Kommunikationstechnik

LAN-Analyse

Vorbereitung und Versuchsauswertung

Autor: Richard Grünert

9.4.2021

1 Vorbereitungsaufgaben

1.1 OSI-Referenzmodell

Das Ziel bei der Kommunikation zwischen zwei oder mehreren Geräten ist es, dass diese trotz Unterschiedlicher Hersteller und verschiedenen Technologien problemlos miteinander kommunizieren können. Um das zu erreichen, wurde das OSI-Referenzmodell (Open System Interconnection) entwickelt, welches die Kommunikation in 7 Teilaufgaben zerlegt, die als hierarchische Schichten (*layers*) dargestellt werden können.

Abbildung 1: Aufbau des OSI-Schichtenmodells mit (stark) vereinfachten Beschreibungen

1.1.1 Schicht 1: Übertragung

Die Übertragungsschicht (engl. *physical layer*) beschreibt als unterste Ebene im OSI-Modell die physikalische Netzanbindung, d.h. die Umsetzung der Daten (Bits) in die messbare/empfangbare physikalische Größe und deren Form, also leitungsgebunden, Funkübertragung oder Lichtwellenübertragung, sowie Modulationsart und auch Kabel, Verbinder etc.

1.1.2 Schicht 2: Sicherungsschicht

Aufgabe der Sicherrungsschicht (engl. *data-link layer*) ist die gewährleistung einer zuverlässigen, möglichst fehlerfreien Kommunikation. Sie regelt den Zugriff auf das Übertragungsmedium (Schicht 1) und führt Maßnahmen zu Flusskontrolle sowie Fehlerkontrolle durch Prüfsummen/Kanalkodierung durch. In dieser Schicht werden die Daten in *Frames* eingeteilt (z.B. Ethernet-Frames (siehe 1.6). Schicht 2 wird unterteilt in *LLC* und *MAC*:

2b) LLC, Logical Link Control
Schnittstelle zwischen Layer 3 und MAC

2a) MAC, Media Access Control
regelt zugriff auf gemeinsames Medium

1.1.3 Schicht 3: Vermittlungs- / Netzwerkschicht

Die Vermittlungsschicht (engl. *network layer*) ist dafür zuständig, die Daten *blockweise*, d.h. als sog. *Pakete* zwischen den Endsystemen weiterzuvermitteln. Außerdem ist sie für die Stauvermeidung (Congestion Control), sowie die Wegsuche zwischen den Zwischensystemen / Netzwerkknoten (Routing) zuständig. Schicht 3 stellt weiterhin Netzwerkadressen bereit (z.B. IP), aktualisiert Routing-Tabellen und fragmentiert die Datenpakete.

1.1.4 Schicht 4: Transportschicht

In der Transportschicht (engl. *transport layer*) wird der Datenstrom segmentiert, d.h. aufgeteilt auf mehrere Pakete (je nach Maximum Segment Size (MSS)), wobei jedem ein Header angefügt wird, welcher u.a. die Segmente nummeriert und das zugrundeliegende Protokoll (z.B. TCP oder UDP) sowie Ziel- und Quellport kennzeichnet. Mit Ziel und Quellport werden hier den segmentierten Datenpaketen Anwendungen auf den Endsystemen zugewiesen.

1.1.5 Schicht 5: Sitzungsschicht

Die Sitzungsschicht (engl. *session layer*), auch Kommunikationsschicht, regelt den Auf- u. Abbau von Kommunikationssitzungen und ermöglicht die Wiederherstellung dieser nach Abbrüchen (Synchronisation).

1.1.6 Schicht 6: Präsentationsschicht

Die Präsentationsschicht (engl. *presentation layer*) setzt die systemabhängige Darstellung der Daten (z.B. ASCII, EBCDIC) in eine unabhängige Form um, um den Datenaustausch zwischen unterschiedlichen Systemen zu ermöglichen. Außerdem fallen Aufgaben wie Datenkompression und -verschlüsselung in diese Ebene.

1.1.7 Schicht 7: Applikationsschicht

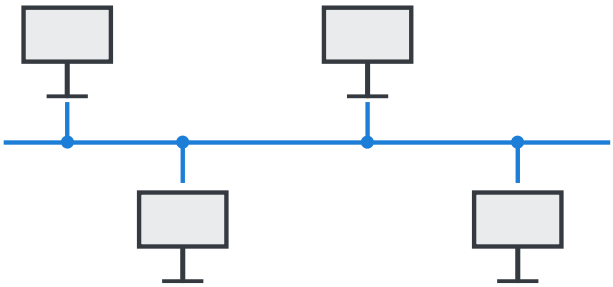
In der Anwendungsschicht befinden sich die Dienste der Anwendungsprogramme auf den jeweiligen Endsystemen. Hier findet man Protokolle wie HTTP, DNS, SMTP, FTP etc. Die Anwendungsprogramme selbst gehören nicht dazu.

1.2 Netztopologien

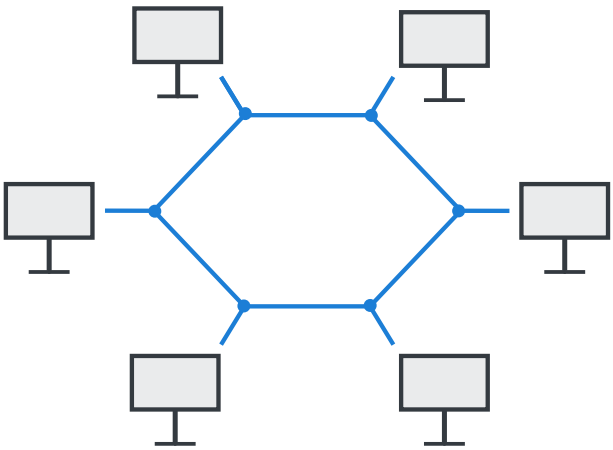
Unterschieden wird in *physikalische* Topologien (realer Aufbau des Netzwerkes, physische Verbindungen, etc.) und *logische* Topologien (prinzipielle Verbindungen der Teilnehmer ohne Details über physische Lage etc.). Je nach Topologie gibt es unterschiedliche Ausfallsicherheiten.

1.2.1 Linie

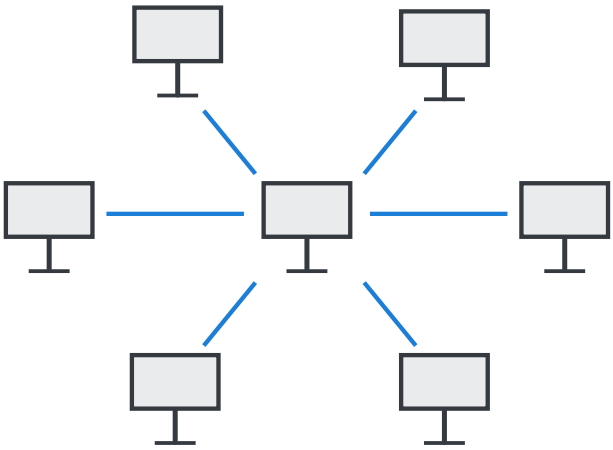
1.2.2 Bus



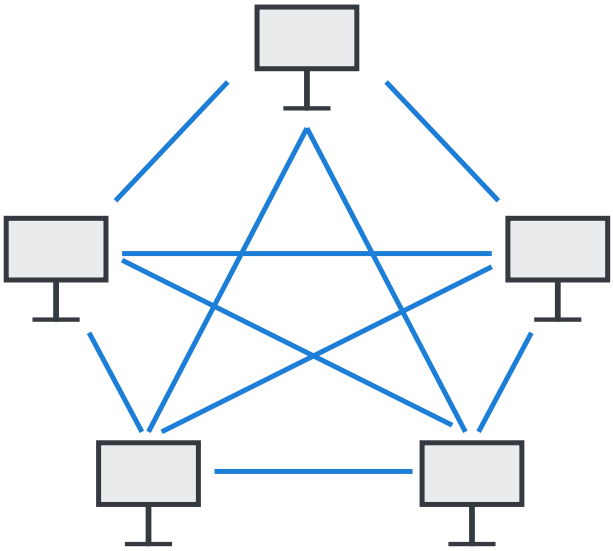
1.2.3 Ring



1.2.4 Stern



1.2.5 Vollvermaschung



1.3 Unterschiede zwischen Ethernet und Token Ring

Ethernet

- Bus-Topologie
- Schreibt, wenn Bus frei
- Bei Kollisionen → CSMA/Cx (Aufg. 1.4)

Token Ring

- Stern-Topologie (log. Ring-Topologie)
- Verwendet *Token* als Signal für Schreibrechte
- keine Kollisionen möglich

1.4 Ethernetzugriffsverfahren

Beim Ethernetzugriffsverfahren Carrier Sense Multiple Access *CSMA* (Mehrfachzugriff mit Trägerprüfung) prüfen alle Teilnehmer das Übertragungsmedium auf dessen Zustand. Ist das Medium frei, kann gesendet werden, ist es belegt, wird gewartet bis es frei ist. Das Medium gilt als frei, wenn es für 96 Bitzeiten nicht belegt ist (z.B. 960 ns bei 100 $\mathrm{Mbit\,s^{-1}}$).

Da nur lokal geprüft wird, kann bei der Ethernet- (Bus-) Übertragung eine Leitung durch Laufzeitunterschiede verschiedener Signale als frei erscheinen, obwohl auf der Leitung noch ein Signal „wandert“, wodruch fälschlicherweise gesendet wird und es es zu einer *Kollision* kommt. Das CSMA-Verfahren wird demnach weiterhin unterschieden nach der Art der Kollisionsbehandlung.

Collision Avoidance (CA): Ready to Send (RTS) Pakete gesendet bei Sendewunsch, Clear to Send (CTS) erhalten, falls frei. Genutzt in WLANs/Funk, da aufgrund der Reichweite keine komplette Überwachung des Mediums möglich.

Collision Detection (CD): Bei Kollision wird eine zufällige Zeit gewartet und dann erneut geprüft. Bei Überschreiten einer maximalen Anzahl von Versuchen tritt ein Fehler auf. Genutzt in LANs.

Collision Resolution (CR): Bei Kollision wird eine Prioritätsanalyse durchgeführt, wer zuerst angefangen hat, erhält das folgende Senderecht.

