Managing Latency in Embedded Streaming Applications under Hard-Real-Time Scheduling

Mohamed A. Bamakhrama and Todor Stefanov

October 8, 2012 - CODES+ISSS - Tampere, Finland

Universiteit Leiden
The Netherlands

Introduction

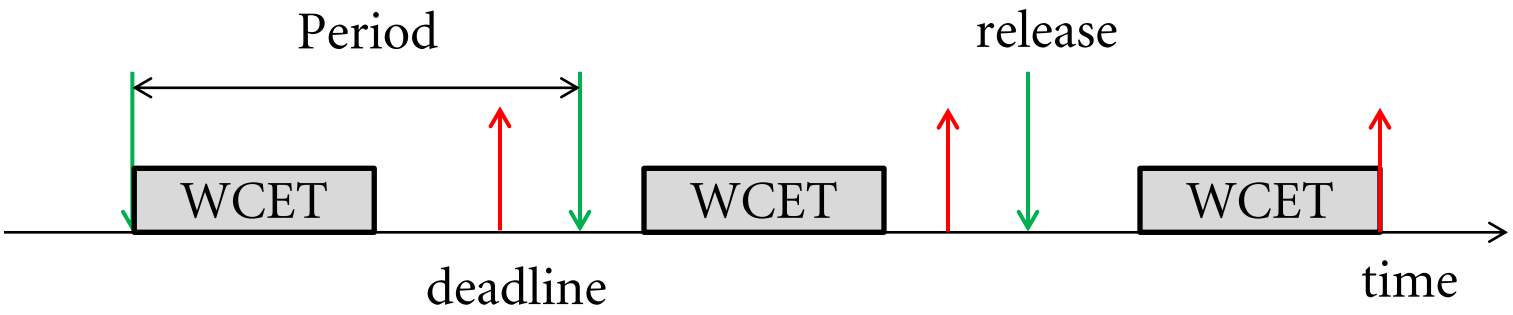
- Many modern embedded streaming systems require:
  - Real-time execution of applications on multiprocessor platforms
  - Ability to add/remove new applications at run-time

- Solution approach: *Apply classical real-time scheduling theory to dataflow models*
  - Dataflow model: Cyclo-Static Dataflow (CSDF)
  - The problem is classical real-time scheduling theory uses *different* task models than dataflow!

The periodic real-time task model

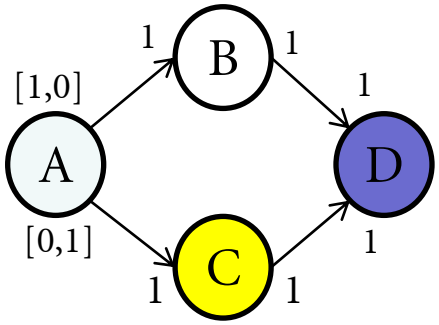
- An application is modeled as a set of *independent* tasks

- A *task* is characterized by 4 parameters: *WCET*, *period*, *start time*, and *deadline*
  - If deadline = period → *implicit-deadline (IDP)*
  - If deadline < period → *constrained-deadline (CDP)*



Cyclo-Static Dataflow (CSDF)

