Grundlagen Rechnernetze und Verteilte Systeme (GRNVS)

IN0010 – SoSe 2026

Prof. Dr.-Ing. Georg Carle

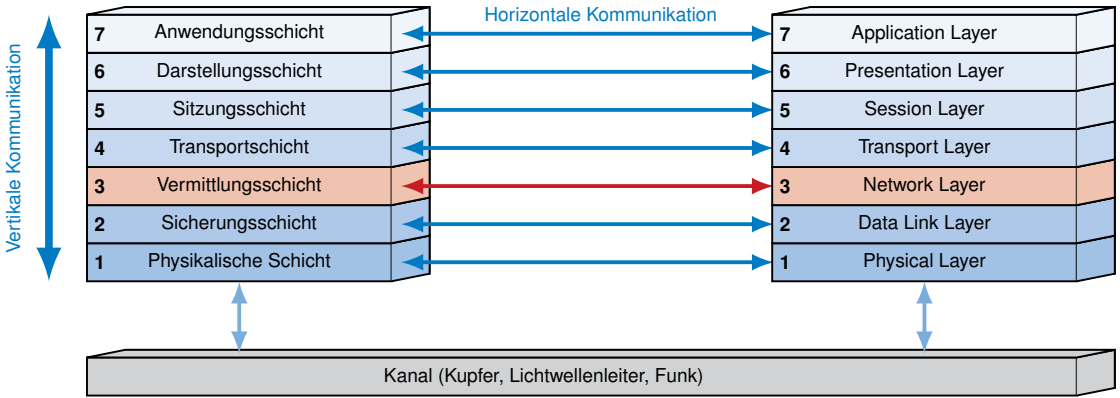
Lorenz Lehle, Christian Dietze, Stefan Lachnit, Manuel Simon, Markus Sosnowski

Lehrstuhl für Netzarchitekturen und Netzdienste
School of Computation, Information and Technology
Technische Universität München

Kapitel 3: Vermittlungsschicht

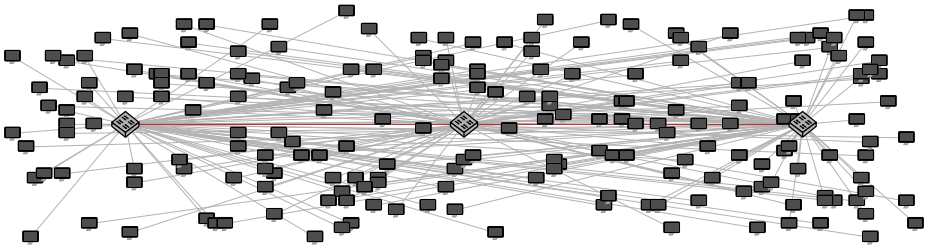
Kapitel 3: Vermittlungsschicht

Einordnung im ISO/OSI-Modell



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

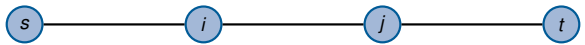
Kapitel 3: Vermittlungsschicht

Vermittlungsarten

Es gibt drei grundlegende Vermittlungsarten:

1. Leitungsvermittlung
   „Reserviere eine dedizierte Leitung zwischen Sender und Empfänger“
2. Nachrichtenvermittlung
   „Wähle für jede Nachricht individuell einen Weg und leite die Nachricht als Ganzes weiter“
3. 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 (i und j) hinsichtlich der Gesamtdauer T einer Übertragung von L Datenbits über die Distanz d und motivieren so die Vorteile der Paketvermittlung.

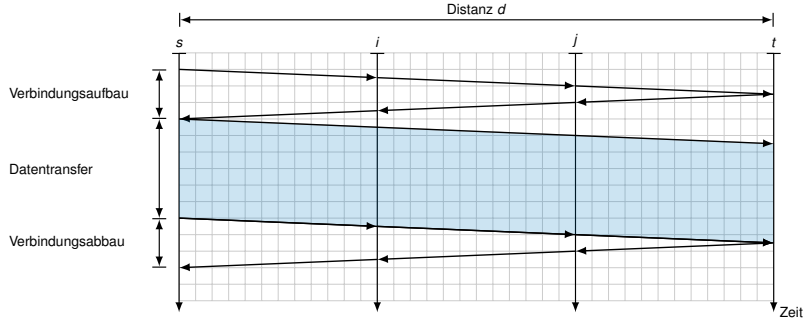
Während einer verbindungsorientierten Übertragung können drei Phasen unterschieden werden:

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.



$$T_{LV} = 2t_p + t_s + 2t_p = t_s + 4t_p = \frac{L}{r} + \frac{4d}{\nu c_0}$$

Vorteile der Leitungsvermittlung

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

Nachteile der Leitungsvermittlung

- Ressourcenverschwendung sofern Leitung nicht dauerhaft ausgelastet wird, 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 beim Schalten physikalischer Verbindungen

Einsatz in heutigen Netzwerken

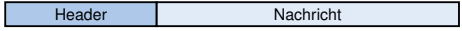
- Leitungsvermittlung wird häufig durch Paketvermittlung ersetzt (z. B. Voice over IP)
- In vielen Vermittlungsnetzen wird Leitungsvermittlung zumindest virtualisiert in Form von Virtual Circuits unterstützt (z. B. Frame Relay, ATM[1], MPLS[2])

[1] Asynchronous Transfer Mode

[2] Multi Protocol Label Switching

Modifikationen gegenüber Leitungsvermittlung:

- 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

Eigenschaften:

