Fachhochschule Frankfurt – University of Applied Sciences  
Fachbereich 2: Informatik und Ingenieurwissenschaften  
Studiengang: Informatik (B.S)

Seminararbeit

**Java.util. \***

Dozent: Maurizio Petrozziello  
Modul 25: Informatik Projek – Software-defined Networking mit Openflow  
Wintersemester 2021/2022

Vorgelegt von:  
Tung Thanh Trinh, James Belmonte, Naghmeh Ghavidel Rostami, Mücahit Sagiroglu  
Matrikelnummer: 1320718, 1340604, 1249307, 1228852

Abgabetermin: 27.01.2022

**Inhaltsverzeichnis**

[Abbildungsverzeichnis 1](#_Toc87990237)

[Einleitung 1](#_Toc87990238)

[Teil eins 1](#_Toc87990239)

[Vorwort 2](#_Toc87990240)

[Abkürzungsverzeichnis 2](#_Toc87990241)

[1. Einleitung 3](#_Toc87990242)

[2. Die Computertomographie 4](#_Toc87990243)

[3. Fazit 5](#_Toc87990244)

[Literaturverzeichnis 6](#_Toc87990245)

[Eidesstaatliche Erklärung 7](#_Toc87990246)

# Abbildungsverzeichnis

# Vorwort

Kurz erläutern, wie die vorliegende Arbeit aufgebaut ist. Erklären welche Aspekte man betrachtet etc.

# Abkürzungsverzeichnis

CT = Computerthomograph / ie / en

UNSCEAR = United Nations Scientific Committee on the Effects of Atomic Radiation

# Einleitung

Seit der Einführung der Computertomographie ist die Industrie, wurde die Interesse an der Technologie

# Organisation

Hier können wir eingehen, welche Tools und Methoden wir benutzt haben, um auf das Ergebnis zu kommen. Besprechungen und Protokolle können helfen, diese zu realisieren.

**4.1 Verwendete Werkzeuge**

**4.1.1 Mininet**

**4.1.1.1 Einführung**

Mininet ist ein Netzwerk Emulator mit der man Netzwerke simulieren kann. Bei Mininet handelt es sich um kostenlose Open-Source-Software, die die virtuelle Maschine und Controller die Recherche in SDN und OpenFlow ermöglichen.

Mininets ermöglichen eine sehr groß angelegte Topologie, wodurch ein Netzwerk von Hosts, Switch-, Controller- und virtuelle Links erstellt wird. Das Ausführen von Tests mit den Komponenten ist unkompliziert und kann über Python Schnittstelle erledigt werden.

**4.1.1.2 Funktionalität**

· Mininet stellt ein einfaches Netzwerk Testbed dar, welches aber auch gleichzeitig auch günstig ist. Da der Switch Openflow in Mininet alle Eigenschaften hat, wie ein echter switch OpenFlow, ist die Anwendung von Netzwerkemulator mit Mininet praktisch sinnvoll.

· Ermöglicht das Debuggen und Ausführen von Tests großer Netzwerke mithilfe von CLI.

· Unterstützung zum Einrichten beliebiger benutzerdefinierter Diagramme

· Die Anwendungen im Mininet können im echten Netzwerk realisiert werden, ohne dass man sein Code ändern muss.

· Mininet bieten eine benutzerfreundliche und erweiterbare Python-API

**4.1.1.3 Nachteil**

Aktuelle Nachteile von Mininet: nur unter Linux lauffähig. Dadurch, dass Mininet nur auf einem Rechner ausgeführt werden kann, ist es leistungsmäßig eingeschränkt. Daher hängt die Leistung von den Ressourcen dieses Rechners ab.

**4.1.1.4** **Komponenten**

· **Links**: Die Links in einem Mininet sind ein Paar virtueller Ethernets, die wie ein Draht funktionieren, der zwei virtuelle Schnittstellen verbindet. Pakete werden von einer Schnittstelle zur anderen gesendet, diese Schnittstellen stellen für alle genau dasselbe wie Ethernet-Ports dar System- und Anwendungssoftware.

· **Hosts:** Die Netzwerk-Namespaces enthalten den Netzwerkstatus (network state). Sie bieten Prozessen (oder Gruppen von Prozessen) die Kontrolle über Schnittstellen, Ports und Routing-Tabellen. Jeder Host hat seine eigenen Ethernet-Schnittstellen (initiiert und gesetzt durch den Befehl ip link add/set) und eine Verbindung zum Mininet.

· **Switches**: OpenFlow-Softswitches bieten die gleiche Semantik für das Senden

Paket wie ein Hardware-Switch. Ein Switch arbeitet auf der Datenverbindungsschicht (Layer 2) und manchmal auf der Netzwerkschicht (Layer 3) des OSI (Open Systems Interconnection) -Referenzmodells und unterstützt daher jedes beliebige Paketprotokoll. LANs, die zur Verbindung von Segmenten Switches verwenden, werden als geswitchte LANs oder, im Falle von Ethernet-Netzwerken, als geswitchte Ethernet-LANs bezeichnet. In Netzwerken ist der Switch das Gerät, das Pakete zwischen LAN-Segmenten filtert und weiterleitet

· **Controller:** Controller können sich überall im realen Netzwerk oder in der Umgebung befinden Simulation.

**4.1.1.5** **Installation**

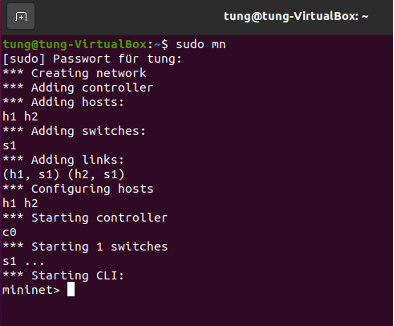
**$sudo apt-get install git**

**$git clone git://github.com/mininet/mininet**

**$sudo mininet/util/install.sh –a**

**4.1.1.6 Aufbau**

Durch die Eingabe von dem Befehl sudo mn erfolgt ein Default Mininet. Auf der virtuellen Maschine laufen 4 Entitäten (2 Hosts, 1 Switch, 1 Controller).



**4.1.2** **Floodlight Controller**

**4.1.2.1 Einführung**

Derzeit gibt es auf dem Markt einige Controller, die in SDN verwendet werden, wie zum Beispiel: OpenDaylight,Ryu, POX, NOX, HP VENTIL... Wir haben uns für Floodlight Controller entschieden, weil er die Anforderungen des Projekts erfüllt und einfach zu installiert ist. Floodlight Controller ist sehr user-friendly und bietet sogar eine Benutzeroberfläche.

