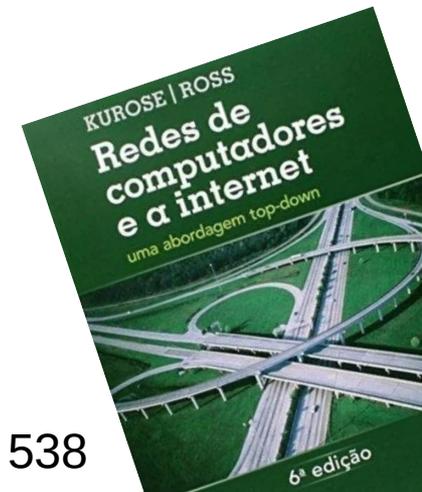


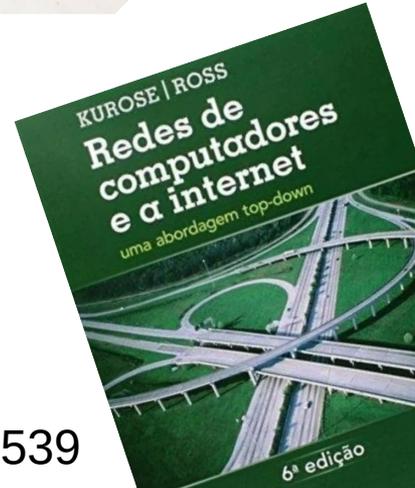
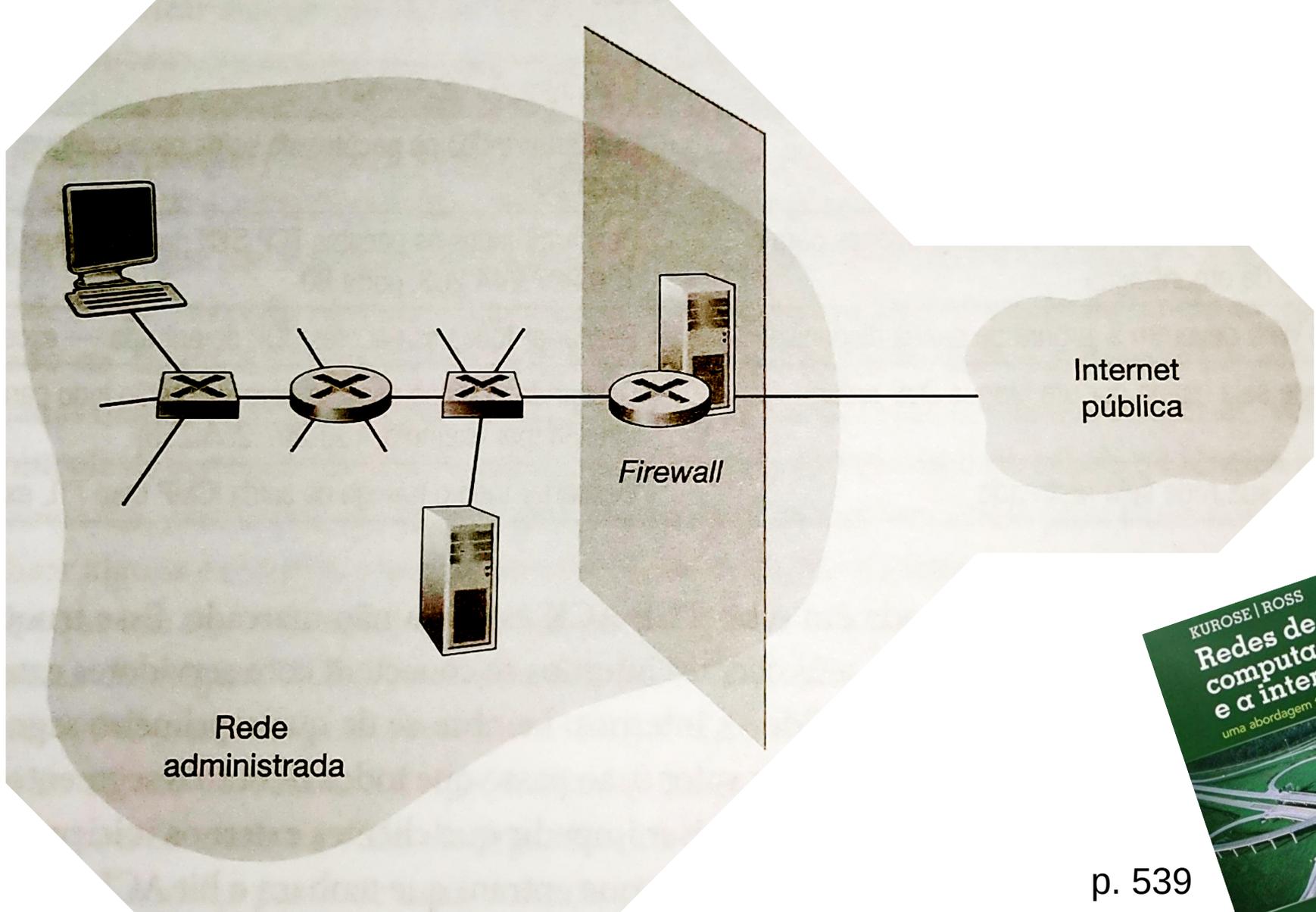
Firewall

