Real-Time Embedded Systems

Jasdeep SINGH
jasdeep.singh@ipsa.fr

Feb 2025 - April 2025

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Computer system

- Machine: Physical system to perform certain function
  - energy to operate
  - certain input
  - expected output
- System: a group of interacting/interrelated elements which follow rules to perform a certain dedicated task
- Computation: Mathematical operations: +, -, / ...
- Computer system is a machine which performs computation

Turing Machine

- The most fundamental definition of computer
- Tape: Divided into cells, each cell contains a symbol
- Head: Can read and write symbols
- State Register: stores the current state of machine
- Table: of instructions
  - Erase/Write symbol
  - Move head
  - Assume same or new state

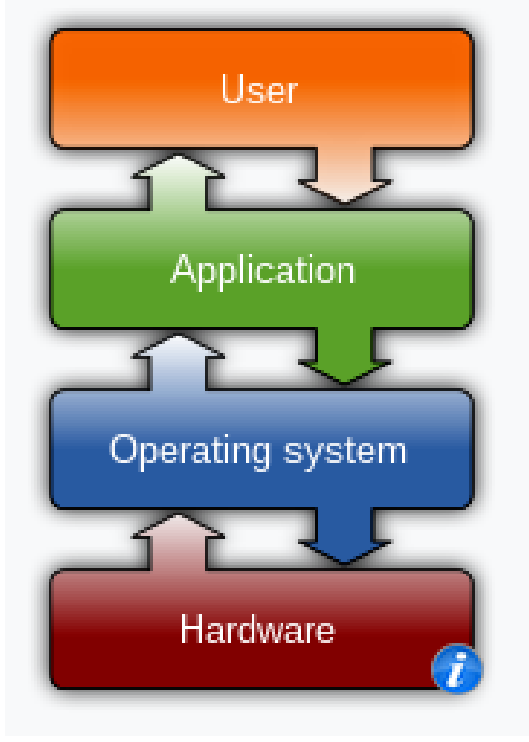
Modern Computer Systems

- Have become a lot more complex
- The usage has also become complex
- Multi-core, large memory areas, with small physical size
- Given way to System On Chips (SoC)
- Microcontrollers have become smaller
- It is generally difficult to impart the need to understand real-time aspects of embedded systems, especially if there is novelty

Operating System

- Software that manages computer hardware and software
- Provides services computer programs: game needs wifi, meeting eed camera, etc.

- Single Tasking OS: can only do one task at a time
- Muti-tasking: Tasks are scheduled
- OS is different from Basic Input Output Service (BIOS)



REAL TIME EMBEDDED SYSTEMS

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Definition

- Embedded System: A computer system which is designed and deployed for a dedicated functionality.
- Real-Time Embedded System: an embedded system which needs to function within certain real world timing constraints or deadlines.
- Real world deadline:
- Clock: measures passage of time
- Time: 1 sec = caesium 133 atom 9192631770 radiation waves
- Eg: Coffee machine Dedicated task: to make coffee, should make coffee in 1 min, not 2 days ! (in human terms)

Hard, Soft Real-Time Systems

- Hard: RT system which if does not produce desired functionality before the deadline at any point in its life, leads to catastrophe.
- Eg.: Aircraft Control system, actuation should happen when there is pilot input
- Soft: R/T system which if does not produce desired functionality before the deadline, does not lead to catastrophe but a quality degradation, data after deadline can have some use
- Eg.: Video streaming, losing some frames leads to bad quality
- Firm: Results after a deadline does not harm but it is of no use to system

Murphy's Law

- "Anything that can go wrong will go wrong"
- Mrs Murphy's Law: "Anything that can go wrong will go wrong, while Mr Murphy is out of town"
- Apply to RT system: If your embedded RT system's functioning can go wrong, it will !!!
- QUESTION: HOW TO MAKE SURE IT DOES NOT ?

A man drowned crossing a stream with an average depth of 6 inches

Real-Time System Model

- Workload are modelled as task τ_i , can be a program code or a function
- Tasks executes on resource(s), processor(s)
- Task have an execution time c_i
- $c_i = f$ (hardware, logic, code, compiler...)
- Tasks have a Worst Case Execution Time (WCET) C_i
- Tasks have a timed deadline D_i
- Tasks can have priorities p_i
- Tasks can have period of execution T_i, at which they execute again
- A real Time System S is tuple

$$S = \{\tau_i, C_i, D_i, T_i, p_i\}, i = 1, 2, 3...n$$

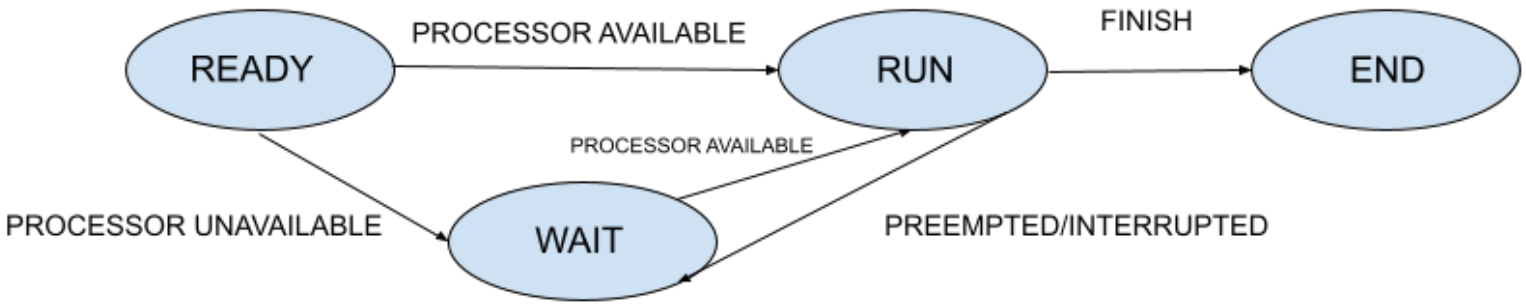
IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Task Execution Model

A task is in one of the following states

- A task is released and is *READY* for execution
- Task executes in *RUN*
- Task can *WAIT* for execution, it can happen immediately after release or it can be interrupted while execution
- Task *END* execution

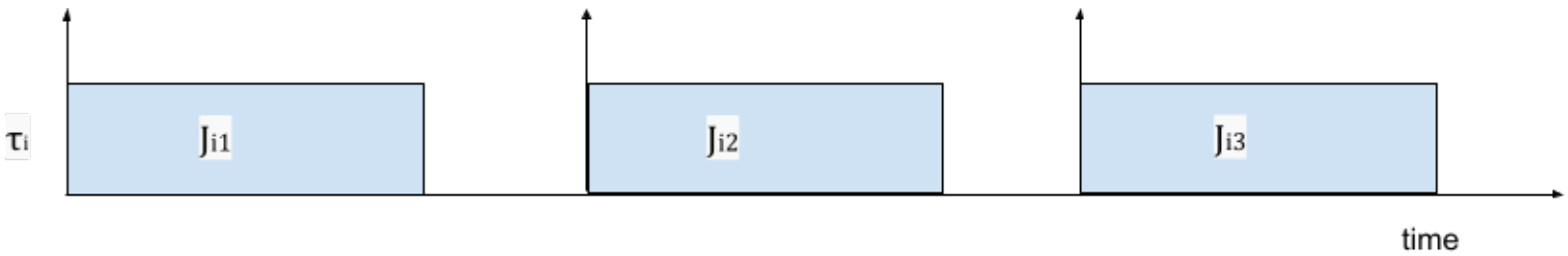
The relationship between these states is:



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Job

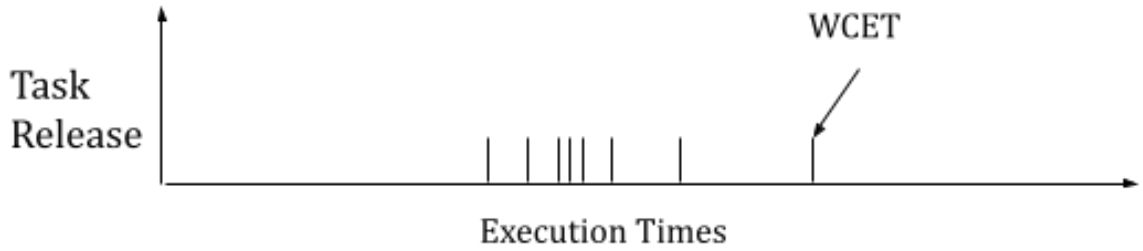
- If task is periodic, each periodic instance of the task is a Job
- for i-th Task τ_i, j-th job is J_{ij}



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Worst Case Execution Time

- WCET: the maximum amount of time a task can take to execute on a particular hardware
- Task takes certain time to finish execution
- This execution time is not deterministic, it can vary
- Variation can come from many sources: environment, other tasks, precedence constraints, memory access problems
- Task will never exceed WCET, it is the most pessimistic estimate



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Priorities

- Tasks can have relative importance over others
- For our discussion, $p_i = 0$ is the highest
- Task with higher priority can arrive later than one lower priority already executing
- Scheduling decisions can use this priority

Resources

- Processors, memories, buses are resources
- Resources are used to execute tasks successfully
- Processors: executes instructions; single core, multi core
- Memories: allow data storage and access; cache, ram, rom, etc.
- Buses: allow data transfer
- All of them influence WCET of a task

Periodic, Aperiodic or Sporadic Tasks

- Periodic tasks are those which need to execute at certain intervals
- Eg.: Temperature sensor and calculations
- Aperiodic tasks are those which only execute once when called
- Eg.: coffee machine, tasks involved execute once to make a cup of coffee
- Sporadic tasks are this which can arrive any point in time
- Eg.: Sudden proximity alert in car while driving (Sporadic tasks are highly undesirable System has to keep room for them all the time, while they arrive only a few times, resources are wasted)

Real Time System Scheduling

- Tasks need time on resources to produce results
- Scheduling: Scheme of allowing time on resources to tasks for their execution so that every tasks is able to execute without any missing its corresponding deadline
- Time on resources corresponds to real world clock
- Scheduling is the act of governing tasks executions

Scheduling Types

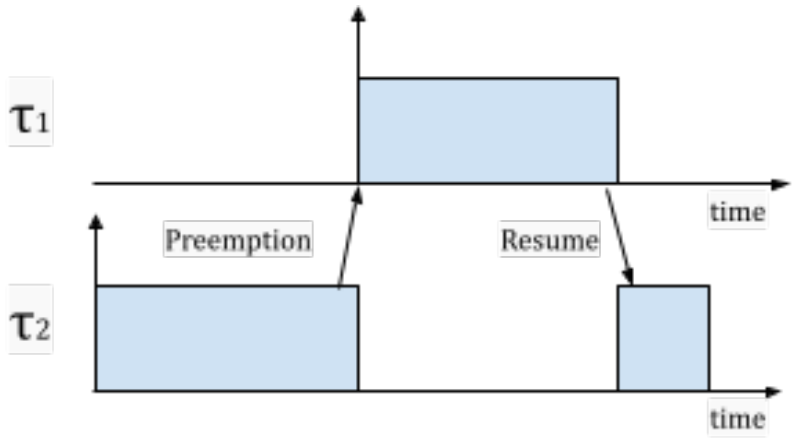
- Static vs Dynamic: Scheduling decisions are predetermined in static, dynaminc schedulers resource allocations can adjust depending on real-time demands
- Offline vs Online: Offline schedulers have predetermined rules, online can change those rules at runtime
- Preemptive vs Non-preemptive: Preemptive schedulers allow task interruption if another high priority task arrives, interrupted task goes to *Wait*
- Optimal vs Heuristic: Optimal algorithm minimizes some cost function; heuristic follows a guiding function which lay not be optimal

Schedule

- The policy/algorithm of allowing task execution
- Most used ones:
  - Fixed Priority (FP): Task priorities are fixed, highest priority task executes
  - Round Robin (RR): Rondo scheme, each task takes turn to execute
  - Rate Monotonic (RM): Task with high frequency (shortest period) has highest priority
  - Earliest Deadline First (EDF): As task jobs arrive, one with earliest deadline is executed first

Scheduling decisions: Preemption

- Each of these scheduling policies can be preemptive or non-preemptive
- Preemption is a scheduling choice, not task choice
- Preemption example:

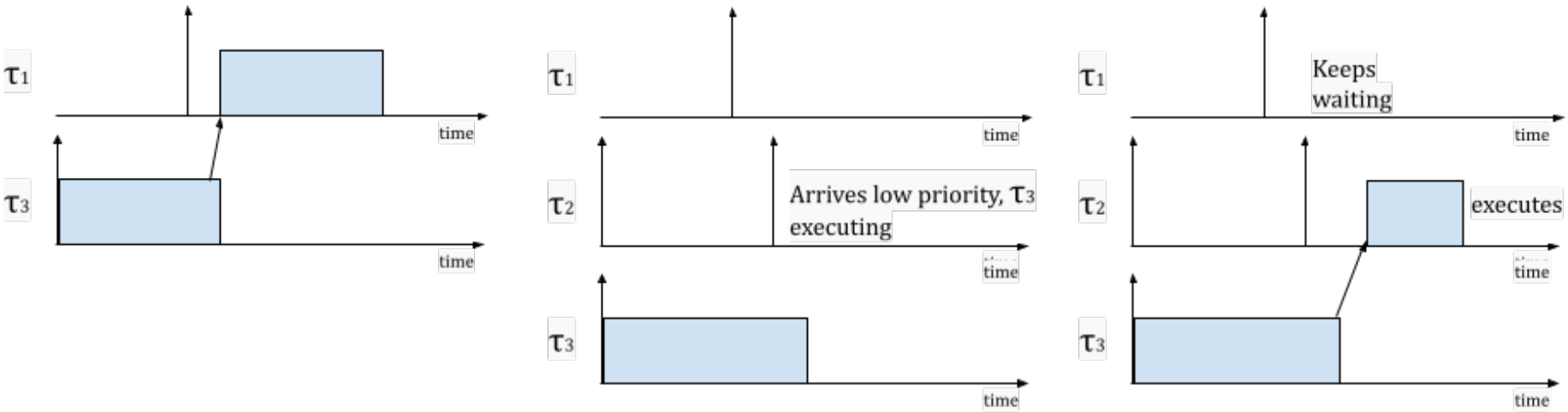


Scheduling decisions: Killing Task

- During execution, if WCET of a task is optimistic, it is possible that the task does not finish by its WCET
- Time allotment for task execution is based in its WCET
- In reality, once task is executing, their is no way of knowing that is has finished until task signals to the scheduler
- If there is no signal and task deadline has arrived, does the scheduler kill the task or allows it to finish ?
- Killing ensures real-time guarantees, not killing ensure functionality !

Priority Inversion

- Priority inversion is a consequence which may happen because of non-preemptive scheduling decision



Schedulability

Given a real Time System S is tuple

$$S = \{\tau_i, C_i, D_i, T_i, p_i\}, i = 1, 2, 3...n$$

- is it schedulable
- is it schedulable under a given scheduling policy? FP, EDF, RR ?
- Scheduability analysis is the process of obtaining functional guarantees of a real-time system

PROVIDE GUARANTEE THAT EACH TASK IN THE SYSTEM WILL FINISH EXECUTION WITHIN ITS DEADLINE FOR THE OPERATIONAL LIFE OF THE EMBEDDED SYSTEM !!

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Schedulability: Utilization

Task	C_i	T_i	p_i
τ_1	2	10	0
τ_2	4	15	1
τ_3	5	20	2

- CPU Utilization for task τ_i, $U_i = \frac{C_i}{T_i}$
- $U_1 = \frac{2}{10}$, $U_2 = \frac{4}{15}$, $U_3 = \frac{5}{20}$
- Total Utilization $= U_1 + U_2 + U_3 = 0.72 < 1$
- Since Total Utilization < 1, it is schedulable
- Does not guarantee yet ! Depends on the policy !

Utilization

Is the following task set schedulable by utilization ?

Task	C_i	T_i	p_i
τ_1	3	10	0
τ_2	2	15	1
τ_3	6	15	2
τ_3	2	20	3

Utilization

Is the following task set schedulable by utilization ?