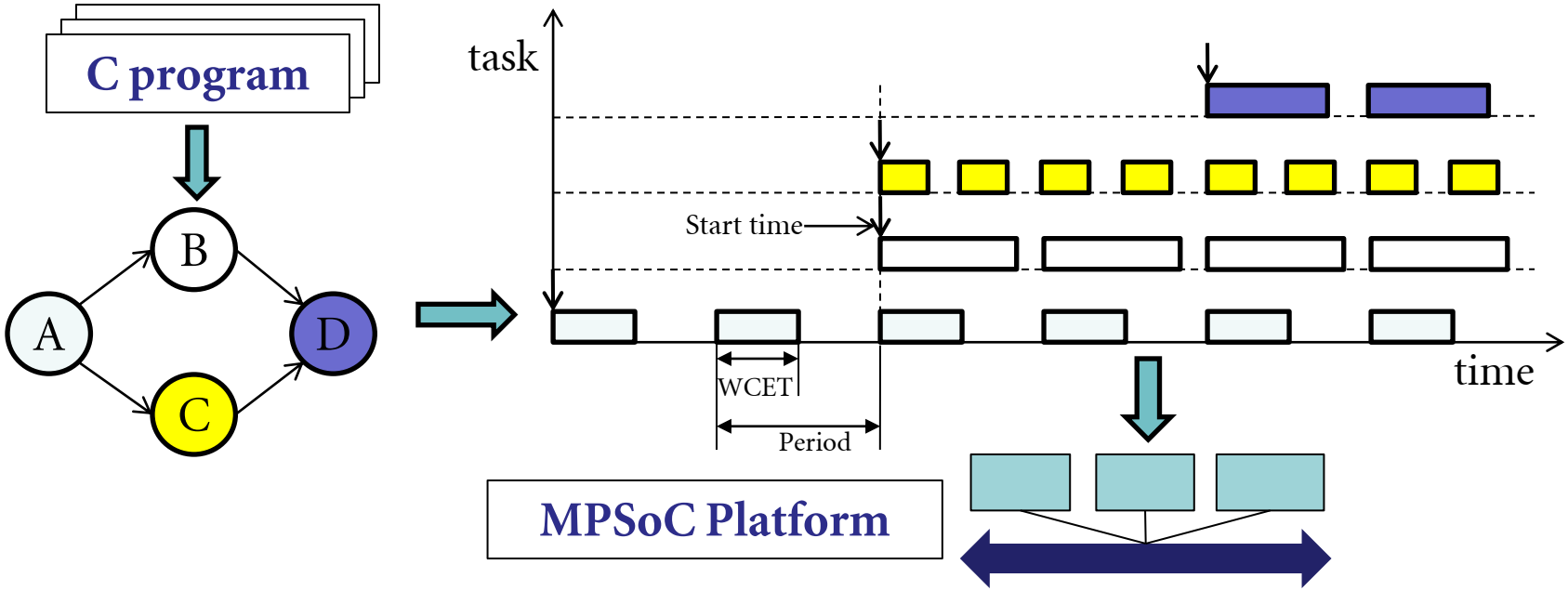
- An application is modeled as a set of *dependent* tasks (represented as a *directed graph)*
  - Graph nodes represent *tasks* (a.k.a. *actors*)
  - Graph edges represent *data dependencies*



- A graph must be *consistent* in order to be scheduled using bounded memory
  - A finite sequence of actor invocations that can be repeated infinitely
    - e. g. **AABCD**AABCDAABCD…
  - Each actor v_i has a repetition rep_i in such sequence. For example: rep_A=2, rep_B=1

Context of This Work

- Both models do not match!
  - Solution: **Daedalus^{RT} framework**
  - *Open-source*, available at: http://daedalus.liacs.nl



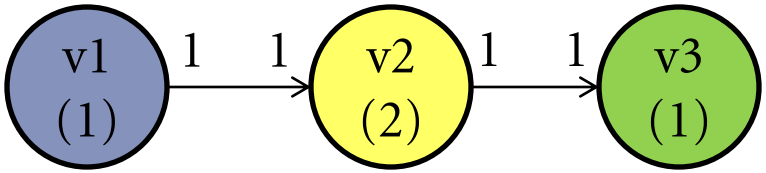
[1] M. Bamakhrama and T. Stefanov. Hard-real-time scheduling of data-dependent tasks in embedded streaming applications. *In EMSOFT*, 2011.
[2] M. Bamakhrama et al. A Methodology for Automated Design of Hard-Real-Time Embedded Streaming Systems. *In DATE*, 2012

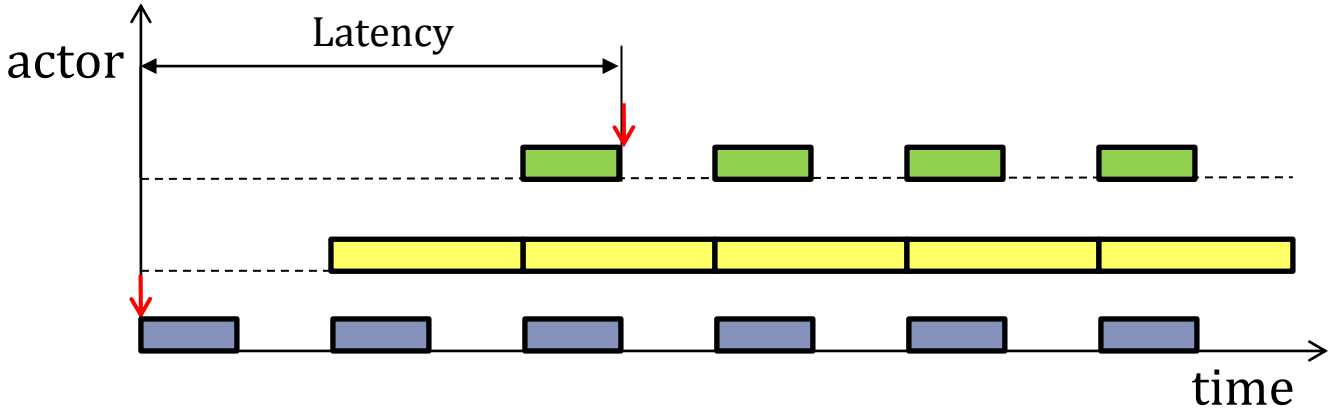
Throughput Analysis

- The CSDF graph actors are scheduled as *periodic real-time tasks*
  - Such scheduling approach is called *strictly periodic scheduling (SPS)*
- SPS delivers the *maximum achievable throughput* for *matched I/O rates* graphs
  - Matched I/O rates graphs represent 80% of streaming applications

What about Latency?