com OpenFlow

Deixar o exemplo 15x15 executando

“Um *firewall* é uma combinação de hardware e software que isola a rede interna de uma organização da Internet em geral, permitindo que alguns pacotes passem e bloqueando outros.”





O que vem em `ryu.app.rest_firewall`?

- vLAN;
- Regras: `<MACs, MACd, P3, IPs/CIDR, IPd/CIDR, P4, Ps, Pd, Act>`
 - CRUD.
- Estatísticas para cada regra;
- Proativo: as regras devem ser adicionadas antes do fluxo de dados.

E se eu quisesse um firewall reativo?

- Começa como um switch “sem regras”;
- Fluxos novos passam pelo controlador, que decide se o fluxo será encaminhado ou descartado, e se essa ação será aplicada aos próximos pacotes.
- ...mas não está implementado.

```
501
502 class SimpleSwitch13(app_manager.RyuApp):
503     OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
504
505     def __init__(self, *args, **kwargs):
506         super(SimpleSwitch13, self).__init__(*args, **kwargs)
507         self.mac_to_port = {}
508         self.firewalls = dict()
509
510     @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
511     def switch_features_handler(self, ev):
512         datapath = ev.msg.datapath
513         ofproto = datapath.ofproto
514         parser = datapath.ofproto_parser
515
516         # install table with flow entry
```

```
577
578     actions = [parser.OFPActionOutput(out_port)]
579
580     # run firewall on it
581     if datapath.id not in self.firewalls:
582         self.firewalls[datapath.id] = Firewall(datapath)
583     actions, save_in_flow_table, *_ = [
584         *list(self.firewalls[datapath.id].actions_for_new_flow(
585             event=ev,
586             message=msg,
587             datapath=datapath,
588             ofproto=ofproto,
589             eth=eth,
590             parser=parser,
591             allow_action=actions
592         )),
593         True
594     ]
EOF
```

595

596

```
if save_in_flow_table:
```

597

```
    # install a flow to avoid packet_in next time
```

598

```
    if out_port != ofproto.OFPP_FLOOD:
```

599

```
        match = parser.OFPMatch(in_port=in_port, eth_dst=dst)
```

600

```
        # verify if we have a valid buffer_id, if yes avoid to send both
```

601

```
        # flow_mod & packet_out
```

602

```
        if msg.buffer_id != ofproto.OFP_NO_BUFFER:
```

603

```
            self.add_flow(datapath, 1, match, actions, msg.buffer_id)
```

