Synchronous DataFlow: A survival guide

Alix Munier-Kordon

LIP6
Sorbonne Université - CNRS
Paris

September 28, 2022

schedulingseminar.com

Join work with Claire Hanen, Jean-Marc Delosme, Olivier Marchetti, Mohamed Benazouz, Abir Benabid, Bruno Bodin, Cédric Klikpo, Jad Khatib, Youen Lesparre, Ning Tang...

Outline

Part 1: Introduction to SDF

1. Basic definitions: SDF, earliest schedule, repetition vector, consistency;
2. Precedence relations, useful tokens, normalisation;
3. Applications.

Part 2: two fundamental questions

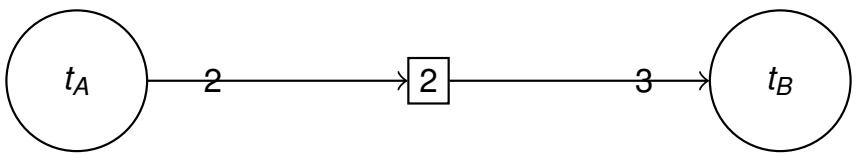
1. Schedulability (or Liveness);
2. Maximum throughput.

Conclusion and perspectives

Synchronous DataFlow Graphs (SDF)

A simple formalism introduced by Lee and Messerschmitt [LM] to model communications for Digital Signal Processor :

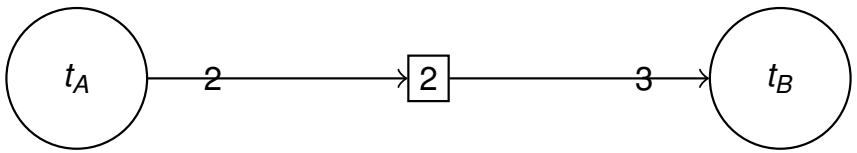
- Nodes $\rightarrow$ Actors (or tasks);
- Arcs $\rightarrow$ buffers;
- Tokens $\rightarrow$ data;
- Each arc $a = (t_A, t_B)$ is tri-valued by 3 integers $u(a) > 0$, $v(a) > 0$ and $M_0(a) \geq 0$;
- Each firing of actor t_A has duration $\ell(t_A)$.



$a = (t_A, t_B)$, $u(a) = 2$, $v(a) = 3$, $M_0(a) = 2$

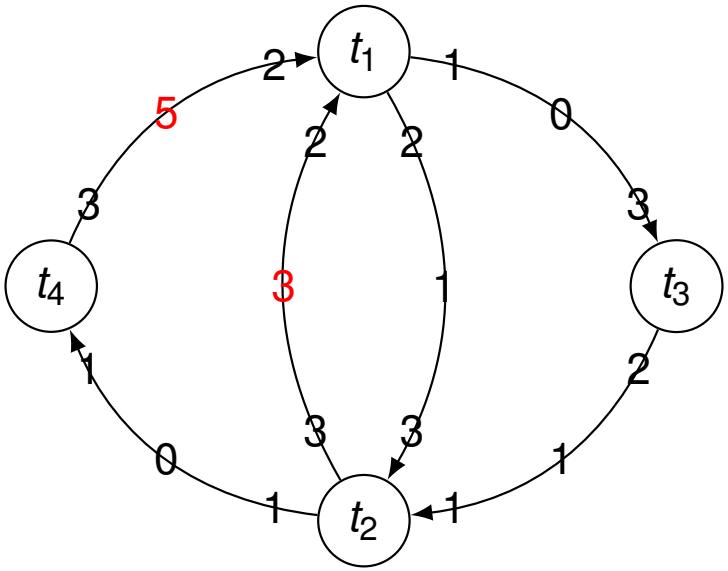
Firing rules of actors

A buffer between two actors t_A and t_B



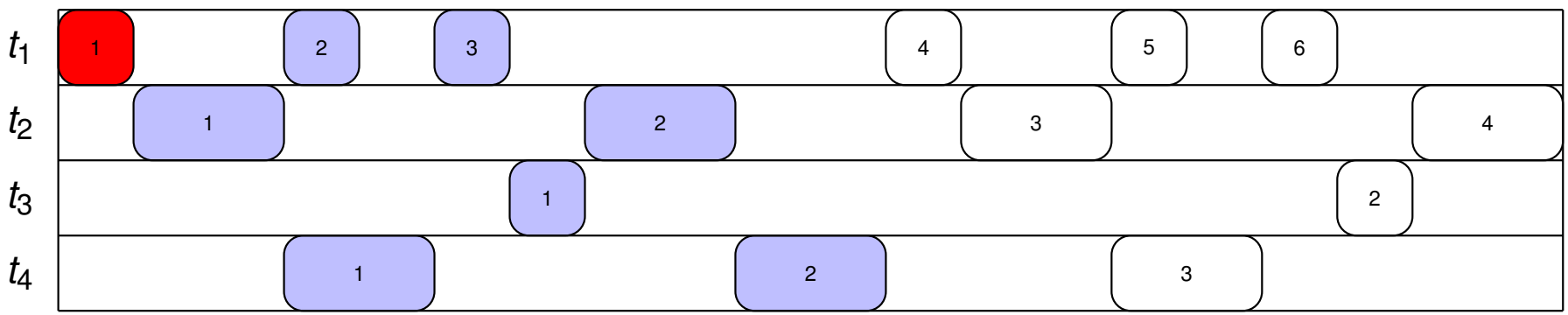
- The buffer contains initially $M_0(a)$ tokens;
- At each firing of t_A, 2 tokens are put in the buffer;
- t_B needs 3 tokens to fire once.
- If the number of tokens is not sufficient to fire t_B, t_B has to wait for executions of t_A.

Example of SDF and earliest schedule

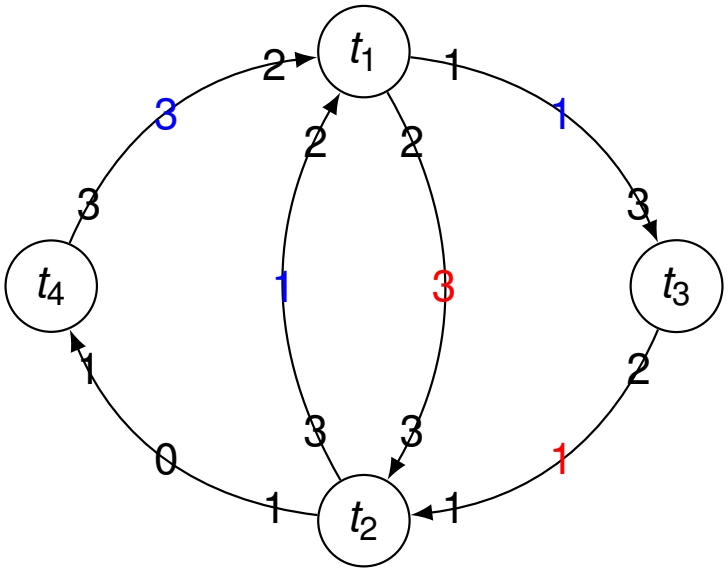


$M_0(t_1, t_3) = 0$
$M_0(t_3, t_2) = 1$
$M_0(t_1, t_2) = 1$
$M_0(t_2, t_1) = 3$
$M_0(t_4, t_1) = 5$
$M_0(t_2, t_4) = 0$

