Characterizing Loss of Digital Evidence due to Abstraction Layers

Felix Freiling
Thomas Glanzmann
Hans P. Reiser

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

UNIVERSITÄT PASSAU

March 23, 2017

Agenda

- Loss of Digital Evidence due to Abstraction Layers
- Contributions
- Definitions
- Identifying the Model in Practice
- Conclusions

Abstraction layers

- hierarchically: higher layer is mapped to lower layer
- efficient resource usage from separation of concerns
- details of lower layers are hidden from higher layers
- evidence is usually collected at lower layers to establish authenticity
- ambiguity and the possibility of interpretation errors
- lower layer might be out of reach

Loss of Digital Evidence due to Abstraction Layers

Under which conditions is it possible to reconstruct higher levels of abstraction from data given at lower layers of abstraction?

- actively used, deleted, overwritten, or otherwise unused data
- unable to interpret: lost through abstraction for forensic expert
- abstraction layers and concurrent access
- attribution due to multi tenancy

Related work

- Abstraction layers are used in software and hardware design
- Abstraction transformation functions (Carrier 2006)
- Virtualization poses problems to incident handling and forensics (Grobauer 2010)
- Correct attribution (Birk 2011, Kolb 2012)
- Recovering deleted files and processes from memory (Dykstra 2012)
- Design cloud environment to aid forensic analysis (Zawoad 2016)
- Cloud-native artifacts can only be aquired by physical access (Roussev 2016)

Contributions

- formal model
  - derive conditions for evidence collection
  - block based storage; might be fragmented
  - reconstruction of active and inactive data at lower layers
  - conditions are only partly satisfied by real data structures therefore data is lost through abstraction
  - covers file systems, virtual memory, guest physical memory, logical volumes, RAID
- API-based storage abstractions like network file systems are excluded because they hide state information

