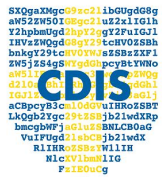


CENTER FOR
CYBER DEFENCE AND
INFORMATION SECURITY

KTH
VETENSKAP
OCH KONST

Provable Security

Roberto Guanciale, KTH
Mads Dam, Henrik Karlsson

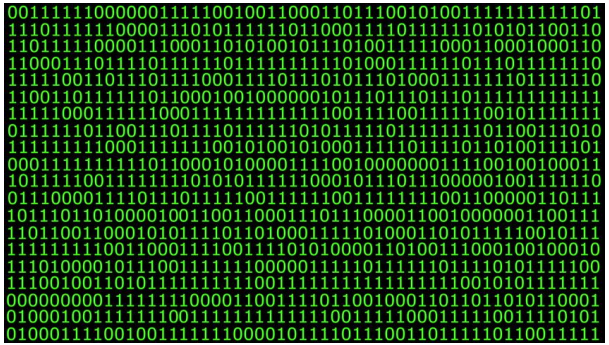
CDIS Spring Conference 2022
KTH Royal Institute of Technology
Tuesday, May 24, 2022

Formal methods

- build our systems in new ways
  - more rigorous techniques.
- **prove** using mathematics and logic the **impossibility** of vulnerabilities in systems

Heavy duty, but remarkable successes for critical SW

Key is to analyse the critical part of the system and guarantee that the other parts do not compromise these properties

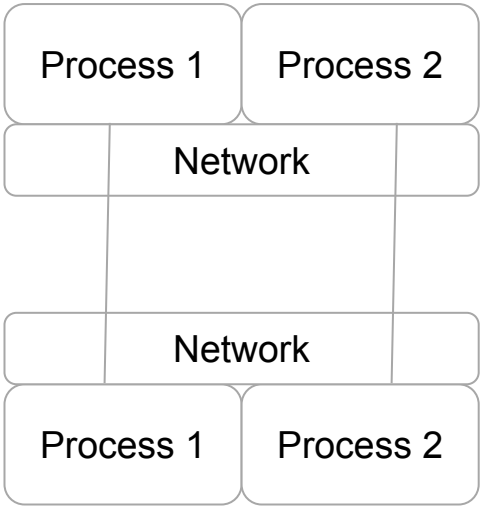


Formal methods

- build our systems in new ways
  - more rigorous techniques.
- **prove** using mathematics and logic the **impossibility** of vulnerabilities in systems

Heavy duty, but remarkable successes for critical SW

Key is to analyse the critical part of the system and guarantee that the other parts do not compromise these properties



