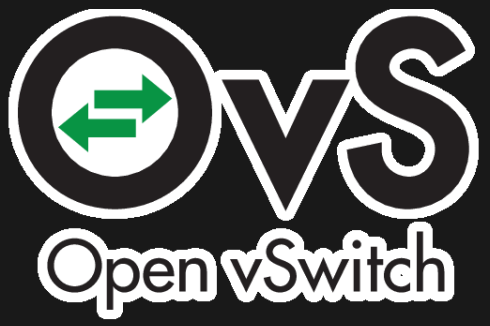
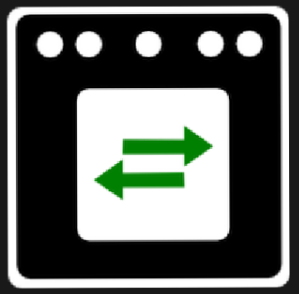
OVN EXPRESSION PARSING: FIGHTING INEQUALITY





Open vSwitch and OVN 2023 Fall Conference

Ilya Maximets, Red Hat
i.maximets@ovn.org

Kubernetes network policy

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
spec:
  egress:
    - to:
        - ipBlock:
            cidr: 172.17.0.0/16
            except:
              - 172.17.1.0/24
```

Kubernetes network policy

```
9               cidr: 172.17.0.0/16
```

OVN ACL match:

```
1       ip4.dst == 172.17.0.0/16
```

Kubernetes network policy

```
 9             cidr: 172.17.0.0/16
10             except:
11               - 172.17.1.0/24
```

OVN ACL match:

```
1      ip4.dst == 172.17.0.0/16 && ip4.dst != {
2          172.17.1.0/24,
3      }
```

Kubernetes network policy

```
 9               cidr: 172.17.0.0/16
10               except:
11                 - 172.17.1.0/24
12                 - 172.17.5.0/24
```

OVN ACL match:

```
1      ip4.dst == 172.17.0.0/16 && ip4.dst != {
2          172.17.1.0/24,
3          172.17.5.0/24,
4      }
```

Kubernetes network policy

```
 9             cidr: 172.17.0.0/16
10             except:
11               - 172.17.1.0/24
12               - 172.17.5.0/24
13               - 172.17.7.0/24
```

OVN ACL match:

```
1      ip4.dst == 172.17.0.0/16 && ip4.dst != {
2          172.17.1.0/24,
3          172.17.5.0/24,
4          172.17.7.0/24,
5      }
```

OVN ACL match:

```
1      ip4.dst == 172.17.0.0/16
```

OpenFlow rules:

```
1 ip,nw_dst=172.17.0.0/16    actions=...
```

OVN ACL match:

```
    ip4.dst == 172.17.0.0/16 && ip4.dst != {
        172.17.1.0/24,
    }
```

OpenFlow rules:

```
ip,nw_dst=172.17.0.0/255.255.1.0    actions=...
ip,nw_dst=172.17.2.0/255.255.2.0    actions=...
ip,nw_dst=172.17.4.0/255.255.4.0    actions=...
ip,nw_dst=172.17.8.0/255.255.8.0    actions=...
ip,nw_dst=172.17.16.0/255.255.16.0  actions=...
ip,nw_dst=172.17.32.0/255.255.32.0  actions=...
ip,nw_dst=172.17.64.0/255.255.64.0  actions=...
ip,nw_dst=172.17.128.0/17           actions=...
```

OVN ACL match:

```
1      ip4.dst == 172.17.0.0/16 && ip4.dst != {
2          172.17.1.0/24,
3          172.17.3.0/24,
4      }
```

OpenFlow rules:

```
 1 ip,nw_dst=172.17.0.0/255.255.1.0     actions=...
 2 ip,nw_dst=172.17.0.0/255.255.3.0     actions=...
 3 ip,nw_dst=172.17.10.0/255.255.10.0   actions=...
 4 ip,nw_dst=172.17.12.0/255.255.12.0   actions=...
 5 ip,nw_dst=172.17.128.0/17            actions=...
 6 ip,nw_dst=172.17.128.0/255.255.129.0 actions=...
 7 ip,nw_dst=172.17.128.0/255.255.130.0 actions=...
 8 ip,nw_dst=172.17.130.0/255.255.130.0 actions=...
 9 ip,nw_dst=172.17.132.0/255.255.132.0 actions=...
10 ip,nw_dst=172.17.136.0/255.255.136.0 actions=...
11 ip,nw_dst=172.17.144.0/255.255.144.0 actions=...
12 ip,nw_dst=172.17.160.0/255.255.160.0 actions=...
13 ip,nw_dst=172.17.16.0/255.255.16.0   actions=...
14 ip,nw_dst=172.17.16.0/255.255.17.0   actions=...
15 ip,nw_dst=172.17.16.0/255.255.18.0   actions=
```

OVN ACL match:

```
    ip4.dst == 172.17.0.0/16 && ip4.dst != {
        172.17.1.0/24,
        172.17.3.0/24,
        172.17.5.0/24,
    }
```

OpenFlow rules:

```
ip,nw_dst=172.17.66.0/255.255.70.0   actions=...
ip,nw_dst=172.17.96.0/255.255.98.0   actions=...
ip,nw_dst=172.17.70.0/255.255.70.0   actions=...
ip,nw_dst=172.17.128.0/17            actions=...
ip,nw_dst=172.17.24.0/255.255.24.0   actions=...
ip,nw_dst=172.17.56.0/255.255.56.0   actions=...
ip,nw_dst=172.17.164.0/255.255.164.0 actions=...
ip,nw_dst=172.17.16.0/255.255.22.0   actions=...
ip,nw_dst=172.17.32.0/255.255.34.0   actions=...
ip,nw_dst=172.17.16.0/255.255.18.0   actions=...
  ... 140 more OpenFlow rules ...
```

Except CIDRs (/24):	OpenFlow rules:
1	8
2	42
4	304
8	1803
16	3366

Except CIDRs (/32):	OpenFlow rules:	Time:
1	16	0 s
2	150	0 s
4	3.7 K	0.03 s
8	214 K	3.3 s
16	1.1 M	1 m 31 s
32	1.7 M	8 m 2 s

Except CIDRs (/32):	OpenFlow rules:	Time:
8	214 K	3.3 s
16	1.1 M	1 m 31 s
32	1.7 M	8 m 2 s

Kubernetes network policy:

```
9               cidr: 172.17.0.0/16
```

Original CIDR contains only 64 K IP addresses ...

... and we generate 1.7 M OpenFlow rules ?!

OVN Logical Flow expressions have a rich syntax:

```
ip4.src == 172.17.1.1
ip4.dst != { 172.17.1.1, 172.17.1.7 }
tcp.dst == 1234 || tcp.dst > 1234
ip4 && (tcp && tcp.src == 443 || udp && udp.dst == 53)
```

OpenFlow rules only allow exact/masked matches:

```
nw_src=172.17.1.1
nw_dst=172.17.0.0/16
ip,nw_src=172.17.1.0/24,tp_src=443
ip,udp,tp_dst=0xab00/0xff00
```

How can we translate one into another?

1. Get rid of all the comparison operations that are not an exact/masked match.

Set match:

```
ip4.dst == { 172.17.1.1, 172.17.1.7 }
```

Translates into:

```
ip4.dst == 172.17.1.1 || ip4.dst == 172.17.1.7
```

Translates into:

```
ip,nw_dst=172.17.1.1, actions= ...
ip,nw_dst=172.17.1.7, actions= ...
```

Range match:

```
433 < tcp.dst < 1024
```

Binary representation:

```
 433 -> 0000 0001 1011 0001
1024 -> 0000 0100 0000 0000
```

```
1        0000 0001 1011 001-   434-435
2        0000 0001 1011 01--   436-439
3        0000 0001 1011 1---   440-447
4        0000 0001 11-- ----   448-511
5        0000 001- ---- ----   512-1023
```

Range match:

```
 433 < tcp.dst < 1024
```

Binary representation:

```
 433 -> 0000 0001 1011 0001
1024 -> 0000 0100 0000 0000
```

```
1        0000 0001 1011 001-   434-435    0x1b2/0xfffe
2        0000 0001 1011 01--   436-439    0x1b4/0xfffc
3        0000 0001 1011 1---   440-447    0x1b8/0xfff8
4        0000 0001 11-- ----   448-511    0x1c0/0xffc0
5        0000 001- ---- ----   512-1023   0x200/0xfe00
```

Range match:

```
 433 < tcp.dst < 1024
```

Translates into:

```
tcp.dst == 0x1b2/0xfffe || tcp.dst == 0x1b4/0xfffc
 || tcp.dst == 0x1b8/0xfff8 || tcp.dst == 0x1c0/0xffc0
 || tcp.dst == 0x200/0xfe00
```

Translates into:

```
1 tcp,tp_dst=0x1b2/0xfffe, actions= ...
2 tcp,tp_dst=0x1b4/0xfffc, actions= ...
3 tcp,tp_dst=0x1b8/0xfff8, actions= ...
4 tcp,tp_dst=0x1c0/0xffc0, actions= ...
5 tcp,tp_dst=0x200/0xfe00, actions= ...
```

Inequality match:

```
tcp.dst[0..7] != 177
```

Binary representation:

```
177 -> 0000 0000 1011 0001
```

```
1       ---- ---- ---- ---1  // tcp.dst[0] != 1
2       ---- ---- ---- --0-  // tcp.dst[1] != 0
3       ---- ---- ---- -0--  // tcp.dst[2] != 0
4       ---- ---- ---- 0---  // tcp.dst[3] != 0
5       ---- ---- ---1 ----  // tcp.dst[4] != 1
6       ---- ---- --1- ----  // tcp.dst[5] != 1
7       ---- ---- -0-- ----  // tcp.dst[6] != 0
8       ---- ---- 1--- ----  // tcp.dst[7] != 1
```

Inequality match:

```
tcp.dst[0..7] != 177
```

Binary representation:

```
177 -> 0000 0000 1011 0001
```

```
1       ---- ---- ---- ---0  // tcp.dst[0] == 0
2       ---- ---- ---- --1-  // tcp.dst[1] == 1
3       ---- ---- ---- -1--  // tcp.dst[2] == 1
4       ---- ---- ---- 1---  // tcp.dst[3] == 1
5       ---- ---- ---0 ----  // tcp.dst[4] == 0
6       ---- ---- --0- ----  // tcp.dst[5] == 0
7       ---- ---- -1-- ----  // tcp.dst[6] == 1
8       ---- ---- 0--- ----  // tcp.dst[7] == 0
```

Inequality match:

```
tcp.dst[0..7] != 177
```

Binary representation:

```
177 -> 0000 0000 1011 0001
```

```
1       ---- ---- ---- ---0  // tcp.dst[0] == 0   0x0/0x1
2       ---- ---- ---- --1-  // tcp.dst[1] == 1   0x2/0x2
3       ---- ---- ---- -1--  // tcp.dst[2] == 1   0x4/0x4
4       ---- ---- ---- 1---  // tcp.dst[3] == 1   0x8/0x8
5       ---- ---- ---0 ----  // tcp.dst[4] == 0   0x0/0x10
6       ---- ---- --0- ----  // tcp.dst[5] == 0   0x0/0x20
7       ---- ---- -1-- ----  // tcp.dst[6] == 1   0x40/0x40
8       ---- ---- 0--- ----  // tcp.dst[7] == 0   0x0/0x80
```

Inequality match:

```
tcp.dst[0..7] != 177
```

Translates into:

```
tcp.dst == 0x0/0x1 || tcp.dst == 0x2/0x2
    || tcp.dst == 0x4/0x4   || tcp.dst == 0x8/0x8
    || tcp.dst == 0x0/0x10  || tcp.dst == 0x0/0x20
    || tcp.dst == 0x40/0x40 || tcp.dst == 0x0/0x80
```

Translates into:

```
1 tcp,tp_dst=0x0/0x1,   actions= ...
2 tcp,tp_dst=0x2/0x2,   actions= ...
3 tcp,tp_dst=0x4/0x4,   actions= ...
4 tcp,tp_dst=0x8/0x8,   actions= ...
5 tcp,tp_dst=0x0/0x10,  actions= ...
6 tcp,tp_dst=0x0/0x20,  actions= ...
7 tcp,tp_dst=0x40/0x40, actions= ...
8 tcp,tp_dst=0x0/0x80,  actions= ...
```

1. Get rid of all the comparison operations that are not an exact/masked match.
2. Normalize.

Normalization

```
ip4 && ( ip4.src == 127.0.0.1 || ip4.src == 192.168.0.1)
  && ( icmp || tcp && ( tcp.dst == 80 || tcp.dst == 443 ) )
```

```
A: ip4                      -> eth_type=0x0800 // or just 'ip'
B: ip4.src == 127.0.0.1     -> nw_src=127.0.0.1
C: ip4.src == 192.168.0.1   -> nw_src=192.168.0.1
D: icmp                     -> ip_proto=1 // or just 'icmp'
E: tcp                      -> ip_proto=6 // or just 'tcp'
F: tcp.dst == 80            -> tp_dst=80
G: tcp.dst == 443           -> tp_dst=443
```