Task	C_i	T_i	p_i
τ_1	3	10	0
τ_2	2	15	1
τ_3	6	15	2
τ_3	2	20	3

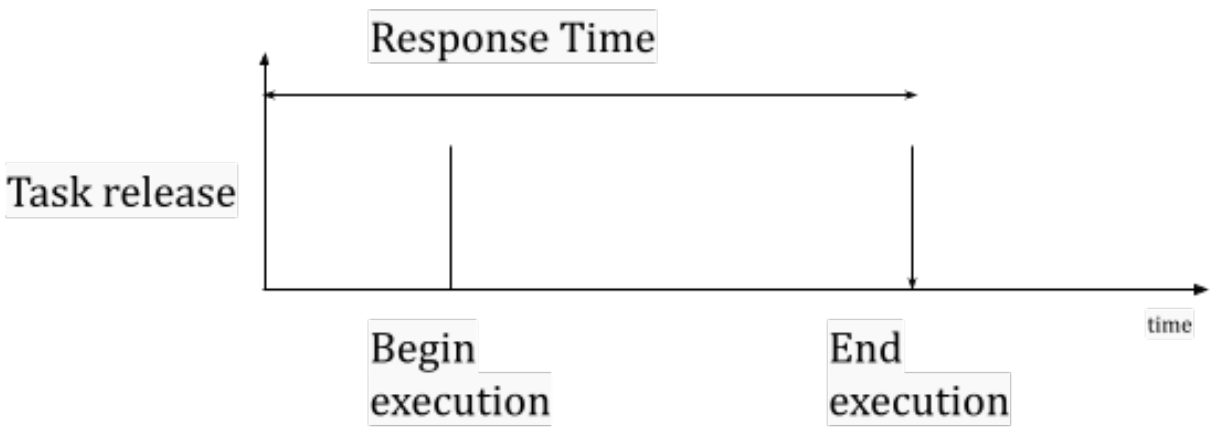
Utilization $=$

$$\frac{3}{10} + \frac{2}{15} + \frac{6}{15} + \frac{2}{20} = 0.3 + 0.133 + 0.4 + 0.1 = 0.933 < 1$$

Schedulability: Response Time Analysis

- Task arrives at a certain time
- Task begins execution a certain time and then it finishes
- Response Time: it is the time taken for a task to finish execution after its arrival
- For a job J_{ij}, response time R_{ij}:

$$R_{ij} = R_{ij}^{p_{ij}-1} + C_{ij}$$



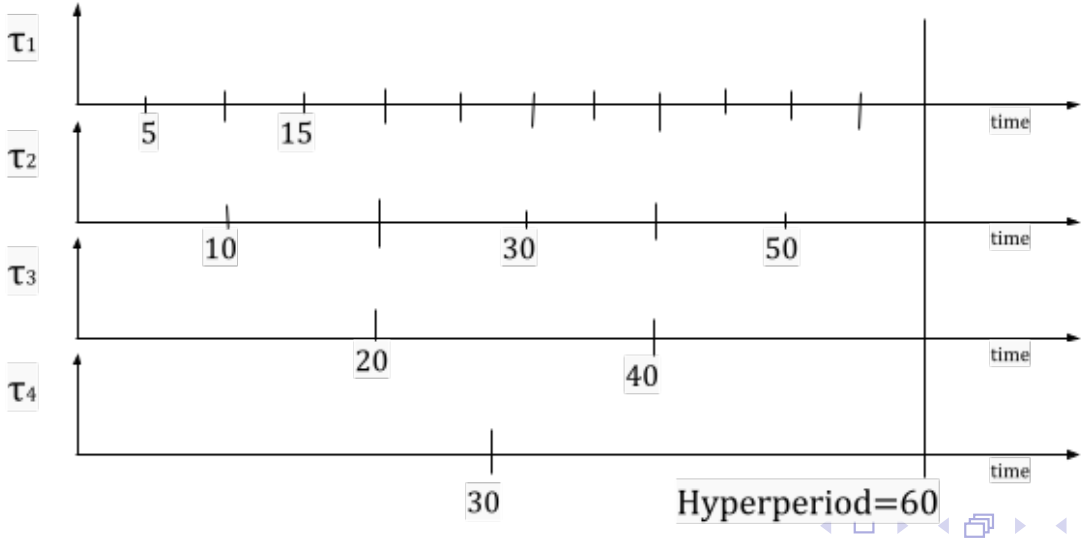
IPSA ÉCOLE D'INGÉNIEURS DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Schedulability: Hyperperiod

- Hyperperiod= $lcm(T_1, T_2, T_3...)$
- If the

  textbftasks are not allowed to exceed their deadline, Task execution pattern repeats every hyperperiod !

Task	C_i	T_i	p_i
τ_1	2	5	0
τ_2	4	10	1
τ_3	5	20	2
τ_4	5	30	3



Response Time Analysis

Using Response Time Analysis, is the following task set schedulable ?

Task	C_i	T_i	p_i
τ_1	2	5	0
τ_2	4	10	1
τ_3	5	20	2

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

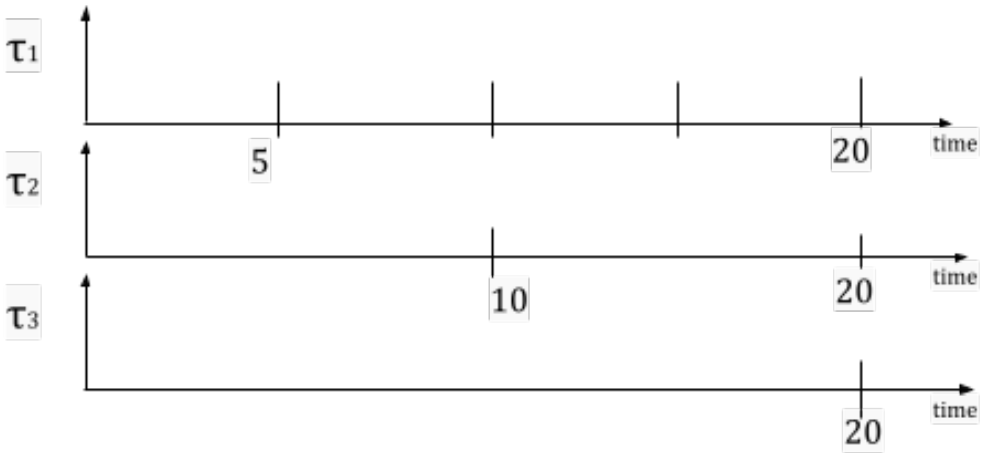
Response Time Analysis

Task	C_i	T_i	p_i
τ_1	2	5	0
τ_2	4	10	1
τ_3	5	20	2

- Hyperperiod ?
- How many jobs does each task has?
- What is the response time for the hyperperiod?

Response Time Analysis

Task	C_i	T_i	p_i
τ_1	2	5	0
τ_2	4	10	1
τ_3	5	20	2



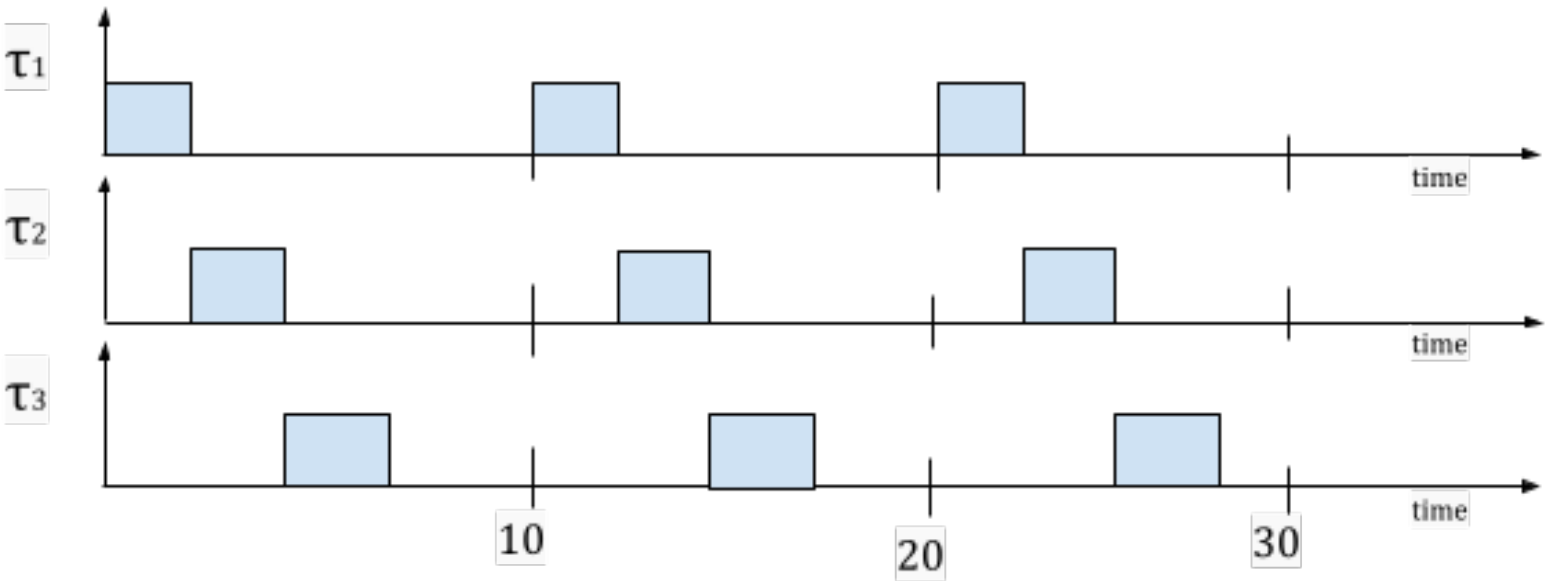
- Hyperperiod =20
- τ_1 has 4, τ_2 has 2, τ_3 has 1
- $RT = 4 * 2 + 2 * 4 + 1 * 5 = 21 > 20$

IPSA ÉCOLE D'INGÉNIEURS DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Round Robin

- Rondo scheme, every task takes turn in fixed order

Task	C_i	T_i	p_i
τ_1	2	10	0
τ_2	3	10	0
τ_3	2	10	0



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Round Robin

Show if the following task set schedulable using Round Robin?

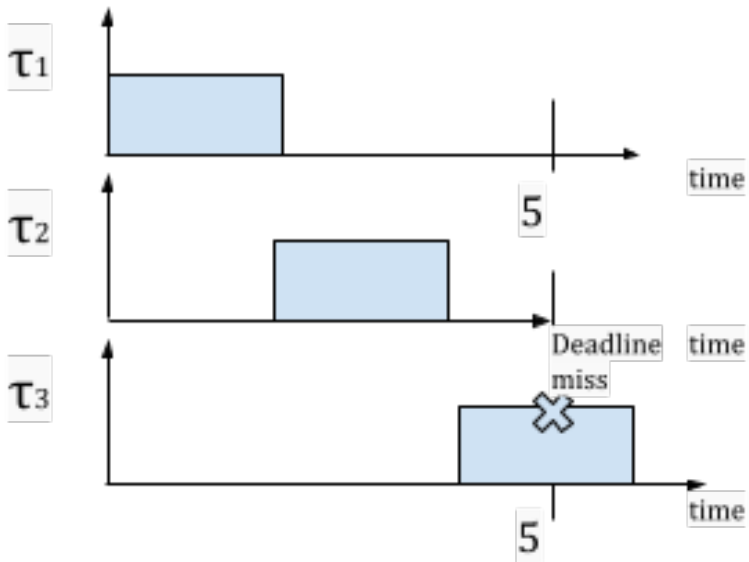
Task	C_i	T_i
τ_1	2	5
τ_2	2	5
τ_3	2	5

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Round Robin

Show if the following task set schedulable using Round Robin?

Task	C_i	T_i
τ_1	2	5
τ_2	2	5
τ_3	2	5

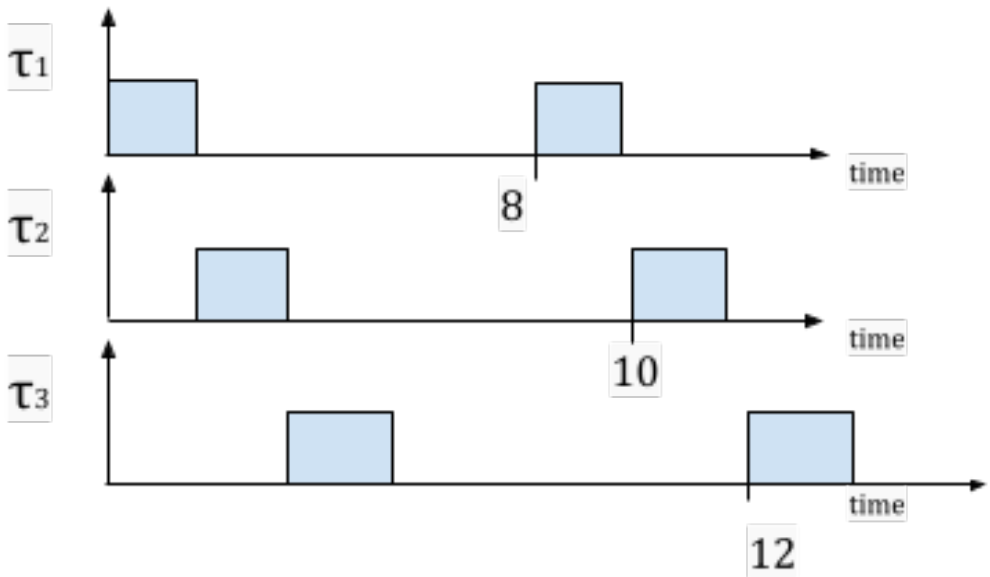


IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Fixed Priority

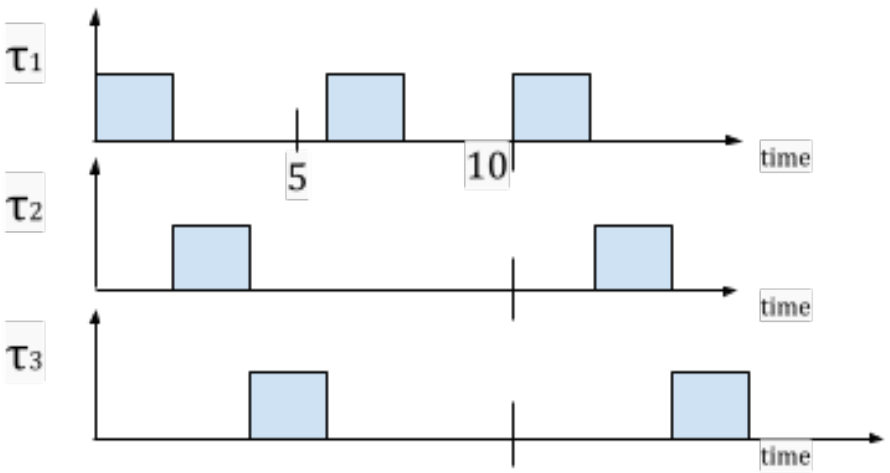
- Each task has a predetermined priority which remains fixed

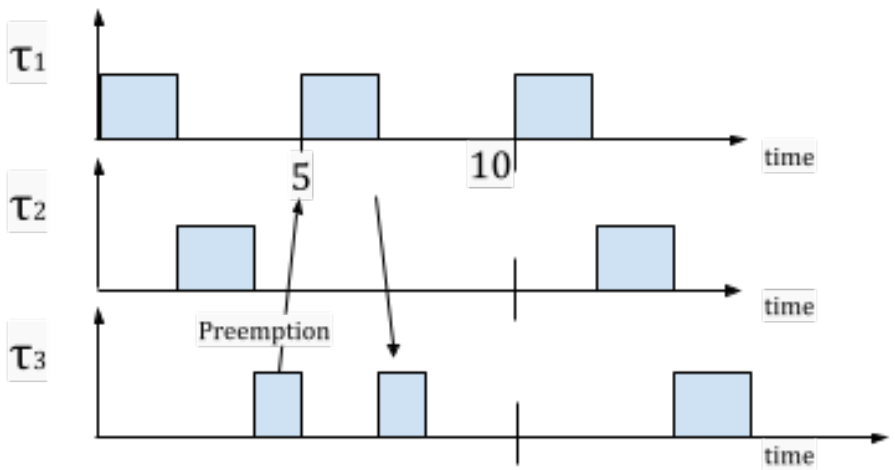
Task	C_i	T_i	p_i
τ_1	2	8	0
τ_2	3	10	1
τ_3	2	12	2



FP Preemptive

Task	C_i	T_i	p_i
τ_1	2	5	0
τ_2	2	10	1
τ_3	2	10	2





IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Fixed Priority

Show the following task schedulability using FP, Preemptive and Non-Preemptive?

