Computernetzwerke
Hochschule für Technik und Wirtschaft – HTW Berlin
Sebastian Bauer
Wintersemester 2022/2023
1. Laboraufgabe vom 5. Oktober 2022
Abgabe bis zum 28. Oktober 2022 via PDF auf der HTW-Cloud und Moodle

---

Wir setzen in diesem virtuellen Labor Mininet (`http://mininet.org`) ein, wobei wir uns im ersten Labor hauptsächlich auf die Einrichtung des Systems konzentrieren und so auch unsere Linux-Kenntnisse trainieren. Im zweiten Aufgabenblatt beschäftigen wir uns dann direkt mit Mininet. Eventuell gibt es dann auch noch ein drittes.

Das erfolgreiche Lösen der Aufgabe besteht

- im Durcharbeiten und Verstehen eines jeden einzelnen Punktes,
- bei Unklarheiten Fragen bei den GitLab Issues zu stellen,
- die Antworten zu den Fragen in einer Datei zu protokollieren sowie
- den Inhalt der Datei zur HTW-Cloud hochzuladen und mit dem Dozenten zu teilen und ihm darauf Schreibzugriff (aka *Kann bearbeiten*) für Kommentare zu geben.

Da das Laboraufgabenblatt im letzten Semester neu war und die Aufgabenstellung neu entwickelt wurde, können sich hier viele Fehler eingeschlichen haben. Der Dozent bedankt sich für jeden Fehlerbericht. Auch über unklare Textstellen und Verbesserungsvorschläge werden angenommen. Das Aufgabenblatt wird deshalb im Laufe der Veranstaltung aktualisiert.

Abgabe: Die Abgabe des Protokolls (Beantwortung der Fragen) erfolgt über die HTW-Cloud (Anleitung unter `https://anleitungen.rz.htw-berlin.de/de/cloud`). Der Name der abzugebenden Datei lautet aus `CNW2022-Labor1-<Vorname(n)>-<Nachname(n)>.pdf` ohne Leerzeichen. Die Datei wird dann mit dem Dozent geteilt, gerne auch vorab des Abgabetermins. Feedback gibt der Dozent dann direkt im PDF.
Hinweis: Dieses Arbeitsblatt setzt voraus, dass Linux als natives System zur Verfügung steht, da wir aus Gründen der Sicherheit auch noch VirtualBox als sogenannte Sandbox einsetzen. Theoretisch ist es möglich, unter einen bereits emulierten Linux via VirtualBox noch einmal VirtualBox zu starten (*nested* VirtualBox), sofern gewisse Voraussetzung erfüllt sind.[1] Es ist aber auch möglich, dass die Mininet-VM und das emulierte Linux Seite an Seite auf in derselben VirtualBox laufen. Hierbei hilft die Ergänzung des Arbeitsblatts.
Weiterer Hinweis: Zum Beantworten der Fragen sind verschiedene Eingaben im Terminal zu machen. Da dies nicht immer im selben Kontext geschieht, werden der Einfachheit halber verschiedenen Kommandoprompts genutzt. Hierbei steht `$h` für Eingaben im Host, `$m` für Eingaben auf der Mininet-VM und `mininet>` für Eingaben im Mininet-CLI.[2]

---

[1] Dieses Feature wurde in Version 6.1 eingeführt, siehe `https://www.virtualbox.org/wiki/Changelog-6.1`
[2] CLI bedeutet ausgeschrieben Command-Line-Interface

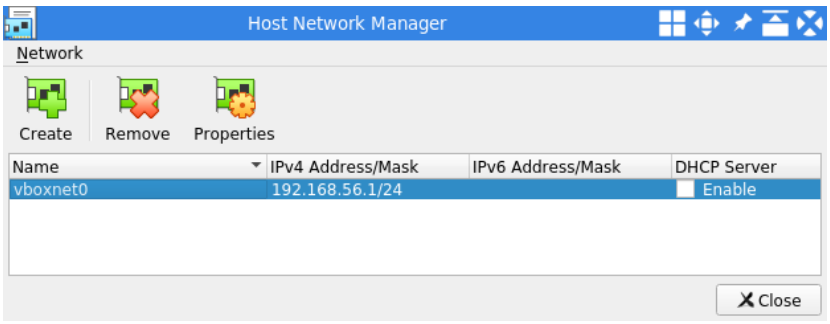

Abbildung 1: Virtuelles Host-only-Netzwerk erstellen. Die Einstellung *DHCP Server* soll bei uns deaktiviert sein.

