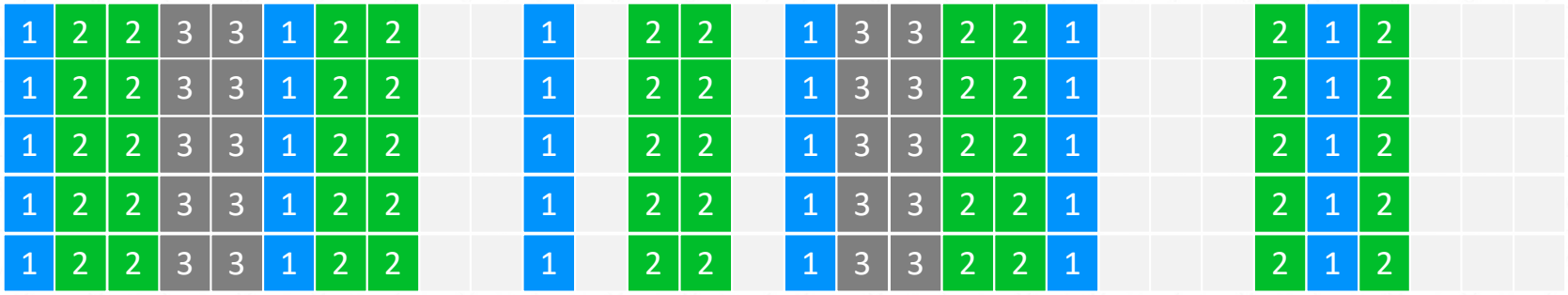
SCHEDULE OBFUSCATION

[RTAS 2016]

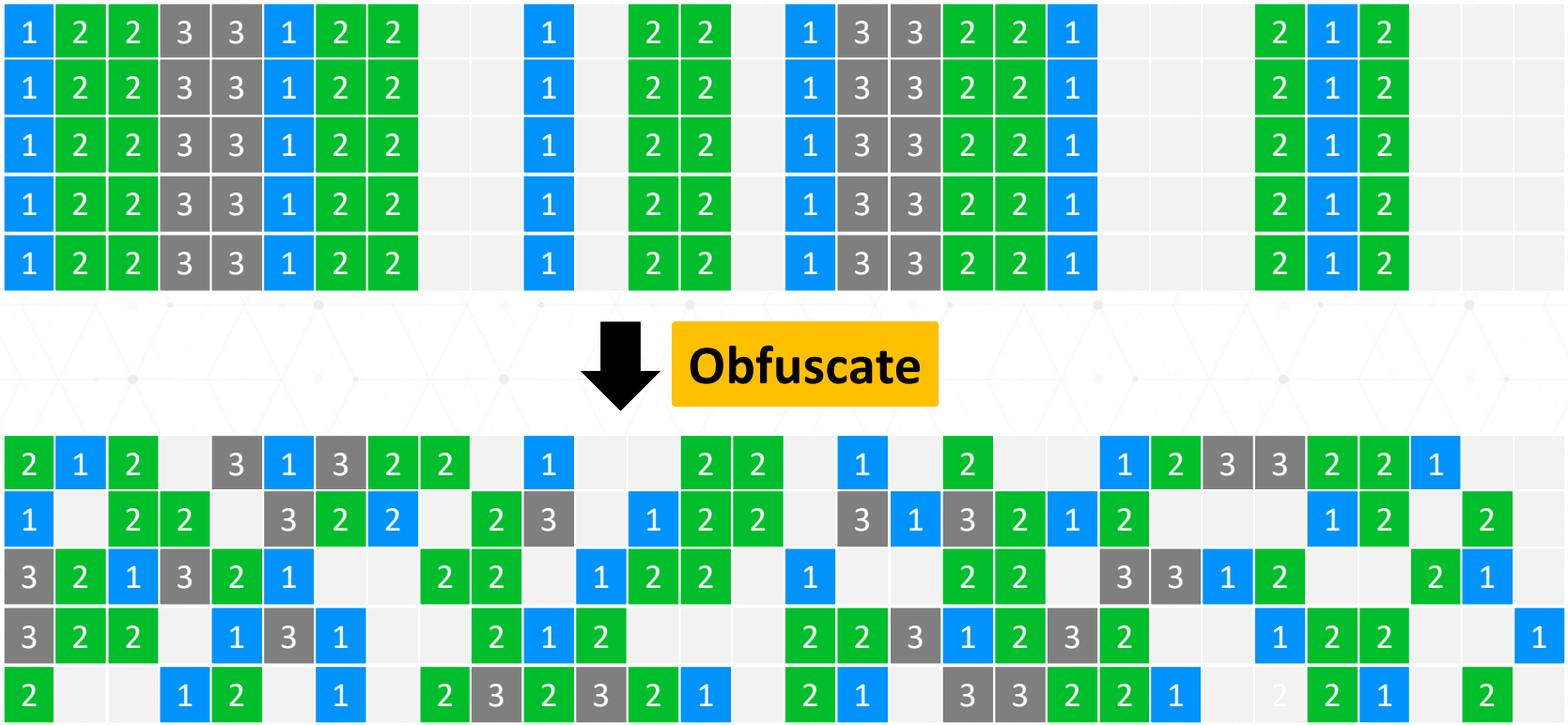
- Integrating security in Real-Time CPS
- Prevent attacks by randomizing schedule

"is it possible to reduce the regularity in real-time task schedules while still guaranteeing that the timing constraints (deadlines) are met?"