Let’s look at an autonomous grass cutter

WiFi

FTP

Motor Control

Voice command

Battery

Proximity sensor

Schedules

Bluetooth

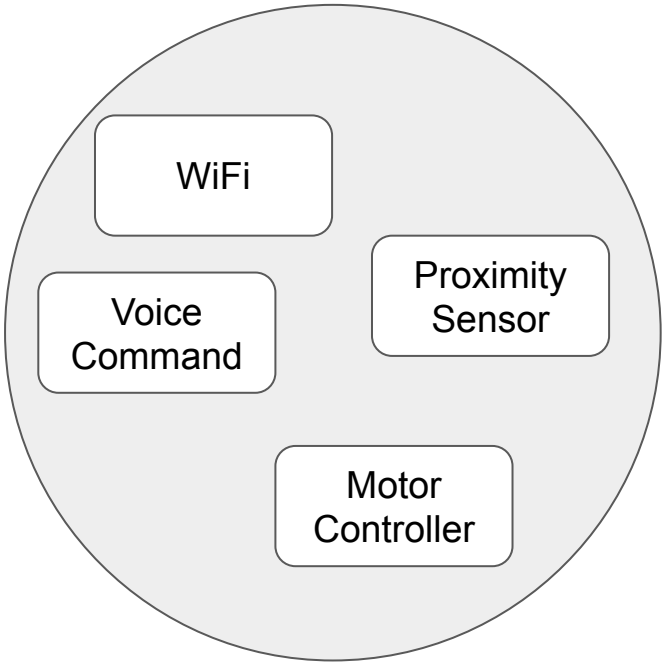
Mapping

Camera

- varying quality
- varying trust
- different supply chains

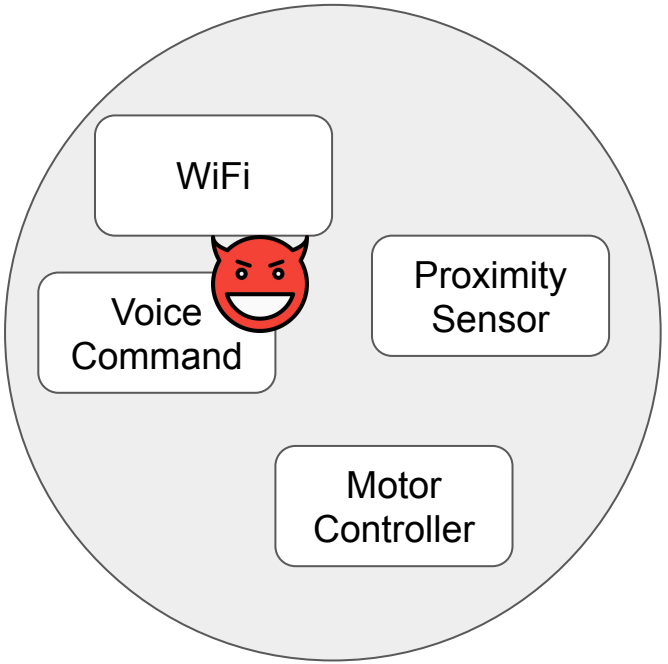
Key problem for security

Complexity



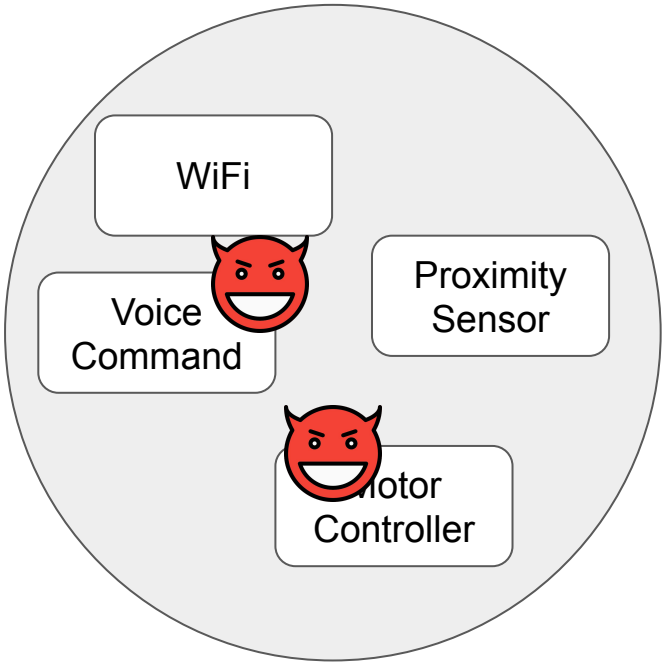
Key problem for security

Complexity



Key problem for security

