XDPeriments: Tinkering with DNS and XDP

Willem Toorop, Luuk Hendriks
NLnet Labs
OARC 34 - virtual

Motivation & goals

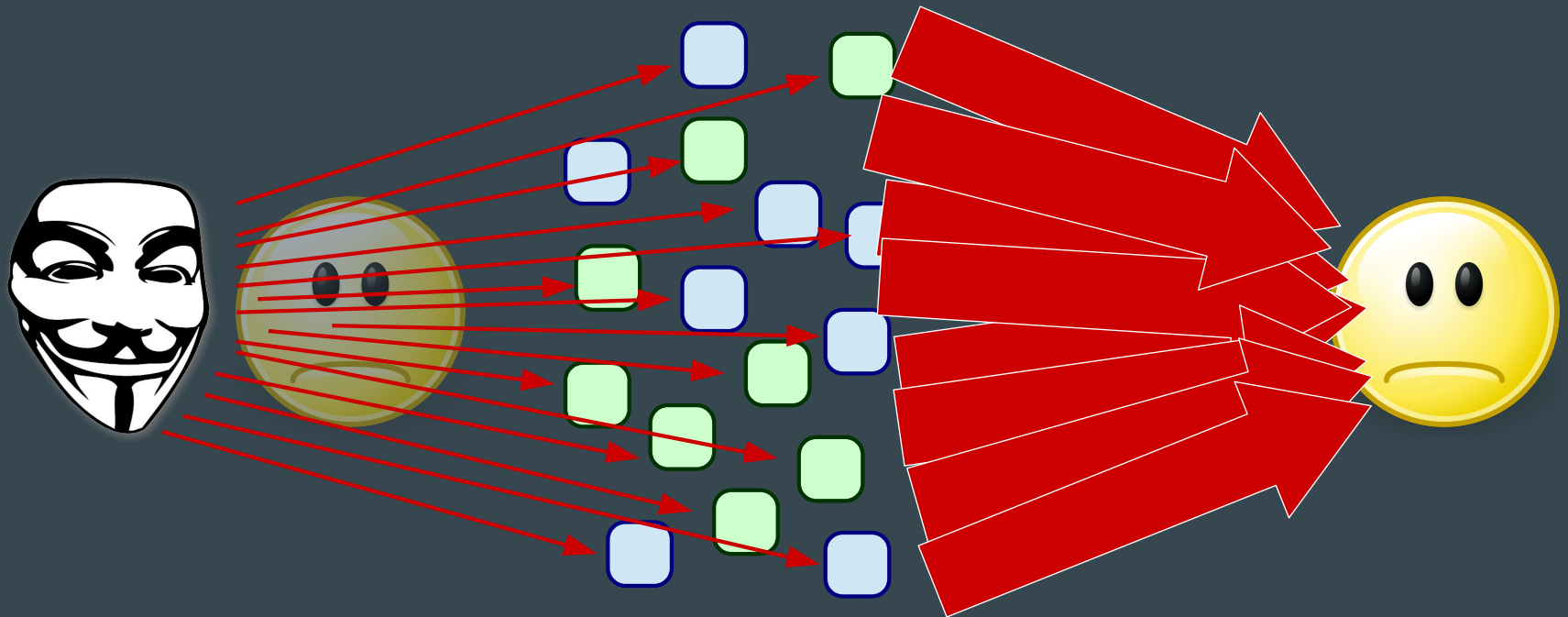
- Programmable networks are hot (see also: P4), and for good reasons!
- Flexibility in the data plane without sacrificing performance
- Specifically using XDP: easy way to perform some parts *in kernel* (heavy lifting) but still have traditional userspace software 'after' that.

XDP does not have to replace everything we do in userspace, it can *augment* it.

-> Focus in this presentation: RRL

Response Rate Limiting 101

- When Queries per Second > X (from certain source IP or Prefix)
- Then Return truncated (or drop)



(e)BPF, XDP, DNS

(Extended) Berkeley Packet Filter:

> Once the VM that handles your `tcpdump` filters, now a much more powerful concept with a slightly deceiving name: run verified code in kernel space without rebooting.

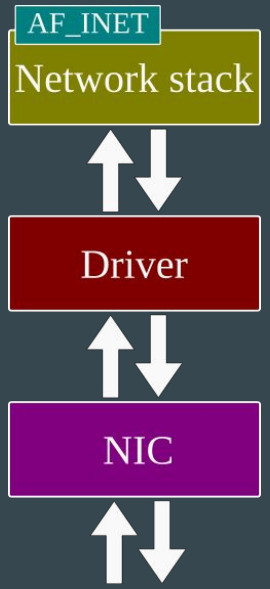
eXpress Data Path:

> Network driver hook to run BPF code. Executed before anything happens in the kernel networking stack.

DNS:

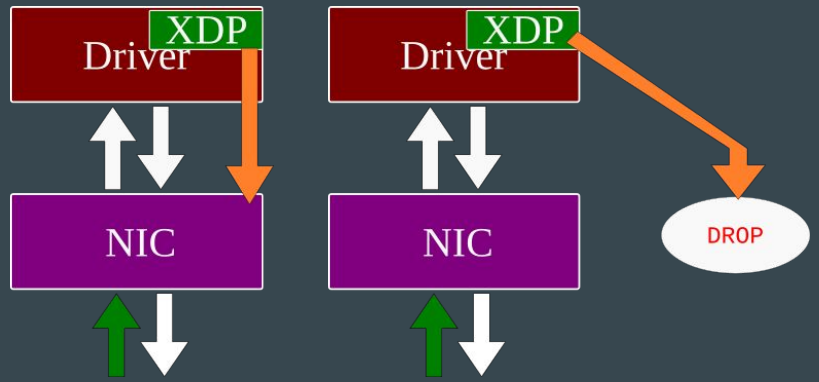
> Just DNS.

