

Praktikumsbericht 9 – IP/ICMP & TCP

Praktikum Teleinformatik 2



I-1d & T-1d
M. Heinzer
Michael.heinzer@edu.hefr.ch
B. Leutwiler
Bernhard.leutwiler@edu.hefr.ch

Inhaltsverzeichnis

1.	Einführung.....	2
2.	Messungen auf Schichte 3 (IP/ICMP)	2
2.1	Route IP (Traceroute)	2
2.2	IP Fragmentierung	6
2.3	Zusatzoptionen von IP/ICMP	11
3.	Messungen auf Schicht 4 (TCP)	12
3.1	TCP-Verbindung – der Dienst „finger“	12
3.2	TCP-Verbindung – der Dienst „chargen“	12
3.3	Rückweisung einer TCP-Verbindung	15
3.4	TCP Optionen.....	15
4.	Schlussfolgerung.....	16

1. Einführung

Unser neunter Laborbericht führt uns in die Protokolle IP, ICMP und TCP ein, von unserer Seite her waren schon Vorkenntnisse da, theoretische sowie auch praktische. Jedoch gab es wie immer Neues und Ungewohntes zu entdecken.

2. Messungen auf Schichte 3 (IP/ICMP)

2.1 Route IP (Traceroute)

P1: Welches sind die für diese Funktion verwendeten Protokolle?

Das Kommando „traceroute“ verwendet die Protokolle **ICMP** und **IP**, und damit auch alle im OSI-Modell darunterliegenden Protokolle.

Nr.	Schicht	Protokoll
3	Vermittlung	ICMP
		IP
2	Sicherung	Ethernet
1	Bitübertragung	

P2: Welches Feld des IP-Kopfes ist für diese Funktion entscheidend? Warum?

Ohne das Feld **TTL (Time To Live)** könnten diese Messungen kaum durchgeführt werden.

Indem der TTL Wert bei eins startet und schrittweise inkrementiert wird, wird Router für Router identifiziert. Denn wenn der TTL null erreicht, sendet der Router normalerweise eine Benachrichtigung an den Sender. Dieser Mechanismus erlaubt es die einzelnen Stationen welche das Paket weiterleiten zu identifizieren.

Ausser wenn einer oder mehrere Router dazu konfiguriert wurden keine Benachrichtigungen bei einem abgelaufenen TTL zu senden.

P3: Wie viele Sprünge (hops) ist das Ziel entfernt?

In unserem Fall ist die Station 160.98.8.105 genau **drei** Sprünge entfernt von unserer Station.

```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\lte>tracert 160.98.8.105
Tracing route to 160.98.8.105 over a maximum of 30 hops
  1  <1 ms    <1 ms    <1 ms    160.98.30.1
  2  <1 ms    <1 ms    <1 ms    160.98.44.1
  3  <1 ms    <1 ms    <1 ms    160.98.12.4
  4  <1 ms    <1 ms    <1 ms    160.98.8.105
Trace complete.
```

Router welche auf dem Weg passiert werden

Die vierte Station stellt das Ziel dar.

P4: Wie erhält man dieselben Informationen (wie bei *tracert*) auch mit ein paar „ping“-Befehlen?

Indem man bei dem Ping Befehl den Parameter TTL definiert. D.h. man sendet einen Ping an den Zielhost, aber mit einem TTL von eins:

```
ping -n 1 -i 1 160.98.8.105
```

Die Option „-i“ definiert das Feld TTL. Wenn man diesen nun auf eins setzt, erhält man eine Antwort vom ersten Router den das Paket passieren will. Den TTL erhört man nun immer ums eins, dass ergibt dann folgende Befehle:

```
ping -n 1 -i 2 160.98.8.105
ping -n 1 -i 3 160.98.8.105
ping -n 1 -i 4 160.98.8.105
```

Anmerkung: Um genau dasselbe Resultat zu erhalten wie bei „tracert“ muss noch die Option „-n 1“ entfernt werden, ansonsten wird nur ein Ping ausgeführt. Der Übersicht zuliebe haben wir aber bei dem folgenden Beispiel aber dies nicht durchgeführt:

```

C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\lte>ping -n 1 -i 1 160.98.8.105
Pinging 160.98.8.105 with 32 bytes of data:
Reply from 160.98.30.1: TTL expired in transit.
Ping statistics for 160.98.8.105:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Documents and Settings\lte>ping -n 1 -i 2 160.98.8.105
Pinging 160.98.8.105 with 32 bytes of data:
Reply from 160.98.44.1: TTL expired in transit.
Ping statistics for 160.98.8.105:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Documents and Settings\lte>ping -n 1 -i 3 160.98.8.105
Pinging 160.98.8.105 with 32 bytes of data:
Reply from 160.98.12.4: TTL expired in transit.
Ping statistics for 160.98.8.105:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Documents and Settings\lte>ping -n 1 -i 4 160.98.8.105
Pinging 160.98.8.105 with 32 bytes of data:
Reply from 160.98.8.105: bytes=32 time<1ms TTL=252
Ping statistics for 160.98.8.105:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Documents and Settings\lte>
    
```

Antwort des 1. Routers

Antwort des 2. Routers

Antwort des 3. Routers

Antwort des Ziels

P5: Dokumentieren und vergleichen sie die beiden Resultate.