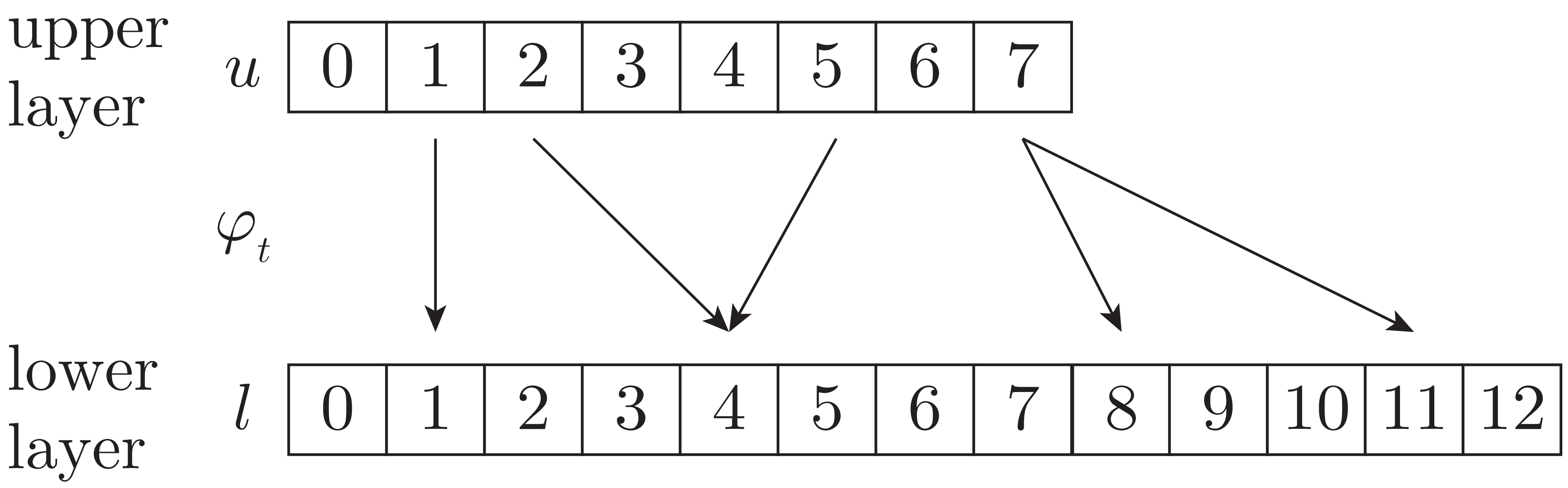
Layers

- Single level of abstraction considered at a time
- Each level has two layers: upper and lower
- Each layer provides computation and storage resources
- Upper layer is implemented by using resource from the lower layer

Storage

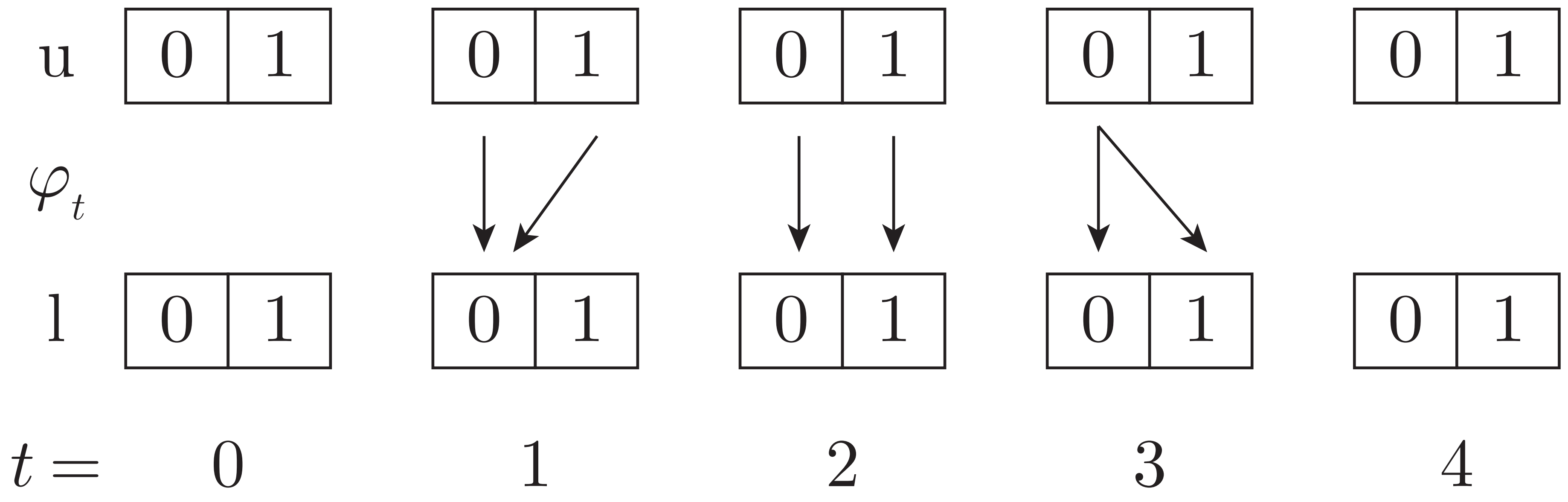
- block-based storage abstraction
- storage can be either memory or disk
- both layers consist of a finite ordered block sequence
- block sizes on upper and lower layer can differ
- storage block is identified by a unique index