- Latency is more important for some applications
  - Defined as the *time elapsed* between the *start of the first firing of the source in a path* and the *completion of the first firing of the sink in the same path*





Problem Statement

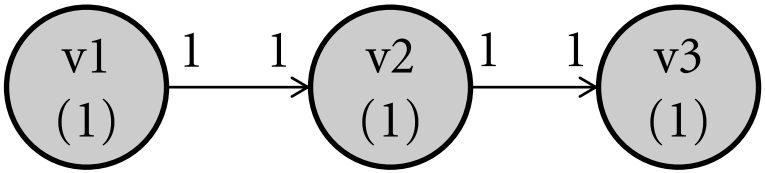
- Daedalus^{RT} uses *implicit-deadline (IDP) model*

- IDP model delivers optimal throughput and latency for *balanced* graphs
  - $rep_i WCET_i = rep_j WCET_j$ for all v_i, v_j

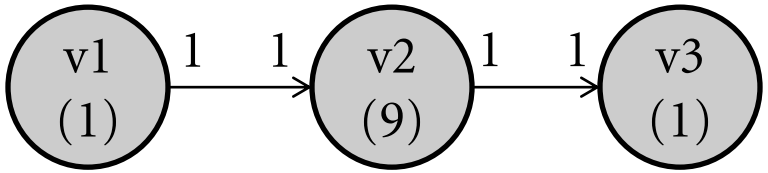
- However, IDP *increases* latency significantly for unbalanced graphs
  - *Constrained-deadline (CDP) model* can be used to manage the latency

Balanced vs. Unbalanced

Graph 1 (Balanced)



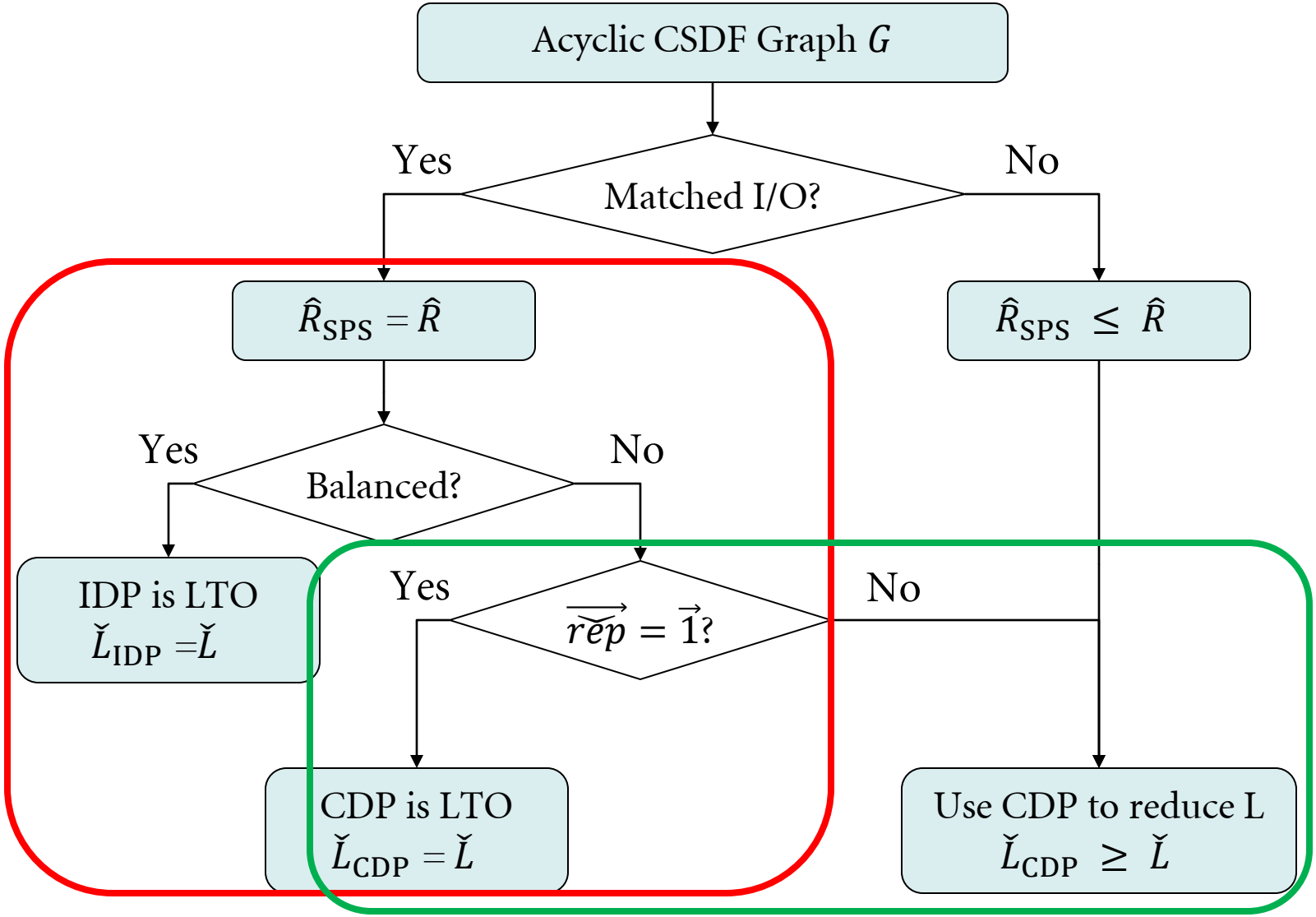
Graph 2 (Unbalanced)