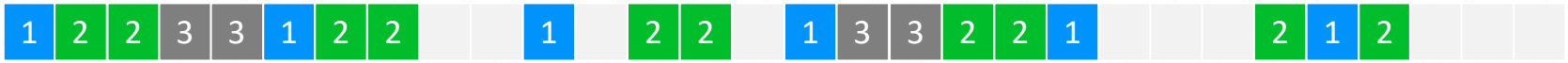
REAL-TIME SCHEDULE OBFUSCATION



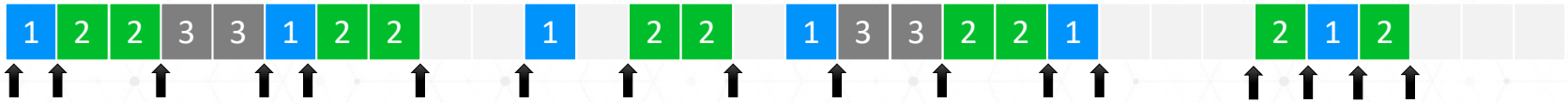
REAL-TIME SCHEDULE OBFUSCATION



SCHEDULE OBFUSCATION: CONCEPT

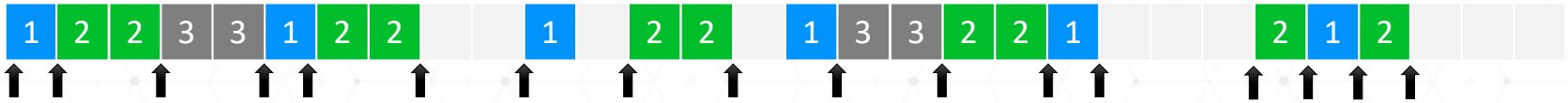


SCHEDULE OBFUSCATION: CONCEPT



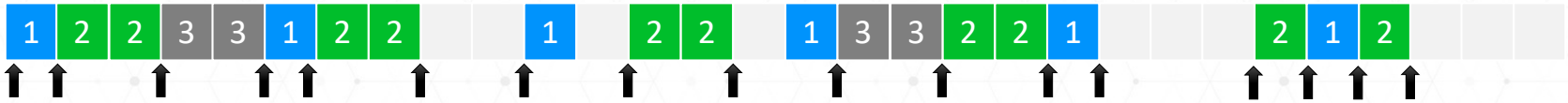
- At each scheduling point

SCHEDULE OBFUSCATION: CONCEPT



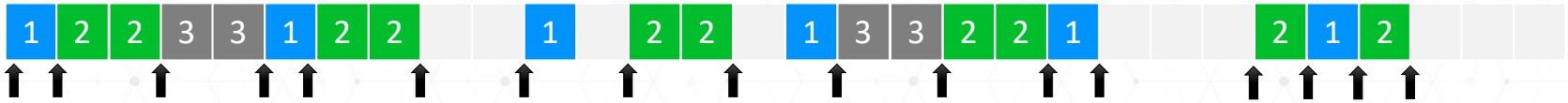
- At each scheduling point
  - Pick a random task from the ready queue
  - Not always the highest priority one

SCHEDULE OBFUSCATION: CONCEPT



- At each scheduling point
  - Pick a **random task** from the ready queue
  - Not always the highest priority one
- **Allow priority inversion**

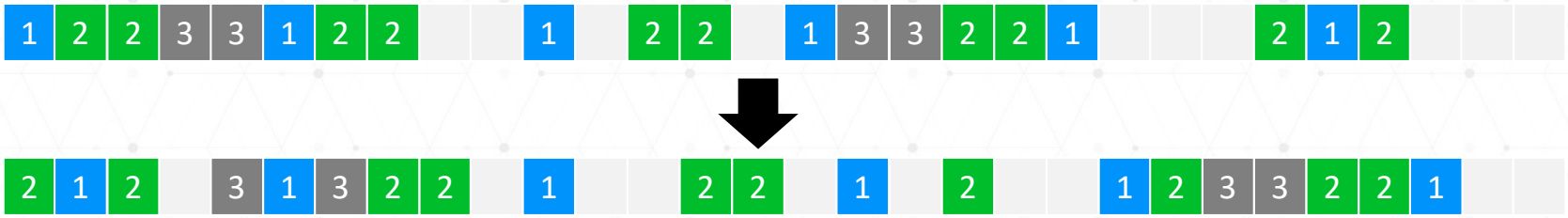
SCHEDULE OBFUSCATION: CONCEPT



- At each scheduling point
  - Pick a **random task** from the ready queue
  - Not always the highest priority one
- **Allow priority inversion**

