netmapとDPDKによるパケット処理プログラミング
導入

XFLAG スタジオ
吉野純平

XFLAG
ケタハズレな冒険を。

このプログラムの目的

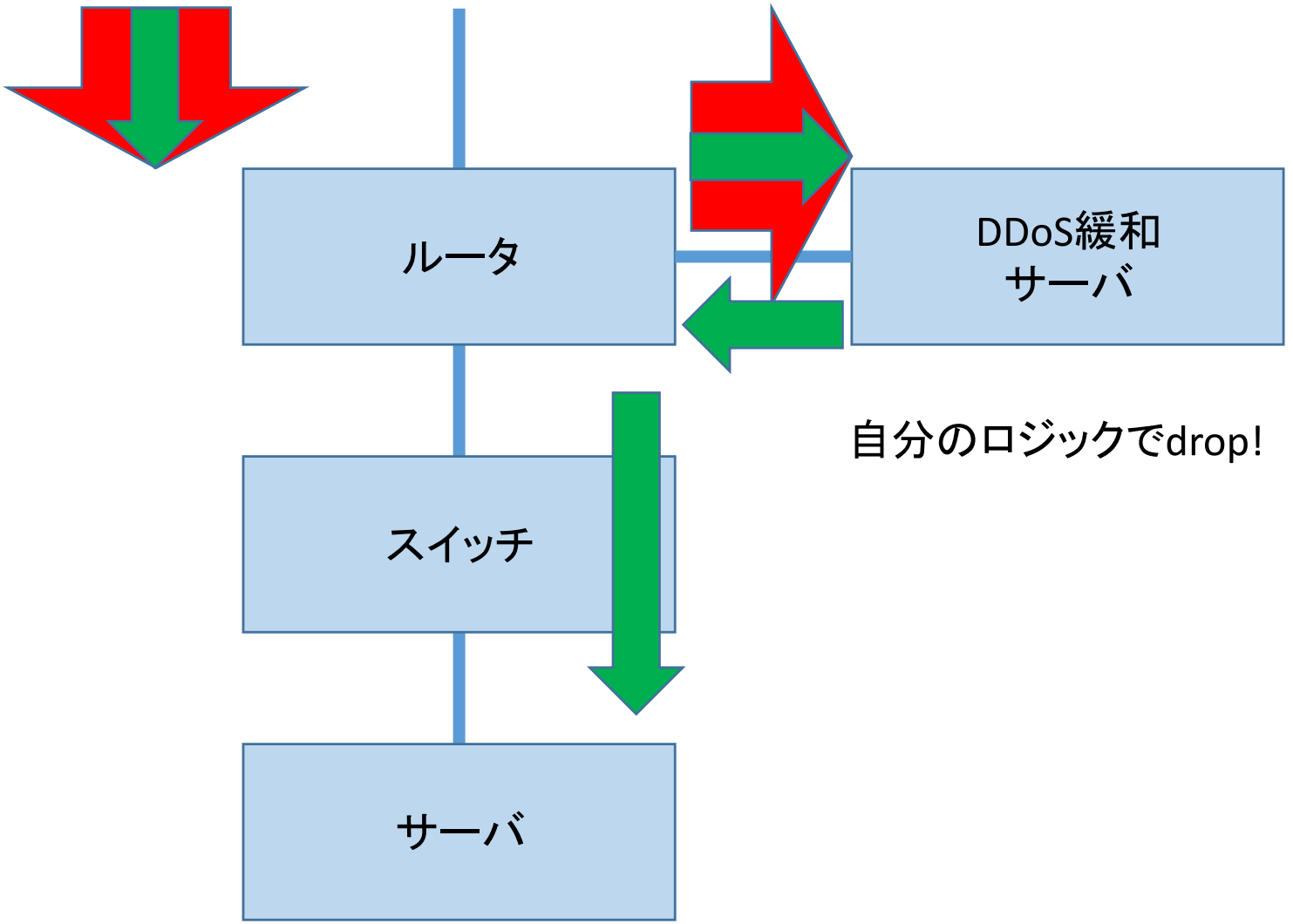
パケット処理したいですよね！

パケット処理したい！
パケット処理したい！
パケット処理したい！
パケット処理したい！

もし可能になれば・・・

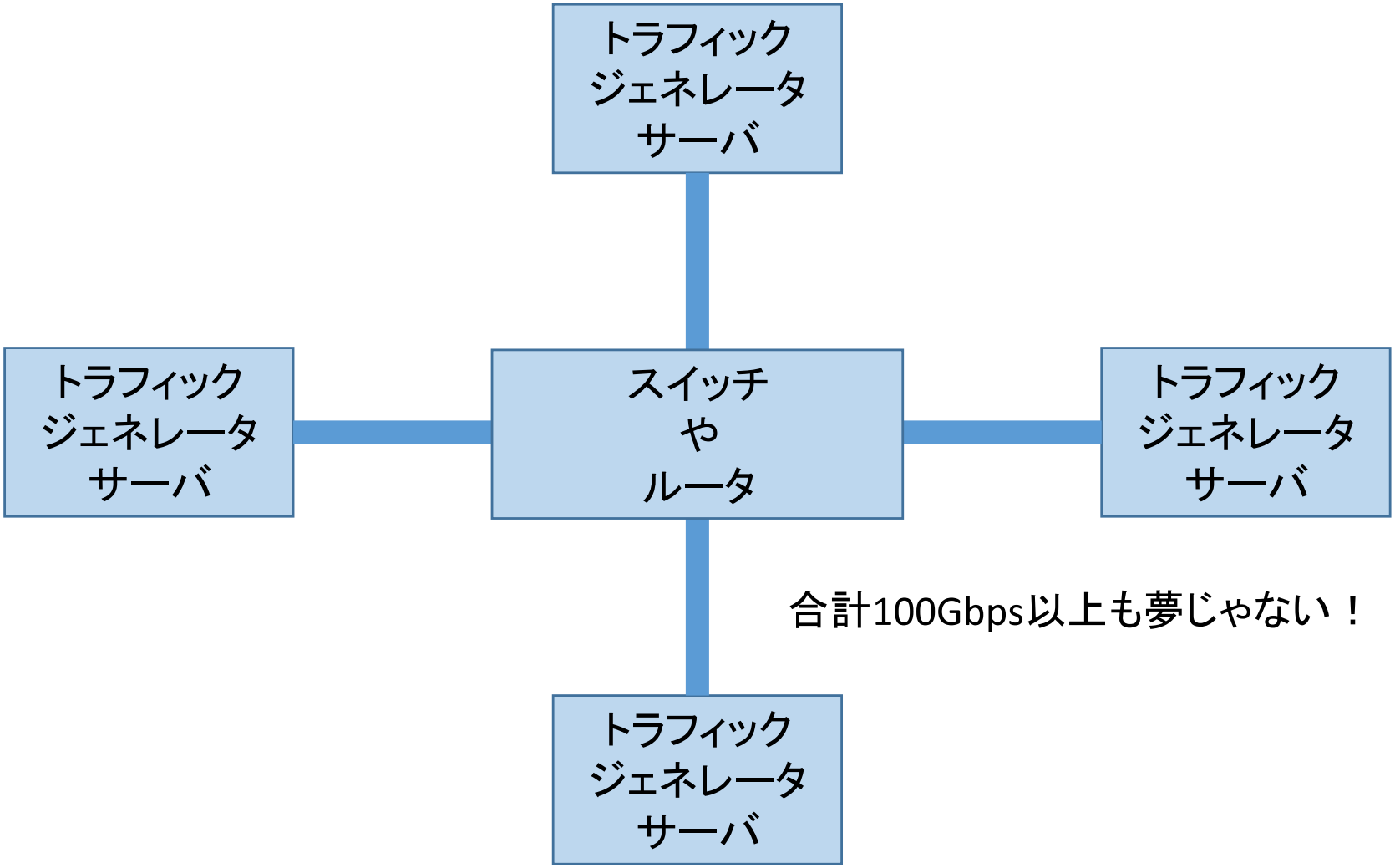
- ブラックボックスな箱からの脱却
- コスト構造の変化

たとえばこんなこと1



自分のロジックでdrop!

たとえばこんなこと2



合計100Gbps以上も夢じゃない！

とはいえ、簡単じゃない？

最初はわかりづらいところもある

↓

開発するためのセッションをしたい

本セッションの流れ

- **導入（この発表）**
  - 前座としてTUN/TAPの場合を紹介
- **netmapによる実践パケット処理プログラミング**
  - 清水さん
- **作って覚えるDPDKプログラミング**
  - 沖さん
- **まとめ**