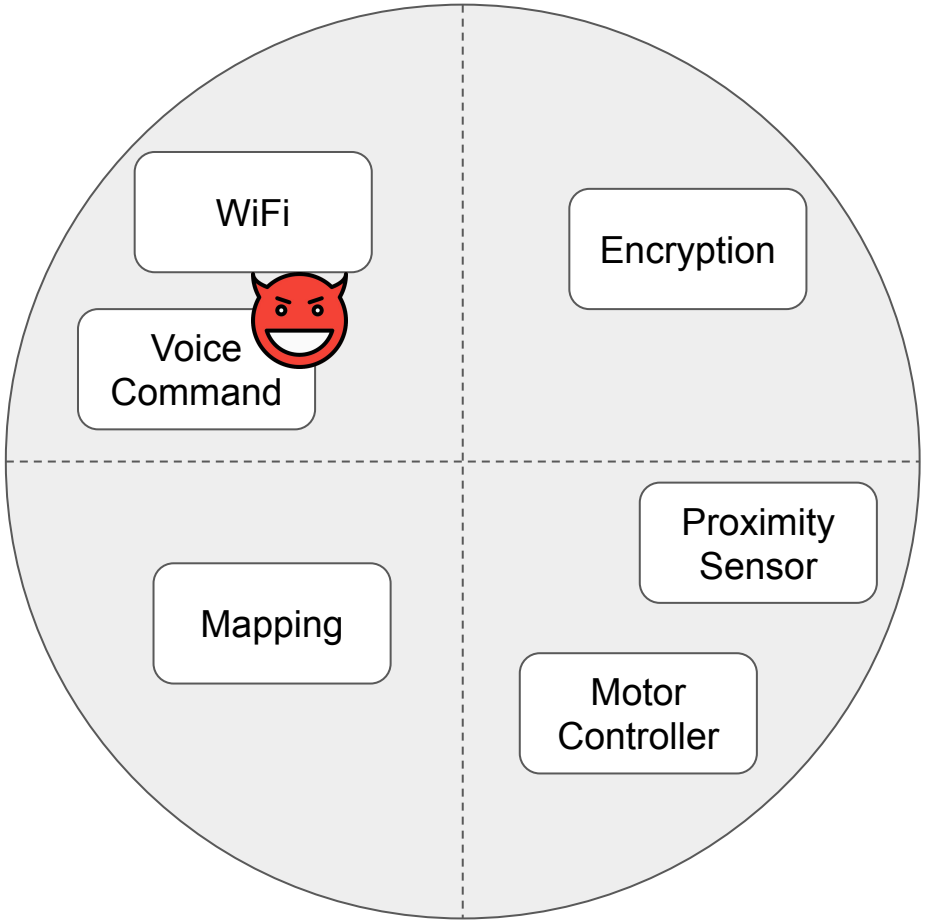
Complexity



Standard OSs (Linux)

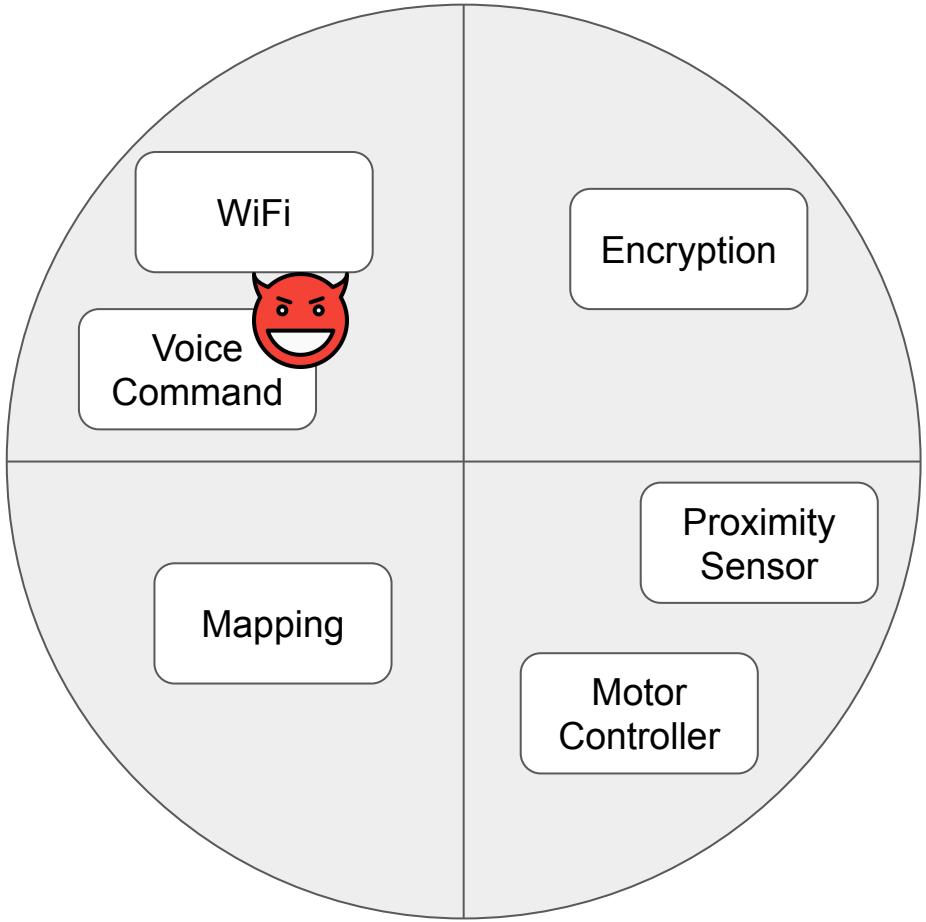
Notoriously buggy

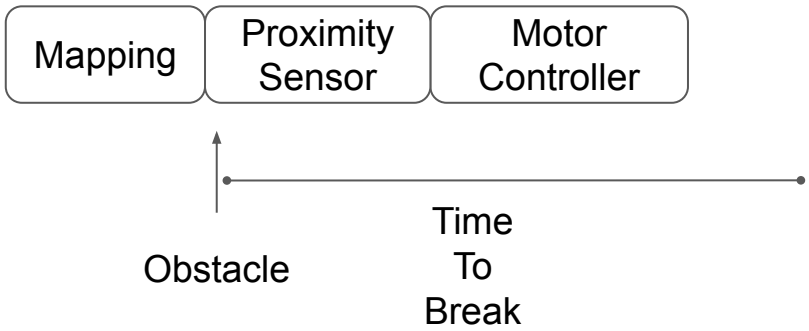
- 27.8 million lines of code
- 75 thousands code commits/year
- Linus Torvalds 3.19% of commits
- 4,189 different contributors



Microkernels

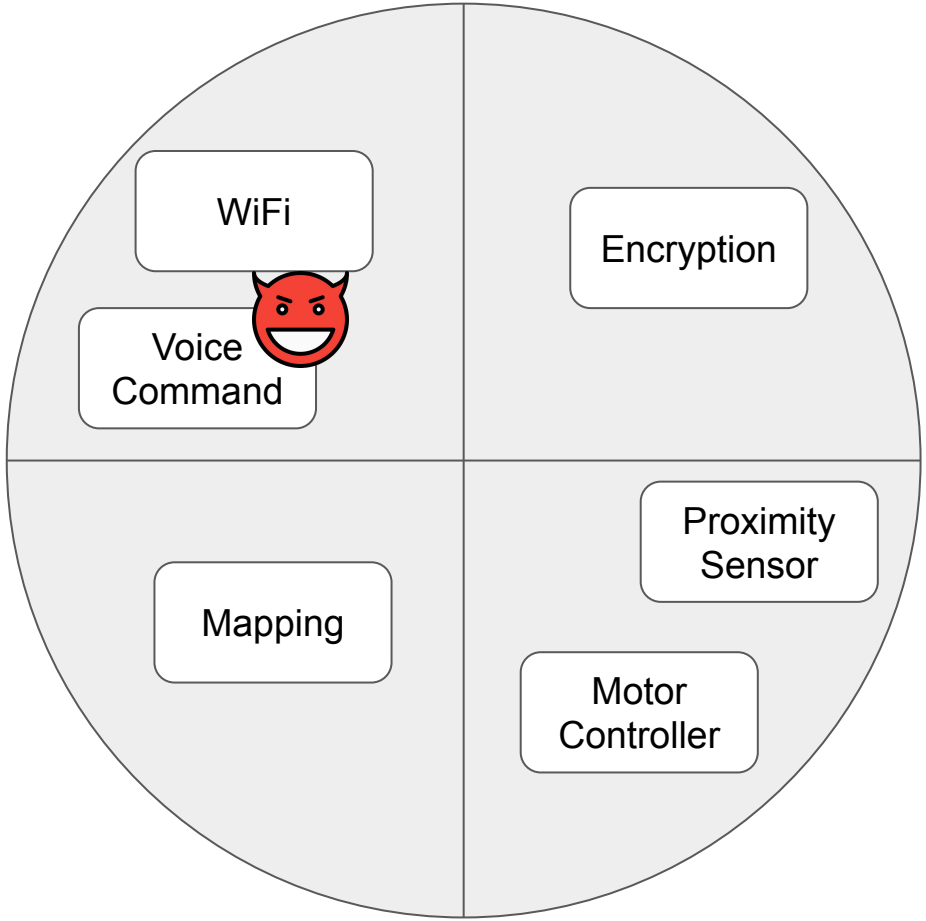
- Functionally correct
- Strong spatial isolation
- NO non-functional guarantees