Task	C_i	T_i	p_i
τ_1	2	5	0
τ_2	4	10	1
τ_3	5	15	2

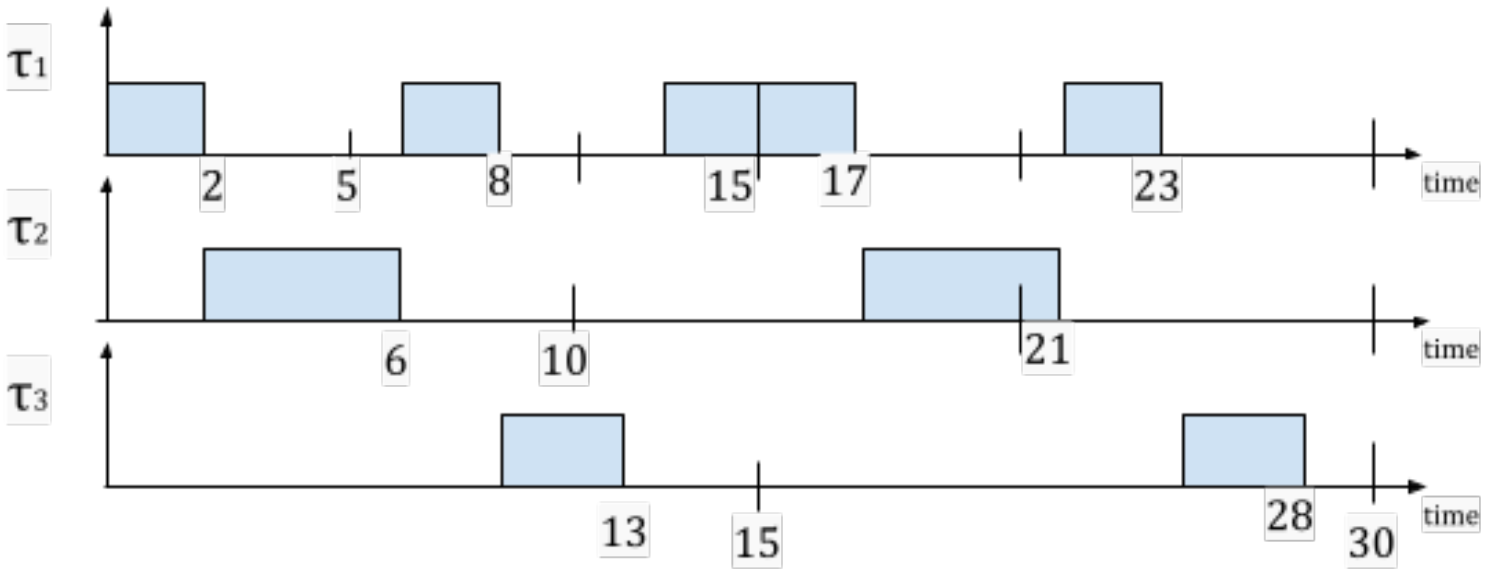
IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Fixed Priority

Show the following task schedulability using FP, Preemptive and Non-Preemptive?

Task	C_i	T_i	p_i
τ_1	2	5	0
τ_2	4	10	1
τ_3	5	15	2

Non-preemptive:



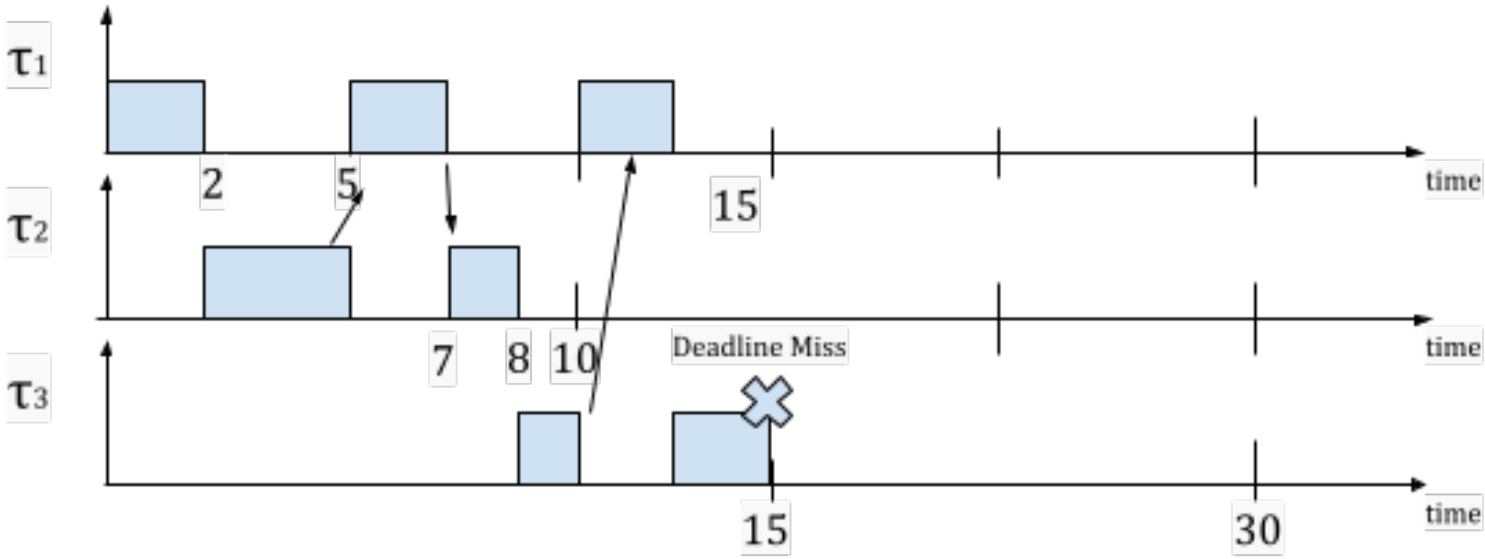
IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Fixed Priority

Show the following task schedulability using FP, Preemptive and Non-Preemptive?

Task	C_i	T_i	p_i
τ_1	2	5	0
τ_2	4	10	1
τ_3	5	15	2

Preemptive:

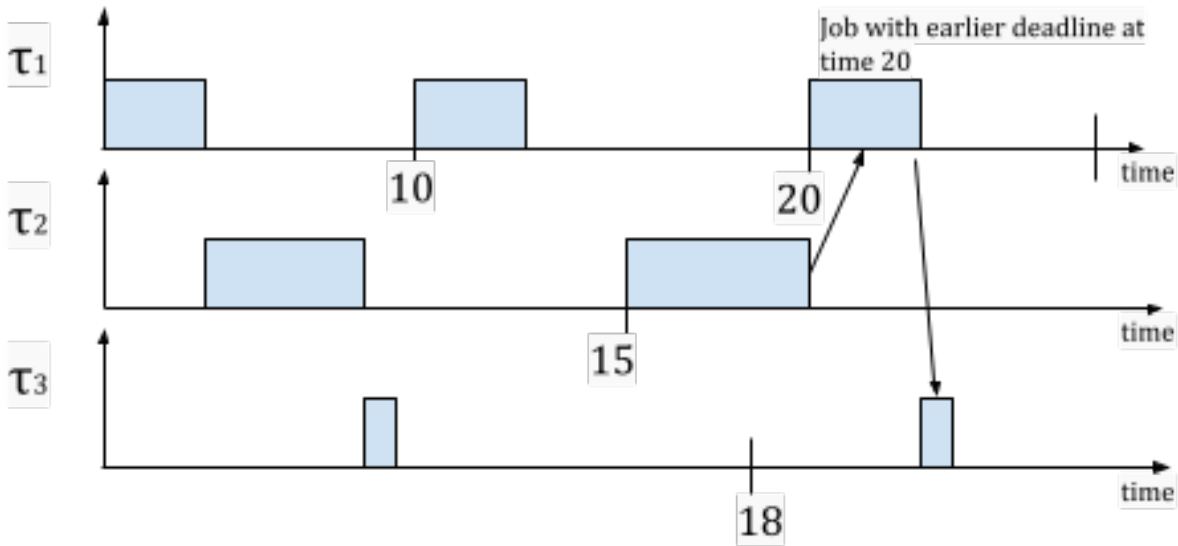


IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Earliest Deadline First

At any given time, an available job with the earliest deadline gets the priority

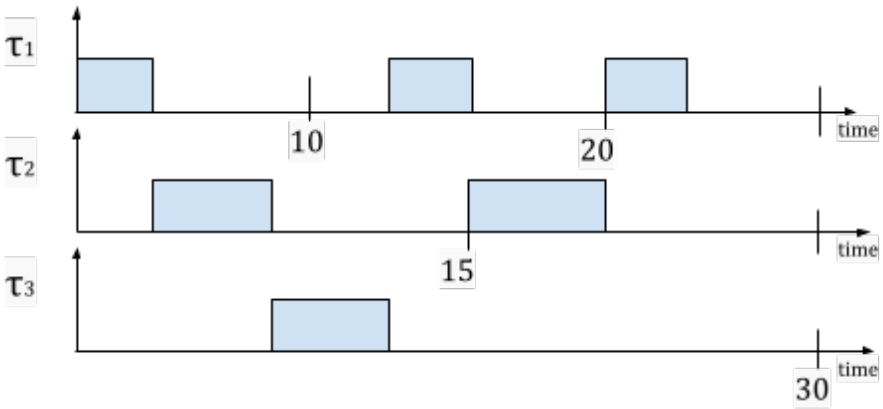
Task	C_i	T_i
τ_1	3	10
τ_2	5	15
τ_3	1	18

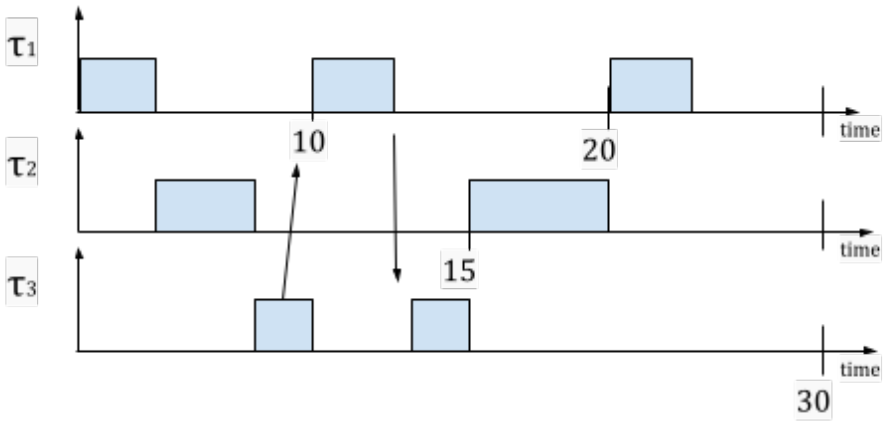


IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

EDF preemptive

Task	C_i	T_i
τ_1	3	10
τ_2	5	15
τ_3	4	30





IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Scheduling Policy: Least Laxity First

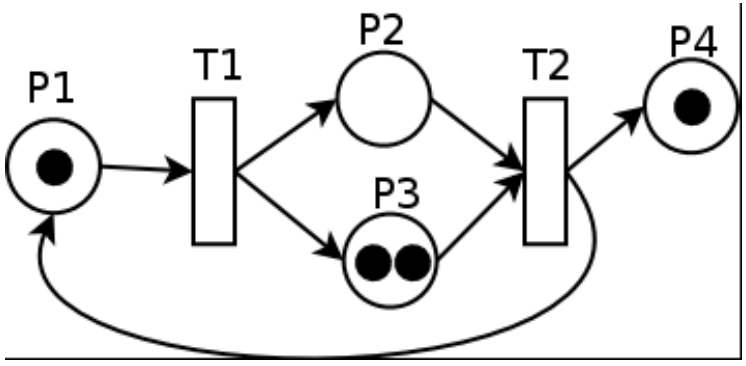
- Dynamic priority scheduling policy
- Laxity: amount of time left after job finished execution
  $l_i = (D_i - C_i)$
- Job with lowest laxity gets the highest priority
- Priority evolves during execution
- Useful for system with aperiodic tasks, since there is no assumption is made for rate of occurrence
- But it does not look ahead, it only works on current state of the system

Schedulability: Formal Methods

- Formal methods are semantics with underlying rules which have provable properties
- Finite state machine:
  - Petri Net
  - Automata
  - Timed Automata
  - Stochastic Automata
  - Markov Chain

Schedulability: Petri Net

- Model of states and transition between those states
- Transitions are enable by a trigger
- States can contain tokens
- Initial state of the model constitutes a token in the first state
- Trigger can be triggered iff there is toke in the previous state
- Tokens move around in the system
- Provable properties:
  - Deadlock
  - Aliveness
  - Reachability



Similar modelling methods exist with Automata and its variants

Markov Chain

- Can be used to model stochastic behaviour of the real-time system
- States and directed transitions between them
- Each transition has a probability of being taken
- Properties:
  - Probability of being a state (Eg: probability of deadline miss)

(Purely academic method so far)

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

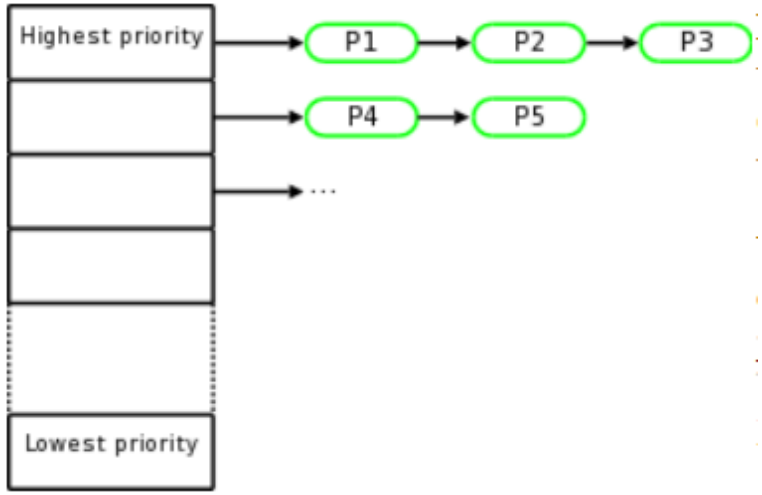
Scheduability

The ideal process would be

- Identify processes or tasks
- Obtain WCET (the most difficult part)
- Schedule and prove schedulability

Linux scheduler

- Currently Linux uses Completely Fair Scheduler (CFS)
- CFS is based on Rotating Staircase Deadline Scheduler(RTDS)
- RTDS: There is a priority list and each priority has processes in it
- At each of those priority list, processes have a time quota
- The scheduler rotates through the priority list in RR
- Say four process with 4,10,6,2 ms of WCET, if time quota is 1ms, each task is given 1ms to execute



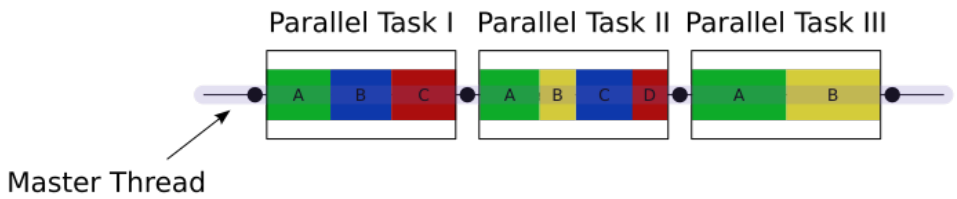
IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