604

```
            return
```

605

```
        else:
```

606

```
            self.add_flow(datapath, 1, match, actions)
```

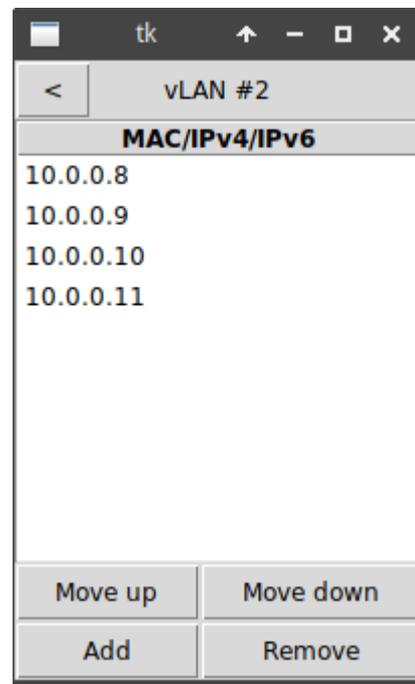
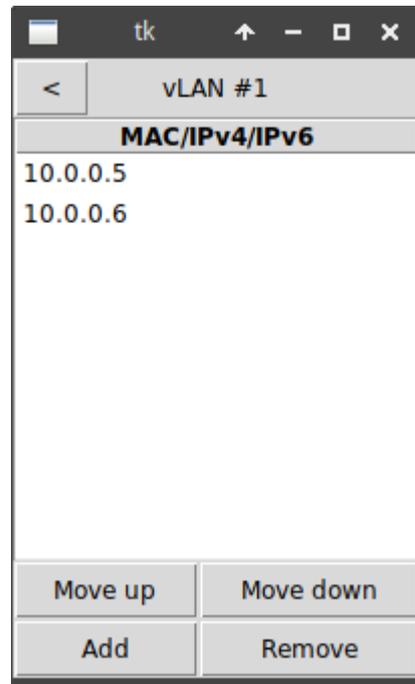
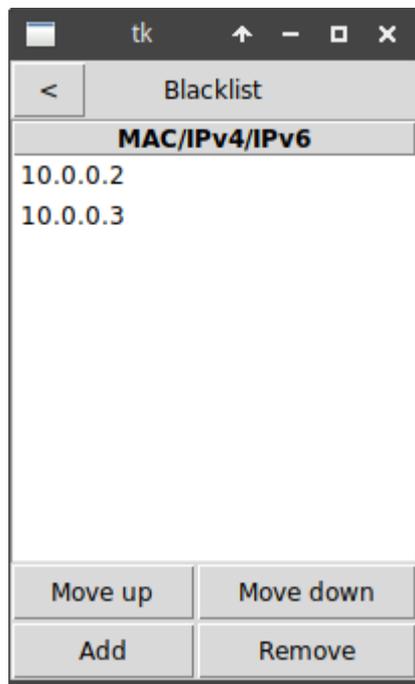
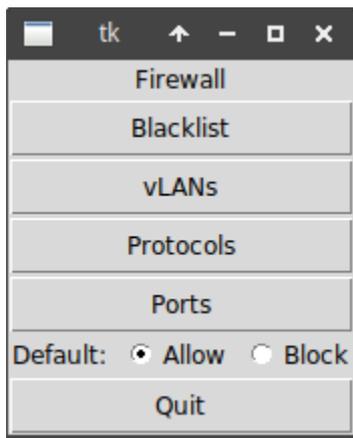
```

72
73 def actions_for_new_flow(self, event, message, datapath, ofproto, eth, parser, allow_action, disallow_action=[]):
74     self.load_rules()
75     default_action = allow_action if self.rules.get(
76         'default_action', True) else disallow_action
77     protocols = list(packet.Packet(message.data).protocols)
78     ipvx = [protocol for protocol in protocols if protocol.__class__.__name__ in [
79         'ipv4', 'ipv6']]
80     ipvx = ipvx[0] if len(ipvx) > 0 else None
81     tcudp = [protocol for protocol in protocols if protocol.__class__.__name__ in [
82         'tcp', 'udp']]
83     tcudp = tcudp[0] if len(tcudp) > 0 else None
84     vlanid_src = self.get_vlanid(
85         eth.src, ipvx.src if ipvx is not None else None)
86     vlanid_dst = self.get_vlanid(
87         eth.dst, ipvx.dst if ipvx is not None else None)
88     # devices banned from the network
89     if vlanid_src < 0 or vlanid_dst < 0:
90         return (disallow_action,)
91     # different vlans
92     if vlanid_src != vlanid_dst:
93         return (disallow_action,)
94     # special protocol on first flow
95     for rule in self.rules.get('special_protocols', list()):
96         match_action = allow_action if rule.get(
97             'action', True) else disallow_action
98         if rule['protocol'].lower() in [protocol.__class__.__name__.lower() for protocol in protocols]:
99             return (match_action, False)
100     # special tcp or udp port on first flow
101     if tcudp is not None:
102         for rule in self.rules.get('special_ports', list()):
103             match_action = allow_action if rule.get(
104                 'action', True) else disallow_action
105             ports = [*([tcudp.src_port] if rule.get('src_matters') else []),
106                 [*([tcudp.dst_port] if rule.get('dst_matters') else [])]]
107             matchrange = list(
108                 range(rule.get('portrange_start', 0), rule.get('portrange_stop', 65535)+1))
109             for port in ports:
110                 if port in matchrange:
111                     return (match_action, False)
112     return (default_action,)
113

```

E como isso seria a prática?

```
python firewall.py
```



tk

< Protocols

#	Protocol	Action
1	ARP	Allow
2	ICMP	Allow
3	UDP	Allow
4	IPV4	Drop

Move up

Add

tk

< Ports

#	Port range	Source/Destination	Action
1	22	Both source and destination	Allow
2	21 .. 23	Both source and destination	Drop