$\ell(t_1) = 1$
$\ell(t_2) = 2$
$\ell(t_1) = 3$
$\ell(t_4) = 2$

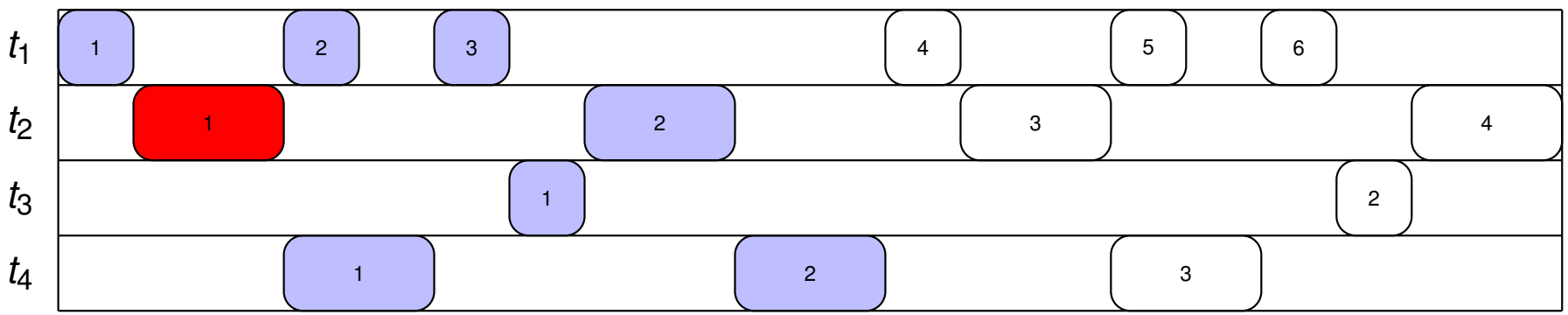


Example of SDF and earliest schedule

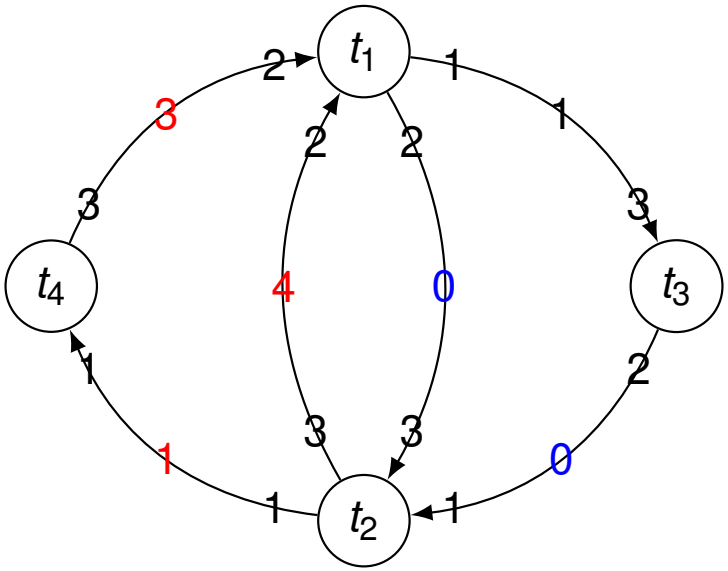


$$M_0(t_1, t_3) = 0$$
$$M_0(t_3, t_2) = 1$$
$$M_0(t_1, t_2) = 1$$
$$M_0(t_2, t_1) = 3$$
$$M_0(t_4, t_1) = 5$$
$$M_0(t_2, t_4) = 0$$

$$\ell(t_1) = 1$$
$$\ell(t_2) = 2$$
$$\ell(t_1) = 3$$
$$\ell(t_4) = 2$$

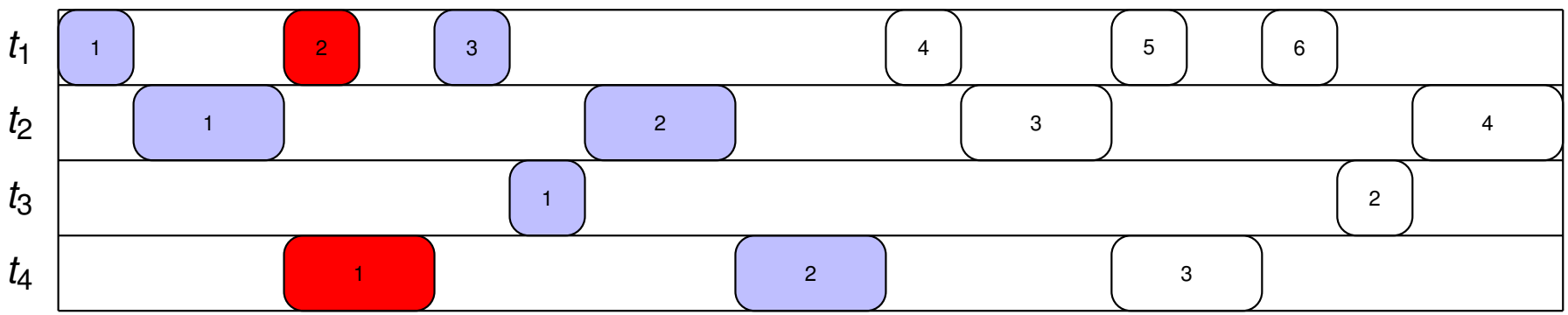


Example of SDF and earliest schedule

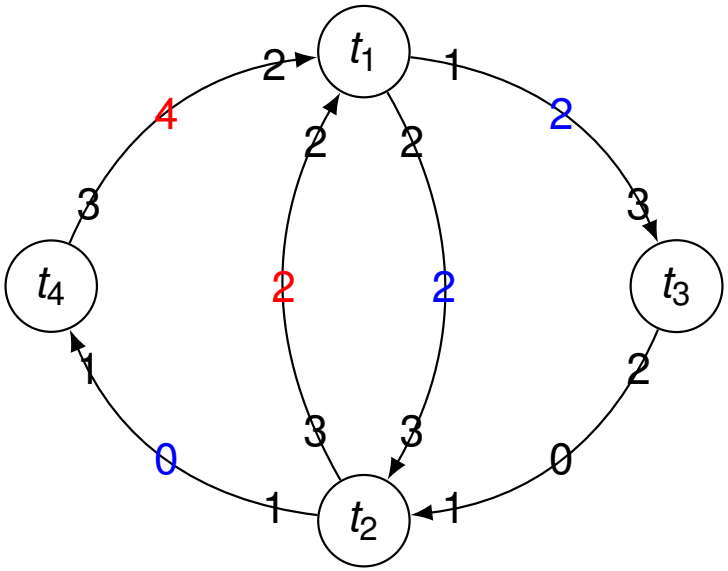


$M_0(t_1, t_3) = 0$
$M_0(t_3, t_2) = 1$
$M_0(t_1, t_2) = 1$
$M_0(t_2, t_1) = 3$
$M_0(t_4, t_1) = 5$
$M_0(t_2, t_4) = 0$

$\ell(t_1) = 1$
$\ell(t_2) = 2$
$\ell(t_1) = 3$
$\ell(t_4) = 2$

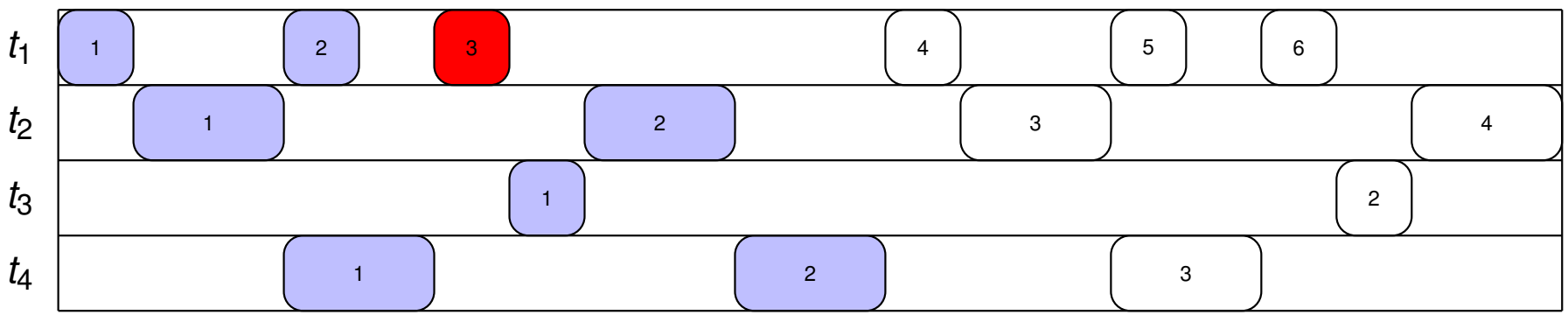


Example of SDF and earliest schedule

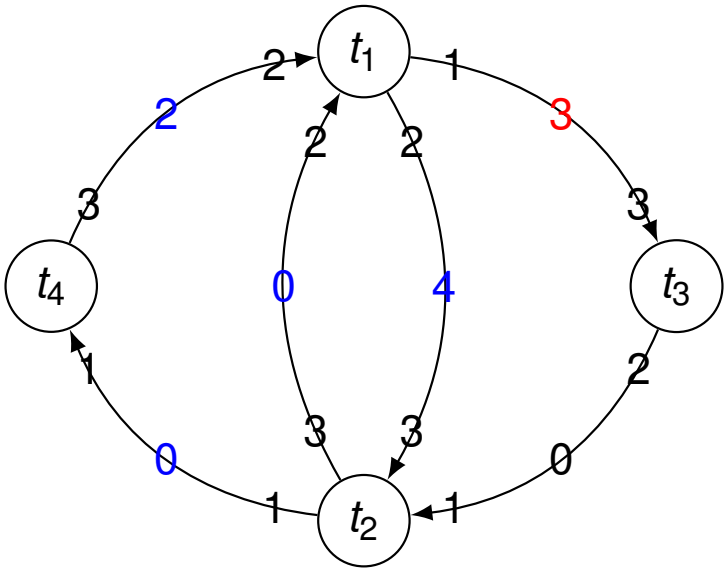


$M_0(t_1, t_3) = 0$
$M_0(t_3, t_2) = 1$
$M_0(t_1, t_2) = 1$
$M_0(t_2, t_1) = 3$
$M_0(t_4, t_1) = 5$
$M_0(t_2, t_4) = 0$