Traceroute aus der Perspektive des PCs:

Das Netz von Switch

```

C:\WINDOWS\system32\cmd.exe
Tracing route to nox.deckpoint.ch [194.38.160.132]
over a maximum of 30 hops:
  0  <1 ms    <1 ms    <1 ms    160.98.30.1
  1  <1 ms    <1 ms    <1 ms    160.98.44.1
  2  <1 ms    <1 ms    <1 ms    160.98.12.4
  3  4 ms     1 ms     2 ms     160.98.6.1
  4  <1 ms    <1 ms    <1 ms    swifr1.unifr.ch [192.47.245.26]
  5  14 ms    6 ms     31 ms    swils2-g2-5.switch.ch [130.59.36.102]
  6  2 ms     2 ms     2 ms     swice2-10ge-1-3.switch.ch [130.59.37.1]
  7  2 ms     2 ms     2 ms     br00.cix.gva.fe0-1-1.ch.vtxnet.net [192.65.185.1]
  8  6 ms     6 ms     6 ms     gve-gix-cr-76-01-ge-2-5.vtxnet.net [212.147.63.3]
  9  3 ms     2 ms     3 ms     gve-gix-er-72-01-ge-0-2.vtxnet.net [212.147.11.2]
 10  6 ms     3 ms     6 ms     gve-dp7-er-73-01-ge-0-0.vtxnet.net [194.38.191.1]
 11  3 ms     3 ms     3 ms     gve-dp7-sw-35-01-v1-100.vtxnet.net [213.162.24.1]
 12  3 ms     3 ms     3 ms     nox.deckpoint.ch [194.38.160.132]
Trace complete.
    
```

Das Netz von vtxnet

Traceroute aus der Perspektive des Servers „traceroute.deckpoint.ch“:

Gateway	IP Address	AS Numbers	Min (ms)	Avg (ms)	Max (ms)	Lost (%)	
1	vl20-br00.dph.gva.ch.ixprime.net	194.38.160.156	AS12350	0.4	0.5	0.8	0%
2	gve-dp7-er-73-01-ge-2-0-100.vtxnet.net	213.162.24.129	AS12350	0.4	0.5	0.8	0%
3	gve-eq1-er-72-01-ge-0-1-115.vtxnet.net	194.38.191.118	AS12350	0.6	0.9	1.2	0%
4	zue-tix-er-72-01-ge-3-0.vtxnet.net	194.38.191.82	AS12350	5.1	5.3	5.7	0%
5	zue-tix-cr-76-02-ge-2-6.vtxnet.net	212.147.11.225	AS12350	5.2	5.5	5.9	0%
6	zue-tix-br-a1-01-ge-0-1-0.vtxnet.net	212.147.63.237	AS12350	5.1	5.4	5.7	0%
7	swiX2-10GE-4-2.switch.ch	91.206.52.53	?	5.0	5.3	5.8	0%
8	swiE22-10GE-1-3.switch.ch	130.59.36.249	AS559	5.2	5.6	5.9	0%
9	swiLS2-10GE-1-1.switch.ch	130.59.36.205	AS559	5.1	5.7	6.1	0%
10	swiFR1-G1-1.switch.ch	130.59.36.101	AS559	6.3	6.6	6.8	0%
11	hefr-ro01.unifr.ch	192.47.245.27	AS559	8.8	8.4	11.7	0%
12	160.98.6.196	160.98.6.196	AS559	6.4	6.7	7.1	0%

Info: customer@deckpoint.com - Tracpoint 0.17 Copyright © 1999-2002 Deckpoint ISP by Guy Baconnière.

Wie man anhand der Grafiken erkennt nehmen die Pakete unterschiedliche Routen. Zwar durchlaufen die beiden ungefähr die gleichen Netzwerke, Switch.ch und vtxnet.net, ohne jedoch die exakt gleichen Stationen zu verwenden.

Desweiteren kann man erkennen dass der Anfang und das Ende jeweils markant anders sind. Aus der Perspektive des Servers „traceroute.deckpoint.ch“ ist der Weg sogar um eine Station kürzer. Dies liegt daran, dass vermutlich die Schulfirewall oder ein anderes Zwischenelement die ankommende Anfrage blockiert, um die Stationen im Schulnetz vor Angriffen von aussen zu schützen.

P6: Allgemeine Frage: Wie werden die logischen Namen (DNS) der Router bestimmt?

Mittels des Befehls „nslookup“ können die Namen gefunden werden welche zu einer IP gehören. In dem folgenden Beispiel wurden die folgenden zwei Befehle eingegeben:

```

C:\Windows\system32\cmd.exe
C:\Users\heinzerm>ping www.switch.ch

Ping wird ausgeführt für aslan.switch.ch [130.59.108.36] mit 32 Bytes Daten:
Antwort von 130.59.108.36: Bytes=32 Zeit=210ms TTL=55
Antwort von 130.59.108.36: Bytes=32 Zeit=86ms TTL=55
Antwort von 130.59.108.36: Bytes=32 Zeit=77ms TTL=55
Antwort von 130.59.108.36: Bytes=32 Zeit=152ms TTL=55

Ping-Statistik für 130.59.108.36:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0 (0% Verlust),
    Ca. Zeitangaben in Millisek.:
        Minimum = 77ms, Maximum = 210ms, Mittelwert = 131ms

C:\Users\heinzerm>nslookup 130.59.108.36
Server: hefrkdc01.hefr.lan
Address: 160.98.2.11

Name:    aslan.switch.ch
Address: 130.59.108.36
  
```

Was dann folgende Pakete (gefiltert nach DNS) zur Folge hat:

160.98.173.30	160.98.2.11	DNS	Standard query A www.switch.ch
160.98.2.11	160.98.173.30	DNS	Standard query response CNAME aslan.switch.ch A 130.59.108.36
160.98.173.30	160.98.2.11	DNS	Standard query PTR 36.108.59.130.in-addr.arpa
160.98.2.11	160.98.173.30	DNS	Standard query response PTR aslan.switch.ch

Die ersten zwei Pakete stellen eine normale DNS-Anfrage dar, das zweite Paar hingegen ist ein sogenannter „reverse lookup“. D.h. man hat eine IP-Adresse und will den oder die dazugehörigen Namen.