A packet's destiny: XDP return codes



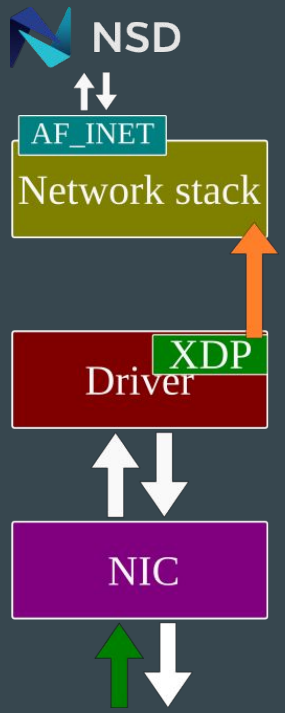
Classic stack, no XDP

A packet's destiny: XDP return codes



XDP_TX: send it out of ingress NIC
XDP_DROP: drop the packet

A packet's destiny: XDP return codes

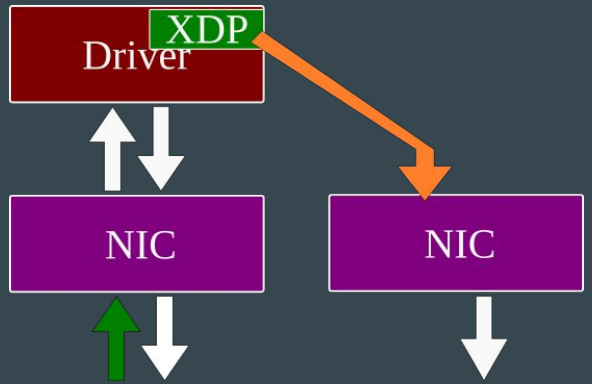


XDP_TX: send it out of ingress NIC

XDP_DROP: drop the packet

XDP_PASS: pass on to network stack

A packet's destiny: XDP return codes



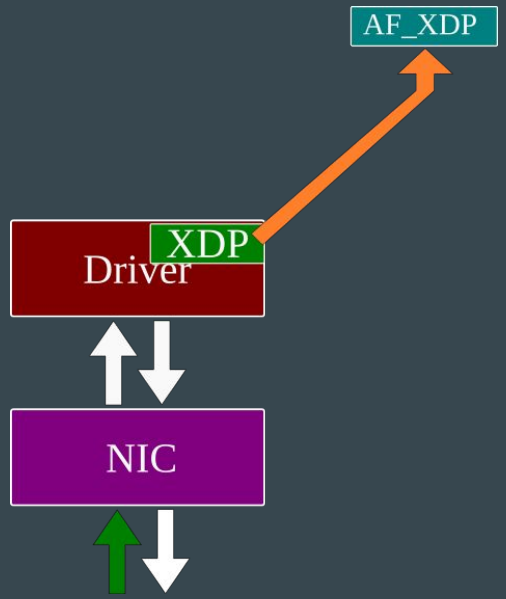
XDP_TX: send it out of ingress NIC

XDP_DROP: drop the packet

XDP_PASS: pass on to network stack

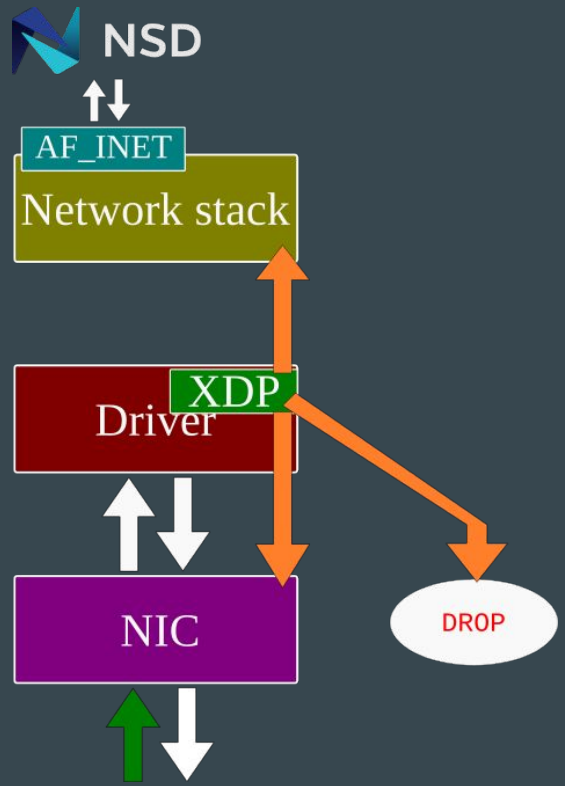
XDP_REDIRECTED: send out other NIC

A packet's destiny: XDP return codes



Using the special AF_XDP socket type one can reach the application while bypassing the entire network stack. (special case of XDP_REDIRECT)

Towards *augmenting* DNS software



<- This work is about:

adding functionality that is agnostic of DNS software running on the OS.

It's not about:

Adapting existing software to use AF_XDP sockets;
Implementing feature complete nameservers/resolvers in XDP

Workflow

- write C code: rrl.c
- compile: rrl.o (NB: successful compilation **does not** guarantee the next step!)
- load rrl.o, e.g. using iproute2:

```
# ip link set dev eno1 xdpgeneric obj rrl.o sec xdp
```

  - verifier checks this code: does it terminate? not too complex?
  - no objections? code is now active on the interface, on ingress, processing incoming packets before the OS network stack sees them

- any further interaction (if any) with the running code goes via *BPF maps*
- no modprobe, no reboot, no reconfiguration of userspace software

Response Rate Limiting

- Check whether incoming packet:
  - is Ethernet/IP/UDP with dst port 53, and,
  - contains a correctly formatted DNS query
    - (if not, XDP_PASS the packet upwards to the stack)
- Now we know we are dealing with a DNS query, we:
  - track the query rate for this src_addr (i.e. keeping state, using *maps*)
  - based on that rate, return:

    XDP_PASS (no rate limiting applied), or

    XDP_DROP (if we want to RRL this query)

Based on student project by Tom Carpay:
https://www.nlnetlabs.nl/downloads/publications/DNS-augmentation-with-eBPF.pdf

On the state of BPF Maps

```
  6 enum bpf_map_type {
  5     BPF_MAP_TYPE_UNSPEC,
  4     BPF_MAP_TYPE_HASH,
  3     BPF_MAP_TYPE_ARRAY,
  2     BPF_MAP_TYPE_PROG_ARRAY,
  1     BPF_MAP_TYPE_PERF_EVENT_ARRAY,
118     BPF_MAP_TYPE_PERCPU_HASH,
  1     BPF_MAP_TYPE_PERCPU_ARRAY,
  2     BPF_MAP_TYPE_STACK_TRACE,
  3     BPF_MAP_TYPE_CGROUP_ARRAY,
  4     BPF_MAP_TYPE_LRU_HASH,
  5     BPF_MAP_TYPE_LRU_PERCPU_HASH,
  6     BPF_MAP_TYPE_LPM_TRIE,
  7     BPF_MAP_TYPE_ARRAY_OF_MAPS,
  8     BPF_MAP_TYPE_HASH_OF_MAPS,
  9     BPF_MAP_TYPE_DEVMAP,
 10     BPF_MAP_TYPE_SOCKMAP,
 11     BPF_MAP_TYPE_CPUMAP,
 12     BPF_MAP_TYPE_XSKMAP,
 13     BPF_MAP_TYPE_SOCKHASH,
 14     BPF_MAP_TYPE_CGROUP_STORAGE,
 15     BPF_MAP_TYPE_REUSEPORT_SOCKARRAY,
 16     BPF_MAP_TYPE_PERCPU_CGROUP_STORAGE,
 17     BPF_MAP_TYPE_QUEUE,
 18     BPF_MAP_TYPE_STACK,
 19     BPF_MAP_TYPE_SK_STORAGE,
 20     BPF_MAP_TYPE_DEVMAP_HASH,
 21 };
```

/usr/include/linux/bpf.h

Datastructures *specific* to BPF, require specific functions to read/write at runtime, e.g.:

bpf_map_lookup_elem()
bpf_map_update_elem()
bpf_map_delete_elem()

NB: Hardware offloading might not support all of these map types

Maps: inter-packet state

Keeping state in-between packets using BPF maps:

- datastructure: hashmap
- key: IPv6/IPv4 src address (of incoming queries)
- value: our own struct `bucket`, enabling rate calculation

```
1  struct bucket {
2          uint64_t start_time;
3          uint64_t n_packets;
4  };
5
6  struct bpf_map_def SEC("maps") state_map = {
7          .type = BPF_MAP_TYPE_PERCPU_HASH,
8          .key_size = sizeof(uint32_t),
9          .value_size = sizeof(struct bucket),
10         .max_entries = 1000000
11 };
12
13 struct bpf_map_def SEC("maps") state_map_v6 = {
14         .type = BPF_MAP_TYPE_PERCPU_HASH,
15         .key_size = sizeof(struct in6_addr),
16         .value_size = sizeof(struct bucket),
17         .max_entries = 1000000
18 };
```

Maps: configuration from userspace

Operator request: *"RRL, but not for $very_important_prefix"*

```
1  struct bpf_map_def SEC("maps") exclude_v4_prefixes = {
2          .type = BPF_MAP_TYPE_LPM_TRIE,
3          .key_size = sizeof(struct bpf_lpm_trie_key) + sizeof(uint32_t),
4          .value_size = sizeof(uint64_t),
5          .max_entries = 10000
6  };
7
8  struct bpf_map_def SEC("maps") exclude_v6_prefixes = {
9          .type = BPF_MAP_TYPE_LPM_TRIE,
10         .key_size = sizeof(struct bpf_lpm_trie_key) + 8, // first 64 bits
11         .value_size = sizeof(uint64_t),
12         .max_entries = 10000
13 };
```

Run-time configuration from userspace using maps:

- datastructure: LPM trie
- key: IPv6/IPv4 src address (of incoming queries)
- value: hit counter
- read/write using `bpftool`, or, your own custom userspace tool.