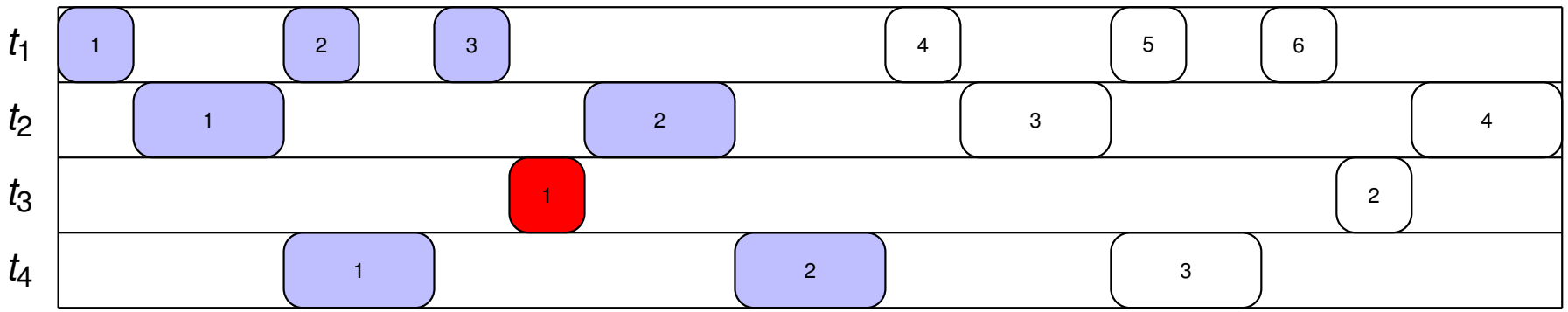
$\ell(t_1) = 1$
$\ell(t_2) = 2$
$\ell(t_1) = 3$
$\ell(t_4) = 2$



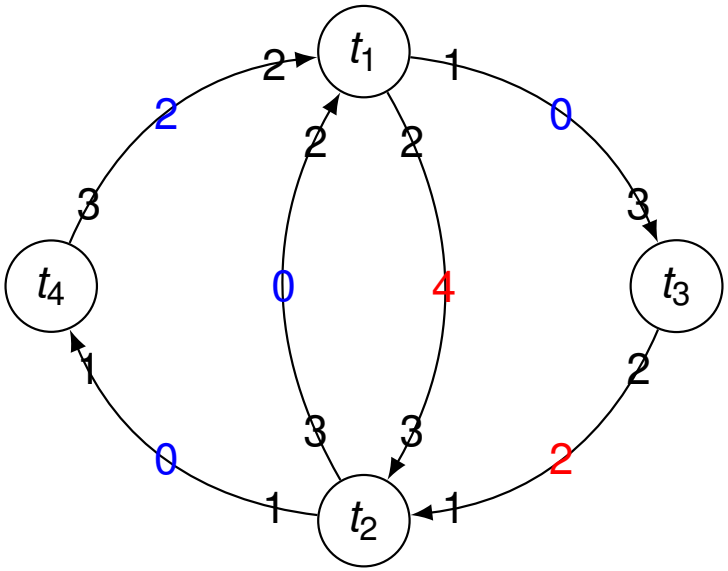
Example of SDF and earliest schedule



$$M_0(t_1, t_3) = 0 \qquad \ell(t_1) = 1$$
$$M_0(t_3, t_2) = 1 \qquad \ell(t_2) = 2$$
$$M_0(t_1, t_2) = 1 \qquad \ell(t_1) = 3$$
$$M_0(t_2, t_1) = 3 \qquad \ell(t_4) = 2$$
$$M_0(t_4, t_1) = 5$$
$$M_0(t_2, t_4) = 0$$

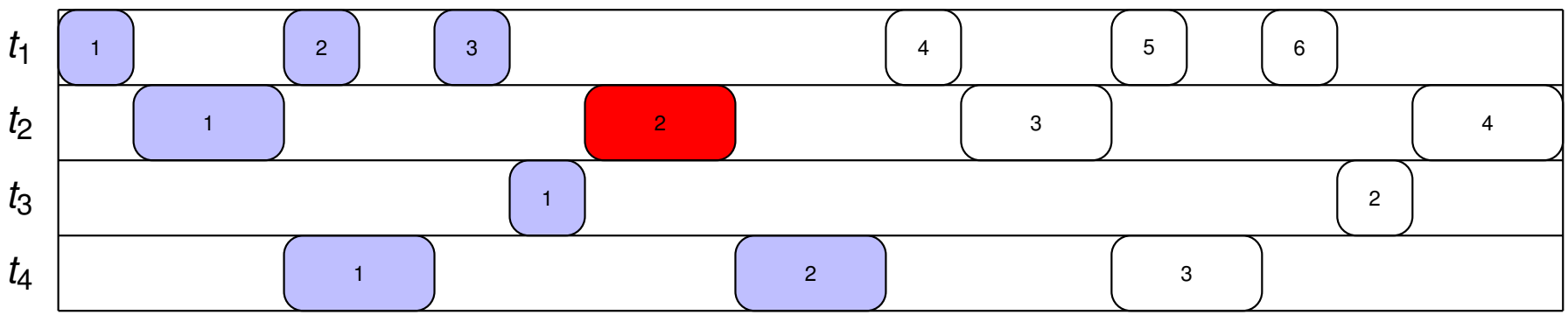


Example of SDF and earliest schedule

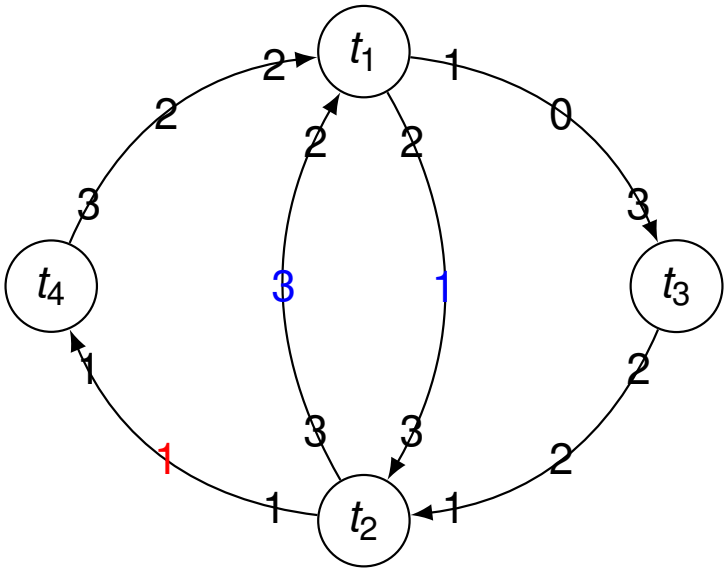


$M_0(t_1, t_3) = 0$
$M_0(t_3, t_2) = 1$
$M_0(t_1, t_2) = 1$
$M_0(t_2, t_1) = 3$
$M_0(t_4, t_1) = 5$
$M_0(t_2, t_4) = 0$

