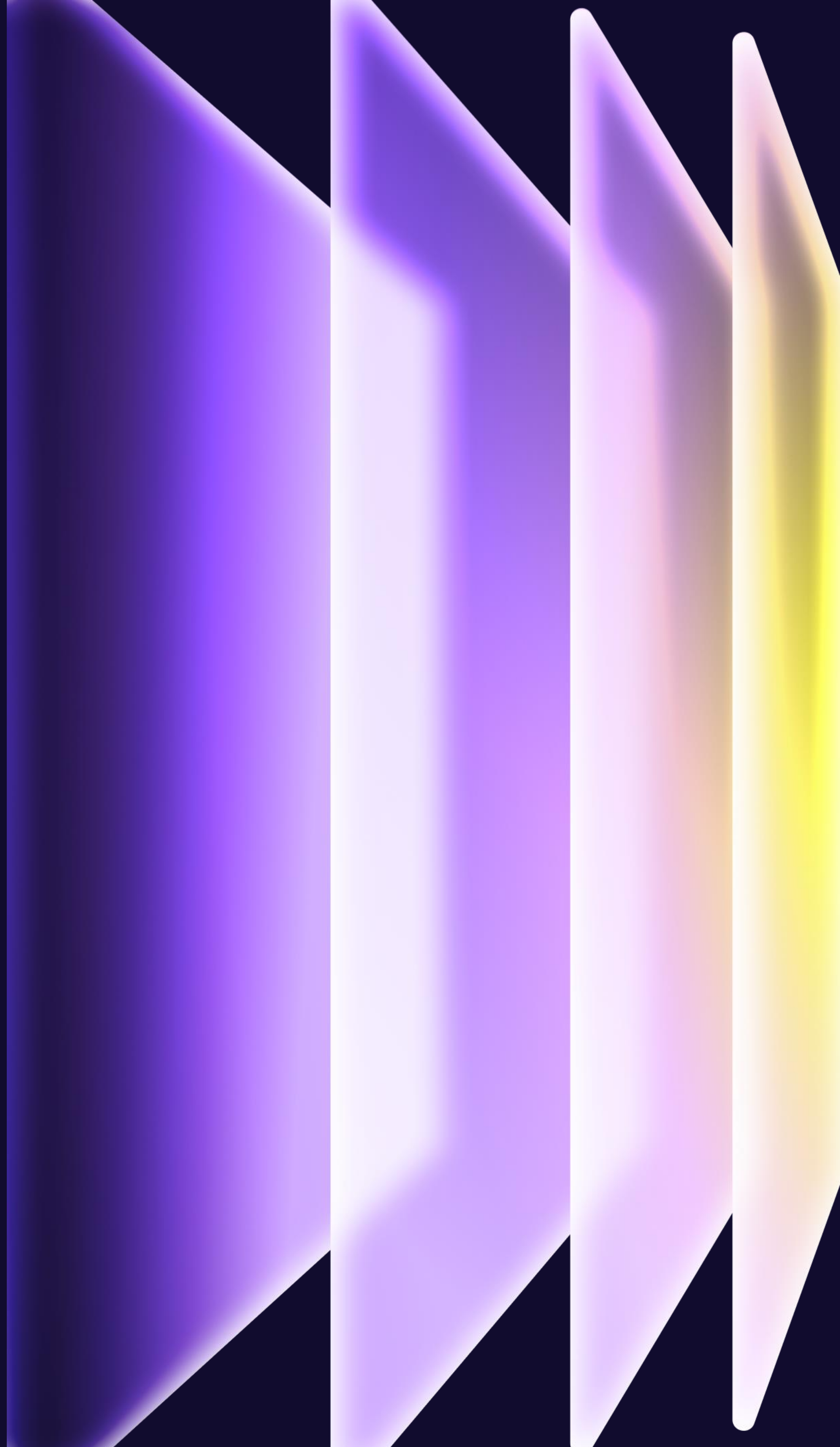
Infrastructure

Can you fit 5$\overset{7}{\cancel{0}}$0M rules into a firewall?

