Matrix-DBP for (m,k)-firm real time guarantee

Ye-Qiong Song

Matrix-DBP For (m, k)-firm Real-Time Guarantee

Based on a common work with E. Poggi, YQ. Song, Z. Wang, A. Koubâa
Partially supported by PRA SI01-04

- Part 1: context
- Part 2: where are the problems
- Part 3: Matrix-DBP
- Part 4: future work

Presentation at

ZHEJIANG UNIVERSITY

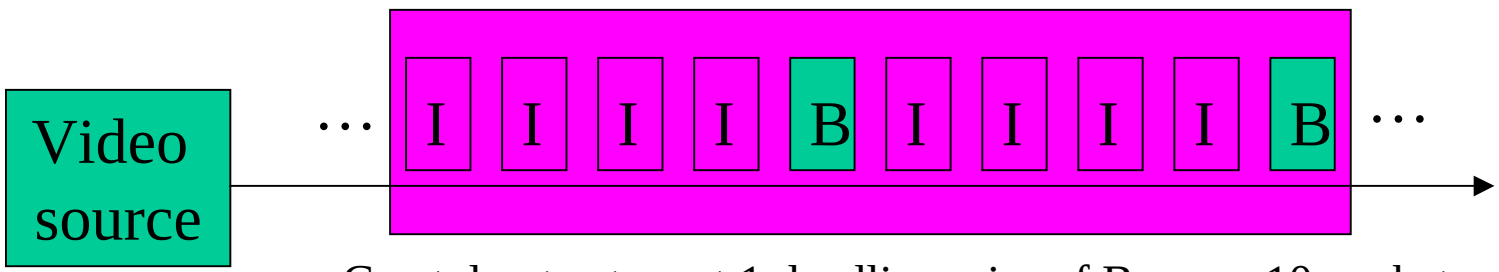
Part 1: (m,k)-firm background

Why we interest in (m,k)-firm?

- HRT uses WC analysis leading to over sized system design (pessimistic performance estimation)
- In practice, many systems being classified HRT are not so « hard ». Occasional deadline misses can be tolerated if they are correctly distributed

Why we interest in (m,k)-firm?

- SRT can resolve the over-sizing problem but does not specify how deadlines can missed
- A probabilistic guarantee of m/k may not be sufficient (e.g. MPEG video stream composed of I, B and P packets)



Can tolerate at most 1 deadline miss of B every 10 packets
Probability of 9/10 can not be suitable

Why we interest in (m,k)-firm?

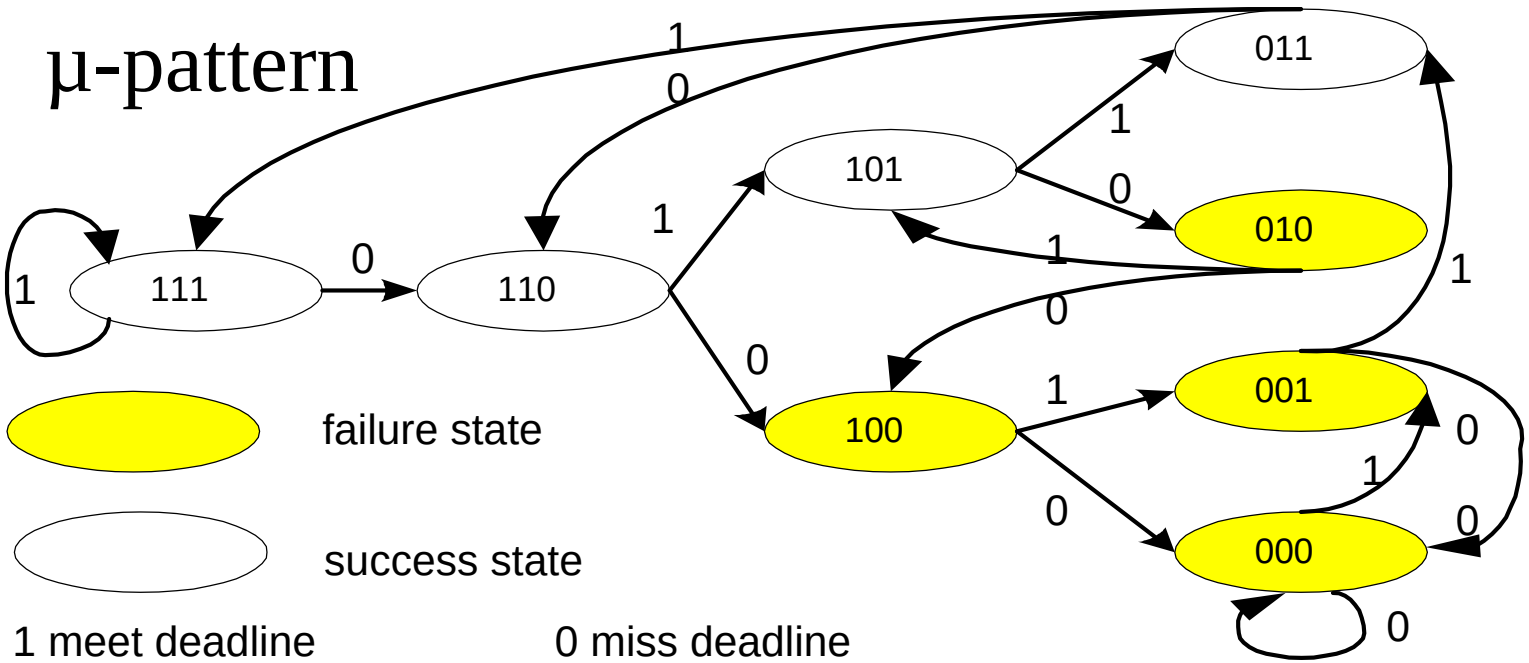
- (m,k)-firm guarantees the deadline meeting of *m* among any *k* consecutive task invocations or message transmissions
- (m,k)-firm provides a general frame for RT constraints description :
  - HRT is a particular case of (m,k)-firm with $m=k$
  - SRT guarantee with probability $p = m/k$ is a particular case of (m,k)-firm with: $p = \lim_{m,k \to \infty} \frac{m}{k}$

(m,k)-firm and WHRT

- (m,k)-firm (i.e. at least m of k meets) is firstly introduced by [Hamdaoui95]
- WHRT is defined by [Bernat98] and [Bernat&Burns01] to generalize (m,k)-firm:
  - $(\overline{m}, \overline{k})$: at most *m* misses. But $(m,k) = (\overline{k} - \overline{m}, \overline{k})$
  - <m,k>: in any *k* consecutive jobs, at least *m* consecutive jobs are with deadline meet
  - $\langle \overline{m}, \overline{k} \rangle = \langle \overline{m} \rangle$: never *m* consecutive deadline misses (notice that $(m,k) \le \langle \overline{k-m} \rangle$)

