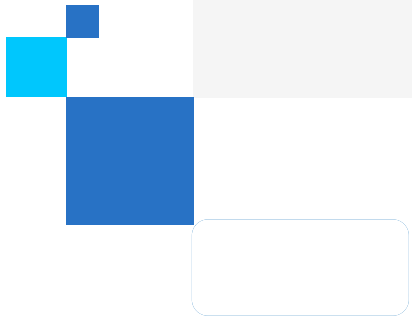
XDP with RDMA(RXE)

Zhu Yanjun, May, 2023

Agenda

- **XDP** (Introduction, XDP Use cases, XDP Actions)
- **RDMA** (Introduction, IB and RoCEv1, RoCEv2, Summary and SoftRoCE)
- **High level Design**
- **XDP user space** (Introduction, new flag, core struct, output)
- **XDP kernel** (NIC driver, RXE initialization, RXE XMIT and RXE RECV)
- **Demo**



XDP Introduction

XDP (eXpress Data Path) is an eBPF-based high-performance data path used to send and receive network packets at high rates by bypassing most of the operating system networking stack. XDP is a programmable and high performance networking data path.

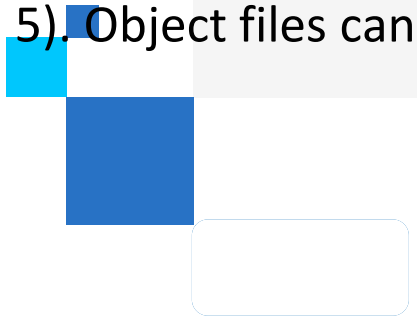
1). XDP is eBPF + early packet processing

2). XDP is short for eXpressed Data Path (it is merged in kernel 4.8+)

3). XDP is Run-time programmable packet processing inside the kernel not kernel-bypass

4). XDP Application are compiled to platform-independent eBPF bytecode

5). Object files can be loaded on multiple kernels and architectures without recompiling



XDP Use cases

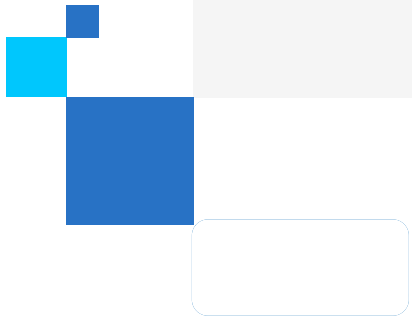
XDP can be used in the following scenarios:

1. DDOS – early dropping malicious packets. When DDOS is detected, the malicious packets are dropped with XDP_DROP/ABORT silently. This decreases host CPU utilizations.

2. Load balancing – with XDP_REDIRECT, XDP will redirect packets to other NICs or user space application to implement load balance.

3. Sample – Monitoring the contents of packets

4. Fast forwarding (destination is networking device or user space application – With AF_XDP, packets are redirected to user space application or other NICs directly without network stack)



XDP Actions

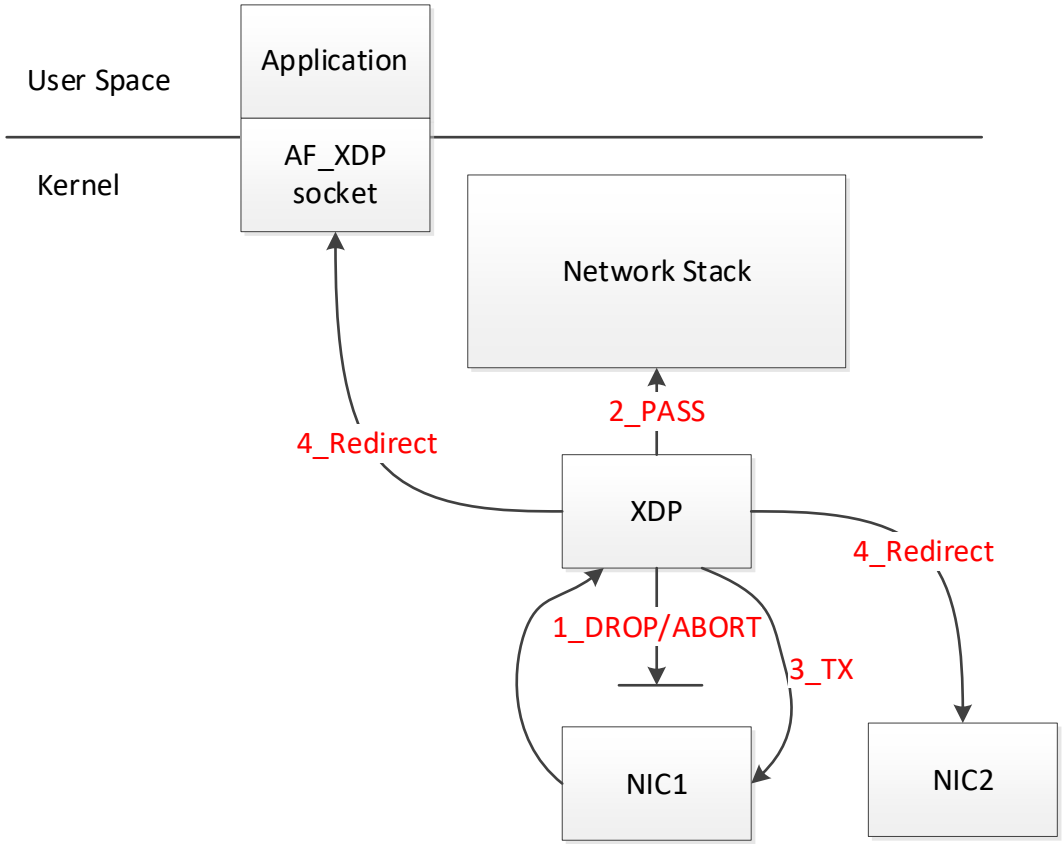
A valid XDP program must return one of these defined values. Unknown return codes will result in packet drops and a warning via bpf_warn_invalid_xdp_action().