Unleash The ipfw Power To Serve

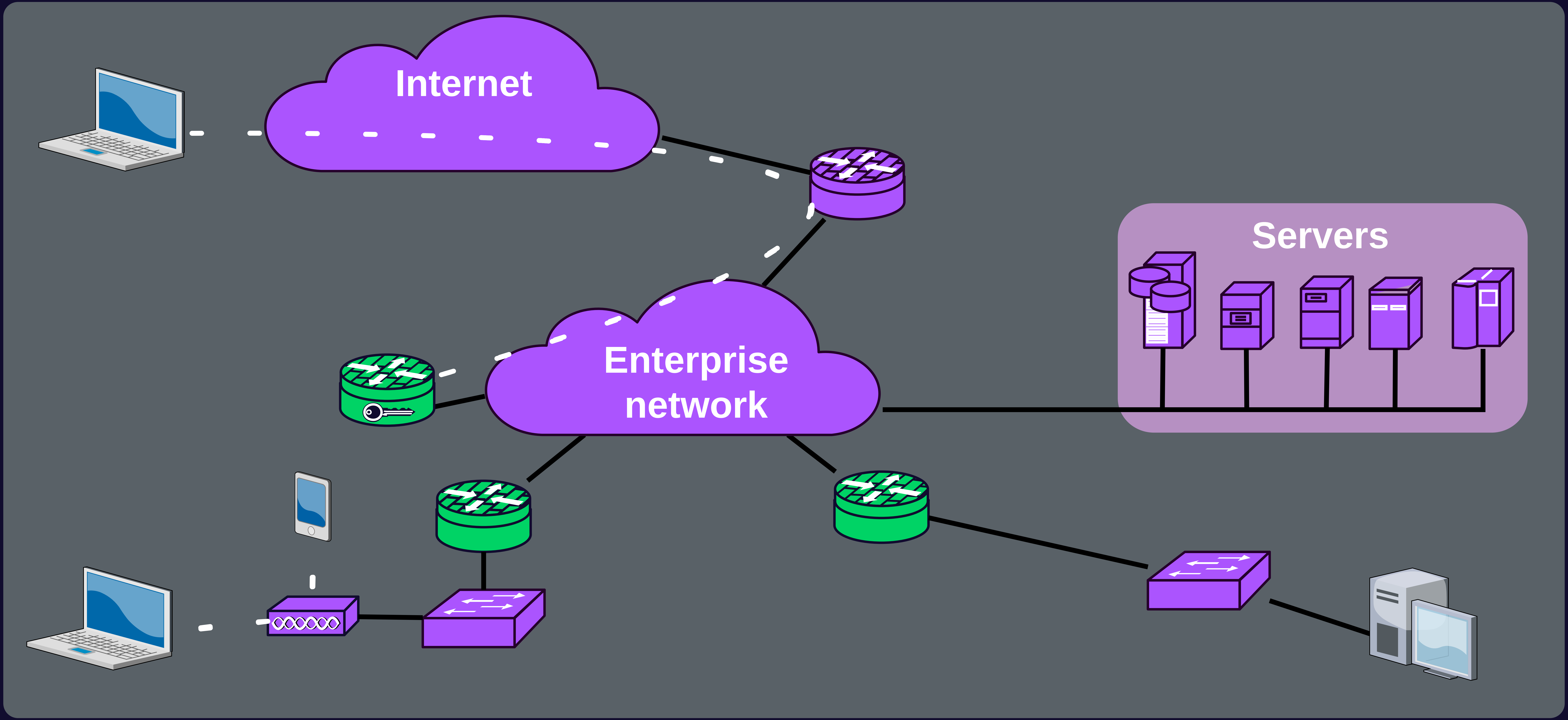
Boris Lytochkin,
Network foreman at Yandex NOC



Who am I?

2002-2008	2006-2008	2008 – …	2017 – …
MSU Physics dep.	**Duty sysadmin at e-port**	**NOC engineer at Yandex**	**Accredited IPv6 trainer @ ENOG**
Laser physics	Servers	Network	Basic IPv6 course
PCB design	Network	Office networks	Advanced IPv6 course
FPGA programming	Security	OOB networks	
		Firewalls	

First hop firewalls



«Dynamic» firewall

1

Limits network access from user laptops to corporate servers and the Internet

2

Combined with first hop router of corporate network segment

3

Capable to run on Intel Xeon-D class servers (slow, low TDP)

4

Dynamic: ready o serve in ~1..3 seconds after a user is connected

Access rules:
– Source is a user
and a media type.
– Destination is a CIDR/FQDN
with TCP/UDP ports.

«Dynamic» firewall: challenges to solve

1

Per-user granularity

2

Different rules for different media types

3

User IPs change, frequently

4

A user can connect in every office or VPN

«Dynamic» firewall: corners to cut

1

TCP
and UDP only

2

Most rules are made for large groups

3

Load users as they connect, remove ruleset upon traffic timeout

Departments as a rule source

```
@dpt_all
  @dpt_devs
    @dpt1_grp1
      %user1
      %user2
    @dpt1_grp2
      %user3
      @dpt1_grp3
        %user4
  @dpt_ops
    @dpt2_grp4
      %user5
      %user6
```

Projects as a rule source

```
@prj_struct
  @prj1
    @prg1_role1
      %user1
      %user4
  @prj2
    @prj2_role2
      %user1
      %user3
      %user5
    @prj2_role3
      %user2
      %user6
```

Some examples

% = user account

```
# Developer tests some service
add allow tcp from %user1% to devserver.example.com 22,80,443 via wired
```

@ = group of users / groups

```
# Everybody needs DNS
add allow { tcp or udp } from @dpt_all@ to dnscache.example.com 53
# expands to
add allow { tcp or udp } from { %user1% or %user2% or %user3% or %user4% \
      or %user5% or %user6% } to dnscache.example.com 53

# Members of Project 1 have full access to the project’s networks
add allow tcp from @prj1@ to 2001:db8:0001::/48
# expands to
add allow tcp from { %user1% or %user4% } to 2001:db8:0001::/48
```