Mappings



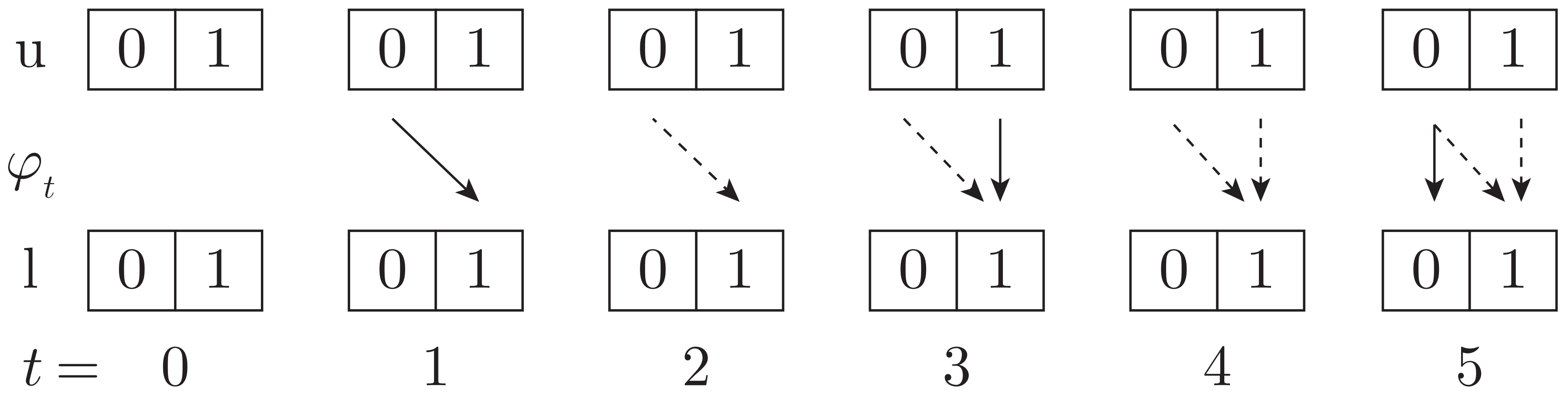
- management data structure

Mappings over time



- Mappings are dynamic and change over time
- If a previous mapping is removed we call this **unmapped**
- If a mapping is added we call this **mapped**
- Unmapped blocks are of special interest for digital forensics

Most Recently Unmapped Blocks



- most recent deletion time
- might still be possible to interpret and recover content of the lower layer in the context of the upper layer

Mapping Aware vs Mapping Agnostic Lower Layers

- Unmapped lower layer block irrelevant for high layer, but might contain evidence and can be reused
- UNMAP and TRIM
- Mapping aware vs. mapping agnostic lower layers

Evidence Collection

- Depending on access level evidence can be collected at upper or lower layer
- Forensic evidence collectors strive for lowest layer necessary to capture all relevant information
- bit-by-bit copy

Forensic Reconstruction Problem

- active mappings
- deleted mappings
- we define three increasingly powerful notions of reconstruction

Finding and Decoding φ

- Finding and identifying management data structure

- Fully understand φ often due to timeconsuming reverse engineering

Definition 1: Active Mappings Reconstruction

Given a copy e of l at time t, solving *active mappings reconstruction* means to

- find and decode φ_t from e and
- enumerate all elements of φ_t.

Definition 2: Deleted Mappings Reconstruction

Given a copy e of l at time t, solving *deleted mappings reconstruction* means to

- find and decode φ_t from e,
- enumerate all elements of φ_t, and
- enumerate a non-trivial subset $\triangle$ of all deleted mappings of φ_t, i.e., the empty set is only allowed as a response if deleted mappings do not exist in φ_t.

Definition 3: Last Deleted Mappings Reconstruction

Given a copy e of l at time t, solving *last deleted mappings reconstruction* means to

- find and decode φ_t from e,
- enumerate all elements of φ_t,
- enumerate a non-trivial subset of all deleted mappings of φ_t, and
- to enumerate the most recently deleted mappings from that subset.

