Dynamic VM Monitoring using Hypervisor Probes

Z. J. Estrada, C. Pham, F. Deng, L. Yan,
Z. Kalbarczyk, R. K. Iyer

European Dependable Computing Conference
2015-09-09

Dynamic VM Monitoring

Goal

On-demand VM Monitoring to reduce the effort required to harden computing systems against failures and attacks.

- ✓ Uptime requirements
- ✓ Effort required
- ✓ QA concerns
- ✓ Lack of knowledge

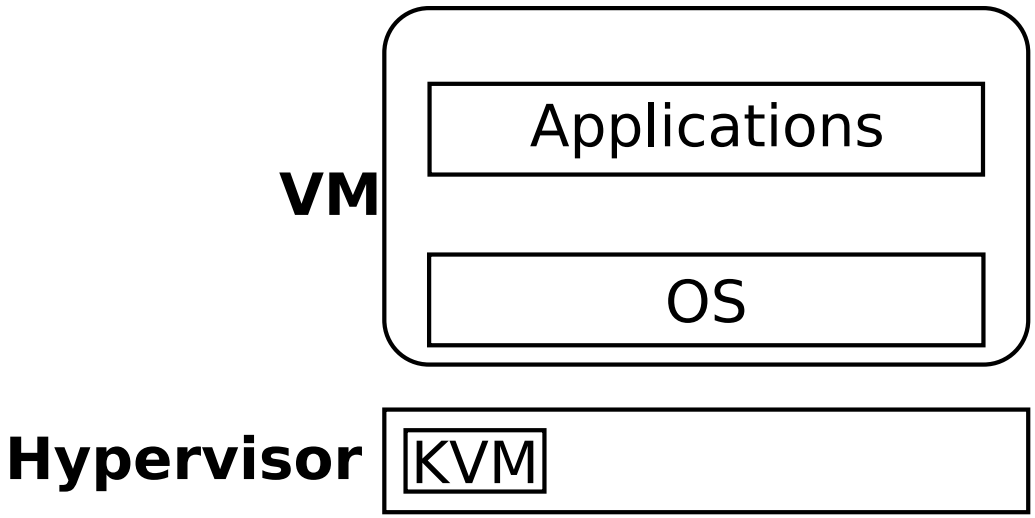
VM Monitoring

Reliability & Security Monitoring

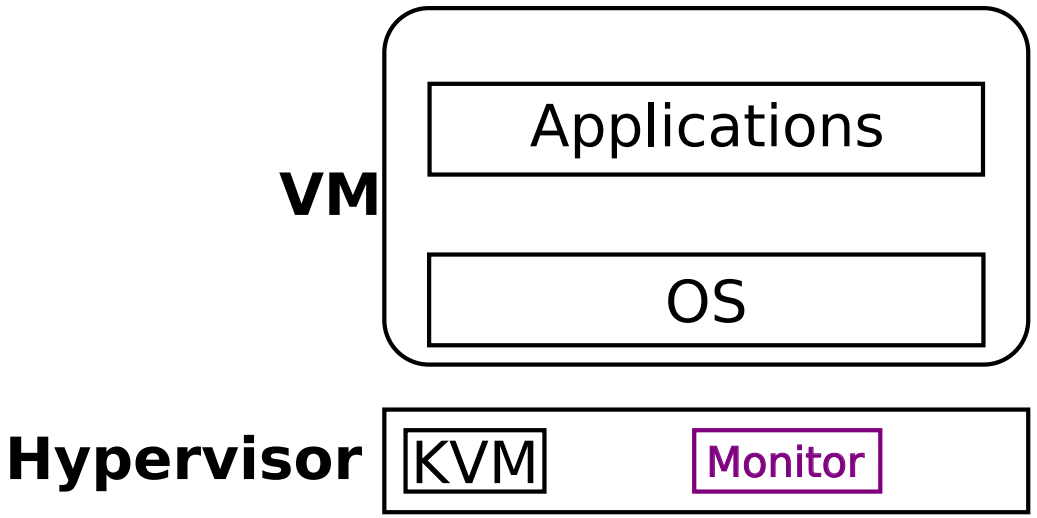
Recording and analyzing a computer system to detect failures and attacks.

- Passive - polling based
- Active - event based

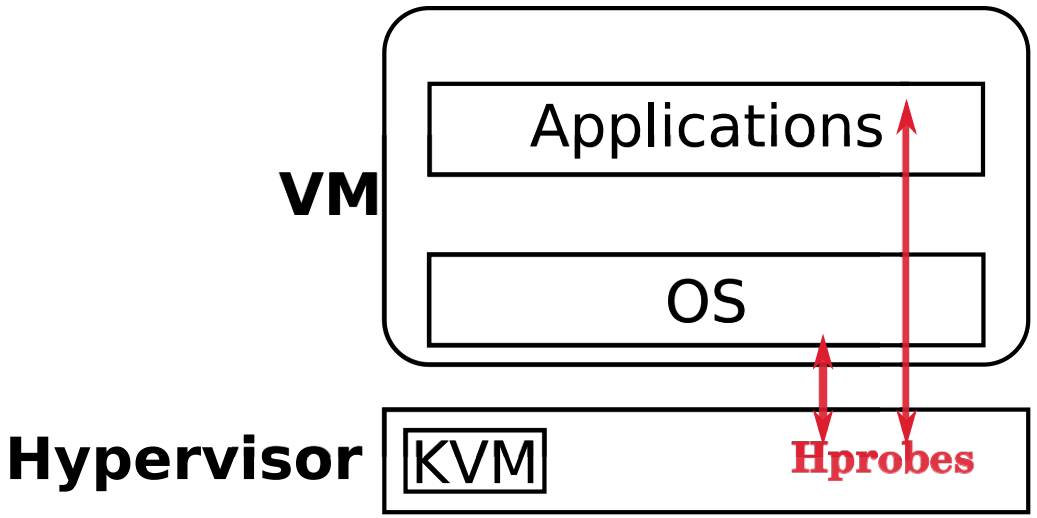
VM Monitoring



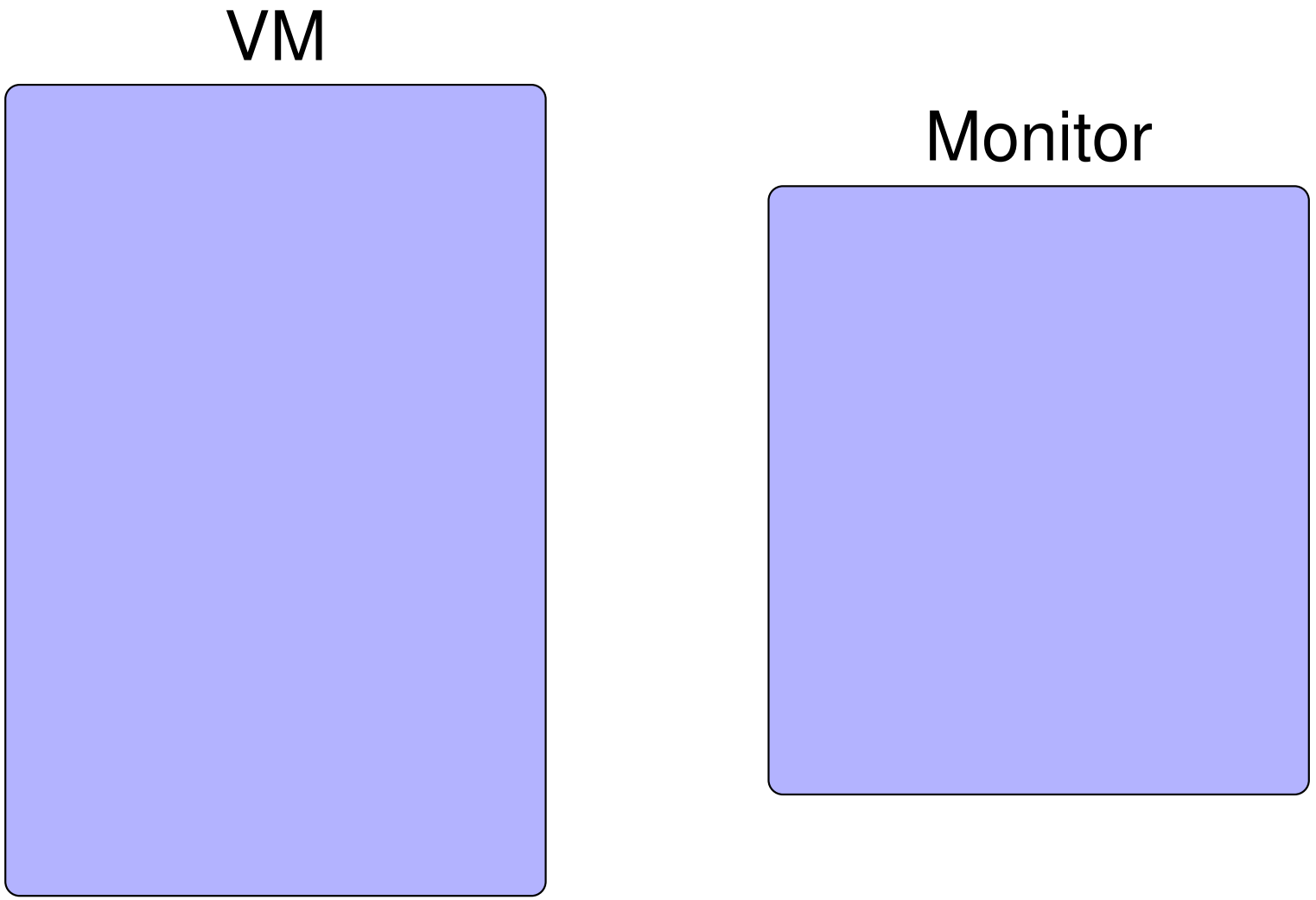
VM Monitoring



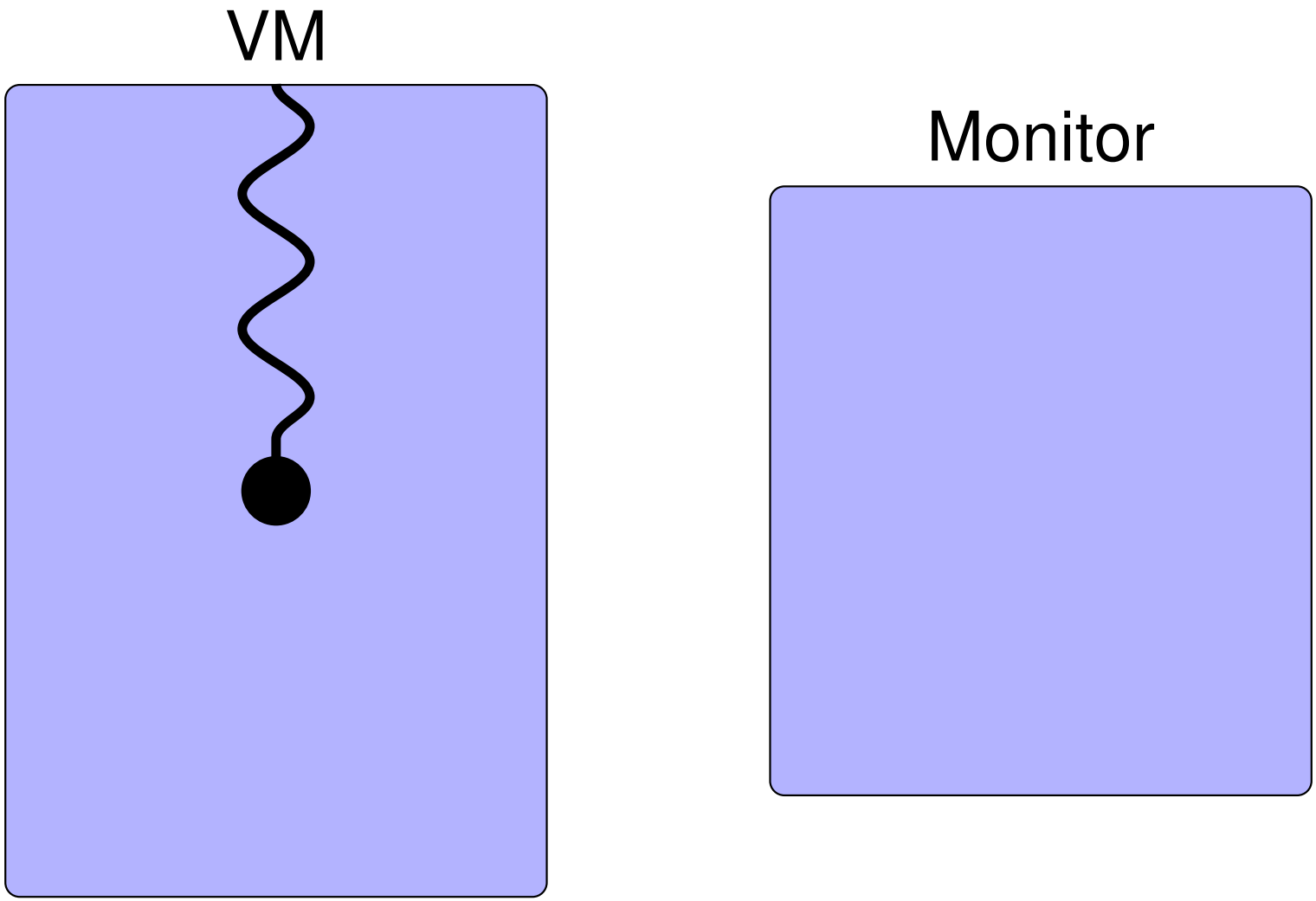
VM Monitoring



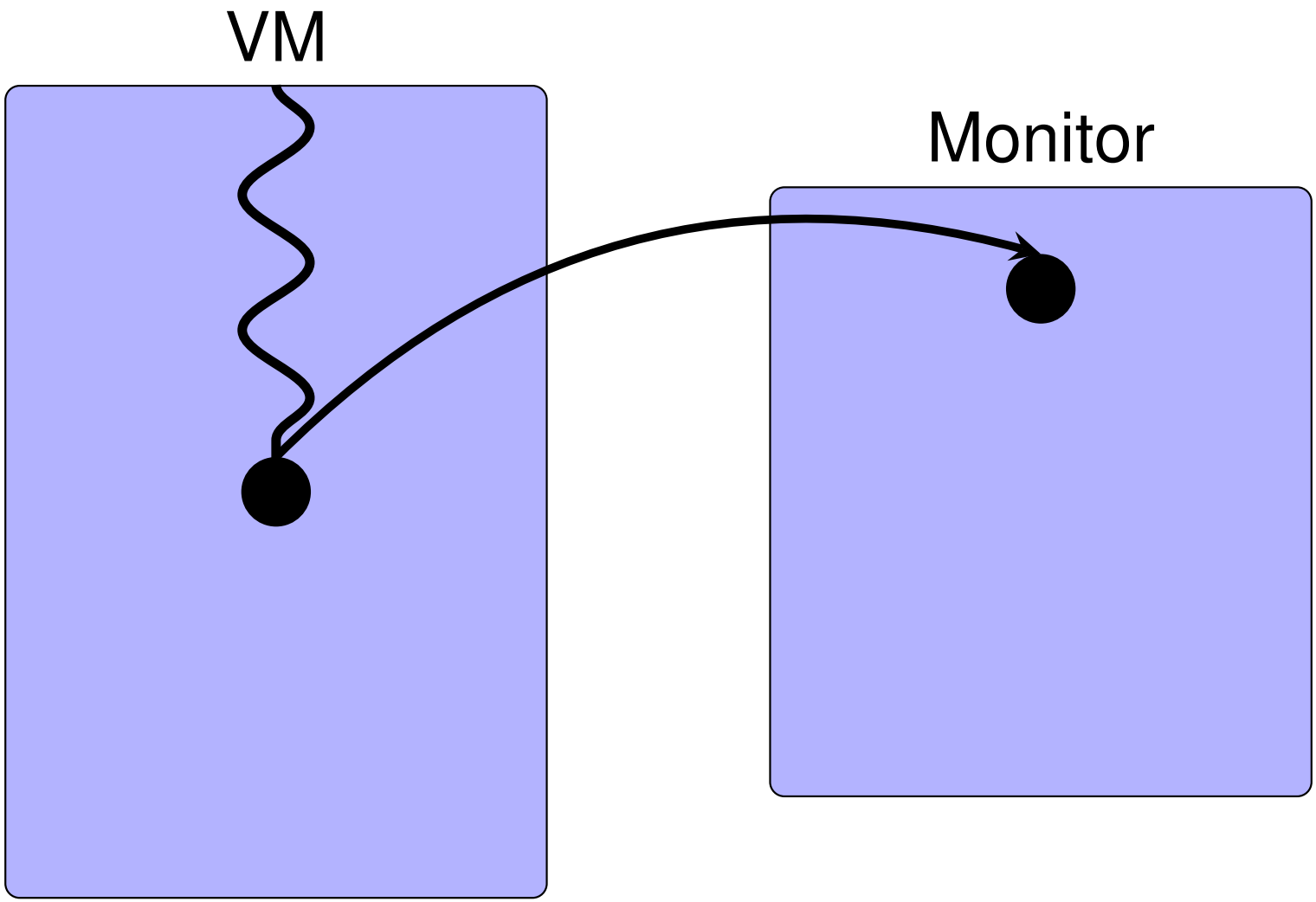
- *Hook based*
- *On-Demand - Add/Removal at Runtime*
- *Vulnerability, Hang, and Infinite Loop Detectors*
- *Userspace support*