Our ruleset

50k	Rules
65k	Unique users
570M	Single user → single CIDR entries

2. Convert user to IP:

```
  add allow tcp from
%user1%
       to
2001:db8:0001::/48
```

User IP addresses are volatile and change frequently (IPv4 + IPv6 + IPv6 security extensions + …)

Use [fw]mark (D39555) as a user ID in rules

[fw]mark (D39555)

- similar to fwmark @ Linux
- 32-bit number stored in mbuf(9) tag
- set: setmark <num>/tablearg
- check: mark <num>
- table lookup:
  lookup mark <tablenum>
- masking before lookup/check

ipfw lookup tables

Type (key)

- **Address**
- **Number**
- Interface
- MAC address
- Flow

Algo

- **Radix tree**
- **Array**
- Hash

Valtype

- **Skipto**
- **Mark**
- Tag
- Nat

…

Using mark as a source

```
add allow tcp from { %user1% or %user4% } to 2001:db8:0001::/48 80,443
```

```
table 1 create type addr valtype mark

table 1 add \
       2001:db8:ffff::1245:1111    0x10 \ // user1 source IP -> mark 0x10
       2001:db8:ffff::1245:2222    0x11   // user4 source IP -> mark 0x11

table 1 info
--- table(1), set(0) ---
 kindex: 11, type: addr
 references: 0, valtype: mark
 algorithm: addr:radix
 items: 2, size: 536

add setmark tablearg ip from table(1) to any in
add allow tcp from any to 2001:db8:0001::/48 { mark 0x10 or mark 0x11 } 80,443
```

Pure ipfw rule!

‘User → IP’ map sidecar

***MQ to fill ‘IP → mark’ table**

Two messages:

1. ‘map: user → IP,media’
2. ‘flush: IP’

Evaluating sequentially 50k rules is not an option

- sequential rules check
- sequential { mark or mark or mark … } in rules check (120M or’s!)
- no ‘change rule’, use ‘delete + add’
- ‘add rule’ is slow!

There are 65k rule numbers only

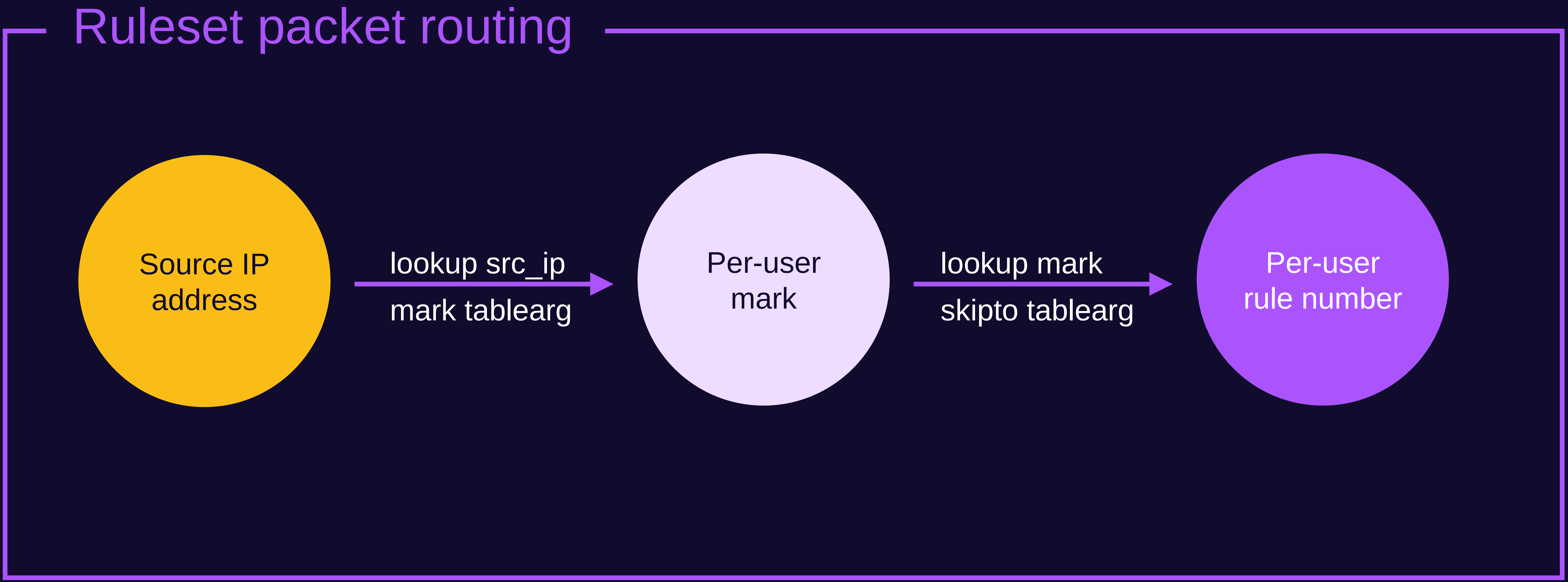
D46183 is here to help!

