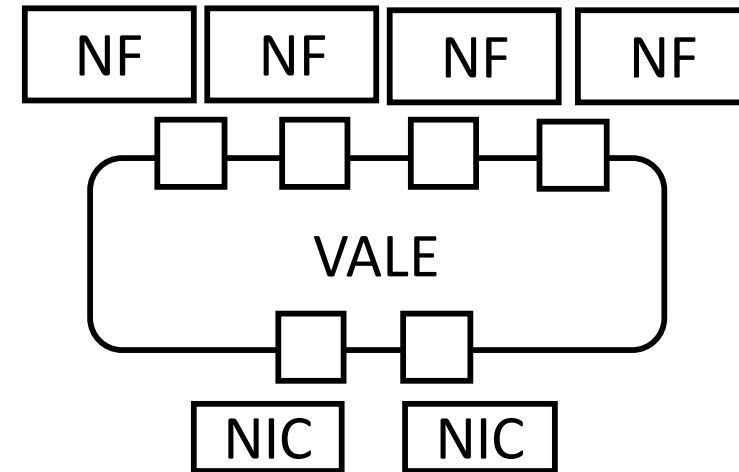


VALE (mSwitch)

Overview

- What?
 - In-kernel virtual switch
 - Like Linux bridge
 - Interconnect NICs and virtual interfaces
- Why?
 - Fast (10 Mpps > w/ a single core)
 - Scalable (100s – 1000s of ports)
 - Flexible (arbitrary packet processing logic)