1.5 Ablauf einer HTTP-Verbindung

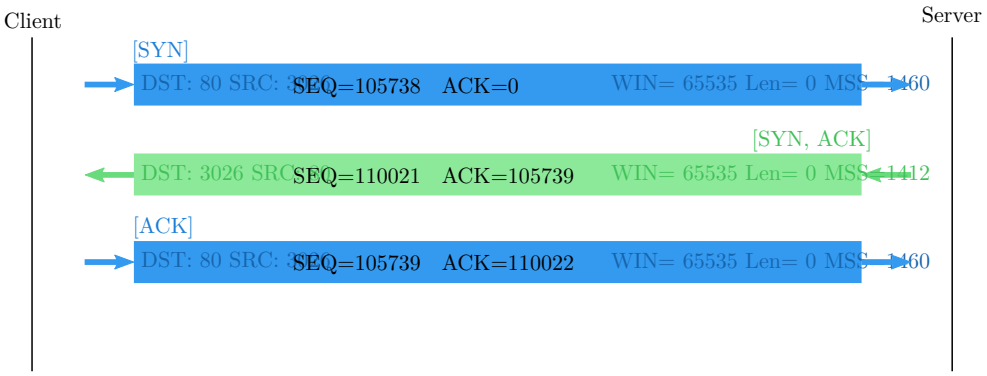
1.5.1 Allgemein

Am Beginn einer HTTP-Verbinung steht der Verbindungsaufbau über das TCP-Protokoll und den sogenannten *three-way-handshake*.

Schritt 1: SYN Der Client schickt ein TCP-Paket mit gesetztem SYN-Flag, um eine Verbindung mit dem Empfänger zu initialisieren und seine Sequenznummer für die Nummerierung der Pakete anzugeben.

Schritt 2: SYN-ACK Der Server antwortet auf Schritt 1 zum einen mit einem ACK, um das Paket (SYN) zu bestätigen und zum anderen mit einem SYN, um seine eigene Sequenznummer anzugeben.

Schritt 3: ACK Der Client bestätigt das SYN des Servers mit einem ACK.



1.5.2 OSI

Die Daten, die an den Server übertragen werden sollen und aus den Anwendungsschichten (7-5) stammen, werden in der Transportschicht (Schicht 4) segmentiert. Jedem Segment wird dann ein 20 bis 60 Byte langer TCP-Header angebracht, welcher u.a. Quell- und Zielport definiert. Der Zielport aus Client-Sicht ist hier der Standard HTTP-Port 80.

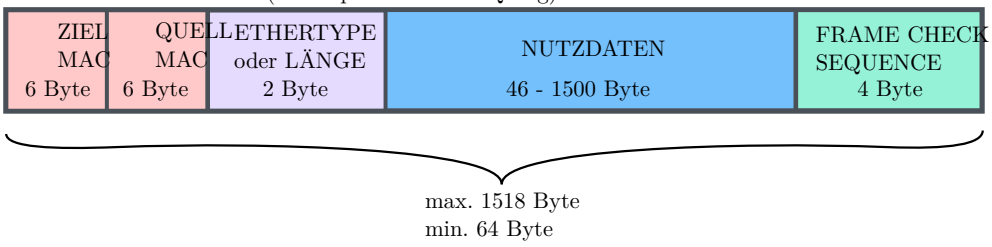
Das Resultat aus Schicht 4 ist dann ein Segment mit TCP-Header, das als Payload für die folgende Schicht 3 (Netzwerkschicht) in einen 20 bis 60 Byte langen IP-Header gebracht wird. Dieser gibt die eindeutigen Quell- und Ziel-IP-Adressen, sowie weitere Optionsfelder (z.B. time to live) an.

Das IP-*Paket* gelangt in Schicht 2, wo es zu einem Ethernet-*Frame* wird. Der Ethernet Header enthält u.a. die ebenfalls einzigartigen MAC-Quell- und Zieladressen (6 Byte) sowie weitere Felder (siehe 1.6).

Sowohl MAC- als auch IP-Adressen werden benötigt. Die MAC-Adressen geben den nächsten Sprung (*hop*) zur folgenden Hardware (z.B. den nächsten Router) an, während die IP-Adressen die Endsysteme eindeutig identifiziert.

1.6 Ethernetframes

Schicht 2 Ethernet Frame (ohne optionales 802.1Q-Tag)



Der Ethernetframe enthält Ziel- und Quell-MAC-Adresse sowie das Feld *Ethertype / Länge*, die Nutzdaten (z.B. IP-Paket) und eine Frame Check Sequence

(FCS).

1.6.1 MAC-Adressen

Die MAC-Adressen werden von der Schicht 2 verwendet, um Ziel- und Quell-Netzwerkschnittstellen (NICs) zu identifizieren.

1.6.2 Ethertype / Länge

Das Ethertype-Feld gibt die Protokollart der Daten im Nutzdatenfeld an, sofern dessen Wert größer gleich 1536 ist. Ist der Wert kleiner gleich 1500, wird damit die Länge der Daten spezifiziert.

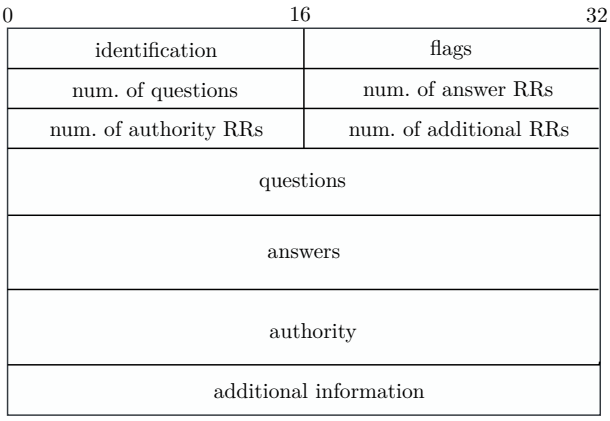
1.6.3 Frame Check Sequence (FCS)

Die Frame Check Sequence ist ein 4-Byte Cyclic Redundancy Check (CRC), der zur Fehlererkennung beim Empfänger genutzt werden kann. Der Wert dieses Feldes errechnet sich aus den Werten der anderen Felder.

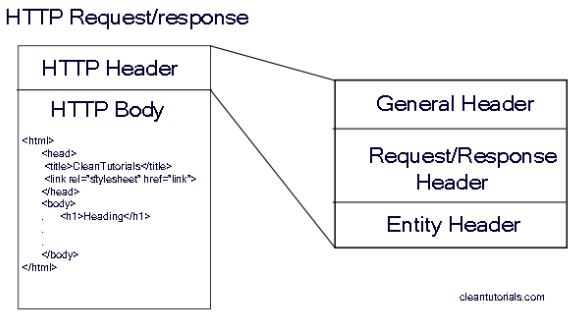
1.7 Internetprotokollapplikationen

Einige Beispiele für Internetprotokollapplikationen und deren Format.

1.7.1 DNS



1.7.2 HTTP



1.7.3 SSH

4 Byte	1 Byte	Variabel	4 Byte	20 Byte
Paketlänge	Padding-länge	Payload	Padding	MAC (message authentication code)

1.8 WIN-Kommandos

ARP-Tabelle:
- arp -a (Tabelle anzeigen)
- arp -d pc3 (Eintrag löschen)
- arp -s pc3 0:0:c:e:12:37 (Eintrag hinzufügen)

Routing Tabelle: netstat -r

Netzwerkinformationen: ipconfig

ping: ping 8.8.8.8

Weg eines Datenpaketes: tracert

1.9 Uni-, Multi-, Broadcast

Unicast: Eine Nachricht wird an einen *einzelnen Empfänger* gesendet

Multicast: Eine Nachricht wird an eine ausgewählte *Gruppe von Empfängern* gesendet (Vorteil: geringere Datenübertragungsrate notwendig, geringere Belastung des Mediums. Z.B. bei IP einstellbar über Adresse)

Broadcast: Eine Nachricht wird an *alle Teilnehmer* des Netzwerkes gesendet (sollte sich auf das eigende Netzsegment beschränken, d.h. nicht durch router weitergeroutet werden)

2 Versuchsaufgaben

2.1 Kennenlernen des Software-Analysators

Nach Öffnen von Wireshark müssen die Capture-Optionen konfiguriert werden. Diese enthalten alle physischen und auch virtuellen Netzwerkschnittstellen. Es muss die Schnittstelle ausgewählt werden, deren Netzwerkverkehr der Analysator aufzeichnen soll. Zudem kann der *Promiskuitive Modus* aktiviert werden, welcher dafür sorgt, dass auch Netzwerkverkehr aufgezeichnet wird, der nicht an die ausgewählte Schnittstelle gerichtet ist. Dies macht keinen Unterschied, wenn das Gerät an einen Switch angeschlossen ist, da dieser (i.d.R.) nur die wirkliche Zieladresse anspricht. In einem Funknetzwerk (WLAN) wäre der Mitschnitt des Verkehrs anderer mit dieser Einstellung jedoch möglich.

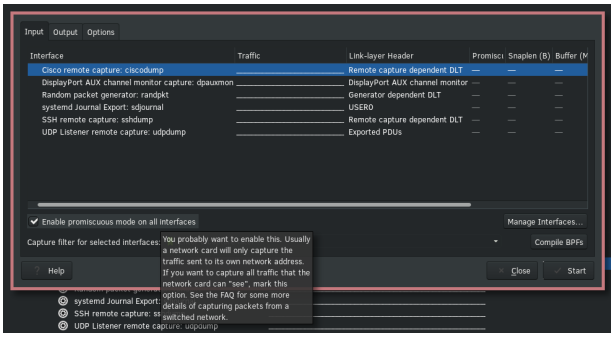


Abbildung 2: Auswahl der Netzwerkschnittstelle in den Capture-Optionen

Im Optionen-Reiter der Capture-Optionen kann außerdem eingestellt werden, ob die angezeigten Adressen (MAC, Netwerk/IP, Transport) in Namen aufgelöst werden sollen (sofern möglich).

Nach der Aufzeichnung im Labor konnten die Daten als Trace-Dateien abgespeichert werden, um später analysiert zu werden. Zur Vereinfachung der Analyse kann man mithilfe von Filtern die Paketliste u.a. auf bestimmte Protokolle einschränken. Beispielsweise wird mit dem Filter `tcp.port == 80` nur der Verkehr mit den TCP-Portnummern 80 (HTTP) angezeigt.

2.2 Testen der Netzwerkkonfiguration

2.2.1 ipconfig

```
C:\Users\Student>ipconfig

Windows-IP-Konfiguration


Ethernet-Adapter Ethernet:

   Verbindungsspezifisches DNS-Suffix:
   Verbindungslokale IPv6-Adresse  . : fe80::4c04:70a5:55bf:b380%20
   Standardgateway . . . . . . . . . :

Ethernet-Adapter Ethernet-Intern:

   Verbindungsspezifisches DNS-Suffix:
   Verbindungslokale IPv6-Adresse  . : fe80::c5d6:6393:a556:db37%29
   IPv4-Adresse  . . . . . . . . . . : 192.168.50.16
   Subnetzmaske  . . . . . . . . . . : 255.255.255.0
   Standardgateway . . . . . . . . . : 192.168.50.1

Ethernet-Adapter Ethernet2-DSL:

   Verbindungsspezifisches DNS-Suffix: mshome.net
   Verbindungslokale IPv6-Adresse  . : fe80::2c00:dfeb:b8db:914%19
   Standardgateway . . . . . . . . . :

C:\Users\Student>arp -a
```

Abbildung 3: Ausgabe des CMD-Befehls *ipconfig*

Der Befehl `ipconfig` zeigt die IP-Konfigurationen der Netzwerkschnittstellen unter Windows an. Für die folgenden Betrachtungen ist das Interface „Ethernet-Adapter Ethernet-Intern“ relevant. Aus Abbildung 3 kann man dessen zugewiesene IP-Adresse `192.168.20.16` , welche das Interface im Netzwerk für IP-Pakete identifiziert. Ob diese statisch oder dynamisch (DHCP) zugewiesen wurde, ist hieraus nicht erkennbar.

