REAWNZ

23 OCTOBER 2019

# THE KILLER TAP

## WHY YOU SHOULD DEPLOY FAUCET

REANNZ

## SDN DOESN'T LET HARDWARE DO ANYTHING IT COULDN'T DO ALREADY

## SIMPLIFY YOUR NETWORK WITH SDN

## SIMPLIFY YOUR NETWORK WITH SDN

"IN ALL EXAMPLES THAT COME TO MIND THE SIMPLE AND ELEGANT SYSTEMS TEND TO BE EASIER AND FASTER TO DESIGN AND GET RIGHT, MORE EFFICIENT IN EXECUTION AND MUCH MORE RELIABLE THAN THE MORE CONTRIVED CONTRAPTIONS THAT HAVE TO BE DEBUGGED INTO SOME DEGREE OF ACCEPTABILITY."

– E.W. Dijkstra

## SIMPLIFY YOUR NETWORK WITH SDN

"[…] THE VISION THAT AUTOMATIC COMPUTING SHOULD NOT BE SUCH A MESS IS OBSCURED, OVER AND OVER AGAIN, BY THE ADVENT OF A MONSTRUM THAT IS SUBSEQUENTLY FORCED UPON THE COMPUTING COMMUNITY AS A DE FACTO STANDARD (COBOL, FORTRAN, ADA, C++, SOFTWARE FOR DESKTOP PUBLISHING, YOU NAME IT)."

– E.W. Dijkstra again, later in the same talk

## SIMPLICITY CAN BE COMPLICATED

- Simpler for whom?
- To do what?
- What are the trade offs?

Simplicity needs to be in aid of a simple goal

## NETWORKING IS SIMPLE

- Take packets from a port
- Send them to another
- (Mostly)

## COORDINATION IS COMPLEX

- SDN adds a level of coordination between the controller and datapath
- But! It greatly reduces the coordination between datapaths
- Coordination isn't just between devices, its between protocols within a device.
  - Maybe we have a chance to turn some of those off?

## NETWORKING IN ONE EQUATION

```
= availability * security / (effort * cost)
```

## (OR MORE FORMALLY)

$$\sum_{f \in \mathbb{F}} \frac{P\left(a_f < u_f\right)}{(c_f . e_f . t_f)}$$

$$\mathbb{F}\ is\ the\ set\ of\ required\ features$$
$$a_x\ is\ the\ required\ availability\ of\ feature\ x$$
$$u_x\ is\ the\ observed\ uptime\ of\ feature\ x$$
$$c_x\ is\ the\ cost\ of\ operating\ feature\ x$$
$$e_x\ is\ the\ effort\ of\ operating\ feature\ x$$
$$t_x\ is\ the\ time\ required\ to\ operate\ feature\ x$$

## NETWORKING IN ONE EQUATION

```
= availability * security / (effort * cost)
```

## REDUCING EFFORT

- Software doesn’t write itself
- SDN needs a long term commitment

## LEARN TO LOVE YOUR FLOW TABLE

```
 ("OFPFlowStats
 "cookie": 1524
{"OXMTlv": {"ma
_timeout": 0, "by
ength": 4, "oxm_fi
pe": 17}}, {"OFPAc
pe": 1}}], "hard_ti
": {"OFPMatch": {"le
{"OFPFlowStats": {"ins
 "cookie": 1524372928,
{"OXMTlv": {"mask": nul
ructionActions": {"len":
": 1, "priority": 9000,
d": "in_port"}}, {"OXMTlv
OFPActionOutput": {"max_len
```

