Parallel & Concurrent Programming

DD1396
Ric Glassey
glassey@kth.se

DD1396 new students

Agenda

Course information

Motivation

Paradigms

Complexities

Course Information

Topics

Short course, but we cover:

- Intro to **Concurrent** and **Parallel** Programming
- Intro to **Go** Programming Language
- Differences between concurrency and parallelism
- Complexities arising (**Data Races** and **Deadlocks**)
- Practical solutions (**Communication**, **Channels**, **Locks**)
- Concurrency Patterns for scalability (e.g. **Map reduce**)

Canvas

Lecture content

- Slides
- Videos (2021 versions already posted)
- Stream (and clips posted within 24hrs)

Try to avoid contacting me via Canvas (use email)

- It hides your KTH id
- KTH id is how we find your Github work
- Involves a round trip to ask, or a lookup

Assignments

3 Assignments

Distributed as KTH Github Repo

All work must be committed and pushed or it will not be graded

Please consult “Using Github” resources if this is new / unfamiliar

Demo: KTH GitHub Repo

Assessment

Pass/Fail assignments

Must pass all three to pass course

You must:

- Attempt all assignments
- Be prepared to explain all assignments
- Do your own work

Groups

DD1338 Students

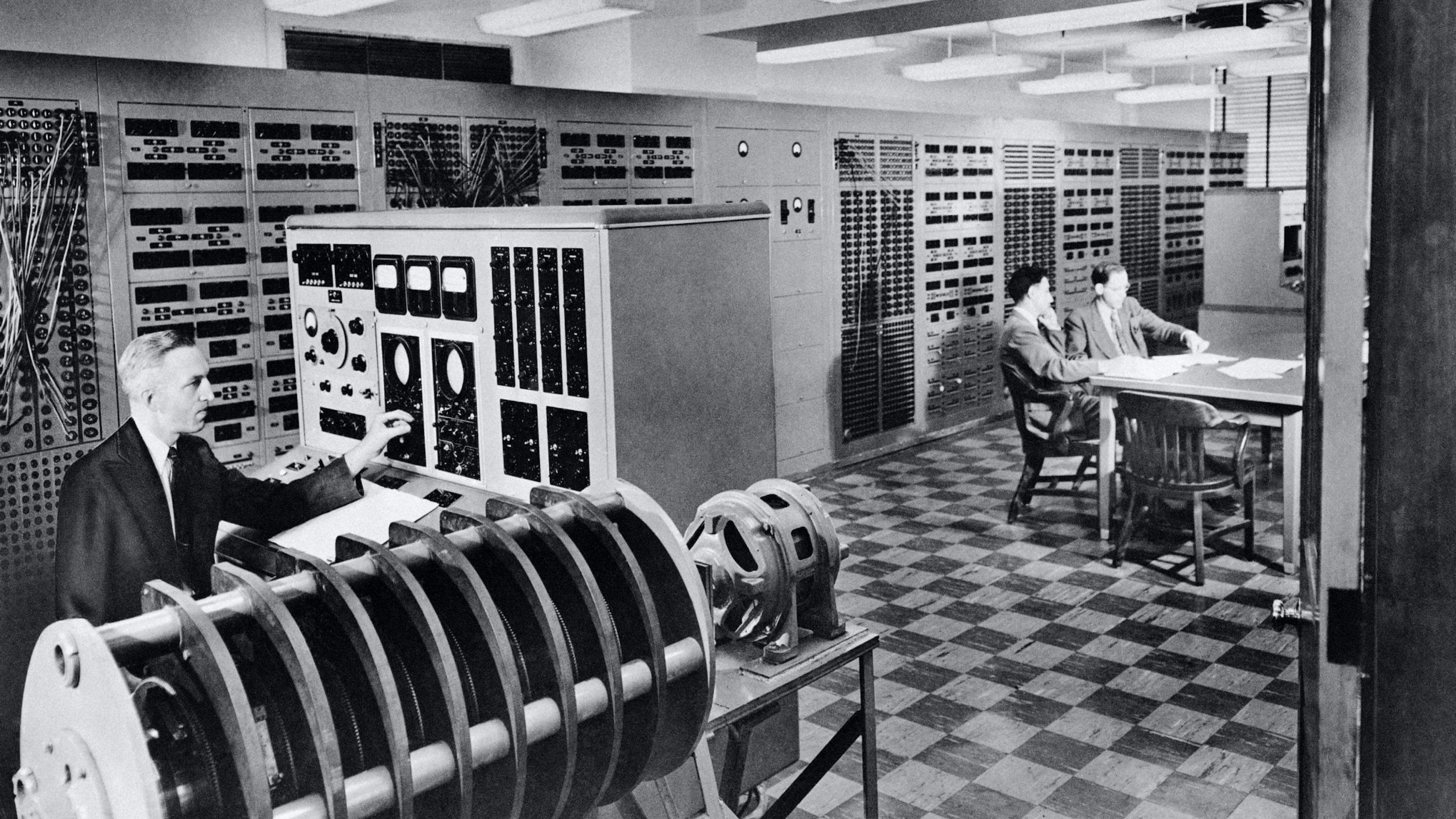
- Remain in the same group

Incoming students

- Will be allocated a TA
- TA will contact you before first övning

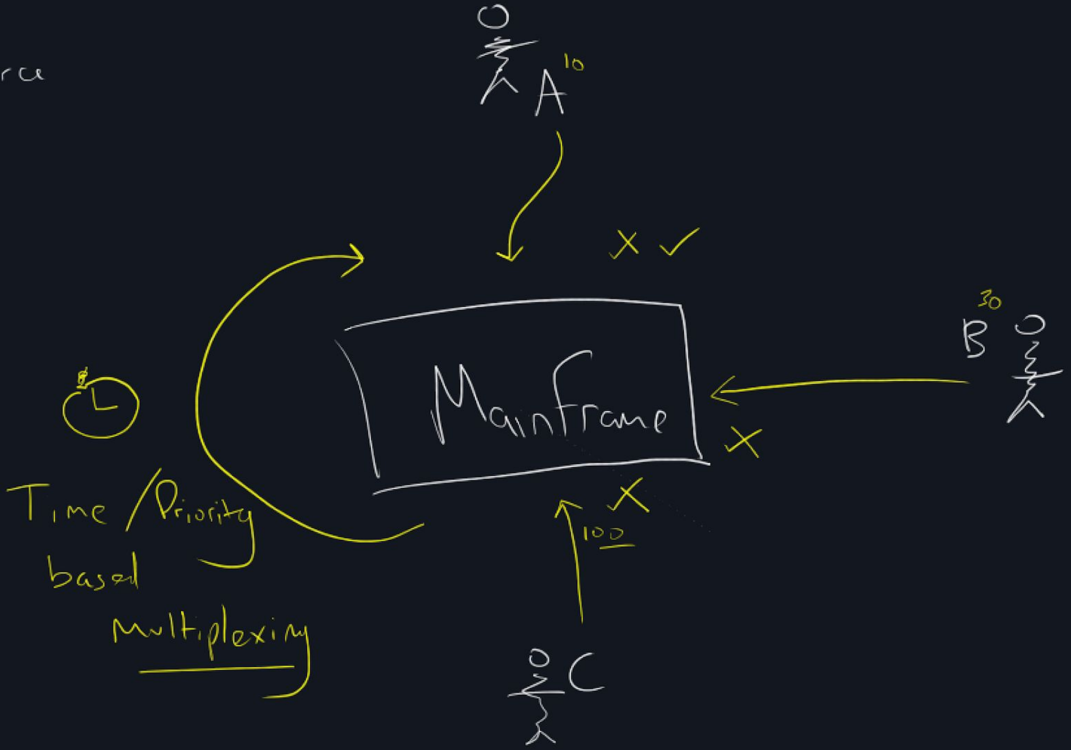
Övningar will be on campus

Motivation

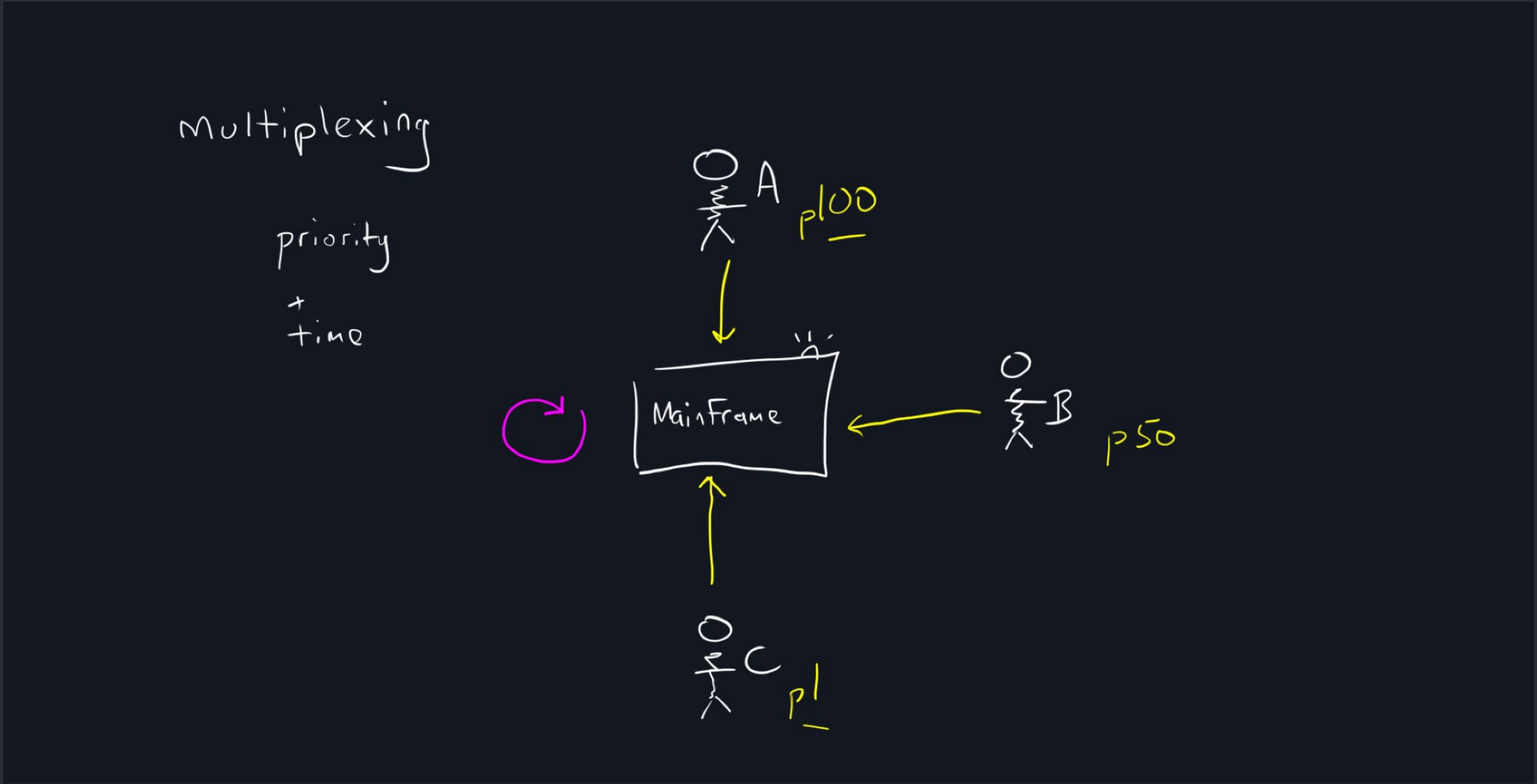


Draw: Resource Sharing

Limited Resource

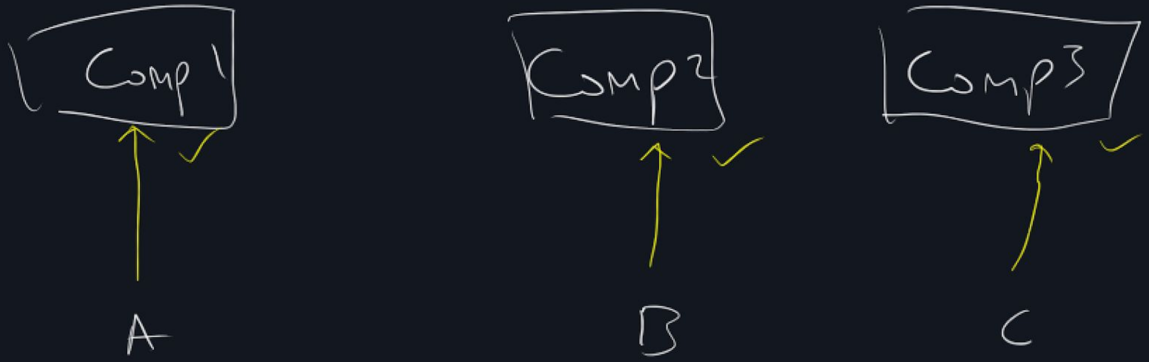


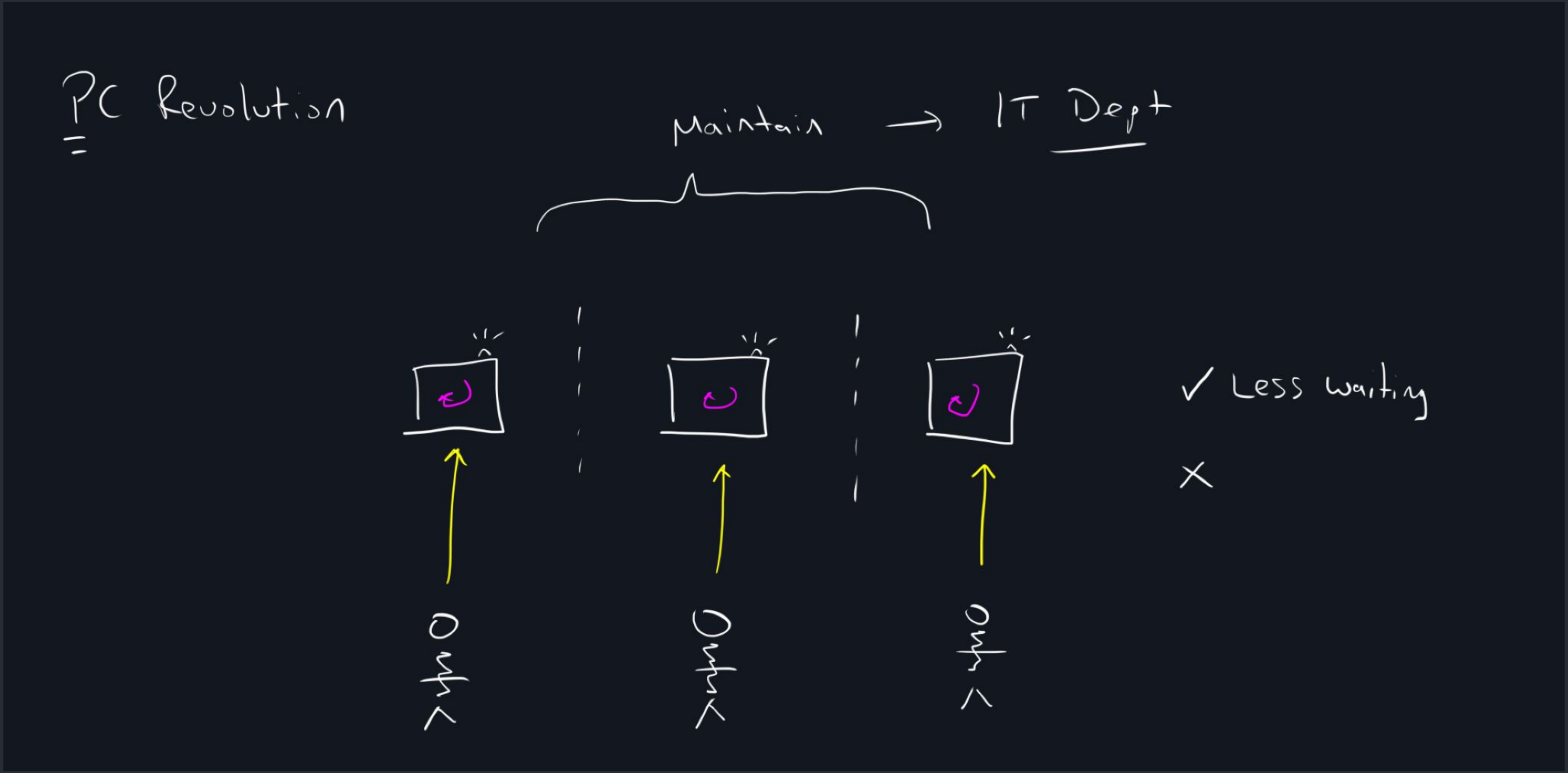
Time / Priority based Multiplexing