Perfect for NFV

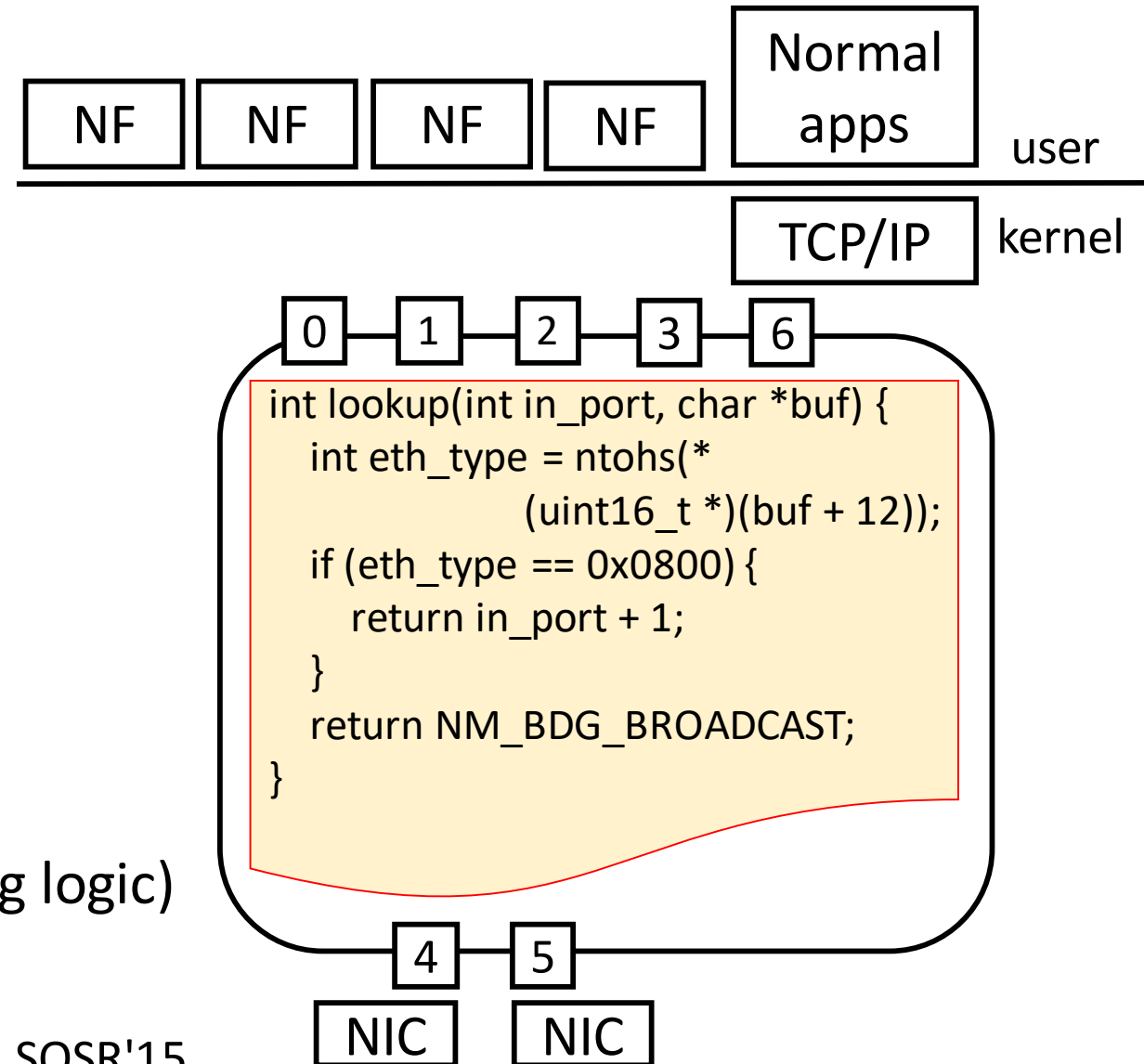


Overview

- What?
 - In-kernel virtual switch
 - Like Linux bridge
 - Interconnect NICs and virtual interfaces
- Why?
 - Fast (10 Mpps > w/ a single core)
 - Scalable (100s – 1000s of ports)
 - Flexible (arbitrary packet processing logic)

Perfect for NFV

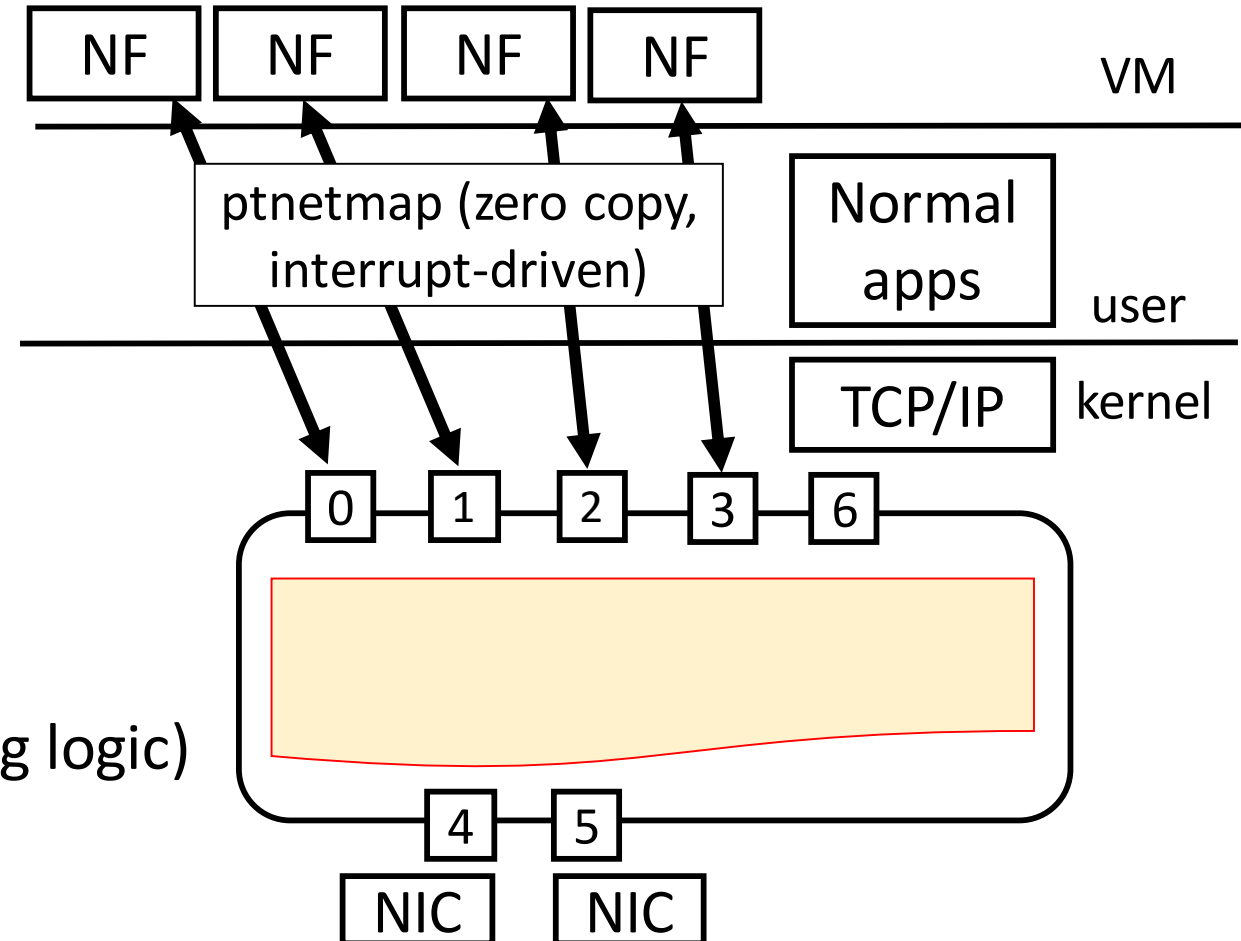
mSwitch: A Highly-Scalable, Modular Software Switch, SOSR'15