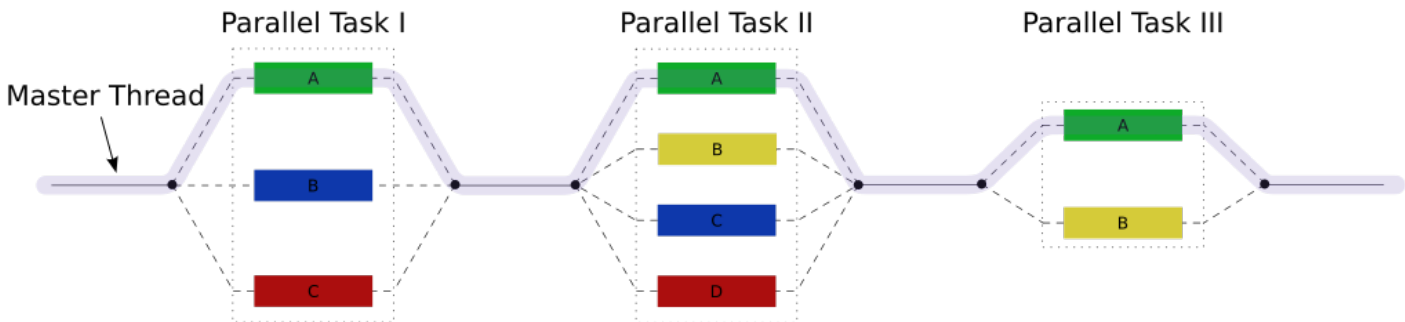
Pessimism

- Pessimism in the context mean over estimation
- Eg.: If a task WCET is measured 20ms, pessimistic number would be 25-30ms
- When designing a real-tims system, worst of the worst cases should be considered and accounted for
- Over pessimism leads to over allocating resources
- Over allocating may result in processor doing nothing if the task actually finished early
- Optimism may result in deadline misses or jitter

Parallel processing

- Executing multiple tasks in parallel at the same time
- Parallel processing on a single core processor is only time slicing
- Fork-join model: Single master thread
- Creates forks with each thread as a process
- Fork join back into master after their designated task is done





Multi-core

- True parallelism can only happen in multi-core, single core is only time slicing
- Performance enhancement in multicore is function of the cores are handles, it still is one computer
- Eg: rp2040 micro-controller has two cores,
  - core 2 is only accessed through core 1
  - core 1 is always the master

Unbounded, Bounded

- Constraints: external situations or implementation that hinder the task execution (apart from itself)
- Bounded constraint: those constraints which are deterministic and their affect on the task execution can be predicted
- Eg: waiting for another task to finish execution based on its WCET is bounded constraint
- Unbounded constraints: those constraints which non deterministic, their affect can't be predicted
- Eg: recursive functions, waiting for memory acces are unbounded constraints

Analysis Tools

- Some commercial tools are available to peform statis analysis of tasks to obtain their WCET
  - RapiTime: By Rapita Systems, it collects execution traces and derives execution time measurement statistics
  - aiT: developed by Absint in the DAEDALUS European project, for avionics (used for A380), analyzes binary executables taking the intrinsic cache and pipeline behavior into account
- Some commercial Schedulability Anlysis tools
  - RTDruid
  - TimeWiz
  - symTA/S
  - chronVAL
- There also exist some Real-Time simulators
  - RTSim: developed by Retis Lab of the Scuola Superiore Sant'Anna of Pisa (Italy)
  - TrueTime: Matlab/Simulink simulator
  - chronSIM

Some considerations

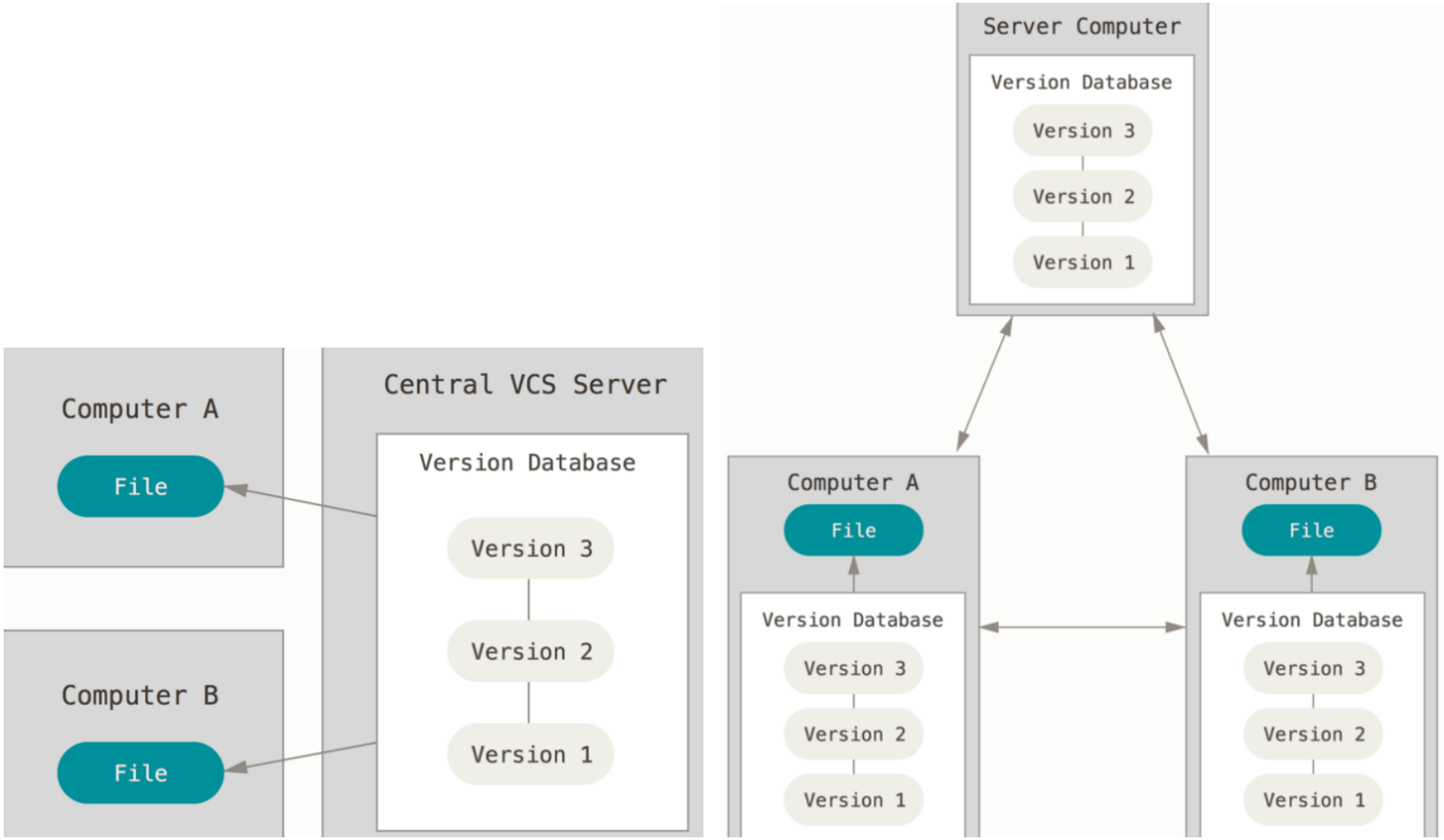
- Unbounded Constraints: Factors affecting the execution time of the system which cannot be predicted
  - Any unbounded constraint must be avoided.
  - Eg: Executing task waiting for a sporadic task
  - Extremey difficult to obtain WCET for such task, or WCET is too pessimistic
- Sporadic Tasks must be avoided
  - Scheduler needs to always have time allotment for a sporadic task which may rarely arrive
- Task waiting for memory, its an unbounded constraint
- Coding Deamons:
  - *goto* statements
  - nested loops, recursing (self-calling) functions
  - Dynamic memory allocation
  - Depends on compiler

Version Control

- A system that records changes to a file or a set of files
- Doing a personal projects
  - try
  - try_1
  - try_better
  - working_version
  - working_better
  - final
  - final_2
- This might even work for yourself
- Working in a large team (50-100) people, you cannot tell everybody which is latest one and everybody stop doing what they are doing and please get the latest copy !
- You need a version control tool

Version Control

- Version can be centralized or distributed



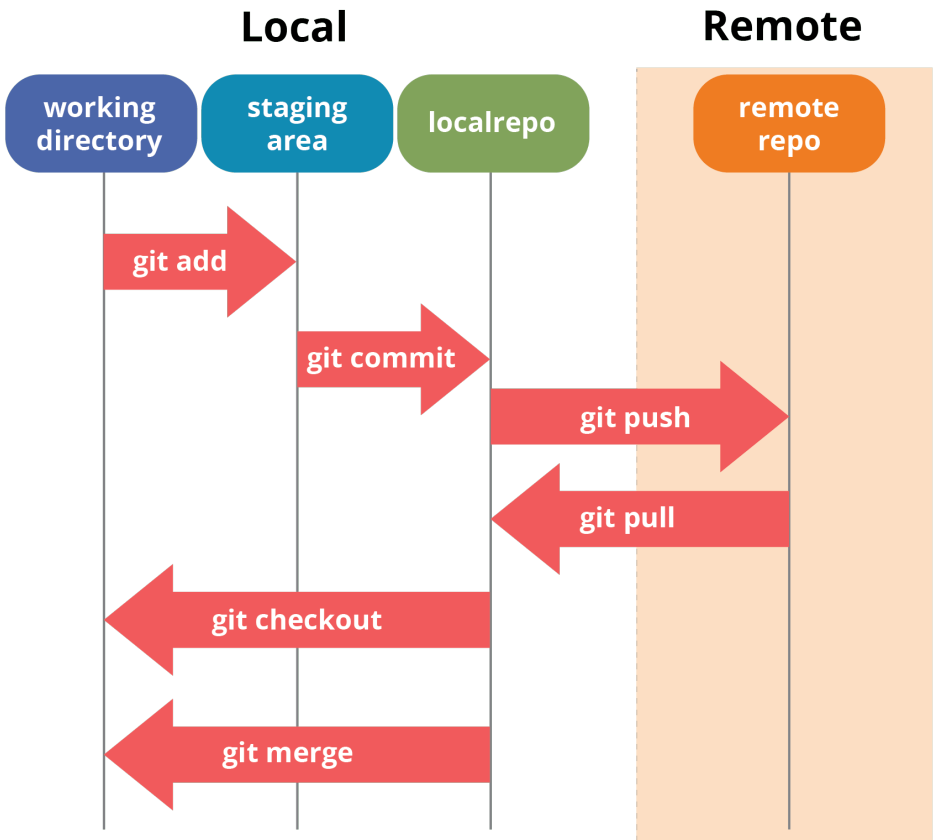
GIT

- GIT is one tool for version control
- You need a GIT account, Github or Gitlab
- Then you create a repository
- Repo: the set of files for your project
- Repo can be online and local

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT First step

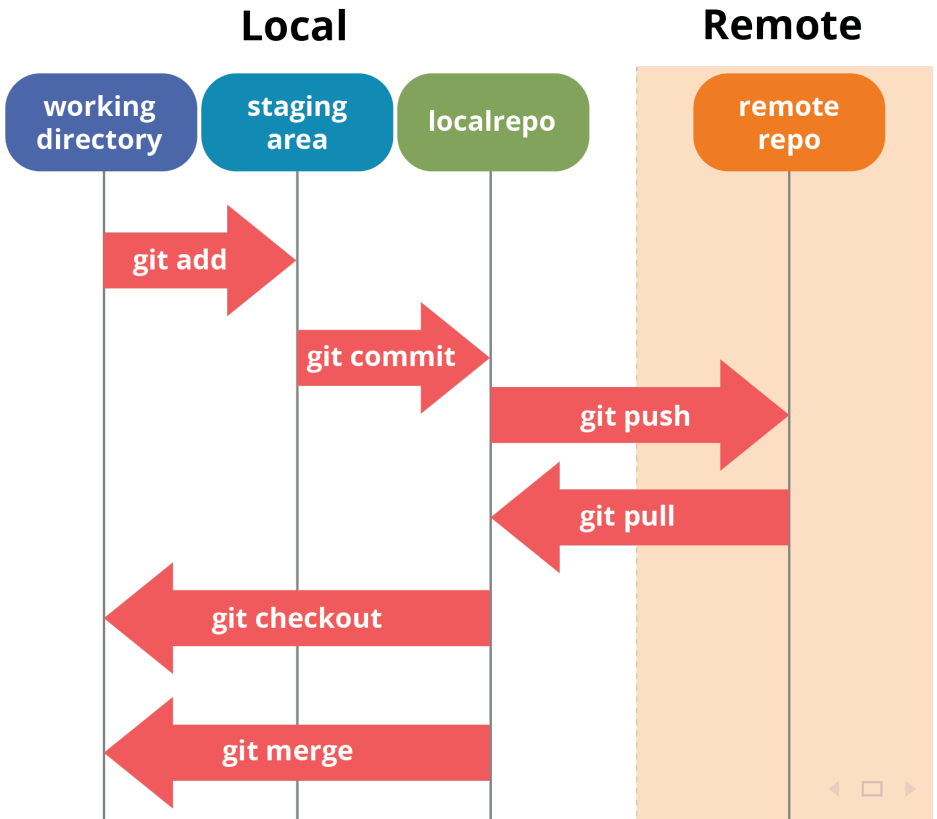
- First you need to create a repo on GIT
- Then you can add files to it
- If there is already a repo, you can *clone* it
- Once you clone, you can can edit, make changes, add or remove files
- After you are done, you *commit* your changes and *push*



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT Workflow

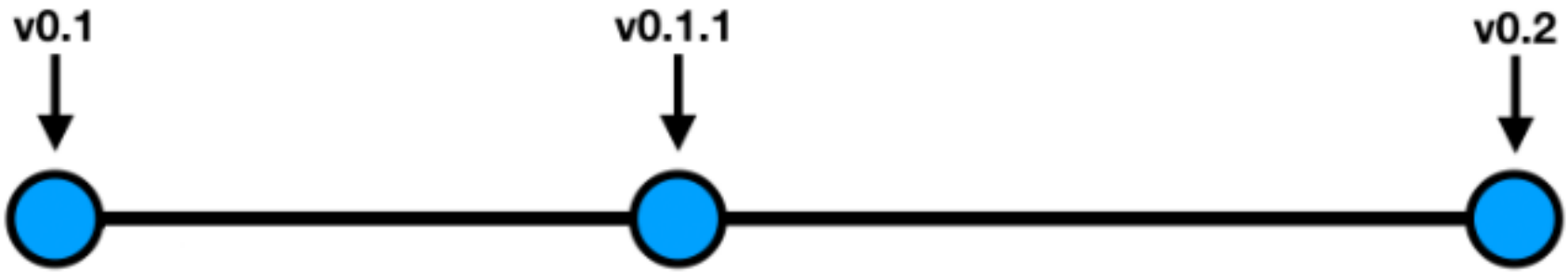
- Once you have started, you have a local repo and remote repo
- Local repo is loca to your machine, remote repo is the one on the server
- Both repos have versions
- In order to get the latest version from remote you *pull*
- In order to update the remote version to your local version you *push*



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT branches

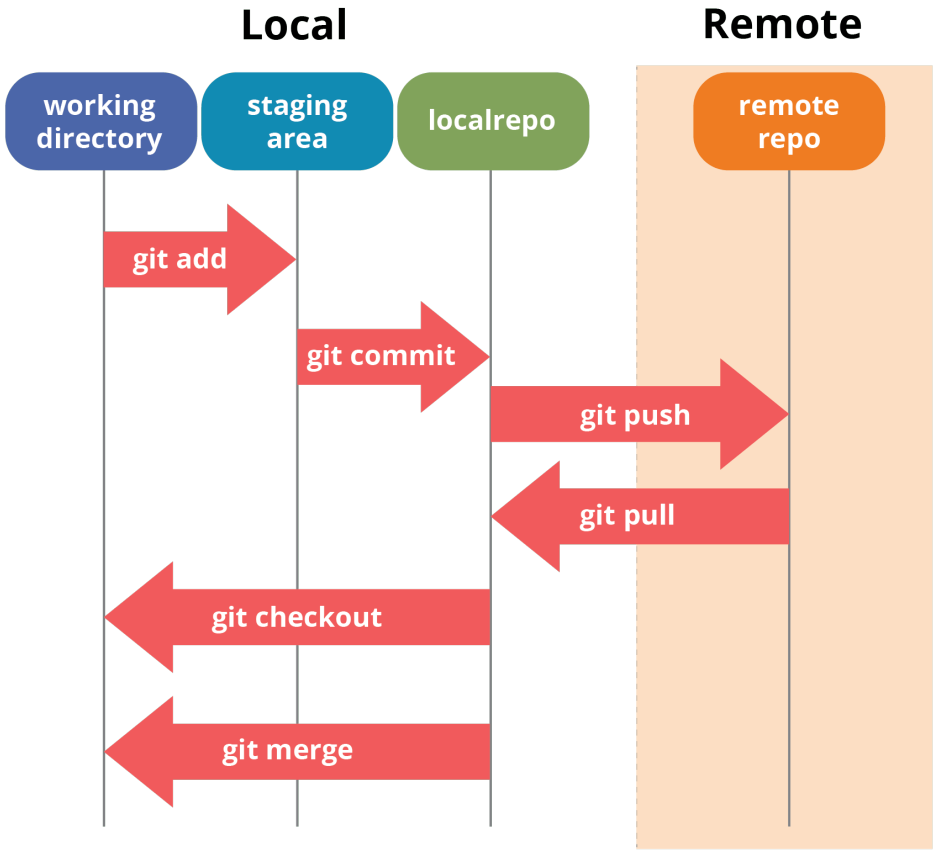
- The remote repo keeps a record of every update ever made
- Say, initial version is v0.1
- Next time you work on it ans push your latest updates, v0.1.1
- It gets added to the repo history with lates update as the head
- This is called a branch
- You can always go to any 'version' of the branch



