Linux Kernel - BPF / XDP

KossLab 유태희, 송태웅

BPF 란 ?

1. Berkeley Packet Filter since 1992
2. Kernel Infrastructure

BPF 란 ?

1. Berkeley Packet Filter since 1992
2. **Kernel Infrastructure**
   - **Interpreter** in-kernel virtual machine
   - **Hook points** in-kernel callback point
   - **Map**
   - **Helper**

BPF 란 ?

“Safe dynamic programs and tools”

"런타임중 안전하게 커널코드를 삽입하는 기술"

BPF Infrastructure:
안전한 code injection 작전

1) Native 머신코드 대신 **BPF instruction** 을 활용하자

2) **Verifier** 를 통해 위험요소를 미리검사하자

3) (기존)커널함수가 필요할때 **Helper** 함수를 통해서만 호출하자

BPF Infrastructure:
안전한 code injection 작전

1) Native 머신코드 대신 **BPF instruction** 을 활용하자

BPF Infrastructure:
안전한 code injection 작전

2) Verifier 를 통해 위험요소를 미리검사하자

BPF Infrastructure:
안전한 code injection 작전

3) (기존)커널함수가 필요할때 Helper 함수를 통해서만 호출하자

BPF Infrastructure:
안전한 code injection 위한 기반기술

Kernel += BPF Interpreter in-kernel virtual machine

\+ Verifier

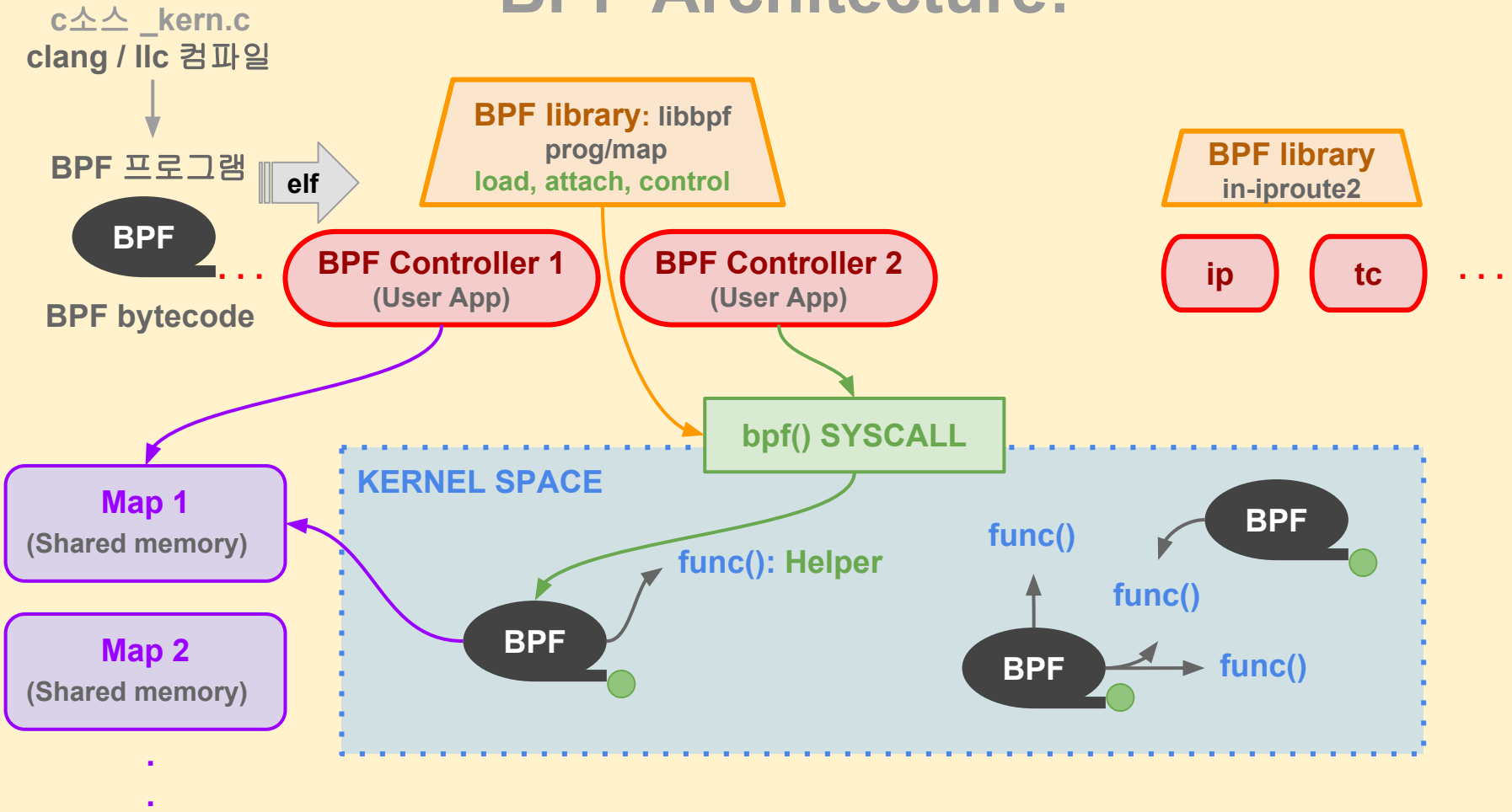
\+ BPF Helper 함수 추가 leveraging kernel func