Overview

- What?
 - In-kernel virtual switch
 - Like Linux bridge
 - Interconnect NICs and virtual interfaces
- Why?
 - Fast (10 Mpps > w/ a single core)
 - Scalable (100s – 1000s of ports)
 - Flexible (arbitrary packet processing logic)

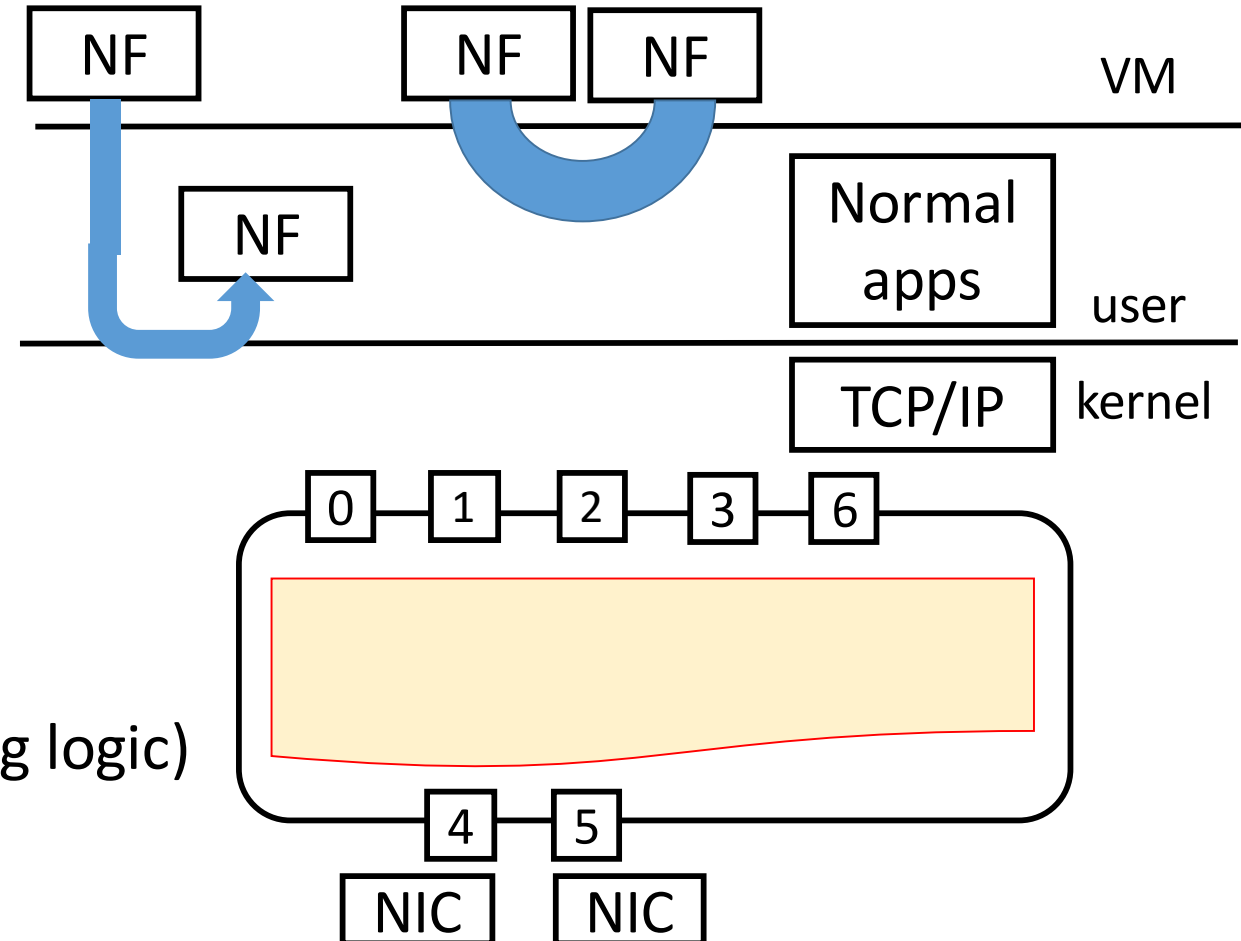
Perfect for NFV



Overview

- What?
 - In-kernel virtual switch
 - Like Linux bridge
 - Interconnect NICs and virtual interfaces
- Why?
 - Fast (10 Mpps > w/ a single core)
 - Scalable (100s – 1000s of ports)
 - Flexible (arbitrary packet processing logic)

