Technische Universität München
Lehrstuhl Informatik VIII
Prof. Dr.-Ing. Georg Carle
Dipl.-Ing. Stephan Günther, M.Sc.
Johannes Naab, M.Sc.

Tutorübung zur Vorlesung Grundlagen Rechnernetze und Verteilte Systeme

Übungsblatt 5 (18. Mai – 22. Mai 2015)

Hinweis: Die mit * gekennzeichneten Teilaufgaben sind ohne Kenntnis der Ergebnisse vorhergehender Teilaufgaben lösbar.

Aufgabe 1 Subnetting

Der Moepinet AG werden die Adressbereiche 131.159.32.0/22 und 131.159.36.0/24 zugewiesen. Für die Aufteilung dieses Adressbereichs ist die Moepinet AG selbst verantwortlich. Nach einer sorgfältigen Bedarfsanalyse ergeben sich die folgenden Anforderungen an die Subnetze und die Mindestanzahl **nutzbarer** IP-Adressen:

Subnetz	NET 1	NET 2	NET 3	NET 4	NET 5
IPs	300	300	15	40	4

Bei der Erhebung dieser Zahlen wurde die an das jeweilige Router-Interface zu vergebende IP-Adresse bereits berücksichtigt.

a)* Geben Sie jeweils die erste und letzte IP-Adresse der beiden vergebenen Adressbereiche an.

- 131.159.32.0/22:
  Erste IP: 131.159.32.0 (Netzadresse)
  Letzte IP: 131.159.35.255 (Broadcast-Adresse)
- 131.159.36.0/24:
  Erste IP: 131.159.36.0 (Netzadresse)
  Letzte IP. 131.159.36.255 (Broadcast-Adresse)

b) Wie viele IP-Adressen stehen der Moepinet AG insgesamt zur Verfügung? Können alle davon zur Adressierung von Hosts verwendet werden?

- 131.159.32.0/22: $2^{32-22} = 2^{10} = 1024$ Adressen
- 131.159.36.0/24: $2^{32-24} = 2^{8} = 256$ Adressen

Insgesamt stehen also 1024 + 256 = 1280 Adressen zur Verfügung. Allerdings sind die erste (Netzadresse) und letzte Adresse (Broadcast-Adresse) eines jeden Netzes nicht zur Adressierung von Hosts nutzbar. Es stehen also zunächst maximal 1022 + 254 = 1276 Adressen zur Hostadressierung zur Verfügung.

c)* Ist es möglich, den von den beiden Adressblöcken gebildeten Adressbereich in einem einzigen Subnetz zusammenzufassen?

Nein. Es gibt keine passende Subnetzmaske, welche den insgesamt 1280 Adressen großen Block definieren könnte. Die Größe eines Netzes kann stets nur eine Zweierpotenz sein.

d)* Teilen Sie nun die beiden Adressbereiche gemäß der Bedarfsanalyse auf, so dass Subnetze der passenden Größe entstehen. Gehen Sie mit den Adressen so sparsam wie möglich um. Es soll am Ende ein möglichst großer zusammenhängender Adressbereich für zukünftige Nutzung frei bleiben. Für jedes Subnetz ist anzugeben:

- die Größe des Subnetzes
- die Anzahl nutzbarer Adressen
- das Subnetz in Präfixschreibweise
- die Subnetzmaske in Dotted-Decimal-Notation
- die Netz- und Broadcastadresse

Um die Vorgaben zu erfüllen, müssen wir die Subnetze gemäß ihrer Größe in absteigender Reihenfolge bearbeiten. Andernfalls könnten wir die folgende Situation erhalten:

- An Netz 3 wird der Adressbereich 131.159.36.0/27 vergeben.
- Vergibt man nun aber an Netz 4 den Bereich 131.159.36.32/26, macht man einen Fehler. Um dies zu verstehen, muss man sich die Binärschreibweise der Netzadresse und Subnetz-Maske ansehen:
  `131.159. 36.0010 0000` (IP)
  `255.255.255.1100 0000` (Subnetz-Maske)
  Eine UND-Verknüpfung beider Zeilen ergibt, dass die IP-Adresse 131.159.36.32 in das Subnetz 131.159.36.0/26 fällt!
- Wir müssten also den Bereich 131.159.36.64/26 an Netz 4 vergeben. Dann allerdings entstünde eine Lücke zwischen Netz 3 und Netz 4.
- Vergibt man die Adressen gemäß der Größe der Subnetze in absteigender Reihenfolge, umgeht man das Problem. Dieses Vorgehen könnte natürlich wieder anderen Kriterien widersprechen – beispielsweise der Vergabe zusammenhängender Adressblöcke an einzelne Niederlassungen.

Subnetz	NET 1	NET 2	NET 3	NET 4	NET 5
Bedarf	300	300	15	40	4
Größe	512	512	32	64	8
Nutzbar	510	510	30	62	6
Präfixnotation	131.159.32.0/23	131.159.34.0/23	131.159.36.64/27	131.159.36.0/26	131.159.36.96/29
Subnetzmaske	255.255.254.0	255.255.254.0	255.255.255.224	255.255.255.192	255.255.255.248
Netzadresse	131.159.32.0	131.159.34.0	131.159.36.64	131.159.36.0	131.159.36.96
Broadcast	131.159.33.255	131.159.35.255	131.159.36.95	131.159.36.63	131.159.36.103

Aufgabe 2 Shortest Path Trees (Hausaufgabe)

In dieser Aufgabe betrachten wir die aus der Vorlesung bekannten Algorithmen zur Konstruktion von SPTs.

a)* Wie lautet die Metrik, die bei der Konstruktion eines SPTs minimiert wird?

Es werden die Kosten jedes einzelnen Pfads von der Wurzel s zu allen anderen Knoten $k \in \mathcal{N} \setminus \{s\}$ minimiert – also ein jeweils kürzester Weg von s zu allen anderen Knoten gesucht. Es handelt sich also um eine vom jeweiligen Zielknoten k abhängige Metrik. Außerdem ist der resultierende SPT natürlich auch von der Wahl des Wurzelknotens abhängig.

b)* Wie lautet hingegen die Metrik, die bei der Konstruktion eines Minimum Spanning Tress (MSTs) minimiert wird?