**4.1.2.2 Installation**

$ git clone git://github.com/floodlight/floodlight.git

$ cd floodlight

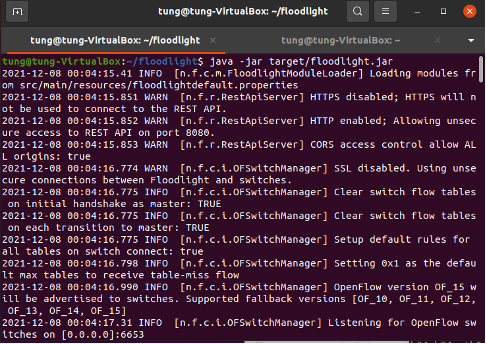
$ git submodule init

$ git submodule update

$ ant

**4.1.2.3 Aufbau**

$ java -jar target/floodlight.jar



**4.1.3** **Ergebnis**

**Please insert a picture of floodlight GUI …**

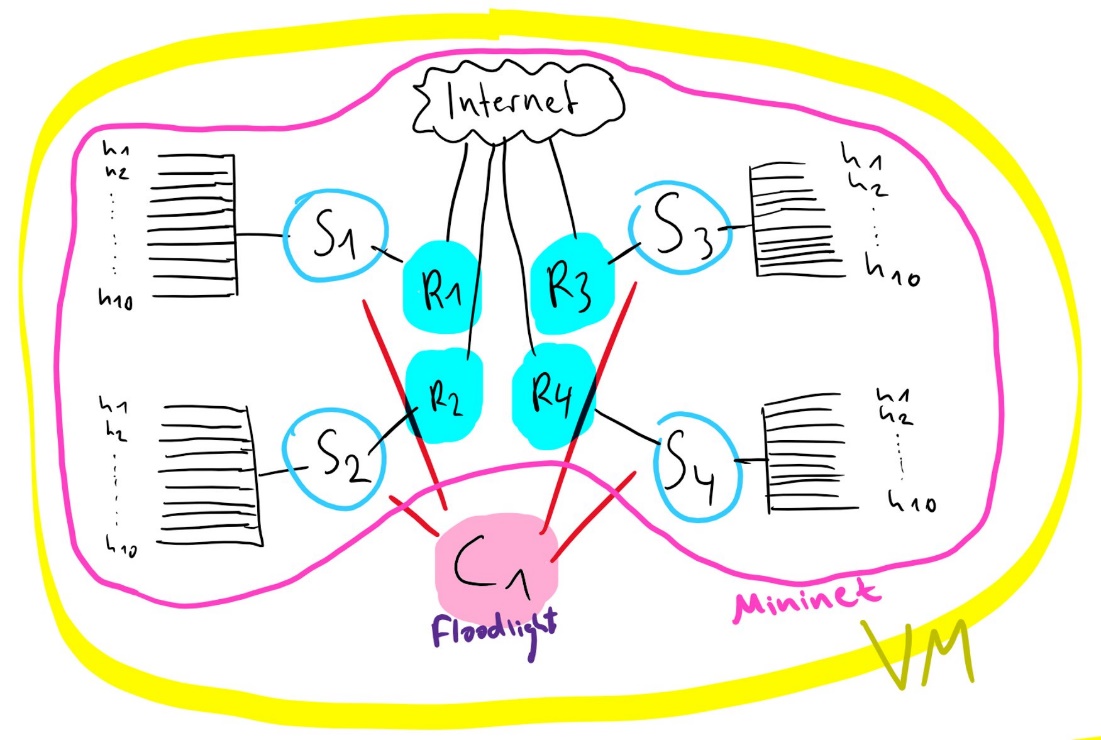
# Die Computertomographie

Die Computertomographie ist ein bildgebendes Verfahren

**Aufgabe 1**

**Vorüberlegungen**

Bereitstellungsmodell:



Das Netzwerkmodell umfasst:

**Mininet:** 4 Switches und 40 Hosts erstellen. Diese Switches und Hosts werden verbunden mit unserem Floodlight Controller. 4 Switches repräsentieren 4 Lokationen mit jeweils 10 Hosts.Damit die Hosts und Switches mit dem Internet verbunden werden können, steht 4 Routers da. Router kann man wie ein Gateway betrachten. Anhand von Headern und Weiterleitungstabellen bestimmt der Router den besten Weg zur Weiterleitung der Pakete.

**Floodlight Controller:** SDN-Controller (SDN-Controller) steuert den Zugriff zwischen Hosts im Netzwerk. Dies wird durch eine Python Datei implementiert

Sollten wir evtl. einen Plan auch zeichnen bzw. mit Tabellen angeben?

Netzwerkplan: <https://creately.com/blog/de/diagramme/netzwerkplan/>

Eventuell DRAW.IO benutzen? Oder die Website mit Mininet Schnittstelle.

Aufgabe: Wichtig im Netzwerkplan Komponente müssen passen, Adressen müssen stimmen und übersichtlich sein, die VPN Tunnel muss eingezeichnet werden (mit Verschlüsselung). Recherchieren wie die Norm ist(wie weit ins Detail gehen)

Problem: Habe versucht die Verbindung zwischen den zwei ´Routern anzuzeigen. Leider zeigt Floodlight auch keine Verbindung zwischen zwei Host, Switches usw. an. Ist wahrscheinlich ein Bug.

**4.1.1** **Durchführung**

In diesem Abschnitt wird der Aufbau der Simulation über Mininet erklärt.

In der Main-Funktion werden die Komponenten eines Netzwerks deklariert und aufgerufen. Das sind eine Topologie, ein Controller mit zugewiesenem Port und ein Mininet Objekt mit der deklarierten Topologie. Anschließend geben wir für unsere Routers Routing-Regeln und Informationen.

**4.2.2.1 Aufbau der Topologie**

Der Mininet-Script besteht aus einer Main-Funktion bei der als allererstes mit der von uns definierten Klasse „Netzwerk()“. Und damit zeigen wir wie das Netzwerk beziehungsweise eine Topologie erstellt wird. Die Klasse übernimmt ein Topo-Objekt an dem er mit der in ihm definierten „build()“ Methode die Konfiguration des Netzwerkes vornimmt.

In der „build()“ Methode definieren wir zuerst einen String der den privaten-IP-Bereich der vier Lokationen enthält. Der defaultIP String bleibt in unvollständiger Form „192.168.%s.1/24“. Somit kann er später durch passende Stellen ersetzt und genutzt werden. Lediglich ist hier im dritten Block ein Platzhalter eingesetzt der beim erstellen der Router in einer Schleife durch die Zahl der Iteration ersetzt wird.

Zunächst wird ein leeres Array/Liste unter dem Namen „Routers“ deklariert. Dies wird dann später genutzt und mit den Router-Objekten gefüllt. Später für die Verlinkung der Router mit dem jeweiligen Switch wird die Liste aufgerufen. Für große Anzahl von Router ist die Bedeutung der Liste sehr praktisch.

Im nächsten Teil der Klasse Netzwerk, Zeile 35 bis 53 wollen wir die Komponenten des Netzwerks implementieren. Dies ist mit Hilfe einer Schleife mit 4 Durchläufe ausgeführt worden. 4 Durchläufe entsprechen 4 Lokationen, jeweils 1 Switch 1 Router und 10 Hosts.

Dabei wird bei jeder Iteration erst ein Router-Objekt mit der Methode „self.addNode()“ erstellt, bei dem der Name, der private IP-Adressen-Bereich, die MAC-Adresse und benutzerdefinierte Parameter für die Konfiguration, dass der Router IP-Forwarding aktiviert bekommt, übergeben. Danach wird der Router der vorher erstellten Liste eingefügt.

Als nächstes wird ein Switch mit der Methode „self.addSwitch()“ erstellt, der nur einen Namen erhält, der anschließend mit der Methode „self.addLink()“ mit dem Router verbunden wird, bei dem die Netzwerkschnittstelle des Router benannt und der private-IP-Adressenbereich vergeben wird.