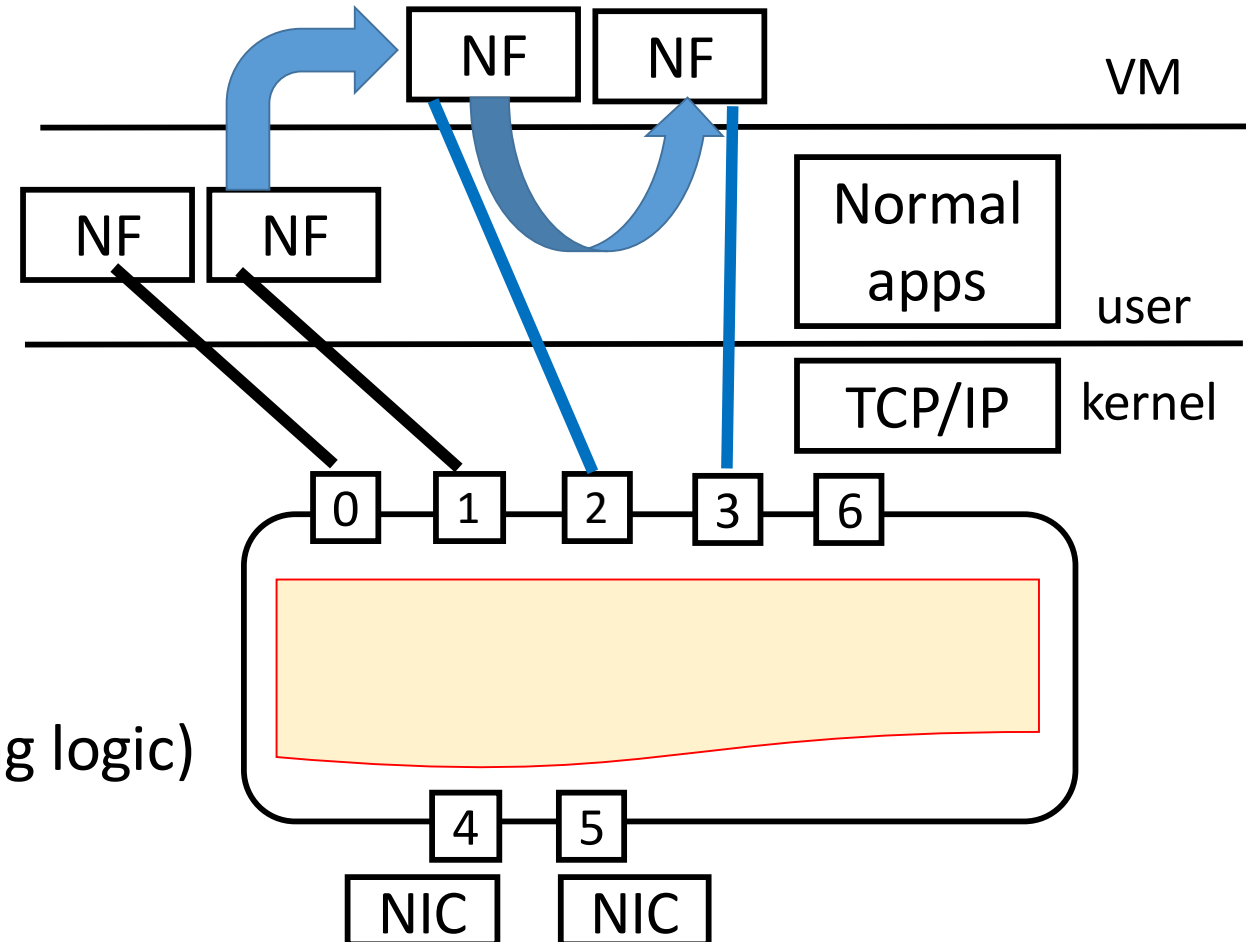
Perfect for NFV



Overview

- What?
 - In-kernel virtual switch
 - Like Linux bridge
 - Interconnect NICs and virtual interfaces
- Why?
 - Fast (10 Mpps > w/ a single core)
 - Scalable (100s – 1000s of ports)
 - Flexible (arbitrary packet processing logic)