P7: Allgemeine Frage: Wie sind Einträge vom Typ „*“ zu erklären?

Diese Einträge erscheinen auf der Liste wenn das Programm keine oder eine zu späte Antwort erhalten hat. Dies kann unterschiedliche Gründe haben, z.B. kann der Router so konfiguriert sein dass er keine Benachrichtigungen verschickt wenn eine Packet verworfen wird.

2.2 IP Fragmentierung

P8: Wie viele IP-Fragmente wurden für den Transport des Datenpakets gebraucht?

Das Datenpaket wurde in **drei** IP-Fragmente unterteilt. Dabei wurden die Daten wie folgt unterteilt:

- 1472 Bytes im ersten Fragment
- 1480 Bytes im zweiten Fragment
- 48 Bytes im dritten Fragment

Wir hatten uns gefragt warum nicht schon im ersten Fragment die theoretisch möglichen 1480 Bytes an Daten versendet wurden. Dies liegt daran dass beim ersten Fragment noch der ICMP-Header hinzugefügt wurde. Dieser hat eine Grösse von 8 Bytes, daher fehlt dieser Platz den Daten.

P9: Welches ist die maximale Grösse für Datenpakete in Ethernet-Netzwerken?

Die maximale Grösse für Datenpakete in Ethernet-Netzwerken beträgt **1500 Bytes**. Davon sind aber 20 Bytes für den IP Header reserviert, d.h. Es werden maximal 1480 Bytes an Daten übertragen pro Paket.

P10: Analysieren sie die Informationen zur IP-Fragmentierung in Ihrer Messung.

Die Resultate unserer Messung haben wir in die Untenstehende Tabelle übertragen:

Nr.	Identification	Fragment Offset	More Fragments	Length (Data)	Length (Fragment)
1	1483	0	1	1472	1500
2	1483	1480	1	1480	1500
3	1483	2960	0	48	68

Wie man anhand der Nummerierung erkennen kann wurde der Ping in drei Pakete fragmentiert. Die Felder haben folgende Bedeutung: Identität

Identification: Die Identifikationsnummer des IP Pakets.

Fragment Offset: Zeigt an in welcher Reihenfolge die Fragmente zusammengesetzt werden müssen.

More Fragments: Zeigt an ob noch mehr Fragmente zum Datenpaket gehören.

Length(Data): Die Daten in Bytes welche mit diesem Paket versendet wurden.

Length(Fragment): Die totale Länge des Fragments.

P11: Dokumentieren und kommentieren Sie das Resultat der Ausgabe des Befehls „netstat -s -p IP“ (vor und nach dem obigen „ping“).

Wir haben die gefragten Messungen durchgeführt und haben folgende Ausgaben erhalten:

```

C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\lte>netstat -s -p IP

IPv4 Statistics

Packets Received                = 5460
Received Header Errors          = 0
Received Address Errors        = 514
Datagrams Forwarded            = 0
Unknown Protocols Received     = 0
Received Packets Discarded     = 2
Received Packets Delivered     = 5297
Output Requests                = 2669
Routing Discards               = 0
Discarded Output Packets       = 0
Output Packet No Route        = 0
Reassembly Required           = 3
Reassembly Successful          = 1
Reassembly Failures            = 0
Datagrams Successfully Fragmented = 2
Datagrams Failing Fragmentation = 0
Fragments Created              = 6

C:\Documents and Settings\lte>ping -n 1 -l 3000 merlin.eif.ch

Pinging merlin.eif.ch [160.98.31.233] with 3000 bytes of data:
Request timed out.

Ping statistics for 160.98.31.233:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),

C:\Documents and Settings\lte>ping -n 1 -l 3000 merlin.eif.ch

Pinging merlin.eif.ch [160.98.31.233] with 3000 bytes of data:
Reply from 160.98.31.233: bytes=3000 time=2ms TTL=255

Ping statistics for 160.98.31.233:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 2ms, Average = 2ms

C:\Documents and Settings\lte>netstat -s -p IP

IPv4 Statistics

Packets Received                = 5529
Received Header Errors          = 0
Received Address Errors        = 519
Datagrams Forwarded            = 0
Unknown Protocols Received     = 0
Received Packets Discarded     = 2
Received Packets Delivered     = 5364
Output Requests                = 2719
Routing Discards               = 0
Discarded Output Packets       = 0
Output Packet No Route        = 0
Reassembly Required           = 6
Reassembly Successful          = 2
Reassembly Failures            = 0
Datagrams Successfully Fragmented = 4
Datagrams Failing Fragmentation = 0
Fragments Created              = 12

```

Vorher

Damit haben wir je 3 Fragmente erzeugt

Nachher (+6)

Wie man anhand der Grafik erkennt hat sich der Wert „Fragments Created“ von 6 auf 12 verdoppelt. Dies könnte daran liegen dass wir zwei Mal ein Paket mit 3000 Bytes, welches dann in 3 Fragmente zerlegt wird, versendet haben.

P12: Welches ist das Resultat des Befehls? Warum?

Wir haben für die Eingabe des folgenden Befehls:

```
Ping -n 1 -l 3000 -f 160.98.30.1
```

Diese Ausgabe als Resultat erhalten:

```
C:\Documents and Settings\lte>ping -n 1 -l 3000 -f 160.98.30.1
Pinging 160.98.30.1 with 3000 bytes of data:
Packet needs to be fragmented but DF set.
Ping statistics for 160.98.30.1:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
```

Das Netzwerk muss Pakete mit dieser Grösse fragmentieren, da wir aber das Flag „don't fragment“ auf „1“ gesetzt haben, wird es verworfen und der Ping geht verloren.