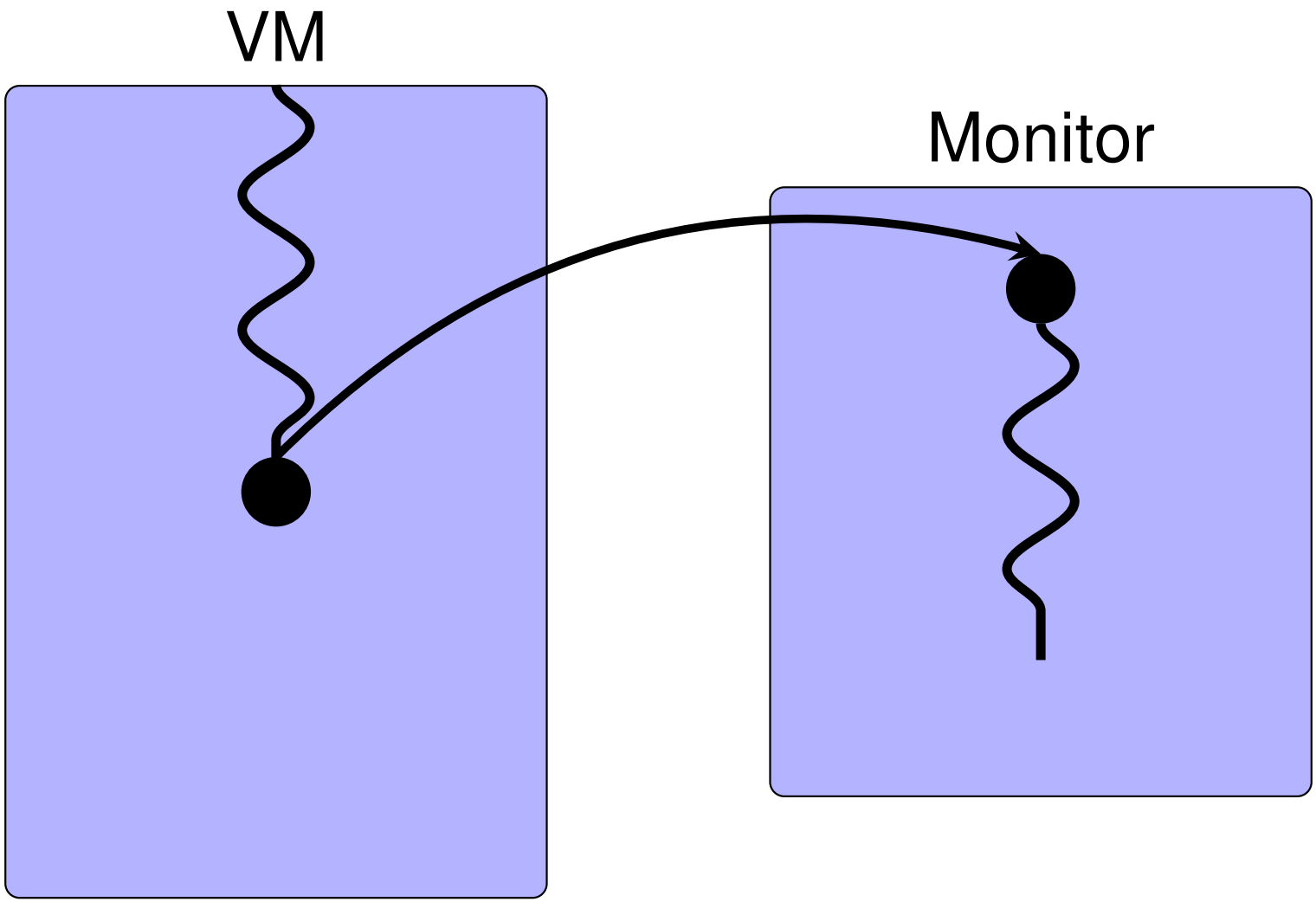
Monitor is running inside the hypervisor



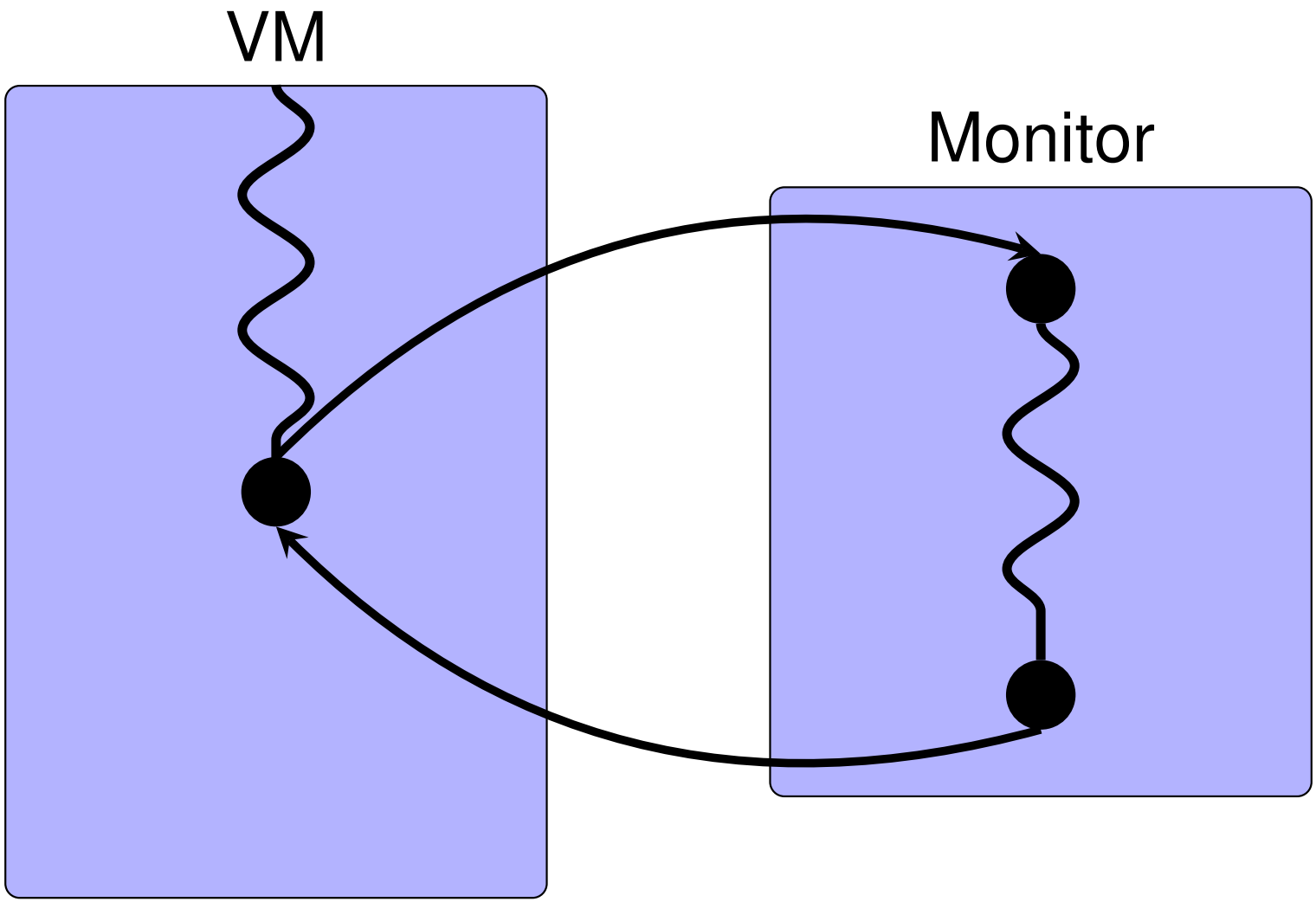
VM execution reaches a hook



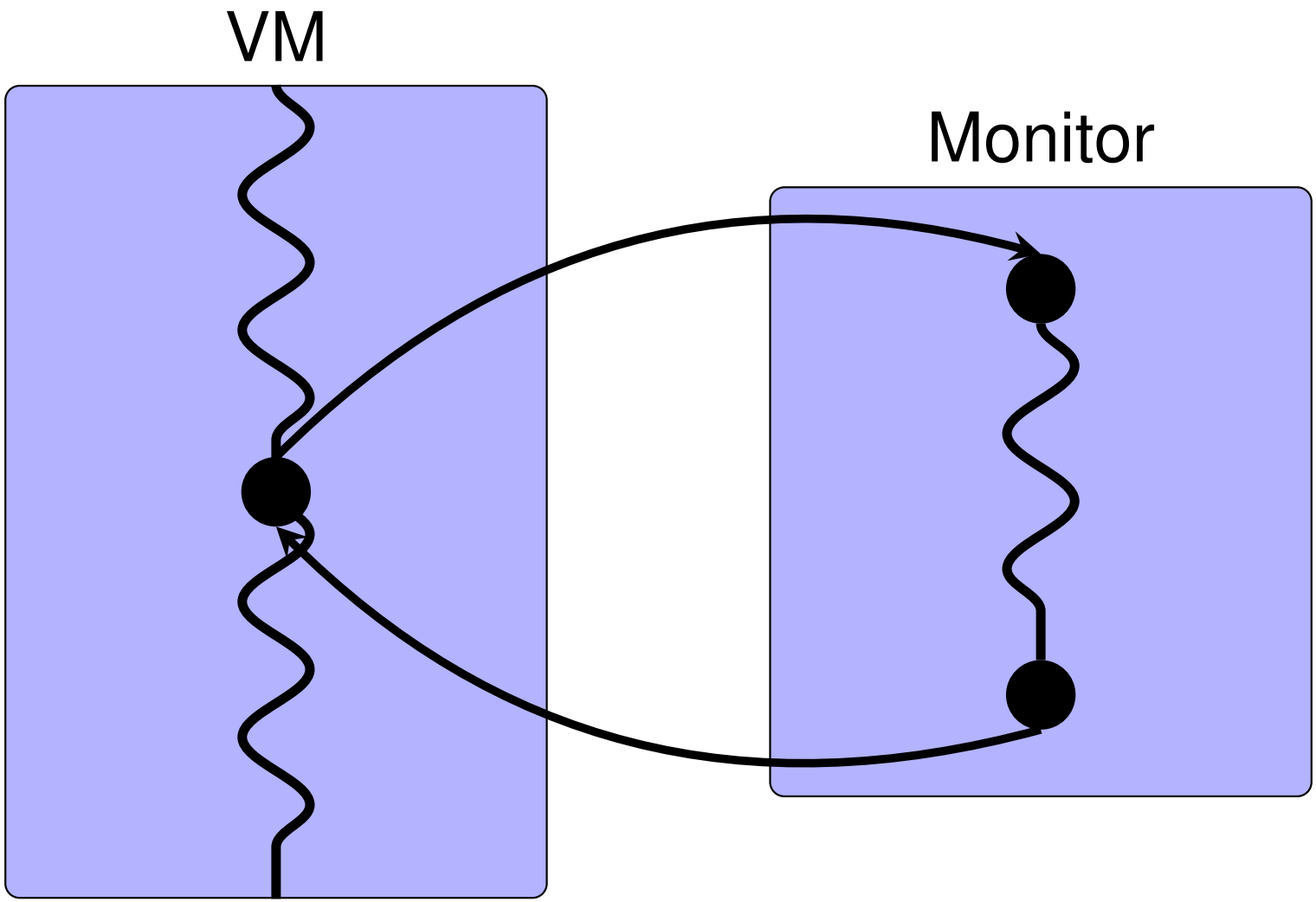
Control is transferred to the monitor



The monitor performs its monitoring function



Control is transferred back to the VM



The VM resumes normal execution

Hook-Based VM Monitoring

Previous techniques:

- \+ Active monitoring
- \+ Protected hooks
- – Guest OS only - no userspace
- – Not dynamic - boot time config
- – Require guest OS modifications

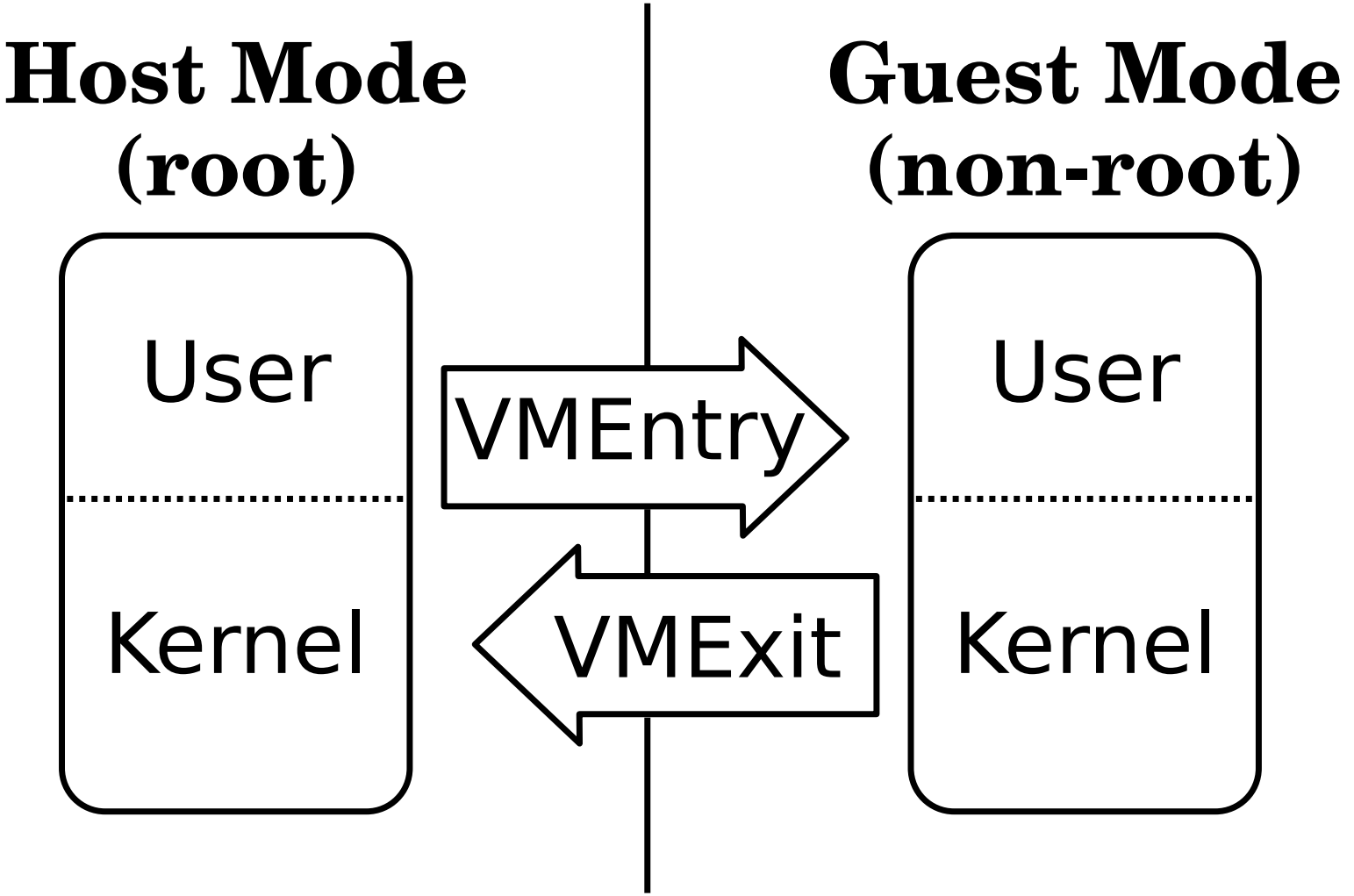
Goals

Hook-based monitoring should:

+ be protected from attacks in the VM
+ be simple to use
+ not require guest OS modification
+ be runtime adaptable
+ allow for arbitrary hook placement

Hypervisor Probes

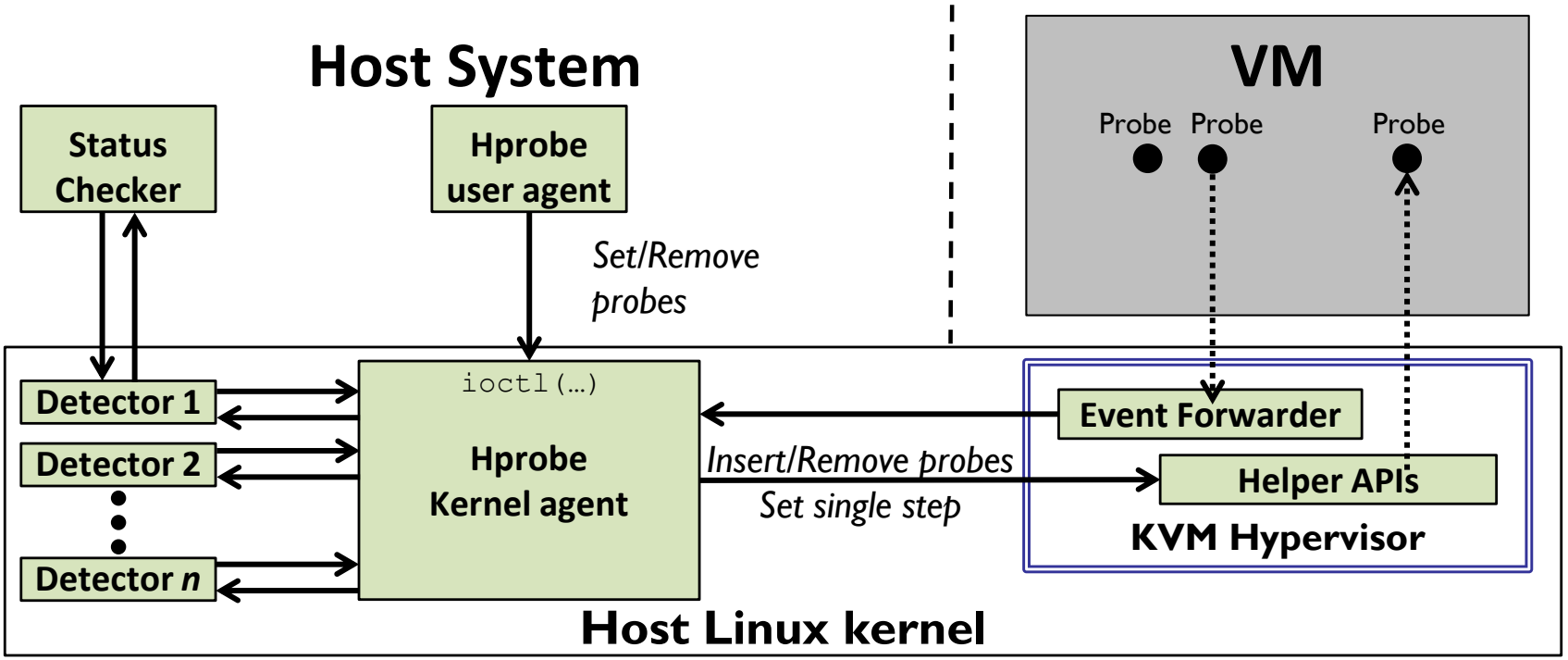
Hardware Assisted Virt.