Außerdem angezeigt ist die Subnetzmaske `255.255.255.0` . Diese trennt den IP-Adressbereich in Netzwerk- und Hostbereich auf, um im gleichen physischen Netzwerk mehrere Unternetze (Subnets) zu erzeugen. Ziel-IP-Adresse und Subnetzmaske werden logisch UND-verknüpft, wodurch man die Netzadresse des Ziels erhält. Ist das Ergebnis gleich der eigenen Netzadresse, kann das Paket direkt an den entsprechenden Host versandt werden. Stimmt es nicht überein, wird es an den *Default Gateway* gesandt, welcher das Paket weiterleitet.

Die Netzadresse der Schnittstelle aus Abb. 3 ist also `192.168.50.0/24` (x.x.x.0 immer reserviert als Netzadresse), die Adresse des Standardgateways ist `192.168.50.1` .

2.2.2 arp -a

```
C:\Users\Student>arp -a

Schnittstelle: 192.168.50.16 --- 0x1d
  Internetadresse       Physische Adresse     Typ
  192.168.50.1          f0-7f-06-1b-7a-01     dynamisch
  192.168.50.13         98-fa-9b-6a-cc-7a     dynamisch
  192.168.50.14         98-fa-9b-72-a1-57     dynamisch
  192.168.50.17         98-fa-9b-6a-b0-7a     dynamisch
  192.168.50.19         98-fa-9b-6c-a3-5a     dynamisch
  192.168.50.255        ff-ff-ff-ff-ff-ff     statisch
  224.0.0.9             01-00-5e-00-00-09     statisch
  224.0.0.22            01-00-5e-00-00-16     statisch
  224.0.0.251           01-00-5e-00-00-fb     statisch
  224.0.0.252           01-00-5e-00-00-fc     statisch
  239.255.255.250       01-00-5e-7f-ff-fa     statisch
  255.255.255.255       ff-ff-ff-ff-ff-ff     statisch

C:\Users\Student>
```

Abbildung 4: Ausgabe des CMD-Befehls arp -a

Mit dem Befehl `arp -a` (ARP: Address Resolution Protocol) kann unter Windows die Routingtabelle (Cache) angezeigt werden. Diese enthält die Zuweisungen der IP-Adressen zu den MAC- bzw. physischen Adressen, sowie, ob diese dynamisch oder statisch eingerichtet wurden. Diese Tabelle wird von Schicht 2 genutzt, um Ziel-IP-Adressen (Schicht 3) in (lokale) Hardwareadressen zu übersetzen.

In Abbildung 4 erkennt man die Einträge der im gleichen Netz (`192.168.50.0`) befindlichen Geräte. Dazu zählen sowohl weitere Hosts, wie z.B. `192.168.50.17` , als auch der Default Gateway und die Broadcast-Adresse `192.168.50.255` / `ff-ff-ff-ff-ff-ff` , welche u.a. für ARP-Anfragen verwendet wird, um eine IP-Adresse in eine physische Adresse umzuwandeln, sollte sie nicht in der Tabelle aufgeführt sein. Die Broadcast-Adresse für einen Host lässt sich berechnen: Zuerst wird die Subnetzmaske negiert und dann ODER-verknüpft mit der Host-Adresse.

$$\begin{aligned} & 192.168.50.16 = 11000000.10101000.00110010.00010000 \\ \text{OR}\ & 255.255.255.0 = 00000000.00000000.00000000.11111111 \\ \hline = \ & 192.168.50.255 = 11000000.10101000.00110010.11111111 \end{aligned}$$

Das Ergebnis deckt sich mit der Adresse aus Abbildung 4.

Auffällig sind weiterhin die folgenden `224.0.0.x` -Adressen, sowie die Adresse `239.255.255.250` . Diese sind Multicast-Adressen (siehe 1.9). IPv4 reserviert einen Multicast-Adressbereich von `224.0.0.0` bis `239.255.255.255` , in welchen diese Adressen gerade fallen.

Da der letzte Eintrag ebenfalls auf die physische Adresse `ff-ff-ff-ff-ff-ff` abgebildet ist, ist klar, dass es sich hierbei ebenfalls um eine Broadcast-Adresse handeln muss, sie ist ebenfalls reserviert.

2.3 Frameanalyse (Ethernet)

2.3.1 Analyse mit CMD

```
Schnittstelle: 192.168.50.16 --- 0x1d
  Internetadresse       Physische Adresse     Typ
  192.168.50.1          f0-7f-06-1b-7a-01     dynamisch
  192.168.50.13         98-fa-9b-6a-cc-7a     dynamisch
  192.168.50.14         98-fa-9b-72-a1-57     dynamisch
  192.168.50.17         98-fa-9b-6a-b0-7a     dynamisch
  192.168.50.19         98-fa-9b-6c-a3-5a     dynamisch
  192.168.50.255        ff-ff-ff-ff-ff-ff     statisch
  224.0.0.9             01-00-5e-00-00-09     statisch
  224.0.0.22            01-00-5e-00-00-16     statisch
  224.0.0.251           01-00-5e-00-00-fb     statisch
  224.0.0.252           01-00-5e-00-00-fc     statisch
  239.255.255.250       01-00-5e-7f-ff-fa     statisch
  255.255.255.255       ff-ff-ff-ff-ff-ff     statisch

C:\Windows\system32>arp -d

C:\Windows\system32>arp -a

Schnittstelle: 192.168.50.16 --- 0x1d
  Internetadresse       Physische Adresse     Typ
  224.0.0.22            01-00-5e-00-00-16     statisch

C:\Windows\system32>
```

Abbildung 5: Löschen des ARP-Caches und folgendes Anzeigen der ARP-Tabelle

Im ersten Schritt wurde die ARP-Tabelle über die Eingabeaufforderung mithilfe von `arp -d` vollständig gelöscht (Abb. 5). Danach wurde der Nachbarrechner mit der IP-Adresse `192.168.50.14` mit dem *ping*-Befehl angesprochen. Der Befehl hat den Zielrechner zuerst nach 4 Versuchen nicht erreicht, was daran lag, dass die Firewall am Ziel nicht deaktiviert war und den ping nicht durchgelassen hat. Nach Deaktivierung der Firewall lieferte der Befehl erfolgreiche Ergebnisse (Abb. 6).

```
C:\Windows\system32>ping 192.168.50.14

Ping wird ausgeführt für 192.168.50.14 mit 32 Bytes Daten:
Antwort von 192.168.50.14: Bytes=32 Zeit<1ms TTL=128
Antwort von 192.168.50.14: Bytes=32 Zeit<1ms TTL=128
Antwort von 192.168.50.14: Bytes=32 Zeit<1ms TTL=128
Antwort von 192.168.50.14: Bytes=32 Zeit<1ms TTL=128

Ping-Statistik für 192.168.50.14:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0
    (0% Verlust),
Ca. Zeitangaben in Millisek.:
    Minimum = 0ms, Maximum = 0ms, Mittelwert = 0ms
```

Abbildung 6: Ping des Nachbarrechners 192.168.50.14

Nach dem ping-Befehl wurde erneut die ARP-Tabelle angezeigt (Abb 7). Man kann erkennen, dass die IP-Adresse des Nachbarrechners in den Cache aufgenommen wurde. Außerdem wurde auch eine andere Adresse als Resultat des Empfangs einer ping-Nachricht aufgenommen, nämlich die `192.168.50.17` .

```
C:\Users\Student>arp -a

Schnittstelle: 192.168.50.16 --- 0x1d
  Internetadresse       Physische Adresse     Typ
  192.168.50.1          f0-7f-06-1b-7a-01     dynamisch
  192.168.50.14         98-fa-9b-72-a1-57     dynamisch
  192.168.50.17         98-fa-9b-6a-b0-7a     dynamisch
  192.168.50.255        ff-ff-ff-ff-ff-ff     statisch
  224.0.0.22            01-00-5e-00-00-16     statisch

C:\Users\Student>
```

Abbildung 7: ARP-Tabelleneinträge nach Ausführen des ping-Befehls

Da der ping-Befehl, wie auch alle anderen Daten, durch Schicht 1 und 2 läuft, benötigt er die MAC-Adressen des Ziels, muss also die angegebene IP-Adresse übersetzen. Da die ARP-Tabelle gelöscht wurde, kann diese nicht zum Nachschlagen verwendet werden, weshalb ein ARP-Broadcast an alle Teilnehmer des Netzwerkes gesendet werden muss, um anzufragen, welche MAC-Adresse zum Host `192.168.50.14` gehört. Die notwendige Broadcast-Adresse ist wie in 2.2.2 ermittelbar, daher erscheint sie auch in Abb. 7.

Die Multicast-Adresse `224.0.0.22` scheint fest eingebaut zu sein, da diese bereits in Abb. 5 unmittelbar nach dem Löschen des ARP-Caches zu sehen war.

2.3.2 Analyse mit Wireshark

Mithilfe des Filters `arp || icmp` können nur die Pakete der für die Auswertung relevanten Protokolle betrachtet werden.

4	2.490860	LCFCHeFe_6c:a2:21	Broadcast	ARP	42 Who has 192.168.50.14? Tell 192.168.50.16
5	2.491407	LCFCHeFe_72:a1:57	LCFCHeFe_6c:a2:21	ARP	60 192.168.50.14 is at 98:fa:9b:72:a1:57
6	2.491451	192.168.50.16	192.168.50.14	ICMP	74 Echo (ping) request id=0x0001, seq=69/17664, ttl=128 (reply in 7)
7	2.491761	192.168.50.14	192.168.50.16	ICMP	74 Echo (ping) reply id=0x0001, seq=69/17664, ttl=128 (request in 6)
8	2.494117	LCFCHeFe_72:a1:57	LCFCHeFe_6c:a2:21	ARP	60 192.168.50.14 is at 98:fa:9b:72:a1:57
9	2.494379	LCFCHeFe_72:a1:57	LCFCHeFe_6c:a2:21	ARP	60 192.168.50.14 is at 98:fa:9b:72:a1:57
10	2.496176	LCFCHeFe_72:a1:57	LCFCHeFe_6c:a2:21	ARP	60 192.168.50.14 is at 98:fa:9b:72:a1:57
11	3.504591	192.168.50.16	192.168.50.14	ICMP	74 Echo (ping) request id=0x0001, seq=70/17920, ttl=128 (reply in 12)
12	3.505284	192.168.50.14	192.168.50.16	ICMP	74 Echo (ping) reply id=0x0001, seq=70/17920, ttl=128 (request in 11)
14	4.521419	192.168.50.16	192.168.50.14	ICMP	74 Echo (ping) request id=0x0001, seq=71/18176, ttl=128 (reply in 15)
15	4.522194	192.168.50.14	192.168.50.16	ICMP	74 Echo (ping) reply id=0x0001, seq=71/18176, ttl=128 (request in 14)
16	5.536807	192.168.50.16	192.168.50.14	ICMP	74 Echo (ping) request id=0x0001, seq=72/18432, ttl=128 (reply in 17)
17	5.537520	192.168.50.14	192.168.50.16	ICMP	74 Echo (ping) reply id=0x0001, seq=72/18432, ttl=128 (request in 16)

Abbildung 8: Gefilterte Ausgabe von Wireshark bei senden des ping-Kommandos

ARP In Abb. 8 erkennt man ersteinmal, dass genau 8 ICMP-Pakete aufgezeichnet wurden. Davon sind 4 Anfragen und 4 Antworten. Dies deckt sich mit Abb. 6, da der ping-Befehl genau 4 Mal gesendet wurde.