	Self-timed			*IDP*			*CDP*		
	R	*L*	*M*	*R*	*L*	*M*	*R*	*L*	*M*
Graph 1	1/1	3	3	1/1	3	3	1/1	3	3
Graph 2	1/9	11	2	1/9	27	2	1/9	11	2

CDP can help in reducing L without compromising R

Paper Contributions

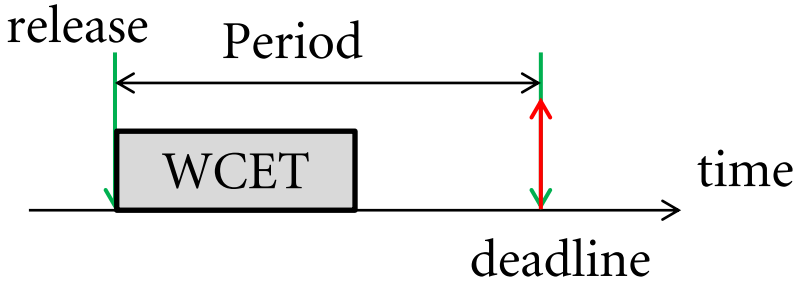


Managing Latency using the CDP model

- We propose an algorithm which checks for the bottleneck actors and assigns them earlier deadlines

- The deadline is controlled by the designer using *deadline factors*. For actor v_i, a deadline factor $d_i \in [0,1]$ specifies the deadline as follows:

$$Deadline_i = WCET_i + d_i(Period_i - WCET_i)$$



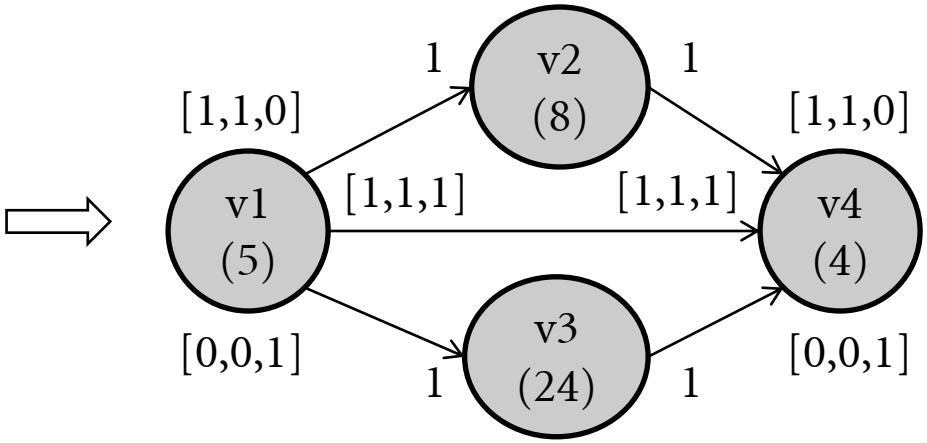
Example 1: $\vec{d} = \vec{1}$

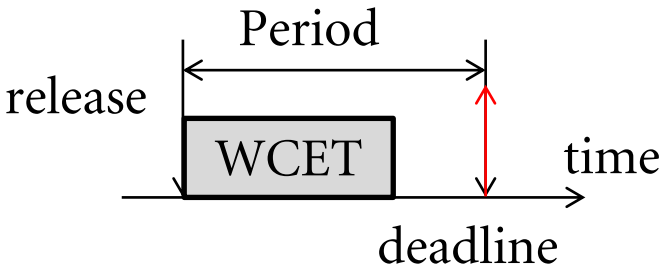
```
int main()
{
  int i, j;
  int img[11][4], img1[11][4];

  while(1) {
    for(i=1;i<=10;i++) {
      for(j=1;j<=3;j++) {
        src(&img[i][j],&img1[i][j]);

        if(j<=2)
          img[i][j]=f1(img[i][j]);
        else
          img[i][j]=f2(img[i][j]);

        snk(img[i][j],img1[i][j]);
      }
    }
  }
  return 0;
}
```