1. **Mininet kennenlernen**. Informiere Dich auf http://mininet.org über Nutzen und Funktion von Mininet. Nützlich ist auch die FAQ auf https://github.com/mininet/mininet/wiki/FAQ.
   (a) Was leistet Mininet?
   (b) Was ist ein Container?
   (c) Was ist ein Container-Orchestrierung-System?
   (d) Wie lautet das Shell-Kommando, um ein Mininet-System zu starten?
   (e) In welcher Programmiersprache kann man eigene Mininet-spezifische Skripte schreiben (wir werden das auch im nächsten Aufgabenblatt tun)?

2. **Mininet installieren**. Installiere Mininet auf Deinem Rechner. Möglichkeiten hierzu werden auf http://mininet.org/download/ angegeben. Der Dozent empfiehlt der Einfachheit halber die Option 1, also die Nutzung einer separaten virtuellen Maschine zusammen mit dem Festplattenimage https://github.com/mininet/mininet/releases/download/2.3.0/mininet-2.3.0-210211-ubuntu-20.04.1-legacy-server-amd64-ovf.zip. Die folgenden Teilaufgaben beziehen sich auf diese Option. Hierbei ist der Host immer der Rechner, auf dem VirtualBox läuft und von dem der Dozent annimmt, dass dieser ein nativer Linux-Rechner ist. Die virtuelle Instanz von VirtualBox ist dann die Mininet-VM bzw. der Gast. Sollte Linux über eine VM laufen, so ist an dieser Stelle auf die Ergänzung des Arbeitsblatts verwiesen.
   (a) Lies Dir die Schritte zum Einrichten von Mininet auf http://mininet.org/vm-setup-notes/ durch. Richte die virtuelle Maschine anhand dieser Schritte ein. Es wird im Folgendem angenommen, dass VirtualBox genutzt wird und eine Version von Mininet installiert ist, die auf Ubuntu 20.04 basiert.
   (b) Wir lautet der Login und das Passwort, um sich in die virtuelle Instanz einzuloggen?
   (c) Wir möchten, dass wir vom Host-Computer auf die eben eingerichtete Maschine zugreifen können. Dafür eignet sich ein sogenanntes *Host-only*-Netzwerk, das VirtualBox erstellen kann und nur eine Verbindung zwischen Host und Gast herstellt, nicht aber zur Außenwelt. Hierfür fügt VirtualBox einen *virtuellen Netzwerkadapter* zum System hinzu.
   Auch wenn der Überblick etwas veraltet ist, ist die Netzwerkkonfiguration ist recht gut auf https://www.thomas-krenn.com/de/wiki/Netzwerkkonfiguration_in_VirtualBox zusammengefasst. Verschaffe Dir einen Überblick und stelle das Netz der Mininet-VM in den Einstellungen von VirtualBox auf *Host-only Adapter*. Zuvor muss ein virtuelles Host-Netzwerk innerhalb von VirtualBox über den Menüpunkt *File > Host Network Manager...* erstellt werden. Abbildung 1 zeigt das Fenster. Das Häkchen vor *Enable* soll bei uns deaktiviert sein. Es wird daraufhin ein virtueller Netzwerkadapter auf dem Host erstellt, dessen Name man spezifiziert hat und im Beispiel `vboxnet0` lautet. Führt man

