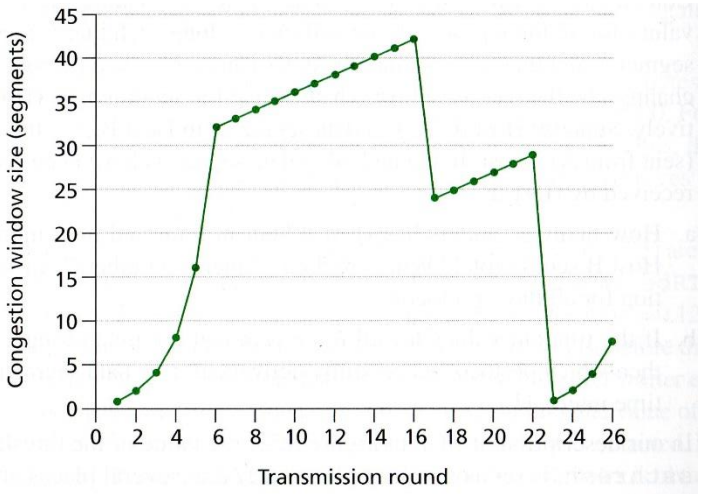
Übung 11: TCP Congestion Control und Wireshark-Analyse, SSH Tunnel

Aufgabe 1: TCP Congestion Control

Die Abbildung zeigt die Größe des Congestion Windows (`cwnd`) nach jeder „Übertragungsrunde“ (RTT Time). Es wird TCP Reno und Fast Retransmit eingesetzt. Ferner werden immer maximal große Pakete versendet (*Maximum Segment Size*).

a) In welchen Zeitintervallen ist *TCP Slow Start* aktiv?
b) In welchen Zeitintervallen wird *TCP Congestion Avoidance* verwendet?
c) 16. Runde: Wird Paketverlust durch Timeout oder durch 3 ACK-Duplikate erkannt?
d) 23. Runde: Wird Paketverlust durch Timeout oder durch 3 ACK-Duplikate erkannt?
e) Welchen Wert hat `ssthres` in der 1. Runde?
f) Welchen Wert hat `ssthres` in der 18. Runde?
g) Welchen Wert hat `ssthres` in der 24. Runde?
h) In welcher Runde wird das 17. Paket gesendet?



Aufgabe 2: TCP mit Wireshark

Der bereitgestellte Wireshark Trace schneidet das Hochladen einer Textdatei per „HTTP-Post“ auf.

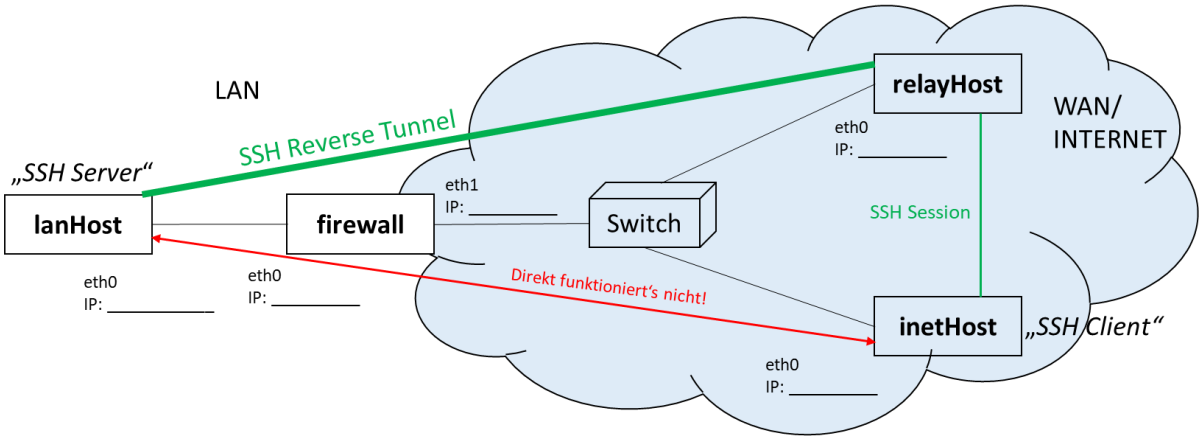
a) Welche IP Adressen und Portnummern hat TCP Client und TCP Server?
b) Welche Sequenznummer hat das erste Paket der TCP Verbindung? Woran kann man erkennen, dass es das erste Paket der Verbindung ist?
c) Welche Sequenz- und Acknowledgement-Nummer hat das dazugehörige `SYNACK` Paket?
d) Suchen Sie das `HTTP Post` Paket. Sie erkennen, dass das HTTP Paket wegen seiner Größe in mehrere TCP Pakete aufgeteilt wurde. Wie viele TCP Pakete wurden verwendet, um den ***HTTP Request*** zu übertragen?
Hinweis: Wireshark macht dies im zuletzt übertragenen TCP Paket durch eine „Zwischenschicht“ zwischen TCP und HTTP mit dem Namen „*Reassembled TCP Segments*“ deutlich.
e) In welchem TCP Segment steht der Text `HTTP Post`? *Tipp*: Ggfs. das „Packet Content Field“ ganz unten in Wireshark beachten.
f) Tragen Sie in folgende Tabelle die Paketnummern (aus Wireshark) und die TCP Sequenznummern der ersten 4 `HTTP-PORT`-Segmente vom Client zum Server ein. Geben Sie an, in welchem Paket der Server **erstmalig** den Erhalt bestätigt.