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT Workflow

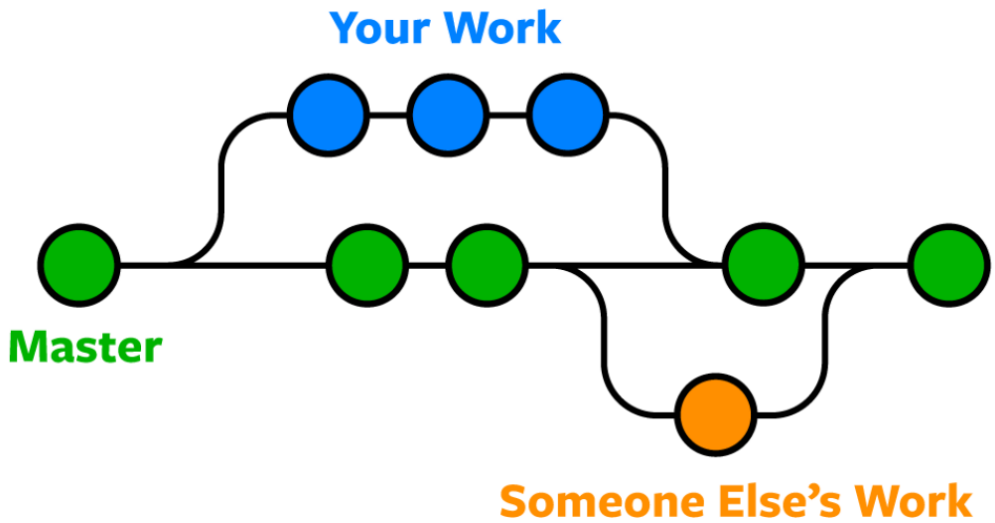
- This going back or forth to a branch is called *checkout* a branch
- Thus, once your local repo is at the same version as remote, you can checkout any branch



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT branches

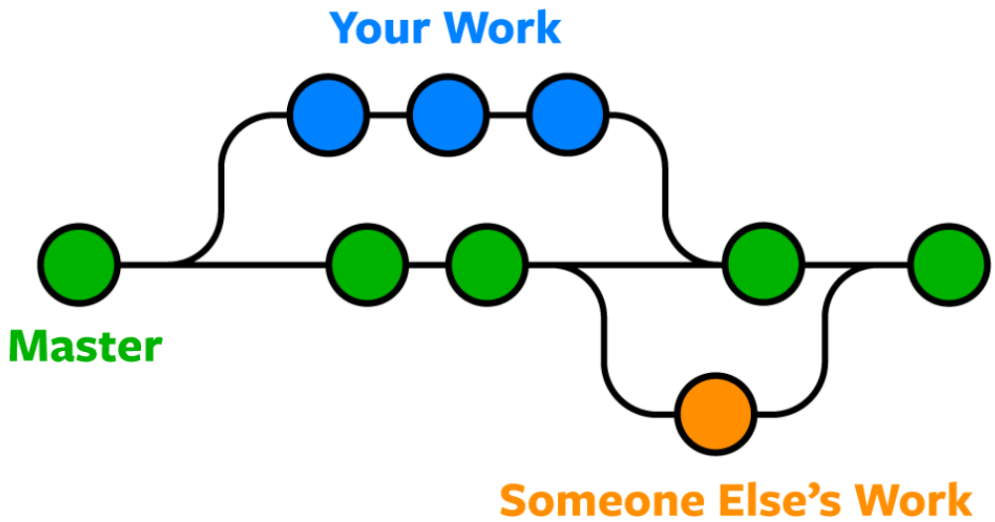
- In software development you have a **master** branch and **dev** branch
- Master branch has the latest stable (official) version of the software
- Dev branch allows development



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT branches

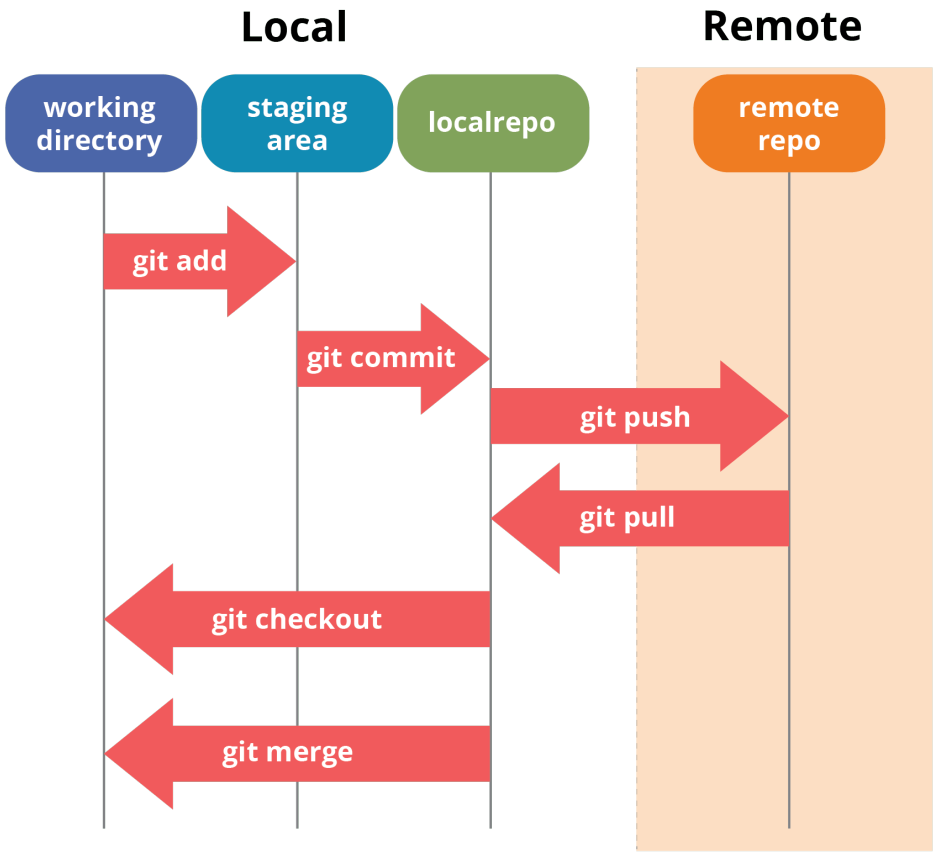
- When you have a team working on a project, each person can have their own modifications to do
- From dev, everyone has their own branches to work on



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT branches

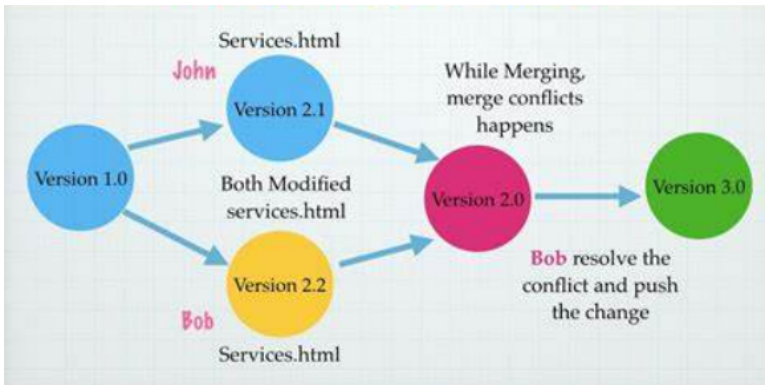
- When you are finished with your changes you *commit* and *push*
- Then you need to merge your changes on the *master* branch !
- Merging makes your changes at the same version as the latest on remote



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT branches

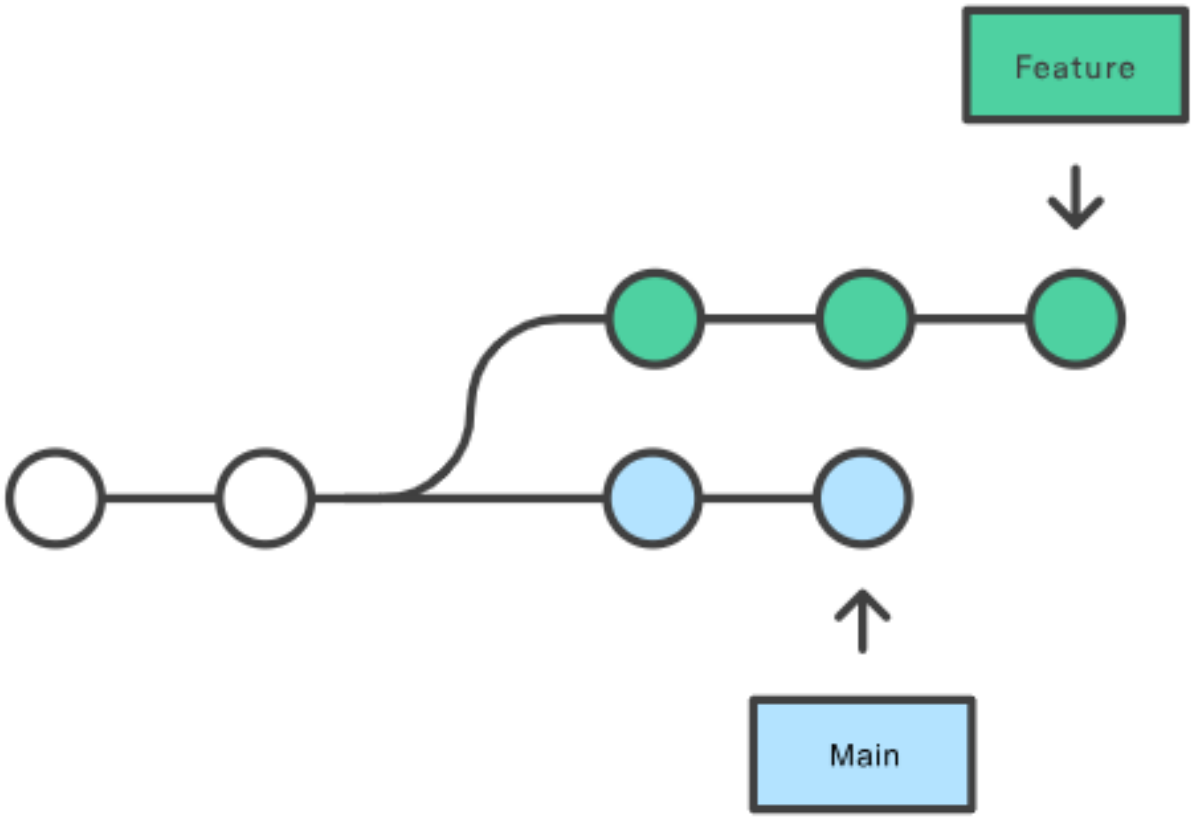
- What if somebody else has worked on the same file as you, or worked on the same line in the same file as you?
- Then you have **conflicts**
- Conflicts need to be resolved
  - You can *accept incoming file*
  - You can *keep you file*
  - You can *keep both changes*



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

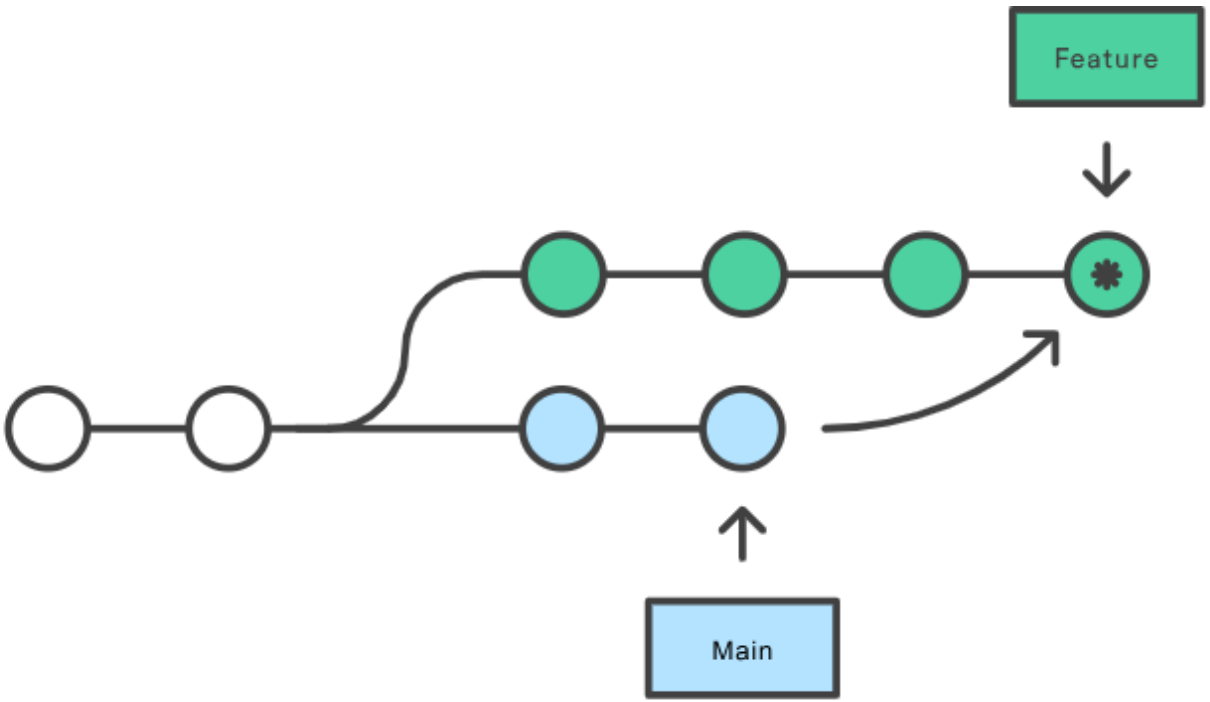
GIT Merge vs Rebase

- Merge: creates another checkout with merged changes
- Rebase: extends the branch as if the checkouts on another branch also happened to the current branch



GIT Merge vs Rebase

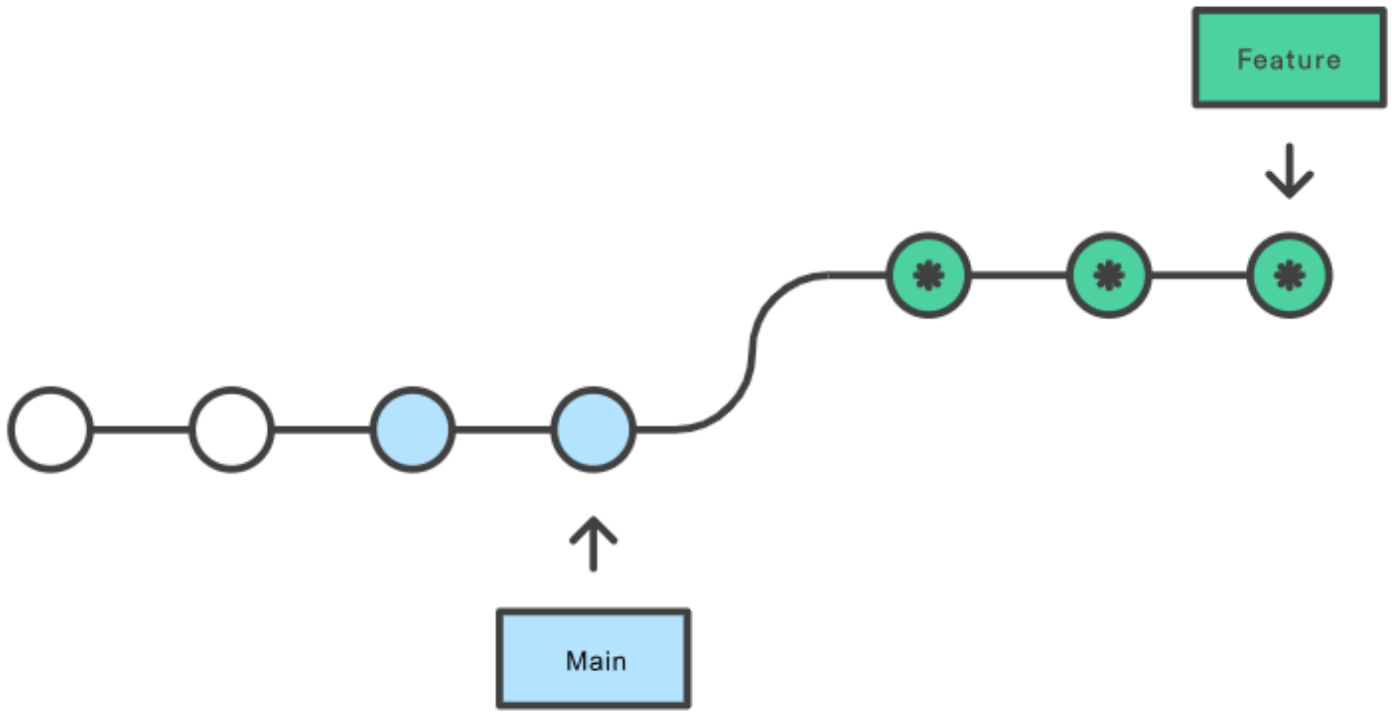
- Merge: creates another checkout with merged changes
- Rebase: extends the branch as if the checkouts on another branch also happened to the current branch