\+ BPF syscall prog/map: loading & attaching 등

BPF Helper 함수:

```
$ grep BPF_CALL
kernel/bpf/helpers.c:
BPF_CALL_2(bpf_map_lookup_elem, struct bpf_map *, map, void *, key)
BPF_CALL_4(bpf_map_update_elem, struct bpf_map *, map, void *, key,
[...]
kernel/trace/bpf_trace.c:
BPF_CALL_2(bpf_override_return, struct pt_regs *, regs, unsigned long, rc)
BPF_CALL_3(bpf_probe_read, void *, dst, u32, size, const void *, unsafe_ptr)
BPF_CALL_3(bpf_probe_write_user, void *, unsafe_ptr, const void *, src,
BPF_CALL_5(bpf_trace_printk, char *, fmt, u32, fmt_size, u64, arg1,
[...]
net/core/filter.c:
BPF_CALL_1(bpf_skb_get_pay_offset, struct sk_buff *, skb)
BPF_CALL_3(bpf_skb_get_nlattr, struct sk_buff *, skb, u32, a, u32, x)
[...]
```

BPF Architecture:

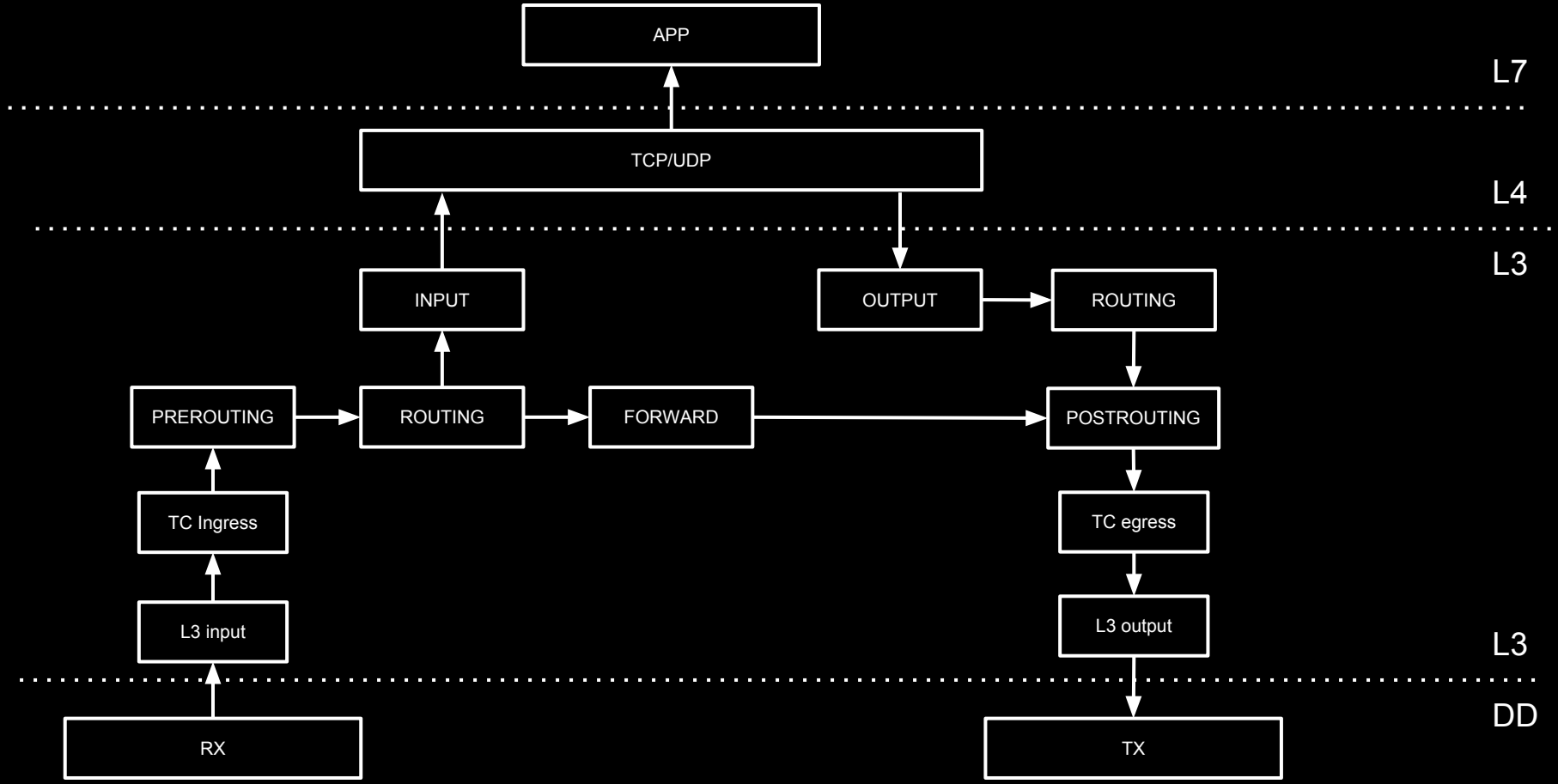


XDP

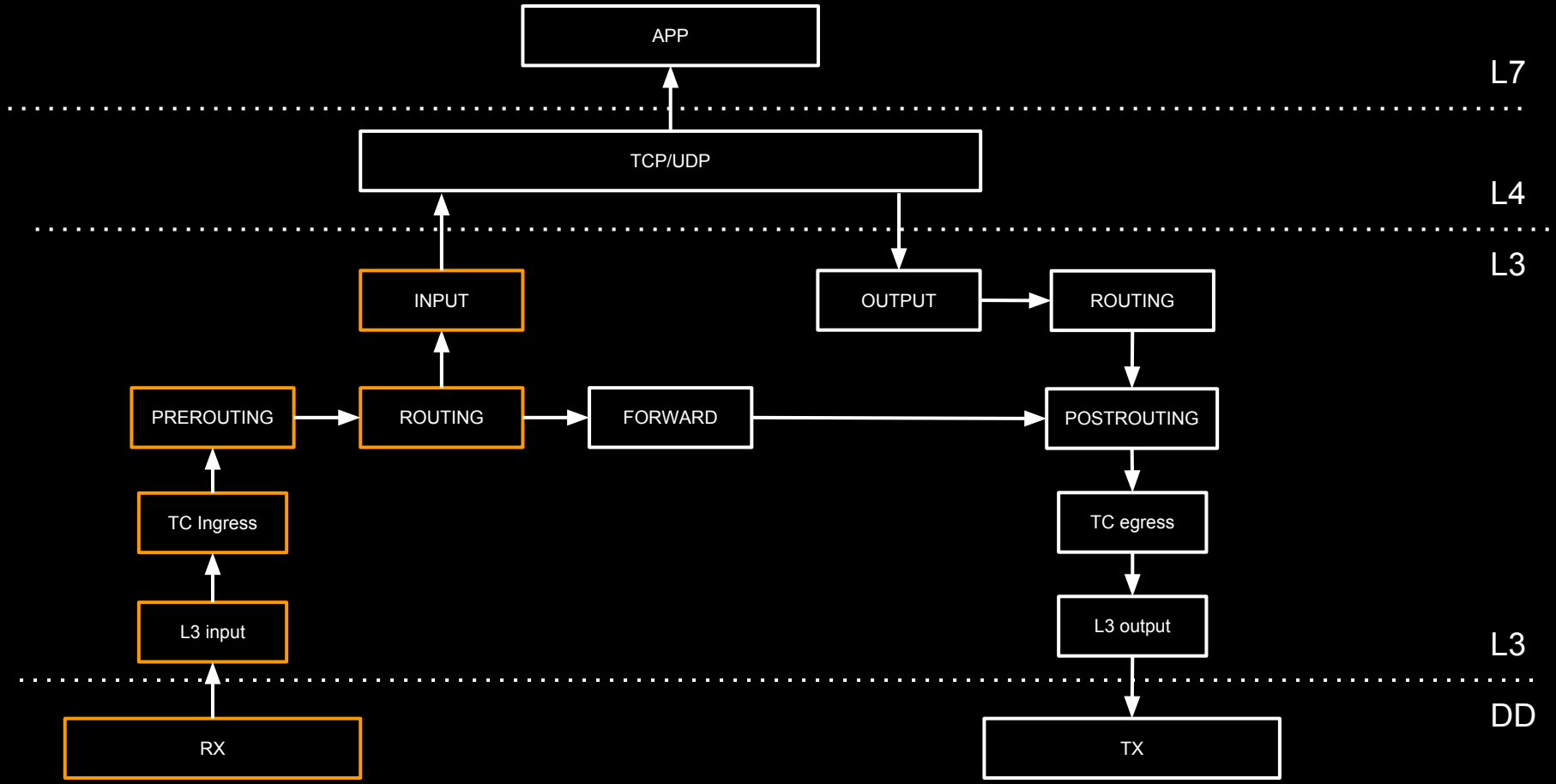
(eXpress Data Path)

Why XDP?