Demo time 😬

- example of how to compile

- example of how to load it

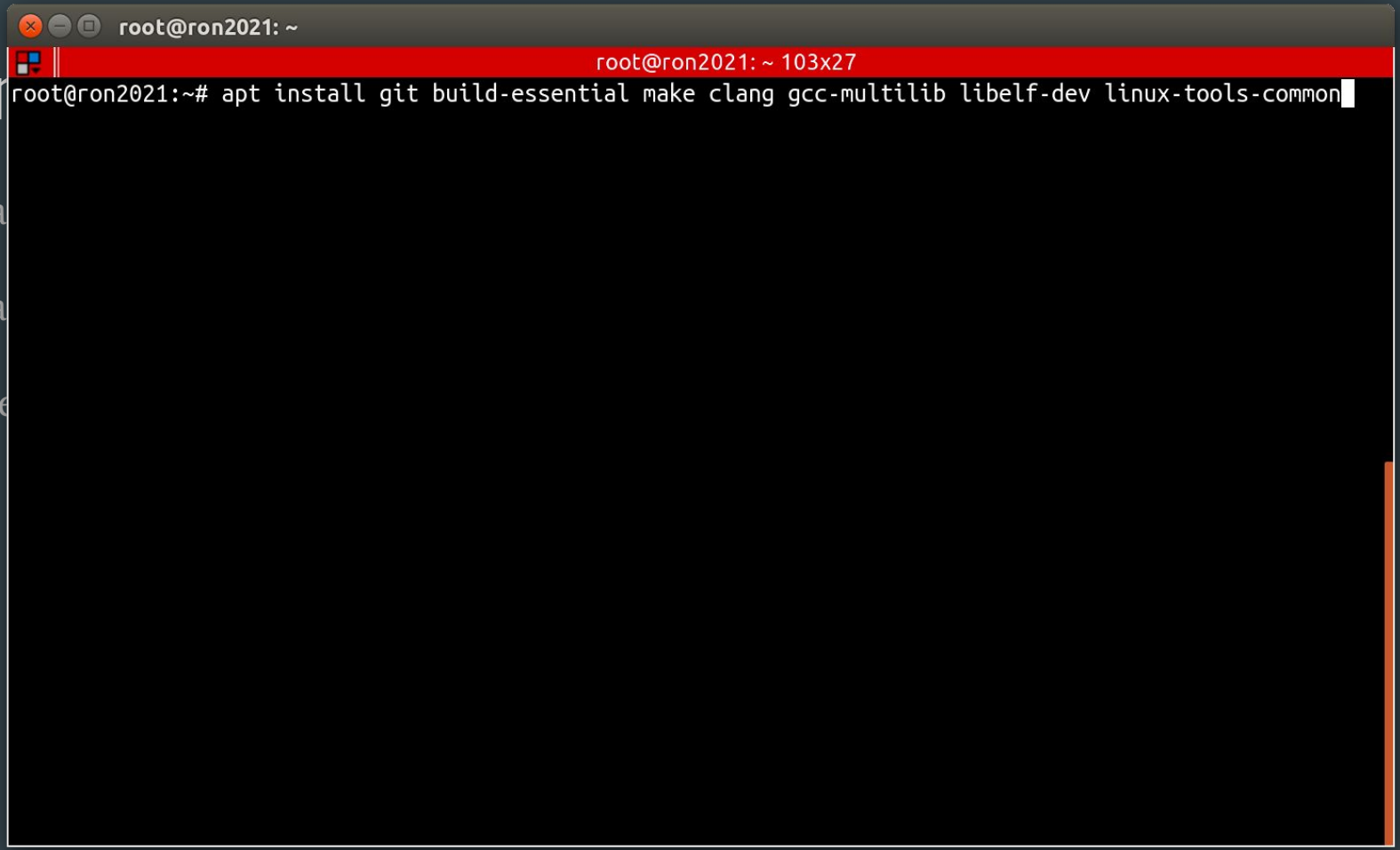
- screenshot of rrl.o in action

Der

- exa

- exa

- scre



Der

- exa
- exa
- scre

root@ron2021: ~/XDPeriments/libbpf/src

root@ron2021: ~/XDPeriments/libbpf/src 103x27

```
Reading state information... Done
build-essential is already the newest version (12.4ubuntu1).
make is already the newest version (4.1-9.1ubuntu1).
gcc-multilib is already the newest version (4:7.4.0-1ubuntu2.3).
git is already the newest version (1:2.17.1-1ubuntu0.7).
libelf-dev is already the newest version (0.170-0.4ubuntu0.1).
linux-tools-common is already the newest version (4.15.0-135.139).
clang is already the newest version (1:6.0-41~exp5~ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@ron2021:~#
root@ron2021:~# git clone https://github.com/NLnetLabs/XDPeriments.git
Cloning into 'XDPeriments'...
remote: Enumerating objects: 107, done.
remote: Counting objects: 100% (107/107), done.
remote: Compressing objects: 100% (71/71), done.
remote: Total 107 (delta 47), reused 87 (delta 33), pack-reused 0
Receiving objects: 100% (107/107), 32.80 KiB | 1.49 MiB/s, done.
Resolving deltas: 100% (47/47), done.
root@ron2021:~#
```

Der

- exa
- exa
- scre

root@ron2021: ~/XDPeriments/libbpf/src

root@ron2021: ~/XDPeriments/libbpf/src 103x27

```
Reading state information... Done
build-essential is already the newest version (12.4ubuntu1).
make is already the newest version (4.1-9.1ubuntu1).
gcc-multilib is already the newest version (4:7.4.0-1ubuntu2.3).
git is already the newest version (1:2.17.1-1ubuntu0.7).
libelf-dev is already the newest version (0.170-0.4ubuntu0.1).
linux-tools-common is already the newest version (4.15.0-135.139).
clang is already the newest version (1:6.0-41~exp5~ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@ron2021:~#
root@ron2021:~# git clone https://github.com/NLnetLabs/XDPeriments.git
Cloning into 'XDPeriments'...
remote: Enumerating objects: 107, done.
remote: Counting objects: 100% (107/107), done.
remote: Compressing objects: 100% (71/71), done.
remote: Total 107 (delta 47), reused 87 (delta 33), pack-reused 0
Receiving objects: 100% (107/107), 32.80 KiB | 1.49 MiB/s, done.
Resolving deltas: 100% (47/47), done.
root@ron2021:~#
root@ron2021:~# cd XDPeriments
root@ron2021:~/XDPeriments# git submodule update --init
Submodule 'libbpf' (https://github.com/libbpf/libbpf) registered for path 'libbpf'
Cloning into '/root/XDPeriments/libbpf'...
Submodule path 'libbpf': checked out '1b42b15b5e6dec568e8826ed908a5acedd32317c'
root@ron2021:~/XDPeriments#
```

Der

- exa
- exa
- scre

root@ron2021: ~/XDPeriments/libbpf/src

root@ron2021: ~/XDPeriments/libbpf/src 103x27

```
Reading state information... Done
build-essential is already the newest version (12.4ubuntu1).
make is already the newest version (4.1-9.1ubuntu1).
gcc-multilib is already the newest version (4:7.4.0-1ubuntu2.3).
git is already the newest version (1:2.17.1-1ubuntu0.7).
libelf-dev is already the newest version (0.170-0.4ubuntu0.1).
linux-tools-common is already the newest version (4.15.0-135.139).
clang is already the newest version (1:6.0-41~exp5~ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@ron2021:~#
root@ron2021:~# git clone https://github.com/NLnetLabs/XDPeriments.git
Cloning into 'XDPeriments'...
remote: Enumerating objects: 107, done.
remote: Counting objects: 100% (107/107), done.
remote: Compressing objects: 100% (71/71), done.
remote: Total 107 (delta 47), reused 87 (delta 33), pack-reused 0
Receiving objects: 100% (107/107), 32.80 KiB | 1.49 MiB/s, done.
Resolving deltas: 100% (47/47), done.
root@ron2021:~#
root@ron2021:~# cd XDPeriments
root@ron2021:~/XDPeriments# git submodule update --init
Submodule 'libbpf' (https://github.com/libbpf/libbpf) registered for path 'libbpf'
Cloning into '/root/XDPeriments/libbpf'...
Submodule path 'libbpf': checked out '1b42b15b5e6dec568e8826ed908a5acedd32317c'
root@ron2021:~/XDPeriments#
root@ron2021:~/XDPeriments# cd libbpf/src/
root@ron2021:~/XDPeriments/libbpf/src# make
```

Der

- exa

- exa

- scre

root@ron2021: ~/XDPeriments/RRL/Round3

root@ron2021: ~/XDPeriments/RRL/Round3 103x27

```
sed -e "s|@PREFIX@|/usr|" \
        -e "s|@LIBDIR@|/usr/lib64|" \
        -e "s|@VERSION@|0.1.0|" \
        < libbpf.pc.template > libbpf.pc
root@ron2021:~/XDPeriments/libbpf/src#
root@ron2021:~/XDPeriments/libbpf/src# cd ../../RRL/Round3
root@ron2021:~/XDPeriments/RRL/Round3# make
clang -target bpf -O2 -Wall -Werror -I ../../libbpf/src -c -o xdp_rrl.o xdp_rrl.c
clang -static -O2 -Wall -Werror -I ../../libbpf/src -o xdp_rrl_vipctl xdp_rrl_vipctl.c -L../../libbpf/s
rc -lbpf -lelf -lz
```

Der

- exa
- exa
- scre

root@ron2021: ~/XDPeriments/RRL/Round3

root@ron2021: ~/XDPeriments/RRL/Round3 103x27

```
sed -e "s|@PREFIX@|/usr|" \
        -e "s|@LIBDIR@|/usr/lib64|" \
        -e "s|@VERSION@|0.1.0|" \
        < libbpf.pc.template > libbpf.pc
root@ron2021:~/XDPeriments/libbpf/src#
root@ron2021:~/XDPeriments/libbpf/src# cd ../../RRL/Round3
root@ron2021:~/XDPeriments/RRL/Round3# make
clang -target bpf -O2 -Wall -Werror -I ../../libbpf/src -c -o xdp_rrl.o xdp_rrl.c
clang -static -O2 -Wall -Werror -I ../../libbpf/src -o xdp_rrl_vipctl xdp_rrl_vipctl.c -L../../libbpf/s
rc -lbpf -lelf -lz
root@ron2021:~/XDPeriments/RRL/Round3#
root@ron2021:~/XDPeriments/RRL/Round3# make vip_maps
sudo mount -t bpf none /sys/fs/bpf
sudo bpftool map create /sys/fs/bpf/rrl_exclude_v4_prefixes flags 1 \
             name exclude_v4_prefixes type lpm_trie key 8 value 8 entries 10000
sudo bpftool map create /sys/fs/bpf/rrl_exclude_v6_prefixes flags 1 \
             name exclude_v6_prefixes type lpm_trie key 12 value 8 entries 10000
```