Hypervisor Probes

- Event on guest execution
  - Event transfers control to hypervisor (VM Exit)
  - Perform monitoring after that event
- Hooks added/removed at runtime
- Monitors applications and the guest OS

Hprobe Architecture

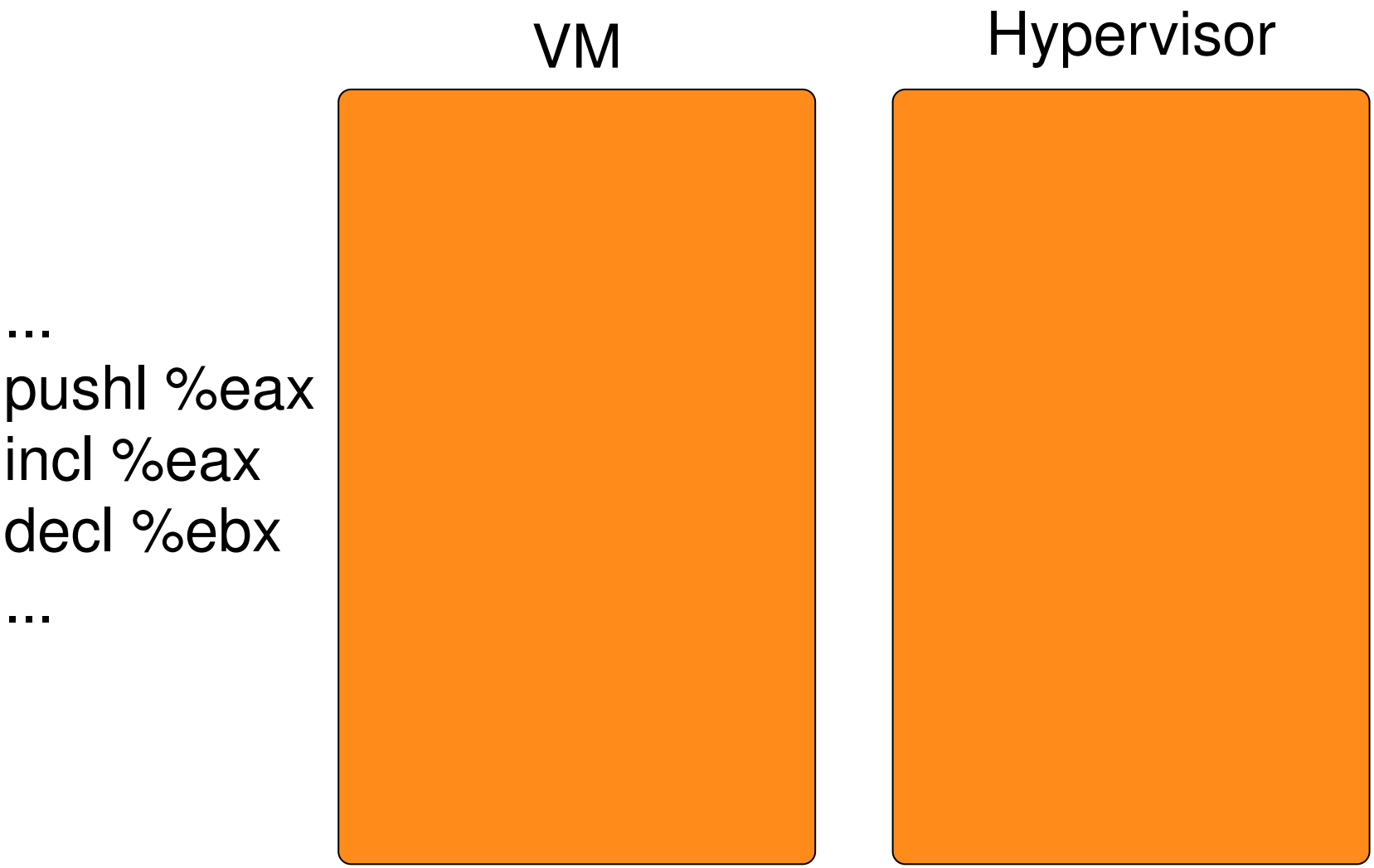


Hprobes API

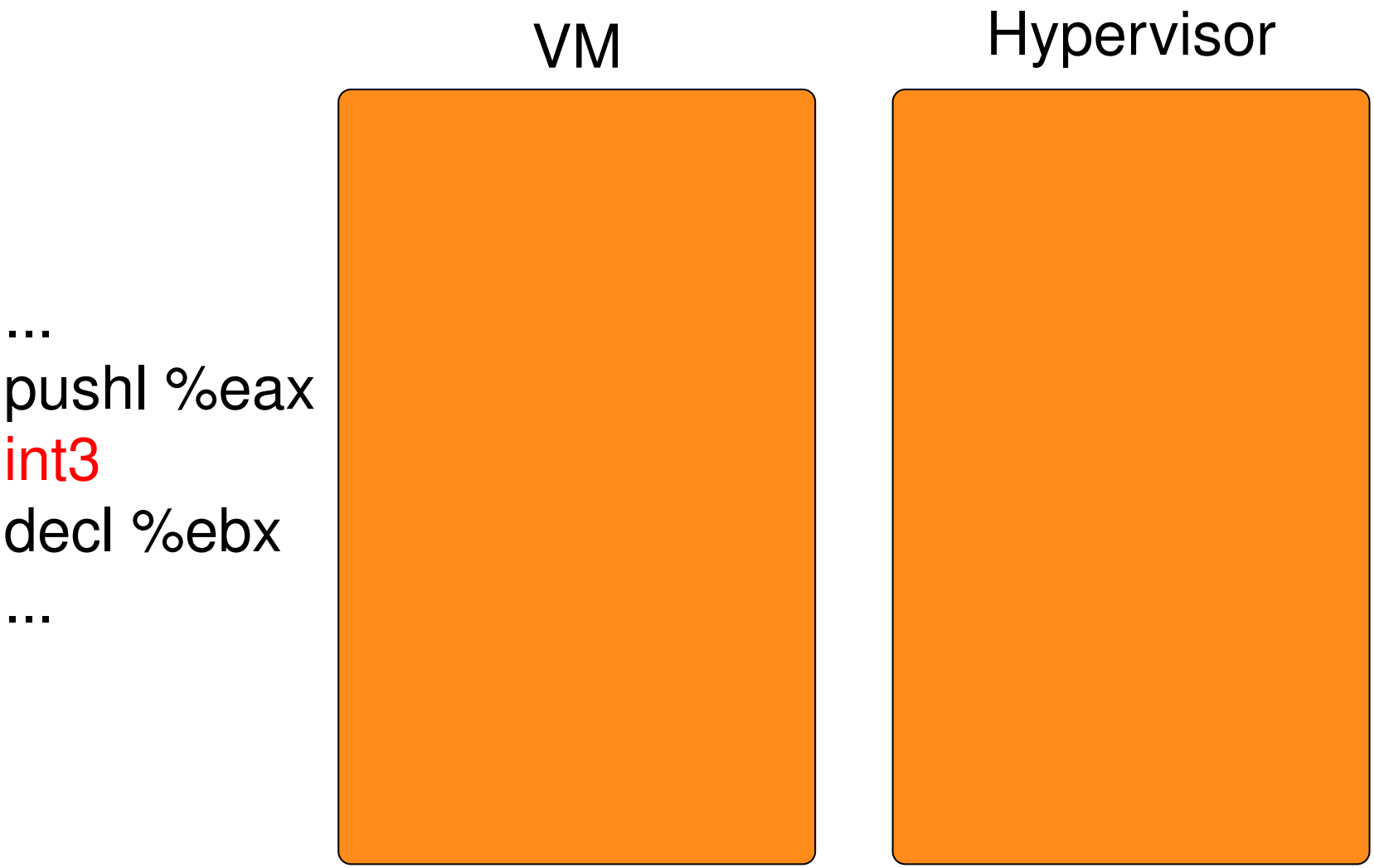
```
int HPROBE_add_probe( );
int HPROBE_remove_probe( );
```

- `addr_info`: gva+cr3
- `vmid`: unique id for VM
- `vcpu_type`: vcpu state

Probe ⇒ Event Forwarder

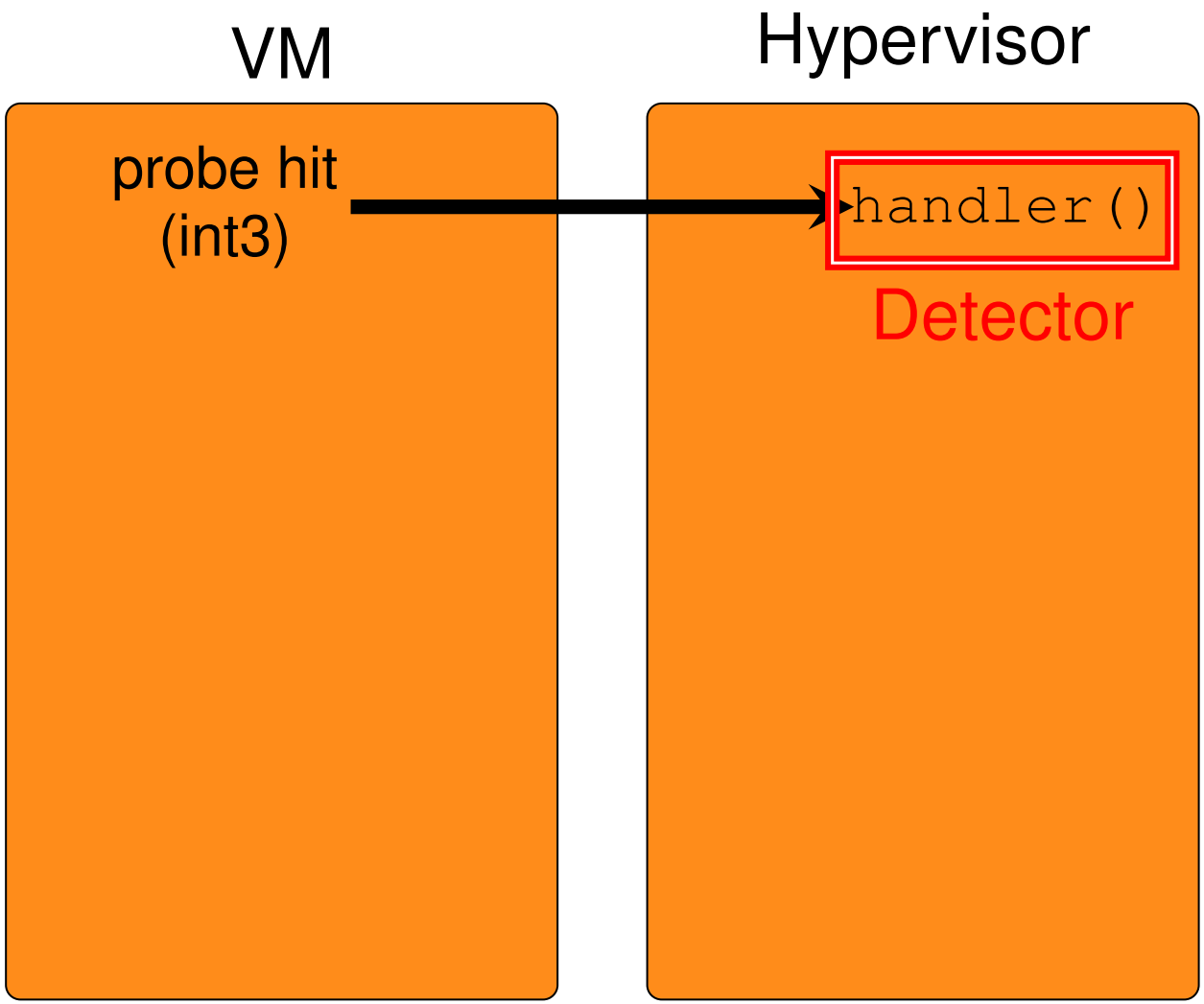


Probe ⇒ Event Forwarder



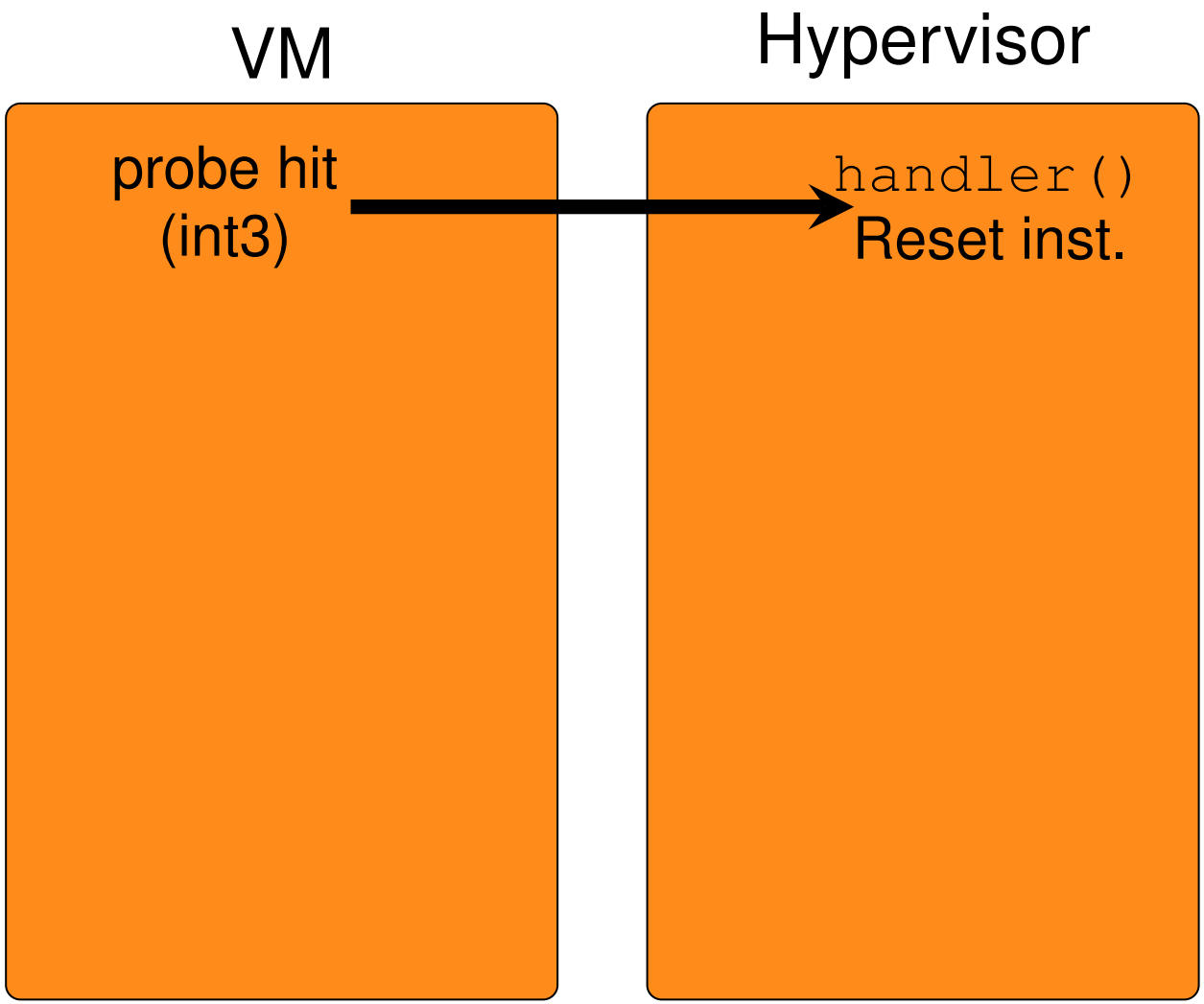
Probe ⇒ Event Forwarder

```
...
pushl %eax
int3
decl %ebx
...
```