Vor Beginn des eigentlichen Pings findet man ein ARP-Paket. Hierbei handelt es sich um eine Broadcast-Nachricht zum Bestimmen der MAC-Adresse des Ziels. Dies ist notwendig, da vorher der ARP-Cache gelöscht wurde und

sich dort kein Eintrag zur Ziel-IP-Adresse befindet (vgl. 2.3.1).

Betachtet man die ARP-Anfrage genauer (Abb. 9), erkennt man im Ethernet-Frame, dass die Ziel-MAC-Adresse die bereits genannte `ff:ff:ff:ff:ff:ff` ist, welche für den Broadcast reserviert ist. Aus dem Quell-Feld erkennt man nun auch die Adresse der eigenen Netzwerkschnittstelle, nämlich `98:fa:9b:6c:a2:21` . Weiterhin ist im Header, wie zu erwarten, der Ethertype ARP, angegeben.

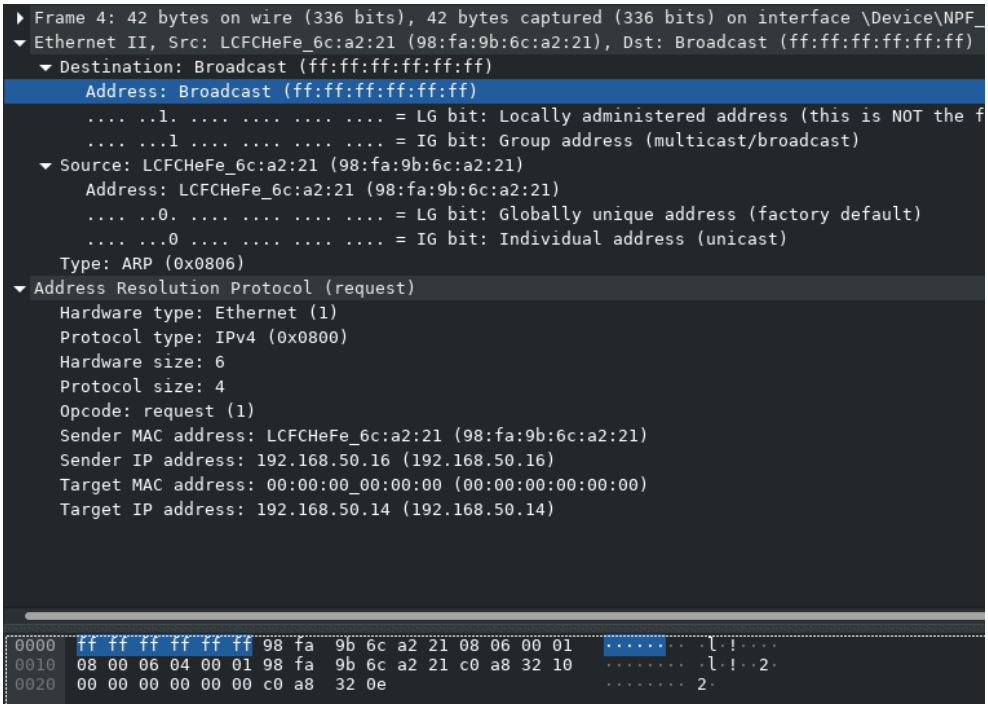


Abbildung 9: Genauere Analyse der ersten ARP-Anfrage

Im folgenden Nutzdatenfeld befindet sich die ARP-Anfrage. Hier steht der Opcode 1 um zu signalisieren, dass es sich um eine Anfrage handelt. Außerdem enthält es Sender-MAC und -IP Adressen, sowie die bekannte Ziel-IP-Adresse. Das Feld Ziel-MAC-Adresse ist frei, da genau diese ja nachgefragt werden soll. Der angesprochene Host (sofern existent) sollte antworten und beide Felder ausfüllen. Dies geschieht dann auch in der zweiten Zeile von Abb. 8, in welcher die ARP-Antwort empfangen wurde.

Es antwortet also genau der Host auf den Broadcast mit einer ARP-Nachricht, dem die angefragte MAC-Adresse gehört. In Abb. 10 ist der genaue Inhalt der ARP-Anwort dargestellt. Die Ziel-MAC ist nun nicht mehr der Broadcast, sondern die Adresse des Anfragers, die aus der Anfrage entnommen werden konnte. Mit dem Empfang dieser Daten ist das Ziel des ping-Befehls identifiziert.

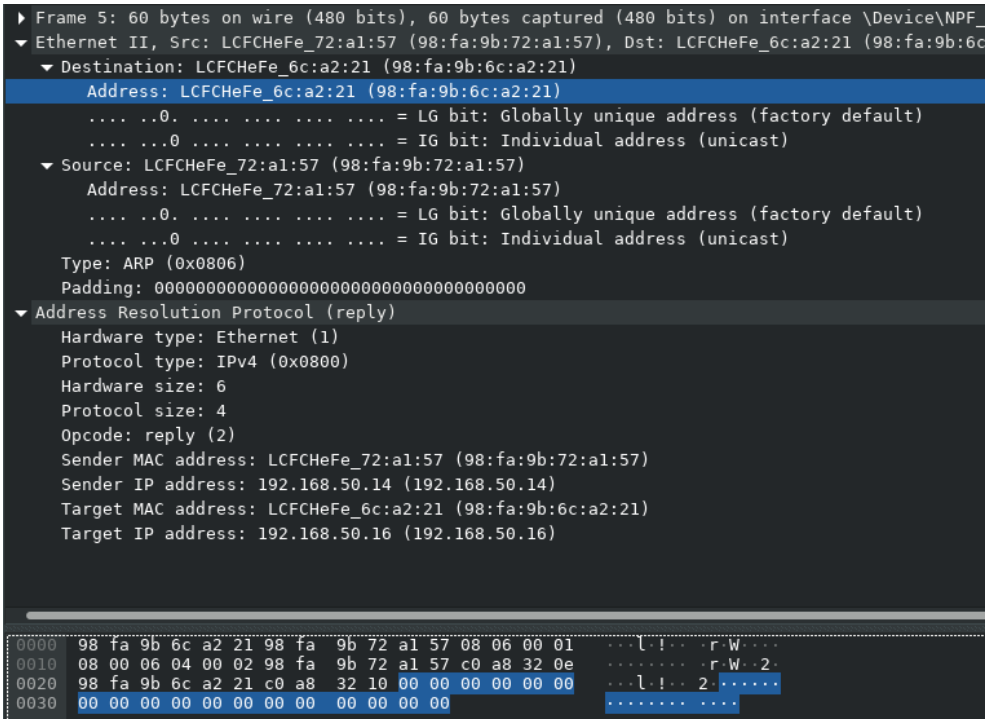


Abbildung 10: Antwort auf die ARP-Anfrage

Auffällig ist, dass in der Antwort aus Abb. 10 im Ethernetframe zusätzlich ein 18-Byte langes Padding-Feld aus Nullen hinzugefügt wurde. Dies wird benötigt, um die minimale Länge des Frames von 64 Byte zu erreichen. In Wireshark sind jedoch nur 60 Byte angezeigt, was daran liegt, dass das 4 Byte lange FCS-Feld zur Fehlererkennung mittels CRC automatisch weggelassen wird.

Dass die Framegröße in Abb. 9 nur 42 und nicht 60 Byte lang ist, liegt daran, dass es sich um ein Paket handelt, das von der Schnittstelle gesendet wurde, die auch aufgezeichnet wird. In diesem Fall werden die Daten aufgezeichnet, bevor das Padding angefügt wird.

ICMP Da der ping-Befehl das ICMP-Protokoll verwendet, findet man dieses auch im Wireshark-Trace wieder, z.B. bei der ersten Anfrage (Abb. 12). Die allgemeine Struktur eines ICMP-Paketes ist in Abb. 11 zu sehen.

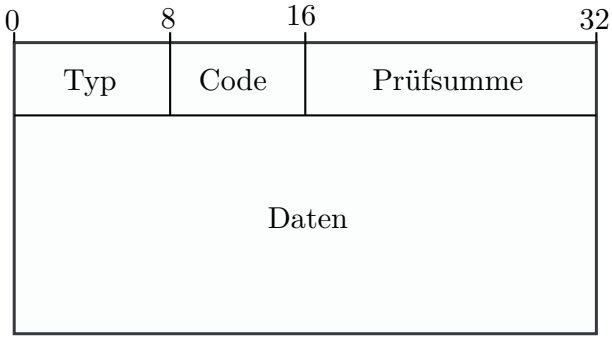


Abbildung 11: Struktur des ICMP-Paketes

Das Typ-Feld kann Werte von 0-255 annehmen und gibt die Art der Kontrollnachricht[1] an. Beispielsweise bedeutet der Typ-Wert 0 eine Echo-Reply Nachricht (Echo Antwort, verwendet von ping) und der Wert 8 ein Echo-Request (Echo-Anfrage). Das Code-Feld gibt hierbei zusätzliche Informationen zum jeweiligen Typ-Feld. Bei den ICMP-Typen des ping-Befehls, also Echo-Reply und -Request ist der Inhalt des Code-Feldes 0.