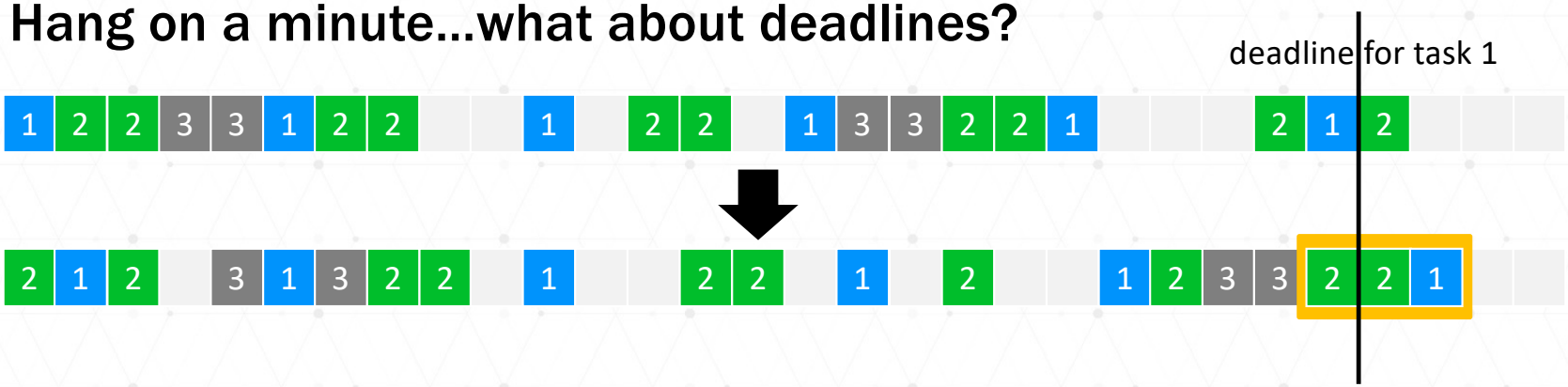
SCHEDULE OBFUSCATION: CONCEPT

- **Hang on a minute…what about deadlines?**



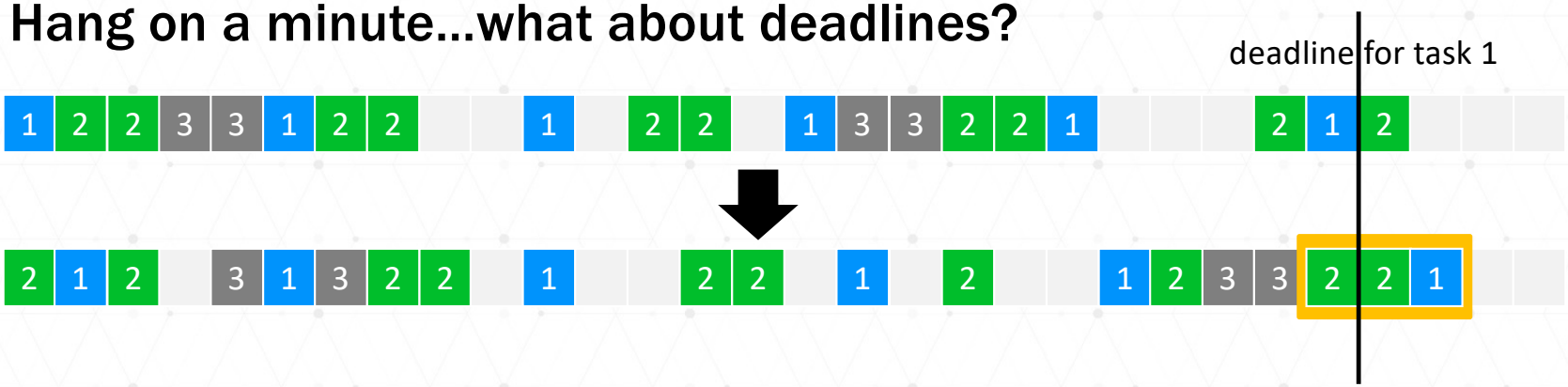
SCHEDULE OBFUSCATION: CONCEPT

- **Hang on a minute…what about deadlines?**



SCHEDULE OBFUSCATION: CONCEPT

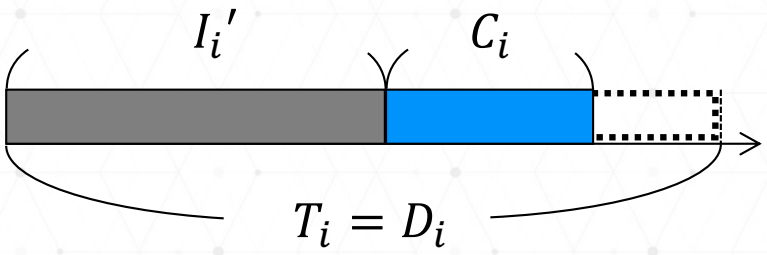
- Hang on a minute…what about deadlines?



- Allow **bounded** priority inversion
- Tasks should still meet their original deadlines
- We must calculate ‘bounds’
  - how long can a higher priority task suffer inversion?

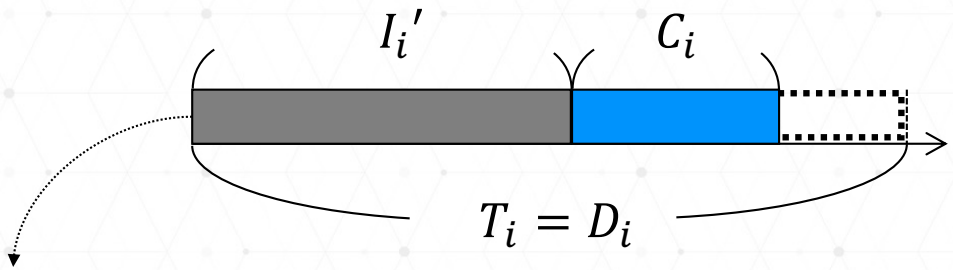
BOUNDING PRIORITY INVERSIONS

- Let's consider a periodic task τ_i



BOUNDING PRIORITY INVERSIONS

- **Let's consider a periodic task τ_i**



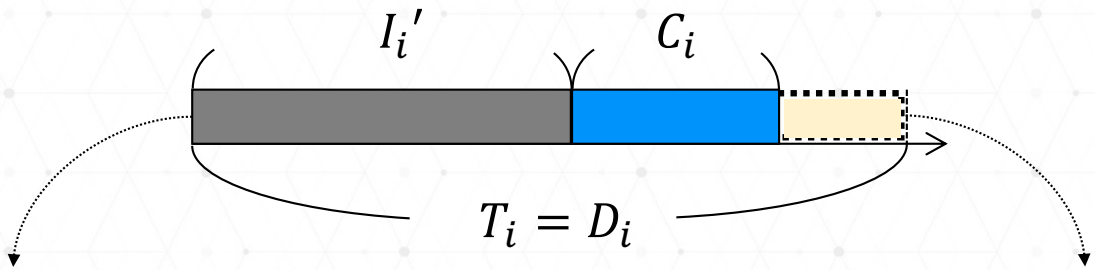
Interference induced by:
- higher priority tasks and
- priority inversion

needs to be taken into account

$$I_i' = \sum_{\tau_j \in hp(\tau_i)} \left(\left\lceil \frac{D_i}{T_j} \right\rceil + 1 \right) \cdot C_j$$

BOUNDING PRIORITY INVERSIONS

- **Let's consider a periodic task τ_i**



Interference induced by:
- higher priority tasks and
- priority inversion

needs to be taken into account

$$I_i' = \sum_{\tau_j \in hp(\tau_i)} \left(\left\lceil \frac{D_i}{T_j} \right\rceil + 1 \right) \cdot C_j$$

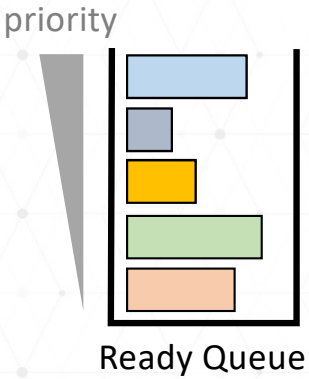
Extra delay that τ_i can tolerate without missing its deadlines