(m,k)-firm

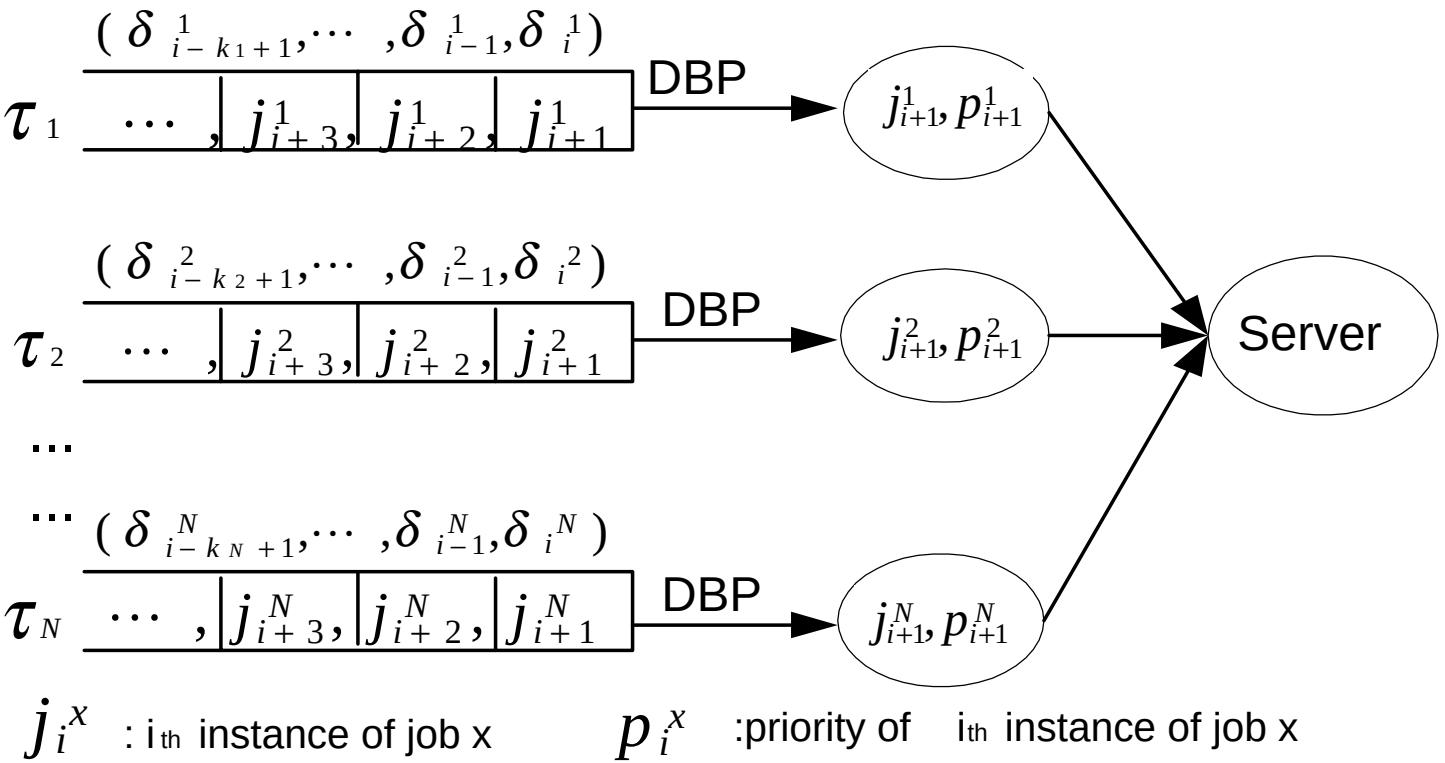
- Failure state in (m,k)-firm context
- Example of (2,3)-firm
- μ-pattern



DBP: Distance Based Priority

- DBP of a μ-pattern is defined as the number of deadline misses to arrive to a failure state
- Example: (3,5)-firm
  - (11011) has a distance of 2
  - (10111) has a distance of 3
  - (10001) has a distance of 0

DBP in MIQSS model



- A job is modeled by: $\tau_i = \{D_i, c_i, m_i, k_i\}$

Gain of DBP for (m,k)-firm

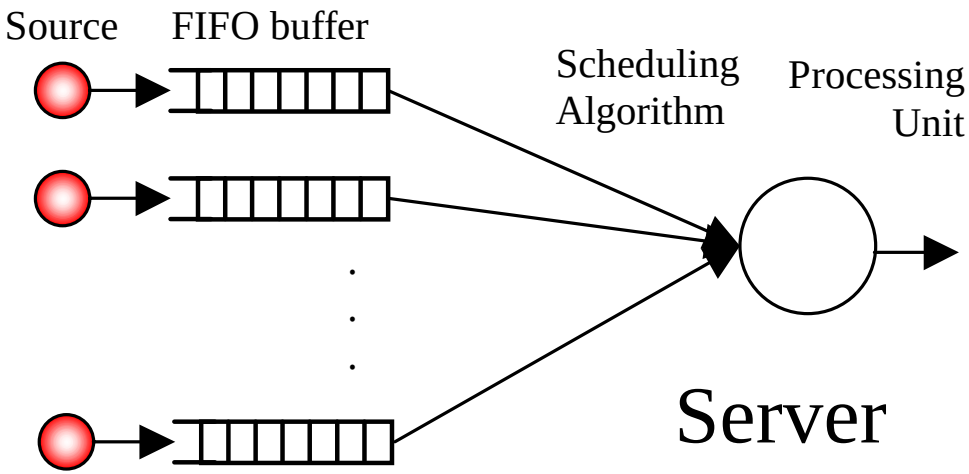
- Dynamic QoS control is possible with:
  - In case of overload, DBP allows on-line drop down of jobs according to (m_i,k_i), reducing thus the need of resources, i.e.:

$$\sum_{i=1}^{N}\left\lceil \frac{c_i}{T_i}\frac{m_i}{k_i}\right\rceil \le 1 \quad \text{instead of} \quad \sum_{i=1}^{N}\left\lceil \frac{c_i}{T_i}\right\rceil \le 1$$

  - Can be extended to (m(t), k(t)) when on-line scheduling algorithm is adopted

Part 2: where are problems?