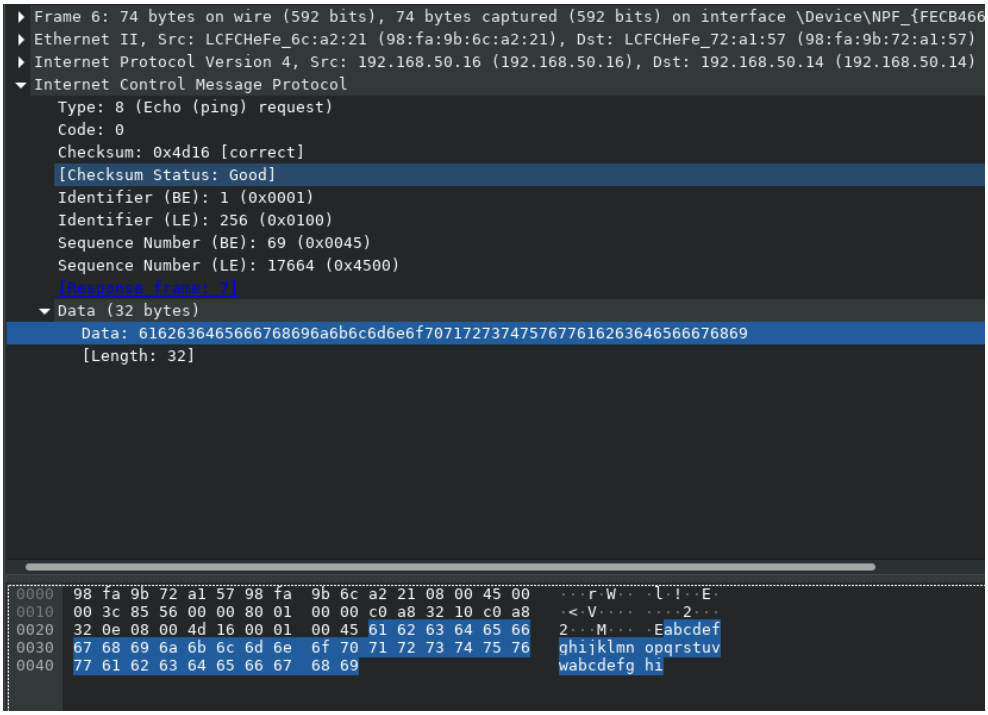


Abbildung 12: ICMP-Anfrage des ping-Befehls, Ausgabe von Wireshark mit geöffnetem ICMP-Paket

Auffällig ist hierbei zuerst, dass, im Gegensatz zum ARP-Beispiel, die ICMP-Daten in ein IP-Paket gepackt sind (Zeile 3 in Abb. 12). Das ist auch sinnvoll, da ICMP auf Ebene 3 agiert und eine konkrete Ziel-IP-Adresse be-

[1] wikipedia.org/Internet_Control_Message_Protocol

nötigt, während ARP auf Ebene 2 mithilfe von Broadcasts arbeitet und somit nicht in Berührung mit dem IP-Header kommt.

Das ICMP-Paket in diesem Beispiel hat ein 32 Byte langes Datenfeld (max.), welches mit 32 diversen ASCII-Zeichen gefüllt ist. Die Ausgabe der Eingabeaufforderung in Abb. 6 bestätigt dies. Die Gesamtgröße des Paketes ist 40 Byte (abzählbar aus Wireshark-Ausgabe), die Header-Größe ist daher die Differenz, also 8 Byte, was auch mit der ICMP-Definition übereinstimmt.

Folgend kommt die ICMP-Antwort, wie sie in Abb. 13 zu sehen ist, vom erwarteten Host mit der IP-Adresse `192.168.50.14` . Die Antwort enthält die gleichen Zeichen im ICMP-Datenfeld, nur das Typ-Feld wurde geändert von 8 (request) auf 0 (reply), was ebenfalls zu erwarten war.

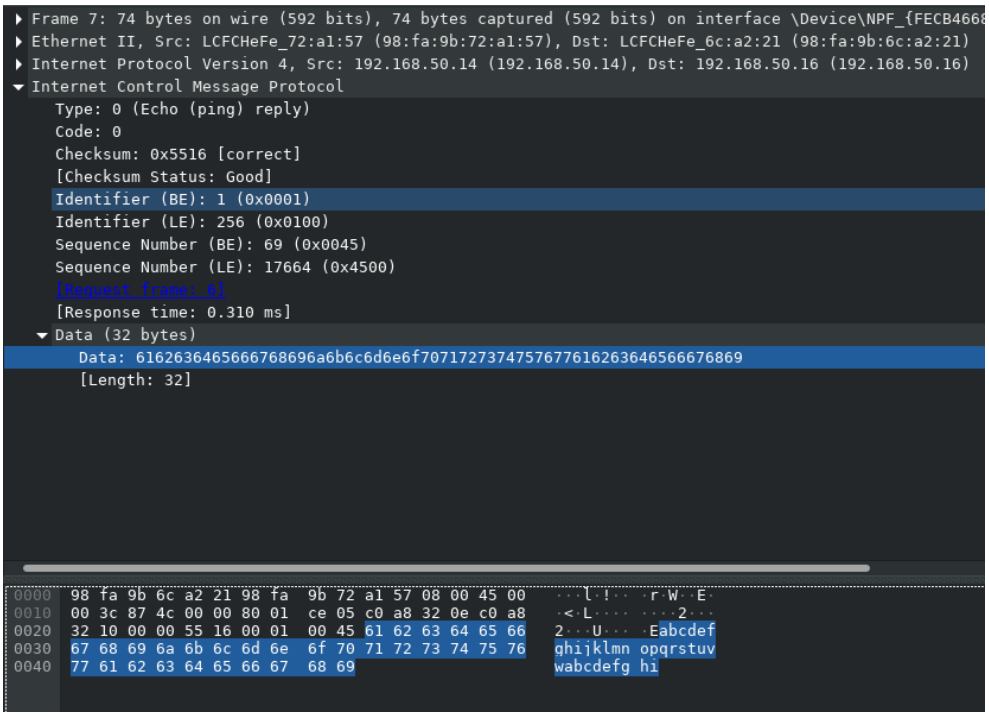


Abbildung 13: ICMP-Antwort auf die Anfrage durch den ping-Befehl

Der gesamte Ping-Prozess wird, wie bereits erwähnt, vier Mal wiederholt. Die dabei gesendeten bzw. empfangenen Pakete/Frames sind einschließlich des Datenfeldes äquivalent zu denen des ersten Pings, der hier behandelt wurde (nur Quell/Ziel-Adressen sind vertauscht).

Die Zeiten, die die CMD-Ausgabe des Befehls angibt, können ebenfalls in Wireshark bestätigt werden. In Abb. 6 wird eine Zeit $< 1\,\text{ms}$ berichtet. Aus der Zeitdifferenz in Wireshark ergeben sich $310\,\mu\text{s}$.

In Abb. 8 erkennt man 3 weitere ARP-Antworten vom Host `192.168.50.14`

, obwohl nur eine Anfrage gestellt wurde. Die Antworten sind der bereits empfangenen ersten Antwort identisch. Weshalb genau diese erscheinen, konnte nicht geklärt werden.

2.3.3 Flussdiagramm der Ping-Kommunikation

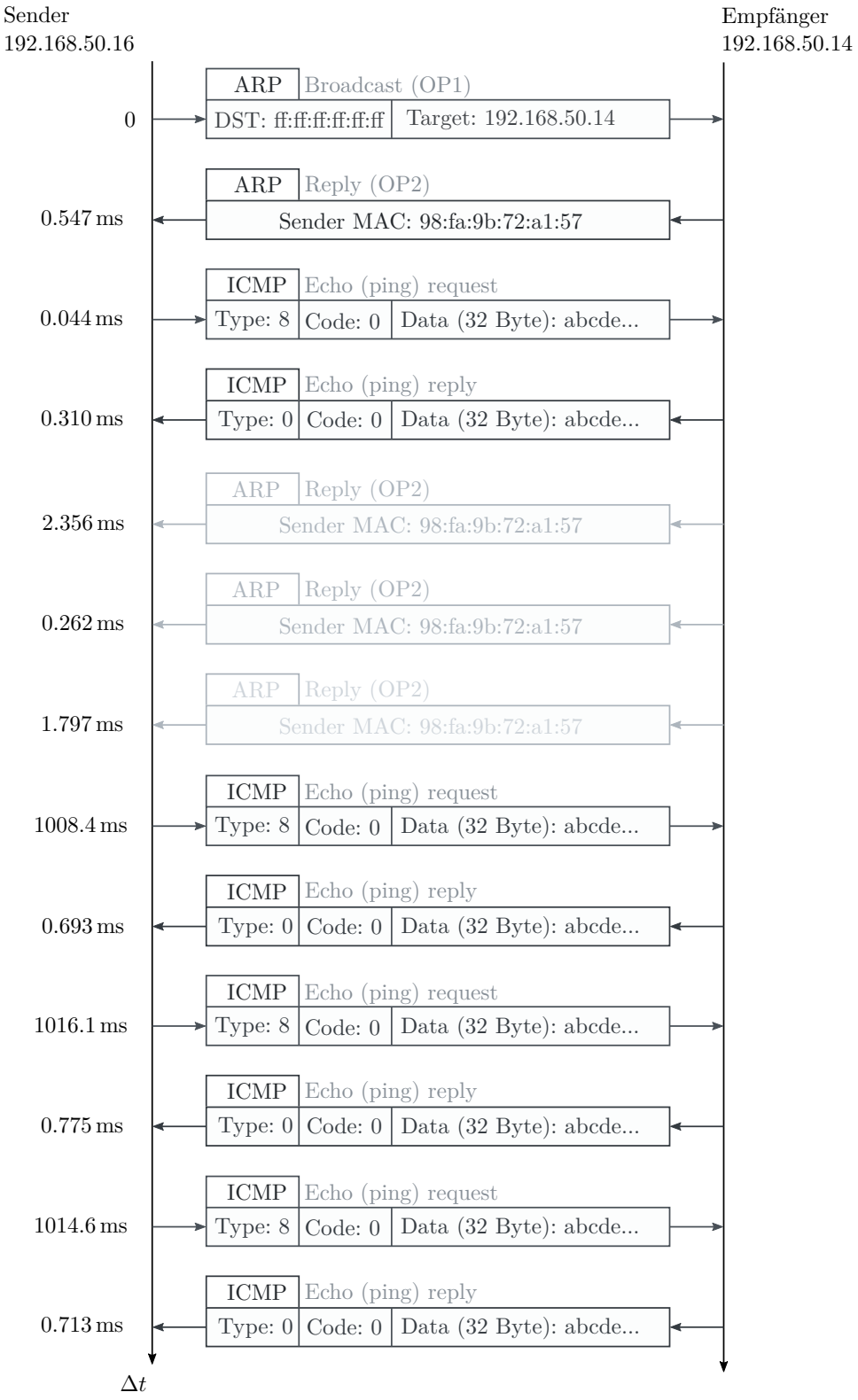


Abbildung 14: Flussdiagramm der Ping-Kommunikation

Das Flussdiagramm stellt den zeitlichen Ablauf der Kommunikation zwischen Sender (`192.168.50.16`) und Empfänger (`192.168.50.16`) dar. Für Sender und Empfänger sind jeweils die IP-Adressen als Endpunkt angegeben, allerdings sind diese nur für das Verständis, da nicht jedes Protokoll (wie z.B. ARP) diese zur Adressierung benötigt. Die Zeitangaben sind jeweils als sogenannte „Delta Zeit“, also der Zeitdifferenz zum vorherigen Ereignis dargestellt (Laufzeit, Wireshark → Einstellungen → Columns → delta Time).

