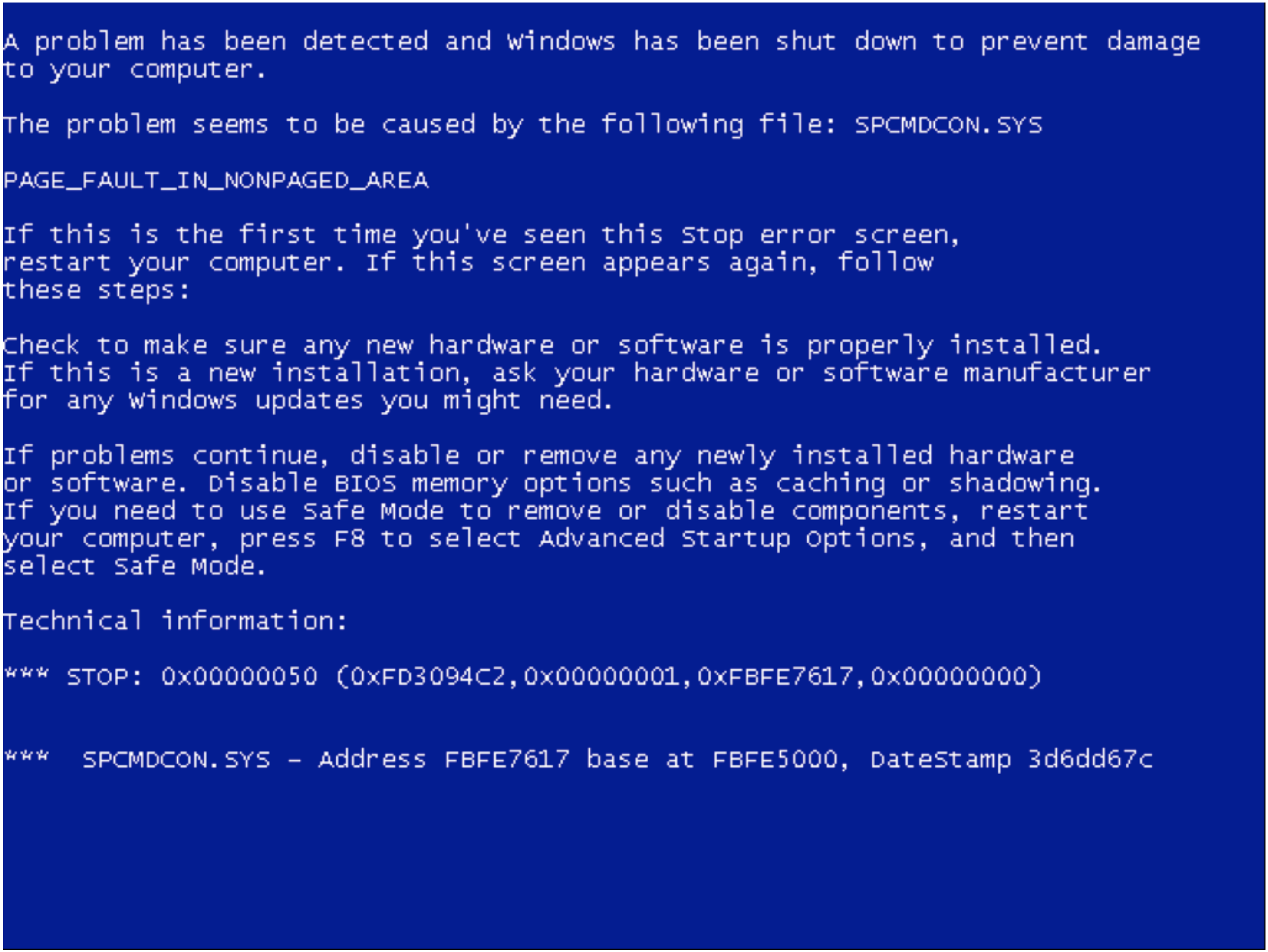
DDT

Automated testing of closed-source device drivers

Volodymyr Kuznetsov, Vitaly Chipounov, George Candea

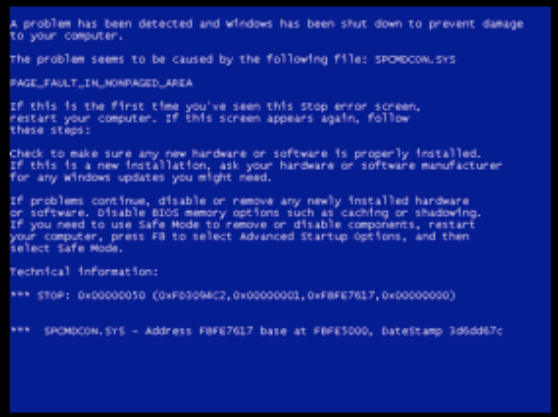
School of Computer & Communication Sciences

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE


A problem has been detected and Windows has been shut down to prevent damage
to your computer.
The problem seems to be caused by the following file: SPCMDCON.SYS
PAGE_FAULT_IN_NONPAGED_AREA
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)
*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

Driver Bugs are Critical

- Lead to system-level malfunction
  Crashes, security issues
- Driver bugs are ubiquitous
  85% of Windows crashes[1]
  3-7 times buggier than the kernel[2]



[1] V. Orgovan and M. Tricker. *An introduction to driver quality.* Microsoft Windows Hardware Engineering Conf., 2003.
[2] A. Chou, et al. *An empirical study of operating systems errors.* In SOSP 2001.

Approach #1: Dynamic Testing

```
01010101010101111010101010101010
01010111110001010111010101101100
11101010101010101010101010100011
11010101010001110011101010110111
10101010101010101010101101010101
01010101010101111010101010101010
01010111110001010111010101101100
11101010101010101010101010100011
11010101010001110011101010110111
10101010101010101010101101010101
01010101010101111010101010101010
01010111110001010111010101101100
11101010101010101010101010100011
11010101010001110011101010110111
10101010101010101010101101010101
11101010101010101010101010100011
```

Approach #1: Dynamic Testing

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet, size);

  } else {
    if(flags & URGENT) {
      send_urgent(packet, size);
    } else {
      queue(packet, size);
    }
  }
}
```

Approach #1: Dynamic Testing

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet, size);

  } else {
    if(flags & URGENT) {
      send_urgent(packet, size);
    } else {
      queue(packet, size);
    }
  }
}
```

```
SendPacket("...", 10, 0);
```

Approach #1: Dynamic Testing

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet, size);

  } else {
    if(flags & URGENT) {
      send_urgent(packet, size);
    } else {
      queue(packet, size);
    }
  }
}
```

```
SendPacket("...", 10, 0);
SendPacket("...", 20, 0);
SendPacket("...", 30, 5);
    ...
SendPacket("...", 123, 321);
```

Approach #1: Dynamic Testing

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet, size);

  } else {
    if(flags & URGENT) {
      send_urgent(packet, size);
    } else {
      queue(packet, size);
    }
  }
}
```

```
SendPacket("...", 10, 0);
SendPacket("...", 20, 0);
SendPacket("...", 30, 5);
    ...
SendPacket("...", 123, 321);
```

Simplest approach

Approach #1: Dynamic Testing

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet, size);

  } else {
    if(flags & URGENT) {
      send_urgent(packet, size);
    } else {
      queue(packet, size);
    }
  }
}
```

```
SendPacket("...", 10, 0);
SendPacket("...", 20, 0);
SendPacket("...", 30, 5);
    ...
SendPacket("...", 123, 321);
```

Approach #2: Exhaustive Path Exploration

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet, size);

  } else {
    if(flags & URGENT) {
      send_urgent(packet, size);
    } else {
      queue(packet, size);
    }
  }
}
```

Approach #2: Exhaustive Path Exploration

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet, size);

  } else {
    if(flags & URGENT) {
      send_urgent(packet, size);
    } else {
      queue(packet, size);
    }
  }
}
```

```
SendPacket(α, β, γ);
```

Approach #2: Exhaustive Path Exploration

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet, size);

  } else {
    if(flags & URGENT) {
      send_urgent(packet, size);
    } else {
      queue(packet, size);
    }
  }
}
```

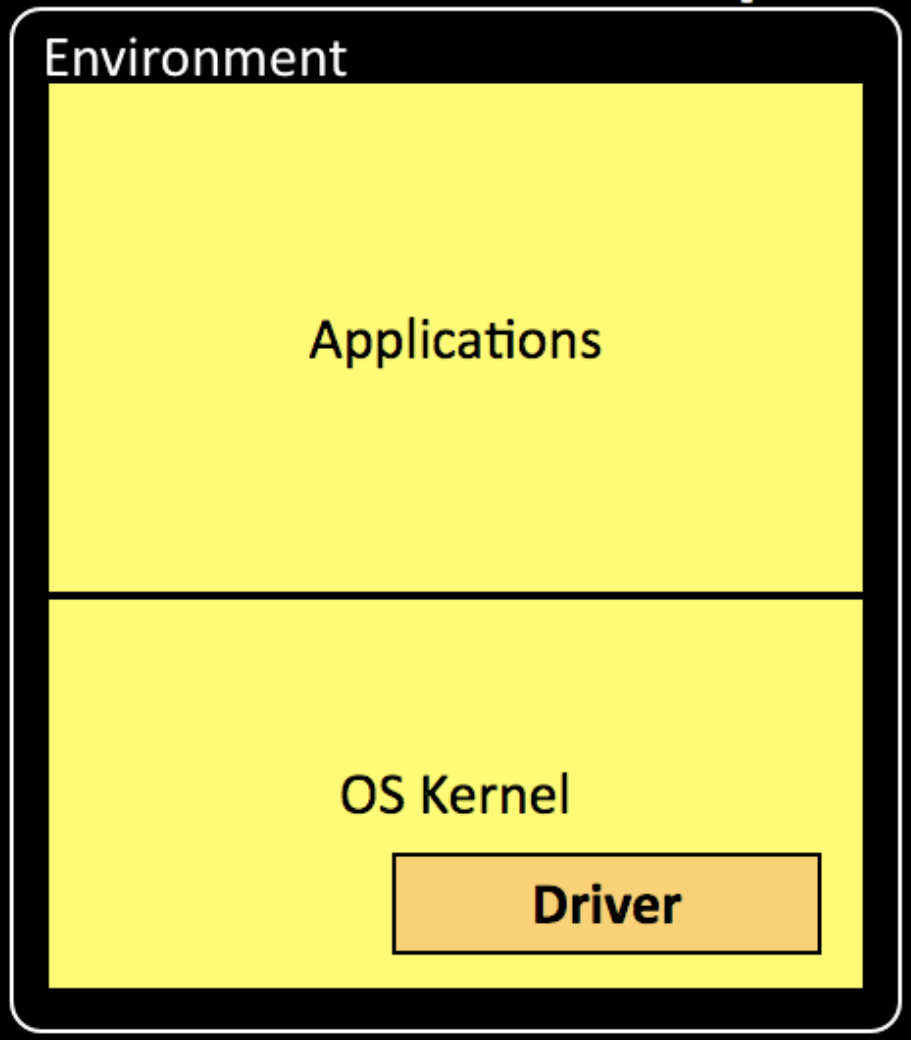
```
SendPacket(α, β, γ);
```

$$\text{paths} = 2^{\text{driver size}}$$

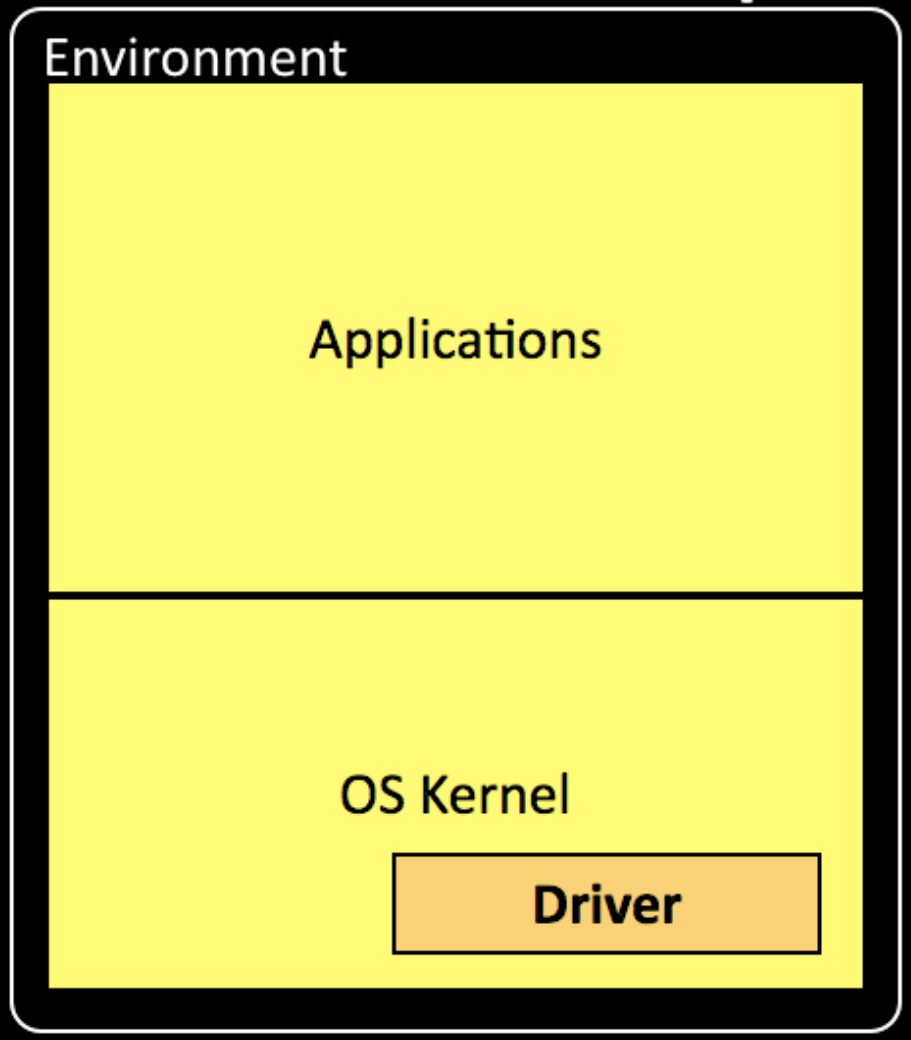
Approach #2: Exhaustive Path Exploration

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet, size);