```
A && ( B || C ) && ( D || E && ( F || G))
```

```
A && ( B || C ) && ( D || E && ( F || G))
```

$$A(B \vee C)(D \vee E(F \vee G))$$

$$\rightarrow A(B \vee C)(D \vee EF \vee EG)$$

$$\rightarrow (AB \vee AC)(D \vee EF \vee EG)$$

$$\rightarrow ABD \vee ABEF \vee ABEG$$
$$\vee ACD \vee ACEF \vee ACEG$$

DNF (Disjunctive Normal Form):

$$ABD \vee ABEF \vee ABEG \vee ACD \vee ACEF \vee ACEG$$

```
A: ip4                      -> eth_type=0x0800 // or just 'ip'
B: ip4.src == 127.0.0.1     -> nw_src=127.0.0.1
C: ip4.src == 192.168.0.1   -> nw_src=192.168.0.1
D: icmp                     -> ip_proto=1 // or just 'icmp'
E: tcp                      -> ip_proto=6 // or just 'tcp'
F: tcp.dst == 80            -> tp_dst=80
G: tcp.dst == 443           -> tp_dst=443
```

DNF (Disjunctive Normal Form):

$$ABD \vee ABEF \vee ABEG$$
$$\vee ACD \vee ACEF \vee ACEG$$

Translates into OpenFlow rules:

```
1 ip,nw_src=127.0.0.1,icmp,                actions= ...
2 ip,nw_src=127.0.0.1,tcp,tp_dst=80,       actions= ...
3 ip,nw_src=127.0.0.1,tcp,tp_dst=443,      actions= ...
4 ip,nw_src=192.168.0.1,icmp               actions= ...
5 ip,nw_src=192.168.0.1,tcp,tp_dst=80,     actions= ...
6 ip,nw_src=192.168.0.1,tcp,tp_dst=443,    actions= ...
```

OVN Logical Flow match:

```
ip4 && ( ip4.src == 127.0.0.1 || ip4.src == 192.168.0.1)
  && ( icmp || tcp && ( tcp.dst == 80 || tcp.dst == 443 ) )
```

Translates into OpenFlow rules:

```
1 ip,nw_src=127.0.0.1,icmp,                actions= ...
2 ip,nw_src=127.0.0.1,tcp,tp_dst=80,       actions= ...
3 ip,nw_src=127.0.0.1,tcp,tp_dst=443,      actions= ...
4 ip,nw_src=192.168.0.1,icmp               actions= ...
5 ip,nw_src=192.168.0.1,tcp,tp_dst=80,     actions= ...
6 ip,nw_src=192.168.0.1,tcp,tp_dst=443,    actions= ...
```

OVN ACL match:

```
1       ip4.dst == 172.17.0.0/16 && ip4.dst != {
2           172.17.1.0/24,
3           172.17.3.0/24,
4       }
```

```
    ip4.dst == 172.17.0.0/16 &&
    ip4.dst != {
        172.17.1.0/24,
        172.17.3.0/24,
    }
```

```
1       ip4.dst == 172.17.0.0/16 &&
2       ip4.dst != 172.17.1.0/24 &&
3       ip4.dst != 172.17.3.0/24
```

```
  172.17.1.0/24 = 10101100.00010001.00000001.--------

(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1 ||
 ip4.dst[11] == 1 || ip4.dst[12] == 1 || ip4.dst[13] == 1 ||
 ip4.dst[14] == 1 || ip4.dst[15] == 1 ||

 ip4.dst[16] == 0 || ip4.dst[17] == 1 || ip4.dst[18] == 1 ||
 ip4.dst[19] == 1 || ip4.dst[20] == 0 || ip4.dst[21] == 1 ||
 ip4.dst[22] == 1 || ip4.dst[23] == 1 ||

 ip4.dst[24] == 1 || ip4.dst[25] == 1 || ip4.dst[26] == 0 ||
 ip4.dst[27] == 0 || ip4.dst[28] == 1 || ip4.dst[29] == 0 ||
 ip4.dst[30] == 1 || ip4.dst[31] == 0 )
```

```
 1      ip4.dst == 172.17.0.0/16 &&
 2 (ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1 |
 3  ip4.dst[11] == 1 || ip4.dst[12] == 1 || ip4.dst[13] == 1 |
 4  ip4.dst[14] == 1 || ip4.dst[15] == 1 ||
 5
 6  ip4.dst[16] == 0 || ip4.dst[17] == 1 || ip4.dst[18] == 1 |
 7  ip4.dst[19] == 1 || ip4.dst[20] == 0 || ip4.dst[21] == 1 |
 8  ip4.dst[22] == 1 || ip4.dst[23] == 1 ||
 9
10  ip4.dst[24] == 1 || ip4.dst[25] == 1 || ip4.dst[26] == 0 |
11  ip4.dst[27] == 0 || ip4.dst[28] == 1 || ip4.dst[29] == 0 |
12  ip4.dst[30] == 1 || ip4.dst[31] == 0 ) &&
13      ip4.dst != 172.17.3.0/24
```

