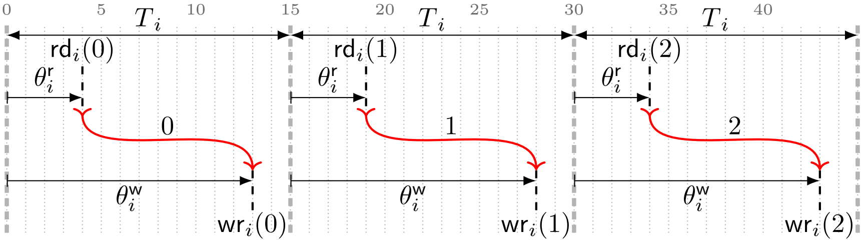
# Zero-Jitter Task Chains via Algebraic Rings

**Enrico Bini**, Paolo Pazzaglia, Martina Maggio

IEEE Transactions on Computers, 2023

WaP @RTSS 2023, December 5th 2023
Taipei, Taiwan

# Model of a LET task $\tau_i$



- A LET *task* $\tau_i$ is composed by *periodic jobs* (curvy arrows)
  - in LET, we only care of the read and write instants

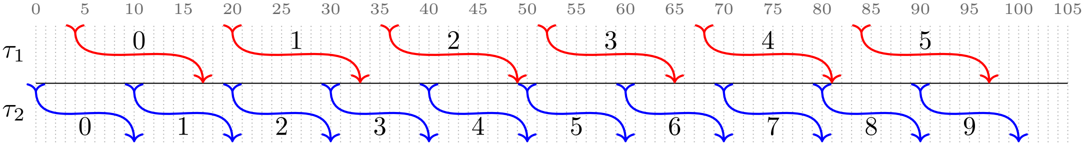
$\theta_i^{\mathsf{r}}$ *read phasing* of $\tau_i$, relative to the *period* $T_i$ (aka the *offset*)

$\theta_i^{\mathsf{w}}$ *write phasing* of $\tau_i$, rel. to *period* $T_i$ ($\theta_i^{\mathsf{w}} - \theta_i^{\mathsf{r}}$ is aka the *deadline*)

$\mathsf{rd}_i(j) = j\,T_i + \theta_i^{\mathsf{r}}$ *read instant* of job $j$ of $\tau_i$

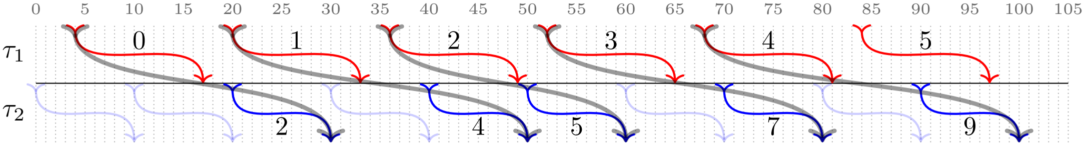
$\mathsf{wr}_i(j) = j\,T_i + \theta_i^{\mathsf{w}}$ *write instant* of job $j$ of $\tau_i$

# Chain of 2 LET tasks



- Chain of LET tasks
  - each task reads data written by the previous one
  - data is on **shared memory**
    - ★ $\tau_1$ may over-write data before $\tau_2$ reads (if $T_1 < T_2$)
    - ★ $\tau_2$ may read again the same data (if $T_2 < T_1$)

# Chain of 2 LET tasks

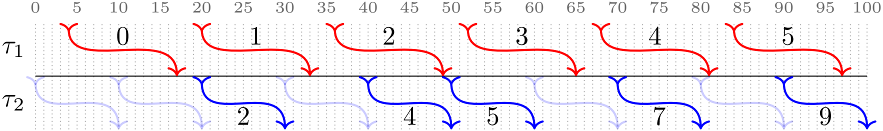


- Chain of LET tasks
  - each task reads data written by the previous one
  - data is on **shared memory**
    - ★ $\tau_1$ may over-write data before $\tau_2$ reads (if $T_1 < T_2$)
    - ★ $\tau_2$ may read again the same data (if $T_2 < T_1$)
- Chains of jobs have **variable** read-to-write delay

| $(j_1, j_2)$ | $\mathsf{rd}_1(j_1)$ | $\mathsf{wr}_2(j_2)$ | $\mathsf{wr}_2(j_2) - \mathsf{rd}_1(j_1)$ |
|---|---|---|---|
| $(0, 2)$ | $4$ | $30$ | $26 = 13 + \mathbf{3} + 10$ |
| $(1, 4)$ | $20$ | $50$ | $30 = 13 + \mathbf{7} + 10$ |
| $(2, 5)$ | $36$ | $60$ | $24 = 13 + \mathbf{1} + 10$ |
| $(3, 7)$ | $52$ | $80$ | $28 = 13 + \mathbf{5} + 10$ |
| $(4, 9)$ | $68$ | $100$ | $32 = 13 + \mathbf{9} + 10$ |
| $\dots$ | $\dots$ | $\dots$ | $\dots$ |

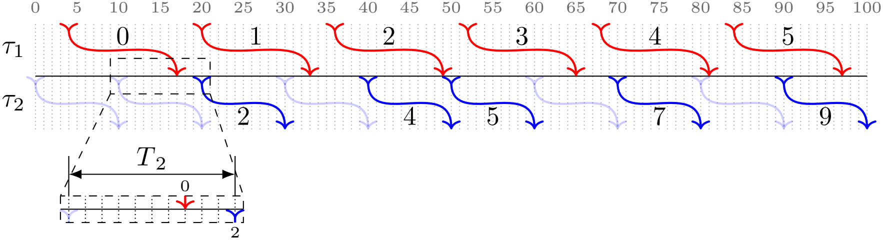
# Variability of the input-output delay

- The source of variability is $\mathsf{rd}_2(j_2) - \mathsf{wr}_1(j_1)$. In the example: $3, 7, 1, 5, 9, 3, 7, \ldots$