	Paketnummer in Wireshark	Sequenznummer	Paketnummer, in der Segment bestätigt wird.
1. Segment			
2. Segment			
3. Segment			
4. Segment			

g) Wie groß ist das Empfangsfenster (`rwnd`, Flusskontrolle), das der Server in der SYN ACK Nachricht an den Empfänger meldet.
h) Gab es in dem Trace Retransmissions? Begründen Sie ihre Antwort mit dem `Time-Sequence-Graph (Stevens)`, den Sie unter `Statistics → TCPStreamGraph` (Stevens) erstellen können.
Hinweis: Es muss in Wireshark ein dazugehöriges Datenpaket markiert sein.

Aufgabe 3: SSH-Tunnel (Hausaufgabe, Demo in Übung)

Der `lanHost` kann SSH Verbindungen zu anderen Hosts im Internet aufbauen, umgekehrt ist das nicht möglich. Stellen Sie sich vor, der `lanHost` ist per NAT mit dem Internet verbunden bzw. Firewalls untersagen den Verbindungsaufbau ins LAN. Eine Lösung für dieses Problem ist `SSH Reverse Tunneling`.



a) Vorbereitung
- Docker muss installiert sein, Details siehe Übung 07 und Übung 08.
- Dateien aus folgendem Archiv in ein Verzeichnis unter Linux extrahieren: https://syncandshare.lrz.de/dl/fiNSUM8U4TreyVATFRrXAxs9/RN_Uebung11-source.zip
- Ausführen: `sudo docker-compose up` in diesem Verzeichnis aus.
- Nur bei Problemen:
  `sudo docker stop $(sudo docker ps -aq); sudo docker system prune`

b) Eigene Shell für die folgenden Docker Container öffnen!
- `lanHost:` `sudo docker exec -it lanHost /bin/bash`
- `inetHost:` `sudo docker exec -it inetHost /bin/bash`
- `relayHost:` `sudo docker exec -it relayHost /bin/bash`
- `firewall:` `sudo docker exec -it firewall /bin/bash`

c) Tragen Sie alle IP-Adressen in das Diagramm ein.

d) Fügen Sie Default-Routen hinzu:
- `lanHost:` `ip route add default via <ip-eth0-firewall> dev eth0`
- `inetHost:` `ip route add default via <ip-eth1-firewall> dev eth0`
- `relayHost:` `ip route add default via <ip-eth1-firewall> dev eth0`

e) Testen: Der `lanHost` kann eine SSH Verbindung zum `inetHost` aufbauen, aber nicht umgekehrt. *Hinweis*: Jeder Host hat einen vorkonfigurierten SSH Server mit dem User „`sshuser`“ und dem Passwort „`ssh`“. Eine SSH Verbindung bauen Sie wie folgt auf: `ssh sshuser@<ip>`

f) Lösen Sie das Problem durch einen **Reverse Tunnel:** Der `lanHost` baut aktiv eine dauerhafte SSH-Verbindung zum einem dritten Host auf (`relayHost`). Letzterer muss aus dem Internet erreichbar sein und leitet Pakete, die er am Port `<remote-port>` empfängt an den `<local-Port>` auf dem `lanHost` weiter.
Kommando auf `lanHost`:
`ssh -N sshuser@<ip-relay> -R <remote-port>`[1]`:localhost:<local-port>`
Ein weiterer Host aus dem Internet (hier: `inetHost`) möchte nun über den `relayHost` eine SSH Verbindung zum `lanHost` aufbauen. Geben Sie hierzu den Port an (Option `-p`), testen Sie!

g) SSH Tunneling funktioniert für alle TCP-Verbindungen, nicht nur für SSH. Angenommen der `lanHost` betreibt auf Port 443 einen Webserver. Wie müssten die Kommandos aus Aufgabe f) lauten, damit der `inetHost` diesen über den `relayHost` erreichen kann? Nur angeben, nicht testen!

[1] Wählen Sie eine Portnummer > 50000.