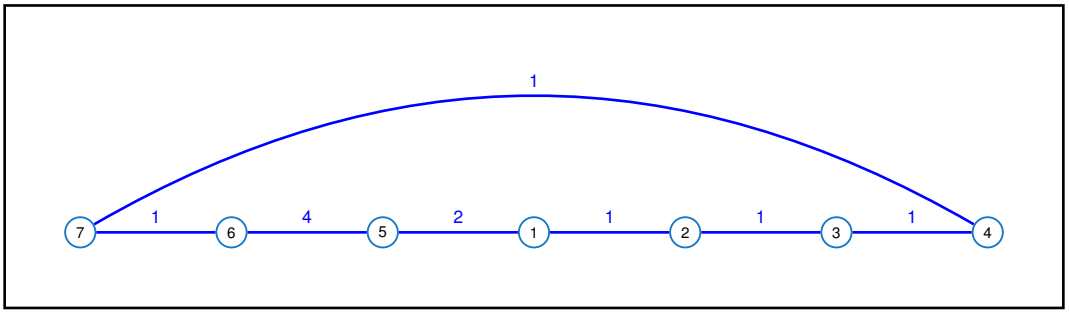
Es werden die Gesamtkosten aller Kanten minimiert, die notwendig sind, um einen Spannbaum zu konstruieren.

Im Folgenden wollen wir die Unterschiede in der Funktionsweise der Algorithmen von Dijkstra und Bellman-Ford genauer beleuchten. Dazu sei ein Netzwerk in Form der Distanzmatrix

$$\boldsymbol{D} = \begin{bmatrix} 0 & 1 & \infty & \infty & 2 & \infty & \infty \\ 1 & 0 & 1 & \infty & \infty & \infty & \infty \\ \infty & 1 & 0 & 1 & \infty & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty & \infty & 1 \\ 2 & \infty & \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 & 1 \\ \infty & \infty & \infty & 1 & \infty & 1 & 0 \end{bmatrix}$$

gegeben.

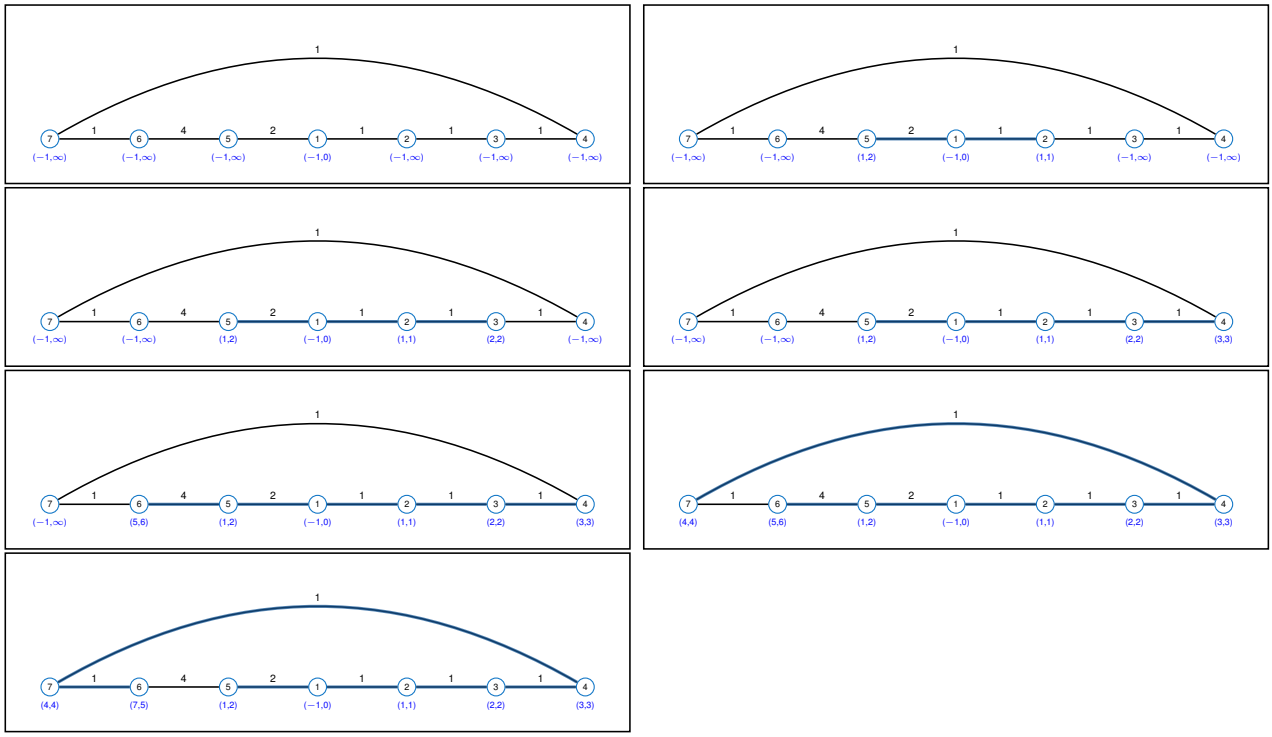
c)* Zeichnen Sie den Graphen, indem Sie die entsprechenden Kanten und Gewichte in untenstehende Abbildung eintragen.



d)* Die Anordnung der Knoten im Vordruck von Teilaufgabe c) lässt bereits vermuten, dass die Knoten weitgehend in einer Kette verschaltet sind. Hätte man das auch direkt aus der Distanzmatrix $\boldsymbol{D}$ erahnen können? Begründen Sie Ihre Antwort.

Ja. Die Matrix ***D*** weißt eine auffällige Diagonalstruktur auf. Der Umstand, dass fast ausschließlich die beiden Nebendiagonalen endliche Einträge besitzen, zeigt bereits, dass die meisten Knoten nur zwei Nachbarn haben – in diesem Fall trifft das sogar auf alle Knoten zu, da sie in einem Ring verschaltet sind. Das allerdings geht zumindest auf den ersten Blick aber nicht aus ***D*** hervor. **Für die folgenden Teilaufgaben sei 1 der Wurzelknoten.** Falls es in einem Schritt mehr als einen Knoten gibt, mit dem Sie fortfahren können, so wählen Sie stets den Knoten mit der niedrigeren Nummer.