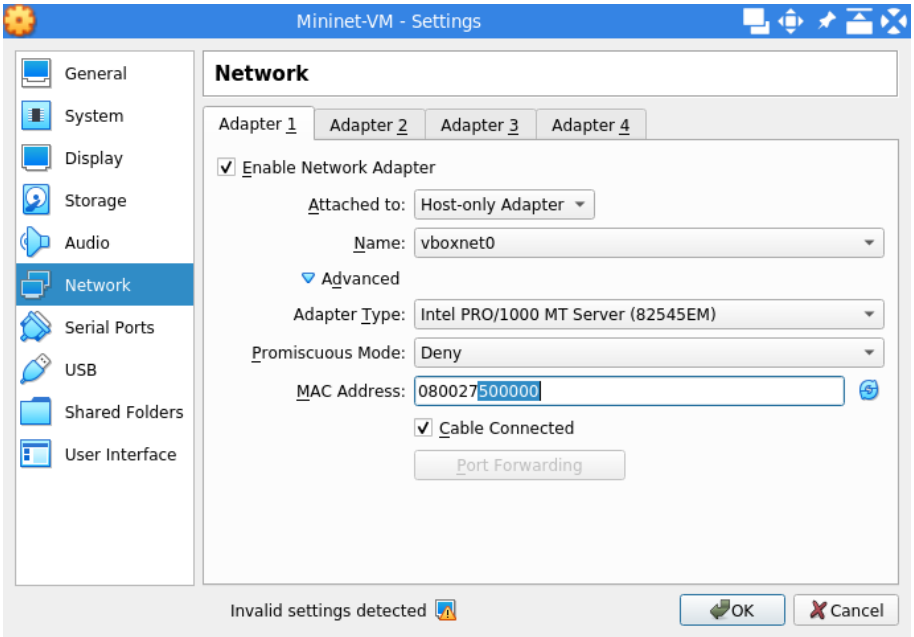

Abbildung 2: Netzwerkeinstellungen für die virtuelle Mininet-Maschine.

Der Chooser *Attached to* muss auf *Host-only Adapter* stehen. Als *MAC Address* wählst Du: `080027XXXXXX`, wobei Du `XXXXXX` durch Deine Matrikelnummer ersetzt. In dem Bild ist der variable Teil markiert.

```
h$ ip a
[...]
12: vboxnet0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group \
                                    default qlen 1000
link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
```

auf dem Host aus, dann kann man sich überzeugen, ob dieser tatsächlich angelegt worden ist. In dem Fall ist also das Interface 12 unser virtuelles. Anstelle `xx:xx:xx:xx:xx:xx` sollten jedoch andere Zahlen zu sehen sein. Wie lauten Sie?

(d) Was hat es mit der Zahl auf sich? Notiere dieselbe Art von Information von einem anderen Netzwerkadapter in Deinem System.

(e) Was stellst Du weiterhin fest, wenn Du die Einträge zwischen einem aktiven, realen Netzwerkadapter (z. B. WLAN) und diesen betrachtest? Welche Informationen fehlen beispielsweise?

(f) Konfiguriere nun die Mininet-VM, das eben erstellte Netz zu nutzen. Abbildung 2 zeigt die Einstellungen. Der Eintrag *Attached to* muss auf *Host-only Adapter* stehen und bei *Name* der Name des eben erstellten Netzes. Als *MAC Address* trage bitte `080027XXXXXX` ein, wobei für `XXXXXX` die eigene Matrikelnummer zu wählen ist. Halte diese Informationen in einem Screenshot fest.

(g) Starte jetzt die virtuelle Maschine. Das System sollte nun in die Mininet-Instanz booten und letztlich einen Login erwarten. Da wir die dynamische Adressvergabe ausgeschaltet haben, kann der Vorgang jedoch eine Weile dauern (ein bis zwei Minuten) bis die Login-Abfrage erscheint.

(h) Falls Dich nach dem Einloggen die auf Englisch eingestellte Tastatur stört, nutze `sudo dpkg-reconfigure keyboard-configuration`.

3. **Netzwerkverbindung zur Mininet-Instanz vom Host herstellen**. Wir könnten jetzt das Interface der virtuellen Maschine nutzen, um die Aufgaben zu erledigen. Über das Terminal des

emulierten Grafikadapters ist dies ist allerdings nicht sehr komfortabel, da wir Ausgaben protokollieren wollen. Also werden wir jetzt die virtuelle Netzwerkverbindung vom Host zur Mininet-Instanz einrichten und dabei die Gelegenheit nutzen, um ein paar grundlegende Dinge zur Netzwerkkonfiguration kennenzulernen.

(a) Zeichne das Netzwerkschema für die uns angedachte Konfiguration.