1. XDP_DROP/ABORT
   drop packet immediately,
   ABORT drop with trace point exception

2. XDP_PASS
   build skb and pass the packet to network stack

3. XDP_TX
   transmit the packet back to the interface

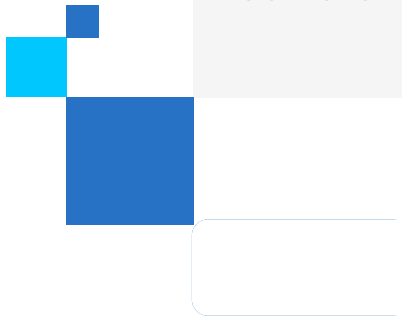
4. XDP_REDIRECT
   redirect packets to other NICs or User space Application



RDMA Introduction

1. RDMA is short for Remote Direct Memory Access.
   It is A (relatively) new method for interconnecting platforms in high-speed networks that overcomes many of the difficulties encountered with traditional networks such as TCP/IP over Ethernet.
   1) new standards
   2) new protocols
   3) new hardware interface cards and switches
   4) new software

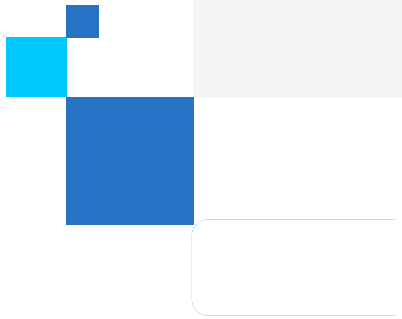
2. RDMA implementations:
   Infiniband and RoCEv1/v2



Infiniband and RoCE

1. Infiniband
   Infiniband implementation needs special devices, for example HCA (Host Channel Adapter), IB switch and special cables.

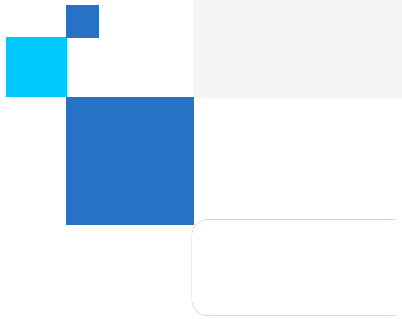
2. RoCE
   1) RoCEv1 and RoCEv2
   2) RoCEv1 is based on TCP/IP ethernet layer. It only works in LAN (Local Area Network). It is not popular.
   3) RoCEv2 is based on UDP. It is popular in public/private Cloud and Data Center.



RoCEv2

1. RoCEv2 is based on UDP/IPv4 or UDP/IPv6. The target udp port 4791 is reserved for RoCEv2. Because RoCEv2 can be routable, RoCEv2 sometimes can be called Routable RoCE or RRoCE.

2. RoCEv2 is supported in Mellanox OFED2.3 or above. In Linux 4.5, RoCEv2 is supported. Intel E810 also supports RoCEv2.



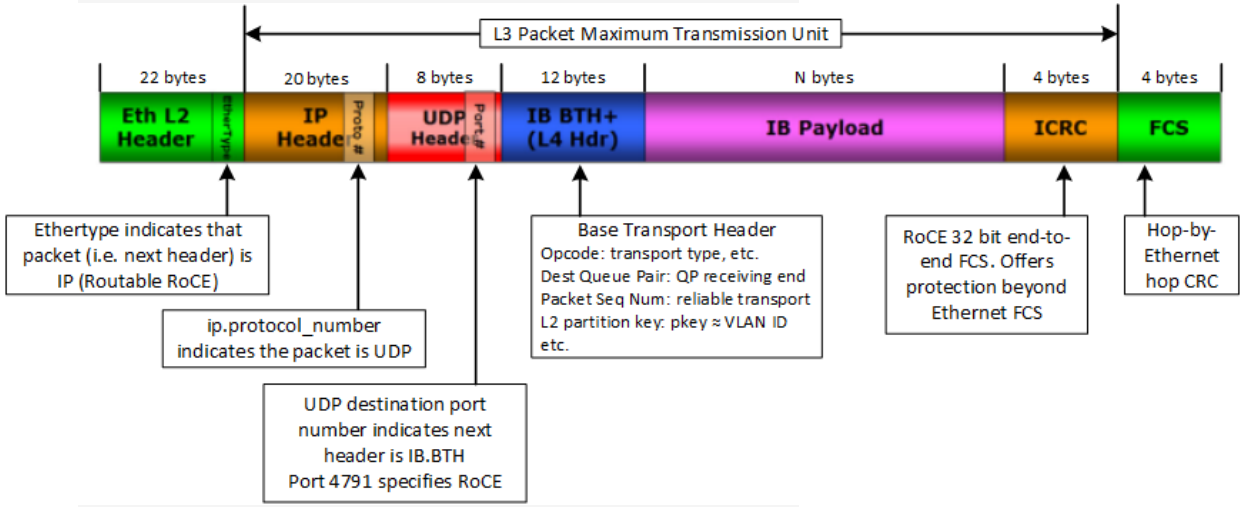
RDMA summary

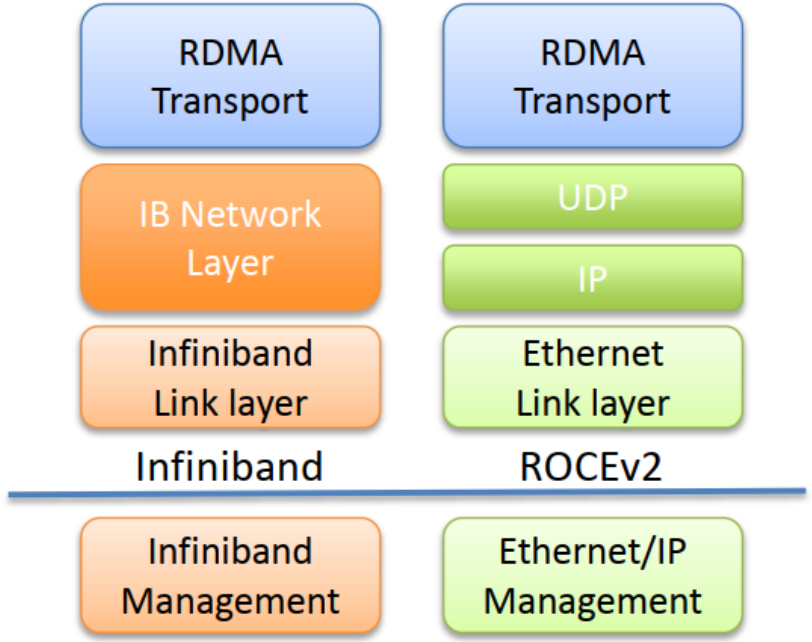
1. RDMA is a networking performance optimization.
   Traditional networking stacks (sockets/TCP/IP/Ethernet) have inherent performance barriers and CPU overhead. RDMA is designed to reduce or eliminate these.