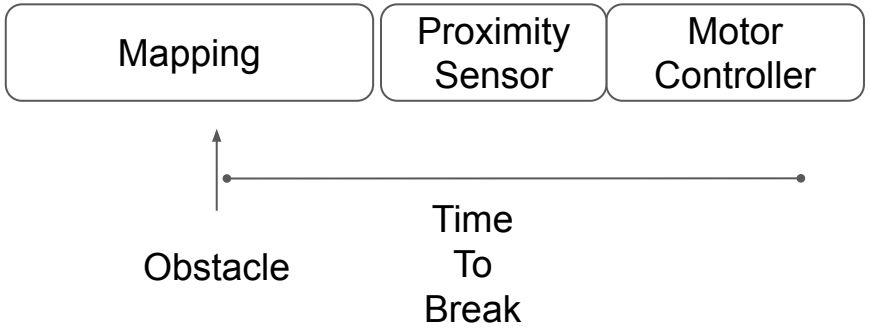




Microkernels

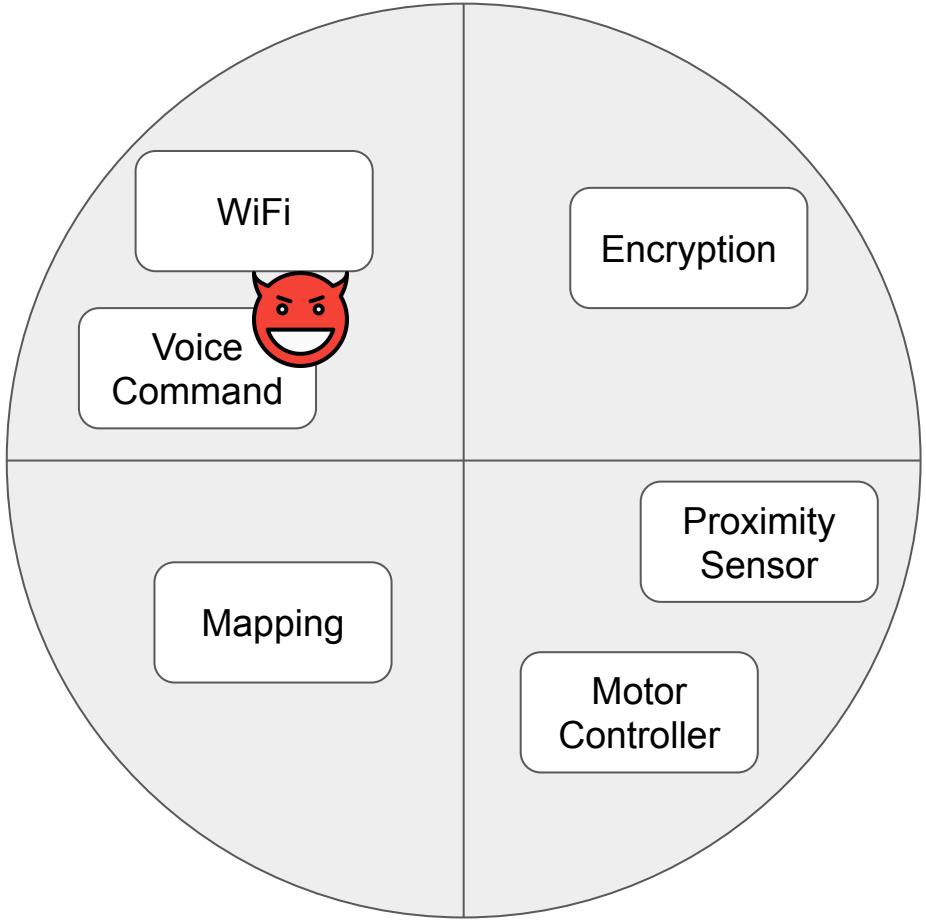
- Functionally correct
- Strong spatial isolation
- NO non-functional guarantees

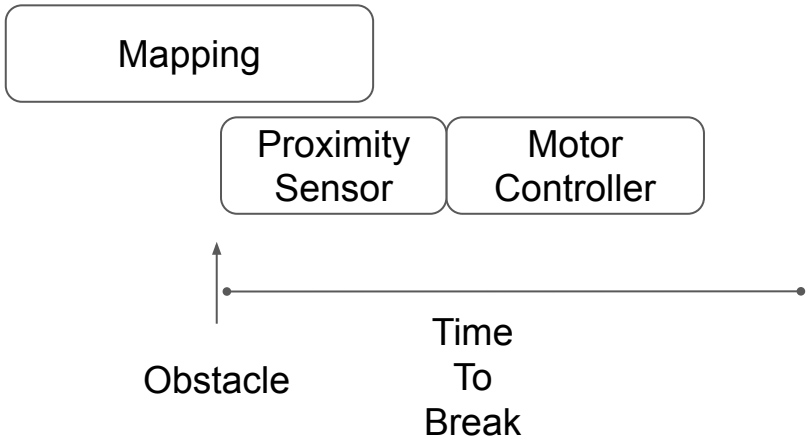




Real Time Systems

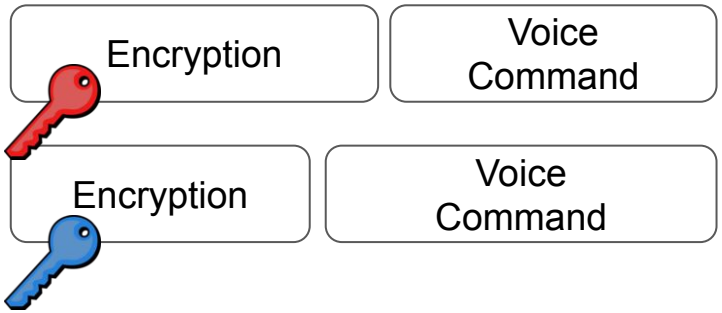
- Priorities to meet deadlines
- No protection against side channel