The worst-case inversion budget = V_i

TASKSHUFFLER RANDOMIZATION PROTOCOL

FIXED-PRIORITY SCHEDULING ALGORITHMS

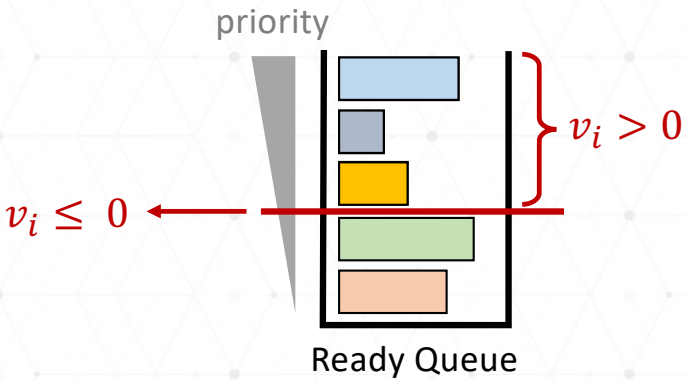
- **At each scheduling point**



TASKSHUFFLER RANDOMIZATION PROTOCOL

FIXED-PRIORITY SCHEDULING ALGORITHMS

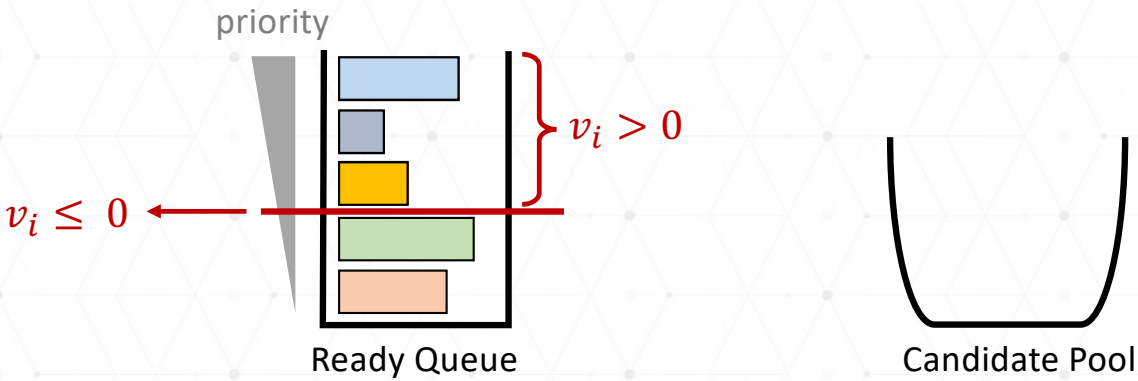
- **At each scheduling point**
  1. **Determine job candidates**



TASKSHUFFLER RANDOMIZATION PROTOCOL

FIXED-PRIORITY SCHEDULING ALGORITHMS

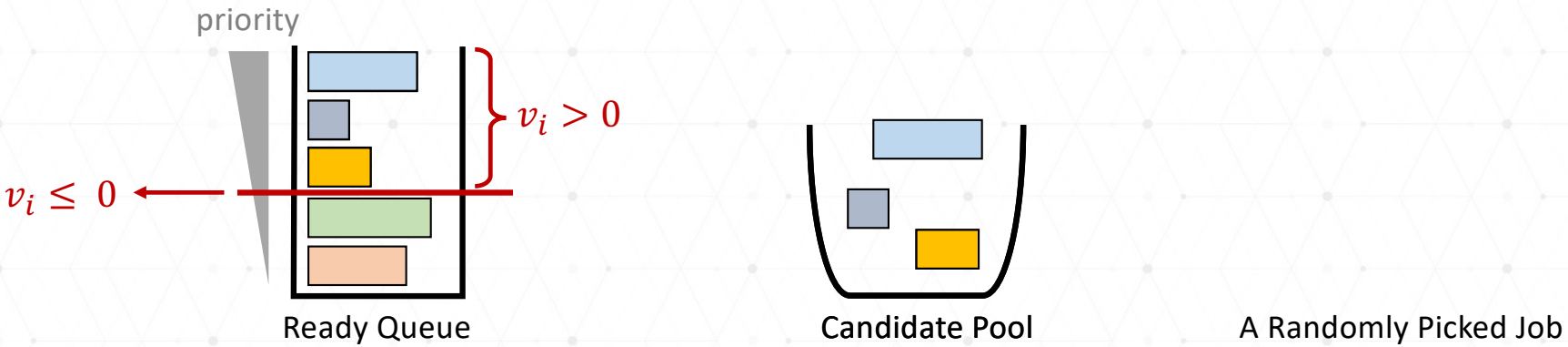
- **At each scheduling point**
  1. **Determine job candidates**



TASKSHUFFLER RANDOMIZATION PROTOCOL

FIXED-PRIORITY SCHEDULING ALGORITHMS

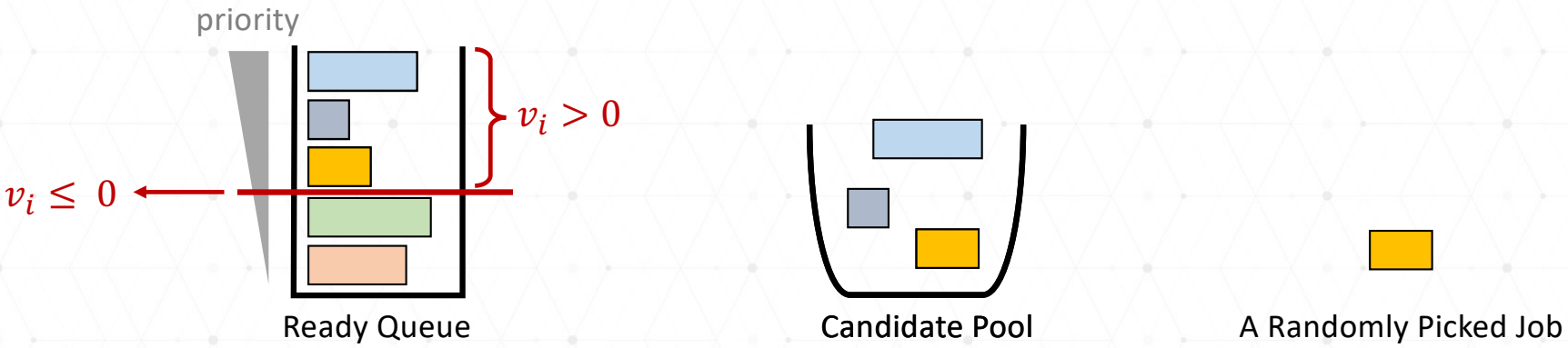
- **At each scheduling point**
  1. **Determine job candidates**
  2. **Randomly pick a job from candidate pool**