RDMA Techniques	Benefit		
	CPU Util	Latency	Mem bw
Offload network transport (e.g. UDP/IP) from Host	☑	☑	
Eliminate receive memory copies with tagged buffers	☑	☑	☑
Reduce context switching with OS bypass (map NIC hw resources into user space)	☑	☑	
Define an asynchronous “verbs” API (socket is synchronous)	☑		☑
Preserve message boundaries to enable application (e. g. SCSI) header/data separation	☑		☑
Message-level (not packet-level) interrupt coalescing	☑		

RDMA SoftRoCE

1. SoftRoCE is a software implementation of the IBTA RoCEv2 specification, RDMA transport services over Ethernet network.

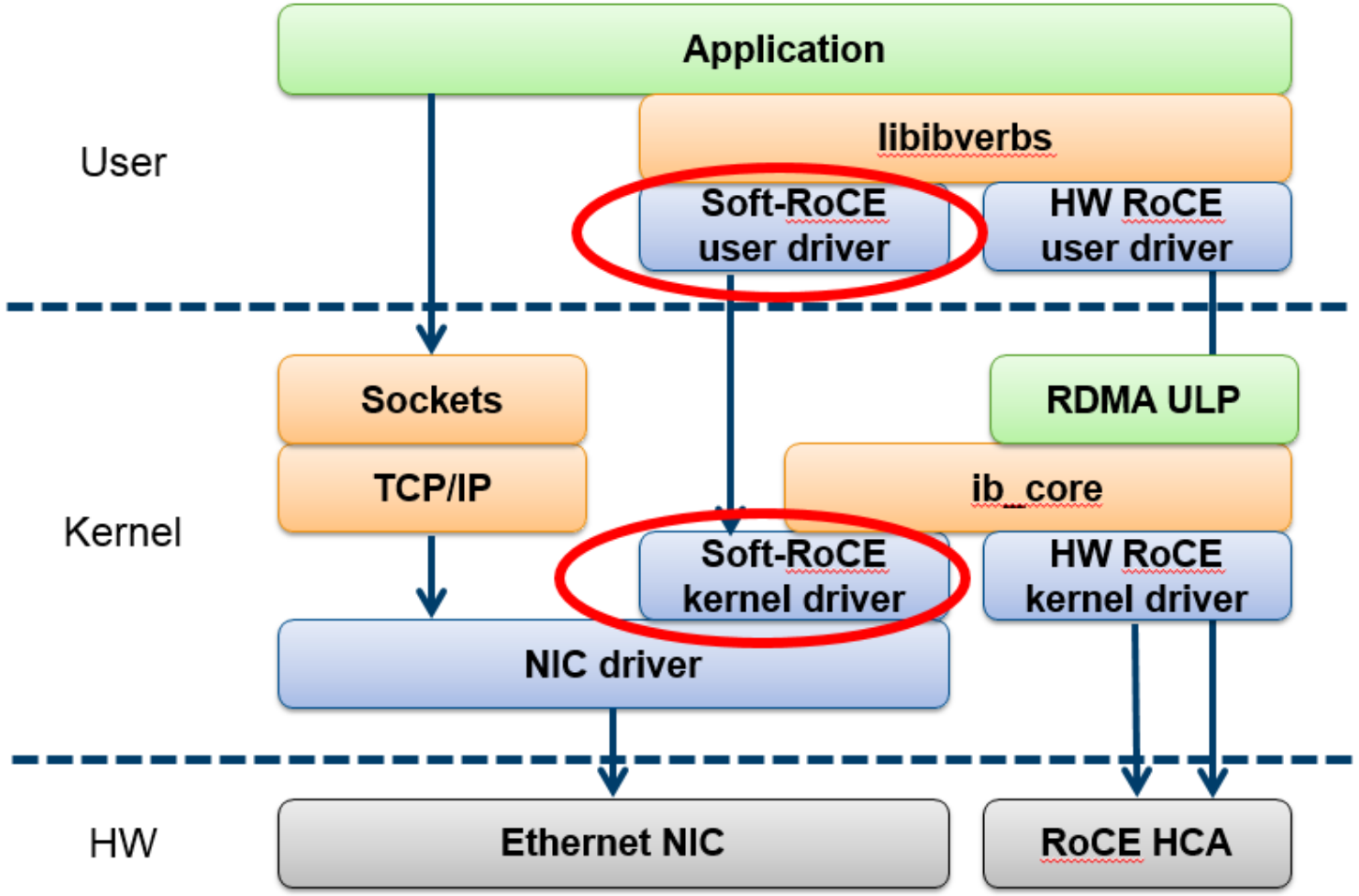




2. Use cases
   1) Develop new RDMA features on SoftRoCE
   2) Easy to debug on developing application on SoftRoCE
   3) Make tests on SoftRoCE
   4) Asymmetric deployments, SoftRoCE connects to RDMA hardware, e.g. MLX5, E810