P13: Dokumentieren Sie die Ausgaben des Befehls „netstat -s -p IP“


Vorher:

```
IPv4-Statistik
Empfangene Pakete = 1021809
Empfangene Ursprungsfehler = 0
Empfangene Adressfehler = 2
Weitergeleitete Datagramme = 0
Empfangene unbekannte Protokolle = 0
Empfangene verworfene Pakete = 1310
Empfangene übermittelte Pakete = 1072938
Ausgabeanforderungen = 277201
Verworfenen Routingpakete = 0
Verworfenen Ausgabepakete = 3329
Ausgabepakete ohne Routing = 17
Reassemblierung erforderlich = 0
Reassemblierung erfolgreich = 0
Reassemblierung erfolglos = 0
Erfolgreiche Datagrammfragmentierung = 0
Erfolgreiche Datagrammfragmentierung = 0
Erzeugte Fragmente = 0
```

Nachher:

```

IPv4-Statistik
Empfangene Pakete = 1022002
Empfangene Vorspannfehler = 0
Empfangene Adressfehler = 2
Weitergeleitete Datagramme = 0
Empfangene unbekannte Protokolle = 0
Empfangene verworfene Pakete = 1313
Empfangene übermittelte Pakete = 1073207
Ausgabeanforderungen = 277464
Verworfenne Routingpakete = 0
Verworfenne Ausgabepakete = 3330
Ausgabepakete ohne Routing = 18
Reassemblierung erforderlich = 0
Reassemblierung erfolgreich = 0
Reassemblierung erfolglos = 0
Erfolgreiche Datagrammfragmentierung = 0
Erfolglose Datagrammfragmentierung = 0
Erzeugte Fragmente = 0
  
```



Unter Windows Vista (Wie im Bild oben): Der Wert „Verworfenne Ausgabepakete“ ist um eins angestiegen, weil der Befehl wie in der Abbildung unter P12 nicht funktioniert hat. Der Inhalt war grösser als 1472 Bytes und durfte nicht fragmentiert werden, dies führte zur Verwerfung des Pakets. Dass der Wert „Ausgabepakete ohne Routing“ in dieser Messung gleichzeitig anstieg scheint ein reiner Zufall zu sein, bei anderen Messungen war dies nicht der Fall.

Unter Windows XP: Der Wert „Erfolglose Datagrammfragmentierung“ steigt um 1. Was uns auch wesentlich logischer erscheint. Das Ganze ist im folgenden Bild dargestellt:

Voher:

```

IPv4-Statistik
Empfangene Pakete = 61
Empfangene Vorspannfehler = 0
Empfangene Adressfehler = 3
Weitergeleitete Datagramme = 0
Empfangene unbekannte Protokolle = 0
Empfangene verworfene Pakete = 3
Empfangene übermittelte Pakete = 58
Ausgabeanforderungen = 83
Verworfenne Routingpakete = 0
Verworfenne Ausgabepakete = 0
Ausgabepakete ohne Routing = 0
Reassemblierung erforderlich = 0
Reassemblierung erfolgreich = 0
Reassemblierung erfolglos = 0
Erfolgreiche Datagrammfragmentierung = 0
Erfolglose Datagrammfragmentierung = 0
Erzeugte Fragmente = 0
  
```



Nachher:

```
IPv4-Statistik
Empfangene Pakete = 589
Empfangene Vorspannfehler = 0
Empfangene Adressfehler = 3
Weitergeleitete Datagramme = 0
Empfangene unbekannte Protokolle = 0
Empfangene verworfene Pakete = 3
Empfangene übermittelte Pakete = 586
Ausgabeanforderungen = 551
Verworfenen Routingpakete = 0
Verworfenen Ausgabepakete = 0
Ausgabepakete ohne Routing = 0
Reassemblierung erforderlich = 0
Reassemblierung erfolgreich = 0
Reassemblierung erfolglos = 0
Erfolgreiche Datagrammfragmentierung = 0
Erfolglose Datagrammfragmentierung = 1
Erzeugte Fragmente = 0
```

+1

P14: Welches ist der maximale Wert von Lmax in „ping -n 1 -l Lmax -f 160.98.30.1“ Erklären sie warum Lmax < 1480.

Der maximale Wert von „Lmax“ berechnet sich wie folgt. 1500 Bytes sind die maximal mögliche Paketgröße mit Ethernet, von dieser Zahl muss noch folgendes abgezogen werden:

- 20 Bytes für den IP Header
- 8 Bytes für den ICMP-Header

Übrig bleiben somit 1472 Bytes. Diese Feststellung wird belegt durch unsere Versuche welche wie folgt abgelaufen sind:

```
C:\Documents and Settings\lte>ping -n 1 -l 1472 -f 160.98.30.1
Pinging 160.98.30.1 with 1472 bytes of data:
Reply from 160.98.30.1: bytes=1472 time<1ms TTL=255

Ping statistics for 160.98.30.1:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Documents and Settings\lte>ping -n 1 -l 1473 -f 160.98.30.1
Pinging 160.98.30.1 with 1473 bytes of data:
Packet needs to be fragmented but DF set.

Ping statistics for 160.98.30.1:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
```

1472 Bytes
funktionieren
noch

1473 Bytes
funktionieren
nicht mehr

2.3 Zusatzoptionen von IP/ICMP

P15: Analysieren sie die Rahmen, welche durch den Befehl „ping -r 5 160.98.8.105“ erzeugt werden.