Der Einfachheit halber wurden im Flussdiagramm nicht alle Felder der jeweiligen Pakete, sondern nur die für das Verständis der Kommunikation notwendigen dargestellt.

2.3.4 ping-Overhead

Für die Betrachtung des prozentualen Overheads wird das gesamte ICMP-Paket, so wie es der IP-Header umfasst, betrachtet. Wie vorher bereits bestimmt, beträgt die Größe dieses Paketes $32\,\mathrm{B} + 8\,\mathrm{B} = 40$ Byte. Der gesamte Frame, so wie er in das Netzwerk übertragen wird, beträgt laut Wireshark $74\,\mathrm{B}$ zu denen noch $4\,\mathrm{B}$ für das Ethernet-FCS Feld hinzukommen (siehe 2.3.2/ARP). Somit sind es $74 + 4 - 32 = 46$ Byte an Overhead. Der Prozentuale Overhead o des ping-Befehls (genauer: des Standard-ping-Befehls unter Windows) ist demnach

$$o = \frac{46\,\mathrm{B}}{78\,\mathrm{B}} \cdot 100\,\% \approx 59\,\%\,.$$

2.4 Analyse der Internetsitzungen

Im ersten Schritt wurden ARP- und DNS-Cache geleert, um entsprechende Anfragen in Wireshark sehen zu können.

Es wurden dann folgende Web-Seiten im *Mozilla Firefox* Browser nacheinander aufgerufen:

acad-30.na.hs-wismar.de
sowie
hermes.fiw.hs-wismar.de/comlab

Währenddessen wurde mithilfe von Wireshark die Netzwerkkommunikation mitgeschnitten. Abb. 15 zeigt einen Ausschnitt aus der Tracedatei, für die Übersicht wurde der Filter `tcp || http || dns` angewandt.

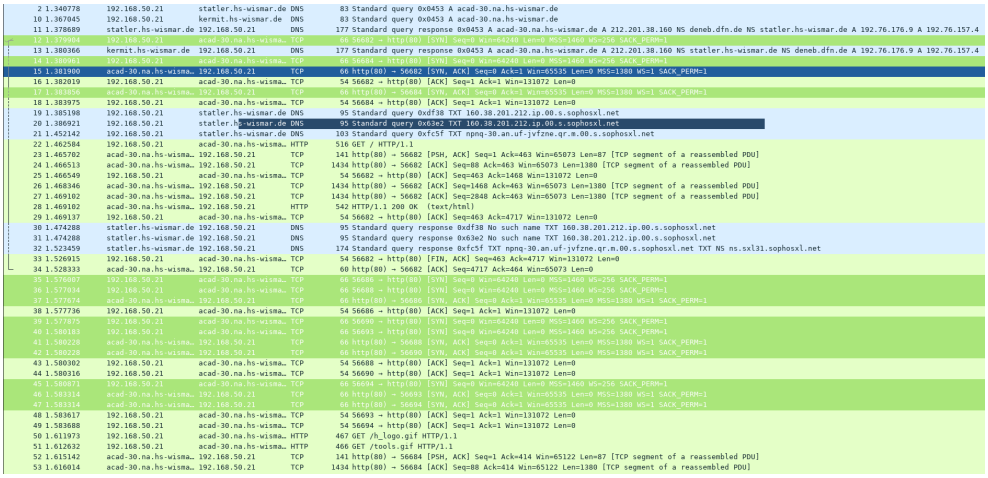


Abbildung 15: Screenshot des Beginns des Wireshark-Mitschnittes, acad-30.na.hs-wismar.de (gefiltert)

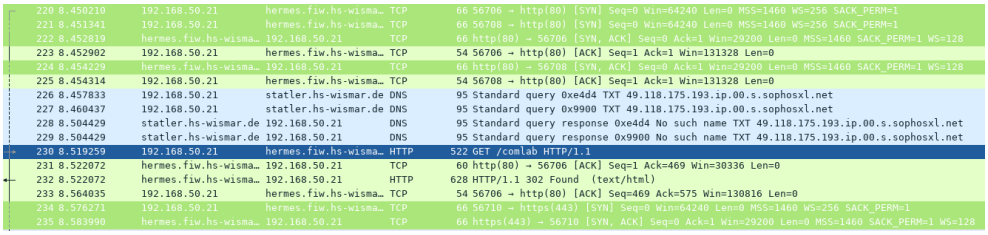


Abbildung 16: Screenshot des Beginns der Kommunikation mit hermes.fiw.hs-wismar.de (gefiltert)

2.4.1 DNS-Anfragen

Das DNS-Protokoll (Domain Name Service) ist ein Protokoll auf Anwendungsebene, das dafür zuständig ist, den jeweiligen Doimainnamen (z.B. acad-30.na.hs-wismar.de) in dessen IP-Adresse umzuwandeln. Ist diese Zuweisung nicht bereits im Zwischenspeicher vorhanden, muss der Client eine DNS-Anfrage an einen DNS-Server stellen. Dies passiert auch am Beginn der Kommunikatio, da der DNS-Cache im Vornherein gelöscht wurde (Abb. 15).

Der Host `192.168.50.21`, auf dem der Name der Website in die Browserleiste eingegeben wurde, stellt eine Anfrage an den Server *statler.hs-wismar.de* (`192.76.157.4`). Dieser befindet sich nicht im gleichen Subnetz, weshalb die Kommunikation über den Standardgateway (`192.168.50.1`) laufen muss, was auch durch das überprüfen der Ziel-MAC-Adresse im Ethernetframe zu erkennen ist.

Es sind keine weiteren ARP-Anfragen von der sendenden Schnittstelle notwendig, da die Hardwareadresse des Standardgateways anscheinend bekannt ist und weitere Hardwareadressauflösungen von den folgenden Netzwerkknoten gehandhabt werden.

Die DNS-Anfrage läuft über Ebene 4 mit dem verbindungslosen UDP-Protokoll zu Port 53 und enthält u.a. den ASCII-codierten Namen der Website, sowie bestimmte Flags, wie z.B. das Opcode-Flag 0 zur Signalisierung einer Anfrage (Abb. 17). Da es verbindungslos ist, sieht man beispielsweise auch keine Bestätigungsantworten (ACK), wie man es bei TCP erwarten würde.

Als Separator zwischen den URN-Bestandteilen ist in den eigentlichen Daten nicht der Punkt (wie man es im Browser eingibt), sondern der Hex-Wert *02*, welcher im ASCII-Code für STX (Start of Text) steht (Abb. 17 unten).

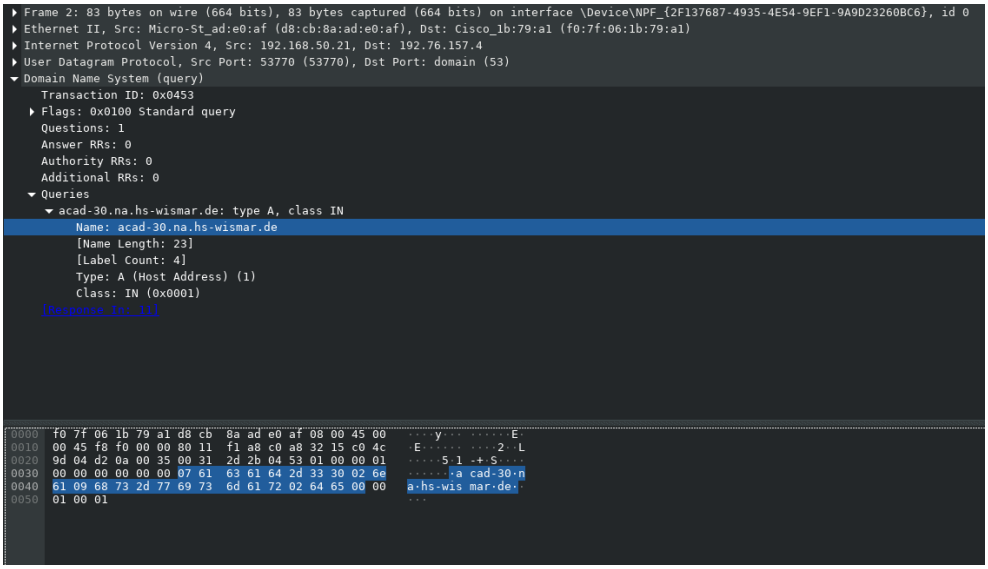


Abbildung 17: Erste DNS-Anfrage über acad-30.na.hs-wismar.de

Es folgt in der nächsten Zeile eine weitere, parallele DNS-Anfrage an einen anderen Server (kermit.hs-wismar.de / `192.76.157.9`) zur Auflösung desselben Namens (acad-30.na.hs-wismar.de). Das geschieht möglicherweise, um im Fehlerfall des „statler“ Servers trotzdem eine DNS-Auflösung zu erhalten.

Die Antwort auf die erste Anfrage erscheint nach etwa 38 ms. Sie enthält, wie erwartet, die zum Namen gehörende IP-Adresse `212.201.38.160`. Die Flags geben hier an, dass es sich um eine Standard-Antwort handelt und dass es keine Fehler gab (Abb. 18).

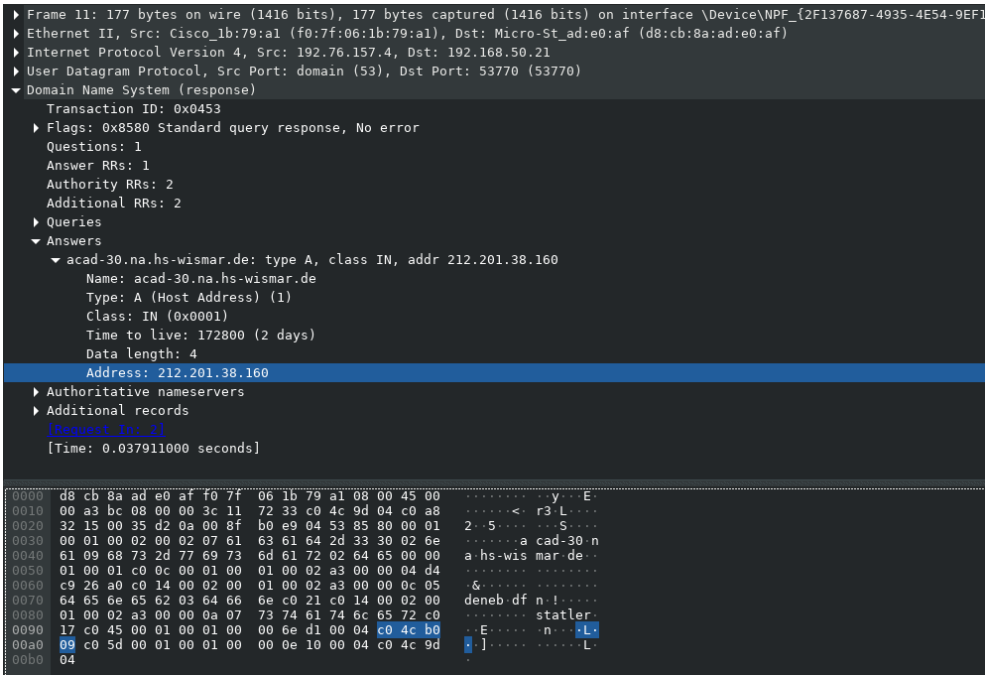


Abbildung 18: Erste DNS-Antwort über acad-30.na.hs-wismar.de

2.4.2 HTTP- und TCP-Verkehr

Nach der Antwort wird die TCP-Kommunikation mit dem nun bekannten acad-30.hs-wismar.de auf Port 80 (HTTP) initiiert. Dies geschieht über den 3-Wege-Handshake (siehe 1.5.1). Der Sender legt einen zufälligen Port zur Identifizierung der Verbindung fest, welcher über dem sogenannten „well known ports“-Bereich (1024) liegt und kleiner als 65535 ist, in diesem Fall 56682.

