Tunneling

Communication among the interfaces

Haichao Zhang

Overview

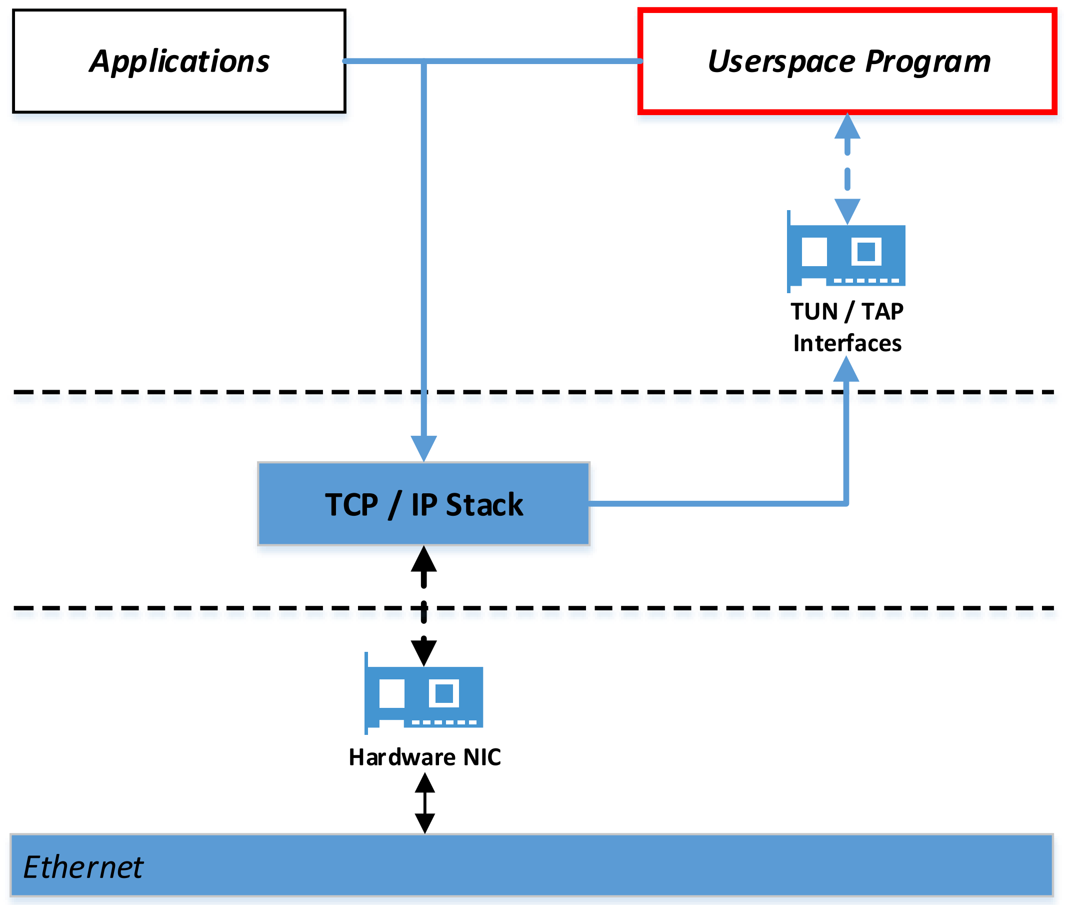
- 1. The TUN/TAP interface Review
- 2. Inter-process communication: fork and pipe

  TCP tunnel and UDP tunnel belongs to different process;