Move up

Add

Move down

Remove

```
1  {
2    "default_action": true,
3    "special_ports": [
4      {
5        "action": true,
6        "dst_matters": true,
7        "portrange_start": 22,
8        "portrange_stop": 22,
9        "src_matters": true
10     },
11     {
12       "action": false,
13       "dst_matters": true,
14       "portrange_start": 21,
15       "portrange_stop": 23,
16       "src_matters": true
17     }
18   ],
```

```
19     "special_protocols": [  
20         {  
21             "action": true,  
22             "protocol": "arp"  
23         },  
24         {  
25             "action": true,  
26             "protocol": "icmp"  
27         },  
28         {  
29             "action": true,  
30             "protocol": "udp"  
31         },  
32         {  
33             "action": false,  
34             "protocol": "ipv4"  
35         }  
36     ],
```

```
37     "static_vlans": {
38         "-1": [
39             "10.0.0.2",
40             "10.0.0.3"
41         ],
42         "1": [
43             "10.0.0.5",
44             "10.0.0.6"
45         ],
46         "2": [
47             "10.0.0.8",
48             "10.0.0.9",
49             "10.0.0.10",
50             "10.0.0.11"
51         ]
52     }
53 }
```

E como seria um pingall nisso?

```
sudo mn --topo=single,15 --mac --switch=ovsk --controller=remote -test=pingall
```

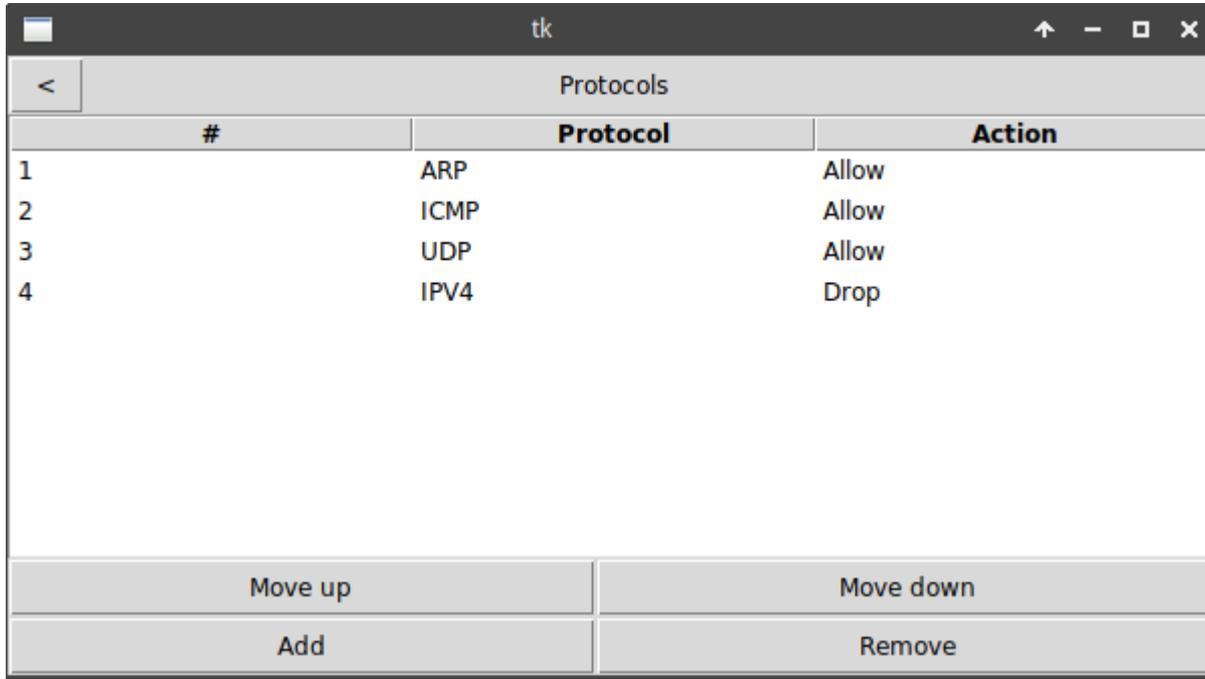
```
sudo ryu-manager firewall.py
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-			✓			✓					✓	✓	✓	✓
2		-													
3			-												
4	✓			-			✓					✓	✓	✓	✓
5					-	✓									
6					✓	-									
7	✓			✓			-					✓	✓	✓	✓
8								-	✓	✓	✓				
9								✓	-	✓	✓				
10								✓	✓	-	✓				
11								✓	✓	✓	-				
12	✓			✓			✓					-	✓	✓	✓
13	✓			✓			✓					✓	-	✓	✓
14	✓			✓			✓					✓	✓	-	✓
15	✓			✓			✓					✓	✓	✓	-

E só resume a definir quem porde pingar quem?

✓ `sudo mn --topo=single,15 --mac --switch=ovsk --controller=remote --test=iperfudp`

x `sudo mn --topo=single,15 --mac --switch=ovsk --controller=remote --test=iperf`



#	Protocol	Action
1	ARP	Allow
2	ICMP	Allow
3	UDP	Allow
4	IPV4	Drop

Move up Move down

Add Remove

Dúvidas?