Perfect for NFV

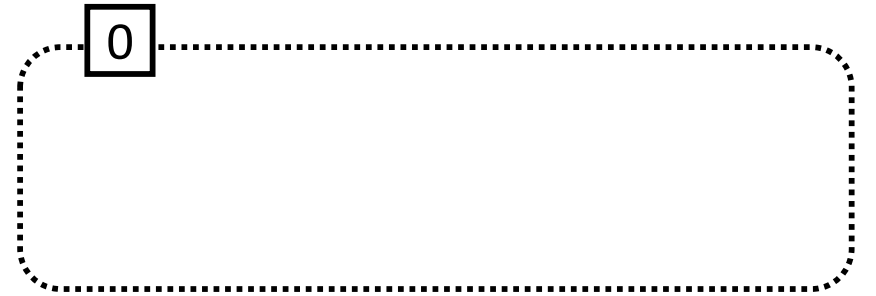


Download and compile

```
% git clone git@github.com:micchie/mymodule.git  
% cd mymodule/LINUX  
% make KSRC=/lib/modules/`uname -r`/build  
NSRC=WHERE_YOUR_NETMAP_IS
```

How to

% vale-ctl -n vi0



How to

```
% vale-ctl -n vi0
```

```
% vale-ctl -a vale0:vi0
```



0

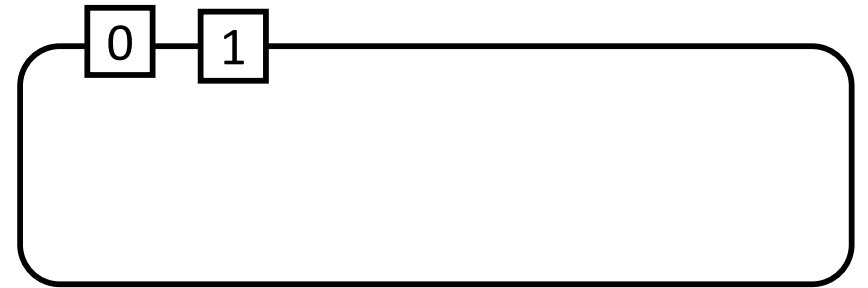
How to

```
% vale-ctl -n vi0
```

```
% vale-ctl -a vale0:vi0
```

```
% vale-ctl -n vi1
```

```
% vale-ctl -a vale0:vi1
```

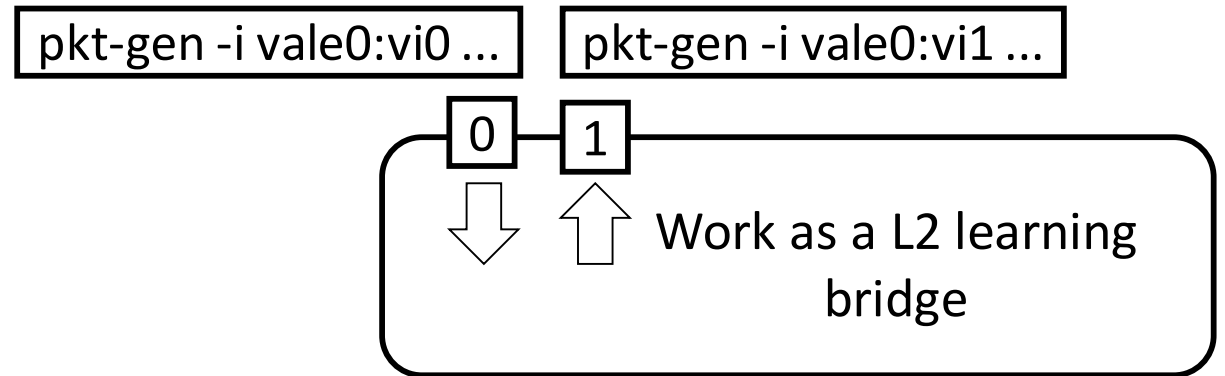


How to

```
% pkt-gen -i vale0:vi0 -f tx  
(another terminal)  
% pkt-gen -i vale0:vi1 -f rx
```