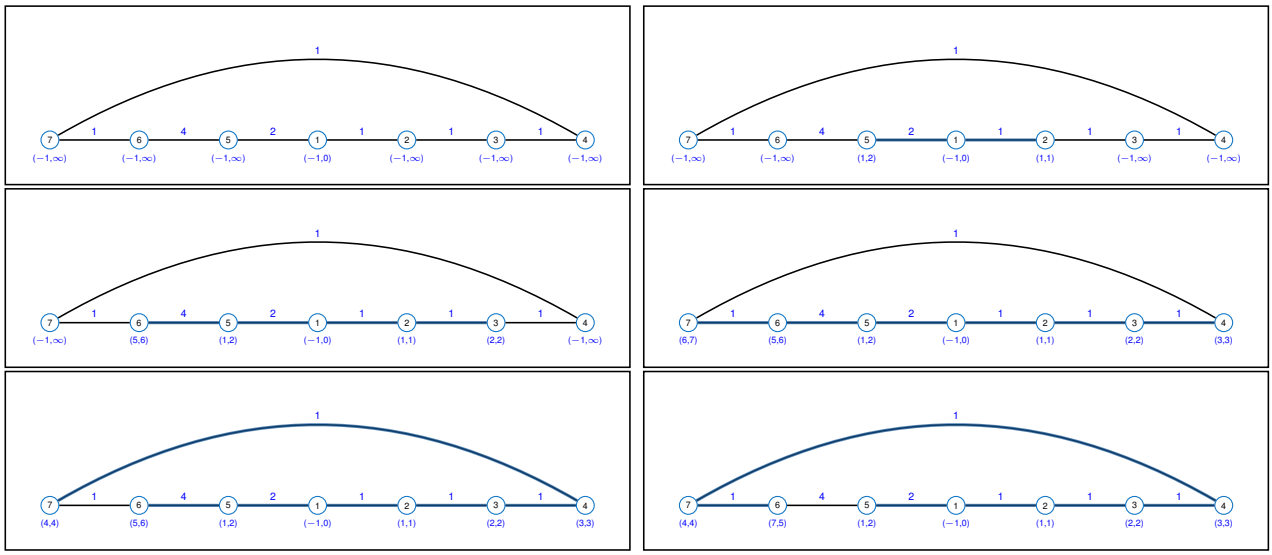
e) Führen Sie auf dem Graphen aus Teilaufgabe c) den Algorithmus von Dijkstra durch. Zeichnen Sie immer den aktuellen Zustand des Netzwerks, wenn ein neuer Knoten in die Menge $\mathcal{S}$ der bearbeiteten Knoten aufgenommen wird. Notieren Sie außerdem zu jedem Knoten i seinen Vorgänger und die derzeitigen Kosten als Tupel ($p[i]$,$d[i]$).



f)* Wieviele Iterationen wird der Algorithmus von Bellman-Ford höchstens benötigen?

Der längste einfache Pfad im Netzwerk beträgt sieben Hops. Der Algorithmus wird also spätestens nach sieben Iterationen terminieren, da dann alle möglichen Pfade berücksichtigt worden sind.

g) Führen Sie auf dem Graphen aus Teilaufgabe c) den Algorithmus von Bellman-Ford durch. Zeichnen Sie immer dann den aktuellen Zustand des Netzwerks, nachdem alle Pfade der Länge N berücksichtigt worden sind. Notieren Sie außerdem zu jedem Knoten i seinen Vorgänger und die derzeitigen Kosten als Tupel ($p[i]$,$d[i]$).



Aufgabe 3 ARP und IP-Fragmentierung

In Abbildung 1 ist eine Anordnung von Netzkomponenten mit ihren IP- und MAC-Adressen dargestellt. Die beiden Computer PC1 und PC2 verwenden den jeweils lokalen Router als Default-Gatway. PC1 sendet ein IP-Paket der Größe l=1000 B (Nutzdaten) an PC2. Die MTU auf dem WAN-Link zwischen R1 und R2 betrage 580 B. Innerhalb der lokalen Netzwerke gelte die für Ethernet übliche MTU von 1500 B.

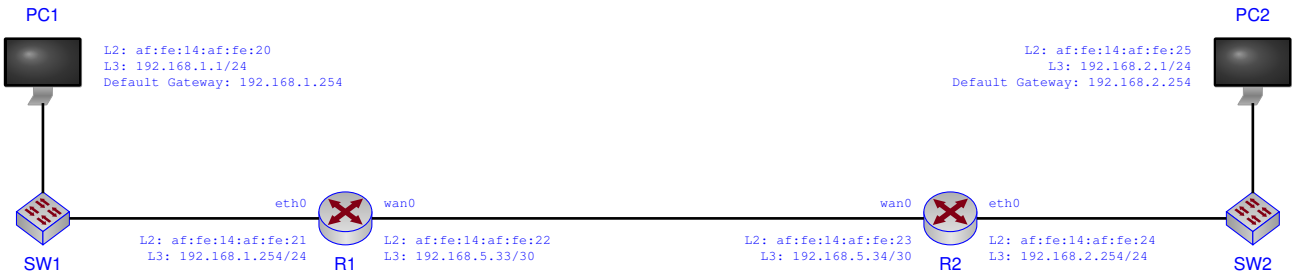


Abbildung 1: Netztopologie

Im Folgenden soll die Übertragung des Pakets mit allen notwendigen Zwischenschritten nachvollzogen werden. Gehen Sie zunächst davon aus, dass die ARP-Caches aller beteiligten Netzwerkkomponenten geleert sind.

a)* Inwiefern wirken sich die beiden Switches SW1 und SW2 in diesem Beispiel aus?

Die Switches haben keinerlei Einfluss auf die ausgetauschten Nachrichten. Switches sind i. A. transparent für die angeschlossenen Hosts. Insbesondere verändern Switches weder Absender noch Empfänger Adresse.

b)* In wie viele Fragmente muss R1 das Paket von PC1 aufteilen?