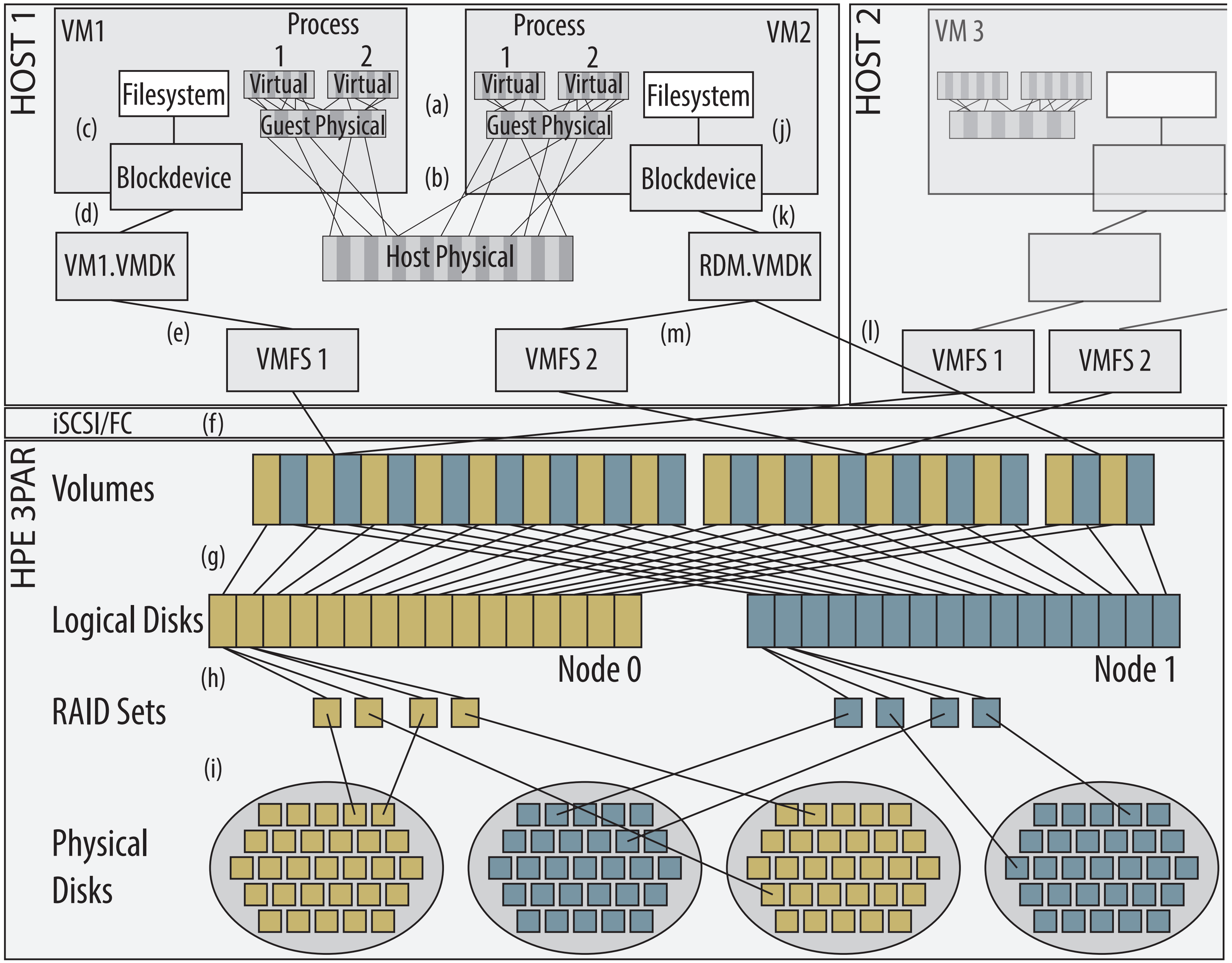
Last Deleted Mappings Reconstruction Dilemma

A method that solves Deleted Mappings Reconstruction could “intentionally” only return deleted pointers that were not deleted last, thus avoiding extra work. It could then return an empty set as fourth element and therefore trivially also solve Last Deleted Mapping Reconstruction. Therefore currently, Last Deleted Mappings Reconstruction is *not* strictly stronger than Deleted Mappings Reconstruction. At present, it is not clear to us how to formulate an appropriate non-triviality condition to make it strictly stronger.