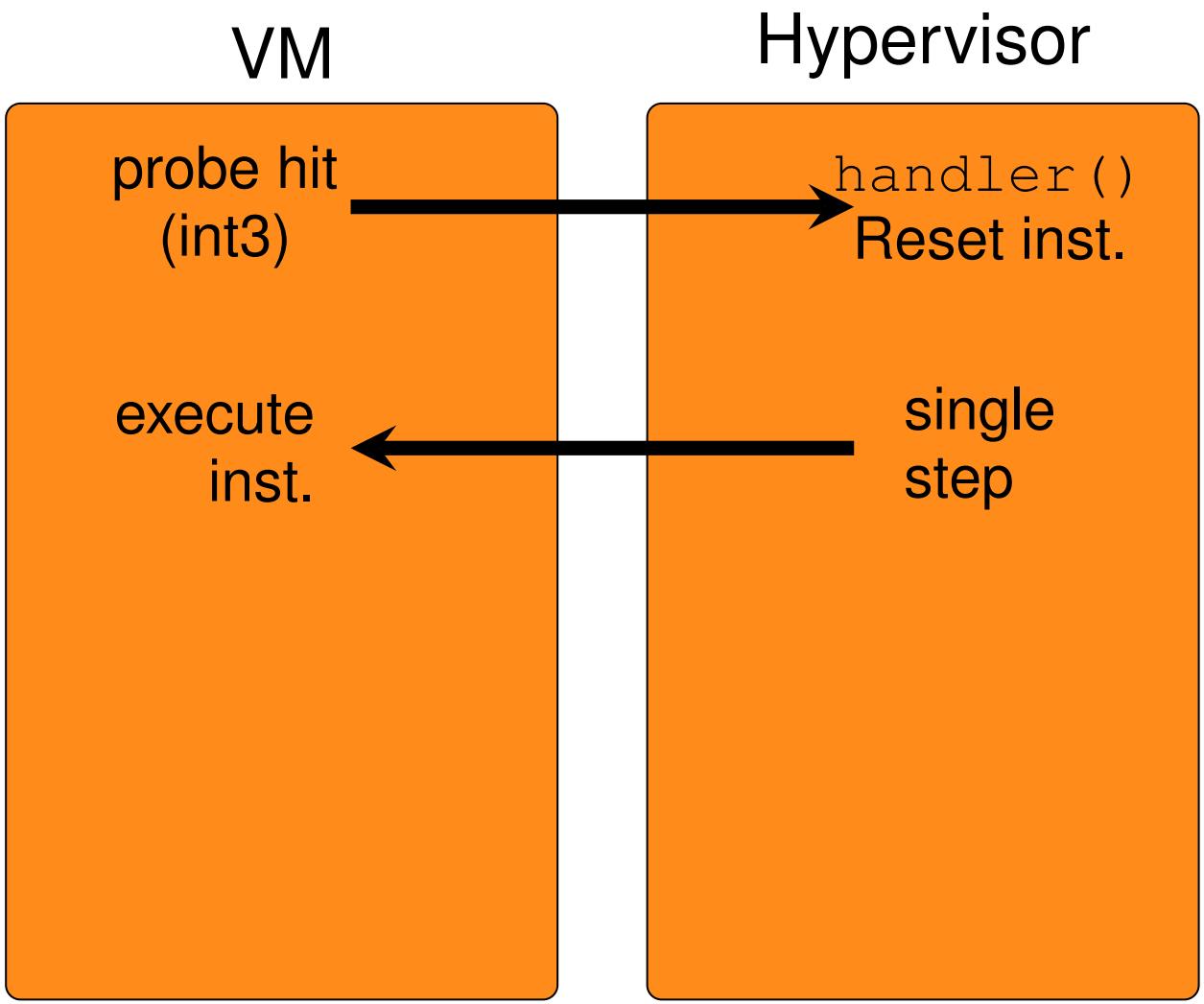
Probe ⇒ Event Forwarder

```
...
pushl %eax
incl %eax
decl %ebx
...
```



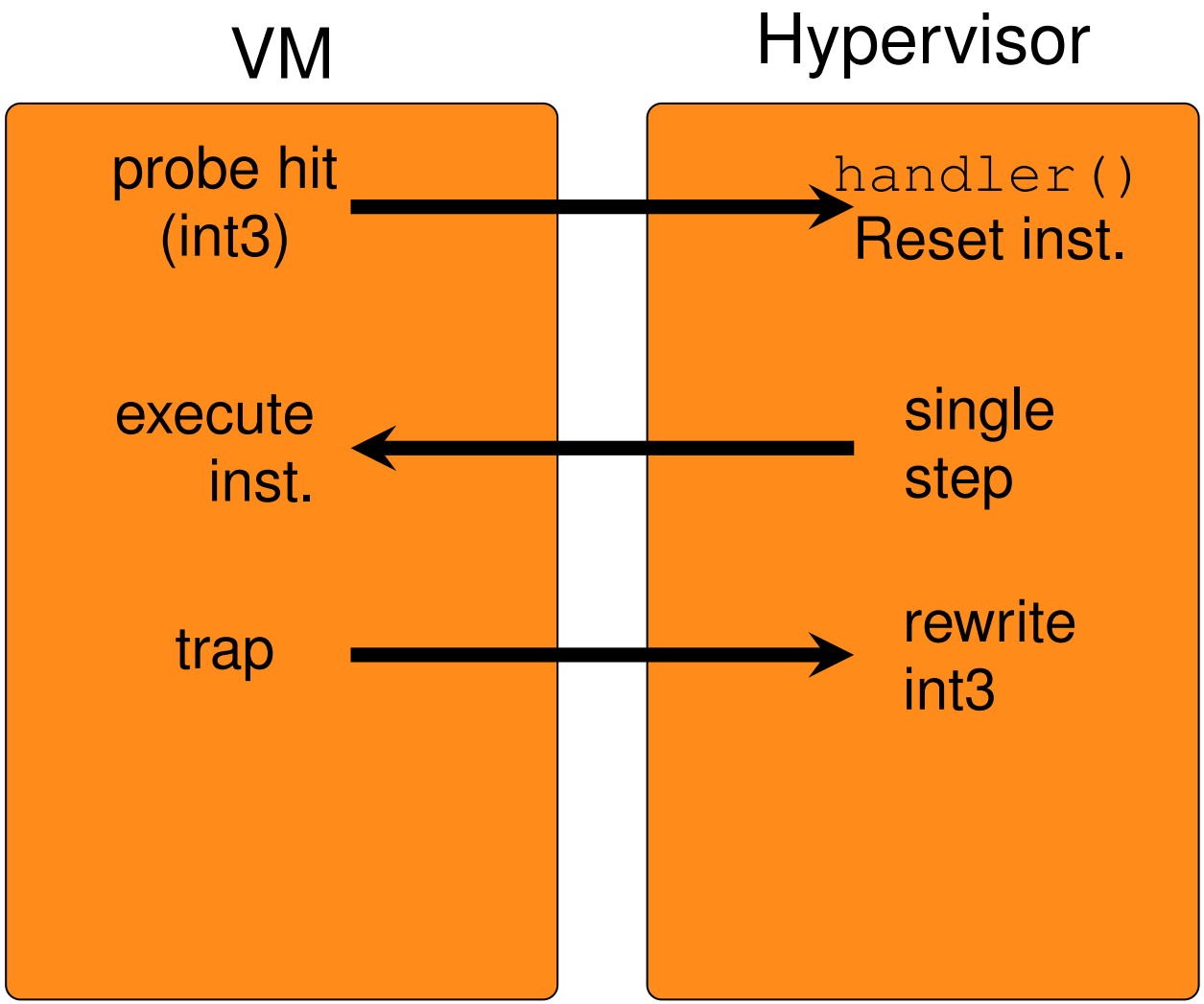
Probe ⇒ Event Forwarder

```
...
pushl %eax
incl %eax
decl %ebx
...
```



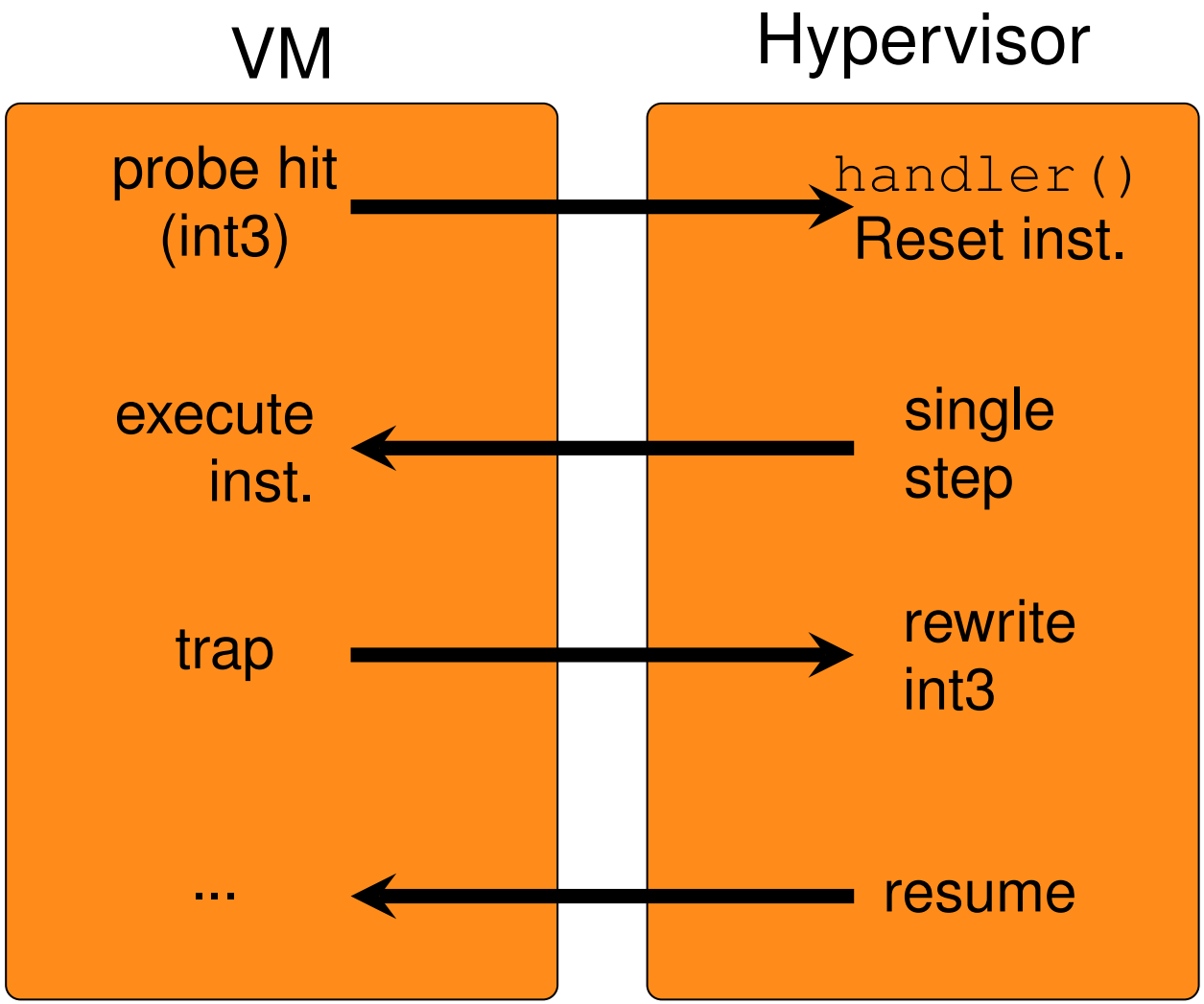
Probe ⇒ Event Forwarder

```
...
pushl %eax
int3
decl %ebx
...
```

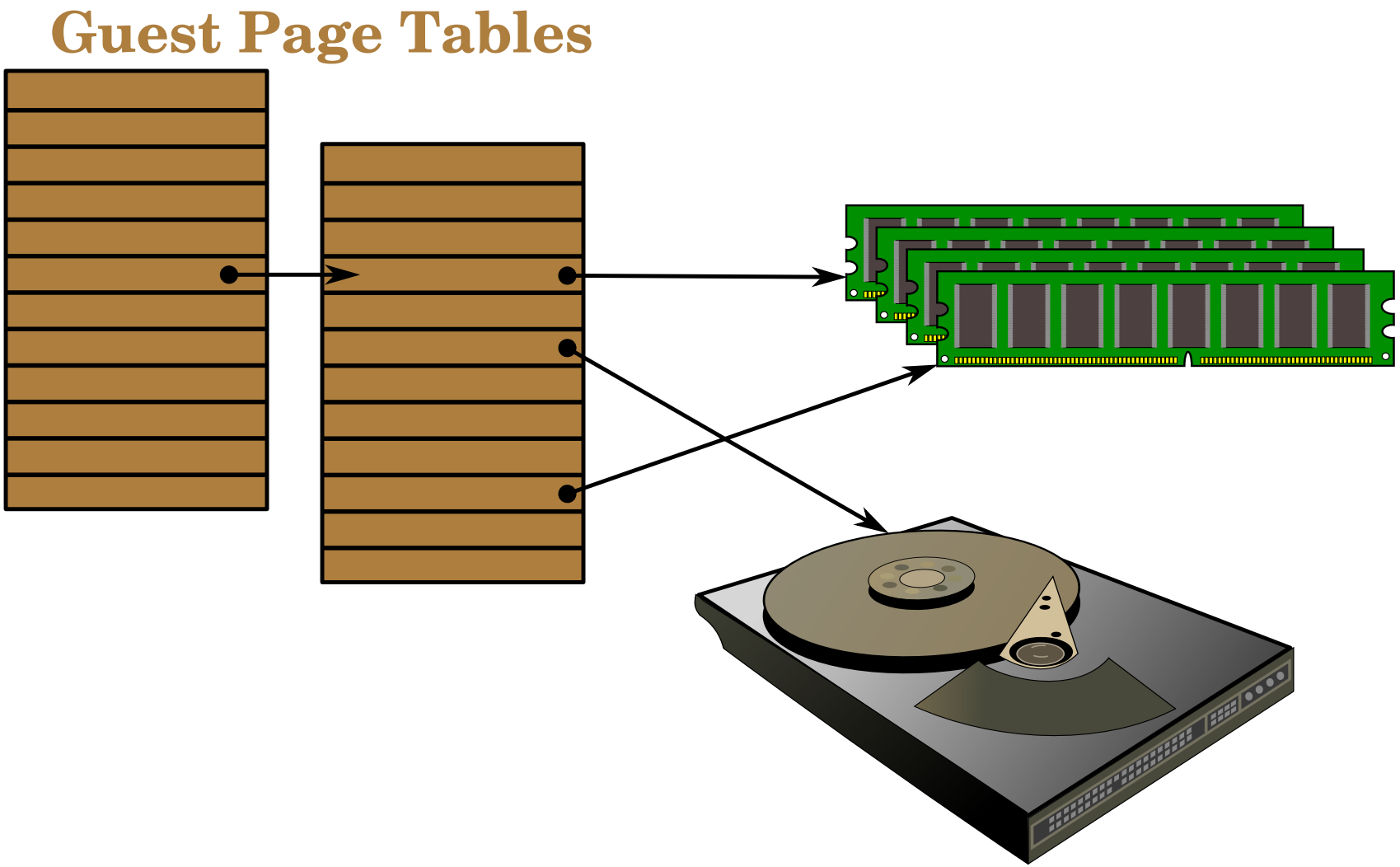


Probe ⇒ Event Forwarder

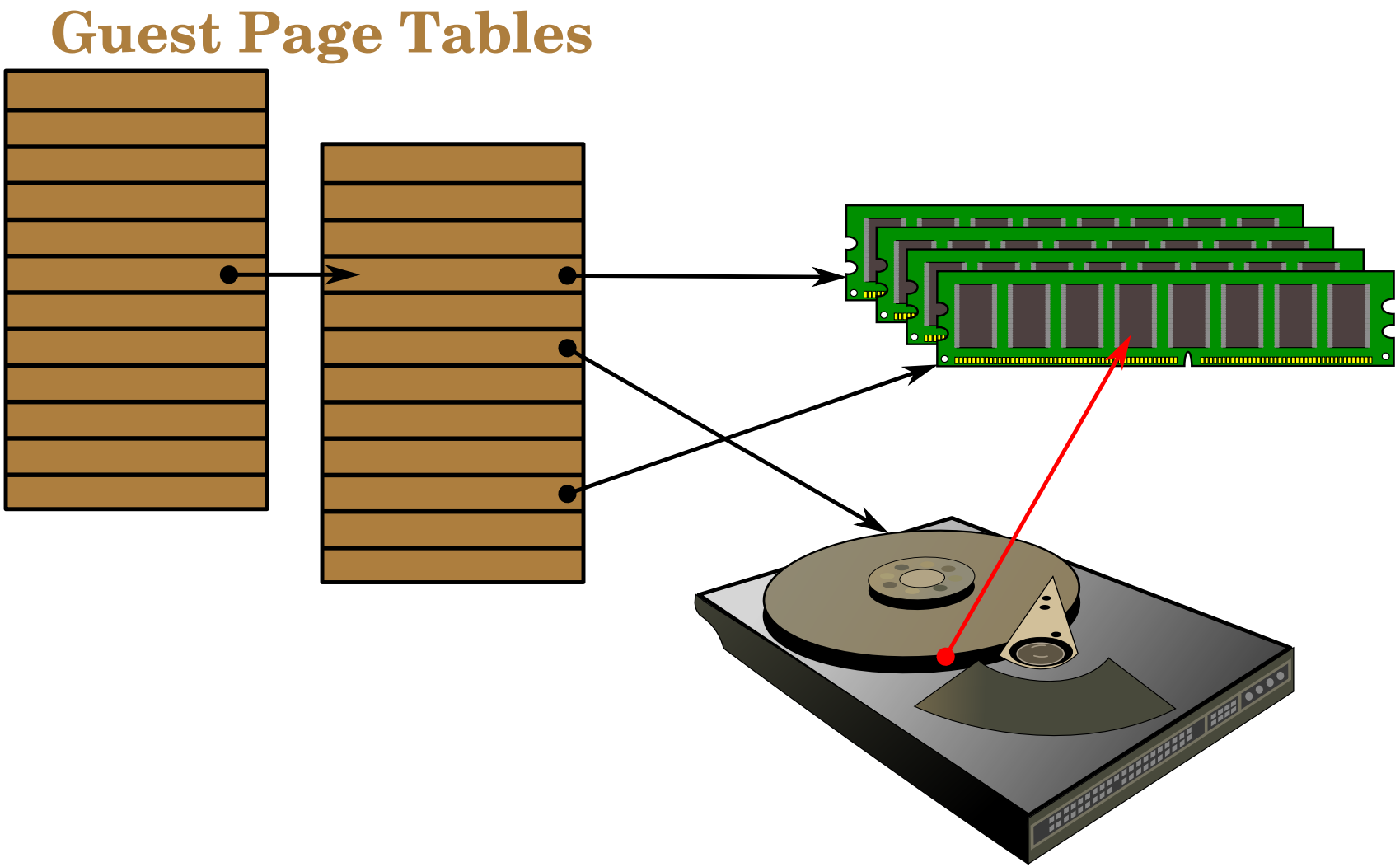
```
...
pushl %eax
int3
decl %ebx
...
```