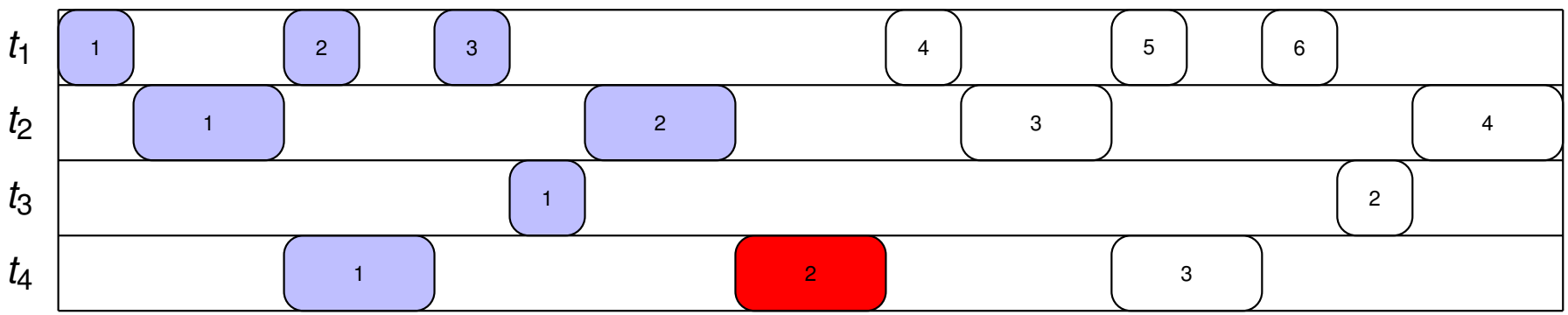
$\ell(t_1) = 1$
$\ell(t_2) = 2$
$\ell(t_1) = 3$
$\ell(t_4) = 2$



Example of SDF and earliest schedule

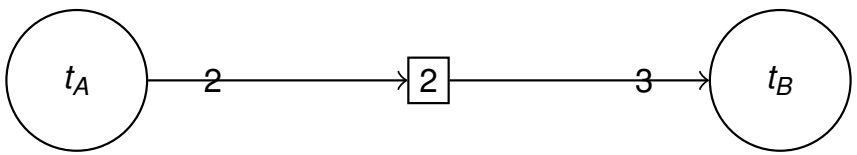


$$M_0(t_1, t_3) = 0 \qquad \ell(t_1) = 1$$
$$M_0(t_3, t_2) = 1 \qquad \ell(t_2) = 2$$
$$M_0(t_1, t_2) = 1 \qquad \ell(t_1) = 3$$
$$M_0(t_2, t_1) = 3 \qquad \ell(t_4) = 2$$
$$M_0(t_4, t_1) = 5$$
$$M_0(t_2, t_4) = 0$$



Repetition vector

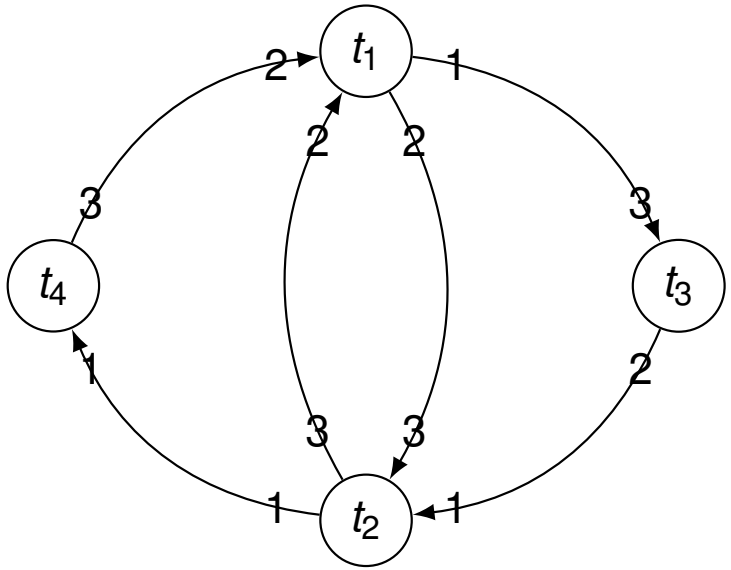
A buffer between two actors t_A and t_B



Balance equation: $N_A \times 2 = N_B \times 3$

After $N_A = 3$ firings of t_A and $N_B = 2$ firings of t_B, the number of tokens (in the buffer) returns to it initial number $M_0(a)$.

Repetition vector



$$N_1 = 3N_3$$
$$2N_3 = N_2$$
$$2N_1 = 3N_2$$
$$N_2 = N_4$$
$$3N_4 = 2N_1$$

The minimum positive integer solution is $N_1 = 3$, $N_2 = 2$, $N_3 = 1$ and $N_4 = 2$.

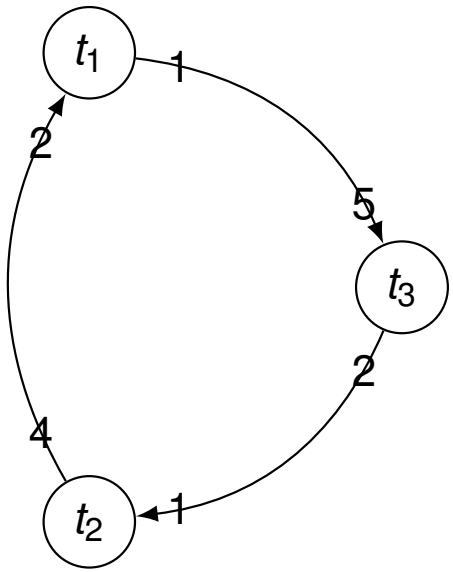
Definition (Lee and Messerschmitt [LM])

An SDF G is consistent if a repetition vector N exists.

Definition (Weight of a circuit)

The weight of a circuit c of an SDF is $W(c) = \Pi_{a \in c} \frac{u(a)}{v(a)}$.

Authors usually consider consistent SDF. Why ?



$$W(c) = \frac{1 \times 2 \times 4}{5 \times 1 \times 2} = \frac{4}{5}$$

Since $W(c) < 1$, the whole number of tokens in the circuit is (roughly) decreasing, and a deadlock will occured !

Authors usually consider consistent SDF. Why ?

Let c be a circuit of an SDF G.

- If $W(c) < 1$, the circuit consumes tokens and a deadlock will happen, even for large values of the initial marking M_0;
- If $W(c) > 1$, the circuit produces tokens and the markings tends to infinity (this in not acceptable for real-life systems).

We only consider SDF G such that $W(c) = 1$ for any circuit c of G.

Definition

A graph G is unitary if any circuit c of G verifies $W(c) = 1$.

Theorem (Lee and Messerschmitt [LM])

An SDF G is unitary iff G is consistent.

Checking the consistency of a SDF is polynomial

Each arc $a = (t_A, t_B)$ of the SDF $\mathcal{G}$ is associated to the balance equation $N_A \times u(a) = N_B \times v(a)$.

Let us consider the valued graph $\mathcal{H} = (\mathcal{T}, E, V)$ builds as follows:

- Nodes of $\mathcal{H}$ are actors (or tasks);
- Each buffer $a = (t_A, t_B)$ is associated to the arcs $\alpha = (t_A, t_B)$ and $\alpha' = (t_B, t_A)$ valued respectively by $V(\alpha) = \log \frac{u(a)}{v(a)}$ and $V(\alpha') = -V(\alpha)$;
- There exists a repetition vector N for the initial SDF $\mathcal{G}$ iff there exists a vector $\delta \in \mathbb{R}^{|\mathcal{T}|}$ such that, for each arc $\alpha = (t_A, t_B) \in E$, $\delta_B - \delta_A \geq V(\alpha)$.

The existence and the determination of δ can be computed by Bellman-Ford Algorithm in time complexity $\mathcal{O}(|\mathcal{T}| \times |E|)$.

A first algorithm for the existence of a feasible schedule

Theorem (Lee and Messerschmitt [LM])

Suppose that G is a consistent SDF. Then, there exists a feasible schedule iff each task t_i can be executed at least N_i times. Moreover, once each task t_i is executed exactly N_i times (if it is possible), the systems returns in its initial state, ı.e the current marking of the buffers equals its initial value.

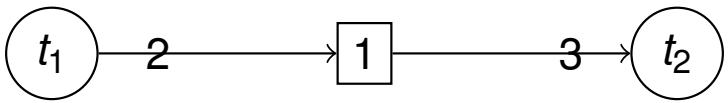
Checking the feasibility of a SDF requires to execute tasks at least N_i times.
The complexity of this algorithm is proportional to $\Sigma_{t_i \in \mathcal{T}} N_i$, which is not polynomial (and may be huge for real-life applications) !!

