Real-Time SMP Scheduling

Ted Baker

Department of Computer Science

Florida State University

Tallahassee, FL 32312

http://www.cs.fsu.edu/∼baker

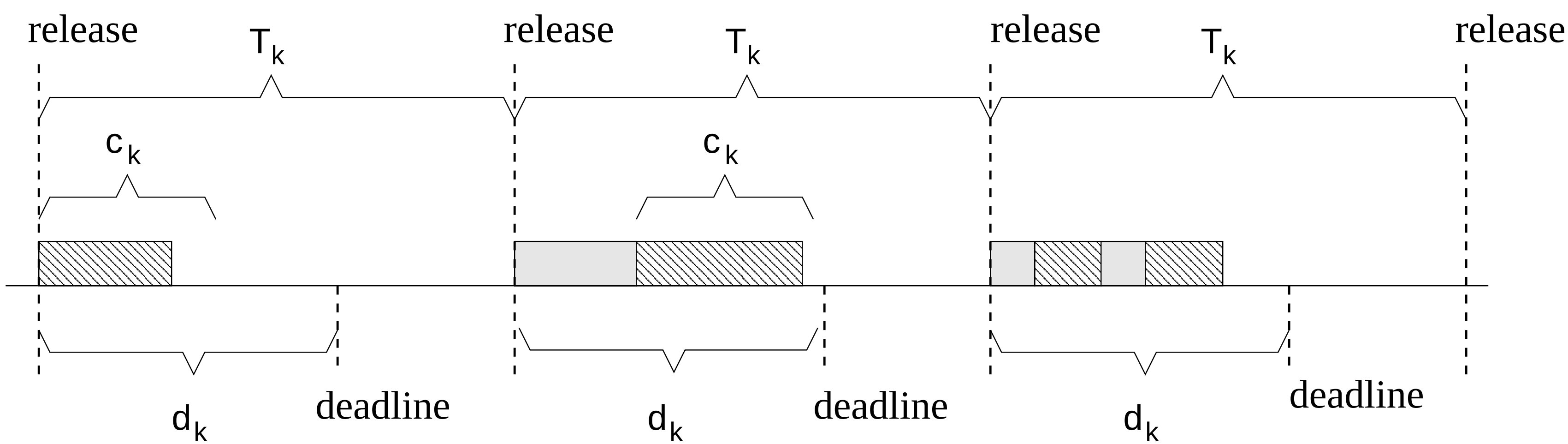
Overview

1. taste of real-time scheduling theory
2. a research process
   - where an idea comes from
   - why doing research in ”backwaters” may lower stress
   - what to do when somebody else ”scoops” you
   - how to publish
   - what to expect from referees
3. a recent research result of mine
4. what I hope to do with the idea next