Latency = 56
Min. Latency = 43

Actor	Min. period	Start time (absolute time)	Deadline
v1	8	0	8
v2	12	8	12
v3	24	24	24
v4	8	32	8

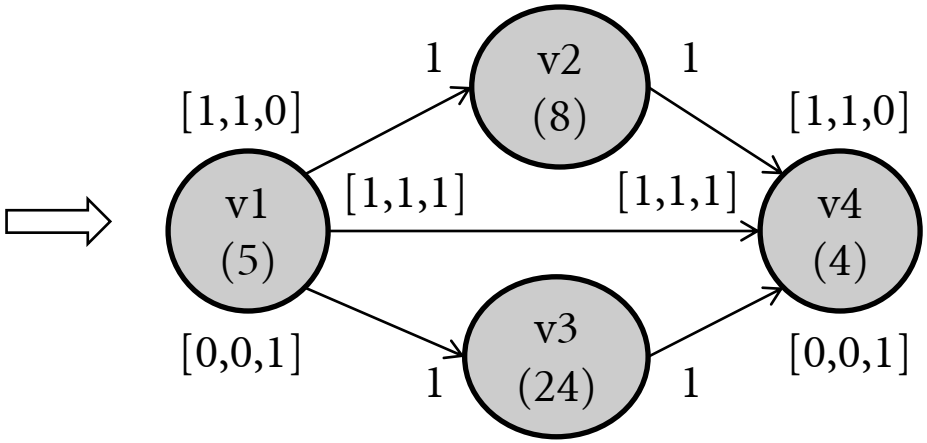
Example 1: $\vec{d} = \overrightarrow{0.5}$

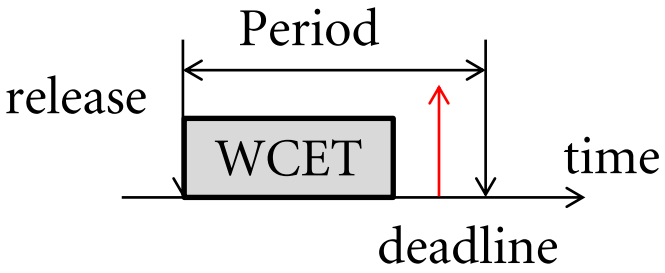
```
int main()
{
  int i, j;
  int img[11][4], img1[11][4];

  while(1) {
    for(i=1;i<=10;i++) {
      for(j=1;j<=3;j++) {
        src(&img[i][j],&img1[i][j]);

        if(j<=2)
          img[i][j]=f1(img[i][j]);
        else
          img[i][j]=f2(img[i][j]);

        snk(img[i][j],img1[i][j]);
      }
    }
  }
  return 0;
}
```





Latency = 52
Min. Latency = 43

Actor	Min. period	Start time (absolute time)	Deadline
v1	8	0	6
v2	12	6	12
v3	24	22	24
v4	8	30	6

