Pip: A Minimal OS Kernel with Provable Isolation[1]

David Nowak

CRIStAL, CNRS & Lille 1 University

Third French-Japanese Meeting on Cybersecurity
April 24, 2017

[1] Joint work with the 2XS team (Lille), partially supported by the Celtic-Plus European Project ODSI C2014/2-12

Memory isolation between applications

Why? For safety and security

How? By software (OS kernel), and hardware (MMU, kernel mode)

Correct? Ensured by a formal proof in Coq

Doable? By reducing the trusted computing base to its bare bone

Outline

What is the Pip protokernel?

How does Pip work?

How are Pip isolation properties proved?

Outline

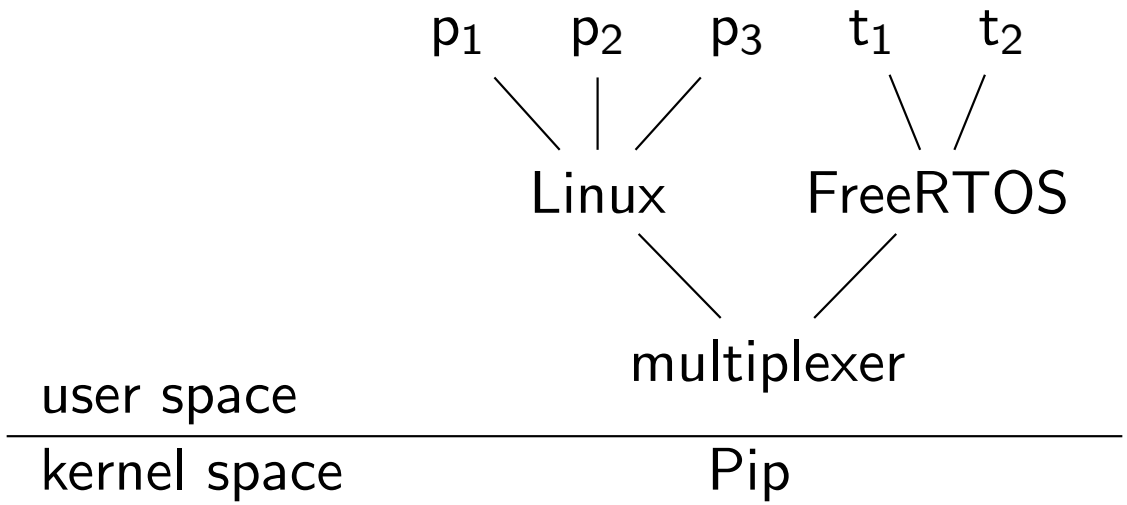
The Pip protokernel: a minimal OS kernel

- A kernel runs in the privileged mode of the CPU.
- Therefore it is highly critical.
- With Pip, the trusted computing base (TCB) is minimal:
  - Scheduling and IPC are done in user mode.
    unlike a microkernel
  - Multiplexing is also done in user mode.
    unlike a hypervisor or an exokernel
  - Kernel mode is only for:
    - multi-level MMU configuration (virtual memory),
    - context switching.
- $$\text{minimal TCB} = \begin{matrix}\text{less risk}\\ \text{of bug}\end{matrix} + \begin{matrix}\text{more feasibility}\\ \text{of formal proof}\end{matrix}$$

Partition tree

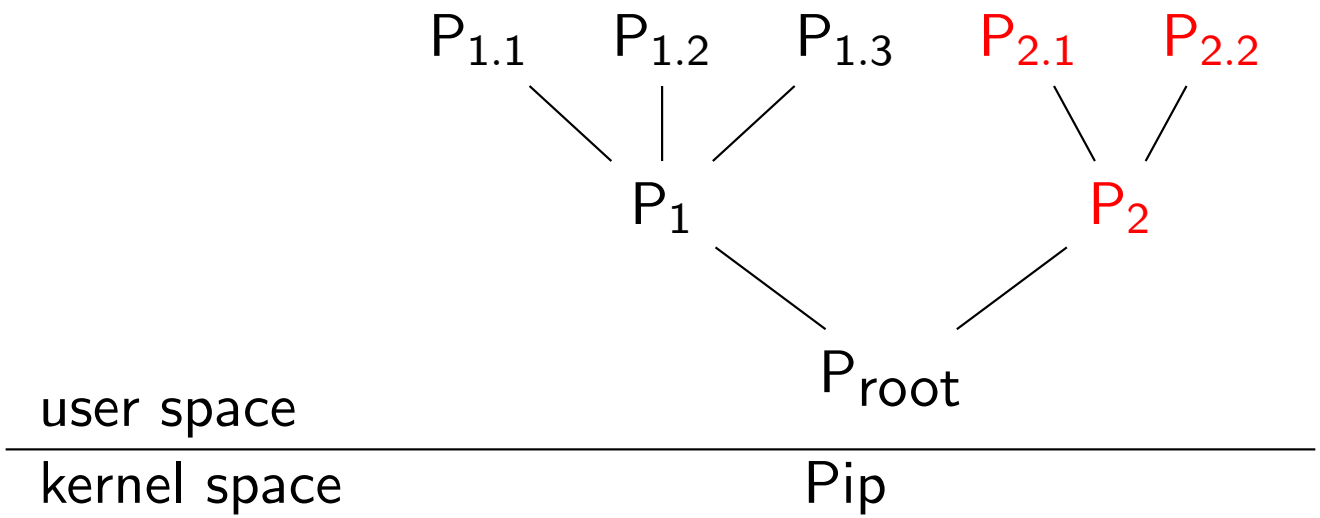
The memory is organized into hierarchical partitions.