## THE FLOW TABLE IS A ONE STOP SHOP FOR ALL YOUR NETWORK DEBUGGING NEEDS

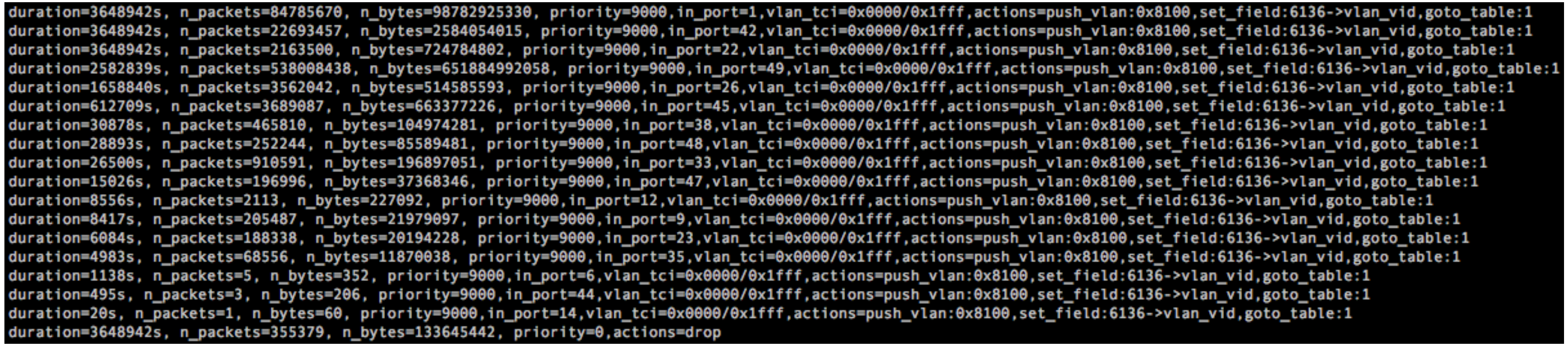
LEARN TO LOVE YOUR FLOW TABLE

## (ALMOST)

- MTU?
- What does happen to fragments in an Open Flow network?
- Maybe the Flow Table is wrong?

LEARN TO LOVE YOUR FLOW TABLE

## FLOW TABLES ARE INTIMIDATING



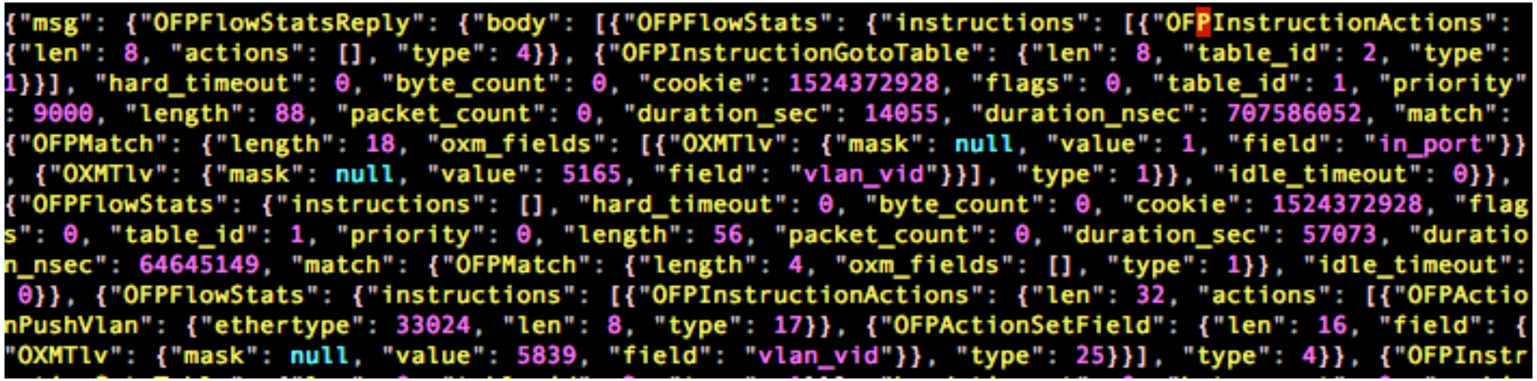
```
duration=3648942s, n_packets=84785670, n_bytes=98782925330, priority=9000,in_port=1,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=3648942s, n_packets=22693457, n_bytes=2584054015, priority=9000,in_port=42,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=3648942s, n_packets=2163500, n_bytes=724784802, priority=9000,in_port=22,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=2582839s, n_packets=538008438, n_bytes=651884992058, priority=9000,in_port=49,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=1658840s, n_packets=3562042, n_bytes=514585593, priority=9000,in_port=26,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=612709s, n_packets=3689087, n_bytes=663377226, priority=9000,in_port=45,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=30878s, n_packets=465810, n_bytes=104974281, priority=9000,in_port=38,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=28893s, n_packets=252244, n_bytes=85589481, priority=9000,in_port=48,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=26500s, n_packets=910591, n_bytes=196897051, priority=9000,in_port=33,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=15026s, n_packets=196996, n_bytes=37368346, priority=9000,in_port=47,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=8556s, n_packets=2113, n_bytes=227092, priority=9000,in_port=12,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=8417s, n_packets=205487, n_bytes=21979097, priority=9000,in_port=9,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=6084s, n_packets=188338, n_bytes=20194228, priority=9000,in_port=23,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=4983s, n_packets=68556, n_bytes=11870038, priority=9000,in_port=35,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=1138s, n_packets=5, n_bytes=352, priority=9000,in_port=6,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=495s, n_packets=3, n_bytes=206, priority=9000,in_port=44,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=20s, n_packets=1, n_bytes=60, priority=9000,in_port=14,vlan_tci=0x0000/0x1fff,actions=push_vlan:0x8100,set_field:6136->vlan_vid,goto_table:1
duration=3648942s, n_packets=355379, n_bytes=133645442, priority=0,actions=drop
```

