Grundlagen Rechnernetze und Verteilte Systeme

SoSe 2013

Kapitel 3: Vermittlungsschicht

Prof. Dr.-Ing. Georg Carle

Dipl.-Ing. Stephan M. Günther, M.Sc.
Nadine Herold, M.Sc.
Dipl.-Inf. Stephan Posselt

Fakultät für Informatik
Lehrstuhl für Netzarchitekturen und Netzdienste
Technische Universität München

Worum geht es in diesem Kapitel?

1 Motivation

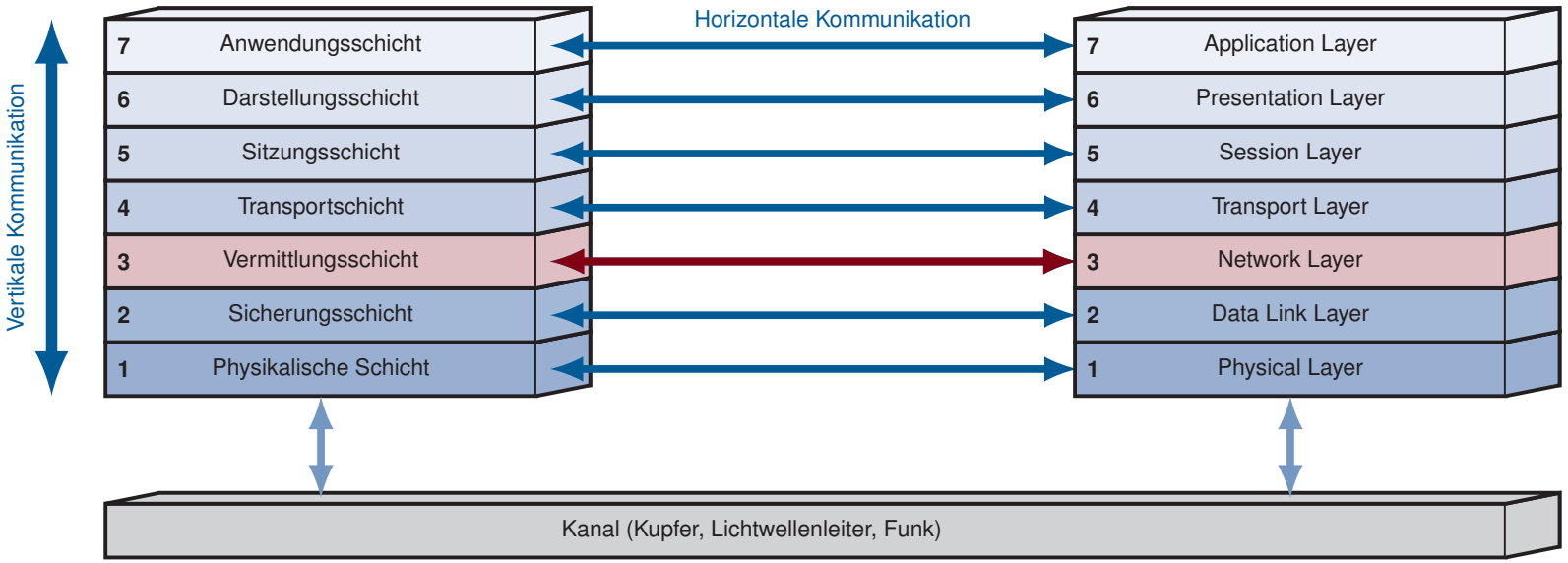
2 Vermittlungsarten

3 Adressierung im Internet

4 Wegwahl (Routing)

5 Nachfolge von IP(v4): IPv6

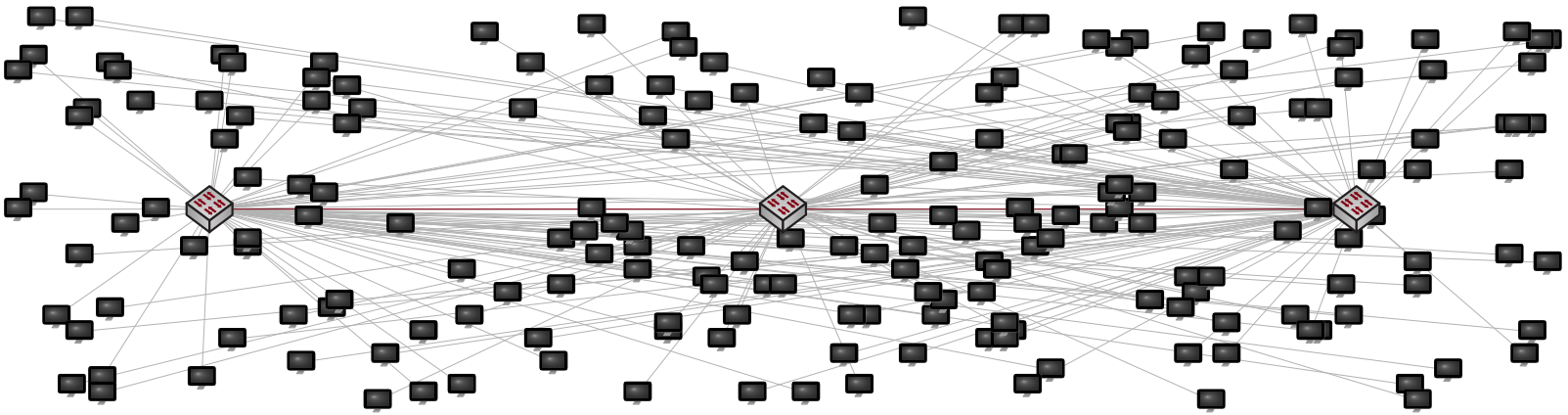
Einordnung im ISO/OSI-Modell



Motivation

Sind Direktverbindungsnetze wie Ethernet skalierbar?

- Alle angeschlossenen Hosts sind direkt bzw. über wenige Switches erreichbar
- MAC-Adressen bieten keine logische Struktur zur Adressierung
- Gruppierung von Geräten in kleinere Netze (Subnetze) durch MAC-Adressen nicht unterstützt



Aufgaben der Vermittlungsschicht:

- Kopplung unterschiedlicher Direktverbindungsnetze
- Strukturierte Aufteilung in kleinere Subnetze
- Logische und global eindeutige Adressierung von Geräten
- Wegwahl zwischen Geräten über mehrere Hops hinweg

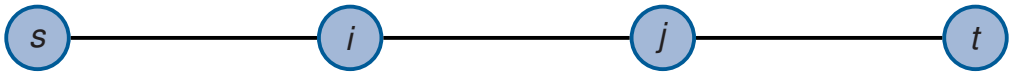
Worum geht es in diesem Kapitel?

Vermittlungsarten

Es gibt drei grundlegende Vermittlungsarten:

- **Leitungsvermittlung**
  „Reserviere eine dedizierte Leitung zwischen Sender und Empfänger“
- **Nachrichtenvermittlung**
  „Wähle für jede Nachricht individuell einen Weg“
- **Paketvermittlung**
  „Teile eine Nachricht in mehrere kleinere Pakete auf und versende jedes Paket unabhängig von den anderen“

Im Folgenden charakterisieren wir diese drei Vermittlungsarten anhand des Beispielnetzwerks



mit $n = 2$ Vermittlungsknoten hinsichtlich der Gesamtdauer T einer Übertragung von l Datenbits über die Distanz d und motivieren so die Vorteile der Paketvermittlung.

Leitungsvermittlung

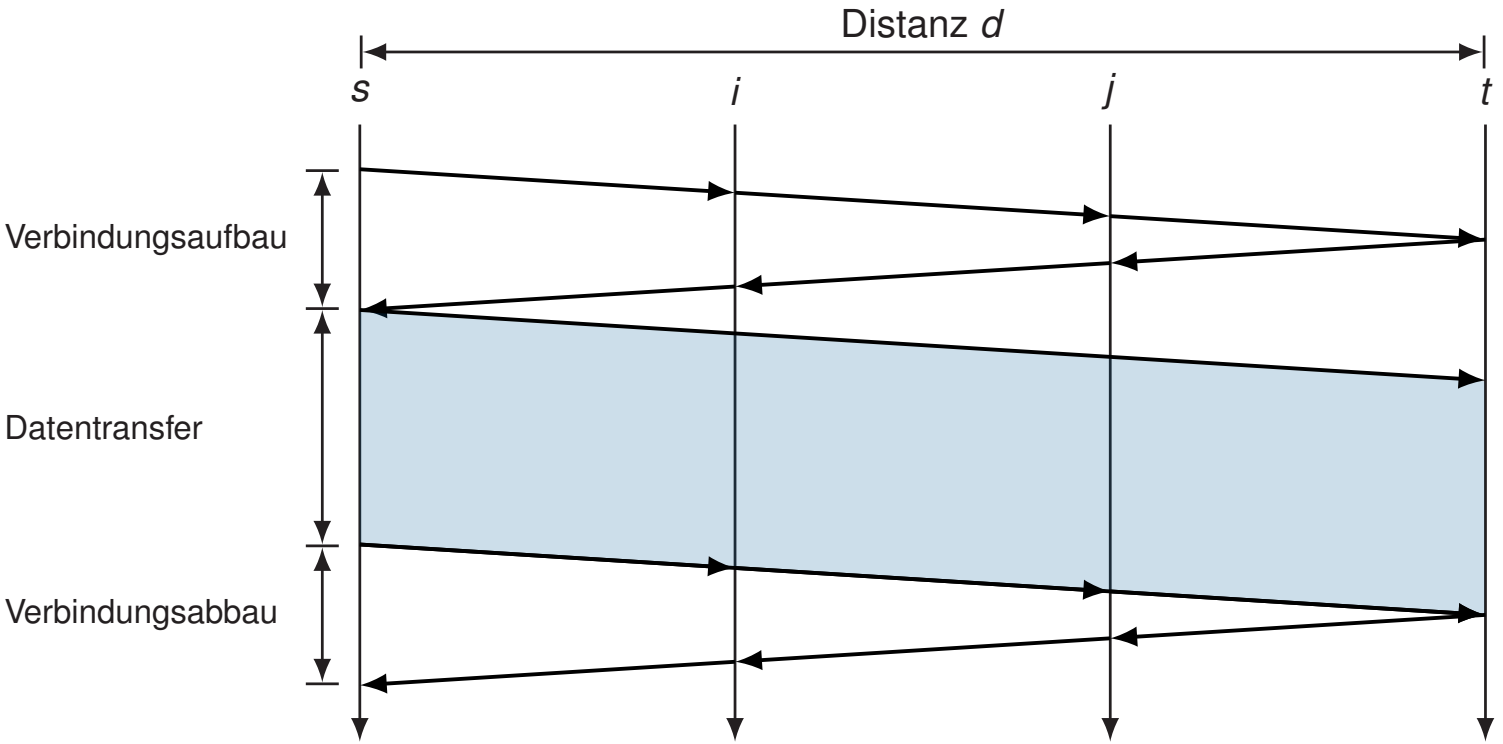
Während einer verbindungsorientierten Übertragung können drei Phasen unterschieden werden:

1. **Verbindungsaufbau**
   - Austausch von Signalisierungsnachrichten zum Aufbau einer dedizierten Verbindung zwischen Sender und Empfänger.
   - Dieser Schritt beinhaltet die Wegwahl, welche vor Beginn der Datenübertragung durchgeführt wird.
2. **Datenaustausch**
   - Kanal steht den Kommunikationspartnern zur exklusiven Nutzung bereit.
   - Auf die Adressierung des Kommunikationspartners kann während der Übertragung weitgehend verzichtet werden (Punkt-zu-Punkt-Verbindung).
3. **Verbindungsabbau**
   - Austausch von Signalisierungsnachrichten zum Abbau der Verbindung.
   - Die durch die Verbindung belegten Ressourcen werden für nachfolgende Verbindungen freigegeben.

Übertragungszeit bei Leitungsvermittlung

Wir nehmen an, dass

- die Serialisierungszeit von Signalisierungsnachrichten vernachlässigbar klein ist,
- die Verarbeitungszeiten und Wartezeiten in jedem Knoten vernachlässigbar klein sind und dass
- der Sender *s* einen Datenblock der Länge *L* an einem Stück übertragen möchte.



$$T_{LV} = 2t_p + t_s + 2t_p = t_s + 4t_p = \frac{L}{r} + 4 \cdot \frac{d}{\nu c}$$

Vorteile der Leitungsvermittlung

- Gleichbleibende Güte der dedizierten Verbindung nach dem Verbindungsaufbau
- Schnelle Datenübertragung ohne Notwendigkeit, weitere Vermittlungsentscheidungen treffen zu müssen

Nachteile der Leitungsvermittlung

- Ressourcenverschwendung, da Leitung zur exklusiven Nutzung reserviert wird
- Verbindungsaufbau kann komplex sein und benötigt u. U. weit mehr Zeit, als die Ausbreitungsverzögerungen vermuten lassen (z. B. Einwahl ins Internet mittels Modem)
- Hoher Aufwand bei der Schaltung physikalischer Verbindungen

Einsatz in heutigen Netzwerken

- Leitungsvermittlung wird häufig durch Paketvermittlung ersetzt (z. B. Voice over IP)
- In vielen Vermittlungsnetzen wird Leitungsvermittlung zumindest virtualisiert in Form von Virtual Circuits unterstützt (z. B. Frame Relay, ATM, MPLS)

Nachrichtenvermittlung

Veränderungen bei der Nachrichtenvermittlung:

- Aufbau und Abbau einer dedizierten Verbindung entfallen
- Der gesamten Nachricht der Länge L wird ein Header der Länge L_H vorangestellt

- Der Header beinhaltet insbesondere Adressinformationen, die geeignet sind, Sender und Empfänger auch über mehrere Zwischenstationen hinweg eindeutig zu identifizieren
- Die so entstehende PDU wird als Ganzes übertragen

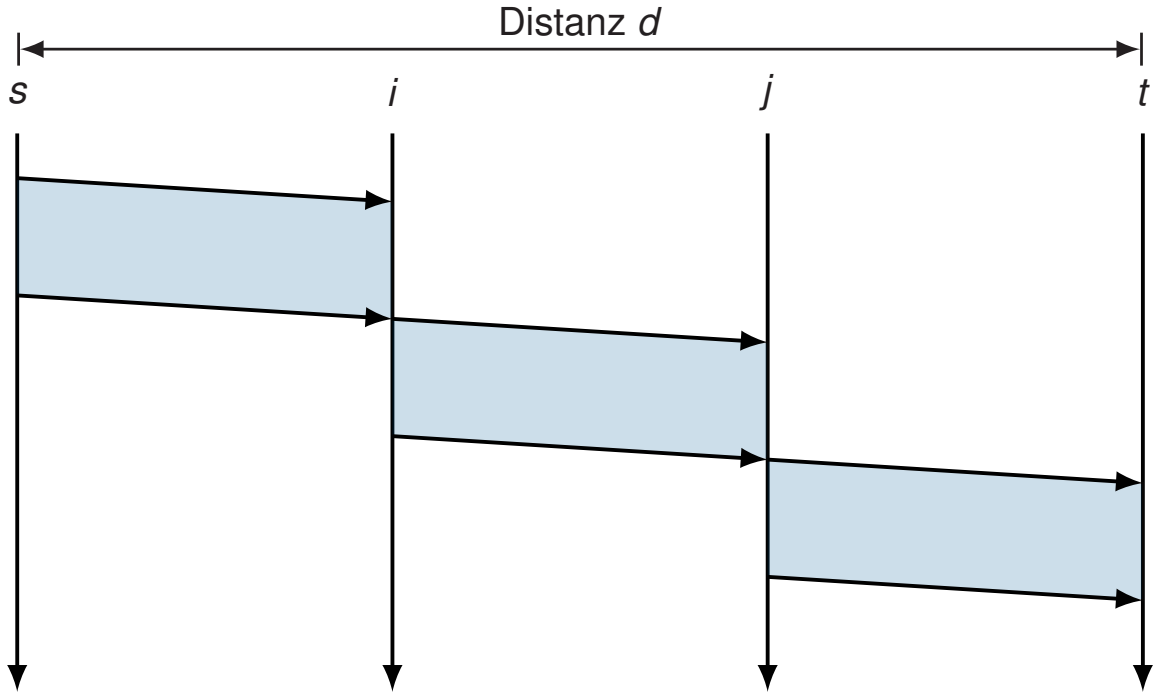
Hoffnung: Zeitersparnis, da die Phasen zum Aufbau und Abbau der Verbindung entfallen.

Analogie: Post / DHL / Paketdienste

- Absender verpackt Ware und versieht das Paket mit Adressinformationen (Header)
- Die Adressen identifizieren Absender und Empfänger weltweit eindeutig und haben eine logische Struktur, die eine effiziente Zuordnung im Transportnetz der Post erlaubt

Übertragungszeit bei Nachrichtenvermittlung

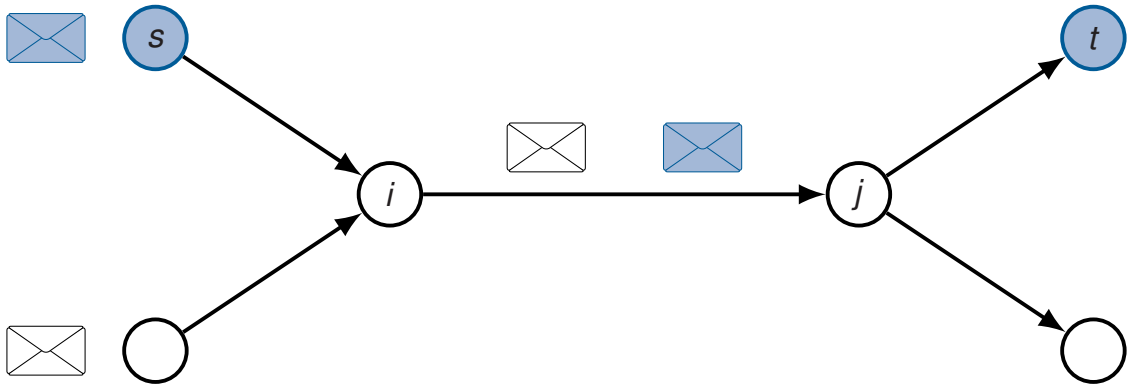
Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.



$$T_{NV} = (n+1) \cdot t_s + t_p = (n+1) \cdot \frac{L_H + L}{r} + \frac{d}{\nu c}$$

Multiplexing auf Nachrichtenebene

- Das Wegfallen fest vorgegebener Pfade ermöglicht die gemeinsame Nutzung von Teilstrecken
- Dies entspricht dynamischem Zeitmultiplex (Time Division Multiplex, TDM)



Vorteile:

- Flexibles Zeitmultiplex von Nachrichten
- Bessere Ausnutzung der Kanalkapazität
- Keine Verzögerung beim Senden des ersten Pakets durch Verbindungsaufbau

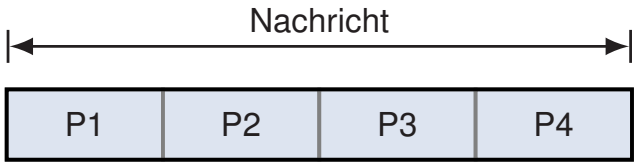
Nachteile:

- Pufferung von Nachrichten, wenn (i, j) ausgelastet
- Verlust von Nachrichten durch begrenzten Puffer möglich (Stausituation $\rightarrow$ Kapitel 4)
- Mehrfache Serialisierung der ganzen Nachricht

Paketvermittlung

Unterschiede zur Nachrichtenvermittlung:

- Nachrichten werden nicht mehr als Einheit übertragen sondern in kleinere Pakete unterteilt:

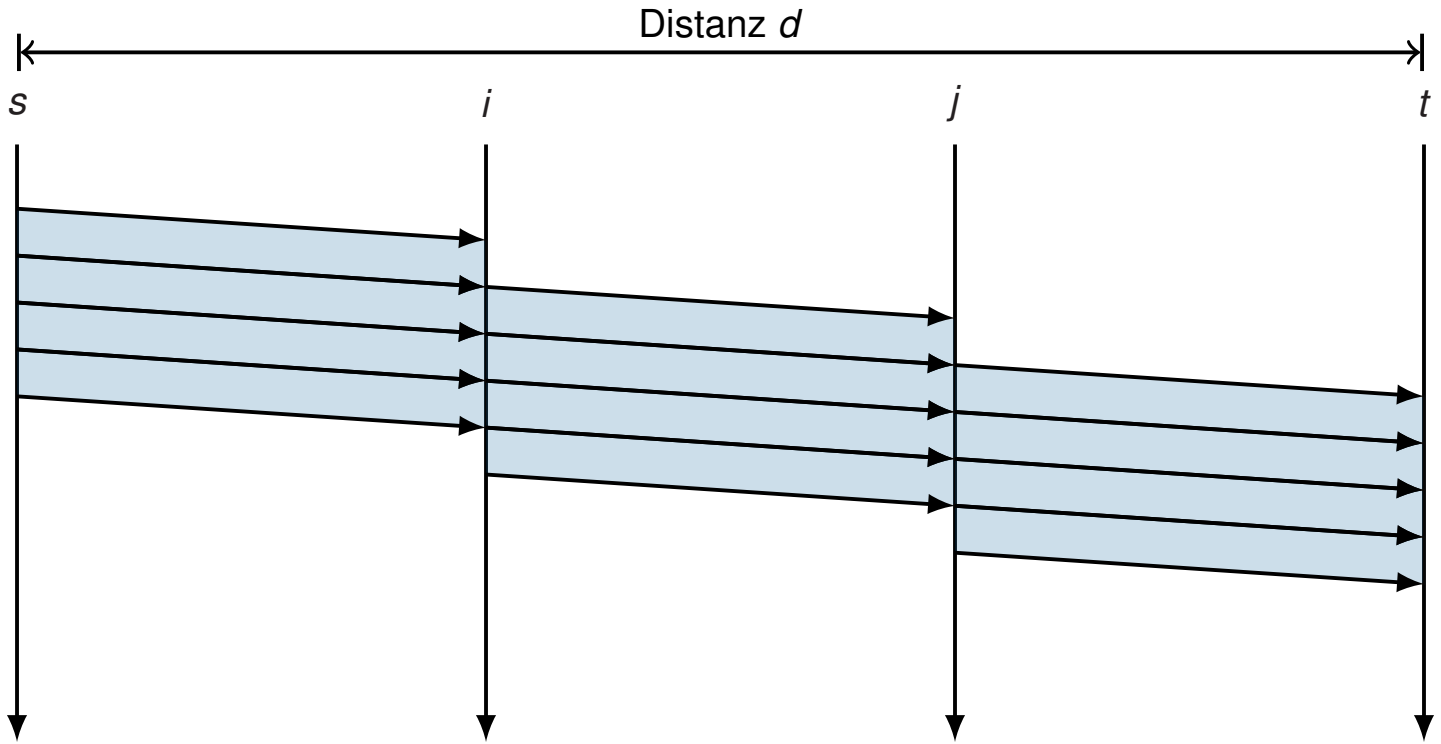


- Jedem Paket wird ein eigener Header vorangestellt, der alle Informationen zur Weiterleitung und ggf. auch zur Reassemblierung enthält:

- Pakete werden unabhängig voneinander vermittelt, d. h. Pakete derselben Nachricht können über unterschiedliche Wege zum Empfänger gelangen.
- Im Allgemeinen müssen die einzelnen Pakete nicht gleich groß sein, es gibt aber Anforderungen an die minimale (p_{min}) und maximale (p_{max}) Paketgröße.

Übertragungszeit bei Paketvermittlung

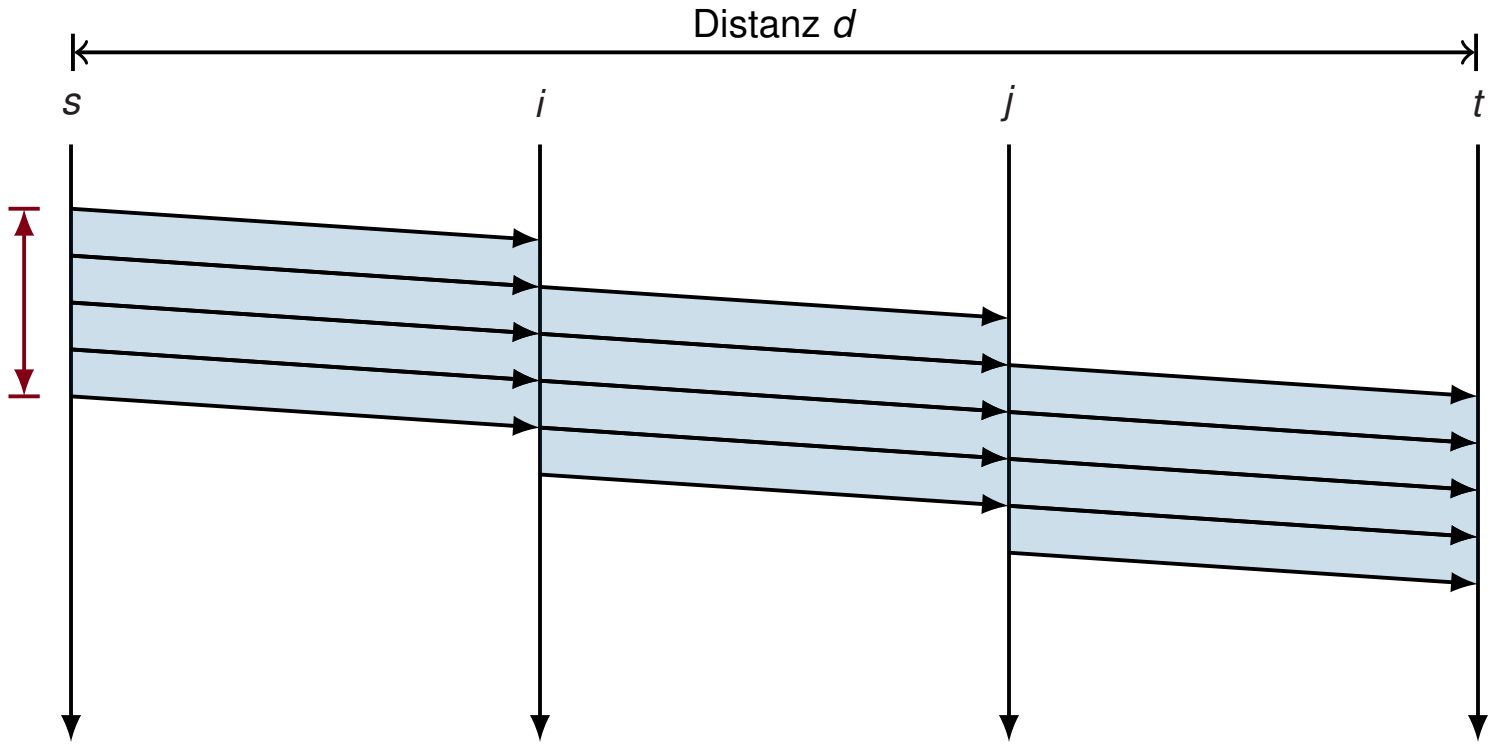
- Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.
- Vereinfachend nehmen wir an, dass die Nachrichtenlänge ein Vielfaches von $p_{\max}$ ist. ($\Rightarrow$ alle Pakete haben dieselbe Größe $p_{\max}$).



$$T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c} + n \cdot \frac{L_h + p_{\max}}{r}$$

Übertragungszeit bei Paketvermittlung

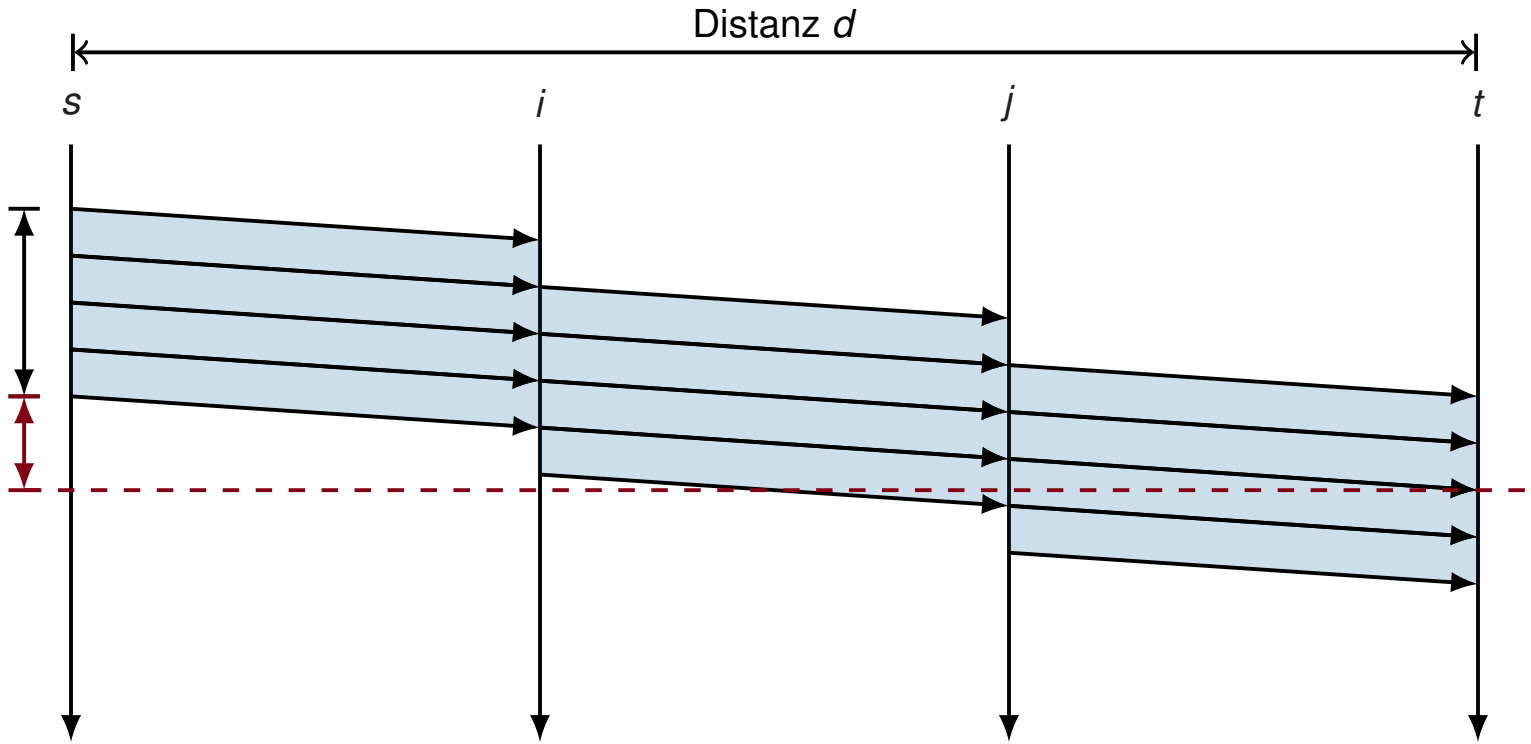
- Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.
- Vereinfachend nehmen wir an, dass die Nachrichtenlänge ein Vielfaches von $p_{\max}$ ist. ($\Rightarrow$ alle Pakete haben dieselbe Größe $p_{\max}$).



$$T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c} + n \cdot \frac{L_h + p_{\max}}{r}$$

Übertragungszeit bei Paketvermittlung

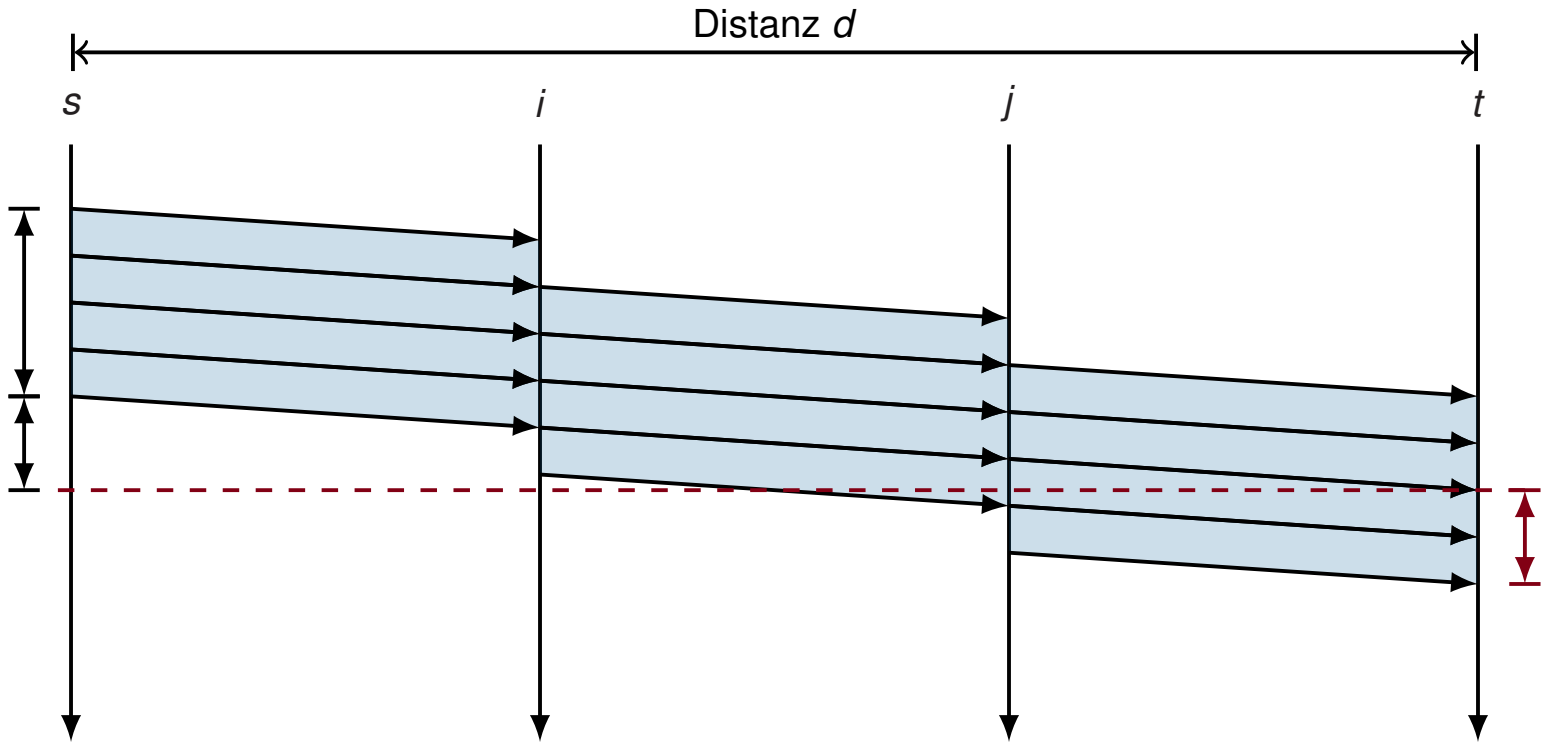
- Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.
- Vereinfachend nehmen wir an, dass die Nachrichtenlänge ein Vielfaches von $p_{\max}$ ist. ($\Rightarrow$ alle Pakete haben dieselbe Größe $p_{\max}$).



$$T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c} + n \cdot \frac{L_h + p_{\max}}{r}$$

Übertragungszeit bei Paketvermittlung

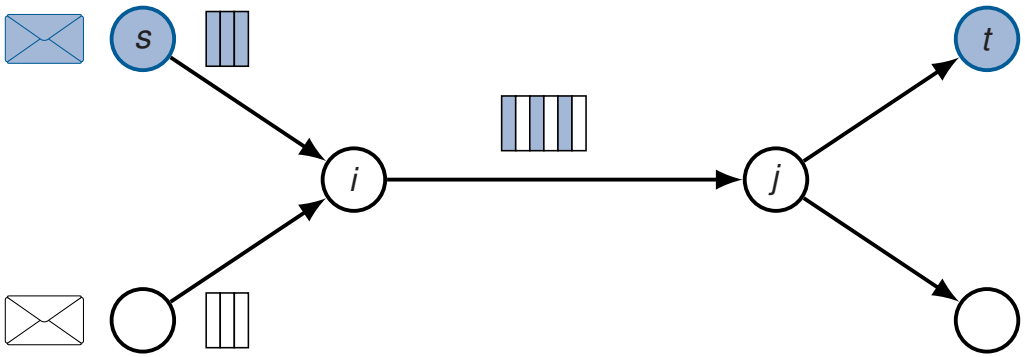
- Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.
- Vereinfachend nehmen wir an, dass die Nachrichtenlänge ein Vielfaches von $p_{\max}$ ist. ($\Rightarrow$ alle Pakete haben dieselbe Größe $p_{\max}$).



$$T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c} + n \cdot \frac{L_h + p_{\max}}{r}$$

Multiplexing auf Paketebene

- Durch die Vermittlung kleiner Pakete statt langer Nachrichten werden Engpässe fairer genutzt
- Gehen Pakete verloren, müssen nur Teile einer größeren Nachricht wiederholt werden



Vorteile:

- Flexibles Zeitmultiplex einzelner Pakete
- Pufferung kleiner Pakete statt ganzer Nachrichten

Nachteile:

- Verlust von Paketen durch begrenzten Puffer möglich
- Jedes Paket benötigt seinen eigenen Header (Overhead)
- Empfänger muss Pakete wieder zusammensetzen

Vergleich der drei Verfahren

$$\text{Leitungsvermittlung:} \quad T_{LV} = \frac{L}{r} + 4\frac{d}{\nu c}$$

$$\text{Nachrichtenvermittlung:} \quad T_{NV} = (n+1)\frac{L_h + L}{r} + \frac{d}{\nu c}$$

$$\text{Paketvermittlung:} \quad T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c} + n \cdot \frac{L_h + p_{\max}}{r}$$

Zahlenbeispiel:

- Die Distanz zwischen Sender und Empfänger beträgt $d = 1000\,\text{km}$
- Für Glasfaserleitungen beträgt $\nu = 0.9$
- Es kommen $n = 5$ Vermittlungsknoten zum Einsatz
- Die Datenrate beträgt auf allen Teilstrecken $r = 1\,\text{Mbit/s}$
- Die Länge der zu sendenden Nachricht beträgt $L = 5\,\text{MiB}$
- Die maximale Paketgröße betrage $p_{\max} = 1480\,\text{B}$
- Die Headergröße pro Nachricht / Paket betrage $L_h = 20\,\text{B}$

Es ergeben sich folgende Zahlenwerte:

$$T_{LV} \approx 42\,\text{s}, \quad T_{NV} \approx 5033\,\text{s} \text{ und } \quad T_{PV} \approx 43\,\text{s}.$$

Wo werden die Verfahren eingesetzt?

