Open Problems in FIFO Scheduling with Multiple Offsets

Mitra Nasri  Robert I. Davis  Björn B. Brandenburg

“Running with offset”
First-In-First-Out (FIFO) scheduling

- **Extremely simple**
- **Very low overheads**
- **Easy to analyze**

**Ideal for**
- IoT-class devices
- deeply embedded systems
- hardware implementations

**Not good for**
- hard real-time systems

**This talk**

Reviewing our recent work [Nasri et al., RTAS’2018] on **Improving FIFO’s schedulability** by assigning **multiple offsets** to each task

**Open problems in**

**multiple-offset assignment**
Intuition

What is the problem with FIFO scheduling

A secret to boost schedulability (for non-preemptive periodic tasks)

How to improve FIFO’s schedulability

From [Nasri et al. RTAS’2018]
What is the problem with the “plain” FIFO?

**FIFO schedule of 3 periodic tasks**

Plain FIFO is **oblivious** to deadlines and priorities

τ₃ comes first → deadline miss for τ₂

**WCET**: worst-case execution time
Work-conserving scheduling

NP-RM and NP-EDF schedule of 3 periodic tasks

In fact, any work-conserving policy (EDF, RM, ...) must schedule $\tau_3$ here $\Rightarrow$ deadline miss for $\tau_2$
Non-work-conserving scheduling

CW-EDF [Nasri et al. ECRTS’2016] schedule of the same 3 periodic tasks

CW-EDF considers future job arrivals in a “critical window” and postpones $\tau_3$ until later.

Non-work-conserving scheduling

• Periodic tasks that pass necessary schedulability tests, constructed in a similar way as Automotive benchmark tasks [Kramer’15]
• About 30 tasks in a task set.
• Deadline is equal to period.
Periodic tasks that pass necessary schedulability tests, constructed in a similar way as Automotive benchmark tasks [Kramer’15]
About 30 tasks in a task set.
Deadline is equal to period.
Periodic tasks that pass necessary schedulability tests, constructed in a similar way as Automotive benchmark tasks [Kramer’15]
About 30 tasks in a task set.
Deadline is equal to period.
Non-work-conserving scheduling

- Periodic tasks that pass necessary schedulability tests, constructed in a similar way as Automotive benchmark tasks [Kramer’15]
- About 30 tasks in a task set.
- Deadline is equal to period.
Non-work-conserving scheduling

CW-EDF looks like a Promising solution
Current implementations of CW-EDF have a considerable runtime overhead! Example: on ATMega2560 @ 16 MHz, the overhead is 9.2x more than RM.

How can we get high schedulability and low overheads?
The secret behind CW-EDF’s success

CW-EDF is able to leave the processor idle at the “right” moment

This is not possible in a work-conserving policy unless the workload is shaped

Offset assignment is one way to shape (here only to “shift”) the workload

τ₃ causes a deadline miss if it is released before time 12

To avoid that, we use an offset!

τ₁

τ₂

τ₃

WCET = 8, period 60
WCET = 6, period 12
WCET = 3, period 10
The secret behind CW-EDF’s success

CW-EDF is able to leave the processor idle at the “right” moment

This is not possible in a work-conserving policy unless the workload is shaped

Offset assignment is one way to shape (here only to “shift”) the workload

τ₃ causes a deadline miss if it is released before time 12

To avoid that, we use an offset!

WCET = 8, period 60

WCET = 6, period 12

WCET = 3, period 10
The state of the art
Single offset assignment for FIFO scheduling

1. Altmeyer, Sundharam, & Navet, 2016:
   Try many randomly assigned offsets
   This approach does not scale with the number of tasks and an increase in utilization

2. Nasri, Davis, & Brandenburg, RTAS’2018:
   Offset of a task is the start time of the first job of that task in a CW-EDF schedule (called FST approach)
Single offset assignment for FIFO scheduling

- Non-preemptive fixed-priority
- FIFO + no offset
- CW-EDF

utilization

schedulability ratio

NP-FP + no offset  FIFO + no offset  CW-EDF
Single offset assignment for FIFO scheduling

- Non-preemptive fixed-priority
- Already a significant achievement
- Room for improvement
- CW-EDF
- FIFO with single offset per task

 schedulability ratio

 utilisation

- NP-FP + no offset
- FIFO + no offset
- FIFO + FST
- CW-EDF
Can we get even better results? Yes!

Our recent work showed that by assigning **multiple offsets** to a task, **FIFO becomes as good as CW-EDF**!

---

[Nasri, Davis, & Brandenburg, RTAS’2018]
Offset tuning technique [Nasri et al. RTAS’2018]

Intuition

Infer offsets from a given feasible reference schedule while greedily reducing the number of offset partitions!