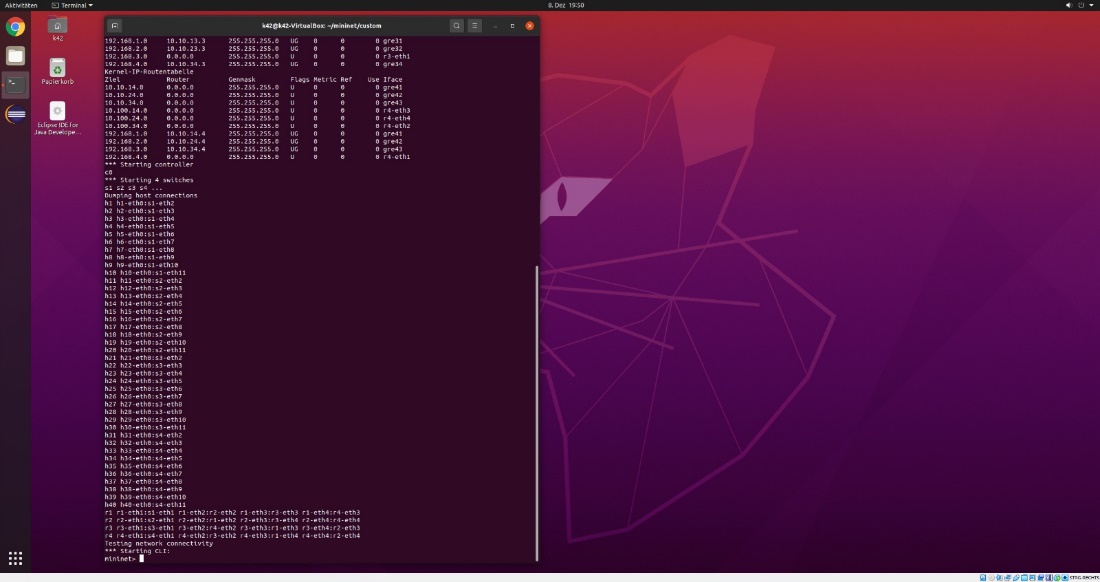
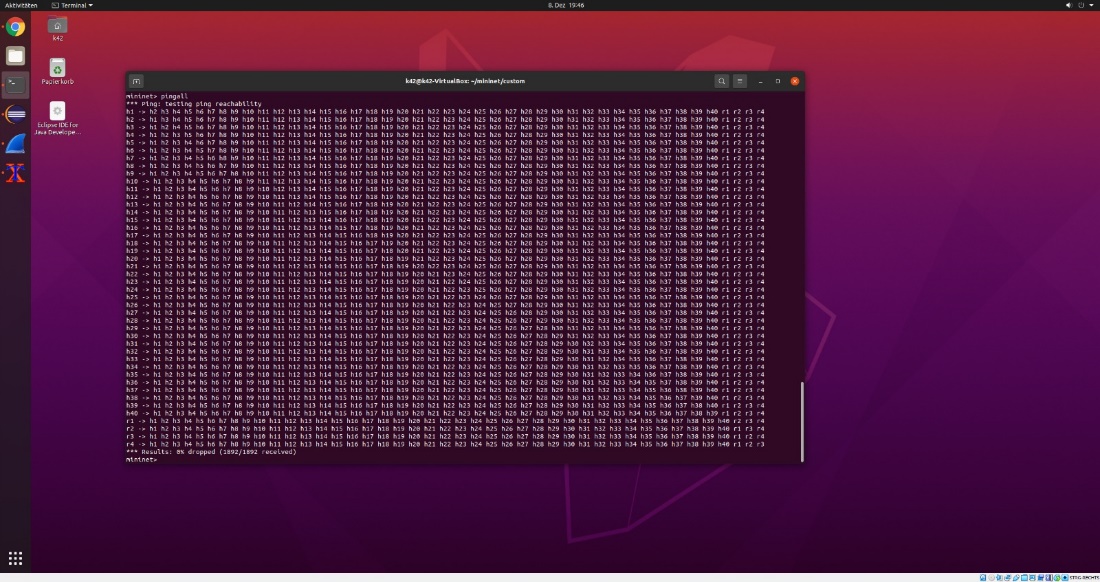
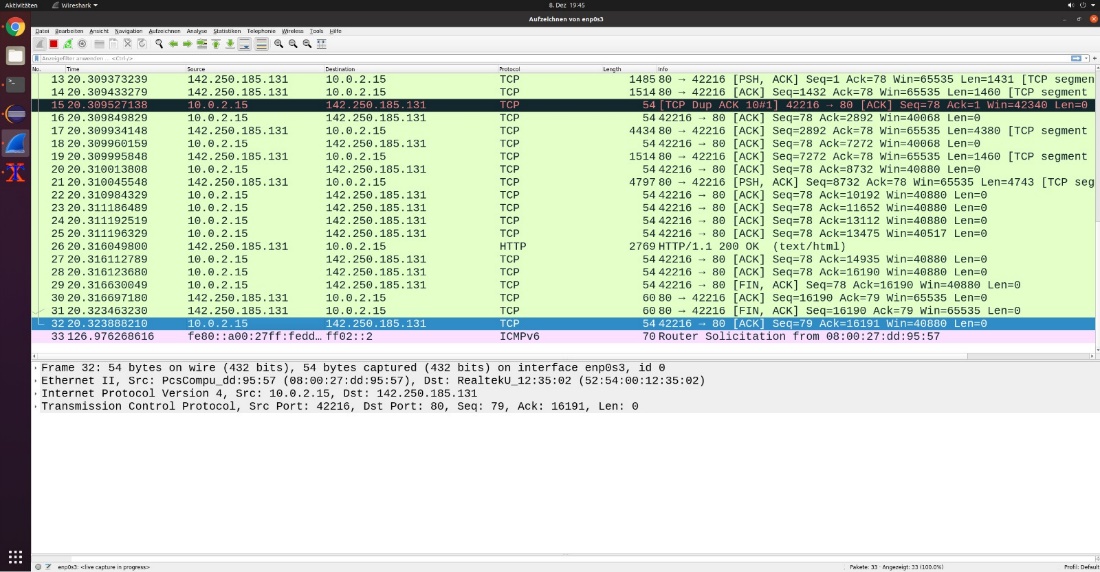
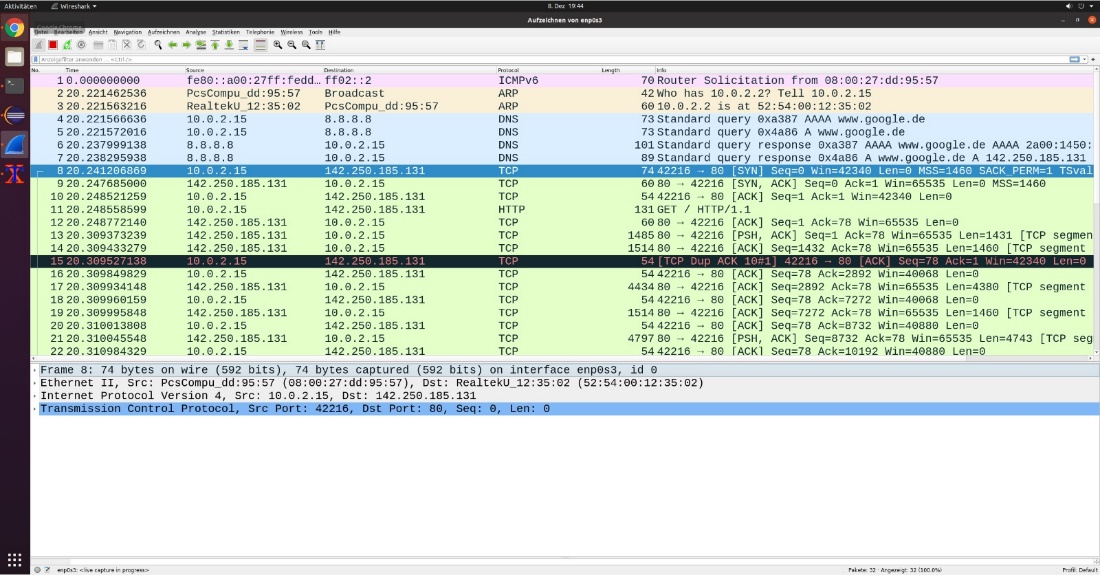
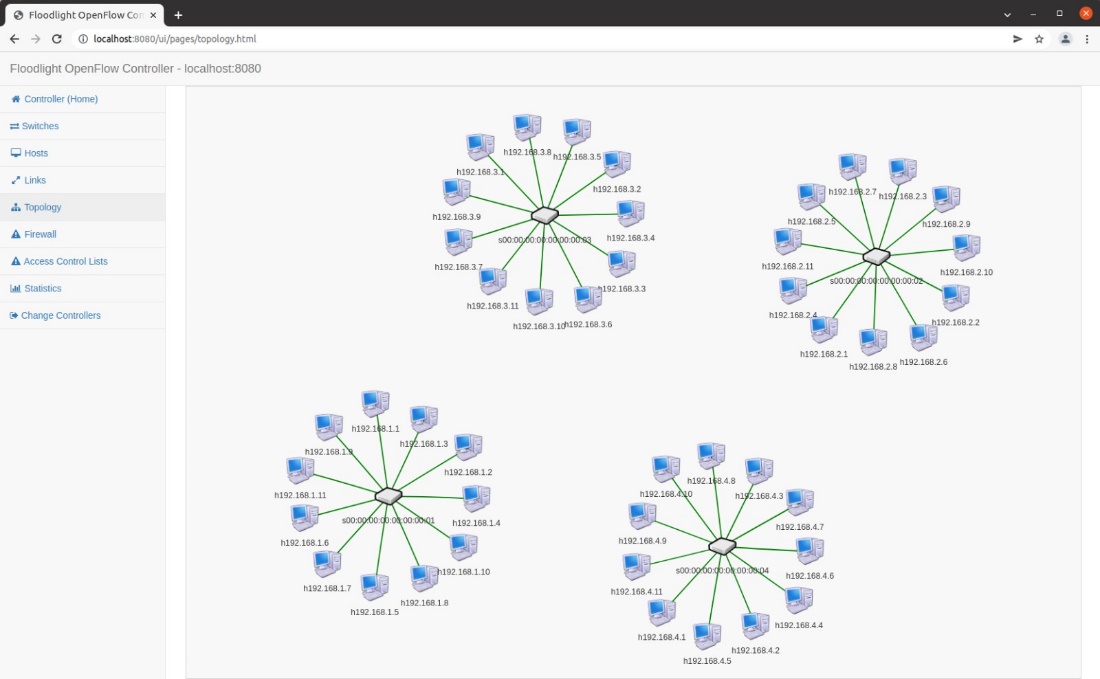
Danach folgt noch eine Schleife, bei der insgesamt n Hosts erstellt und mit dem Switch verbunden werden. Die Hosts erhalten für den jeweiligen privaten-IP-Bereich eine IP, eine MAC-Adresse und die IP des jeweiligen Routers als Standard-Route zugewiesen.