Der

- exa
- exa
- scre

root@ron2021: ~/XDPeriments/RRL/Round3

root@ron2021: ~/XDPeriments/RRL/Round3 103x27

```
sed -e "s|@PREFIX@|/usr|" \
        -e "s|@LIBDIR@|/usr/lib64|" \
        -e "s|@VERSION@|0.1.0|" \
        < libbpf.pc.template > libbpf.pc
root@ron2021:~/XDPeriments/libbpf/src#
root@ron2021:~/XDPeriments/libbpf/src# cd ../../RRL/Round3
root@ron2021:~/XDPeriments/RRL/Round3# make
clang -target bpf -O2 -Wall -Werror -I ../../libbpf/src -c -o xdp_rrl.o xdp_rrl.c
clang -static -O2 -Wall -Werror -I ../../libbpf/src -o xdp_rrl_vipctl xdp_rrl_vipctl.c -L../../libbpf/s
rc -lbpf -lelf -lz
root@ron2021:~/XDPeriments/RRL/Round3#
root@ron2021:~/XDPeriments/RRL/Round3# make vip_maps
sudo mount -t bpf none /sys/fs/bpf
sudo bpftool map create /sys/fs/bpf/rrl_exclude_v4_prefixes flags 1 \
             name exclude_v4_prefixes type lpm_trie key 8 value 8 entries 10000
sudo bpftool map create /sys/fs/bpf/rrl_exclude_v6_prefixes flags 1 \
             name exclude_v6_prefixes type lpm_trie key 12 value 8 entries 10000
root@ron2021:~/XDPeriments/RRL/Round3#
root@ron2021:~/XDPeriments/RRL/Round3# make load
sudo bpftool prog load xdp_rrl.o /sys/fs/bpf/rrl type xdp \
        map name exclude_v4_prefixes \
        pinned /sys/fs/bpf/rrl_exclude_v4_prefixes \
        map name exclude_v6_prefixes \
        pinned /sys/fs/bpf/rrl_exclude_v6_prefixes
sudo ip --force link set dev eth0 xdpgeneric \
        pinned /sys/fs/bpf/rrl
root@ron2021:~/XDPeriments/RRL/Round3#
```

Der

- exa

- exa

- scr

root@ron2021: ~/XDPeriments/RRL/Round3

root@ron2021: ~/XDPeriments/RRL/Round3 103x27

```
sudo bpftool map create /sys/fs/bpf/rrl_exclude_v4_prefixes flags 1 \
            name exclude_v4_prefixes type lpm_trie key 8 value 8 entries 10000
sudo bpftool map create /sys/fs/bpf/rrl_exclude_v6_prefixes flags 1 \
            name exclude_v6_prefixes type lpm_trie key 12 value 8 entries 10000
root@ron2021:~/XDPeriments/RRL/Round3#
root@ron2021:~/XDPeriments/RRL/Round3# make load
sudo bpftool prog load xdp_rrl.o /sys/fs/bpf/rrl type xdp \
        map name exclude_v4_prefixes \
        pinned /sys/fs/bpf/rrl_exclude_v4_prefixes \
        map name exclude_v6_prefixes \
        pinned /sys/fs/bpf/rrl_exclude_v6_prefixes
sudo ip --force link set dev eth0 xdpgeneric \
        pinned /sys/fs/bpf/rrl
root@ron2021:~/XDPeriments/RRL/Round3#
root@ron2021:~/XDPeriments/RRL/Round3# bpftool map  | tail -8
20: lpm_trie  name exclude_v4_pref  flags 0x1
        key 8B  value 8B  max_entries 10000  memlock 524288B
21: lpm_trie  name exclude_v6_pref  flags 0x1
        key 12B  value 8B  max_entries 10000  memlock 561152B
23: percpu_hash  name state_map  flags 0x0
        key 4B  value 16B  max_entries 1000000  memlock 320778240B
24: percpu_hash  name state_map_v6  flags 0x0
        key 16B  value 16B  max_entries 1000000  memlock 328777728B
root@ron2021:~/XDPeriments/RRL/Round3#
root@ron2021:~/XDPeriments/RRL/Round3# bpftool map dump id 24
Found 0 elements
root@ron2021:~/XDPeriments/RRL/Round3#
```

Der

- exa

- exa

- scre

root@ron2021: ~/XDPeriments/RRL/Round3

root@ron2021: ~/XDPeriments/RRL/Round3 103x20

```
root@ron2021:~/XDPeriments/RRL/Round3# bpftool map dump id 23
key:
2d 5f 40 00
value (CPU 00): 40 e0 5d 75 81 03 00 00  01 00 00 00 00 00 00 00
value (CPU 01): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 02): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 03): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 04): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 05): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 06): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 07): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 08): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 09): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 10): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 11): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 12): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 13): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
value (CPU 14): 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
Found 1 element
root@ron2021:~/XDPeriments/RRL/Round3#
```

willem@makaak: ~ 103x5

```
willem@makaak:~$ dig -4 @ron2021.nlnetlabs.nl nlnetlabs.nl A +short
185.49.140.10
willem@makaak:~$
```

Der

- exa
- exa
- scre

root@ron2021: ~/XDPeriments/RRL/Round3

root@ron2021: ~/XDPeriments/RRL/Round3 103x27

```
/*
 * DNS Response Rate Limiting module in XDP.
 *
 * October 2020 - Tom Carpay & Willem Toorop
 */

#define RRL_N_CPUS            2
/* This should be the number of CPUs on your system. Get it by running:
 *
 *      echo "CPUs: $(grep -c processor /proc/cpuinfo)"
 */

#define RRL_SIZE        1000000
/* This option gives the size of the hashtable. More buckets
 * use more memory, and reduce the chance of hash collisions.
 */

#define RRL_RATELIMIT       200
/* The max qps allowed (from one query source). If set to 0 then it is disabled
 * (unlimited rate). Once the rate limit is reached, responses will be dropped.
 * However, one in every RRL_SLIP number of responses is allowed, with the TC
 * bit set. If slip is set to 2, the outgoing response rate will be halved. If
 * it's set to 3, the outgoing response rate will be one-third, and so on.  If
 * you set RRL_SLIP to 10, traffic is reduced to 1/10th.
 */

"xdp_rrl.c" 625L, 18102C                                      1,1           Top
```

Der

- exa
- exa
- scre

root@ron2021: ~/XDPeriments/RRL/Round3

root@ron2021: ~/XDPeriments/RRL/Round3 103x27

```
#define RRL_RATELIMIT       200
/* The max qps allowed (from one query source). If set to 0 then it is disabled
 * (unlimited rate). Once the rate limit is reached, responses will be dropped.
 * However, one in every RRL_SLIP number of responses is allowed, with the TC
 * bit set. If slip is set to 2, the outgoing response rate will be halved. If
 * it's set to 3, the outgoing response rate will be one-third, and so on.  If
 * you set RRL_SLIP to 10, traffic is reduced to 1/10th.
 */

#define RRL_SLIP             2
/* This option controls the number of packets discarded before we send back a
 * SLIP response (a response with "truncated" bit set to one). 0 disables the
 * sending of SLIP packets, 1 means every query will get a SLIP response.
 * Default is 2, cuts traffic in half and legit users have a fair chance to get
 * a +TC response.
 */

#define RRL_IPv4_PREFIX_LEN  24
/* IPv4 prefix length. Addresses are grouped by netblock.
 */

#define RRL_IPv6_PREFIX_LEN  48
/* IPv6 prefix length. Addresses are grouped by netblock.
 */
                                                              26,0-1          2%
```