Video

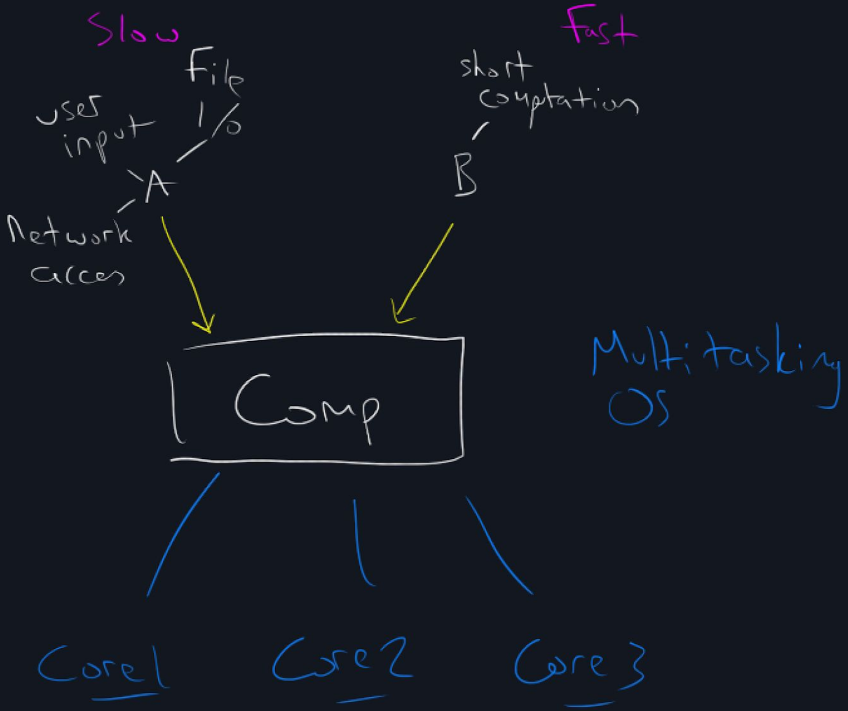
More Resources

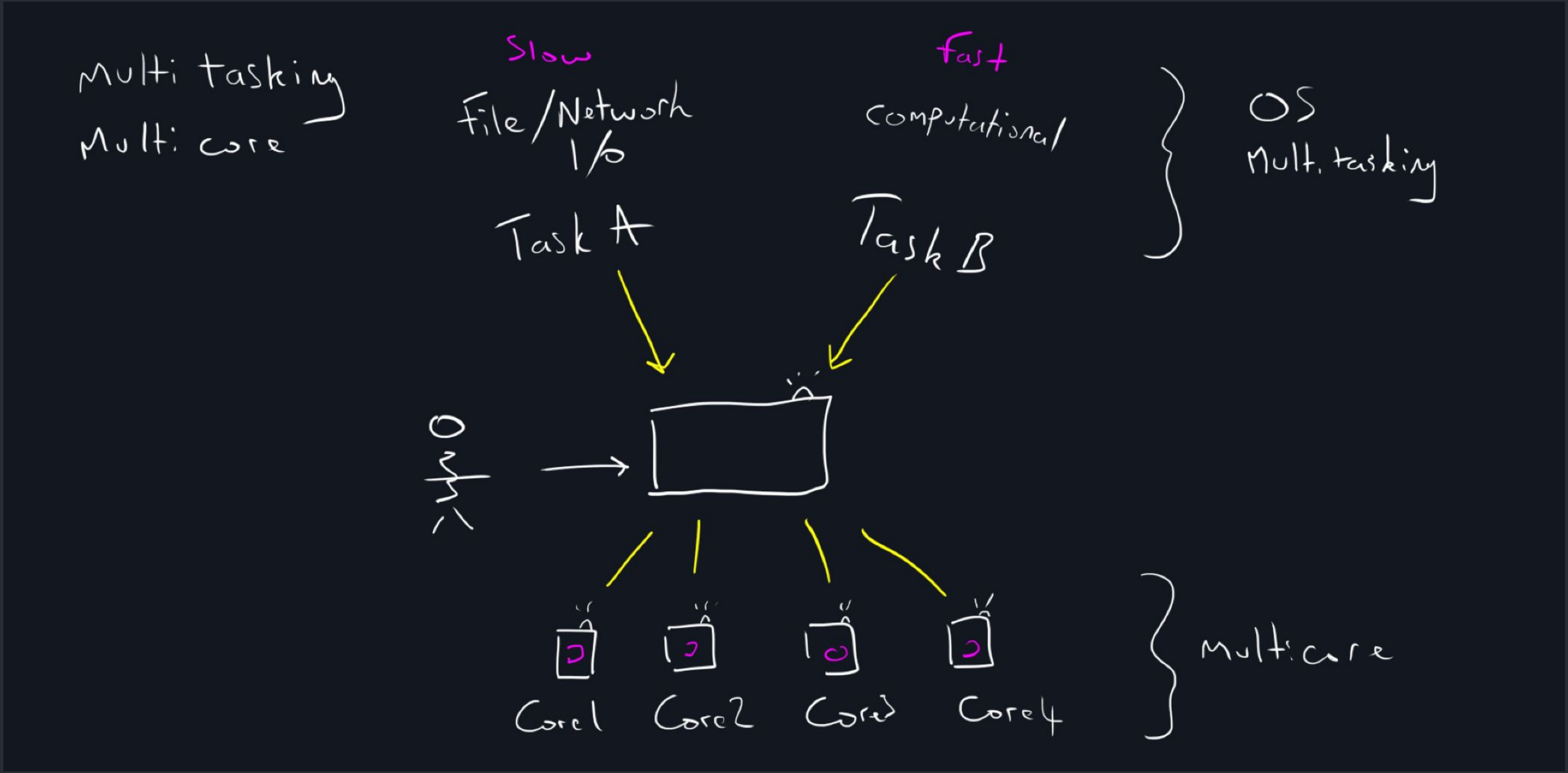




Video

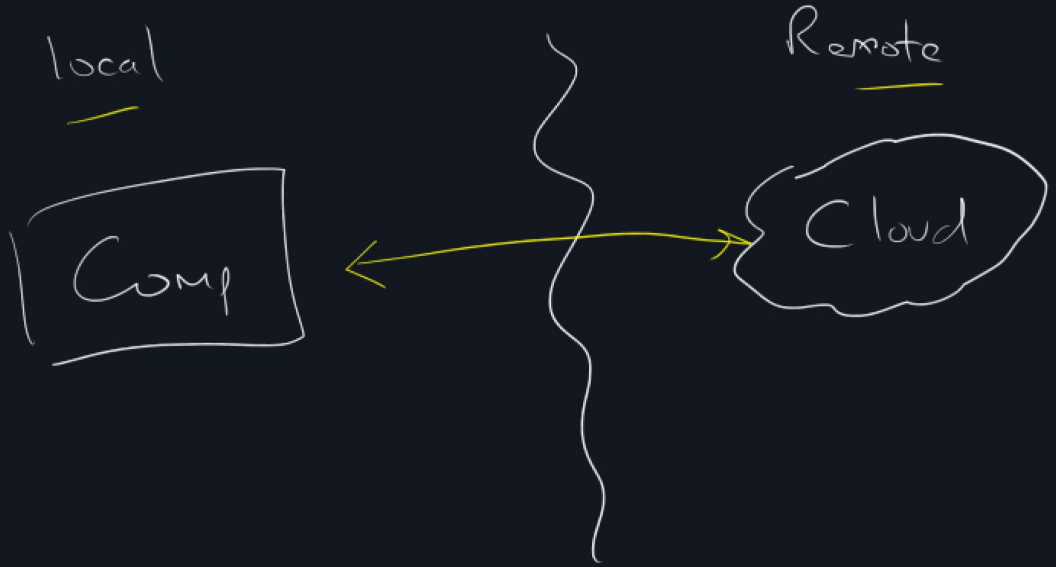
Going Deeper

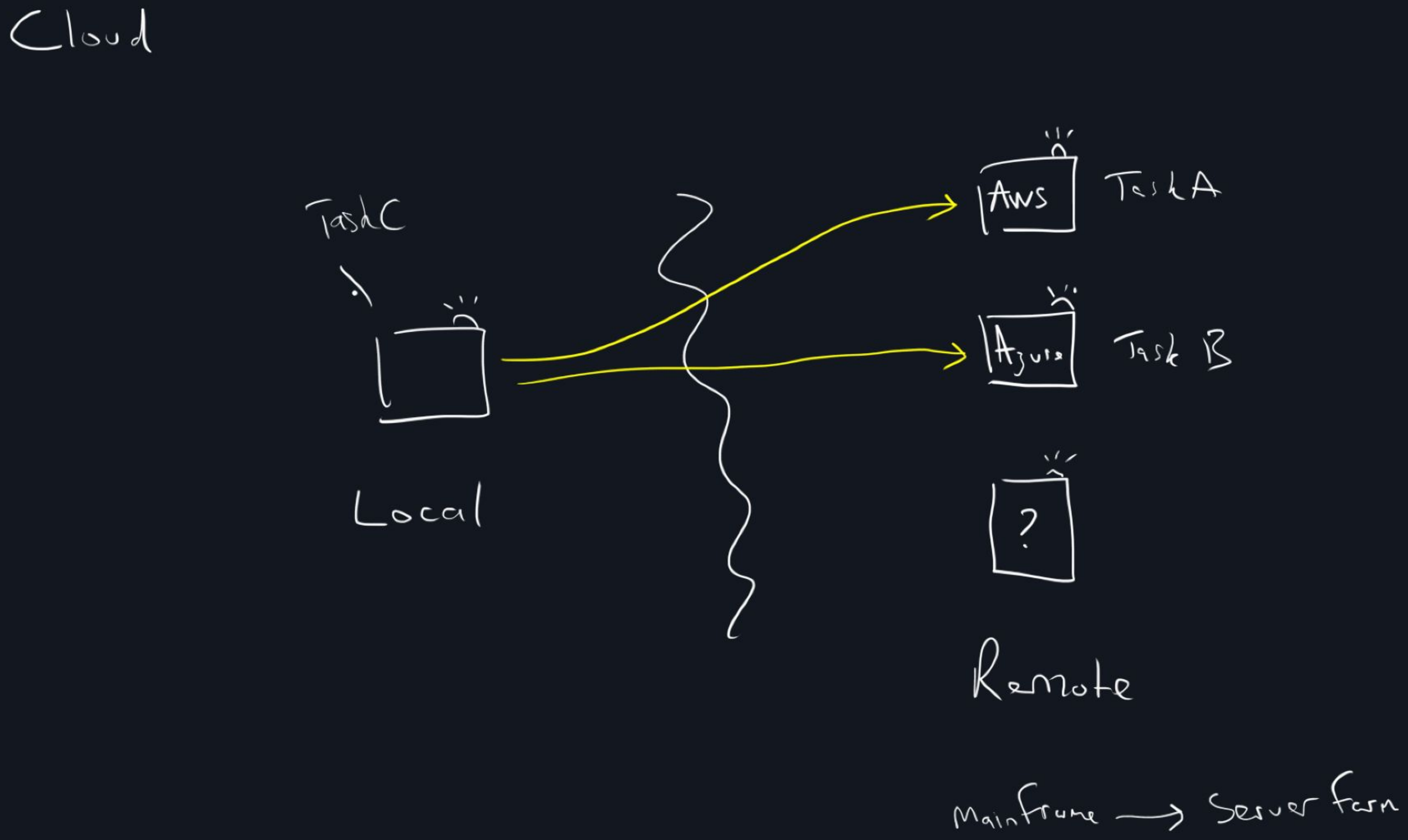



Multi tasking
Multi core
Slow
Fast
File/Network
I/O
Computational
OS
Multi tasking
Task A
Task B
Core1
Core2
Core3
Core4
multicore

Video

Mix




Cloud
TaskC
Local
Aws
Task A
Azure
Task B
?
Remote
Mainframe → Server farm

Video

Scarcity to abundance

Availability of resources has exploded in terms of:

- Number of devices
- Number of processors per device
- Number of interconnected devices

Multiplexing mainframe and independent computers are metaphors for:

- Concurrency
- Parallel programming

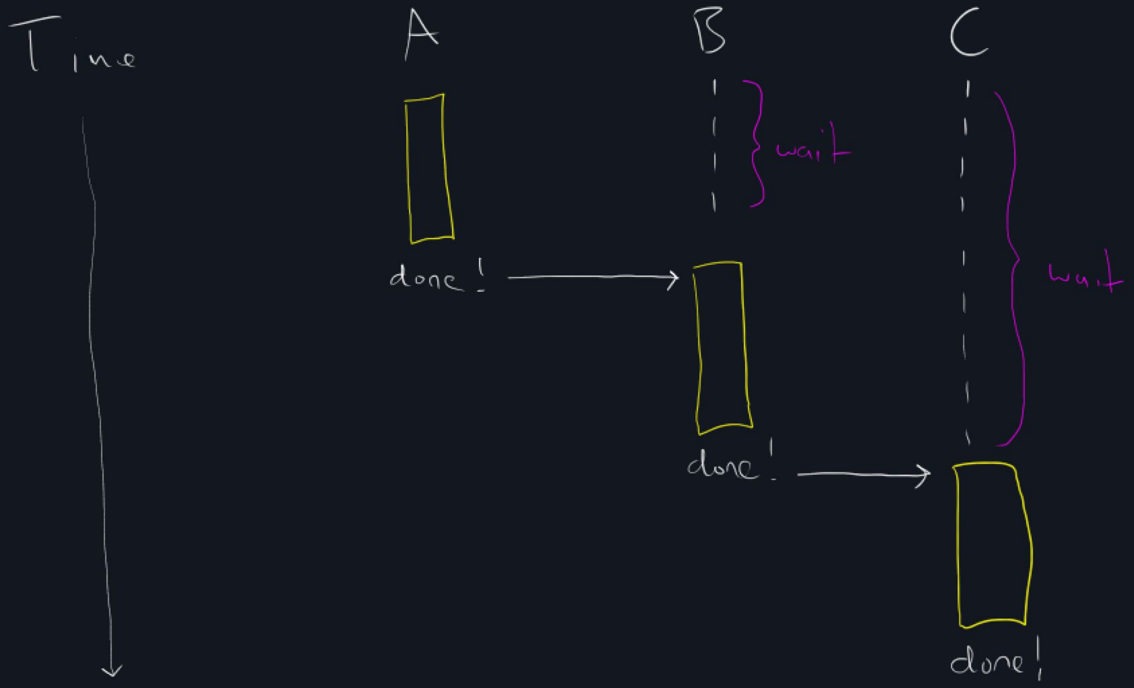
But we mostly develop software in a sequential, imperative style