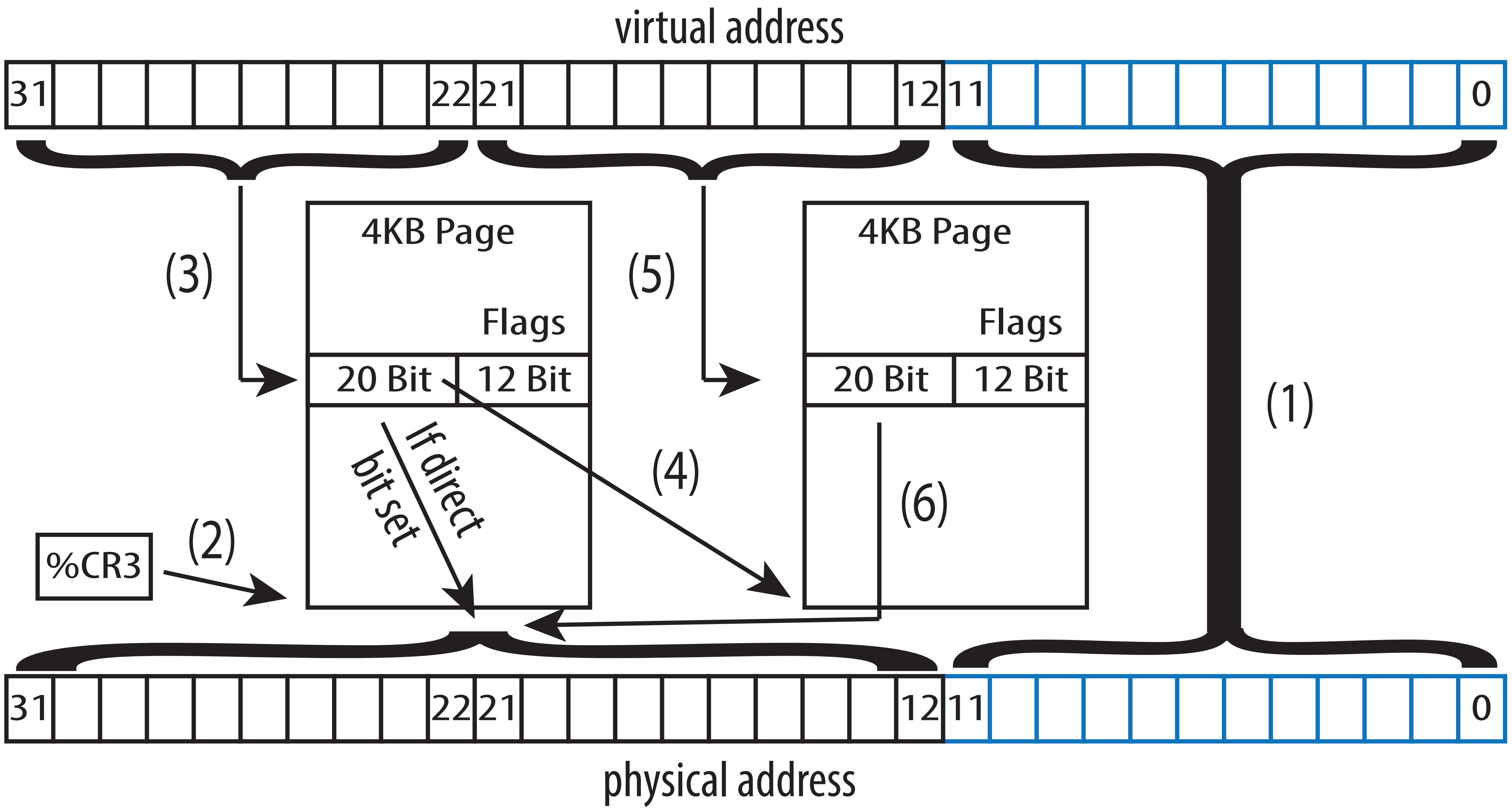
Identifying the Model in Practice

- Focus in particular abstraction layer and concrete implementation
- Identification of upper and lower layer
- Name and collect evidence
- Backups, memory and disk snapshots are excluded
- Focus on “pure” management mechanisms
- Difference between backups and “pure” management mechanisms blurred.