Der

- exa
- exa
- scre

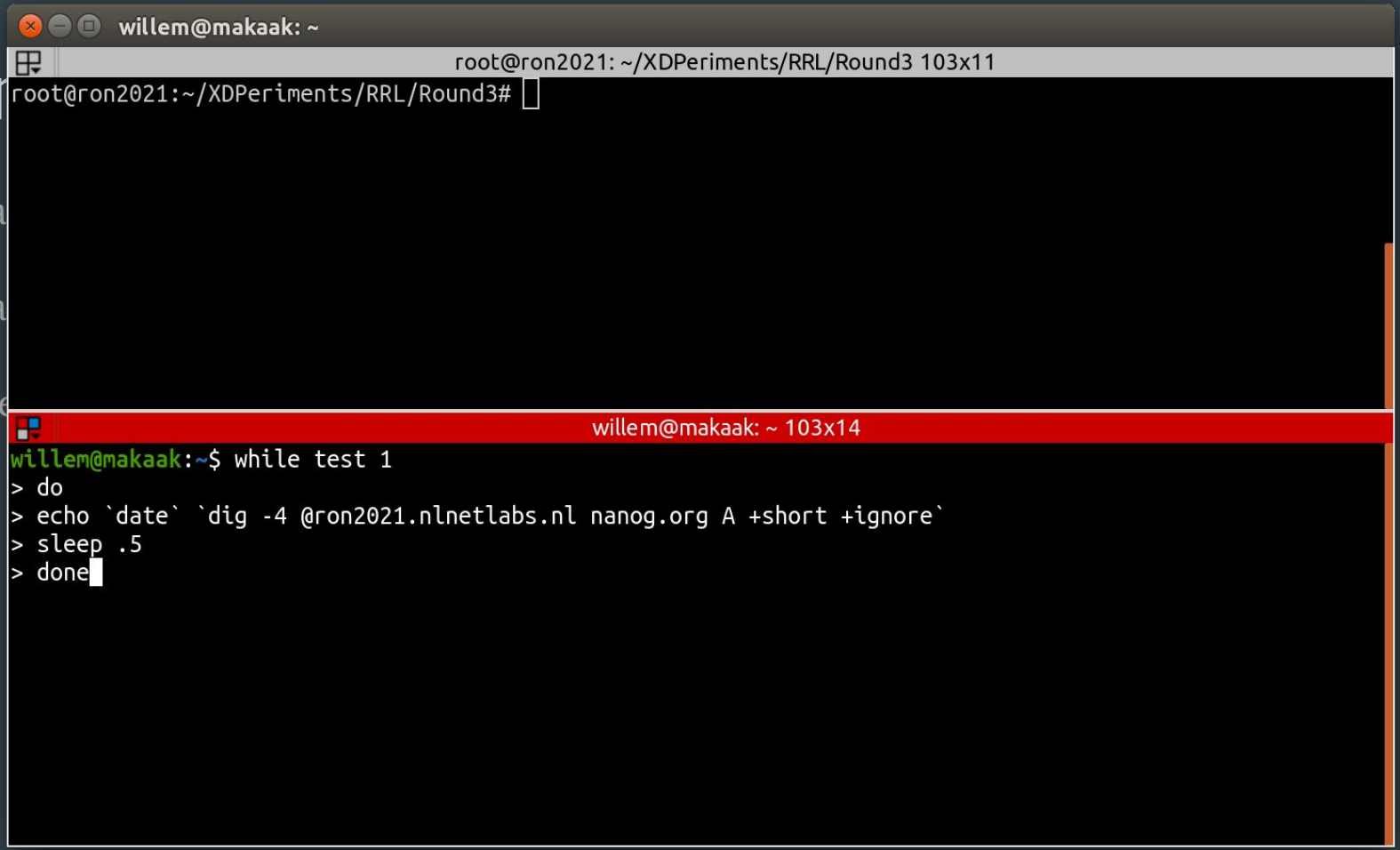
root@ron2021: ~/XDPeriments/RRL/Round3

root@ron2021: ~/XDPeriments/RRL/Round3 103x27

```
#define RRL_SIZE        1000000
/* This option gives the size of the hashtable. More buckets
 * use more memory, and reduce the chance of hash collisions.
 */

#define RRL_RATELIMIT        5
/* The max qps allowed (from one query source). If set to 0 then it is disabled
 * (unlimited rate). Once the rate limit is reached, responses will be dropped.
 * However, one in every RRL_SLIP number of responses is allowed, with the TC
 * bit set. If slip is set to 2, the outgoing response rate will be halved. If
 * it's set to 3, the outgoing response rate will be one-third, and so on.  If
 * you set RRL_SLIP to 10, traffic is reduced to 1/10th.
 */

#define RRL_SLIP             1
/* This option controls the number of packets discarded before we send back a
 * SLIP response (a response with "truncated" bit set to one). 0 disables the
 * sending of SLIP packets, 1 means every query will get a SLIP response.
 * Default is 2, cuts traffic in half and legit users have a fair chance to get
 * a +TC response.
 */

#define RRL_IPv4_PREFIX_LEN  24
/* IPv4 prefix length. Addresses are grouped by netblock.
 */
                                                          26,0-1          1%
```