TASKSHUFFLER RANDOMIZATION PROTOCOL

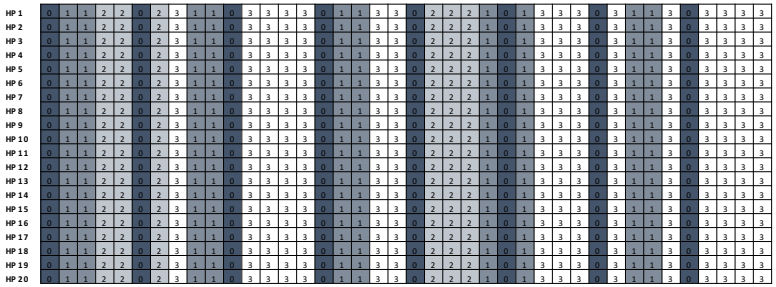
FIXED-PRIORITY SCHEDULING ALGORITHMS

- **At each scheduling point**
  1. **Determine job candidates**
  2. **Randomly pick a job from candidate pool**
  3. **Set next scheduling point and run picked job**



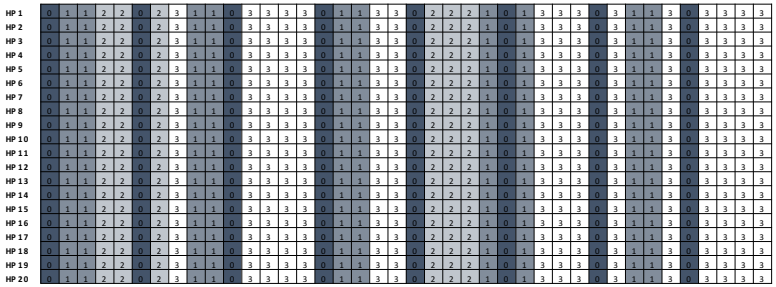
RANDOMIZATION SCHEMES

- **Without Randomization**

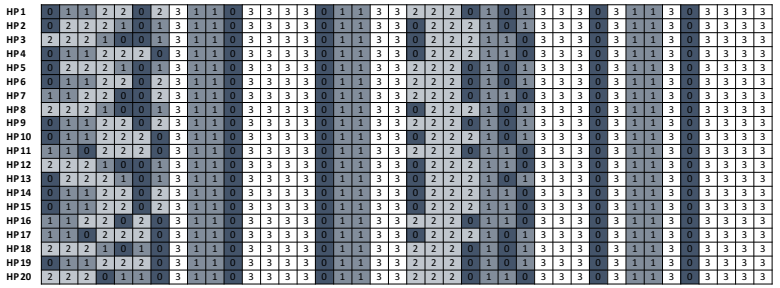


RANDOMIZATION SCHEMES

- **Without Randomization**

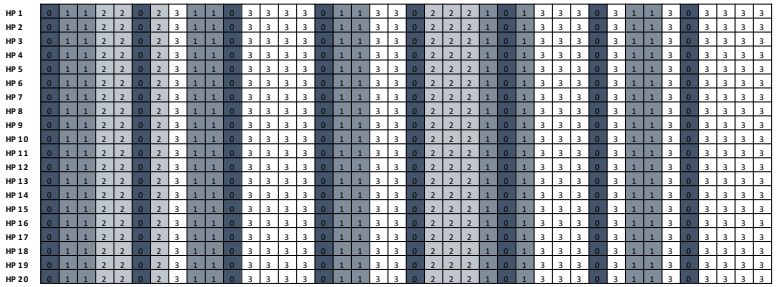


- **Task-only Randomization**

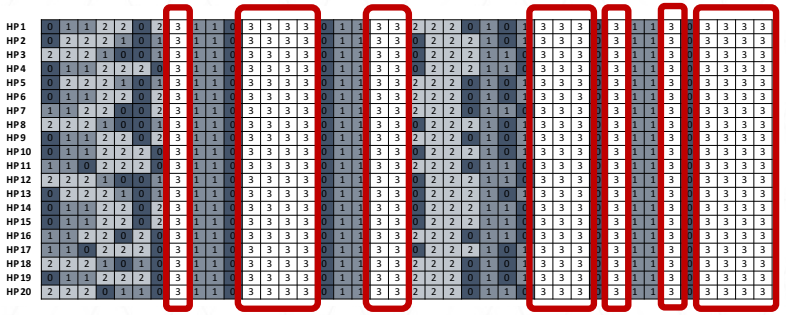