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT Merge vs Rebase

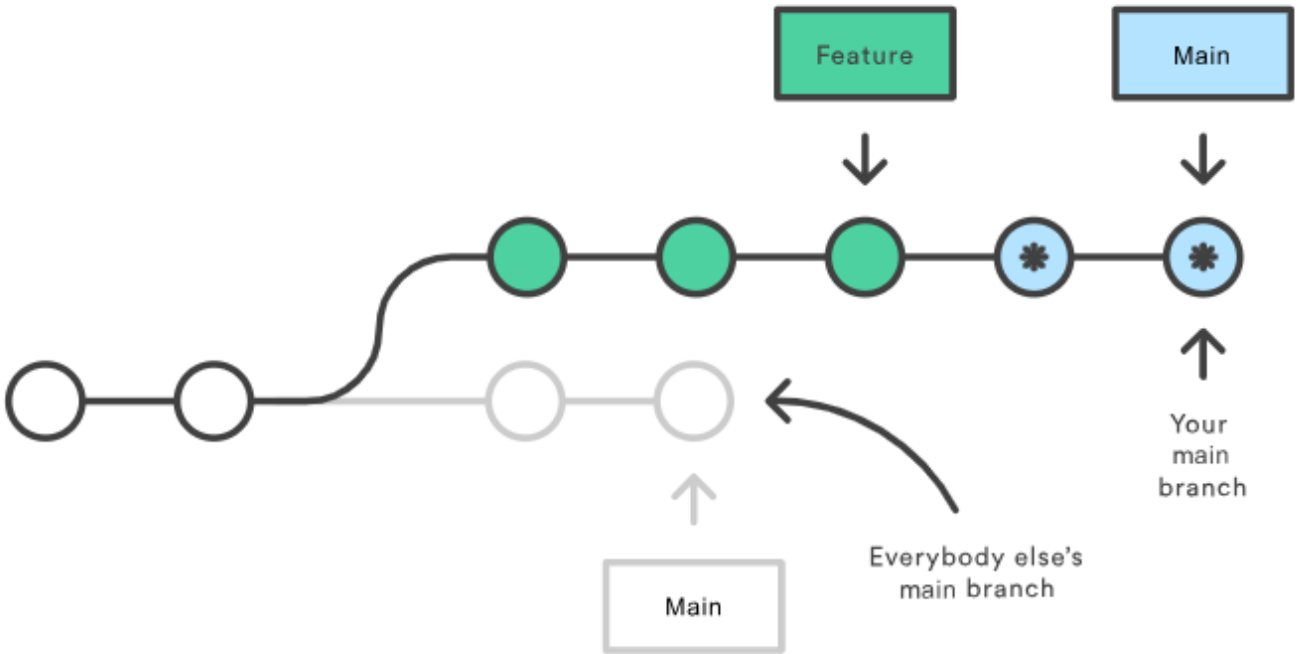
- Merge: creates another checkout with merged changes
- Rebase: extends the branch as if the checkouts on another branch also happened to the current branch



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

GIT: careful

- Rebase on public branch



GIT: careful !

- Force Push: when you have a conflict you have the option of force push
- It pushes your changes and overwrites the existing ones
- For example, you can checkout an earlier version and want to remove everything after that, you will force push

Further reading: https://www.atlassian.com/git

Pull Request

- Once you have finished your work, you need to merge them to master
- You open a Pull Request
- It is an official request for the code owner to review your changes, ask for modifications or corrections
- PR also accounts for testing
- Once everything is ok, your work can be integrated

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

CI-CT

- Continuous Integration Continuous Testing
- During software development, you have various releases while undergoing development in parallel
- It is not that a final product is delivered only after being finished, then if there are some problems you dedicate teams to solve them
- GIT is extensively used for development
- Testing is done in parallel to development

CI-CT workflow

- There is a master and dev branches
- You get a task to do, eg.: develop a function
- You create your own branch and work on it
- You finish, commit, push, merge/rebase so that your branch is up to date and your work is included
- You set up a pull request
- Your complete code goes for jenkins pipeline
- Jenkins pipeline is set up to perform various tests on your code like unit tests, coding standard tests, etc.
- Once it is green, the code owner will integrate your changes on master

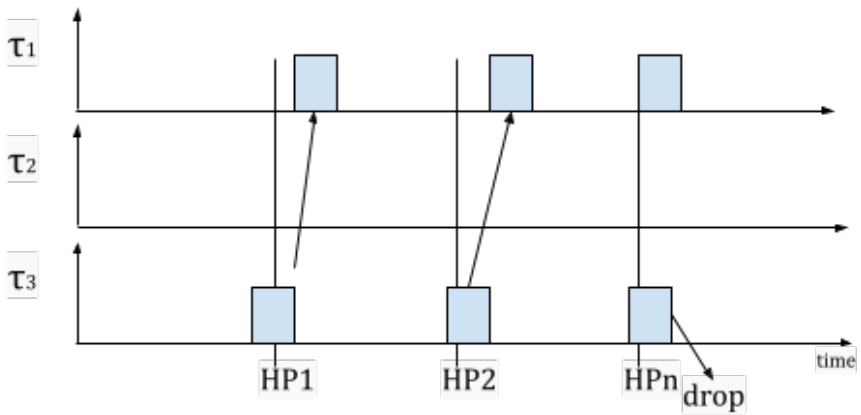
IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Deadline Miss handling

- If there is a deadline miss, what are your actions
- Are they predetermined or decided on the go ?
- Do you evaluate, during execution or beforehand, the cost of deadline miss (cost is not necessarily financial)
- Eg; Do you kill the task or let it finish ?
- Some ways you can deal with this.

Deadline Miss handling: Job suspension

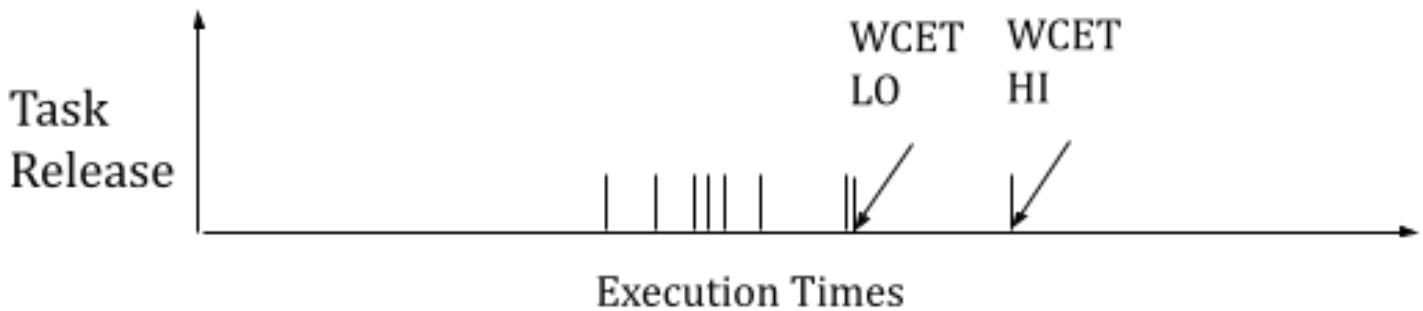
- If you allow task to execute beyond its deadline,
- And if this keeps happening while system execution, you can collect how much overshoot is happening
- Set a predetermined threshold to the overshoot
- If overshoot exceeds, you can drop a job execution
- It can be difficult or impossible to implement in hard real-time system
- Eg: Dropping a frame in a video stream



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Deadline Miss handling: Mixed Criticality

- Two types of tasks *HI* and *LO* criticality tasks
- *LO* criticality tasks are 'normal' tasks
- *HI* criticality tasks have two WCETs C_i^{HI} and C_i^{LO}, ($C_i^{LO} < C_i^{HI}$)
- Real Time system has two modes of functioning *LO* and *HI*



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Deadline Miss handling: Mixed Criticality

- While the system executes, the tasks are scheduled using the LO WCET for all tasks
- If one task exceeds C_i^{LO}, the system switches to HI criticality mode
- Then, all the LO criticality tasks are dropped from execution
- The time slots gained from dropping all the LO tasks are given to HI criticality tasks
- HI criticality tasks are scheduled using C_i^{HI}
- System can switch back to LO criticality mode using certain condition like after a certain time (or other more robust condition)

System Modelling and Application: a philosophy

- Why would you need a model of system: too complex to calculate by hand or to compute, certain properties you can't prove with measurements, etc.
- When you "need" a model of real-time system, you need to know how and what to choose
- You could have a theorem which covers your implementation and prove it
- You could have a formal method to chose from
- Graph theory: model system with states and transitions, define semantics
- Always ask yourself: what am I getting out of this model ? !

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Fluid Dynamics

- Study of fluid flow in or around objects (around circle, sphere; in a pipe, etc)
- What are the properties of fluid flow you are interested in ?
- For exmaple, flow rate: quantity of fluid passing through a cross section per unit time
- Does it map to R/T systems ?
- Maybe task execution can be mapped to fluid flow rate
- If task executions in a system can be modelled with fluid flow through pipes and a network of pipes, could it represent real-time system execution ?
- What am I getting out of it ?
- Deadline miss could be fluid clog ? or backflow ?

Game theory or Chaos Theory

To Think

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

USE A THEORY WITH ITS ASSOCIATED PROVABLE PROPERTIES TO YOUR ADVANTAGE BY MAPPING AND INTERPRETING IN YOUR CONTEXT !

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Modelling: Hidden Assumptions

- Any statement you make can have, most likely does have, hidden assumptions !
- A theorem without specifying the assumptions is incomplete
- Eg.: Spuri, Buttazo: Given **a set** of **periodic tasks** with **processor utilization** U_p and **DPE server** with processor utilization U_s, the **whole set** is schedulable iff $U_p + U_s \leq 1$
- It is a well bound statement
- (ExB)

Scheduling vs Finding Schedule

- Scheduling is the method of assigning time for task executions
- Scheduling policies are well known generalized ways of scheduling tasks
- Sometimes those policies are not applicable
- EDF: is made for optimality, a policy with max allowed processor utilization across other policies; (Optimality is a hidden assumption)
- Sometimes you are required to keep processor utilisation between some limits (eg.: 50-60 percent)
- You will have to find a schedule

Finding Schedule

- One way to find a schedule is to find a patter, theorize and prove
- Eg.: For all periodic tasks...on a unicore processor...other assumptions...is schedulable...iff... etc
- Another way to 'go look a schedule'
- Implies exploratory methods where you find a schedule that fits your needs
- Graph theory, find combinations of schedule, calculate paramaters, decide on a schedule
- Exploratory methods a computationally expensive, they can be complex !
- (ExB)

Complexity

- Definition of complexity depends on your system
- We are talking about complexity of algorithm, particularly complexity of method to find a schedule which fits the requirements
- Complexity can be expressed as the order of the relationship between input and output
- Complexity expresses how long and resource intensive the algorithm will be if implemented
- Bog O notation is used as relationship between input and implementation of algorithm
- Usually there is time complexity and Space complexity
- Time complexity refers to the time it takes for the algorithm
- Space complexity refers to the memory it needs

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Complexity: Search

- Say you have a unordered list and you want to look a particular value/entry in the list
- Linear search: The algorithm is simple, go through the list one-by-one and stop when the element is found
- The worst case is when the element is in the last entry of the list
- It will take iterations equal to the length of the list to find the element
- As the length of list increases, so does the worst case time for finding the element
- The complexity is linear $O(n)$

Complexity: Search

- Binary search: if the list is ordered, say in ascending, then binary search can be used
- Compare the middle value to the value you are looking for
- If middle value is less, look right, else look left
- Then divide the list in half, compare the middle value, and so on..
- Worst case complexity is $O(logn)$

You can perform same operation more efficiently !

Complexity: Sorting

- The objective is sort a list in ascending or descending order
- Selection sort: Comparing each element with the unsorted list
- Take first element, compare it with all the list, if a lower value is found, exchange the positions
- Second element, compare with the rest of the list, if a lower is found, exchange
- For each element the list went through once, but the list to go through becomes shorter with each iteration
- Time complexity is $O(n^2)$; space complexity $O(1)$

Complexity: Sorting

- Bubble sort: comparing element to the next
- Traverse from left and compare adjacent elements and the higher one is placed at right side
- The largest element is moved to the rightmost end at first
- Continue to find the second largest and place it and so on
- Similar number of passes, time complexity $O(n^2)$; space complexity $O(1)$
- Relatively easier to implement

Complexity: Sorting

- Merge sort: sorting by recursively dividing the list
- Divide the list until it cannot be divided further
- Single element array is always sorted
- Merge them back same way together in increasing order
- Time complexity $O(n\ log(n))$, space complexity $O(n)$

Complexity

- When looking for schedule in exploration model, searching and sorting can be implemented
- But different combinations of job execution order leads to searching, sorting, computing metrics across a tree
- Such algorithms can be expensive
- Therefore, in real-time systems, find a schedule can be a complex task
- If finding a schedule is done online, extra computation power is needed to do that with its own real-time constraints !

Safety of Critical System

- Hard real-time systems are safety critical
- It means the functioning of such a system is directly related to safety, of human lives, operation of the device, etc.
- Notion of **safety** requires a clear definition depending on context
- How do you define safety for satellite ? car ? sitting and working in office?
- How do we ensure safety in real-time embedded system
- Say, safety is defined as no task should miss a deadline
- Schedulability analysis
  - Response Time analysis
  - Formal methods
  - etc

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Safety of Critical Systems

- Worst Case scenarios: for a given system, what is the worst that could happen ?
- If you can see what is the worst that can happen, it probably might (Murphy's Law)
- In real-time system, one worst case scenario is when all the tasks are released at the beginning
- When the system initializes, all the tasks are ready for execution
- This implies the processor demand is the highest, it is the maximum load

Safety of Critical Systems

- Can your scheduler deal with highest processor demand at the beginning ?
- Can your scheduler handle any overload conditions ? (job reaching deadline, maybe job suspension ?)
- Can your scheduler deal with domino effects ?
- Is there any priority inversion ?
- Are precedence constraints being respected ?