Problems of DBP in MIQSS for RT



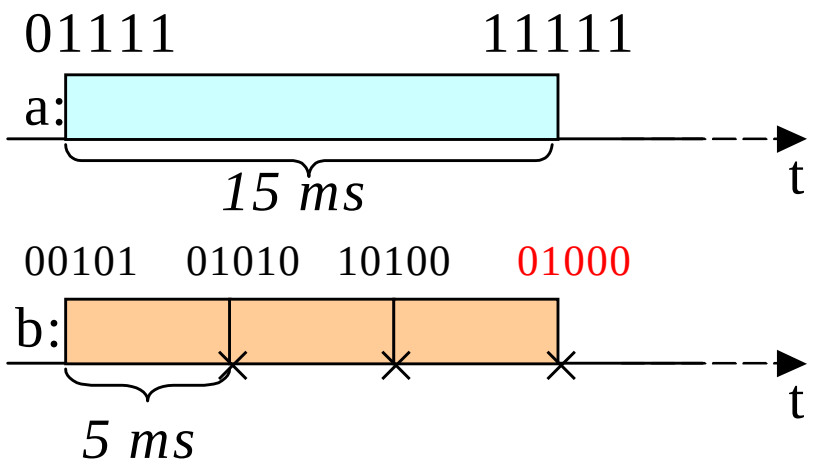
- A job is modeled by: $\tau_i = \{T_i, D_i, c_i, m_i, k_i\}$
- Pb1: how to choose when several jobs have the same priority? (DBP+EDF?)
- Pb2: how to take into account the relative importance among jobs of all sources ?

Un example of Pb2

	(m,k)-constraint	Service time (ms)	Period/ Deadline	Initial μ-pattern
Sa	(4,5)	15	30	{01111}
Sb	(2,5)	2	5	{00101}

2

3



processing firstly Sa will lead to failure state of Sb !

So notion of relative criticality should be defined

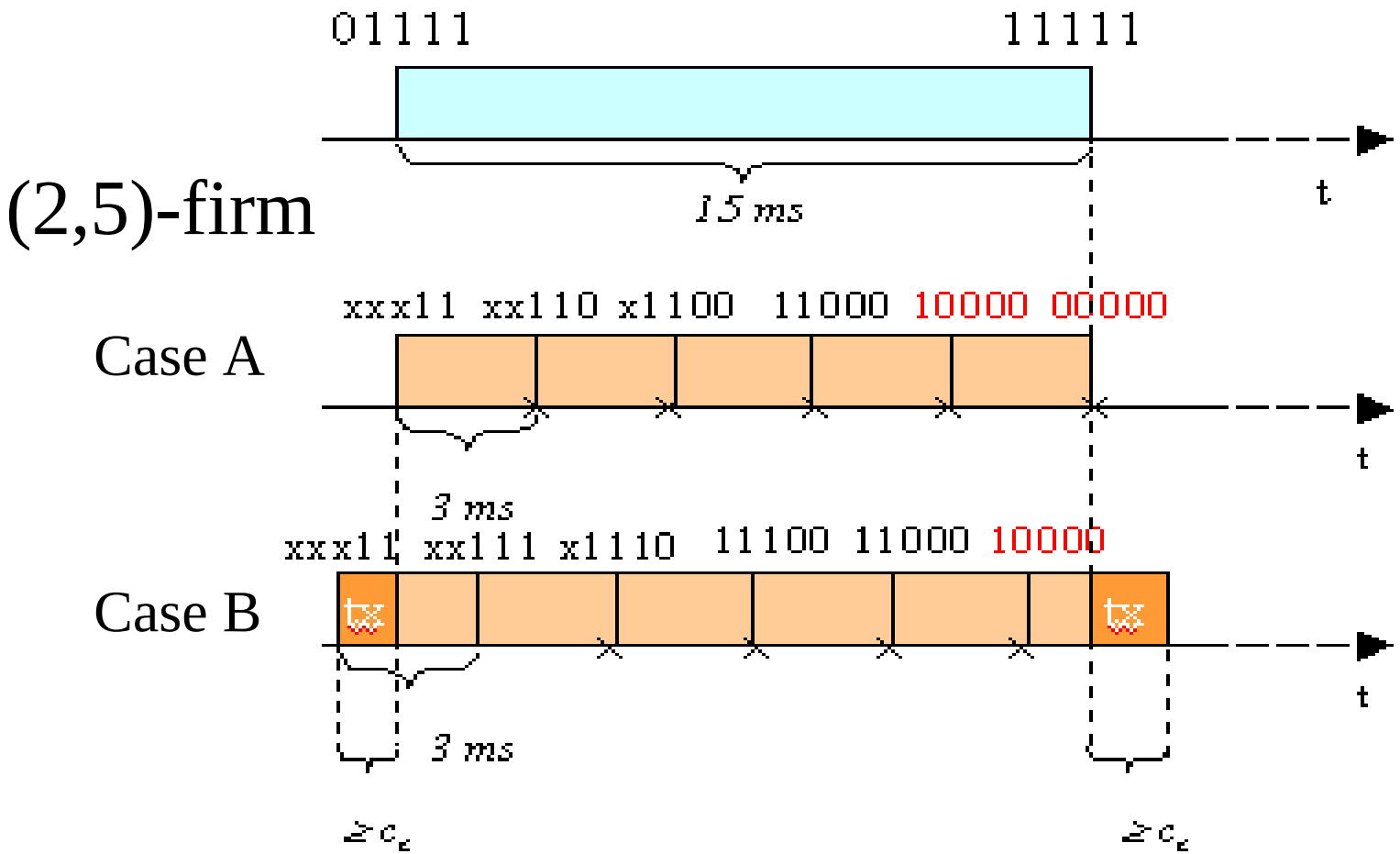
Need of new *non-preemptive* scheduling algo.