## BUT THEY ARE VERY SWEET WHEN YOU GET TO KNOW THEM

- At least the nicely formatted ones you get directly from your switch usually are
- Priority, Matches and Actions
    - Hard to simplify things further without losing information
- Remember to subtract 4096 from your vlan vids
- Just follow your packet through the table

## GAUGE FLOW TABLE MONITORING

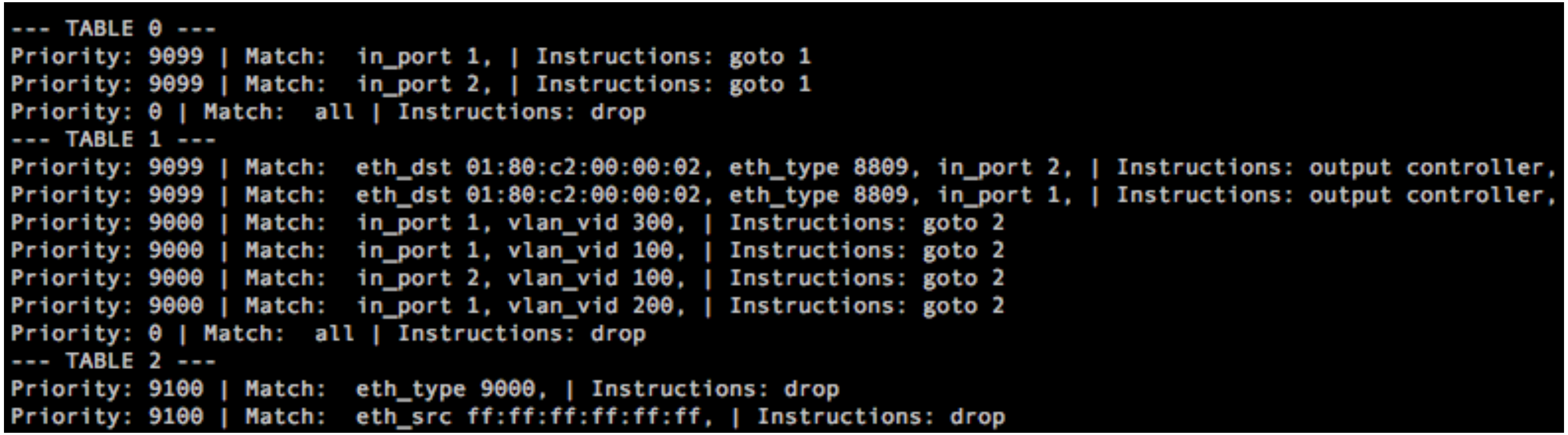
- Not particularly nicely formatted (for people)
- But it does save you logging into every device on the network



```
{"msg": {"OFPFlowStatsReply": {"body": [{"OFPFlowStats": {"instructions": [{"OFPInstructionActions":
{"len": 8, "actions": [], "type": 4}}, {"OFPInstructionGotoTable": {"len": 8, "table_id": 2, "type":
1}}], "hard_timeout": 0, "byte_count": 0, "cookie": 1524372928, "flags": 0, "table_id": 1, "priority"
: 9000, "length": 88, "packet_count": 0, "duration_sec": 14055, "duration_nsec": 707586052, "match":
{"OFPMatch": {"length": 18, "oxm_fields": [{"OXMTlv": {"mask": null, "value": 1, "field": "in_port"}}
, {"OXMTlv": {"mask": null, "value": 5165, "field": "vlan_vid"}}], "type": 1}}, "idle_timeout": 0}},
{"OFPFlowStats": {"instructions": [], "hard_timeout": 0, "byte_count": 0, "cookie": 1524372928, "flag
s": 0, "table_id": 1, "priority": 0, "length": 56, "packet_count": 0, "duration_sec": 57073, "duratio
n_nsec": 64645149, "match": {"OFPMatch": {"length": 4, "oxm_fields": [], "type": 1}}, "idle_timeout":
 0}}, {"OFPFlowStats": {"instructions": [{"OFPInstructionActions": {"len": 32, "actions": [{"OFPActio
nPushVlan": {"ethertype": 33024, "len": 8, "type": 17}}, {"OFPActionSetField": {"len": 16, "field": {
"OXMTlv": {"mask": null, "value": 5839, "field": "vlan_vid"}}, "type": 25}}], "type": 4}}, {"OFPInstr
```