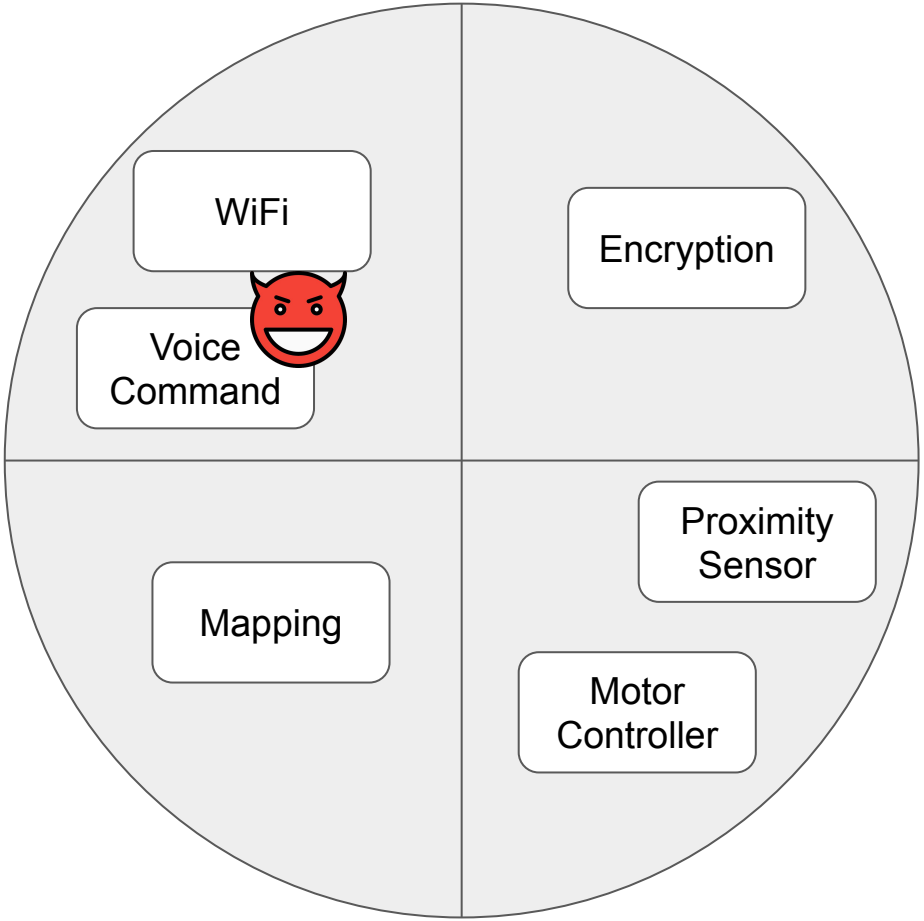




Real Time Systems

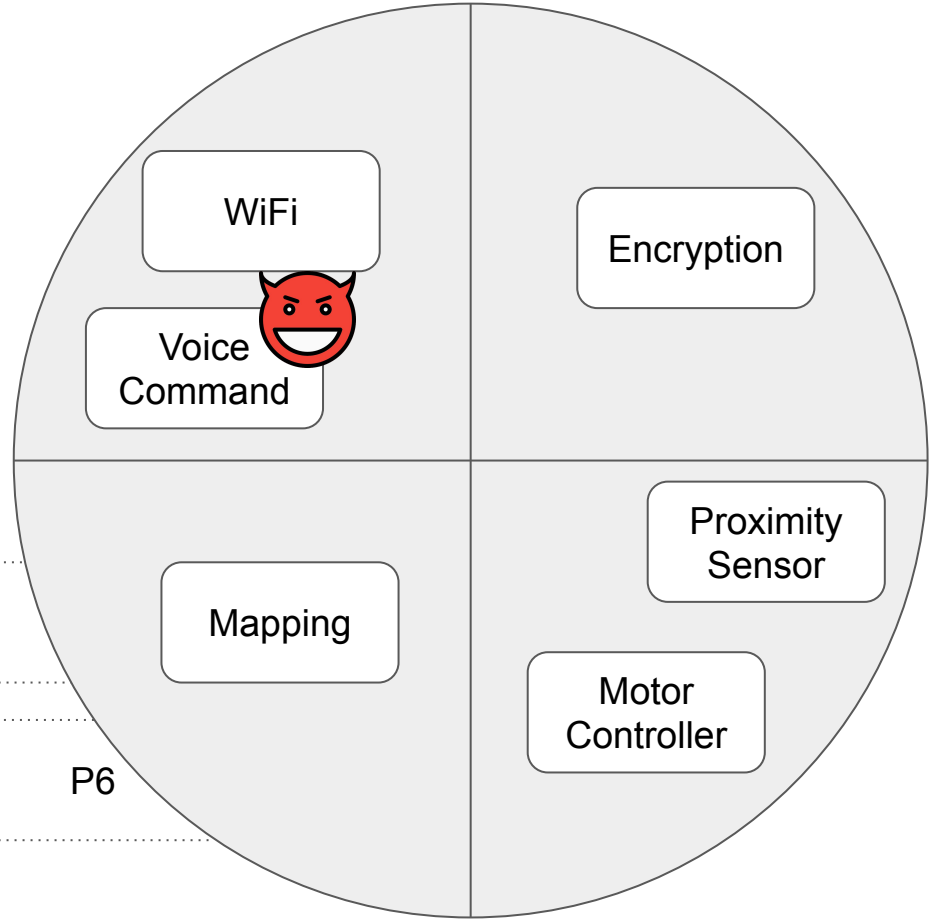
- Priorities to meet deadlines
- No protection against side channel

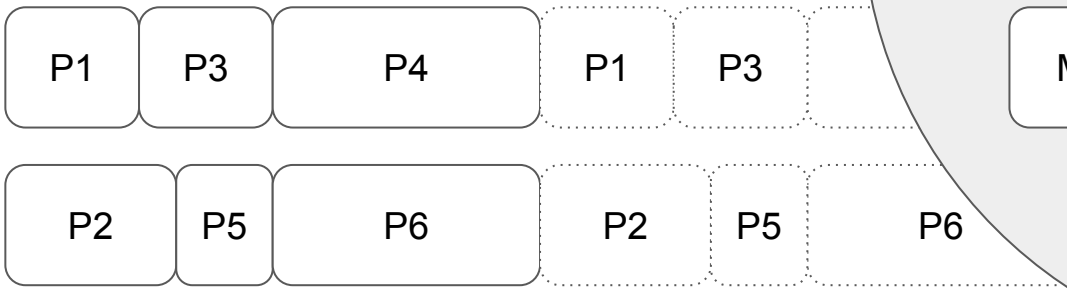




Separation Kernel

- Both strong spatial and temporal isolation
- Used in Avionics (ARINC 653)
- Everything is static





Minor Frame

Mayor Frame

A new model for separation kernels

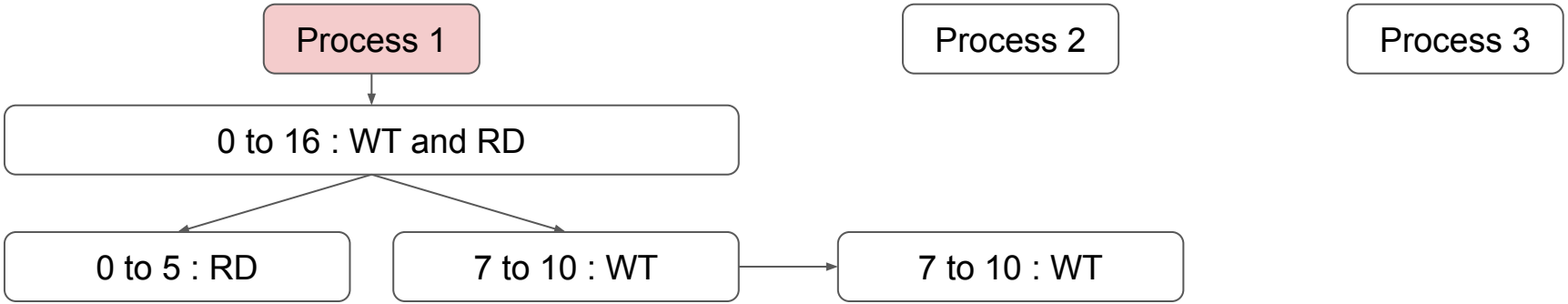
Use the capability model for all types of resources:

- Process management
- Process communication
- Memory
- Time

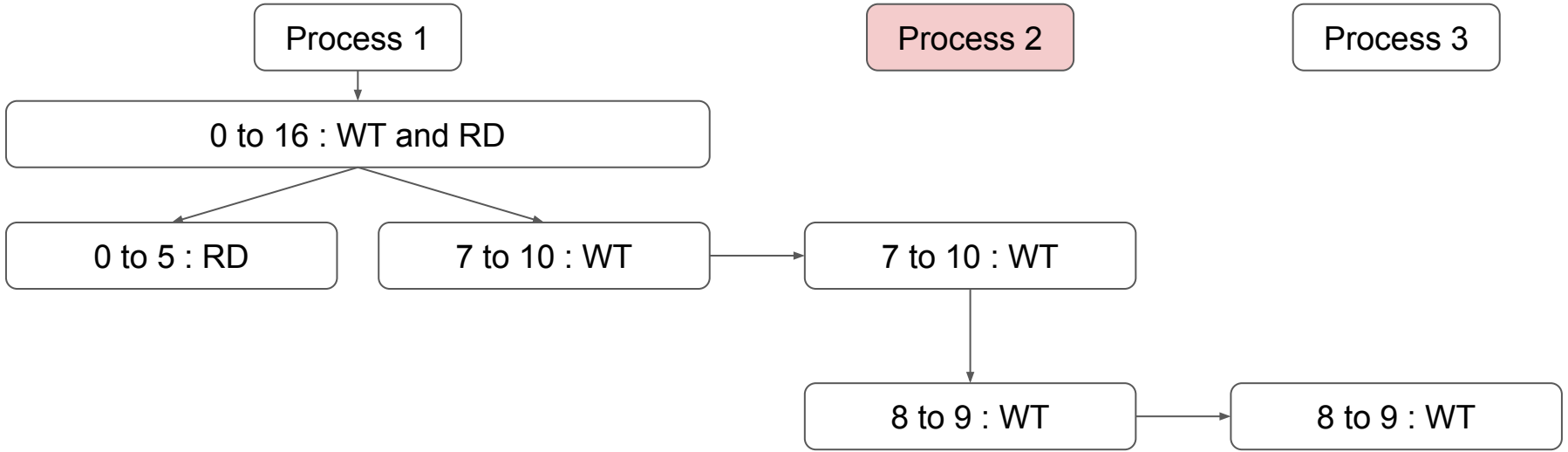
Capability model

Capability model: derivation

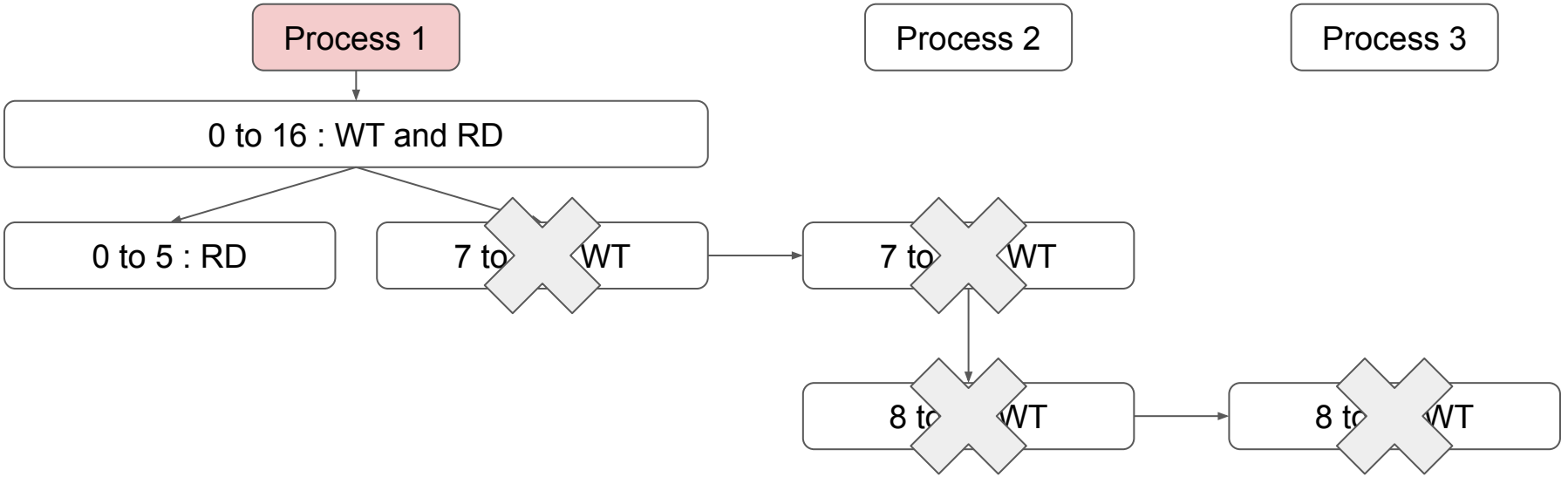
Capability model: delegation



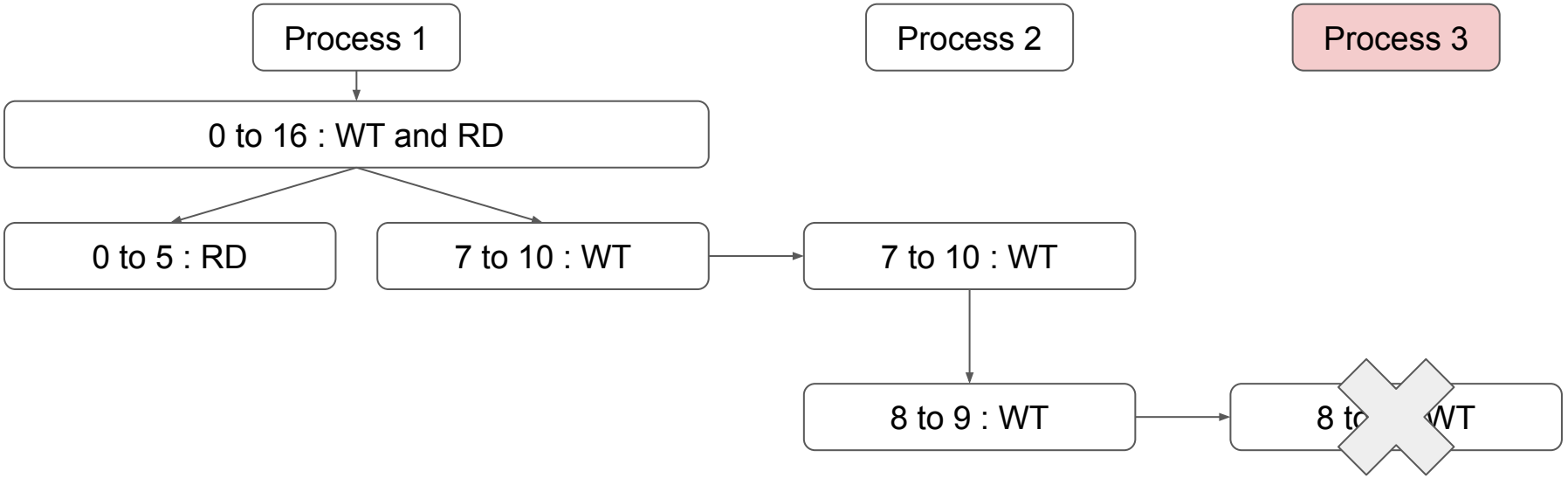
Capability model: delegation



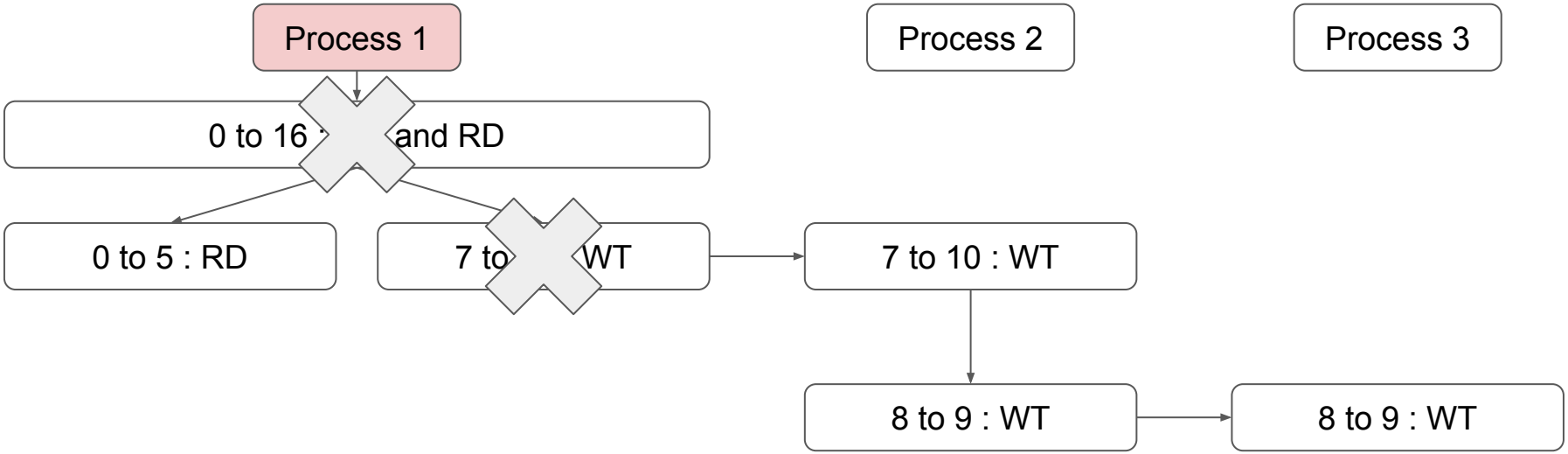
Capability model: revocation



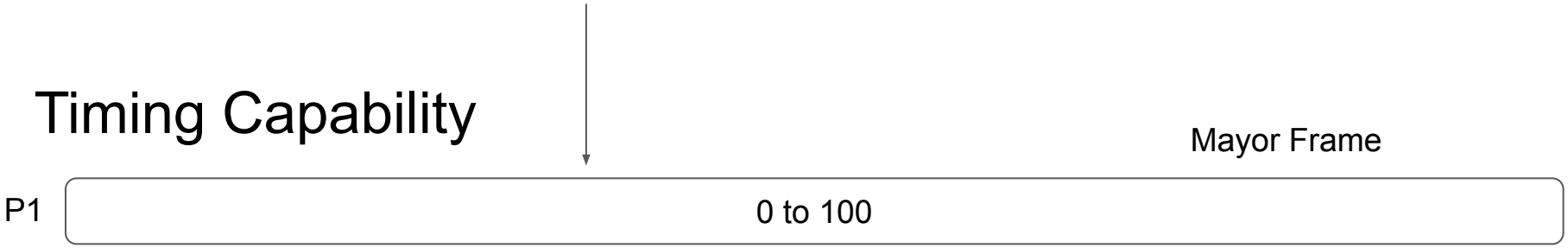
Capability model: deletion