Leitungsvermittlung:

- Analoge Telefonverbindungen (POTS)
- Interneteinwahl („letzte Meile“)
- Standortvernetzung von Firmen
- Virtuelle Kanäle (engl. Virtual Circuits) in unterschiedlichen Arten von Vermittlungsnetzen (Frame Relay, ATM, MPLS, . . .)

Nachrichtenvermittlung:

- Kaum praktische Anwendung auf Schicht 3
- Aber: Nachrichtenvermittlung existiert aus Sicht höherer Schichten (ab Schicht 5 aufwärts)

Paketvermittlung:

- In den meisten modernen Datennetzen
- Zunehmend auch zur Sprachübertragung (Voice over IP)
- Digitales Radio / Fernsehen
- Viele Peripherieschnittstellen an Computern (PCI, USB, Thunderbolt)

Übersicht

Adressierung im Internet

Die Sicherungsschicht (Schicht 2) bietet

- mehr oder weniger fairen Medienzugriff bei von mehreren Hosts geteilten Medien,
- einen „ausreichenden“ Schutz vor Übertragungsfehler und
- Adressierung innerhalb eines Direktverbindungsnetzes.

Die Vermittlungsschicht (Schicht 3) ergänzt dies um

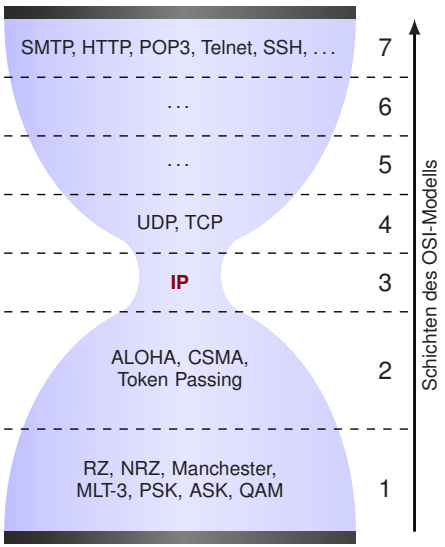
- Möglichkeiten zur global eindeutigen und strukturierten / logischen Adressierung sowie
- Verfahren zur Bestimmung von (möglichst) optimalen Pfaden.

Wir beschränken uns in diesem Teilkapitel auf die Betrachtung von

- IP (Internet Protocol, 1981) bzw.
- seinem Nachfolger IPv6 (1998).

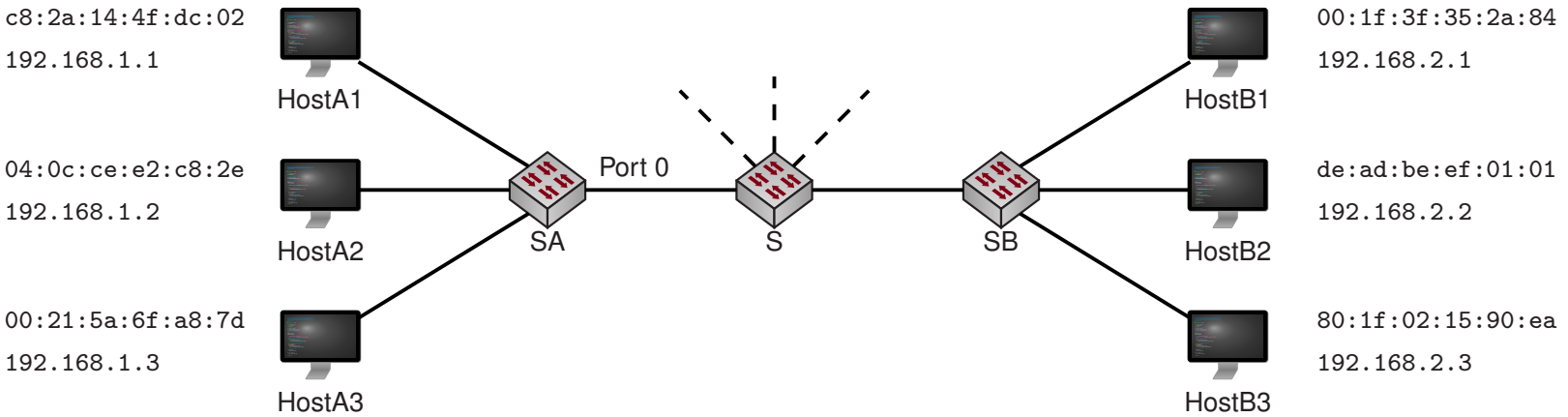
Beispiele für alternative Protokolle der Netzwerkschicht:

- IPX (Internetwork Packet Exchange, 1990)
- DECnet Phase 5 (1987)
- AppleTalk (1983)



Internet Protocol (IPv4) [3]

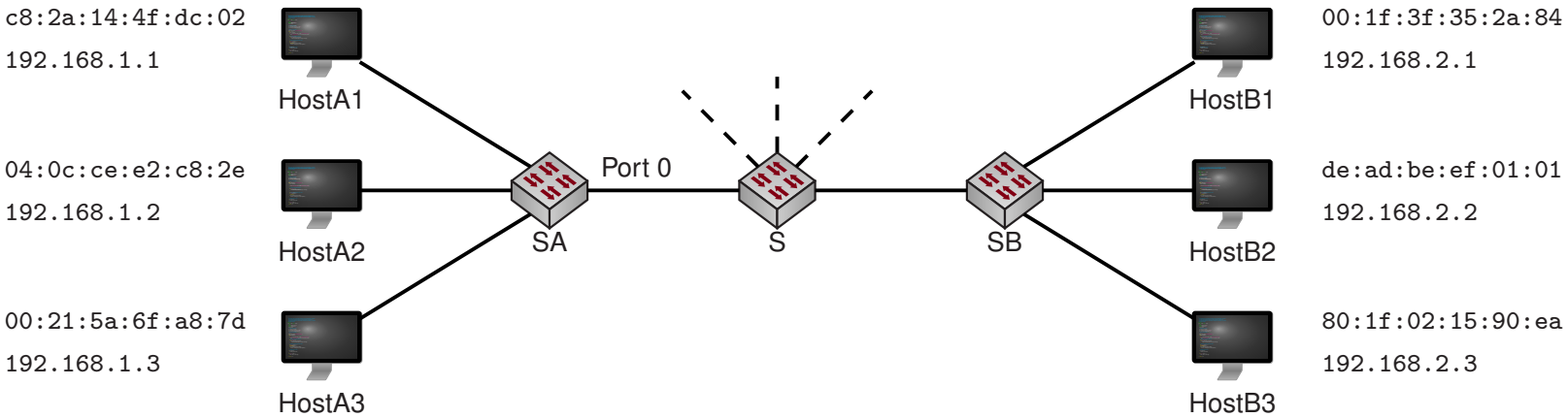
Wir betrachten das folgende Beispielnetz, welches auf einem aktuellen Ethernet-Standard basiert:



Wie viele Einträge enthält die Switching-Tabelle von Switch SA?

Internet Protocol (IPv4) [3]

Wir betrachten das folgende Beispielnetz, welches auf einem aktuellen Ethernet-Standard basiert:



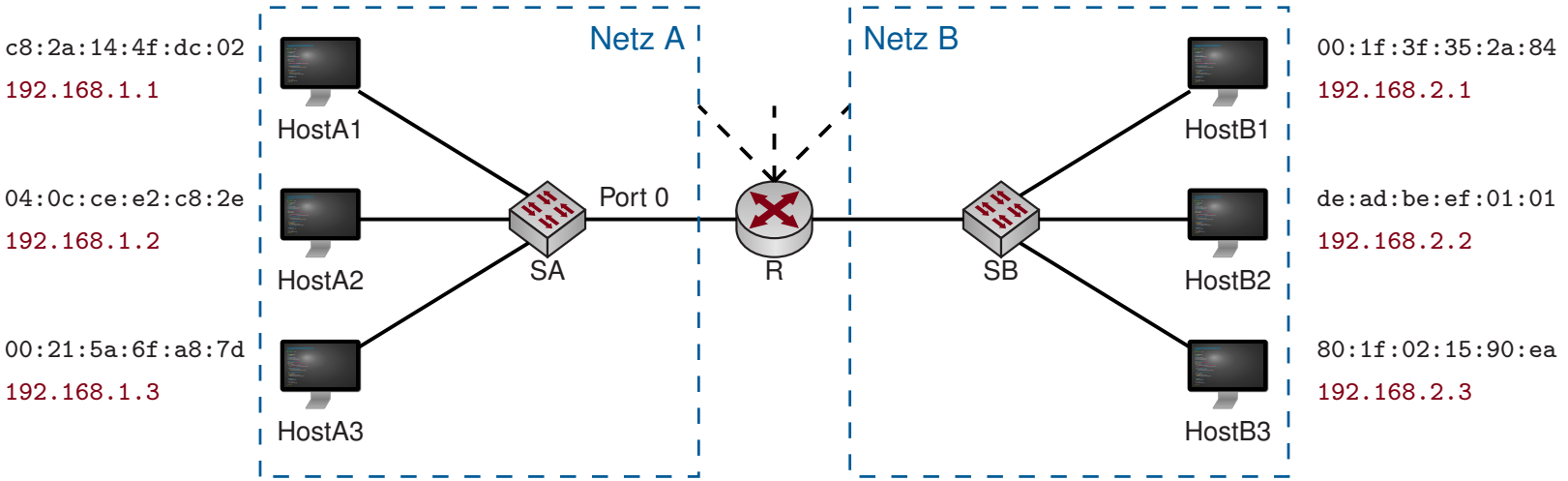
Wie viele Einträge enthält die Switching-Tabelle von Switch SA?

- Genau einen Entrag für jeden bekannten Host
- Die meisten MAC-Adressen werden auf Port 0 abgebildet
- Eine Zusammenfassung von Einträgen ist i. A. nicht möglich, da MAC-Adressen nicht die Position eines Knotens innerhalb eines Netzwerks wiederspiegeln

Port	MAC
0	00:1f:3f:35:2a:84
0	de:ad:be:ef:01:01
0	80:1f:02:15:90:ea
1	c8:2a:14:4f:dc:02
2	04:0c:ce:e2:c8:2e
3	00:21:5a:6f:a8:7d
⋮	⋮

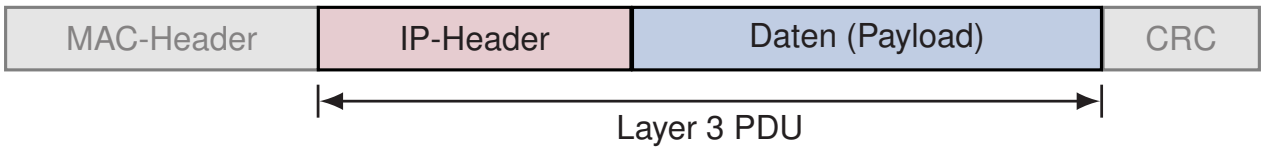
$\Rightarrow$ Es erscheint sinnvoll, von physikalischen Adressen zu abstrahieren

Wir betrachten das Beispielnetz mit einem Router (R) in der Mitte:



- Jedem Host ist eine IP-Adresse zugewiesen. Jede IP-Adresse ist in vier Gruppen zu je einem Byte, durch Punkte getrennt, dargestellt (Dotted Decimal Notation).
- In diesem Beispiel identifiziert das 4. Oktett einen Host innerhalb eines Netzes.
- Die ersten drei Oktette identifizieren das Netzwerk, in dem sich der Host befindet.
- Der Router R trifft Weiterleitungsentscheidungen auf Basis der Ziel-IP-Adresse.

$\Rightarrow$ Jedes Paket muss mit einer Absender- und Ziel-IP-Adresse (im IP-Header) versehen werden:



IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

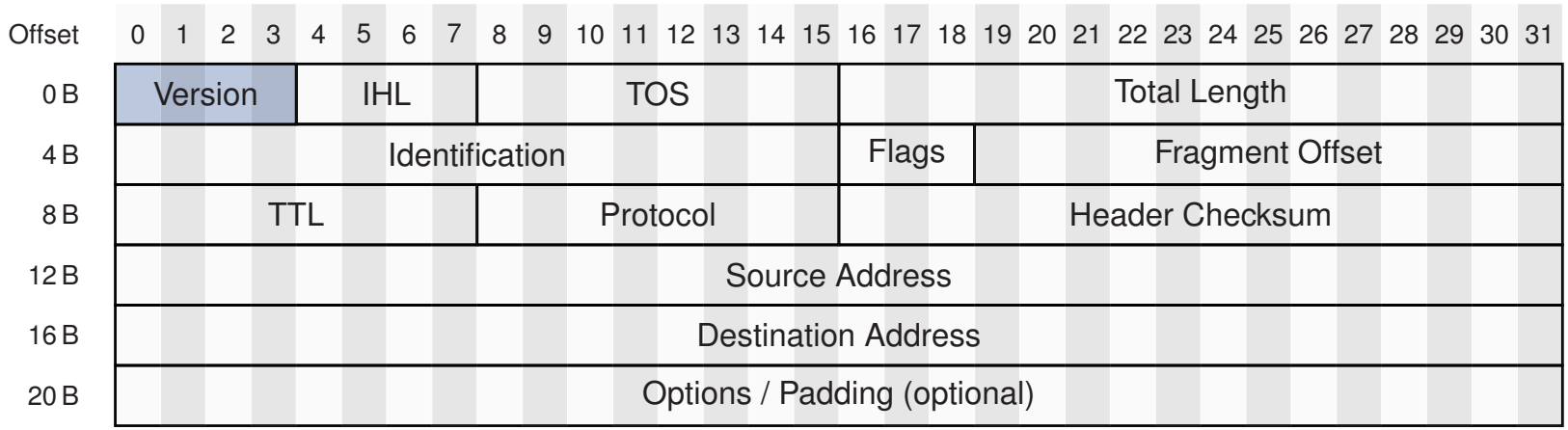


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Version

- Gibt die verwendete IP-Version an.
- Gültige Werte sind 4 (IPv4) und 6 (IPv6).

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

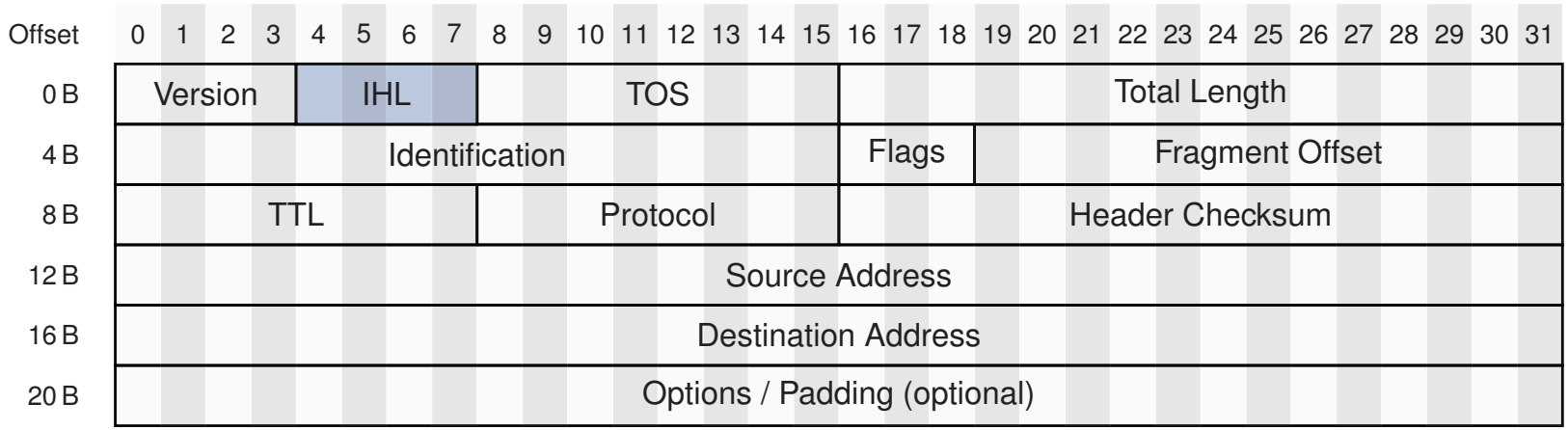


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

IHL (IP Header Length)

- Gibt die Länge des IP Headers inkl. Optionen in Vielfachen von 32 Bit an.
- Wichtig, da der IPv4-Header durch Optionsfelder variable Länge hat.

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

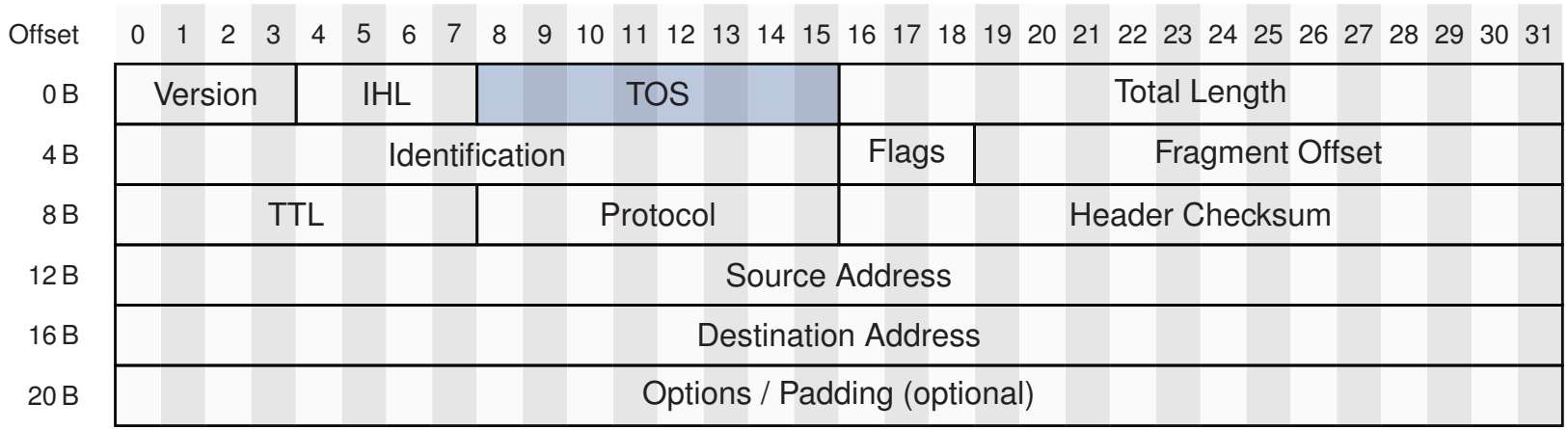


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

TOS (Type of Service)

- Dient der Klassifizierung und Priorisierung von IP-Paketen (z. B. Hinweis auf zeitsensitive Daten wie Sprachübertragungen).
- Möglichkeit zur Staukontrolle (Explicit Congestion Notification) auf Schicht 3 (optional).

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

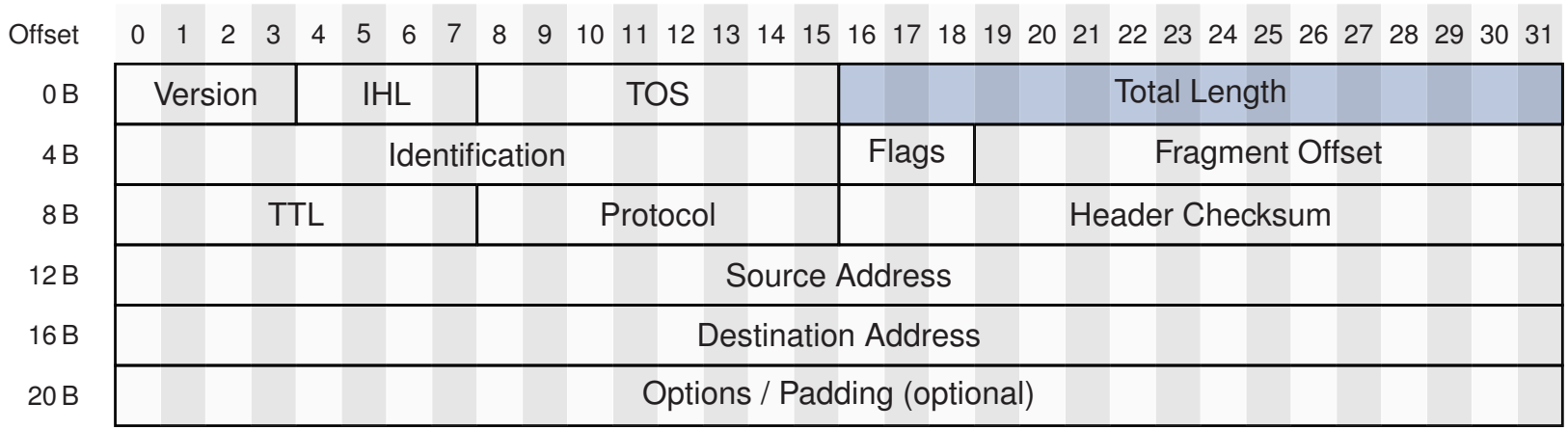


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Total Length

- Gibt die Gesamtlänge des IP-Pakets (Header + Daten) in Bytes an.
- Die Maximallänge eines IP-Pakets beträgt damit 65535 Byte.
- Der Sender passt die Größe ggf. an, um Fragmentierung zu vermeiden.
- Die maximale Paketlänge, so dass keine Fragmentierung notwendig ist, bezeichnet man als Maximum Transmission Unit (MTU). Diese ist Abhängig von Schicht 2/1 und beträgt bei FastEthernet 1500 Byte.

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

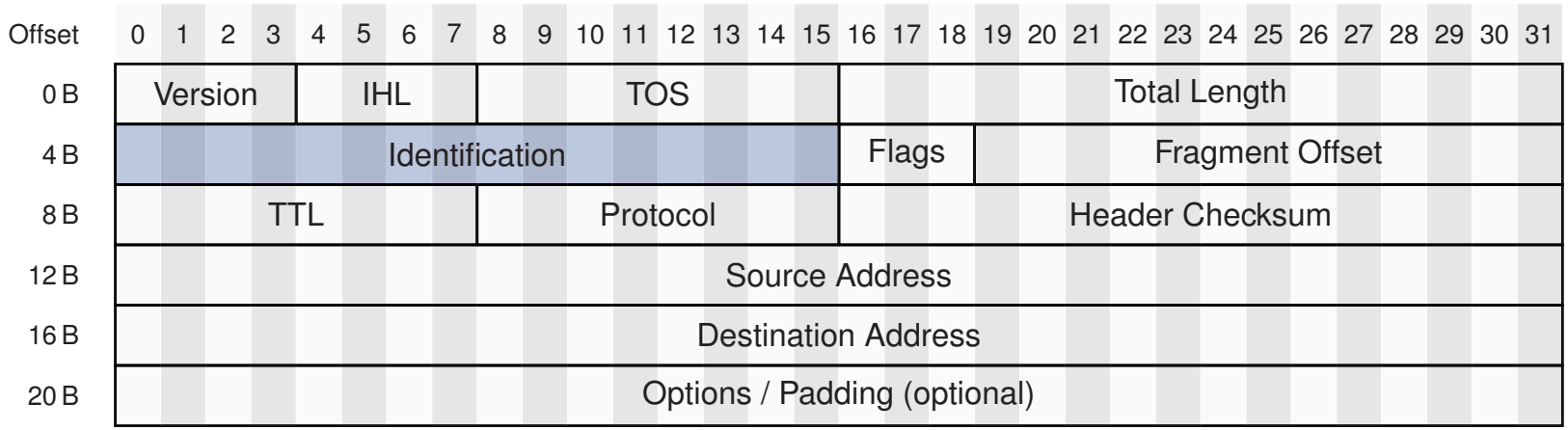


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Identification

- Für jedes IP-Paket (zufällig) gewählter 16 Bit langer Wert.
- Dient der Identifikation zusammengehörender Fragmente (IP-Fragmentierung $\rightarrow$ später).

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

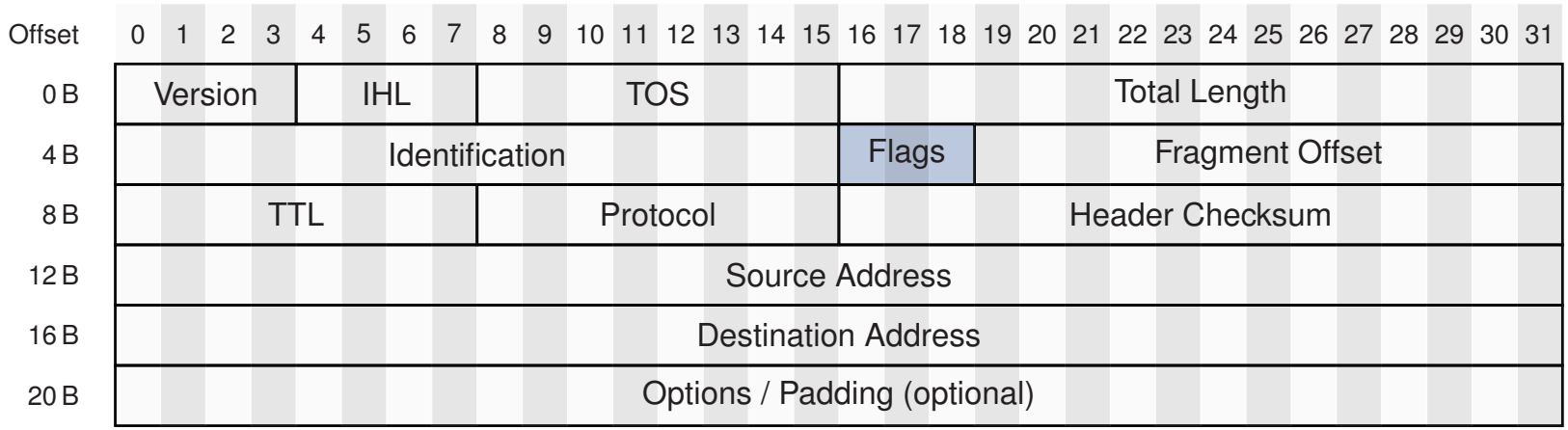


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Flags

- Bit 16: Reserviert und wird auf 0 gesetzt.
- Bit 17: Don't Fragment (DF). Ist dieses Bit 1, so darf das IP-Paket nicht fragmentiert werden.
- Bit 18: More Fragments (MF). Gibt an, ob weitere Fragmente folgen (1) oder dieses Paket das letzte Fragment ist (0). Wurde das Paket nicht fragmentiert, wird es ebenfalls auf 0 gesetzt.

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

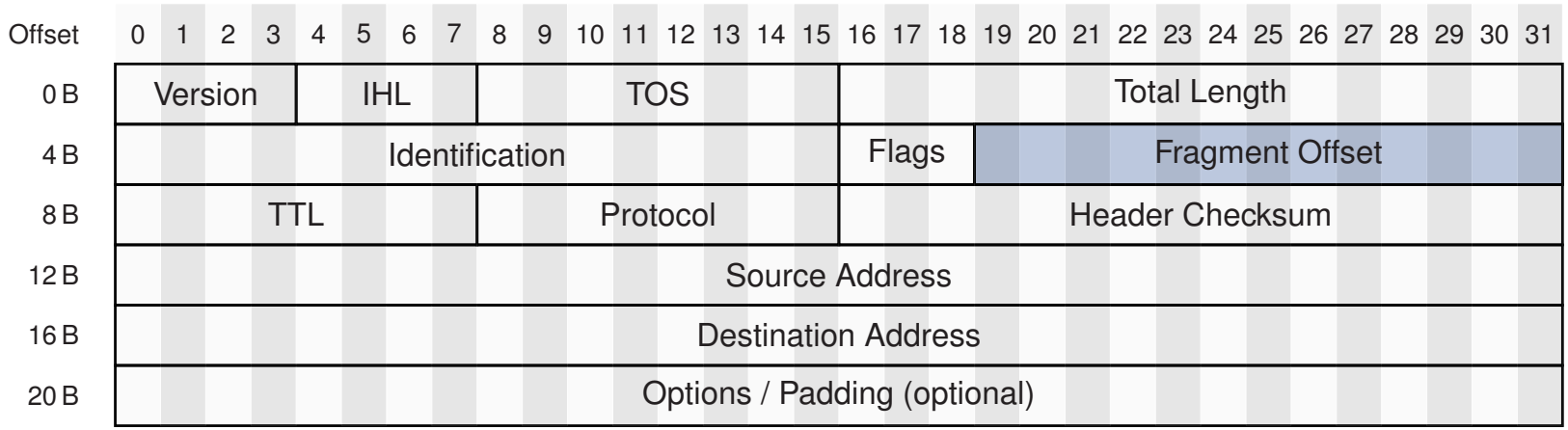


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Fragment Offset

- Gibt die absolute Position der Daten in diesem Fragment bezogen auf das unfragmentierte Paket in ganzzahligen Vielfachen von 8 Byte an.
- Ermöglicht zusammen mit dem Identifier und MF-Bit die Reassemblierung fragmentierter Pakete in der richtigen Reihenfolge.

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

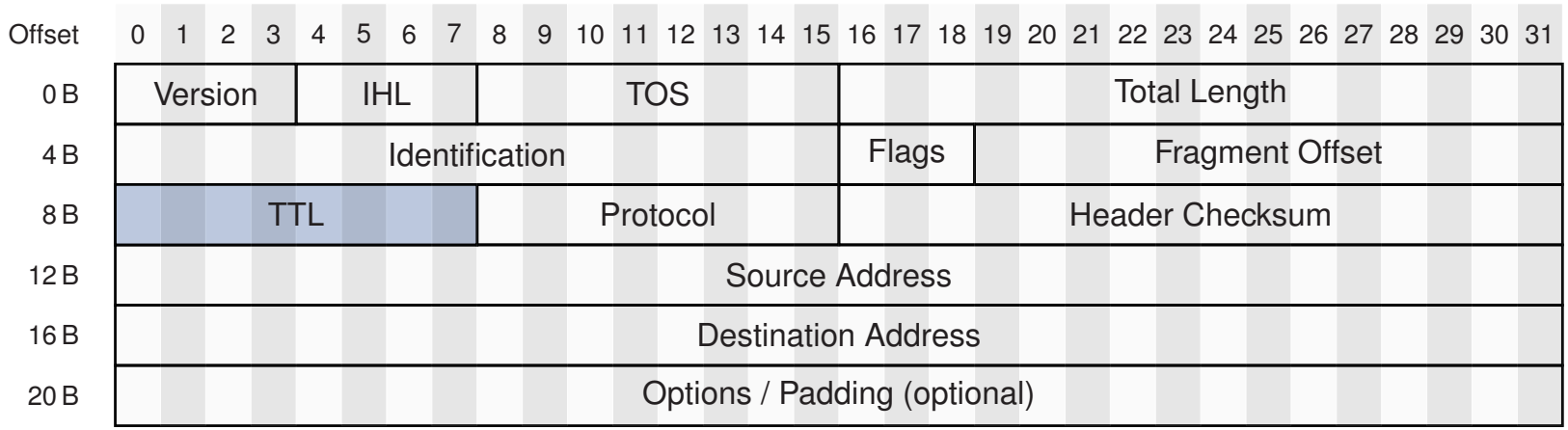


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

TTL (Time to Live)

- Leitet ein Router ein IP-Paket weiter, so dekrementiert er das TTL-Feld um 1.
- Erreicht das TTL-Feld den Wert 0, so verwirft ein Router das Paket und sendet eine Benachrichtigung an den Absender (ICMP Time Exceeded → später).
- Dieser Mechanismus beschränkt die Pfadlänge im Internet und verhindert endlos kreisende Pakete infolge von Routing Loops.

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

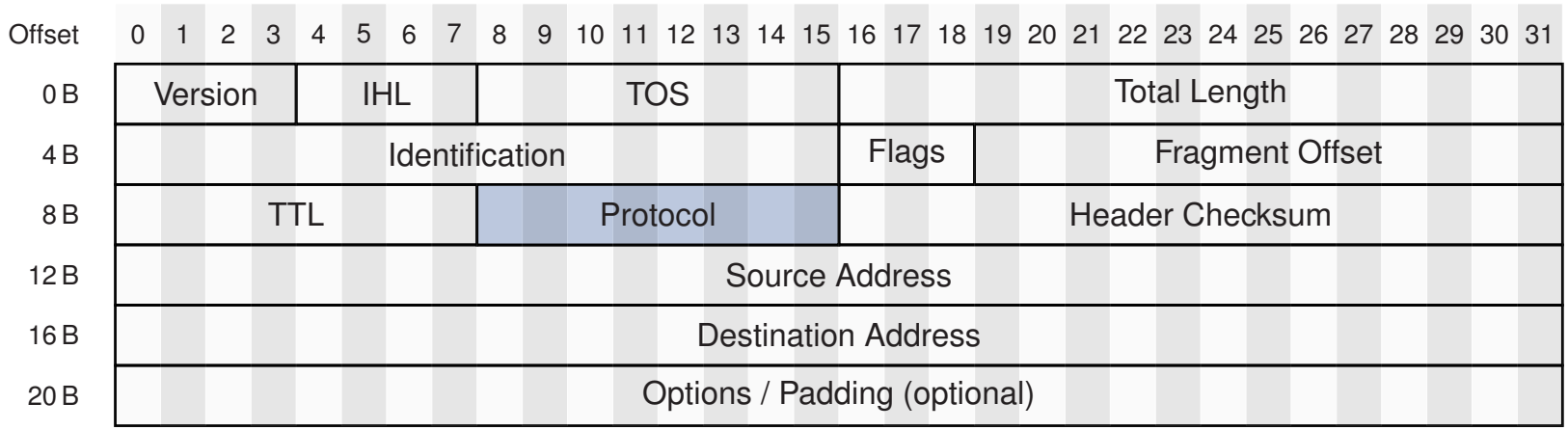


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Protocol

- Identifiziert das Protokoll auf Schicht 4, welches in der Payload (Daten) des IP-Pakets enthalten ist.
- Relevant u. a. für das Betriebssystem, um Pakete dem richtigen Prozess zuordnen zu können.
- Gültige Werte sind beispielsweise `0x06` (TCP) und `0x08` (UDP).