- Should take into account:
  - (m,k)-firm constraint
  - Other temporal parameters & constraints: T_i, D_i, c_i
  - Relative criticality between streams (a matrix can be used to represent it)
- Should be of low implementation cost

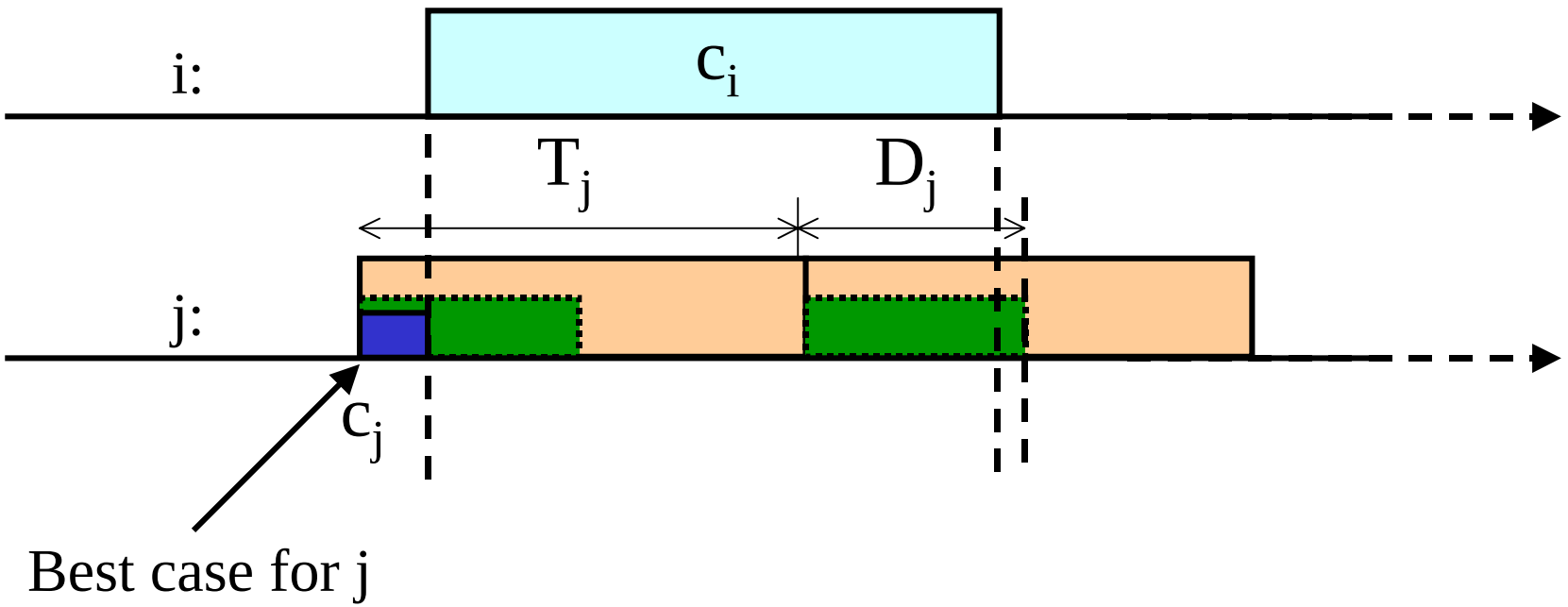
Matrix-DBP can meet all these requirements