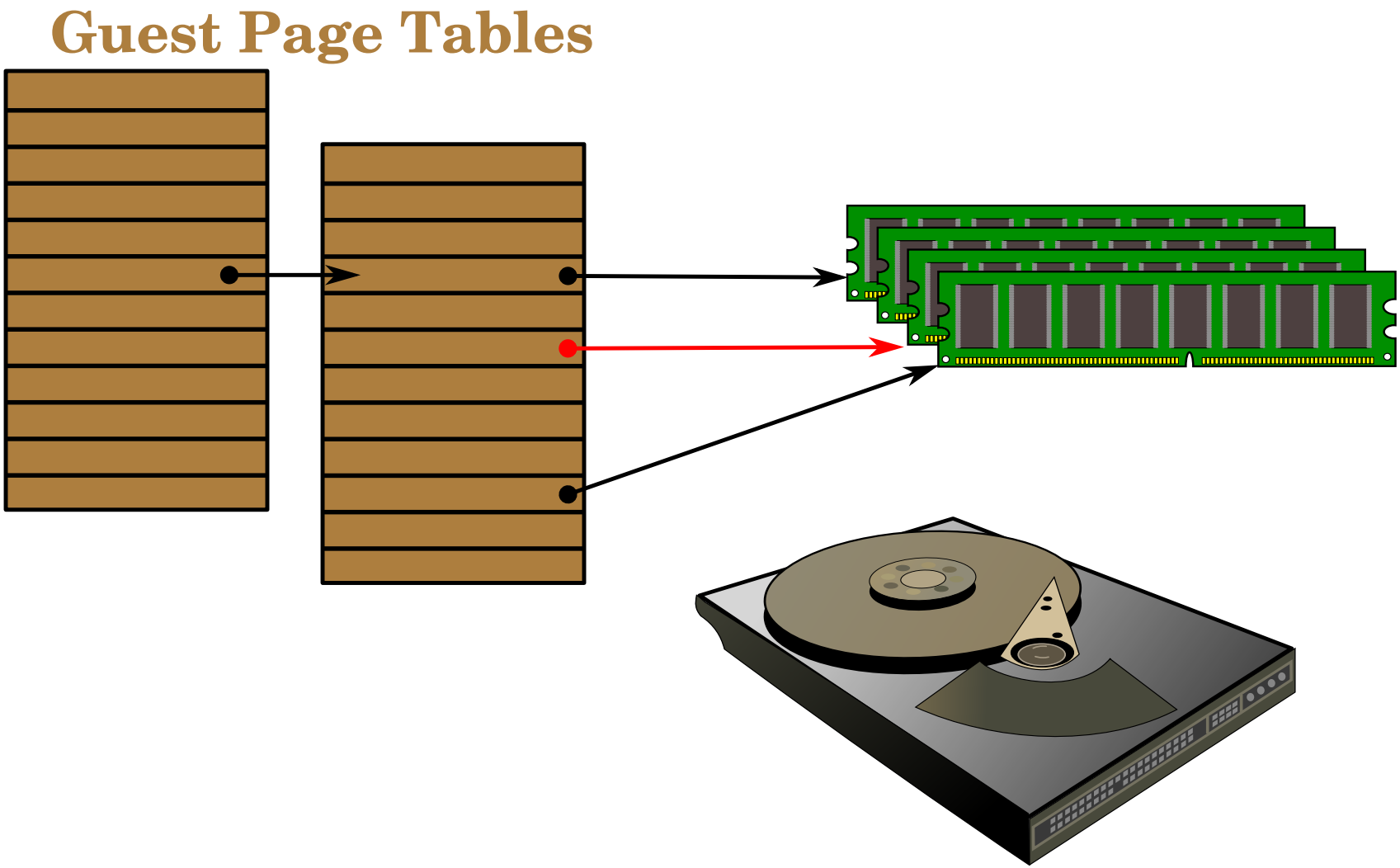
Userspace Probe Challenge



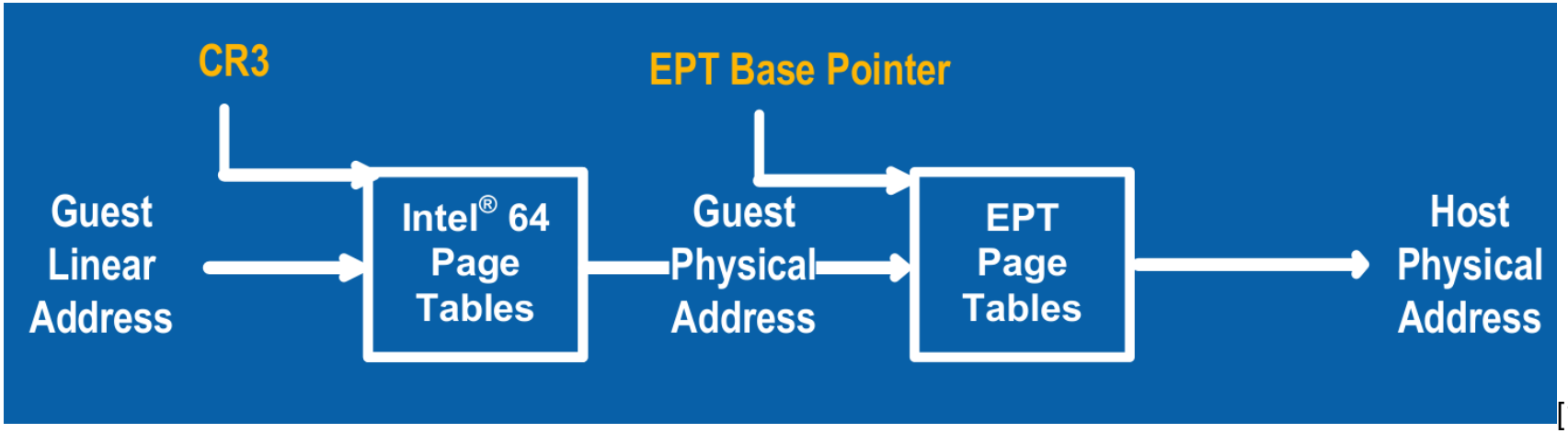
Userspace Probe Challenge



Userspace Probe Challenge



Extended Page Tables (EPT)



[1]

- Guest OS has full control over PTs
- 2nd set of HW PTs for GPA→HPA
- Use EPT to write-protect Guest Page Table

[1] http://www-archive.xenproject.org/files/xensummit_4/VT_roadmap_d_Nakajima.pdf

Goals

Hook-based monitoring should:

+ be protected from attacks in the VM
+ be simple to use
+ not require guest OS modification
+ be runtime adaptable
+ allow for arbitrary hook placement

Goals

Hook-based monitoring should:

- ✓ be protected from attacks in the VM
- + be simple to use
- + not require guest OS modification
- + be runtime adaptable
- + allow for arbitrary hook placement

Goals

Hook-based monitoring should:

- ✓ be protected from attacks in the VM
- ✓ be simple to use
- + not require guest OS modification
- + be runtime adaptable
- + allow for arbitrary hook placement

Goals

Hook-based monitoring should:

- ✓ be protected from attacks in the VM
- ✓ be simple to use
- ✓ not require guest OS modification
- + be runtime adaptable
- + allow for arbitrary hook placement

Goals

Hook-based monitoring should:

- ✓ be protected from attacks in the VM
- ✓ be simple to use
- ✓ not require guest OS modification
- ✓ be runtime adaptable
- + allow for arbitrary hook placement

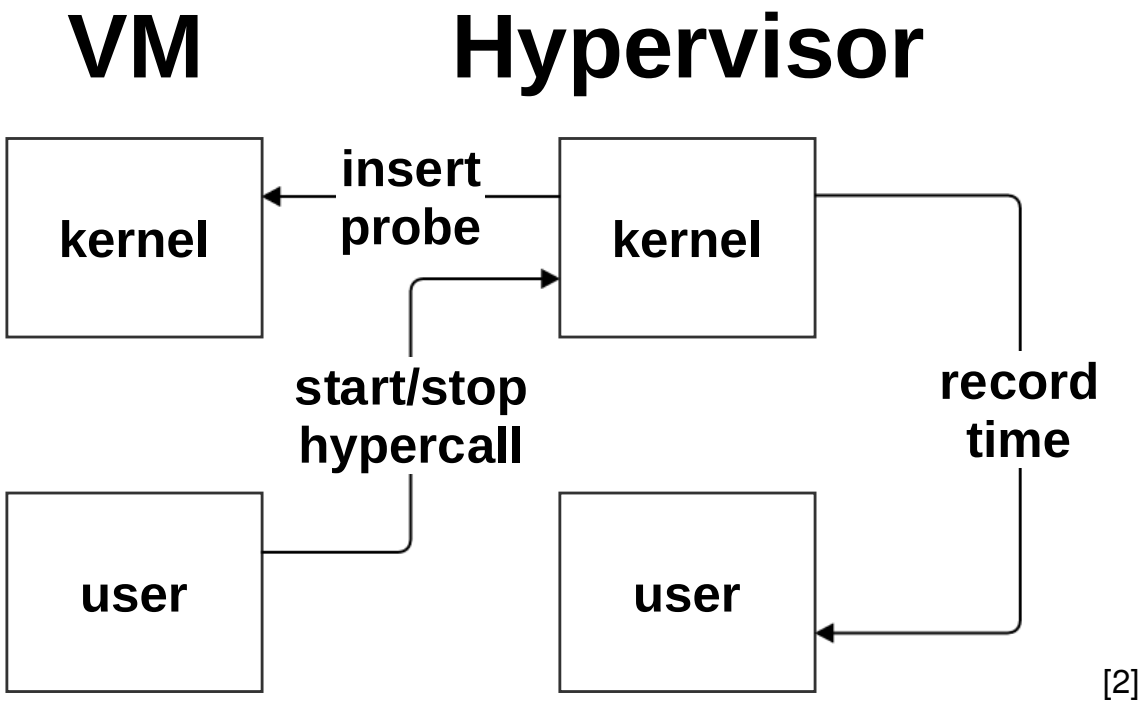
Goals

Hook-based monitoring should:

- ✓ be protected from attacks in the VM
- ✓ be simple to use
- ✓ not require guest OS modification
- ✓ be runtime adaptable
- ✓ allow for arbitrary hook placement

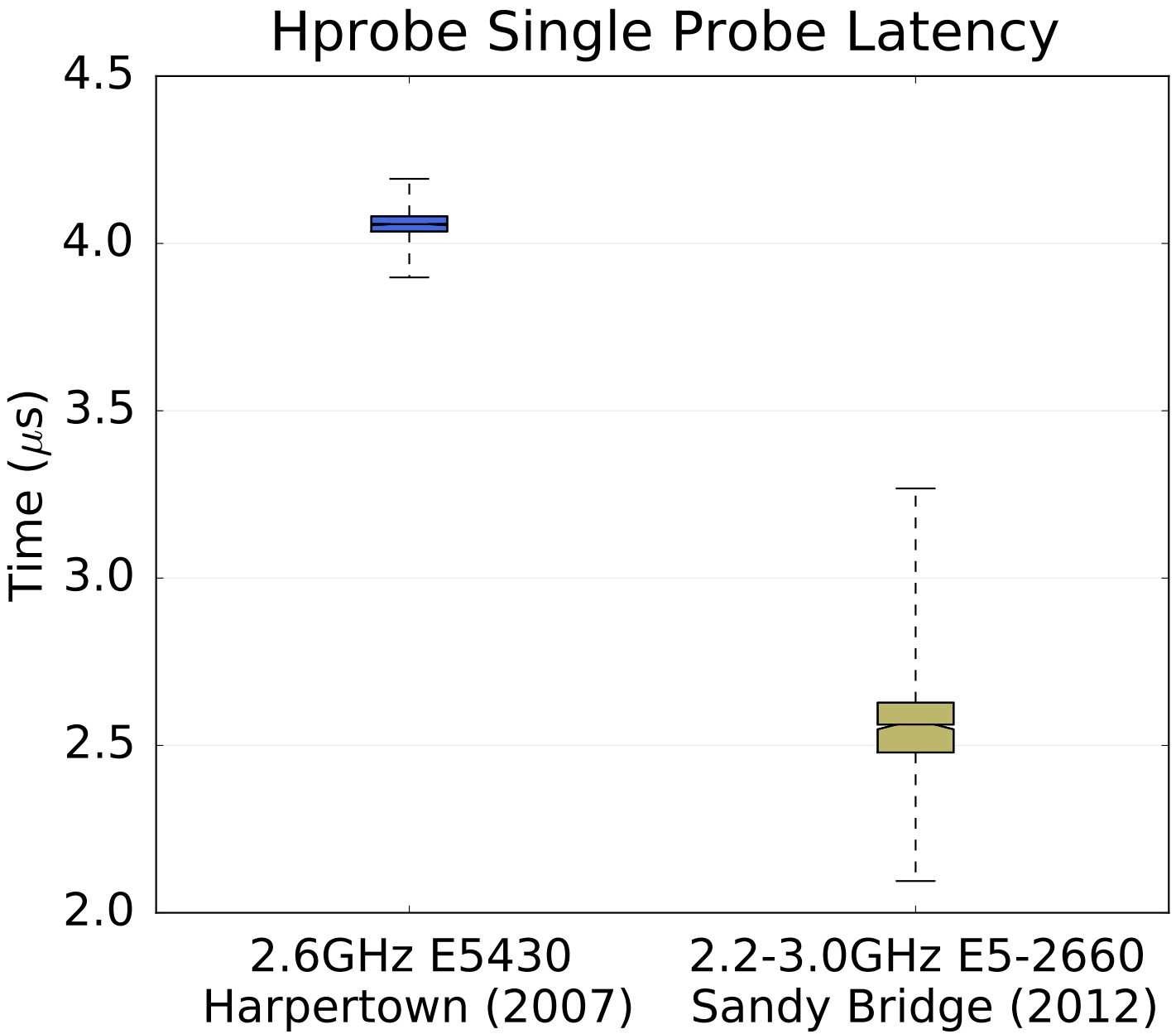
Hprobe Microbenchmarks

- probe @ `noop` kernel function
- execute 1M times



[2]

[2] Adapted from an image by Fei Deng


Hprobe Single Probe Latency
Time (μs)
4.5
4.0
3.5
3.0
2.5
2.0
2.6GHz E5430
Harpertown (2007)
2.2-3.0GHz E5-2660
Sandy Bridge (2012)

Hook-based VM Monitoring

Name	Latency	User	Dynamic	Modifications
Lares	28μs	No	No	Hypervisor/Guest
SIM	0.40μs	No	No	Hypervisor/Guest
hprobes	2.6μs	Yes	Yes	Hypervisor

Hook-based VM Monitoring

Name	Latency	User	Dynamic	Modifications
Lares	28μs	No	No	Hypervisor/Guest
SIM	0.40μs	No	No	Hypervisor/Guest
hprobes	2.6μs	Yes	Yes	Hypervisor

- as-a-Service is worth slight performance cost

Detectors

What detectors can we build with hprobes?