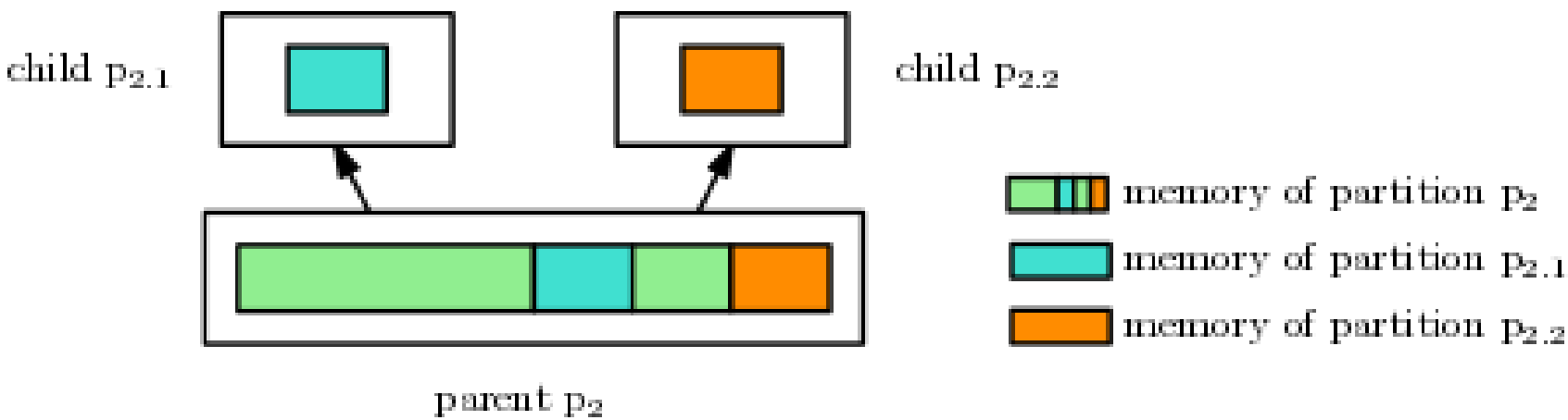
Example



- FreeRTOS is a real-time OS that does not isolate its tasks.
- by porting it on Pip, we easily secured it with task isolation.

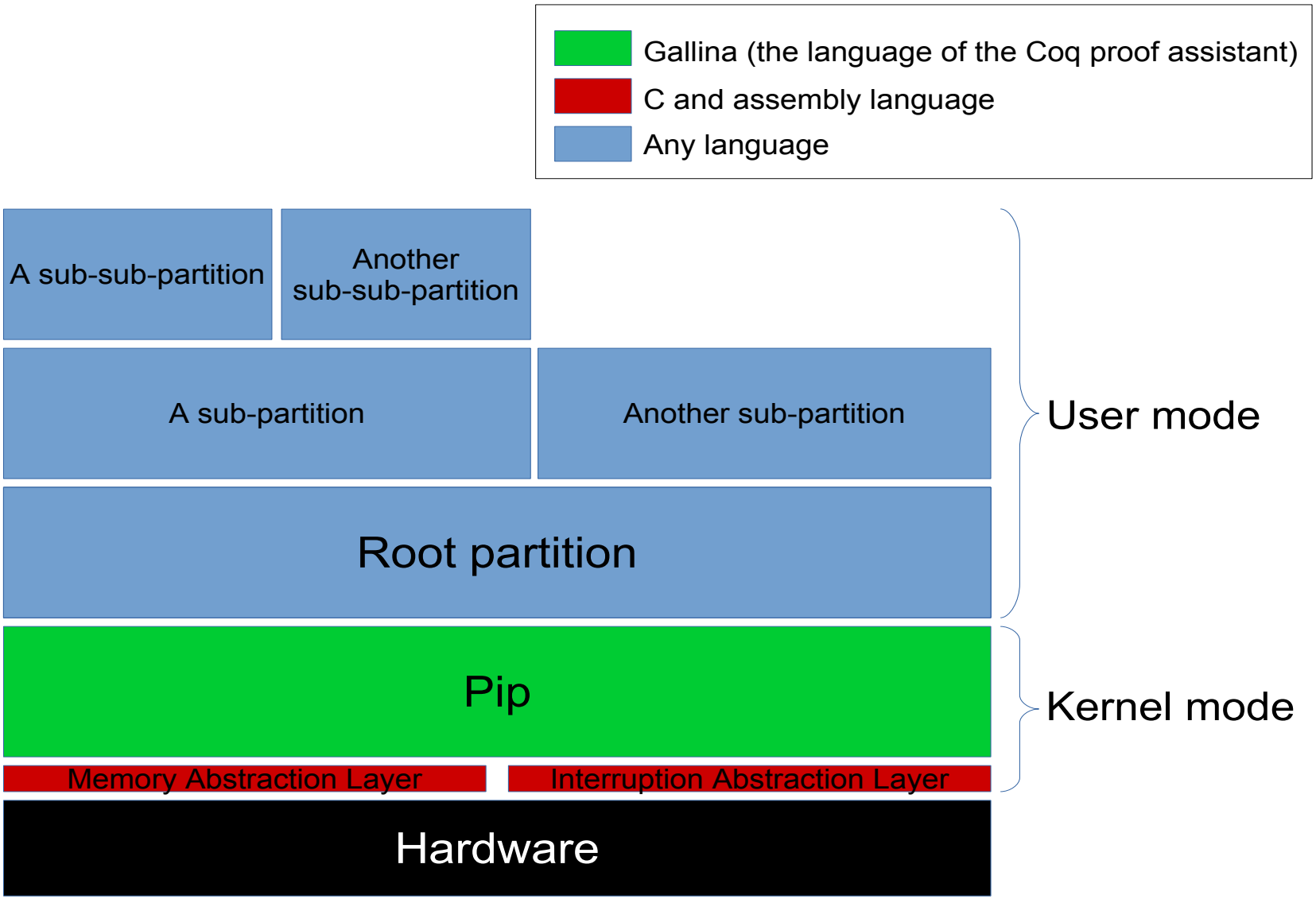
Horizontal isolation and vertical sharing