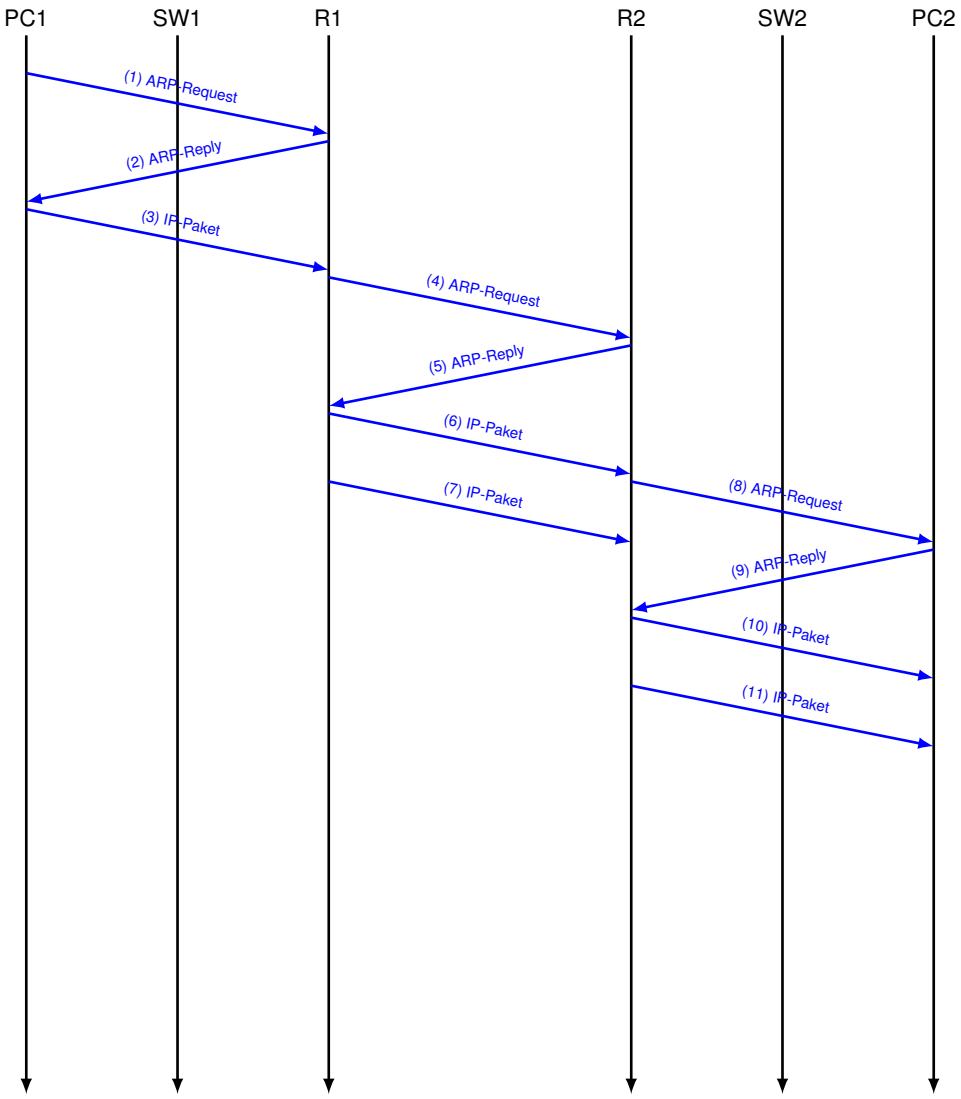
Die MTU (Maximum Transmission Unit) ist die maximale Größe eines Pakets auf Schicht 3 inkl. Header. Sie entspricht also genau der maximalen Größe der Payload auf Schicht 2. Mit dem Wissen, dass ein IP-Header 20 B lang ist (Ausnahme bei Verwendung von Optionen), erhalten wir:

$$N = \left\lceil \frac{1000\,\mathrm{B}}{580\,\mathrm{B} - 20\,\mathrm{byte}} \right\rceil = 2$$

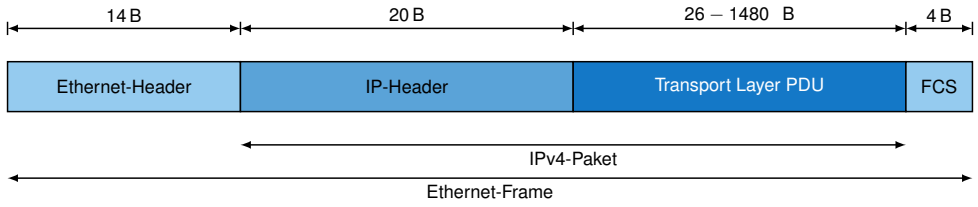
c)* An welcher Stelle im Netzwerk werden die Fragmente reassembliert?

Erst der Empfänger, hier also PC2, reassembliert die Fragmente wieder. Tatsächlich kann i. A. kein anderer Knoten die Reassemblierung durchführen, da die Fragmente jeweils einzelne und voneinander unabhängige Pakete darstellen. Dies bedeutet insbesondere, dass sie unabhängig voneinander geroutet werden und daher u. U. verschiedene Wege zum Ziel nehmen können – das sieht man aus dem einfachen Beispiel in Abbildung 1 natürlich nicht, da es hier nur einen Pfad zwischen PC1 und PC2 gibt.

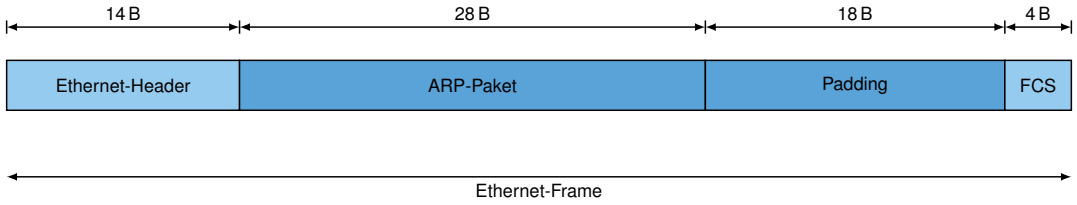
d) Skizzieren Sie ein einfaches Nachrichtenflussdiagramm, welches **alle Rahmen** berücksichtigt, die auf den jeweiligen Verbindungen übertragen werden müssen. **Nennen Sie die Art der ausgetauschten Rahmen und geben Sie den Rahmen Nummern (1,2,3,...).** (Das Diagramm muss nicht Maßstabsgetreu sein, Serialisierungszeiten können vernachlässigt werden.)