Part 3: Matrix-DBP

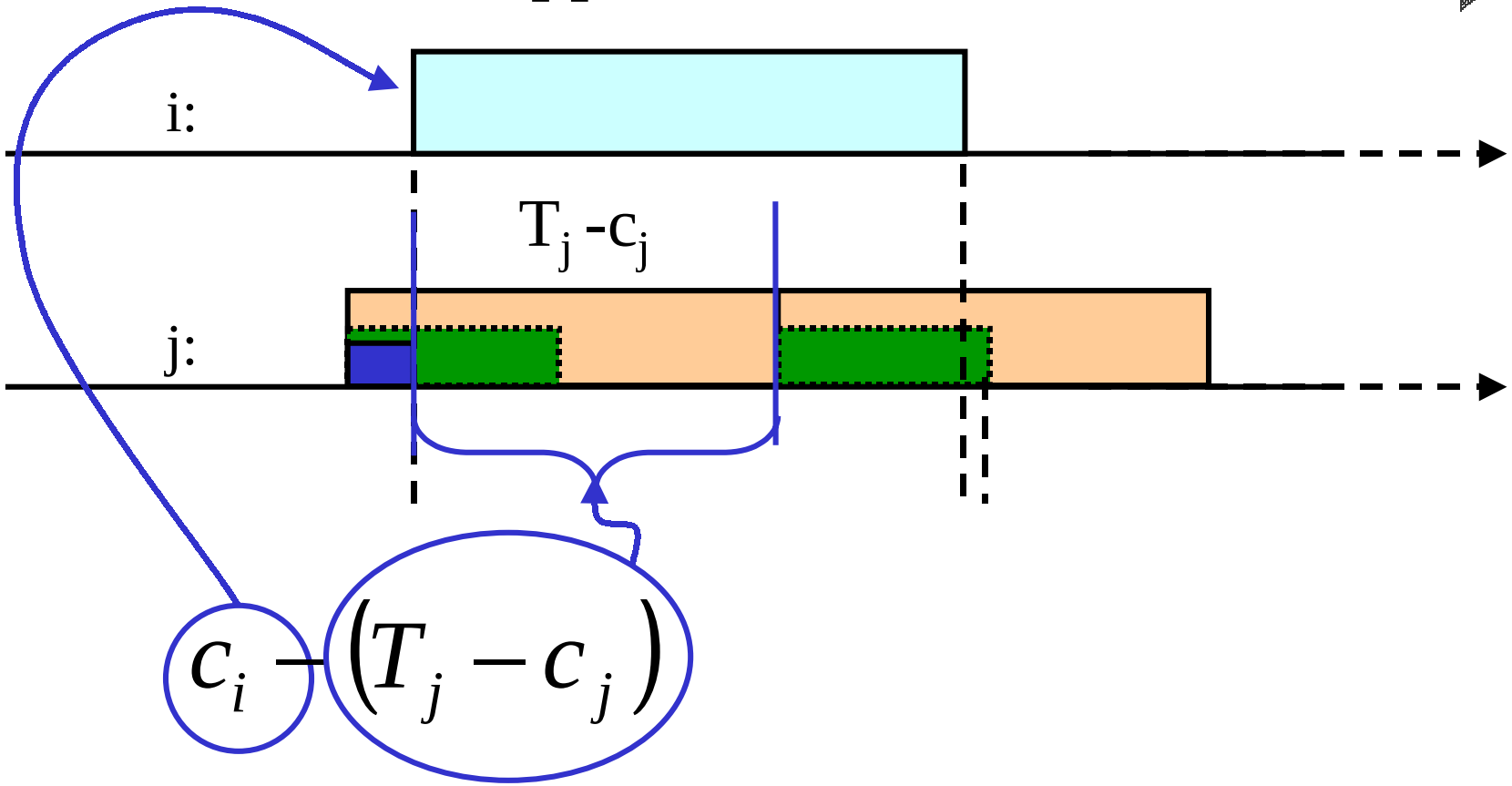
Offset between jobs invocations



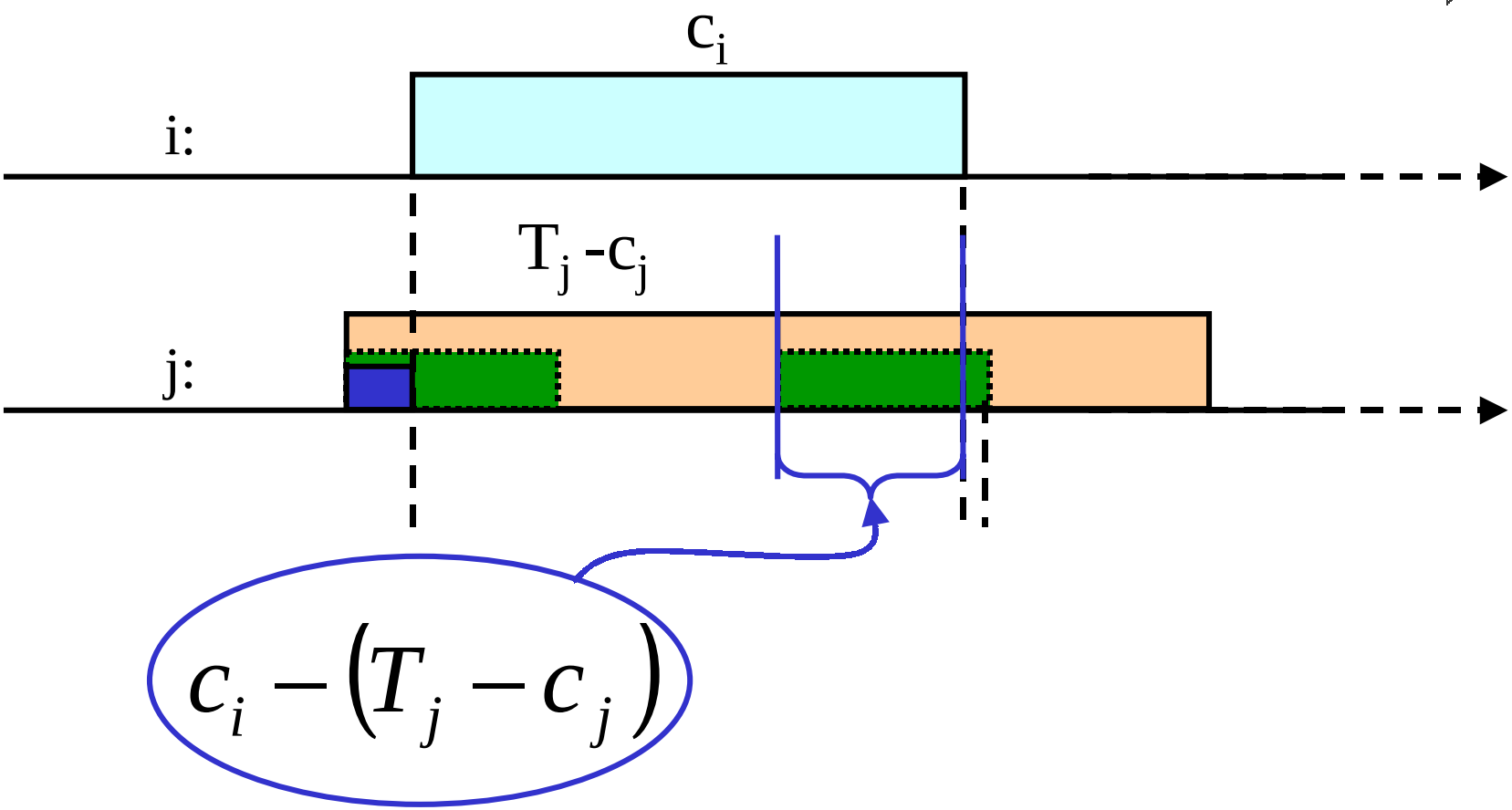
When happens 1 deadline miss



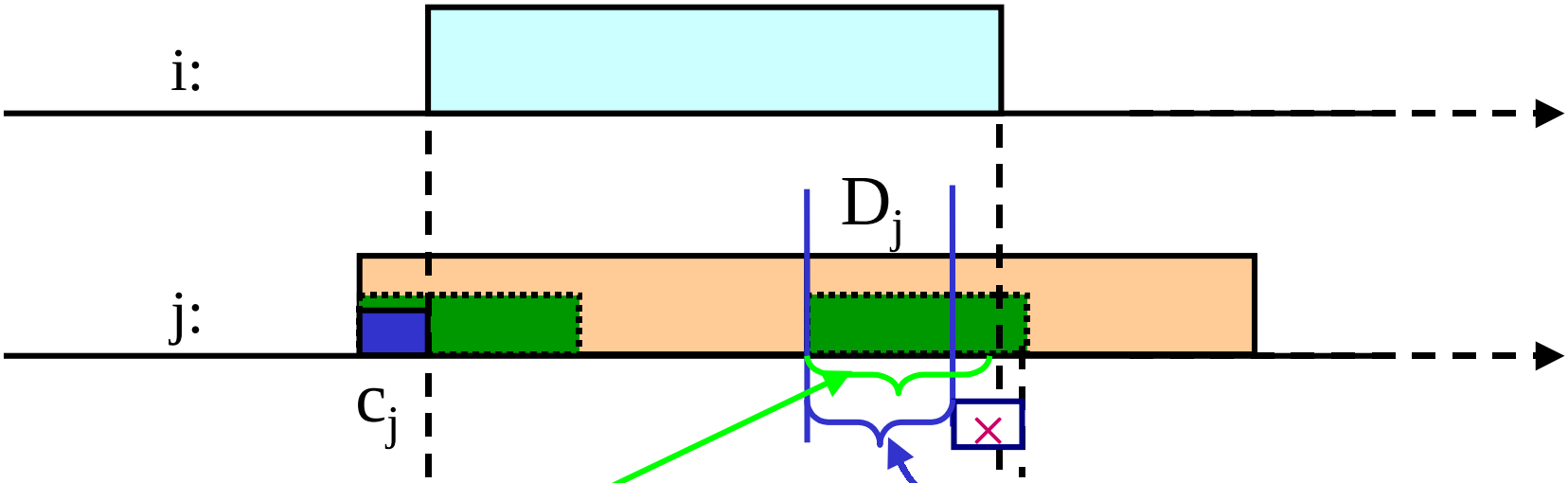