1. Es gibt den Host-Rechner.
2. Er enthält u. a. einen (virtuellen) Netzwerkadapter.
3. Es gibt die virtuelle Mininet-Instanz.
4. Sie enthält einen virtuellen Netzwerkadapter.
5. Beide Netzwerkadapter sind verbunden.

Die beiden Rechner können hierbei als physisch getrennt betrachtet werden.

(b) Wähle die IP-Adressen und Subnetmasken und notiere sie in dem Bild. Nutze die CIDR-Notation. Achtung: Die IP-Adressen auf Host-Seite sowie die Subnetmaske wird in VirtualBox spezifiziert (im Host-Network-Manager). Die IP-Adresse des Netzwerkadapters auf Seite der virtuellen Maschine ist selbst sinnvoll festzulegen. Die IP-Adresse muss im selben Subnetz liegen.

(c) Wieso überhaupt müssen sich beide IP-Adresse im selben Subnetz befinden?

(d) Logge Dich in die virtuelle Instanz mit den in Aufgabe 2b) notierten Credentials ein.

(e) Erhalte nun dauerhaft Root-Rechte via

```
m$ sudo su
```

(f) Wie lautet die MAC-Adresse des virtuellen Netzwerkadapters auf Seite des Gastes? Nutze `ip a` (die modernere Variante) oder klassisch `ifconfig`. Notiere sie.

(g) Notiere den Namen des Netzwerkinterfaces (nutze die MAC-Adresse zur eindeutigen Identifikation). Im Folgenden wird dieser mit `<if>` abgekürzt.

(h) Versichere Dich, dass dem Netzwerkadapter keine IP-Adresse zugewiesen ist. Falls doch, musst Du in den VirtualBox-Netzwerkeinstellungen die Option *Enable* deaktivieren, wie in Abbildung 1 zu sehen. Möglicherweise musst Du die virtuelle Maschine und VirtualBox noch einmal neu starten.

(i) Da wir selbst die Konfiguration der Netzwerkadresse durchführen möchten. müssen wir zunächst die automatische Konfiguration via DHCP deaktivieren, da dies für die Netzwerkschnittstelle per Vorgabe so eingestellt ist. Neuere Ubuntu-Versionen nutzen für die Netzwerkkonfiguration `netplan`, das über die Datei `/etc/netplan/01-netcfg.yaml` konfiguriert wird. Ändere dort den Eintrag von `dhcp4: yes` auf `dhcp4: no` z.B. mit dem Editor `nano`. Führe anschließend

```
m$ netplan apply
```

aus.

(j) Wir legen jetzt die IP-Adresse auf dem Gast (die Mininet-VM) für den virtuellen Netzwerkadapter fest. Wir nutzen hierfür `ip`[3]. Auf dem Gast geben wir jetzt ein:

```
m$ ip addr add <guest-cidr> dev <if>
```

Hiermit wird dem Interface `<if>`, dessen Name vorher ermittelt wurde, die unter `<guest-cidr>` angegebene Adresse zugewiesen. Diese Adresse haben wir in Teilaufgabe b) festgelegt.

Eine weitere nützliche Übersicht zu `ip` gibt es hier: https://www.thomas-krenn.com/de/wiki/Linux_ip_Kommando.

[3]Das Programm *ifconfig* erfüllte auch den Zweck, ist von vielen Linux-Distributionen bereits vor 20 Jahren als veraltet eingestuft. Trotzdem erfreut es sich heute noch größter Beliebtheit, inkl. beim Dozenten. Wir nutzen hier dennoch `ip` aus dem Paket `iproute2`. Einen nützlichen Vergleich gibt es z. B. hier: https://www.linux.com/training-tutorials/replacing-ifconfig-ip/.

(k) Verifiziere nun, dass die Adresse wirklich gesetzt wurde:

```
m$ ip a
```

Ergibt die Ausgabe Sinn?

(l) Anschließend pingen wir vom Gastrechner zum Host-Rechner. Auf der **Mininet-VM** geben wir ein:

```
m$ ping <host-ipaddr>
```

Das Programm via `Strg+C` bzw. `Ctrl+C` abgebrochen werden. Hat der Ping funktioniert, kannst Du zum nächsten Schritt übergehen. Ansonsten versuche einen Fehler zu finden oder bitte in der Lehrveranstaltung oder bei Mitstudierenden um Hilfe.