PC1
SW1
R1
R2
SW2
PC2
(1) ARP-Request
(2) ARP-Reply
(3) IP-Paket
(4) ARP-Request
(5) ARP-Reply
(6) IP-Paket
(7) IP-Paket
(8) ARP-Request
(9) ARP-Reply
(10) IP-Paket
(11) IP-Paket

Vergegenwärtigen Sie sich den Zusammenhang, dass einem IP-Paket bei seinem Versand ein Ethernet-Header vorangestellt wird. Dieser wird bei jedem Zwischenempfänger (Hop) auf dem Weg zum Zielsystem modifiziert beziehungsweise ersetzt. Der *Payload* eines Ethernet-Frames beinhaltet also insbesondere auch den IP-Header.

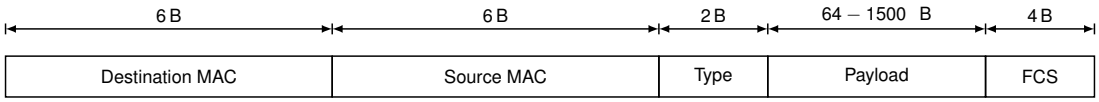


Genauso werden ARP-Requests und -Replies gekapselt.



Um Ihnen das Suchen der entsprechenden Vorlesungsfolien zu ersparen, sind im Folgenden der Ethernet- und IP-Header sowie ARP-Pakete nochmals schematisch dargestellt und kurz erklärt:

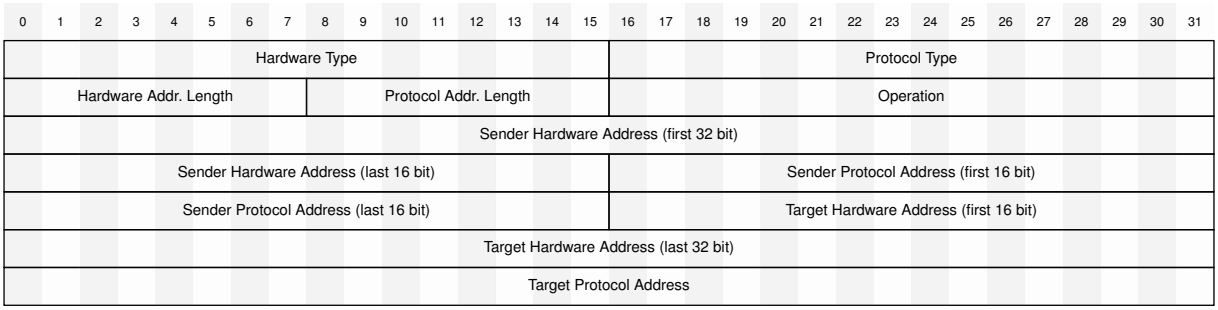
Ethernet-Header



Das Type-Feld des Ethernet-Headers gibt die Art der Payload an. Die für uns relevanten Werte sind:

- IPv4 $\triangleq$ `0x0800`
- ARP $\triangleq$ `0x0806`

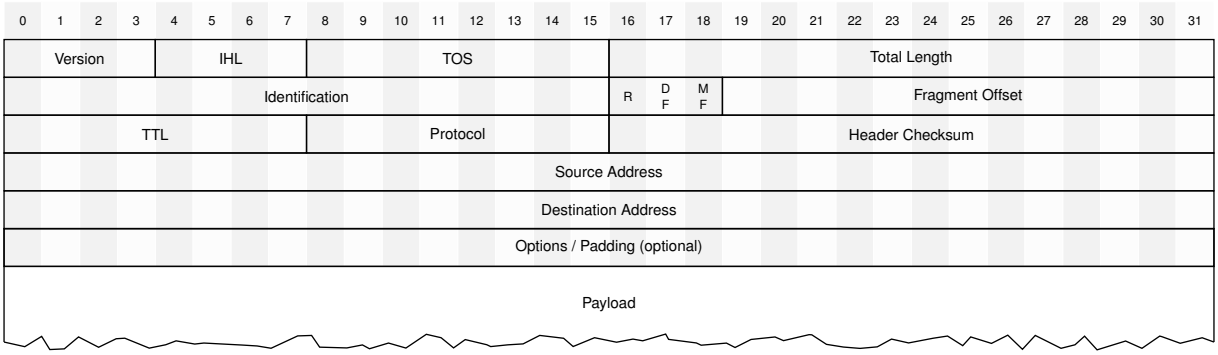
ARP-Pakete



Die Felder nehmen dabei folgende Werte an:

- Hardware Address Type für Ethernet-Adressen ist `0x0001`
- Protocol Address Type für IPv4 ist `0x0800`
- Hardware bzw. Protocol Address Length geben die Länge der jeweiligen Adresse in Byte an
- Opcode ist `0x0001` (ARP-Request) oder `0x0002` (ARP-Reply)

IP-Header



Auch hier wieder einige Hinweise zu den Feldern:

- „Version“ ist für IPv4 stets `0x4`
- „IHL“ (IP Header Length) gibt die Länge des IP-Headers in ganzzahligen Vielfachen von 4 B an
- „Total Length“ gibt die Gesamtlänge des IP-Pakets inkl. Header in Vielfachen von 1 B an
- „Identification“ ist ein eindeutiger 16 bit langer Wert, welcher zum Reassemblieren von Fragmenten benötigt wird
- Die drei Flags „R“, „DF“ und „MF“ stehen für „Reserved“, „Do not Fragment“ und „More Fragments“. Ersteres hat also keine Bedeutung, die beiden letzteren geben an, ob das Paket fragmentiert werden darf (0 falls ja) und ob mehr Fragmente folgen (1 falls ja).
- „Fragment Offset“ gibt an, an welche Position die in der Payload enthaltenen Daten beim Reassemblieren geschrieben werden sollen. Der Offset wird in ganzzahligen Vielfachen von 8 B angegeben!
- Die TTL wird pro Hop (also Weiterleitung durch einen Router) um 1 dekrementiert. Erreicht sie 0, wird das Paket verworfen und ein ICMP TTL Exceeded an den Sender zurückgeschickt. Der Initialwert der TTL bleibt dem Sender überlassen – ein gebräuchlicher Wert ist 64.