NORMAL PATH

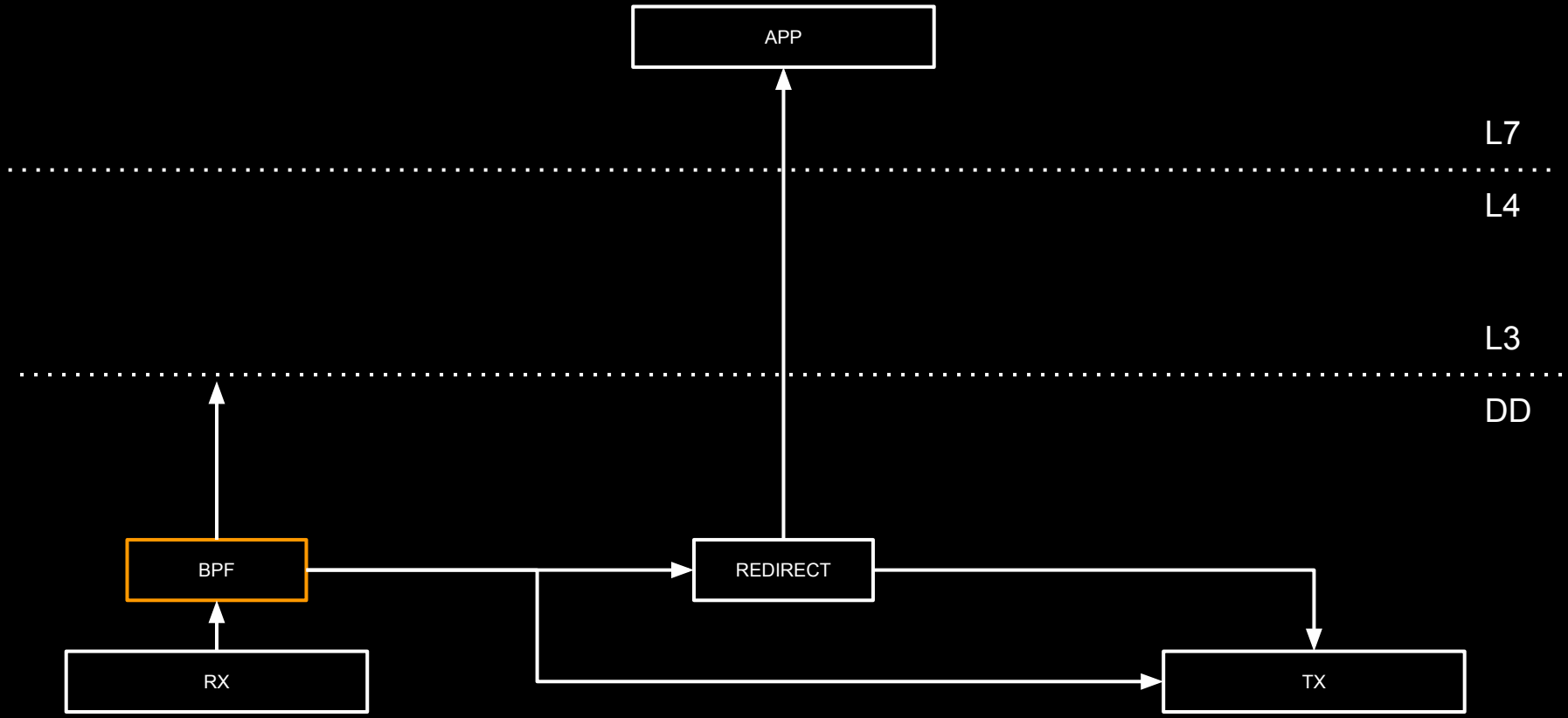

APP
L7
TCP/UDP
L4
L3
INPUT
OUTPUT
ROUTING
PREROUTING
ROUTING
FORWARD
POSTROUTING
TC Ingress
TC egress
L3 input
L3 output
L3
DD
RX
TX

NORMAL PATH


APP
L7
TCP/UDP
L4
L3
INPUT
OUTPUT
ROUTING
PREROUTING
ROUTING
FORWARD
POSTROUTING
TC Ingress
TC egress
L3 input
L3 output
L3
DD
RX
TX