TUN/TAPの場合

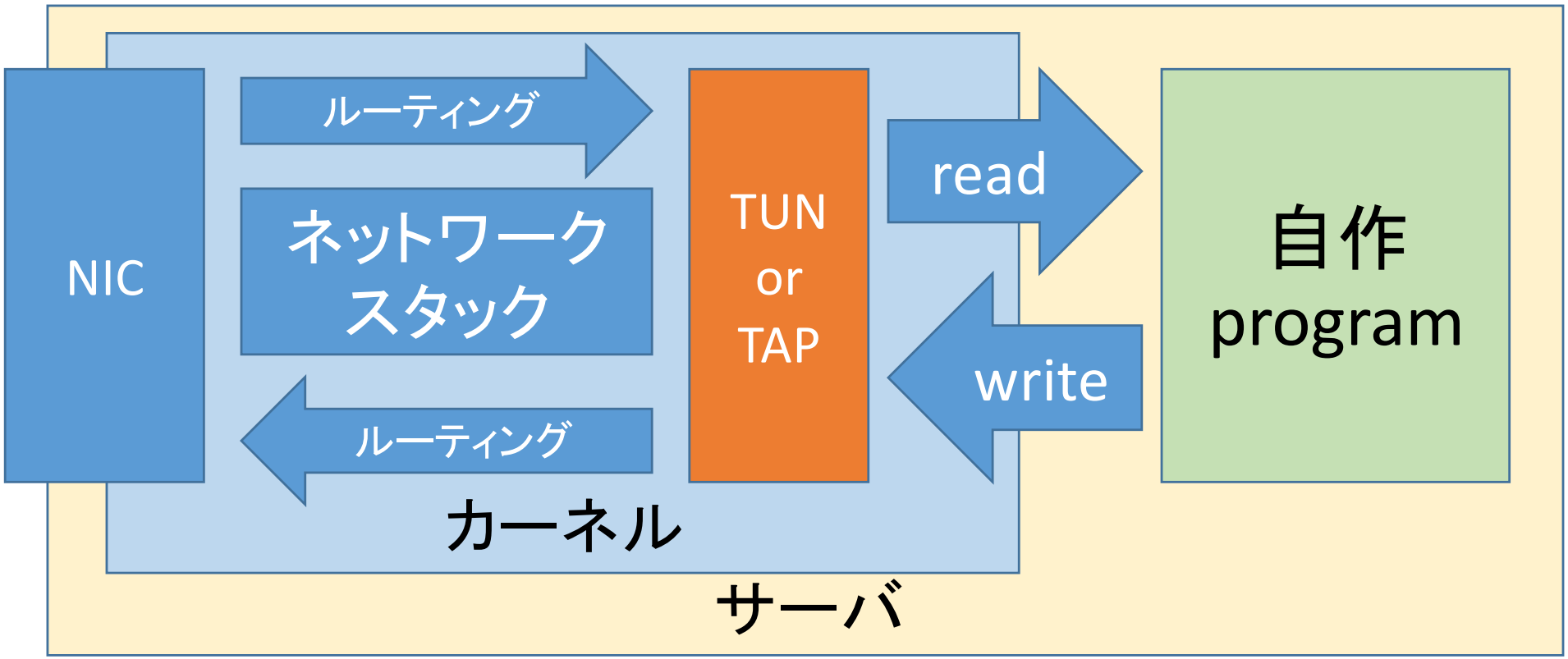
TUN/TAPとは何か

- 仮想デバイスのドライバ
- 簡単にパケットを送受信できる
- 性能は高くない

- サーバにNICが増えたように見える
- TAPはkvmでもおなじみ

どう使うもの？（概要）

デバイスを開いて、
readするとTUN/TAP向けに来たパケットが読めて、
writeするとTUN/TAPからパケット出て行く



TAPとTUNの違い

- **TAP**
  - L2のデバイス
  - Ethernetがトンネル内を流れる
  - 自分でEthernetヘッダーも書く必要がある
- **TUN**
  - L3のデバイス
  - IPがトンネル内を流れる
  - IPヘッダーから書く必要がある

- **早速どんなコードを書くか見てみましょう**

どのように使うのか(Cでの例)

こちらのドキュメントを引用し、紹介します。

https://www.kernel.org/doc/Documentation/networking/tuntap.txt

```
#include <linux/if.h>
#include <linux/if_tun.h>

int tun_alloc(char *dev) {
        struct ifreq ifr;
        int fd, err;
        if( (fd = open("/dev/net/tun", O_RDWR)) < 0 ) return tun_alloc_old(dev);
続く
```

デバイスを開くと
file descriptorが
返って来る

準備の続き

```
続き        memset(&ifr, 0, sizeof(ifr));

            /* Flags:  IFF_TUN - TUN device (no Ethernet headers)
                     * IFF_TAP - TAP device *
                     * IFF_NO_PI - Do not provide packet information */

           ifr.ifr_flags = IFF_TUN;

           if( *dev ) strncpy(ifr.ifr_name, dev, IFNAMSIZ);
           if( (err = ioctl(fd, TUNSETIFF, (void *) &ifr)) < 0 ){
                     close(fd);
                     return err;
           }
           strcpy(dev, ifr.ifr_name);
           return fd;
}
```