Am Ende dieses Übungsblatts finden Sie Vordrucke für Ethernet-Header, ARP-Pakete (Header und Payload) und IP-Header (mehr als benötigt). Es ist nicht notwendig, den Header binär auszufüllen. Achten Sie lediglich darauf, dass Sie die Zahlenbasis deutlich Kennzeichnen – z. B. `0x10` für hexadezimal oder $63_{(10)}$ für dezimal.

e) Füllen Sie für die ersten drei Rahmen aus Teilaufgabe (d) jeweils einen Ethernet-Header und die passende Payload (ARP-Paket oder IP-Header mit angedeuteter Payload) aus. Beschriften Sie die gestrichelte Box neben dem jeweiligen Header/Paket mit der in (d) vergebenen Rahmennummer.

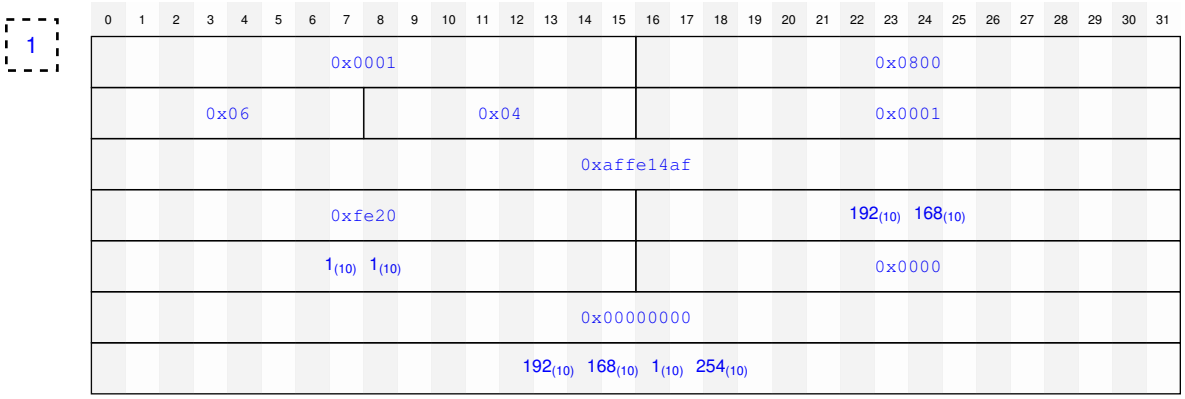
f) Füllen Sie für alle übrigen Rahmen, welche eine IP-Payload transportieren, jeweils einen Ethernet- und IP-Header aus. Beschriften Sie die gestrichelte Box neben dem jeweiligen Header mit der in (d) vergebenen Rahmennummer.

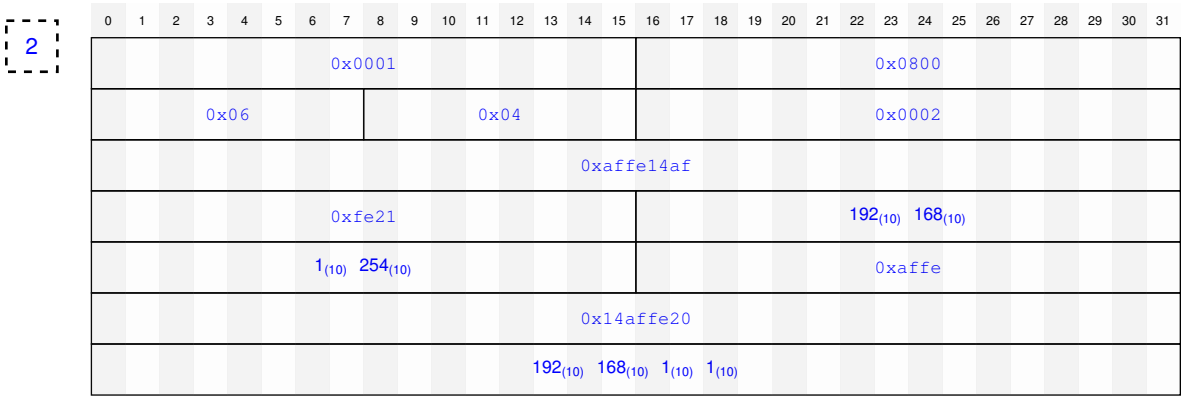
Vordrucke für Protokoll-Header:

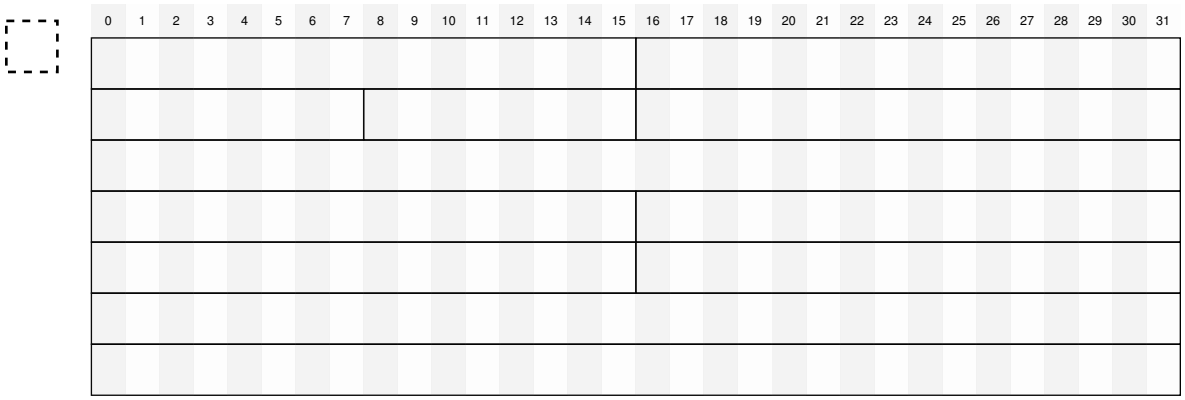
Ethernet Frames

1	ff:ff:ff:ff:ff:ff	af:fe:14:af:fe:20	0x0806	Payload	FCS
2	af:fe:14:af:fe:20	af:fe:14:af:fe:21	0x0806	Payload	FCS
3	af:fe:14:af:fe:21	af:fe:14:af:fe:20	0x0800	Payload	FCS
6	af:fe:14:af:fe:23	af:fe:14:af:fe:22	0x0800	Payload	FCS
7	af:fe:14:af:fe:23	af:fe:14:af:fe:22	0x0800	Payload	FCS
10	af:fe:14:af:fe:25	af:fe:14:af:fe:24	0x0800	Payload	FCS
11	af:fe:14:af:fe:25	af:fe:14:af:fe:24	0x0800	Payload	FCS