  } else {
    if(flags & URGENT) {
      send_urgent(packet, size);
    } else {
      queue(packet, size);
    }
  }
}
```

Approach #2: Exhaustive Path Exploration

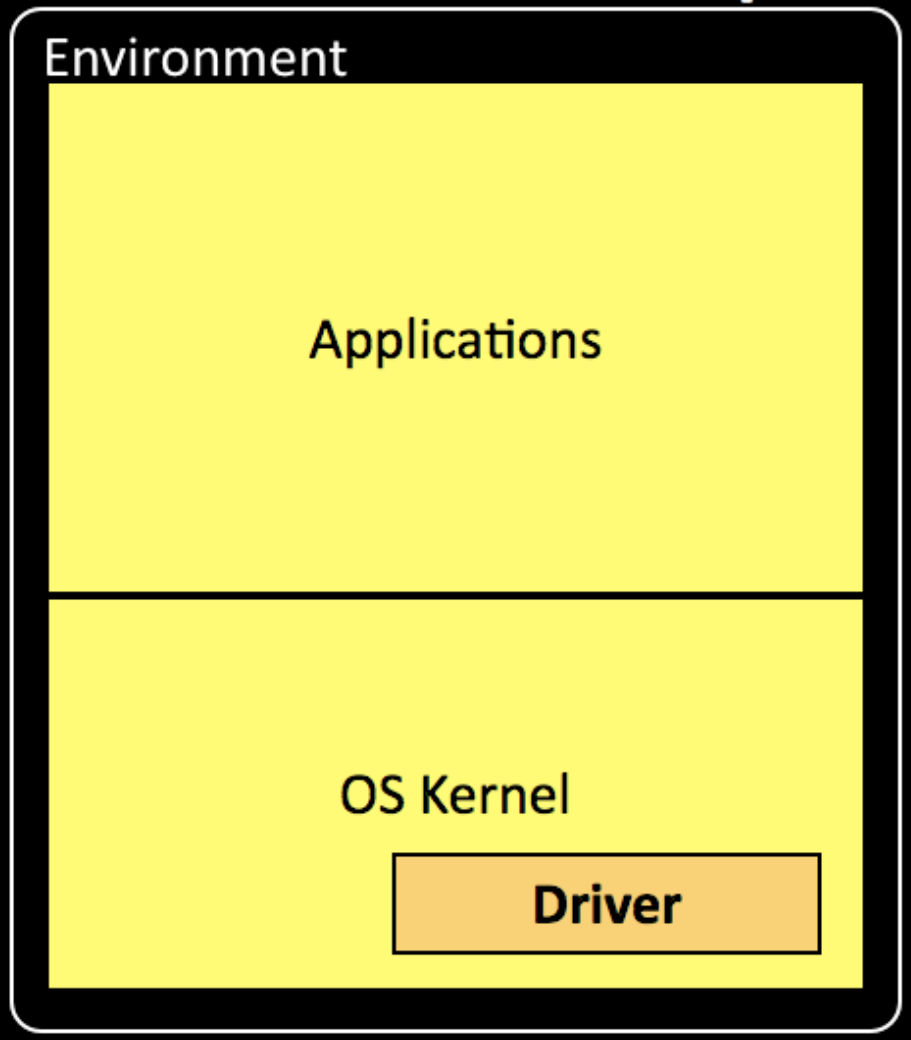


Approach #2: Exhaustive Path Exploration



- $2^{\text{system size}}$ is off limits

Approach #2: Exhaustive Path Exploration



- $2^{\text{system size}}$ is off limits
- hard to control hardware devices

Approach #3: Modeling

Approach #3: Modeling

- Models abstract the environment
  Static analysis
  Symbolic execution

- Hard to write and hard to maintain

Challenges

- ***Hard*** to achieve coverage of a driver that interacts with a large system
- ***Hard*** to control hardware and OS interactions

- ***Hard*** to achieve coverage of a driver that interacts with a large system
- ***Hard*** to control hardware and OS interactions

- ***Hard*** to achieve coverage of a driver that interacts with a large system
- ***Hard*** to control hardware and OS interactions

- ***Hard*** to achieve coverage of a driver that interacts with a large system
- ***Hard*** to control hardware and OS interactions

	Dynamic	**Static**
Closed-source drivers		
Unmodified environment		
Works without hardware		
Multi-path		

- ***Hard*** to achieve coverage of a driver that interacts with a large system
- ***Hard*** to control hardware and OS interactions

	Dynamic	Static
Closed-source drivers	✔	
Unmodified environment	✔	
Works without hardware	✖	
Multi-path	✖	

- ***Hard*** to achieve coverage of a driver that interacts with a large system
- ***Hard*** to control hardware and OS interactions

	Dynamic	Static
Closed-source drivers	✔	✖
Unmodified environment	✔	✖
Works without hardware	✖	✔
Multi-path	✖	✔

- ***Hard*** to achieve coverage of a driver that interacts with a large system
- ***Hard*** to control hardware and OS interactions

	Dynamic	Static	DDT
Closed-source drivers	✔	✖	✔
Unmodified environment	✔	✖	✔
Works without hardware	✖	✔	✔
Multi-path	✖	✔	✔

Contributions

High coverage testing of drivers
On closed-source drivers
➔Consistent mixing of symbolic and concrete execution

Testing without physical hardware
Accounting for buggy hardware