When happens 1 deadline miss



When happens 1 deadline miss



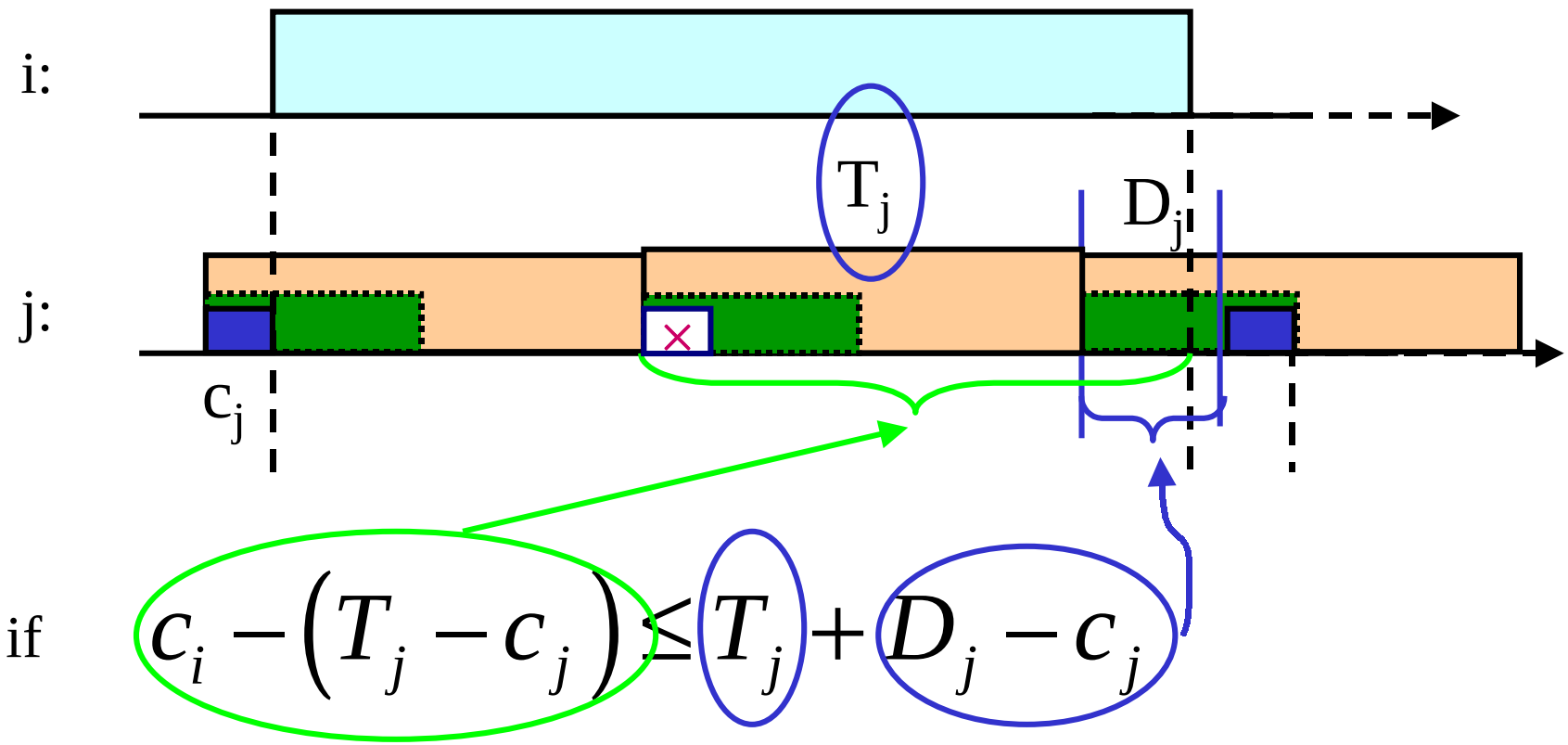
When happens 1 deadline miss



if $c_i - (T_j - c_j) > D_j - c_j$, then 1 miss in the best case.