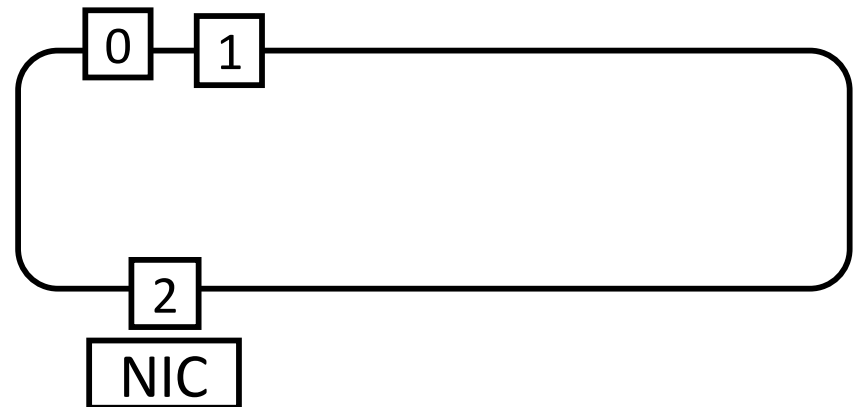
Cleanup:

```
% vale-ctl -d vale0:vi0  
% vale-ctl -r vi0  
% vale-ctl -d vale0:vi1  
% vale-ctl -r vi1
```



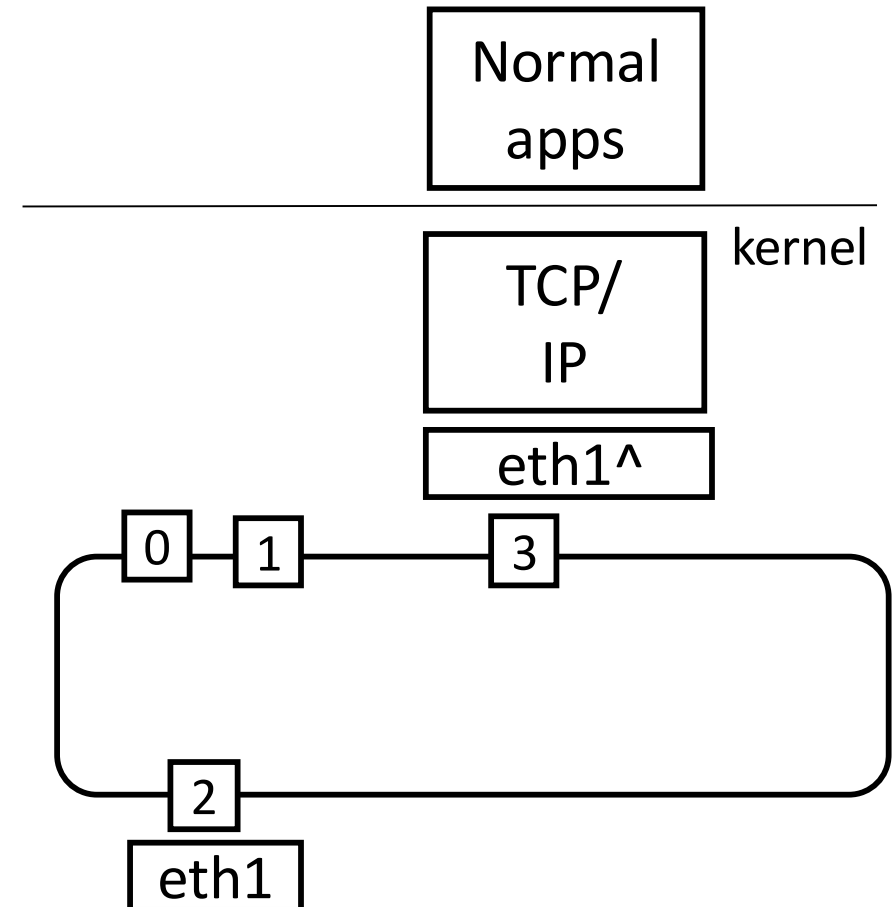
Playing with a NIC

```
% vale-ctl -a vale0:eth1  
(-d to detach)
```



Playing with a NIC

% vale-ctl **-h** vale0:eth1
(-d to detach)



Loading a custom module

```
% git clone git@github.com:micchie/mymodule.git
```

```
% cd mymodule/LINUX
```

```
% make KSRC=WHERE_YOUR_KERNEL_IS  
NSRC=WHERE_YOUR_NETMAP_IS
```

```
% vale-ctl -n vi0
```

```
% vale-ctl -a vale0:vi0
```

```
% sudo insmod mymodule_lin.ko
```