➔Symbolic hardware

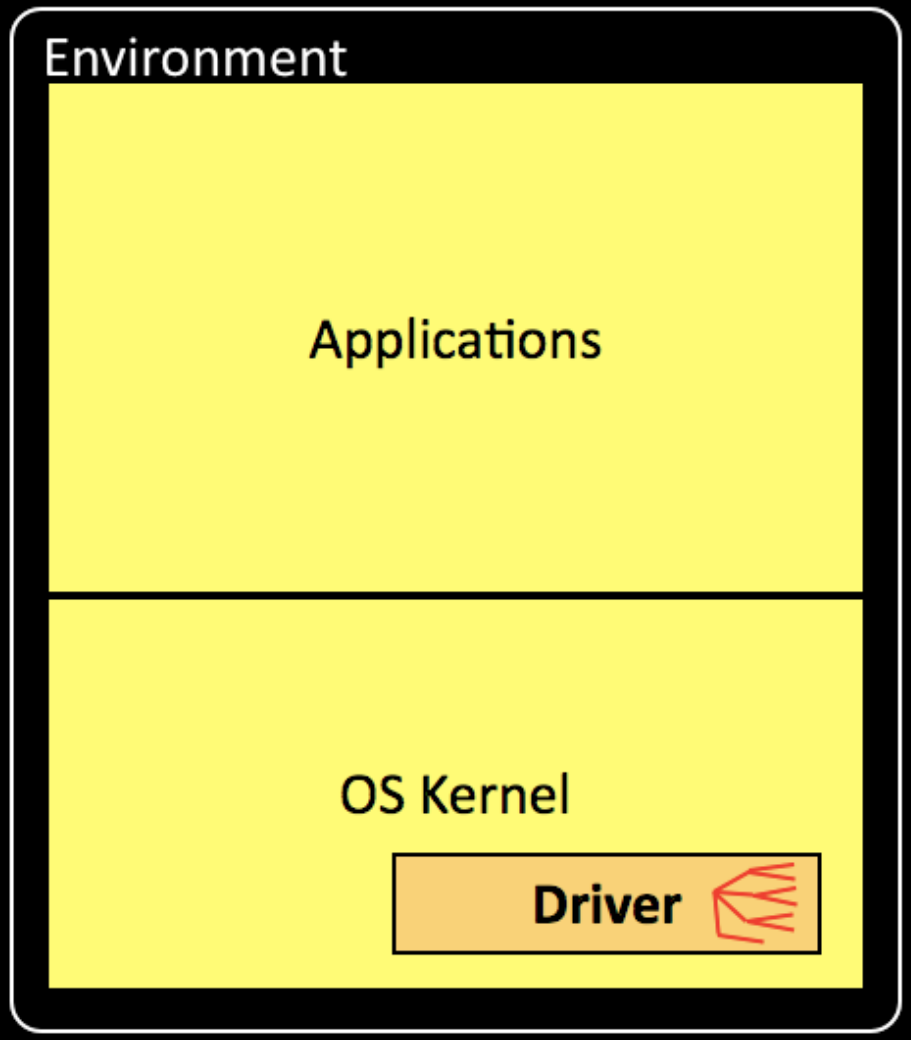
DDT

Volodymyr Kuznetsov - http://ddt.epfl.ch

DDT

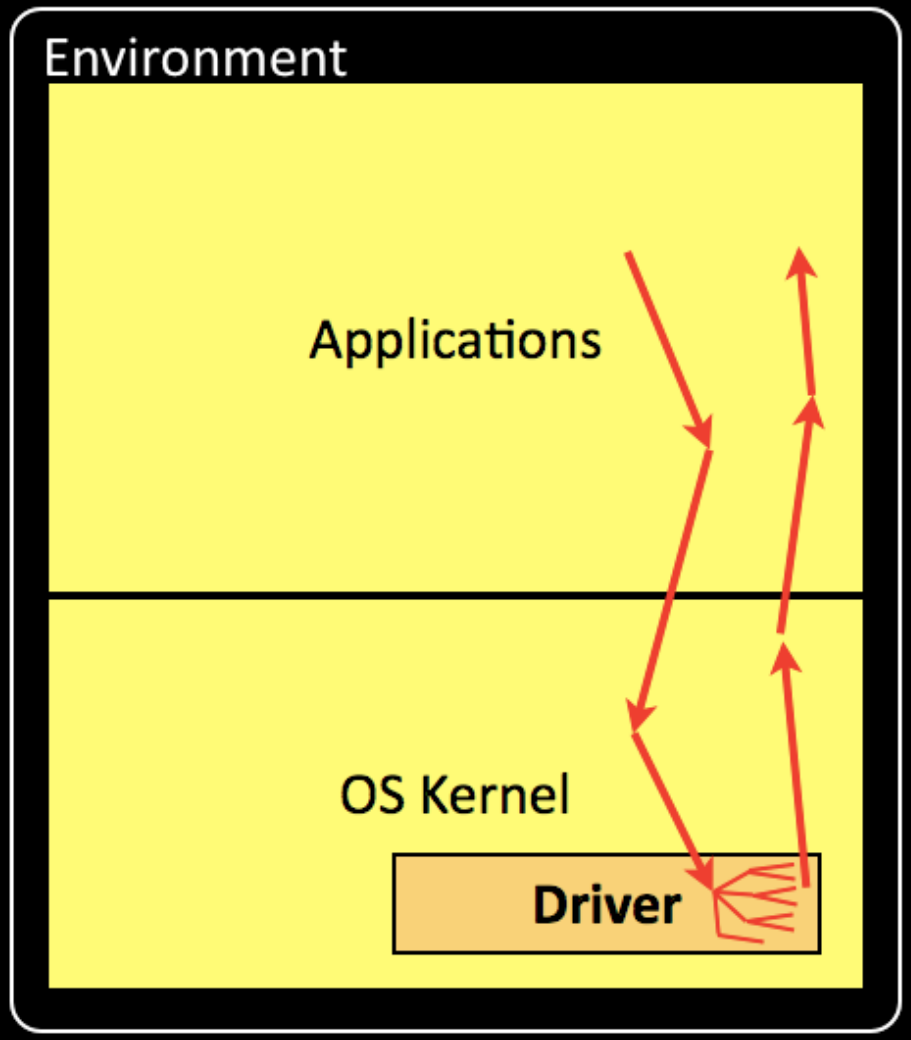
- Execute the driver symbolically

DDT



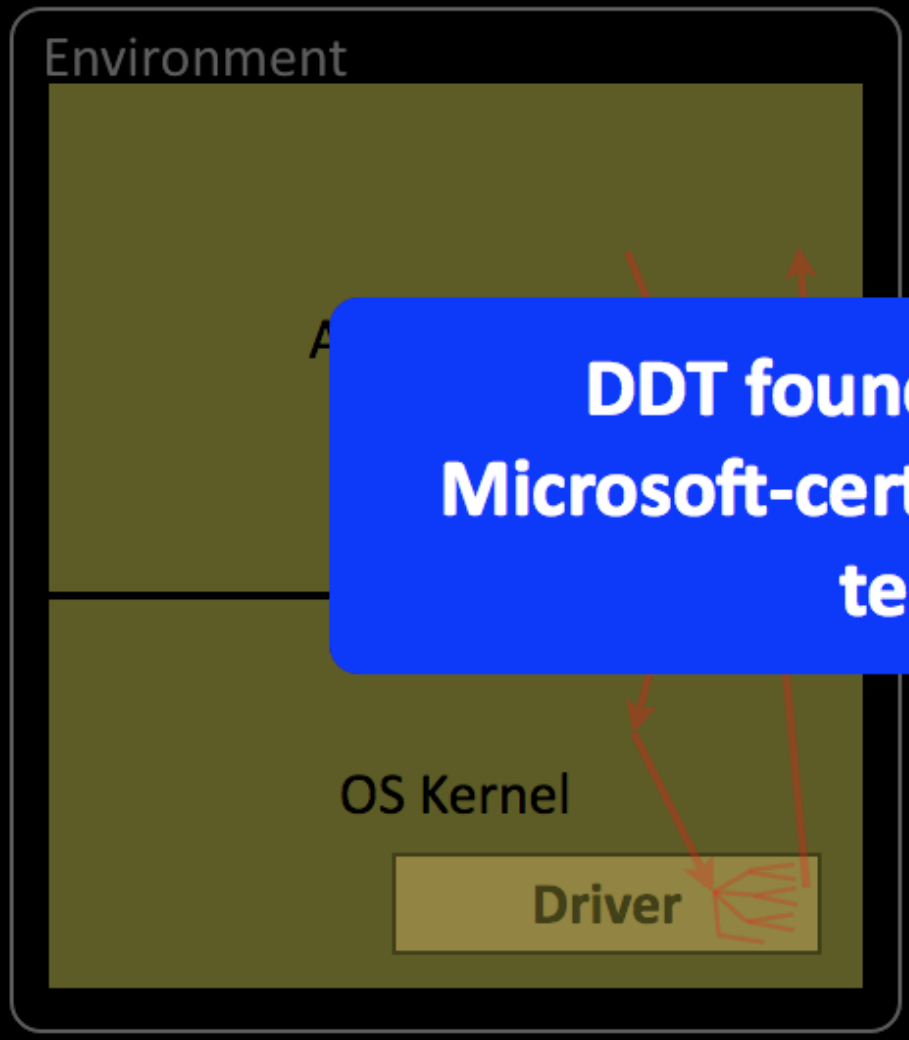
- Execute the driver symbolically
- Inside its native environment

DDT



- Execute the driver symbolically
- Inside its native environment
- Execute the environment concretely

DDT



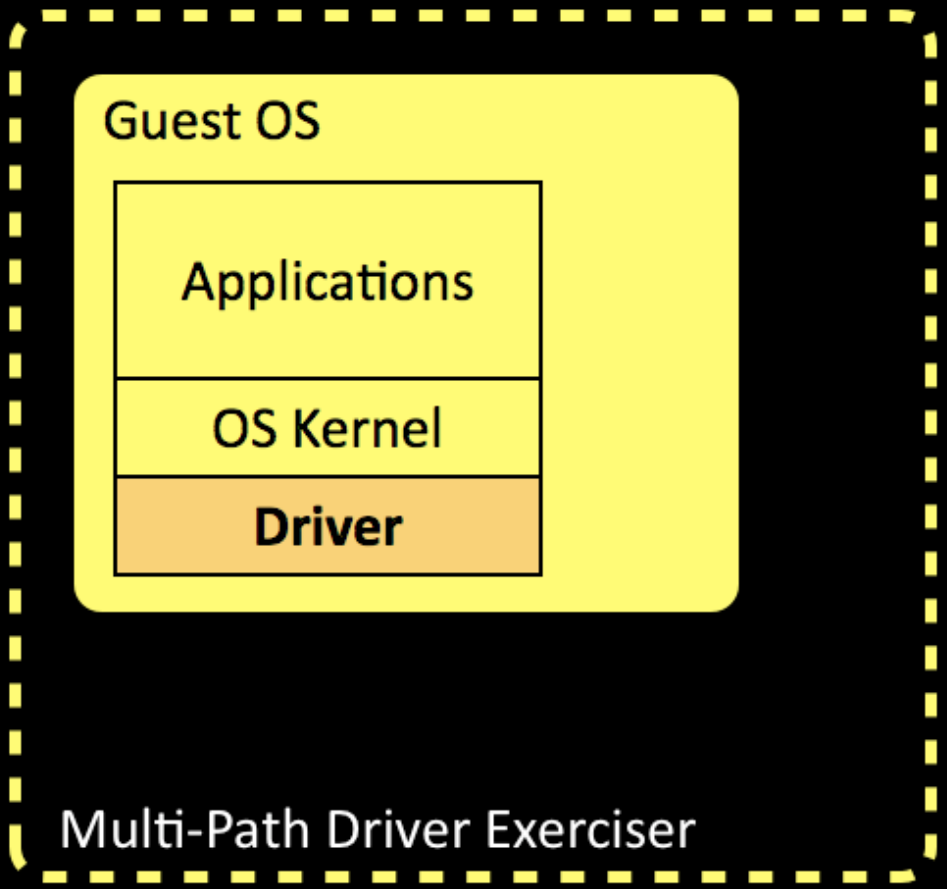
- Execute the driver symbolically

DDT found bugs in all Microsoft-certified drivers we tested

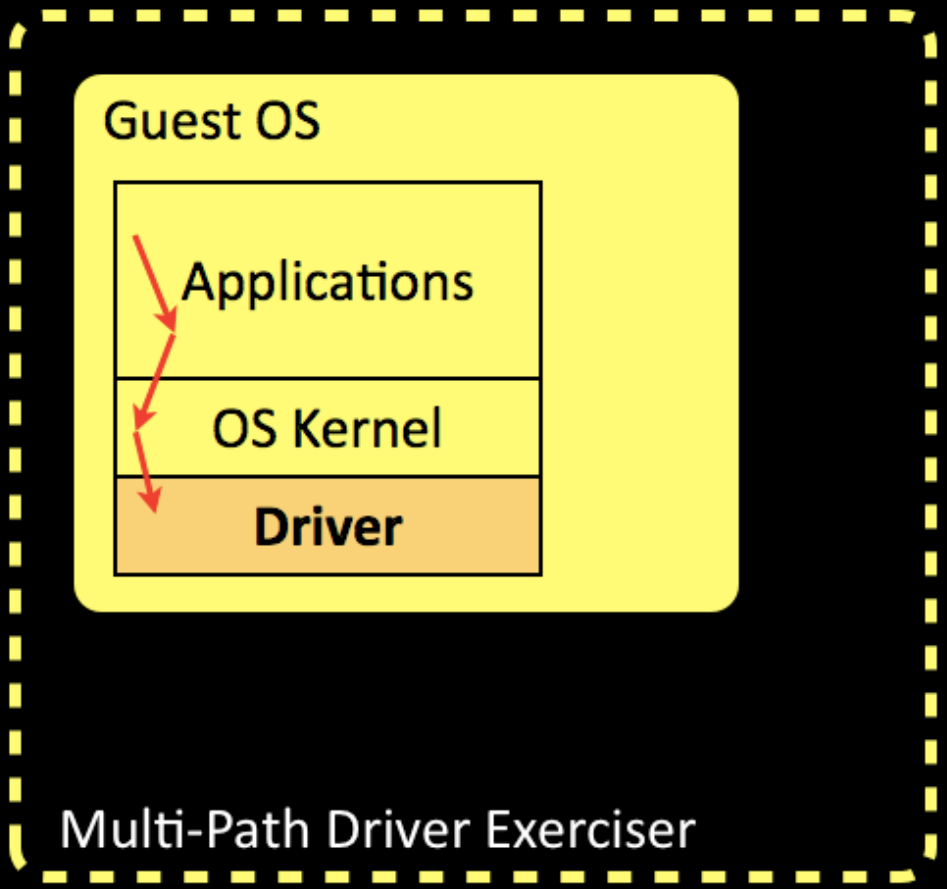
- Execute the environment concretely

Outline

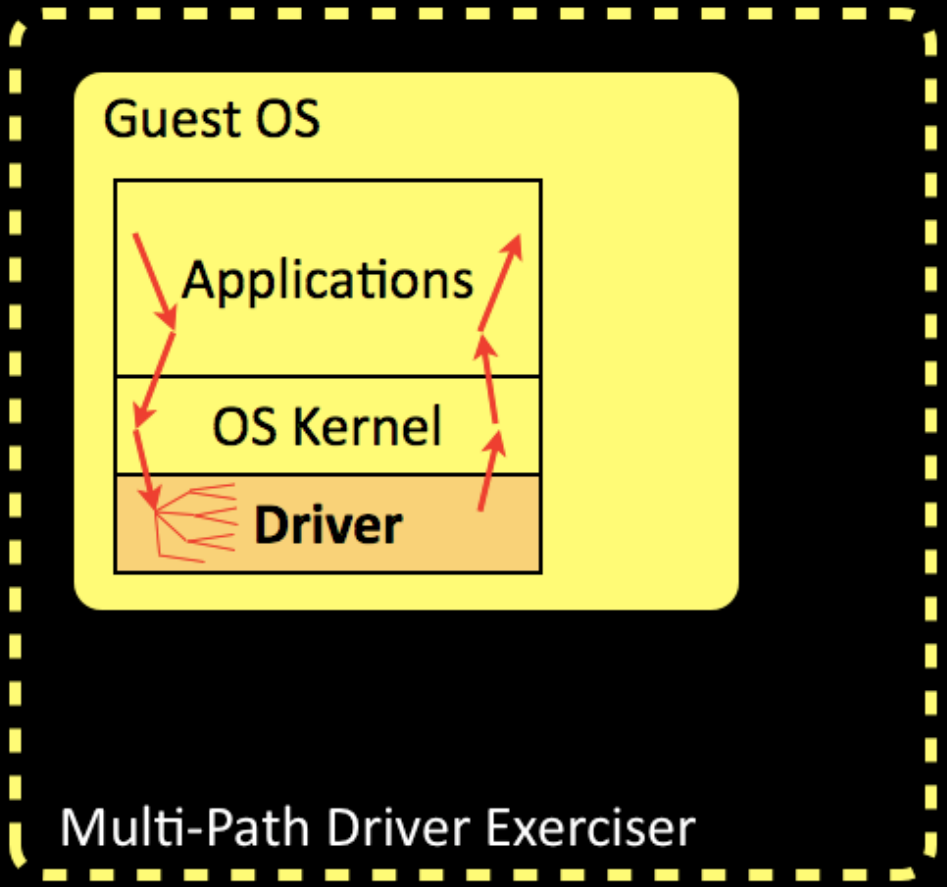
Outline


Guest OS
Applications
OS Kernel
Driver
Multi-Path Driver Exerciser

Outline

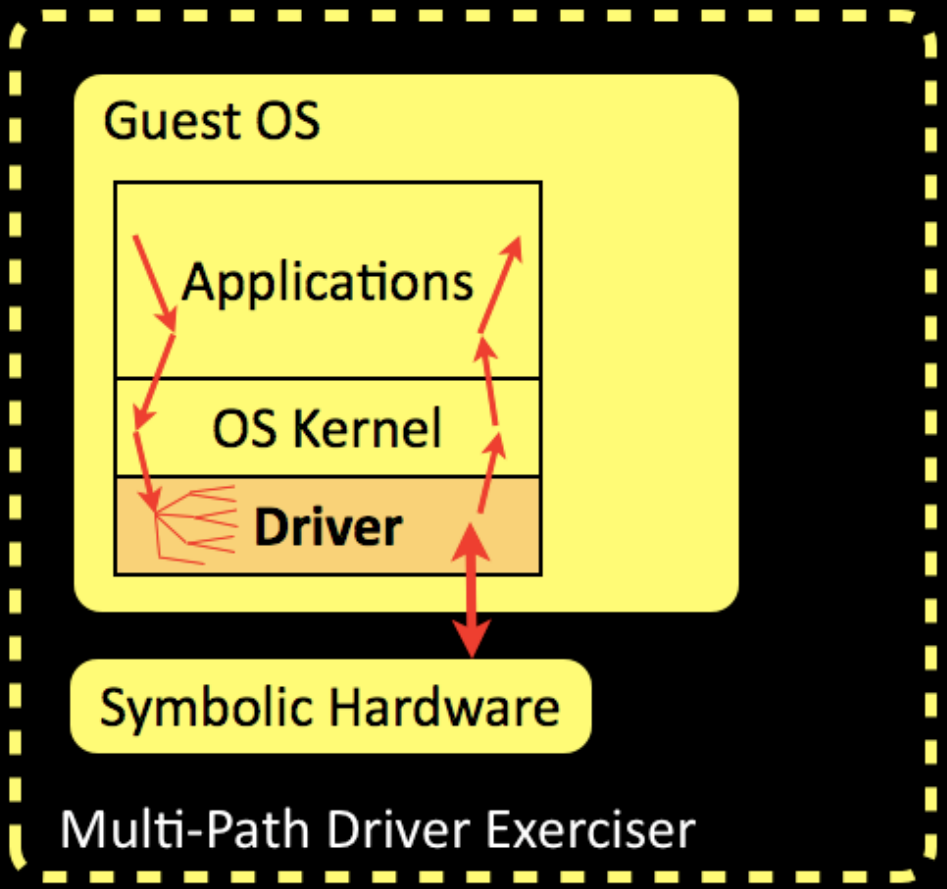

Guest OS
Applications
OS Kernel
Driver
Multi-Path Driver Exerciser

Outline



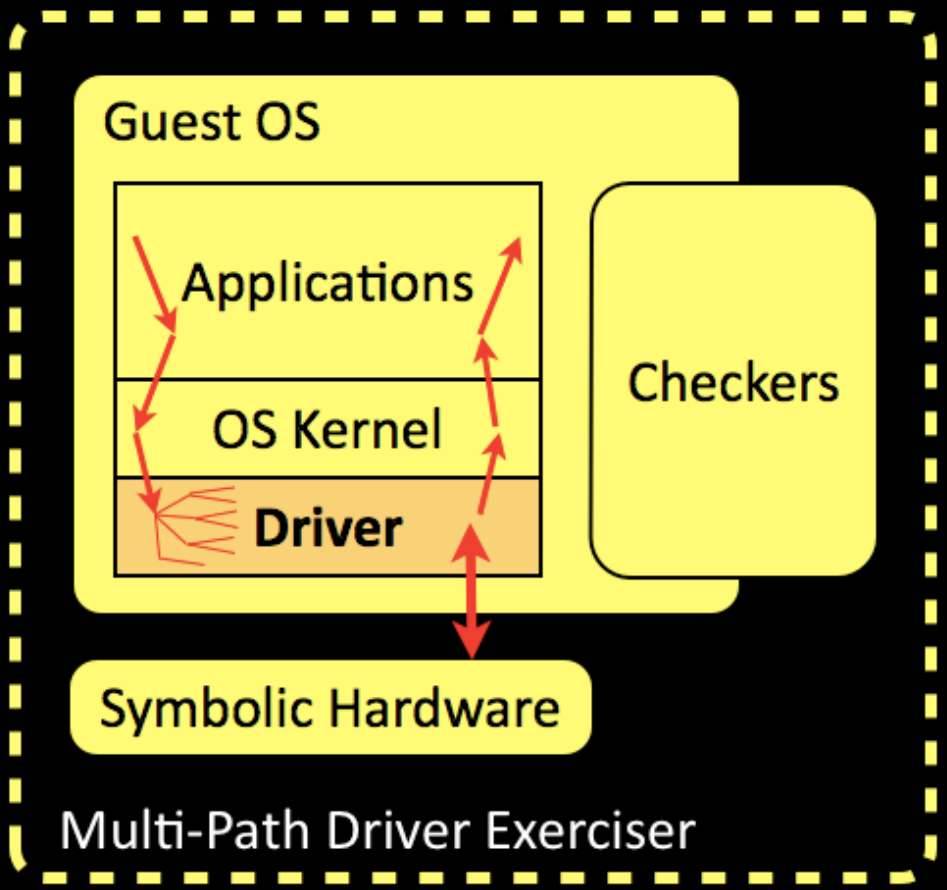
- Driver Exerciser

Outline



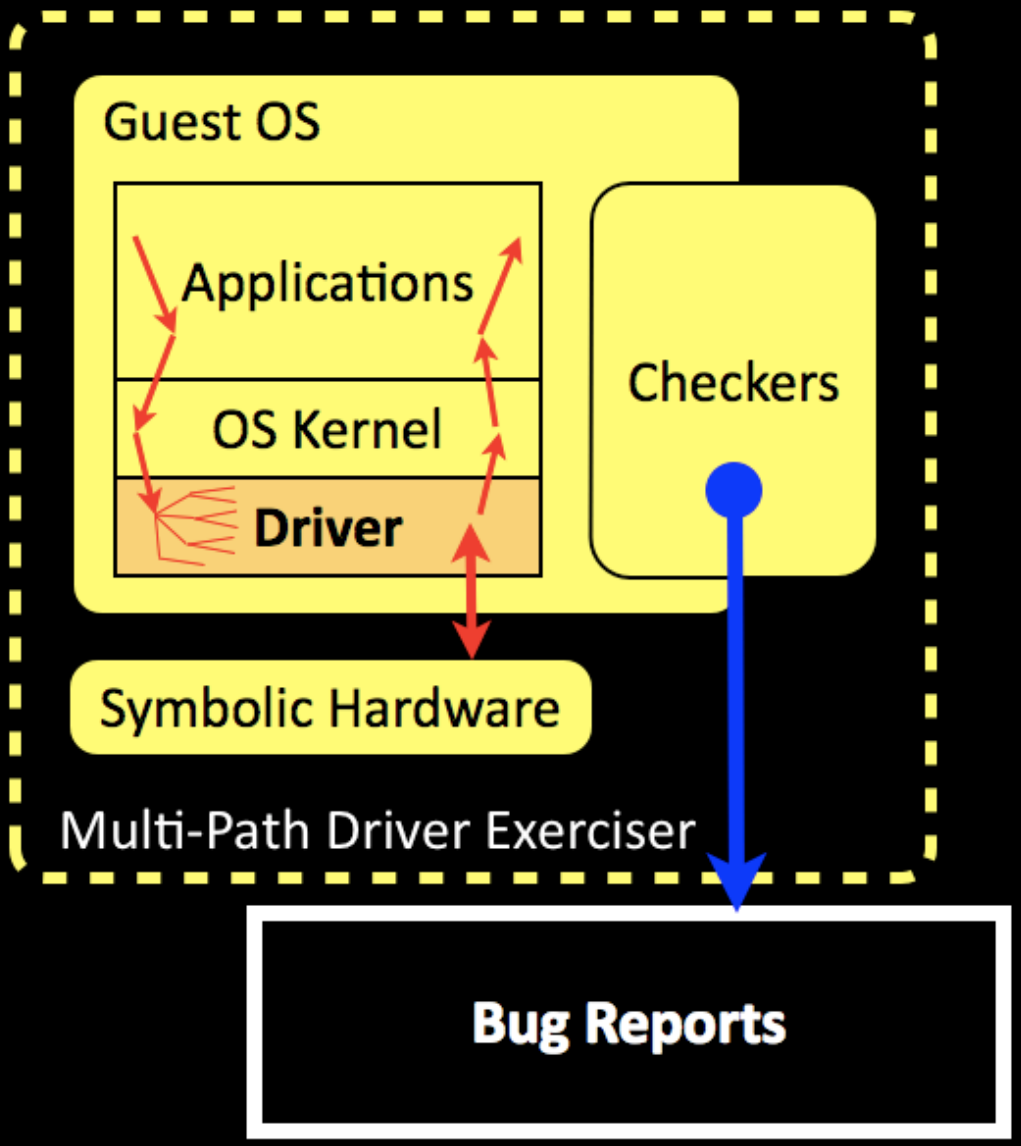
- Driver Exerciser

Outline



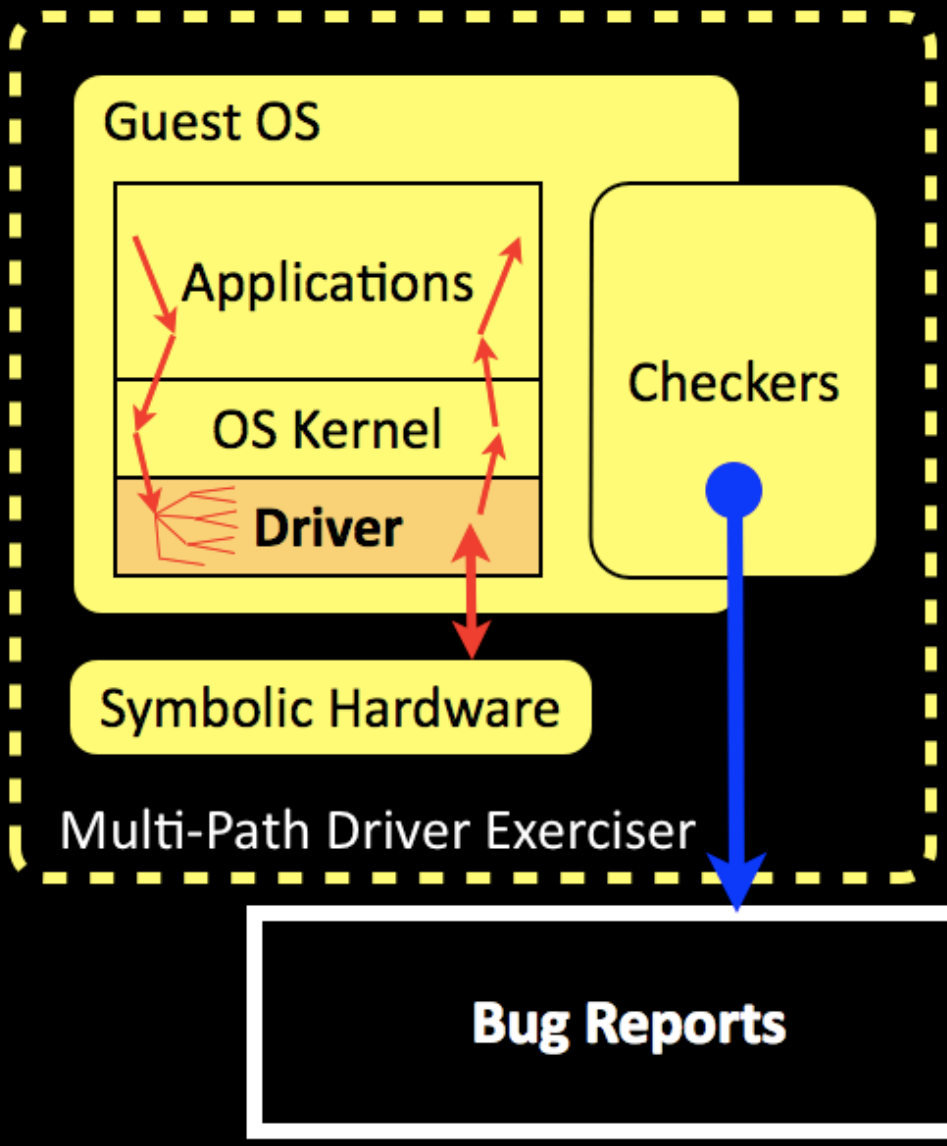
- Driver Exerciser
- Bug Checkers

Outline



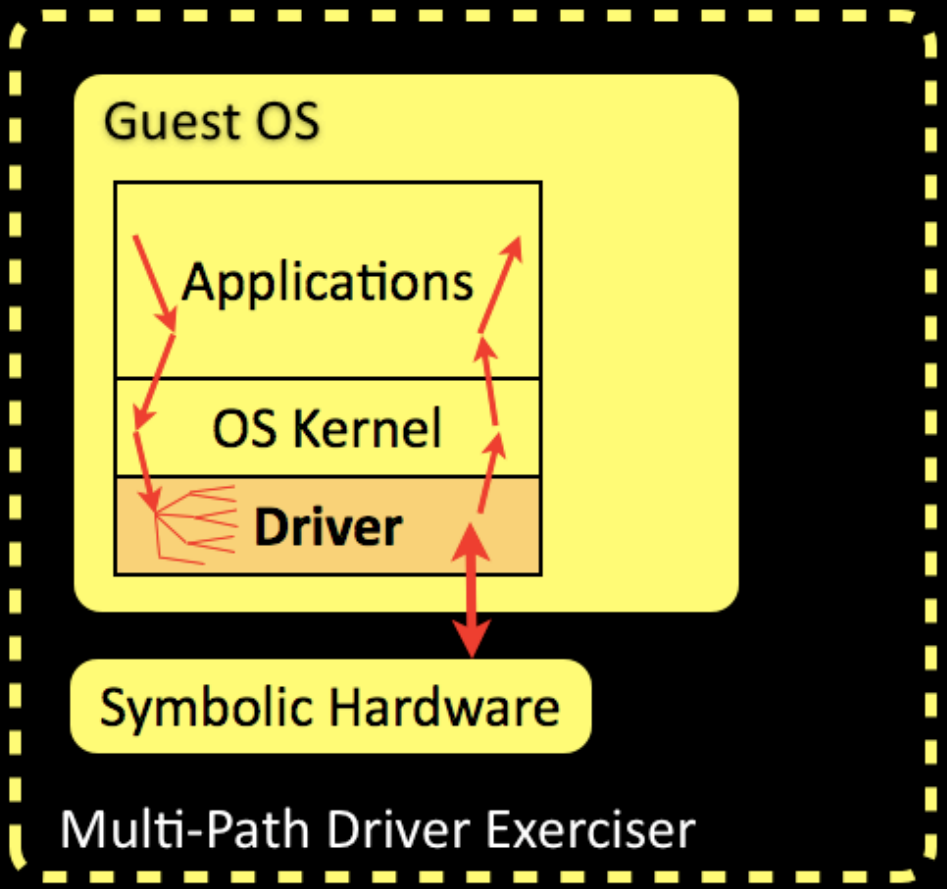
- Driver Exerciser
- Bug Checkers
- Bug Reports

Outline



- Driver Exerciser
- Bug Checkers
- Bug Reports
- Evaluation

Outline



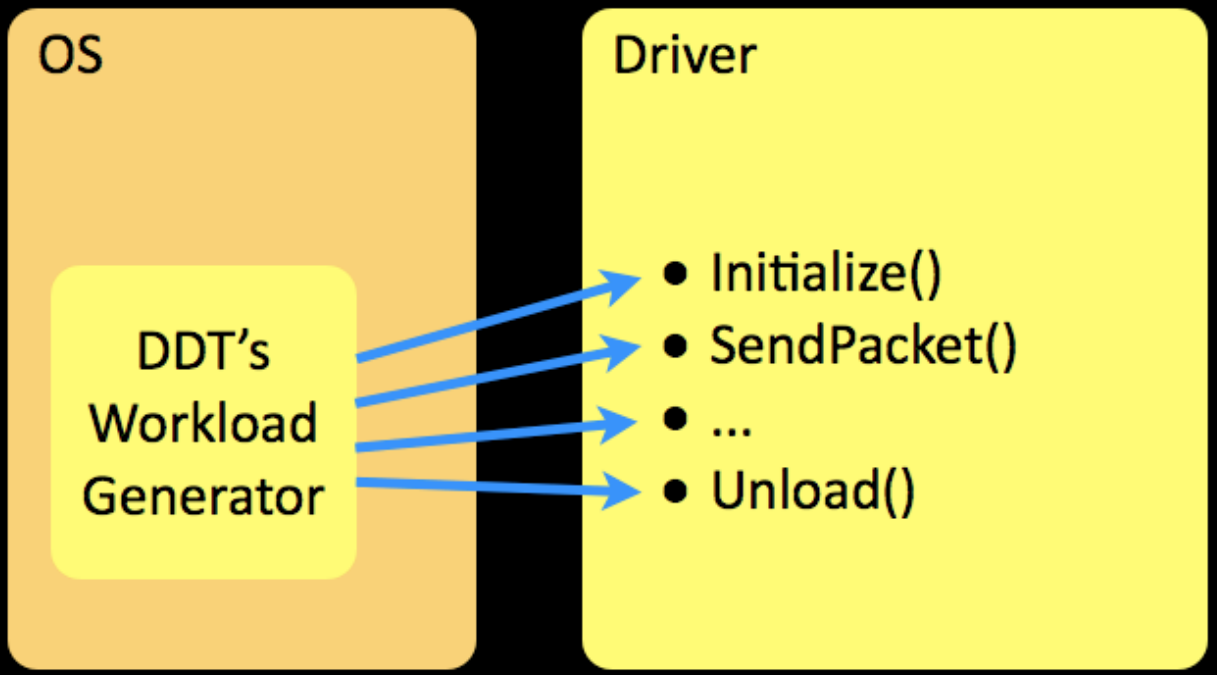
- **Driver Exerciser**
- Bug Checkers
- Bug Reports
- Evaluation

Drivers

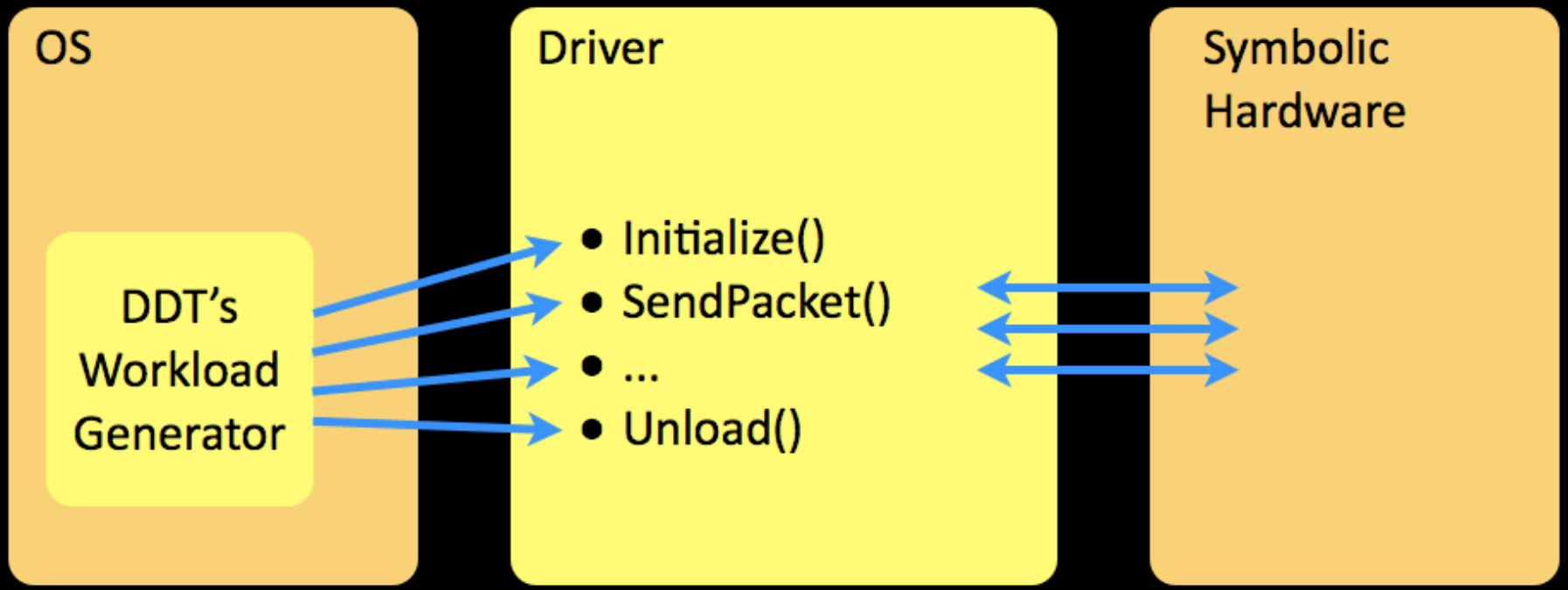
Driver

- Initialize()
- SendPacket()
- ...
- Unload()

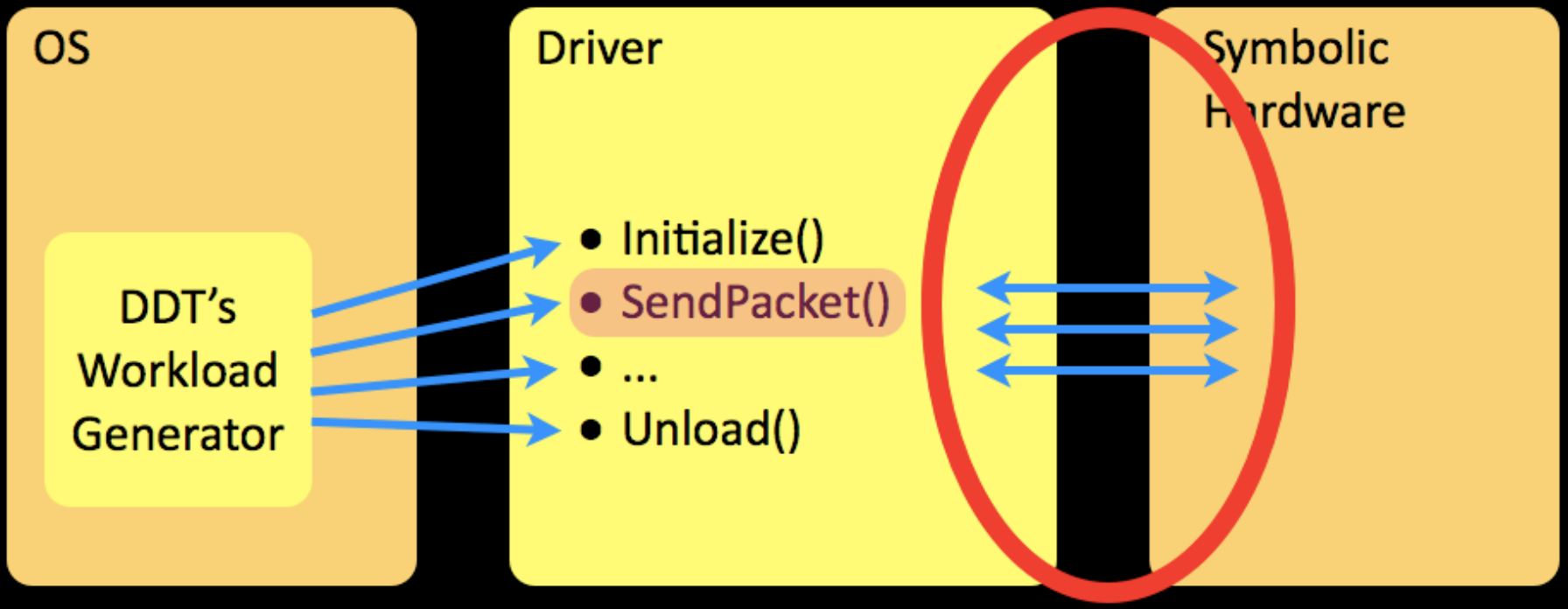
Drivers


OS
DDT's
Workload
Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()

Drivers


OS
DDT's Workload Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()
Symbolic Hardware

Drivers


OS
DDT's Workload Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()
Symbolic Hardware

Symbolic Execution

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```

status = READY

Symbolic Execution

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```

status = α

Symbolic Execution

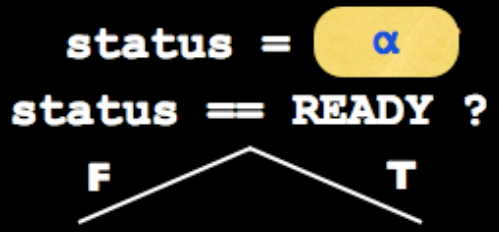
```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```