Detectors

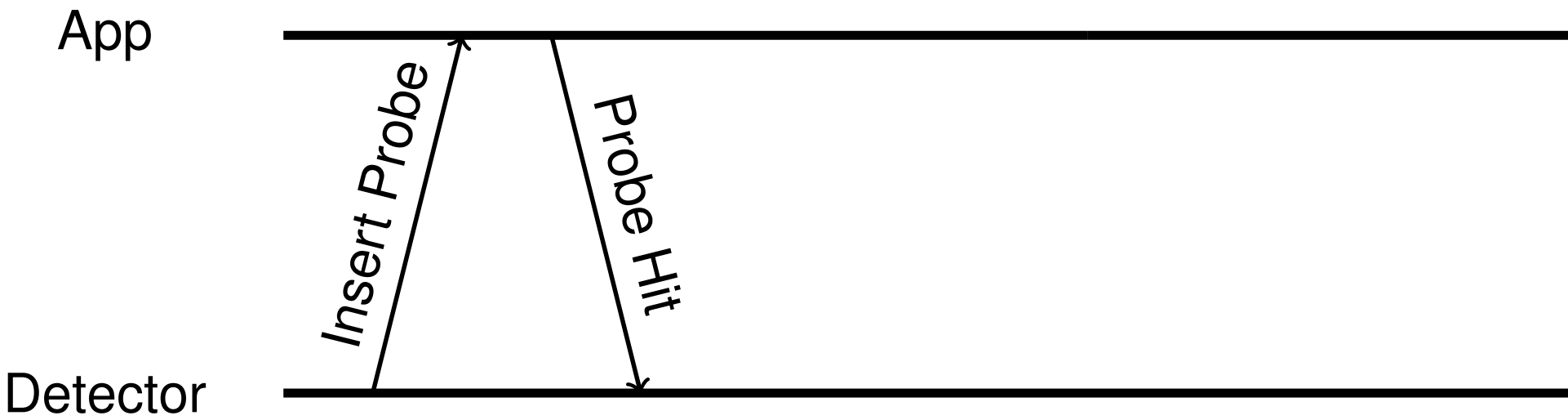
What detectors can we build with hprobes?