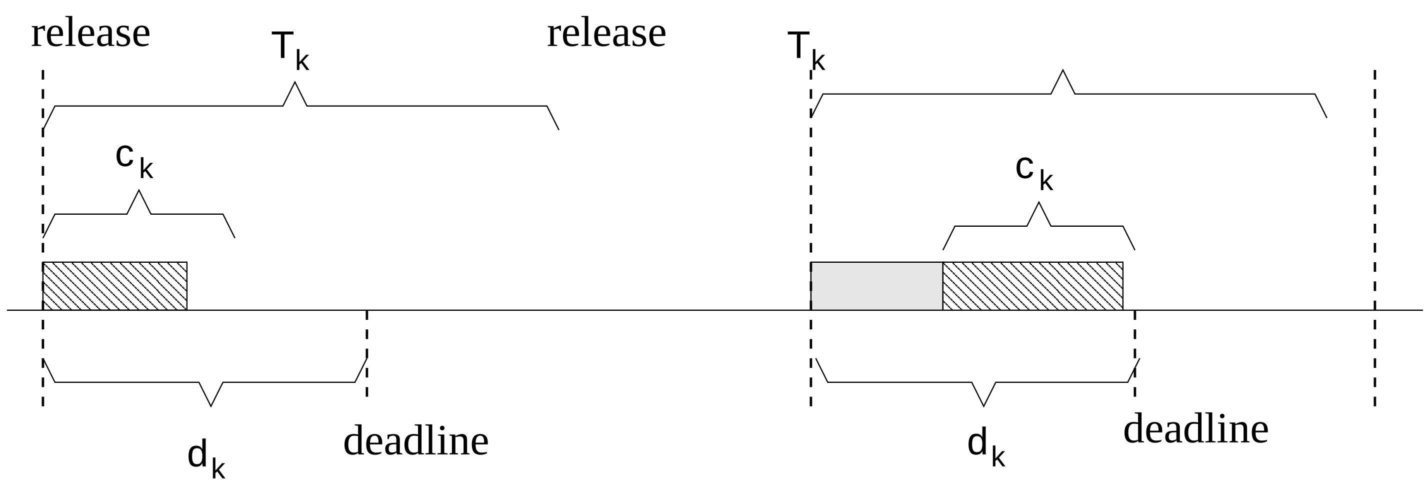
Background: Periodic Task Model

- set of tasks $\tau_1, \ldots, \tau_n$
- each task has a *period* T_i
- each task has a *worst case compute time* c_i
- each task has a *relative deadline* d_i.
- processor *utilization* of task $\tau_i = \frac{c_i}{T_i}$
- *total* utilization = $\sum_{i=1}^{n} \frac{c_i}{T_i}$

Gannt Chart of a Periodic Task's Execution



Period is Just a Lower Bound



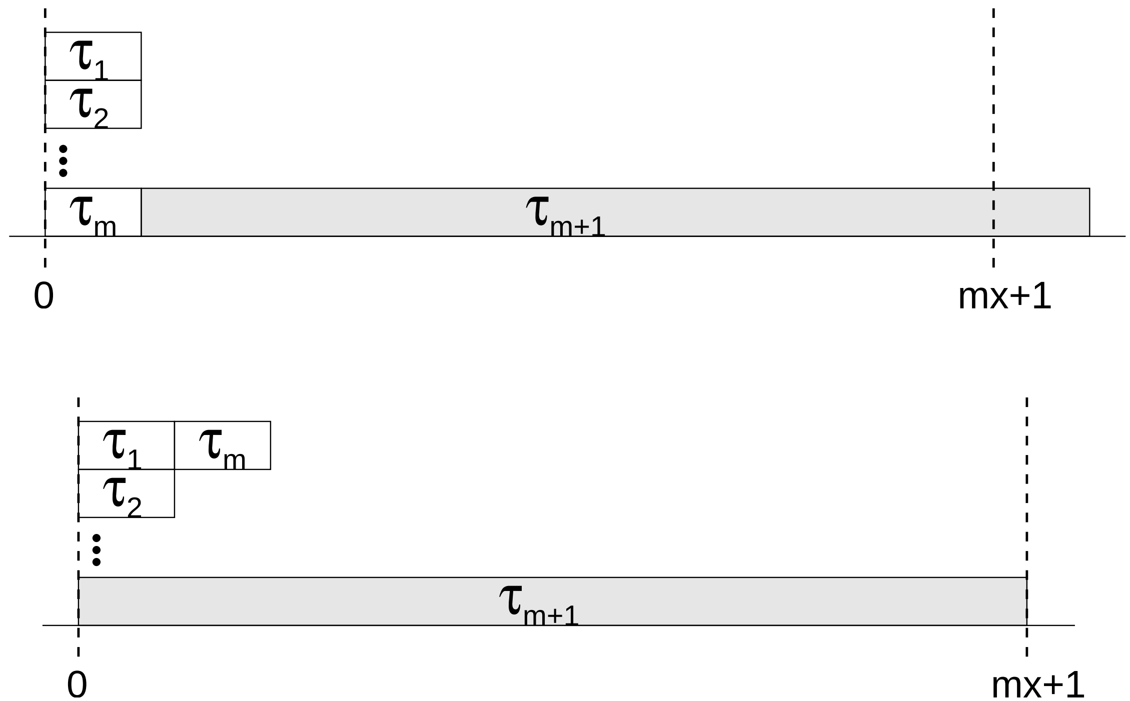
Background: Liu & Layland EDF Utilization Bound