status = α

status == READY ?

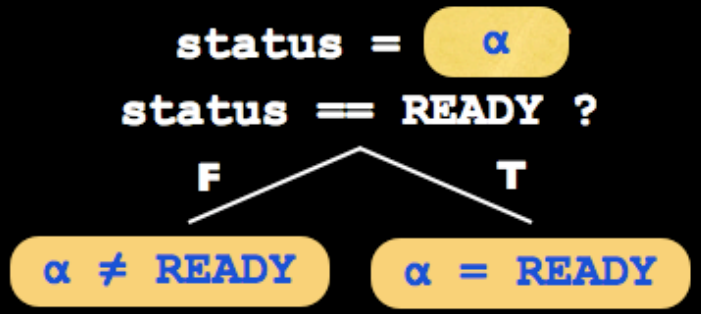
Symbolic Execution

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```

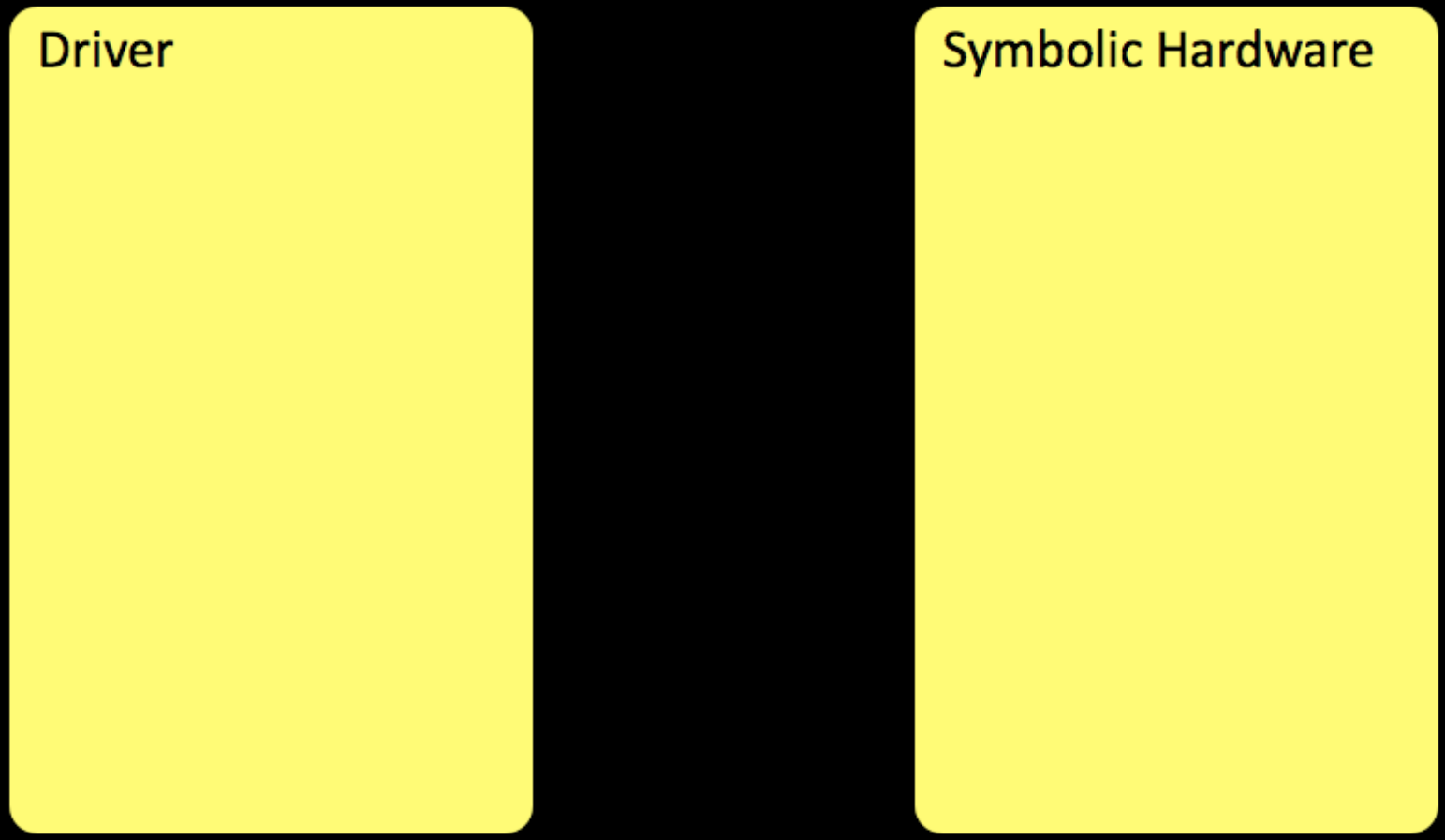


Symbolic Execution

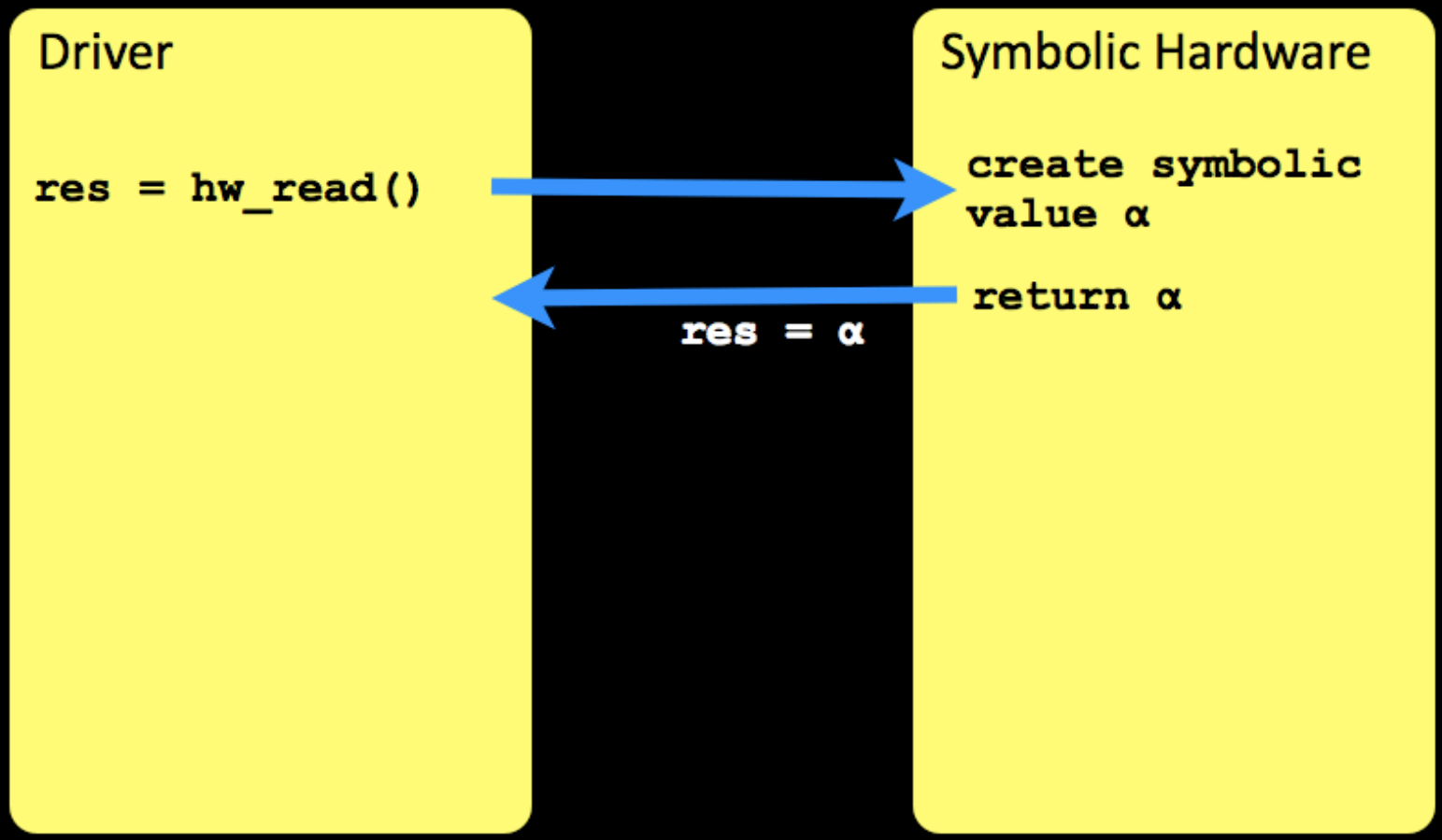
```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```



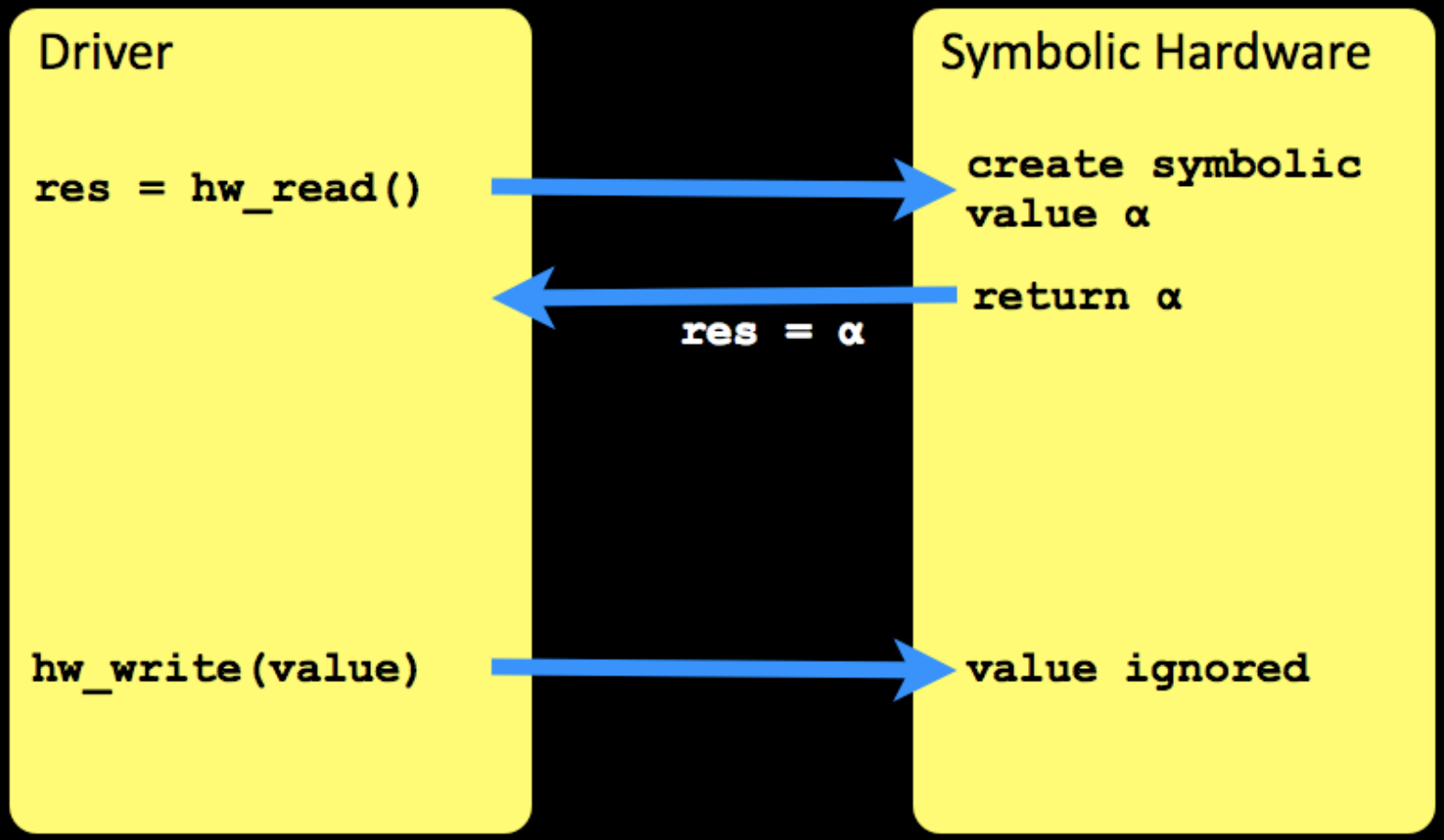
Driver-Hardware Interface



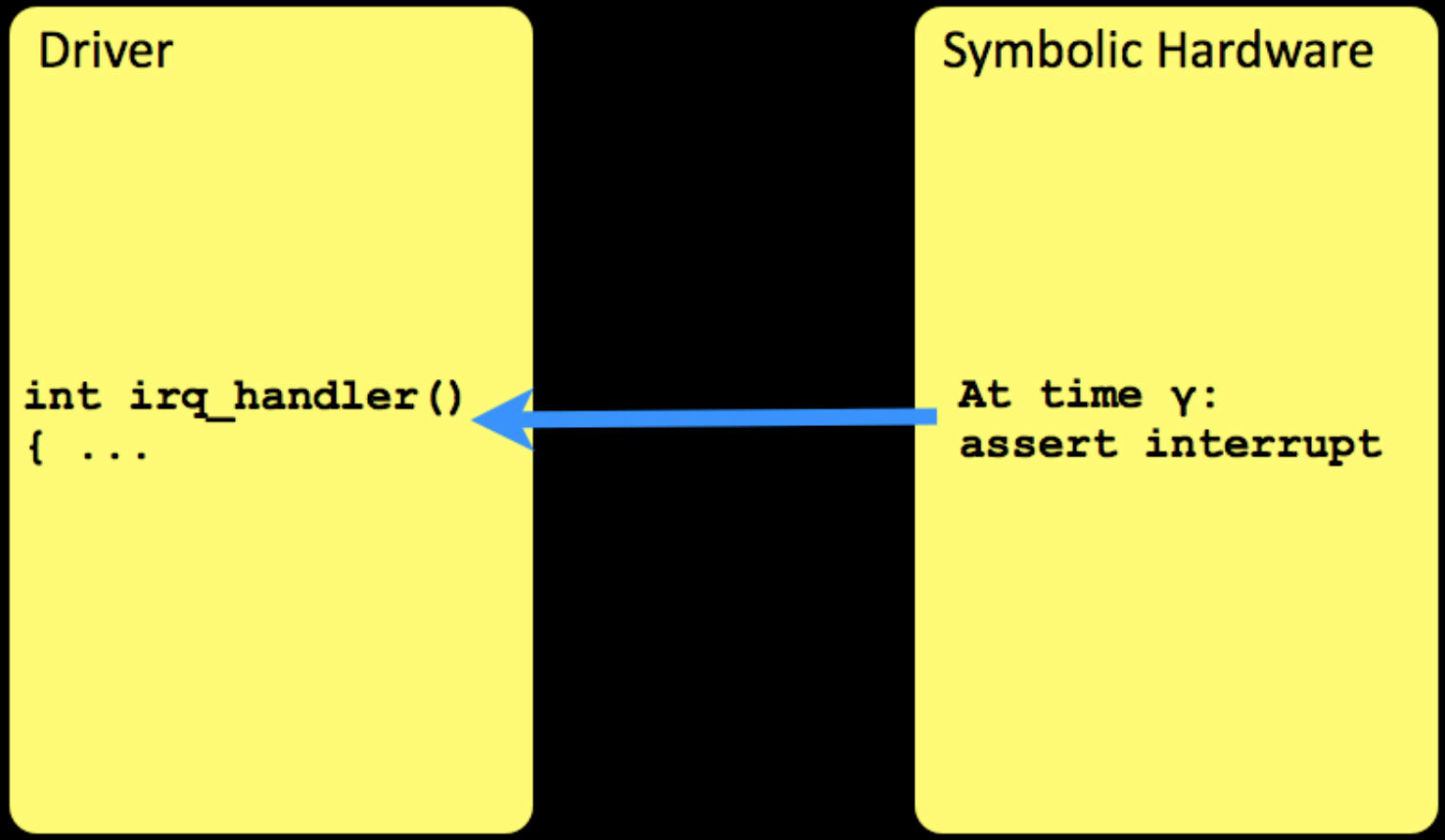
Driver-Hardware Interface



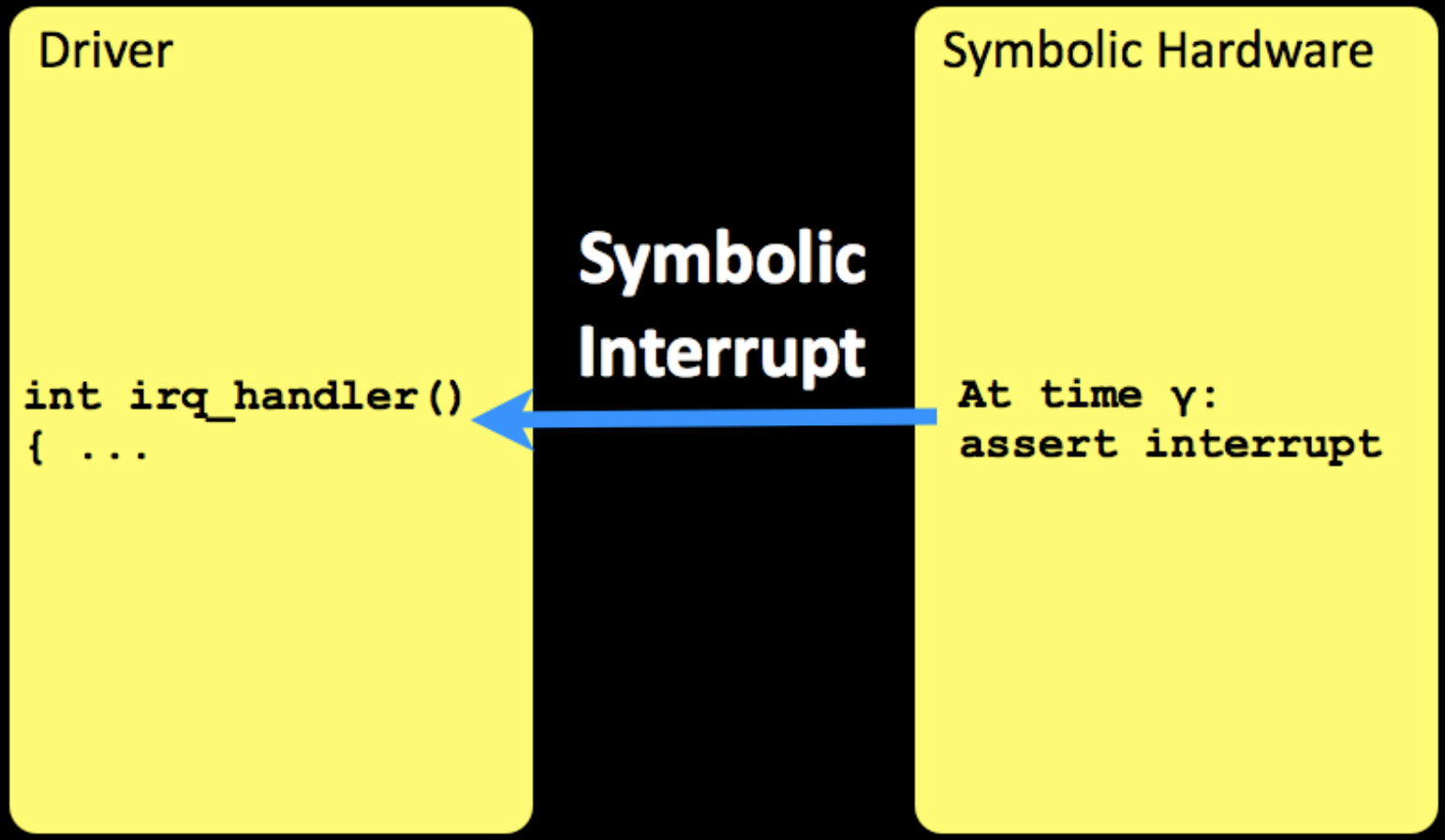
Driver-Hardware Interface



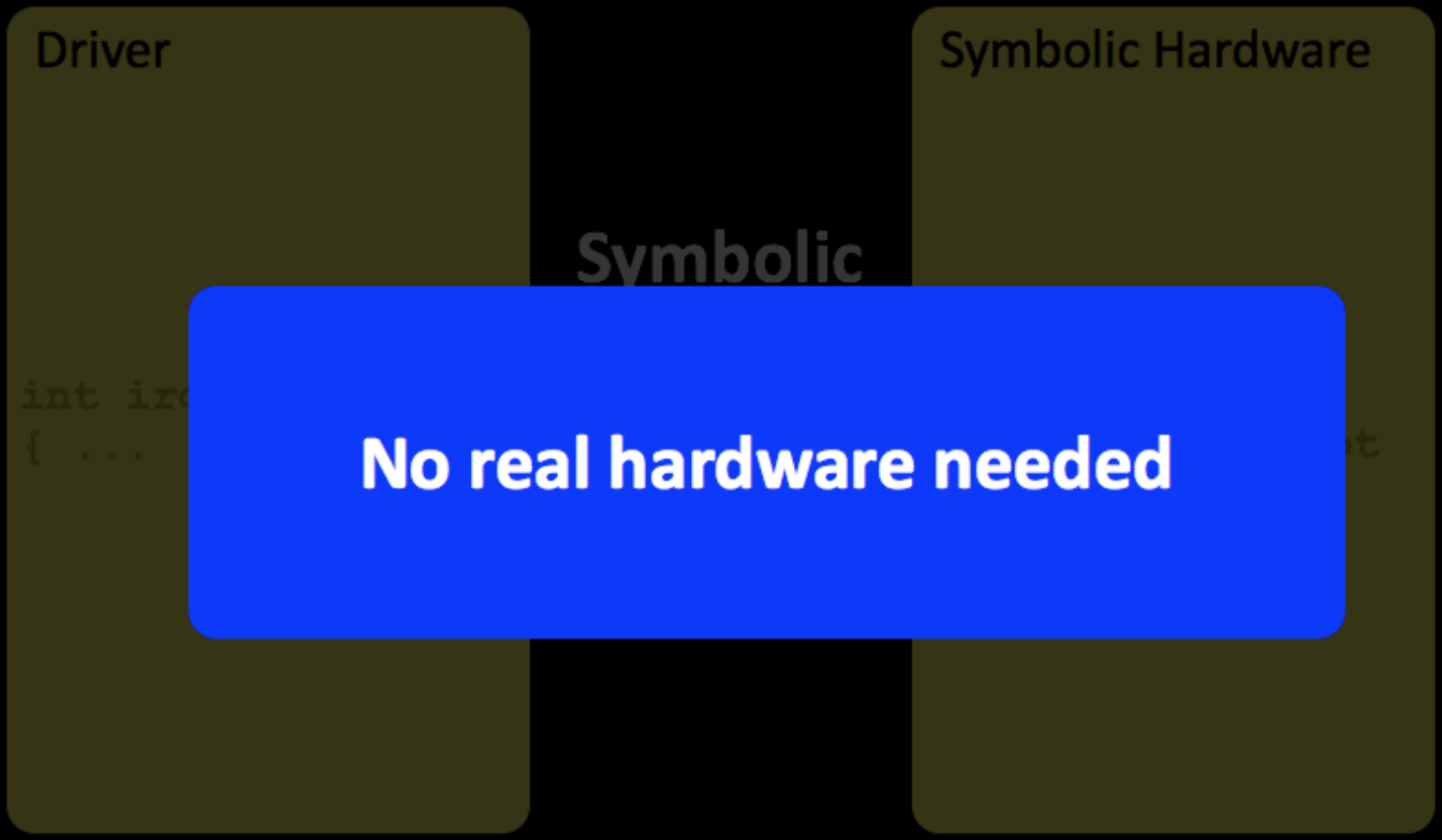
Driver-Hardware Interface



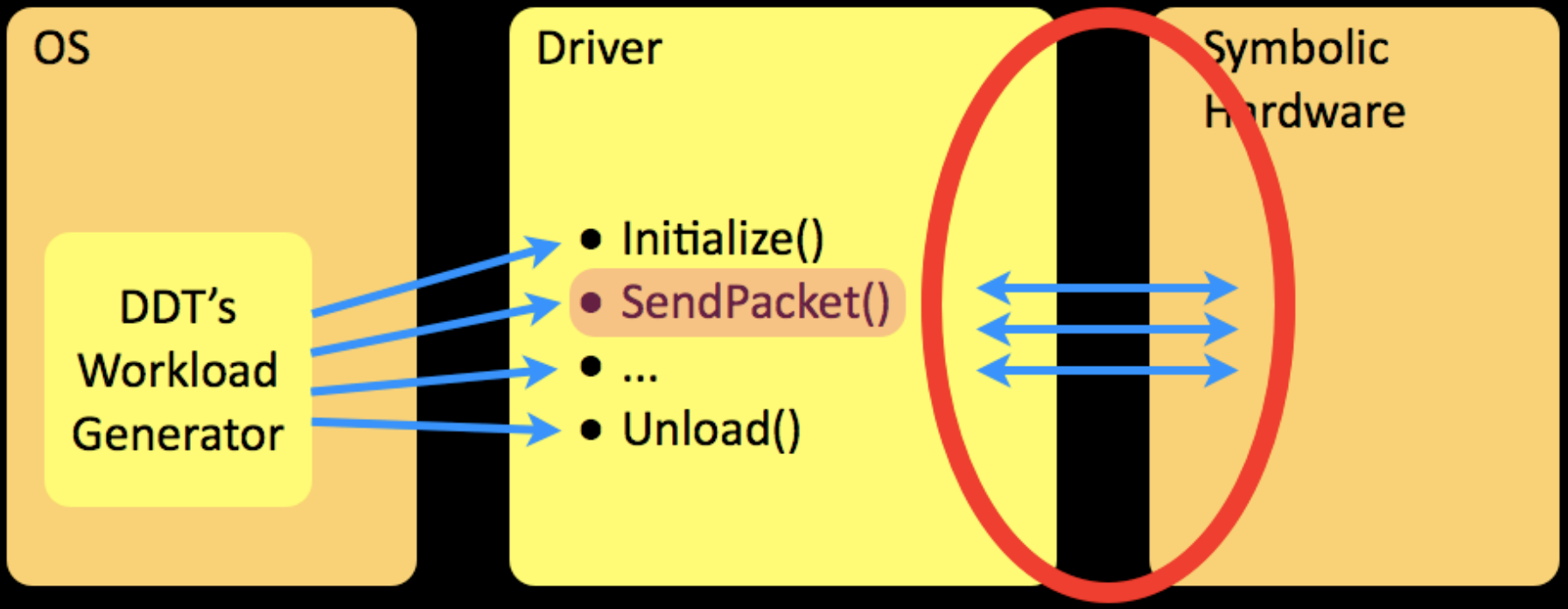
Driver-Hardware Interface



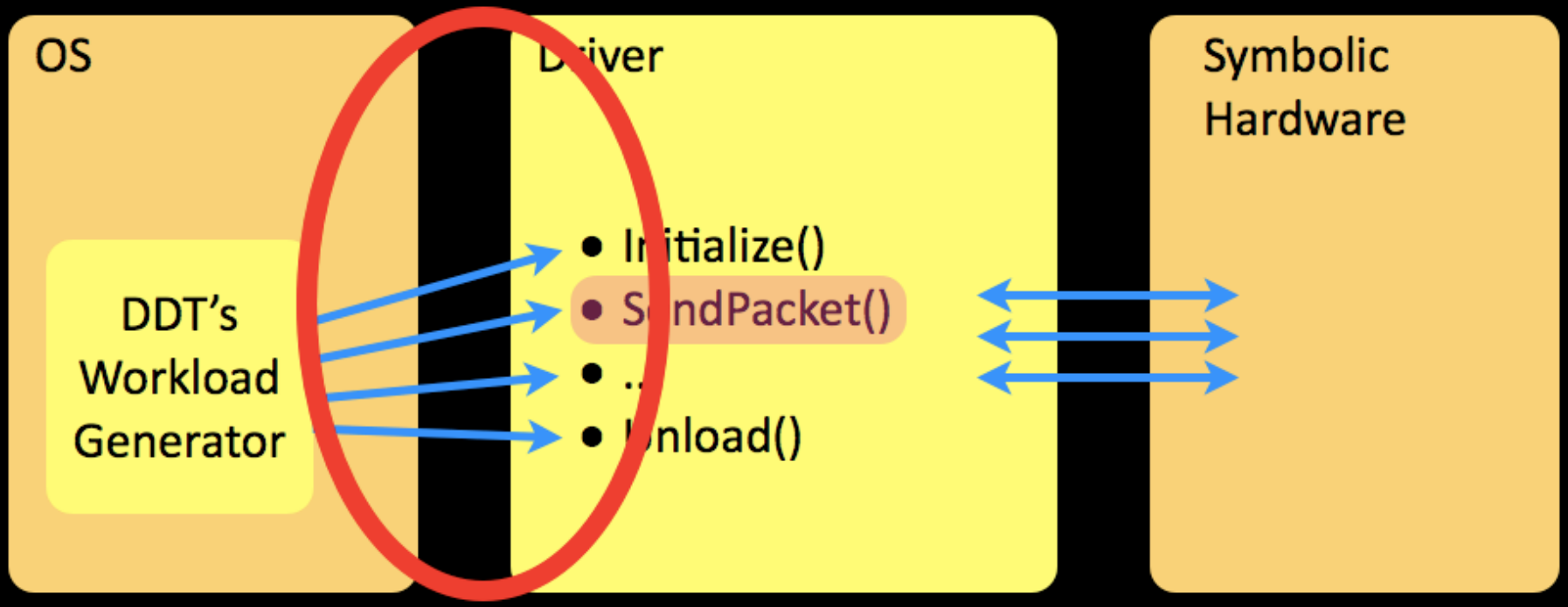
Driver-Hardware Interface



Drivers

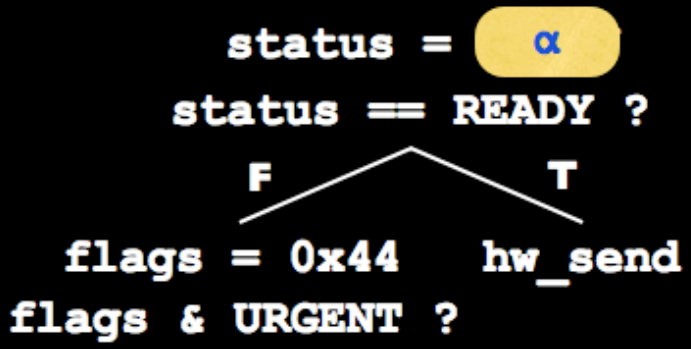

OS
