Computer Networks
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: 320301
Date: 2015-09-30
Due: 2015-10-07

Problem Sheet #2

Problem 2.1: *IP over TCP tunnel* (10 points)

The Linux operating system supports so called Tun/Tap network interfaces that emulate a networking interface towards the protocol stack residing in the operating system kernel and that hand packets (or frames) over to a program running in user space. A Tun interface attaches to the IP network layer and exposes IP packets to a user space program while a Tap interface attaches to the data link layer and exposes Ethernet frames. The code shown below demonstrates how to create a Tun or a Tap interface and how to read packets or frames from it.

Extend the program written for the previous assignment so that it reads IP packets from a Tun interface and forwards them over a TCP connection to a remote instance of the program that puts the IP packets via a Tun interface back into the local networking stack and vice versa. The TCP connection should run either over IPv4 or IPv6 (your program should be written to be IP version agnostic). The program must implement the following options:

`-s`	run as a server
`-c <host>`	run as a client connecting to a specified server
`-p <port>`	set the port number to be used
`-i <ifname>`	name of the interface to be created
`-C`	enable check-summing of transmitted packets

Make sure your source code is clearly structured and that the program can be easily extended. In particular, you should be prepared that we will replace TCP with other mechanisms to transport IP packets to a remote server. Make sure your program can handle arbitrarily sized IP packets.

```
/*
 * http://backreference.org/2010/03/26/tuntap-interface-tutorial/
 *
 * TUN=tun0
 * ip link set $TUN up
 * ip addr add 10.0.0.1/24 dev $TUN
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <net/if.h>
#include <linux/if_tun.h>
#include <sys/ioctl.h>

/*
 * Create a tun/tap interface. Return the interface name in dev which
 * should be at least of size IFNAMSIZ. The flags control whether a
 * TUN (layer 3) or TAP (layer 2) interface is created. The function
 * returns -1 if system calls fail.
 */

int tun_alloc(char *dev, size_t dev_len, int flags)
{
    struct ifreq ifr;
    int fd, err;
    const char *clonedev = "/dev/net/tun";

    /*
     * Open the clone device that is used to create TUN/TAP
     * interfaces.
     */

    fd = open(clonedev , O_RDWR);
    if (fd < 0) {
```

```
        perror("open(\"/dev/net/tun\")");
        return fd;
    }

    /*
     * Initialize a structure used to request the creation of a
     * TUN/TAP interface. If the caller requests a specific interface
     * name, copy the name to the interface request structure.
     */

    memset(&ifr, 0, sizeof(ifr));
    ifr.ifr_flags = flags;
    if (*dev) {
        strncpy(ifr.ifr_name, dev, IFNAMSIZ);
    }

    /*
     * Try to allocate the interface using an ioctl on the clone
     * device.
     */

    err = ioctl(fd, TUNSETIFF, (void *) &ifr);
    if (err < 0 ) {
        perror("ioctl(TUNSETIFF)");
        (void) close(fd);
        return err;
    }

    /*
     * Copy the interface name that got allocated into the
     * dev buffer (if it is large enough).
     */

    if (strlen(ifr.ifr_name) >= dev_len) {
        errno = EINVAL;
        perror("interface device name string too short");
        (void) close(fd);
        return -1;
    }

    strncpy(dev, ifr.ifr_name, dev_len);
    return fd;
}

int main(int argc, char *argv[])
{
    int fd = 0;
    int flags = IFF_TUN;     /* layer three */
    // int flags = IFF_TAP;        /* layer two */
    char if_name[IFNAMSIZ] = "";

    char buf[65535];
    ssize_t len;

    fd = tun_alloc(if_name, sizeof(if_name), flags | IFF_NO_PI);
    if (fd < 0) {
        fprintf(stderr, "Error creating tun/tap interface %s!\n", if_name);
        exit(1);
    }

    printf("tun/tap interface '%s' created\n", if_name);

    while (1) {
        len = read(fd, buf, sizeof(buf));
        if (len == 0) {
            break;
        }
        if (len < 0) {
            perror("read()");
            break;
        }
        fprintf(stderr, "[%d]\n", (int) len);
    }

    (void) close(fd);
    return 0;
}
```