Nachdem für alle Lokationen der Rumpf erstellt worden ist, stellen wir die Verbindungen zwischen den Routern mit dem Befehl „self.addLink()“ her. Dabei wird jeder Router mit allen anderen Routern verbunden. Dieser Vorgang wird das Internet simulieren, worauf ebenfalls der Tunnel und die Verschlüsselung implementiert wird. Dabei wird der Netzwerkschnittstellen-Name für beide Router und die jeweilige öffentliche-IP-Adresse definiert. Zusätzlich setzen wir per „bw=20“ Befehl die Bandbreite der Leitung auf 20 Megabit, wobei dies die geforderte SDSL-Leitung der Aufgabe sieben darstellen soll.

**4.2.2.2 Controller Implementierung**

Es würde weiter bei der Main-Funktion gehen. Dort initialisieren wir ein RemoteController-Objekt der einen Namen, die Konfiguration um was für einen Controller es sich handelt, die IP-Adresse und den Port, wo er zu erreichen ist, bekommt. Hier ist wichtig zu erwähnen, dass der Controller auf Ubuntu läuft und das der Controller per „localhost“ zu erreichen ist. Hier könnte man ebenfalls den Controller auf einer anderen VM laufen lassen und ihn per Internes Netzwerk verbinden oder auf der echten Welt laufen lassen und ihn per NAT-Verbindung verbinden. Auch besteht noch die Möglichkeit den Controller auf dem Betriebssystem, auf dem Virtualbox läuft, laufen zu lassen und ihn per Host-Only-Adapter zu verbinden. Anschließend wird ein Mininet-Objekt erstellt, bei dem die erstellte Topologie, der erstellte Controller, ein TCLink-Objekt für die Einstellung der Bandbreite der Netzwerkadapter und ein OVSKernelSwitch-Objekt für die Erstellung der Switche als Open vSwitche. Die Open vSwitche nutzen wir später, um den Quality of Service zu implementieren, bei dem wir per ovs-vsctl-Befehle Queues an den Ports der Switche erstellen und die Priorisierung der Pakete vornehmen.

Ergebnis



Als nächstes folgt in der Main-Funktion die Einrichtung der Tunnel zwischen den Routern beziehungsweise den Lokationen. Dafür baut jeder Router mit jedem Router einen GRE-Tunnel auf, für den wir den Befehl „ip tunnel add Tunnel-Name mode gre local Router-Schnittstelle-des-eigenen-Routers remote Router-Schnittstelle-des-anderen-Routers ttl 255“ bei jedem Router mit der Mininet-Methode „info(net[‚Router-Name‘].cmd(Befehl))“ ausgeben und ausführen. Nachdem die Verbindung definiert wurde fahren wir den Tunnel-Adapter per „ip link set Tunnel-Name up“ hoch. Anschließend geben wir dem Tunnel Adapter mit dem Befehl „ip addr add Tunnel-IP dev Tunnel-Name“ die jeweilige Tunnel IP. Hier ist wichtig zu erwähnen, dass die IP zwischen zwei Lokationen im selben Netzwerkbereich, wie bei der Erstellung und Simulierung des Internets zwischen den Routern, sein muss. Nachdem der Tunnel aufgesetzt worden ist, sind alle Pakete, die durch den Tunnel versendet werden, nun als Payload eines neuen Paketes, wo der IP-Header der dem Tunnel entspricht. Auch ist sehr wichtig, dass nun Pakete, die den MTU erreichen, jetzt eine geringere Länge annehmen müssen, da der neue Payload aus dem alten Payload und dem IP-Header besteht. Wenn dies der Fall ist schickt der Router eine Aufforderung an den Absender zurück das Paket kleiner zu gestalten.

Im nächsten Schritt definieren wir für jeden Router die Route zum anderen Sub-Netzwerkadressenbereich, welches wir per „ip route add IP-der-anderen-Lokation via IP-über-welchen-Adapter dev Adapter-Name“ Befehl in die Routing-Tabelle einfügen. Hier ist sehr wichtig, dass die Lokationen jeweils andere privaten-Adressenbereiche besitzen, da es sonst bei Überschneidungen Probleme auftreten können, weil wir das Internet simulieren. Bei einer echten Umgebung mit echten öffentlichen-IP-Adressen könnte man ohne Bedenken die gleichen privaten-IP-Adressen für die Hosts unterschiedlicher Lokationen benutzen, wie es auch im echten Leben üblich ist.

Nachdem der Tunnel aller Lokationen von und zu den Lokationen aufgesetzt ist, verschlüsseln wir als nächstes alle Pakete, die am Router aus dem privaten-IP-Adressenbereich ankommen und entschlüsseln alle Pakete die am Router aus dem öffentlichen-IP-Adressenbereich ankommen. Für diese Methoden wird die Verschlüsselung über IPSEC im Transport-Modus benutzt, welches wir per „ip xfrm state add src IP-Adresse-des-Routers dst IP-Adresse-des-Zie-Routers proto esp spi V…….