XDP == FAST PATH

XDP FAST PATH



Tutorial

준비물

1. 컴파일 컴퓨터 1대
2. 테스트 컴퓨터 1대(x86추천)
3. 커널 소스코드
4. clang + llvm(컴파일러)
5. bpftool(bpf 프로그램 로더)
6. bpf를 지원하는 iproute2 패키지

컴파일러

clang + llvm

커널 소스코드

git.kernel.org 의 bpf tree

https://git.kernel.org/pub/scm/linux/kernel/git/bpf/bpf.git

BPF 프로그램 로더

bpftool

https://git.kernel.org/pub/scm/linux/kernel/git/bpf/bpf.git/tree/tools/bpf/bpftool

XDP 설정도구

iproute2

https://git.kernel.org/pub/scm/linux/kernel/git/dborkman/iproute2.git

예제

kernel source code 및 bpf sample code

samples/bpf

BPF 프로그램 컴파일 실습

samples/bpf

$make samples/bpf/

컴파일 결과

xdp_rxq_info_kern.o

xdp_rxq_info

프로그램 실행

./xdp_rxq_info --dev $DEV --action

실행

실행결과

프로그램 분석

xdp_rxq_info == xdp_rxq_info_user.c

xdp_rxq_info 역할?

xdp_rxq_info_kern.o 로드 및 실행!

프로그램 로드

```
$ mount bpffs /sys/fs/bpf -t bpf

$ bpftool prog load ./xdp_rxq_info_kern.o /sys/fs/bpf/xdp
```

프로그램 확인

```
$ ls /sys/fs/bpf/
$ ./bpftool prog list
$ ./bpftool prog dump xlated id X
                       jited
```