0

```
int lookup(int in_port, char *buf) {  
    int eth_type = ntohs(*  
                          (uint16_t *)(buf + 12));  
    if (eth_type == 0x0800) {  
        return in_port + 1;  
    }  
    return NM_BDG_BROADCAST;  
}
```

Loading a custom module

```
% git clone git@github.com:micchie/mymodule.git
```

```
% cd mymodule/LINUX
```

```
% make KSRC=WHERE_YOUR_KERNEL_IS  
NSRC=WHERE_YOUR_NETMAP_IS
```

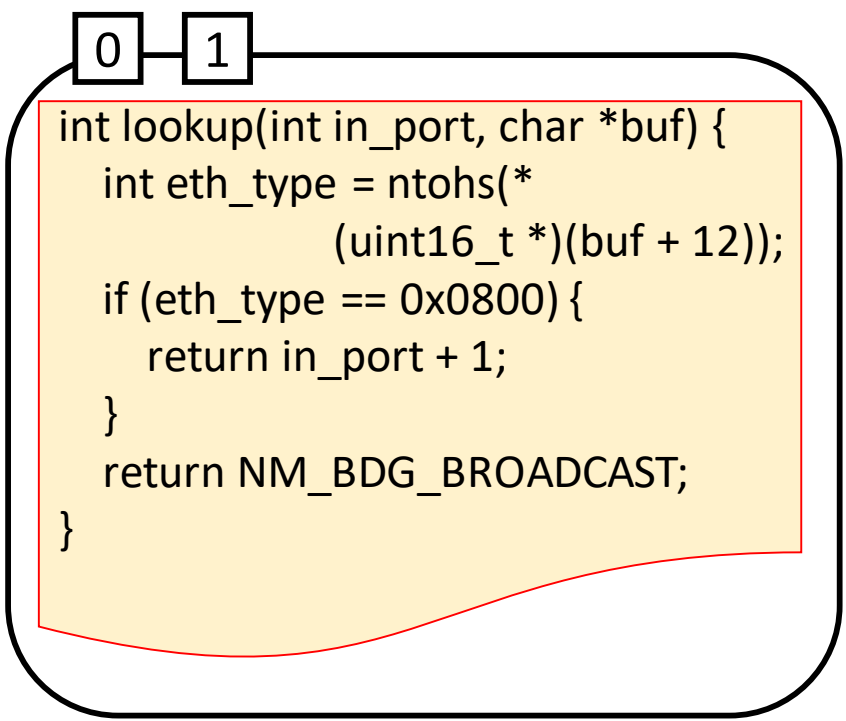
```
% vale-ctl -n vi0
```

```
% vale-ctl -a vale0:vi0
```

```
% sudo insmod mymodule_lin.ko
```

```
% vale-ctl -n vi1
```

```
% vale-ctl -a vale0:vi1
```



```
0 1  
int lookup(int in_port, char *buf) {  
    int eth_type = ntohs(*  
                          (uint16_t *) (buf + 12));  
    if (eth_type == 0x0800) {  
        return in_port + 1;  
    }  
    return NM_BDG_BROADCAST;  
}
```

Loading a custom module

```
% pkt-gen -i vale0:vi0 -f tx  
(another terminal)  
% pkt-gen -i vale0:vi1 -f rx  
% vale-ctl -n vi2  
% vale-ctl -a vale0:vi2  
(another terminal)  
% pkt-gen -i vale0:vi2 -f rx  
(you don't see packets)
```

0

1

```
int lookup(int in_port, char *buf) {  
    int eth_type = ntohs(*  
                          (uint16_t *)(buf + 12));  
    if (eth_type == 0x0800) {  
        return in_port + 1;  
    }  
    return NM_BDG_BROADCAST;  
}
```


Loading a custom module

% vale-ctl -d vale0:vi1

0

1

```
int lookup(int in_port, char *buf) {  
    int eth_type = ntohs(*  
                          (uint16_t *)(buf + 12));  
    if (eth_type == 0x0800) {  
        return in_port + 1;  
    }  
    return NM_BDG_BROADCAST;  
}
```

Loading a custom module

```
% vale-ctl -d vale0:vi1  
% vale-ctl -r vi1
```

0

```
int lookup(int in_port, char *buf) {  
    int eth_type = ntohs(*  
                          (uint16_t *)(buf + 12));  
    if (eth_type == 0x0800) {  
        return in_port + 1;  
    }  
    return NM_BDG_BROADCAST;  
}
```

Loading a custom module

```
% vale-ctl -d vale0:vi1
```

```
% vale-ctl -r vi1
```

```
% vale-ctl -d vale0:vi0
```