Theorem A set of n independent periodic tasks is schedulable by preemptive EDF scheduling on one processor if

$$\sum_{i=1}^{n} \frac{c_i}{d_i} \leq 1$$

Q: *How does this generalize for m processors?*

Bad Example for MP EDF Scheduling



Consequences

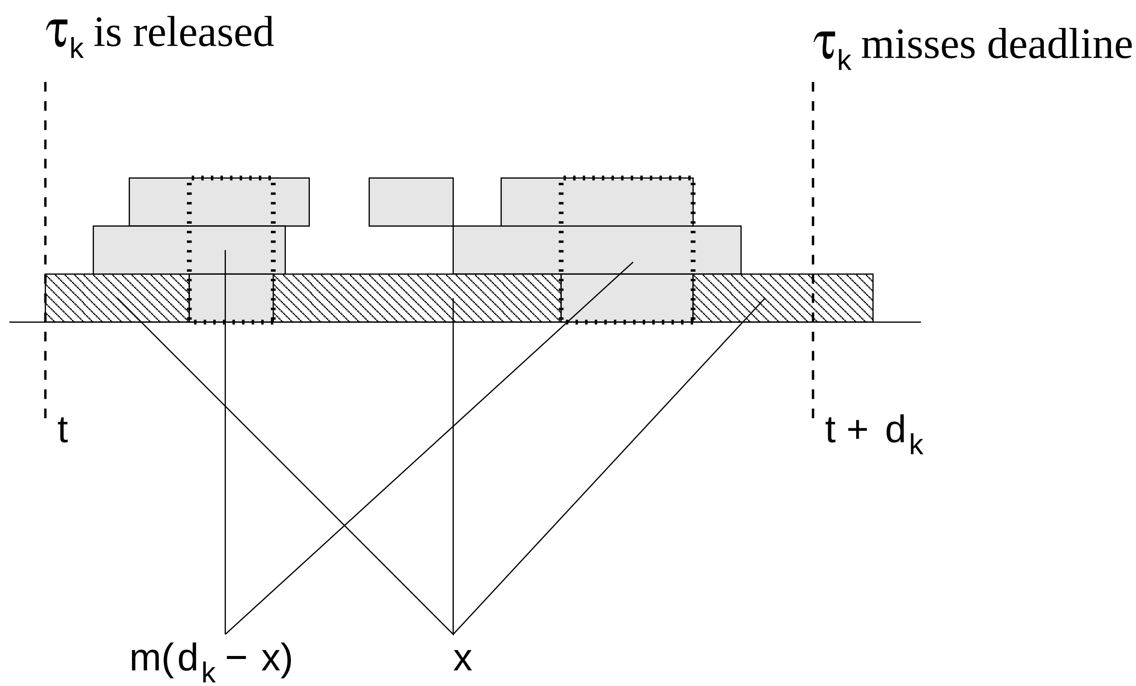
This example, which shows worst-case achievable processor utilization can be as bad as 1 (compared to ideal value of m).

Everybody says EDF scheduling is no good for multiprocessors.

Everybody assumes tasks must be bound to processors in a static (or nearly static) way, and single-processor scheduling applied to each processor.

Papers are written on how to partition tasks between processors, a bin packing problem.

1990: My Observation about a "Problem Window"



1990: My Conjecture

If we have an upper bound on individual task utilizations we have a lower bound on the worst-case achievable utilization.

Looking at the example, the worst-case achieveable utilization with EDF seems to be close to $m(1-\lambda)$ where $lambda = \max_{i=1}^{n} \frac{c_i}{T_i}$.

Years Go By

I talk to Lui Sha (then CMU/SEI and now UIUC) about the idea. He doesn't seem to understand what I am talking about enough pick up on it.

I am still convinced it should not be too hard to prove something here.