- 32-bit rule numbers
- return next-rule

Evaluating 50k rules sequentially is still not an option

- Convert high-level rules into per-user rulesets
- One rule number per user
- Ruleset of an average user: 13k entries

Source IP processing



One rule number per user

```
add allow tcp from { %user1% or %user4% } to 2001:db8:0001::/48 80,443
add allow tcp from { %user4% } to 2001:db8:0000::/40 22
```

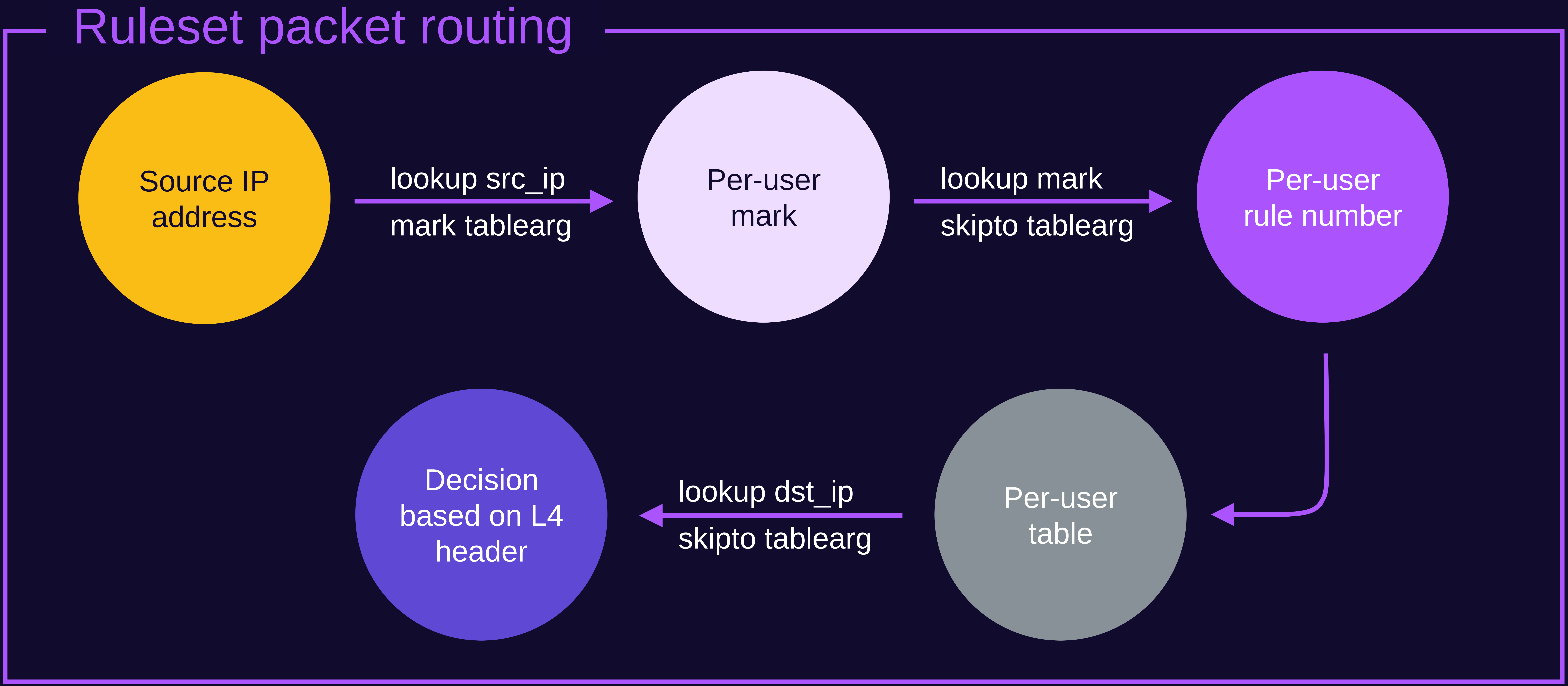
```
table 2 create type number valtype skipto // mark -> skipto map
table 2 add     \
        0x11 11 \     // user1 mark -> user rule number
        0x14 14       // user4 mark

add skipto tablearg ip from any to any lookup mark 2 // lookup mark in table(2)

add 11 allow tcp from any to 2001:db8:0001::/48 80,443 // user1 ruleset start
// … 13k rules for an average user
add 11 deny  ip from any to any                        // user1 ruleset end

add 14 allow tcp from any to 2001:db8:0001::/48 80,443 // user4 ruleset start
add 14 allow tcp from any to 2001:db8:0000::/40 22
add 14 deny  ip from any to any                        // user4 ruleset end
```

13k rules per user is still not an option

```
allow tcp from any to 2001:db8:0000::/40   22
allow tcp from any to 2001:db8:0000::/48   3306
allow tcp from any to 2001:db8:0001::/48   80,443
allow udp from any to 2001:db8:0001::7/128 53
allow tcp from any to 2001:db8:0001::8/128 8080
deny  ip  from any to any
```