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

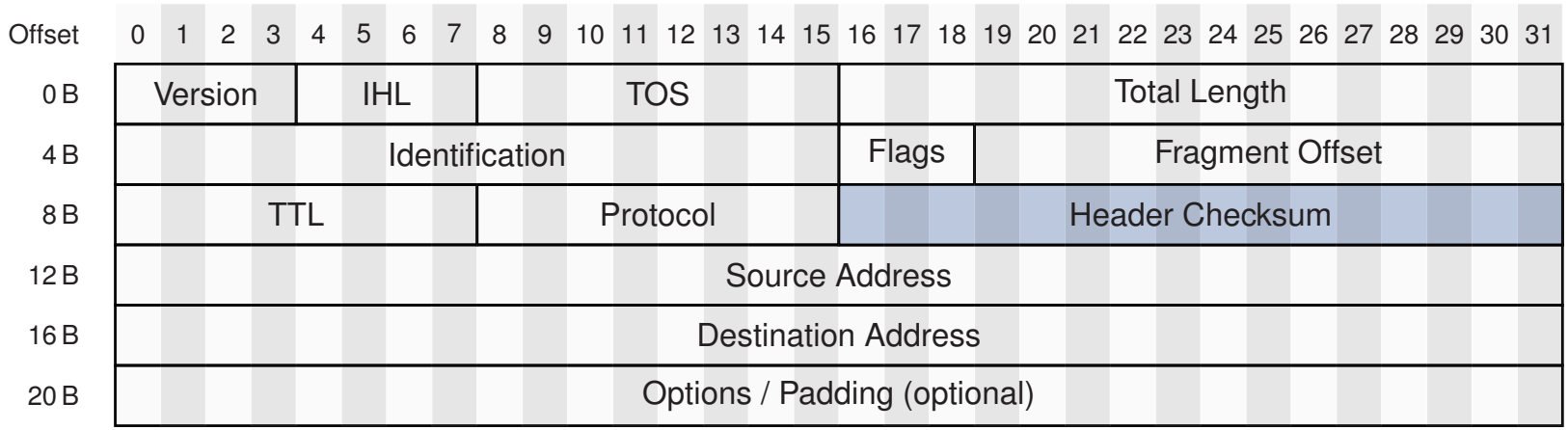


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Header Checksum

- Einfache, auf Geschwindigkeit optimierte Prüfsumme, welche nur den IP-Header (ohne Daten) schützt.
- Die Checksumme ist so ausgelegt, dass die Dekrementierung des TTL-Felds einer Inkrementierung der Checksumme entspricht. Es ist also keine Neuberechnung der Checksumme bei der Weiterleitung von Paketen notwendig.
- Es ist lediglich Fehlererkennung aber keine Korrektur möglich.

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

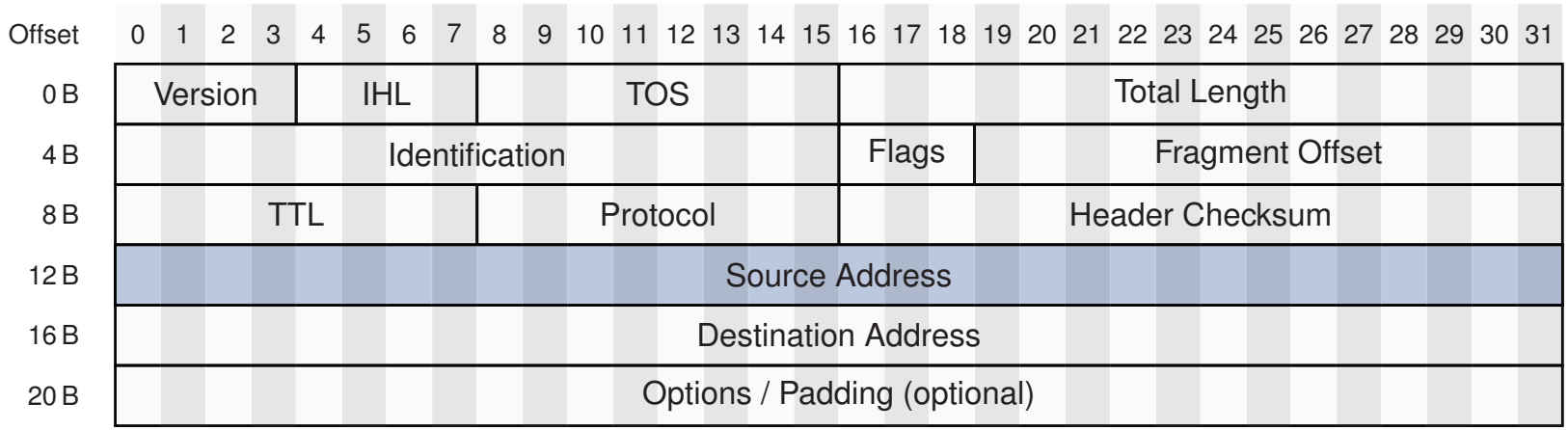


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Source Address

- IP-Adresse des Absenders.

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

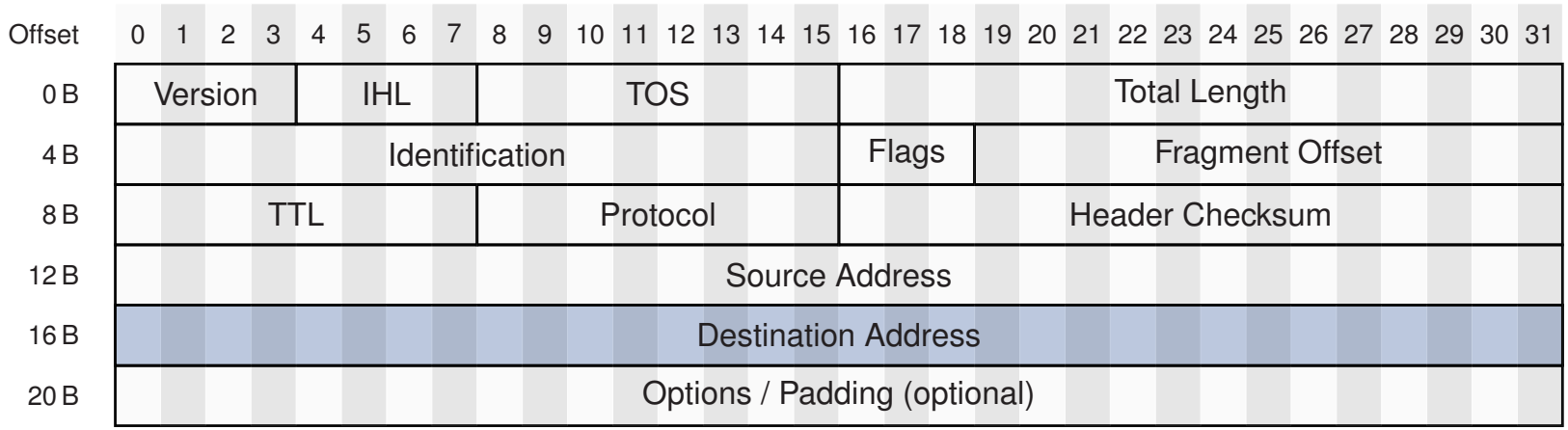


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Destination Address

- IP-Adresse des Empfängers.

IP-Header

- Der IP-Header beinhaltet nicht nur Quell- und Ziel-Adresse.
- Die Verwendung der wichtigsten Felder wird später in diesem Kapitel anhand von Beispielen genauer erläutert werden.

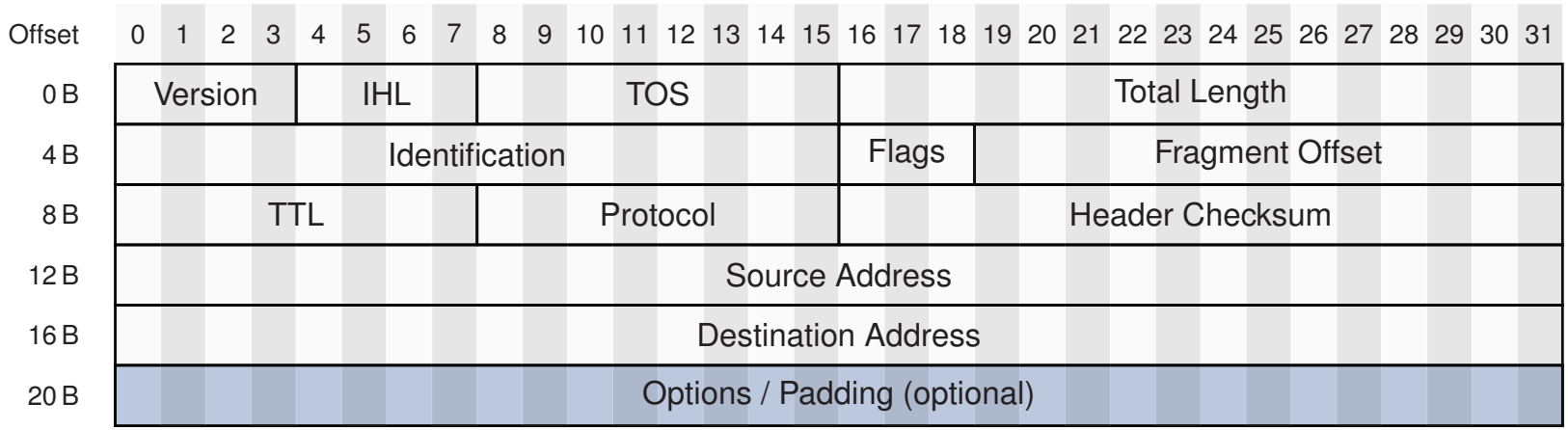


Abbildung: IPv4-Header (minimale Länge: 20 Byte)

Options / Padding

- IP unterstützt eine Reihe von Optionen (z. B. Zeitstempel, Route Recording, . . .), welche als optionale Felder an den IP-Header angefügt werden können.
- Nicht alle diese Optionen sind 4 Byte lang. Da die Länge des IP-Headers jedoch ein Vielfaches von 4 Byte betragen muss, werden kürzere Optionen ggf. durch Padding auf den nächsten gültigen Wert ergänzt.

Einschub: Host-Byte-Order und Network-Byte-Order

Es gibt zwei unterschiedle Byte-Orders:

1. Big Endian: „Niederwertigstes Byte steht an höchstwertigster Adresse“
   Intuitve Schreibweise entspricht der Reihenfolge im Speicher, z. B. die Dezimalzahl 256 in hexadezimaler Schreibweise als 0x0100.
2. Little Endian: „Niederwertigstes Byte steht an niederwertigster Adresse“
   Kontraintuitiv, da die Daten im Speicher „verkehrt herum“ abgelegt werden, z. B. die Dezimalzahl 256 in hexadezimaler Schreibweise als 0x0001.

x86-kompatible Computer verwenden intern Little Endian. Bei der Kommunikation mit anderen Computern stellt das aber möglicherweise ein Problem dar. Deswegen: Konvertierung in Network Byte Order.

Network Byte Order

Vor dem Versenden von Daten müssen Daten aus der Host Byte Order (Little Endian bei x86) in Network Byte Order (Big Endian) konvertiert werden. Analoges gilt beim Empfang von Daten.

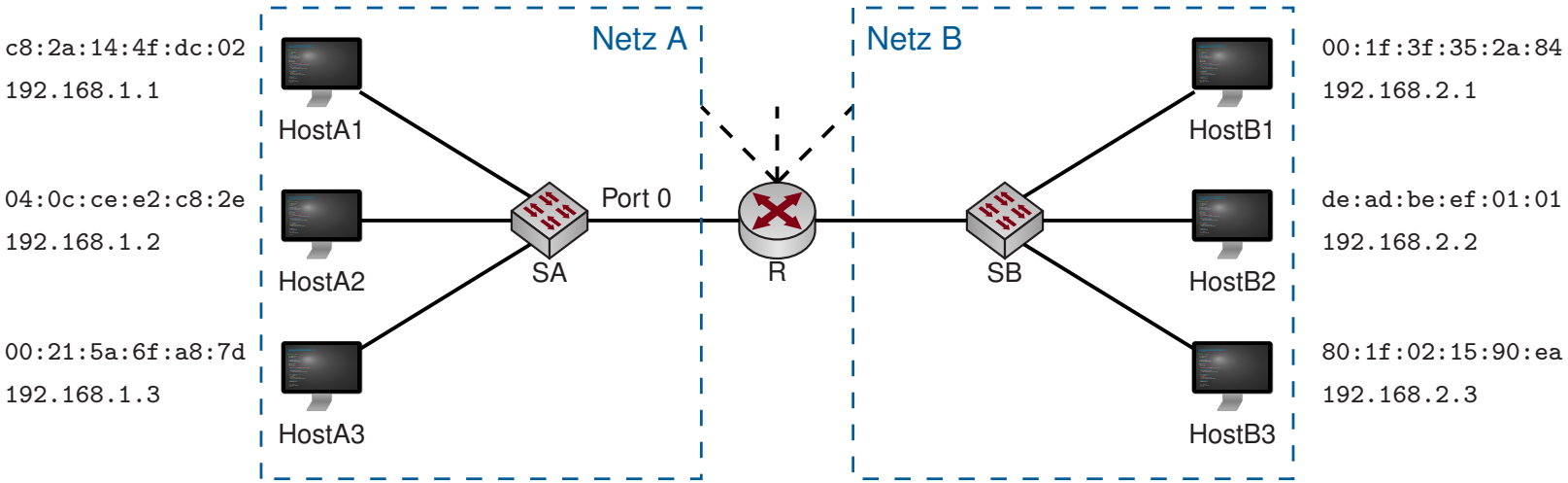
- Dies betrifft z. B. die Felder Total Length, Identification, Flags+Fragment Offset und Header Checksum aus dem IP-Header.
- Nicht betroffen sind 1 Byte lange Felder. Quell- und Zieladresse liegen gewöhnlich als Array von 1 Byte langen Werten vor, weswegen auch diese kein Problem darstellen.

Die Konvertierung zwischen beiden Formaten erfolgt für 16 Bit lange Werte in C beispielsweise durch die Funktionen `htons()` und `ntohs()`:

- `htons(0x0001) = 0x0100` „Host to Network Short“
- `ntohs(0x0100) = 0x0001` „Network to Host Short“

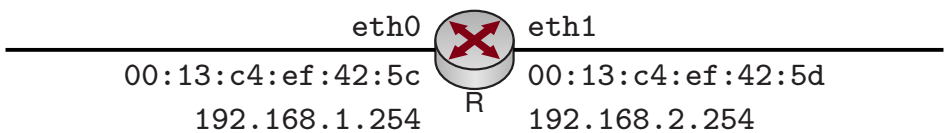
Für 32 Bit lange Werte gibt es analog `htonl()` und `ntohl()` („l“ für Long).

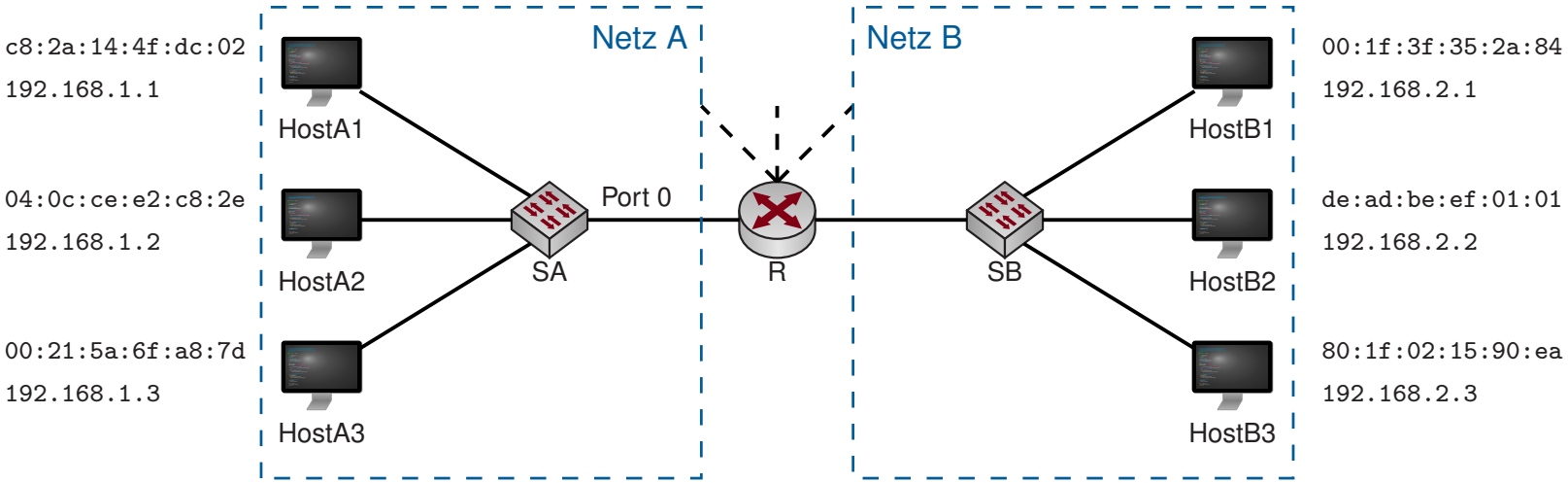
Zurück zu unserem Beispiel:



- Innerhalb von Netz A erreichen sich alle Hosts weiterhin direkt über SA.
- Will HostA1 nun ein Paket an HostB2 schicken, so geht das nicht mehr direkt:
  - Die Weiterleitung von A nach B erfolgt über R **auf Basis von IP-Adressen**
  - HostA1 muss also dafür sorgen, dass R das Paket erhält
  - Dazu trägt HostA1 als Ziel-MAC-Adresse die Adresse des lokalen Interfaces von R ein
  - Als Ziel-IP-Adresse wird die IP-Adresse von HostB2 verwendet

$\Rightarrow$ R muss adressierbar sein und verfügt daher auf jedem Interface über eine eigene MAC-Adresse und (wie wir gleich sehen werden) auch über eine eigene IP-Adresse:



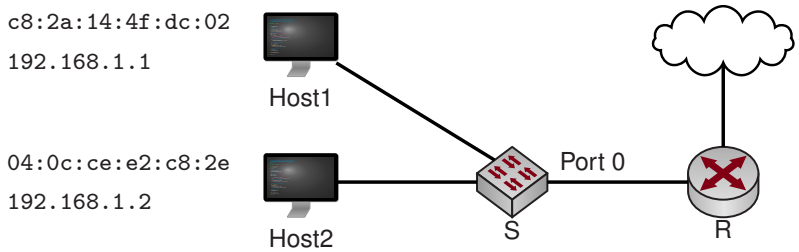


Offene Probleme:

- Angenommen der Sender kennt nur die IP-Adresse des Ziels. Wie kann die MAC-Adresse des Next-Hops bestimmt werden? (Adressauflösung)
- Woher weiß HostA1, dass er Netz B über R erreichen kann? (Routing Table)
- Angenommen das Ziel eines Pakets befindet sich nicht in einem direkt an R angeschlossenen Netz. Woher kennt R die richtige Richtung? (Routing Table)
- Wie wird diese „Routing Table“ erzeugt? (statisches Routing, Routing Protokolle)
- Was ist, wenn R mehrere Wege zu einem Ziel kennt? (Longest Prefix Matching)
- Wie funktioniert die Unterteilung einer IP-Adresse in Netz- und Hostanteil? (Classfull Routing, Classless Routing, Subnetting)
- Woher kennt der Sender überhaupt die IP des Ziels? (DNS $\rightarrow$ Schicht 7)

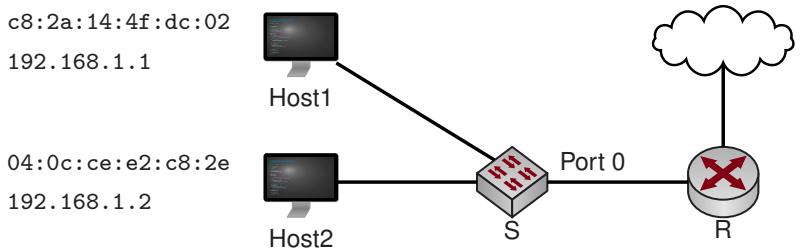
Adressauflösung [4]

- Host1 will eine Nachricht an Host2 senden
- Die IP-Adresse von Host2 (192.168.1.2) sei ihm bereits bekannt
- Wie erhält Host1 die zugehörige MAC-Adresse?



Adressauflösung [4]

- Host1 will eine Nachricht an Host2 senden
- Die IP-Adresse von Host2 (192.168.1.2) sei ihm bereits bekannt
- Wie erhält Host1 die zugehörige MAC-Adresse?



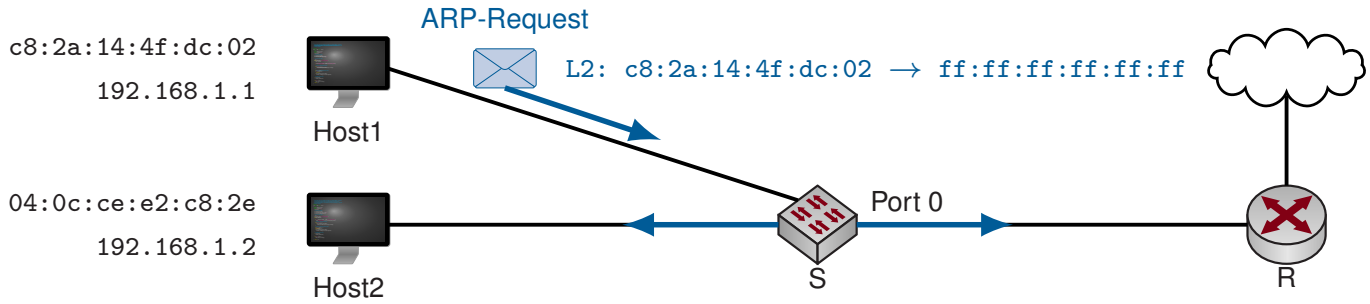
Address Resolution Protocol (ARP)

1. Host1 sendet einen ARP Request: „Who has 192.168.1.2? Tell 192.168.1.1“
2. Host2 antwort mit einem ARP Reply: „192.168.1.2 is at `04:0c:ce:e2:c8:2e`“

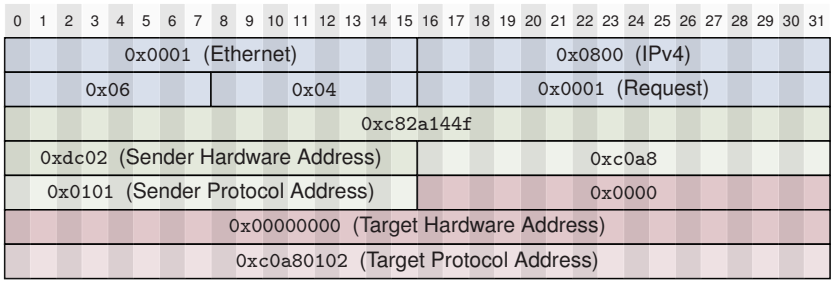
Offset	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 B	Hardware Type	Protocol Type
4 B	Hardware Addr. Length	Protocol Addr. Length	Operation
8 B	Sender Hardware Address (first 32 Bit)	
12 B	Sender Hardware Address (last 16 Bit)	Sender Protocol Address (first 16 Bit)
16 B	Sender Protocol Address (last 16 Bit)	Target Hardware Address (first 16 Bit)
20 B	Target Hardware Address (last 32 Bit)	
24 B	Target Protocol Address	

Abbildung: ARP-Paket für IPv4 über Ethernet

Beispiel:



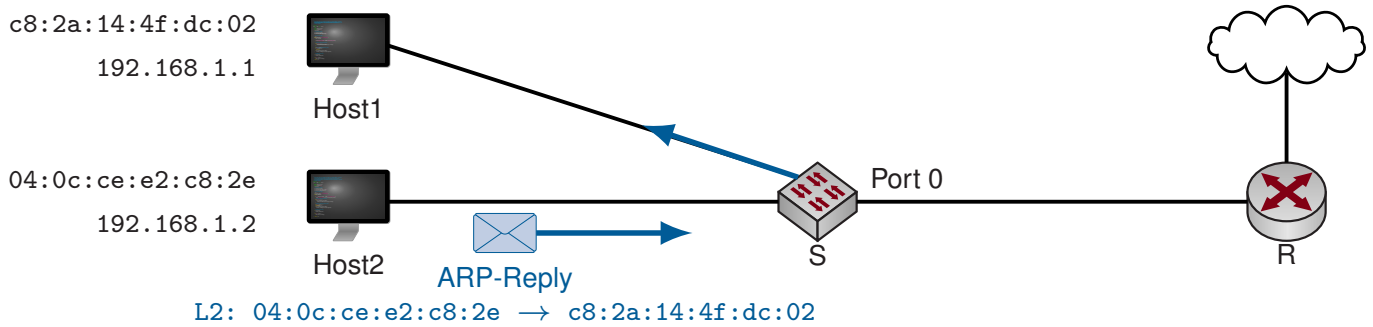
Hinweis: `L2: xx:xx:xx:xx:xx:xx → yy:yy:yy:yy:yy:yy` stellt Absender- und Ziel-MAC auf Schicht 2 dar.



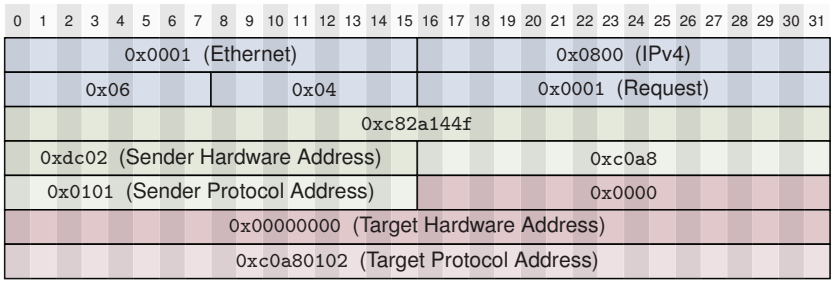
(a) ARP Request

- Der ARP-Request wird an die MAC-Broadcastadresse `ff:ff:ff:ff:ff:ff` geschickt, weswegen der Switch S den Rahmen an alle angeschlossenen Hosts weiterleitet.

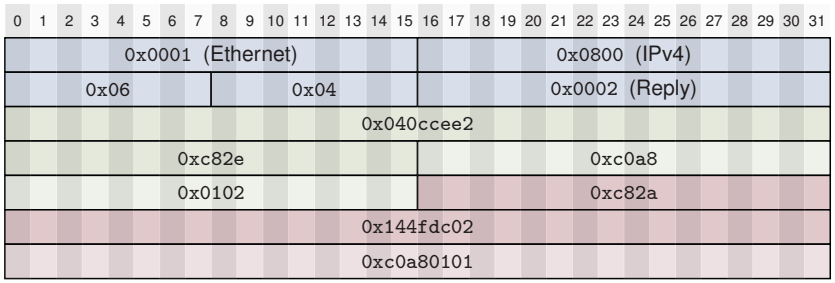
Beispiel:



Hinweis: `L2: xx:xx:xx:xx:xx:xx → yy:yy:yy:yy:yy:yy` stellt Absender- und Ziel-MAC auf Schicht 2 dar.



(c) ARP Request

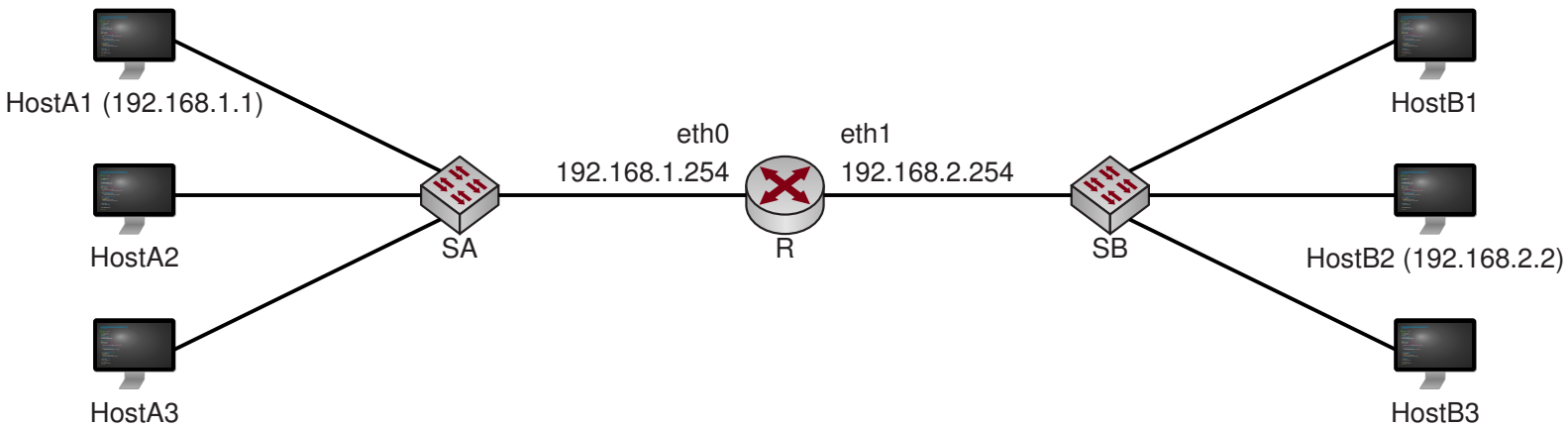


(d) ARP Reply

- Der ARP-Request wird an die MAC-Broadcastadresse `ff:ff:ff:ff:ff:ff` geschickt, weswegen der Switch S den Rahmen an alle angeschlossenen Hosts weiterleitet.
- Der ARP-Reply wird als MAC-Unicast versendet (adressiert an Host1).
- Die Rollen Sender / Target sind zwischen Request und Reply vertauscht (vgl. Inhalte der grünen und roten Felder).

Was ist nun, wenn das Ziel nicht im selben Netz liegt (z. B. HostA1 an HostB2)?

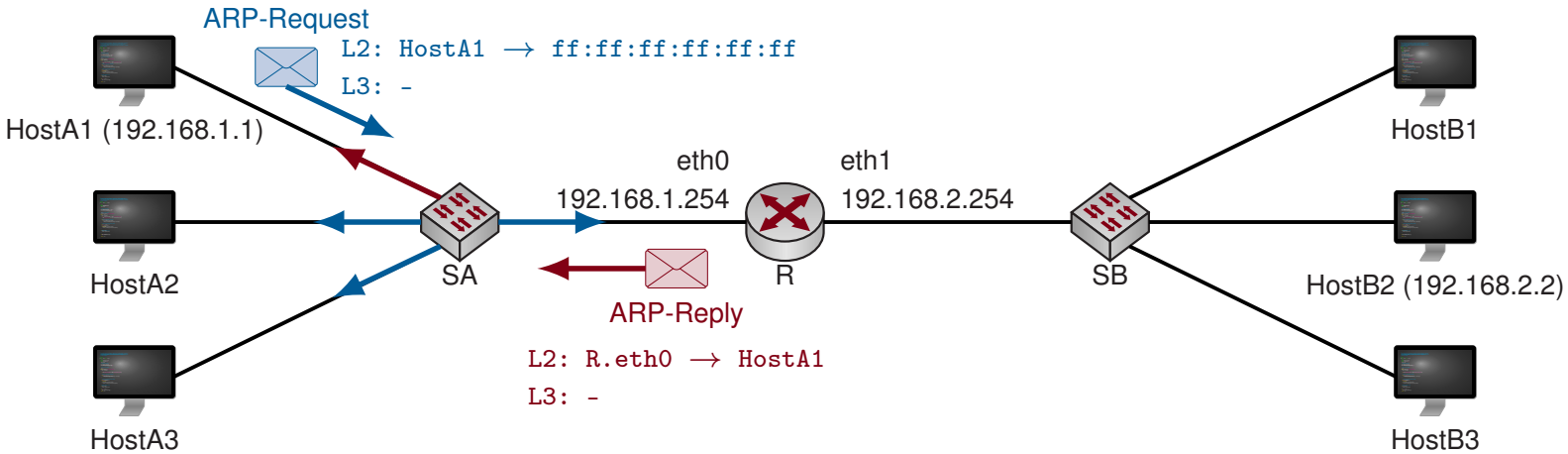
- Jeder Host sollte ein Default Gateway kennen, an das er alle Pakete schickt, deren Zieladressen nicht im eigenen Netz liegen.
- Die Zugehörigkeit einer Adresse zum jeweiligen Netz erkennt ein Host durch Vergleich der eigenen Adresse mit der Zieladdresse.
- Im Moment gehen wir noch davon aus, dass die ersten 3 Oktette einer IP-Adresse das Netz identifizieren
  $\Rightarrow$ 192.168.1.1 und 192.168.2.2 liegen in unterschiedlichen Netzen.



1 HostA1 erkennt, dass 192.168.2.2 nicht im eigenen Netz liegt. Sein Default-Gateway ist 192.168.1.254.

Was ist nun, wenn das Ziel nicht im selben Netz liegt (z. B. HostA1 an HostB2)?

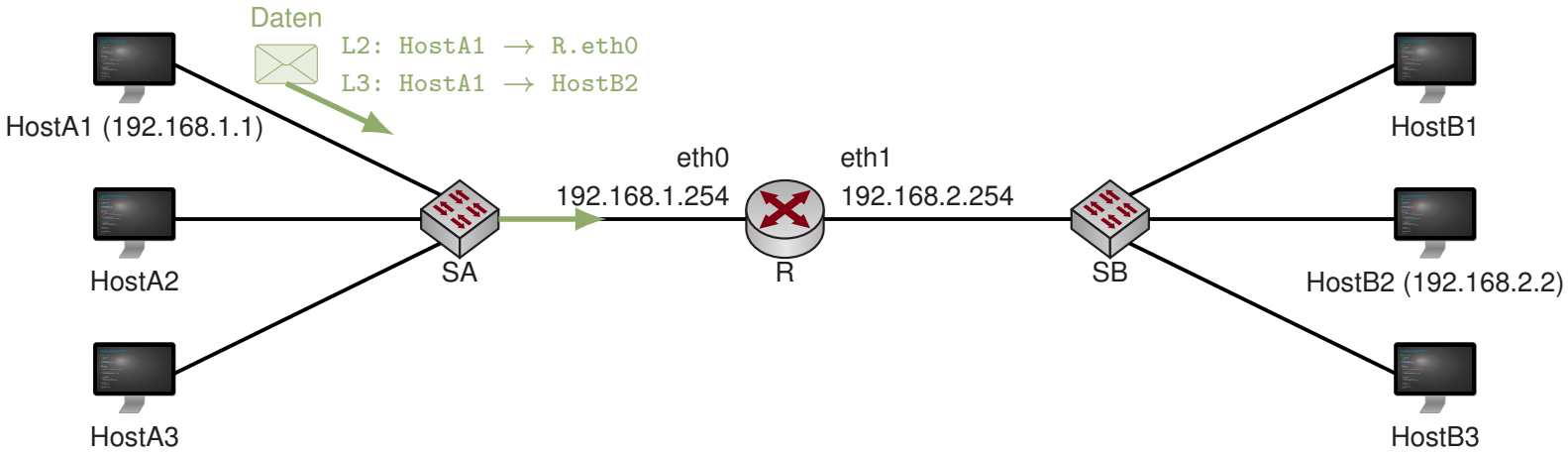
- Jeder Host sollte ein Default Gateway kennen, an das er alle Pakete schickt, deren Zieladressen nicht im eigenen Netz liegen.
- Die Zugehörigkeit einer Adresse zum jeweiligen Netz erkennt ein Host durch Vergleich der eigenen Adresse mit der Zieladdresse.
- Im Moment gehen wir noch davon aus, dass die ersten 3 Oktette einer IP-Adresse das Netz identifizieren
  $\Rightarrow$ 192.168.1.1 und 192.168.2.2 liegen in unterschiedlichen Netzen.



1. HostA1 erkennt, dass 192.168.2.2 nicht im eigenen Netz liegt. Sein Default-Gateway ist 192.168.1.254.
2. HostA1 löst die MAC-Adresse zu 192.168.1.254 auf.

Was ist nun, wenn das Ziel nicht im selben Netz liegt (z. B. HostA1 an HostB2)?