I suggest to three different Ph.D. students that they work on the problem. They get nowhere.

2003: Revisiting the Problem

No longer department chair, with no current Ph.D. students, I decide to work out the result myself.

The Key Lemma

Lemma (upper bound on EDF load) For any busy window $[t, t+\Delta)$ with respect to τ_k the EDF load W_i/Δ due to τ_i is at most β_i, where

$$\beta_i = \begin{cases} \frac{c_i}{T_i}(1 + \frac{T_i - d_i}{d_k}) & \text{if } \lambda \geq \frac{c_i}{T_i} \\ \frac{c_i}{T_i}(1 + \frac{T_i - d_i}{d_k}) + \frac{c_i - \lambda T_i}{d_k} & \text{if } \lambda < \frac{c_i}{T_i} \end{cases}$$

The Final Result

Theorem (EDF schedulability test) A set of periodic tasks $\tau_1, \ldots, \tau_n$ is schedulable on m processors using preemptive EDF scheduling if, for every task τ_k,

$$\sum_{i=1}^{n} \min\{1, \beta_i\} \leq m(1 - \frac{c_k}{d_k}) + \frac{c_k}{d_k}$$

where β is as defined in the Lemma above.

The Nice Corollary

Corollary A set of periodic tasks $\tau_1, \ldots, \tau_n$, all with deadline equal to period, is guaranteed to be schedulable on m processors using preemptive EDF scheduling if

$$\sum_{i=1}^{n} \frac{c_i}{T_i} \leq m(1-\lambda)+\lambda$$

How I was "Scooped"

A periodic task set $\{\tau_1, \tau_2, \ldots \tau_n\}$ is *light on m processors* if:

1. $\sum_{i=1}^{n} \frac{c_i}{T_i} \leq \frac{m^2}{2m-1}$
2. $\frac{c_i}{T_i} \leq \frac{m}{2m-1}$, for $1 \leq i \leq n$.

Theorem (Srinivasan, Baruah[4]) Any periodic task system that is light on m processors is scheduled to meet all deadlines on m processors by EDF.

What I Thought I was able to Salvage

- new proof technique
- pre-period deadlines
- more general utilization bound test: $\frac{m^2}{2m-1}$ is just a special case of $m(1-\lambda)+\lambda$
- proof that the utlization bound is tight

The Second Scoop

Theorem (Goossens, Funk, Baruah[3]) A set of periodic tasks $\tau_1, \ldots, \tau_n$, all with deadline equal to period, is guaranteed to be schedulable on m processors using preemptive EDF scheduling if

$$\sum_{i=1}^{n} \frac{c_i}{T_i} \leq m(1-\lambda)+\lambda$$

where $\lambda = \max\{c_i/T_i \mid i = 1, \ldots, n\}$.

They also provided a proof (like mine) that this result is “tight”.

What I was able to Salvage

- pre-period deadlines
- new proof technique
- decided to merge fixed-priority results into same paper

What Referee 1 Said

“...Although the paper has some contributions to be presented .. the topic and motivation is not that exciting. ...”

Consequence of being scooped.

What Referee 1 Said

“Quantitative justification of the proposed analysis is required. ... more general in the sense that it can handle preperiod deadlines. ... we can simply ... change the original execution time C to C+(P-D) to assure P-D (D is the preperiod deadline) earlier completion prior to the period P. Obviously, this simple modification of the previous analysis may be much less accurate than the proposed analysis. However, how much accuracy improvement can be achieved by the proposed analysis is questionable....”

There is an improvement, but to show it is a good idea for more research. One way to do this is via simulation on a large randomly chosen collection of task sets.

What Referee 1 Said

“... for some important theorems, only sketch of proof is given referring their two technical reports. This makes readers hard to follow the theorems ...”

You can’t win on this, given the 20-page limit for papers. Putting in more proofs means less results, and maybe an even less exciting paper.

What Referee 2 Said

"...Given the originality of this work, I strongly recommend that this paper be accepted. ..."

