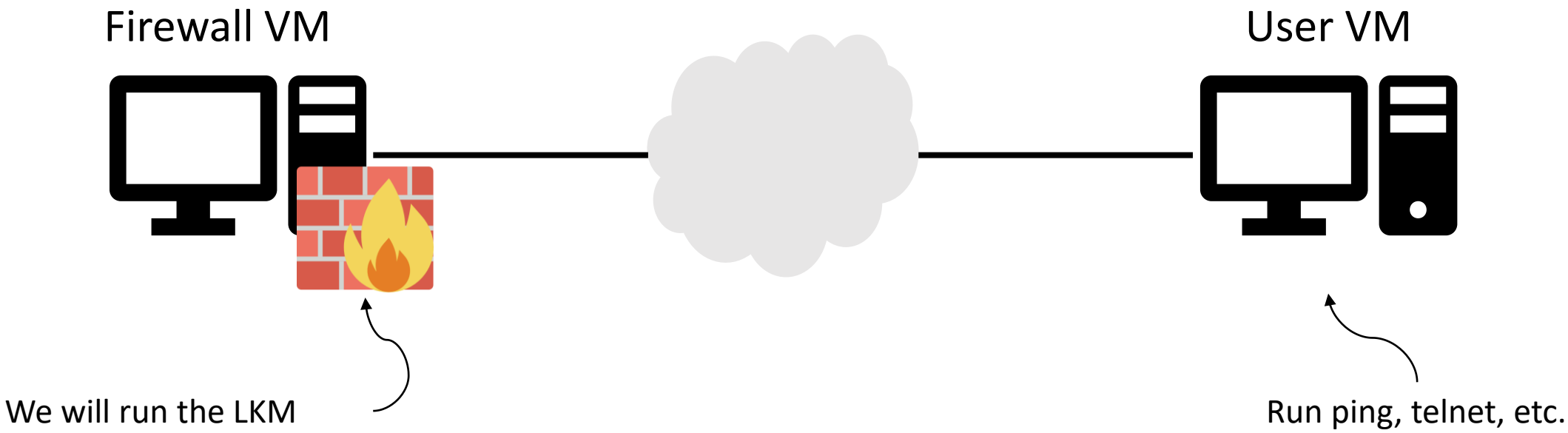
Lab 11

Three Tasks

- Implement a loadable kernel module (to perform packet filtering)
- Implement:
  - stateless firewall
  - stateful firewall

Task 1 Setup



Task 1

- Run provided code:
  - block outgoing telnet traffic
- Implement two new hook functions to block:
  - the firewall VM from pinging 8.8.8.8
  - other machines from pinging the firewall VM

Task 1: Writing the Logic

```
unsigned int yourFilter(void *priv, struct sk_buff *skb,
                        const struct nf_hook_state *state)
{
        // Some Parsing
        iph = ip_hdr(skb);

        if (some_condition) {
                return NF_DROP;
        }
        return NF_ACCEPT;
}
```

Task 1: Registering the hook

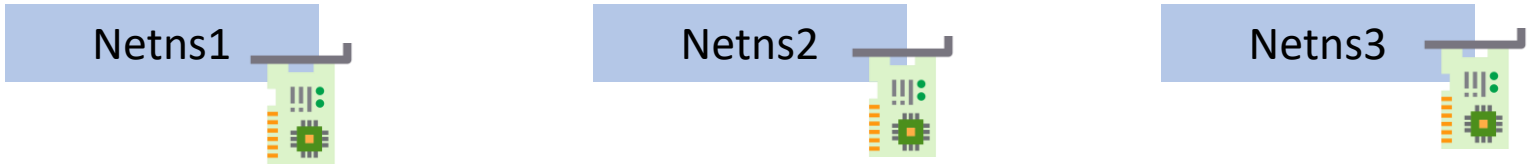
```
static struct nf_hook_ops yourFilterHook;

int addFilters(void) {
        yourFilterHook.hook = yourFilter;
        yourFilterHook.hooknum = <HOOK_NUM>;
        yourFilterHook.pf = PF_INET;
        yourFilterHook.priority = NF_IP_PRI_FIRST;

        // Register the hook
        nf_register_net_hook(&init_net, &yourFilterHook);
        return 0;
}
```

NF_INET_PRE_ROUTING
NF_INET_LOCAL_IN
NF_INET_FORWARD
NF_INET_LOCAL_OUT
NF_INET_POST_ROUTING

What is init_net?