**Aufgabe 2**

Wir haben den Verkehr zwischen allen 4 Routern verschlüsselt verschickt. Die Methode die wir benutzt haben heißt IPSEC over GRE und bedeutet, dass wir erst ein Paket per IPSEC (esp Methode) verschlüsseln und dann durch ein GRE-Tunnel versenden. Angekommen auf der anderen Seite wird das Packet entschlüsselt und zum Zielort weitergeroutet.

**Aufgabe 3**

a)

Preisunterschiede von 20 bis 30 Euro pro DSL-Vertrag ist für das Unternehmen irrelevant. Eine Leitung mit höherer Bandbreite wäre vorteilhafter, als eine Preiseinsparung.

-Tabelle in Latex einfügen und Bezug drauf nehmen

Gesamtkosten = 70€ x 4 = 280€

Bandbreite Schätzung (10 Hosts): Downlod= Variable Mbit Upload= 4Mbit pro Host

b)

Anfrage machen:

<https://geschaeftskunden.telekom.de/internet-dsl-tarife/produkt/deutschlandlan-connect-ip>

<https://www.mpcservice.com/company-connect/>

<https://www.savecall.de/standleitung/standleitung-colt-ip-access/>

-Ist Standleitung mit Switch verbunden?

-

Hier hat Herr P. spitze gefunden, dass wir eine Tabelle mit den Preisangeben anfertigen. Kosten jeder Site und Gesamtkosten wären angebracht. Wir sollten noch diskutieren, wie wir uns die WAN-Verbindung (Standleitung) vorstellen. (Da war mal kurz ein Gespräch, ob wir den Controller ganz woanders haben und alle Sites per Standleitung verbunden werden)

a) Und b) sind reine kostenfragen: Fallls man ohne WAN Verbindung eine stabile Internetleistung hat, kann man ohne WAN weiterarbeiten à ADSL, (100mbit mindestens download)

Aufgaben: für jeden Standort(frankfurt, hamburg, berlin, München) ADSL Pakete finden. Von jedem ISP(Telekom, Vodafon, 1&1 usw.) 2 angebote.

Aufgabe: preise für standleitungen zwischen den lokationen finden (Näherungspreise finden).

Zu beiden Aufgaben geschwindigkeit notieren (download, upload), verfügbarkeit, garantierte geschwindigkeit. Andere Vorgaben von Aufgaben beachten à Tabelle anfertigen.

**Aufgabe 4**

Für die Aufgabe vier wird verlangt, dass wir auf dem Router die NAT-Firewall Funktion implementieren, sodass jede Anfrage ins World-Wide-Web mit der öffentlichen-IP-Adresse des Routers und nicht mit der privaten-IP-Adresse durchgeführt wird. Wichtig ist zu wissen, dass dies zum Verstecken der privaten-IP-Adressen beziehungsweise der Geräte führt und damit auch keine Informationen über die Geräte ins World-Wide-Web geschickt werden, welches zu einer noch größeren Sicherheit führt. Hinter jeder öffentlichen-IP-Adresse können mehrere tausende Geräte stehen und das Internet erreichen. Des Weiteren ist die Anzahl der IPv4-Adressen durch den eigenen Aufbau begrenzt. Für unsere vier Lokationen würden wir vier öffentliche-IP-Adressen über unseren Internet-Service-Provider zugeteilt bekommen. Diese wird vom Internet-Service-Provider in bestimmten Zeitintervallen immer wieder neu vergeben, welches ebenfalls zu einer gewissen Sicherheit beiträgt.

Um überhaupt den Hosts der Lokationen den Internetzugang zu ermöglichen, müssen wir den Routern erst einmal den Zugang ermöglichen. Dafür haben wir in Virtualbox alle vier Verfügbaren Netzwerk-Schnittstellen aktiviert und ans NAT des realen Hosts angeschlossen. Danach haben wir die vier Schnittstellen an die Router r1, r2, r3 und r4 per „Intf('*Schnittstellen-Name*', node=*Router-Objekt*)“ Mininet-Befehl zugewiesen. Ein Problem welches bei diesem Schritt aufgetreten ist, war das die Netzwerkschnittstellen beim Beenden von Mininet auch für Ubuntu nicht mehr verfügbar waren. Deshalb mussten wir immer die virtuelle Maschine beziehungsweise Ubuntu neustarten, wenn wir Mininet beendet hatten, um Mininet wieder ausführen zu können, da sonst der Befehle „Int(…)“ die Netzwerkschnittstelle nicht findet und ein Fehler zurückgibt. Um dies zu beheben wird beim Beenden von Mininet den Befehl „ip link set Schnittstellen-Name netns 1“ für jeden Router und seiner zugewiesenen Netzwerkschnittstelle ausgeführt. Hier vll weitererklären Anschließend haben wir den „info(net['Router-Name'].cmd("dhclient *Schnittstellen-Name*"))“ Mininet-Befehl für jeden Router ausgeführt, um den Routern eine Öffentliche-IP-Adresse des Virtualbox Nat-Services zuzuweisen. Hier ist es wichtig zu verstehen, dass die Öffentliche-IP von Virtualbox nur simuliert wird und dies nicht die Öffentliche-IP des realen Hosts ist. In unserem Fall haben die Router jeweils die Öffentliche-IP [10.0.2.15; 10.0.3.15; 10.0.4.15; 10.0.5.15] zugewiesen bekommen.

Da nun eine Internetverbindung für alle Router besteht, müssen wir ebenfalls den DNS-Server definieren, damit die Hosts nicht nur per IPv4 ins Internet zugreifen können. Eine Möglichkeit bestand darin die Datei /etc/resolv.conf per Admin-Rechte zu bearbeiten und dort die IPv4 des Nameservers auf ein beliebiges wie etwa von Google 8.8.8.8 und 8.8.4.4 zu ändern. Jedoch wird bei dieser Variante nach jedem Neustart der Nameserver auf die IPv4-Adresse 127.0.0.53 gesetzt, welches wir bei jedem Neustart des Betriebssystems immer wieder neusetzen müssen. Um dem entgegenzuwirken haben wir das Paket Resolvconf per „sudo apt install resolvconf“ installiert und in die Datei „/etc/resolvconf/resolv.conf.d/head“ den gewünschten Nameserver, in unserem Fall 8.8.8.8 und 8.8.4.4, gesetzt und gespeichert, welches für ein permanenten Eintrag des Nameservers sorgt. Nach diesem Schritt war es den Routern möglich das Internet auch per Domain-Namen zu erreichen.

Der Default-Route der Hosts ist der jeweilige Router in der Lokation. Wenn jetzt ein Host eine Website aufruft, schickt er eine Anfrage an den Router, der die Anfrage für den Host mit seiner öffentlichen-IP der Virtualboxmaschine durchführt und dem Host die Antwort zurückgibt. Damit genau dies gewährleistet haben wir auf allen Routern den Befehl „sudo iptables -t nat -A POSTROUTING -o Netzwerkschnittstellen-Name -j MASQUERADE“ ausgeführt. Dabei nutzen wir das Tool iptables, welches das Linux-Kernel konfigurieren kann. Der Befehl schreibt in die Tabelle „nat“, dass alle Pakete, die an der Netzwerkschnittstelle weitergeroutet werden, die eigene IPv4-Adresse dieser Netzwerkschnittstelle als Quelladresse erhaltet. Es wird im Befehl MASQUERADE benutzt, weil zum Zeitpunkt der Ausführung des Befehls die IPv4-Adresse der Schnittstelle unbekannt sein kann beziehungsweise sich ändern kann. Würden der Router eine statische IPv4-Adresse besitzen, so würden wir statt MASQUERADE direkt die IPv4-Adresse angeben. Genau dieser Schritt hat ermöglicht, dass die Router eine NAT-Firewall Funktion haben, sodass die Hosts in den Lokationen mit der öffentlichen-IPv4-Adresse des Routers beziehungsweise die der von Virtualbox ins Internet gehen können.