Der

- exa
- exa
- scre

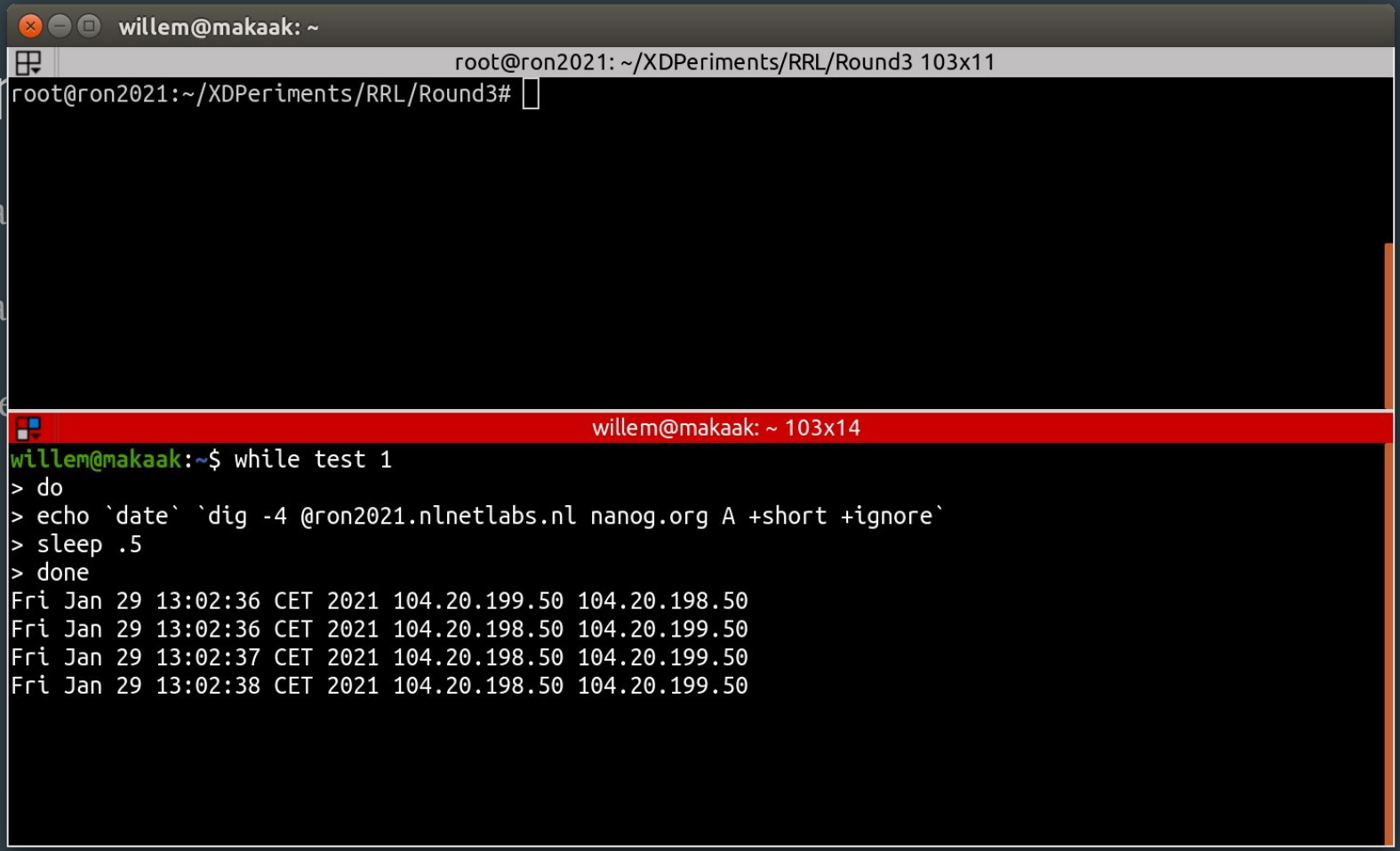


Der

- exa

- exa

- scre

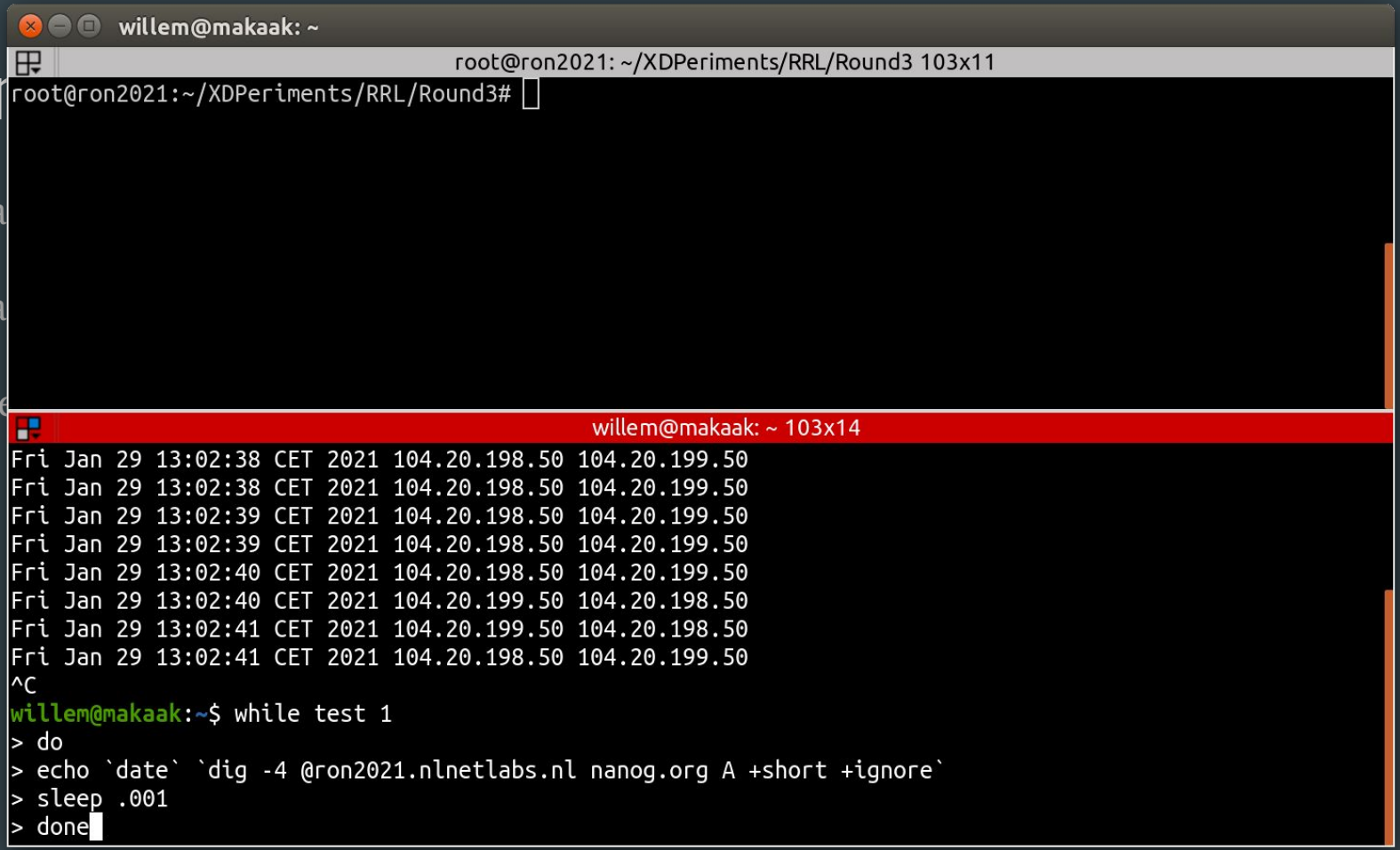


Der

- exa

- exa

- scre

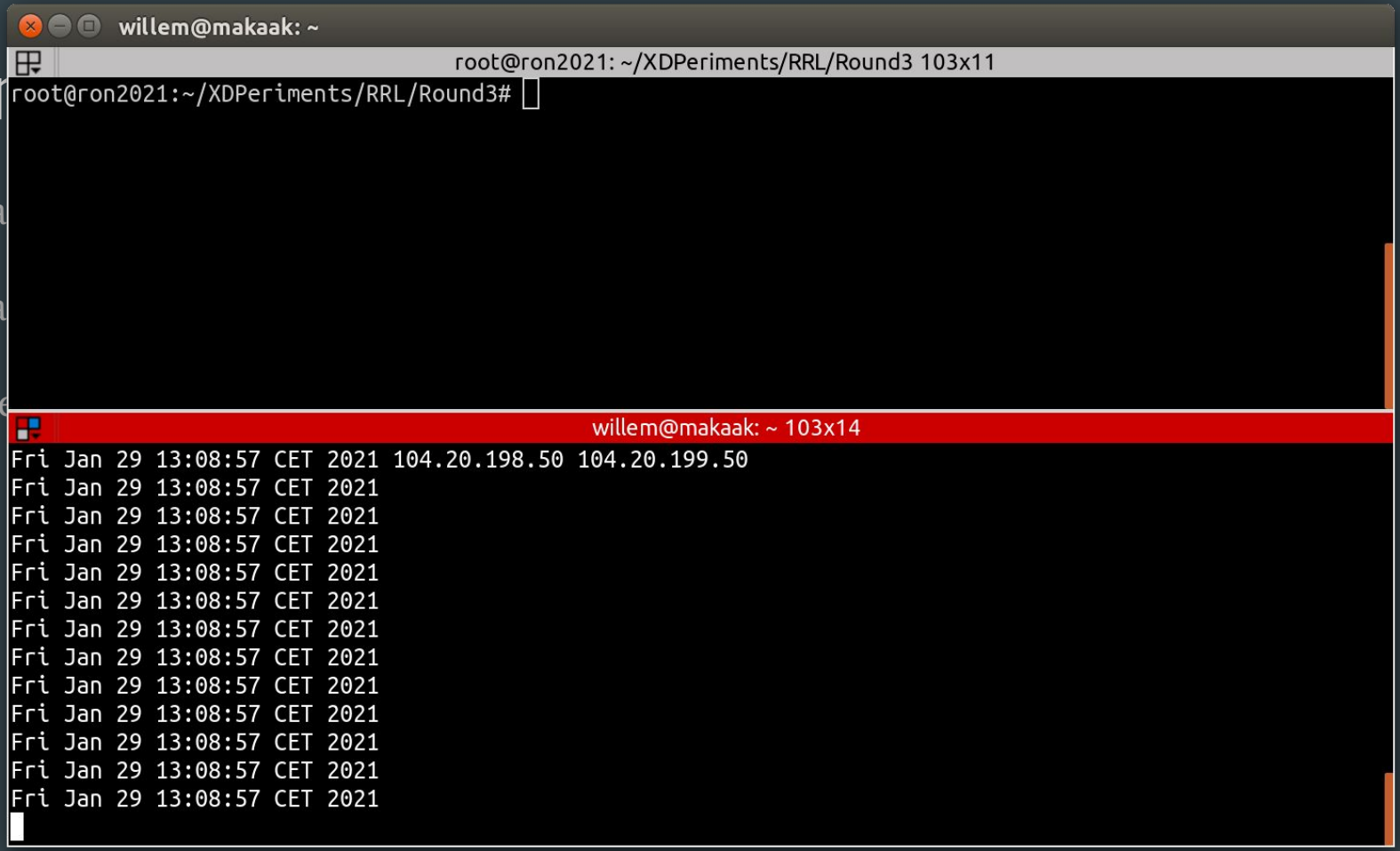


Der

- exa

- exa

- scre

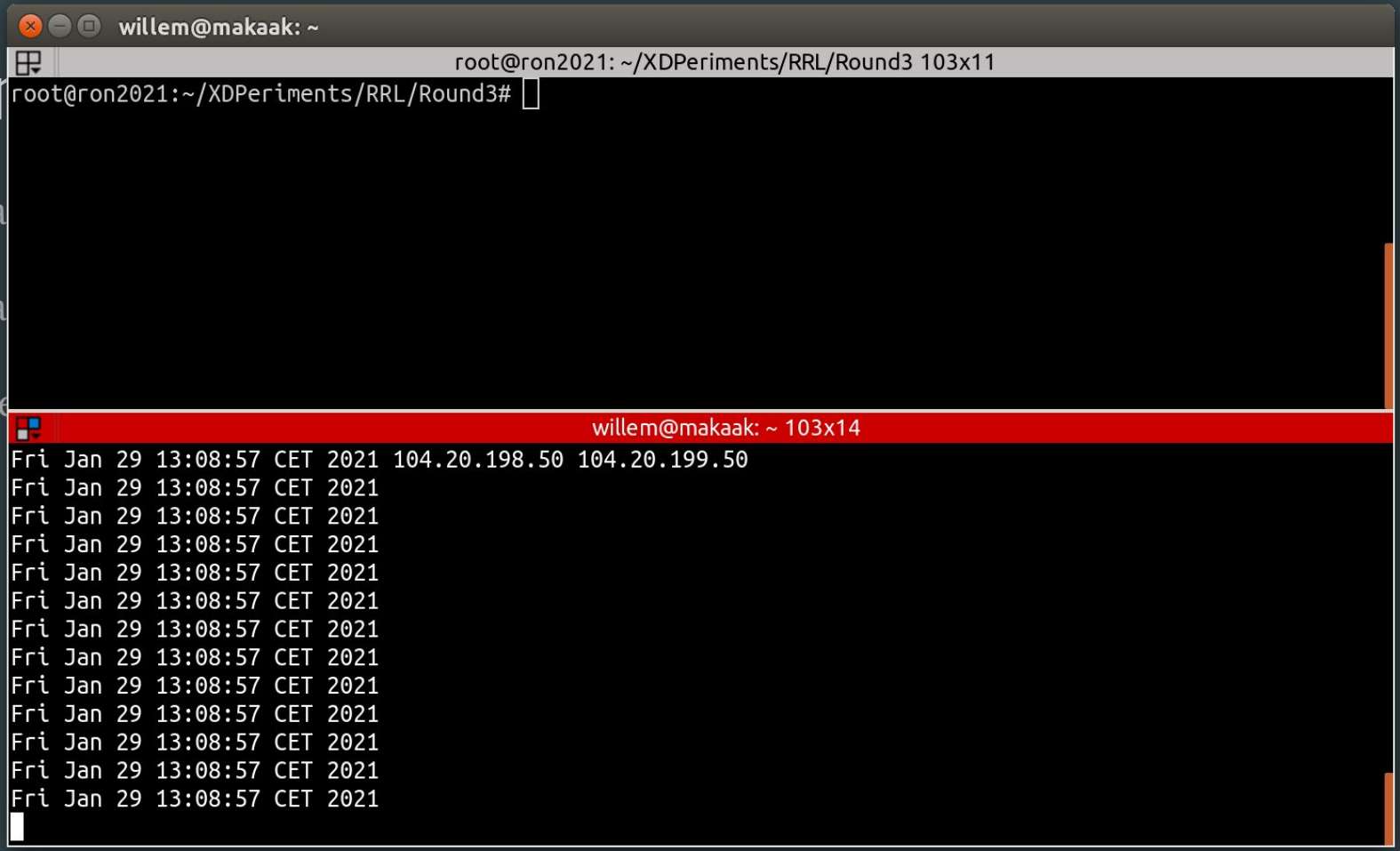


Der

- exa

- exa

- scre



Der

- exa

- exa

- scre

root@ron2021: ~/XDPeriments/RRL/Round3

root@ron2021: ~/XDPeriments/RRL/Round3 103x11

```
root@ron2021:~/XDPeriments/RRL/Round3# ./xdp_rrl_vipctl add 185.49.140.0/22
root@ron2021:~/XDPeriments/RRL/Round3#
```