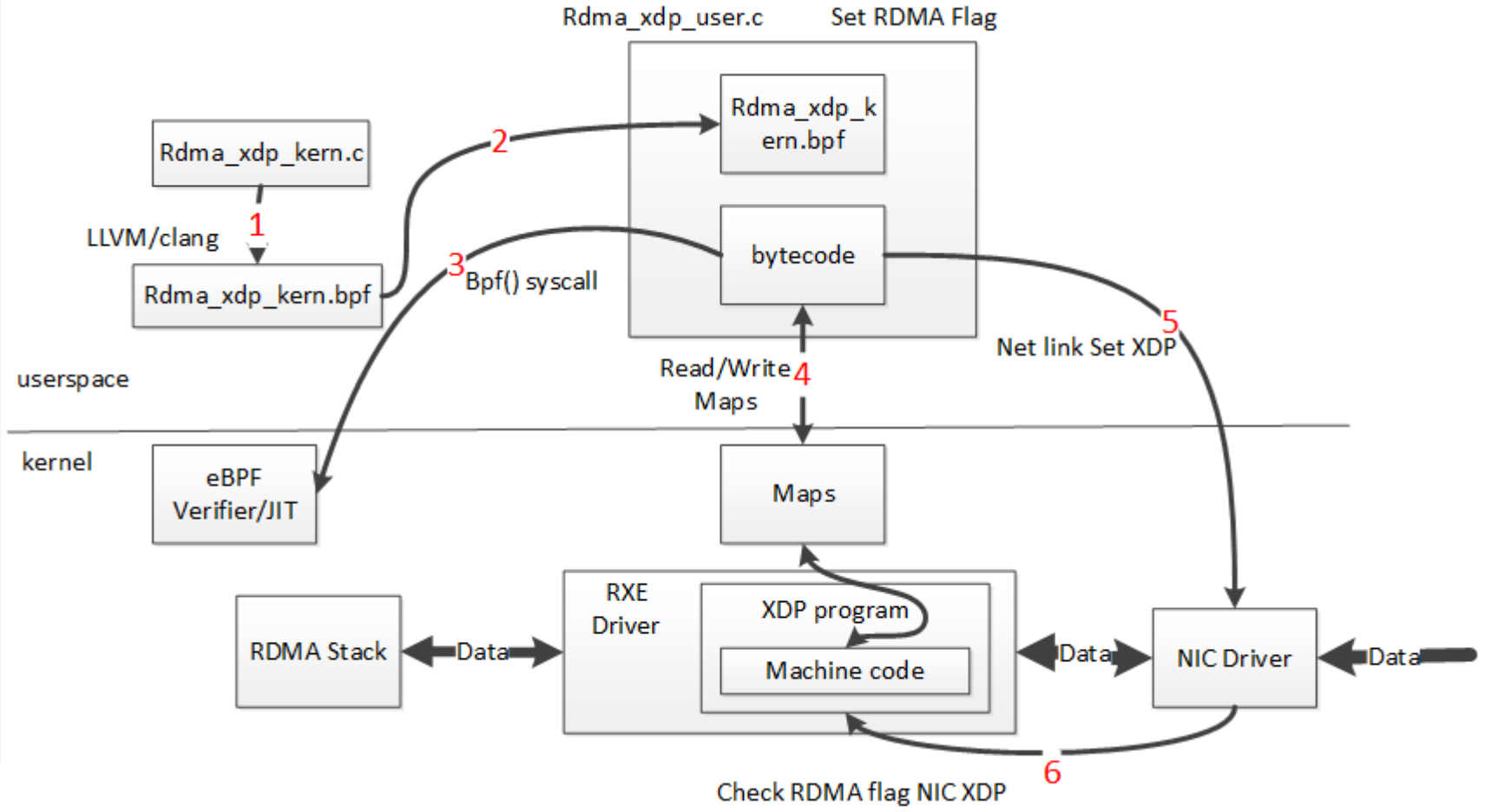
RDMA SoftRoCE Architecture

User Space: SoftRoCE user driver is in rdma-core software package. IB verbs generic APIs finally call the specific implementations of SoftRoCE user driver.
Kernel: SoftRoCE kernel driver is attached on a specific NIC driver(ICE driver). It receives rdma packets from RDMA stack, build a skb and put rdma data into UDP payload, then send skb to TCP/IP stack.
HW: Any NIC including E810 (Intel). In my demo, I use E810 without irdma, attach SoftRoCE on E810