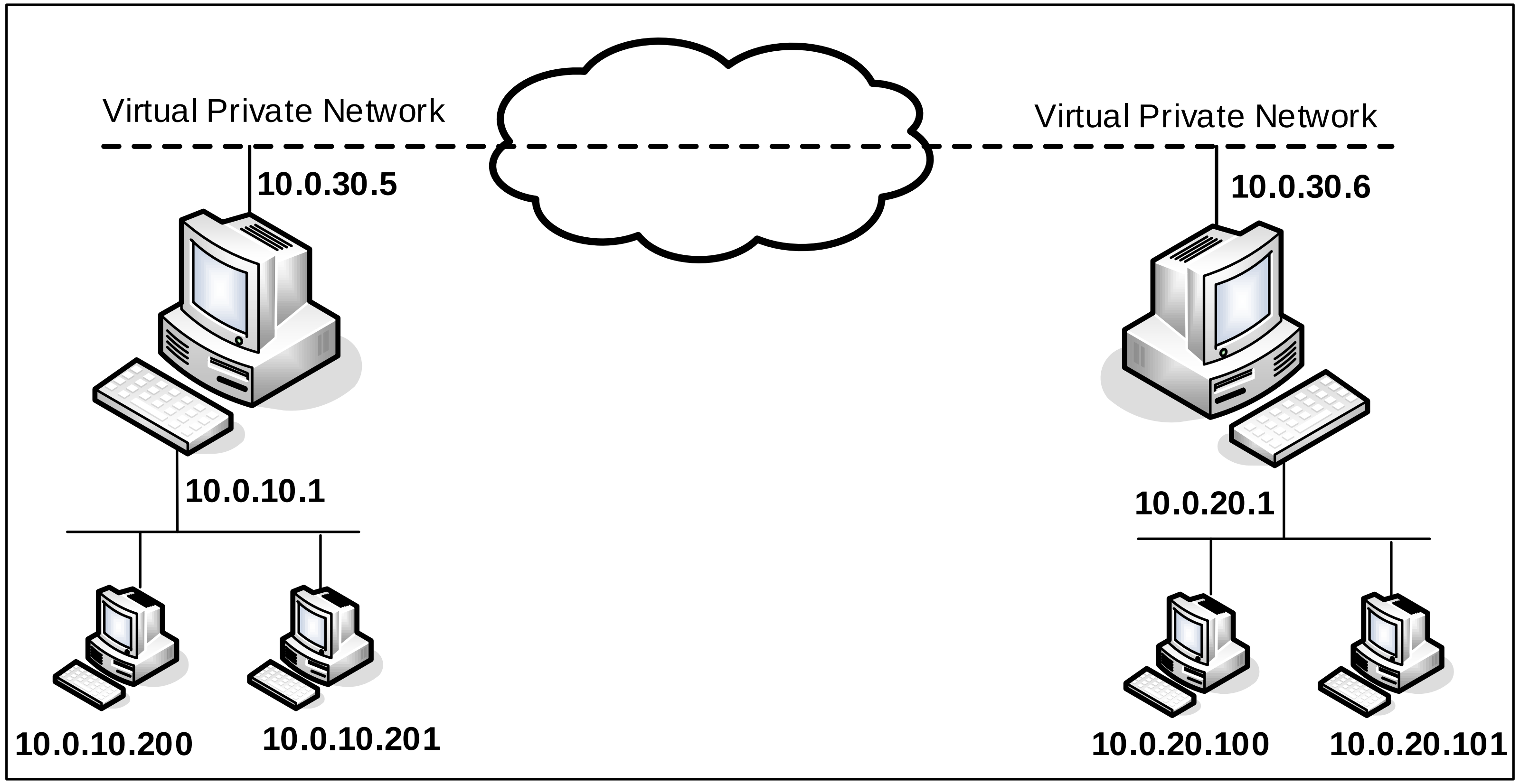
  The two tunnels need to communicate using some technique.