DDT's Workload Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()
Symbolic Hardware

Drivers


OS
Driver
Symbolic Hardware
DDT's Workload Generator
• Initialize()
• SendPacket()
• ..
• Unload()

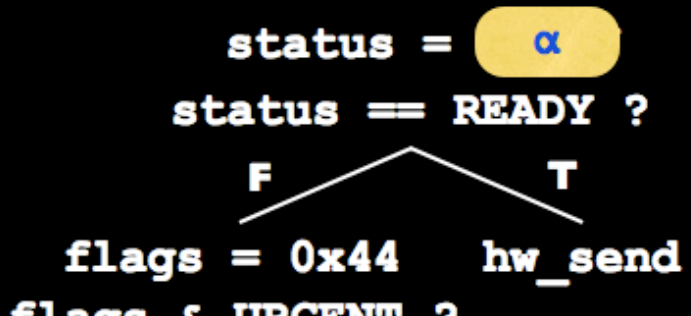
Kernel-Driver Interface

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```



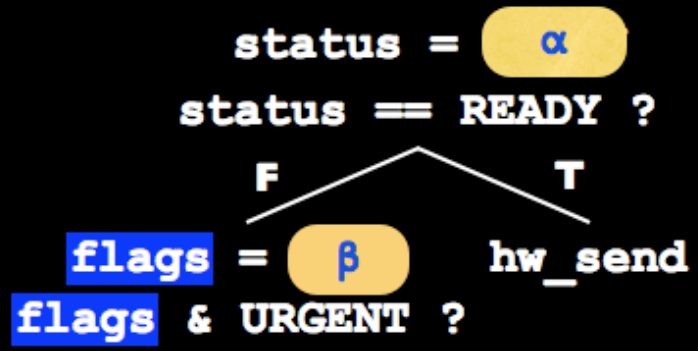
Kernel-Driver Interface

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```



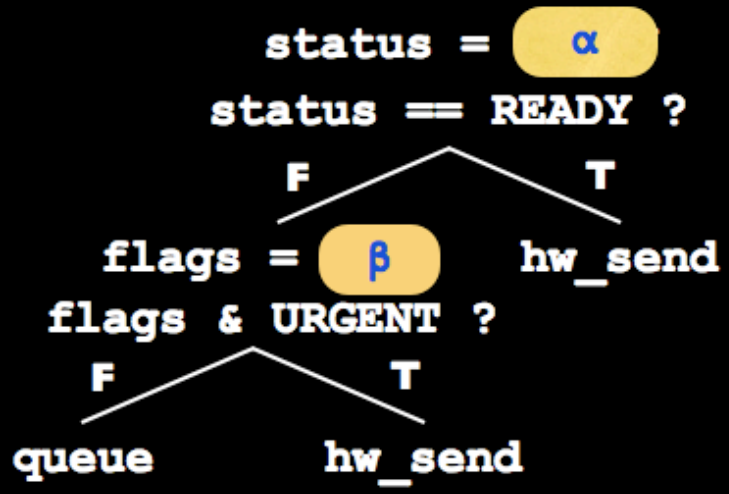
Kernel-Driver Interface

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```

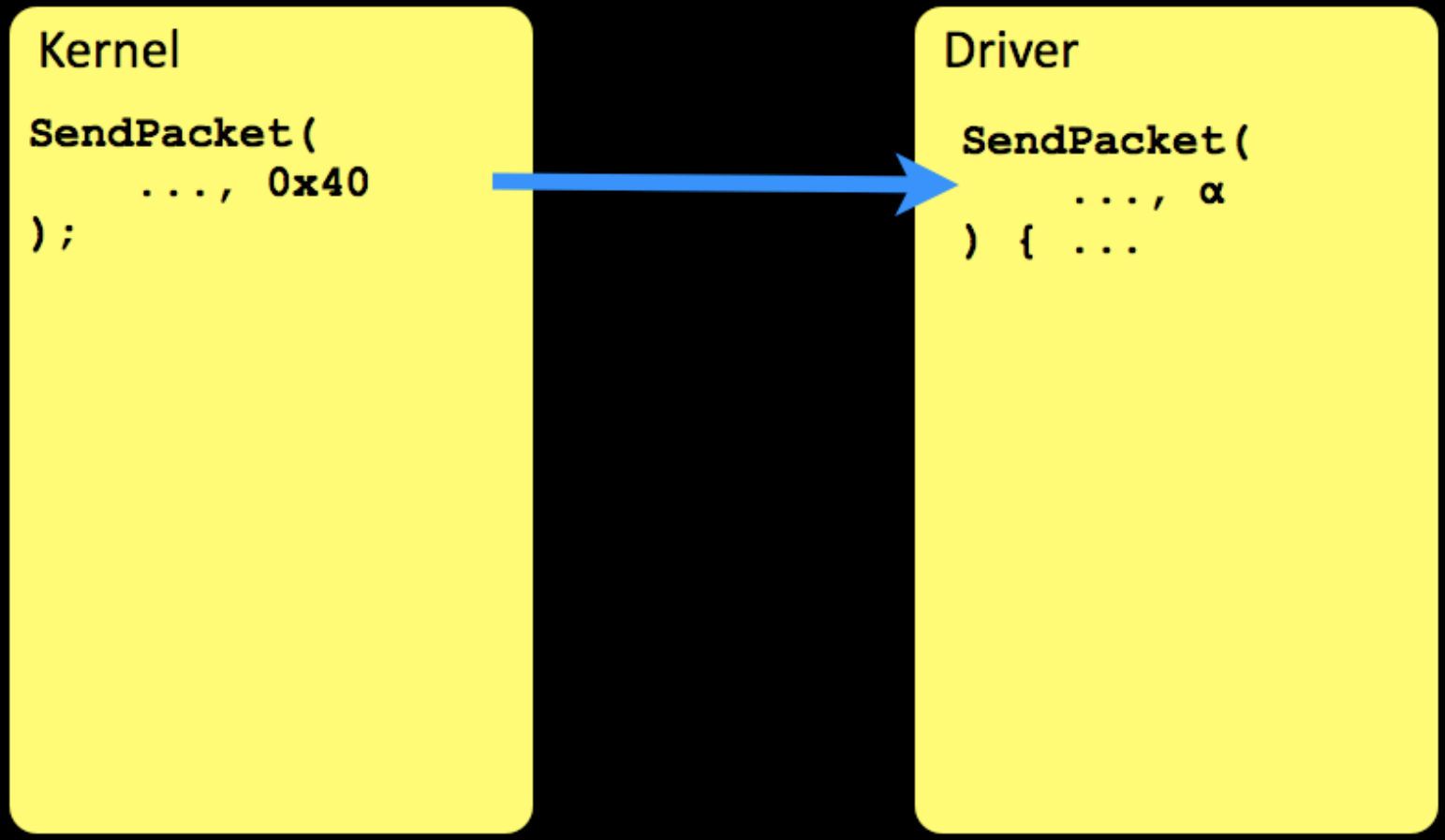


Kernel-Driver Interface

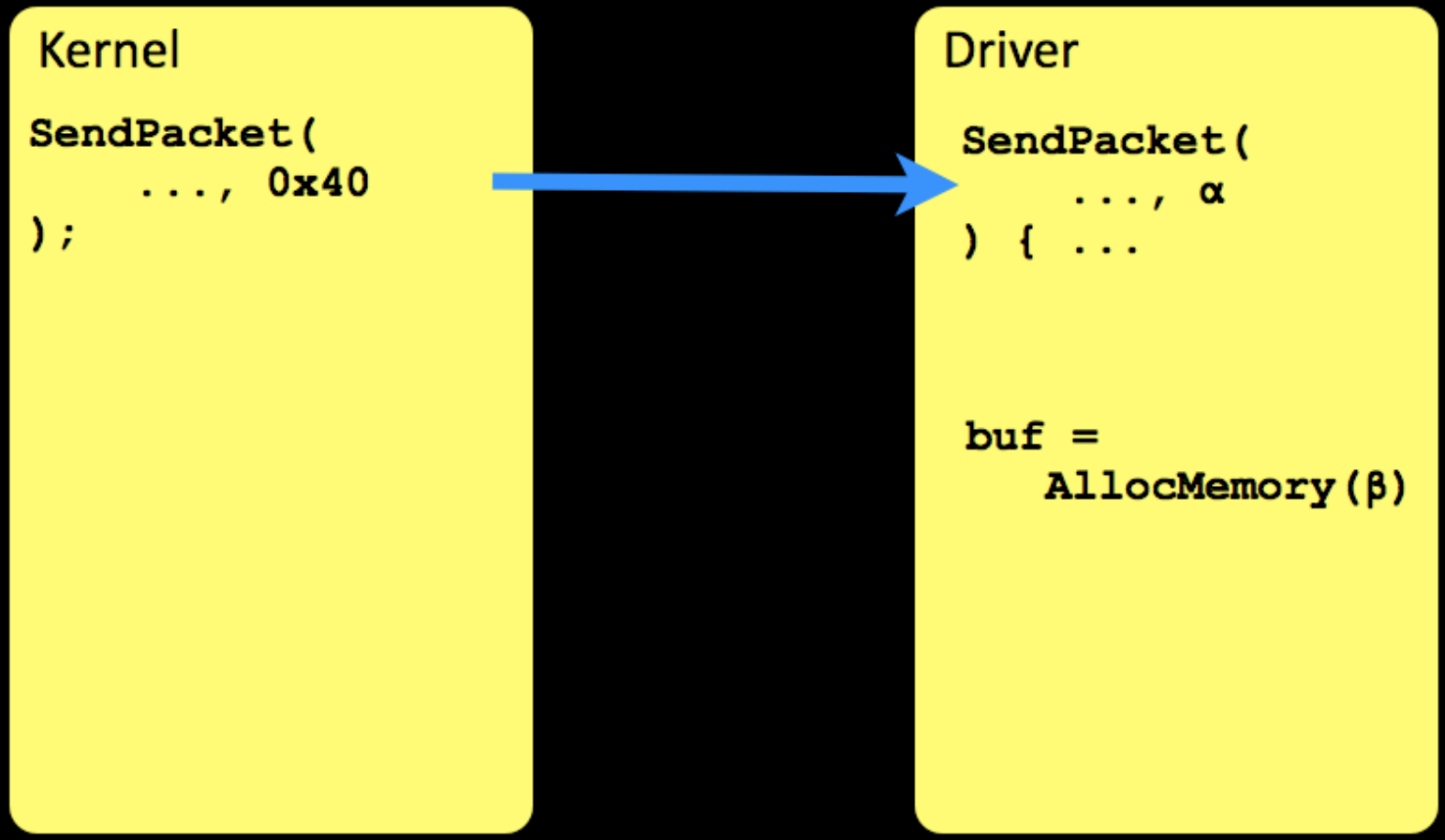
```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if(status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```