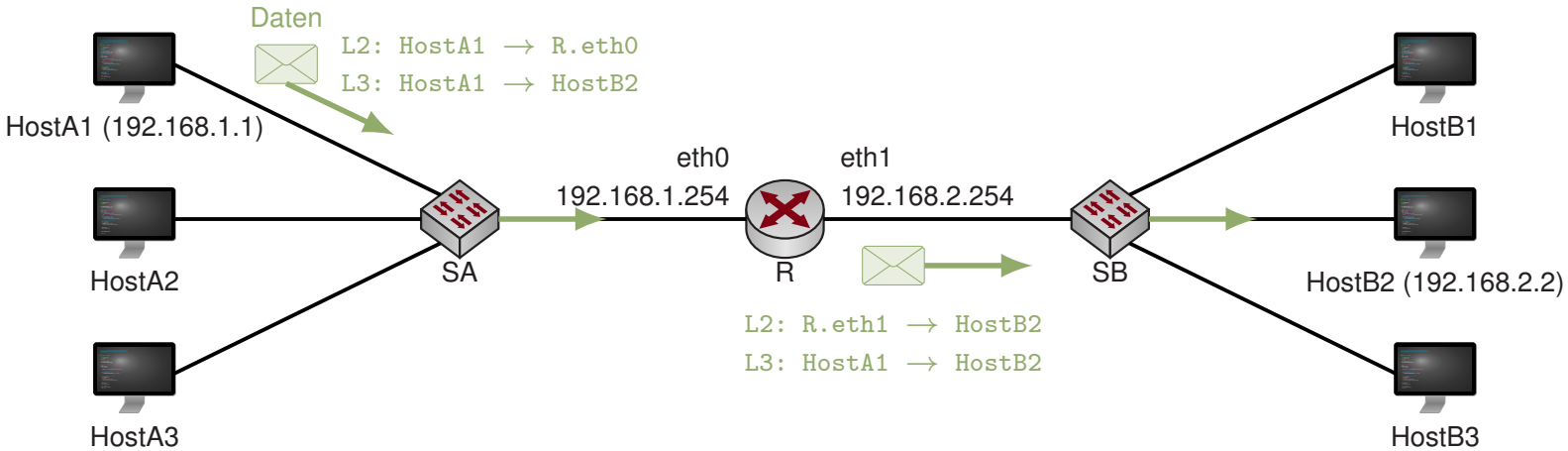
- Jeder Host sollte ein Default Gateway kennen, an das er alle Pakete schickt, deren Zieladressen nicht im eigenen Netz liegen.
- Die Zugehörigkeit einer Adresse zum jeweiligen Netz erkennt ein Host durch Vergleich der eigenen Adresse mit der Zieladdresse.
- Im Moment gehen wir noch davon aus, dass die ersten 3 Oktette einer IP-Adresse das Netz identifizieren
  $\Rightarrow$ 192.168.1.1 und 192.168.2.2 liegen in unterschiedlichen Netzen.



1. HostA1 erkennt, dass 192.168.2.2 nicht im eigenen Netz liegt. Sein Default-Gateway ist 192.168.1.254.
2. HostA1 löst die MAC-Adresse zu 192.168.1.254 auf.
3. HostA1 sendet das Datenpaket an R: Dabei adressiert er R mittels der eben bestimmten MAC-Adresse (Schicht 2). Als Ziel-IP-Adresse (Schicht 3) verwendet er die IP-Adresse von HostB2.

Was ist nun, wenn das Ziel nicht im selben Netz liegt (z. B. HostA1 an HostB2)?

- Jeder Host sollte ein Default Gateway kennen, an das er alle Pakete schickt, deren Zieladressen nicht im eigenen Netz liegen.
- Die Zugehörigkeit einer Adresse zum jeweiligen Netz erkennt ein Host durch Vergleich der eigenen Adresse mit der Zieladdresse.
- Im Moment gehen wir noch davon aus, dass die ersten 3 Oktette einer IP-Adresse das Netz identifizieren
  $\Rightarrow$ 192.168.1.1 und 192.168.2.2 liegen in unterschiedlichen Netzen.



1. HostA1 erkennt, dass 192.168.2.2 nicht im eigenen Netz liegt. Sein Default-Gateway ist 192.168.1.254.
2. HostA1 löst die MAC-Adresse zu 192.168.1.254 auf.
3. HostA1 sendet das Datenpaket an R: Dabei adressiert er R mittels der eben bestimmten MAC-Adresse (Schicht 2). Als Ziel-IP-Adresse (Schicht 3) verwendet er die IP-Adresse von HostB2.
4. R akzeptiert das Paket, bestimmt das ausgehende Interface und leitet das Paket weiter an HostB2. Dabei adressiert R wiederum HostB2 anhand seiner MAC-Adresse (erfordert ggf. einen weiteren ARP-Schritt).

Merke:

- MAC-Adressen dienen zur Adressierung innerhalb eines (Direktverbindungs-)Netzes und werden beim Routing verändert.
- IP-Adressen dienen der End-zu-End-Adressierung zwischen mehreren (Direktverbindungs-)Netzen und werden beim Routing nicht verändert.

Anmerkungen

- Das Ergebnis einer Adressauflösung wird i. d. R. im ARP-Cache eines Hosts zwischengespeichert, um nicht bei jedem zu versendenden Paket erneut eine Adressauflösung durchführen zu müssen.
- Die Einträge im ARP-Cache altern und werden nach einer vom Betriebssystem festgelegten Zeit invalidiert (5-10 Minuten).
- Den Inhalt des ARP-Caches kann man sich unter Linux, OS X und Windows mittels des Befehls `arp -a` anzeigen lassen.
- ARP-Replies können auch als MAC-Broadcast verschickt werden, so dass alle Hosts innerhalb einer Broadcast-Domain den Reply erhalten. Abhängig vom Betriebssystem werden derartige „unaufgeforderten ARP-Replies“ (engl. unsolicited ARP replies) häufig ebenfalls im ARP-Cache gespeichert.

Fragen: Was würde passieren, wenn . . .

- zwei Hosts innerhalb derselben Broadcast-Domain identische MAC-Adressen aber unterschiedliche IP-Adressen haben?
- ein Host absichtlich auf ARP-Requests antwortet, die nicht an ihn gerichtet waren?
- ein Host unsinnige ARP-Replies via MAC-Broadcasts verschickt?

Internet Control Message Protocol (ICMP) [5]

IP dient der logischen Adressierung von Knoten. Dabei kann es zu Fehlern kommen, z. B.

- ein Paket wird zu oft weitergeleitet (evtl. durch eine Routing-Schleife),
- ein Router kennt keinen Weg zum Zielnetz,
- der letzte Router zum Ziel kann die MAC-Adresse des Empfängers nicht auflösen . . .

Das Internet Control Message Protocol (ICMP) dient dazu,

- in derartigen Fällen den Absender über das Problem zu benachrichtigen und
- stellt zusätzlich Möglichkeiten bereit, um z. B.
  - die Erreichbarkeit von Hosts zu prüfen („Ping“) oder
  - Pakete umzuleiten (Redirect).

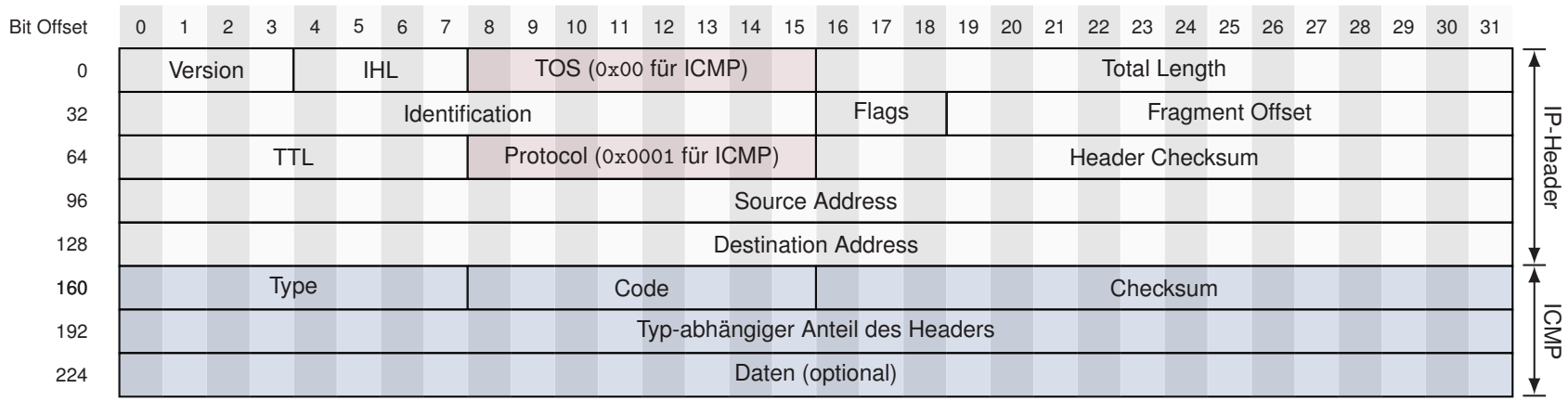
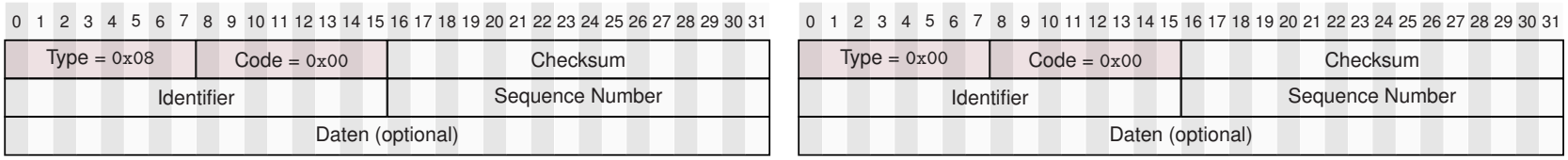


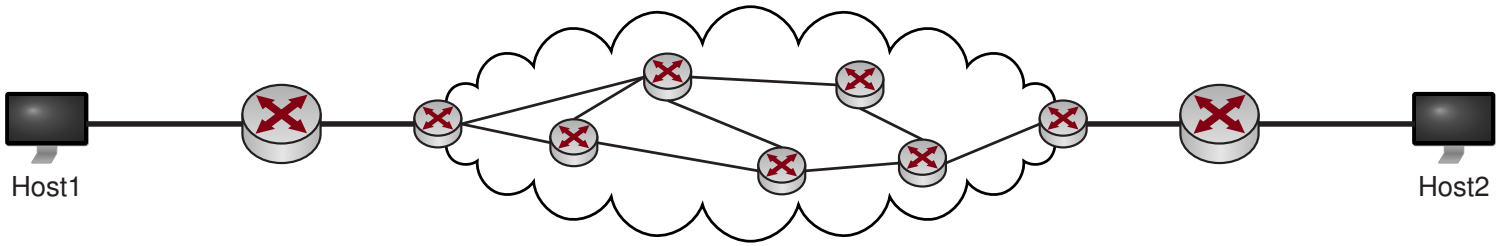
Abbildung: Allgemeiner ICMP-Header mit vorangestelltem IP-Header

ICMP Echo Request / Echo Reply



(a) ICMP Echo Request

(b) ICMP Echo Reply

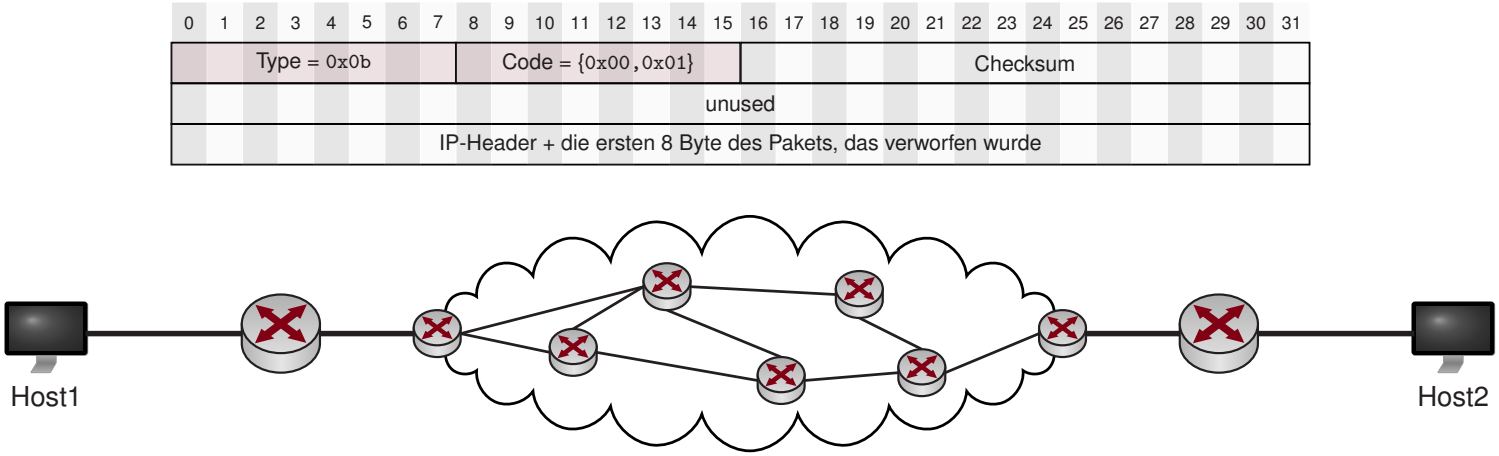


„Ping“ von Host1 nach Host2:

- Host1 wählt einen zufälligen Identifier (16 Bit Wert), die Sequenznummer wird für jeden gesendeten Echo-Request um eins inkrementiert.
- Der Echo Request wird von Routern wie jedes IP-Paket weitergeleitet.
- Erhält Host2 den Echo Request, so antwortet er mit einem Echo Reply. Dabei werden Identifier, Sequenznummer und Daten aus dem Request kopiert und zurückgeschickt.
- Sollte die Weiterleitung zu Host2 fehlschlagen, so wird eine ICMP-Nachricht mit dem entsprechenden Fehlercode an Host1 zurückgeschickt.

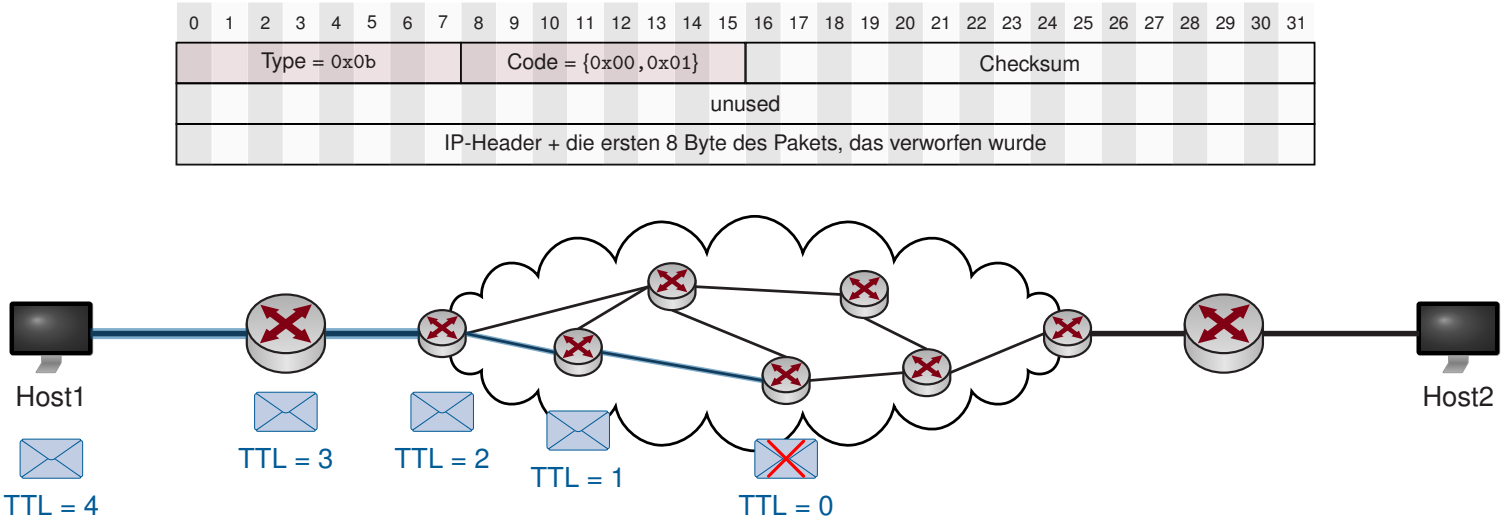
Frage: Wozu dient der Identifier?

ICMP Time Exceeded



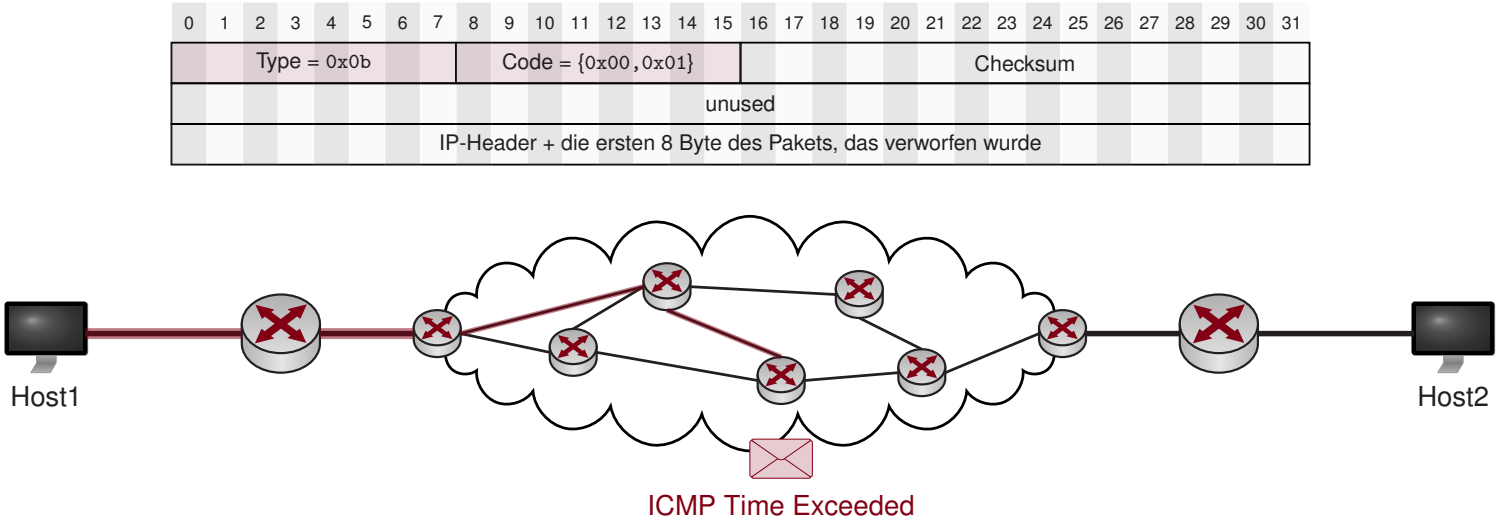
- Der IP-Header besitzt ein TTL-Feld, welches bei der Weiterleitung eines Pakets durch den jeweiligen Router um eins dekrementiert wird.

ICMP Time Exceeded



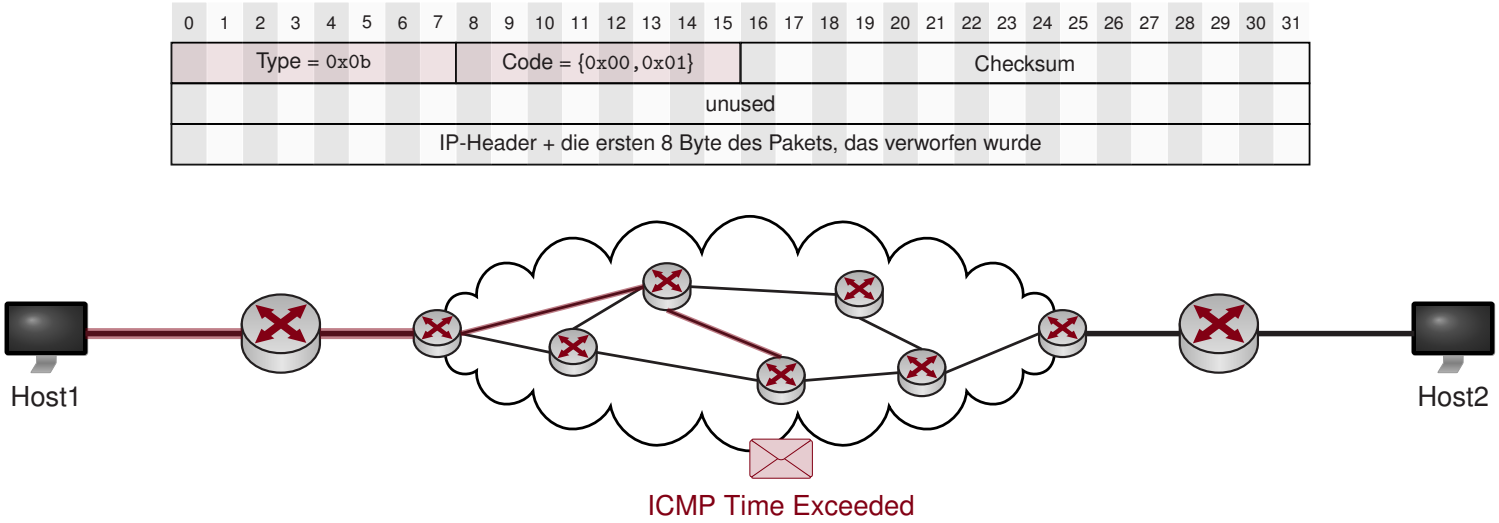
- Der IP-Header besitzt ein TTL-Feld, welches bei der Weiterleitung eines Pakets durch den jeweiligen Router um eins dekrementiert wird.
- Erreicht es den Wert 0, so wird das betreffende Paket verworfen.

ICMP Time Exceeded



- Der IP-Header besitzt ein TTL-Feld, welches bei der Weiterleitung eines Pakets durch den jeweiligen Router um eins dekrementiert wird.
- Erreicht es den Wert 0, so wird das betreffende Paket verworfen.
- Der Router generiert ein ICMP Time Exceeded und schickt es an den Absender des verworfenen Pakets zurück.

ICMP Time Exceeded

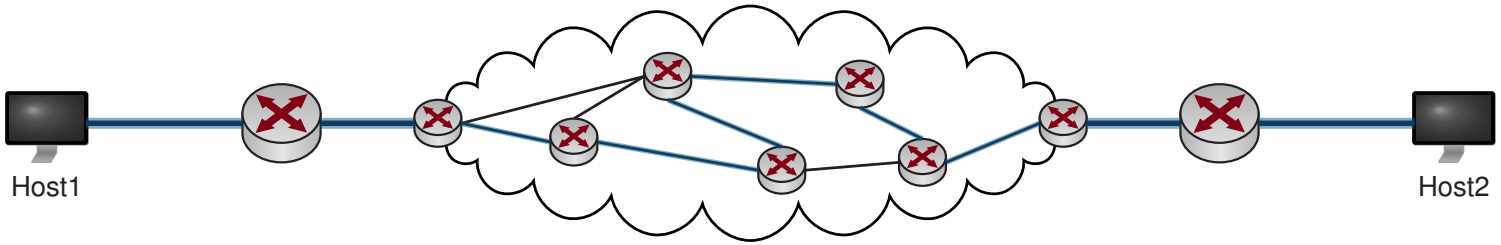


- Der IP-Header besitzt ein TTL-Feld, welches bei der Weiterleitung eines Pakets durch den jeweiligen Router um eins dekrementiert wird.
- Erreicht es den Wert 0, so wird das betreffende Paket verworfen.
- Der Router generiert ein ICMP Time Exceeded und schickt es an den Absender des verworfenen Pakets zurück.

Da der Header und die ersten 8 Byte des verworfenen Pakets an den Absender zurückgeschickt werden, kann dieser genau bestimmen, welches Paket verworfen wurde.

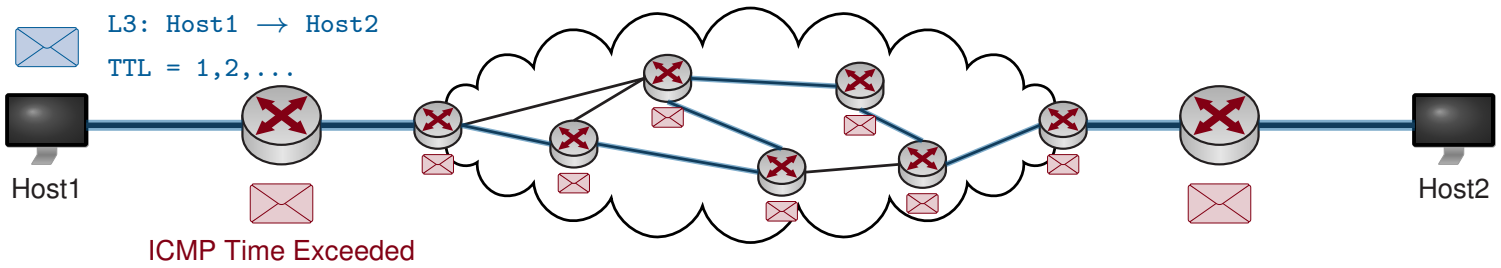
Frage: Welche Informationen sind in diesen ersten 8 Byte enthalten, wenn das verworfene Paket ein ICMP Echo Request war?

ICMP Traceroute



- Obwohl Pakete zwischen Host1 und Host2 (in Hin- und Rückrichtung) jeweils unterschiedliche Wege nehmen können, werden in der Praxis meist nur einer oder sehr wenige genutzt.
- Welchen Pfad nehmen Pakete von Host1 nach Host2?

ICMP Traceroute



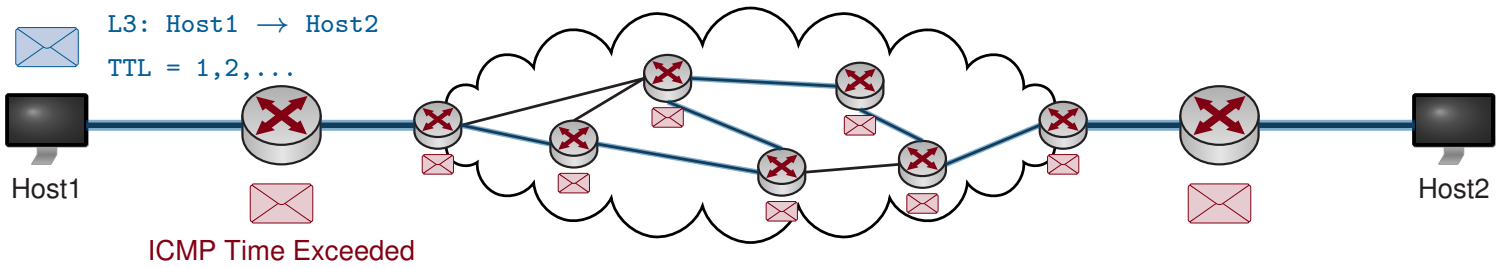
- Obwohl Pakete zwischen Host1 und Host2 (in Hin- und Rückrichtung) jeweils unterschiedliche Wege nehmen können, werden in der Praxis meist nur einer oder sehr wenige genutzt.
- Welchen Pfad nehmen Pakete von Host1 nach Host2?

Traceroute: Host1 sendet z. B. ICMP Echo Requests an Host2

- wobei das TTL-Feld zu Beginn auf 1 gesetzt wird und
- danach schrittweise um jeweils 1 erhöht wird.

$\Rightarrow$ Router entlang des Pfads von Host1 nach Host2 werden schrittweise die Pakete verwerfen und jeweils ein TTL Exceeded an Host1 zurücksenden. Anhand der IP-Quelladresse dieser Fehlernachrichten kann Host1 den Pfad hin zu Host2 nachvollziehen.

ICMP Traceroute



- Obwohl Pakete zwischen Host1 und Host2 (in Hin- und Rückrichtung) jeweils unterschiedliche Wege nehmen können, werden in der Praxis meist nur einer oder sehr wenige genutzt.
- Welchen Pfad nehmen Pakete von Host1 nach Host2?

Traceroute: Host1 sendet z. B. ICMP Echo Requests an Host2

- wobei das TTL-Feld zu Beginn auf 1 gesetzt wird und
- danach schrittweise um jeweils 1 erhöht wird.

$\Rightarrow$ Router entlang des Pfads von Host1 nach Host2 werden schrittweise die Pakete verwerfen und jeweils ein TTL Exceeded an Host1 zurücksenden. Anhand der IP-Quelladresse dieser Fehlernachrichten kann Host1 den Pfad hin zu Host2 nachvollziehen.

$\Rightarrow$ 2. Programmieraufgabe: „Implementiere Traceroute“

Beispiel:

```
slacky: moepi$ traceroute -n www.net.in.tum.de
traceroute to www.net.in.tum.de (131.159.15.49), 64 hops max, 52 byte packets
 1  192.168.1.1  2.570 ms  1.808 ms  2.396 ms
 2  82.135.16.28  15.036 ms  14.359 ms  13.760 ms
 3  212.18.7.189  14.118 ms  13.801 ms  13.845 ms
 4  82.135.16.102  20.062 ms  20.137 ms  20.251 ms
 5  80.81.192.222  22.707 ms  23.049 ms  23.215 ms
 6  188.1.144.142  31.068 ms  36.542 ms  30.823 ms
 7  188.1.37.90  30.815 ms  30.671 ms  30.808 ms
 8  129.187.0.150  30.272 ms  30.602 ms  30.845 ms
 9  131.159.252.1  30.885 ms  30.551 ms  30.992 ms
10  131.159.252.150  30.886 ms  30.955 ms  30.621 ms
11  131.159.252.150  30.578 ms  30.699 ms *
```

Fragen / Probleme:

- Ein Router hat mehrere IP-Adressen. Welche wählt er als Absender-Adresse der Fehlerbenachrichtigungen? Wählt er immer dieselbe Adresse?
- Was ist, wenn es tatsächlich mehrere gleichzeitig genutzte Pfade oder Pfadabschnitte gibt?
- Müssen Router überhaupt Fehlerbenachrichtigungen versenden?
- Angenommen der Pfade von $A \rightarrow B$ ist symmetrisch. Warum werden sich die Ausgaben von Traceroute dennoch unterscheiden?

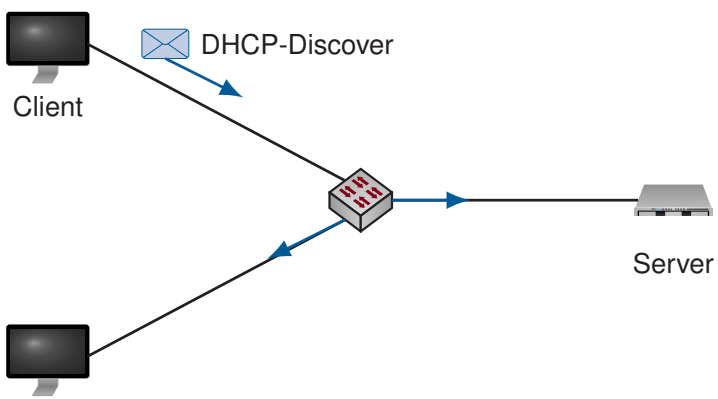
Dynamic Host Configuration Protocol (DHCP) [5]

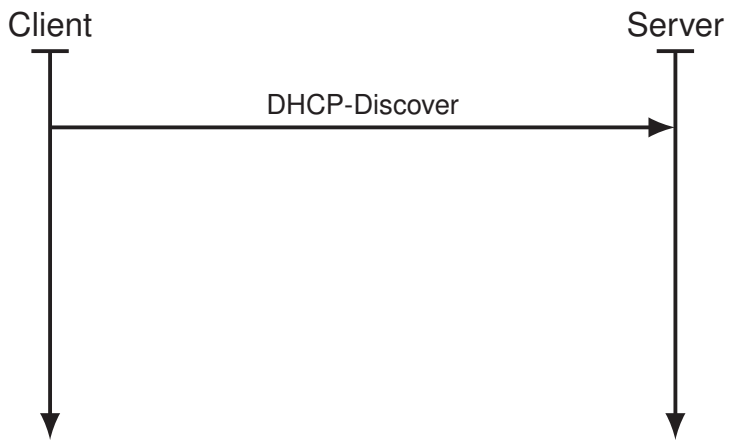
Woher bekommen Hosts eigentlich ihre IP-Adresse?

- Statische Konfiguration von Hand
- Dynamisch von einem DHCP-Server zugewiesene IP-Adresse

Ablauf:

1. Client sendet DHCP-Discover (Layer 2 Broadcast)





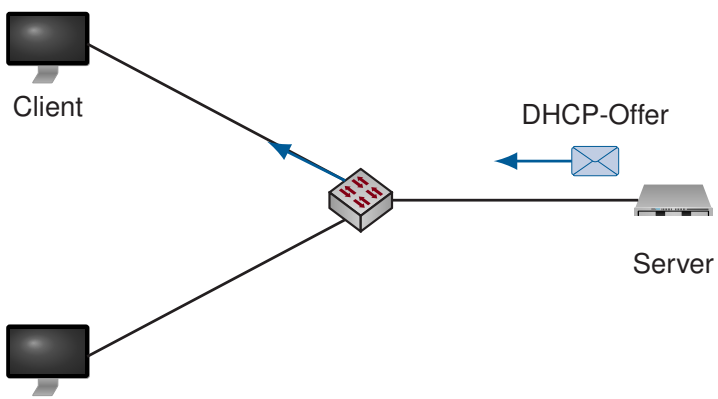
Dynamic Host Configuration Protocol (DHCP) [5]

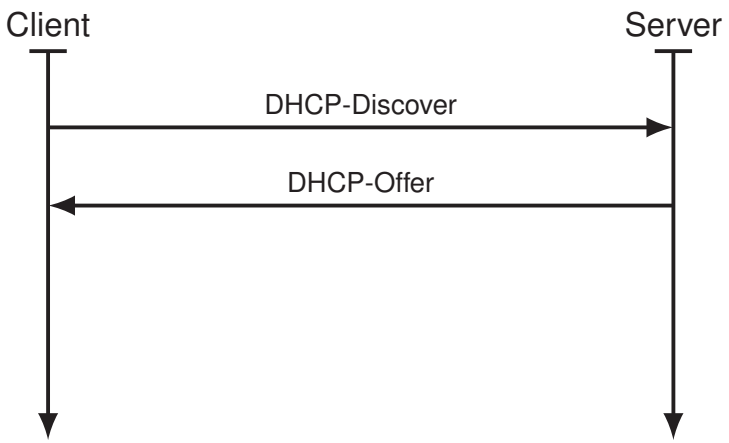
Woher bekommen Hosts eigentlich ihre IP-Adresse?

- Statische Konfiguration von Hand
- Dynamisch von einem DHCP-Server zugewiesene IP-Adresse

Ablauf:

1. Client sendet DHCP-Discover (Layer 2 Broadcast)
2. DHCP-Server antwortet mit DHCP-Offer, wodurch er dem Client eine IP-Adresse anbietet





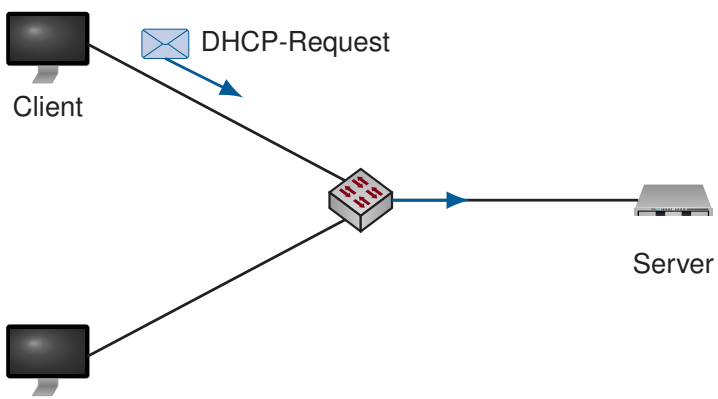
Dynamic Host Configuration Protocol (DHCP) [5]

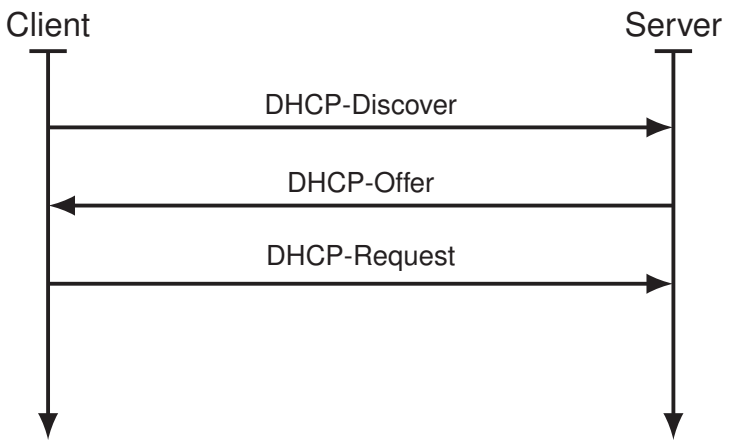
Woher bekommen Hosts eigentlich ihre IP-Adresse?

