Understanding the Genetic Makeup of Linux Device Drivers

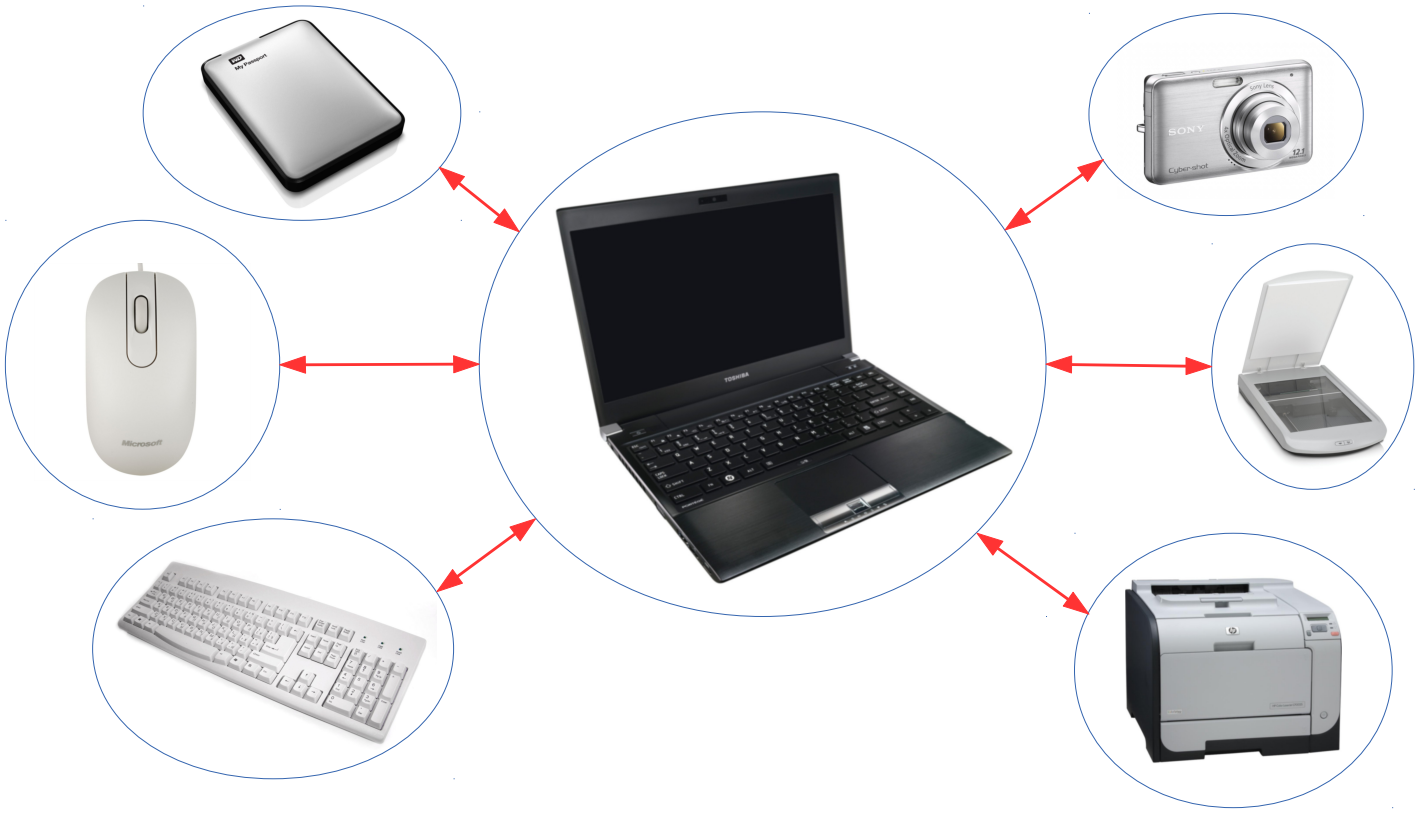
(Work in Progress)

Peter Senna Tschudin, Laurent Réveillère, Lingxiao Jiang,
David Lo, Julia Lawall, Gilles Muller

LIP6 Inria & UPMC, LaBRI, Singapore Management University
Preliminary work published at PLOS 2013