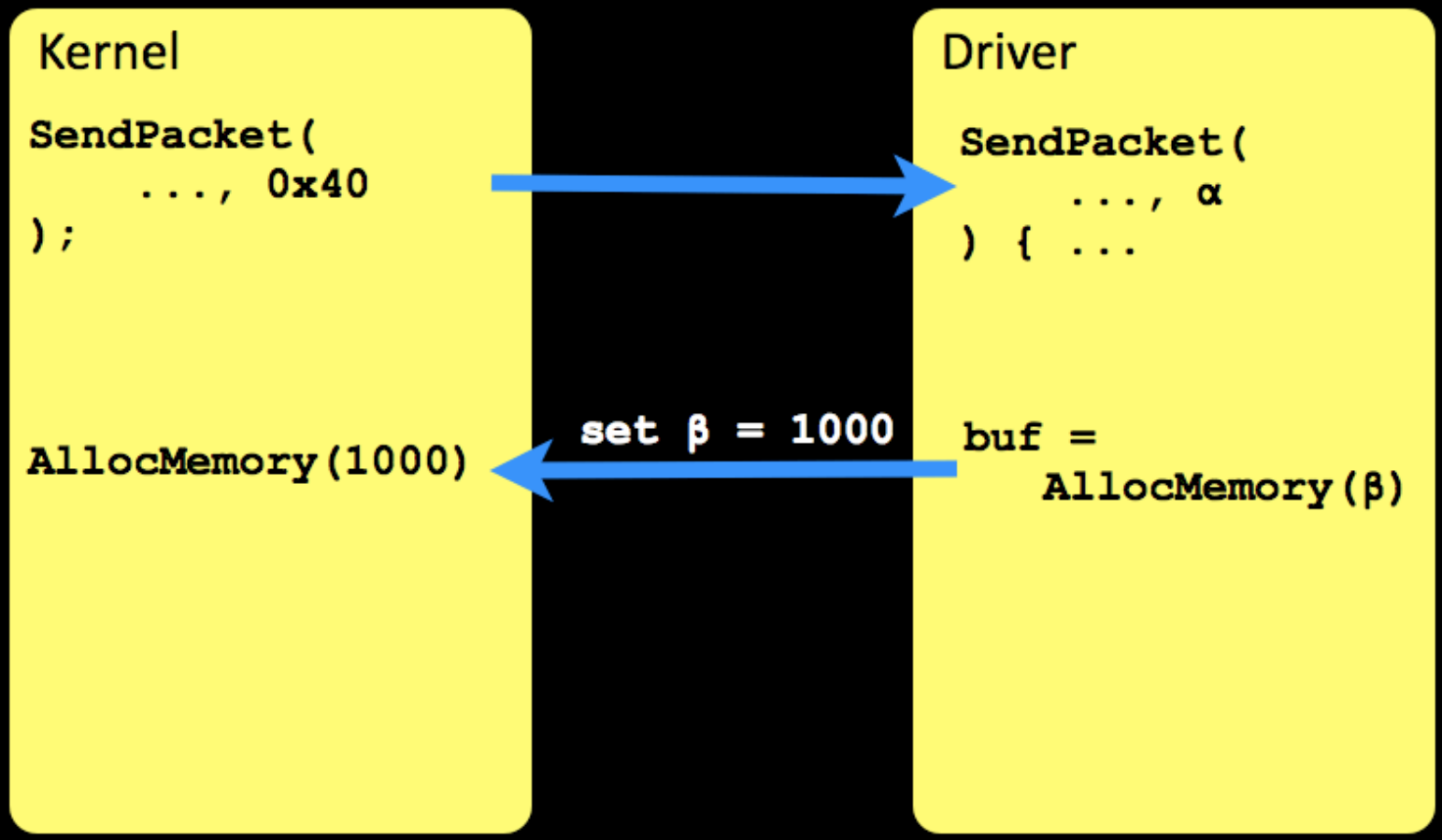
Kernel-Driver Interface



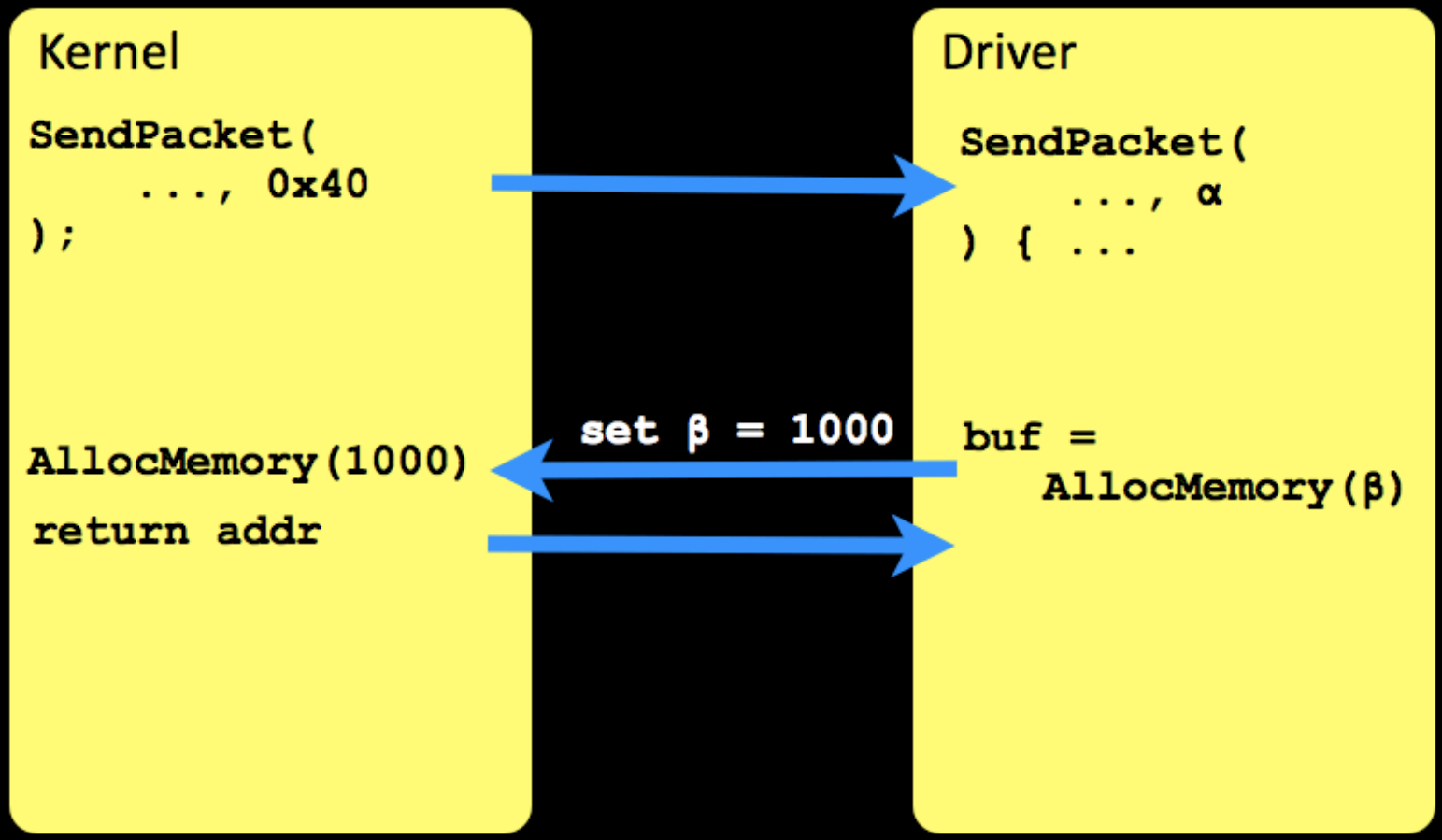
Kernel-Driver Interface



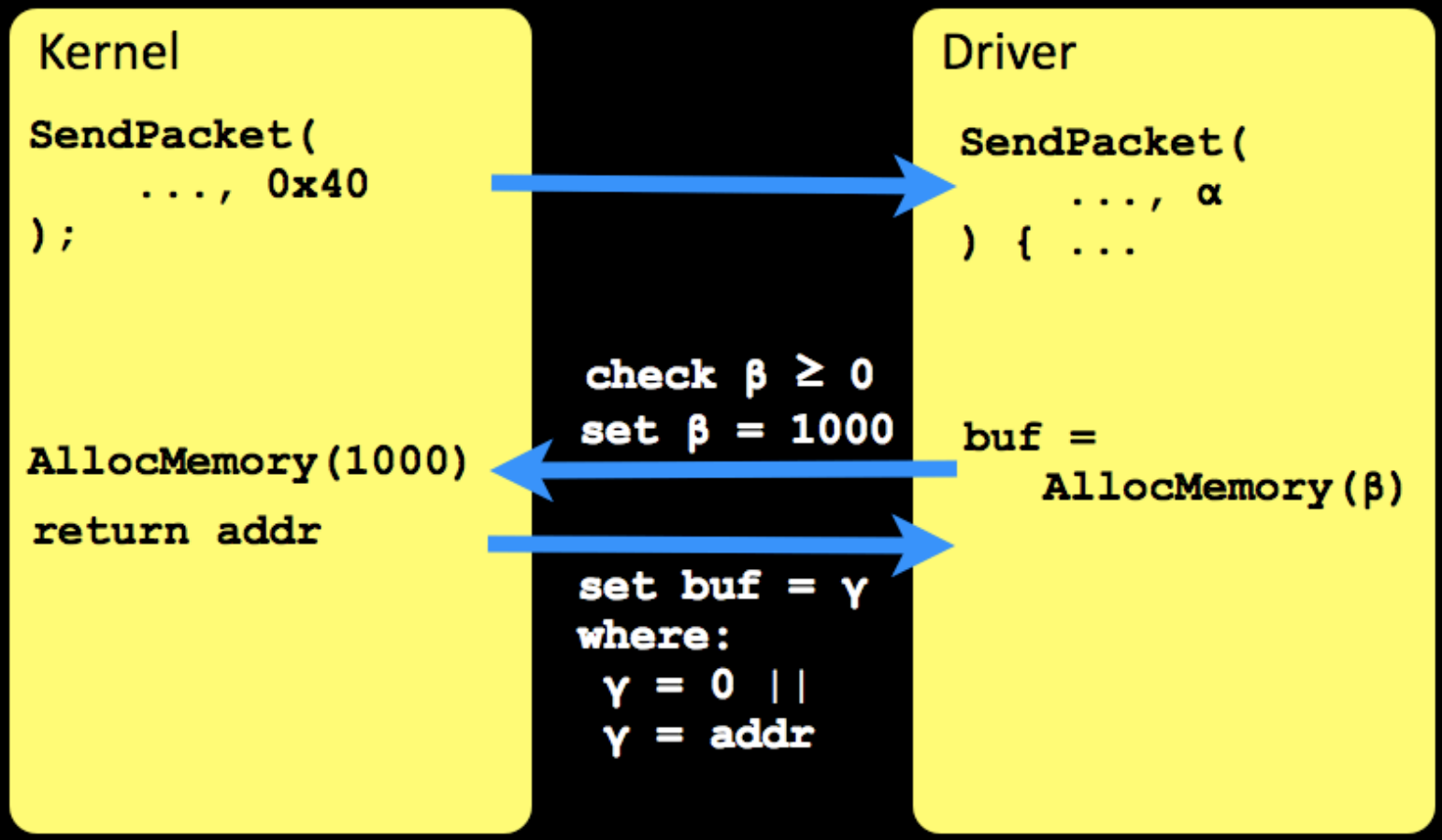
Kernel-Driver Interface



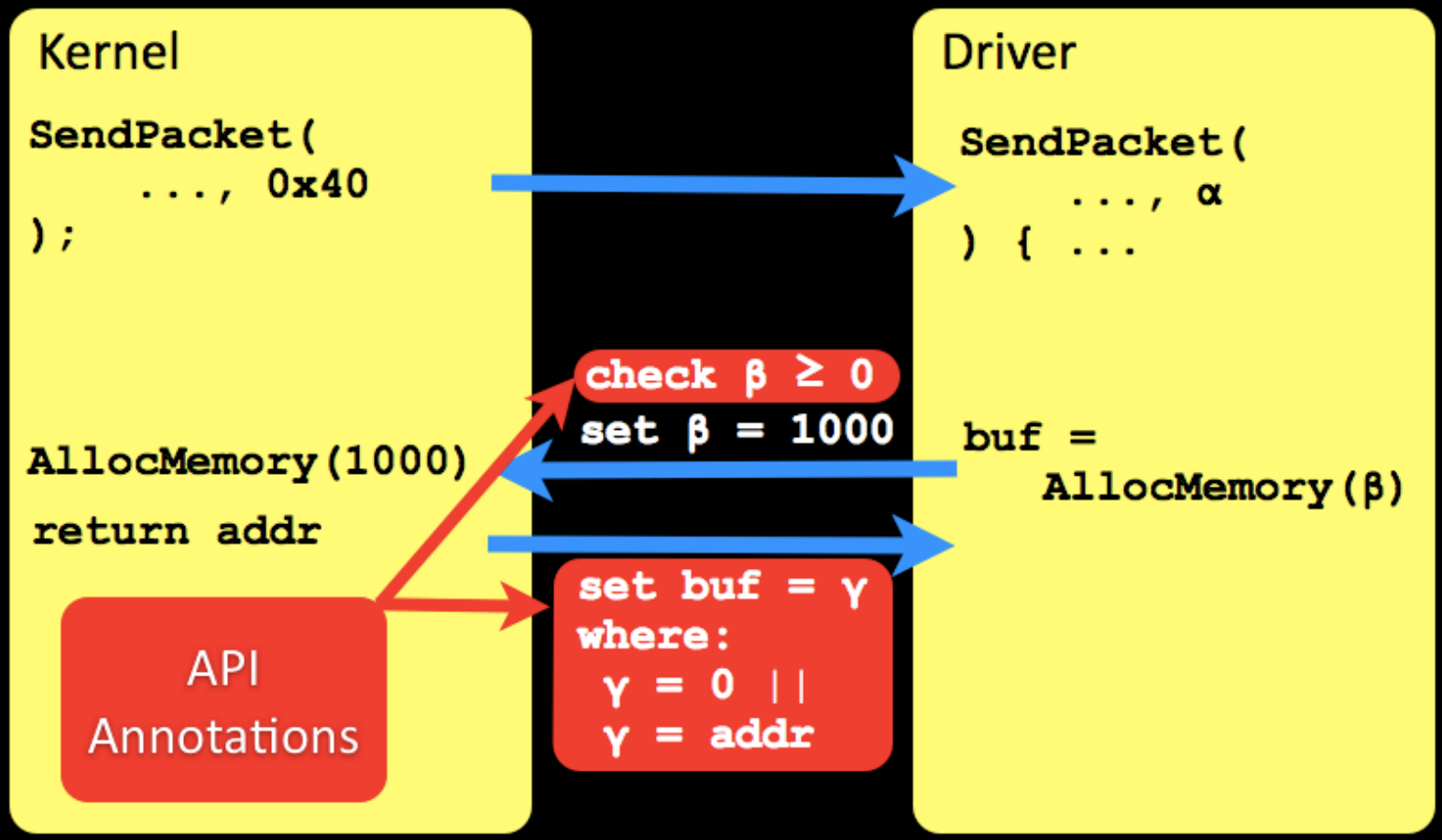
Kernel-Driver Interface



Kernel-Driver Interface



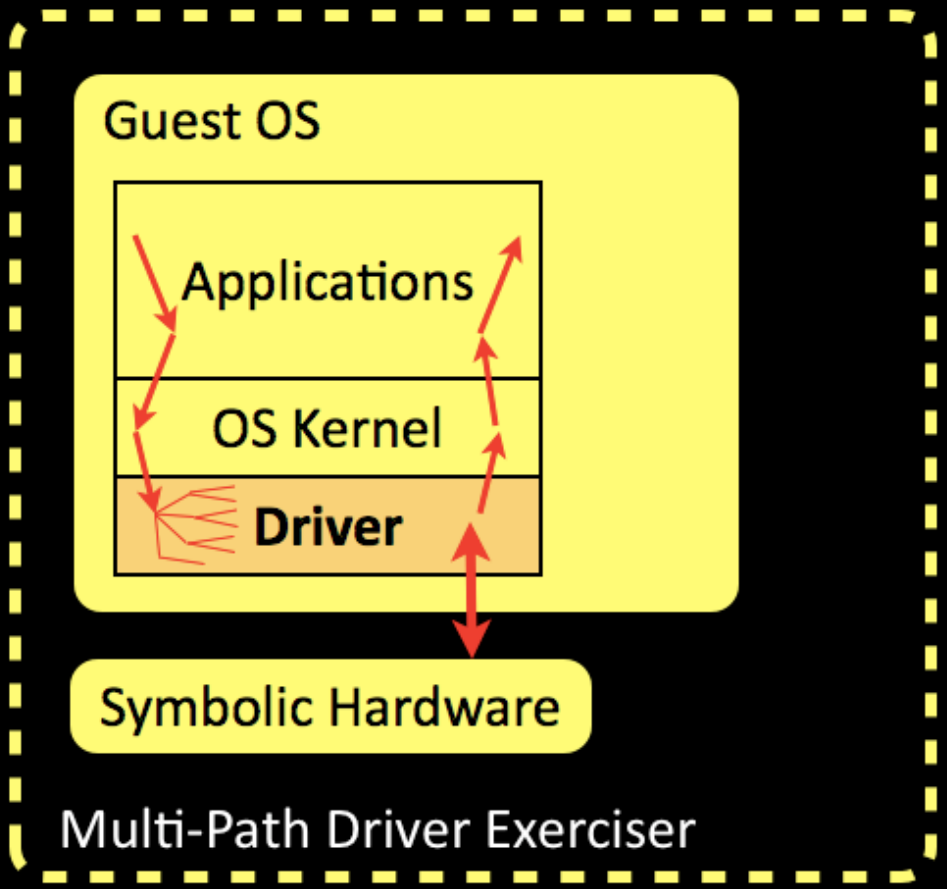
Kernel-Driver Interface



API Annotations

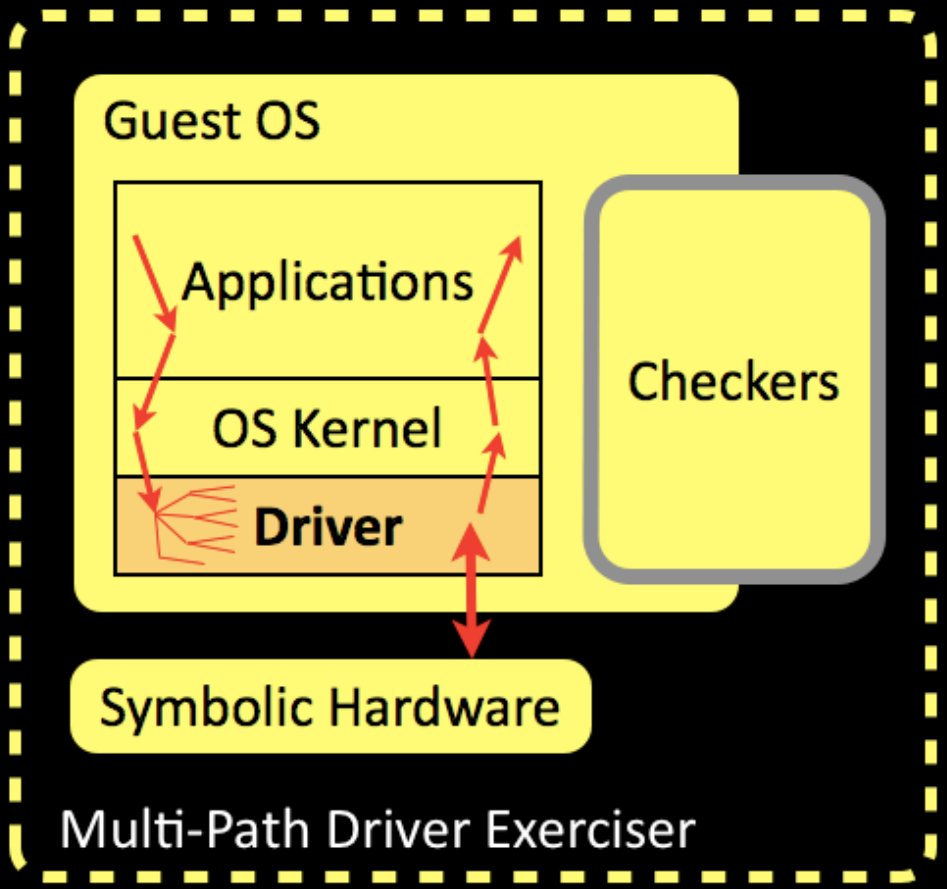
- Encode developer's knowledge of the API
  Enhance code coverage
  Check for API-specific bugs
- Easier to write than API models
  Days instead of years to develop
- Optional

Outline



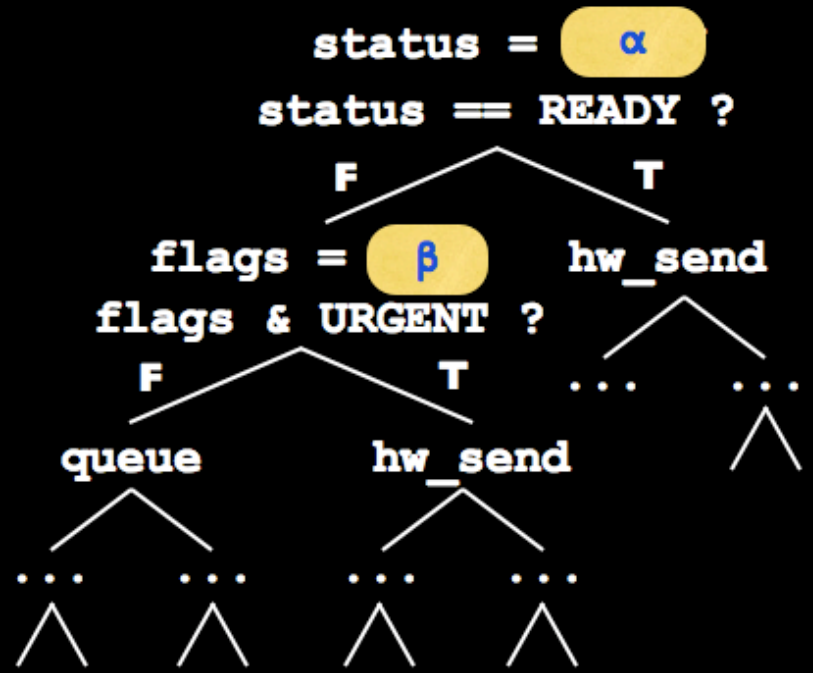
- **Driver Exerciser**
- Bug Checkers
- Bug Reports
- Evaluation

Outline



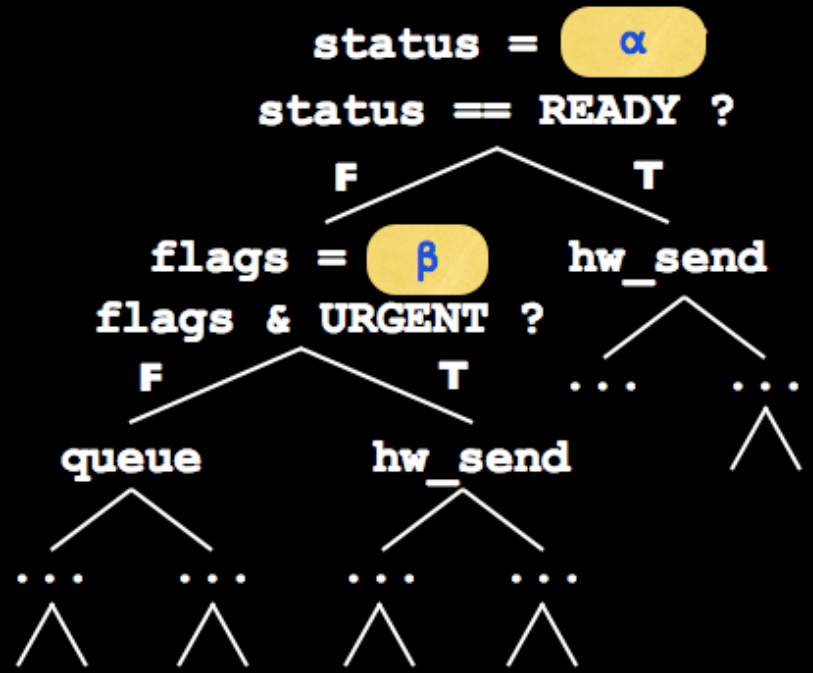
- Driver Exerciser
- **Bug Checkers**
- Bug Reports
- Evaluation

OS-Level Checkers



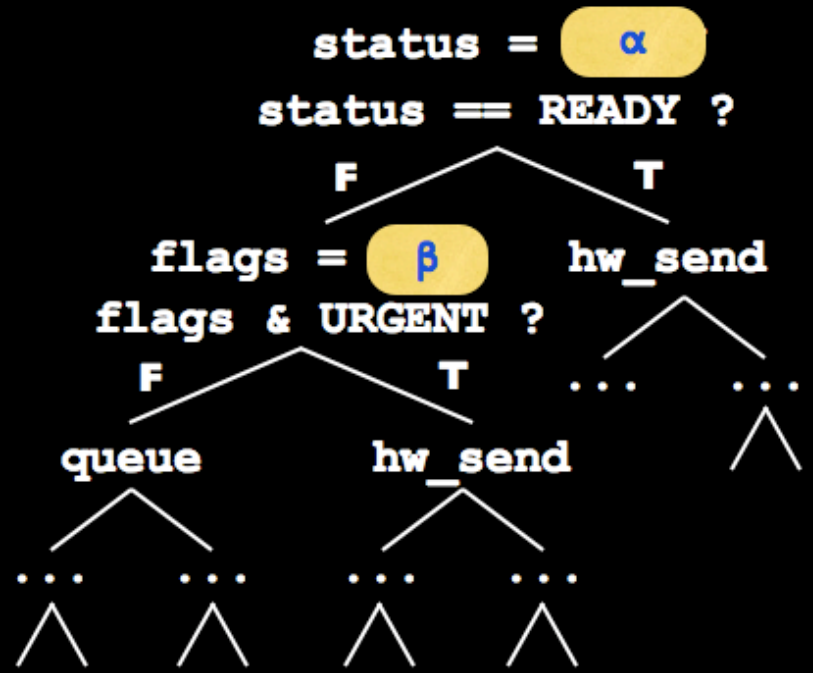
OS-Level Checkers

- Run in ***guest OS***



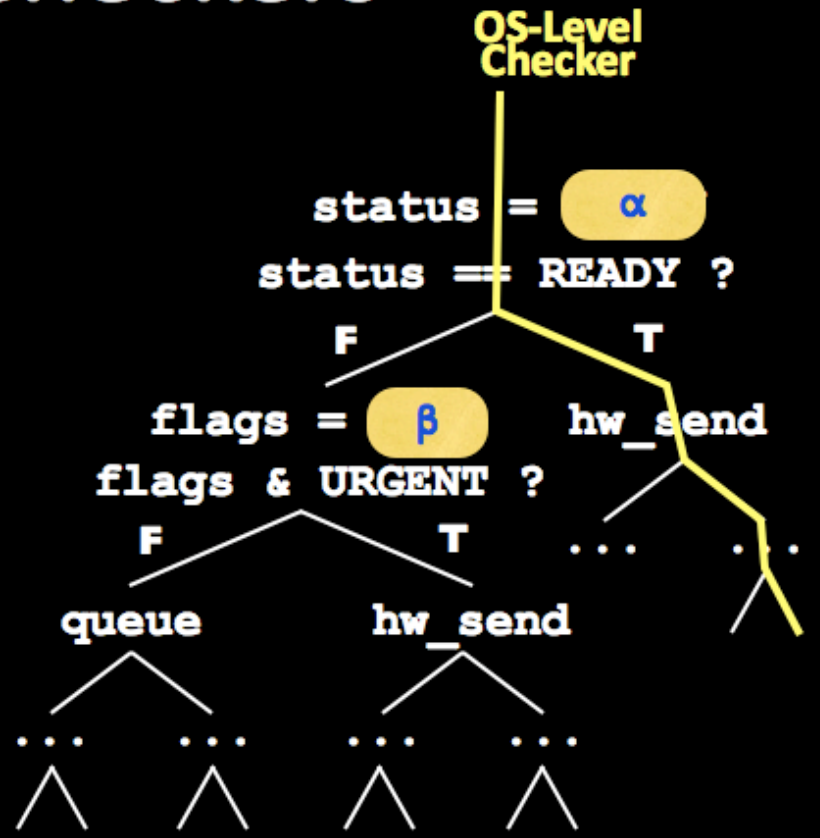
OS-Level Checkers

- Run in ***guest OS***
- ***Reuse*** existing single-path dynamic bug-finding tools