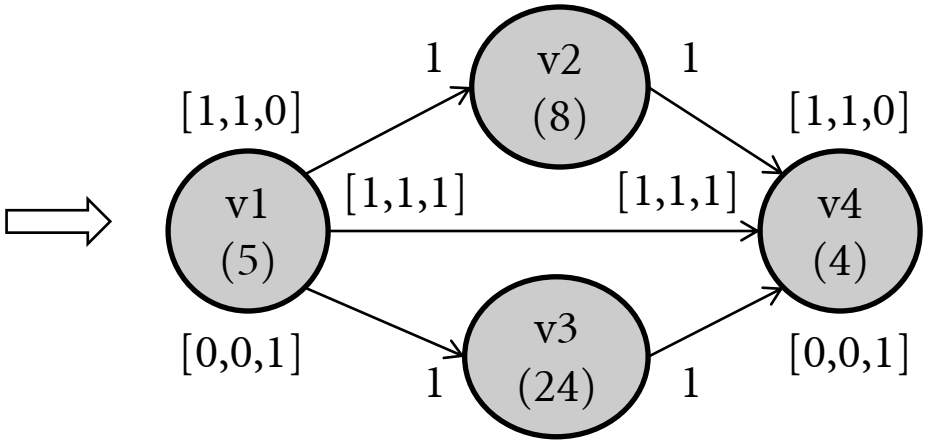
Example 1: $\vec{d} = \vec{0}$

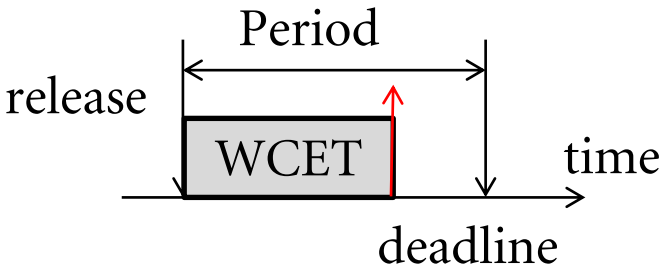
```
int main()
{
  int i, j;
  int img[11][4], img1[11][4];

  while(1) {
    for(i=1;i<=10;i++) {
      for(j=1;j<=3;j++) {
        src(&img[i][j],&img1[i][j]);

        if(j<=2)
          img[i][j]=f1(img[i][j]);
        else
          img[i][j]=f2(img[i][j]);

        snk(img[i][j],img1[i][j]);
      }
    }
  }
  return 0;
}
```

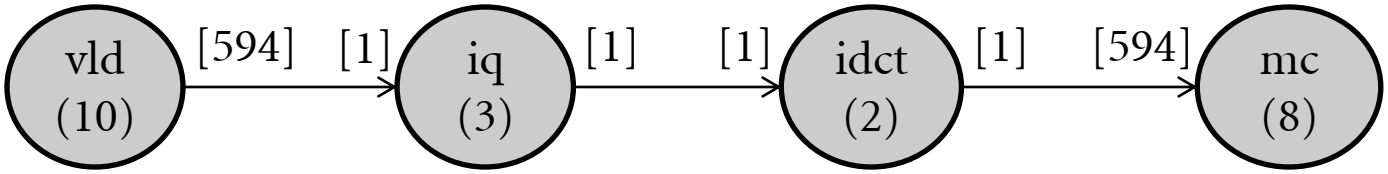




Latency = 49
Min. Latency = 43

Actor	Min. period	Start time (absolute time)	Deadline
v1	8	0	5
v2	12	5	12
v3	24	21	24
v4	8	29	4

Example 2: H.263 decoder



Actor	Min. Period	Deadline d = 1	Deadline d = 0.5	Deadline d = 0
vld	1782	1782	896	10
iq	3	3	3	3
idct	3	3	2	2
mc	1782	1782	895	8
	Latency	5349	3575	1802