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Safety: Semaphores/Mutex

- Two tasks can share same memory space, especially if there is inter-task communication
- That is, two tasks need each other to accomplish their respective tasks
- To have such a "communication" between two tasks, a "medium" is needed
- If not, tasks may read obsolete data, or overwrite an existing data which might be needed by another task
- That's when you need semaphores or mutex

Critical Section of a task

- A process/task has a segment of its code which reads/writes data on a memory space; it performs a data update or a table update
- When the task is in this segment, it is said to be in a critical section
- If a task enters a critical section, no other task is allowed to enter a critical section
- This "allowing" or "not allowing" is implemented as semaphores or mutex

Race Condition

- Another usage of mutex is to prevent **race condition**
- Multiple threads or processes are accessing and changing shared data
- Especially once you create multiple threads as a fork from one thread
- You don't know how the scheduler is scheduling the tasks, you don't know which thread executes first, which one second, when it goes back to first, etc.
- It leads to nondeterminism in the system
- The threads will the 'race' to access the shared data
- It can lead to software bugs

Mutex

- Mutex or Mutual Exclusion, is a lock used to protect critical section and prevent **race condition**
- A process has to wait to get the mutex lock to enter critical section

Algorithm 1: *acquire()*

```
while !available do
    busy wait
end
available = false
```

-
- Once it has finished and exits critical section, it gives back the lock

Algorithm 2: *release()*

-

```
available = true
```

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Mutex

- Whenever mutex is implemented in a real-time embedded system, it adds an unbounded constraint
- The task could be scheduled and would have finished execution
- However, the task might have to wait until it can write/read data before finishing
- Duration of this wait can't be pre-determined and guaranteed
- While the task is waiting for a mutex, the time could have been used for some other task to execute
- On the other hand, allowing a task switch while it is waiting for mutex is a complex mechanism to put in place

Semaphore

- There is a library with three study rooms
- There is one librarian with a key for each study room
- As students arrive one after another asking for a study room, the librarian hands them on over three students
- Fourth student arrive asking for a study room, librarian has no more keys to give
- Fourth student has to wait until one of the students who have the study room returns the key
- Once he/they have, fourth student can enter a study room

Semaphore

- Semaphore is an integer value locking mechanism
- A task waits until semaphore integer value is greater than zero
- semaphore > 0 implies task can enter critical section, it reduces semaphore integer value by one

Algorithm 3: *wait*(S)

```
while S≤0 do
 |  busy wait
end
S−−
```

- Task exits critical section, increases semaphore value by one

Algorithm 4: *signal*(S)

```
S++
```

- Binary semaphore where semaphore can be either 0 or 1, it essentailly acts as a mutex

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Operating Systems Standards

- Standardization involves developing and applying technical standards
- They are usually done by involving firms, interest groups, etc. who are affected by it
- If you don't have standards everyone will do their own way
- If you end up changing your OS for some reason, you will have to re-do the whole development process
- It means understanding the philosophy of the design, how those ideas are implemented, then eventually how do you align your ideas to the new design to accomplish what you want, develop tests, verify, validate your product
- It is a solution to coordination problem

OS standard: POSIX

- POSIX: Portable Operating System Interface based on UNIX operating systems
- UNIX: a family of operating systems developed by AT&T in 1970s
- UNIX lead to development of a variety of OS, like Microsoft Xenix
- UNIX philosophy is says OSs should provide a set of simple tools, each performs a dedicated tasks
- Inter process communication should be done using pipes
- A shell scripting facilitates this execution and communication
- POSIX is a standard defined by IEEE to maintain compatibility between OSs

POSIX

- POSIX.1: Core services like Process creation, signals, pipes, C library, I/O ports
- POSIX.1b: Real-time extensions like priority scheduling, clocks and timers, semaphores
- POSIX.1c: Thread extensions like thread creation, thread scheduling, etc
- POSIX.1-2017: last version

RT-POSIX

- The real-time extension of POSIX
- Specifies set of system calls for real-time programming
- Services like mutex sync, wait and signal sync, data sharing by shared memory objects
- Includes FP preemptive scheduling, sporadic server scheduling, high resolution time management, etc
- There are four real-time profiles defined by POSIX.13:
  - Minimal Real-Time System profile: for small embedded systems
  - Real-Time Controller profile: Similar to previous plus file read, write, create; for robot controllers
  - Dedicated Real-Time System profile: for large embedded systems, extends previous with multiple process; eg.: avionics
  - Multi-Purpose Real-Time System profile: general purpose computing systems running real-time or non real-time requirements

OSEK

- OSEK: *Offene Systeme und deren Schnittstellen f-ur die Elektronik im Kraftfahrzeug* (Open systems and the corresponding interfaces for automotive electronics)
- To support efficient utilization of resources for automotive industries
- Intended for distribution control units in vehicles
- Its like an application program interface (API) for real-time OSs on network management system
- Typically for applications with tight real-time constraints, high criticality and large production volumes
- Strongly recommended for code optimization, reduce memory footprint

OSEK

- Scalability: Intended for a wide range of hardware platforms
- Software Portability: an ISO/ANSI-C interfance between application and OS is adopted
- Configurability: the designer should be able to tune the system
- Static allocation of software components: all kernel objects and application objects are statically allocated
- Kernal awareness: To allow debugging, task run tracing, real time values of variables, etc
- Support from Time Triggered Architectures: provides specification for OSEKTime OS, time trigerred OS
- AUTOSAR (AUTomotive Open System ARchitecture): It is an automotive standard derived from OSEK,

ARINC

- ARINC 653 (Avionics Application Standard Software Interface) is software sepcification for avionic software
- Specifies how to host multiple applications on same hardware
- ARINC 653 defines an API called APEX (APplication/EXecutive)
- APEX allows analyzable safety critical real-time applications implementation
- Traditionally, avionics computer systems had separate functions allocated to dedicated computing blocks
- ARINC has pushed towards Integrated Modular Avionics to save physical resources

ARINC IMC

- Physical memory is subdivided into partitions, and software sub-systems occupy distinct partitions at run-time
- An off-line cyclic schedule is used to schedule partitions
- Each partition is temporally isolated from the others and cannot consume more processing time than that allocated to it in the cyclic schedule
- Each partition contains one or more application processes, having attributes such as period, time capacity, priority, and running state
- Processes within a partition are scheduled on a fixed priority basis.

RT-OS

- An OS which allows real-time scheduling of functions on a hardware
- It needs to have a scheduler (usually hard), and facilities to execute threads, communicate data, memory management, etc.
- Some commercial RTOS:
  - VxWorks (Wind River)
  - OSE (OSE Systems)
  - Windows CE (Microsoft)
  - QNX
  - Integrity (Green Hills).
- Some Open source RTOS:
  - FreeRTOS
  - RTLinux
  - OSRTOS: others in the list: https://www.osrtos.com/

RTLinux

- RTLinux works as a small executive with a real-time scheduler
- It runs Linux as its lowest priority thread
- Linux thread is preemptable so that real-time threads
- When an interrupt is raised, the micro-kernel is interrupted
- If the interrupt is related to a real-time activity, a real-time thread is notified and the realtime kernel executes its own scheduler
- If the interrupt is not related to a real-time activity, then it is "flagged"
- When no real-time threads are active, Linux is resumed and executes its own code
- In this way, Linux is executed as a background activity in the real-time executive
- Scheduler is a simple FP scheduler and is POSIX compliant
- Lacks proper resource management, like a quick fix to the RT problem

FreeRTOS

- Open source RTOS for microcontrollers
- Very simple to work with, basically you requie 3 C files
- Overall small and light, very small memory footprint
- Has a range of supported devices
- For easy implementation, projects templates for the supported devices already exist and can be developed upon
- Have Windows, POSIX simulators to work and practice without having a proper microcontroller
- Developing FreeRTOS from scratch for a particular microcontroller is difficult

FreeRTOS

- FreeRTOS is configured by a header file called FreeRTOSConfig.h
- It is used to tailor the usage, eg: allow preemption or not *configUSE_PREEMPTION*
- Mainly you FreeRTOSConfig.h, tasks.c and list.c for any project
- You also have timers.c, event_groups.c and croutine.c
  - queue.c: provides queue and semaphore services
  - timer.c: provide timer functionality
  - event_group.c
  - croutine.c : for co-routine functionality
- You must include FreeRTOS.h followed by whatever you need, task.g, queue.h, etc.
- As soon as FreeRTOS launches there is an Idle task

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Semester 2

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Real-Time Embedded System analysis

- Verify mathematically
- Set theory, graph theory
- Theorems, proofs, etc
- Use logical reasoning and use mathematical approach

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Exercises

- Prove the Pythagorus theorem: $a^2 + b^2 = c^2$
- Find more than one ways

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Exercises

False or True, Prove:

- For a task set Γ with tasks τ_i and WCET C_i, it is sufficient that $\Sigma_i(C_i/T_i) < 1$ and the tasks are ordered in nondecreasing period T_i, that it is schedulable using Preemptive Fixed Priority scheduling.
- For a task set Γ with tasks τ_i and WCET C_i, it is sufficient that $\Sigma_i(C_i/T_i) < 1$ and the tasks are ordered in nondecreasing period T_i, that it is schedulable using Non-Preemptive Fixed Priority scheduling.
- For a task set Γ with tasks τ_i and WCET C_i, it is sufficient that $\Sigma_i(C_i/T_i) < 1$ and the tasks are ordered in nondecreasing period T_i, that it is schedulable using Preemptive Earliest Deadline First scheduling.

Schedulability: Formal Methods

- Formal methods are semantics with underlying rules which have provable properties
- Finite state machine:
  - Petri Net
  - Automata
  - Timed Automata
  - Stochastic Automata
  - Markov Chain

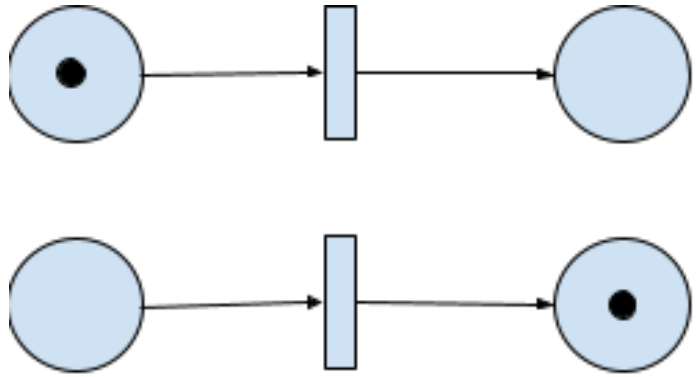
Schedulability: Petri Net

- Tuple $N = (S, T, W)$, S is the set of states, T is the set of transitions S and T are disjoint, W is the set of directed arcs
- Model of states and transition between those states
- Transitions are enable by a trigger
- States can contain tokens
- Initial state of the model constitutes a token in the first state
- Trigger can be triggered iff there is toke in the previous state
- Tokens move around in the system

Similar modelling methods exist with Automata and its variants

Petri Net

Token from transition



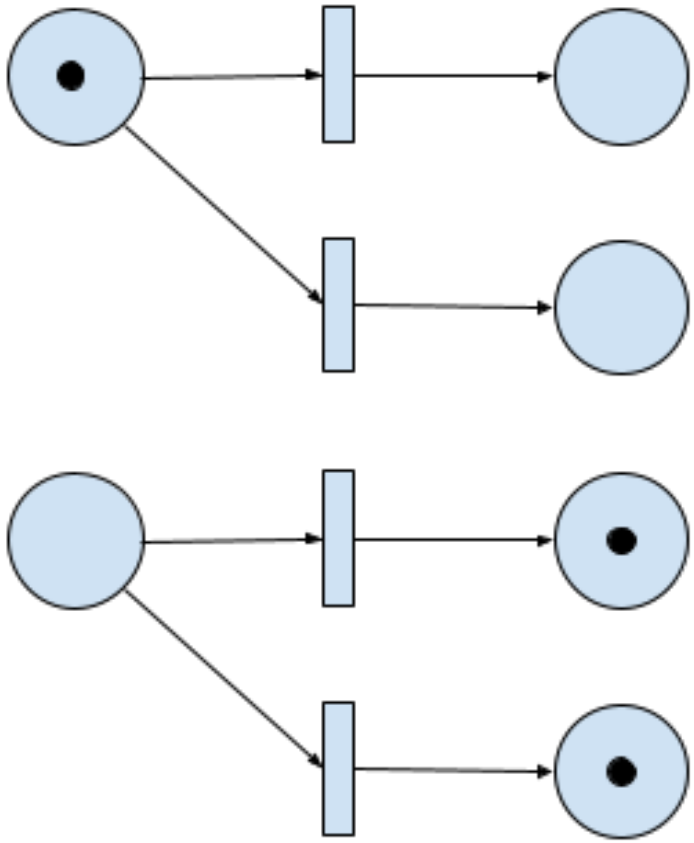
Petri Net

Prepare a Petri Net to represent light switch.

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Petri Net

Two transition for shared token space

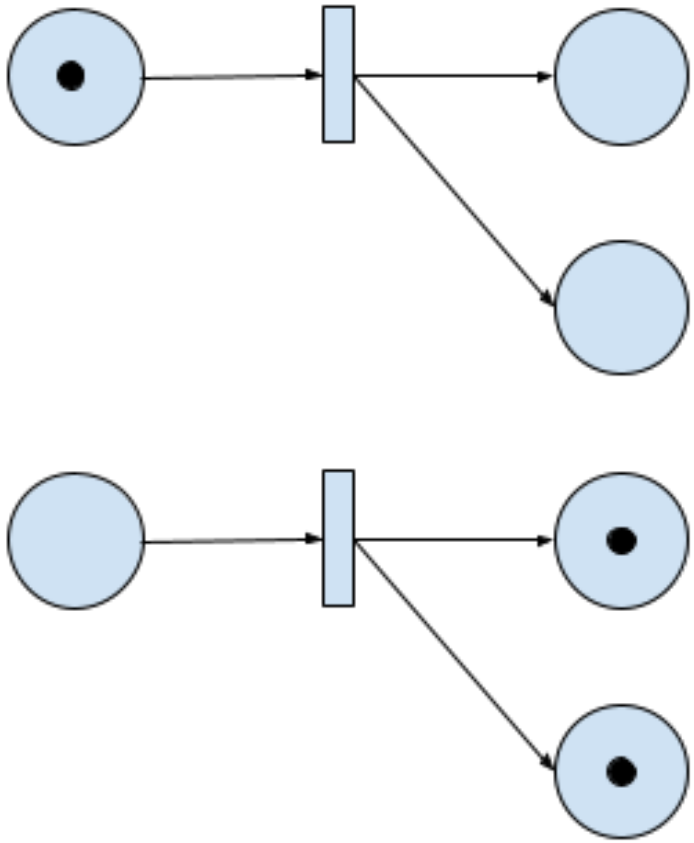


Petri Net

Prepare a Petri Net to represent two switches for one light, in the sense that if one switch is turned on, the light switches on, then if the other switch is turned on, the light switches off.

Petri Net

Two token space for one transition

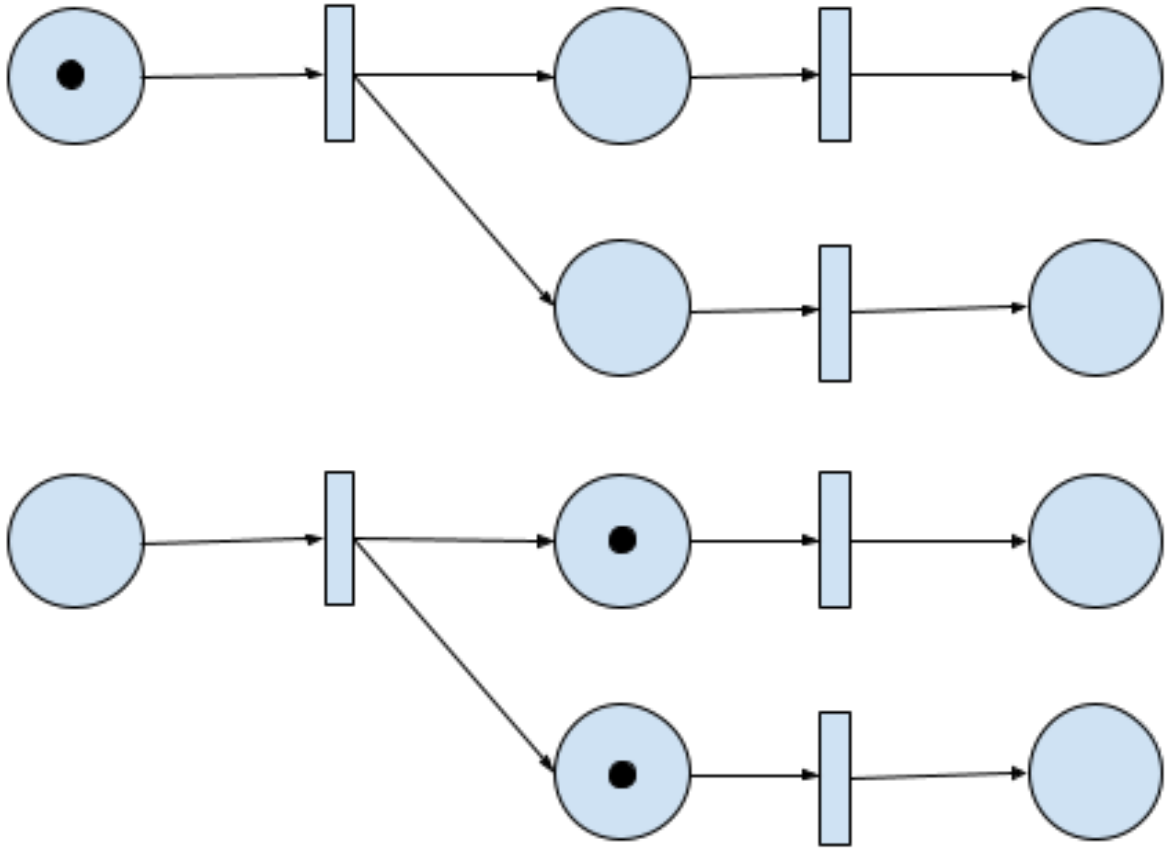


Petri Net

- Prepare a Petri Net to represent two switches for one light, in the sense that if one switch is turned on, the light switches on, then if the other switch is turned on, the light switches off.
- Prepare a Petri Net to represent one switch for one light, but two different switches to turn them off individually.
- Prepare a Petri Net to represent one switches for one light, which only on when two switches are on. Once the two switches are on AND the light is on, a fan and the television has the possibility to be switched on. Implement a kill switch the close everything.