What Referee 3 Said

“... The paper is well written, and the results are of theoretical interest. ...”

What Referee 3 Said

"... practical usage ... is limited ... unrealistic system model ... scalability and processor cache considerations ... modern operating systems use a priority queue per processor ... schedule the task on the processor where its previous instance executed ... not ... the processor that is executing the lowest priority task ... ... introduces a form of priority inversion when tasks are dynamically dispatched ... challenging to dynamically schedule tasks in a multiprocessor in consistent priority order ... many other factors make the assumption of perfect preemption invalid."

A valid question. This is something we need to look into further. Clearly, there are trade-offs involved.

What Referee 3 Said

"...Are the bounds tight, in the sense that Liu and Layland bound is, while many subsequent schedulability are not? Some statement on tightness of the bounds is needed."

We actually answered in the paper, but since it was just a few senteces and a reference to the people who "scooped" me, the reviewerd missed it.

What Referee 3 Said

“...Lemma 9 is obvious. The proof obscures the result...”

You can’t please everybody on this kind of issue. Referee 1 wanted more details on proofs.

What I Hope to Do Next

- Try to resolve Referee 1's issue about how much is gained, and how often, by the tighter preperiod deadline schedulability test
- Try to resolve Referee 3's issue about fixed *vs.* dyamic binding of tasks to processors
  1. simulate
  2. implement and test, to determine real switching overheads
  3. distribute implementation
- Extend analysis to include blocking for mutexes
- Revisit aperiodic server scheduling algorithms, in the MP context

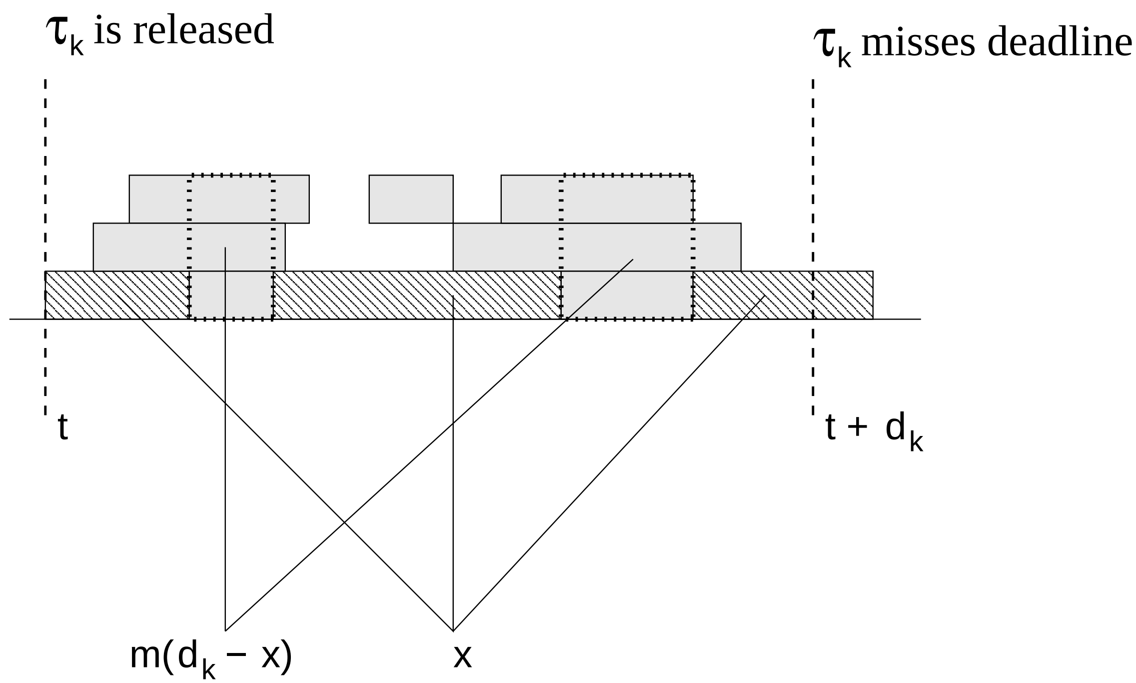
The Reasoning

Definition The *demand* of a time interval is the total amount of computation that would need to be completed within the window for all the deadlines within the interval to be met.