## GAUGE FLOW TABLE MONITORING

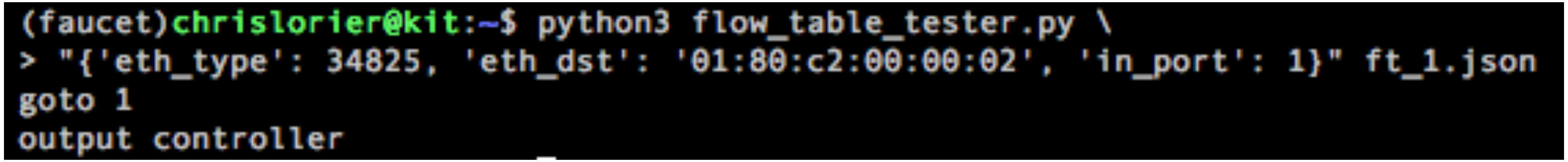
- And Software fixes everything!



```
--- TABLE 0 ---
Priority: 9099 | Match:  in_port 1, | Instructions: goto 1
Priority: 9099 | Match:  in_port 2, | Instructions: goto 1
Priority: 0 | Match:  all | Instructions: drop
--- TABLE 1 ---
Priority: 9099 | Match:  eth_dst 01:80:c2:00:00:02, eth_type 8809, in_port 2, | Instructions: output controller,
Priority: 9099 | Match:  eth_dst 01:80:c2:00:00:02, eth_type 8809, in_port 1, | Instructions: output controller,
Priority: 9000 | Match:  in_port 1, vlan_vid 300, | Instructions: goto 2
Priority: 9000 | Match:  in_port 1, vlan_vid 100, | Instructions: goto 2
Priority: 9000 | Match:  in_port 2, vlan_vid 100, | Instructions: goto 2
Priority: 9000 | Match:  in_port 1, vlan_vid 200, | Instructions: goto 2
Priority: 0 | Match:  all | Instructions: drop
--- TABLE 2 ---
Priority: 9100 | Match:  eth_type 9000, | Instructions: drop
Priority: 9100 | Match:  eth_src ff:ff:ff:ff:ff:ff, | Instructions: drop
```

## QUERYABLE

Why aren’t my LACP packets getting forwarded???



```
(faucet)chrislorier@kit:~$ python3 flow_table_tester.py \
> "{'eth_type': 34825, 'eth_dst': '01:80:c2:00:00:02', 'in_port': 1}" ft_1.json
goto 1
output controller
```

## QUERYABLE

Three possibilities:

- That rule shouldn’t be there!
- Some other rule is missing!
- Oh! I’m an idiot!

But this requires the ability to know why any given rule in the flow table is there

## THE FLOW TABLE IS A POWERFUL TOOL

- The more complex we make it, the higher we raise the bar for people to take advantage of it
- Documentation and tooling helps
- But ultimately the simpler it is, the easier it will be

## MORE FLOW TABLE FUN

## TESTABLE

- Write a test that verifies that your network will never forward LACP

## FLOW TABLES AREN'T THAT BIG

- You can run this in reverse too
- Find all possible packets that could output a packet to port 1
- If port 1 has reasonably tight access control, this would not be too onerous

## SUPER-CYBER-RFP

- You could automatically build test packets to do a complete search of all possible rule combinations from a given flow table
- 100% flow table test coverage

## WHAT IF IT ISN'T THE FLOW TABLE

- The device is doing something I think the flow table says it shouldn't
- Try loading the flow table onto Open vSwitch, is the behavior consistent?
    - Am I being an idiot?
- Try loading the flow table onto another of the same device, is the behavior consistent?
    - Is it a hardware fault or a software fault?

REAVNZ