We used CW-EDF schedule as a reference (since it has a very good schedulability ratio)

However, offset tuning is capable to force FIFO to rebuild “any” desired schedule at runtime

- ILP/SAT solving
- bespoke planning heuristics
- ...
How to reduce the number of offsets? [Nasri et al. RTAS’2018]

Can we use only one offset for both jobs?
How to reduce the number of offsets? [Nasri et al. RTAS’2018]

because FIFO schedules jobs with their **release order**

Any offset assignment within interval (12, 18] creates the **same job ordering**
(\(\tau_3\) will be scheduled after the second job of \(\tau_2\))

Any offset assignment from the **intersection** of (12, 18] and (10, 14]
creates the desired **job ordering** for both jobs \(J_{3,1}\) and \(J_{3,2}\)
How to reduce the number of offsets? [Nasri et al. RTAS’2018]

1. We defined **schedule equivalency**
2. We defined **potential offset intervals (POI)**
3. We introduced a **greedy heuristic** to find largest **job partitions** that can use the **same relative offset**

Jobs of the task:

- POI\(_i,1\)
- POI\(_i,2\)
- POI\(_i,3\)
- POI\(_i,4\)
- POI\(_i,5\)
- POI\(_i,6\)

Job partition 1: \(\{J_i,1, J_i,2\}\)
Job partition 2: \(\{J_i,3, J_i,4, J_i,5\}\)
Job partition 3: \(\{J_i,6\}\)
Some results

86% of the 9000 generated task sets needed only 4 offsets per task.

In average 122 Bytes are required to store all offsets.

Same experimental setup: About 30 tasks in a task set. Deadline is equal to period. Periodic tasks that pass necessary schedulability tests, constructed in a similar way as Automotive benchmark tasks [Kramer’15].
Open problems
Open Problem 1: How to find offsets?

Open Problem 2: How to minimize the number of offsets?

Open Problem 3: How to deal with release jitters?
Given a set of \( n \) periodic tasks
(characterized \( C_i, T_i, D_i, O_i \), where \( O_i \) is the initial offset),

**Find** a set of offset pairs \( \hat{\Omega} = \{ (k_{i,1}, o_{i,1}), (k_{i,2}, o_{i,2}), \ldots, (k_{i,m_i}, o_{i,m_i}) \} \)

**such that** the resulting task set is FIFO schedulable.

We assume that job's relative deadline is not affected by relative offsets!
Open Problem 1

Given a set of \( n \) periodic tasks
(characterized \( C_i, T_i, D_i, O_i \), where \( O_i \) is the initial offset),

Find a set of offset pairs \( \hat{\Theta} = \{(k_{i,1}, o_{i,1}), (k_{i,2}, o_{i,2}), ..., (k_{i,m_i}, o_{i,m_i})\} \)
such that the resulting task set is **FIFO schedulable**.

**Visualization of offset pairs:**

\[
(k_{i,1}, o_{i,1}) = (1, 3ms) \quad \quad (k_{i,2}, o_{i,2}) = (5, 7ms) \quad \quad (k_{i,3}, o_{i,3}) = (9, 1ms)
\]
Challenges of Open Problem 1

As an **extreme case**, assume we assign an **offset** to each **job** of a task

Now, the problem is reduced to finding a non-preemptive schedule for a set of periodic tasks

Open problem 1 is strongly NP-Hard

Since non-preemptive scheduling of periodic tasks is a strongly NP-Hard problem [Jeffay 1991]

In our recent work [Nasri et al. RTAS’2018] we find solution **only if** the task set is **CW-EDF schedulable**.
Challenges of Open Problem 1

As an extreme case, assume we assign an **offset** to each **job** of a task.

Now, the problem is reduced to **finding a non-preemptive schedule for a set of periodic tasks**.

**Open problem 1 is strongly NP-Hard**
Since non-preemptive scheduling of periodic tasks is a strongly NP-Hard problem [Jeffay 1991]

The space of possible offsets is **large and unstructured**.

- **Iterative approaches cannot be easily applied**
- **Storing too many offsets per task might not be feasible**
  - When the system has a **limited memory**
Open Problem 2

**Given** a set of \( n \) periodic tasks

(characterized \( C_i, T_i, D_i, O_i \), where \( O_i \) is the initial offset),

**Find** a set of offset pairs \( \hat{O} = \{(k_{i,1}, o_{i,1}), (k_{i,2}, o_{i,2}), \ldots, (k_{i,m_i}, o_{i,m_i})\} \)

such that the resulting task set is **FIFO schedulable**

and the **total number of offset pairs** is **minimized**, i.e.,

\[
\text{Min } \sum_{i=1}^{n} |\hat{O}_i|
\]

**In our prior work** [Nasri, Davis, Brandenburg RTAS’2018],

we solve Open Problem 1

while trying to **reducing** the number of offset pairs
Other practical aspects

In practice, systems usually have release jitter
due to interrupt handling routine, buffers, networking delays, etc.

FIFO scheduling is **NOT sustainable**

w.r.t. release jitter

An offset assignment is needed that guarantees
schedulability in the presence of release jitter
Open Problem 3

**Given** a set of $n$ periodic tasks
(characterized $C_i, T_i, D_i, O_i, J_i$, where $J_i$ is the **release jitter**),

**Find** a set of offset pairs $\hat{O} = \{(k_{i,1}, o_{i,1}), (k_{i,2}, o_{i,2}), ..., (k_{i,1}, o_{i,m_i})\}$
**such that** the resulting task set is **FIFO schedulable**.

**Challenge**
there is no FIFO schedulability analysis that considers release jitters
Our recent work [Nasri et al. RTAS’2018] showed that FIFO schedulability can be **significantly improved** with the help of **offsets**

**Open problems**

- How to **find** offsets that make FIFO schedulable?
- How to **assign** offsets in the presence of **release jitters**?
- How to **minimize** The total number of offsets?
Plain FIFO (no offset)

FIFO with one offset per task

FIFO with multiple offsets

NP-FP + no offset

FIFO + no offset

FIFO + FST

FIFO + offset tuning

CW-EDF

Thank you