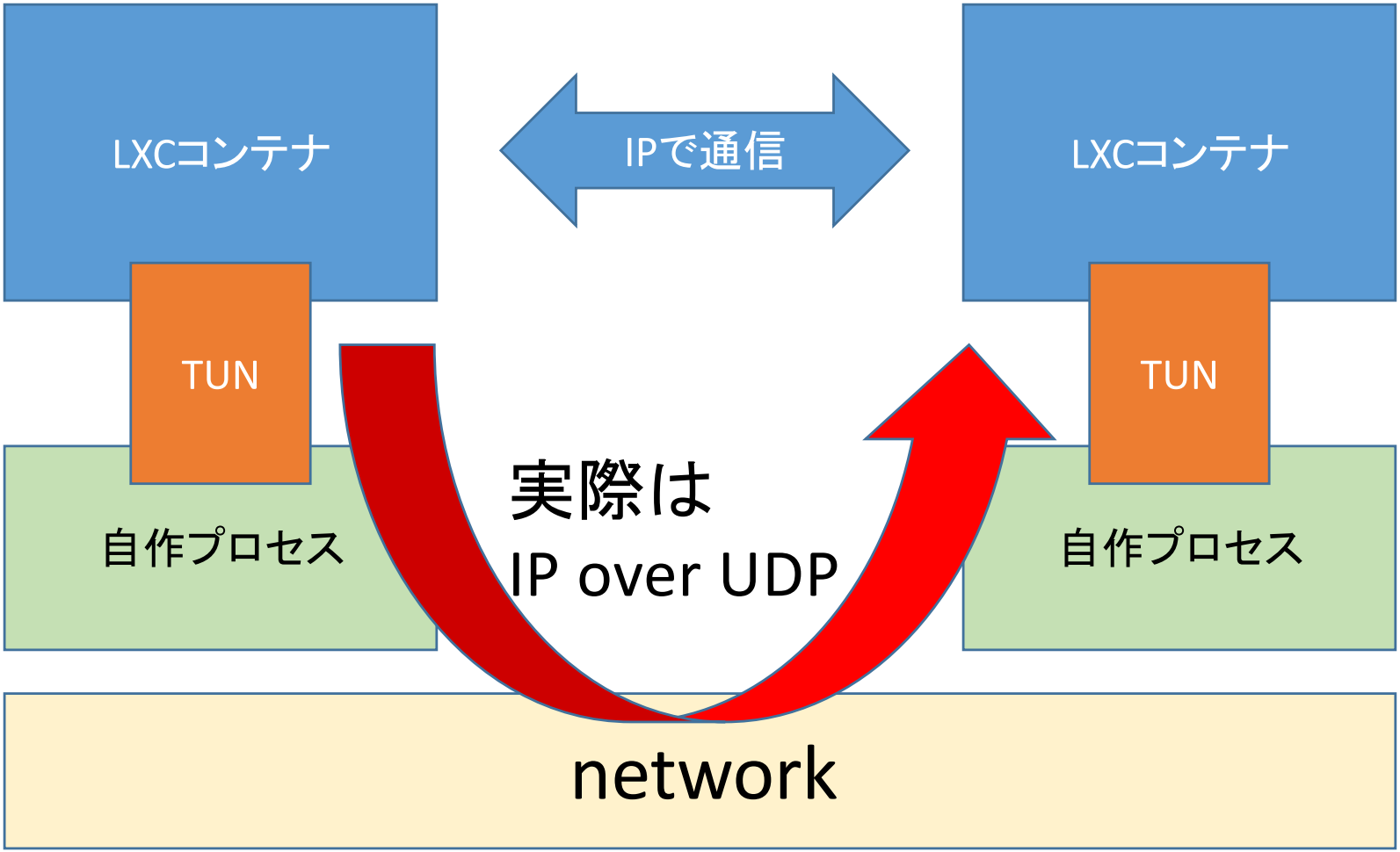
Ifrを初期化

TUNとして設定準備

インターフェイス名を設定準備

準備した設定を反映

UDPで自作overlayネットワーク



パケットの送受信(rubyでの例)

```
sock = UDPSocket.open
sock.bind("0.0.0.0", PORT)
peer = Socket.pack_sockaddr_in(PORT, ADDR)
tunio = tun.to_io
```

続く

パケットの送受信

続き

```
while true  ret = IO::select([tunio,sock])
  ret[0].each do |d|
    if d == tunio
      packet = tunio.read_nonblock(1500) #mtu size
      encaped = Encap.new()
      encaped.user_payload = packet
      sock.send(encaped.to_binary_s,0,ADDR,PORT)
    else
```

続く

パケットの送受信

続き

```
        encaped = Encaped.new()
        d=sock.recv_nonblock(1500)
        encaped.read(d)
        tunio.write(encaped.user_payload)
        tunio.flush
      end
  end
end
```


XFLAG