Als erstes habe ich auf dem Router R1 den Internetzugang ermöglicht. Dafür habe ich ihm den Interface enp0s3 von Virtualbox als Interface übernehmen lassen und dann per „r1 dhclient enp0s3“ sich die IP-Adresse vergeben lassen.

Als nächstes muss ich einen Proxy host für die Site wo R1 ist erstellen und ihn über R1 ins internet zugreifen lassen. -.-

Musste soeben den nameserver über nano /etc/resolv.conf einstellen, damit r1 nicht nur per 8.8.8.8 sondern auch per google.de pingen kann.

Masquerade?

Masquerade richtig!

Schritte: r1 dhclient enp0s3, r1 sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

Kann jetzt von P1 google.de anpingen.

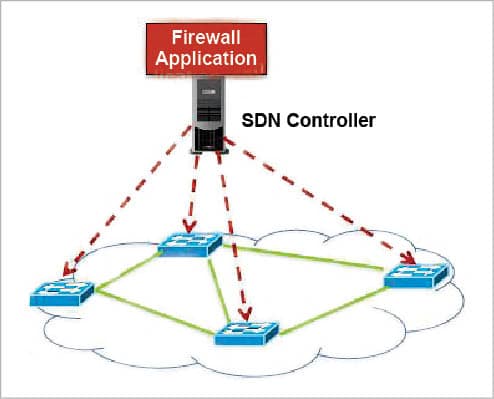
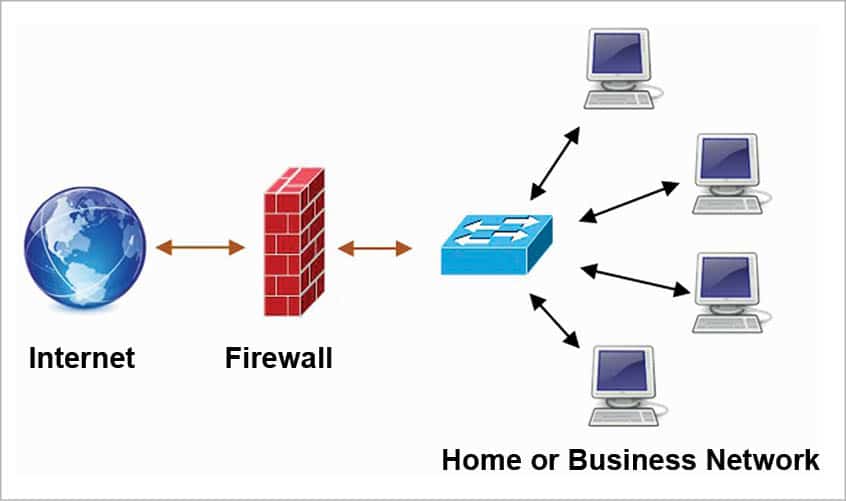
Firewall kann im Moment durch ein Script aktiviert werden. Achtung: Die Firewall liegt am Switch und nicht am Router. Unterscheiden bitte. Zusätzlich wird ARP, ICMP und TCP im Netz 192.168.0.0/16 erlaubt. PING und http funktionieren mit der Regel. Fragen ob an Router oder Switch :S

Sollten nachfragen, ob weitere Regeln eingefügt werden sollten und/oder ob die Regeln zu **durchlässig** sind. Sollten wir für jede Switch **exakte** Regeln anlegen?

Ping funktioniert nicht ganz!

Aufgabe: Firewall muss bis Layer 7 gehen

<https://www.opensourceforu.com/2016/07/implementing-a-software-defined-network-sdn-based-firewall/>



<https://youtu.be/WrLsiT65igY?t=681>

Im Video sieht man wie er Attacken ausführt

<https://ieeexplore.ieee.org/document/8962096>

<https://www.net.in.tum.de/fileadmin/bibtex/publications/theses/2015-IDP-OpenFlow-Firewall.pdf>

Gut erklärt.

Die Firewall die wir aufsetzen sollte „Stateful“ sein. Das heißt, wenn wir in eine Richtung durchlassen, sollte die Firewall die Rückrichtung automatisch erkennen! Dazu muss man in der Flow-Table alle Flows an den Controller senden (Auch die die gedropt werden). Siehe Link Video Min 8:25 <https://www.youtube.com/watch?v=7R91K0d2r2E>

Wir können nun mit aktivierter Firewall ein pingall Command ohne Probleme ausführen. Stand: 26.11.2021. Leider klappt das mit dem stateful nicht. Egal, weiter geht es mit dem Proxy.

Note: Eventuell die Rest-API über die Software „Postman“ betreiben. Leichter und übersichtlich.

Note: In der Aufgabe steht „Nat-Firewall“, welches ein Prinzip meint, wo ausgehende Pakete mit der öffentlichen IP versehen werden. Beim Rückweg dann wieder mit der privaten…

<https://whatismyipaddress.com/nat>

Privater Netzwerkbereich bekommt eine Public IP. (Wegen Ipv4 knappheit, keeping individual IP addresses hidden) Das ganze soll im Router geschehen.

* Pro Router nur eine public IP
* Standleitung ist garantiert stabil

**Aufgabe 5**

Bei der Aufgabe fünf muss ein Web-Proxy-Server in allen Lokationen eingerichtet werden, sodass der Web-Proxy jede http- oder https-Anfrage (request) aller Geräte im gleichen Subnetz selbst durchführt und die Antwort (response) dem Anfrager zurückschickt. Der Vorteil hierbei ist, dass der Web-Proxy-Server für eine Sicherheit in allen Schichten des OSI-Modells sorgen kann. Es muss lediglich nur an dem Web-Proxy-Server Einstellungen bezüglich gewünschter Inhalte, IP-Adressen oder MAC-Adressen vorgenommen werden, um die Sicherheit für alle Geräte, die über den Web-Proxy-Server eine Anfrage (request) machen, zu gewährleisten. Des Weiteren wäre die Bandbreite weniger Ausgelastet, da der Web-Proxy-Server jede neue Antwort (response) in seinem Cache speichert und bei erneuter Anfrage (response), die Antwort aus seinem Cache, statt durch erneute Abfrage aus dem World-Wide-Web, ausgibt.

Um den Web-Proxy-Server zu realisieren haben wir einen Host in Mininet erstellt, ihm den Namen p1, eine IPv4-Adresse im privaten Netzwerkbereich vergeben und ihn mit dem Switch (Mininet-Switch) verbunden. Dies haben wir zuerst für eine Lokation implementiert, da die Funktionalität noch unbekannt ist. Auf dem Web-Proxy-Server haben wir per „Xterm p1“ Befehl einen Terminal auf P1 gestartet und die Internetverbindung durch aufrufen einer Website mit dem Befehl „curl www.google.de“ als funktionsfähig getestet.

Als nächstes wollten wir eine Schnittstelle des Web-Proxy-Servers mit einer Schnittstelle des Ubuntu-Betriebssystems beziehungsweise der Virtualbox-Maschine verbinden und den Web-Proxy-Server p1 als eine Schnittstellenverbindung und nicht als Server benutzen. Anschließend hatten wir vor einen Squid-Proxy-Server auf der Ubuntu-VM laufen zu lassen, um die Web-Proxy-Funktion zu realisieren. Wichtig ist, dass dies der Realität fern ist und der Web-Proxy-Server eine Software selber laufen lassen würden und nicht als Schnittstelle zu anderen Servern dient. Zu unserem bedauern konnten wir den in Mininet erstellten Host p1 nicht mit einer Schnittstelle des Ubuntu-Vms verbinden, weil nur die Schnittstellen der Switches für den VM sichtbar sind.