- Arbitrarily chose events
- On-demand
- Access to VM memory & CPU state

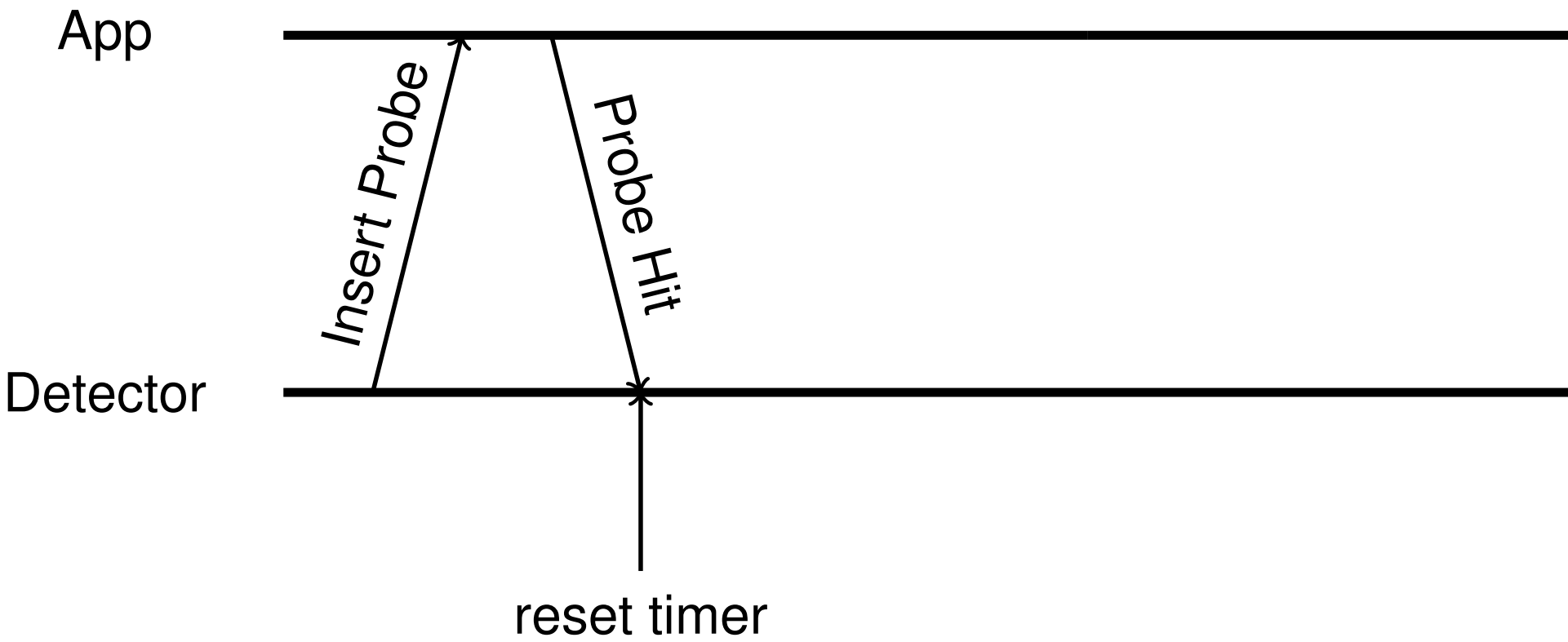
Heartbeat/watchdog

Heartbeat/watchdog

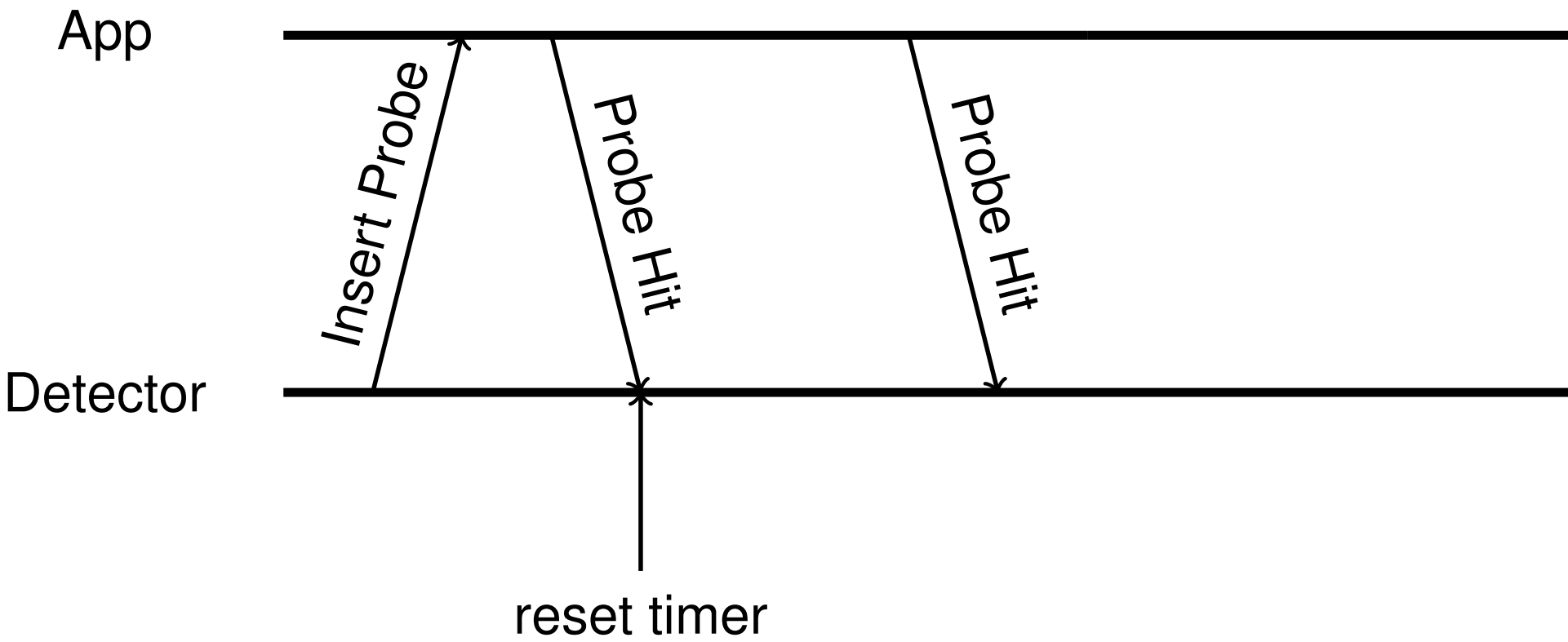
Heartbeat/watchdog



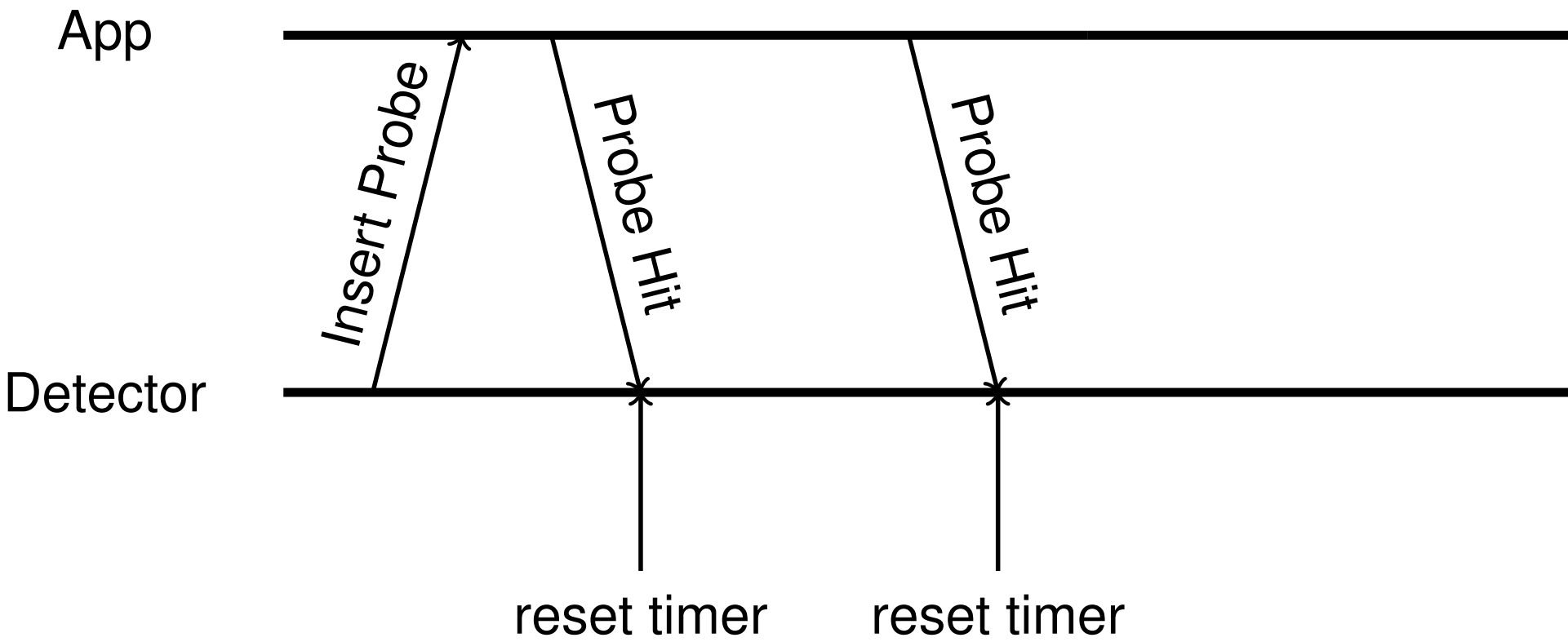
Heartbeat/watchdog



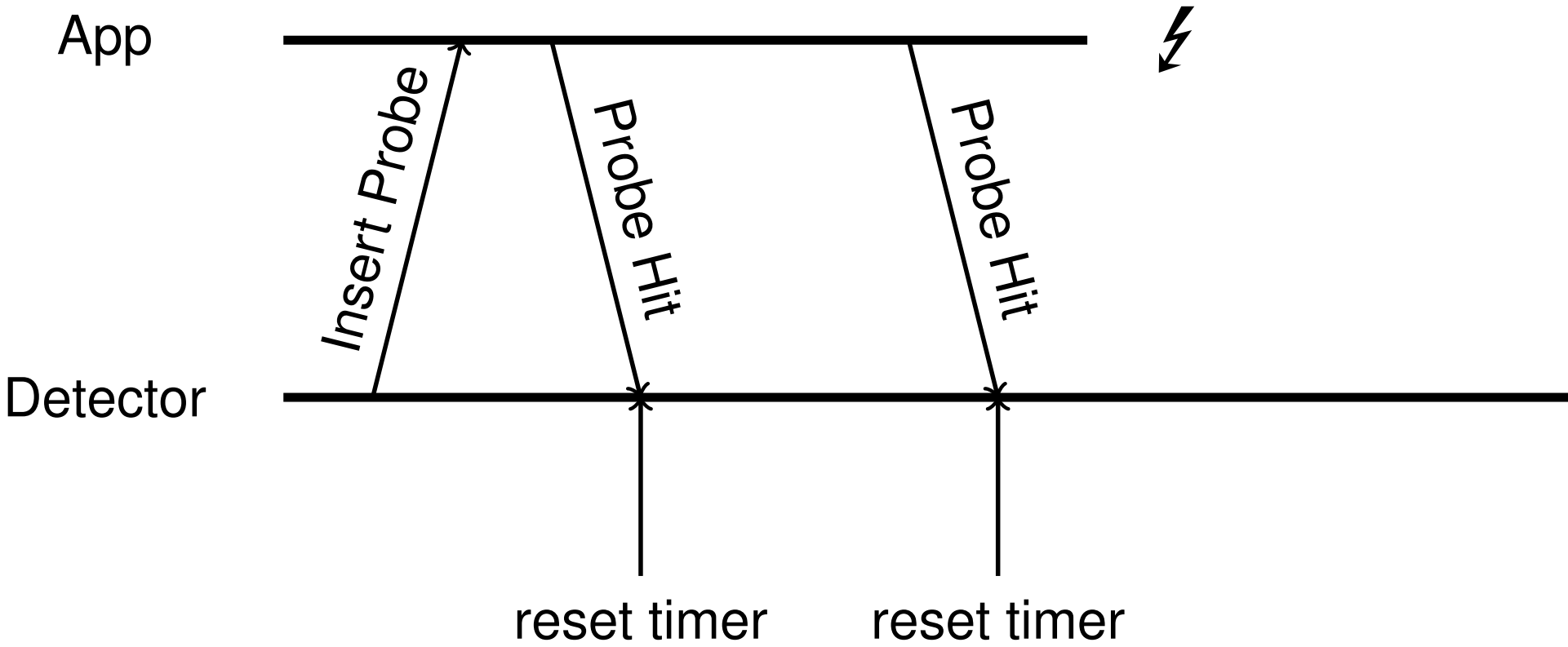
Heartbeat/watchdog



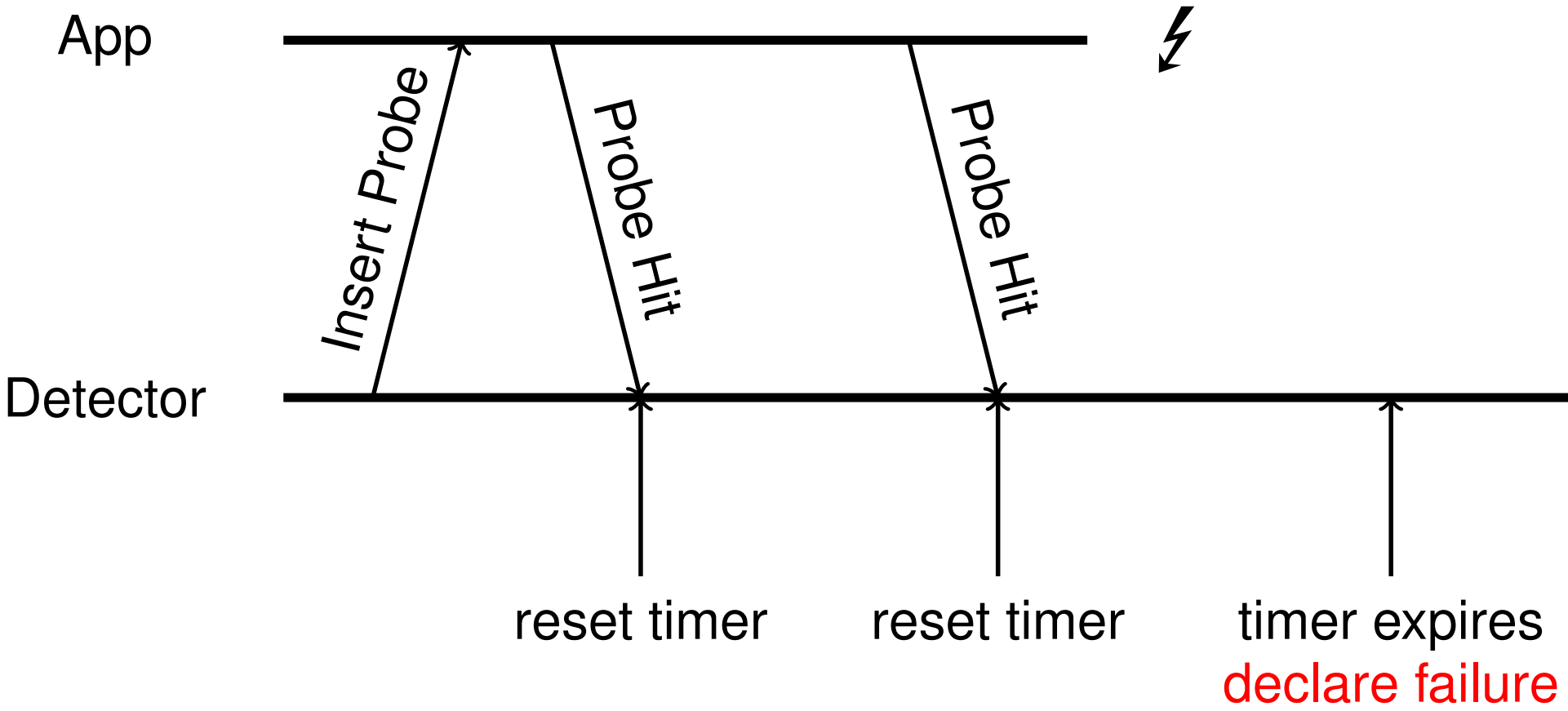
Heartbeat/watchdog



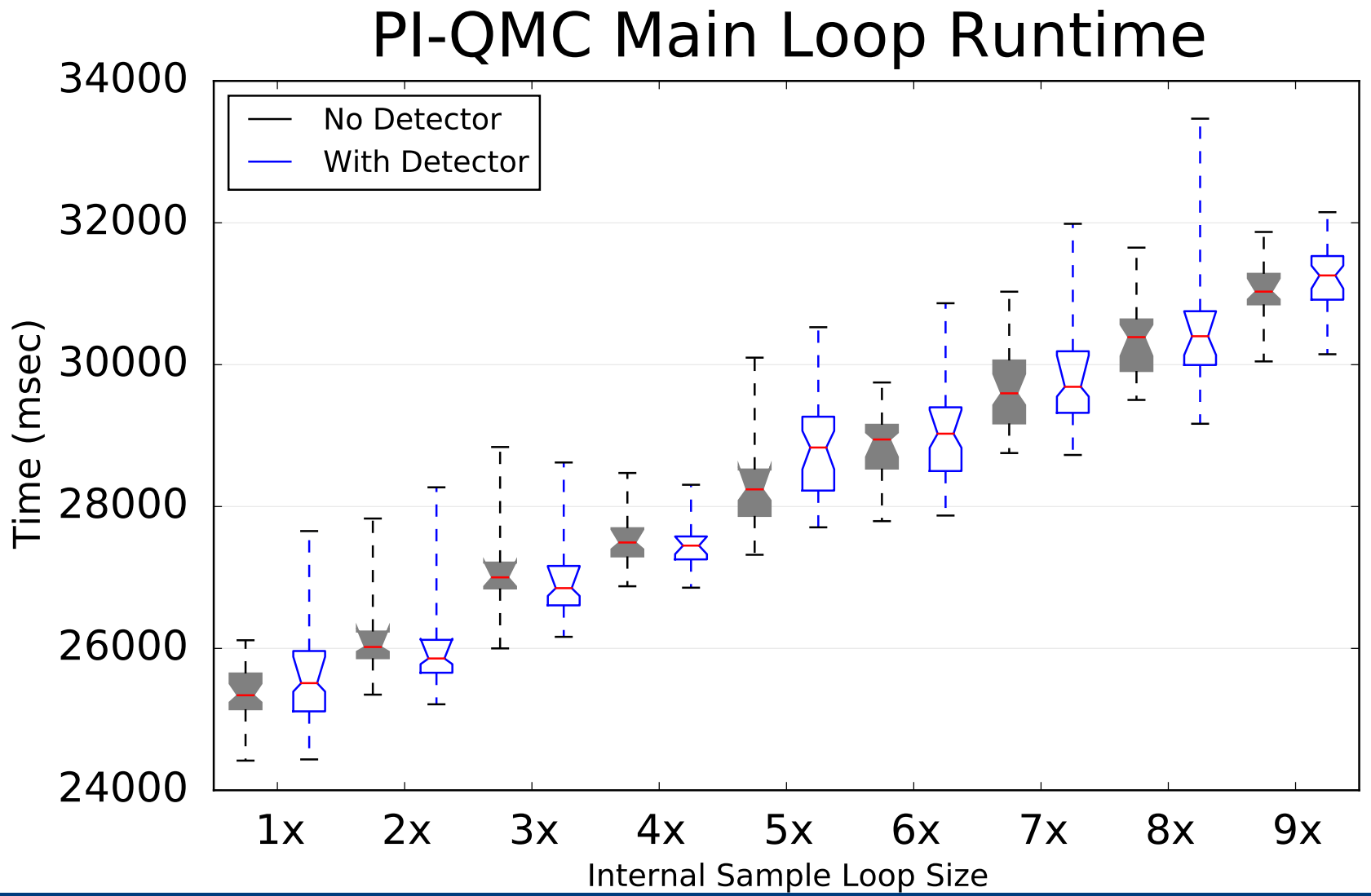
Heartbeat/watchdog



Heartbeat/watchdog



Watchdog - Performance



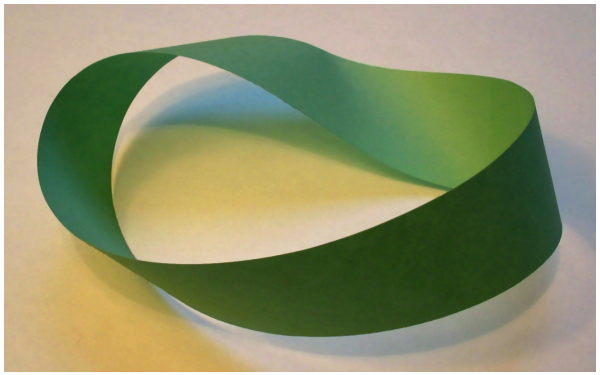
Detectors

Infinite Loop Detector

Detectors

Infinite Loop Detector

- Kernel or App-level
- Previously determined threshold
- Or register



Infinite Loop Detector

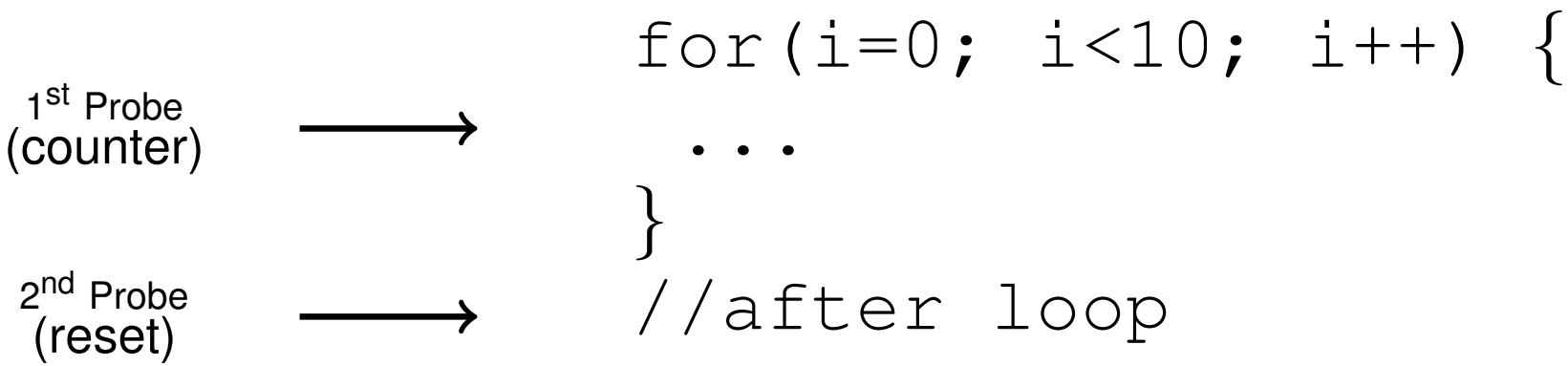
```
for(i=0; i<10; i++) {
  ...
}
//after loop
```

Infinite Loop Detector

1st Probe
(counter)

```
for(i=0; i<10; i++) {
  ...
}
//after loop
```

Infinite Loop Detector

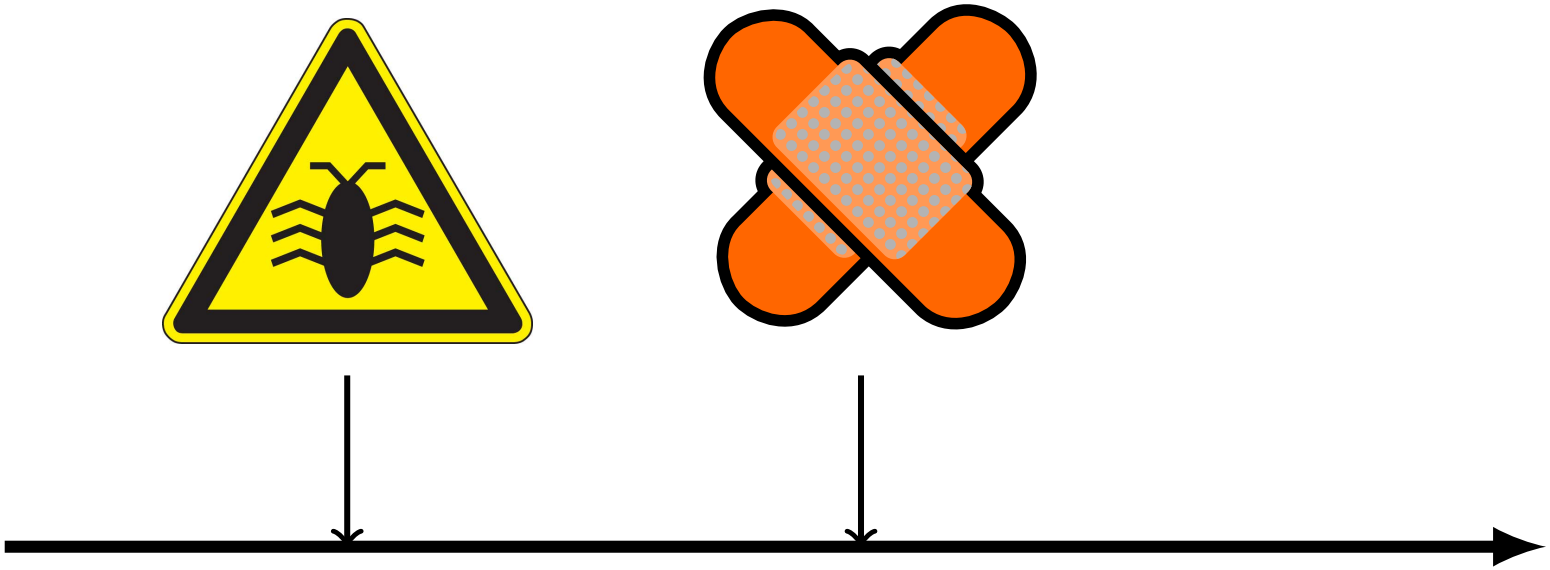


Without Infinite Loop

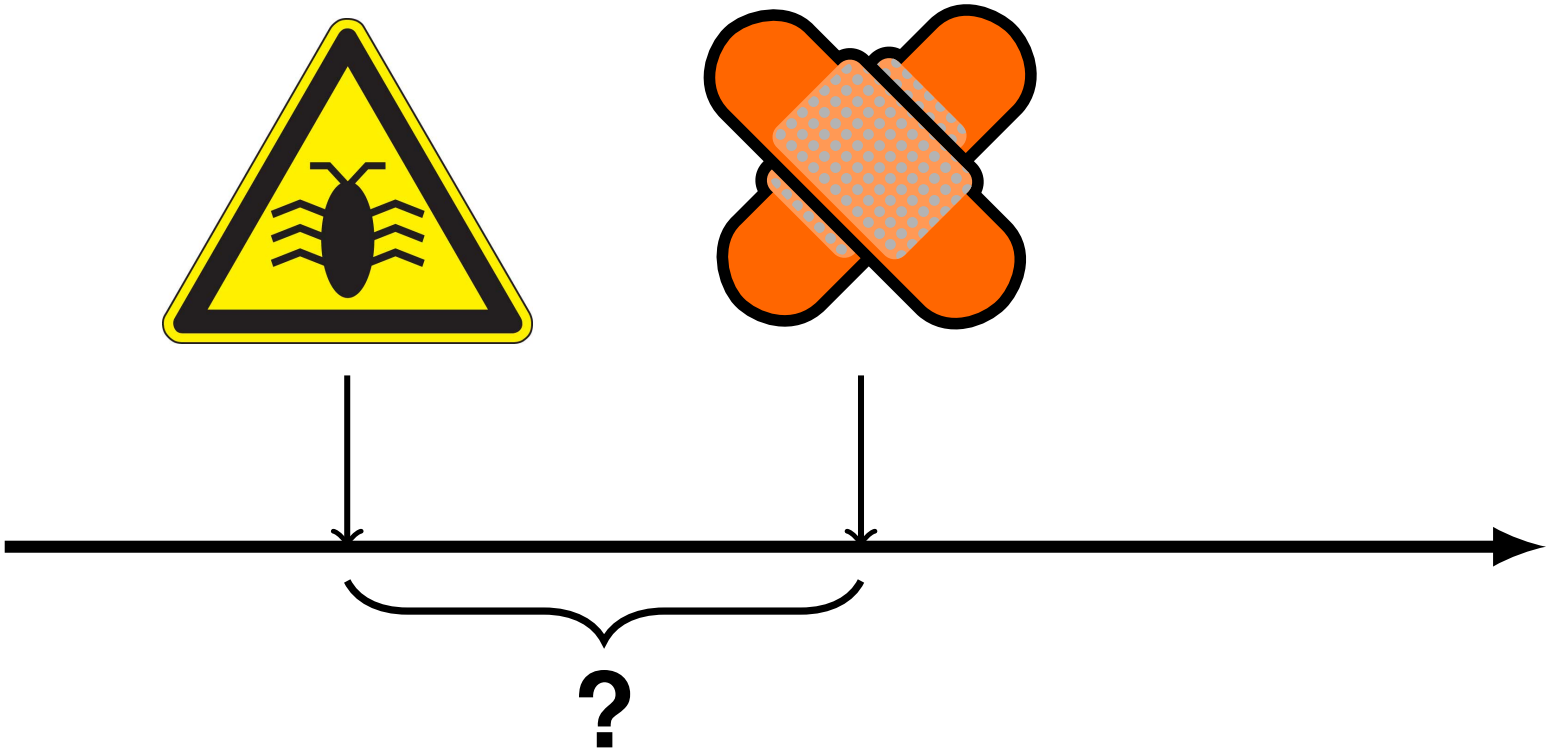
Application	**Time (s)**	**95% CI** (s)	**% overhead**
Normal	1.13	0.0325	N/A
Naïve ILD - Page	1.26	0.0229	11.5
Naïve ILD - No Page	1.26	0.0265	11.8
Smart ILD - Page	1.14	0.0267	1.15
Smart ILD - No Page	1.15	0.0215	1.9