Definition The *load* of an interval $[t, t+\Delta)$ is W/Δ, where W is the demand of the interval.

If we can find a lower bound on the load of a problem window that is necessary for a job to miss its deadline, and we can show that a given set of tasks could not possibly generate so much load in the problem window, that would be sufficient to serve as a schedulability condition.

Lower Bound on Load



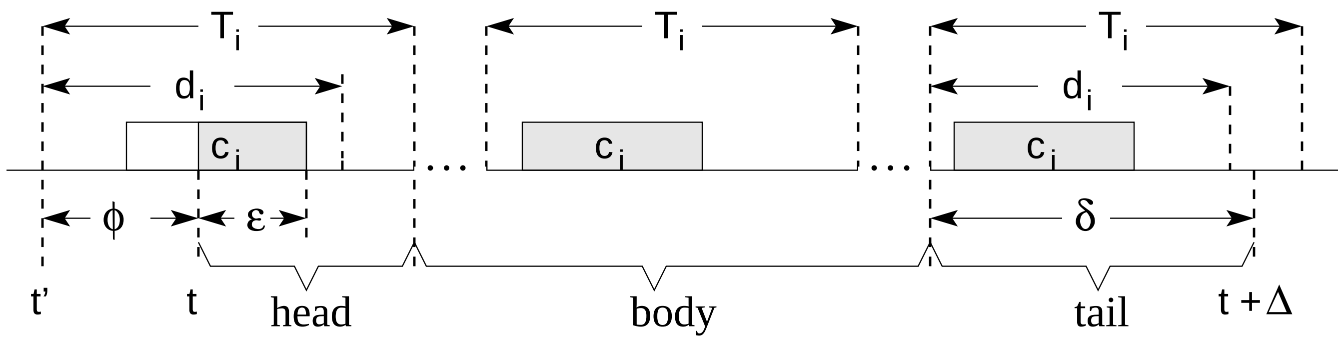
Since the problem job misses its deadline, the sum of the lengths of all the time intervals in which the problem job does not execute must exceed its slack time, $d_k - c_k$.

Lower Bound on Load

Lemma(lower bound on load) If W/d_k is the load of the interval $[t, t+d_k)$, where $t+d_k$ is a missed deadline of τ_k, then

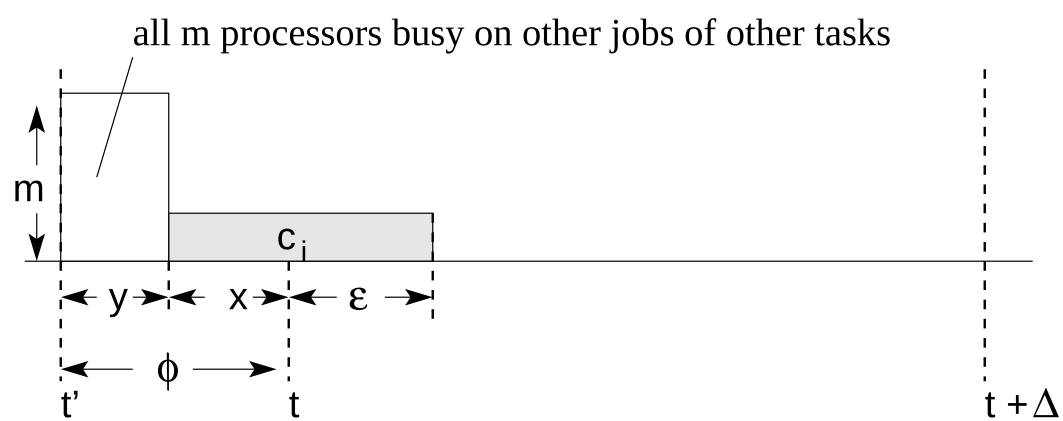
$$\frac{W}{d_k} > m(1 - \frac{c_k}{d_k}) + \frac{c_k}{d_k}$$

Analysis of Maximum Load



Carried-in Load

Definition The *carry-in* of τ_i at time t is the residual compute time of the last job of task τ_i released before t, if any, and is denoted by the symbol ε.