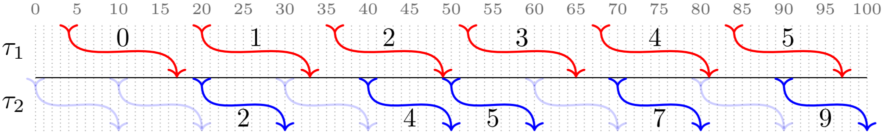
# Variability of the input-output delay

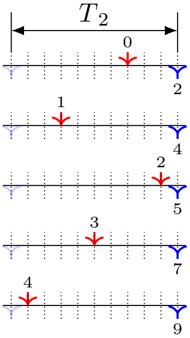
- $\mathsf{rd}_2(j_2) - \mathsf{wr}_1(j_1) < T_2$, always. Let's zoom in ...



# Variability of the input-output delay

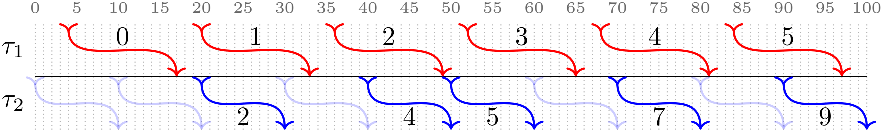
- …and align the read instants of consecutive $\tau_2$ jobs.

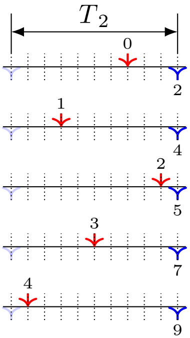


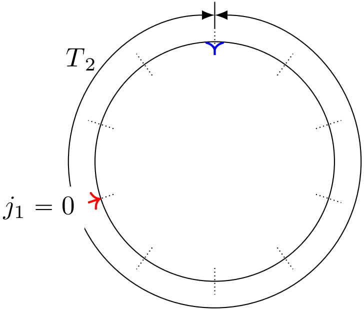


# Variability of the input-output delay

- A repetitive requence... mmhhh... let's move over the (algebraic) ring $\mathbb{Z}/T_2\mathbb{Z}$.

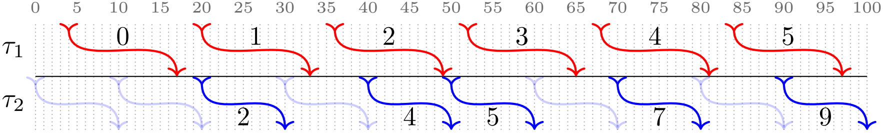


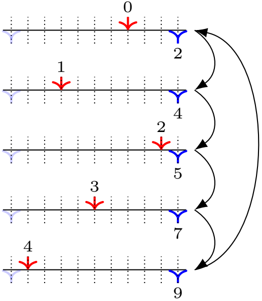


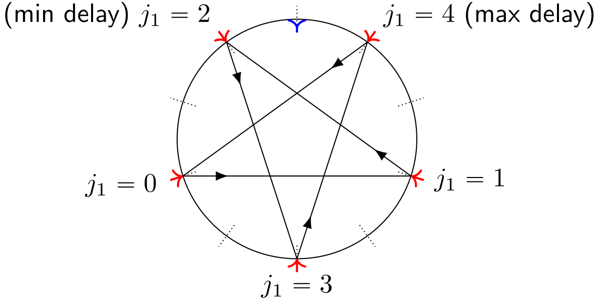


# Variability of the input-output delay

- By inverting over the ring $\mathbb{Z}/T_2\mathbb{Z}$, max/min delay is found **without** unrolling the schedule.

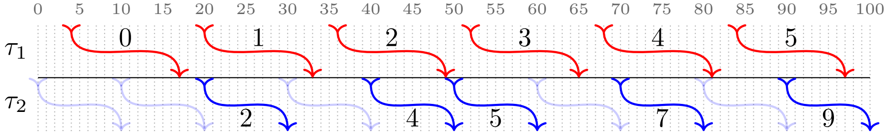


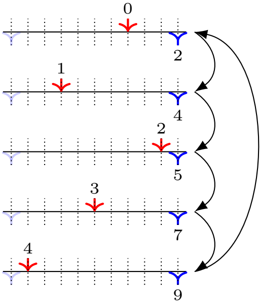


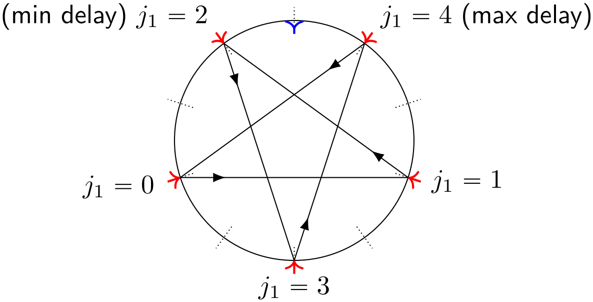


# Variability of the input-output delay

- We can **eliminate the jitter** of a 2-tasks chain, by adding a copier task.


0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
$\tau_1$
$\tau_2$
0
1
2
3
4
5
2
4
5
7
9


0
2
1
4
2
5
3
7
4
9


(min delay) $j_1 = 2$
$j_1 = 4$ (max delay)
$j_1 = 0$
$j_1 = 1$
$j_1 = 3$