  Extra tip: Select() – read all the coming traffic smoothly

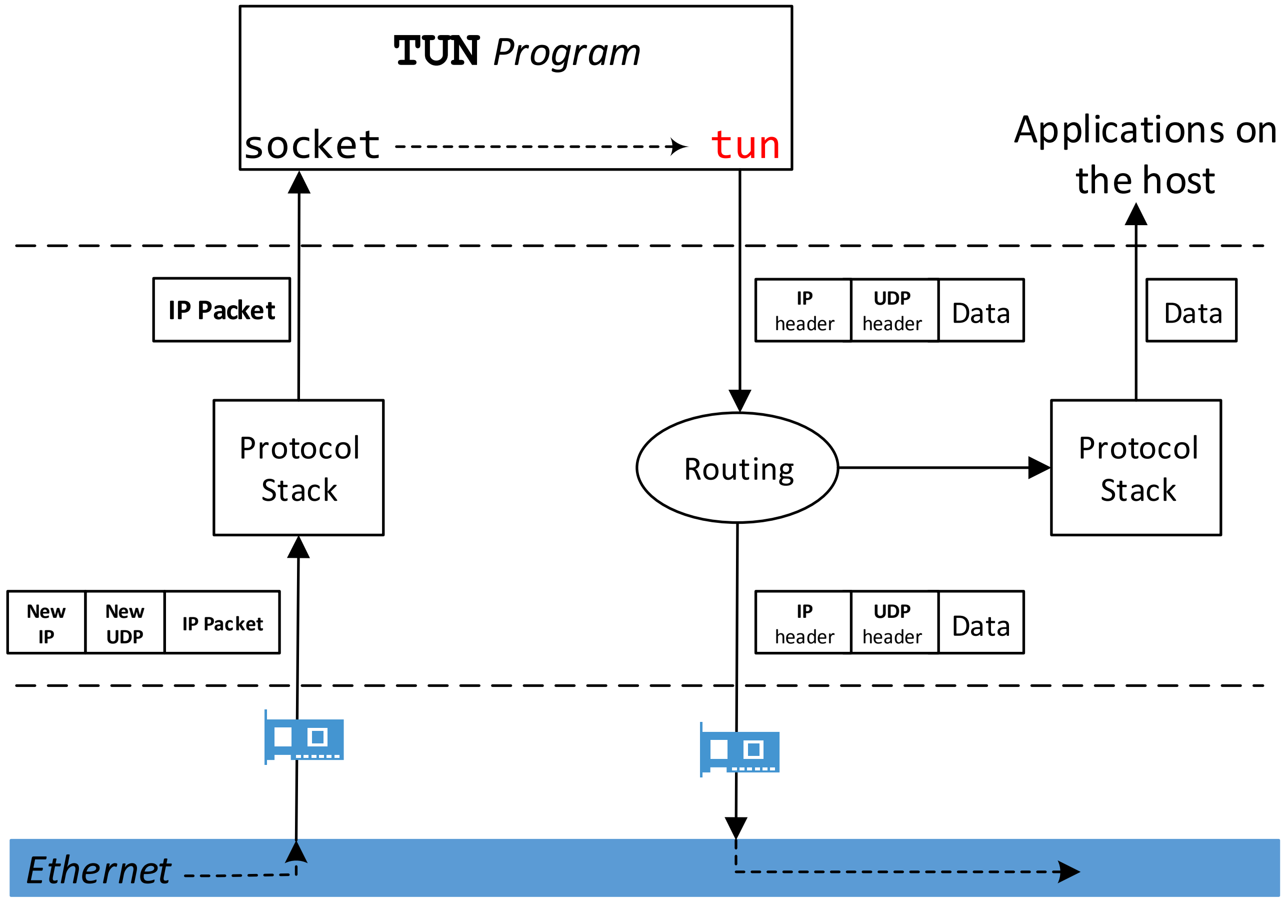
TUN/TAP: direct access from the user space