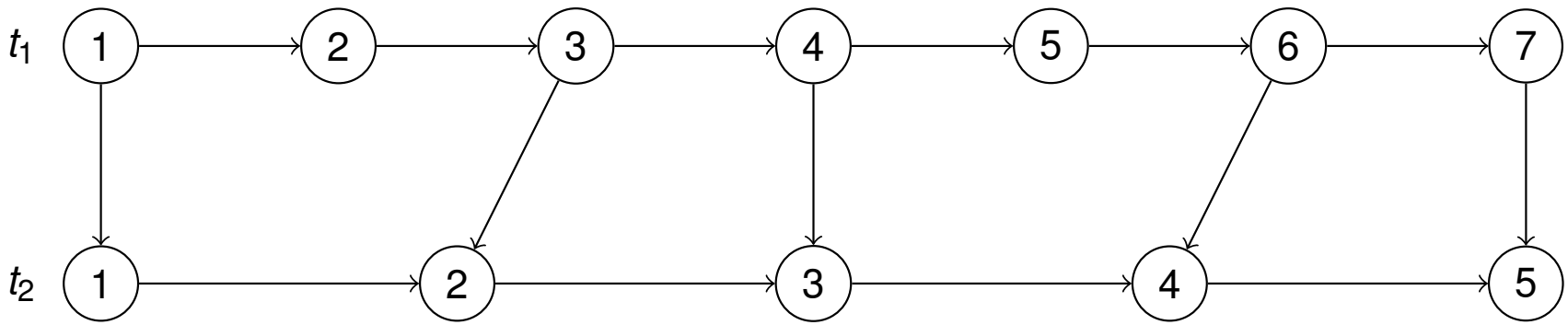
The complexity of the feasibility of a SDF is of unknown complexity

Precedence relations associated to a SDF

Definition

An actor t_i is non re-entrant if two executions of t_i cannot overlap.





The couples of indexes (n_1, n_2) such that there exists a precedence relation are then $\{(1+3k, 1+2k), k \in \mathbb{N}\}$ and $\{(3+3k, 2+2k), k \in \mathbb{N}\}$.

Precedence constraints associated to a SDF

Theorem (Munier [Mun93])

Let t_i and t_j be two re-entrant tasks. A FIFO queue $a = (t_i, t_j) \in A$ with initially $M_0(a)$ tokens models a precedence relation between the n_ith execution of t_i and the n_jth execution of t_j iff

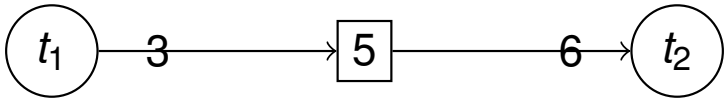
$$u(a) > M_0(a) + u(a) \cdot n_i - v(a) \cdot n_j \geq \max\{u(a) - v(a), 0\}.$$

For our example, $u(a) = 2$, $v(a) = 3$ and $M_0(a) = 1$;

$$2 > 1 + 2n_1 - 3n_2 \geq 0$$

1. If $1 + 2n_1 - 3n_2 = 0$, then $(n_1, n_2) = (1 + 3k, 1 + 2k)$, $k \in \mathbb{N}$;
2. If $1 + 2n_1 - 3n_2 = 1$, then $(n_1, n_2) = (3 + 3k, 2 + 2k)$, $k \in \mathbb{N}$.

Useful tokens



There is always at least 2 tokens in the buffer. They thus can be removed without any influence on the precedence constraints !

Definition

A useful initial marking is such that, for any arc $a = (t_i, t_j)$, $M_0(a)$ is a multiple of $\gcd(u(a), v(a))$.

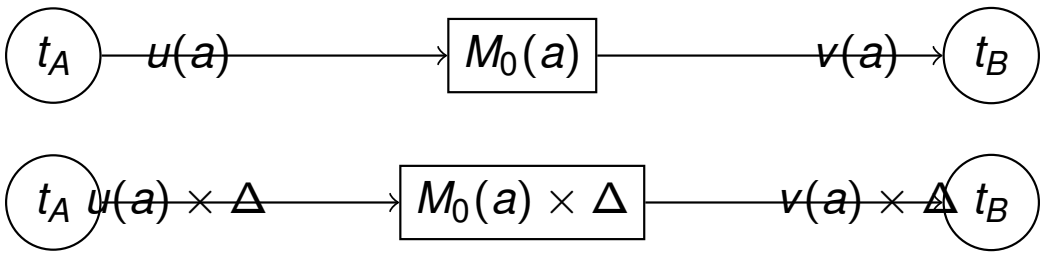
Theorem (Marchetti et Munier [MMK09])

Useful initial markings are dominant. Moreover, the initial marking $M_0(a)$ of a may be replaced by $\lfloor \frac{M_0(a)}{\gcd(u(a),v(a))} \rfloor \times \gcd(u(a), v(a))$ without any influence on the precedence constraints associated to a.

For our example, $\gcd(3, 6) = 3$ and the equivalent useful initial marking is $\lfloor \frac{5}{3} \rfloor \times 3 = 3$.

Only consistent SDF with useful initial markings are considered

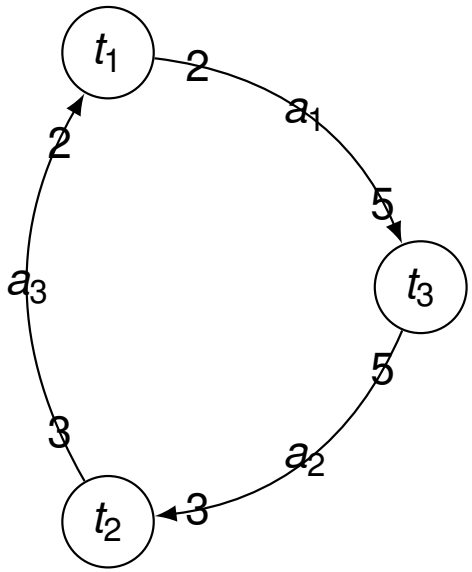
Scaling



Theorem (Marchetti et Munier [MMK09])

Let us consider $\Delta \in \mathbb{Q}^+ - \{0\}$ *such that* $u(a) \times \Delta \in \mathbb{N}$, $v(a) \times \Delta \in \mathbb{N}$ *and* $M_0(a) \times \Delta \in \mathbb{N}$. *Then, the arc* $a = (t_i, t_j)$ *may be replaced by* $a' = (t_i, t_j)$ *with* $u(a') = u(a) \times \Delta$, $v(a') = v(a) \times \Delta$ *and* $M_0(a') = M_0(a) \times \Delta$ *without any influence on the precedence constraints associated to a.*

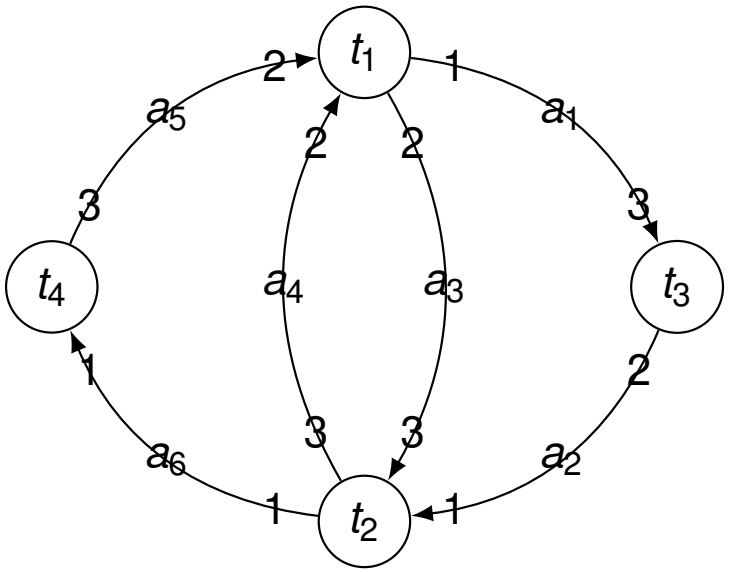
Normalised circuit



$$Z_1 = 2 = u(a_1) = v(a_3)$$
$$Z_2 = 3 = u(a_3) = v(a_2)$$
$$Z_3 = 5 = u(a_2) = v(a_1)$$

The number of tokens is constant !