XDP프로그램 설정

```
$ ip link set dev lo xdp pin /sys/fs/bpf/xdp
```

XDP프로그램 설정 확인

```
$ ip link show dev lo
```

XDP프로그램 설정 제거

```
$ ip link set dev lo xdp off

$ rm /sys/fs/bpf/xdp
```

간단한 방화벽 만들기

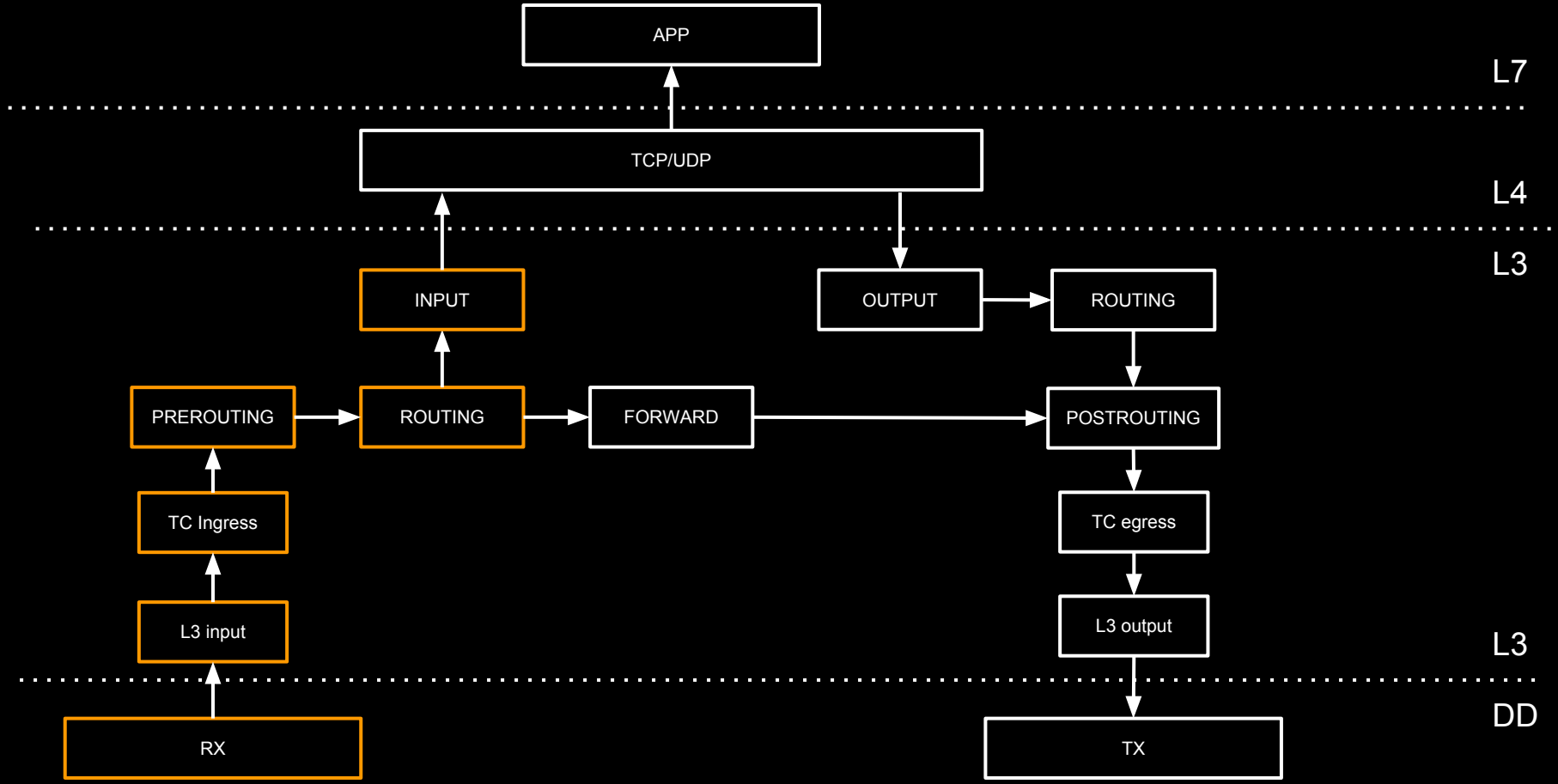
TEST NETWORK

iptables를 사용하여 패킷을 버리기

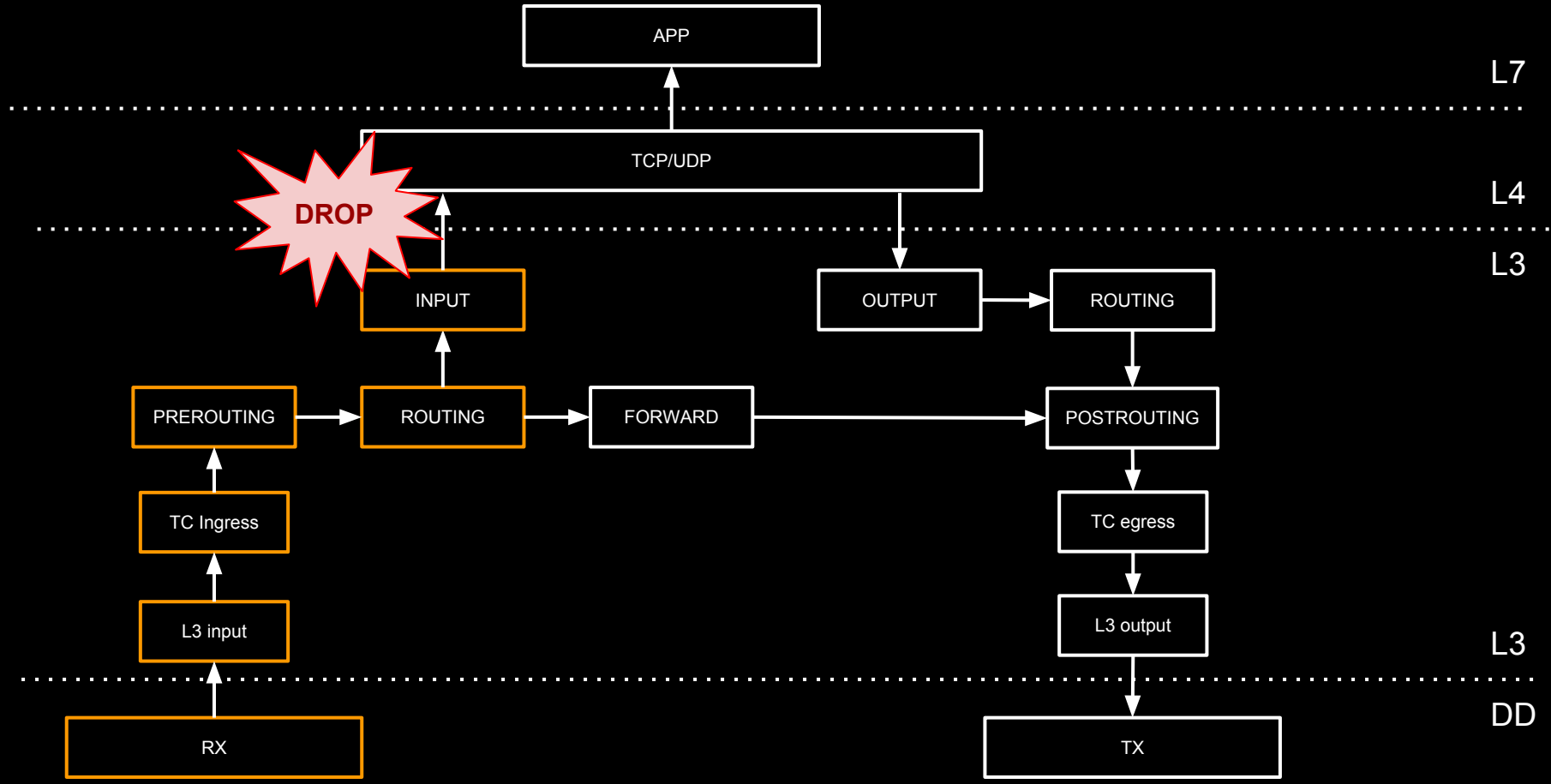
DROP

```
#Guest PC

$ ping 8.8.8.8


$ iptables -A INPUT -s 8.8.8.8 -p icmp -j DROP
```