0

```
int lookup(int in_port, char *buf) {  
    int eth_type = ntohs(*  
                          (uint16_t *)(buf + 12));  
    if (eth_type == 0x0800) {  
        return in_port + 1;  
    }  
    return NM_BDG_BROADCAST;  
}
```

Loading a custom module

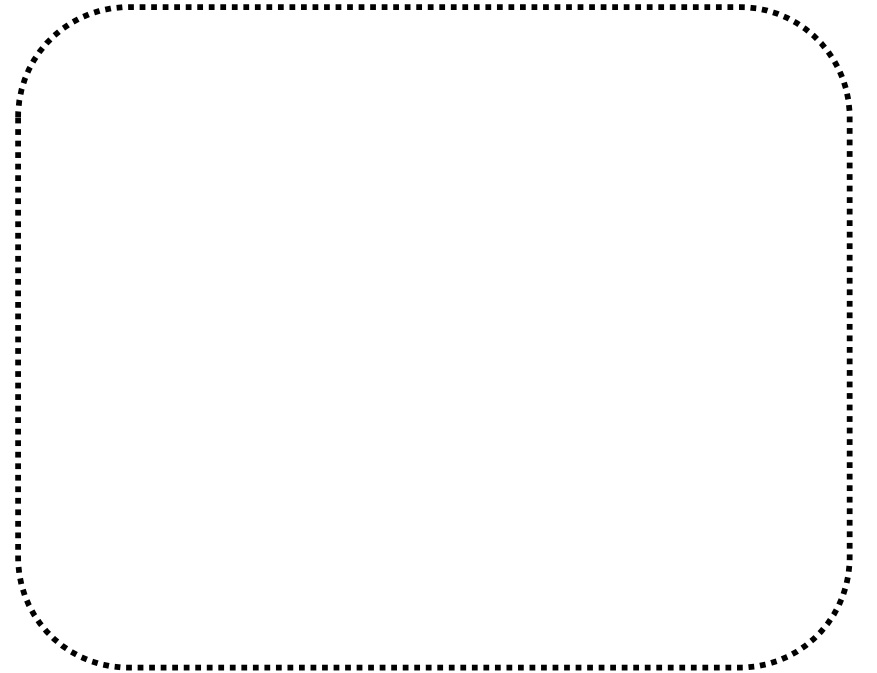
```
% vale-ctl -d vale0:vi1
```

```
% vale-ctl -r vi1
```

```
% vale-ctl -d vale0:vi0
```

(module is automatically unloaded)

```
% vale-ctl -r vi0
```



Writing a module

- Open `sys/contrib/mymodule/mymodule.c`
- Find a function `my_lookup()`
 - This function extracts the ethernet type (at 12 byte offset)
 - It returns the source switch port + 1 for IPv4 packets
 - It indicates broadcast for the other packets

Writing a module

- Open `sys/contrib/mymodule/mymodule.c`
- Find a function `my_lookup()`
 - This function extracts the ethernet type (at 12 byte offset)
 - It returns the source switch port + 1 for IPv4 packets
 - It indicates broadcast for the other packets

Writing a module

- Let's change this module to forward packets based on the source port and user-specified destination port
- Remove code under `#endif` in `my_lookup()`
- Activate `#if 0 -- #endif`
- See `my_config()`, another callback which is plugged in to the VALE switch
 - `struct mmreq` is in `sys/net/mymodule.h`
 - User gives `mreq->mr_sport` and `mreq->mr_dport`

References

- *Netmap: a novel framework for fast packet I/O* (USENIX ATC'11)
 - NIC I/O and Basic API
- *Vale, a switched ethernet for virtual machines* (ACM CoNext'12)
 - Learning bridge between VMs
- *mSwitch: A Highly-Scalable, Modular Software Switch* (ACM SOSR'15)
 - Many ports, modular switching logic
 - Papers using it:
 - Rekindling network protocol innovation with user-level stacks (ACM CCR April 2014)
 - Prism: A Proxy Architecture for Datacenter Networks (ACM SoCC'17)
- *Flexible Virtual Machine Networking Using Netmap Passthrough* (IEEE LANMAN'16)
 - Pipes, ptnetmap
- *A Study of Speed Mismatches Between Communicating Virtual Machines* (ACM ANCS'16)
 - Modeling producer (sender) - consumer (receiver) speeds and resulting performance
- *PASTE: Network Stacks Must Integrate with NVMM Abstractions* (ACM HotNets'16)
 - netmap with persistent memory (ongoing)