Normalisation of a consistent graph



Find α_j, $j \in \{1, \cdots, 6\}$

and Z_1, $i \in \{1, \cdots, 4\}$

$$Z_1 = \alpha_1 = 2\alpha_3 = 2\alpha_4 = 2\alpha_5$$

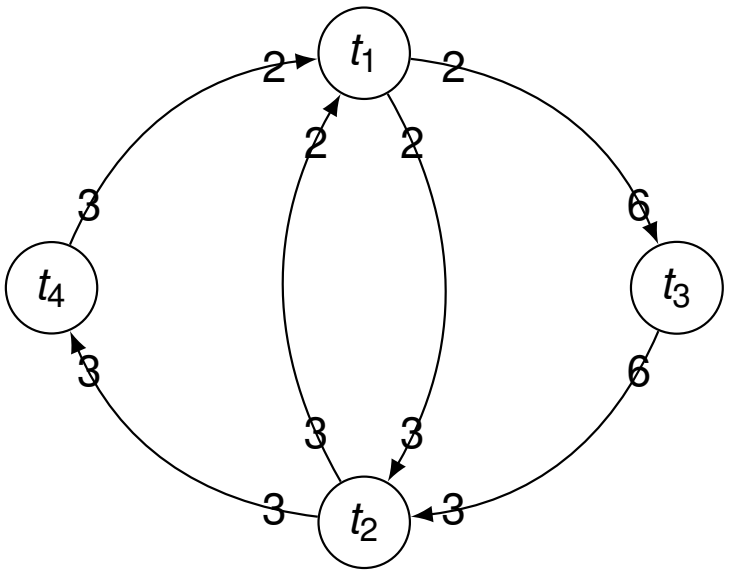
$$Z_2 = \alpha_2 = 3\alpha_3 = 3\alpha_4 = \alpha_6$$

$$Z_3 = 3\alpha_1 = 2\alpha_2$$

$$Z_4 = 3\alpha_5 = \alpha_6$$

The smallest solution is $\alpha_1 = 2$, $\alpha_2 = 3$, $\alpha_3 = 1$, $\alpha_4 = 1$, $\alpha_5 = 2$, $\alpha_6 = 3$, $Z_1 = 2$, $Z_2 = 3$, $Z_3 = 6$ and $Z_4 = 3$.

Normalisation of a consistent graph



$N_1 = 3, N_2 = 2$

$N_3 = 1, N_4 = 2$

We observe that $Z_1 \times N_1 = Z_2 \times N_2 = Z_3 \times N_3 = Z_4 \times N_4 = 6$.

Normalisation of a consistent graph

Theorem (Marchetti et Munier [MMK09])

Let suppose that G is a consistent graph and N its smallest repetition vector. Let consider the values $M = lcm(N_i)$ and for any task t_i, $Z_i = \frac{M}{N_i}$. Then, the normalized SDF G' built from G by setting, for any arc $a = (t_i, t_j)$, $u'(a) = Z_i$, $v'(a) = Z_j$ and $M'_0(a) = M_0(a) \times \frac{Z_i}{u(a)}$ generates the same set of precedence constraints as G.

Every consistent graph can be normalized. The complexity of the algorithm is linear in the number of arcs if the repetition vector is known.

Moreover, a normalized graph is clearly consistent.

Our studies are thus restricted to normalized SDF

Applications modelled using a SDF

SDF can be considered directly to model streaming applications.

- tasks: repetitive treatments;
- arcs: FIFO buffers between two communication tasks;
- initial markings: tokens initially in buffers.

Some interesting questions:

- Is the SDF is consistent ? Schedulable (Liveness) ?
- What is its maximum throughput ?
- What is the influence of the size of the buffers on the throughput ?
- How can I execute the SDF on a fixed given multi-processors architecture with a maximum throughput ?

Examples of applications

Table: Benchmark coming from industry applications

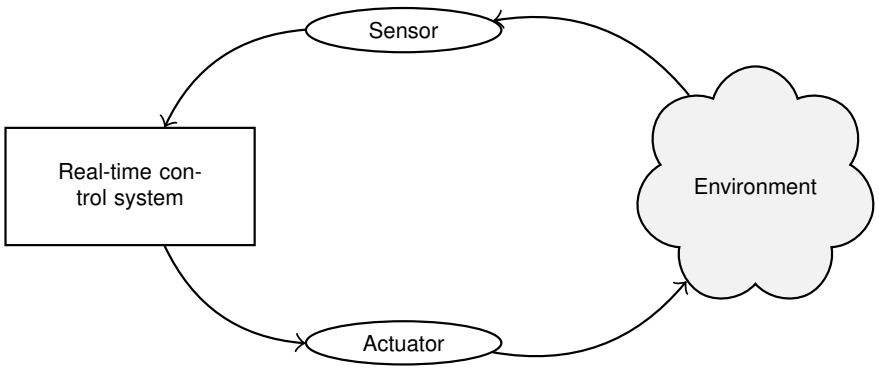
Application	Actors	Buffers	$\sum_{t_i \in T} N_i$
BlackScholes	41	40	11895
JPEG2000	240	703	802971540
Echo	38	82	336024
Pdetect	58	76	3883200
H264 Encoder	665	3128	24094980

The time and space complexity of the algorithms for the previous questions depends usually on the size $\sum_{t_i \in T} N_i$ of the repetition vector.

$\Rightarrow$ They cannot be considered for real-life applications such as JPEG2000.

Embedded Real-time Reactive System

1. Reactive = permanently in interaction with its environment
2. Real-time = subject to (hard) timing-constraints
3. Embedded = automotive, avionics, cellular phone,. . .



Multi-periodic Tasks

1. $\mathcal{T} = \{t_1, \ldots, t_n\}$ is a set of multi-periodic tasks following the Liu and Layland model.
2. Each task $t_i \in \mathcal{T}$ is associated to the triple (r_i, D_i, T_i)
   - $r_i \stackrel{\text{def}}{=}$ release date (the offset) of the first execution of t_i;
   - $D_i \stackrel{\text{def}}{=}$ the relative deadline of t_i;
   - $T_i \stackrel{\text{def}}{=}$ the period of t_i.
3. For any positive interger n
   - $\langle t_i, n \rangle \stackrel{\text{def}}{=} n$th execution of t_i;
   - $\mathcal{S}(t_i, n) \stackrel{\text{def}}{=} r_i + (n-1) \times T_i$ is the release time of $\langle t_i, n \rangle$;
   - $D_i + (n-1) \times T_i$ is the deadline of $\langle t_i, n \rangle$.

Logical Execution Time paradigm

Definition (LET paradigm)

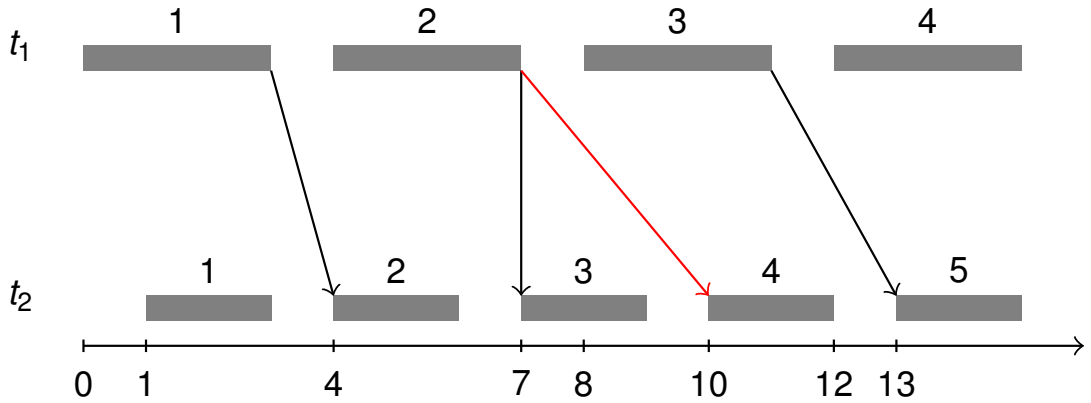
Let suppose that $(t_i, t_j) \in \mathcal{T}^2$ such that each execution of t_i sends data to executions of t_j using a shared memory.