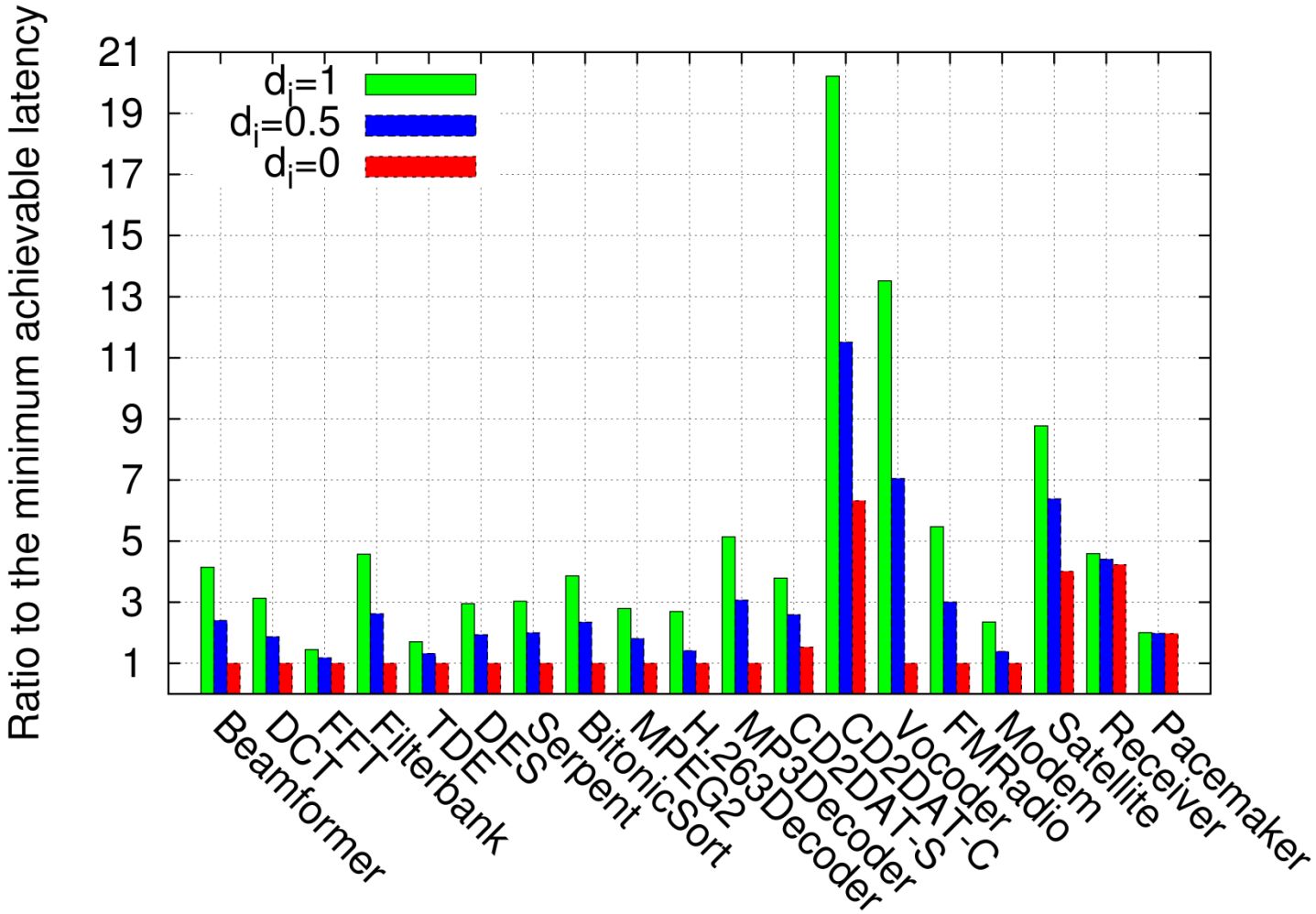
Min. Latency = 1802

Evaluation

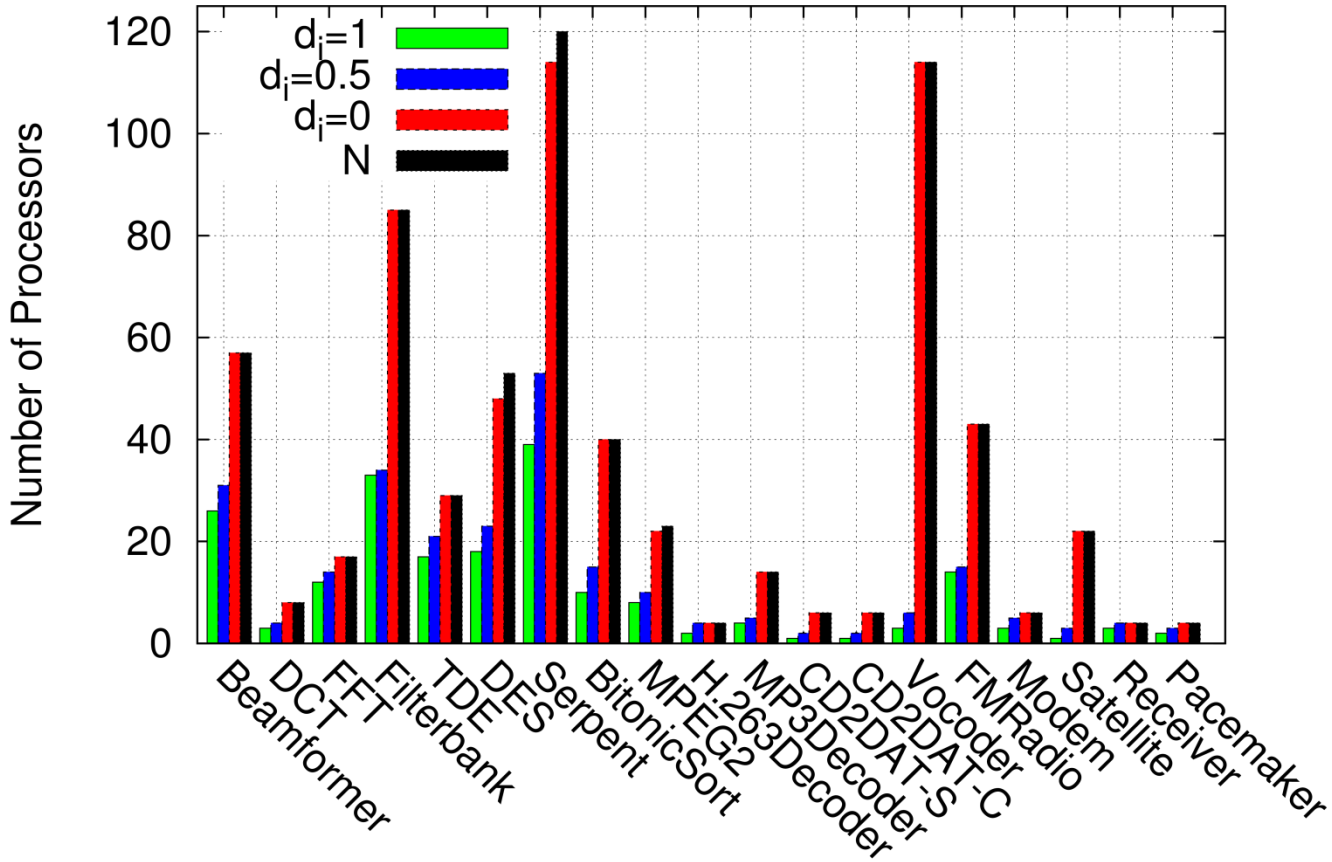
- We evaluate the proposed technique using 19 real applications