(m) Nun pingen wir in die andere Richtung, d. h., vom Host-Rechner zum Gastrechner. Auf dem Host geben wir ein:

```
h$ ping <guest-ipaddr>
```

Da wir jetzt auf dem Host sind, können wir die Ausgabe leicht per Copy'n'Paste protokollieren. Die ersten fünf Zeilen einer erfolgreichen Ausgabe sind als Lösung dieser Teilaufgabe abzugeben.

4. **SSH-Zugang konfigurieren**. In dieser Aufgabe wollen wir einen komfortablen SSH-Zugang zur Mininet-VM einrichten.

(a) Wir loggen uns per SSH vom Host-Rechner auf dem Gastrechner ein, das war unser eigentliches Ziel:

```
h$ ssh mininet@<guest-ipaddr>
```

Nach Eingabe des Passworts können wir über das Host-Terminal auf dem Gastrechner zugreifen.

(b) Die Eingabe des Passworts ist lästig und wir werden deshalb eine Public-Key-basierte Methode zur Authentifizierung einrichten. Wir legen im Home-Verzeichnis auf der Mininet-VM ein Verzeichnis mit Namen `.ssh` via noch bestehender SSH-Verbindung an:

```
m$ mkdir ~/.ssh
```

Danach schließen die Verbindung zum Gastrechner gleich wieder.

```
m$ exit
```

(oder kurz `Strg+D` bzw `Ctrl+D`)

(c) Überzeuge Dich im Terminal von Mininet-VM (das von VirtualBox angezeigt wird), dass das Verzeichnis `.ssh` im Home nun vorhanden ist:

```
m$ ls -la ~
```

(d) Wir machen den lokalen öffentlichen Schlüssels des eigenen Benutzers auf dem Host, dem Gast bekannt. Zunächst überprüfen wir, ob auf dem Host für uns bereits ein Schlüsselpaar vorliegt, indem wir die öffentlichen Schlüssel auflisten:

```
h$ ls ~/.ssh/*.pub
```

Ist dies nicht der Fall, so erzeugen wir ein neues Schlüsselpaar mit `ssh-keygen`. Überprüfe im Anschluss erneut, ob der öffentliche Schlüssel nun vorliegt.

(e) Jetzt kopieren wir den öffentlichen Schlüssel zur Mininet-VM und legitimieren einen passwortlosen Zugriff, für diejenigen, die einen passenden privaten Schlüssel zu diesem Schlüssel haben (das ist man im Idealfall nur höchst persönlich). Das geht z. B. mit folgendem Einzeiler

```
h$ cat ~/.ssh/*.pub | ssh mininet@<guest-ipaddr> "tee -a .ssh/authorized_keys"
```

Hierbei müssen wir noch einmal das Passwort des Nutzers `mininet` eingeben. Durch `tee` wird der öffentliche Schlüssel dabei auf die Standardausgabe und in die angegebene Datei ausgegeben.

(f) Wir loggen uns erneut vom Host zum Gast ein. Dieses Mal sollte keine Passwortabfrage erfolgen. Warum?

(g) Warum ist die schlüsselbasierte Authentifizierung in der Regel wesentlich sicherer als eine Passwort-basierte Authentifizierung?

(h) Die meiste Zeit werden wir unter dem Benutzer `root` arbeiten wollen. Wir nutzen die virtuelle Maschine nur für diesen Zweck, also ist das akzeptabel. Finde einen Weg, wie man sich per ssh gleich als `root` anmelden kann. Ob dies sicherheitstechnisch sinnvoll ist, sei an dieser Stelle einmal offen gelassen. Beachte, dass eine Passwort-Authentifizierung für `root` explizit ausgeschaltet ist.