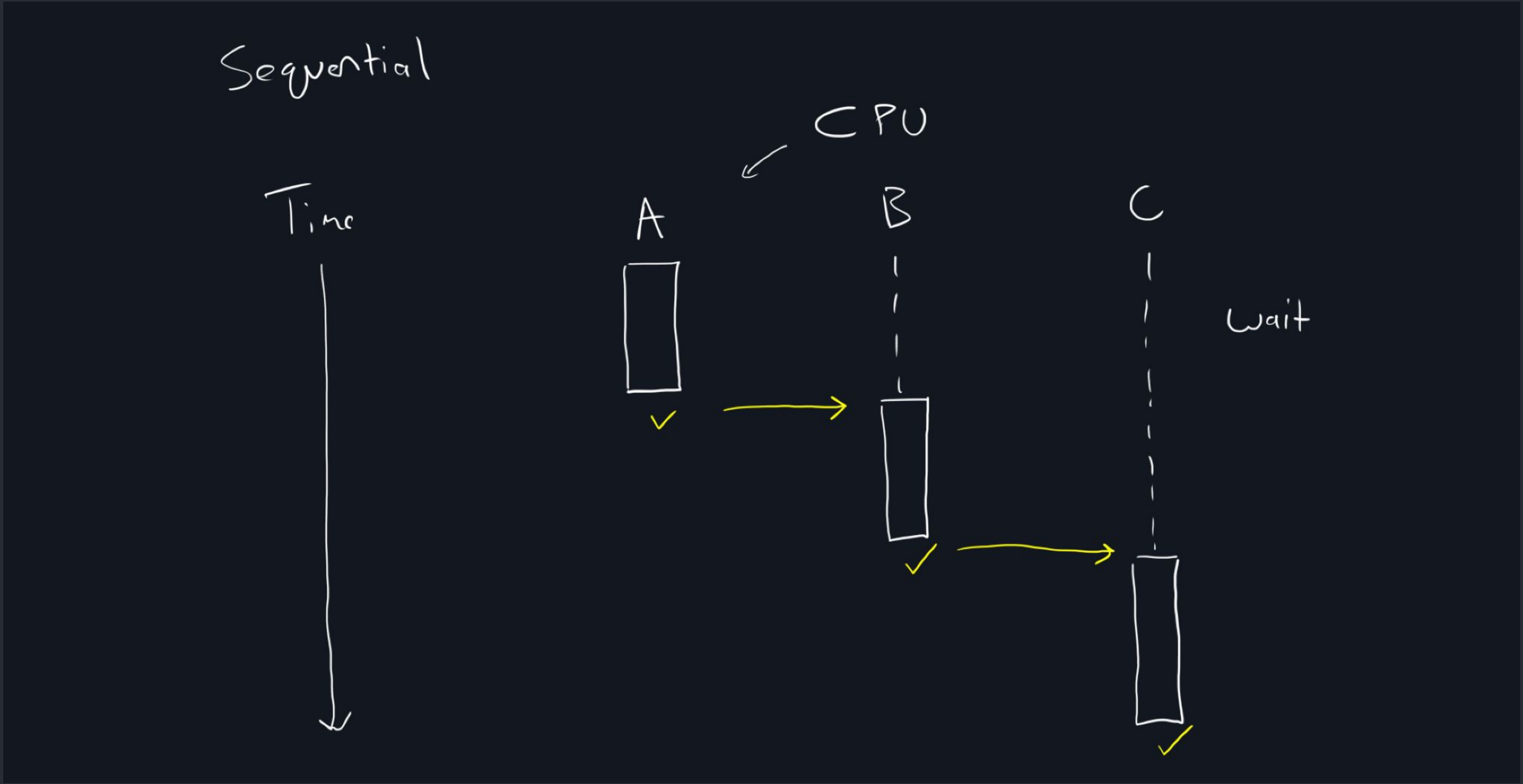
Paradigms

Draw: Sequential, Parallel & Concurrent

Sequential

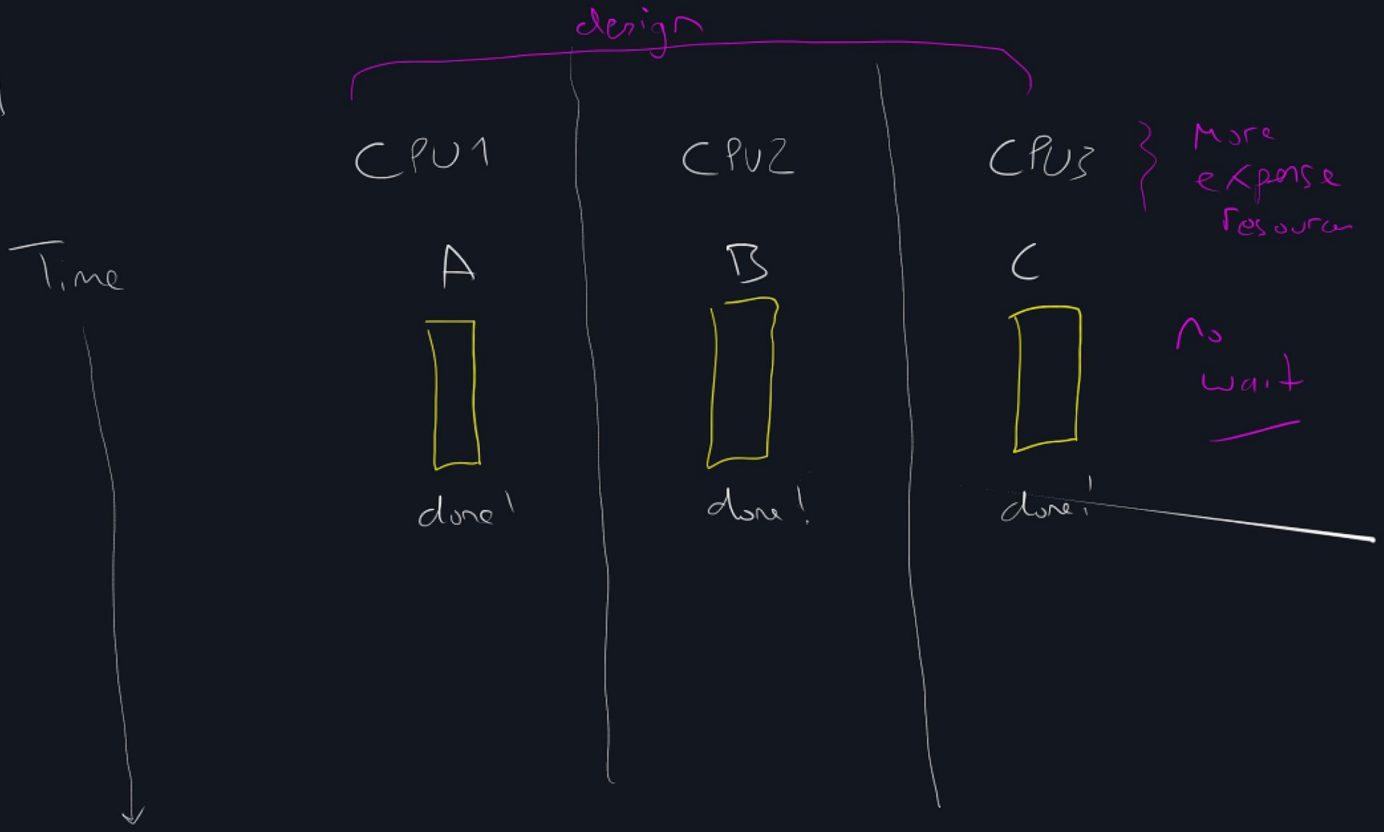
x1 CPU