RANDOMIZATION SCHEMES

- **Without Randomization**

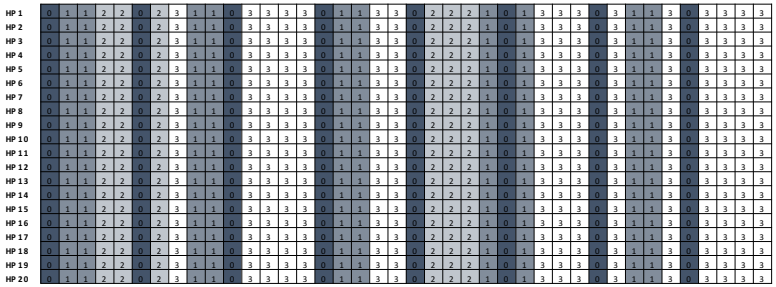


- **Task-only Randomization**

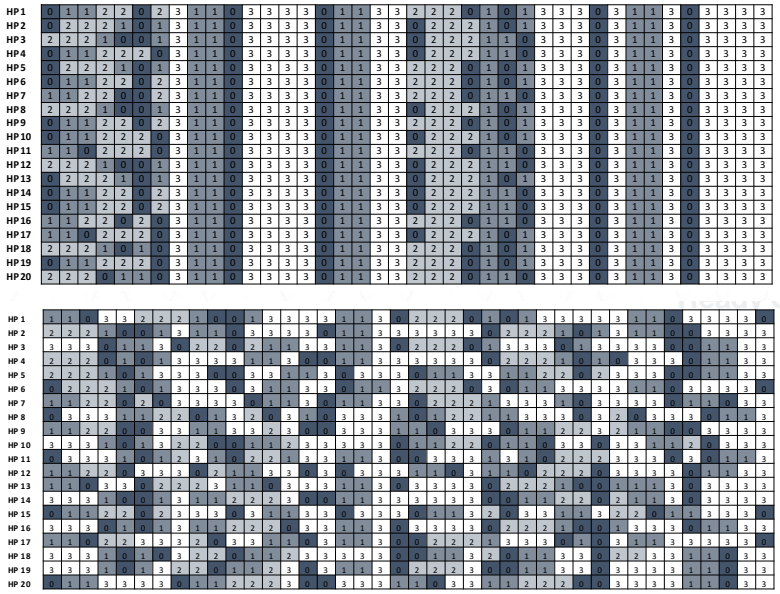


RANDOMIZATION SCHEMES

- Without Randomization

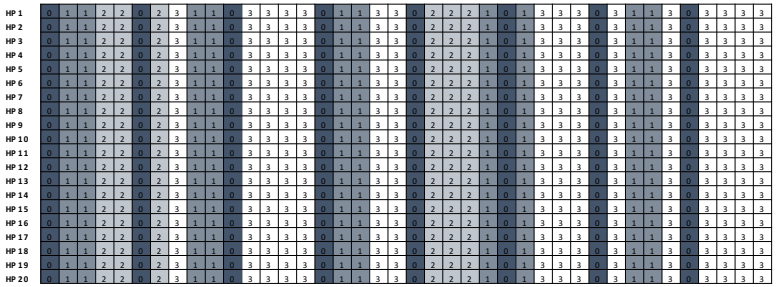


- Task-only Randomization
- With Idle Time Scheduling

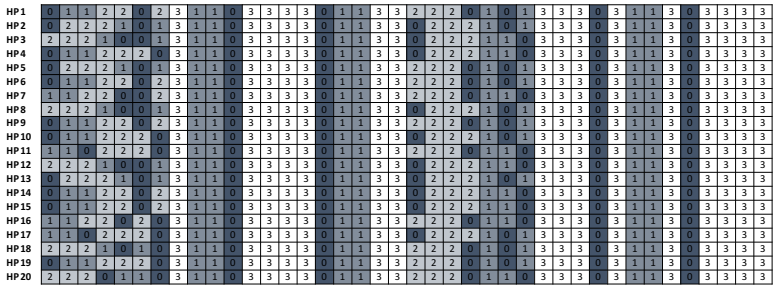


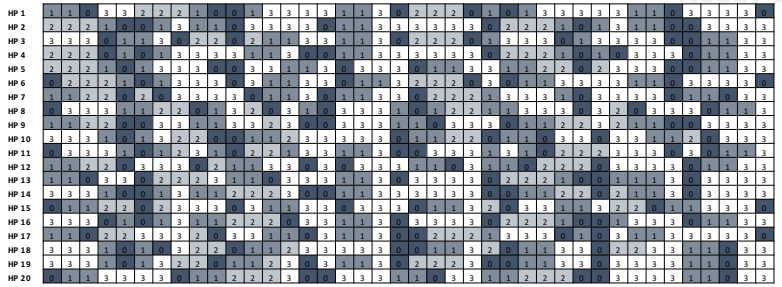
RANDOMIZATION SCHEMES

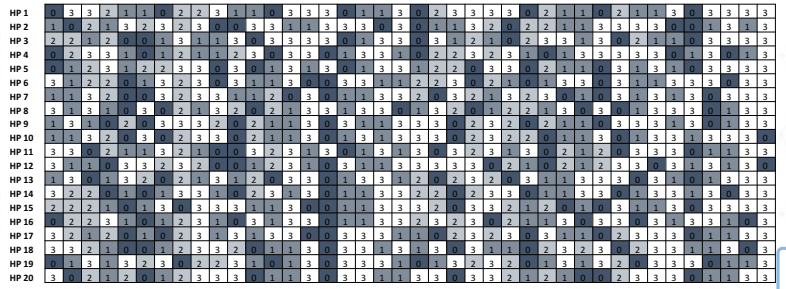
- Without Randomization



- Task-only Randomization
- With Idle Time Scheduling
- Fine-grained Switching