NORMAL PATH


APP
L7
TCP/UDP
L4
L3
INPUT
OUTPUT
ROUTING
PREROUTING
ROUTING
FORWARD
POSTROUTING
TC Ingress
TC egress
L3 input
L3 output
L3
DD
RX
TX

NORMAL PATH


APP
L7
TCP/UDP
DROP
L4
L3
INPUT
OUTPUT
ROUTING
PREROUTING
ROUTING
FORWARD
POSTROUTING
TC Ingress
TC egress
L3 input
L3 output
L3
DD
RX
TX

XDP를 사용하여 패킷을 버리기

DROP

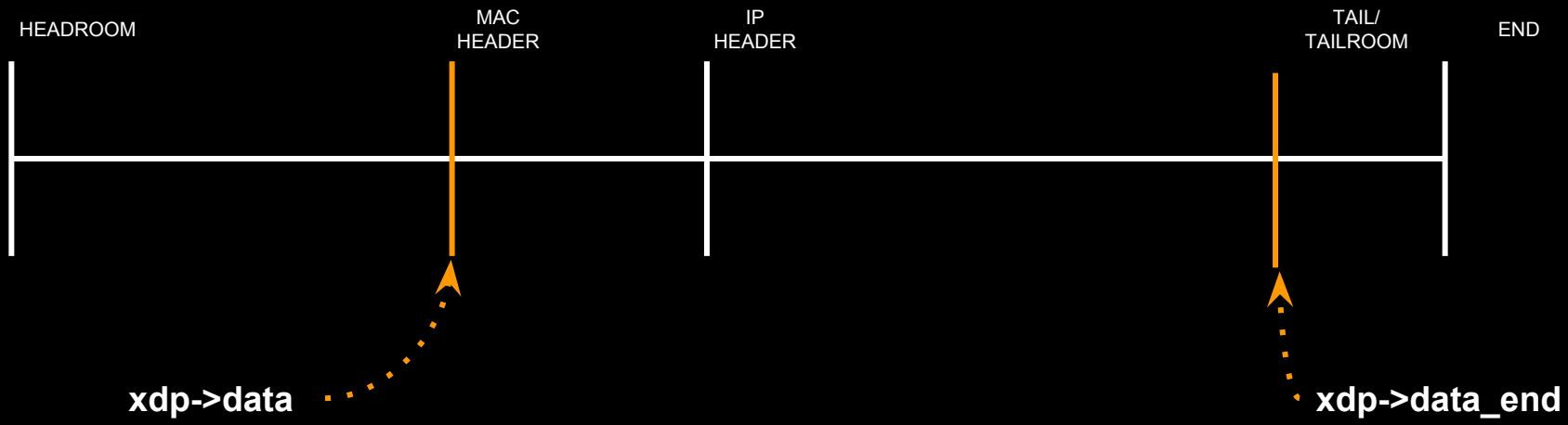
xdp_rxq_info_kern.c 수정

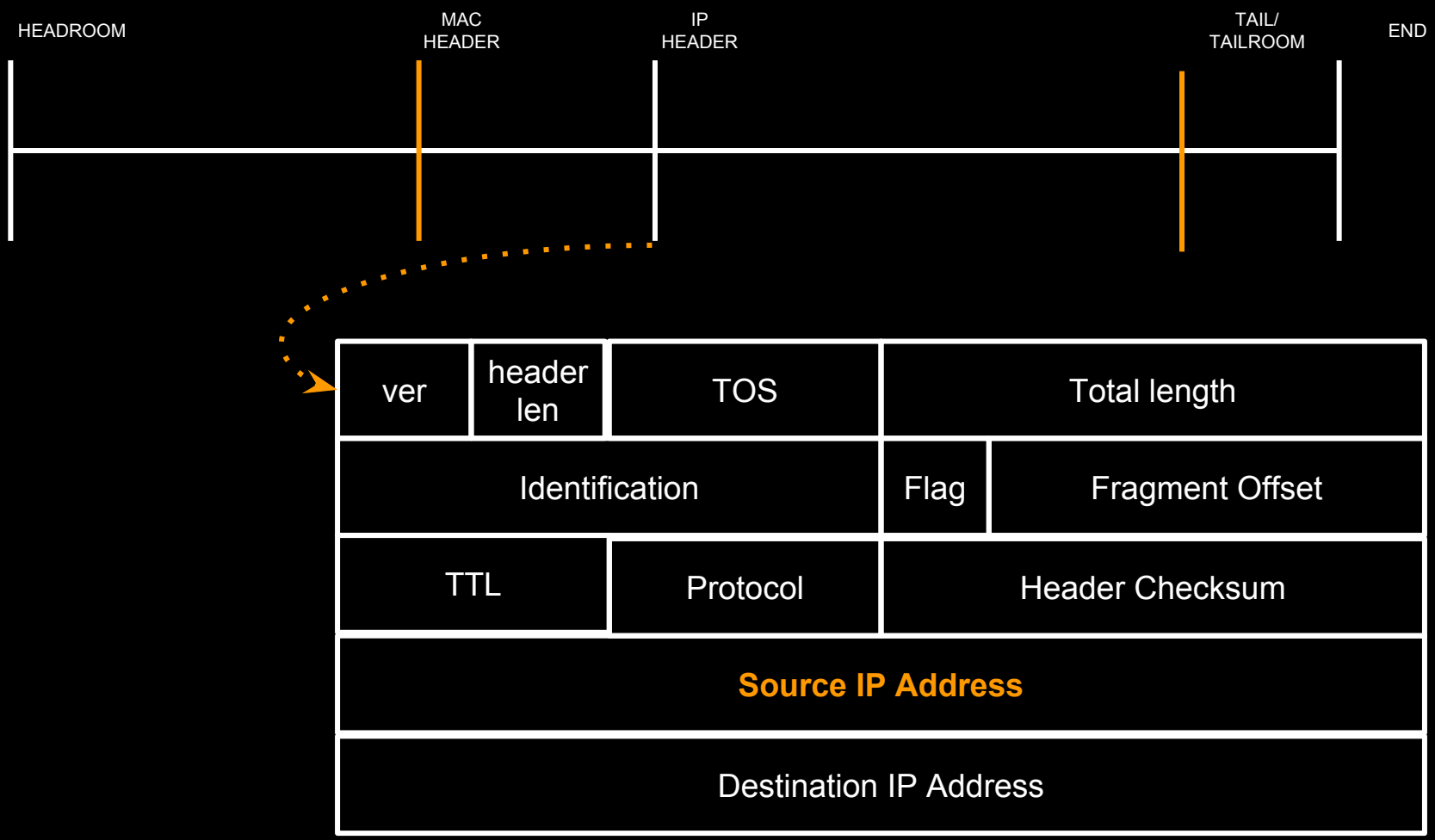
XDP FILTER

ip source address 기준 packet filtering 구현

XDP 자료구조와 SKB

```
struct xdp_buff {
        void *data;
        void *data_end;
        void *data_meta;
        void *data_hard_start;
        unsigned long handle;
        struct xdp_rxq_info *rxq;
};

struct xdp_frame {
        void *data;
        u16 len;
        u16 headroom;
        u16 metasize;
        /* Lifetime of xdp_rxq_info is limited to NAPI/enqueue time,
         * while mem info is valid on remote CPU.
         */
        struct xdp_mem_info mem;
        struct net_device *dev_rx; /* used by cpumap */
};
```


HEADROOM
MAC HEADER
IP HEADER
TAIL/ TAILROOM
END
xdp->data
xdp->data_end


HEADROOM
MAC HEADER
IP HEADER
TAIL/ TAILROOM
END
ver
header len
TOS
Total length
Identification
Flag
Fragment Offset
TTL
Protocol
Header Checksum
Source IP Address
Destination IP Address

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8    ihl:4,
                version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
        __u8    version:4,
                ihl:4;
#else
#error  "Please fix <asm/byteorder.h>"
#endif
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here. */
};
```

프로그램 수정

버전정보 추가

프로그램 수정

map

프로그램 수정

bpf kernel program과 user-space의 공유된 영역

map의 종류

array

cpumap

devmap

...

map을 통한 정책 설정