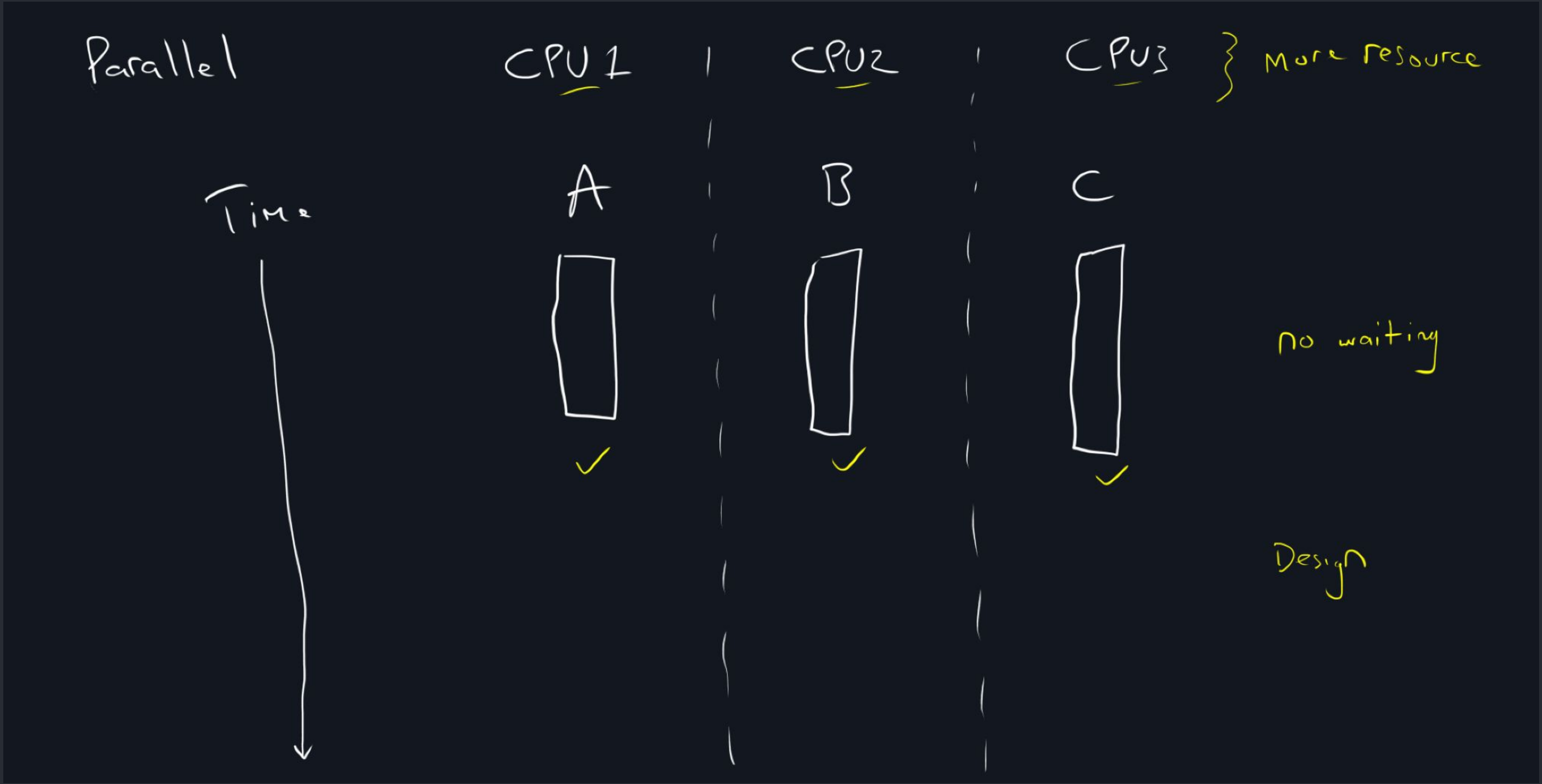




Video

Parallel

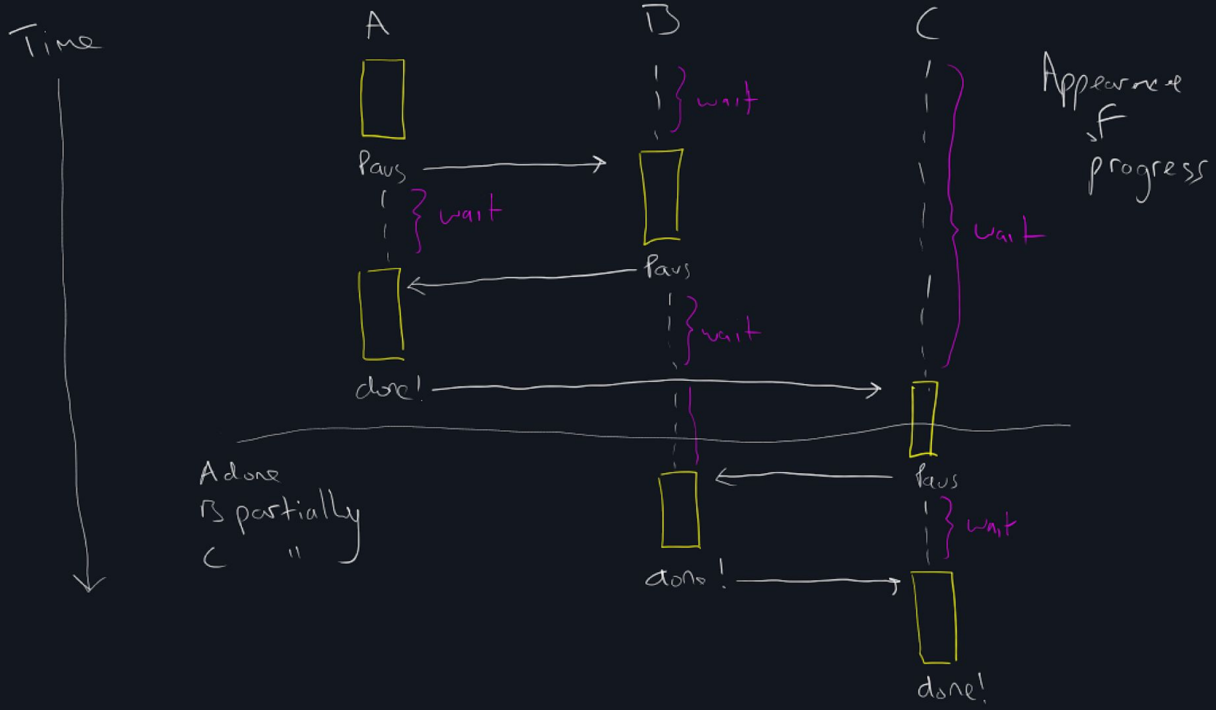

design
CPU1
CPU2
CPU3
More expense resource
Time
A
B
C
no wait
done!
done!
done!