5. **Netzwerkkonfiguration auf der Mininet-VM persistieren**. Wir erinnern uns, dass wir in Aufgabe 3 die Netzwerkkonfiguration manuell eingerichtet haben. Leider geht diese nach einem Reboot der virtuellen Maschine verloren. Da wir das nicht wollen, werden wir jetzt die Netzwerkkonfiguration persistieren. Das von der Mininet-VM benutzte Ubuntu nutzt wie weiter oben festgestellt Netplan als Frontend für die Netzwerkkonfiguration.[4] Netplan liest eine in YAML[5] geschriebene Konfigurationsdatei und schreibt beim Startvorgang weitere Konfigurationsdateien, die dann von Netzwerkmanager-Tool eingelesen wird.[6]

   Da wir im Folgenden noch einmal die Netzwerkkonfiguration verändern, was schief gehen kann, ist es empfohlen das Ganze über die von VirtualBox angezeigte Konsole durchzuführen, also nicht `ssh` zu benutzen. Alle folgenden Kommandos führen wir mit Root-Rechten auf der **Mininet-VM** (nicht auf dem Host!) aus. Gib ggf. vorab `sudo su` ein. Im Kommandoprompt ist jederzeit ersichtlich, unter welchen Nutzer man sich gerade bewegt.

   (a) Studiere kurz die Webseite von Netplan, um herauszufinden, wie das Dateiformat lautet. Hilfreich ist auch das Manual von `netplan`:

   ```
   m$ man netplan
   ```

   (b) Mit neuen Kenntnissen ausgerüstet, modifiziere jetzt `/etc/netplan/01-netcfg.yaml` entsprechend der vorher erarbeiten Netzwerkkonfiguration. Rufe dabei `netplan apply` auf, um die Konfiguration anzuwenden.

   (c) Wie lautet jetzt die Ausgabe von

   ```
   m$ ip a
   ```

   Erscheint sie korrekt? Wenn nein, untersuche die Datei erneut auf Fehler, behebe sie und starte das Netzwerk erneut wie in diesem Schritt. Wiederhole den Vorgang solange, bis alles korrekt erscheint.

   (d) Logge Dich erneut per SSH vom Host in den Gast ein. Funktioniert es nicht, dann gehe zurück zu b).

   (e) Fahre die virtuelle Maschine runter.

   ```
   m$ poweroff
   ```

   (f) Wir machen jetzt noch ein Snapshot vom aktuellen Zustand der Mininet-VM (Menüpunkt *Machine* > *Tools* > *Snapshot* und dann *Take*). Auf diese Weise können wir bei Problemen schnell wieder auf einem funktionierenden Zustand zurück.

   (g) Starte die virtuelle Maschine erneut. Der Startvorgang sollte nun deutlich schneller gehen.

   (h) Warum überhaupt geht der Startvorgang nun schneller?

   (i) Logge Dich nun wieder per ssh vom Host-Rechner auf dem Gastrechner ein.

   als Nutzer *root* erledigen.

[4] https://netplan.io/

[5] https://yaml.org/

[6] Ob dieser von neueren Ubuntu-Versionen gewählte weg so sinnvoll ist, sei dahin gestellt, dennoch müssen wir damit an dieser Stelle umgehen.

6. **Mininet testen**. In der letzten Aufgabe dieses Arbeitsblattes testen wir, ob die Installation von Mininet auf dem bereitgestellten Image funktionstüchtig ist. Wenn noch nicht getan, loggen wir uns vom Host-Rechner per `ssh` auf dem Gastrechner ein und erhalten gleich Root-Rechte. Desweiteren starten wir auch Wireshark. Wir geben entweder

   ```
   h$ ssh -Y mininet@<guest-ipaddr>
   m$ sudo -E wireshark &
   m$ sudo su
   ```

   ein[7] oder, wer Aufgabe 4h) absolviert hat,

   ```
   h$ ssh -Y root@<guest-ipaddr>
   m$ wireshark &
   ```

   Die Option `-Y` ist an der Stelle wichtig, da sie das sogenannte X11-Forwarding einschaltet, sodass GUI-Programme auf dem Remote-Host über den lokalen Rechner „ferngesteuert“ werden können. Wir nutzen nämlich gleich Wireshark, den Packet-Sniffer.

   Des Weiteren bewirkt das `&` am Ende einer Shell-Zeile, dass das zu startende Programm in den Hintergrund gelegt wird und man weitere Kommandos im Terminal eingeben kann.