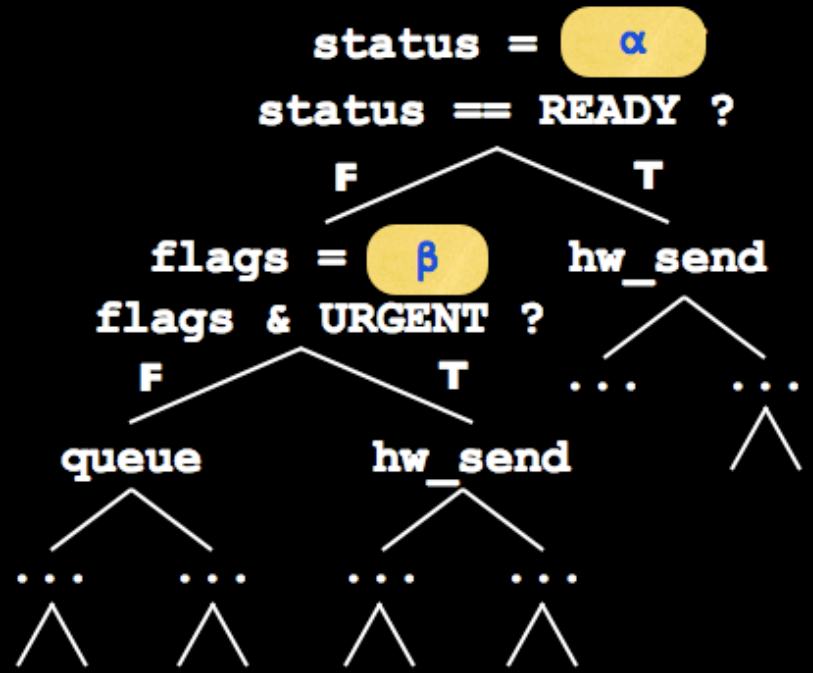


OS-Level Checkers

- Run in ***guest OS***
- ***Reuse*** existing single-path dynamic bug-finding tools
- On ***multiple*** paths

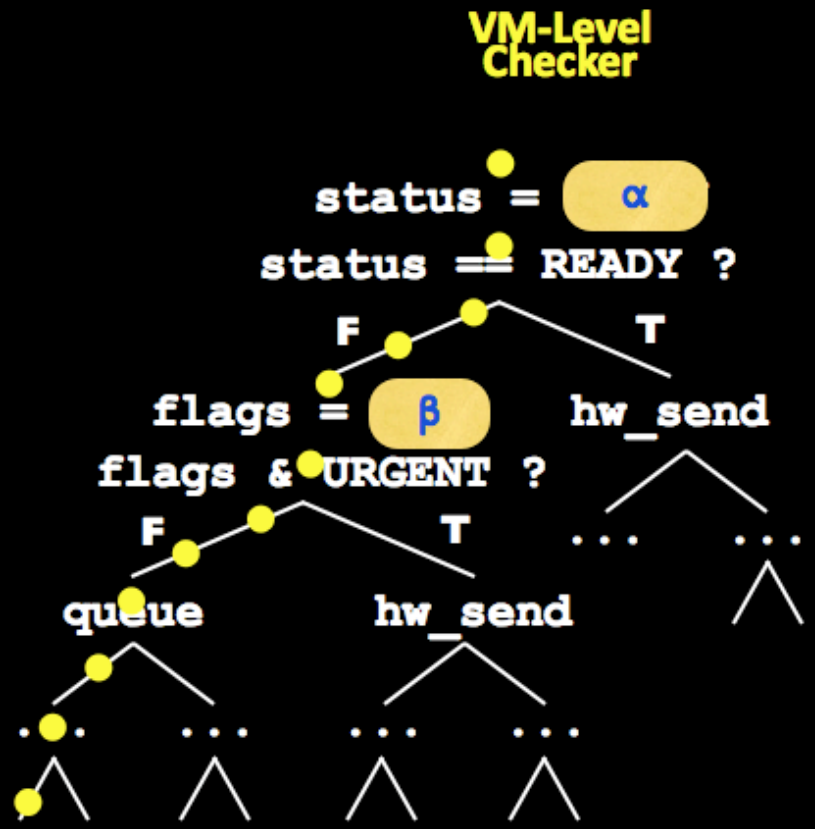


VM-Level Checkers



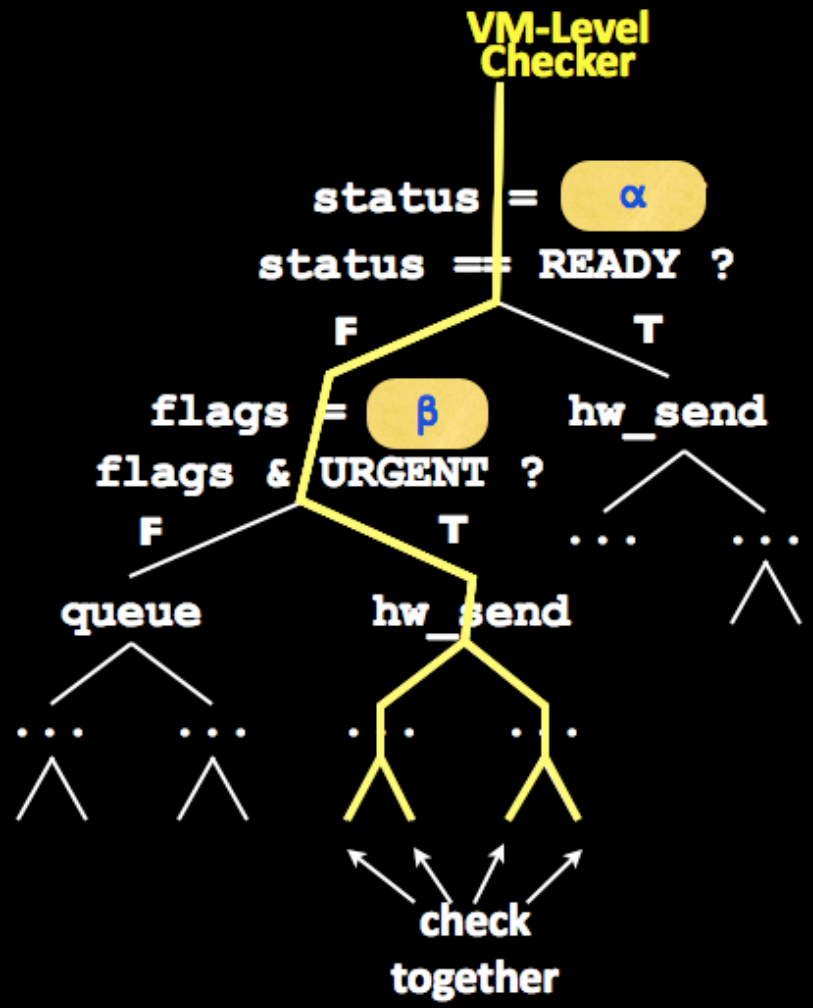
VM-Level Checkers

- Instruction-granularity instrumentation

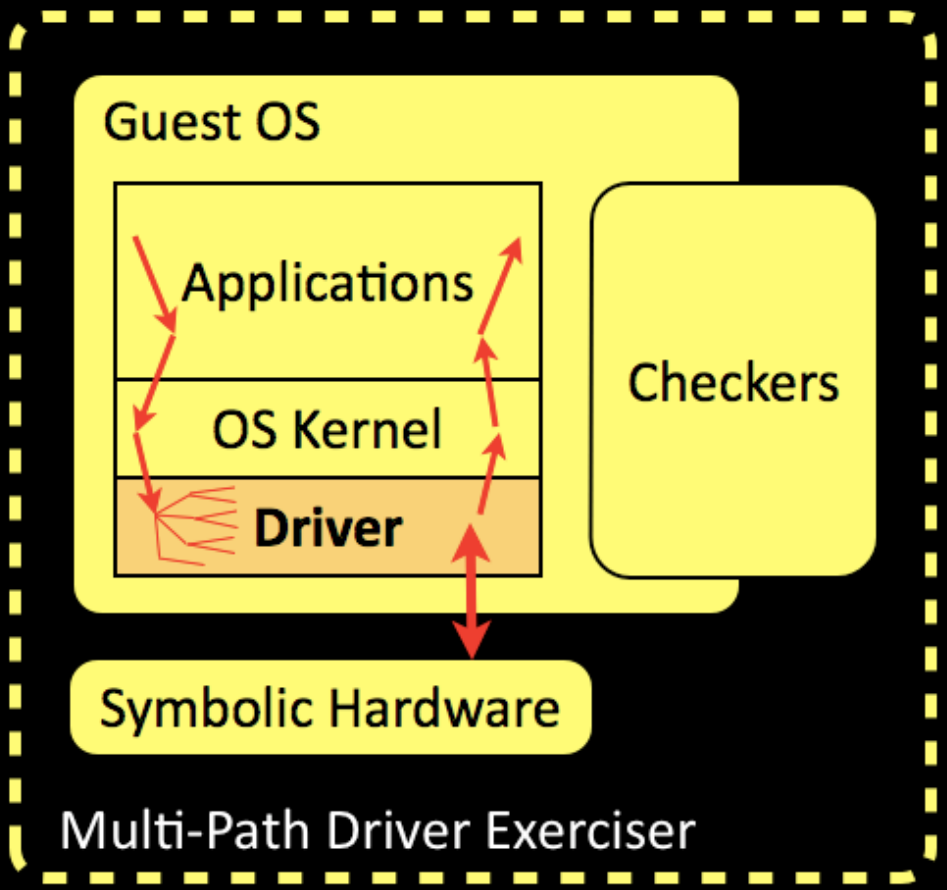


VM-Level Checkers

- Instruction-granularity instrumentation
- Multi-path analysis

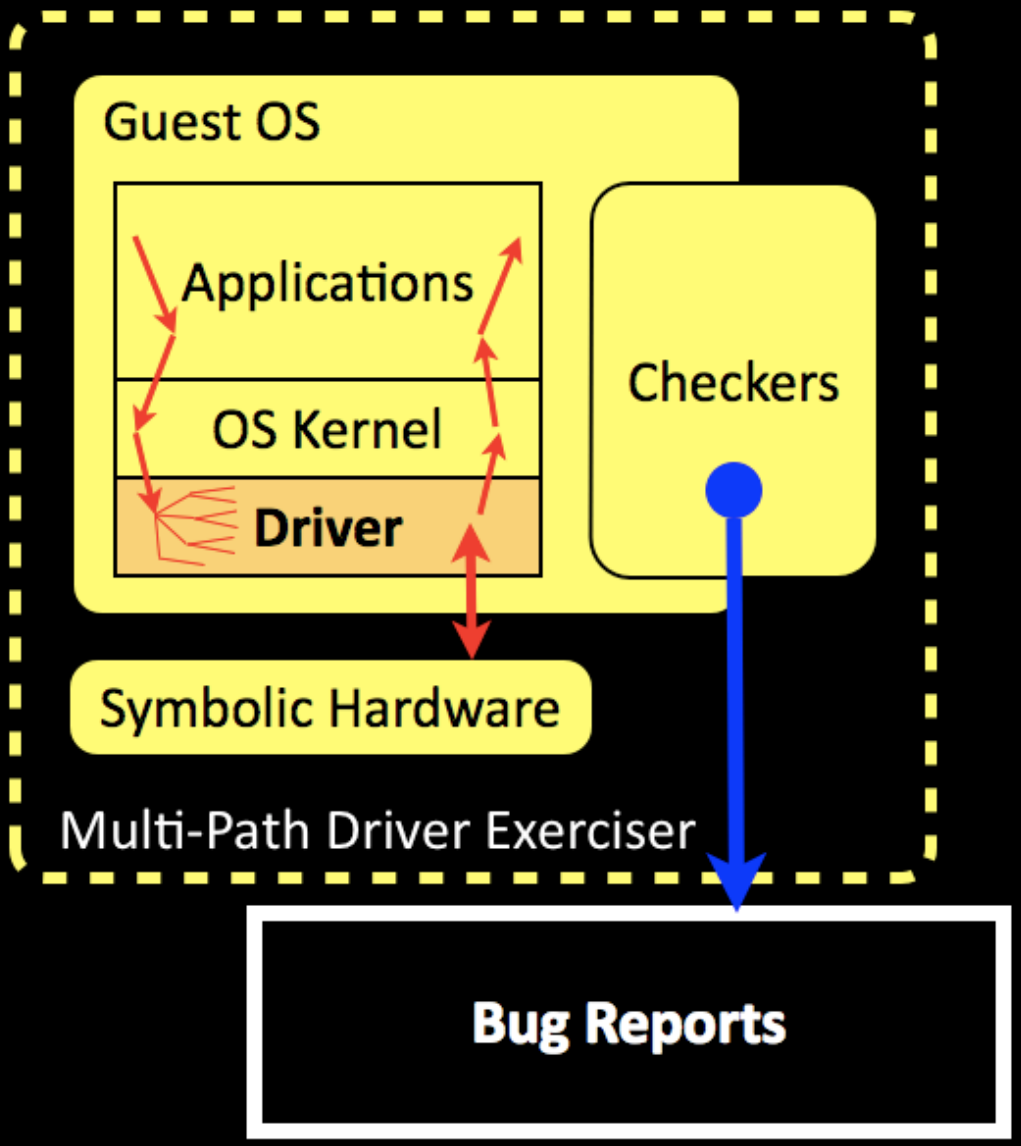


Outline



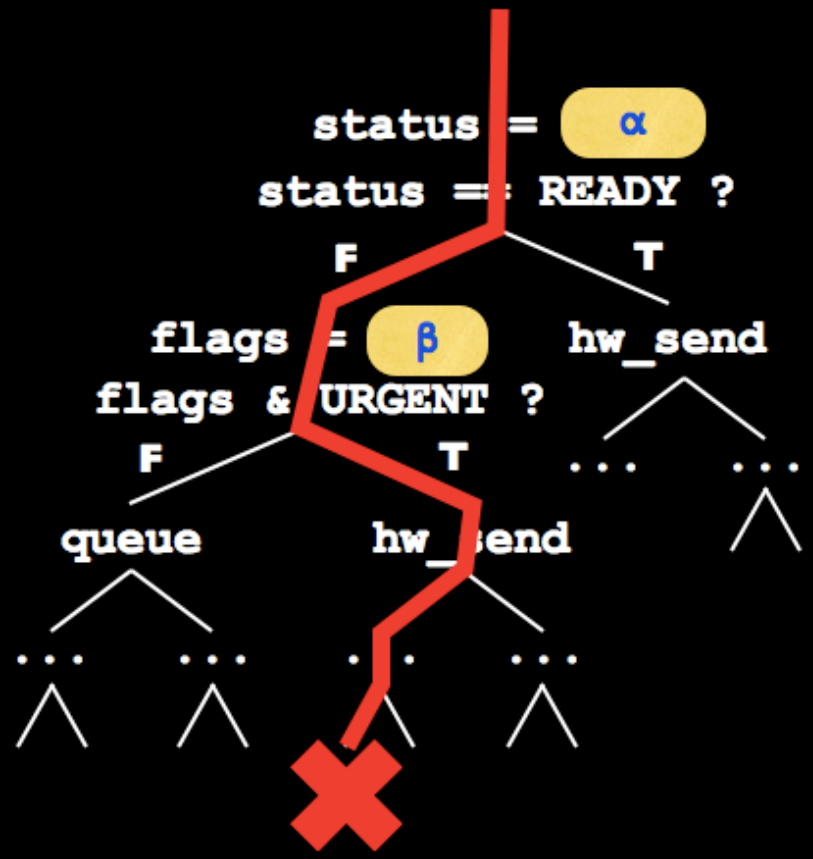
- Driver Exerciser
- **Bug Checkers**
- Bug Reports
- Evaluation

Outline

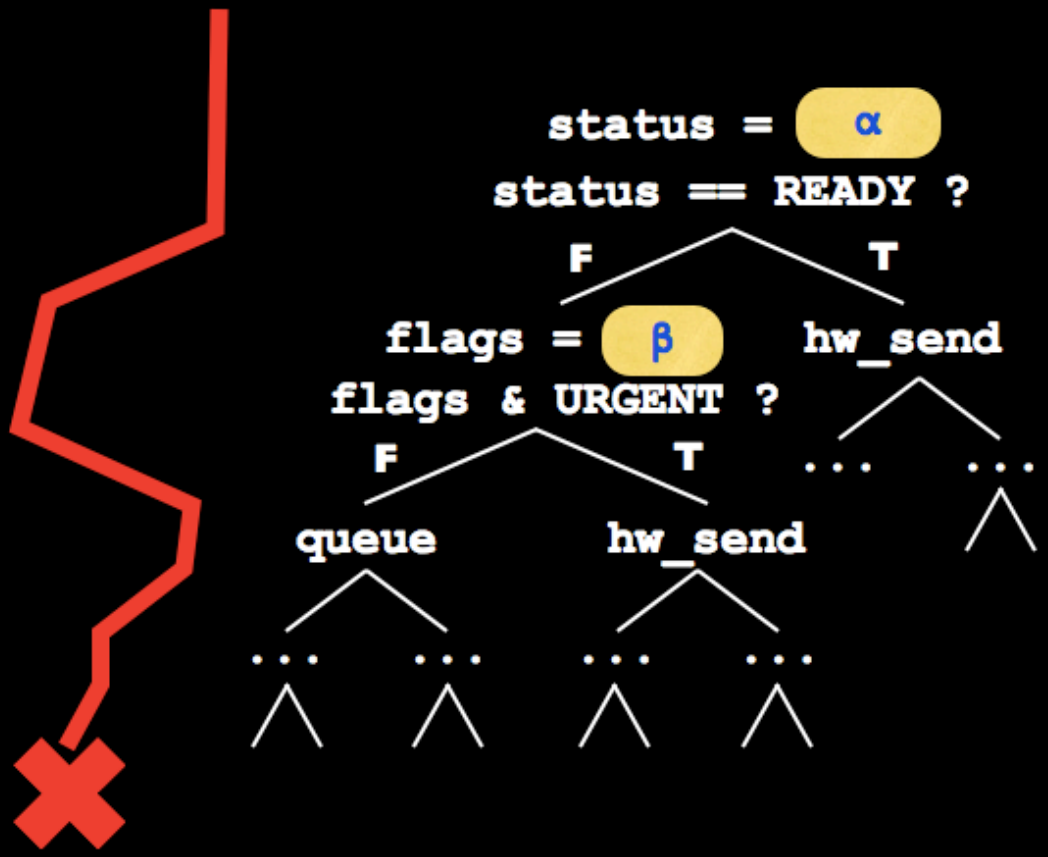


- Driver Exerciser
- Bug Checkers
- **Bug Reports**
- Evaluation

Bug Reports

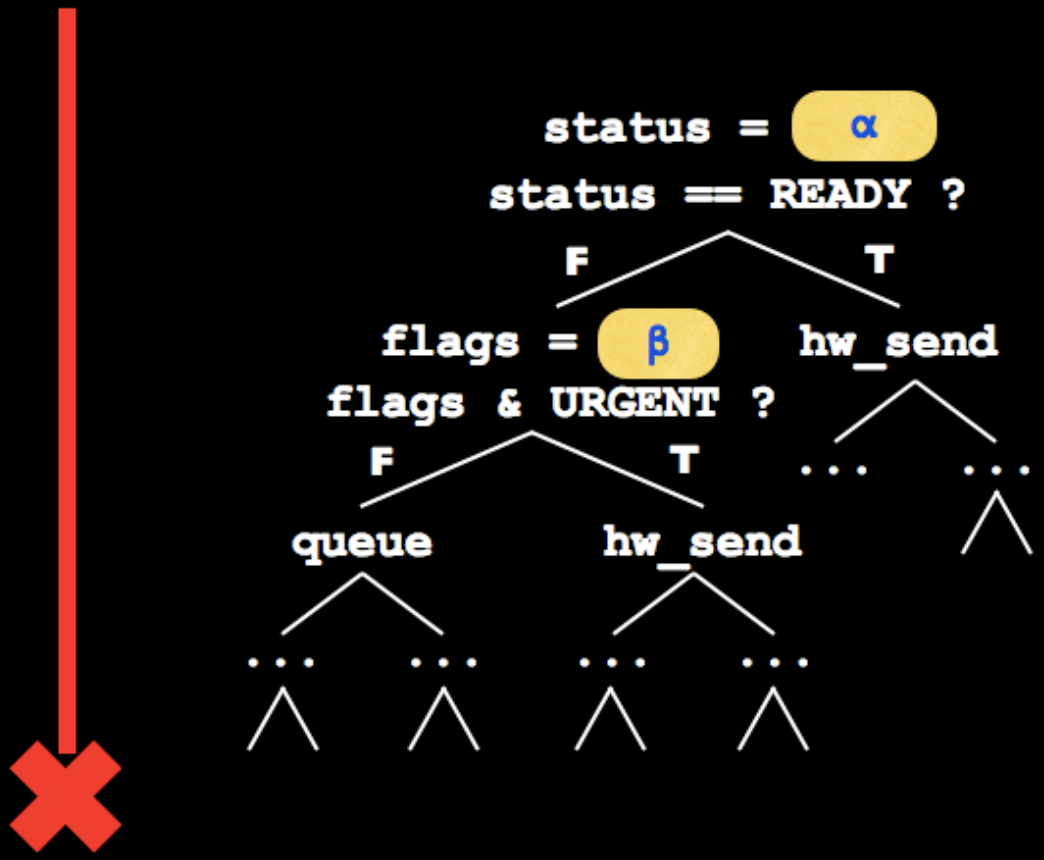

status = α
status == READY ?
F
T
flags = β
hw_send
flags & URGENT ?
F
T
queue
hw_send