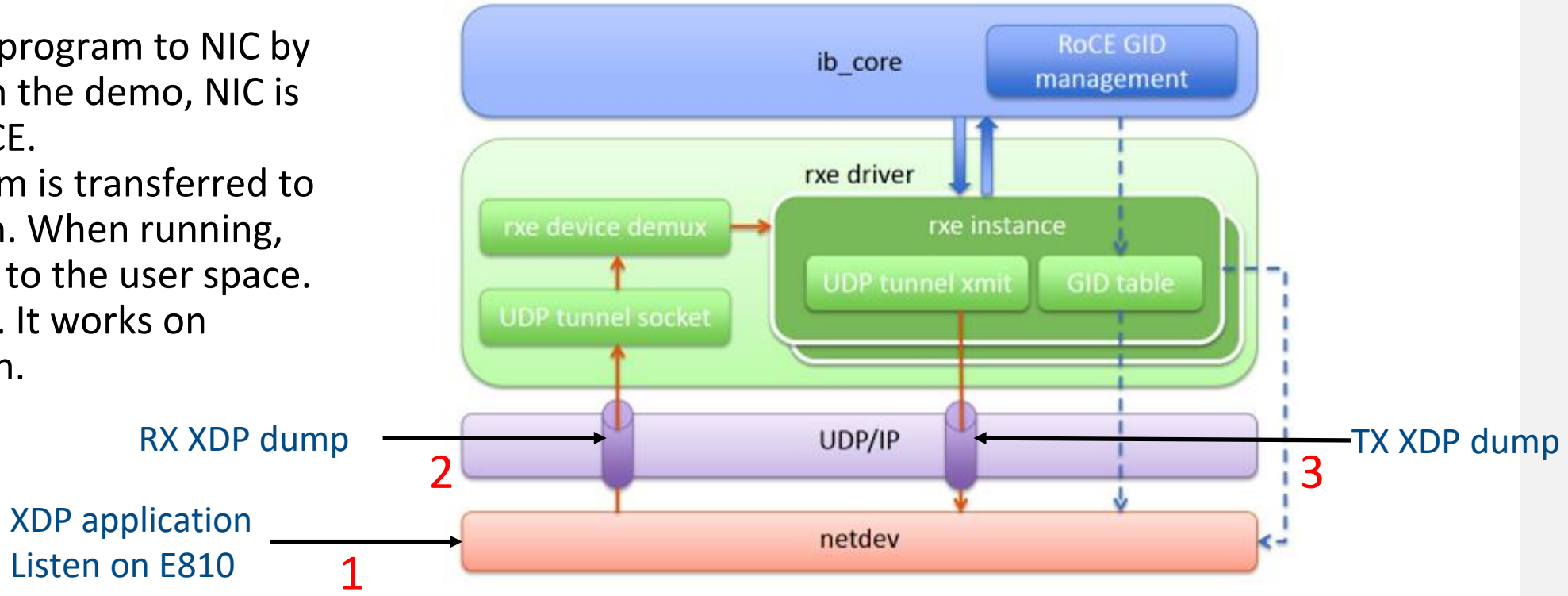
XDP with RDMA Design

User Space:
Path1 and 2: Rdma_xdp user space c files are compiled into a bin file;
Path 3: This bin is checked by eBPF verifier.
Kernel:
Path 5: This bin file is attached to NIC (E810).
Path 6: The bin is transferred to SoftRoCE driver.
Path 4: Mapping rdma data between kernel and user space.



XDP with RDMA

Path 1: Attach xdp program to NIC by ip link command. In the demo, NIC is E810, its driver is ICE.
Path 2: XDP program is transferred to SoftRoCE Recv path. When running, dump rdma packet to the user space.
Path 3: Similar to 2. It works on SoftRoCE XMIT path.




```
RDMA/rxe: Add rxe rx_xdp_prog
RDMA/rxe: Recv netdev state change and set xdp_prog
RDMA/rxe: add rxe_xmit xdp capture packets
ice: Set netdev xdp_prog
samples/bpf: Add rdma userspace files into Makefile
samples/bpf: add rdma user/kernel files
```

Userspace Applications

1. In userspace, two files: xdp_rdma_pkts_kern.c, xdp_rdma_pkts_user.c

2. In xdp_rdma_pkts_kern.c, the returned value is XDP_PASS. This means that the packets is resumed to travel in the kernel stack.

3. In xdp_rdma_pkts_user.c, the followings will do:
   1) A new XDP flag XDP_FLAGS_RDMA is introduced;
   2) Parse the RDMA packets;
   3) Output the information of packets;

xdp_rdma_pkts_user.c new Flag

1. A new flag XDP_FLAG_RDMA is introduced, shown in the following

```
#define XDP_FLAGS_REPLACE               (1U << 4)
#define XDP_FLAGS_RDMA                  (1U << 5)
#define XDP_FLAGS_MODES                 (XDP_FLAGS_SKB_MODE | \
                                         XDP_FLAGS_DRV_MODE | \
                                         XDP_FLAGS_HW_MODE)
#define XDP_FLAGS_MASK                  (XDP_FLAGS_UPDATE_IF_NOEXIST | \
                                         XDP_FLAGS_MODES | XDP_FLAGS_REPLACE | \
                                         XDP_FLAGS_RDMA)
```

2. This flag is set in do_attach of xdp_rdma_pkts_user.c (xdp program in user space) and checked in ice_xdp of ice_main.c (in kernel)

```
static int do_attach(int idx, int fd, const char *name)
{
        struct bpf_prog_info info = {};
        __u32 info_len = sizeof(info);
        int err;

        err = bpf_xdp_attach(idx, fd, xdp_flags|XDP_FLAGS_RDMA, NULL);
        if (err < 0) {
                printf("ERROR: failed to attach program to %s\n", name);
                return err;
        }
```

```
        switch (xdp->command) {
        case XDP_SETUP_PROG:
                if (xdp->flags & XDP_FLAGS_RDMA) {
                        old_prog = xchg(&dev->xdp_prog, xdp->prog);
                        if (old_prog)
                                bpf_prog_put(old_prog);
```

Userspace Applications core struct

1. With this struct, RDMA packets are parsed, the core struct is as below:

```
struct rxe_opcode_info {
        char    *name;
        int     length;
        int     offset[NUM_HDR_TYPES];
};
```

name: the operation name

length: the distance between the beginning of BTH and the beginning of this struct;

offset: the structs in this rdma packets;

The core struct is initialized as below:

```
struct rxe_opcode_info rxe_opcode[RXE_NUM_OPCODE] = {
        [IB_OPCODE_RC_SEND_FIRST] = {
                .name   = "IB_OPCODE_RC_SEND_FIRST",
                .length = RXE_BTH_BYTES,
                .offset = {
                        [RXE_BTH]       = 0,
                        [RXE_PAYLOAD]   = RXE_BTH_BYTES,
                }
        },
        //RC
        …
        //RD
        …
        //UC
        …
        //UD
        …
}
```

Userspace Applications output

1. The output contains src/dst ip address, udp information and RDMA packets information

This is a rping packet analysis. Including IP UDP and RDMA headers and payload

```
Pkt len: 124   bytes. IP hdr:
struct iphdr {
        __be32  saddr; 192.168.1.149
        __be32  daddr; 192.168.1.156
};
udp hdr:
struct udphdr {
        __be16  source; 62933
        __be16  dest;   4791
        __be16  len;    104
        __sum16 check;  0
};
IB_OPCODE_RC_RDMA_WRITE_ONLY
struct rxe_bth {
        __u8          opcode; 0xa
        __u8          flags;  0x0
        __be16          pkey;   0xffff
        __be32          qpn;    0x16
        __be32          apsn;   0x22ea59
};
RXE_RETH
struct rxe_reth {
        __be64          va;
        __be32          rkey;
        __be32          len;
};
RXE_PAYLOAD
rdma-ping-0: ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr00<ffffffc8070bffffffdeffffffac1effffffe009000000
```