Outline

Software layers



The API of Pip

9 system calls can be called by the code of any partition

`createPartition`	create a partition
`deletePartition`	delete a partition
`addVAddr`	map an address
`removeVAddr`	remove a mapping
`pageCount`	return the number of indirections to map an address
`prepare`	add the indirections to map an address
`collect`	delete all empty indirections
`dispatch`	send a signal to a child partition
`resume`	return control to another partition

Some Pip internals

- Pip redirects:
  - a sofware interrupt to the parent of the caller,
  - a hardware interrupt to the root partition.

- Data structures
  - The MMU pages tables (used by Pip and MMU)

    for translation of a virtual address into a physical address

  - two shadow MMUs and a linked list (used by Pip only).
    - for storing additional information about of virtual addresses
    - for optimization

- The kernel is always mapped but not accessible in user mode.

  for efficient system calls

Outline

The hardware monad

- Gallina is a purely functional language.
- But, in order to access hardware, we need imperative features:
  - updatable state;
  - undefined behaviors:
    - out-of-bound physical address,
    - type error,
    - ...;
  - halting.
- We wrap those imperative features in a monad.
- We define a Hoare logic on top of this monad.

Memory isolation (1/2)

- not from the point of view of information flow
- but at the lower level of page table management
- A state is **isolated** iff, for any two distinct processes P_1 and P_2, any page used by P_1 is not used by P_2.
  - By *pages used by a process* P_i, we mean the pages referenced in its page table $ptp(P_i)$ and the page $ptp(P_i)$ itself.
  - By *two distinct processes* P_1 *and* P_2, we mean $ptp(P_1) \neq ptp(P_2)$
- Our goal is to show that this property is preserved.

Memory isolation (2/2)

- We would be satisfied if we could prove the following triple for each system call c:

$$\{\textbf{Isolated}\}\ c\ \{\textbf{Isolated}\}$$

- But it is false in general:
  - The precondition must be strenghened with consistency properties.
  - Those consistency properties must also be preserved.

$$\{\textbf{Isolated} \wedge \textbf{Consistent}\}\ c\ \{\textbf{Isolated} \wedge \textbf{Consistent}\}$$

- consistency $\approx$ well-formedness of Pip's data structures

Translating Gallina into C

- Word-for-word translation: possible because of monadic style

- **Example**: In Gallina, we write:

```
Definition getFstShadow (partition : page) : page :=          1
  perform idx := getSh1idx in                                 2
  perform idxSucc := MALInternal.Index.succ idx in            3
  readPhysical partition idxSucc.                             4
```

Its translation in C is:

```
uintptr_t getFstShadow(const uintptr_t partition) {           1
  const uint32_t idx = getSh1idx();                           2
  const uint32_t idxSucc = succ(idx);                         3
  return readPhysical(partition, idxSucc); }                  4
```

- **Work in progress**: proving the correctness of this translation

Applications

- Supported by the European project ODSI
  - PhD students:
    Quentin Bergougnoux, Narjes Jomaa, Mahieddine Yaker
  - Postdoc:
    Paolo Torrini
  - Case studies by industrial partners: IoT, M2M, SCADA
- Discussion with the European branch of a Japanese company
  Isolate the CAN network and the Ethernet network in a car

Conclusions

- A new design of OS kernel amenable to formal proof
- An implementation: the Pip protokernel
- To find out more:

`http://pip.univ-lille1.fr`