Im „reply „ Paket hat es Optionen, im IP Paket, das sieht dann wie folgt aus:

```

Options: (24 bytes)
  Record route (23 bytes)
    Pointer: 24
    160.98.44.2
    160.98.12.1
    160.98.8.4
    160.98.8.105
    160.98.12.4
    EOL
  
```

0020	1e 19 07 17 18 a0 62 2c 02 a0 62 0c 01 a0 62 08	..b...b, ..b...b.
0030	04 a0 62 08 69 a0 62 0c 04 00 00 00 f0 5b 02 00	..b.i.b. ...[..
0040	63 00 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e	c.abcdef ghijklmn
0050	6f 70 71 72 73 74 75 76 77 61 62 63 64 65 66 67	opqrstuvwxyz wabcdefg
0060	68 69	hi

Wie man unschwer erkennen kann ist eine Anzahl IP-Adressen aufgezeichnet worden. Diese sind die vom Paket durchquerten Router. Die Grösse berechnet sich wie folgt:

$$\text{Grösse} = r * 4 \text{ Bytes} + 3 \text{ Bytes im Header} + 1 \text{ Byte EOL}$$

r = Anzahl Adressen

Die IP Option 7 sieht folgendermassen aus:

Byte	1	2	3	4-Variabel
Aufgabe	Type	Length	Pointer	Route Data

Type: Beinhaltet das Copy Flag (Muss diese Option in alle Fragmente geschrieben werden?), die „Class“ Option und „Option“ die Nummer der Option laut der IP Spezifizierung(in diesem Fall 7).

Length: Die totale Länge der IP Option in Bytes.

Pointer: Zeigt wo die Adressen im Paket geschrieben werden müssen

Route Data: Beinhaltet die Adressen welche aufgezeichnet werden + 1 Byte EOL

P16: Welches ist die Länge des IP-Paketkopfs? Erklären sie den hexadezimalen Wert des Feldes „header length“.

Das Problem

Hexadezimal: 4B -> Dezimal: 75 -> nicht die Bytes welche im Wireshark angegeben werden, Wireshark sagt uns 44 Bytes.

Die Lösung:

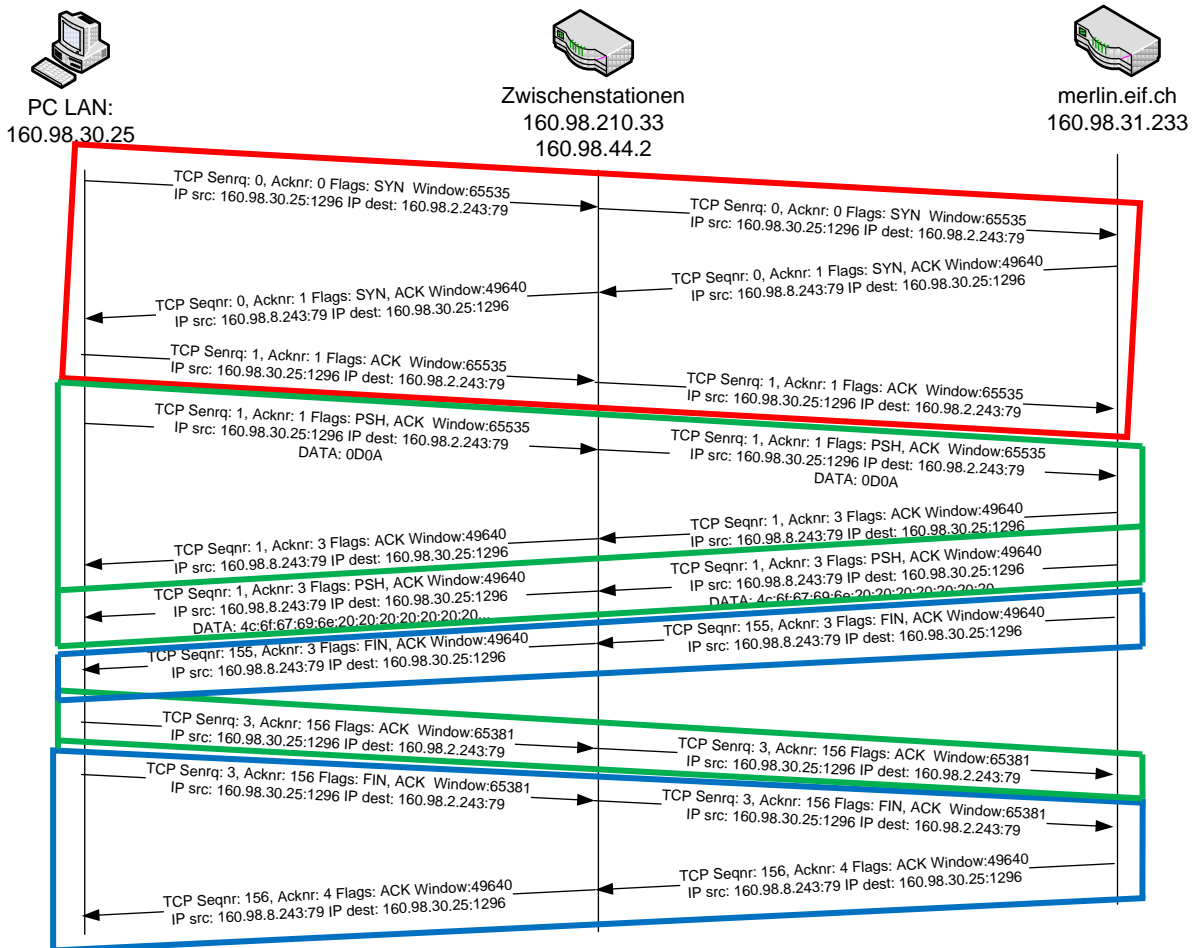
$$4 = \text{IP Version}, B = 11 * 4 = 44 \text{ Bytes}$$

Geprüft mit anderen Messungen, zweites Feld wird immer mit 4 multipliziert um die Länge zu erhalten.