Abstraction Layers in Practice



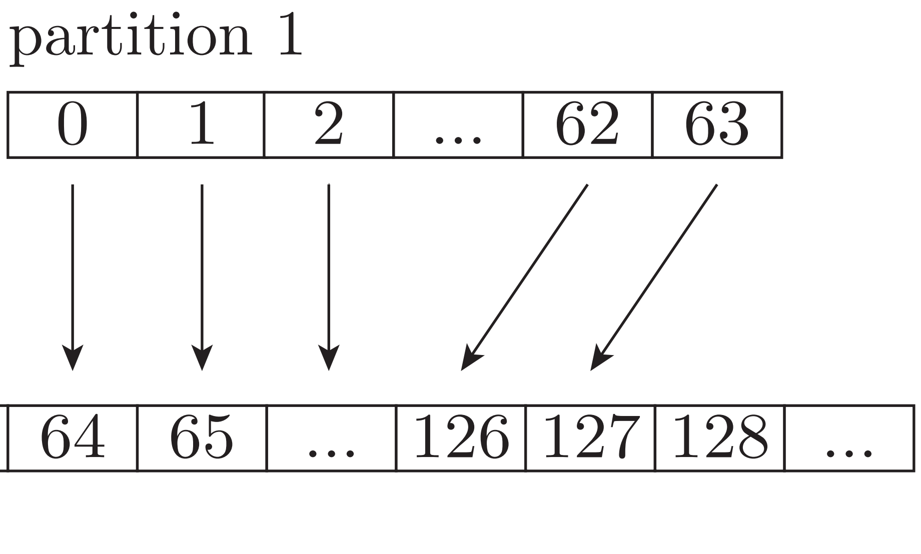
Virtual Memory Management in Linux



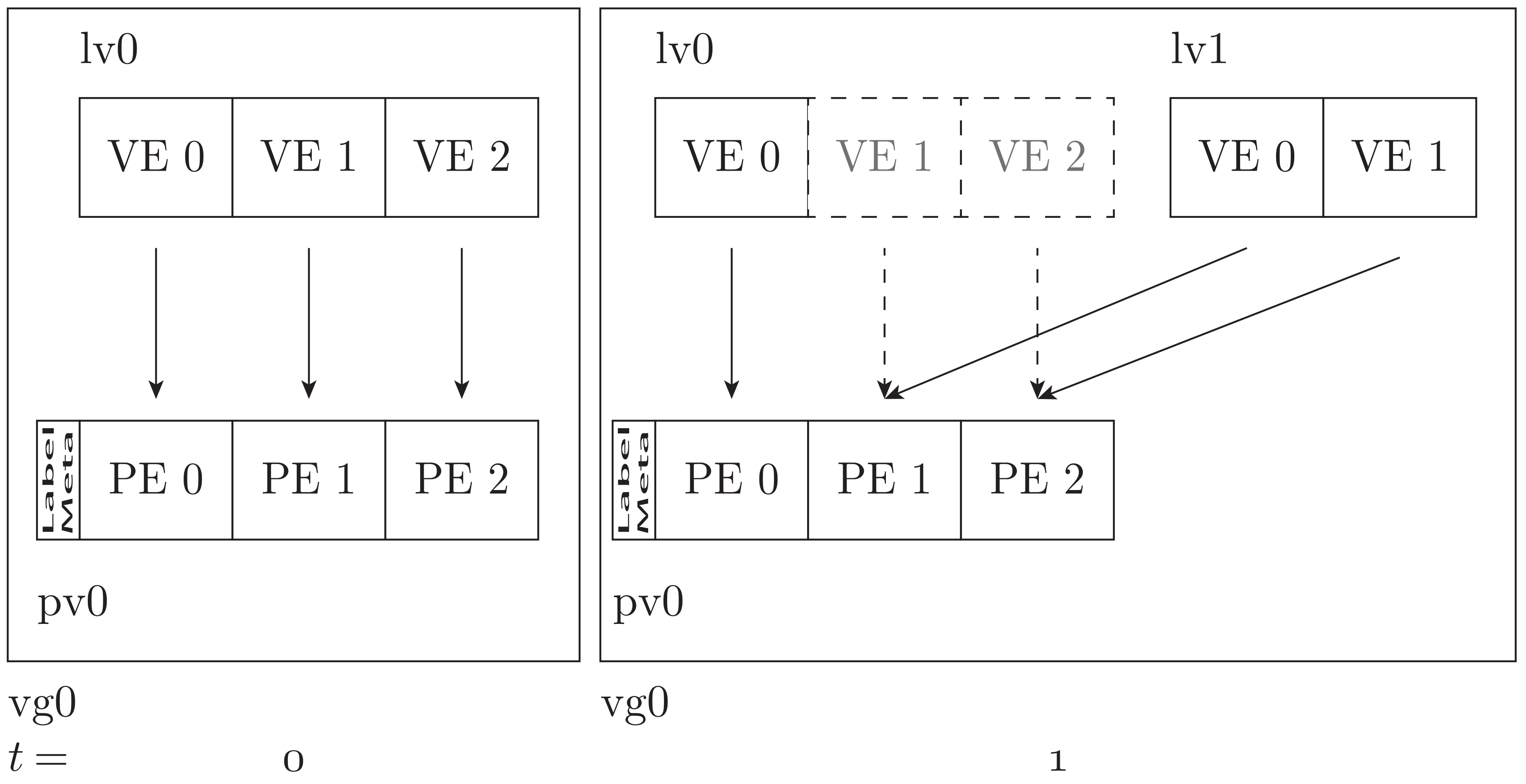
DOS/MBR Partition Management

partition table

partition	start	stop
#1	64	64
⋮		



Logical Volume Management



LVM2 Text Format Volume Group Metadata

```
creation_time = 1483598692

vg0 {
 id = "Ubd6dH-...-Lmhjb0"
 seqno = 2
...
 extent_size = 8192 # 4 Megabytes
...
 physical_volumes {
  pv0 {
   id = "kEeQij-...-BanVIt"
   device = "/dev/loop0" # Hint only
...

   dev_size = 26624 # 13 Megabytes
   pe_start = 2048
   pe_count = 3    # 12 Megabytes
 }
}

 logical_volumes {
  lv0 {
   id = "bihcBw-...-z51XDO"
...
   creation_time = 1483598689
   segment_count = 1
   segment1 {
    start_extent = 0
    extent_count = 3 # 12 Megabytes
    type = "striped"
    stripe_count = 1 # linear
    stripes = [
     "pv0", 0
    ]
   }
  }
 }
}
```

Conclusions

- We investigated the effect of abstraction layers on forensic evidence collection.
- Evidence collection on higher layers is inferior to lower layers collection (Roussev 2016)
- Semantic gap (Birk 2011)
- Logical access restricts evidence for mapping-aware systems (Nisbet 2013)
- Evidence collection using APIs improves (Roussev 2016)
- On mapping agnostic systems deleted mappings can often be recovered unless over-written
- Model does not fit abstractions by file systems due to hierarchy which does not fit our “flat” model
- Investigate compositionality in the future.