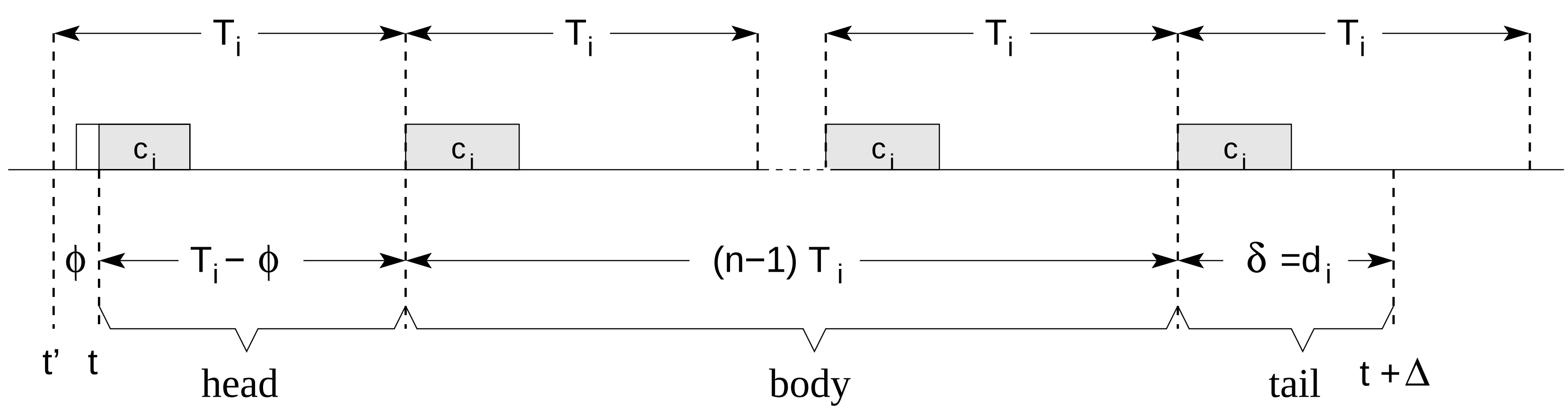


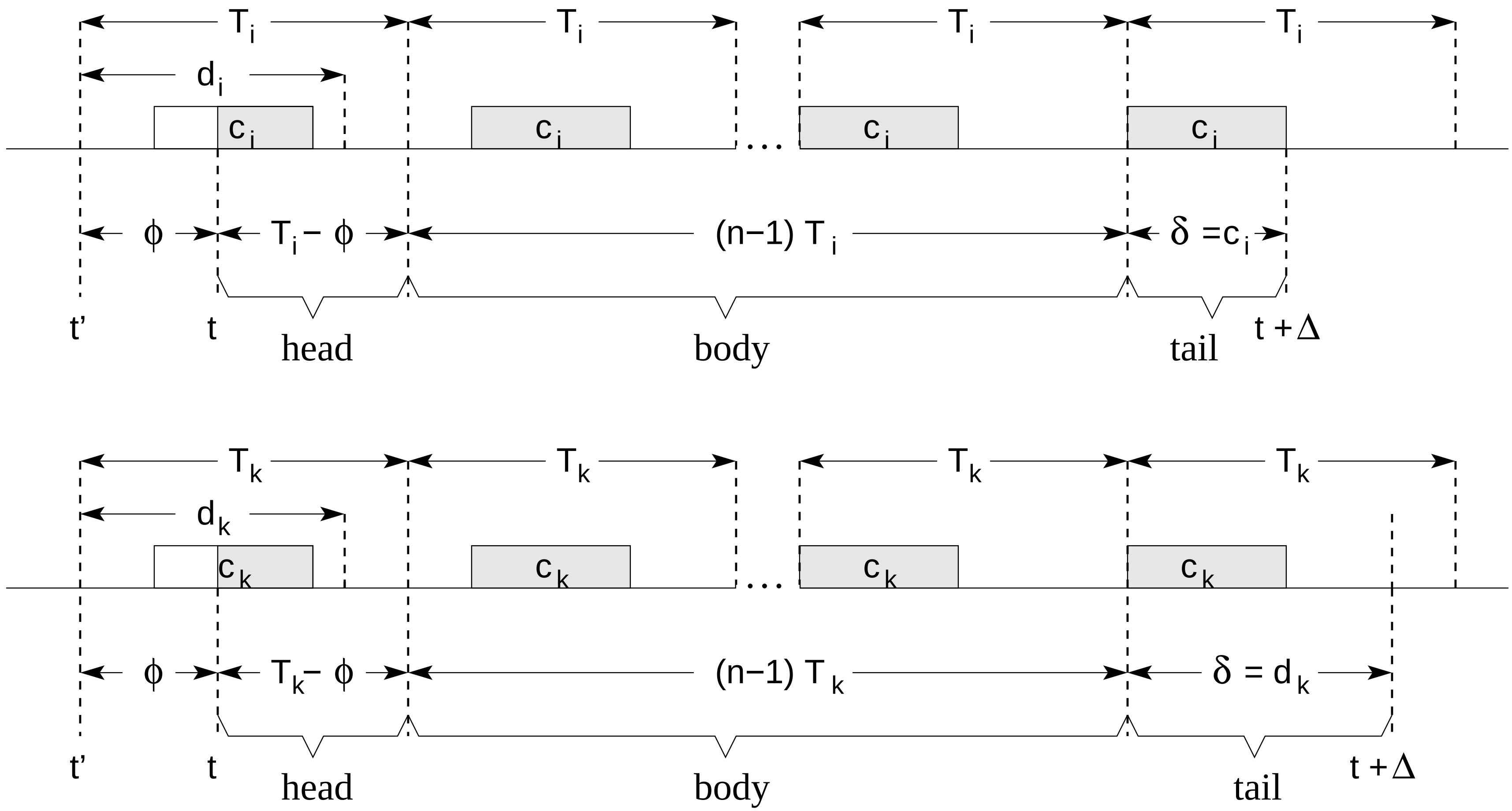
Upper Bound on EDF Demand

Lemma (EDF demand) For any busy window $[t, t+\Delta)$ of task τ_k (*i.e., the maximal* λ*-busy downward extension of a problem window*) and any task τ_i, the EDF demand W_i of τ_i in the busy window is no greater than

$$nc_i + \max\{0, c_i - \phi\lambda\}$$

where $\phi = nT_i + d_i - \Delta$, $n = \lfloor(\Delta - d_i)/T_i\rfloor + 1$ if $\Delta \geq d_i$, and $n = 0$ otherwise.





T_i
d_i
c_i
φ
T_i − φ
(n−1) T_i
δ = c_i
t'
t
head
body
tail
t + Δ
T_k
d_k
c_k
T_k − φ
(n−1) T_k
δ = d_k

References

1. C.L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time environment”, *JACM 20.1* (January 1973) 46-61.
2. T.P. Baker, *Multiprocessor EDF and Deadline Monotonic Schedulability Analysis*, http://www.cs.fsu.edu/research/reports/TR-030401.pdf
3. J. Goossens, S. Funk, S. Baruah, “Priority-driven scheduling of periodic task systems on multiprocessors”, technical report UNC-CS TR01-024, University of North Carolina Computer Science Department, *Real Time Systems*, Kluwer, (to appear).
4. A. Srinivasan, S. Baruah, “Deadline-based scheduling of periodic task systems on multiprocessors”, *Information Processing Letters 84* (2002) 93-98.x
5. C.A. Phillips, C. Stein, E. Torng, J Wein, “Optimal time-critical scheduling via resource augmentation”, *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing* (El Paso, Texas, 1997) 140-149.