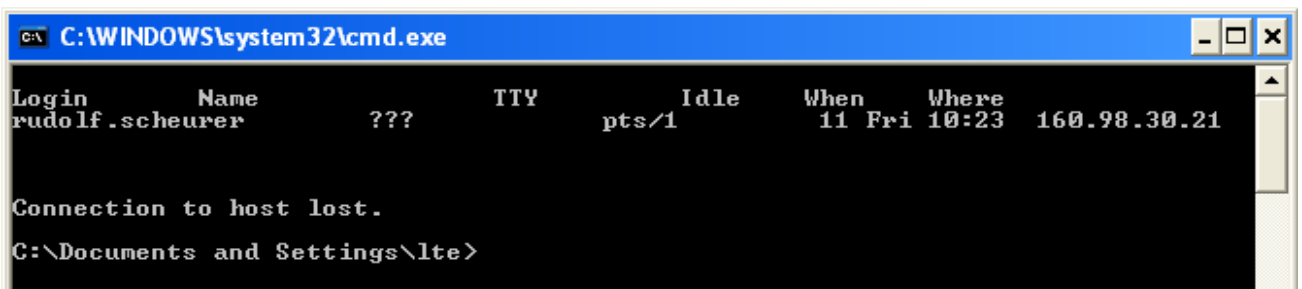
3. Messungen auf Schicht 4 (TCP)

3.1 TCP-Verbindung – der Dienst „finger“

P17: Dokumentieren und erklären Sie den gesamten Ablauf der TCP Verbindung.



Der „Finger“-Dienst erlaubt uns, die authentifizierten User auf einem Server anzuzeigen. Der Verbindungsablauf ist wie eine gewöhnliche TCP/IP verlaufen **3-way Handshake beim Verbindungsaufbau, Request-ACK Request-ACK, 3-way Handshake beim Verbindungsabbau).**



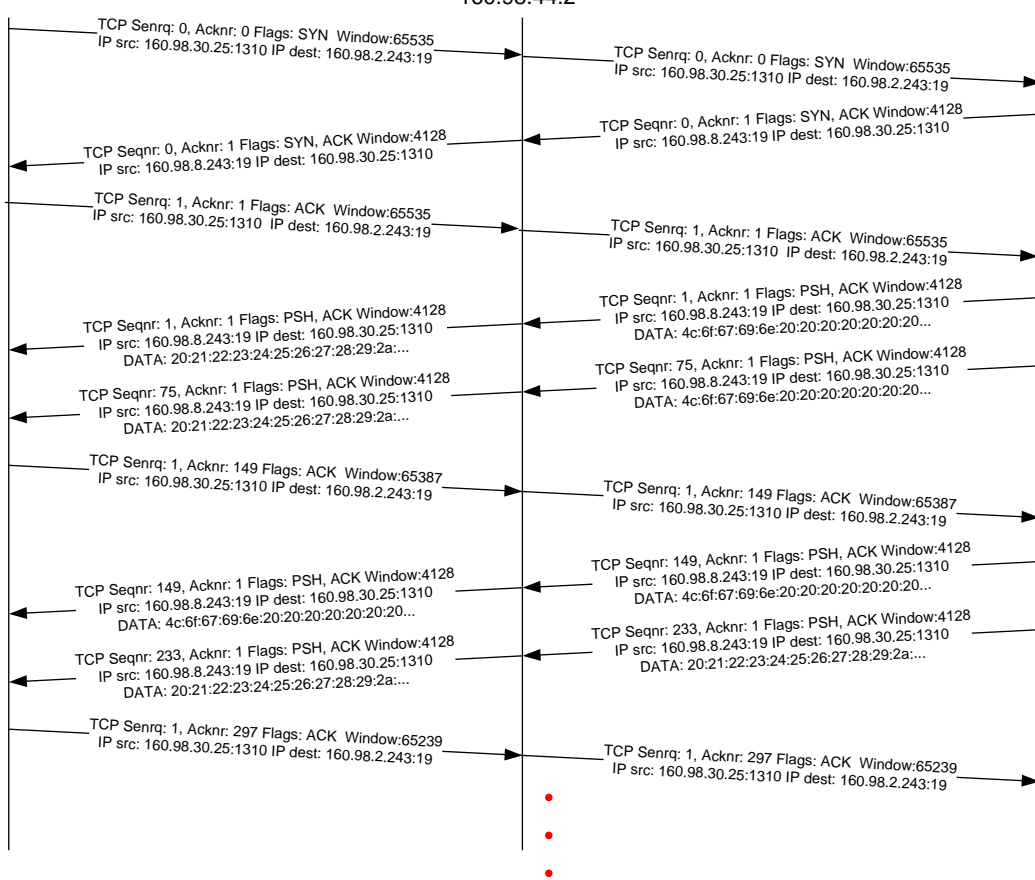
3.2 TCP-Verbindung – der Dienst „chargen“