```
    ip4.dst == 172.17.0.0/16 &&

(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1 |
 ip4.dst[11] == 1 || ip4.dst[12] == 1 || ip4.dst[13] == 1 |
 ip4.dst[14] == 1 || ip4.dst[15] == 1 ||
 ip4.dst[16] == 0 || ip4.dst[17] == 1 || ip4.dst[18] == 1 |
 ip4.dst[19] == 1 || ip4.dst[20] == 0 || ip4.dst[21] == 1 |
 ip4.dst[22] == 1 || ip4.dst[23] == 1 ||
 ip4.dst[24] == 1 || ip4.dst[25] == 1 || ip4.dst[26] == 0 |
 ip4.dst[27] == 0 || ip4.dst[28] == 1 || ip4.dst[29] == 0 |
 ip4.dst[30] == 1 || ip4.dst[31] == 0 ) &&

(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1 |
 ip4.dst[11] == 1 || ip4.dst[12] == 1 || ip4.dst[13] == 1 |
 ip4.dst[14] == 1 || ip4.dst[15] == 1 ||
```

```
1      ip4.dst == 172.17.0.0/16 &&
2 (ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1 ||
3  ip4.dst[11] == 1 || ip4.dst[12] == 1 || ip4.dst[13] == 1 ||
4  ip4.dst[14] == 1 || ip4.dst[15] == 1 ) &&
5 (ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1 ||
6  ip4.dst[11] == 1 || ip4.dst[12] == 1 || ip4.dst[13] == 1 ||
7  ip4.dst[14] == 1 || ip4.dst[15] == 1 )
```

1 * 8 * 8 = 64 terms

Almost a half of them are exactly the same:

- `ip4.dst[ 8] == 0 && ip4.dst[11] == 1`
- `ip4.dst[11] == 1 && ip4.dst[ 8] == 0`

Bit 9 is different, so we end up with 42 (not 32) rules.

What happens if we use /32 instead of /24 ?

Need to also match on low 8 bits now.

1 * 16 * 16 = 256 terms

More than 128 rules (150 in practice).

What happens if we also remove the
ip4.dst == 172.17.0.0/16 ?

Can no longer assume the value of the top 16 bits.
32 * 32 = 1024 terms

And these numbers are for just 2 excluded CIDRs.

Number of terms hits 1 billion with just 6 CIDRs.

Visible effects:

1. High CPU usage on ovn-controller
2. High memory usage on ovn-controller
3. High CPU usage on ovs-vswitchd.
   Packet classification becomes expensive.
   Potentially consuming all cores.

What can we do?

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0 && ip4.dst[ 8] == 0
2 ip4.dst[ 8] == 0 && ip4.dst[ 9] == 0
3 ip4.dst[ 8] == 0 && ip4.dst[10] == 1
4 ip4.dst[ 9] == 1 && ip4.dst[ 8] == 0
5 ip4.dst[ 9] == 1 && ip4.dst[ 9] == 0
6 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
7 ip4.dst[10] == 1 && ip4.dst[ 8] == 0
8 ip4.dst[10] == 1 && ip4.dst[ 9] == 0
9 ip4.dst[10] == 1 && ip4.dst[10] == 1
```

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0 && ip4.dst[ 8] == 0
2 ip4.dst[ 8] == 0 && ip4.dst[ 9] == 0
3 ip4.dst[ 8] == 0 && ip4.dst[10] == 1
4 ip4.dst[ 9] == 1 && ip4.dst[ 8] == 0
5 ip4.dst[ 9] == 1 && ip4.dst[ 9] == 0
6 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
7 ip4.dst[10] == 1 && ip4.dst[ 9] == 0
8 ip4.dst[10] == 1 && ip4.dst[10] == 1
```

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0 && ip4.dst[ 8] == 0
2 ip4.dst[ 8] == 0 && ip4.dst[ 9] == 0
3 ip4.dst[ 8] == 0 && ip4.dst[10] == 1
4 ip4.dst[ 9] == 1 && ip4.dst[ 8] == 0
5 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
6 ip4.dst[10] == 1 && ip4.dst[ 9] == 0
7 ip4.dst[10] == 1 && ip4.dst[10] == 1
```

OVN v23.03 doesn't perform any further optimizations.