- We consider the impact of deadline factors on latency and resource requirements by using different values of the deadline factor

Impact on Latency

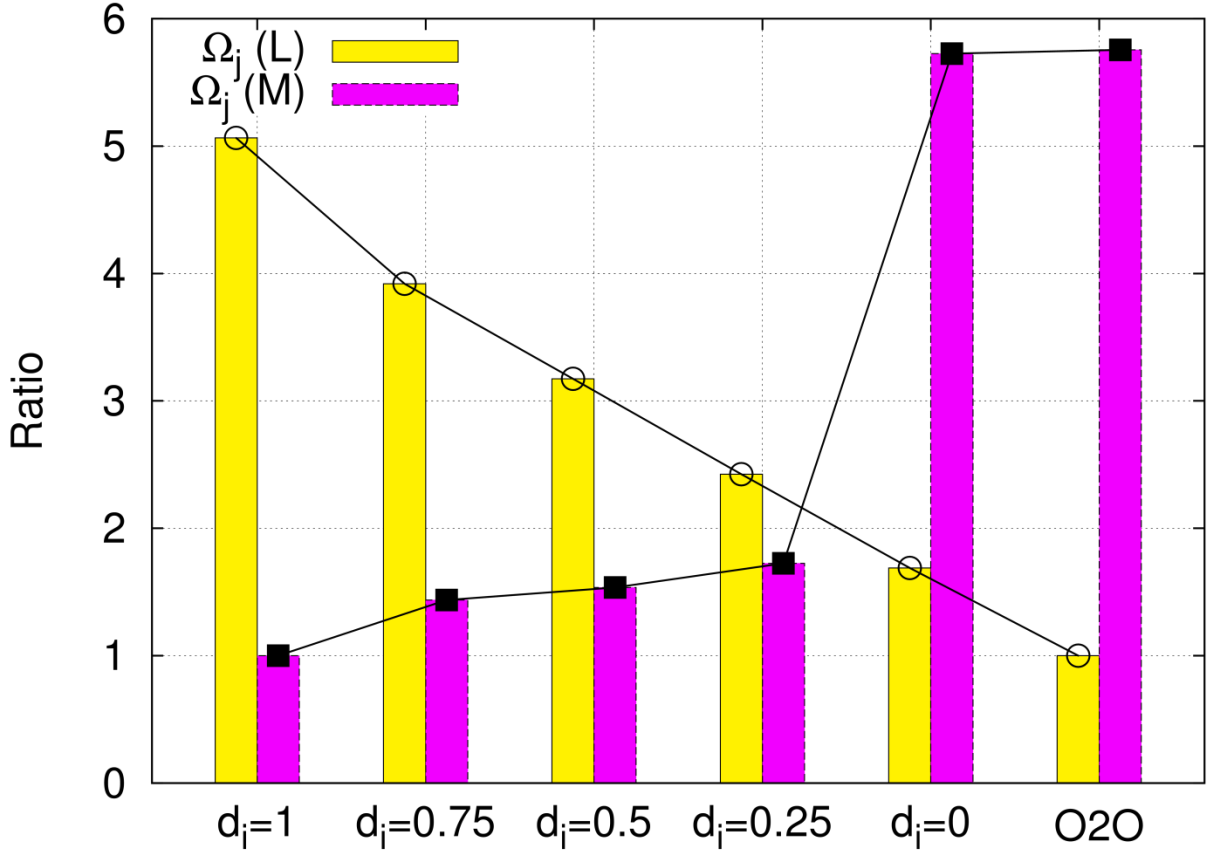


Impact on Resource Usage



- The number of processors is computed using the following sufficient test: $M = \left\lceil \sum \frac{WCET_i}{\min(Deadline_i, Period_i)} \right\rceil$

Relationship between Latency and Required Resources



L scales linearly while *M* does not for the sufficient test

Conclusions

- We propose a *decision tree* to assist the designer in choosing the suitable task model in terms of latency and throughput

- *IDP model is latency and throughput optimal* for *balanced graphs*, however, it increases latency of unbalanced graphs

- CDP model can be used to *reduce latency* without impacting the throughput

- Choosing the deadline involves a tradeoff between latency and resource requirements

Thank you

?