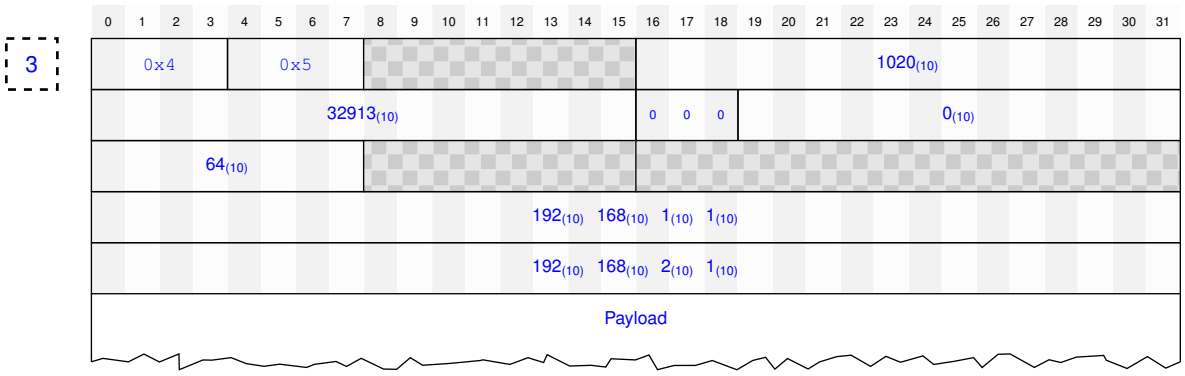
ARP-Pakete

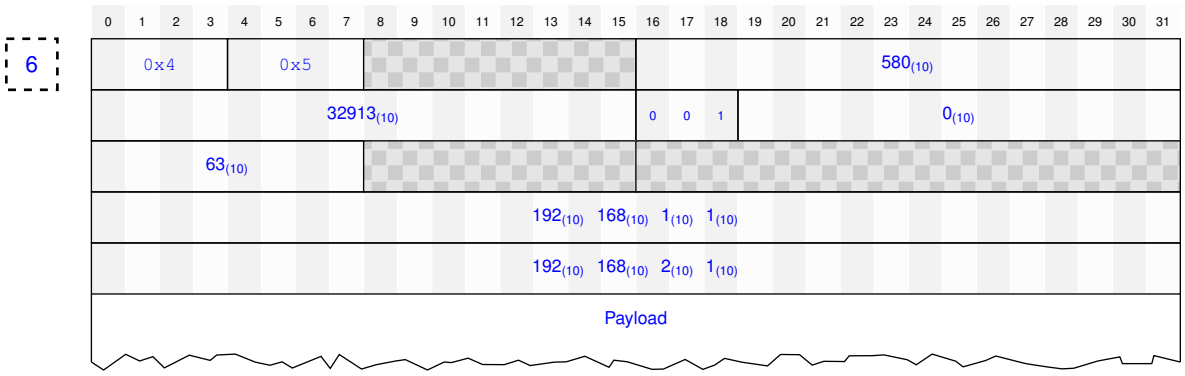


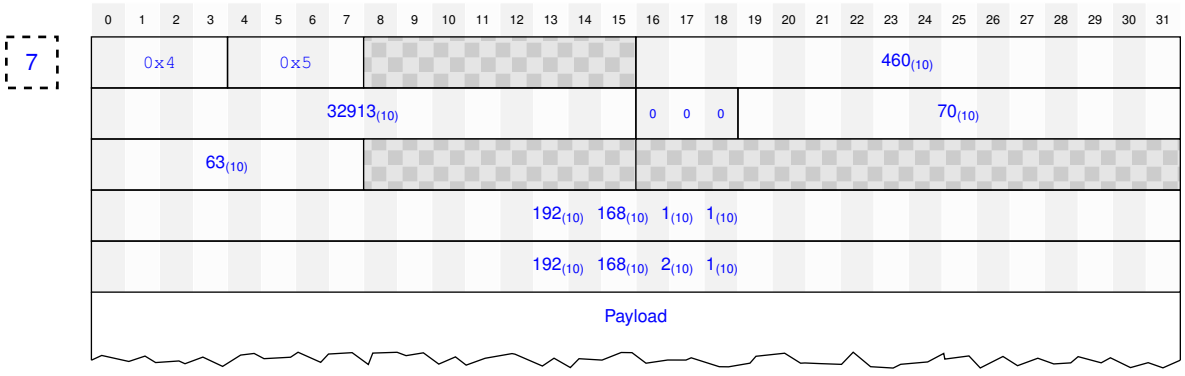


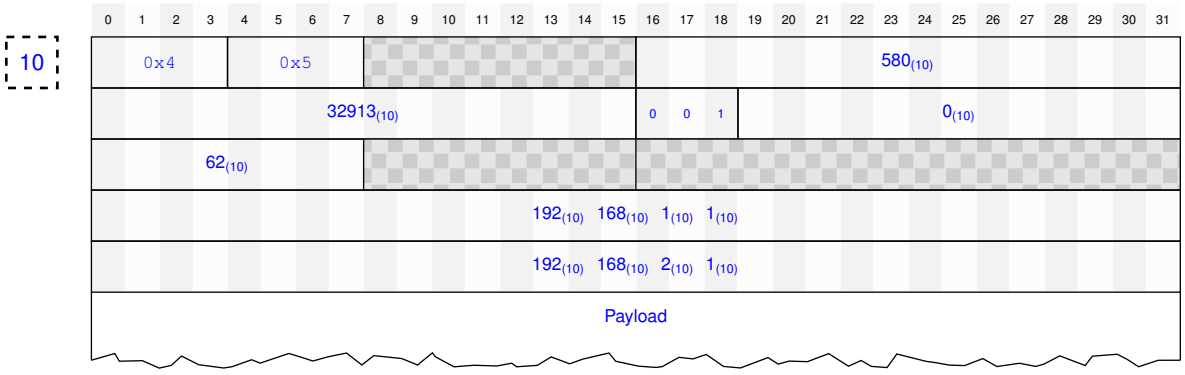


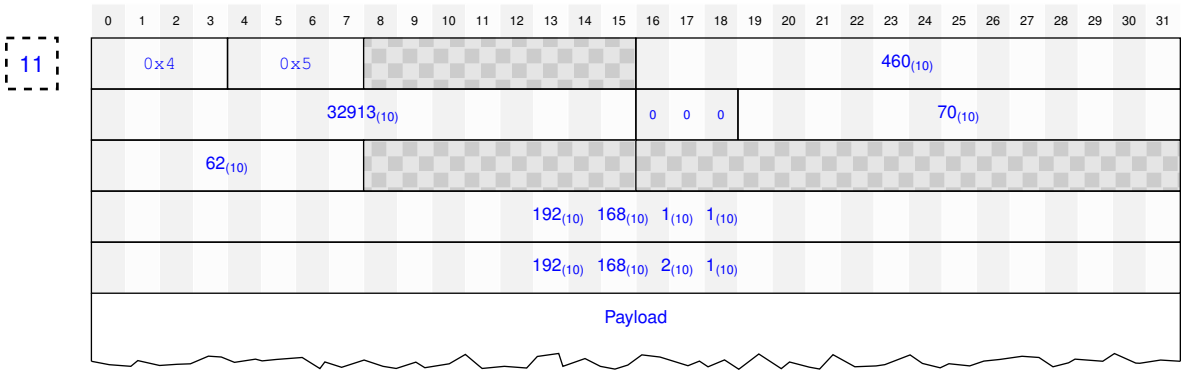
IP-Pakete

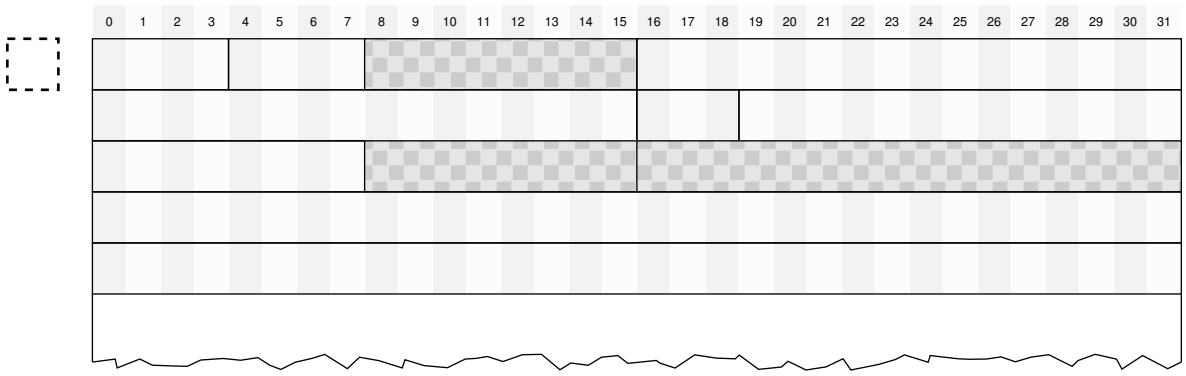


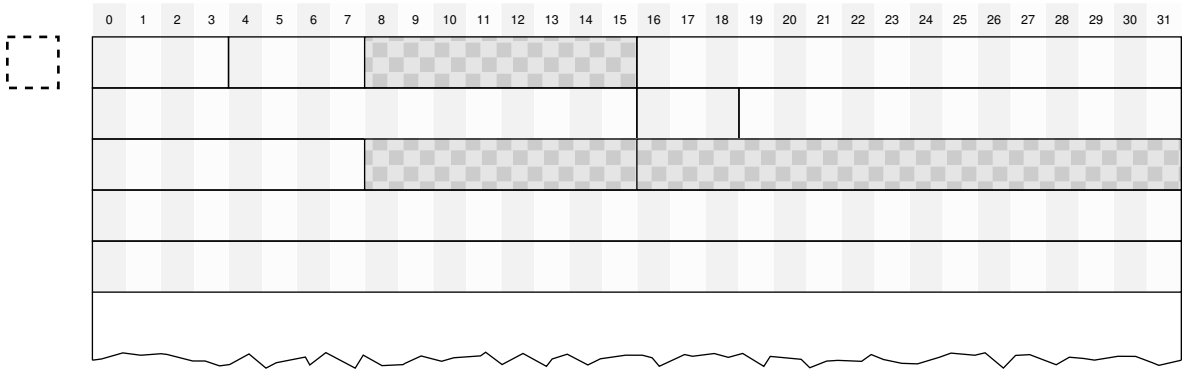

6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0x4
0x5
580(10)
32913(10)
0 0 1
0(10)
63(10)
192(10) 168(10) 1(10) 1(10)
192(10) 168(10) 2(10) 1(10)
Payload


7
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0x4
0x5
460(10)
32913(10)
0 0 0
70(10)
63(10)
192(10) 168(10) 1(10) 1(10)
192(10) 168(10) 2(10) 1(10)
Payload


10
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0x4
0x5
580(10)
32913(10)
0 0 1
0(10)
62(10)
192(10) 168(10) 1(10) 1(10)
192(10) 168(10) 2(10) 1(10)
Payload


11
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0x4
0x5
460(10)
32913(10)
0 0 0
70(10)
62(10)
192(10) 168(10) 1(10) 1(10)
192(10) 168(10) 2(10) 1(10)
Payload


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31