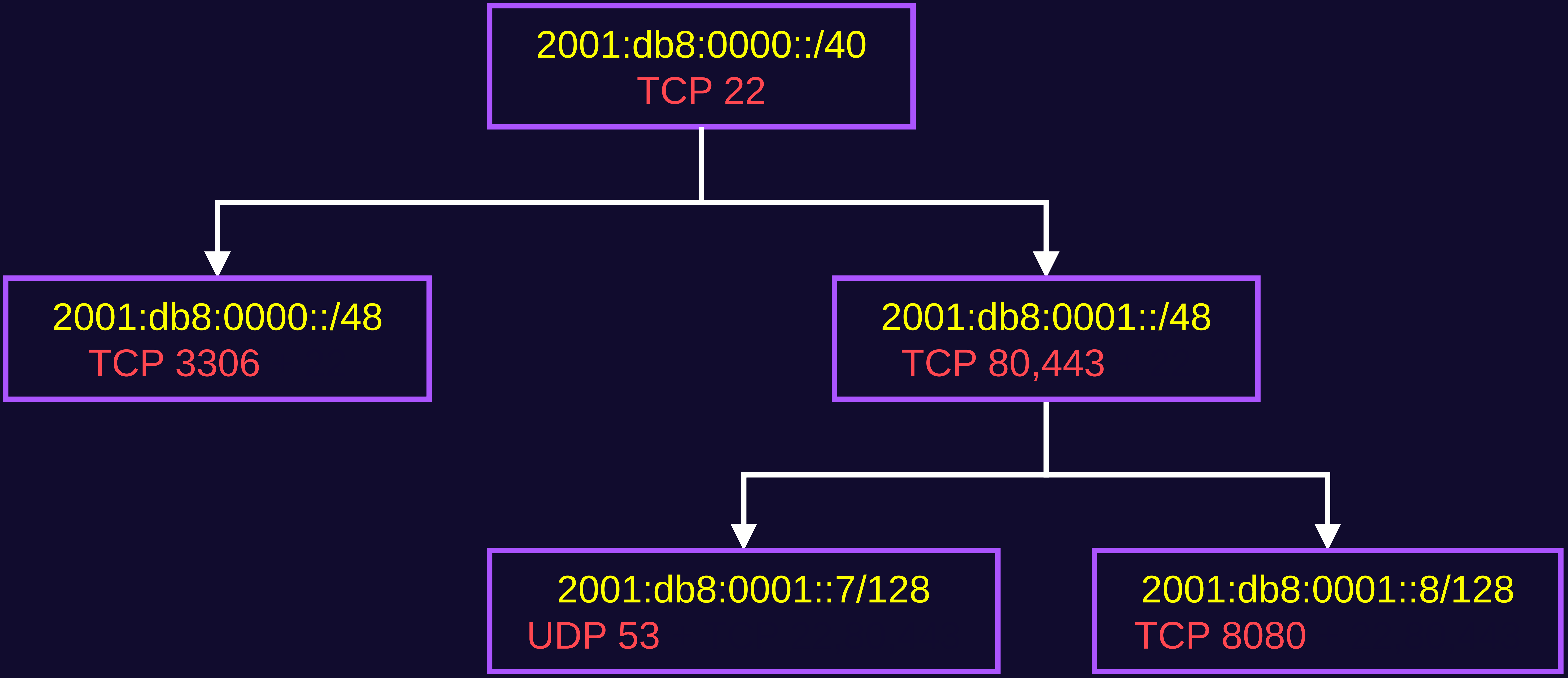
Map: CIDR → proto, ports

Ruleset as a radix tree



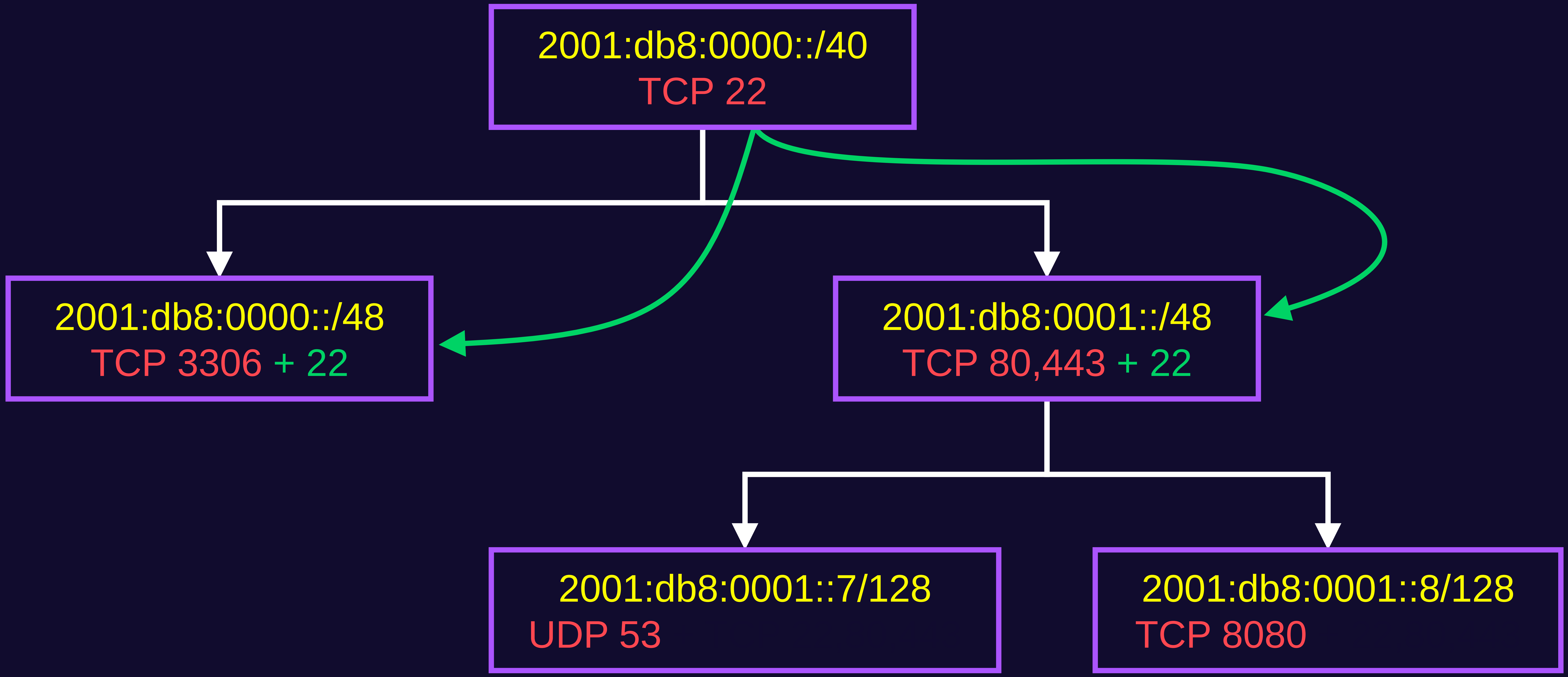
Ruleset as a radix tree

tcp 2001:db8:0001::8 22



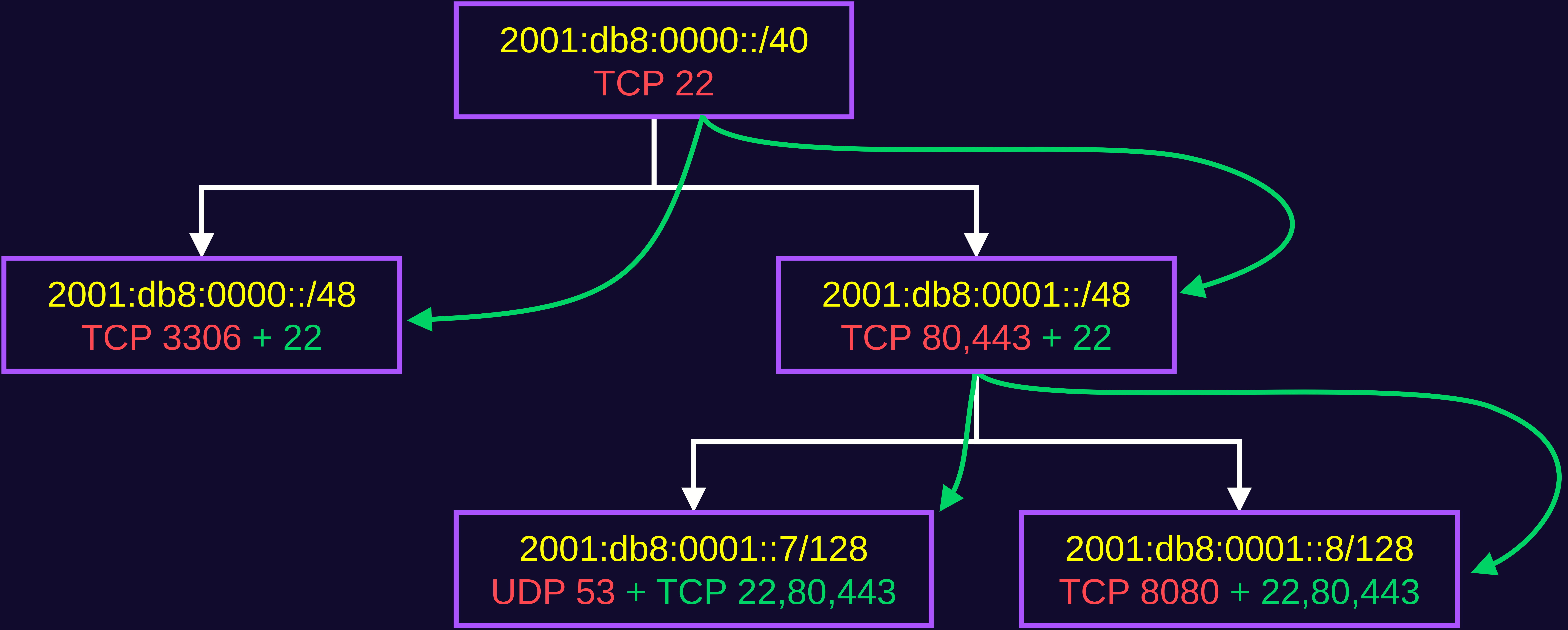
Ruleset as a radix tree

tcp 2001:db8:0001::8 22



Ruleset as a radix tree

tcp 2001:db8:0001::8 22



Ruleset as a radix tree

TCP 22	`add 50001 allow tcp from any to any 22`
TCP 3306 + 22	`add 50002 allow tcp from any to any 22,3306`
TCP 80,443 + 22	`add 50003 allow tcp from any to any 20,80,443`