Bug Reports


status = α
status == READY ?
F
T
flags = β
hw_send
flags & URGENT ?
F
T
queue
hw_send

Bug Reports

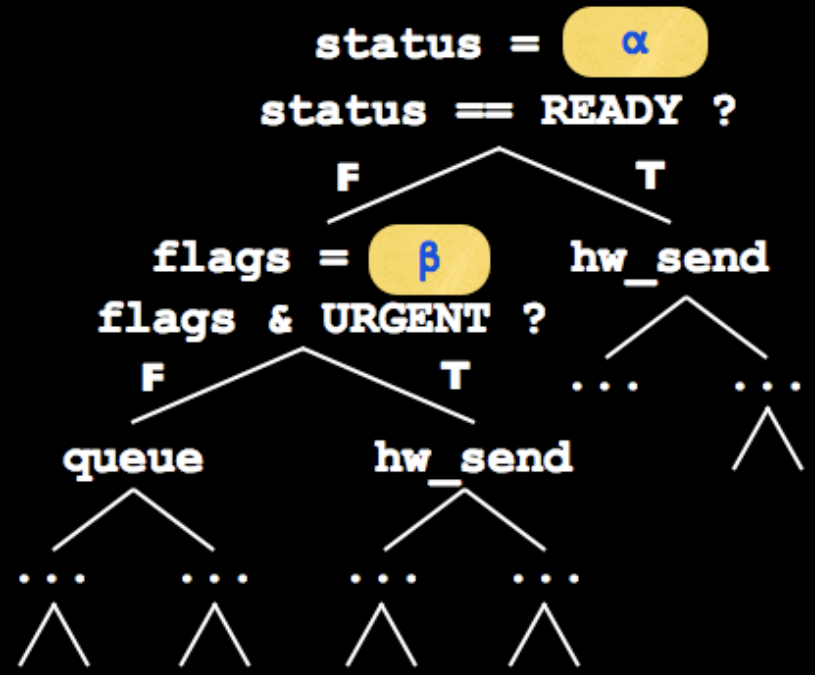
Bug: NULL pointer dereference



Bug Reports

Bug: NULL pointer dereference

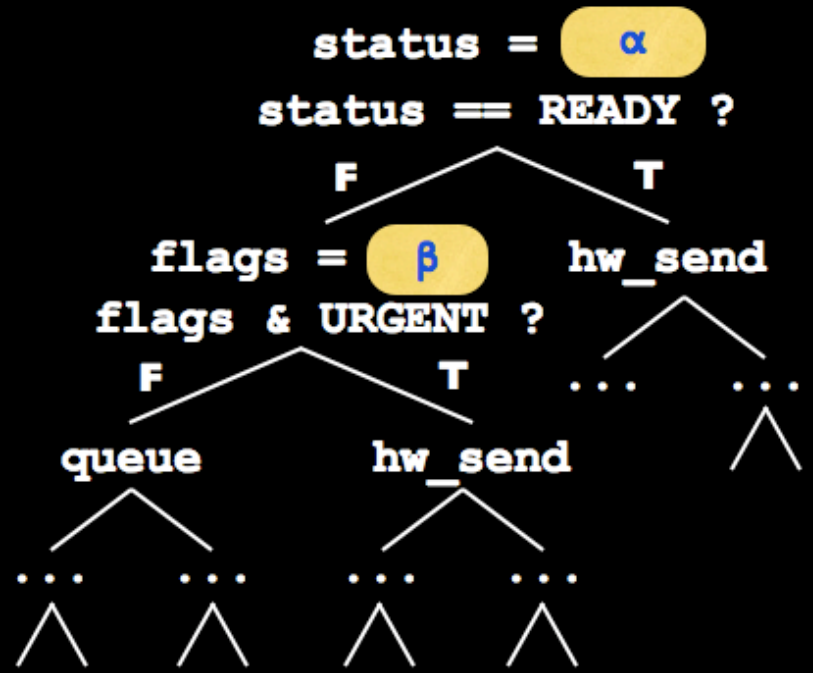
```
...
lea    edx, [ebp+var_8]
push   edx
call   ds:NdisInitializeW...
cmp    [ebp+var_8], 0
...
```



Bug Reports

Bug: NULL pointer dereference

```
...
lea    edx, [ebp+var_8]
push   edx
call   ds:NdisInitializeW...
cmp    [ebp+var_8], 0
...


...
status = 0x01
flags = 0x20
...
```



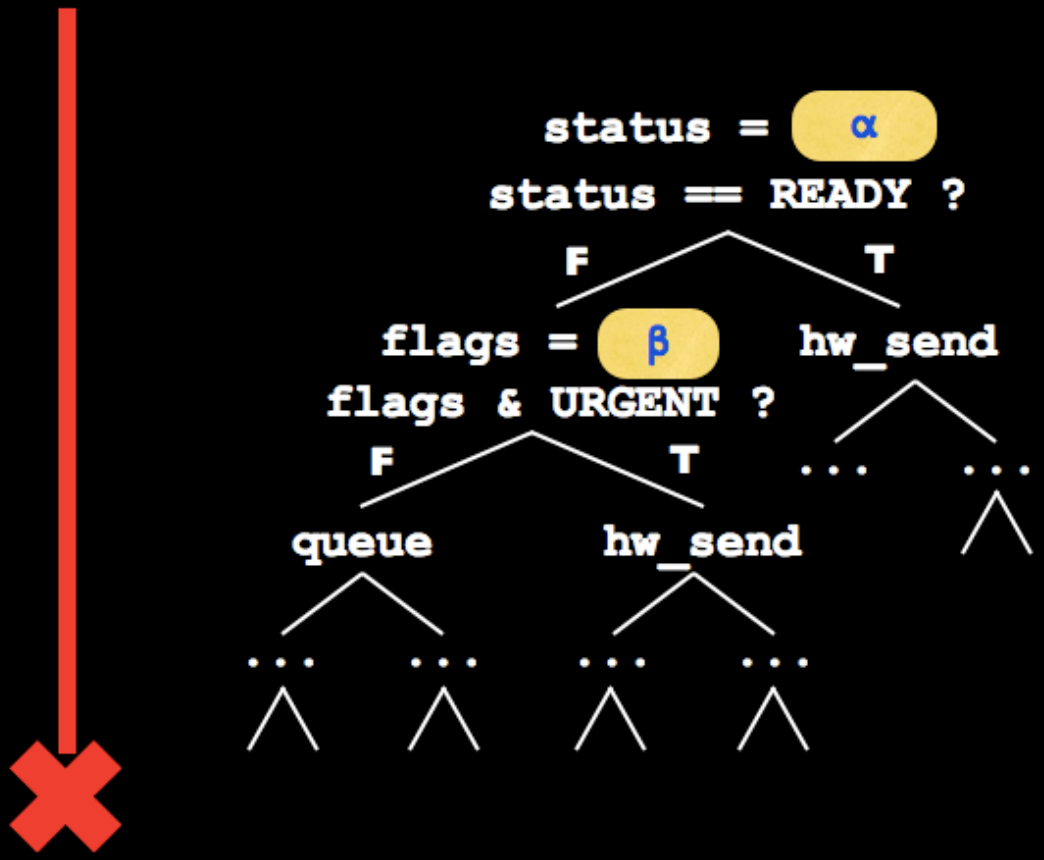
Bug Reports

Bug: NULL pointer dereference

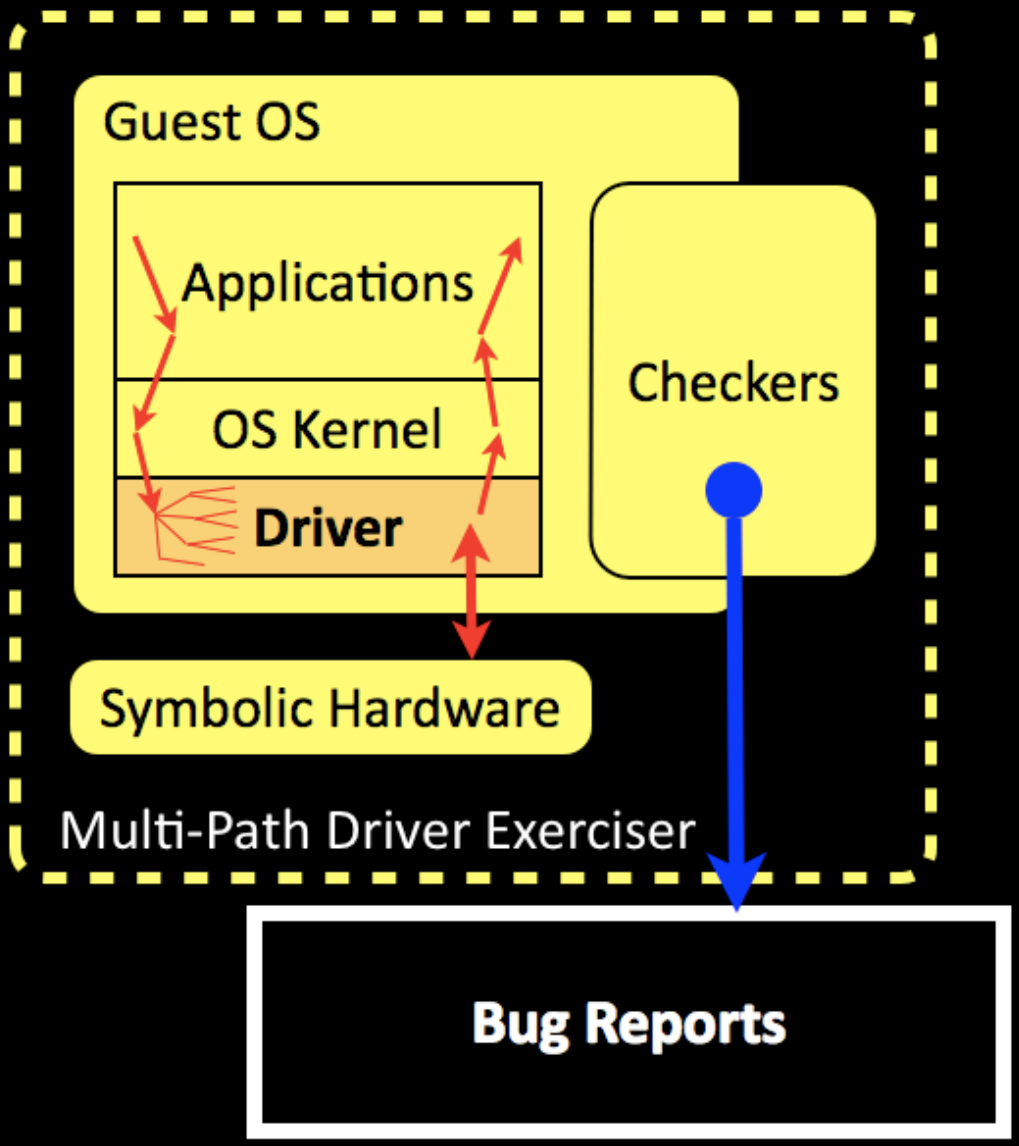
```
...
lea    edx, [ebp+var_8]
push   edx
call   ds:NdisInitializeW...
cmp    [ebp+var_8], 0
...
```

```
...
status = 0x01
flags = 0x20
...
```

Data flow information

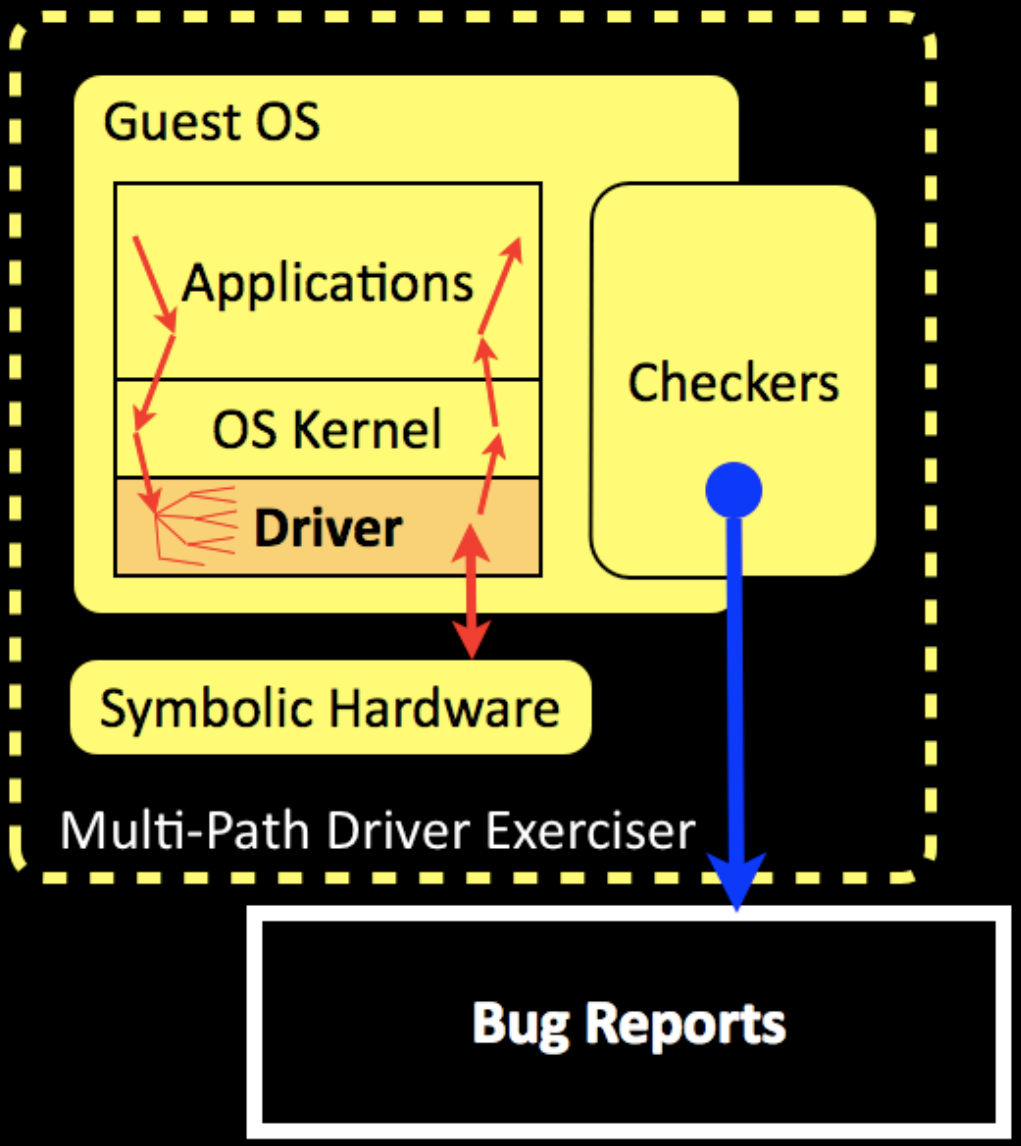


Outline



- **Driver Exerciser**
- **Bug Checkers**
- **Bug Reports**
- Evaluation

Outline



- Driver Exerciser
- Bug Checkers
- Bug Reports
- Evaluation

Implementation

- QEMU virtual machine[1]
- x86-to-LLVM translator
- KLEE symbolic execution engine[2]

[1] F. Bellard. *QEMU, a Fast and Portable Dynamic Translator.* In USENIX 2005.
[2] C. Cadar et al. *KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs.* In OSDI 2008.

Evaluation

- Effectiveness
  Found 14 bugs in 6 Microsoft-certified closed-source drivers
- Bug Example
  Demo

Tested Drivers

Driver	Binary Size	New Bugs Found
Intel Pro/1000	168 KB	Memory leak
Intel Pro/100	70 KB	Kernel crash
Realtek RTL8029	18 KB	Memory corruption, race condition, resource leak, multiple segmentation
Ensoniq AudioPCI	37 KB	Multiple segmentation faults, multiple resource leaks
AMD PCNet	35 KB	Multiple resource leaks
Intel 828001AA AC97	39 KB	Race condition

Microsoft-certified drivers

Tested Drivers

Driver	Binary Size	New Bugs Found
Intel Pro/1000	168 KB	Memory leak
Intel Pro/100	70 KB	Kernel crash
Realtek RTL8029	18 KB	Memory corruption, race condition, resource leak, multiple segmentation
Ensoniq AudioPCI	37 KB	Multiple segmentation faults, multiple resource leaks
AMD PCNet	35 KB	Multiple resource leaks
Intel 828001AA AC97	39 KB	Race condition

Microsoft-certified drivers

Tested Drivers

Driver	Binary Size	New Bugs Found
Intel Pro/1000	168 KB	Memory leak
Intel Pro/100	70 KB	Kernel crash
Realtek RTL8029	18 KB	Memory corruption, race condition, resource leak, multiple segmentation
Ensoniq AudioPCI	37 KB	Multiple segmentation faults, multiple resource leaks
AMD PCNet	35 KB	Multiple resource leaks
Intel 828001AA AC97	39 KB	Race condition

Microsoft-certified drivers

Tested Drivers

Driver	Binary Size	New Bugs Found
Intel Pro/1000	168 KB	Memory leak
Intel Pro/100	70 KB	Kernel crash
Realtek RTL8029	18 KB	Memory corruption, race condition, resource leak, multiple segmentation
Ensoniq AudioPCI	37 KB	Multiple segmentation faults, multiple resource leaks
AMD PCNet	35 KB	Multiple resource leaks
Intel 828001AA AC97	39 KB	Race condition

Microsoft-certified drivers

Impact of ...

Component	# bugs found
MS Driver Verifier + DDT (as an OS-Level checker)	3 bugs
VM-Level checkers ***without*** API annotations	4 bugs
VM-Level checkers ***with*** API Annotations	7 bugs
Total	14 bugs

Impact of ...

Component	# bugs found
MS Driver Verifier + DDT (as an OS-Level checker)	3 bugs
VM-Level checkers ***without*** API annotations	4 bugs
VM-Level checkers ***with*** API Annotations	7 bugs
Total	14 bugs

Impact of ...

Component	# bugs found
MS Driver Verifier + DDT (as an OS-Level checker)	3 bugs
VM-Level checkers ***without*** API annotations	4 bugs
VM-Level checkers ***with*** API Annotations	7 bugs
Total	14 bugs

Impact of ...

Component	# bugs found
MS Driver Verifier + DDT (as an OS-Level checker)	3 bugs
VM-Level checkers ***without*** API annotations	4 bugs
VM-Level checkers ***with*** API Annotations	7 bugs
Total	14 bugs

Impact of ...

Component	# bugs found
MS Driver Verifier + DDT (as an OS-Level checker)	3 bugs
VM-Level checkers ***without*** API annotations	4 bugs
VM-Level checkers ***with*** API Annotations	7 bugs
Total	14 bugs

Crash on Invalid Parameters (RTL8029)

```
NdisReadConfiguration(..., "MaximumMulticastList", &size, ...)
if (size > 32)
  goto HandleError
...

HandleError:
...
driver->multicastListSize = size;
...

FillMulticastRegs:
...
for(i=0; i < driver->multicastListSize; ++i) {
  driver->arr[i] = ...;
}
```

Crash on Invalid Parameters (RTL8029)

```
NdisReadConfiguration(..., "MaximumMulticastList", &size, ...)
if (size > 32)
  goto HandleError
...

HandleError:
  ...
  driver->multicastListSize = size;
  ...

FillMulticastRegs:
  ...
  for(i=0; i < driver->multicastListSize; ++i) {
    driver->arr[i] = ...;
  }
```

Crash on Invalid Parameters (RTL8029)

```
  NdisReadConfiguration(..., "MaximumMulticastList", &size, ...)
  if (size > 32)
    goto HandleError
  ...

HandleError:
  ...
  driver->multicastListSize = size;
  ...

FillMulticastRegs:
  ...
  for(i=0; i < driver->multicastListSize; ++i) {
    driver->arr[i] = ...;
  }
```

Crash on Invalid Parameters (RTL8029)

```
NdisReadConfiguration(..., "MaximumMulticastList", &size, ...)
if (size > 32)
  goto HandleError
...

HandleError:
...
driver->multicastListSize = size;
...

FillMulticastRegs:
...
for(i=0; i < driver->multicastListSize; ++i) {
  driver->arr[i] = ...;
}
```

Crash on Invalid Parameters (RTL8029)

```
NdisReadConfiguration(..., "MaximumMulticastList", &size, ...)
if (size > 32)
  goto HandleError
...

HandleError:
...
driver->multicastListSize = size;
...

FillMulticastRegs:
...
for(i=0; i < driver->multicastListSize; ++i) {
  driver->arr[i] = ...;
}
```

✖overflow

Crash on Invalid Parameters (RTL8029)

```
NdisReadConfiguration(..., "MaximumMulticastList", &size, ...)
if (size > 32)
  goto HandleError
...

HandleError:
...
driver->multicastListSize = size;
...

FillMulticastRegs:
...
for(i=0; i < driver->multicastListSize; ++i) {
  driver->arr[i] = ...;
}
```

Bug Report
System crash
Key: \\HKLM\Services\...\...\0003
Value: MaximumMulticastList
size=999999

✖overflow

Crash on Invalid Parameters (RTL8029)

```
NdisReadConfiguration(..., "MaximumMulticastList", &size, ...)
if (size > 32)
  goto HandleError
...

HandleError:
  ...
  driver->multicastListSize = size;
  ...

FillMulticastRegs:
  ...
  for(i=0; i < driver->multicastListSize; ++i) {
    driver->arr[i] = ...;
  }
```

Bug Report
System crash
Key: \\HKLM\Services\...\...\0003
Value: MaximumMulticastList
size=999999

✖overflow

DDT

- Consistent mixing of symbolic and concrete execution
  Achieves high coverage testing of drivers
  Works on closed-source drivers
- Symbolic hardware
  Testing does not require physical hardware
  Accounts for buggy hardware