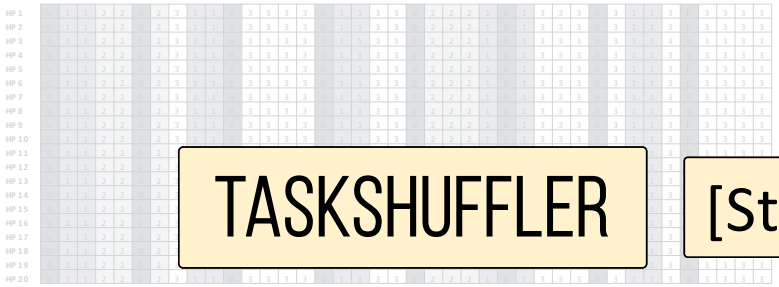






RANDOMIZATION SCHEMES

- Without Randomization



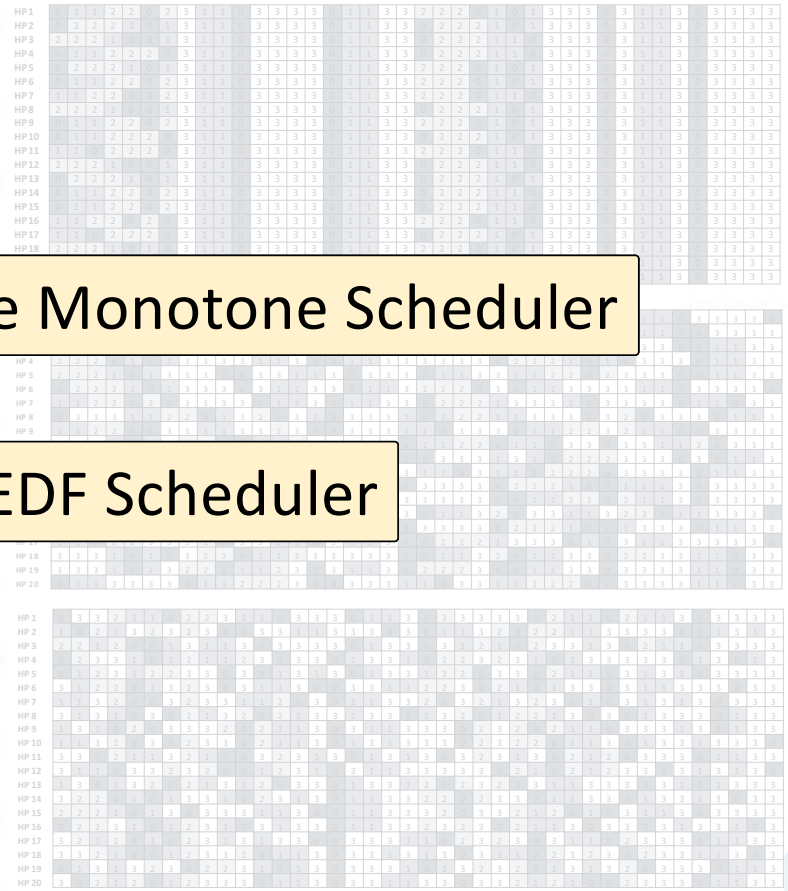
TASKSHUFFLER

[Static] Rate Monotone Scheduler

REORDER

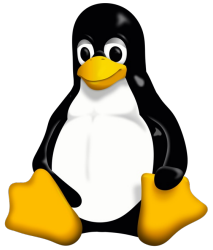
[Dynamic] EDF Scheduler

- Task-only R n
- With Idle Time Scheduling
- Fine-grained Switching



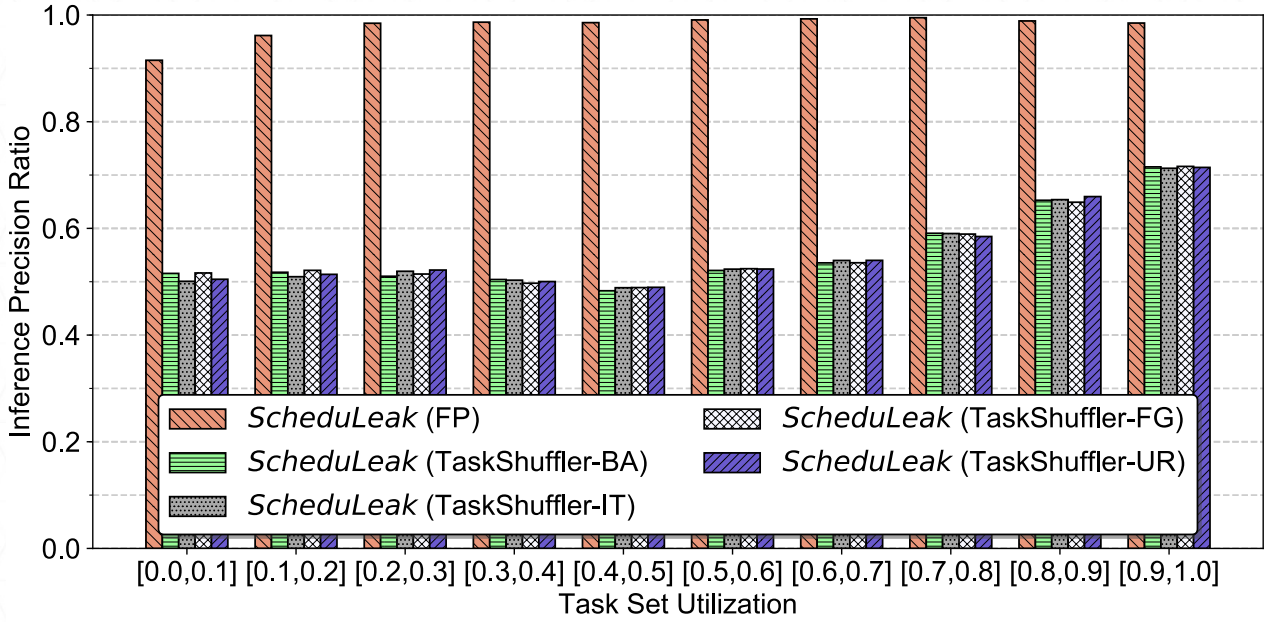
IMPLEMENTATION

- Platform
  - **Raspberry P**i 3 Model B
  - 1.2 GHz 64-bit quad-core ARM Cortex-A53
- Operating System
  - **Linux** kernel version: 4.9.48
  - Raspbian (a variant of Debian Linux)
  - Patched with **PREEMPT_RT**