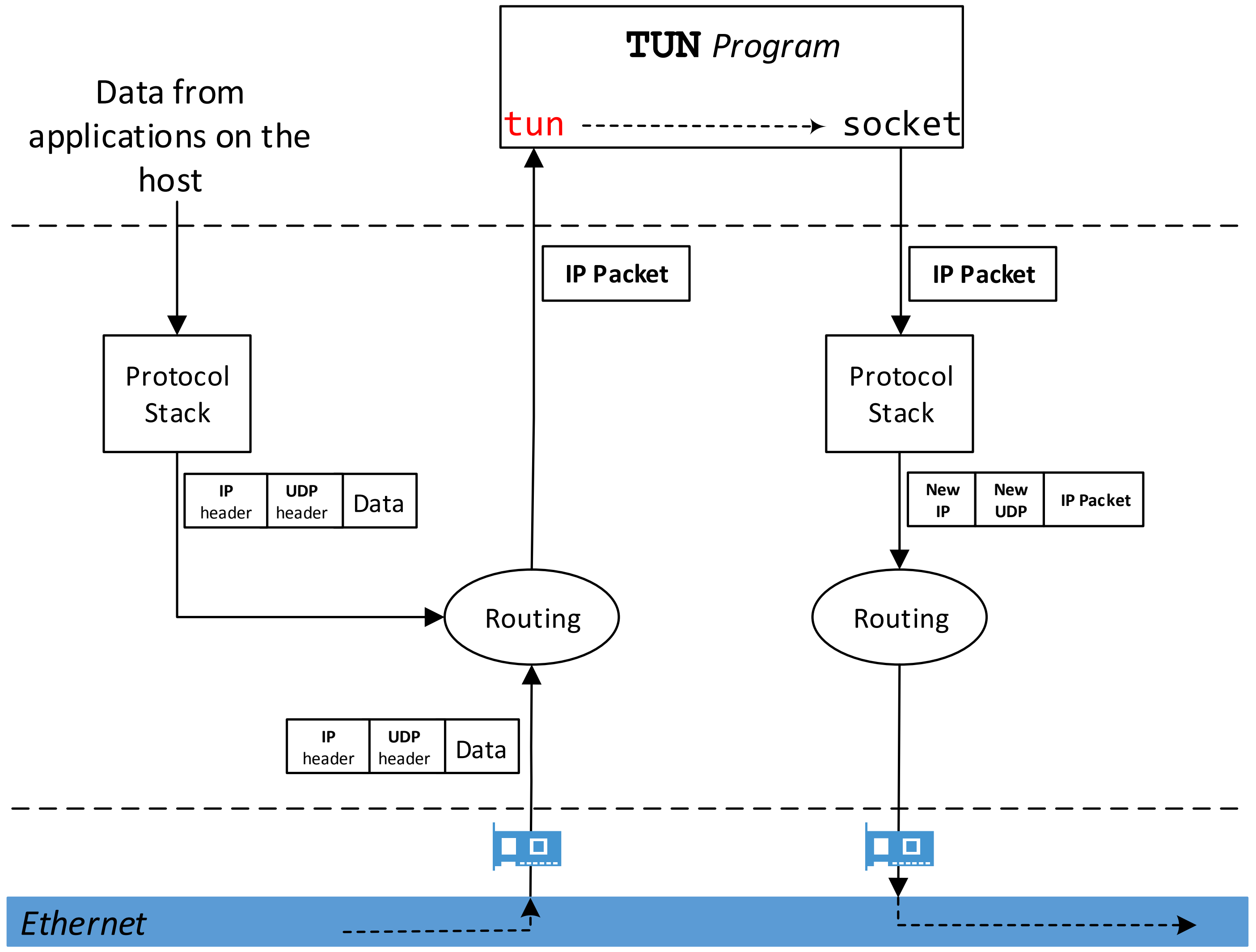
Environment review



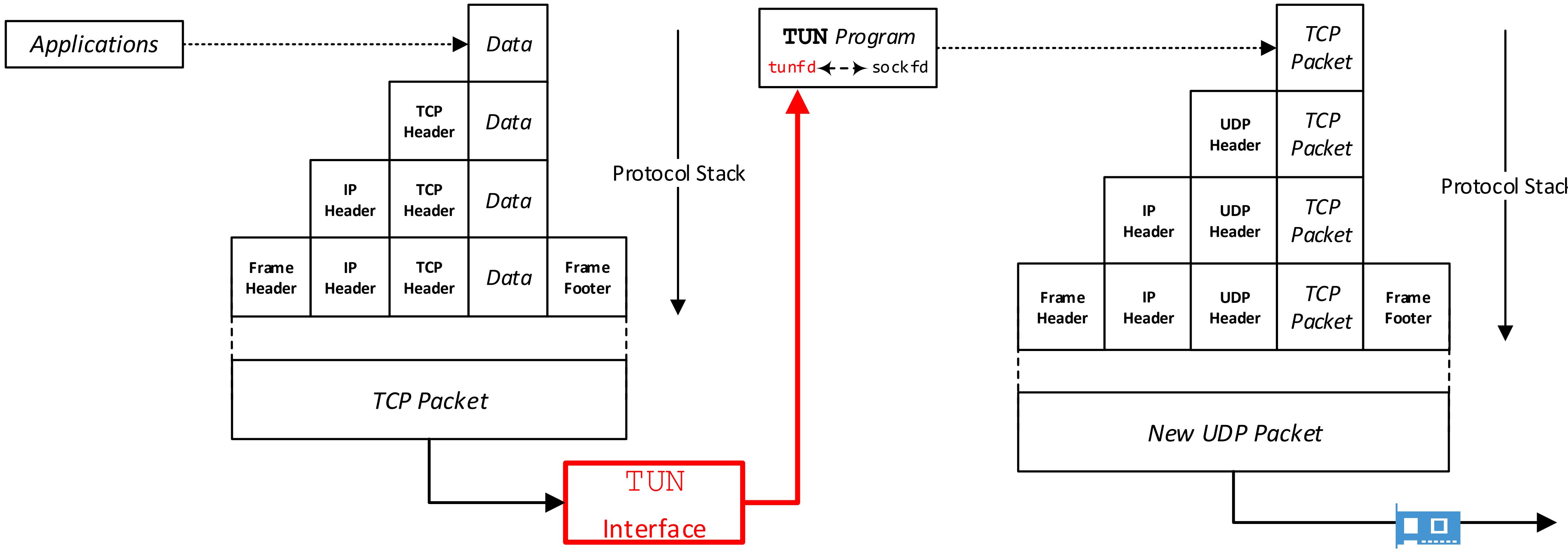
The flow from the sender



Flow at the receiver



Wrap up the packet



TUN/TAP Programming

- No major difference with the socket programming.
- The socket address become the file path instead of the IP/port
- Other details can refer to the simpletun.c or tunproxy.c

Inter Process Communication

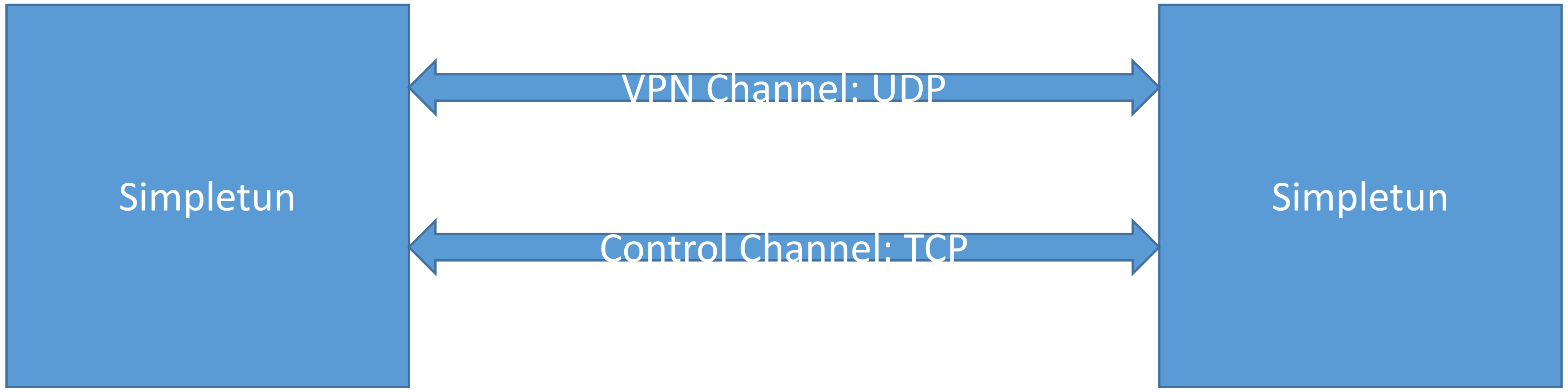
- 1. fork(): set up two tunnels: UDP for the packet tunnel; TCP for the control tunnel
- 2. select()
- 3. pipe()

Fork()

- There would be two processes in the program that handle the UDP and TCP tunnel.

```
int pid = fork();
if(pid >0)
        { handle UDP Tunnel: sendto/recvfrom}
else if(pid ==0)
        { handle TCP Tunnel: sendto/recvfrom}
else
        {handle error}
```

Why we need to fork()?



