Networked Systems
ETH Zürich — seit 2015

Communication Networks

Prof. Laurent Vanbever

Exercise 13 – Applications

13.1 DNS

For this and the following task you will have to change the DNS resolver your VM is using. To do that execute the following command (this change will be lost on reboot):

```
resolvectl dns enp1s0 192.168.122.98
```

Now answer the following questions using `dig(1)` on your VM:

a) What is the IP address of the web server at `www.lan`? Which group hosts the web server?

b) What is the name of the default gateway? The command to inspect the routing table is `ip route`.

c) If you resolve the name from the previous subtask, what do you get? Is that to be expected?

d) Resolve the name `self.lan`. What IP address is it pointing to? Can you explain what is happening here?

13.2 HTTP

We know interrogate a web server on three levels: Once directly using `telnet(1)`, once using a command line utility called `curl` and finally via the web browser running on your local system.

a) What port is HTTP using? You can consult the file `/etc/services` on your VM. Using this file can you work out what port `ntp` is using? What protocol is it using?

b) Now connect to the web server using `telnet(1)`. Every time you press return, `telnet(1)` sends a carriage return (*CR*) and a line feed (*LF*) to the remote host. Type your request for the root directory like this:

```
GET / HTTP/1.0
```

and press enter twice to first end the current line and then to send an empty line.

c) Can you learn what software the web server is using?

d) Now use the `curl(1)` tool to request `www.lan` like this:

```
curl www.lan
```

Is the result different from the first task? Can you work out why?

e) Using `ssh(1)` you can connect the web browser on your computer to the web server like this:

```
ssh -L 127.0.0.1:8080:<web-server-name>:<www-port> -p <2000+X> student@duvel.ethz.ch
```

X your group number. Now navigate your browser to `127.0.0.1:8080` and look at the shown web page.

f) Your friend says there is a secret web server called `www2.lan` running on the same host as `www.lan`, but when you check with `dig(1)` there is no IP for `www2.lan`. Can you find a way to access the secret server?

13.3 (Extra) IP Sockets – Establishing a TCP connection with C

In this task you will establish a TCP connection to a server using C. All the tools you need (a C compiler and the appropriate headers) are already installed on you VM. To make sure you are not interfering with other members of your group, work inside a directory named after your `nethz` username. You can create a directory using `mkdir(1)` and switch to it with `cd`:

```
mkdir <nethz-name>
cd <nethz-name>
```

Test your setup To test the setup use the file `example.c` from the course web site. You can either download it on your computer and then upload it with `scp(1)` to your VM, or download it directly to your VM using `curl(1)`. You can now compile it using:

```
cc example.c
```

and execute the resulting file using:

```
./a.out
```

You should see the content of `example.c` in your terminal.

Useful functions, structs etc. *Hint:* As shown in the presentation at the beginning of the session, text followed by a number in parentheses like `socket(2)` are references to a manual page. `socket` is the name of the page and 2 is the section number. To read the page use the `man(1)` command. For example to read the manual of `socket(2)` do:

```
man 2 socket
```

Type *q* to exit the man page again.

The following functions, structs and man pages might be helpful to implement your TCP connection with C:

- **`struct sockaddr_in`** Holds an IP address and a port. Can be used as an argument to `connect(2)`. The structure layout is documented in `ip(7)`. Recall that a dot (`.`) is used to access `struct` elements and `->` if the variable is a pointer to a `struct`.
- **`inet_aton(3)`** Converts a string to `struct in_addr`.
- **`htons(3)`** Converts a 16-bit integer from *host* to *network* byte order. As your VMs use Little Endian, and network byte order is Big Endian you will have to use this to convert the destination port accordingly.
- **`socket(2)`** Used to obtain a descriptor. `socket(2)` is responsible for allocating descriptors for local (`AF_UNIX`) connections and remote connections (`AF_INET`, `AF_INET6`) among others. To select a protocol one has to use one of the `SOCK_*` flags. `SOCK_STREAM` is used for TCP, `SOCK_DGRAM` for UDP.
- **`connect(2)`** This connects a file descriptor obtained from `socket(2)` to a remote host.
- **`write(2)`** Write data to a file descriptor.
- **`read(2)`** Read data from a file descriptor.
- **`close(2)`** Closes a file descriptor (and the connection with it).

The following functions/built-ins can be of use as well:

- **`err(1)`** Exit the program with an error message.
- **`sizeof`** C built-in which returns the size of the argument in bytes as `size_t`.
- **`fwrite(3)`** Writes a buffer to a file. `stdin`, `stdout` and `stderr` are `FILE *` which read from standard input and write to standard output/error respectively.

Implementation As a starting point for your implementation we provide a skeleton file on the course website (`skeleton.c`), which already contains the relevant includes. Be careful to check the return values of the system calls. `err(1)` exits the process and prints a message according to the value of `errno(3)`.

Try to complete the skeleton file following the hints in the comments. Are you able to establish the TCP connection?

Debugging You can see all outgoing TCP connections using the `ss(8)` command like this:

```
ss -t
```

Next steps Here are some ideas on how to extend your program:

- Use `getaddrinfo(3)` to convert `www.lan` to the corresponding IP address.
- Switch to use `HTTP/1.1` instead. For that you will need to include a `Host` header.
- Allow the user to specify a hostname on the command line.