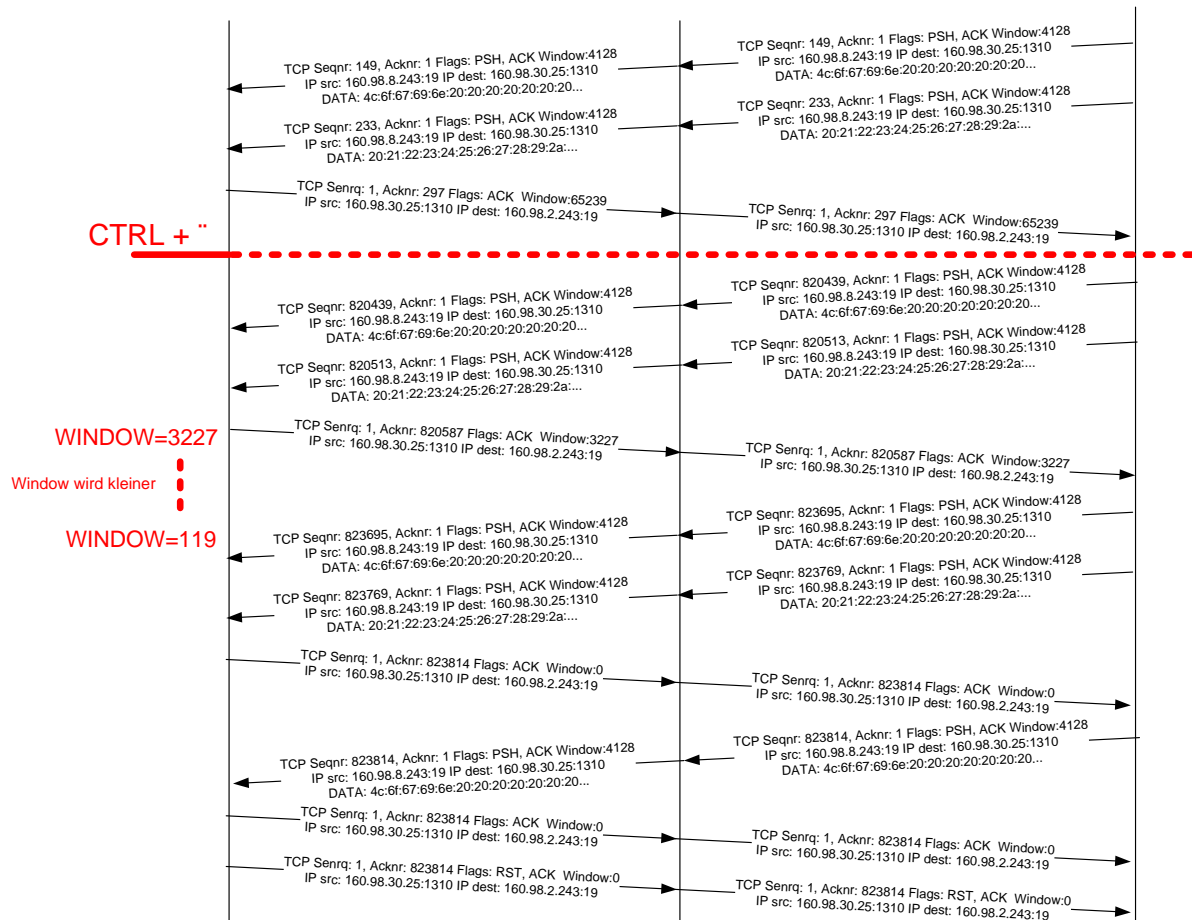
P18: Dokumentieren und erklären sie den TCP-Verbindungsaufbau, den Beginn des Datentransfers und den Abbau der TCP-Verbindung.

PC LAN:
160.98.30.25

Zwischenstationen
160.98.210.33
160.98.44.2

merlin.eif.ch
160.98.31.233





Der charges Dienst sendet bei Verbindung einen endlosen Character Strom an den Client. Die Verbindung verläuft am Anfang wie eine Standardmässige TCP/IP Verbindung (3-way Handshake beim Verbindungsaufbau, bestätigte Pakete während der Verbindung) wie in Frage 17 beschrieben. Der Verbindungsabbau hingegen funktioniert eher nach „Holzhackermethode“, da der Server keine Anweisungen, bzw. Kommandos vom Client entgegen nimmt. Dies hat zur Folge, dass der Client den Verbindungsabbruch künstlich erzeugen muss, indem er den Empfangsbuffer volllaufen lässt, und somit die Verbindung am Schluss bloss mit einem Reset Flag abgebrochen wird (und somit unbestätigt von Clientseite geschlossen wird).

P19: Dokumentieren Sie allfällige Anomalien/Auffälligkeiten während der Datentransferphase.

Diese Verbindung hat eine Anomalie im Verbindungsabbau. Da der Server (absichtlich) eine gewagte Konfiguration hat, dass er auf Anfrage sofort mit Senden beginnt, ohne auf ankommende „Befehle“ oder Requests zu reagieren, kann die Verbindung nicht TCP/IP Standard konform beendet werden. Dies weil der Server, da er nicht auf Requests reagiert, keine FIN Pakete beantwortet und somit die Verbindung bestehen bleibt. Daher ist die einzige Möglichkeit, die Verbindung per Reset Paket abzurechnen. Dadurch bleibt die Verbindung aber geöffnet, und bleibt eine Weile offen und ungenutzt.