1. Each execution of t_i writes the data at its periodic deadline,
2. each execution of t_j reads the data at its periodic release date.

- Introduced by Kisrsch and Sokolova [KS12] and implemented in the time-triggered programming language Giotto;
- Communications are fixed before the executions.

Communications between executions of two adjacent tasks

Let consider $e = (t_1, t_2) \in E$, with $(r_1, D_1, T_1) = (0, 3, 4)$ and $(r_2, D_2, T_2) = (1, 2, 3)$.



There exists a dependence from $\langle t_1, 2\rangle$ to $\langle t_2, 4\rangle$ since $\langle t_1, 3\rangle$ ends before the starting time of $\langle t_2, 4\rangle$.

Characterization of the dependencies

Theorem (Munier and Tang [KT20])

Let suppose that $e = (t_i, t_j) \in E$, $gcd_T^e = \gcd(T_i, T_j)$ *and* $M^e = T_j + \left\lceil \frac{r_i - r_j + D_i}{gcd_T^e} \right\rceil \times gcd_T^e$.

For any pair $(\nu_i, \nu_j) \in \mathbb{N} - \{0\}^2$, *there exists a dependency from* $\langle t_i, \nu_i \rangle$ *to* $\langle t_j, \nu_j \rangle$ *iff* $T_i \geq M^e + T_i\nu_i - T_j\nu_j > 0$.

For the previous example, $e = (t_1, t_2)$, $(r_1, D_1, T_1) = (0, 3, 4)$, $(r_2, D_2, T_2) = (1, 2, 3)$. Thus, $gcd_T^e = 1$, $M^e = 5$ and we get

$$4 \geq 5 + 4\nu_1 - 3\nu_2 > 0.$$

Challenging questions associated to LET paradigm

An instance of the problem is given by a set of multi-periodic tasks $\mathcal{T} = \{(r_i, D_i, T_i)\}$ and a directed graph $G = (\mathcal{T}, E)$ modelling LET communications.

- The liveness is necessarliy true;
- The throughput is fixed;
- A first challenging question is the evaluation of the latency of the system, ie. the maximum lenght between the execution of a task corresponding to a sensor (input task) to the execution of a task corresponding to an actuator (output task);
- Another question is the execution of the system on a fixed multi-processor architecture.

Schedulability

Let G be a normalized strongly connected initially marked SDF. Is there an infinite feasible schedule *i.e.* each task can be executed a non bounded number of times?

First solution: schedule the tasks as soon as possible until each task t_i is executed at least N_i times. Following Lee and Messerschmitt [LM], if such a schedule exists, G is schedulable.

The number of tasks of this schedule is upper-bounded by $\Sigma_{t_i \in T} N_i$. The algorithm is not polynomial !

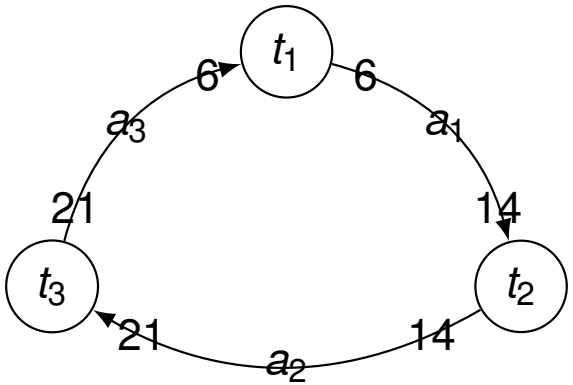
Now, if G is not schedulable, then there exists a circuit c for which each arc $a = (t_i, t_j)$ has at most $Z_j - \gcd(Z_i, Z_j)$ tokens, thus:

Theorem (Marchetti et Munier [MMK09])

Let G be a normalized strongly connected initially marked SDF. If, for any circuit c from G, the inequality $\sum_{a=(t_i,t_j)\in c} M_0(a) > \sum_{a=(t_i,t_j)\in c}(Z_j - \gcd(Z_i, Z_j))$ holds, then G is schedulable.

This condition can be checked in $\mathcal{O}(|T||A|)$ using Bellman-Ford algorithm coupled with a DFS.

The condition is not necessary !



$$M_0(a_1) = 0$$
$$M_0(a_2) = 0$$
$$M_0(a_3) = 21$$

1. $N_1 = 7$, $N_2 = 3$ and $N_3 = 2$.
2. The sequence $t_3 t_1 t_1 t_1 t_2 t_3 t_1 t_1 t_1 t_1 t_2 t_2$ can be repeated infinitely, the circuit is thus schedulable.
3. $\sum_{t_l \in c} Z_i - \sum_{a \in c} M_0(a) - \sum_{a=(t_i,t_j) \in c} \gcd(Z_i, Z_j) = 41 - 28 - 12 > 0$, the sufficient condition does not hold !

Problem Formulation

Let G a strongly connected normalizes SDF.

Definition

A schedule is a function $s : \mathcal{T} \times \mathbb{N}^\star \to \mathbb{N}$ where $s(t, n)$ denotes the nth execution of t. s is feasible is the numbers of tokens in any buffer remain non negative.

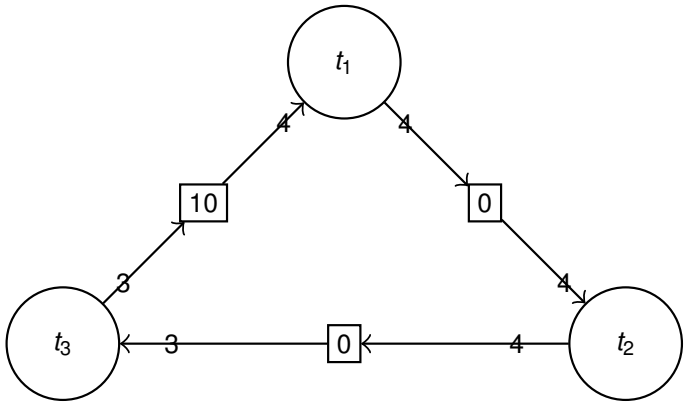
Definition

Let s be a feasible schedule. The throughput of t following s is $\lambda^s(t) = \lim_{n\to\infty} \frac{n}{s(t,n)}$.

If G is consistent and strongly connected, $\forall t \in \mathcal{T}$, $\lambda^s(t) \times Z_t = \lambda^s$ is a constant.

How to evaluate efficiently the maximum throughput of G?

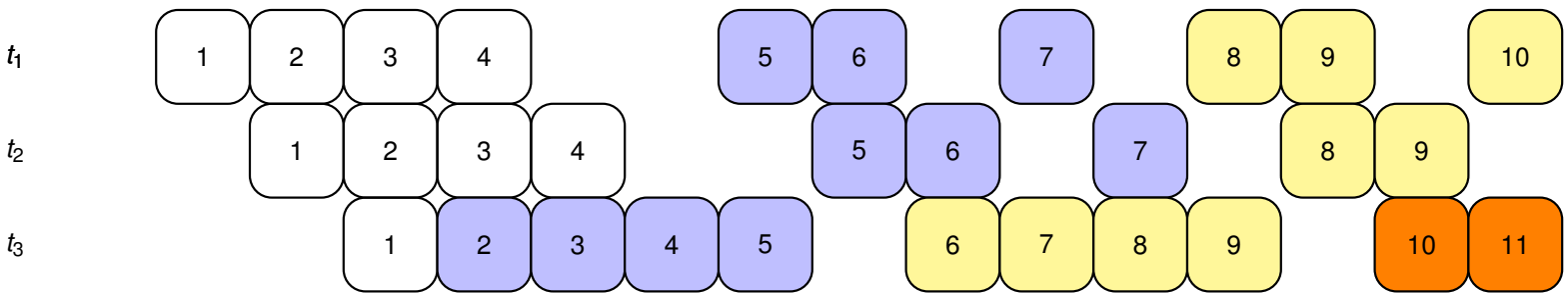
Example of the earliest schedule



$N_1 = 3$

$N_2 = 3$

$N_3 = 4$



An K-Periodic schedule of (exact) maximum throughput $\lambda^\star = \frac{12}{5}$. The throughput of the actors are $\lambda(t_1)^\star = \frac{3}{5}$, $\lambda^\star(t_2) = \frac{3}{5}$ and $\lambda^\star(t_2) = \frac{4}{5}$.