Kernel

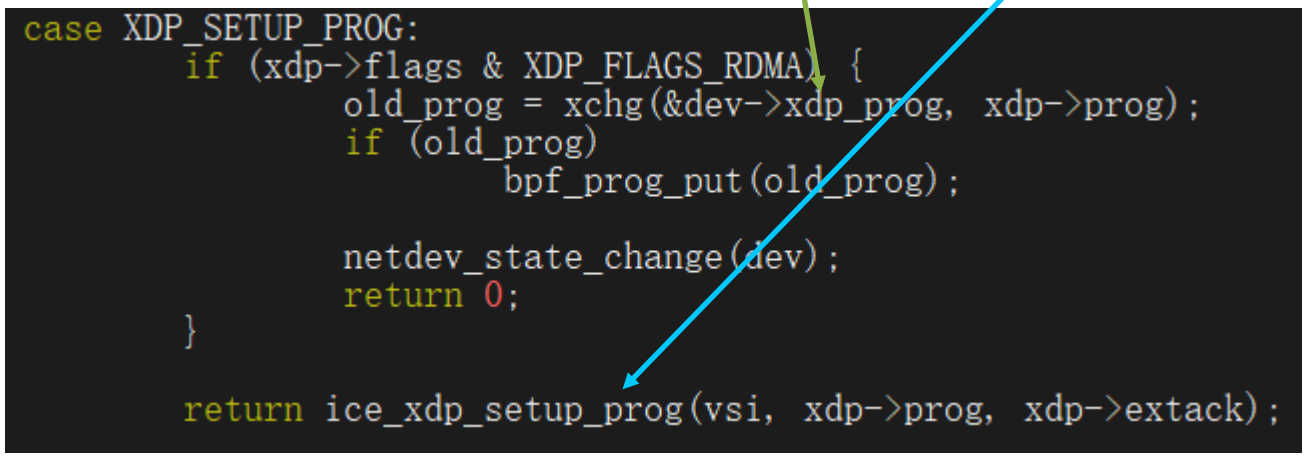
1. In kernel, two drivers are involved:
   1) NIC driver (ICE, driver of E810)
   2) RXE driver

2. In NIC driver, the function that ndo_bpf points will check the flag XDP_FLAGS_RDMA. If XDP_FLAGS_RDMA is set, xdp program is transferred to SoftRoCE. If not, this xdp program is used in the NIC driver.

3. In RXE driver, 3 parts are involved:
   1) rxe_xdp_prog initialization
   2) xmit packets capture
   3) recv packets capture

Kernel: NIC driver

1. NIC driver checks the flag XDP_FLAGS_RDMA. If the flag is set, the xdp command XDP_SETUP_PROG will set netdev->xdp_prog as below. If not, xdp command will work as usual. In ICE driver, that is, the original ice_xdp_setup_prog is called to perform original tasks.

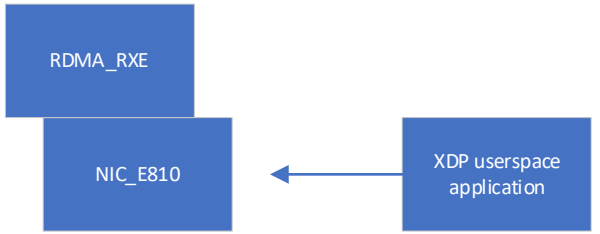


```
case XDP_SETUP_PROG:
        if (xdp->flags & XDP_FLAGS_RDMA) {
                old_prog = xchg(&dev->xdp_prog, xdp->prog);
                if (old_prog)
                        bpf_prog_put(old_prog);

                netdev_state_change(dev);
                return 0;
        }

        return ice_xdp_setup_prog(vsi, xdp->prog, xdp->extack);
```

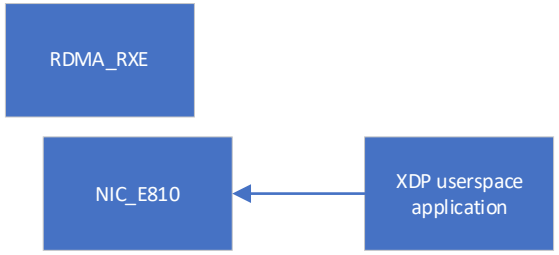
Kernel: SoftRoCE driver initialization

In softRoCE driver, the XDP initialization includes XMIT and RECV initializations. And the following scenarios should be considered.

1) When SoftRoCE driver has been attached to the NIC, the event NETDEV_CHANGE is sent out. SoftRoCE driver receives this event and makes TX/RX initializations.



2) Before SoftRoCE driver is attached to the NIC, the xdp userspace application has already been attached to the NIC. During SoftRoCE driver configuring network, the SoftRoCE XMIT/RECV initializations are done.