- Statische Konfiguration von Hand
- Dynamisch von einem DHCP-Server zugewiesene IP-Adresse

Ablauf:

1. Client sendet DHCP-Discover (Layer 2 Broadcast)
2. DHCP-Server antwortet mit DHCP-Offer, wodurch er dem Client eine IP-Adresse anbietet
3. Client antwortet mit DHCP-Request, wodurch er die angebotene Adresse anfordert





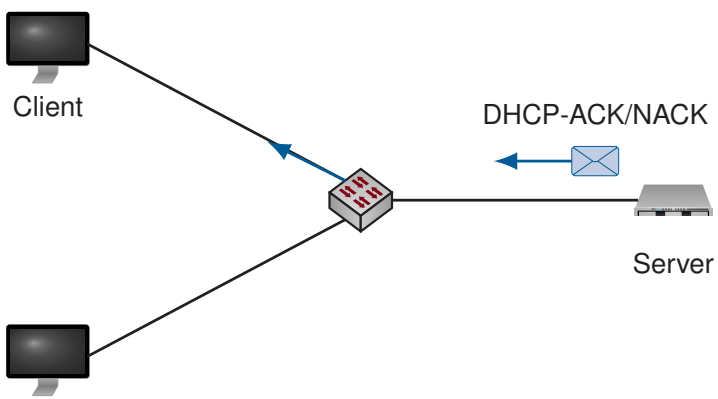
Dynamic Host Configuration Protocol (DHCP) [5]

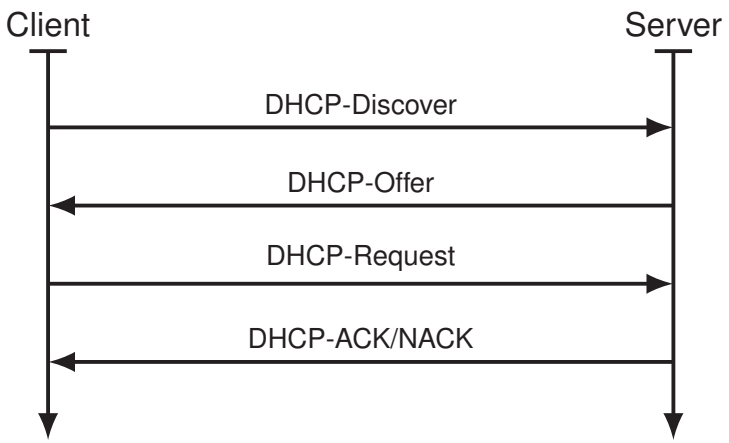
Woher bekommen Hosts eigentlich ihre IP-Adresse?

- Statische Konfiguration von Hand
- Dynamisch von einem DHCP-Server zugewiesene IP-Adresse

Ablauf:

1. Client sendet DHCP-Discover (Layer 2 Broadcast)
2. DHCP-Server antwortet mit DHCP-Offer, wodurch er dem Client eine IP-Adresse anbietet
3. Client antwortet mit DHCP-Request, wodurch er die angebotene Adresse anfordert
4. DHCP-Server antwortet mit DHCP-ACK, wodurch er die angeforderte Adresse zur Nutzung freigibt, oder mit DHCP-NACK, wodurch er die Nutzung der Adresse untersagt





Anmerkungen

- Die vom DHCP-Server zugewiesene Adresse wird auch als Lease bezeichnet und ist in ihrer Gültigkeit zeitlich begrenzt
- Clients erneuern ihr Lease in regelmäßigen Abständen beim DCHP-Server.
- Gerade in kleineren (privaten) Netzwerken übernimmt häufig ein Router die Rolle des DHCP-Servers.
- Zusammen mit der IP-Adresse und Subnetzmaske können DHCP-Server viele weitere Optionen ausliefern:
  - DNS-Server zur Namensauflösung ($\rightarrow$ Kapitel 6)
  - Hostname und Domänen-Suffix ($\rightarrow$ Kapitel 6)
  - Statische Routen, insbesondere einen Default-Gateway
  - Maximum Transmission Unit
  - NTP-Server zur Zeitsynchronisation
  - ...
- Aus Redundanzgründen werden manchmal mehrere DHCP-Server pro Netzwerk verwendet, wobei
  - sich die Adressbereiche der beiden Server nicht überschneiden dürfen oder
  - zusätzliche Mechanismen zur Synchronisation der Adresspools notwendig sind.
- Ein versehentlich ins Netzwerk eingebrachter DHCP-Server (z. B. durch einen achtlos angeschlossenen Router oder Access Point) kann beträchtliche Auswirkungen auf das Netzwerk haben und ist häufig schwer zu finden:
  - Im Netz eines Lehrstuhls gab es einst einen solchen Rogue-DHCP-Server,
  - der trotz intensiver Suche nicht gefunden werden konnte.
  - Über die MAC-Adressen der vom Server stammdenden DHCP-Nachrichten stellte sich am Ende heraus, dass es sich um ein **Nokia-Smartphone** im WLAN handelte!
  - Die Anzahl der Verdächtigen hatte sich damit stark eingeschränkt (;

Adressklassen (Classful Routing)

Zu Beginn des Kapitels hatten wir in einem Beispiel beschrieben, dass

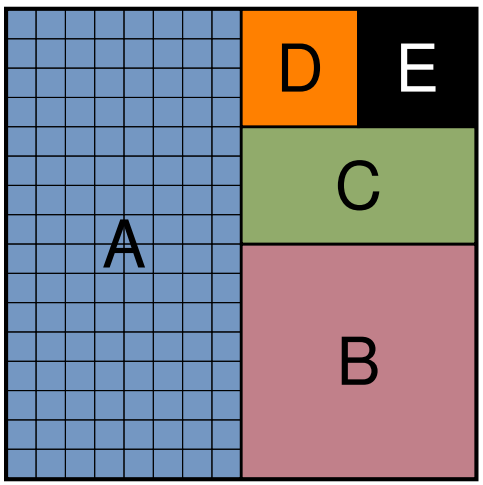
- die ersten drei Oktette einer IP-Adresse das Netz identifizieren und
- das vierte Oktett Hosts innerhalb eines Netzes adressiert.

Diese Aufteilung war lediglich ein Beispiel, entsprechend der Adress-Klasse C. Historisch ist der IP-Adresseraum in folgende fünf Klassen unterteilt:

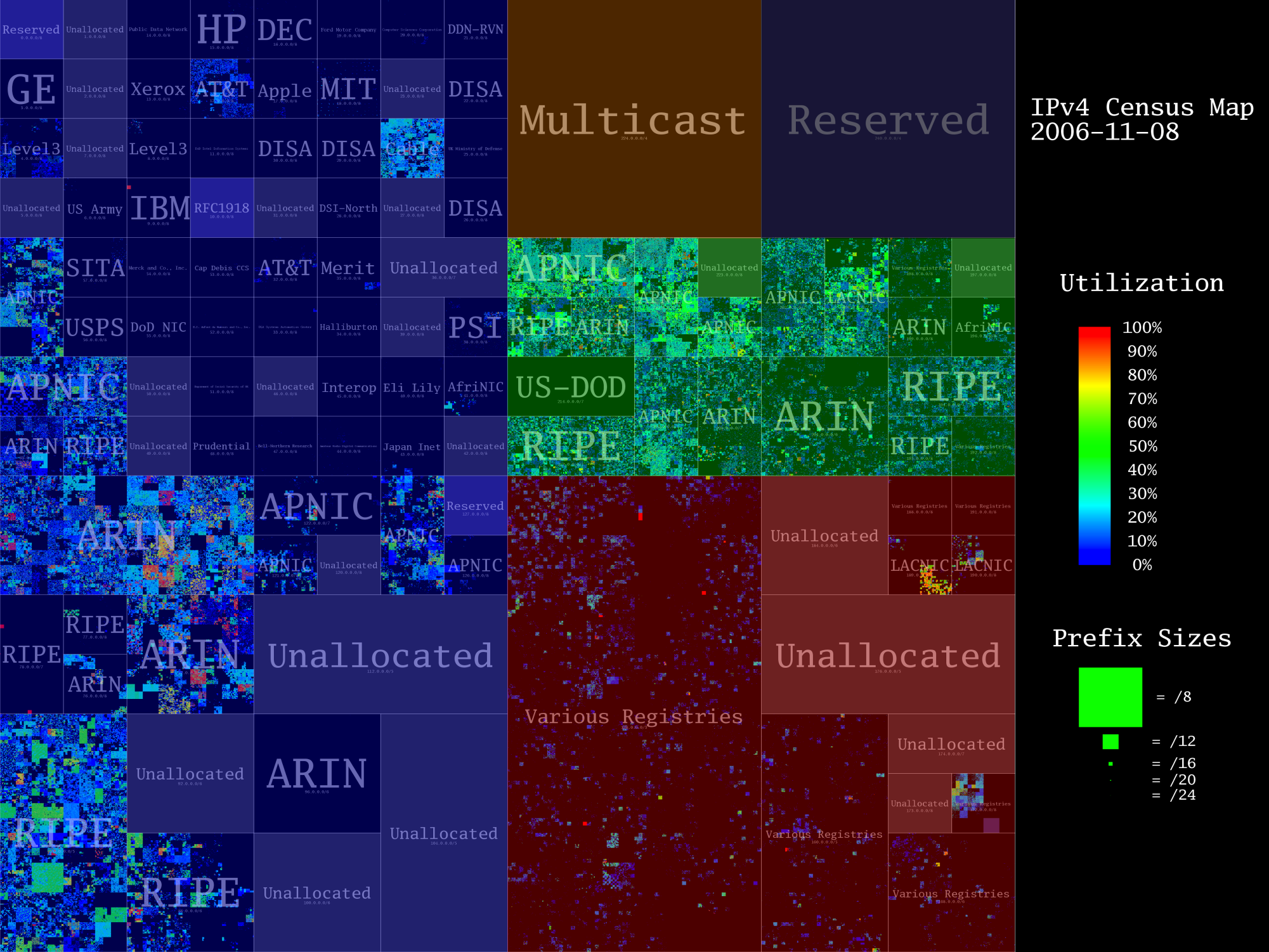
Klasse	1. Oktett	Erste Adresse	Letzte Adresse	Netz-/Hostanteil	Anzahl Netze	Adressen pro Netz
A	`0xxxxxxx`	0.0.0.0	127.255.255.255	N.H.H.H	$2^7 = 128$	$2^{24} = 16777216$
B	`10xxxxxx`	128.0.0.0	191.255.255.255	N.N.H.H	$2^{14} = 16384$	$2^{16} = 65536$
C	`110xxxxx`	192.0.0.0	223.255.255.255	N.N.N.H	$2^{21} = 2097152$	$2^8 = 256$
D	`1110xxxx`	224.0.0.0	239.255.255.255	Reserviert für Multicast		
E	`1111xxxx`	240.0.0.0	255.255.255.255	Reserviert		

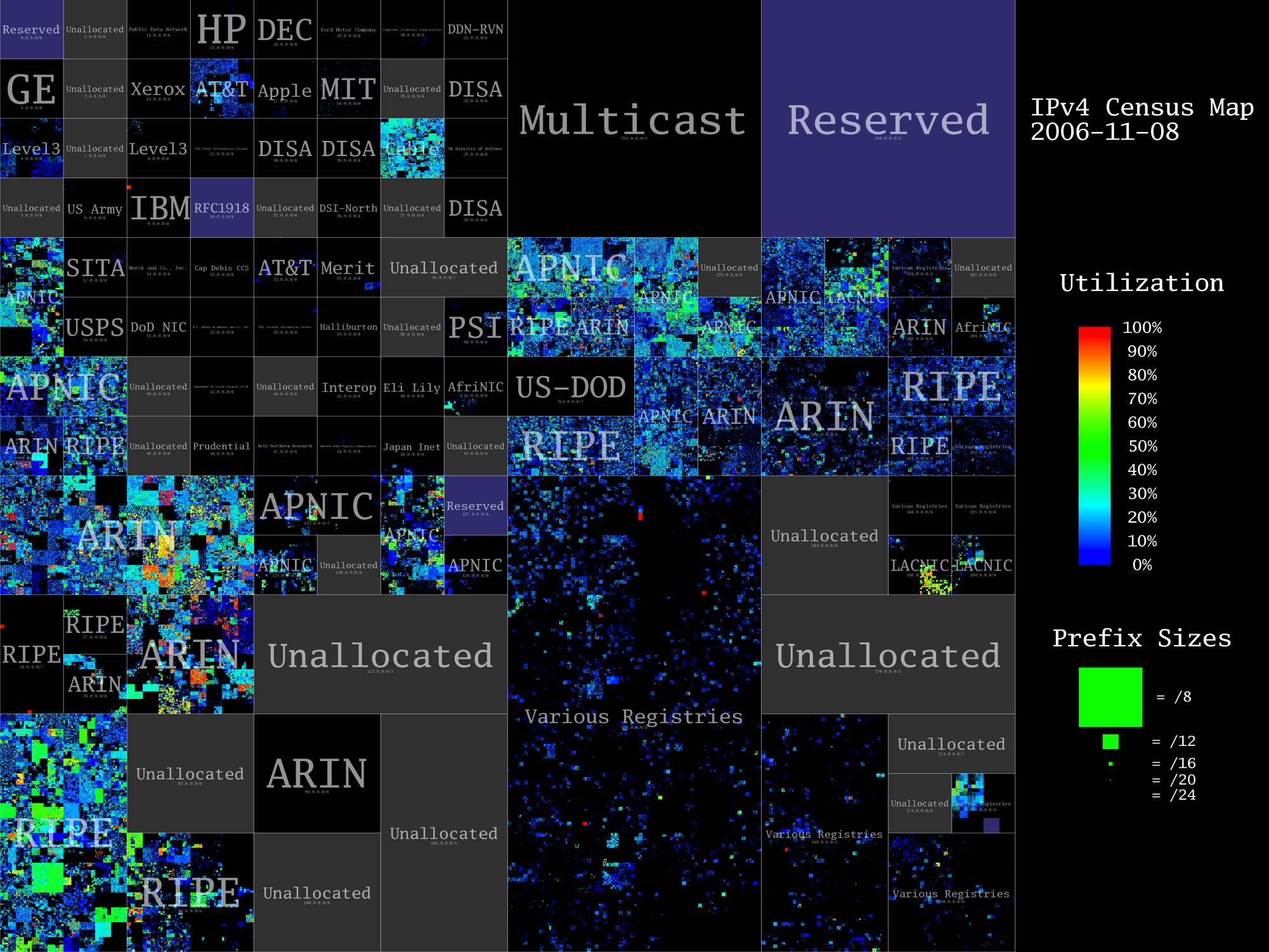
Grafische Darstellung:

- IP-Adressraum als Quadrat
- Farbige Flächen markieren die fünf Adressklassen
- Die einzelnen Klasse-A-Netze sind als Drahtgitter angedeutet



→ Nächste Folie zeigt die tatsächliche Verteilung und Ausnutzung des Adressraums [1] (selbe Farbkodierung).


IPv4 Census Map
2006-11-08
Utilization
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Prefix Sizes
= /8
= /12
= /16
= /20
= /24
Reserved
Unallocated
Public Data Network
HP
DEC
Ford Motor Company
Computer Sciences Corporation
DDN-RVN
GE
Xerox
AT&T
Apple
MIT
DISA
Level3
DoD Intel Information Systems
UK Ministry of Defense
US Army
IBM
RFC1918
DSI-North
Multicast
Reserved
SITA
Merck and Co., Inc.
Cap Debis CCS
Merit
USPS
DoD NIC
Halliburton
PSI
APNIC
Interop
Eli Lily
AfriNIC
ARIN
RIPE
Prudential
Bell-Northern Research
Japan Inet
Various Registries
LACNIC
US-DOD


IPv4 Census Map
2006-11-08
Utilization
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Prefix Sizes
= /8
= /12
= /16
= /20
= /24
Reserved
Unallocated
Public Data Network
HP
DEC
Ford Motor Company
Computer Sciences Corporation
DDN-RVN
GE
Xerox
AT&T
Apple
MIT
DISA
Multicast
Level3
DoD Intel Information Systems
UK Ministry of Defense
US Army
IBM
RFC1918
DSI-North
SITA
Merck and Co., Inc.
Cap Debis CCS
Merit
USPS
DoD NIC
Halliburton
PSI
APNIC
US-DOD
Interop
Eli Lily
AfriNIC
ARIN
RIPE
Prudential
Bell-Northern Research
Japan Inet
LACNIC
Various Registries

Verschwendung des Adressraums

- Als IPv4 1981 eingeführt wurde, konnte man sich nicht vorstellen, dass ~2^{32} Adressen aufgeteilt in die Klassen A, B und C nicht ausreichend würden.
- Es wurden große Adresseblöcke an Firmen, Behörden und Bildungseinrichungen vergeben, z. B. ganze Klasse-A Netze an HP, IBM, AT&T, Apple, MIT, Generel Electric, US Army, ...

Folge:

- Ineffiziente Aufteilung und Nutzung des Adressraums
- Große Netze mit internen Routern $\Rightarrow$ weitere Unterteilung in Subnetze notwendig

Situation heute:

- Vergabe des letzten IPv4 Adressblocks am 3.2.2011 durch die IANA[1] an eine der fünf Regional Internet Registries (RIRs), das APNIC[2]
- IPv4-Adressraum praktisch aufgebraucht
- Immernoch schleppende Einführung von IPv6

[1] Internet Assigned Numbers Authority
[2] Asia-Pacific Network Information Center (APNIC)

Subnetting (Classless Routing)

Bereits 1993 wurde mit CIDR[3] ein Verfahren zur Unterteilung von IP-Netzen eingeführt:

- Zusätzlich zur IP-Adresse erhält ein Interface eine ebenfalls 32 Bit lange Subnetzmaske
- Die Subnetzmaske unterteilt die IP-Adresse in einen Netzanteil und einen Hostanteil
- Eine logische 1 in der Subnetzmaske bedeutet Netzanteil, eine logische 0 Hostanteil
- UND-Verknüpfung von IP-Adresse und Subnetzmaske ergibt die Netzadresse
- Die übliche Klassenzugehörigkeit hat damit nur noch im Sprachgebrauch eine Bedeutung

Beispiel 1:

IP-Adresse	11000000 . 10101000 . 00000000 . 10110010	192.168.0.178
Subnetz Maske	11111111 . 11111111 . 11111111 . 00000000	255.255.255.0

- 24 Bit Netzanteil, 8 Bit Hostanteil $\Rightarrow 2^8 = 256$ Adressen
- Netzadresse: 192.168.0.0
- Broadcastadresse: 192.168.0.255
- Nutzbare Adressen für Hosts: $2^8 - 2 = 254$

[3] Classless Inter-Domain Routing

Subnetting (Classless Routing)

Bereits 1993 wurde mit CIDR[3] ein Verfahren zur Unterteilung von IP-Netzen eingeführt:

- Zusätzlich zur IP-Adresse erhält ein Interface eine ebenfalls 32 Bit lange Subnetzmaske
- Die Subnetzmaske unterteilt die IP-Adresse in einen Netzanteil und einen Hostanteil
- Eine logische 1 in der Subnetzmaske bedeutet Netzanteil, eine logische 0 Hostanteil
- UND-Verknüpfung von IP-Adresse und Subnetzmaske ergibt die Netzadresse
- Die übliche Klassenzugehörigkeit hat damit nur noch im Sprachgebrauch eine Bedeutung

Beispiel 2:

IP-Adresse	11000000 . 10101000 . 00000000 . 10110010	192.168.0.178
Subnetz Maske	11111111 . 11111111 . 11111111 . 10000000	255.255.255.128

- 25 Bit Netzanteil, 7 Bit Hostanteil $\Rightarrow 2^7 = 128$ Adressen
- Netzadresse: 192.168.0.128
- Broadcastadresse: 192.168.0.255
- Nutzbare Adressen für Hosts: $2^7 - 2 = 126$

[3] Classless Inter-Domain Routing

Subnetting (Classless Routing)

Bereits 1993 wurde mit CIDR[3] ein Verfahren zur Unterteilung von IP-Netzen eingeführt:

- Zusätzlich zur IP-Adresse erhält ein Interface eine ebenfalls 32 Bit lange Subnetzmaske
- Die Subnetzmaske unterteilt die IP-Adresse in einen Netzanteil und einen Hostanteil
- Eine logische 1 in der Subnetzmaske bedeutet Netzanteil, eine logische 0 Hostanteil
- UND-Verknüpfung von IP-Adresse und Subnetzmaske ergibt die Netzadresse
- Die übliche Klassenzugehörigkeit hat damit nur noch im Sprachgebrauch eine Bedeutung

Beispiel 3:

IP-Adresse	11000000 . 10101000 . 00000000 . 10110010	192.168.0.178
Subnetz Maske	11111111 . 11111111 . 11111111 . 11000000	255.255.255.192

- 26 Bit Netzanteil, 6 Bit Hostanteil $\Rightarrow 2^6 = 64$ Adressen
- Netzadresse: 192.168.0.128
- Broadcastadresse: 192.168.0.191
- Nutzbare Adressen für Hosts: $2^6 - 2 = 62$

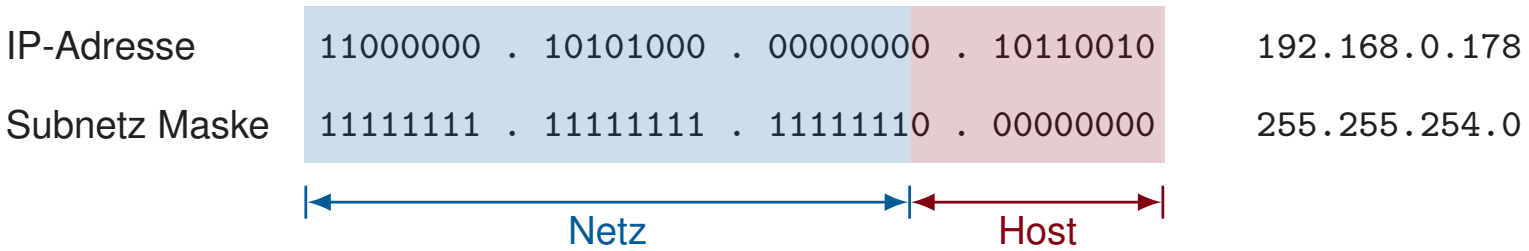
[3] Classless Inter-Domain Routing

Supernetting

Dasselbe Prinzip funktioniert auch in die andere Richtung:

- Zusammenfassung mehrerer zusammenhängender kleinerer Netze zu einem größeren Netz
- Wird häufig von Routern angewendet, um die Anzahl der Einträge in der Routingtabelle zu reduzieren

Beispiel:



- 23 Bit Netzanteil, 9 Bit Hostanteil $\Rightarrow 2^9 = 512$ Adressen
- Netzadresse: 192.168.0.0
- Broadcastadresse: 192.168.1.255
- Nutzbare Adressen für Hosts: $2^9 - 2 = 510$

Präfixschreibweise:

Anstelle die Subnetzmaske auszuschreiben, wird häufig nur die Länge des Netzanteils (Anzahl führender Einsen in der Subnetzmaske) angegeben, z. B. 192.168.0.0/23.

Frage: Können die beiden Netze 192.168.1.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Frage: Können die beiden Netze 192.168.1.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn es gibt keine passende Subnetzmaske:

- 192.168.0.0/23 enthält die Netze 192.168.{0,1}.0/24
- 192.168.2.0/23 enthält die Netze 192.168.{2,3}.0/24
- 192.168.1.0/23 ist keine gültige Netzadresse, denn UND-Verknüpfung der Adresse 192.168.1.0 mit der Subnetzmaske 255.255.254.0 ergibt 192.168.0.0 als Netzadresse!
  $\Rightarrow$ 192.168.1.0 ist eine Hostadresse im Netz 192.168.0.0/23.

Frage: Können die beiden Netze 192.168.1.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn es gibt keine passende Subnetzmaske:

- 192.168.0.0/23 enthält die Netze 192.168.{0,1}.0/24
- 192.168.2.0/23 enthält die Netze 192.168.{2,3}.0/24
- 192.168.1.0/23 ist keine gültige Netzadresse, denn UND-Verknüpfung der Adresse 192.168.1.0 mit der Subnetzmaske 255.255.254.0 ergibt 192.168.0.0 als Netzadresse!
  $\Rightarrow$ 192.168.1.0 ist eine Hostadresse im Netz 192.168.0.0/23.

Frage: Können die beiden Netze 192.168.0.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Frage: Können die beiden Netze 192.168.1.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn es gibt keine passende Subnetzmaske:

- 192.168.0.0/23 enthält die Netze 192.168.{0,1}.0/24
- 192.168.2.0/23 enthält die Netze 192.168.{2,3}.0/24
- 192.168.1.0/23 ist keine gültige Netzadresse, denn UND-Verknüpfung der Adresse 192.168.1.0 mit der Subnetzmaske 255.255.254.0 ergibt 192.168.0.0 als Netzadresse!
  $\Rightarrow$ 192.168.1.0 ist eine Hostadresse im Netz 192.168.0.0/23.

Frage: Können die beiden Netze 192.168.0.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn die beiden Netze sind nicht benachbart:

- Das Netz 192.168.0.0/22 würde zwar beide Subnetze enthalten, zusätzlich aber auch die beiden Netze 192.168.{1,3}.0/24.

Frage: Können die beiden Netze 192.168.1.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn es gibt keine passende Subnetzmaske:

- 192.168.0.0/23 enthält die Netze 192.168.{0,1}.0/24
- 192.168.2.0/23 enthält die Netze 192.168.{2,3}.0/24
- 192.168.1.0/23 ist keine gültige Netzadresse, denn UND-Verknüpfung der Adresse 192.168.1.0 mit der Subnetzmaske 255.255.254.0 ergibt 192.168.0.0 als Netzadresse!
  $\Rightarrow$ 192.168.1.0 ist eine Hostadresse im Netz 192.168.0.0/23.

Frage: Können die beiden Netze 192.168.0.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn die beiden Netze sind nicht benachbart:

- Das Netz 192.168.0.0/22 würde zwar beide Subnetze enthalten, zusätzlich aber auch die beiden Netze 192.168.{1,3}.0/24.

Frage: Können die vier Netze 192.168.{4,5,6,7}.0/24 zu einem größeren Netz zusammengefasst werden?

Frage: Können die beiden Netze 192.168.1.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn es gibt keine passende Subnetzmaske:

- 192.168.0.0/23 enthält die Netze 192.168.{0,1}.0/24
- 192.168.2.0/23 enthält die Netze 192.168.{2,3}.0/24
- 192.168.1.0/23 ist keine gültige Netzadresse, denn UND-Verknüpfung der Adresse 192.168.1.0 mit der Subnetzmaske 255.255.254.0 ergibt 192.168.0.0 als Netzadresse!
  $\Rightarrow$ 192.168.1.0 ist eine Hostadresse im Netz 192.168.0.0/23.

Frage: Können die beiden Netze 192.168.0.0/24 und 192.168.2.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Nein, denn die beiden Netze sind nicht benachbart:

- Das Netz 192.168.0.0/22 würde zwar beide Subnetze enthalten, zusätzlich aber auch die beiden Netze 192.168.{1,3}.0/24.

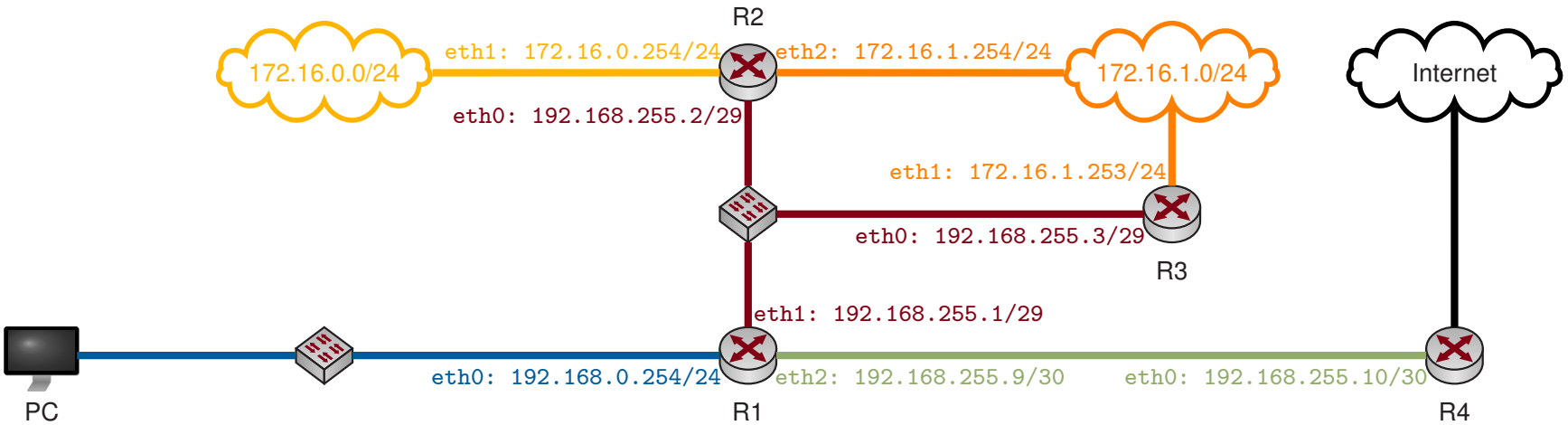
Frage: Können die vier Netze 192.168.{4,5,6,7}.0/24 zu einem größeren Netz zusammengefasst werden?

Antwort: Ja, das Netz 192.168.4.0/22 umfasst genau diese vier Netze.

Übersicht

Statisches Routing

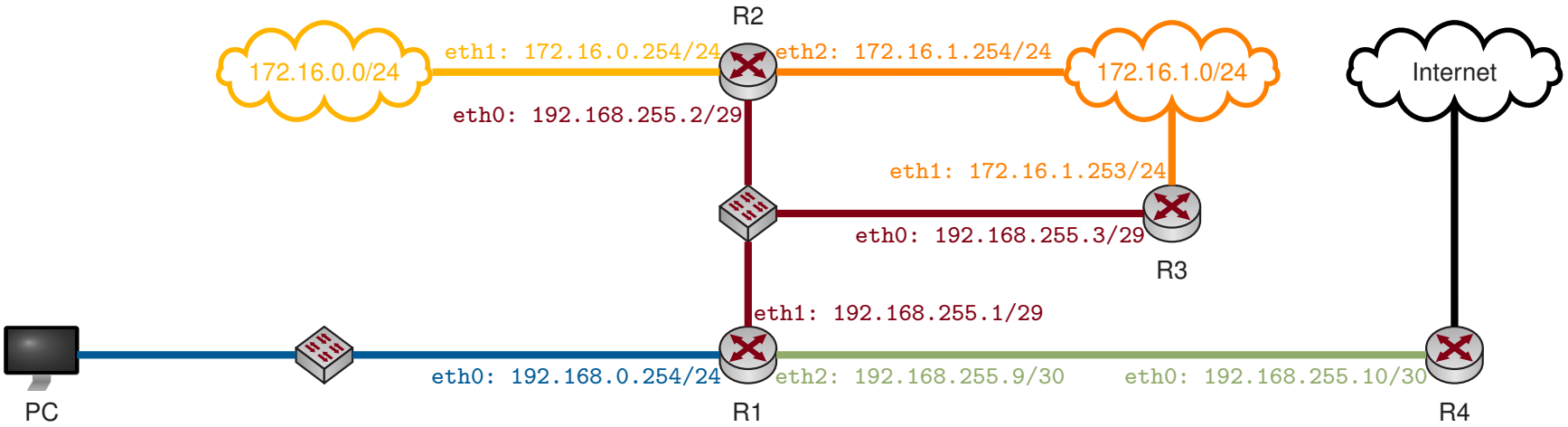
Wir betrachten im Folgenden das unten abgebildete Beispielnetzwerk:



- Die Farben der Links und Interface-Adressen verdeutlichen die einzelnen Subnetze
- Das Netzwerk 192.168.255.0/29 (rot) verfügt über 6 nutzbare Hostadressen
- Das Netzwerk 192.168.255.8/30 (grün) ist ein Transportnetz mit nur 2 nutzbaren Hostadressen
- Die übrigen Netze sind /24 Netze mit jeweils 254 nutzbaren Hostadressen

Frage: Wie entscheidet R1, an welchen Next-Hop ein Paket weitergeleitet werden soll?

Routing Table

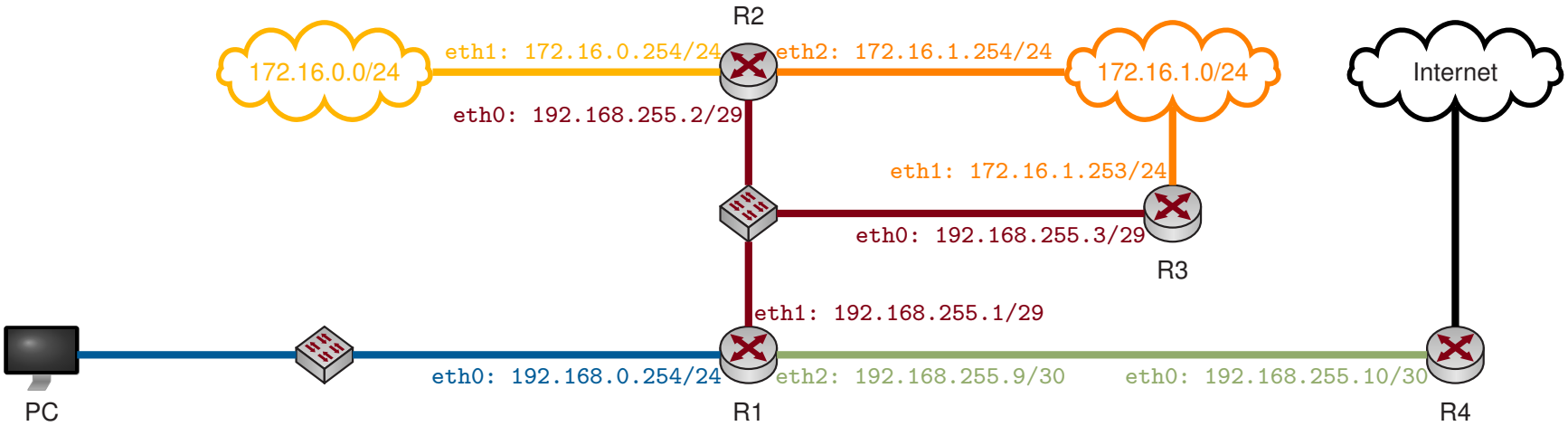


Definition: Routing Table

In der Routing-Tabelle speichert ein Router (oder Host)

- die Netzadresse eines Ziels,
- die zugehörige Subnetzemaske (oder das Präfix),
- den zugehörigen Next-Hop (auch Gateway genannt),
- das Interface, über welches dieser Next-Hop erreichbar ist, und
- die Kosten bzw. die Metrik bis zum Ziel.

Routing Table



Beispiel: Routing-Tabelle für R1

Destination	Mask	Gateway	Metric	Iface
192.168.255.8	255.255.255.252	0.0.0.0	0	eth2
192.168.255.0	255.255.255.248	0.0.0.0	0	eth1
192.168.0.0	255.255.255.0	0.0.0.0	0	eth0
172.16.1.0	255.255.255.0	192.168.255.3	1	eth1
172.16.0.0	255.255.254.0	192.168.255.2	1	eth1
0.0.0.0	0.0.0.0	192.168.255.10	0	eth2

- Die Netze 172.16.{0.1}.0/24 wurden zusammengefasst
- Die Route 0.0.0.0 wird auch als Default Route bezeichnet
- Interessant: R1 kennt zwei (eigentlich sogar drei) Routen zum Netz 172.16.1.0/24 !