UDP 53 + TCP 22,80,443	`add 50004 allow tcp from any to any 20,80,443`
	`add 50004 allow udp from any to any 53`

TCP 8080 + 22,80,443	`add 50005 allow tcp from any to any 22,80,443,8080`

Ruleset as a radix tree

50001
2001:db8:0000::/40
TCP 22

50002
2001:db8:0000::/48
TCP 3306 + 22

50003
2001:db8:0001::/48
TCP 80,443 + 22

50004
2001:db8:0001::7/128
UDP 53 + TCP 22,80,443

50005
2001:db8:0001::8/128
TCP 8080 + 22,80,443

```
table 14 create type address valtype skipto // user4 pfx table
table 14 add \
    2001:db8:0000::/40   50001 \
    2001:db8:0000::/48   50002 \
    2001:db8:0001::/48   50003 \
    2001:db8:0001::7/128 50004 \
    2001:db8:0001::7/128 50005


add 14 skipto tablearg ip from any to table(14) // user4 rs start
add 14 deny ip from any to any // user4 ruleset end

…
add 50003 allow tcp from any to any 20,80,443
add 50003 deny tcp from any to any
…
```

We’ve split
570M hard-to-modify
rules into easy-to-update
per-user tables and
a user-count-proportional ruleset

Our ruleset

570M	Single user → single CIDR entries
5k ☑ Optimized	Unique [Proto + Port Mask + Media] combinations throughout all rules
80k ☑ Optimized	Destination IP addresses / prefixes, including FQDNs resolved

IRL lots of rules are company-wide

```
# Everybody needs DNS
add allow { tcp or udp } from @dpt_all@ to dnscache.example.com 53

# Everybody needs a task tracker
add allow tcp from @dpt_all@ to tracker.example.com 80,443
add allow udp from @dpt_all@ to tracker.example.com 443 // QUIC too

# Everybody needs a jump host
add allow tcp from @dpt_all@ to bastion.example.com 22
add allow udp from @dpt_all@ to bastion.example.com 60000-61000 // mosh too

# Developer tests some service..
add allow tcp from @dpt_devs@ to devserver.example.com 22,80,443 via wired

# Members of Project 1 have full access to the project's networks
add allow tcp from @prj1@ to 2001:db8:0001::/48
```