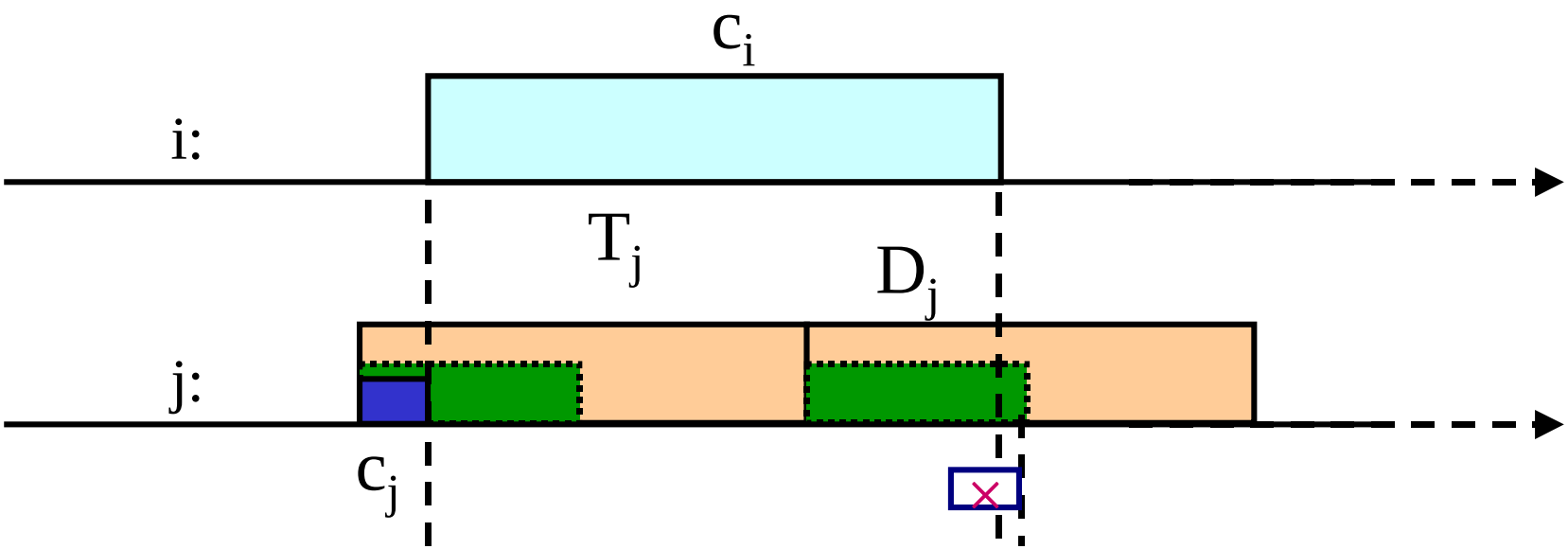
i.e., at least 1 miss

When happens 1 deadline miss



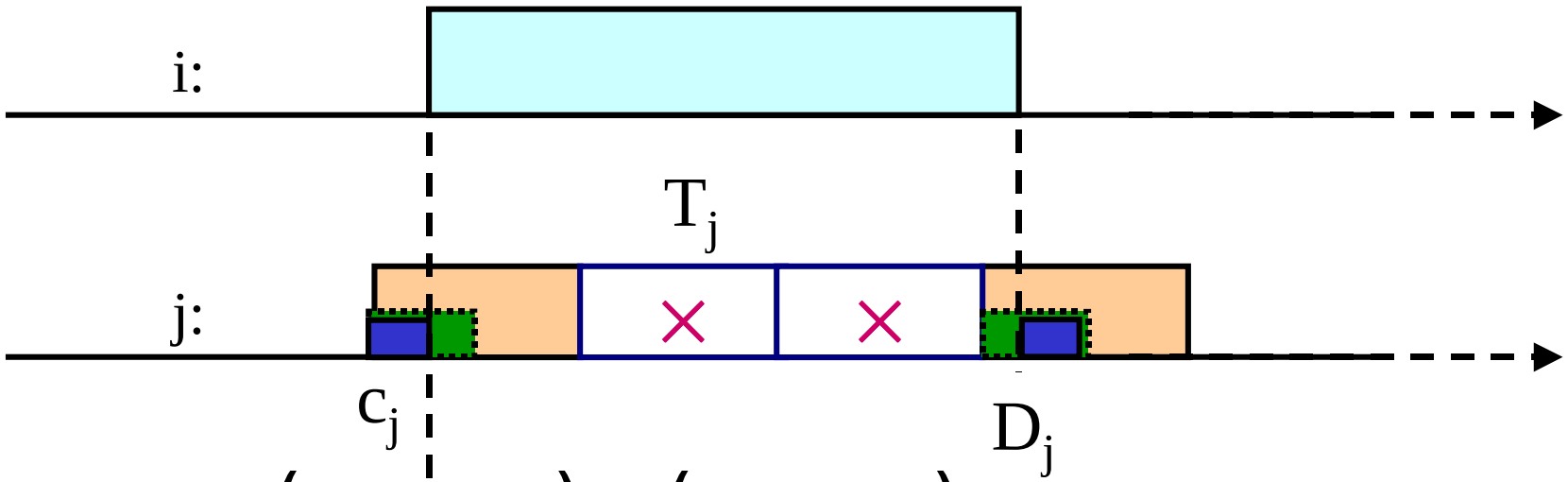
then no more than 1 miss

When happens 1 deadline miss



$$\begin{cases} c_i - (T_j - c_j) > D_j - c_j, & \text{At least 1 miss} \\ c_i - (T_j - c_j) \le T_j + D_j - c_j. & \text{No more than 1 miss} \end{cases}$$

Generalization to *n* deadline misses



$$\begin{cases} c_i - (T_j - c_j) > (n_{j,i} - 1) \cdot T_j + D_j - c_j, \\ c_i - (T_j - c_j) \le n_{j,i} \cdot T_j + D_j - c_j. \end{cases}$$

which gives $\boldsymbol{x\text{-}1 \le n < x}$

Minimum number of deadline misses

$$n_{j,i} = \left\lceil \frac{c_i + 2c_j - D_j}{T_j} \right\rceil - 1.$$

Minimum number of jobs that stream *j* misses while a job of stream *i* is being served

$$M = \left[\begin{array}{cccc} & & & \\ & n_{j,i} & & \\ & & & \\ & & & \end{array} \right] \quad N \times N$$

Necessary conditions

Necessary Load condition

$$\sum_{i=1}^{N} \left\lceil \frac{c_i}{T_i} \frac{m_i}{k_i} \right\rceil \leq 1$$

Mutual feasibility condition

$$\forall i, j \leq N,$$

$$n_{i,j} \leq k_i - m_i$$

New Necessary Condition

Matrix-DBP

	(m, k)	Execution time	Period Deadline	Initial μ-pattern	Priority
Stream a	(4, 5)	15ms	30ms	01111	2
Stream b	(2, 5)	2ms	5ms	00101	3

$$M = \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix}$$

$$\text{Priority}_{a,b} = \text{DBP}_a - n_{a,b} = 2$$

$$\text{Priority}_{b,a} = \text{DBP}_b - n_{b,a} = 1$$

Simulation scenario

	(m,k)-constraints	Job Process time: c_i	Period (T_i)/ Deadline(D_i)
Stream 0	(2,5)	8/c	12
Stream 1	(4,5)	10/c	20
Stream 2	(3,6)	2/c	5
Stream 3	(1,5)	4/c	6