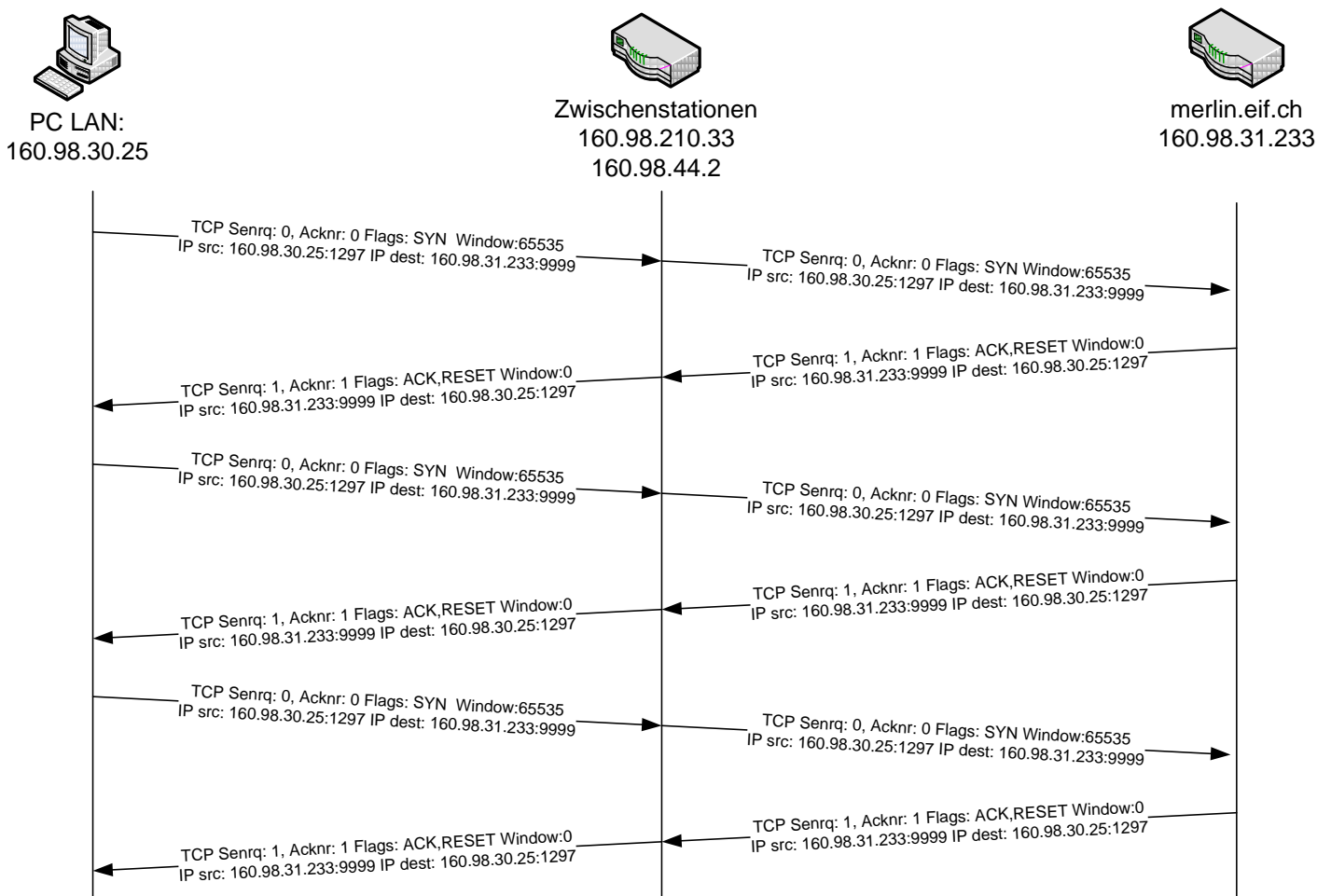
Der Effektive Verbindungsabbau ist bei Frage P18 näher erklärt.

P20: Analysieren und erklären Sie die sich verändernden Werte des Feldes „window“ in den TCP-Segmenten gegen Ende der Verbindung.

Das Window Feld bezeichnet die verfügbare Grösse des empfang Buffers auf Clientseite. Da der „chargen“ Dienst auf Serverseite auf keine Anweisungen wartet, bzw. auf keiner offenen Leitung horcht, sendet er ununterbrochen Daten zum Client. Im Normalfall ist das kein Problem, der Buffer ist längstens gross genug, um alle Daten aufzunehmen und zu Verarbeiten (Pendelt zwischen 65535 (max, 16 bit) und ca. 63000). Will man nun diesen Datenstrom anhalten, kann man auf Clientseite mit dem Kommando „CTRL+““ der Konsole mitteilen, dass der Buffer nicht mehr geleert wird, bzw. die darin befindlichen Daten nicht mehr verarbeitet werden. Dies führt dazu, dass das Window Feld immer kleiner wird (der Buffer füllt sich), und sobald der Buffer voll ist macht der Client automatisch einen Reset der Verbindung (es entsteht kein Verbindungsabbau per FIN-Flag).

3.3 Rückweisung einer TCP-Verbindung

P21: Dokumentieren und erklären sie den gesamten Ablauf der Messung.



Die Verbindung wird direkt bei Anfrage auf einem geschlossenen Port relativ brüsk beendet. Der Server sendet ein gesetztes RESET Flag zurück und setzt gleichzeitig die Window Size auf „0“. Der Client versucht die Anfrage insgesamt 3-mal zu machen, ehe er „einsieht“, dass der Server auf diesem Port keine Anfragen will.

3.4 TCP Optionen

P22: Wie weiss TCP, welches die maximale Segmentgrösse ist, so dass keine IP-Fragmentierung ausgelöst wird?

Beim Verbindungsaufbau (SYN = 1) hat es eine Rubrik „Options“ in welchem die maximale Segmentgrösse mitgeteilt wird. 1460 Bytes in unserem Fall.

```
Frame 3 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: IntelCor_1c:14:7f (00:1c:c0:1c:14:7f), Dst: Cisco_14:18:00 (00:02:7e:14:18:00)
Internet Protocol, Src: 160.98.30.25 (160.98.30.25), Dst: merlin.eif.ch (160.98.31.233)
Transmission Control Protocol, Src Port: sdproxy (1297), Dst Port: distinct (9999), Seq: 0, Len: 0
  Source port: sdproxy (1297)
  Destination port: distinct (9999)
  [Stream index: 1]
  Sequence number: 0 (relative sequence number)
  Header length: 28 bytes
  Flags: 0x02 (SYN)
  window size: 65535
  Checksum: 0xb60a [validation disabled]
  options: (8 bytes)
    Maximum segment size: 1460 bytes
    NOP
    NOP
    SACK permitted
```

4. Schlussfolgerung

Grössere Probleme gab es zum Glück nicht, nur die Headerlänge des ICMP Protokolls gab zu reden. Da auf der deutschen Website von Wikipedia nur 4 Bytes angegeben waren, Wireshark aber 8 Bytes angab kam es zu einer kurzzeitigen Verwirrung welche dann aber durch die englische Wikipedia Seite gelöst werden konnte.