Consider this situation

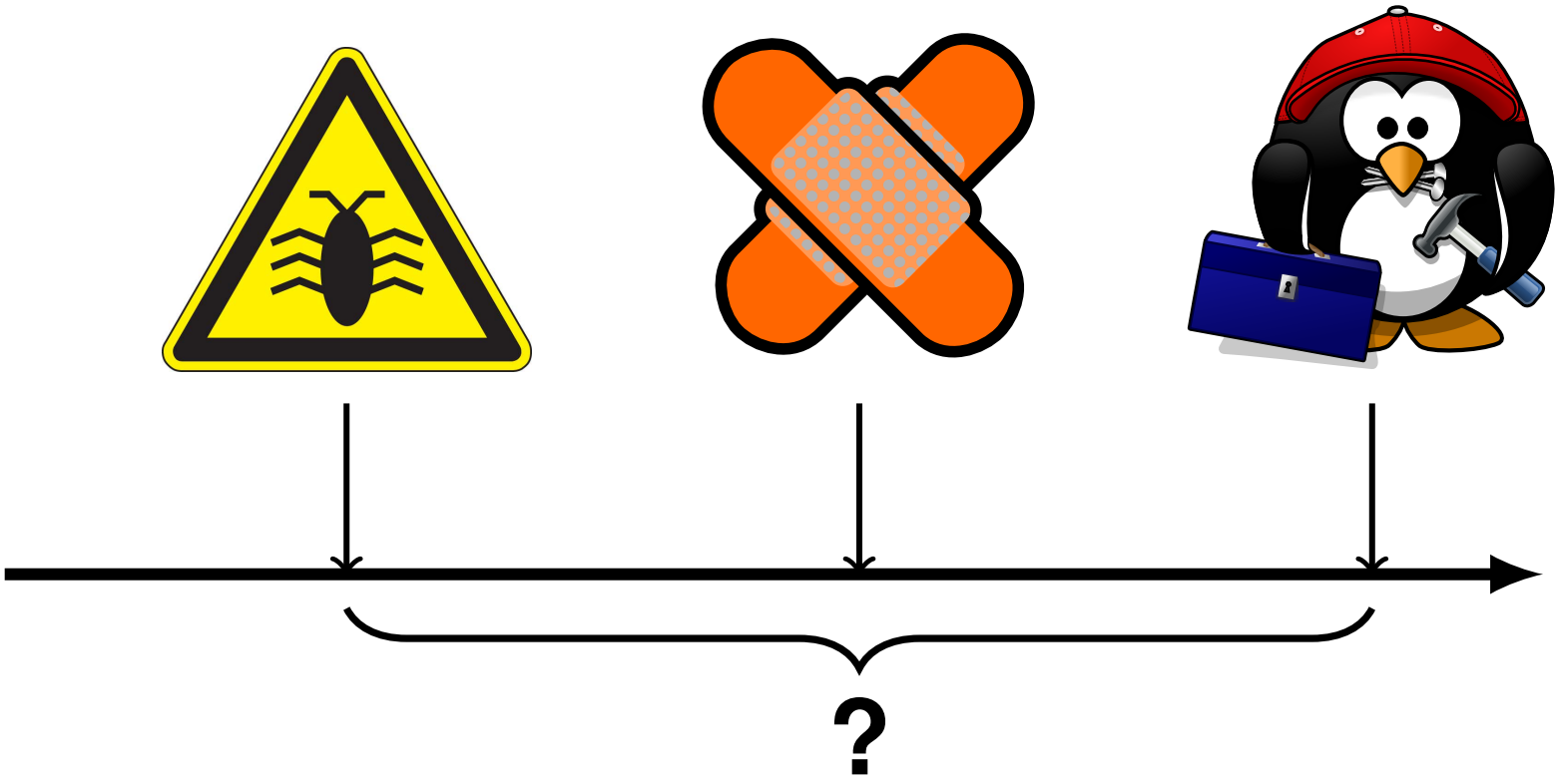
A vulnerability is announced



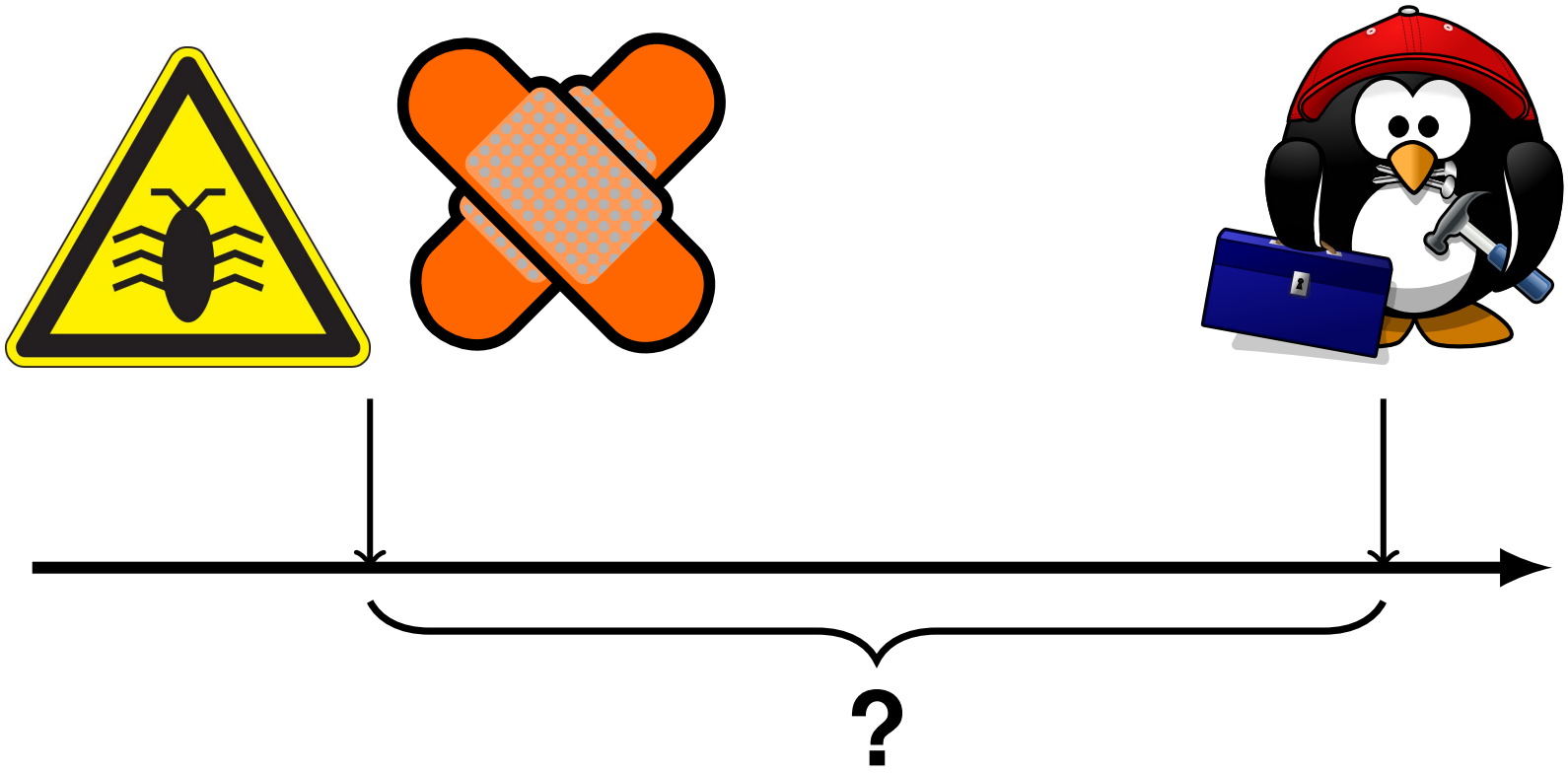
At a later time, a patch is released



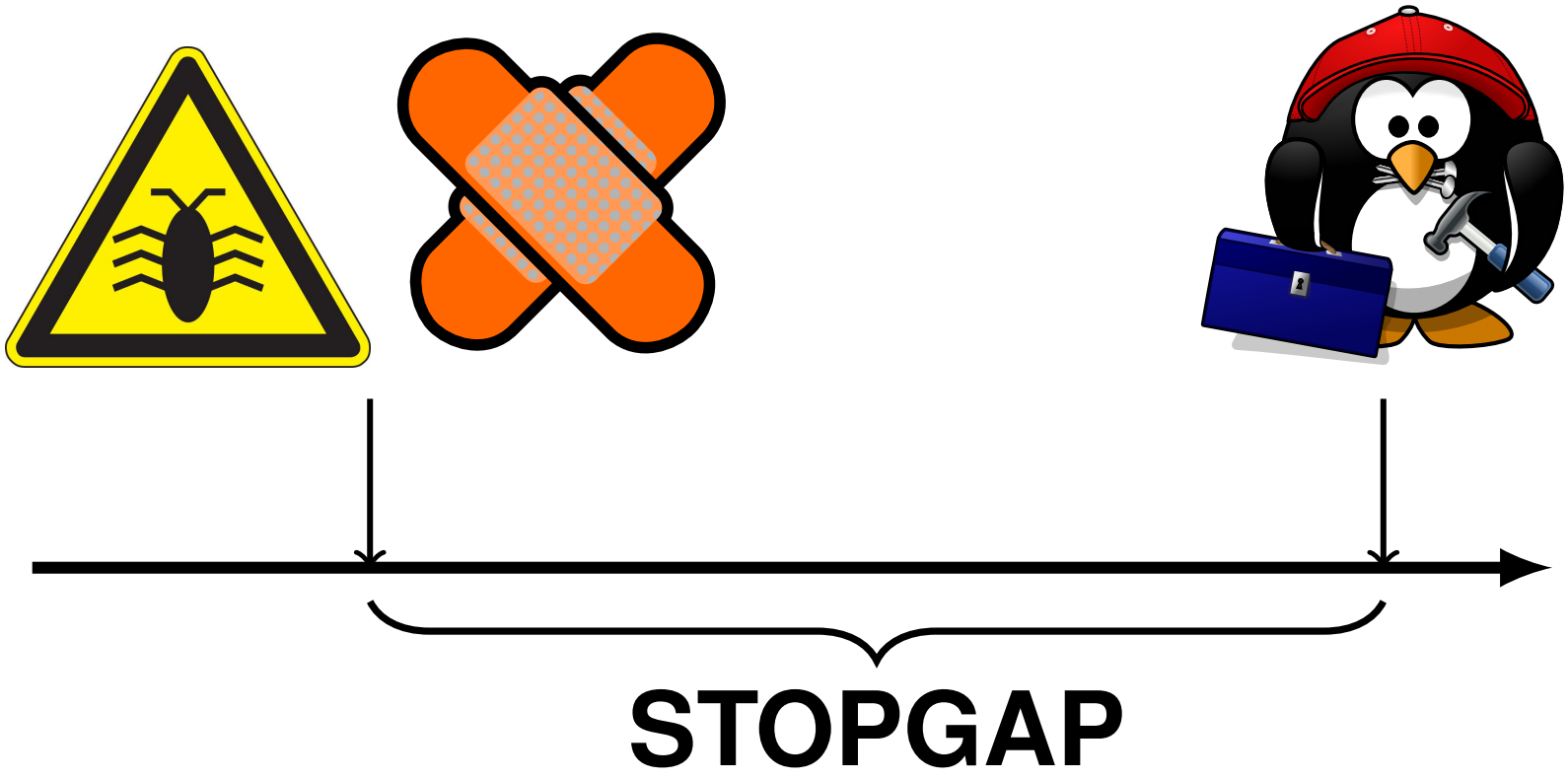
What can we do?



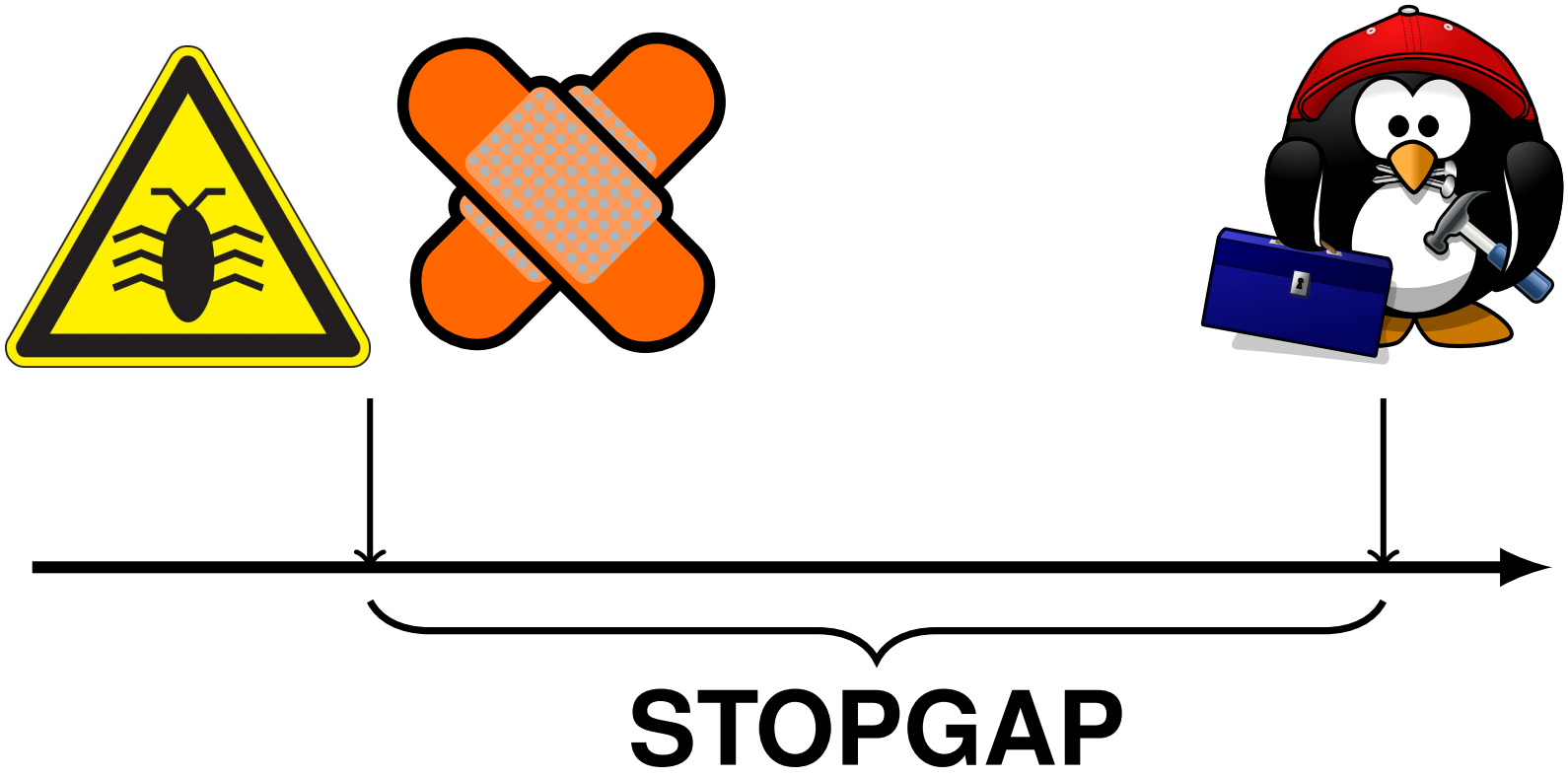
We may have to follow a maintenance window



Even when the bug and patch are coreleased



To mitigate risk, we would like a stopgap



Solution

Use an *Hprobe-based Detector*

Emergency Detector

Should be...

- easier than a patch
- simpler than a patch
- less disruptive than a patch
- less risky than a patch

Emergency Detector

- CVE-2008-0600 - Privilege Escalation in `vmsplice()` [3]
- Integer overflow in a `struct iovec` argument
- Corrupts OS (kernel) stack
- Execute attack payload

```
struct iovec {
  void  *iov_base;
  size_t iov_len;
};
```

[3] http://www.win.tue.nl/∼aeb/linux/hh/hh-12.html#ss12.4

Emergency Detector

- Added to running guest OS
- Detects malicious value that causes overflow
- Two modes of operation
  - Read-only mode: does not change anything
  - Fix mode: malicious value $\Rightarrow$ benign value

Emergency Detector

- Probe at `vmsplice()` syscall
- Get value of `iov_len` off of the stack

Emergency Detector

```
procedure VMSPLICE_HANDLER(vcpu)
    iov_pointer ← read_guest(esp+arg_offset)
    iov_len ← read_guest_virt(iov_pointer)

    if iov_len ≥ BAD_VALUE then
        HANDLE_EXPLOIT_ATTEMPT(vcpu)
    end if
end procedure
```

Detector Performance

- Checkpoint/Restart In Userspace
- Two scientific computing applications
  - Folding @ Home
  - Path-integral Quantum Monte Carlo
- Three cases:
  - Normal: base case without monitoring
  - hprobe: only monitor `sys_vmsplice`
  - Naïve: monitor all system calls

Detector Performance

Application	**Runtime** $\pm$ 95% CI (s)	**overhead (%)**
F@H Normal	0.221 ± 0.0092	0
F@H w/hprobe	0.228 ± 0.012	3.30
F@H w/Naïve	0.253 ± 0.0085	14.4
pi-qmc Normal	0.137 ± 0.0063	0
pi-qmc w/hprobe	0.140 ± 0.0073	1.73
pi-qmc w/Naïve	0.152 ± 0.0051	11.1

Thoughts

- Zero overhead without `vmsplice()`
- Cloud provider doesn't need tenant to update
- Can be used while official fix is in QA
- Don't need full understanding of bug

VM Monitoring Techniques

	Hprobes (EDCC'15)	HyperTap (DSN '14)	LiveWire (NDSS'03)	LibVMI (ACSAC'07)	SIM (CCS'09) Lares (SP'08)	Lycosid (VEE'08) Antfarm (ATC'06)	Osck (ASPLOS'11)	Virtuoso (SP'11)	VMST (SP'12)
On-demand Add/Remove	✓	✗	✗	✗	✗	✗	✗	✗	✗
Changes to VM	✗	✗	✗	✗	✓	✗	✗	✗	✗
Userspace Monitoring	✓	✗	✓	✗	✗	✗	✗	✗	✗
Root-of-trust (invariant)	OS	HW	OS	OS	OS	HW	OS	OS	OS
Active/**P**assive Mon.	A (Hook)	A	P	P	A (Hook)	A	P	P	P
Auto-generate Monitoring	✗	✗	✗	✗	✗	✗	✓	✓	✓

This Presentation

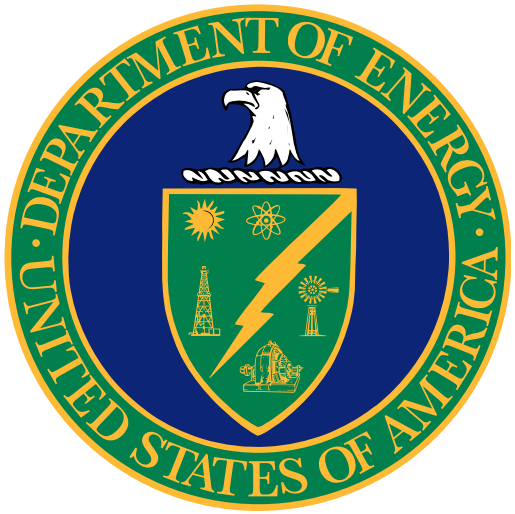
Desirable Feature

✓ = Supported Feature

✗ = Unsupported Feature

Acknowledgements

- Collaborators:
  Cuong Pham, Fei Deng, Dr. Lok Yan,
  Prof. Zbigniew Kalbarczyk, Prof. Ravi Iyer



Summary

- VM Monitoring
- How hprobes work
- Microbenchmarks
- Emergency Detector