65k users

25k users

65k x 5 = ~325k table entries

25k table entries

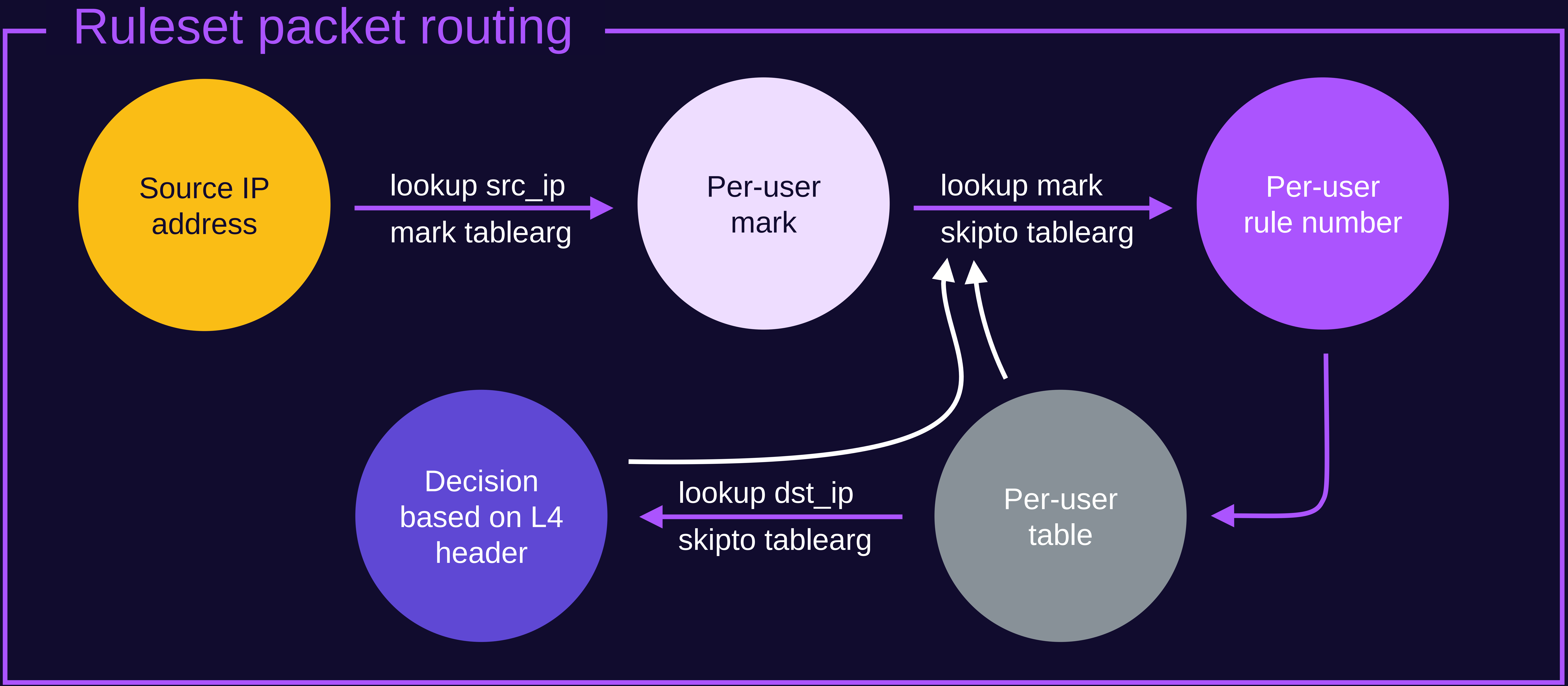
IRL lots of rules are company-wide

Now it’s
a “user”

```
# Everybody needs DNS
add allow { tcp or udp } from %dpt_all% to dnscache.example.com 53

# Everybody needs a task tracker
add allow tcp from %dpt_all% to tracker.example.com 80,443
add allow udp from %dpt_all% to tracker.example.com 443 // QUIC too

# Everybody needs a jump host
add allow tcp from %dpt_all% to bastion.example.com 22
add allow udp from %dpt_all% to bastion.example.com 60000-61000 // mosh too


# Developer tests some service
add allow tcp from %dpt_devs% to devserver.example.com 22,80,443 via wired

# Members of Project 1 have full access to the project’s networks
add allow tcp from @prj1@ to 2001:db8:0001::/48
```

Left as
a group

Ruleset as a radix tree



Ruleset with @dpt_all@ converted into a «user»

```
add 14 call tablearg ip from any to table(14) // user4 rs start, no dpt_all here
add 14 deny ip from any to any                // user4 rs end
```

Terminal rule

```
add 40001 call tablearg ip from any to table(40001) // dpt_all ruleset start
add 40001 return next-rule ip from any to any       // dpt_all ruleset end
```

“No match” is not fatal

```
add 50003 allow tcp from any to any 20,80,443
add 50003 return next-rule tcp from any to any
```

“No match” is not fatal

Referencing %dpt_all% pseudo user

```
table 2 create type number valtype skipto // mark -> skipto map
table 2 add    0x11 11 \      // user1 mark -> user1 rule number
               0x14 14 \      // user4
               0x15 15 \      // user5

add skipto tablearg ip from any to any lookup mark 2 // lookup mark in table(2)


table 3 create type number valtype skipto // mark -> skipto map to #1 pseudo user rs
table 3 add    0x11 40001 \   // user1 mark -> %dpt_all%  rule number (40001)
               0x14 40001 \   // user4
               0x15 40001     // user5

add call tablearg ip from any to any lookup mark 3 // lookup mark in table(3)


// more tables & skipto tablearg as needed
```

Put everything together

```
table 2 create type number valtype skipto // mark -> skipto map to user ruleset
table 3 create type number valtype skipto // mark -> skipto map to #1 pseudo user ruleset
table 4 create type number valtype skipto // mark -> skipto map to #2 pseudo user ruleset


// entering table walker
add 2 call tablearg ip from any to any lookup mark 3 // check call #1 pseudo user
add 3 call tablearg ip from any to any lookup mark 4 // check call #2 pseudo user
add 4 skipto tablearg ip from any to any lookup mark 2 // check real user ruleset at the end
add 5 deny ip from any to any // not reached unless no match in table(2)


add 14 call tablearg ip from any to table(14) // user4 ruleset start, no pseudo users here
add 14 return next-rule ip from any to any    // user4 ruleset end

add 40001 call tablearg ip from any to table(40001) // dpt_all ruleset start
add 40001 return next-rule ip from any to any       // dpt_all ruleset end
```

Dedup results

~~570M~~ ☑ Optimized

Single user → single CIDR entries

25M

Entries with deduplicated common rules

```
> vmstat -m | egrep 'MemUse|ipfw'
  Type      InUse    MemUse    HighUse Requests Size
  ipfw_tbl 17042852 2130357K  -        88973812 128
```

Dedup limitations

- One call to pseudo user ruleset implies an additional radix table lookup
- Convert large groups only

Some tips

1

Check single field in a packet once

2

Keep ruleset as small as possible

3

Use tables,
use <action> tablearg

4

Use mark
it rocks!

Questions?

Boris Lytochkin

Network foreman
Yandex Core Infrastructure
lytboris@yandex-team.com