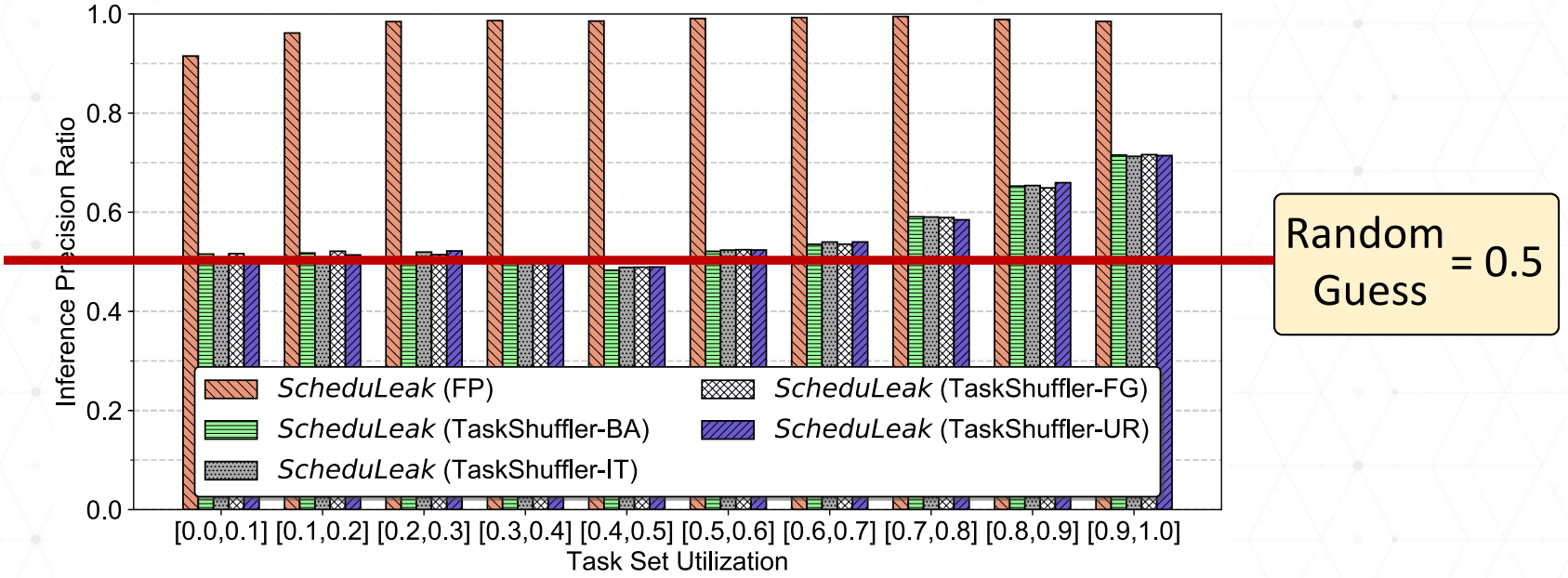
TASKSHUFFLER VS SCHEDULEAK

TASKSHUFFLER VS SCHEDULEAK



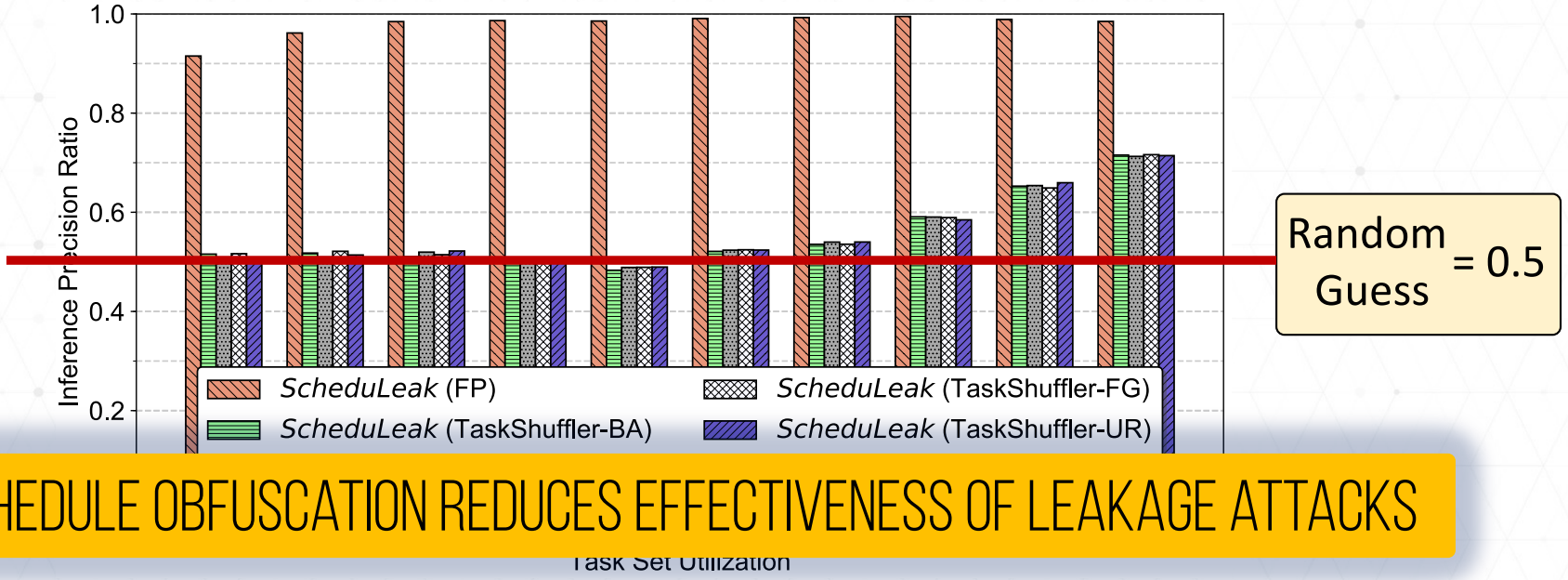
- Experiment Configurations
  - 6000 task sets tested
  - 600 each utilization group
  - 5, 7, 9, 11, 13, 15 tasks per task set
  - Each bar is averaged from 600 task sets

TASKSHUFFLER VS SCHEDULEAK



- Experiment Configurations
  - 6000 task sets tested
  - 600 each utilization group
  - 5, 7, 9, 11, 13, 15 tasks per task set
  - Each bar is averaged from 600 task sets

TASKSHUFFLER VS SCHEDULEAK



SCHEDULE OBFUSCATION REDUCES EFFECTIVENESS OF LEAKAGE ATTACKS

- Experiment Configurations
  - 6000 task sets tested
  - 600 each utilization group
  - 5, 7, 9, 11, 13, 15 tasks per task set
  - Each bar is averaged from 600 task sets

>