Matrix of the scenario

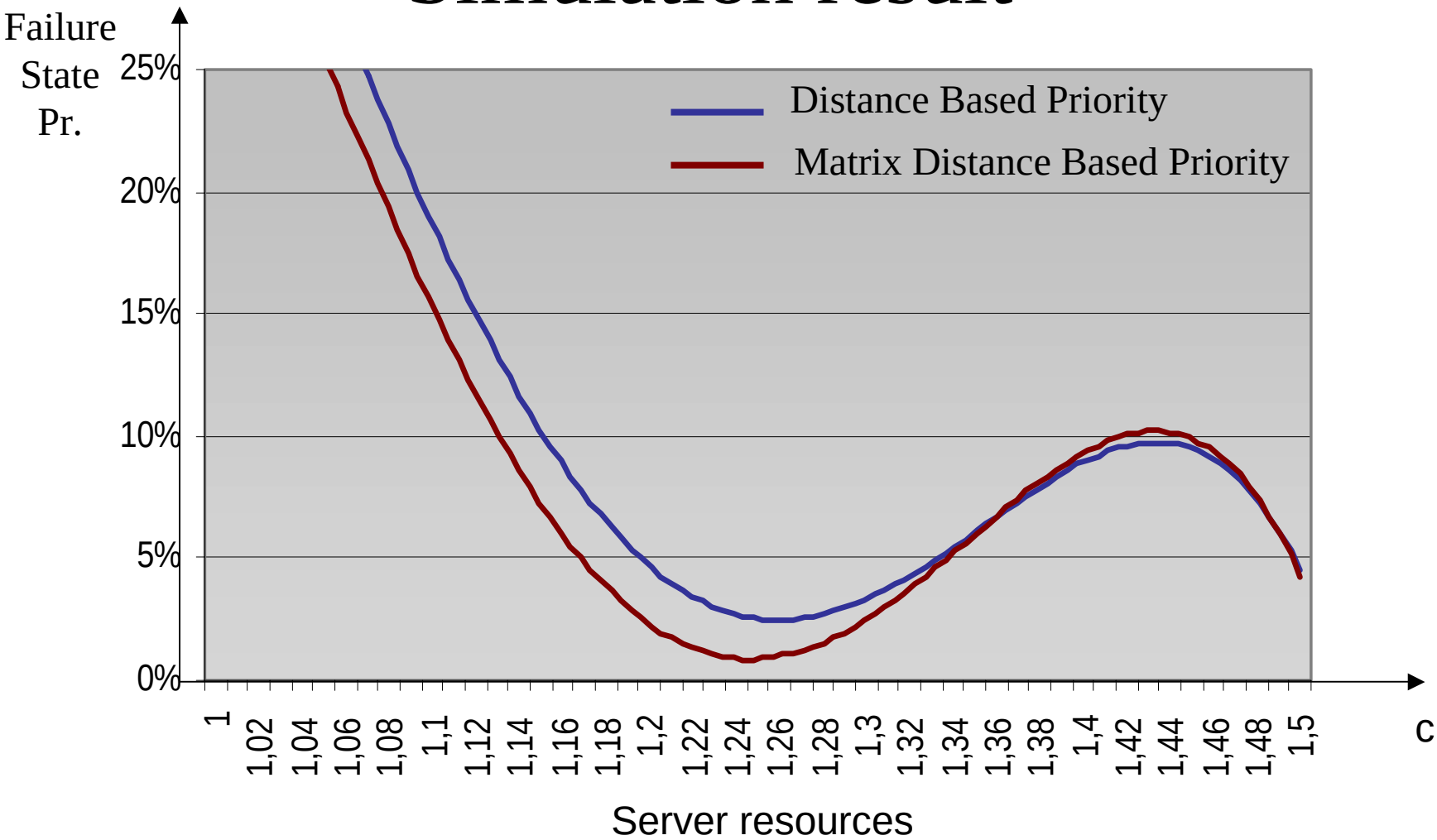
- When c = 1

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

- When c = 1.5
  Matrix-DBP
  becomes DBP

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Simulation result



Contributions on (m,k)-firm

- Pointed out the drawback of the classic DBP when it is applied to a more general real-time context
- Provided an additional necessary condition call mutual schedulability test
- Proposed matrix-DBP to correct DBP

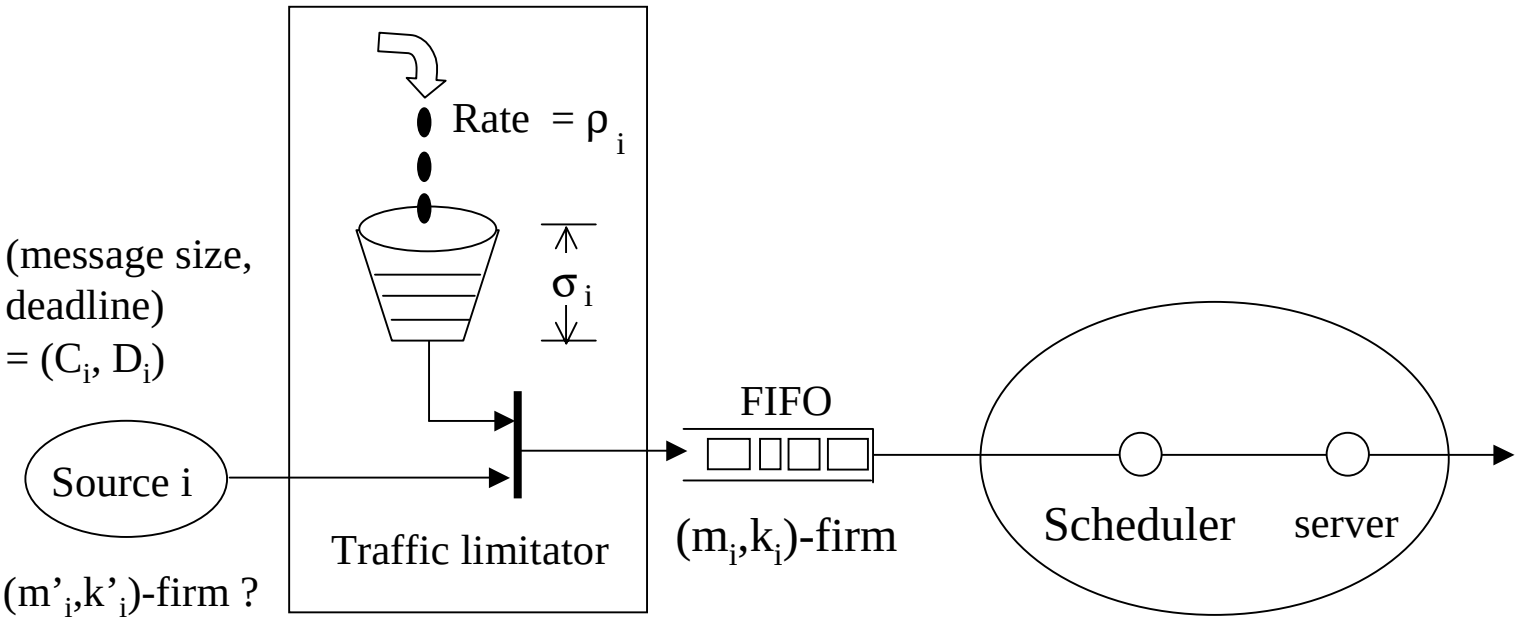
Part 4: Future work

On Matrix-DBP for (m,k)-firm

- Analysis of necessary condition on *c* for (m,k)-firm guarantee (i.e. find the value of *c* from which the system always has *0* failure state)
- Taking other criteria than the global failure state percentage (e.g., fairness in failure state percentage for each stream)
- Extension to multi-hops case by taking global deadline and local deadlines

On (m,k)-firm

- Enhancement of DBP for aperiodic streams
- (m,k)-firm for a (σ,ρ) upper bounded traffic



Application domains of (m,k)-firm

- Handling real-time traffic in Switched Industrial Ethernet (an Intranet) and in Internet
- Evaluating dependability in process control systems (in which we suppose that the state variables are over-sampled vs. Nyquist frequency)

Questions and remarks?