Dann wird ein TCP-Paket mit gesetzem SYN-Bit (Synchronisation) an den Server geschickt, um diesem die eigene Sequenznummer (3291467299) für die Paketnummerierung, die Fenstergröße (64240) zur Flusskontrolle, sowie die maximale Segmentgröße (Datengröße) mitzuteilen. In Wireshark kann man für TCP-Streams relative Sequenznummern einstellen, wodurch die Nummerierung automatisch bei 0 beginnt (so auch in Abb. 19). Außerdem wird während des Handshakes der Skalierungsfaktor für die Fenstergröße festgelegt. Dessen TCP-Feld ist 8, was einen Faktor von $2^8 = 256 bedeutet$.

Kurz danach öffnet der Client einen weiteren Port (56684) für die TCP-Kommunikation mit dem gleichen Server auf Port 80. Er fährt dabei genauso vor wie bei der vorherigen Initiierung. Die parallelen Verbindungen werden dann genutzt, um jeweils verschiedene Ressourcen vom Server „gleichzeitig“ anzufragen (z.B. HTML, Bilder etc.).

Auf die erste SYN-Anfrage antwortet der Server in gleicher Weise mit einem TCP-Paket, in welchem das SYN-Bit sowie nun auch das ACK-Bit zur Bestätigung gesetzt sind. Der Server teilt dem Client ebenfalls seine Window-

größe (65535 · 1), Sequenznummer (167188764) und maximale Segmentgröße (1380) mit und richtet sie an Port 56682.

Dazwischen erscheint noch die Antwort auf die zweite DNS-Anfrage von kermit.hs-wismar.de, welche wahrscheinlich ignoriert wird, da die Auflösung bereits bekannt ist.

Im dritten Schritt des three-way-handshakes erhält der Server vom Client ein ACK, um den Empfang des vorherigen TCP-Paketes zu bestätigen (Abb. 20). Hierin ist die Sequenznummer auf 1 gesetzt, um zu kennzeichnen, dass das nächste Paket, das empfangen werden kann die (relative) Sequenznummer 1 besitzt. Man erkennt außerdem, dass die Fenstergröße nur 512 Byte beträgt. Da im Handshake allerdings der Fensterskalierungsfaktor 256 festgelegt wurde, ist die tatsächliche Fenstergröße 131072, was auch im durch Wireshark hinzugefügten Feld [Calculated window size] zu sehen ist.

Der andere TCP-Handshake auf Client-Port 56684 verläuft analog.

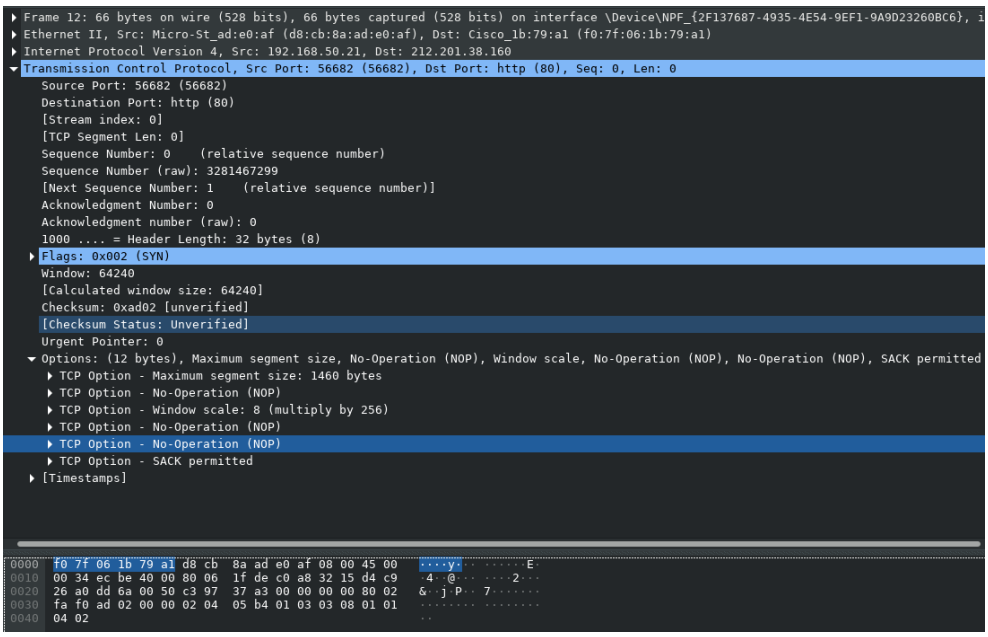


Abbildung 19: Erstes TCP-Paket für Three-Way-Handshake mit SYN-Bit

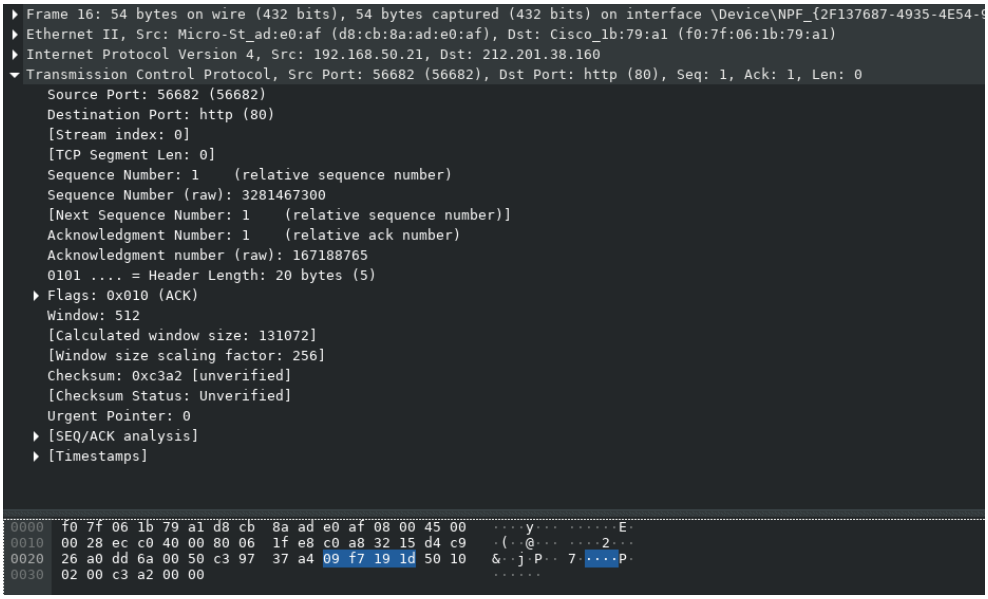


Abbildung 20: Beenden des Three-Way-Handshakes mit ACK

Vor der ersten HTTP-Anfrage erscheinen noch drei DNS-Anfragen zur Auflösung der Namen „160.38.201.212.ip.00.s.sophosxl.net“ sowie „npnq-30.an.uf-jvfzne.qr.m.00.s.sophosxl.net“ an den Server `192.76.157.4`, welche jedoch, wie in der DNS-Antwort bei Nummern 30-32 zu sehen, von diesem nicht aufgelöst werden können.

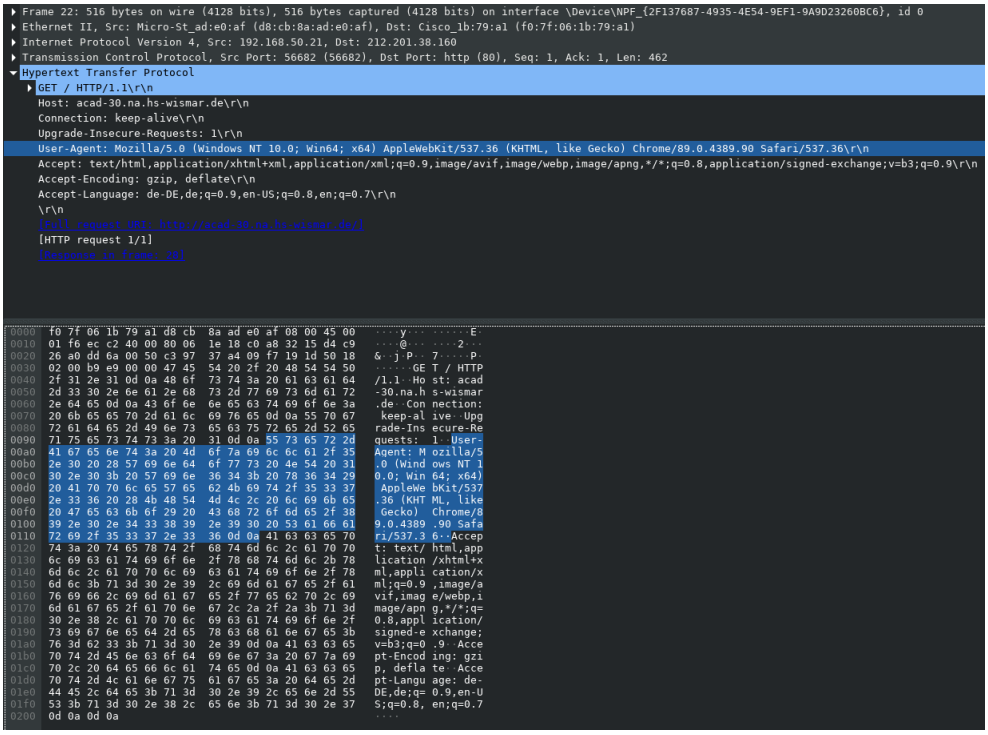


Abbildung 21: HTML-Anfrage

Es folgt die HTTP-Anfrage zum Grundverzeichnis der Website „/“ . Diese enthält u.a. Informationen über Sprache, Kodierung, Browser und Plattform. Sie wird vom Server sofort zusammen mit den ersten Daten bestätigt. Da die maximale Segmentgröße (MSS) beim Handshake zu 1380 Byte festgelegt wurde, können die Daten nicht als ein einziges Segment verschickt werden, sondern müssen in maximal dieser Größe segmentiert sein. In Abb. 15 ab Paket 23 erkennt man dieses Verhalten. Zuerst werden Daten der Länge 84, dann Daten der Länge 1380 übertragen, also insgesamt 1467 Byte. Dies überschreitet zwar noch nicht die festgelegte Fenstergröße, allerdings ist das nur die maximale Grenze und nach Konvention wird jedes zweite (oder spätere) Paket bereits mit einem ACK bestätigt, wie bei Paket 25 zu sehen. Wireshark kennzeichnet TCP Segmente, die zu größeren, zusammenhängenden Daten gehören außerdem als [TCP segment of a reassembled PDU].