Longest Prefix Matching

1. R1 berechnet das logische UND aus der Zieladresse des Pakets und den Masken („Genmasks“) in seiner Routingtabelle
2. Das Ergebnis wird mit dem Eintrag in der Spalte „Destination“ verglichen
3. Stimmt das Ergebnis damit überein, werden Gateway und zugehöriges Interface bestimmt
4. Nachdem die MAC-Adresse des Gateways ggf. via ARP aufgelöst wurde, wird das Paket mit einem neuen Ethernet Header versehen und weitergeleitet

Beispiel: R1 erhalte ein Paket mit Zieladresse 172.16.1.23

Destination	Mask	Gateway	Metric	Iface
192.168.255.8	255.255.255.252	0.0.0.0	0	eth2
192.168.255.0	255.255.255.248	0.0.0.0	0	eth1
192.168.0.0	255.255.255.0	0.0.0.0	0	eth0
172.16.1.0	255.255.255.0	192.168.255.3	1	eth1
172.16.0.0	255.255.254.0	192.168.255.2	1	eth1
0.0.0.0	0.0.0.0	192.168.255.10	0	eth2

Longest Prefix Matching

1. R1 berechnet das logische UND aus der Zieladresse des Pakets und den Masken („Genmasks“) in seiner Routingtabelle
2. Das Ergebnis wird mit dem Eintrag in der Spalte „Destination“ verglichen
3. Stimmt das Ergebnis damit überein, werden Gateway und zugehöriges Interface bestimmt
4. Nachdem die MAC-Adresse des Gateways ggf. via ARP aufgelöst wurde, wird das Paket mit einem neuen Ethernet Header versehen und weitergeleitet

Beispiel: R1 erhalte ein Paket mit Zieladresse 172.16.1.23

	Destination	Mask	Gateway	Metric	Iface
→	192.168.255.8	255.255.255.252	0.0.0.0	0	eth2
	192.168.255.0	255.255.255.248	0.0.0.0	0	eth1
	192.168.0.0	255.255.255.0	0.0.0.0	0	eth0
	172.16.1.0	255.255.255.0	192.168.255.3	1	eth1
	172.16.0.0	255.255.254.0	192.168.255.2	1	eth1
	0.0.0.0	0.0.0.0	192.168.255.10	0	eth2

IP-Adresse	10101100 . 00010000 . 00000001 . 00010111	172.16.1.23
Subnetz Maske	11111111 . 11111111 . 11111111 . 11111100	255.255.255.252
Netzadresse	10101100 . 00010000 . 00000001 . 00010100	172.16.1.20

$\Rightarrow$ kein Match, da 172.16.1.20 $\neq$ 192.168.255.8

Longest Prefix Matching

1. R1 berechnet das logische UND aus der Zieladresse des Pakets und den Masken („Genmasks“) in seiner Routingtabelle
2. Das Ergebnis wird mit dem Eintrag in der Spalte „Destination“ verglichen
3. Stimmt das Ergebnis damit überein, werden Gateway und zugehöriges Interface bestimmt
4. Nachdem die MAC-Adresse des Gateways ggf. via ARP aufgelöst wurde, wird das Paket mit einem neuen Ethernet Header versehen und weitergeleitet

Beispiel: R1 erhalte ein Paket mit Zieladresse 172.16.1.23

	Destination	Mask	Gateway	Metric	Iface
	192.168.255.8	255.255.255.252	0.0.0.0	0	eth2
→	192.168.255.0	255.255.255.248	0.0.0.0	0	eth1
	192.168.0.0	255.255.255.0	0.0.0.0	0	eth0
	172.16.1.0	255.255.255.0	192.168.255.3	1	eth1
	172.16.0.0	255.255.254.0	192.168.255.2	1	eth1
	0.0.0.0	0.0.0.0	192.168.255.10	0	eth2

IP-Adresse	10101100 . 00010000 . 00000001 . 00010111	172.16.1.23
Subnetz Maske	11111111 . 11111111 . 11111111 . 11111000	255.255.255.248
Netzadresse	10101100 . 00010000 . 00000001 . 00010000	172.16.1.16

$\Rightarrow$ kein Match, da 172.16.1.16 $\neq$ 192.168.255.0

Longest Prefix Matching

1. R1 berechnet das logische UND aus der Zieladresse des Pakets und den Masken („Genmasks“) in seiner Routingtabelle
2. Das Ergebnis wird mit dem Eintrag in der Spalte „Destination“ verglichen
3. Stimmt das Ergebnis damit überein, werden Gateway und zugehöriges Interface bestimmt
4. Nachdem die MAC-Adresse des Gateways ggf. via ARP aufgelöst wurde, wird das Paket mit einem neuen Ethernet Header versehen und weitergeleitet

Beispiel: R1 erhalte ein Paket mit Zieladresse 172.16.1.23

	Destination	Mask	Gateway	Metric	Iface
	192.168.255.8	255.255.255.252	0.0.0.0	0	eth2
	192.168.255.0	255.255.255.248	0.0.0.0	0	eth1
→	192.168.0.0	255.255.255.0	0.0.0.0	0	eth0
	172.16.1.0	255.255.255.0	192.168.255.3	1	eth1
	172.16.0.0	255.255.254.0	192.168.255.2	1	eth1
	0.0.0.0	0.0.0.0	192.168.255.10	0	eth2

IP-Adresse	10101100 . 00010000 . 00000001 . 00010111	172.16.1.23
Subnetz Maske	11111111 . 11111111 . 11111111 . 00000000	255.255.255.0
Netzadresse	10101100 . 00010000 . 00000001 . 00000000	172.16.1.0

$\Rightarrow$ kein Match, da 172.16.1.0 $\neq$ 192.168.0.0

Longest Prefix Matching

1. R1 berechnet das logische UND aus der Zieladresse des Pakets und den Masken („Genmasks“) in seiner Routingtabelle
2. Das Ergebnis wird mit dem Eintrag in der Spalte „Destination“ verglichen
3. Stimmt das Ergebnis damit überein, werden Gateway und zugehöriges Interface bestimmt
4. Nachdem die MAC-Adresse des Gateways ggf. via ARP aufgelöst wurde, wird das Paket mit einem neuen Ethernet Header versehen und weitergeleitet

Beispiel: R1 erhalte ein Paket mit Zieladresse 172.16.1.23

	Destination	Mask	Gateway	Metric	Iface
	192.168.255.8	255.255.255.252	0.0.0.0	0	eth2
	192.168.255.0	255.255.255.248	0.0.0.0	0	eth1
	192.168.0.0	255.255.255.0	0.0.0.0	0	eth0
→	172.16.1.0	255.255.255.0	192.168.255.3	1	eth1
	172.16.0.0	255.255.254.0	192.168.255.2	1	eth1
	0.0.0.0	0.0.0.0	192.168.255.10	0	eth2

IP-Adresse	10101100	00010000	00000001	00010111	172.16.1.23
Subnetz Maske	11111111	11111111	11111111	00000000	255.255.255.0
Netzadresse	10101100	00010000	00000001	00000000	172.16.1.0

$\Rightarrow$ Match, da 172.16.1.0 = 172.16.1.0 $\Rightarrow$ Gateway ist 192.168.255.3

Definition: Longest Prefix Matching

Die Routingtabelle wird von längeren Präfixen (spezifischeren Routen) hin zu kürzeren Präfixen (weniger spezifische Routen) durchsucht. Der erste passende Eintrag liefert das Gateway (Next-Hop) eines Pakets. Diesen Prozess bezeichnet man als Longest Prefix Matching.

Beachte:

Destination	Mask	Gateway	Metric	Iface
192.168.255.8	255.255.255.252	0.0.0.0	0	eth2
192.168.255.0	255.255.255.248	0.0.0.0	0	eth1
192.168.0.0	255.255.255.0	0.0.0.0	0	eth0
172.16.1.0	255.255.255.0	192.168.255.3	1	eth1
172.16.0.0	255.255.254.0	192.168.255.2	1	eth1
0.0.0.0	0.0.0.0	192.168.255.10	0	eth2

- Der Eintrag für 172.16.0.0/23 liefert ebenfalls einen Match, ist aber weniger spzifisch als der für 172.16.1.0/24 (1 Bit kürzeres Präfix).
- Die Default Route 0.0.0.0 liefert immer einen Match (logisches UND mit der Maske 0.0.0.0).
- Es ist nicht garantiert, dass das Gateway der Default Route („Gateway of last resort“) eine Route zum Ziel kennt ($\rightarrow$ ICMP Destination Unreachable / Host Unreachable).
- Routen zu direkt verbundenen Netzen (also solchen, zu denen ein Router selbst gehört) können automatisch erzeugt werden. Das Gateway ist in diesem Fall der Router selbst (bzw. eine seiner IP-Adressen).
- Routen zu entfernten Netzen müssen „gelernt“ werden – entweder durch händisches Eintragen (statisches Routing) oder durch Routing Protokolle (dynamisches Routing).

Dynamisches Routing

Mittels Routing Protokollen können Router miteinander kommunizieren und Routen untereinander austauschen. Routingprotokolle können nach Ihrer Funktionsweise wie folgt gruppiert werden:

Distanz-Vektor-Protokolle

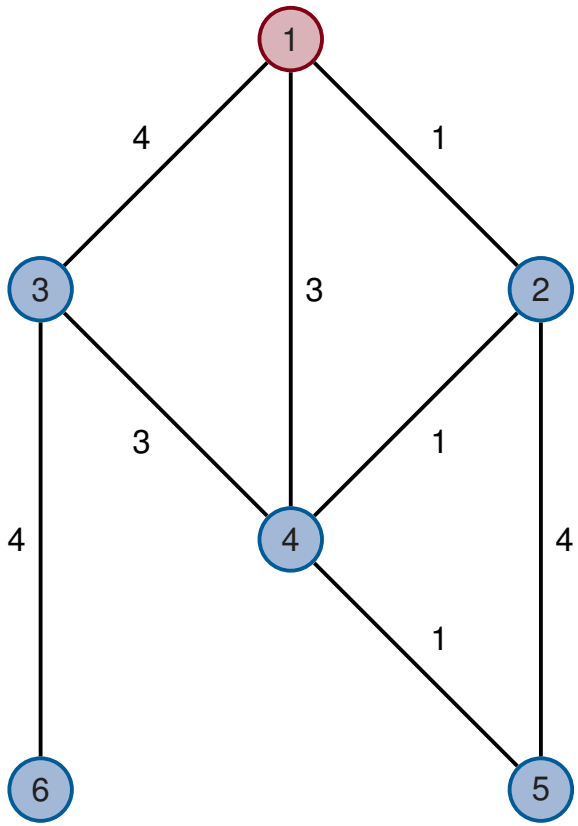
- Router kennen nur Richtung (Next Hop) und Entfernung (Kosten) zu einem Ziel (vgl. Straßenschild mit Richtungs- und Entfernungsangabe)
- Router haben keine Information über die Netzwerktopologie
- Router tauschen untereinander lediglich kummulierte Metriken aus (z. B. den Inhalt Ihrer Routingtabellen)
- Funktionsprinzip basiert auf dem Algorithmus von Bellman-Ford

Link-State-Protokolle

- Router informieren einander detailliert über den aktuellen Zustand einzelner Links
- Router verfügen über vollständige Topologieinformationen
- Häufig komplexe Nachbarschaftsbeziehungen und Update-Nachrichten
- Basierend auf den Topologieinformationen bestimmt jeder Router kürzeste Pfade, z. B. mittels Dijkstras Algorithmus

Algorithmus von Bellman-Ford

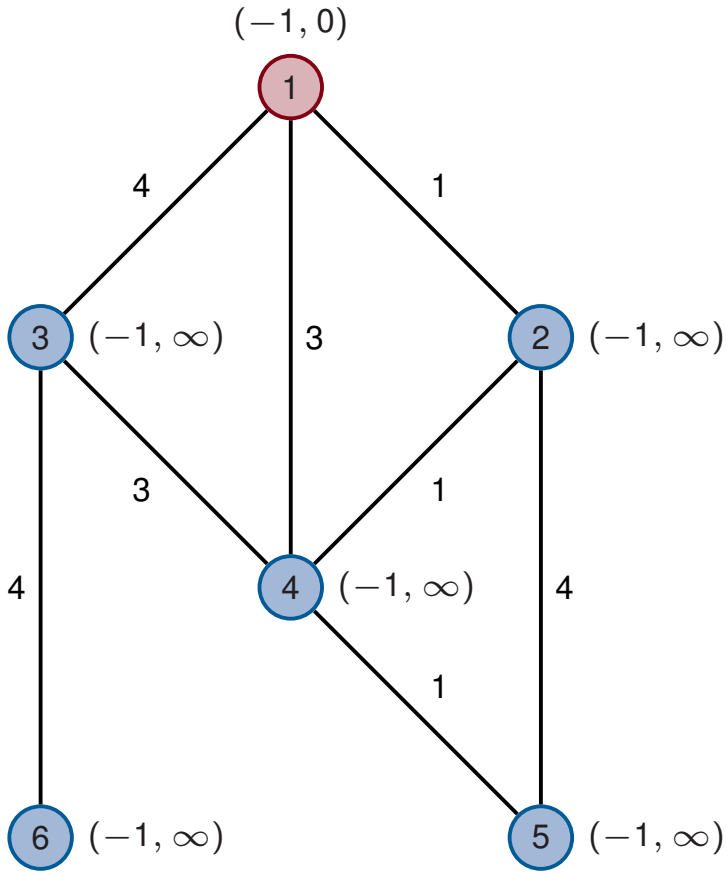
- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i



Algorithmus von Bellman-Ford

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i

// Initialisierung
for $i \in \mathcal{N}$ **do**
$p[i] = -1$
$d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
end for
$T = \{s\}$ // Menge der erreichbaren Knoten

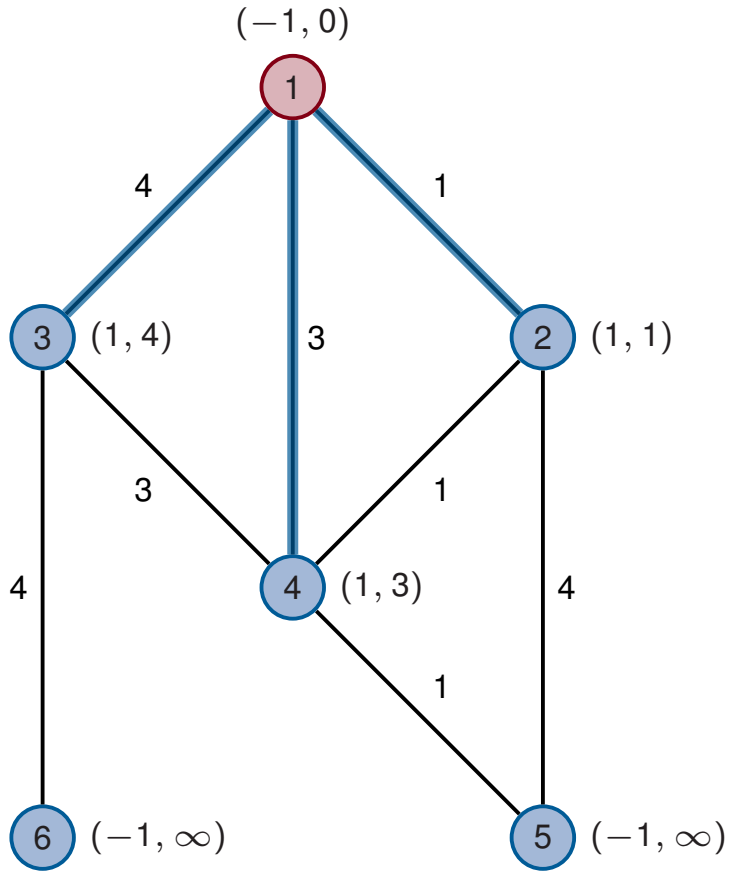


Algorithmus von Bellman-Ford

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i

// Initialisierung
for $i \in \mathcal{N}$ **do**
  $p[i] = -1$
  $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
end for
$T = \{s\}$ // Menge der erreichbaren Knoten

// Berechnung der Pfade
while $d[j]$ changes for some $j \in \mathcal{N}$ **do**
  $S = \{\}$ // Aktualisierte Knoten
  for $i \in T$ **do**
    // Für alle Nachbarn von i:
    for $\forall j : (i, j) \in \mathcal{E}$ **do**
      if $d[i] + c_{ij} < d[j]$ **then**
        $p[j] = i$
        $d[j] = d[i] + c_{ij}$
        $S = S \cup \{j\}$
      end if
    end for
  end for
  $T = T \cup S$
end while

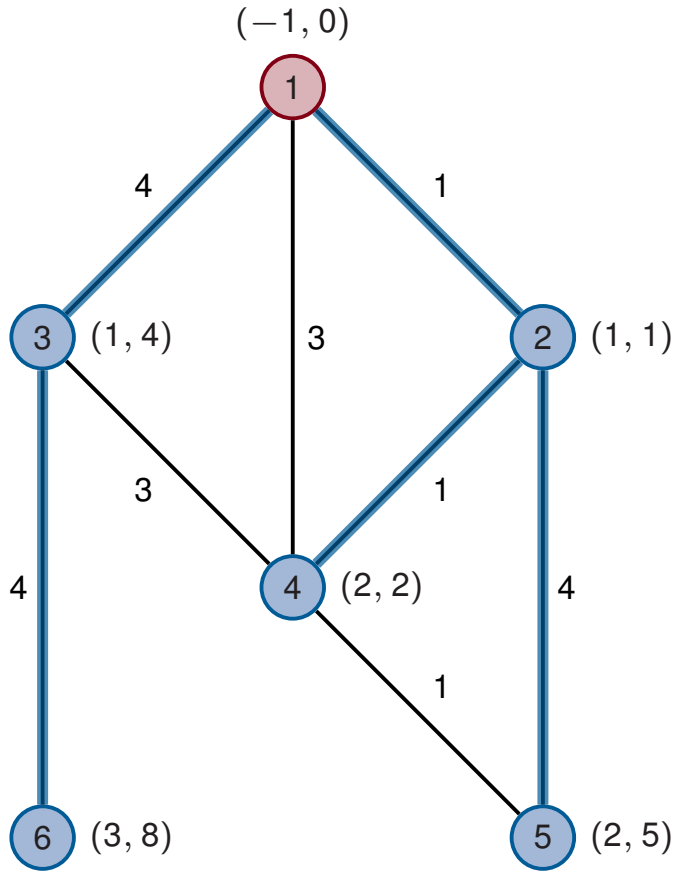


Algorithmus von Bellman-Ford

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i

// Initialisierung
for $i \in \mathcal{N}$ **do**
  $p[i] = -1$
  $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
end for
$T = \{s\}$ // Menge der erreichbaren Knoten

// Berechnung der Pfade
while $d[j]$ changes for some $j \in \mathcal{N}$ **do**
  $S = \{\}$ // Aktualisierte Knoten
  for $i \in T$ **do**
    // Für alle Nachbarn von i:
    for $\forall j : (i, j) \in \mathcal{E}$ **do**
      if $d[i] + c_{ij} < d[j]$ **then**
        $p[j] = i$
        $d[j] = d[i] + c_{ij}$
        $S = S \cup \{j\}$
      end if
    end for
  end for
  $T = T \cup S$
end while

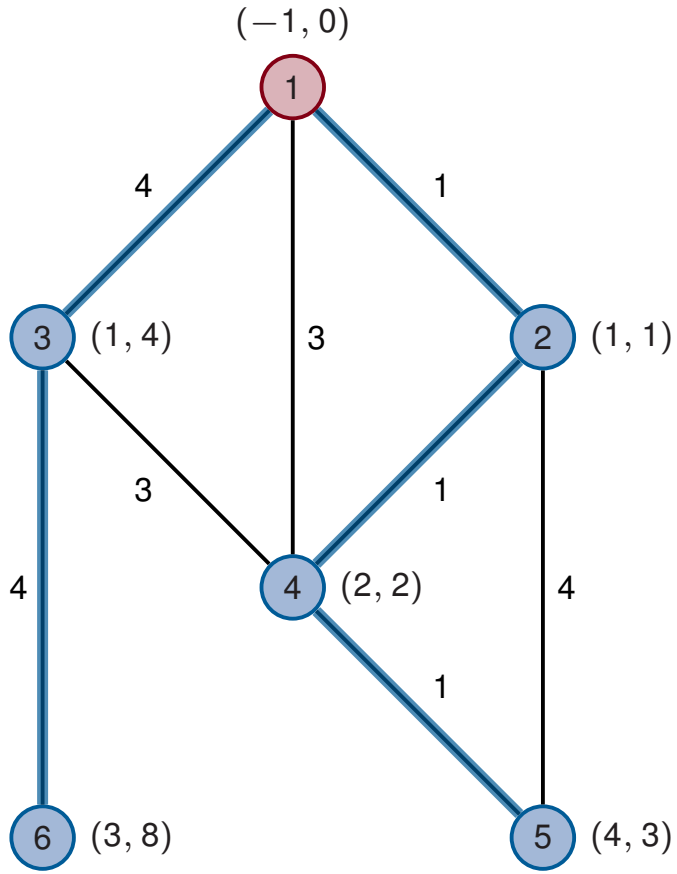


Algorithmus von Bellman-Ford

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i

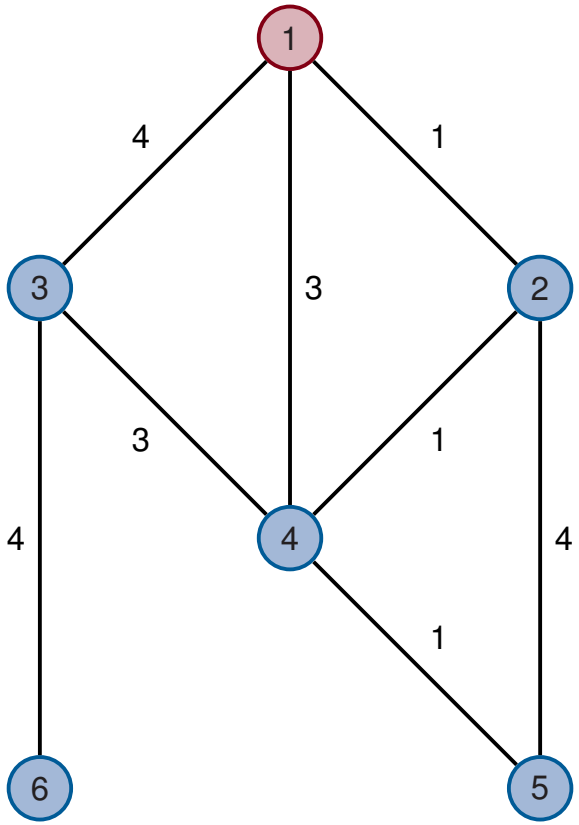
// Initialisierung
for $i \in \mathcal{N}$ **do**
  $p[i] = -1$
  $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
end for
$T = \{s\}$ // Menge der erreichbaren Knoten

// Berechnung der Pfade
while $d[j]$ changes for some $j \in \mathcal{N}$ **do**
  $S = \{\}$ // Aktualisierte Knoten
  for $i \in T$ **do**
    // Für alle Nachbarn von i:
    for $\forall j : (i, j) \in \mathcal{E}$ **do**
      if $d[i] + c_{ij} < d[j]$ **then**
        $p[j] = i$
        $d[j] = d[i] + c_{ij}$
        $S = S \cup \{j\}$
      end if
    end for
  end for
  $T = T \cup S$
end while



Dijkstras Algorithmus

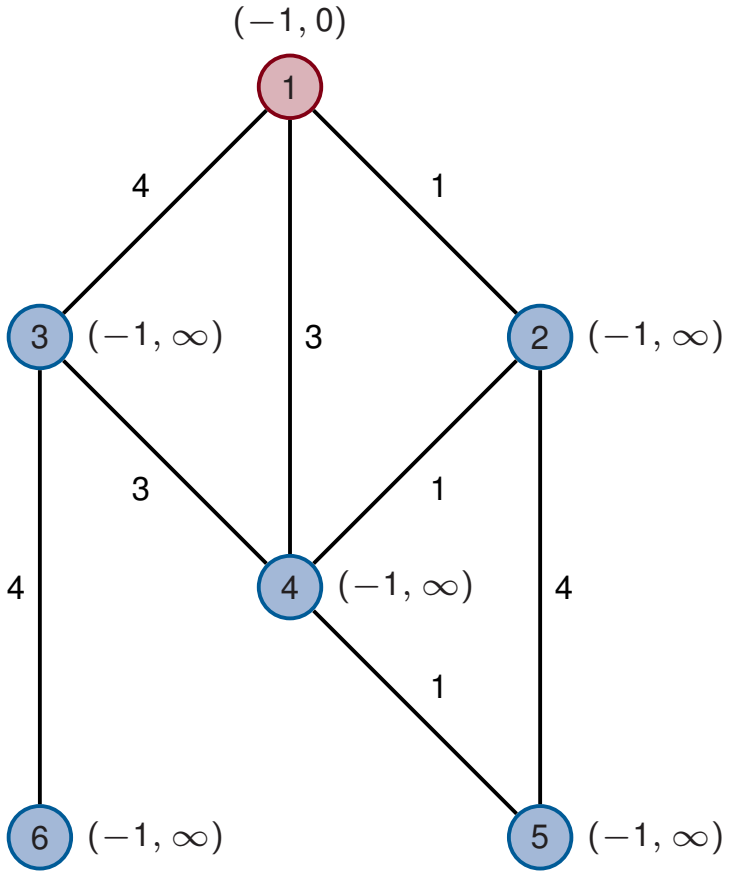
- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i
- Priority Queue Q, welche Elemente nach Schlüsseln sortiert ausgibt



Dijkstras Algorithmus

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i
- Priority Queue Q, welche Elemente nach Schlüsseln sortiert ausgibt

for $i \in \mathcal{N}$ **do**
  $p[i] = -1$
  $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
  $Q.enqeue(i, d[i])$
end for
$T = \{\}$ // Menge der bearbeiteter Knoten

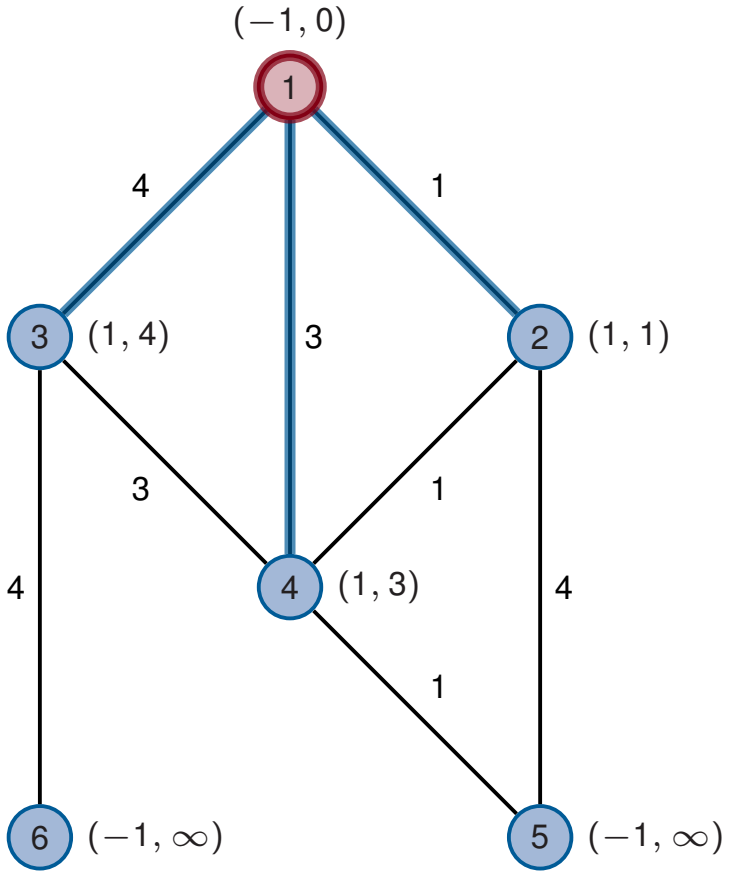


Dijkstras Algorithmus

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i
- Priority Queue Q, welche Elemente nach Schlüsseln sortiert ausgibt

for $i \in \mathcal{N}$ **do**
  $p[i] = -1$
  $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
  $Q.enqeue(i, d[i])$
end for
$T = \{\}$ // Menge der bearbeiteter Knoten

// Berechnung der Pfade
while $T \neq \mathcal{N}$ **do**
  $i = Q.dequeue()$
  $T = T \cup \{i\}$
  // Für alle Nachbarn von i:
  for $\forall j : (i, j) \in \mathcal{E}$ **do**
    if $d[i] + c_{ij} < d[j]$ **then**
      $Q.decreaseKey(j, d[i] + c_{ij})$
      $p[j] = i$
    end if
  end for
end while

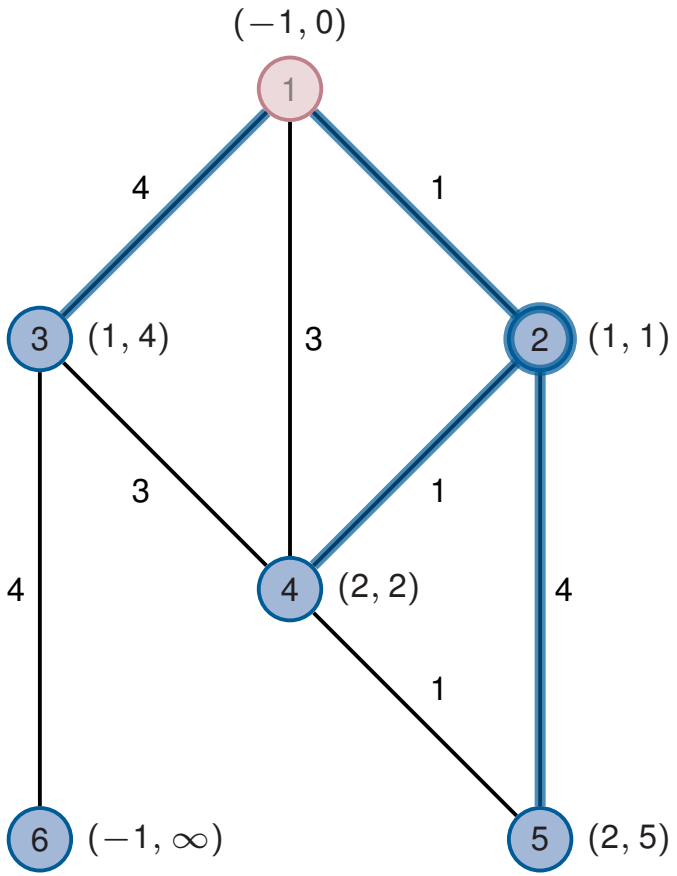


Dijkstras Algorithmus

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i
- Priority Queue Q, welche Elemente nach Schlüsseln sortiert ausgibt

for $i \in \mathcal{N}$ **do**
  $p[i] = -1$
  $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
  $Q.enqeue(i, d[i])$
end for
$T = \{\}$ // Menge der bearbeiteter Knoten

// Berechnung der Pfade
while $T \neq \mathcal{N}$ **do**
  $i = Q.dequeue()$
  $T = T \cup \{i\}$
  // Für alle Nachbarn von i:
  for $\forall j : (i, j) \in \mathcal{E}$ **do**
    if $d[i] + c_{ij} < d[j]$ **then**
      $Q.decreaseKey(j, d[i] + c_{ij})$
      $p[j] = i$
    end if
  end for
end while

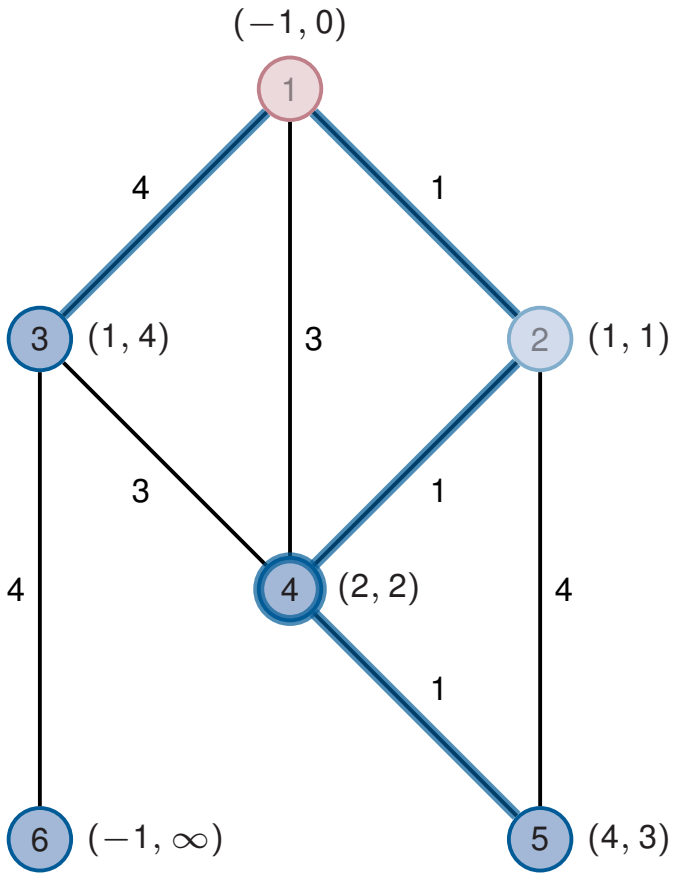


Dijkstras Algorithmus

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i
- Priority Queue Q, welche Elemente nach Schlüsseln sortiert ausgibt

for $i \in \mathcal{N}$ **do**
  $p[i] = -1$
  $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
  $Q.enqeue(i, d[i])$
end for
$T = \{\}$ // Menge der bearbeiteter Knoten

// Berechnung der Pfade
while $T \neq \mathcal{N}$ **do**
  $i = Q.dequeue()$
  $T = T \cup \{i\}$
  // Für alle Nachbarn von i:
  for $\forall j : (i, j) \in \mathcal{E}$ **do**
    if $d[i] + c_{ij} < d[j]$ **then**
      $Q.decreaseKey(j, d[i] + c_{ij})$
      $p[j] = i$
    end if
  end for
end while

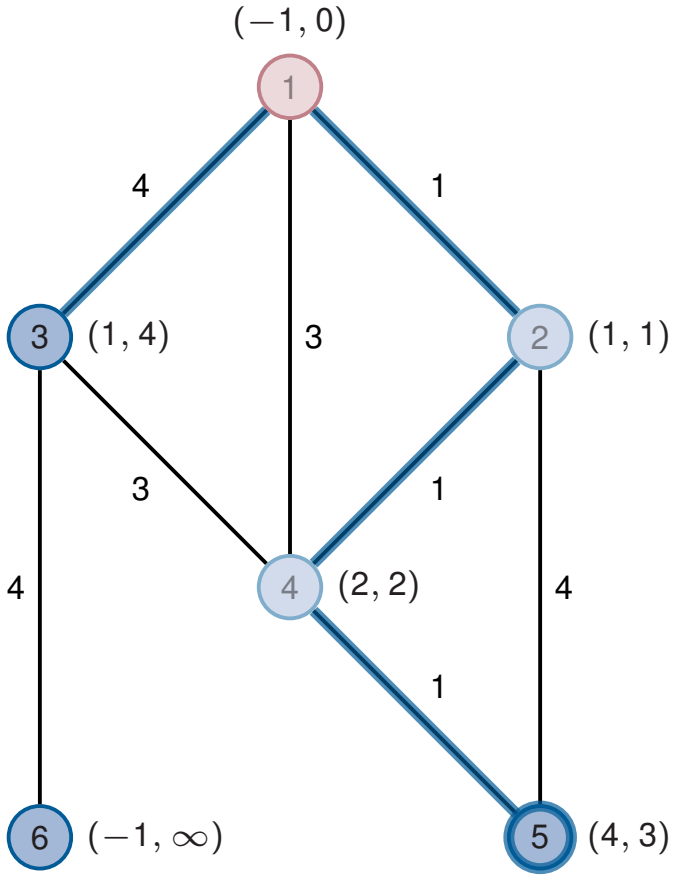


Dijkstras Algorithmus

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i
- Priority Queue Q, welche Elemente nach Schlüsseln sortiert ausgibt

for $i \in \mathcal{N}$ **do**
    $p[i] = -1$
    $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
    $Q.enqeue(i, d[i])$
end for
$T = \{\}$ // Menge der bearbeiteter Knoten

// Berechnung der Pfade
while $T \neq \mathcal{N}$ **do**
    $i = Q.dequeue()$
    $T = T \cup \{i\}$
    // Für alle Nachbarn von i:
    for $\forall j : (i, j) \in \mathcal{E}$ **do**
        if $d[i] + c_{ij} < d[j]$ **then**
            $Q.decreaseKey(j, d[i] + c_{ij})$
            $p[j] = i$
        end if
    end for
end while

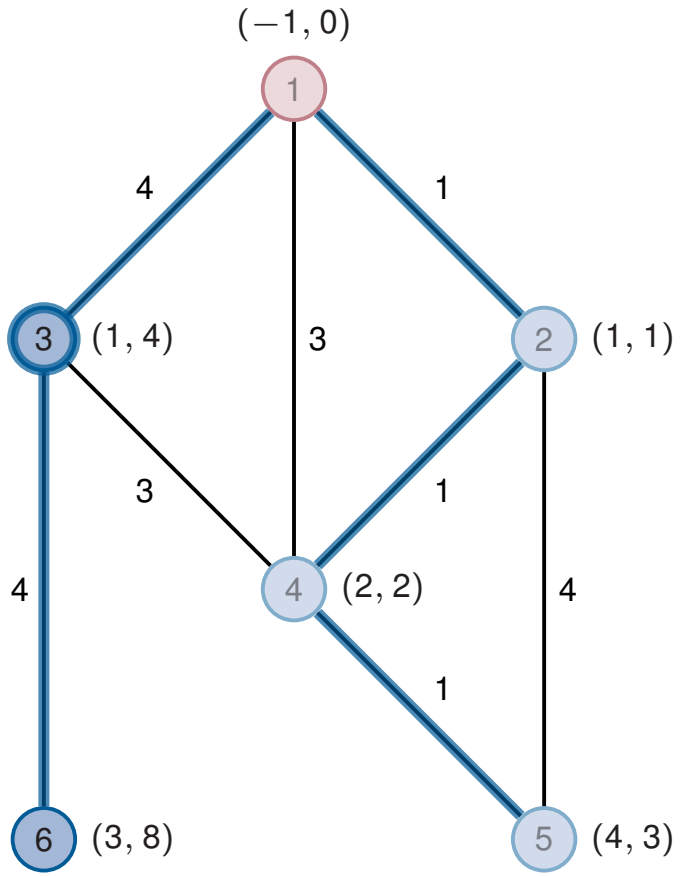


Dijkstras Algorithmus

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i
- Priority Queue Q, welche Elemente nach Schlüsseln sortiert ausgibt

for $i \in \mathcal{N}$ **do**
  $p[i] = -1$
  $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
  $Q.enqeue(i, d[i])$
end for
$T = \{\}$ // Menge der bearbeiteter Knoten