Network Namespace (netns)

- A netns is a copy of the network stack:
  - With its own routes, firewall rules, etc.
- Defining routing tables and firewall rules *per* netns



Network Namespace (netns)

- init_net is the initial netns in Linux

```
149   /* Init's network namespace */
150   extern struct net init_net;
```

- `nf_register_net_hook(&init_net, &yourFilterHook):`
  - We register the hook to the initial netns
  - The rules will be applied to all traffic in this netns

Task 1: Registering the hook

```
static struct nf_hook_ops yourFilterHook;

int addFilters(void) {
      yourFilterHook.hook = yourFilter;
      yourFilterHook.hooknum = <HOOK_NUM>;
      yourFilterHook.pf = PF_INET;
      yourFilterHook.priority = NF_IP_PRI_FIRST;

      // Register the hook
      nf_register_net_hook(&init_net, &yourFilterHook);
      return 0;
}
```

NF_INET_PRE_ROUTING
NF_INET_LOCAL_IN
NF_INET_FORWARD
NF_INET_LOCAL_OUT
NF_INET_POST_ROUTING

```
void rmFilters(void) {
      nf_unregister_net_hook(&init_net, &yourFilterHook);
}
```

Task 1: Initialize the module

```
module_init(addFilters);
module_exit(rmFilters);
```

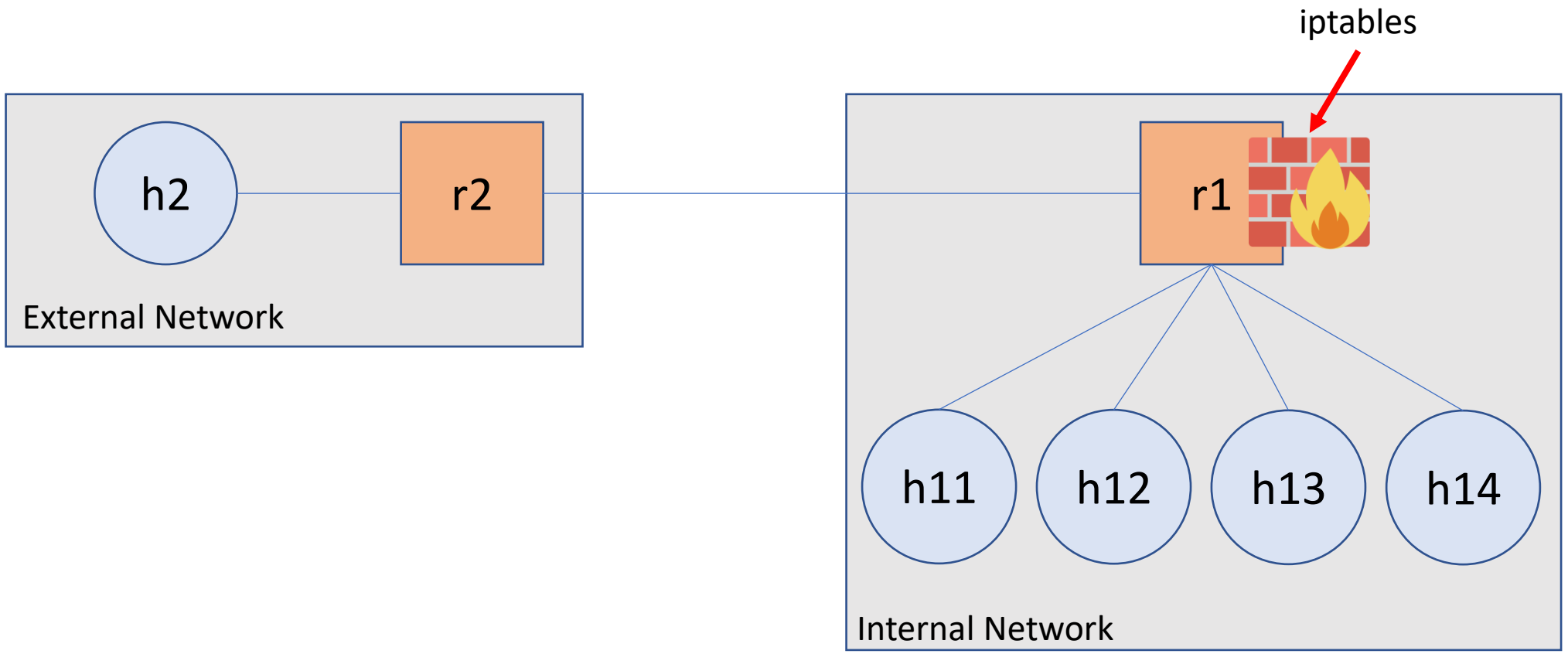
Task 2 and Task 3

- Writing stateless and stateful firewall rules:
  - Protect an internal network
  - Protect internal services
  - Ensure that initiated connections from internal network are not blocked