Der HTML-Status „200 OK“ wird dann gesendet, um eine erfolgreiche Anfrage zu signalisieren. Der Client bestätigt folglich noch das Empfangen der Daten.

Auf diese Weise wurde eine 4715 Byte große HTML-Datei übertragen, welche der Browser dann darstellen kann. Diese kann man auch erkennen, wenn man sich die (zusammengesetzten) Daten der Pakete anschaut.

Von der Server-Seite wird mit dem letzen HTTP-Paket mittels [FIN, ACK] das Kommunikationsende auf dem Port 80/56682 angefragt, was dann vom Client mit einem [FIN, ACK] bestätigt wird, worauf der Server erneut mit einem ACK antwortet.

Später wird noch eine Datei *h_logo.gif*, also eine Bilddatei im GIF-Format, angefragt und auch übertragen (Abb. 22). Man kann die gesendeten Daten mit Wireshark extrahieren und wieder zusammensetzen. Dazu geht man auf File → Export Objects → HTTP und wählt die entsprechende Datei aus. Das Ergebnis ist in Abb. 23 zu sehen.

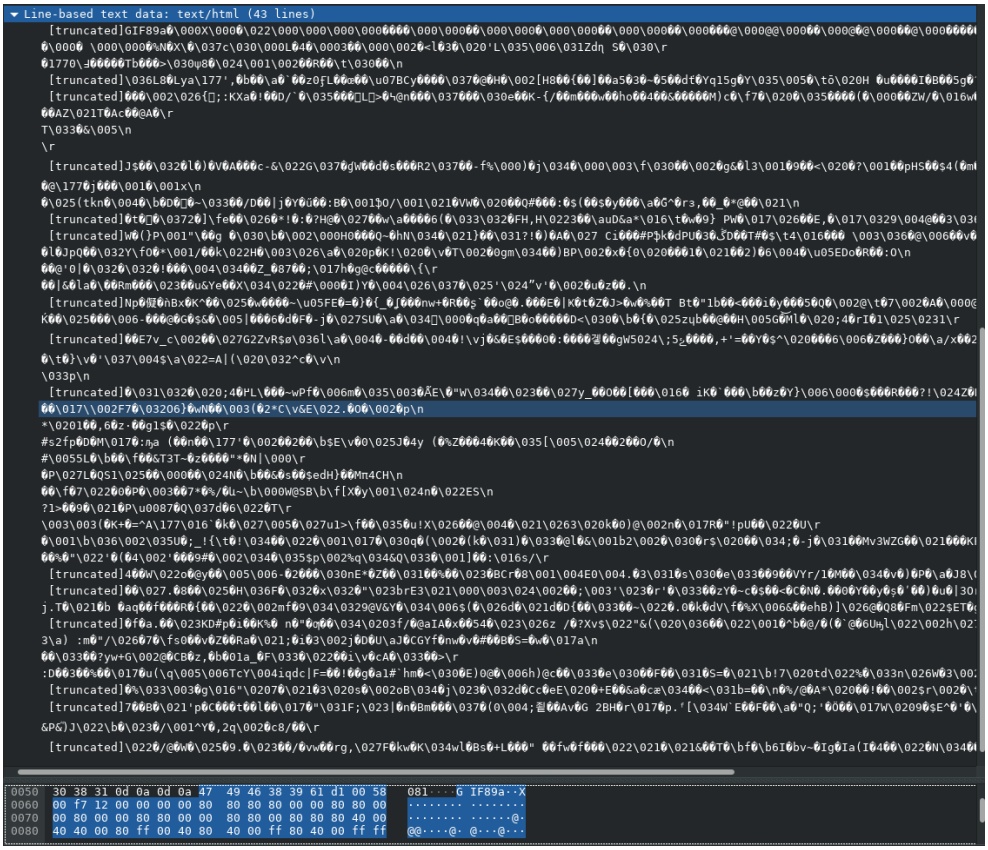


Abbildung 22: Ausschnitt aus dem HTTP-Paket für die Datei h_logo.gif

Abbildung 23: Entnommene GIF-Datei

2.4.3 Flussdiagramm der Sitzungen

Da die Kommunikation aus vielen HTTP-Anfragen besteht, wurden in den Flussdiagrammen jeweils nur die DNS-Anfragen und die ersten TCP-Verbindungen bis zu deren Ende (FIN) dargestellt. Außerdem wurden nur die „höchsten“ Protokolle in den jeweiligen Paketen gekennzeichnet (z.B. HTTP, obwohl es natürlich in einem TCP Paket eingebettet ist).

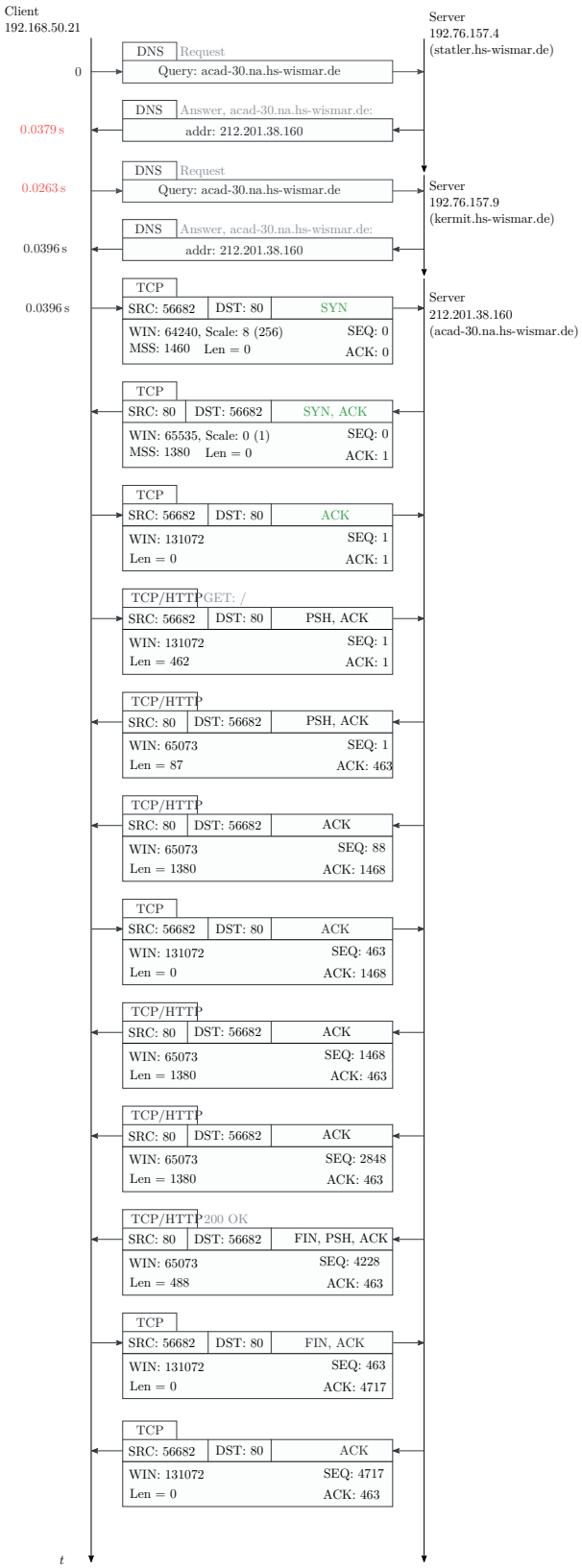


Abbildung 24: Flussdiagramm der Verbindung mit acad-30.na.hs-wismar.de

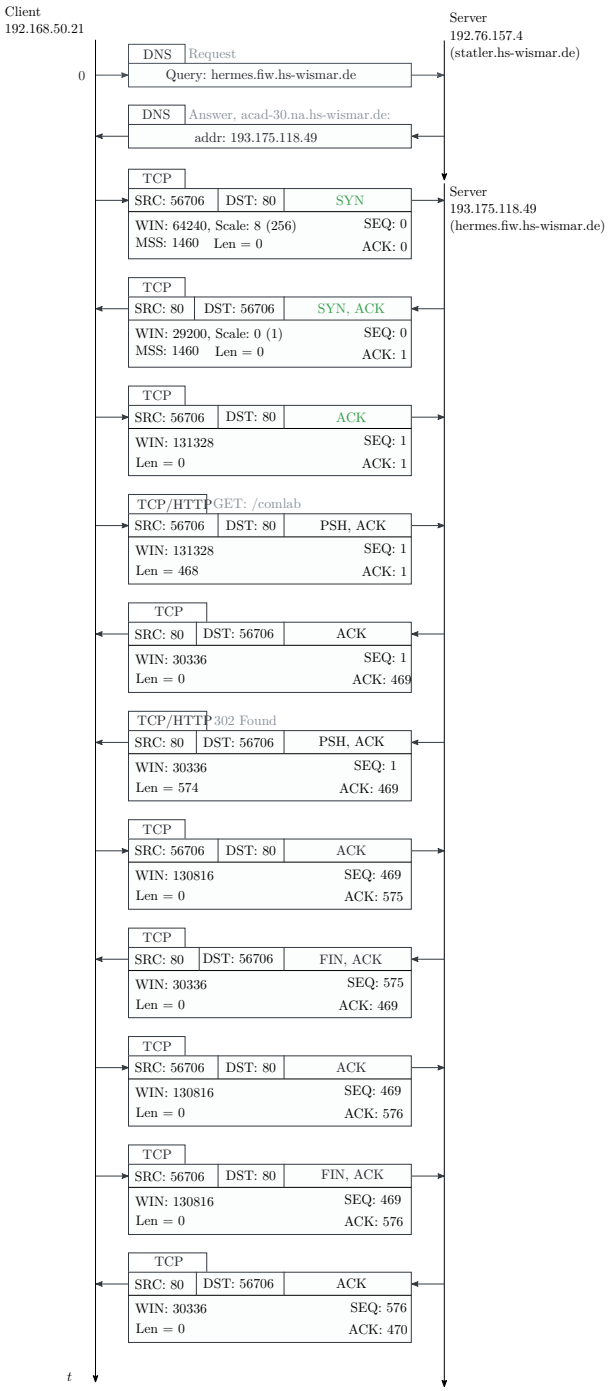


Abbildung 25: Flussdiagramm der Verbindung mit hermes.fiw.hs-wismar.de