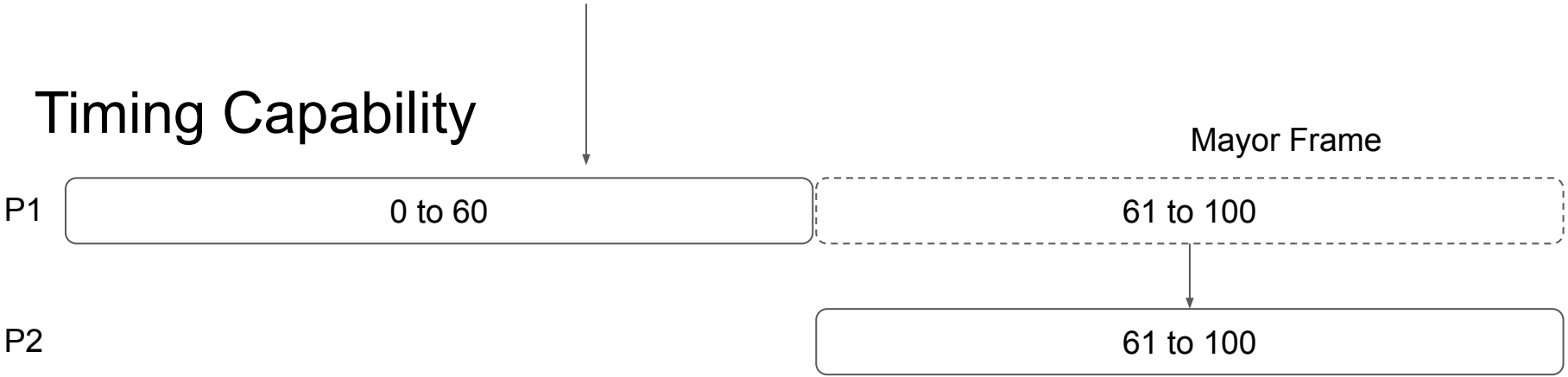
Capability model: drop



Timing Capability



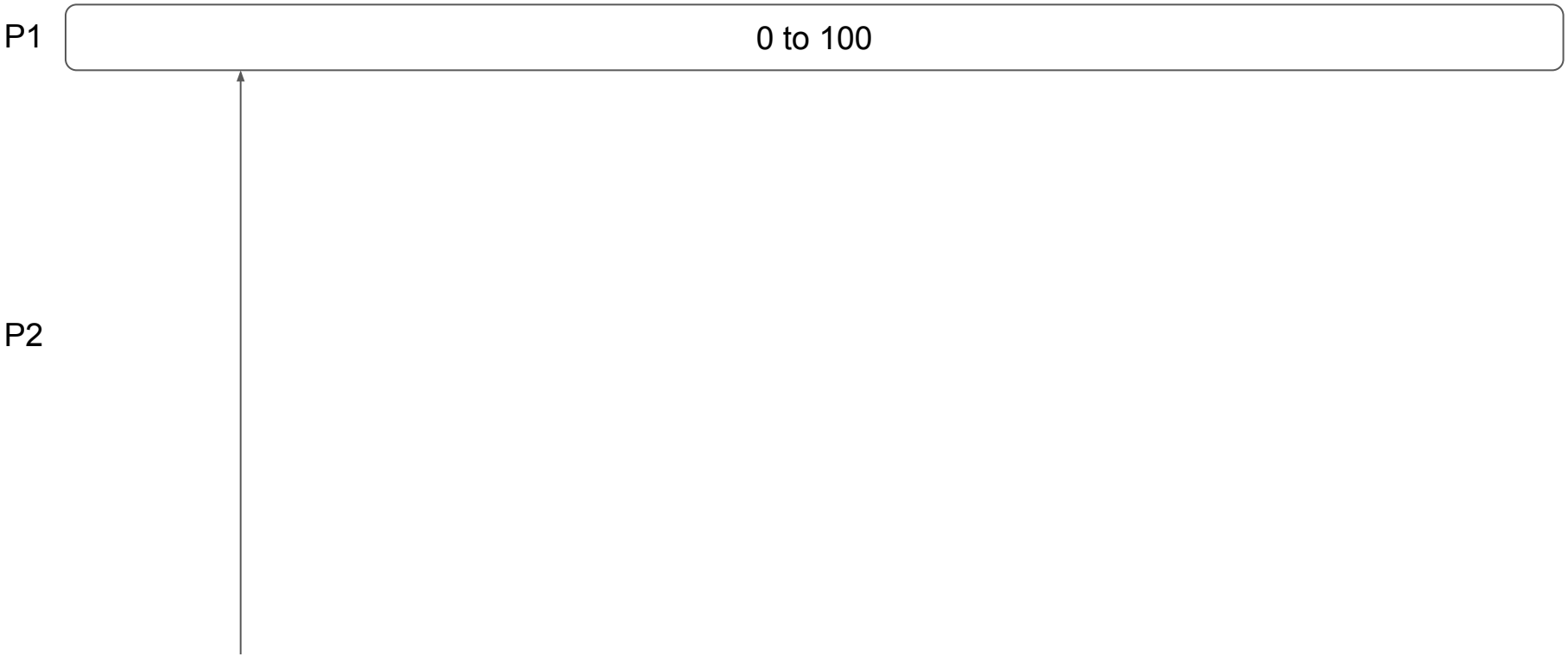
Timing Capability



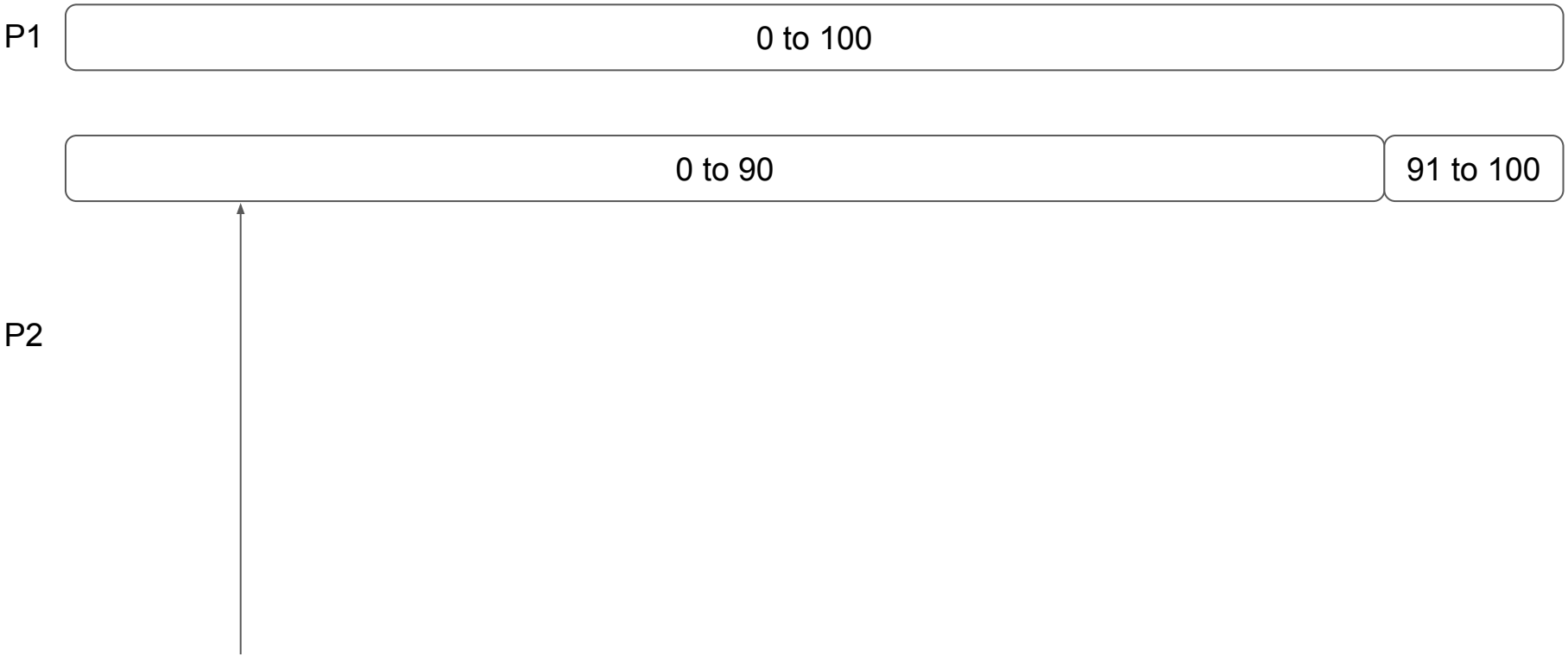
Few APIs but expressive

- Timing Capabilities are per core.
- A process can run only on one core, but it can migrate.
- Kernel is (mostly) preemptive. We use slack time that should cover the non-preemptive parts of the kernel
- 2kLoC of C/Assembly for 64-bit RISC-V