// Berechnung der Pfade
while $T \neq \mathcal{N}$ **do**
  $i = Q.dequeue()$
  $T = T \cup \{i\}$
  // Für alle Nachbarn von i:
  for $\forall j : (i, j) \in \mathcal{E}$ **do**
    if $d[i] + c_{ij} < d[j]$ **then**
      $Q.decreaseKey(j, d[i] + c_{ij})$
      $p[j] = i$
    end if
  end for
end while

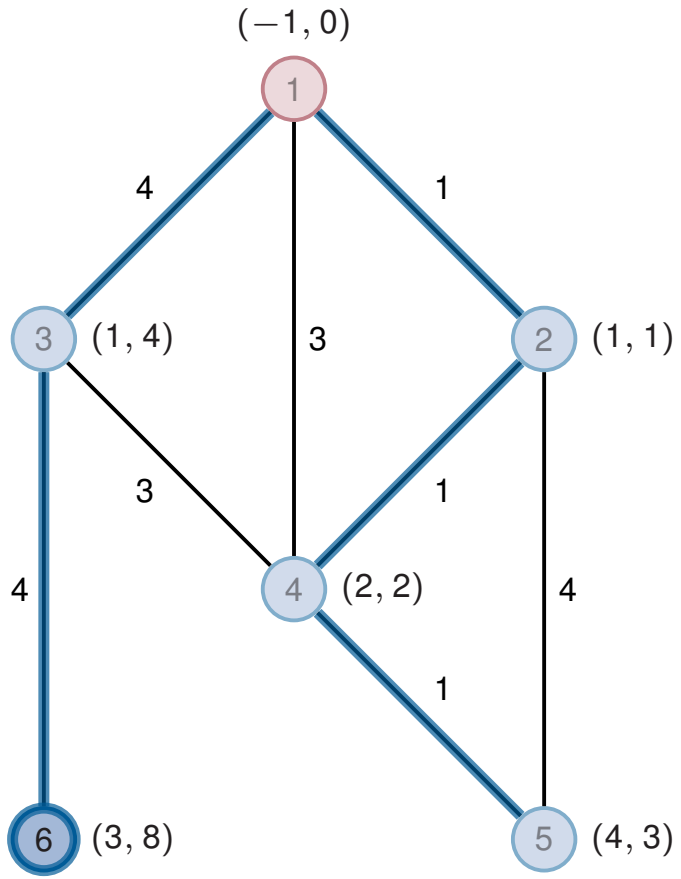


Dijkstras Algorithmus

- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i
- Priority Queue Q, welche Elemente nach Schlüsseln sortiert ausgibt

for $i \in \mathcal{N}$ **do**
  $p[i] = -1$
  $d[i] = \begin{cases} 0 & i = s \\ \infty & \text{sonst} \end{cases}$
  $Q.enqeue(i, d[i])$
end for
$T = \{\}$ // Menge der bearbeiteter Knoten

// Berechnung der Pfade
while $T \neq \mathcal{N}$ **do**
  $i = Q.dequeue()$
  $T = T \cup \{i\}$
  // Für alle Nachbarn von i:
  for $\forall j : (i, j) \in \mathcal{E}$ **do**
    if $d[i] + c_{ij} < d[j]$ **then**
      $Q.decreaseKey(j, d[i] + c_{ij})$
      $p[j] = i$
    end if
  end for
end while



Eigenschaften des Algorithmus von Bellman-Ford:

- Im *n*-ten Durchlauf der while-Schleife werden alle Pfade der Länge höchstens *n* berücksichtigt (vergleiche min-plus-Produkt in *n*-ter Potenz)
- Keine komplexen Datenstrukturen notwendig
- Verteilte (dezentrale) Implementierung ohne Kenntnis der Topologie möglich (Beispiel)

Eigenschaften des Algorithmus von Dijkstra:

- Es werden immer Pfade über den im jeweiligen Schritt am günstigsten erreichbaren Knoten gesucht (Greedy-Prinzip)
- Wurde ein Knoten abgearbeitet, so ist garantiert, dass der kürzeste Pfad zu diesem Knoten gefunden ist
- Resourcenintensiver als der Algorithmus von Bellman-Ford, da komplexere Datenstrukturen notwendig (aber asymptotisch bessere Laufzeit)
- Vollständige Kenntnis der Netzwerktopologie erforderlich

Routing Information Protocol (RIP)

- Einfach Distanz-Vektor-Protokoll
- RIPv1 standardisiert in RFC 1058 (1988)
- Unterstützung für CIDR in RIPv2 hinzugefügt (RFC 2453, 1998)
- Einzige Metrik: Hop Count
- Hop Count Limit von 15, weiter entfernte Ziele sind nicht erreichbar

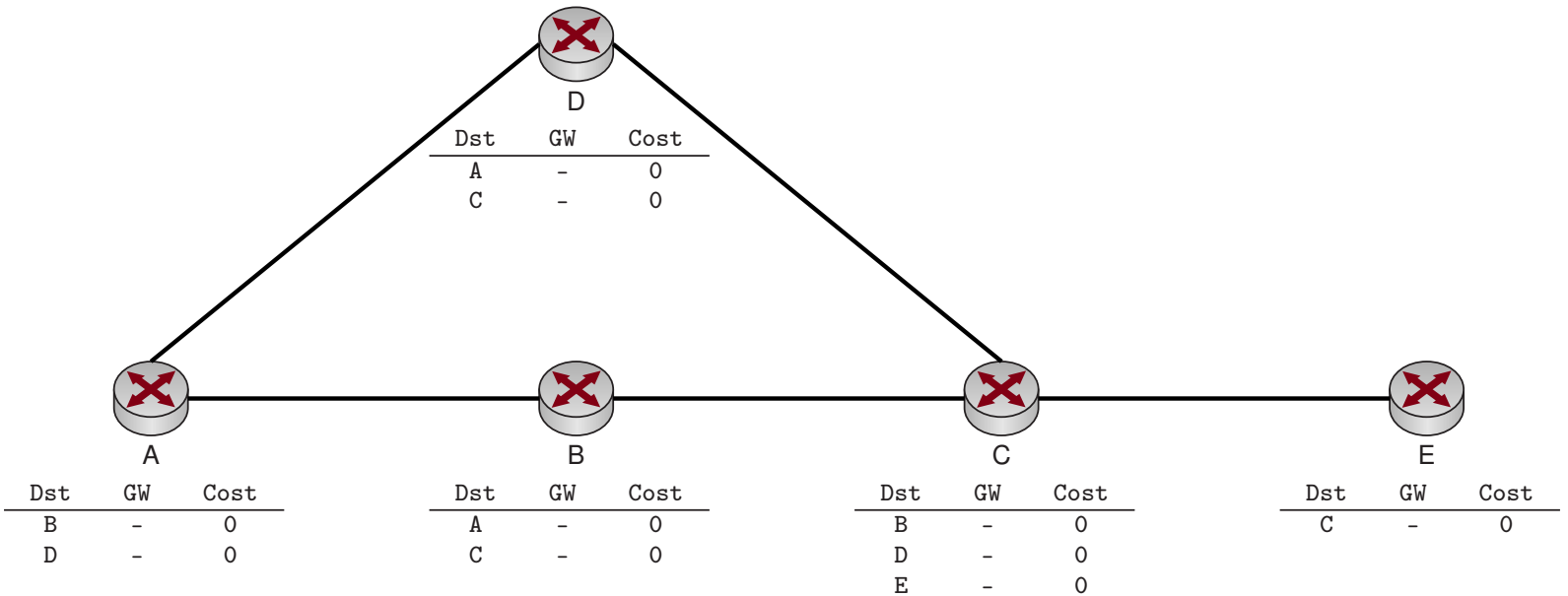
Funktionsweise:

- Router senden in regelmäßigen Abständen (Standardwert 30 s) den Inhalt ihrer Routingtabelle an die Multicast-Adresse 224.0.0.9[4].
- Jeder RIP-Router akzeptiert diese Update-Nachrichten, inkrementiert die Metrik der enthaltenen Routen um 1 und vergleicht die Routen mit bereits vorhandenen Routen aus seiner Routingtabelle:
  - Enthält das Update eine noch unbekannte Route, wird diese in die eigene Routingtabelle übernommen
  - Enthält das Update eine Route zu einem bekannten Ziel aber mit niedrigeren Kosten, so wird die vorhandene Route durch das Update ersetzt
  - Andernfalls wird die vorhandene Route beibehalten
- Bleiben fünf aufeinanderfolgende Updates von einem Nachbarn aus, so werden alle Routen über diesen Next Hop aus der Routingtabelle entfernt.

[4] Multicast wird im Rahmen der Vorlesung nicht näher behandelt. Stellen Sie sich die Funktionsweise hier wie einen Broadcast im lokalen Subnetz vor.

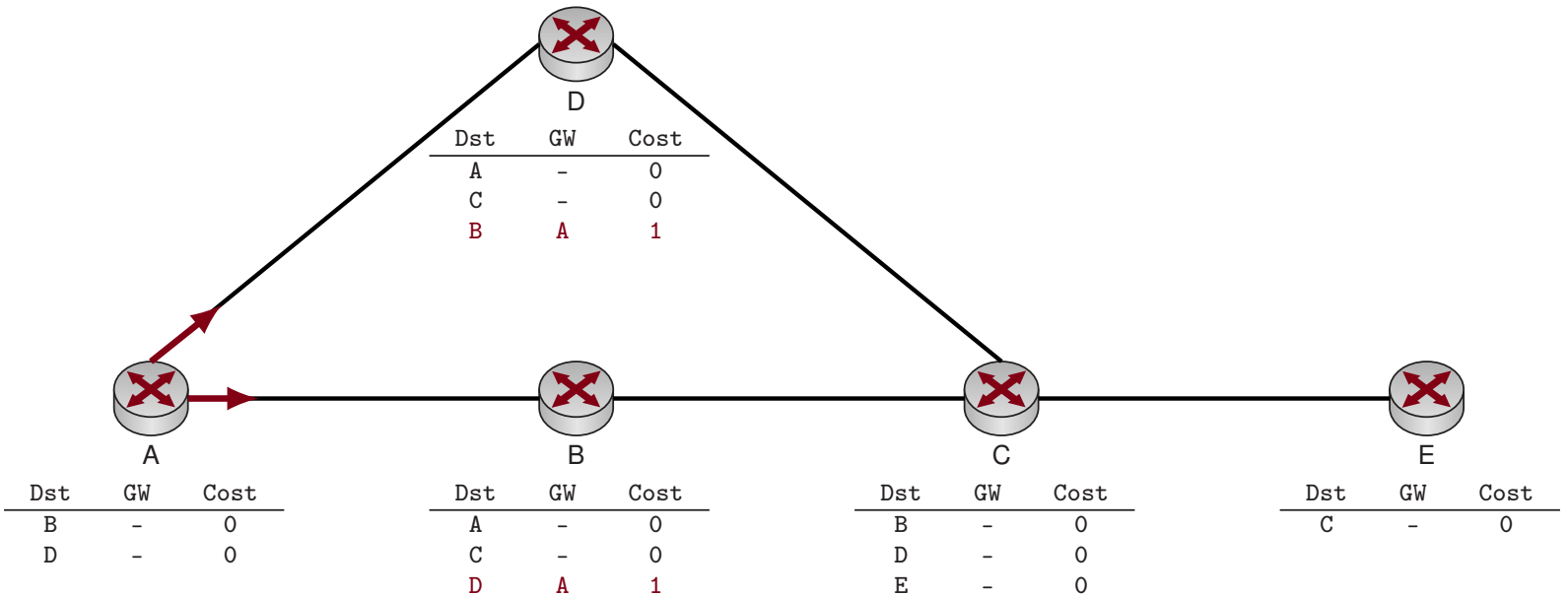
Beispiel mit vereinfachter Darstellung:

- Zwischen jeweils zwei Routern befindet sich sinnvollerweise ein Switch, welcher den Anschluss weiterer Computer an das jeweilige Subnetz ermöglicht
- Anstelle von IP- und Netzadressen tragen wir in die Routingtabellen lediglich die Namen der Router ein
- Für direkt erreichbare Nachbarn tragen wir kein Gateway ein
- Update-Nachrichten werden rundenweise verschickt (erst A, dann B, . . .)



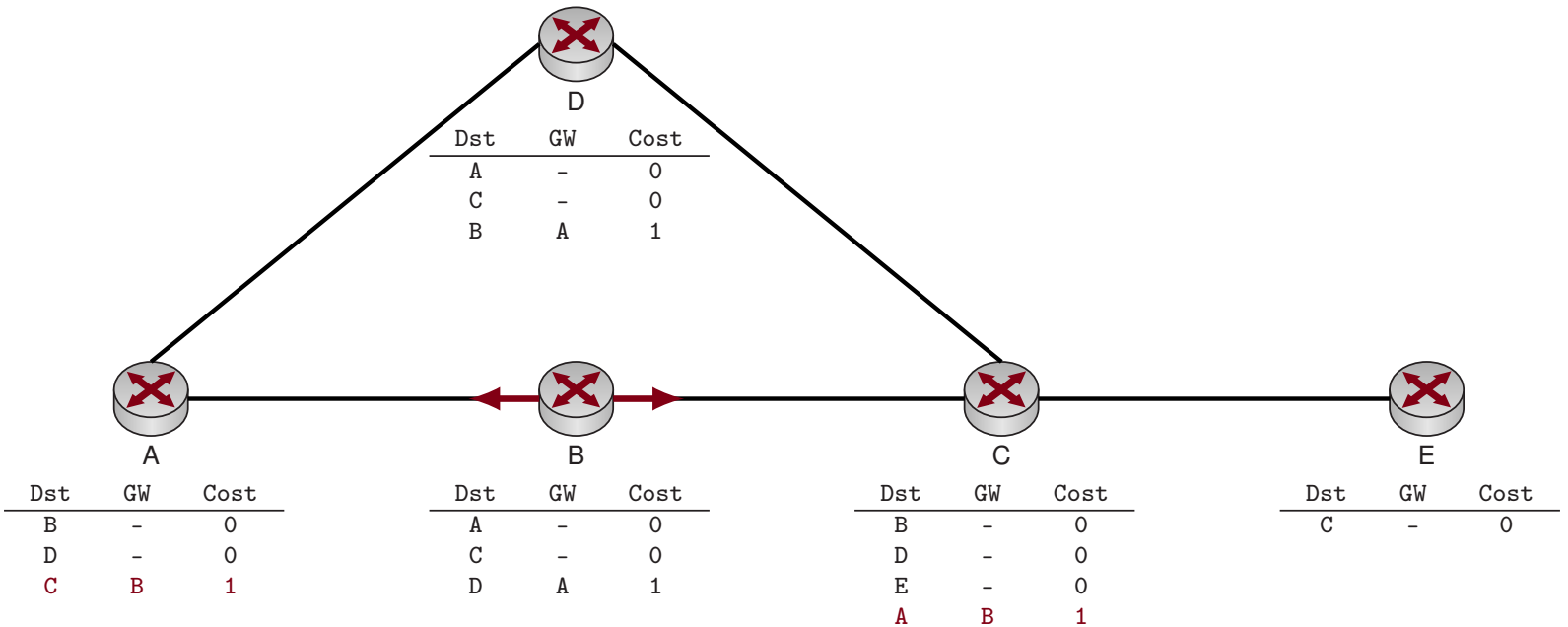
Beispiel mit vereinfachter Darstellung:

- Zwischen jeweils zwei Routern befindet sich sinnvollerweise ein Switch, welcher den Anschluss weiterer Computer an das jeweilige Subnetz ermöglicht
- Anstelle von IP- und Netzadressen tragen wir in die Routingtabellen lediglich die Namen der Router ein
- Für direkt erreichbare Nachbarn tragen wir kein Gateway ein
- Update-Nachrichten werden rundenweise verschickt (erst A, dann B, . . .)



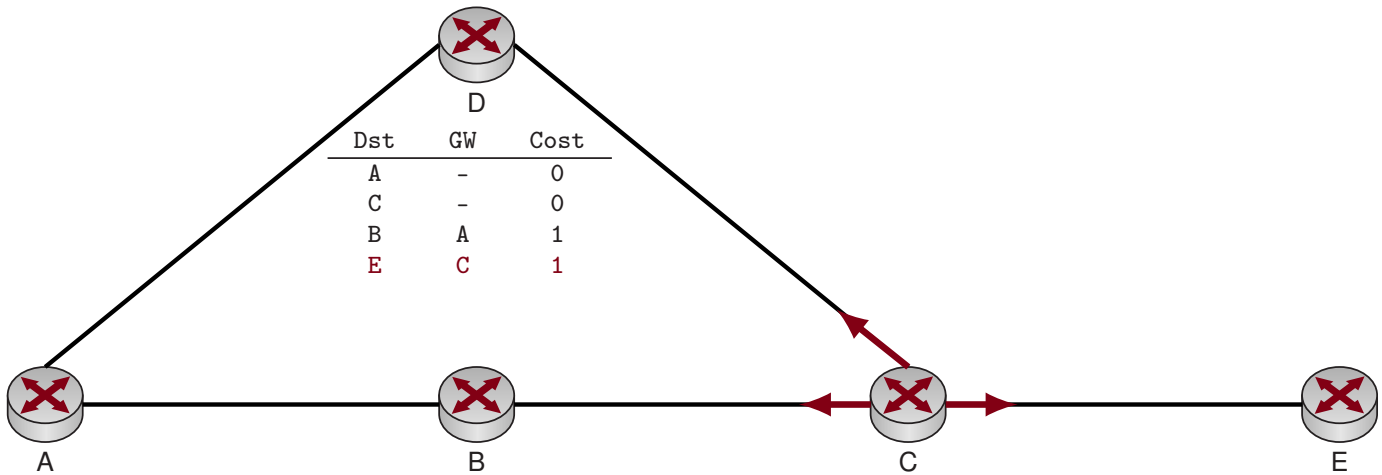
Beispiel mit vereinfachter Darstellung:

- Zwischen jeweils zwei Routern befindet sich sinnvollerweise ein Switch, welcher den Anschluss weiterer Computer an das jeweilige Subnetz ermöglicht
- Anstelle von IP- und Netzadressen tragen wir in die Routingtabellen lediglich die Namen der Router ein
- Für direkt erreichbare Nachbarn tragen wir kein Gateway ein
- Update-Nachrichten werden rundenweise verschickt (erst A, dann B, . . .)



Beispiel mit vereinfachter Darstellung:

- Zwischen jeweils zwei Routern befindet sich sinnvollerweise ein Switch, welcher den Anschluss weiterer Computer an das jeweilige Subnetz ermöglicht
- Anstelle von IP- und Netzadressen tragen wir in die Routingtabellen lediglich die Namen der Router ein
- Für direkt erreichbare Nachbarn tragen wir kein Gateway ein
- Update-Nachrichten werden rundenweise verschickt (erst A, dann B, . . .)



D

Dst	GW	Cost
A	-	0
C	-	0
B	A	1
E	C	1

A

Dst	GW	Cost
B	-	0
D	-	0
C	B	1

B

Dst	GW	Cost
A	-	0
C	-	0
D	A	1
E	C	1

C

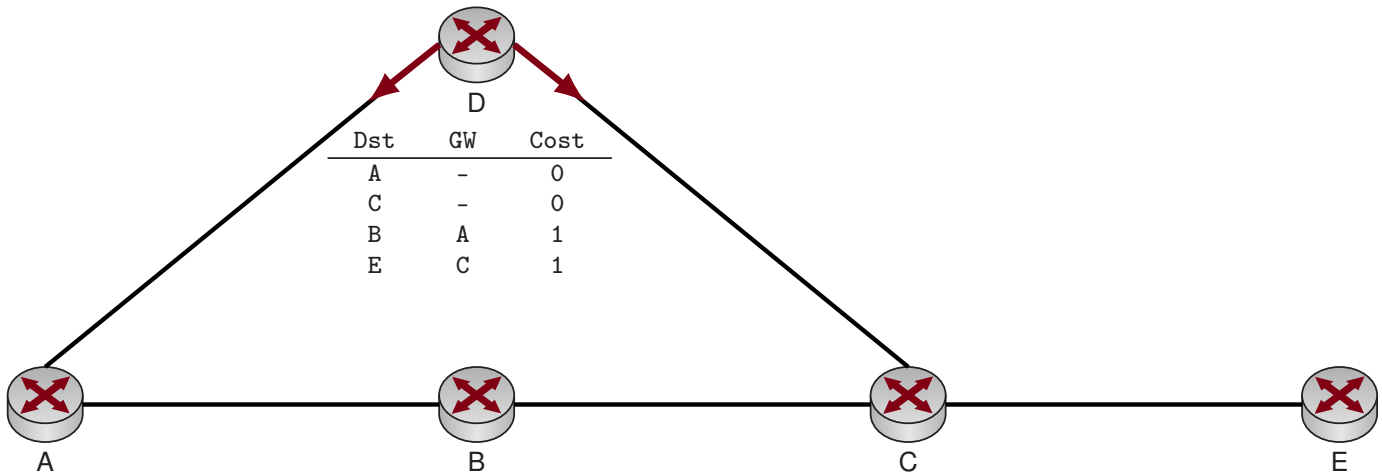
Dst	GW	Cost
B	-	0
D	-	0
E	-	0
A	B	1

E

Dst	GW	Cost
C	-	0
B	C	1
D	C	1
A	C	2

Beispiel mit vereinfachter Darstellung:

- Zwischen jeweils zwei Routern befindet sich sinnvollerweise ein Switch, welcher den Anschluss weiterer Computer an das jeweilige Subnetz ermöglicht
- Anstelle von IP- und Netzadressen tragen wir in die Routingtabellen lediglich die Namen der Router ein
- Für direkt erreichbare Nachbarn tragen wir kein Gateway ein
- Update-Nachrichten werden rundenweise verschickt (erst A, dann B, . . .)



D

Dst	GW	Cost
A	-	0
C	-	0
B	A	1
E	C	1

A

Dst	GW	Cost
B	-	0
D	-	0
C	B	1
E	D	2

B

Dst	GW	Cost
A	-	0
C	-	0
D	A	1
E	C	1

C

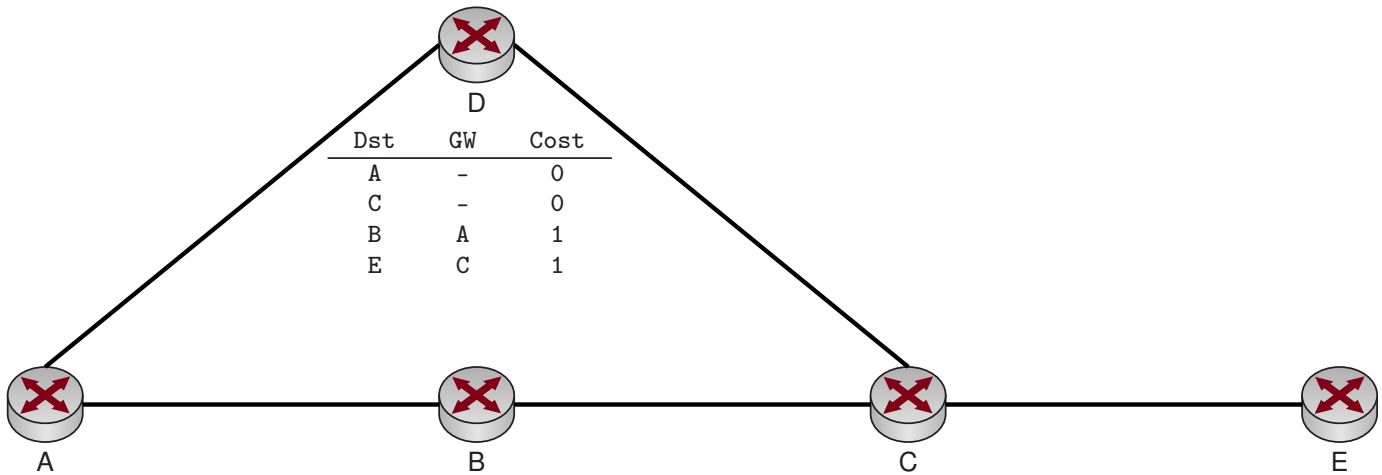
Dst	GW	Cost
B	-	0
D	-	0
E	-	0
A	B	1

E

Dst	GW	Cost
C	-	0
B	C	1
D	C	1
A	C	2

Beispiel mit vereinfachter Darstellung:

- Zwischen jeweils zwei Routern befindet sich sinnvollerweise ein Switch, welcher den Anschluss weiterer Computer an das jeweilige Subnetz ermöglicht
- Anstelle von IP- und Netzadressen tragen wir in die Routingtabellen lediglich die Namen der Router ein
- Für direkt erreichbare Nachbarn tragen wir kein Gateway ein
- Update-Nachrichten werden rundenweise verschickt (erst A, dann B, . . .)



- Nach diesem Schritt kennt jeder Router eine kürzeste Route zu jedem anderen Router
- Da mehrere gleich lange Pfade existieren, bleibt es dem Zufall (der Reihenfolge der Updatenachrichten) überlassen, ob beispielsweise Router A als Next Hop zu E Router D oder B lernt

Das vorangegangene Beispiel lässt sich leicht auf gewichtete Kanten erweitern:

- Metrik ist nicht mehr die Anzahl der Hops zum Ziel, sondern das Kantegewicht
- Router addieren auf erhaltene Updates das Kantengewicht des Links, über den das Update empfangen wurde

$\Rightarrow$ Tutorübungen

Problem:

- Bis jeder Router den besten Next Hop bestimmen kann, dauert es ggf. mehrere „Runden“
- Eine obere Schranke für die Anzahl der notwendigen Nachrichten, die jeder Router senden muss, ist die maximale Entfernung zwischen zwei Routern in Hops
- Die maximale Entfernung beträgt bei RIP 15 Hops
- Da Updates nur alle 30 s verschickt werden, ergibt sich eine maximale Verzögerung von $15 \cdot 30\,\text{s} = 7.5\,\text{min}$

Das vorangegangene Beispiel lässt sich leicht auf gewichtete Kanten erweitern:

- Metrik ist nicht mehr die Anzahl der Hops zum Ziel, sondern das Kantegewicht
- Router addieren auf erhaltene Updates das Kantengewicht des Links, über den das Update empfangen wurde

$\Rightarrow$ Tutorübungen

Problem:

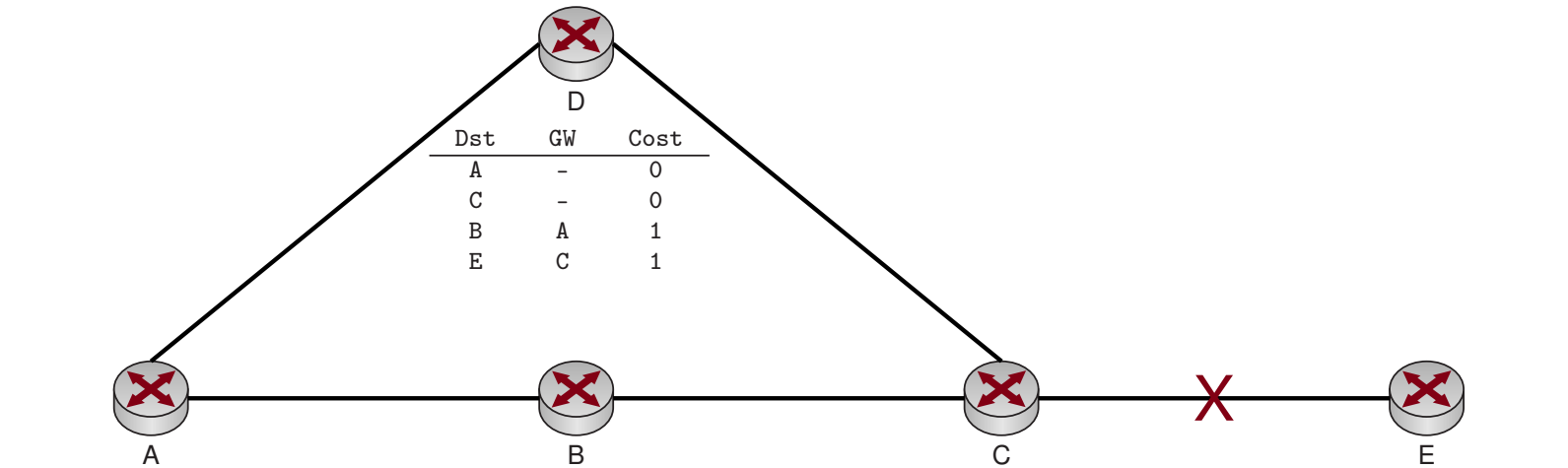
- Bis jeder Router den besten Next Hop bestimmen kann, dauert es ggf. mehrere „Runden“
- Eine obere Schranke für die Anzahl der notwendigen Nachrichten, die jeder Router senden muss, ist die maximale Entfernung zwischen zwei Routern in Hops
- Die maximale Entfernung beträgt bei RIP 15 Hops
- Da Updates nur alle 30 s verschickt werden, ergibt sich eine maximale Verzögerung von $15 \cdot 30\,\text{s} = 7.5\,\text{min}$

Lösung: Triggered Updates

- Sobald ein Router eine Änderung an seiner Routingtabelle vornimmt, sendet er sofort ein Update
- Dies führt zu einer Welle von Updates durch das Netzwerk
- Konvergenzzeit wird reduziert, aber das Netzwerk während der Updates ggf. stark belastet

Anderes Problem: Count to infinity

- Link zwischen C und E fällt aus
- Reihenfolge, in der Updates versendet werden, ist dem Zufall überlassen



D

Dst	GW	Cost
A	-	0
C	-	0
B	A	1
E	C	1

A

Dst	GW	Cost
B	-	0
D	-	0
C	B	1
E	D	2

B

Dst	GW	Cost
A	-	0
C	-	0
D	A	1
E	C	1

C

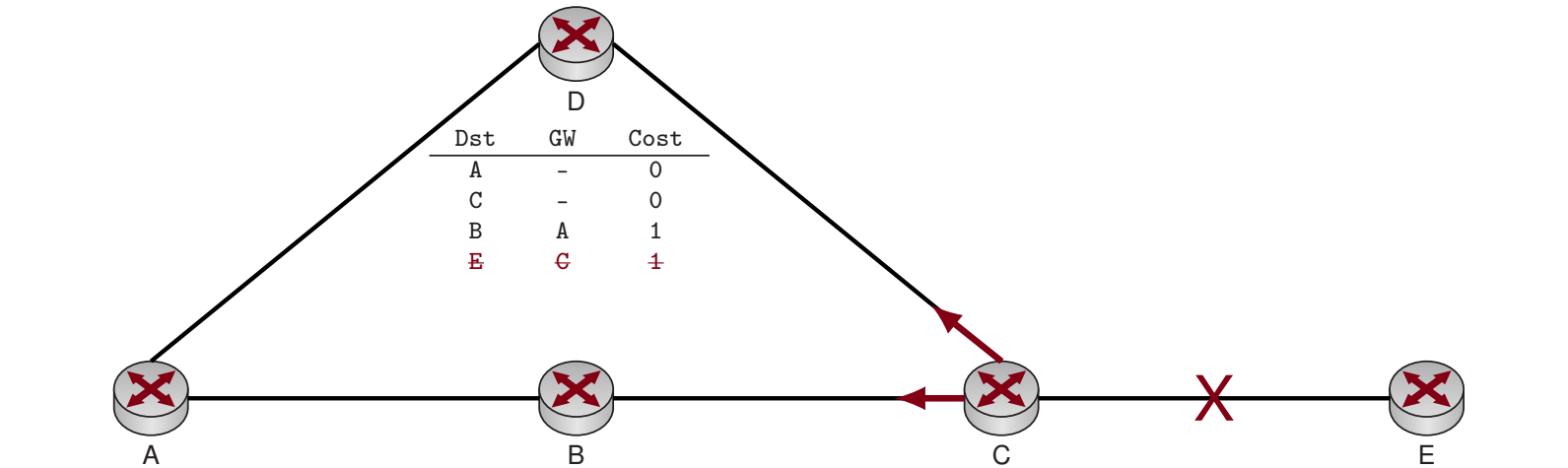
Dst	GW	Cost
B	-	0
D	-	0
~~E~~	~~-~~	~~0~~
A	B	1

E

Dst	GW	Cost

Anderes Problem: Count to infinity

- Link zwischen C und E fällt aus
- Reihenfolge, in der Updates versendet werden, ist dem Zufall überlassen



D

Dst	GW	Cost
A	-	0
C	-	0
B	A	1
~~E~~	~~C~~	~~1~~

A

Dst	GW	Cost
B	-	0
D	-	0
C	B	1
E	D	2

B

Dst	GW	Cost
A	-	0
C	-	0
D	A	1
~~E~~	~~C~~	~~1~~

C

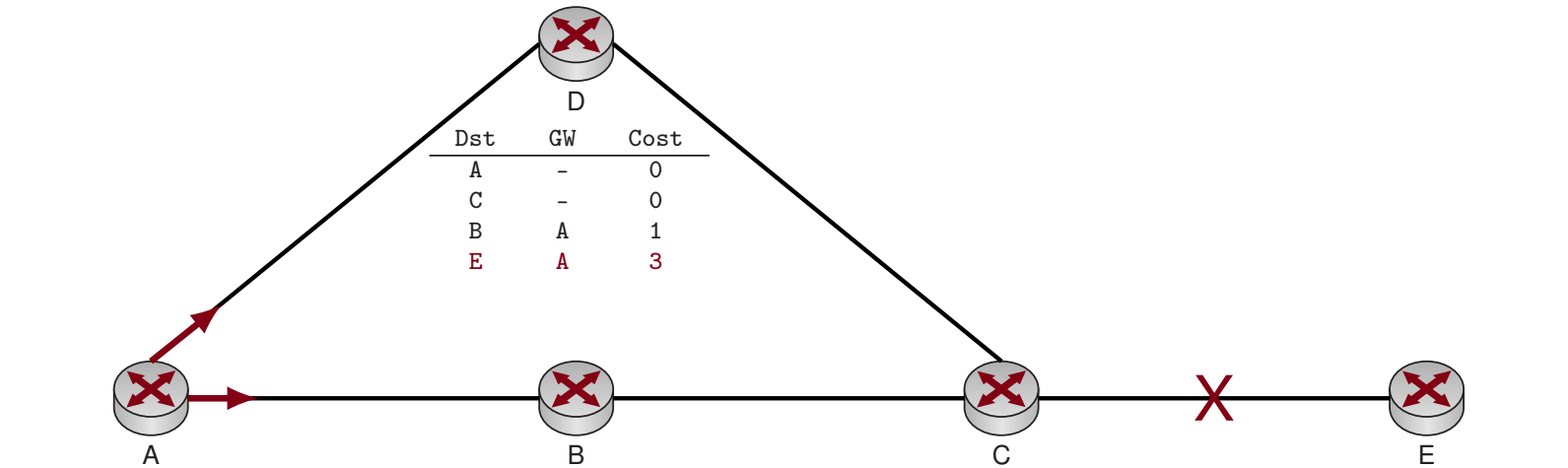
Dst	GW	Cost
B	-	0
D	-	0
A	B	1

E

Dst	GW	Cost

Anderes Problem: Count to infinity

- Link zwischen C und E fällt aus
- Reihenfolge, in der Updates versendet werden, ist dem Zufall überlassen



D

Dst	GW	Cost
A	-	0
C	-	0
B	A	1
E	A	3

A

Dst	GW	Cost
B	-	0
D	-	0
C	B	1
E	D	2

B

Dst	GW	Cost
A	-	0
C	-	0
D	A	1
E	A	3

C

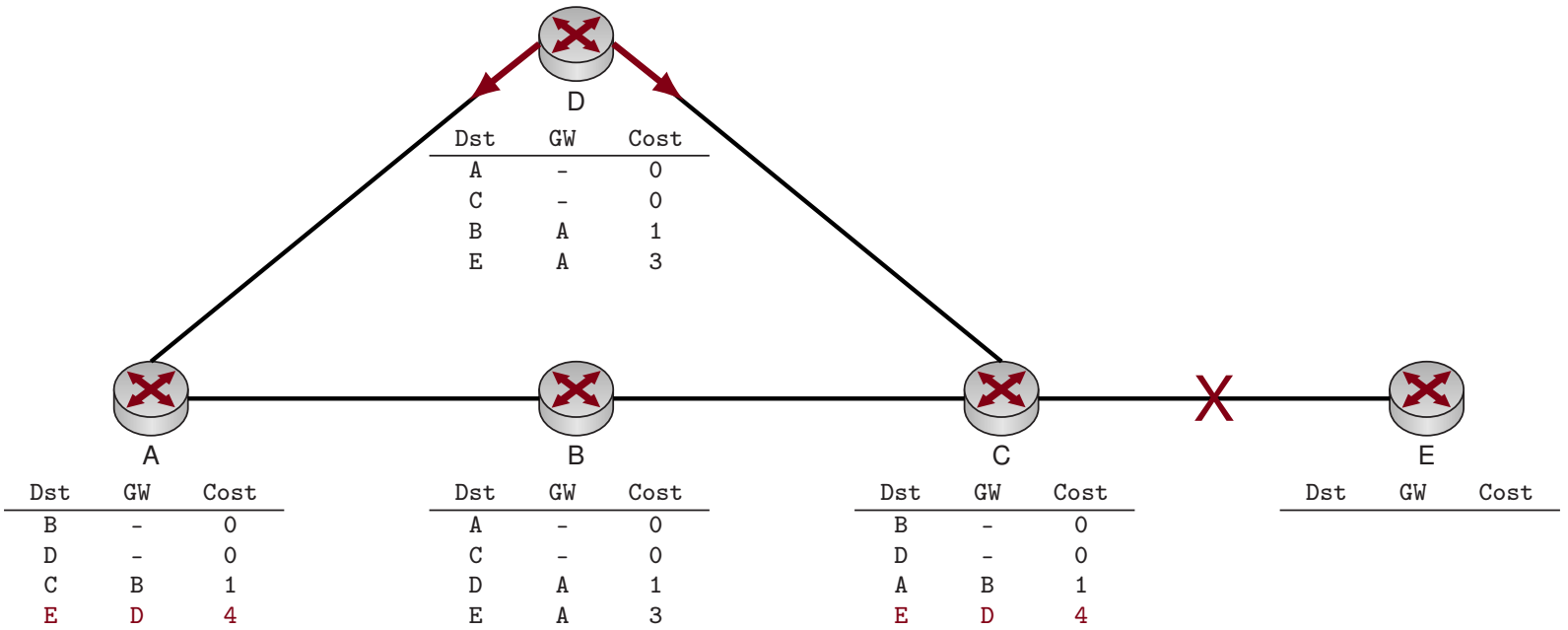
Dst	GW	Cost
B	-	0
D	-	0
A	B	1

E

Dst	GW	Cost

Anderes Problem: Count to infinity

- Link zwischen C und E fällt aus
- Reihenfolge, in der Updates versendet werden, ist dem Zufall überlassen



Je nach Reihenfolge der Updates

- wird die fehlerhafte Route zu E weiterverbreitet und
- die Metrik stets incrementiert
- bis schließlich das Hop Count Limit von 15 erreicht ist.