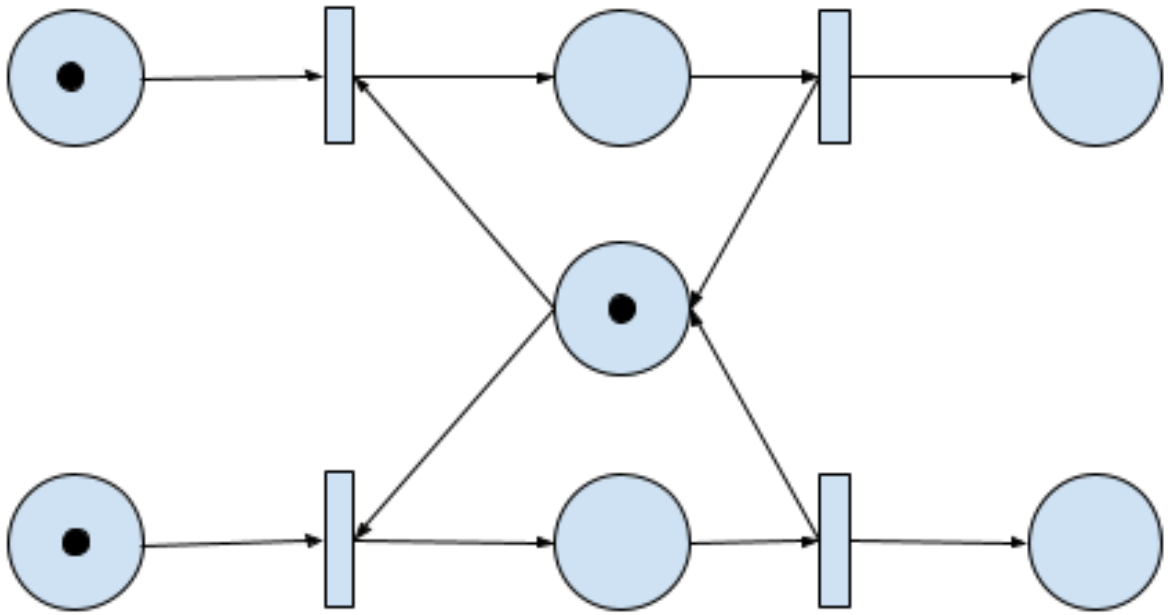
Petri Net

Two sequences can occur in parallel



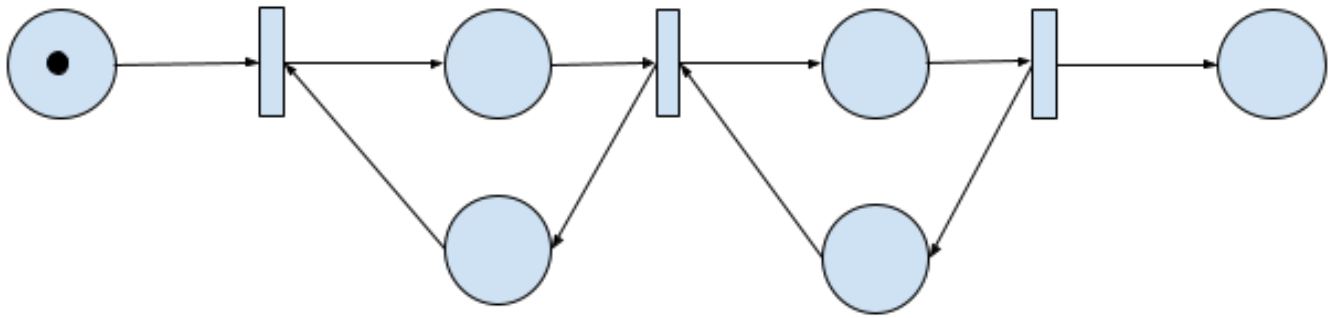
Petri Net

Resource sharing



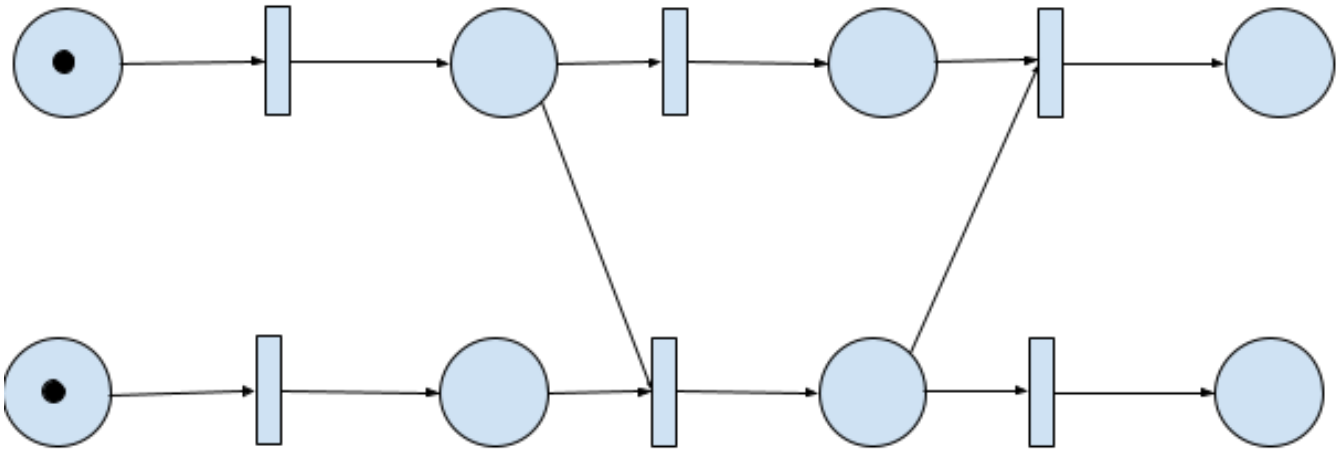
Petri Net

Buffer, queue



Petri Net

Communication between two petri nets



Petri Net

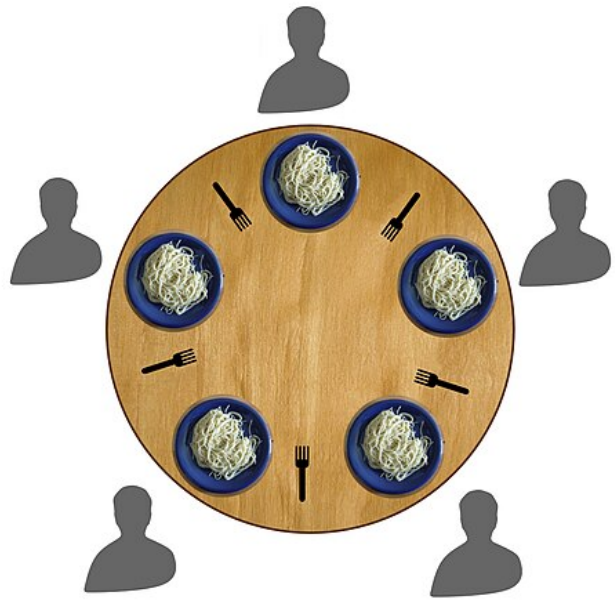
Prepare a Petri Net to represent one switches for one light, which only on when two switches are on. Once the two switches are on AND the light is on, a fan and the television has the possibility to be switched on. Implement a kill switch the close everything.

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Dining Philosophers problem

- Five philosophers sit a table to eat, there are five plates and five forks.
- Each philosopher can only alternately think and eat.
- a philosopher can only eat their spaghetti when they have both a left and right fork.

Two forks will only be available when their two nearest neighbors are thinking, not eating. After an individual philosopher finishes



eating, they will put down both forks.

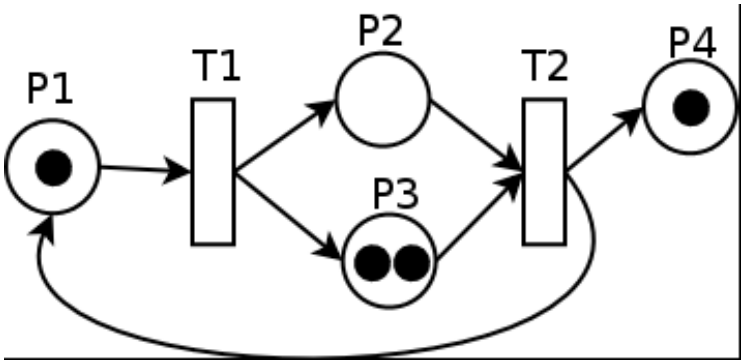
Dining Philosophers problem

The problem is how to design a regimen policy such that any philosopher will not starve.
Each philosopher can forever continue to alternate between eating and thinking, assuming that no philosopher can know when others may want to eat or think

Schedulability: Petri Net

- Provable properties:
  - Deadlock: None of the states lead to deadlock
  - Aliveness: A state is reached at least once
  - Reachability: A state is reachable
  - Boundedness: number of tokens is bounded

Similar modelling methods exist with Automata and its variants



Peteri Nets

Test your previous light bulb implementations for the properties: Liveness, Reversibility and Boundedness.

Petri Net

Ensure schedulability of the following task set using Non-Preemptive FP. Prepare a Petri Net to represent task execution of the following task set.

Task	C_i	T_i
τ_1	2	10
τ_2	2	15

Note that, alternate Task τ_1 execution can't execute before τ_2.

- Is this task set schedulable ?

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Petri Net

Ensure schedulability of the following task set using Non-Preemptive EDF . Prepare a Petri Net to represent task execution of the following task set.

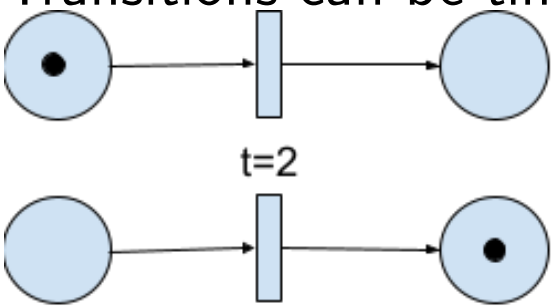
Task	C_i	T_i
τ_1	2	10
τ_2	3	15
τ_3	1	30

Note that, Task τ_2 can't execute before τ_3. Also, τ_1 leaves a data value to be read by τ_2 so that τ_2 can execute.

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

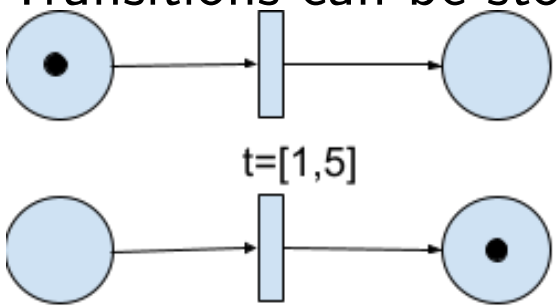
Petri Net

Transitions can be timed



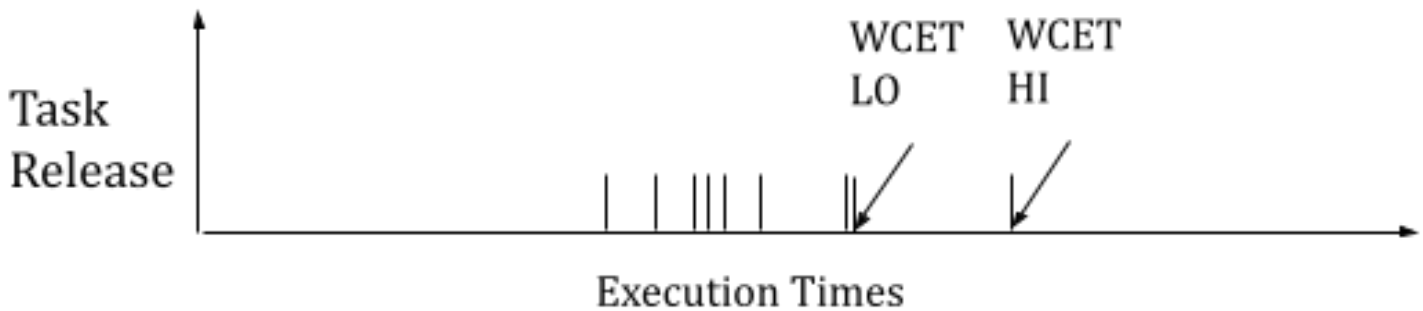
Petri Net

Transitions can be stochastic



Mixed Criticality

- Two types of tasks *HI* and *LO* criticality tasks
- *LO* criticality tasks are 'normal' tasks
- *HI* criticality tasks have two WCETs C_i^{HI} and C_i^{LO}, ($C_i^{LO} < C_i^{HI}$)
- Real Time system has two modes of functioning *LO* and *HI*



IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Schedulability of Mixed Criticality

- While the system executes, the tasks are scheduled using the LO WCET for all tasks
- If one task exceeds C_i^{LO}, the system switches to HI criticality mode
- Then, all the LO criticality tasks are dropped from execution
- The time slots gained from dropping all the LO tasks are given to HI criticality tasks
- HI criticality tasks are scheduled using C_i^{HI}
- System can switch back to LO criticality mode using certain condition like after a certain time (or other more robust condition)

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Schedulability of Mixed Criticality

Is the following task set schedulable under non-preemptive fixed priority?

Task	C	T_i
τ_1	7	10
τ_2	7	15

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Schedulability of Mixed Criticality

The previous task set is modified using the notion of Mixed Criticality, ensure schedulability in *HI* and *LO* system criticality modes of the following mod task set using Non-Preemptive Fixed Priority.

Task	Criticality	C^{LO}	C^{HI}	T_i
τ_1	*HI*	5	7	10
τ_2	*LO*	7	NA	15

Schedulability of Mixed Criticality

Ensure schedulability in *HI* and *LO* system criticality modes of the following task set using Non-Preemptive Fixed Priority.

Task	Criticality	C^{LO}	C^{HI}	T_i
τ_1	*HI*	5	7	10
τ_2	*HI*	1	3	10
τ_3	*LO*	2	NA	15
τ_4	*LO*	3	NA	30

Schedulability of Mixed Criticality

Ensure schedulability in *HI* and *LO* system criticality modes of the following task set using Non-Preemptive Fixed Priority.

Task	Criticality	C^{LO}	C^{HI}	T_i
τ_1	*HI*	5	7	10
τ_2	*HI*	1	2	10
τ_3	*HI*	1	2	15
τ_4	*LO*	3	NA	30

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

Exercise

- Implement a code in C to do a multiplication of 10 random numbers which you generate. After the executable is ready, time the execution enough times. Sample the execution times in intervals of your preference. Based on this sampling, determine the C_i^{HI} and C_i^{LO}.
- Implement another C code to calculate *log* of multiplication of two random numbers, determine their WCET, treat this task as low-criticality task
- You only have 20ms for the first task, and 10ms for the second, is the task set schedulable? If not, propose and argue different time period values.