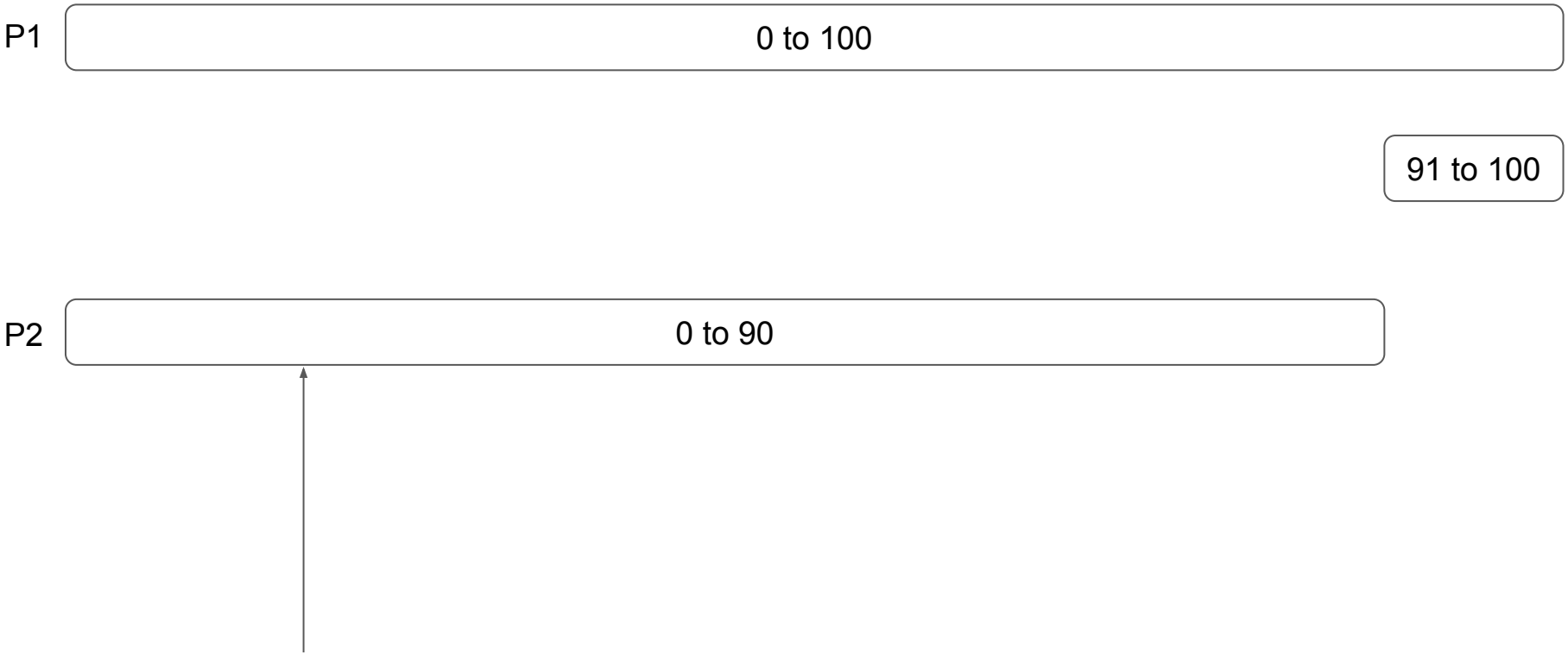
Remote Procedure Call



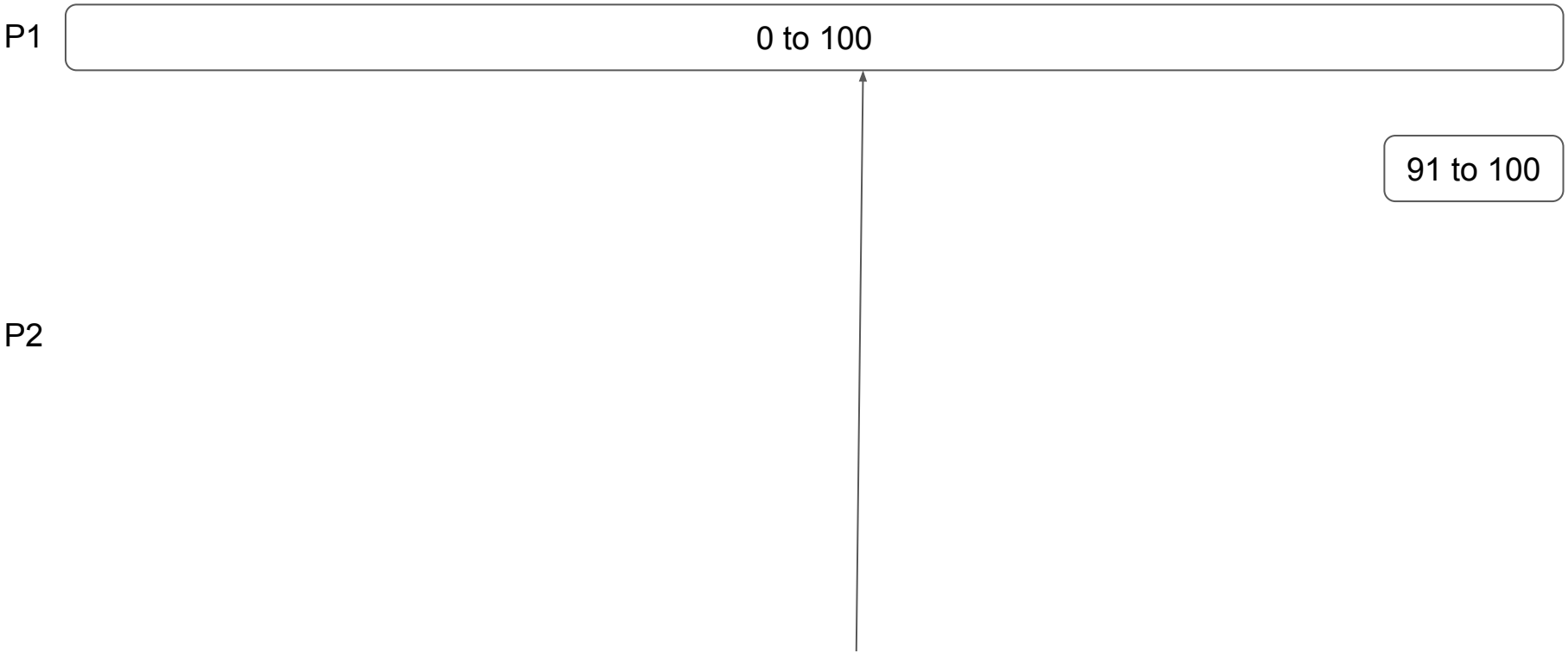
Remote Procedure Call: Derive



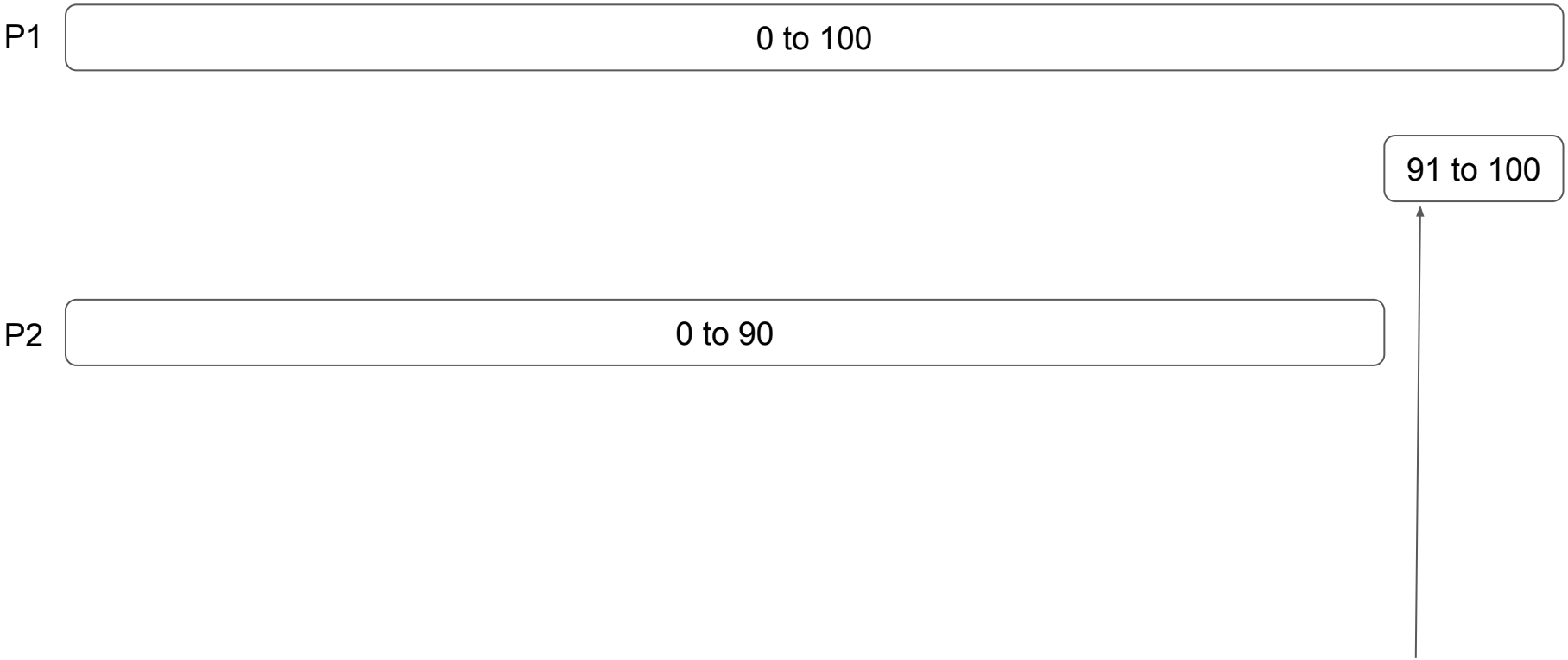
Remote Procedure Call: Delegate



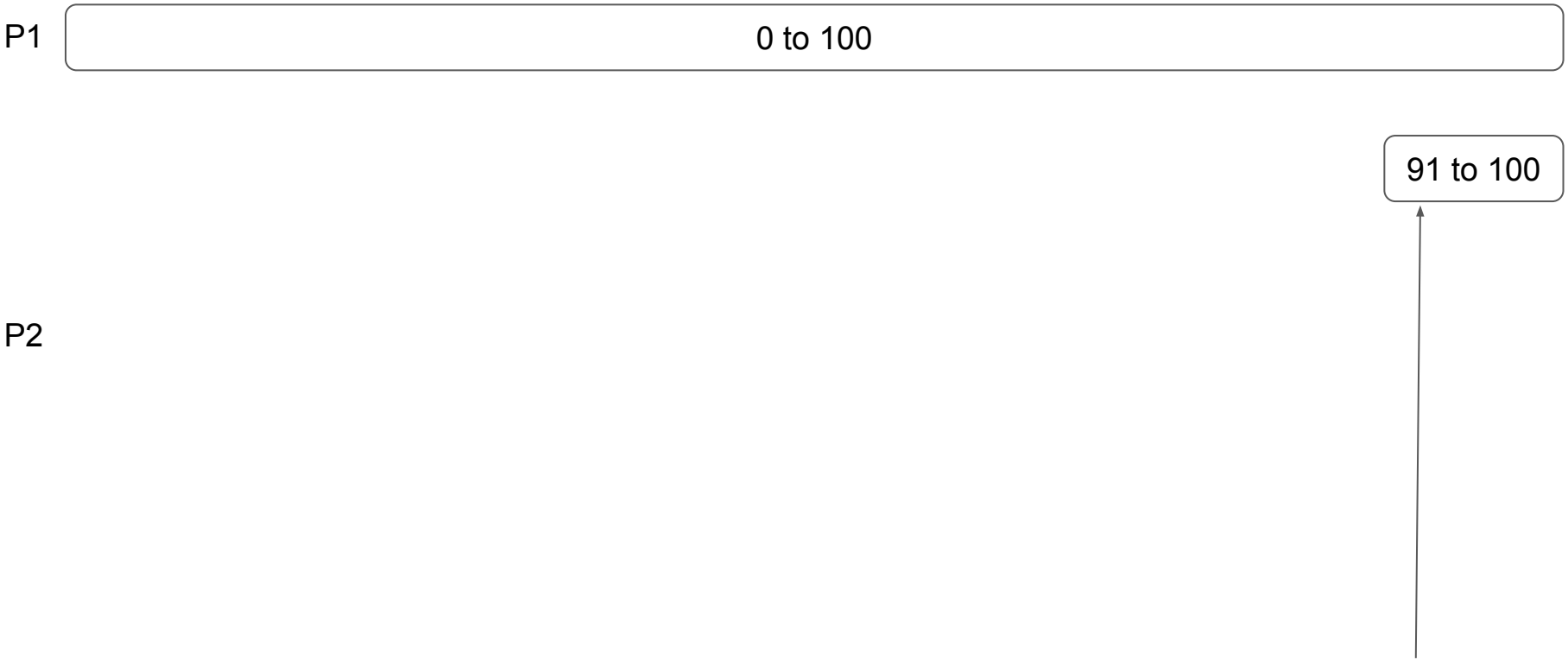
Remote Procedure Call: Delete



Remote Procedure Call: Revoke



Remote Procedure Call: Revoke



Ongoing work

- WCET analysis of non-preemptive kernel
- Demonstrating application:
  - Cyber physical system with multiple modes of operations
  - Reconfigurations to handle faults
- Userland (Posix) and drivers
- Formal verification of kernel system calls
  - Model of RISC-V memory model
- Formal verification of non-functional properties
  - WCET
  - Constant time execution