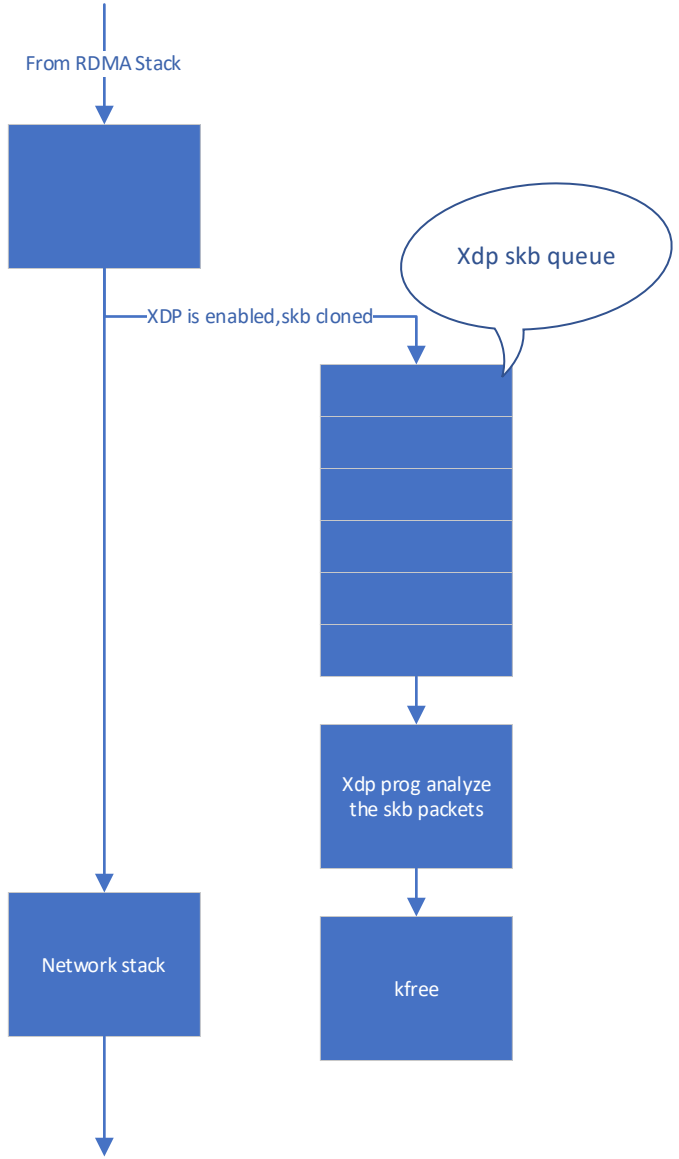


Kernel SoftRoCE driver xmit path

1. In SoftRoCE driver XMIT path, the xdp fetch the skb to a new xdp skb list. The xdp userspace application fetch the SKBs and analyze them.

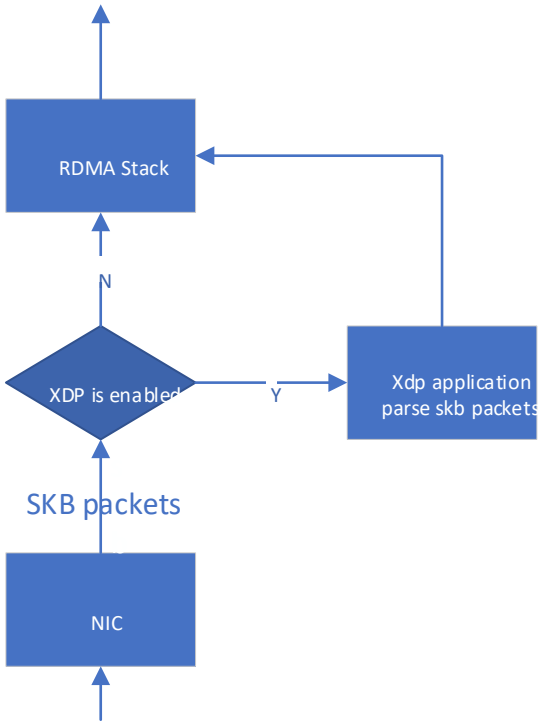
   In the right diagram, the rdma data from RDMA stack is built into a skb packet and stored in udp payload at the same time it is cloned to a xdp skb queue. Then xdp program fetches a skb packet from this queue and analyze the rdma data. Finally this skb packet is freed.

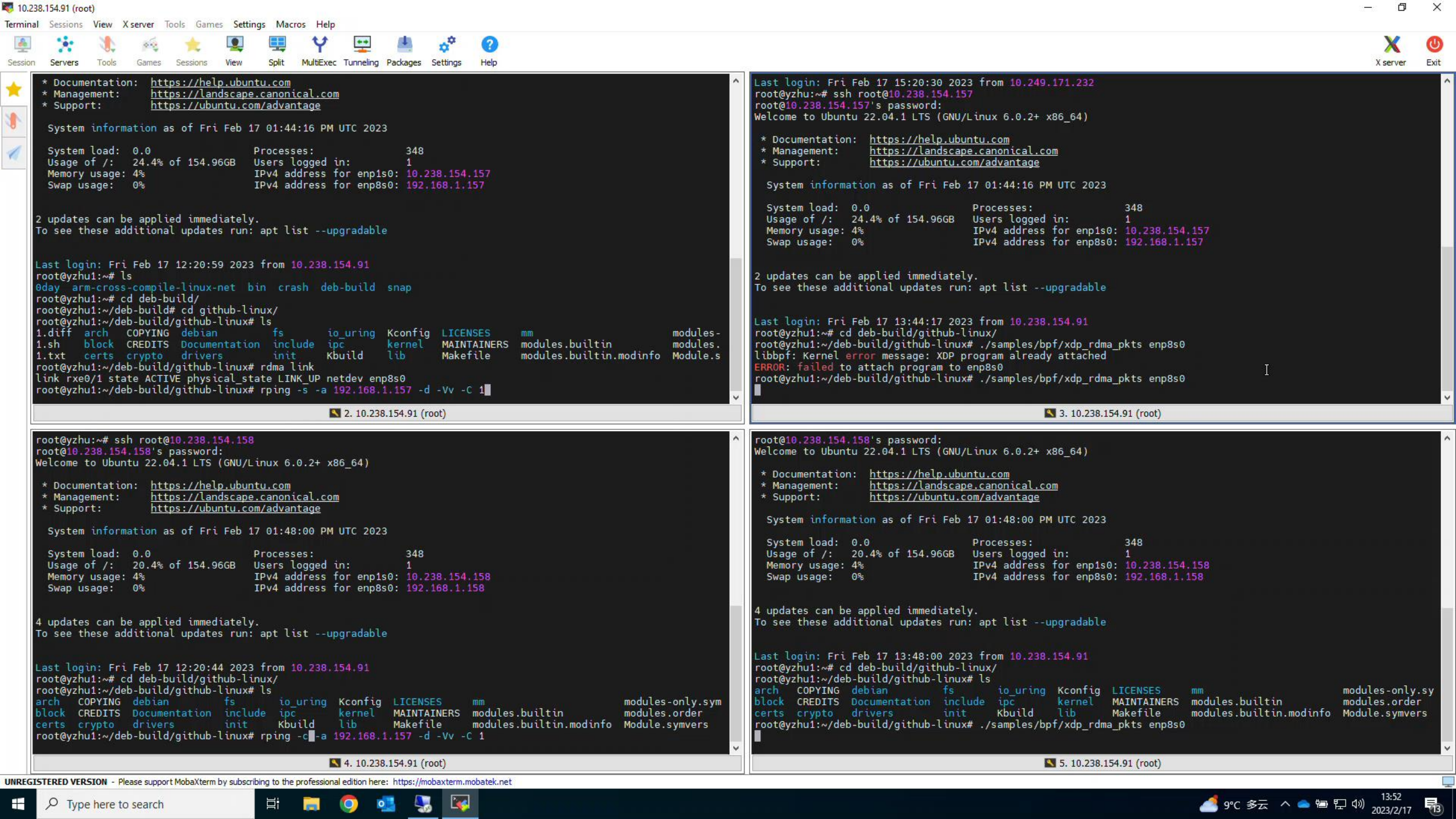
   The built skb packets are still sent to network stack.