K-Periodic schedules

Definition

A schedule s is K-periodic if for any task $t_i \in \mathcal{T}$, there exists the integers $q_i \geq 0$ and $K_i > 0$ and a vector $w_i \in \mathbb{Q}^{+n}$ such that

$$\forall q > q_i, s(t_i, q + K_i) = s(t_i, q) + w_i$$

The throughput of t_i is $\lambda^s(t_i) = \frac{K_i}{w_i}$ while the throughput of the schedule equals $\lambda^s = \frac{K_i Z_i}{w_i}$.

1. If $q_i = 0$ for any task t_i, the schedule is strictly K-periodic;
2. If $K_i = 1$ for any task t_i, the schedule is periodic;
3. The size of a K-periodic schedule is proportional to $\sum_{t_i \in \mathcal{T}} K_i$.

Theorem

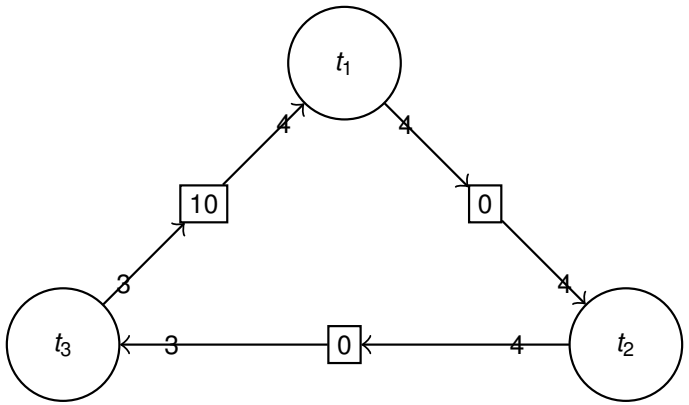
Let G be a schedulable SDF. The earliest schedule is K-periodic, such that, for any task $t_i \in \mathcal{T}$, N_i is a divisor of K_i. Moreover, there exists a strictly K-periodic schedule with, $\forall t_i \in \mathcal{T}$, $K_i = N_i$ which throughput is maximum (equal to the throughput of the earliest schedule).

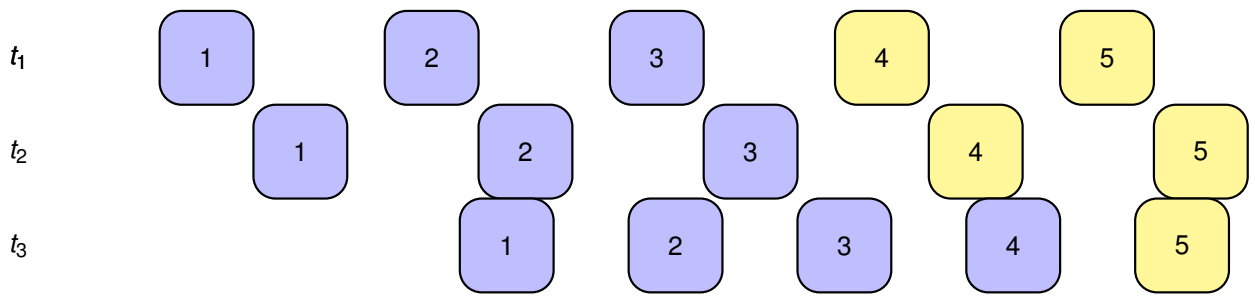
Two usual methods to evaluate the maximum throughput

1. **Earliest Schedule**: tasks are performed as soon as possible until a stabilization is reached. A K-periodic steady state is always reached after a temporary phase.
2. **Expansion**: An equivalent SDF $\mathcal{G}_{exp}$ with unit weight (*i.e* $Z_t = 1, \forall t \in \mathcal{T}$) may be built by expanding each actor t N_t times. The throughput may then be polynomially computed using classical critical circuits algorithms (Chrétienne 1982) or Max-Plus algebra (Cohen et al. 1987).

- *Advantage* $\rightarrow$ the evaluation is exact (as the earliest schedule maximizes the throughput of each actor);
- *Drawback* $\rightarrow$ not polynomial, complexity proportional at least to $\sum_{t \in \mathcal{T}} N_t$. Not possible to use this method in an optimization process, nor for SDF with a large number of actors.

Periodic schedule: the simplest way to executes tasks





A Periodic schedule of throughput $\tilde{\lambda} = \frac{5}{3}$. $\tilde{\lambda}(t_1) = \frac{5}{12}$, $\tilde{\lambda}(t_2) = \frac{5}{12}$ and $\tilde{\lambda}(t_3) = \frac{5}{9}$.

Characterization and computation of a Periodic Schedule with maximum throughput

Theorem (Benabid et al. [BHMMK12])

Let a normalized SDF G. For any feasible periodic schedule s, there exists $\lambda \in \mathbb{Q}^{\star +}$ *such that for any couple of actors* $(t_i, t_j) \in \mathcal{T}^2$, $\frac{Z_i}{w_i} = \frac{Z_j}{w_j} = \lambda$. *Moreover, s is feasible if, for any buffer* $a = (t_i, t_j) \in \mathcal{B}$,

$$s(t_j, 1) - s(t_i, 1) \geq \ell(t_i) - \frac{H(a)}{\lambda}.$$

with $gcd_a = \gcd(Z_i, Z_j)$ *and* $H(a) = M_0(a) + gcd_a - Z_j$.

Computation of a strictly periodic schedule of maximum throughput can be expressed by a linear program:

$$\begin{array}{l} max \quad \lambda \quad subject\ to \\ \begin{cases} \forall a = (t_i, t_j) \in \mathcal{B}, & s(t_j, 1) - \mathcal{S}(t_i, 1) \geq \ell(t_i) - \frac{H(a)}{\lambda} \\ \forall t_i \in T, & s(t_i, 1) \geq 0 \end{cases} \end{array}$$

The maximum value $\tilde{\lambda}$ is a lower bound of the maximum throughput of *G*.

Benchmarks

Table: Evaluation of the normalized period $\frac{1}{\lambda}$ using an optimal periodic scheduling vs. an optimal algorithm earliest schedule.

Application	Periodic Sched.		Optimal Sched.	
BlackScholes	210	13ms	210	42ms
Echo	40816	10ms	40816	65ms
JPEG2000	66225	69ms	N/A	6sec
Pdetect	1953	80ms	294	216ms
H264 Encoder	3906	544ms	-	>60h

Conclusions and perspectives

- SDF is a simple model for repetitive tasks communicating through buffers;
- Constraints associated to arcs can be viewed as a cyclic scheduling problem with some particular precedence constraints;
- Dominance properties and mathematical results can be considered to avoid simulation to evaluate the SDF (Schedulability, maximum throughput);

Challenging questions:

- The complexity of the schedulability and the determination of the maximum throughput are still of unknown complexity. There exists fixed parameter tractable algorithms pamaterized by $p = \sum_{t_i \in \mathcal{T}} N_i$;
- New strategies using partial expansions on critical circuits were tested by several authors;
- Resource limitations (processors for tasks, memory for buffers) must be considered to solve real-life applications.

References I


Abir Benabid, Claire Hanen, Olivier Marchetti, and Alix Munier-Kordon, *Periodic Schedules for Bounded Timed Weighted Event Graphs*, IEEE Transactions on Automatic Control **57** (2012), no. 5, 1222 – 1232.


Christoph M. Kirsch and Ana Sokolova, *The logical execution time paradigm*, Advances in Real-Time Systems (Samarjit Chakraborty and Jörg Eberspächer, eds.), Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 103–120.


Alix Munier Kordon and Ning Tang, *Evaluation of the age latency of a real-time communicating system using the LET paradigm*, 32nd Euromicro Conference on Real-Time Systems, ECRTS 2020, July 7-10, 2020, Virtual Conference, 2020, pp. 20:1–20:20.


Edward A. Lee and David G. Messerschmitt, *Synchronous data flow*, Proceeding of the IEEE **vol. 75**, no. no. 9.


Olivier Marchetti and Alix Munier Kordon, *Cyclic scheduling for the synthesis of embedded systems*, Introduction to scheduling, ch. 6, Chapman and Hall/CRC Press, November 2009, ISBN: 978-1420072730.


Alix Munier, *Régime asymptotique optimal d'un graphe d'événements temporisé généralisé: application à un problème d'assemblage*, RAIRO-Automatique Productique Informatique Industrielle **27** (1993), no. 5, 487–513.