willem@makaak: ~ 103x14

```
Fri Jan 29 13:12:01 CET 2021 104.20.199.50 104.20.198.50
Fri Jan 29 13:12:01 CET 2021 104.20.199.50 104.20.198.50
Fri Jan 29 13:12:01 CET 2021 104.20.198.50 104.20.199.50
Fri Jan 29 13:12:01 CET 2021 104.20.199.50 104.20.198.50
Fri Jan 29 13:12:02 CET 2021 104.20.198.50 104.20.199.50
Fri Jan 29 13:12:02 CET 2021 104.20.199.50 104.20.198.50
Fri Jan 29 13:12:02 CET 2021 104.20.199.50 104.20.198.50
Fri Jan 29 13:12:02 CET 2021 104.20.199.50 104.20.198.50
Fri Jan 29 13:12:02 CET 2021 104.20.199.50 104.20.198.50
Fri Jan 29 13:12:02 CET 2021 104.20.199.50 104.20.198.50
Fri Jan 29 13:12:02 CET 2021 104.20.198.50 104.20.199.50
Fri Jan 29 13:12:02 CET 2021 104.20.198.50 104.20.199.50
Fri Jan 29 13:12:02 CET 2021 104.20.199.50 104.20.198.50
```

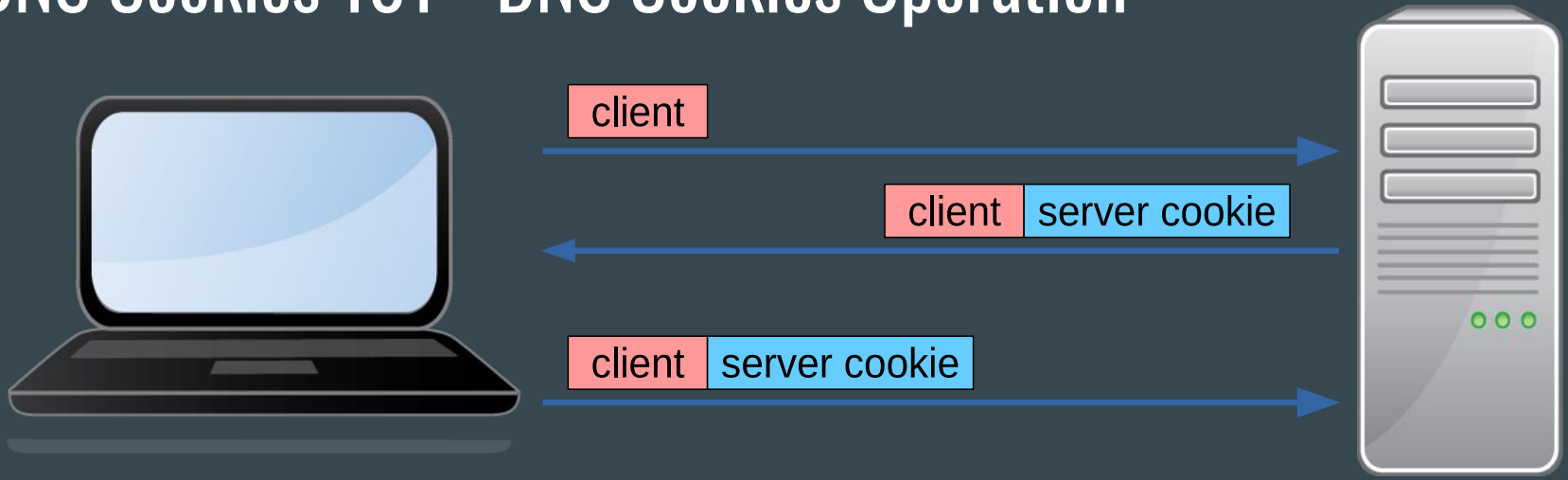
Response Rate Limiting - lessons learned

We can leverage XDP to *augment* DNS services:
handle the packet in XDP, or,
decide to point it upwards to a userspace nameserver

Maps enable keeping state,
not only for e.g. statistics and rates calculations,
but moreover for configuration from userspace at runtime

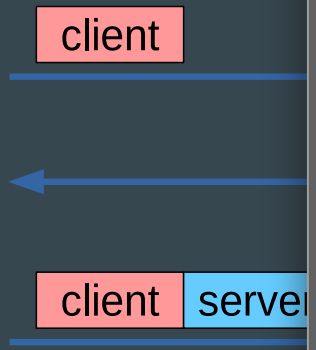
When choosing a BPF map type, consider concurrency (PERCPU or not)
and possible performance hits