Kernel SoftRoCE recv path

1. In SoftRoCE recv path, the skb packets are from NIC. If xdp is enabled, the skb packets are parsed by xdp userspace application, then still goes RDMA stack. If not, the skb packets goes to RDMA stack directly.
   In the right diagram, skb packets from NIC are sent to SoftRoCE recv path because of udp port 4791.
   In recv path, if XDP is enabled, xdp program will parse the skb packets. If not, the skb packets are handled and sent to RDMA stack.




10.238.154.91 (root)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Fri Feb 17 01:44:16 PM UTC 2023
System load: 0.0 Processes: 348
Usage of /: 24.4% of 154.96GB Users logged in: 1
Memory usage: 4% IPv4 address for enp1s0: 10.238.154.157
Swap usage: 0% IPv4 address for enp8s0: 192.168.1.157
2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Last login: Fri Feb 17 12:20:59 2023 from 10.238.154.91
root@yzhu1:~# ls
0day arm-cross-compile-linux-net bin crash deb-build snap
root@yzhu1:~# cd deb-build/
root@yzhu1:~/deb-build# cd github-linux/
root@yzhu1:~/deb-build/github-linux# ls
1.diff arch COPYING debian fs io_uring Kconfig LICENSES mm modules-
1.sh block CREDITS Documentation include ipc kernel MAINTAINERS modules.builtin modules.
1.txt certs crypto drivers init Kbuild lib Makefile modules.builtin.modinfo Module.s
root@yzhu1:~/deb-build/github-linux# rdma link
link rxe0/1 state ACTIVE physical_state LINK_UP netdev enp8s0
root@yzhu1:~/deb-build/github-linux# rping -s -a 192.168.1.157 -d -Vv -C 1
2. 10.238.154.91 (root)
Last login: Fri Feb 17 15:20:30 2023 from 10.249.171.232
root@yzhu:~# ssh root@10.238.154.157
root@10.238.154.157's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 6.0.2+ x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Fri Feb 17 01:44:16 PM UTC 2023
System load: 0.0 Processes: 348
Usage of /: 24.4% of 154.96GB Users logged in: 1
Memory usage: 4% IPv4 address for enp1s0: 10.238.154.157
Swap usage: 0% IPv4 address for enp8s0: 192.168.1.157
2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Last login: Fri Feb 17 13:44:17 2023 from 10.238.154.91
root@yzhu1:~# cd deb-build/github-linux/
root@yzhu1:~/deb-build/github-linux# ./samples/bpf/xdp_rdma_pkts enp8s0
libbpf: Kernel error message: XDP program already attached
ERROR: failed to attach program to enp8s0
root@yzhu1:~/deb-build/github-linux# ./samples/bpf/xdp_rdma_pkts enp8s0
3. 10.238.154.91 (root)
root@yzhu:~# ssh root@10.238.154.158
root@10.238.154.158's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 6.0.2+ x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Fri Feb 17 01:48:00 PM UTC 2023
System load: 0.0 Processes: 348
Usage of /: 20.4% of 154.96GB Users logged in: 1
Memory usage: 4% IPv4 address for enp1s0: 10.238.154.158
Swap usage: 0% IPv4 address for enp8s0: 192.168.1.158
4 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Last login: Fri Feb 17 12:20:44 2023 from 10.238.154.91
root@yzhu1:~# cd deb-build/github-linux/
root@yzhu1:~/deb-build/github-linux# ls
arch COPYING debian fs io_uring Kconfig LICENSES mm modules-only.sym
block CREDITS Documentation include ipc kernel MAINTAINERS modules.builtin modules.order
certs crypto drivers init Kbuild lib Makefile modules.builtin.modinfo Module.symvers
root@yzhu1:~/deb-build/github-linux# rping -c -a 192.168.1.157 -d -Vv -C 1
4. 10.238.154.91 (root)
root@10.238.154.158's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 6.0.2+ x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Fri Feb 17 01:48:00 PM UTC 2023
System load: 0.0 Processes: 348
Usage of /: 20.4% of 154.96GB Users logged in: 1
Memory usage: 4% IPv4 address for enp1s0: 10.238.154.158
Swap usage: 0% IPv4 address for enp8s0: 192.168.1.158
4 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Last login: Fri Feb 17 13:48:00 2023 from 10.238.154.91
root@yzhu1:~# cd deb-build/github-linux/
root@yzhu1:~/deb-build/github-linux# ls
arch COPYING debian fs io_uring Kconfig LICENSES mm modules-only.sy
block CREDITS Documentation include ipc kernel MAINTAINERS modules.builtin modules.order
certs crypto drivers init Kbuild lib Makefile modules.builtin.modinfo Module.symvers
root@yzhu1:~/deb-build/github-linux# ./samples/bpf/xdp_rdma_pkts enp8s0
5. 10.238.154.91 (root)
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
Type here to search
9°C 多云
13:52
2023/2/17

Thanks for your attention!