Diesen Vorgang bezeichnet man als Count to infinity.

Lösungen:

- Split Horizon
  - Neue Regel: „Sende dem Nachbarn, von dem Du die Route zu X gelernt hast, keine Route zu X“
  - Im vorherigen Beispiel würde A die Route zu E nicht an D, wohl aber an B schicken
  - Split Horizon verbessert die Situation, kann das Problem aber nicht lösen
- Poison Reverse
  - Anstelle dem Nachbarn, von dem eine Route zu X gelernt wurde, keine Route zu X mehr zu schicken, wird eine Route mit unendlicher Metrik gesendet
  - Im vorherigen Beispiel würde A die Route zu E an D mit Metrik 15 schicken
  - Die fehlerhafte Route würde nach wie vor B erreichen
  - Auch Poison Reverse kann das Problem nicht vollständig lösen
- Path Vector
  - Sende bei Updates nicht nur Ziel und Kosten, sondern auch den vollständigen Pfad, über den das Ziel erreicht wird
  - Jeder Router prüft vor Installation der Route, ob er selbst in diesem Pfad bereits vorhanden ist
  - Falls ja, handelt es sich um eine Schleife und das Update wird verworfen
  - Path Vector verhindert Routing Loops und damit auch Count to Infinity, vergrößert jedoch die Update-Nachrichten und die Protkollkomplexität

Übersicht: Ausgewählte Routing-Protokolle

Distanz-Vektor-Protokolle

- RIP (Routing Information Protocol)
  Sehr einfaches Protokoll, Hop-Count als einzige Metrik, geeignet für eine geringe Anzahl von Netzen, wird von den meisten Routern unterstützt (sogar einige Heimgeräte).
- IGRP (Interior Gateway Routing Protocol)
  Proprietäres Routing Protokoll von Cisco, unterstützt komplexere Metriken als RIP
- EIGRP (Enhanced Interior Gateway Routing Protocol)
  Proprietäres Routing Protokoll von Cisco, Nachfolger von IGRP, deutlich verbesserte Konvergenzeigenschaften
- AODV (Ad hoc On-Demand Distance Vector)
  Einsatz in kabellosen vermaschten Netzwerken, Routen werden nicht proaktiv ausgetauscht sondern on-demand gesucht (reaktives Protokoll)

Link-State-Protokolle

- OSPF (Open Shortest Path First)
  Industriestandard für mittlere bis große Anzahl von Netzwerken
- IS-IS (Intermediate System to Intermediate System)
  Seltener eingesetztes, leistungsfähiges Routingprotokoll, welches unabhängig von IP ist (es handelt sich um ein ISO-standardisiertes Layer-3-Protokoll)
- HWMP (Hybrid Wireless Mesh Protocol)
  Ermöglicht Routing in IEEE 802.11s (Wireless Mesh Networks)

Autonome Systeme

Alle bislang vorgestellten Routingprotokolle

- bestimmen beste Pfade anhand objetiver Kriterien (Hopcount, Bandbreite, Delay, . . .),
- bieten aber keine bzw. nur eingeschränkte Möglichkeiten, Routen direkt zu beeinflussen.

Manchmal ist es aber wünschenswert, Routen auf Basis anderer Kriterien zu wählen:

- Tatsächlich anfallende monetäre Kosten
- Netze / Länder, durch die Datenverkehr zu einem Ziel weitergeleitet wird
- Infrastrukturentscheidungen (z. B. Belastung einzelner Router)

Routingentscheidungen auf Basis derartiger Kriterien bezeichnet man als Policy-Based Routing.

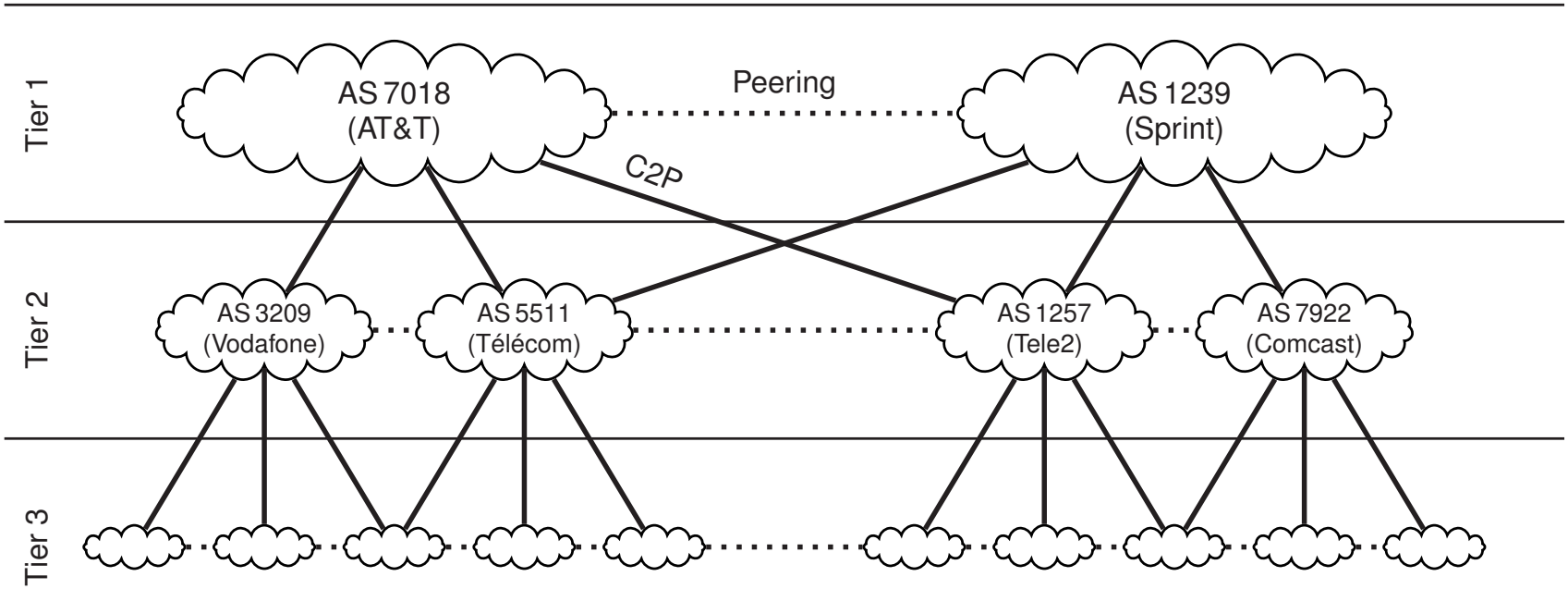
Definition: Autonomes System

Eine Menge von Netzwerken, die unter einheitlicher administrativer Kontrolle stehen, bezeichnet man als Autonomes System (AS). Ein AS wird i. d. R. durch einen 16 Bit Identifier, der sog. AS-Nummer identifiziert. Beim Einsatz von Routingprotokollen wird unterschieden:

- Innerhalb eines autonomen Systems werden Interior Gateway Protocols (IGPs) wie RIP, OSPF, EIGRP oder IS-IS eingesetzt.
- Zum Austausch von Routen zwischen Autonomen Systemen wird ein Exterior Gateway Protocol (EGP) verwendet.

Das einzige in der Praxis verwendete EGP ist das Border Gateway Protocol (BGP), und zwar in der Version BGP4.

Internet: Stark vereinfachte schematische Darstellung



- Autonome Systeme können durch Upstream-Provider oder durch Peering miteinander verbunden sein.
- An Internet Exchange Points, z. B. DE-CIX, existieren zahlreiche Peering-Verbindungen.
- Peering-Verbindungen sind aus Kostengründen gegenüber Customer-Provider (C2P) Verbindungen zu bevorzugen.
- Die Border-Router eines AS „announcen“ Präfixe, die über dieses AS erreichbar sind.
- AS 5511 announced seine eigenen Customer an seine Peerings und Upstream-Provider.
- AS 5511 würde evtl. die Netze von Vodafone an Tele2 announcen, sicher aber nicht an Sprint oder AT&T.

Faustregel: Für vertikale Verbindungen muss der jeweilige Customer („kleinere Provider“) bezahlen, weshalb Horizontaler Verbindungen (Peerings) bevorzugt werden.

Übersicht

Nachfolge von IP(v4): IPv6

Wir haben bereits gesehen, dass

- der IPv4 Adressraum aus heutiger Sicht zu knapp bemessen ist und
- historisch bedingt Fehler bei der Adressvergabe gemacht wurden.

Außerdem ist die Verarbeitung des IPv4 Headers unnötig komplex:

- Header variabler Länge ($\Rightarrow$ Angabe der Header- und Datenlänge)
- Ungenutzte bzw. im Laufe der Zeit redefinierte Felder (TOS, DSCP, ECN)
- Fragmentierung und Reassemblierung bedeutet erheblichen Mehraufwand

IPv6 begegnet diesen Problemen:

- 128 Bit Adressraum $\Rightarrow 2^{128} \approx 10^{38}$ Adressen
  Das sind etwa $6.67 \cdot 10^{23}$ Adressen pro m^2 Erdoberfläche (mit Wasserflächen)!
- Header fester Länge für schnelle Verarbeitung
- Erweiterbarkeit durch sog. Extension Headers
- Keine Fragmentierung von IP-Paketen mehr (Anpassung der MTU beim Sender)
- Viele neue Features zur automatischen Konfiguration, Adressvergabe, Auffindung lokaler Gateways, etc.

Notation

IPv6 Adressen werden infolge ihrer Länge kompakt

- in hexadezimaler Schreibweise
- in Gruppen zu je 16 Bit
- getrennt durch : dargestellt.

Beispiel:

`2001:4ca0:2001:0011:2a37:37ff:fe02:3241/64`

- Wie IPv4-Adressen bestehen IPv4-Adressen aus einem Netz- und Hostanteil variabler Länge
- Die von IPv4-Adressen bekannte Präfixschreibweise wurde übernommen

Abkürzende Schreibweisen:

- In jedem Block von 16 Bit Länge können führende Nullen weggelassen werden
- Bestehen einer oder mehrere aufeinanderfolgende Blöcke nur aus Nullen, so können diese durch :: abgekürzt werden:

`fe80::2a37:37ff:fe02:41/64 = fe80:0000:0000:0000:2a37:37ff:fe02:3241/64`

IPv6-Header (ohne Extension Headers)

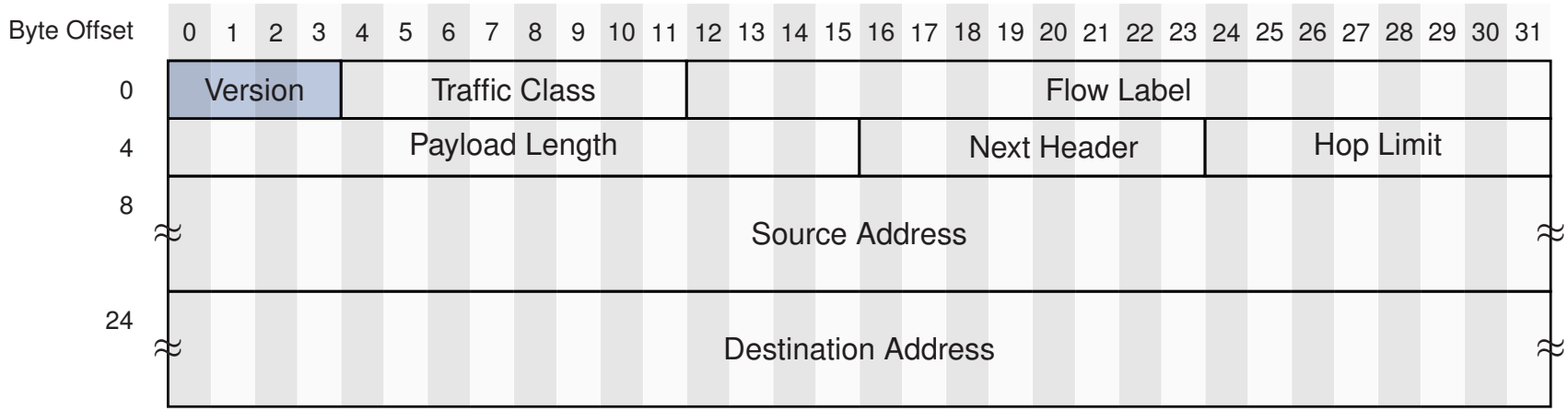


Abbildung: IPv6-Header (minimale Länge: 40 Byte)

Version

- Gibt die verwendete IP-Version an.
- Gültige Werte sind 6 (IPv6) und 4 (IPv4).

IPv6-Header (ohne Extension Headers)

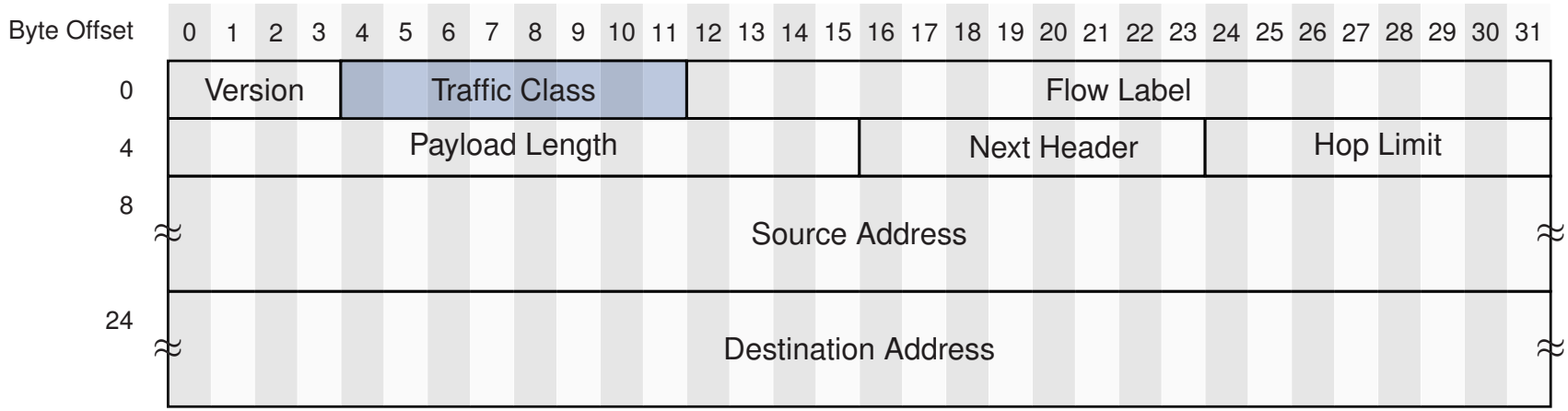


Abbildung: IPv6-Header (minimale Länge: 40 Byte)

Traffic Class

- Äquivalent zum TOS-Feld des IPv4 Headers.
- Wird zur Verkehrspriorisierung (QoS) eingesetzt.

IPv6-Header (ohne Extension Headers)

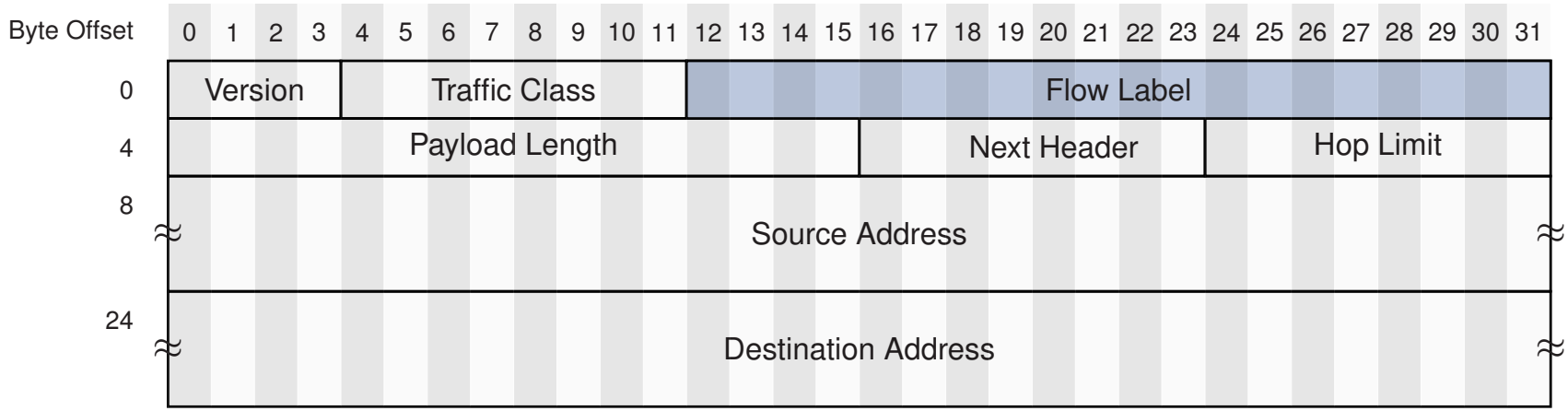


Abbildung: IPv6-Header (minimale Länge: 40 Byte)

Flow Label

- Ebenfalls zur Verkehrspriorisierung eingesetzt.
- Ermöglicht es, zu demselben Datenstrom (Flow) gehörende Pakete auf Schicht 3 zu identifizieren und gleich zu behandeln.

IPv6-Header (ohne Extension Headers)

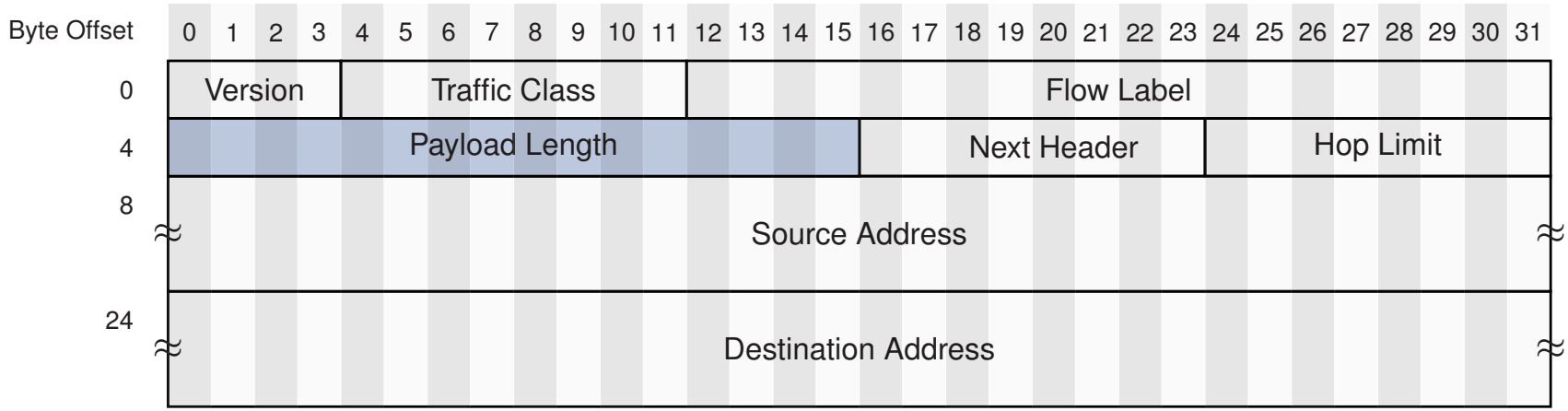


Abbildung: IPv6-Header (minimale Länge: 40 Byte)

Payload Length

- Länge der Daten inkl. möglicherweise vorhandener Extension Headers.
- Angabe in Vielfachen von 1 Byte.

IPv6-Header (ohne Extension Headers)

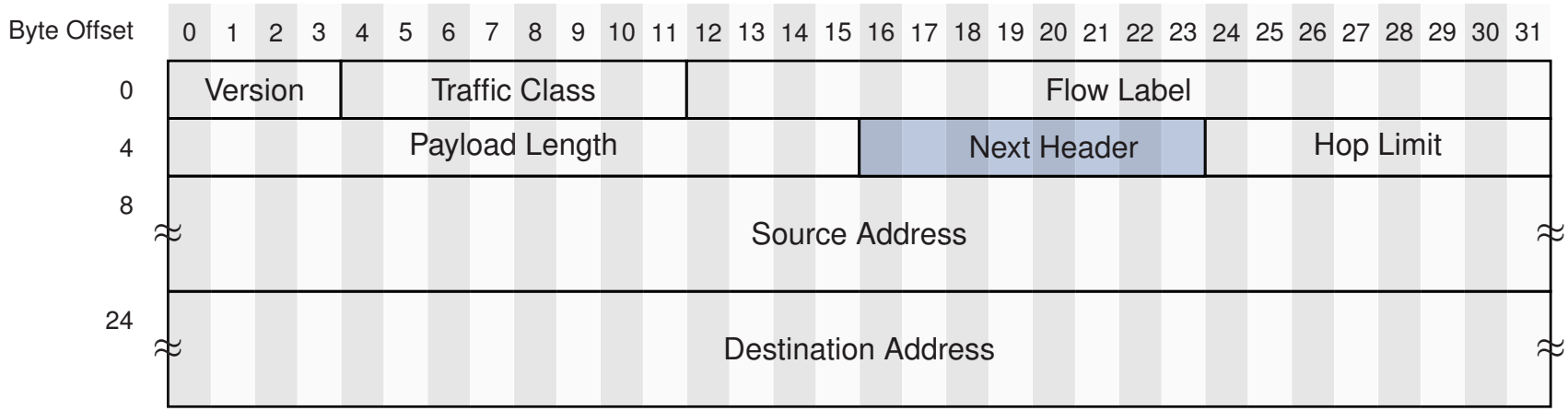


Abbildung: IPv6-Header (minimale Länge: 40 Byte)

Next Header

- Identifiziert den Typ des nächsten Headers.
- Dieser kann entweder ein Extension Header von IPv6 sein oder
- der Header der im Paket transportierten Daten (z. B. ICMP-, TCP-, UDP-Header).

IPv6-Header (ohne Extension Headers)

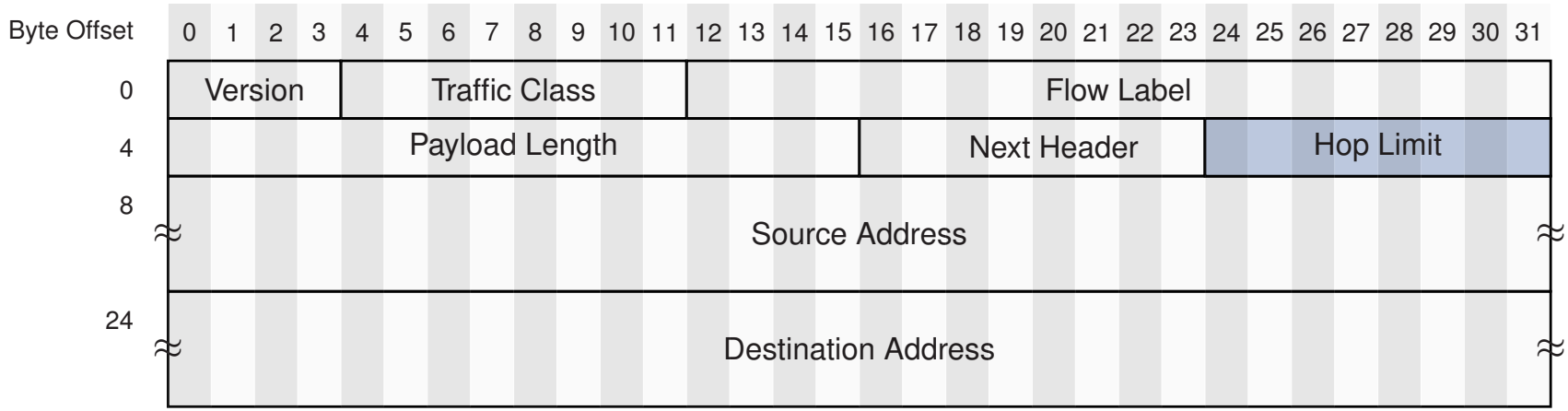


Abbildung: IPv6-Header (minimale Länge: 40 Byte)

Hop Limit

- Entspricht dem TTL-Feld des IPv4-Headers.
- Wird beim Weiterleiten des Pakets durch einen Router um 1 dekrementiert.
- Erreicht das Feld den Wert 0, wird das Paket verworfen und ein ICMPv6 Time Exceeded an den Absender zurückgeschickt.

IPv6-Header (ohne Extension Headers)

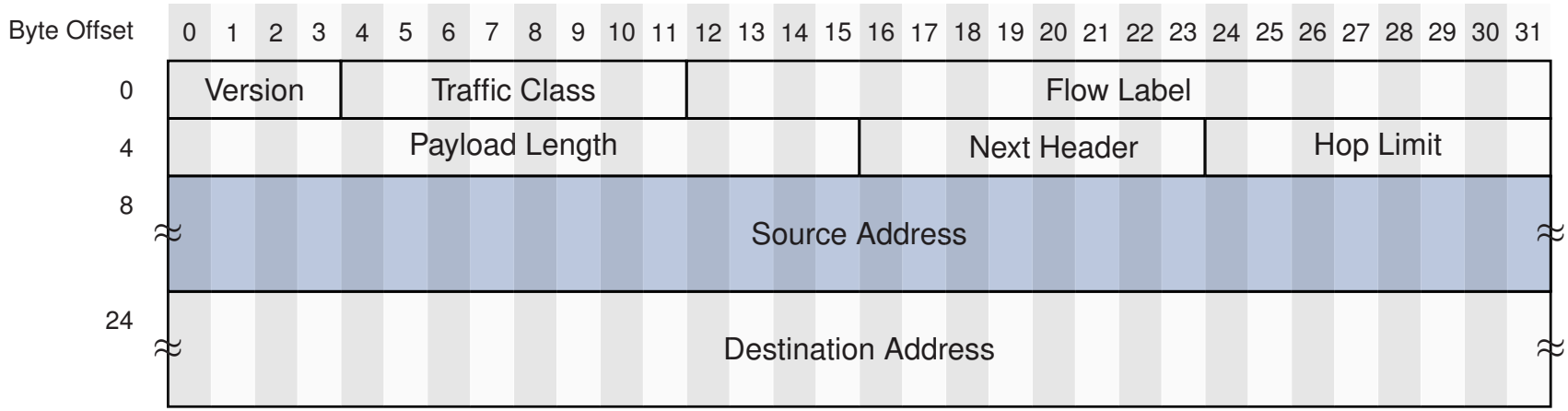


Abbildung: IPv6-Header (minimale Länge: 40 Byte)

Source Address

- 128 Bit (16 Byte) lange Absenderadresse.

IPv6-Header (ohne Extension Headers)

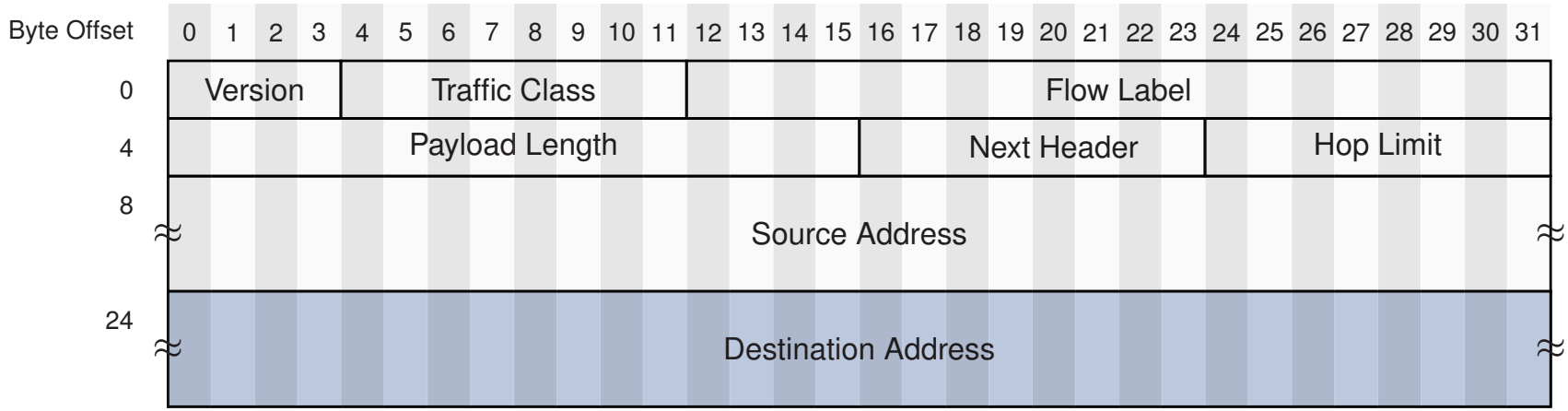


Abbildung: IPv6-Header (minimale Länge: 40 Byte)

Destination Address

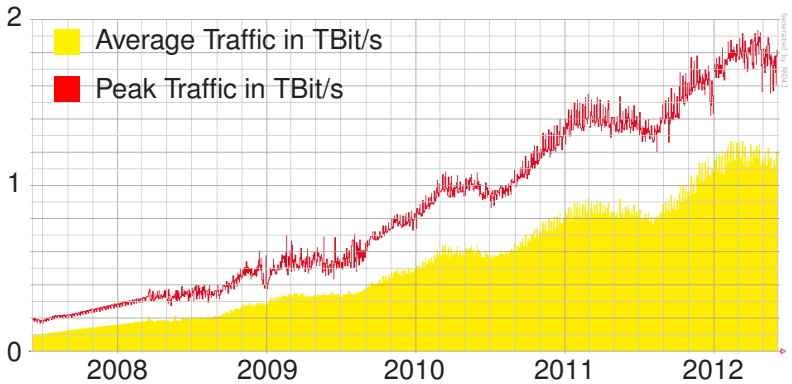
- 128 Bit (16 Byte) lange Zieladresse.

Kompatibilität:

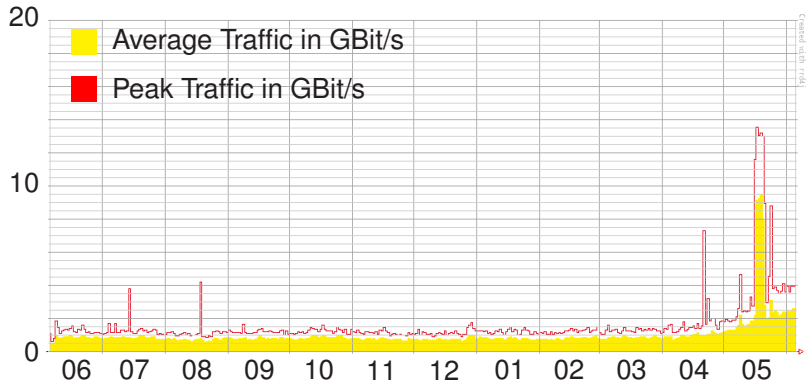
- IPv4 und IPv6 sind nicht kompatibel, können aber nebeneinander existieren
- Knoten verwenden heute häufig eine IPv4 und eine IPv6 Adresse
- Router müssen für beide Protokollversionen Routinginformationen getrennt voneinander austauschen und verarbeiten

Stand heute:

- Obwohl IPv6 bereits seit 1998 standardisiert ist (RFC 2460), ist die Umstellung auf IPv6 noch lange nicht abgeschlossen.
- Der mit Abstand größte Teil des weltweiten Datenverkehrs ist noch immer IPv4:



(a) IPv4-Verkehr 2008 - 2012



(b) IPv6-Verkehr 2011 - 2012

Abbildung: IPv4- und IPv6-Datenverkehr am DE-CIX [2]

Zusammenfassung

In diesem Kapitel haben wir

- die Vorteile von Paketvermittlung gegenüber Leitungs- und Nachrichtenvermittlung erarbeitet,
- die Notwendigkeit logischer Adressen zur End-zu-End Adressierung erkannt,
- zwei unterschiedliche Protokolle zur End-zu-End Adressierung im Internet kennen gelernt,
- Methoden zur weiteren logischen Unterteilung von Netzen in Subnetze kennengelernt und
- ein grundlegendes Verständnis bzgl. des Austauschs von Routinginformationen im Internet entwickelt.

Was wir wissen sollten:

- Was sind die Unterschiede zwischen Leitungs-, Nachrichten- und Paketvermittlung?
- Worin besteht der technische und logische Unterschied zwischen MAC- und IP-Adressen?
- Wie werden IP-Adressen in Netz- und Hostanteil aufgeteilt?
- Wie werden IP-Adressen in MAC-Adressen übersetzt?
- Was ist eine Routing Tabelle?
- Wie treffen Hosts und Router Weiterleitungsentscheidungen (Forwarding)?
- Wie tauschen Router untereinander Routinginformationen aus?
- Welche grundlegenden Typen von Routingprotokollen gibt es?
- Was sind die wesentlichen Unterschiede zwischen IPv4 und IPv6?

Literaturhinweise und Quellenangaben I

[1] CAIDA: IPv4 Census Map.
`http://www.caida.org/research/id-consumption/#ipv4-census-map`.

[2] DE-CIX: IP Traffic Statistics.
`http://www.de-cix.net/about/statistics`.

[3] Peterson, L. L. und S. Davie B.: Computer Networks – A System Approach, Kapitel Internetworking, Seiten 234 – 242.
Elsevier, 4. Auflage, 2007.
Auszug s. Moodle/SVN.

[4] Peterson, L. L. und S. Davie B.: Computer Networks – A System Approach, Kapitel Internetworking, Seiten 248 – 256.
Elsevier, 4. Auflage, 2007.
Auszug s. Moodle/SVN.

[5] Peterson, L. L. und S. Davie B.: Computer Networks – A System Approach, Kapitel Internetworking, Seiten 259 – 262.
Elsevier, 4. Auflage, 2007.
Auszug s. Moodle/SVN.