- We will use IPMininet

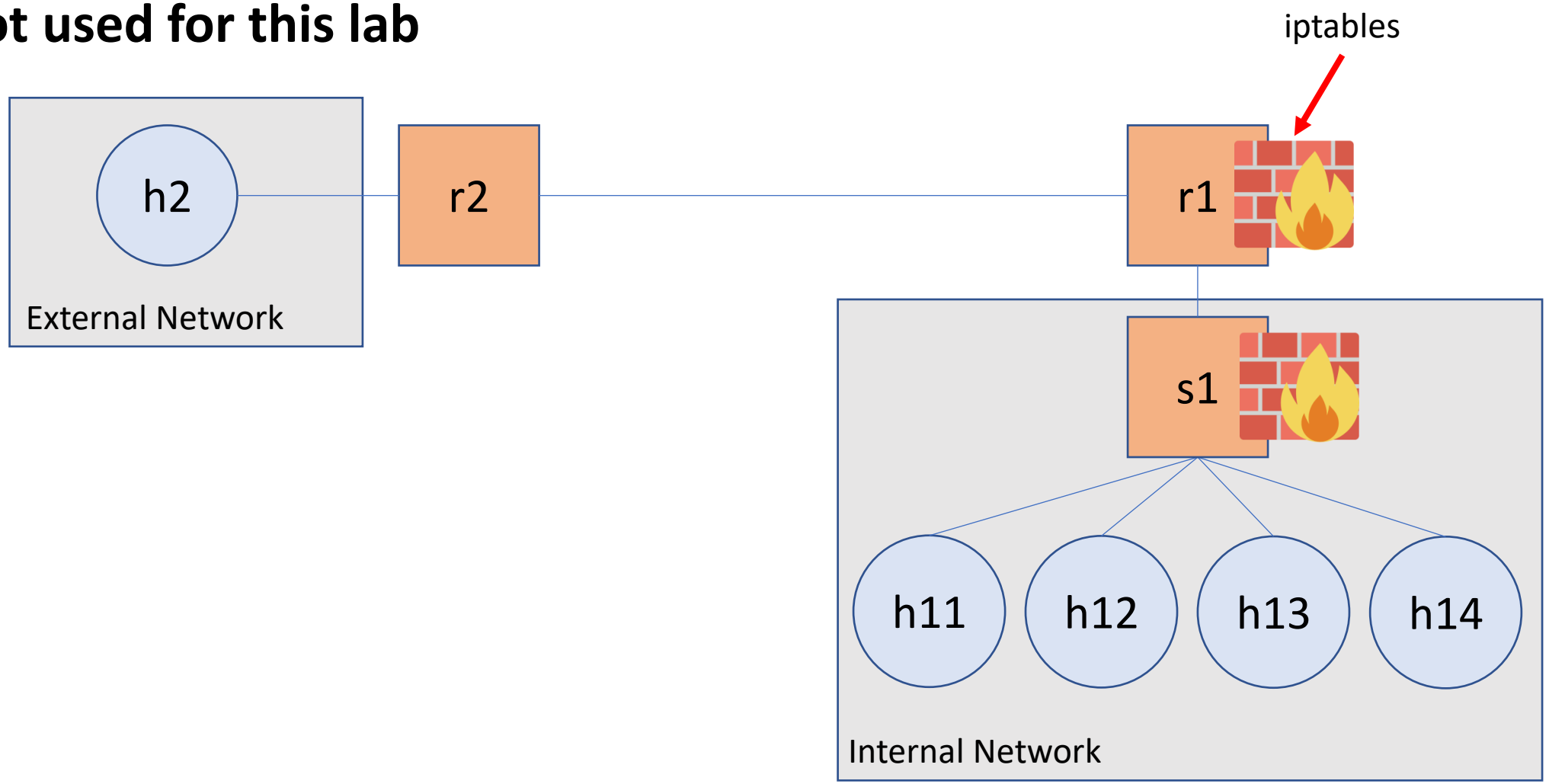
- iptables: https://linux.die.net/man/8/iptables

Task 2 and Task 3: IPMininet Setup



Switch-based Approach

Not used for this lab



iptables: Commands

Goal	Command
Specify a table	`-t <table>`
Append a rule to a chain	`-A <chain> <rule_spec>`
Delete a rule from a chain	`-D <chain> <rule_spec>`
Show packet count	`-L <chain> -v`
List rules	`-L [chain] [-t table]`
Flush rules	`-F [chain] [-t table]`
Insert a rule to a chain	`-I <chain> [rule-number]`

iptables: Parameters

Goal	Parameter
Protocol	`-p <protocol>`
Jump to target	`-j <target>`
Input interface	`-i <intf>`
Output interface	`-o <intf>`
Source	`-s <src>`
Destination	`-d <dst>`
Extended match module	`-m <module>`

iptables: Connection Tracking

- Make sure that conntrack-tools is installed at your Vagrant VM
  - (Instructions in the document)

- `--cstate: NEW, RELATED, ESTABLISHED`

Installing iptables Rules in IPMininet

```
r1_rules = [Rule('-A OUTPUT -j DROP')]
r1.addDaemon(IPTables, rules=r1_rules)
```

Questions?