Aus diesem Grund haben wir einen alternativen http-Proxy-Script aus Github benutzt, um die Web-Proxy-Funktionalität auf p1 per Python-Script einzurichten. Der http-Proxy-Python-Script nimmt nur Anfragen entgegen und führt Sie selber durch und gibt für einen bestimmten Zeitintervall http-Seiten-Daten aus dem Cache zurück. Hier ist nochmal zu verdeutlichen, dass der Python-Script für keine umfangreiche Web-Proxy-Funktionalität ausgelegt ist, jedoch wir diesen aus Testzwecken benutzt haben. Als nächstes haben wir alle Pakete, die beim Switch aus den Workstation-Ports eingehen und einen Ziel-Port als 80 haben, an den Proxy weitergeleitet. Hierfür haben wir über den Controller einen Match und die dazugehörigen Actions-Liste implementiert und dem Switch die Anweisungen geschrieben (Flow-modification). Dadurch haben wir bewirkt, dass der Web-Proxy-Server die Anfragen bekommt, ohne dass die Geräte an dem Switch davon wissen. Der Proxy nimmt die Anfrage entgegen und führt sie selber durch und gibt die Antwort an das jeweilige Gerät weiter. Hier tritt das Problem auf, dass das Gerät eine Antwort von der Website erwartet, jedoch die Antwort vom Proxy geschickt bekommt. Wir könnten hier die erhaltene Antwort ebenfalls per Switch modifizieren und an das jeweilige Gerät weiterschicken. Leider fehlt dazu die Information des Absenders im World-Wide-Web zu dem Zeitpunkt an dem die Antwort von dem Web-Proxy an das Gerät geschickt wird. Resultierend war es uns nicht möglich einen Web-Proxy zu simulieren.

Alternativ könnte man eine transparente Web-Proxy-Funktion an dem Router der jeweiligen Lokationen einrichten, der die Anfragen selber durchführt, verfolgt, speichert und die Antwort an das jeweilige Gerät sendet. Auch das Eintragen der Proxy-Server an den jeweiligen Geräten, die mit dem Switch verbunden sind, würde sich als funktionsfähig erweisen.

Ich habe 0 Ahnung, wie wir mit dem Controller eine Webproxy-Funktion hinbekommen sollen.

Webproxy-Definition: <https://www.youtube.com/watch?v=5cPIukqXe5w>

Nehmen wir an, dass wir hinbekommen haben, dass der Switch die Anfragen im WWW macht/übernimmt. Wie sollen wir die Responses im Chache des Controllers speichern und wieder weitergeben…?

Denke, dass wir Herrn P. nochmal fragen, was er genau verlangt.

Oder sollen wir einfach die IP beim Herausgeben des Paketes an den Router verändern? H1 -> S1 (IP wird geändert) -> R1 -> R2 -> S2 -> H12

H12 – S2 – R2 -> R1 -> S1 (was passiert hier?) -> H1

**Update:** Sollen wir ein Proxy-Server auf dem Switch laufen lassen und alles was http raus und reinkommt darüber laufen lassen?

Alle webanfragen von host wird von switch zu proxy weitergeleitet, die proxy macht den request und erhaltet die Antwort und leitet sie dem Angefragten wieder zurück.

Vorteile: 1) Sicherheit 2) bandbreite weniger ausgelastet 3.) geschwindigkeit

**Stand: 26.11.2021** à Ziel ist es jetzt in Mininet für jede Site einen Proxy zu erstellen. Alle Anfragen von den Hosts müssen über den Switch erst zum Proxy, danach macht der Proxy die Anfrage und sendet die Antwort an den jeweiligen Host.

Bei Squid muss ich jetzt die IP von R1 einfügen, sodass der Service auf R1 läuft. Nur ist die Frage, wie ich das bei 4 Standorten machen kann!

Update 27.11.2021: Der Squid-Proxy läuft auf dem Ubuntu host. Dort werde ich die 4 Interfaces (P1-P4) angeben. Die http requests werden vom Controller zu P1-P4 weitergeleitet. P1-P4 kümmern sich dann um die Anfrage. Im Moment versuche ich bei P1 eine Internetverbindung zu bekommen. Der Router hat eine Verbindung, nur muss P1 durch den Switch zum Router und der Router führt die Anfrage durch. Stichwort

Habe auf P1 einen webproxy laufen. Will nun per floodlight die eingehenden http requests von allen ports auf den proxy weiterleiten. Stichwort: StaticEntryPusher.

Leider nicht geschafft. Die Erkennung des Packetes macht mir zu schaffen.

Update 28.11.2021: Hab das Packet (tcp 80) von H1 auf P1 redirected. Der P1 macht zwar die Anfragen, doch beim Zurücksenden muss das packet wieder geändert werden. Nur woher weiß ich dann wer die Anfrage gemacht hatte….?

Als nächstes Eclipse installieren und per Java-Code das ganze hinzubekommen. (Vll könnte man auch den Firewallcode darüber laufen lassen =D)

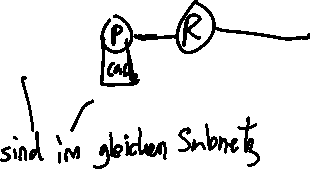
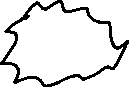
Note: Hab eben bemerkt, dass die ganze Zeit h10 und p1 die gleiche IP hatten. Wahrscheinlich collisions ohne Ende….

**Eclipse:** New class auf „src/main/java“ -> Packet Name eingeben -> Interfaces aussuchen -> Implementieren -> Modul registrieren und in die load-liste setzen!

Protokolle bei https: DNS, TCP, TL

Eine Möglichkeit wäre „connection tracking“. Dabei würden wir das Paket weiterleiten und die connection tracken. Bei antworten darüber die Antwort zurückleiten.

* Wenn man eine http oder https anfrage hat, wird das über proxy geschickt
* Es besteht die Möglichkeit mit dem switch den proxy server anzufragen
* à Adresse von host zu switch und switch zu proxy server besteht noch
* Mininet(transparenter Proxy geht evtl. nicht) In diesem Fall müsste man alle Adressen in den Proxy eintragen
* Die anfrage geht durch und wird gecached im Webproxy Server
* Ein transparenter Proxy gibt google die adresse an
* Sie erklären den Weg und wie weit sie kamen und das wäre auch gelöst
* Anfragen, wenn über proxy nicht möglich, sollen nur über die Nat-Firewall gehen
* Eine ausführliche Erklärung wird bewertet
* Proxy ist ein Stellvertreter, also muss es funktionieren(wie oben genannt, solange ausführlich begründet wird, ist es egal)
* Man kann es bei der host konfigurieren
* Direkter Proxy hätte funktioniert, ein transparenter Proxy auch, da über NAT-Firewall(geht bis Layer 7) funktioniert
* In der Firewall würde ein extra Modul das machen



**Aufgabe 6**

Bei unserer Topologie sehen wir 4 topo’s aber nicht die Verbindung zwischen den 4 Routern. Nachfragen, nachfragen!

**Gefunden:** Es ist eine API verbaut, wo man die paths und routes der Geräte einstellen kann. Eventuell verändert das die Topology beim viewer.

Was genau heißt „Zentrales Monitoring“?

Wollen wir den ganzen Traffic für alle 4 Standorte sehen oder nur jeweils für ein Standort.