   (a) Für Wireshark bestätigen wir nun zahlreiche Warnungen, die erscheinen, weil wir Wireshark normalerweise nicht unter `root` laufen lassen sollten. Wir machen es dennoch, weil unsere Maschine nur eine Sandbox ist.

   (b) Wir starten nun ein Capture in Wireshark auf dem Interface *Loopback: lo*, also auf dem lokalen Loopback-Interfaces. Hierzu klicken wir genau dieses Interface in der Liste an (ggf. scrollen) und betätigen den grünen Haiflossen-Button.

   (c) Jetzt erscheinen schon allerhand Pakete, die für uns allerdings nicht sehr interessant sind. Interessant ist die OpenFlow-Kommunikation und nur die möchten wir sehen. Hierzu geben wir bei Filter `openflow_v1` oder ähnliches ein, das Texteingabefeld sollte jetzt grün erscheinen. Mache nun einen Screenshot vom Wireshark-Fenster fürs Protokoll. Im Anschluss können wir das Fenster erst einmal beiseite legen.

   (d) Das Programm Mininet wird über das Kommando `mn` verfügbar gemacht. Wir geben in die Mininet-VM ein:

   ```
   m$ mn -h
   ```

   Was ist zu sehen? Inspiziere und protokolliere die Ausgabe.

   (e) Jetzt starten wir Mininet richtig.

   ```
   m$ mn
   ```

   Was ist zu sehen? Inspiziere, protokolliere und interpretiere die Ausgabe.

   (f) Mininet hat auf diese Weise ein sehr einfache Netzwerk-Topologie erstellt. Zeichne anhand der obigen Ausgabe die Netzwerk-Toppologie, wobei Hosts als Kreise dargestellt werden soll und der Switch als Rechteck. Die selbst gezeichnete Topologie ist Lösung dieser Aufgabe und diese ist zu protokollieren.

   (g) Das Ganze kann man auch nachträglich anfragen, indem man weitere Kommandos ausführt. Gib im Mininet-Kommandoprompt ein:

   ```
   mininet> dump
   ```

   und protokolliere die Ausgabe. Was zeigt das Kommando offensichtlich an? Wie lauten die IP-Adressen der Hosts?

[7] Die Option `-E` bei `sudo` bewirkt, dass Umgebungsvariablen beibehalten werden, was für das Funktionieren der UI von Wireshark wichtig ist.

(h) Jetzt geben wir

```
mininet> links
```

ein und protokolliere die Ausgabe. Interpretiere auch diese Ausgabe.

(i) Anhand der Ausgaben können wir jetzt wieder die Topologie zeichnen, was dasselbe Ergebnis wie oben ergeben sollte. Glücklicherweise gibt es auch eine Web-Applikation, die diese Aufgabe für uns abnimmt. Die URL lautet http://demo.spear.narmox.com/app/?apiurl=demo#!/mininet. Wir fügen jetzt beide eben protokollierten Ausgaben in die entsprechenden Eingabefelder der Webseite und machen von der Webseite einen Screenshot.

(j) Wir senden jetzt ein Ping vom Host `h1` zum Host `h2`:

```
mininet> h1 ping h2
```

und protokollieren die ersten fünf Zeilen der Ausgabe (Abbruch wie gehabt mit `Strg+C` bzw. `Ctrl+C`). Die Eingabe bedeutet, dass wir auf `h1` das Kommando `ping h2` aufrufen möchten.

(k) Wir holen kurz das Wireshark-Fenster zurück in den Fokus. Was ist zu erkennen?

(l) Auch hiervon machen wir einen Screenshot und widmen uns dann wieder dem CLI.

(m) Wir können auch andere Kommandos auf den Hosts ausführen:

```
mininet> h2 ls
```

(n) Gib nun folgende Sequenz ein (einige Kommandos werden einen Fehler quittieren):

```
mininet> h2 ls test
mininet> h1 touch test
mininet> h2 ls test
mininet> h2 rm test
```

Protokolliere und interpretiere die Ausgabe.

(o) Wer an der Stelle nicht selbst weiter experimentieren möchte, schließt Mininet jetzt

```
mininet> exit
```

Damit schließen wir das erste Laborarbeitsblatt. Herzlichen Glückwunsch :)

To be continued...