Parallel
CPU 1
CPU2
CPU3
More resource
Time
A
B
C
✓
✓
✓
no waiting
Design

Video

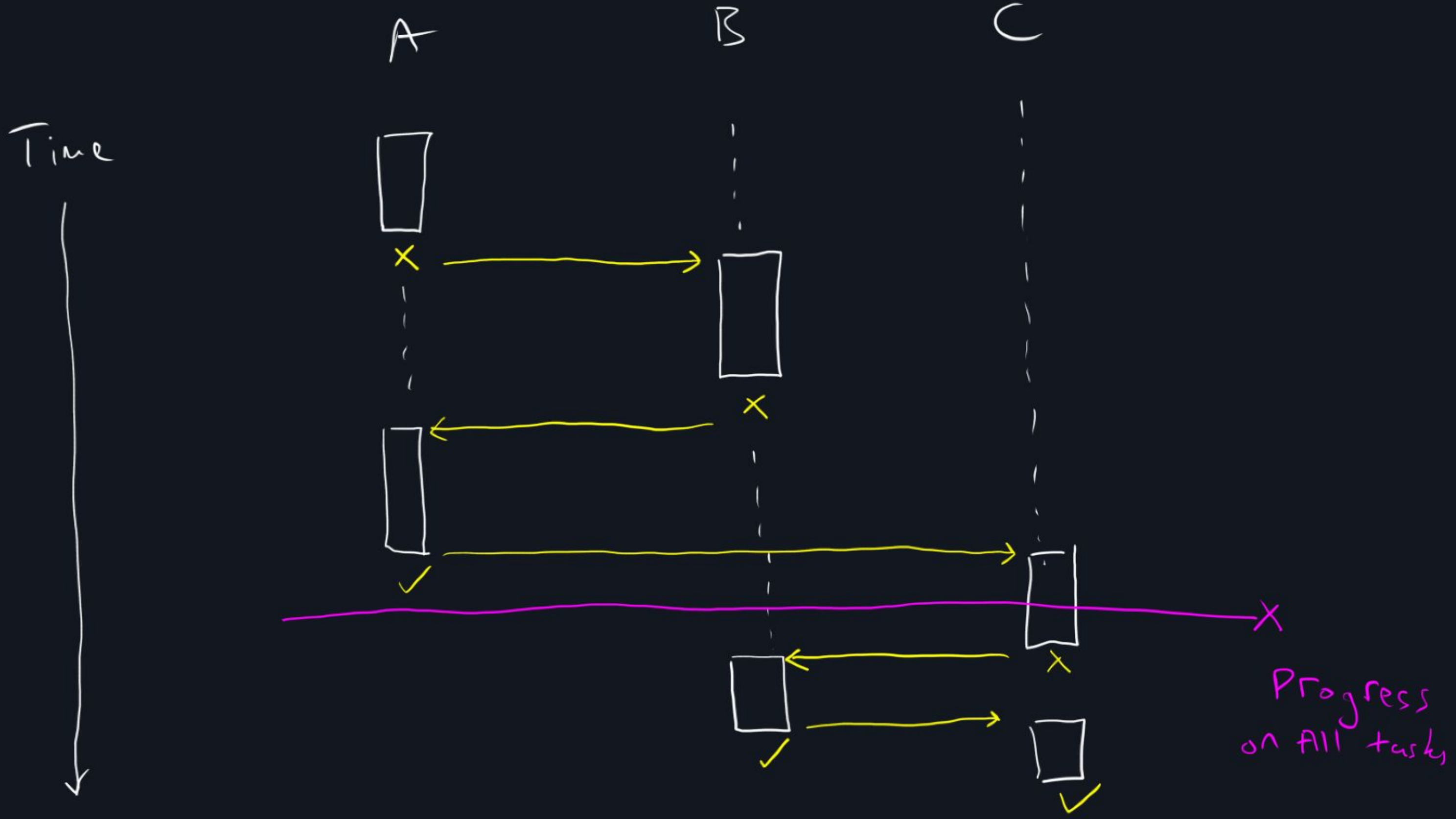
Concurrent

×1 CPU (multiple CPUs)



Concurrent

1 CPU (More CPUs)



Video

Concurrency vs parallelism

A **parallel program** will use a **multiplicity of independent processing units** to accomplish **one or more tasks**

A **concurrent program** is one structured with **multiple threads of control**, which give the **impression of simultaneous execution**

“Difference between **dealing with** and **doing multiple things** at the same time” - Rob Pike (co-inventor of Go)

Complexities

Increased design effort

What is the appropriate level of **division of task**?

What **inter-task communication** is required?

How can **task correctness** be maintained?

How can **multiple processing units** be exploited?

No longer any **guarantees of order**

Non-determinism is the norm

Increased debugging effort

Execution is potentially non-deterministic

- Can go wrong in **multiple places**
- Can go wrong at **different times**

No two runs will give the **same sequence** that lead to a bug

New area **for bugs to hide** and go unnoticed

Nasty side effects

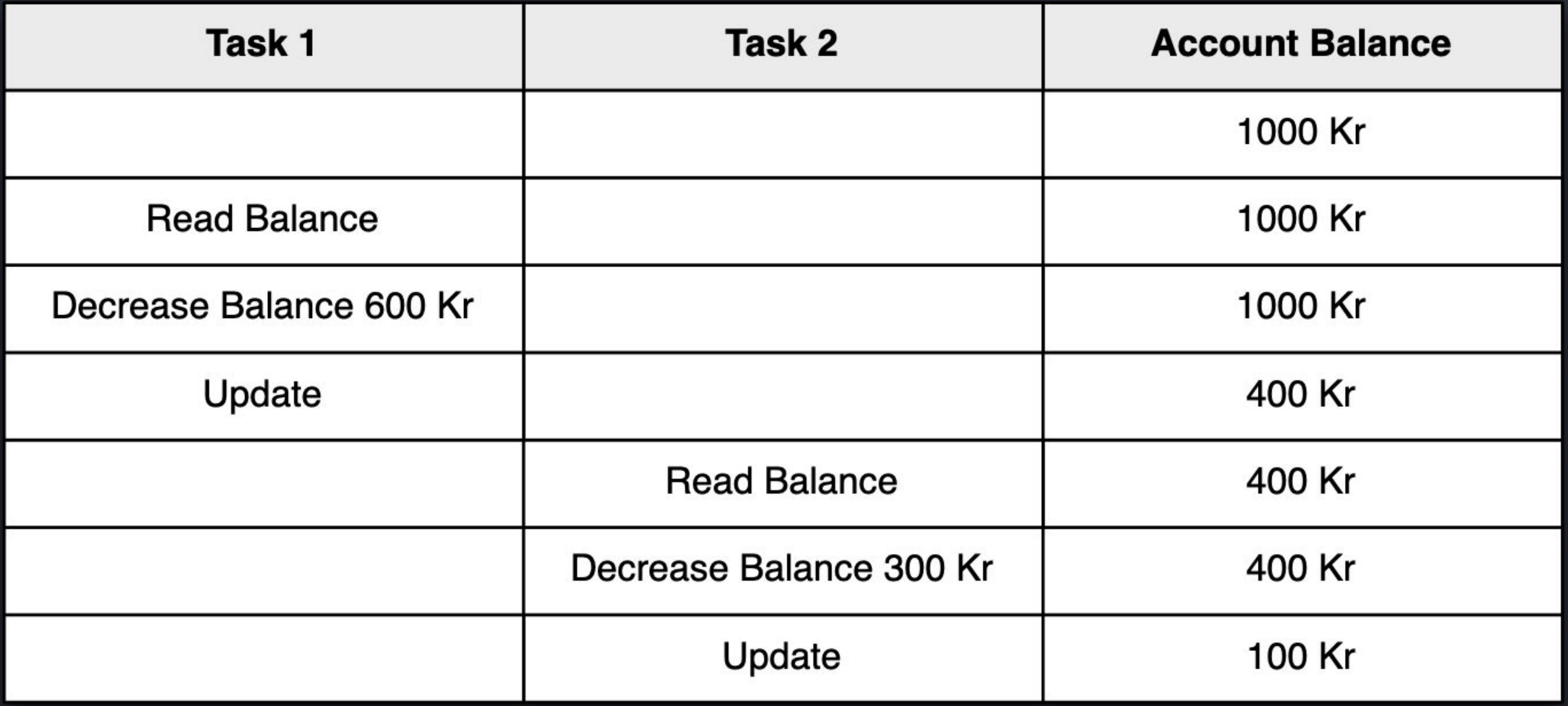
Even if programs appear bug-free:

- No compilation errors
- No logical errors
- No runtime errors

Can have **concurrency issues:**

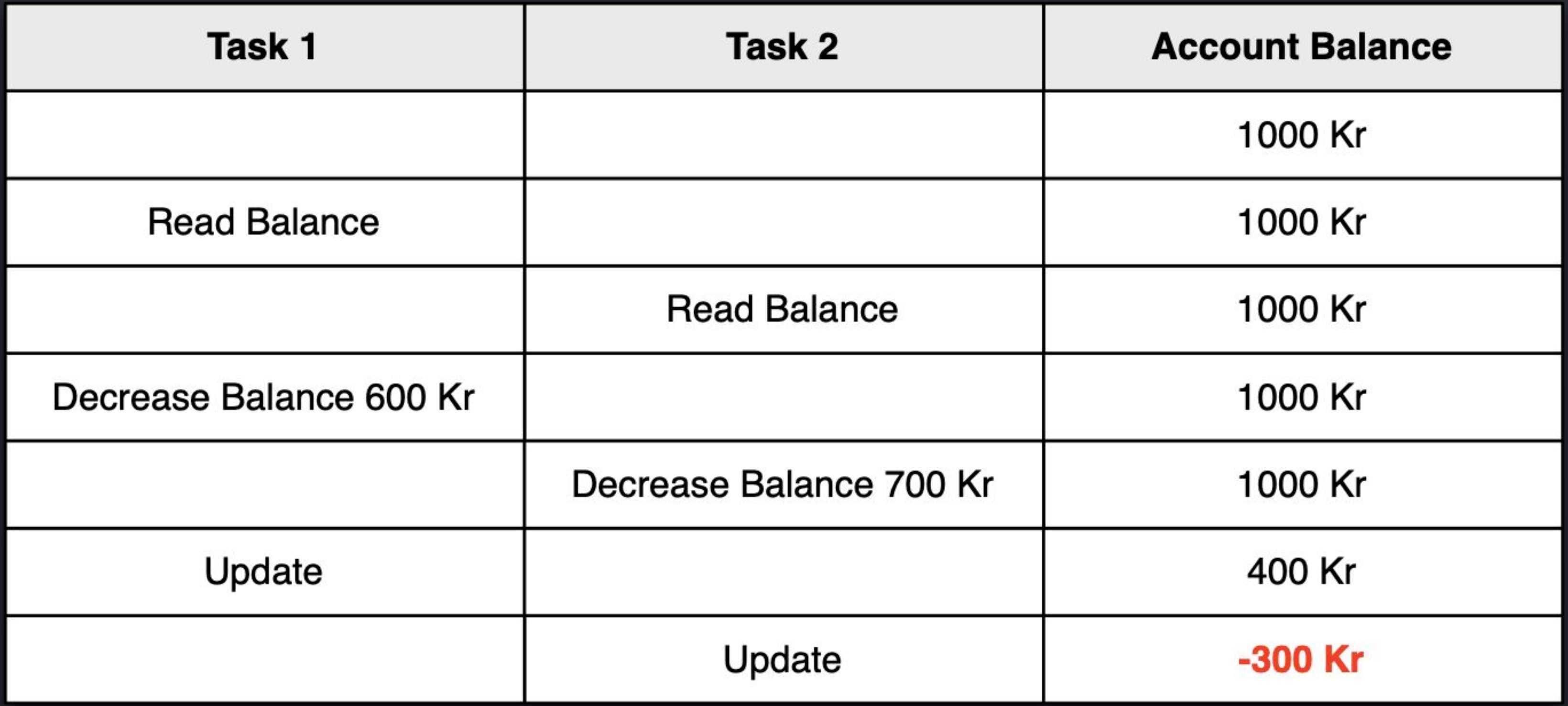
- Race Conditions
- Deadlock
- Resource Starvation

Bug #1: Transactions in sequence



Task 1	Task 2	Account Balance
		1000 Kr
Read Balance		1000 Kr
Decrease Balance 600 Kr		1000 Kr
Update		400 Kr
	Read Balance	400 Kr
	Decrease Balance 300 Kr	400 Kr
	Update	100 Kr