Falls das zweite eintrifft könnte man in jedem Switch einen Dummy-Port einrichten und jede Nachricht die im Switch ankommt nochmal als Kopie dort weiterleiten. Dann könnte man für 4 Standorte 4 Monitore haben

Falls das erste eintrifft habe ich erstmal ka.

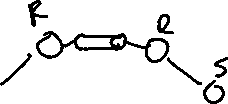
Aufgabe: Über Netzwerkdesign informieren.

https://www.inf-schule.de/rechnernetze/filius/vernetzungrechner/konzept\_topologie

Monitoring sinnvoll im Fehlerfall. Per SDN ein Fow erkennen und zu Monitoring kopieren und weiterleiten. Fehleranalyse. Bei 4 Lokationen monitor machen und über remote zugreifbar machen. Webbrowserbasierend??

(Ein paket kann man eins zu eins kopieren à sollte man aus der Aufgabe lernen.)

* Solange man beweisen kann, dass alles funktioniert, kann man argumentieren das alles funktioniert
* Muss bei der präsi dokumentiert werden, dass diese leitung nicht dargestellt werden können à solange es funktioniert, alles gut(Kein kriterium weiterzukommen)
* Nächste woche info dazu
* Wenn etwas Technisches nicht geht einfach dokumentieren
* Petro konfig.



* Graphische Darstellung nicht relevant, solange funktionsfähig

NAGIOS, SFlow, Controller Statistik?

Wenn es kein Broadcast ist, sieht der Switch die Pakete nicht?

**Aufgabe 7**

Bei der Aufgabe sieben wird die Verbindung zwischen allen vier Routern auf 20 Mbit eingestellt (Mininet Befehl: bw=20). Bei der Einstellung wird die Vorgabe, dass es eine 20Mbit SDSL Leitung ist, genau erfüllt. Als nächstes muss herausgefunden werden, wie viel „genug Bandbreite“ ist, um bei der QoS über SDN diese Geschwindigkeit gewährleisten zu können.

**QoS Definition:** [Quality of Service (QoS)](https://www.paloaltonetworks.com/content/pan/en_US/documentation/80/pan-os/pan-os/quality-of-service) is a set of technologies that work on a network to guarantee its ability to dependably run high-priority applications and traffic under limited network capacity.

Wie über Controller QoS gewährleisten. UDP? TCP? Zoom?..

<https://www.geeksforgeeks.org/video-conferencing-protocols/>

<https://www.c21video.com/technical-papers/videoconferencing/videoconferencing-standards---terminology>

**Überlegung:** Eventuell muss man über den Controller mit der StaticEntryPusher API die priority von UDP Paketen auf hochstellen (vll. Sogar am höchsten). Dies würde heißen, dass alle anderen Verbindungen an nächste Stelle kommen und die Verbindung der Videokonferenzen die Bandbreite gewährleistet bekommen…

Zum Schluss sollte man testen, ob bei einem Videokonferenz genug Bandbreite verfügbar wird, während das Netz überlastet ist.

Überlegen wie man testet. Per Iperf würde es machbar sein.

**Überlegung:** Das Netzt mit einem anderen Protokoll überlasten und schauen was passiert, wenn UDP (Video-Audio) ins Spiel kommt. Wenn Bandbreite für Video gesichert wird  OK

**Bis jetzt:** Habe versucht per iperf() Befehl TCP und UDP Pakete zwischen site1 und site2 zu schicken. Mit TCP funktioniert das, doch leider nicht per UDP, da die MTU beim Verschlüsseln zu groß wird (Vermutung bestätigt da bei der unverschlüsselten Version alles durchging). Wenn ich iperf() für UDP Pakete nutzen kann, werden wir als nächstes die Priorität einstellen und das Netzwerk so austesten mit dem Ziel das UDP priorisiert durchkommt, also die QoS gewährleistet wird, da UDP für Videokonferenzen benutzt wird.

**Gelöst:** Habe die iperf() Funktion umgeschrieben und erfolgreich UDP, auch mit ESP Verschlüsselung, erfolgreich zwischen zwei Sites gesendet. Das umschreiben war nötig, da ich das „-l 1k“ Argument nicht einfügen konnte.

**Als nächstes:** Jetzt den Test mit der Priorisierung durchführen. Hoffe der Ansatz ist richtig.

**Gedanken:** Anscheinend nutzt bsp. Zoom UDP und TCP, d.h. die Priorisierung sollte für beide Protokolle geltend gemacht werden. Nebenbei: Rein theoretisch nutz man für Video-Dienste immer UDP. Quelle: <https://www.lifesize.com/en/blog/tcp-vs-udp/> Siehe Tabelle, letzte Zeile

Bei dem Test sollte das Netzwerk überlastet sein (Beweis dafür!). Danach müsste man ein Videoanruf durchführen (Simulieren?). Wenn der Videoanruf Vorrang erhaltet ist die Aufgabe gelöst. Normalerweise erkennt man das auch wenn das Video nicht hängt.

**Tool für Videokonferenz:** <https://roll.urown.net/server/videocon/index.html>

Aufgabe: Flowtable soll es sein. Recherchieren was QoS gibt à gibt verschiedene Services, welches passt zu unserem Problem?

<https://en.wikipedia.org/wiki/Type_of_service> --> Sehr wichtig für die Aufgabe

<https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/15040526/How+to+Use+OpenFlow+Queues>

Wir können in die Switches per Befehl Queues an die Ports einrichten und die Flow per Controller separieren.

Brauchen wir mehrere Queues? Was schicken wir durch welche Queue?

**Idee:** Wir setzen an dem Port der Switch vom Switch zu Router zwei Queues. Queue 0 wird für low-priority-flows dienen und Queue 1 wird für high-priority-flows dienen. Hier müssen wir Queue 0 als standard einstellen. Dann an der Queue1 nur die Flows reinschicken die wir priorisieren wollen.

**Versuch:** Jitsi Meet auf zweitem VM als Server installieren und an eine Lokation hinzufügen. Auf einem dritten VM einen Clienten und an eine andere Lokation hinzufügen. Anschließend die Flows separieren.

**Test:** Leitung vollpumpen und ein Videoanruf starten.

Video/Audio = UDP(Richtige Annahme? Oder doch TCP?) Eigener Port: 8082

Befehl auf Switch, um flow auf queue zuzuweisen: sudo ovs-ofctl add-flow s1 priority=65535,tcp,tcp\_dst=8082,actions=set\_queue:122,normal

http://manpages.ubuntu.com/manpages/xenial/man8/ovs-ofctl.8.html

1. Qos auf der VSwitch eingestellt. 2 Queues pro Switchausgang nach Router
2. Alle Pakete gehen auf Queue 0. Video-Telefon Pakete (anhand Ports bestimmen) gehen auf Queue 1
3. Leitung vollpumpen und Video-Telefon (port) Pakete schieben und schauen.

Stichpunkte 16 Dezember Petro

* QoS beweisen indem eigenen Port darstellen --> mit TCP anstatt UDP
* UDP ist eigentlich Standard Protocol für Videochat ect.
* Iperf beweis reicht teilweise
* Wäre ganz schick video konferenz tool umsetzen xmpp. Vielleicht wäre das sinnvoll (Vor Ort)
* Schauen, ob es nicht doch auf einer virtualbox doch geht

**Aufgabe 8**

Für Aufgabe acht muss bei Befehl einer Software, über die Controller-API ein bestimmter Flow, über bestimmte Knoten, vom Startpunkt zum Endpunkt, priorisiert geleitet werden.  StaticEntryPusher API

# Fazit

# Literaturverzeichnis

**Im aktuellen Dokument sind keine Quellen vorhanden.**

# Eidesstaatliche Erklärung