March, 2014

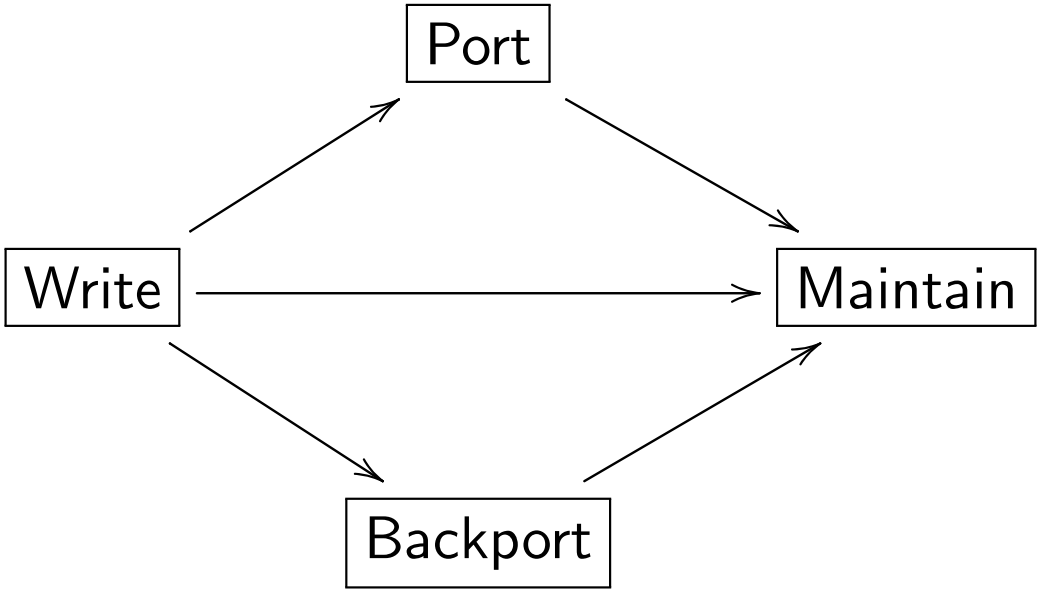
What is a device driver?



Data translator between OS and device

- May also perform computations, such as checksums

Device Driver Development Issues



- Complex
- Expensive
- Slow
- Error-Prone

What has been proposed

Domain-specific languages: Devil [OSDI 2000]

Automatic synthesis from specifications: Termite [SOSP 2009]

Templates: RevNIC [Eurosys 2010]

What has been proposed

Domain-specific languages: Devil [OSDI 2000]

- Only covers hardware interaction code.

Automatic synthesis from specifications: Termite [SOSP 2009]

Templates: RevNIC [Eurosys 2010]

What has been proposed

Domain-specific languages: Devil [OSDI 2000]

- Only covers hardware interaction code.

Automatic synthesis from specifications: Termite [SOSP 2009]

- Requires access to specifications
- Only covers input/output behavior

Templates: RevNIC [Eurosys 2010]

What has been proposed

Domain-specific languages: Devil [OSDI 2000]

- Only covers hardware interaction code.

Automatic synthesis from specifications: Termite [SOSP 2009]

- Requires access to specifications
- Only covers input/output behavior

Templates: RevNIC [Eurosys 2010]

- Templates must be handwritten
- Legality questionable

An observation

There are already good drivers

- Billions of instances of device drivers running right now
- Some have being developed for many years
- Used in mission critical environments (we are all alive)

How to exploit the knowledge found in good drivers to help us make new ones?

Our vision

A driver is a collection of entry points

- Probe, remove, suspend, resume, ...

Entry points are made up of building blocks for common features

- Create and register device structure
- Initialize memory mapped I/O
- Enable reception of interrupts

Porting *e.g.* from Linux to BSD

- Identify building blocks used in the Linux driver.
- Replace them with the corresponding BSD building blocks.

Our proposal:
Drivers are made of genes

A gene:

- Motivated by device features and OS API
- Set of possibly non-contiguous code fragments
- Express the behavior of a feature

How to identify genes?

Our proposal:
Drivers are made of genes

A gene:

- Motivated by device features and OS API
- Set of possibly non-contiguous code fragments
- Express the behavior of a feature

How to identify genes?

- Alternatively, how to decompose the code as a product line?

Clone detection?

Urban legend:
Device drivers are implemented by copy-paste

CP-Miner [OSDI 2004]:

- 22.3% of Linux 2.6.6 code is involved in clones
- Most common copy-paste groups contain only 2 copies
- Most common copy-paste segment size: 5-16 statements

Clone detection issues

```
static int ftmac100_probe(...) {
  ...
  irq = platform_get_irq(pdev, 0);
  if (irq < 0) return irq;
  netdev = alloc_etherdev(...);
  if (!netdev) { ... }
  ...
  priv->irq = irq;
  ...
  err = register_netdev(netdev);
  if (err) { ... }
  ...
  return 0;
  ...
  free_netdev(netdev);
}
```

drivers/net/ethernet/faraday/ftmac100.c

```
static int am79c961_probe(...) {
  ...
  dev = alloc_etherdev(...);
  if (!dev) { ... }
  ...
  ret = platform_get_irq(pdev, 0);
  if (ret < 0) { ... }
  dev->irq = ret;
  ...
  ret = register_netdev(dev);
  if (ret == 0) {
    ...
    return 0;
  }
  ...
  free_netdev(dev);
}
```

drivers/net/ethernet/amd/am79c961a.c

Protocol mining?

Urban legend?
Device drivers are programmed according to implicit rules for the use of various combinatins of API functions.

Bugs as deviant behavior [SOSP 2001], PR-Miner [FSE 2005], . . . :

- When A appears, if B often also appears, then maybe they are related.
- PR-Miner finds 1,075 API function rules in Linux 2.6.11.
- 2.6.11 contains over 4 million lines of code

Protocol mining concepts and issues

Key concepts:

- Support: How often a set of things occurs together.
- Confidence: If part of a set of things is present, then how often is the rest present as well.