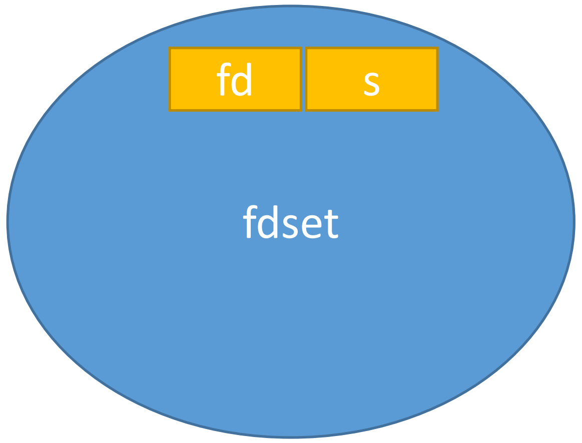
UDP channel: transfer the wrapped packet
TCP channel: control the other side – update key, vi; shutdown; etc.

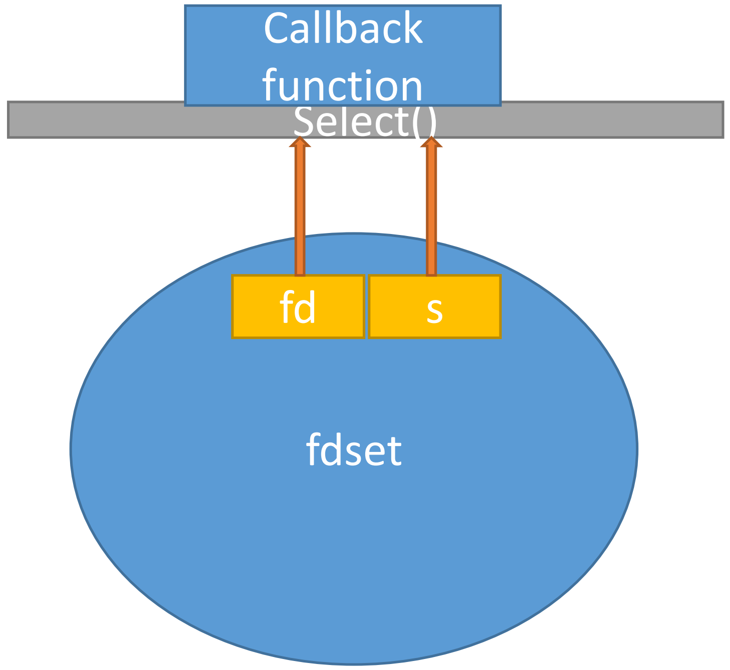
Subsection:How to organize the multiple socket? (you have tun/tap, Internet socket, pipe at the same time)

- Select():

```
fd_set fdset; // claim the variable
FD_ZERO(&fdset); // initialize the fd set
FD_SET(fd, &fdset); // add fd socket into the fdset
FD_SET(s, &fdset); // add s socket into the fdset
if (select(fd+s+1, &fdset,NULL,NULL,NULL) < 0) PERROR("select");
// begin the select operation.
if (FD_ISSET(fd, &fdset)){ // if the coming traffic is from fd socket, then… do something
}
```

General idea for select





Pipe() the communication between the child and the parent

```
char instruction[] = “Go back to sleep.”;
int pipe_fd[2];
pipe2(pipe_fd,O_NONBLOCK);
int pid = fork();
if(pid==0){
        close(pipe_fd[1]);
        ….
        read(pipe_fd[0],instruction,sizeof(instruction) );
        write(pipe_fd[0], instruction,sizeof(instruction) );
        …
} else if(pid>0){
        close(pipe_fd[0]);
        …
        read(pipe_fd[1],instruction,sizeof(instruction) );
        write(pipe_fd[1], instruction,sizeof(instruction) );
        …
}
```