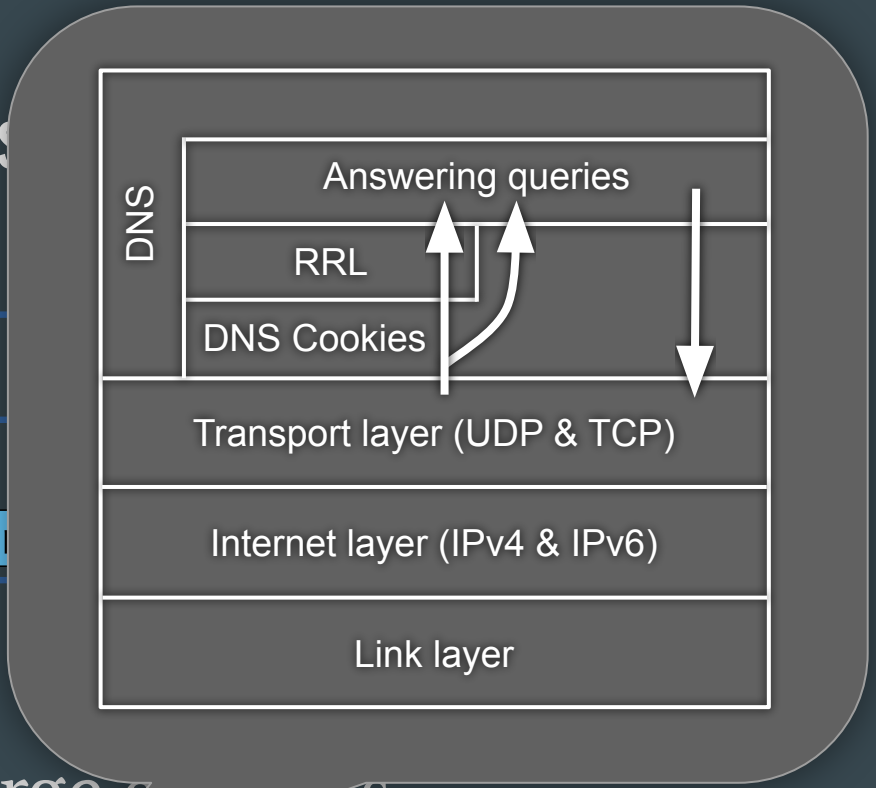
DNS Cookies 101 - DNS Cookies Operation



- Valid Server Cookie? Large answers
- Valid Server Cookie? RRL disabled

DNS Cookies 101 - DNS Cookies

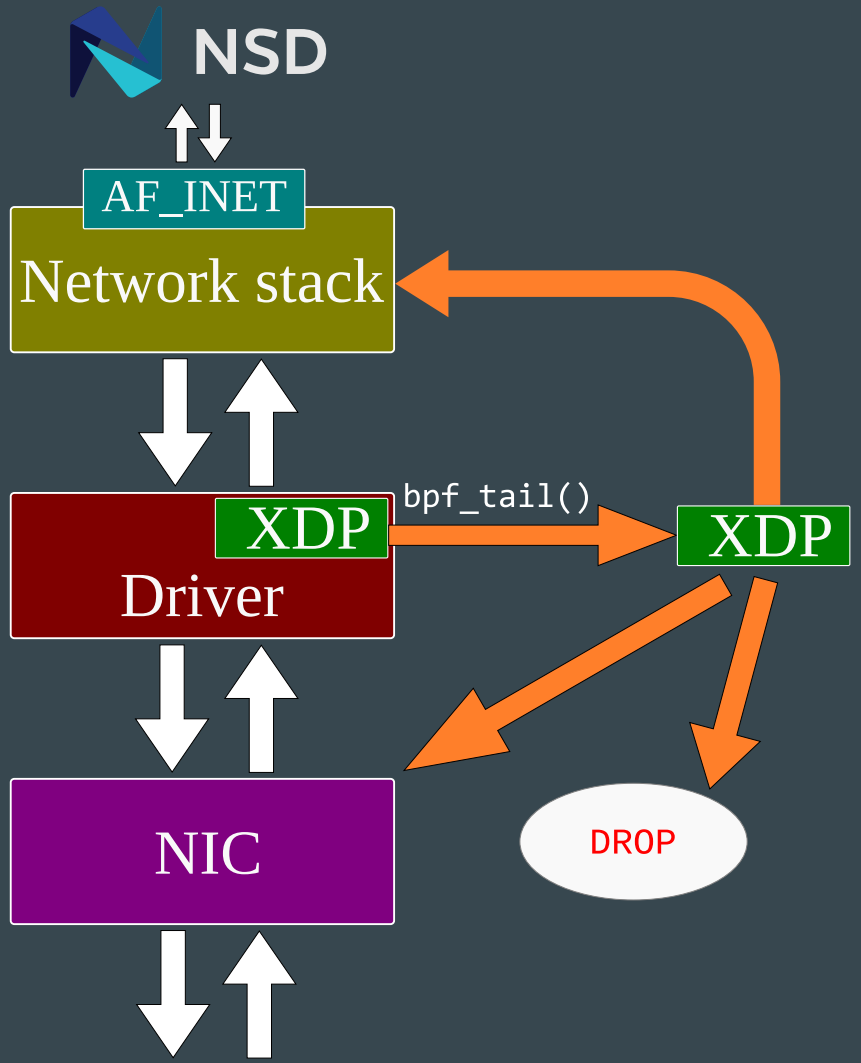




- Valid Server Cookie? Large answers
- Valid Server Cookie? RRL disabled

DNS Cookies - Pass info with meta data

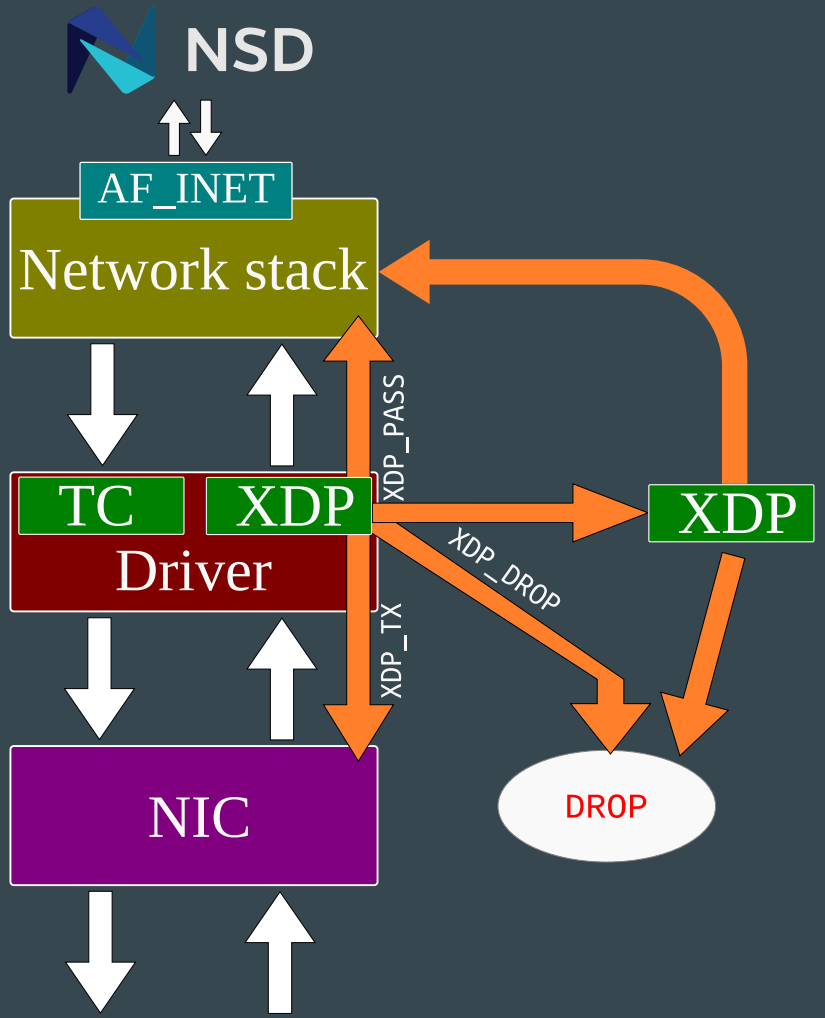
- `bpf_tail_call()` is like `goto`



DNS Cookies
- Also Creating Cookies ... ongoing

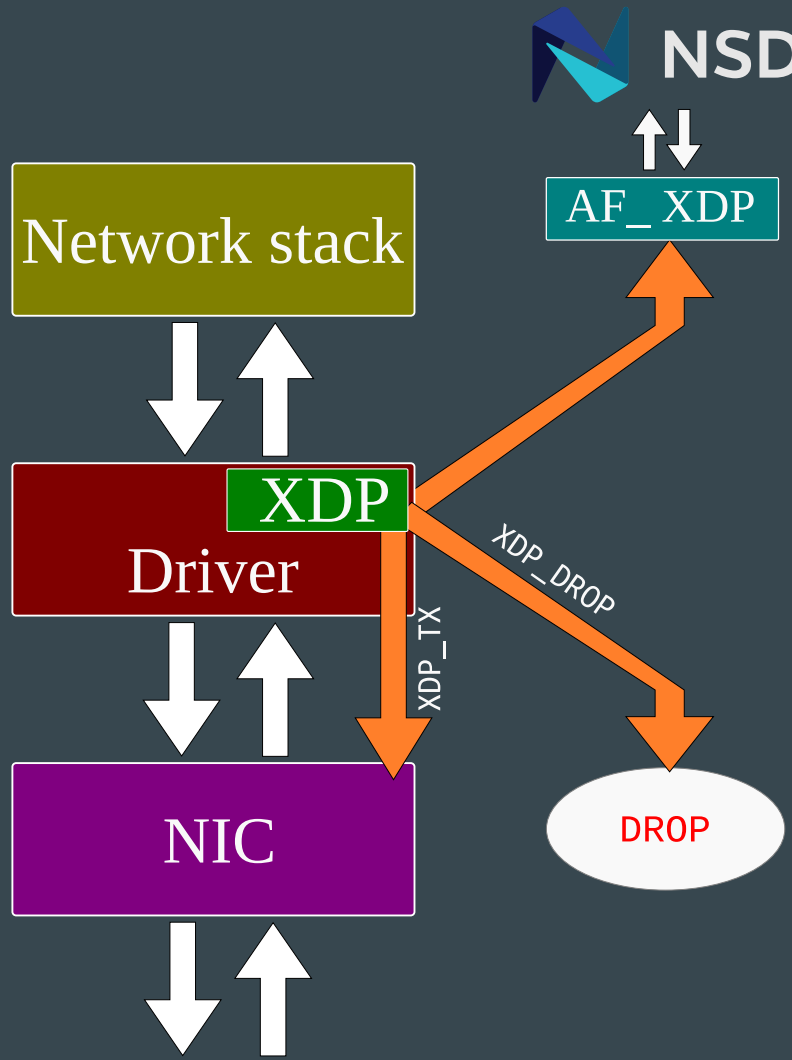
- Outgoing eBPF on Traffic Control (TC) layer
- Edit Socket Buffer instead of packet
- Can grow with:
  - `bpf_skb_change_tail()`
- Checksum recalculations with:
  - `bpf_skb_store_bytes()`
- Connect in with out with:
  - `BPF_MAP_TYPE_LRU_HASH`
- Outgoing less performant, but...

... Augmenting ... Interoperable



Looking ahead

- **AF_XDP support for NSD**
  Adapt NSD to use the AF_XDP socket type provided by BPF/XDP
- **Hot self-managing cache**
  Write outgoing answers in a LRU hashmap, answer queries directly from XDP
- **Zone sharding / load balancing**
  Load balance based on the qname, so that nameservers only have to load part of (big) zones.
- **root zone from XDP?**



XDPeriments: Tinkering with DNS and XDP

{willem,luuk}@nlnetlabs.nl
https://github.com/NLnetLabs/XDPeriments
https://blog.nlnetlabs.nl/tag/research/

Many thanks to
Ronald van der Pol
at
SURF