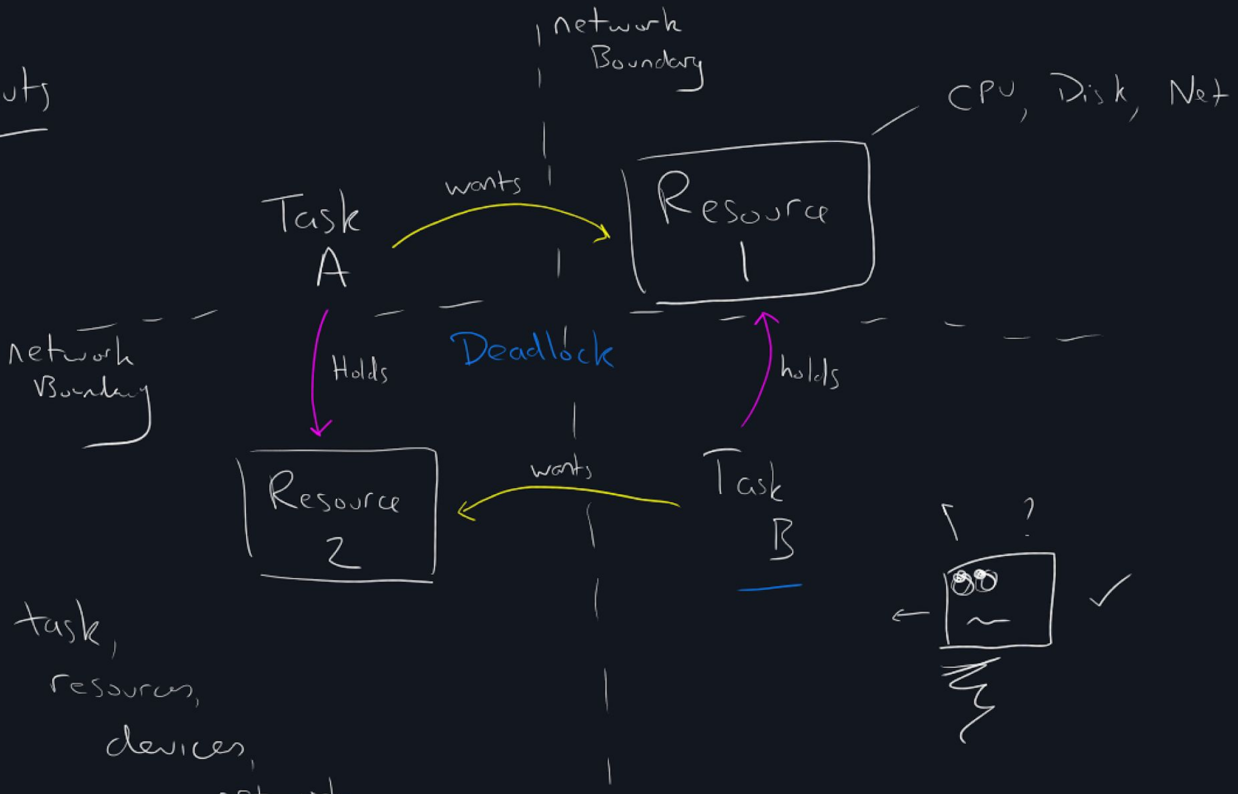
Bug #1: Transactions in parallel



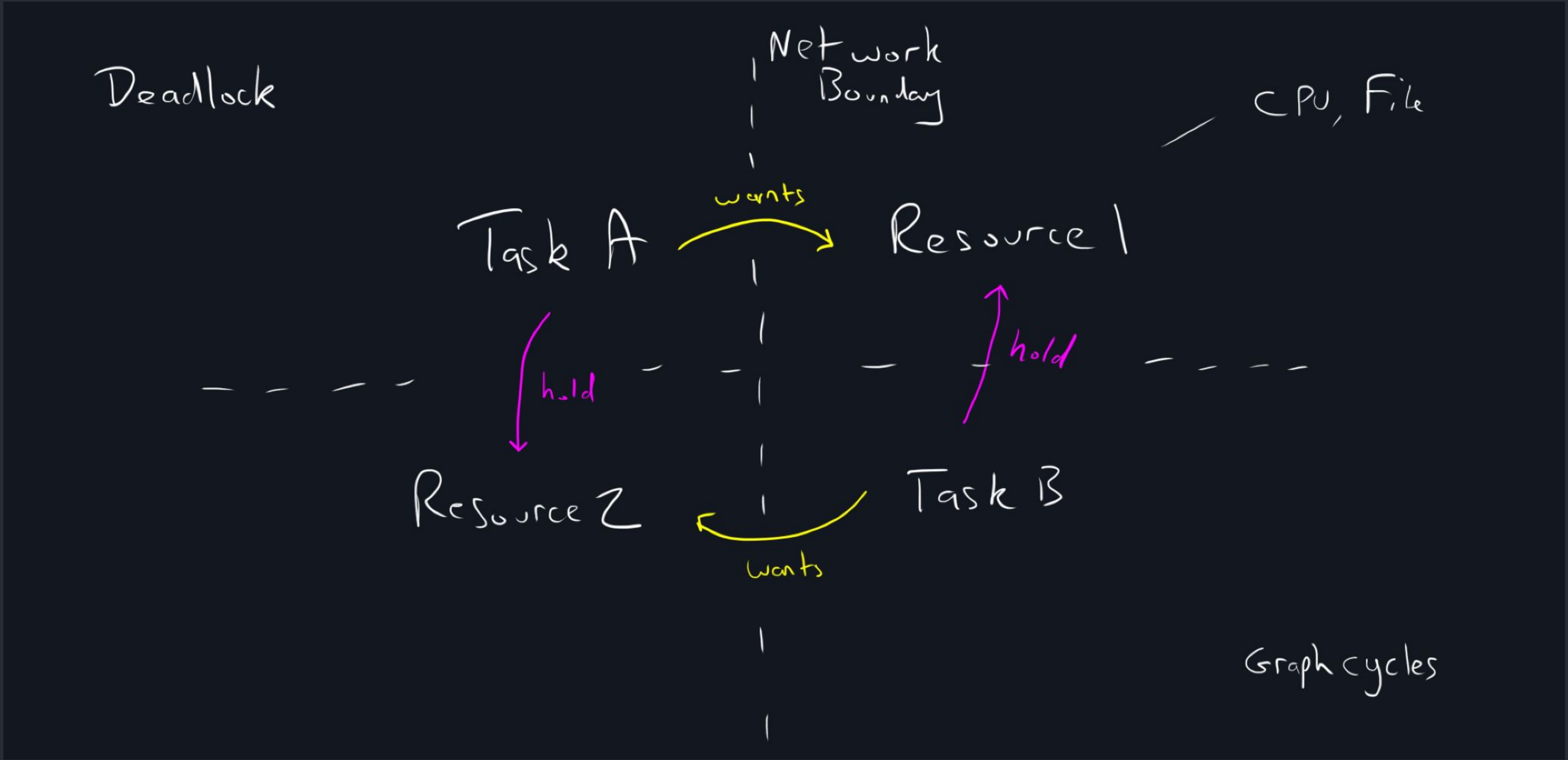
Task 1	Task 2	Account Balance
		1000 Kr
Read Balance		1000 Kr
	Read Balance	1000 Kr
Decrease Balance 600 Kr		1000 Kr
	Decrease Balance 700 Kr	1000 Kr
Update		400 Kr
	Update	**-300 Kr**

Draw: Bug #2 Deadlock

Timeouts



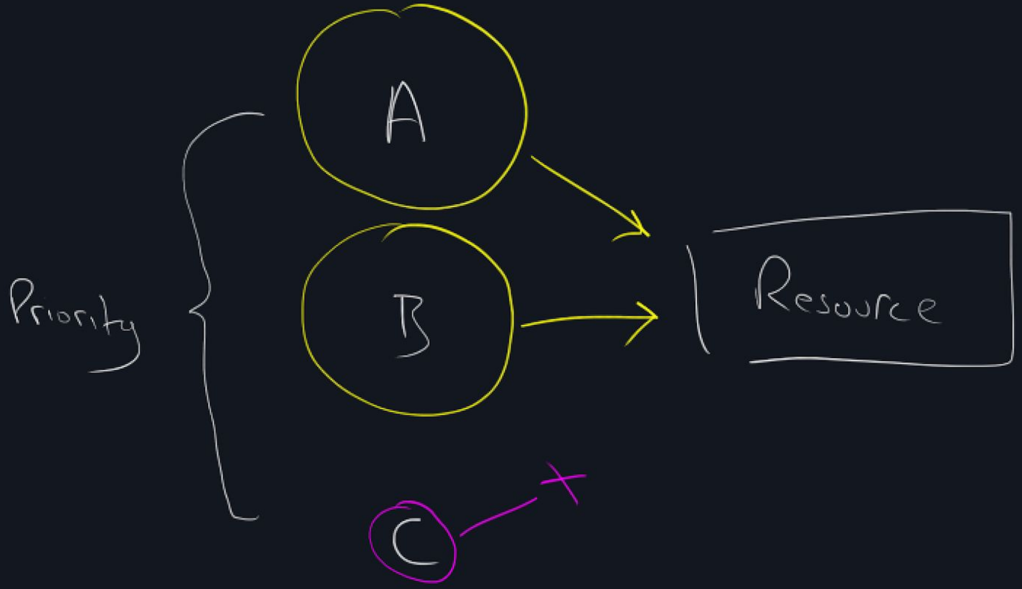
- Many task,
  resources,
  devices,
  network

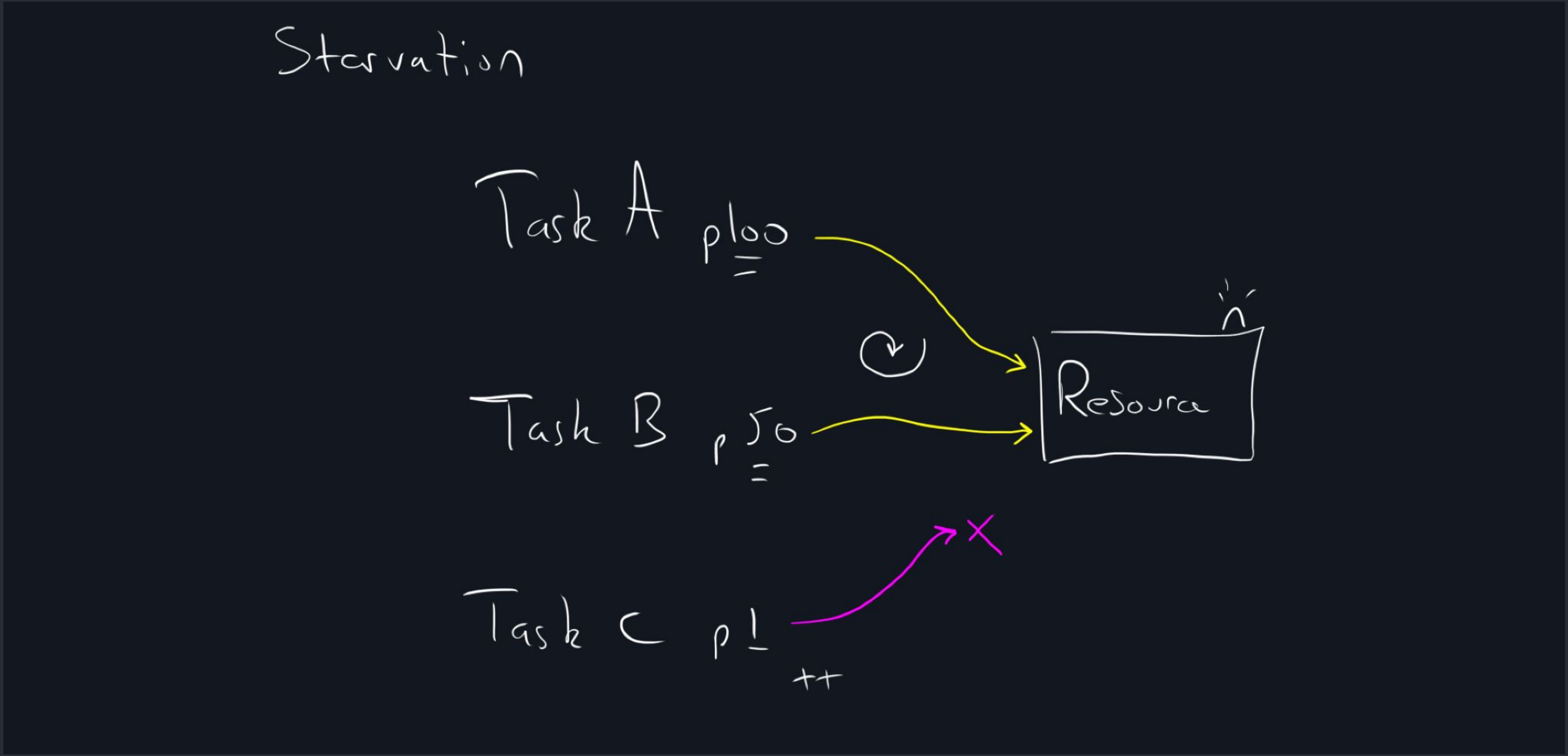

Deadlock
Network
Bounday
CPU, File
wants
Task A
Resource 1
hold
hold
Resource 2
Task B
wants
Graph cycles

Video

Draw: Bug #3 Starvation

Starvation





Video

Concept Review

Concept review

Progress in processing power and plurality gives more opportunity

We can create programs that exploit concurrency and parallelism

But, we must handle new levels of complexity and design challenge