Can we do somthing else?

Yes!

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0 && ip4.dst[ 8] == 0
2 ip4.dst[ 8] == 0 && ip4.dst[ 9] == 0
3 ip4.dst[ 8] == 0 && ip4.dst[10] == 1
4 ip4.dst[ 9] == 1 && ip4.dst[ 8] == 0
5 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
6 ip4.dst[10] == 1 && ip4.dst[ 9] == 0
7 ip4.dst[10] == 1 && ip4.dst[10] == 1
```

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0
2 ip4.dst[10] == 1
3 ip4.dst[ 8] == 0 && ip4.dst[ 9] == 0
4 ip4.dst[ 8] == 0 && ip4.dst[10] == 1
5 ip4.dst[ 9] == 1 && ip4.dst[ 8] == 0
6 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
7 ip4.dst[10] == 1 && ip4.dst[ 9] == 0
```

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0
2 ip4.dst[10] == 1
3 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
4 ip4.dst[10] == 1 && ip4.dst[ 9] == 0
```

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 1) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0
2 ip4.dst[10] == 1
```

9 terms reduced to 2.

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 0) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0 && ip4.dst[ 8] == 0
2 ip4.dst[ 8] == 0 && ip4.dst[ 9] == 0
3 ip4.dst[ 8] == 0 && ip4.dst[10] == 1
4 ip4.dst[ 9] == 1 && ip4.dst[ 8] == 0
5 ip4.dst[ 9] == 1 && ip4.dst[ 9] == 0
6 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
7 ip4.dst[10] == 0 && ip4.dst[ 8] == 0
8 ip4.dst[10] == 0 && ip4.dst[ 9] == 0
9 ip4.dst[10] == 0 && ip4.dst[10] == 1
```

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 0) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0 && ip4.dst[ 8] == 0
2 ip4.dst[ 8] == 0 && ip4.dst[ 9] == 0
3 ip4.dst[ 8] == 0 && ip4.dst[10] == 1
4 ip4.dst[ 9] == 1 && ip4.dst[ 8] == 0
5 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
6 ip4.dst[10] == 0 && ip4.dst[ 8] == 0
7 ip4.dst[10] == 0 && ip4.dst[ 9] == 0
```

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 0) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0
2 ip4.dst[ 8] == 0 && ip4.dst[ 9] == 0
3 ip4.dst[ 8] == 0 && ip4.dst[10] == 1
4 ip4.dst[ 9] == 1 && ip4.dst[ 8] == 0
5 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
6 ip4.dst[10] == 0 && ip4.dst[ 8] == 0
7 ip4.dst[10] == 0 && ip4.dst[ 9] == 0
```

```
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 1 || ip4.dst[10] == 0) &&
(ip4.dst[ 8] == 0 || ip4.dst[ 9] == 0 || ip4.dst[10] == 1)
```

```
1 ip4.dst[ 8] == 0
2 ip4.dst[ 9] == 1 && ip4.dst[10] == 1
3 ip4.dst[10] == 0 && ip4.dst[ 9] == 0
```

9 terms reduced to 3.

More different are the initial values, more rules are required.

Optimization steps:

1. Sort the terms based on the number of bits in the mask.
2. Starting from the smallest, eliminate all the larger terms that are supersets of the current one.
3. Go to the next smallest term.

$$N^2$$

-ish complexity in the worst case.

Techniques: array-based lists with path compression.

Luckily, people are not using sets of completely random IP addresses.

They do not, right... ? :)

Except CIDRs (/32)	OF rules (v23.03)	OF rules (v23.06)
1	16	16
2	150	15
4	3.7 K	14
8	214 K	18
16	1.1 M	32
32	1.7 M	31

ACL with just an inequality match on 20 completely random /32 IP addresses generates ~126 K OpenFlow rules and takes a lot of time to compute.

Conclusions:

- Don't write ACLs like that !!! :)
- Use priorities instead of negative matches ! E.g. deny all + allow specific. (any version of OVN)
- Use ACL tiers (since v23.06).
- Use K8s Admin Network Policies.
- Try to split large ACLs into smaller ones.
- If none of the above is possible, make sure you're on OVN v23.06+.

Thank you!

Special credit to **Nadia Pinaeva** (Red Hat) for highlighting unoptimally generated OpenFlow rules leading to this improvement.