Problem: Many variants of similar protocols:

- `SET_NETDEV_DEV -> alloc_etherdev netdev_priv`
- `SET_NETDEV_DEV -> netdev_priv register_netdev`

Problem: Multiple variants exist for some functions:

- `alloc_etherdev`, `alloc_etherdev_mq`, `alloc_etherdev_mqs`

Protocol mining issues, contd.

Problem: Inferred protocols may relate irrelevant information:

- `INIT_WORK -> SET_NETDEV_DEV alloc_etherdev netdev_priv register_netdev`

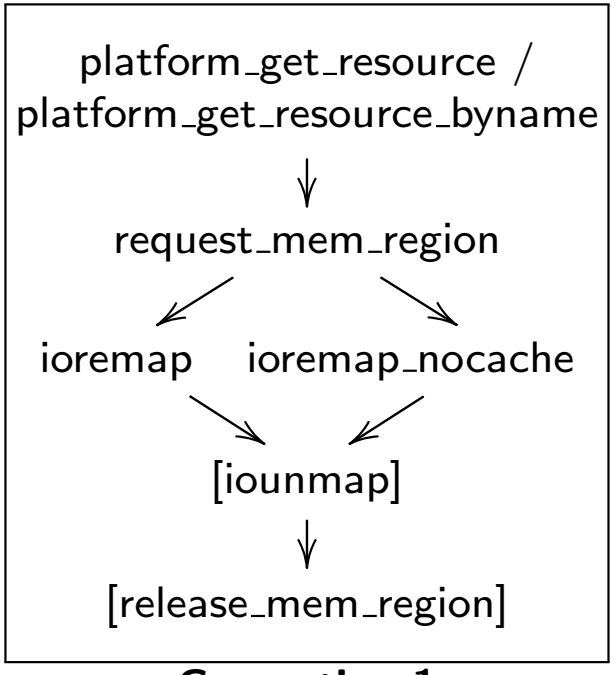
Problem: Some functions used in multiple ways:

- Neighbors of `ioremap`:

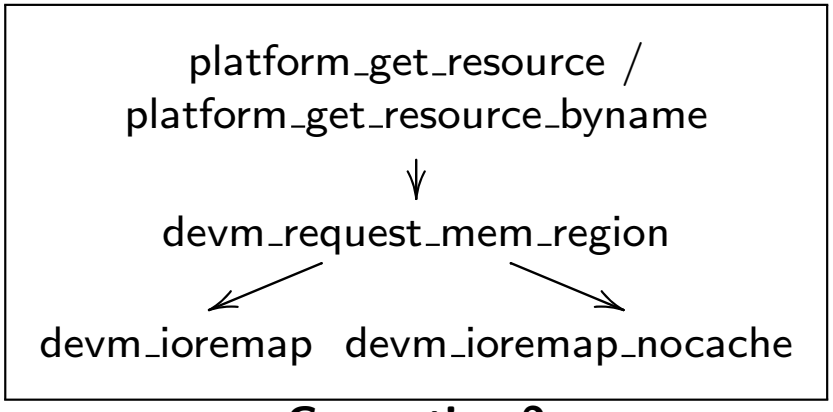
in platform drivers	in pci drivers

`platform_get_resource`
`resource_size`
`request_mem_region`
	`pci_resource_start`
	`pci_resource_len`

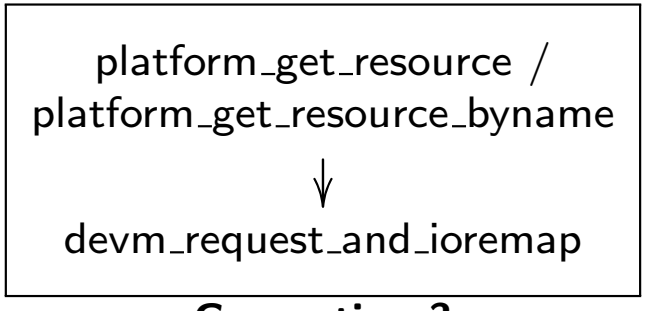
Some manually obtained results



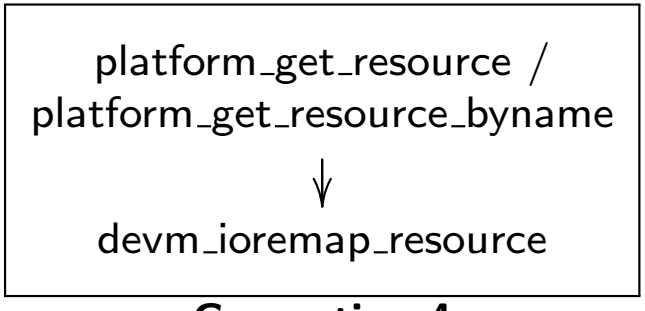
Generation 1



Generation 2



Generation 3



Generation 4

Ongoing work

Sequencing and mapping genes

- Currently manual
- Exploring variants of protocol mining

Finding gene instances

- Coccinelle
- Writing semantic patches automatically

Organizing genes

- Feature-oriented gene database