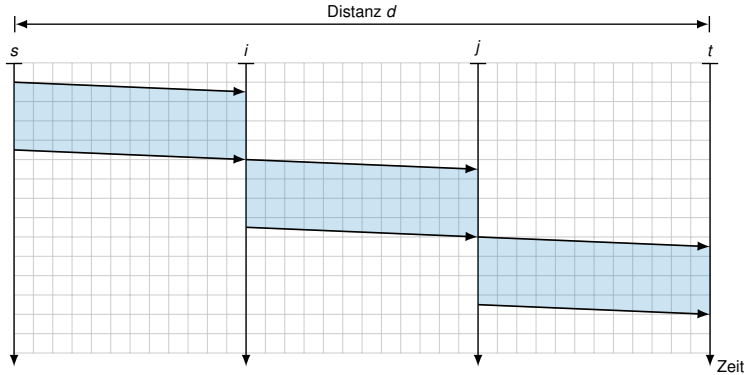
- Möglichkeit für asynchrone Kommunikation, d. h. Nachrichten können ggf. an Empfänger versendet werden, die zum Zeitpunkt des Sendens nicht empfangsbereit sind
- Mögliche Zeitersparnis, da die Phasen zum Aufbau und Abbau der Verbindung entfallen

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

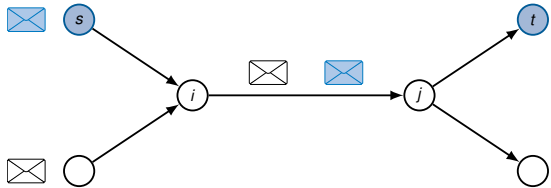
Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.



$$T_{NV} = (n+1) \cdot t_s + t_p = (n+1) \cdot \frac{L_H + L}{r} + \frac{d}{\nu c_0}$$

Multiplexing auf Nachrichtenebene

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



Vorteile:

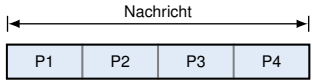
- Flexibles Zeitmultiplex von Nachrichten
- Bessere Ausnutzung der Kanalkapazität
- Keine Verzögerung beim Senden der Nachricht durch Verbindungsaufbau

Nachteile:

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

Unterschiede zur Nachrichtenvermittlung:

- Nachrichten werden nicht mehr als Einheit übertragen sondern in kleinere Einheiten, den Datenteilen von Paketen, unterteilt:



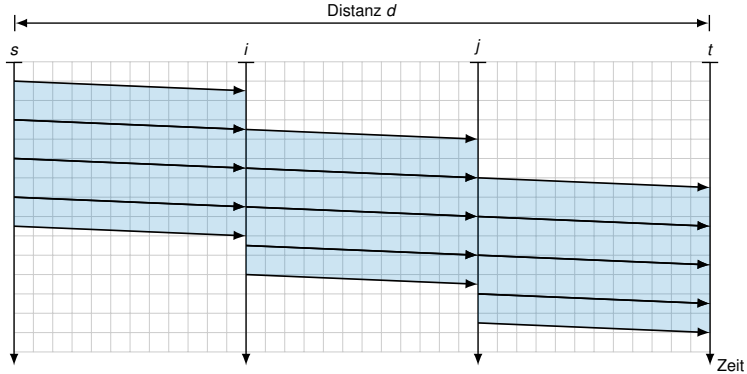
- Jedes Paket wird mit einem eigenen Header versehen, der alle Informationen zur Weiterleitung und ggf. auch zur Reassemblierung enthält:

- 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 maximale Paketgröße einhergehend mit Datenteilen maximaler Länge $p_{\max}$.

Übertragungszeit bei Paketvermittlung

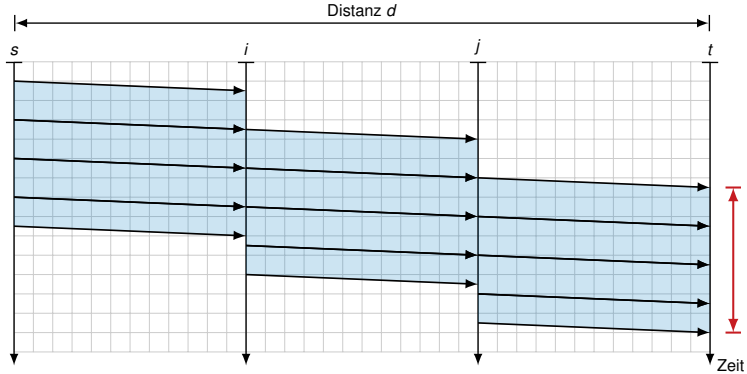
- Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.
- Die Nachrichtenlänge L muss kein Vielfaches von $p_{\max}$ sein.



$$T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c_0} + n \cdot \frac{L_h + p_{\max}}{r}$$

Übertragungszeit bei Paketvermittlung

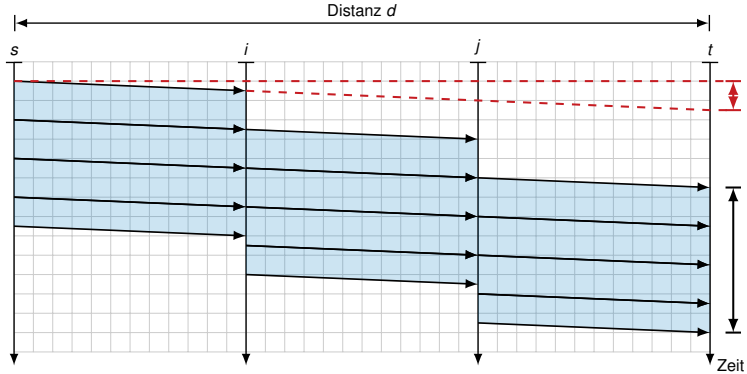
- Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.
- Die Nachrichtenlänge L muss kein Vielfaches von $p_{\max}$ sein.



$$T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c_0} + n \cdot \frac{L_h + p_{\max}}{r}$$

Übertragungszeit bei Paketvermittlung

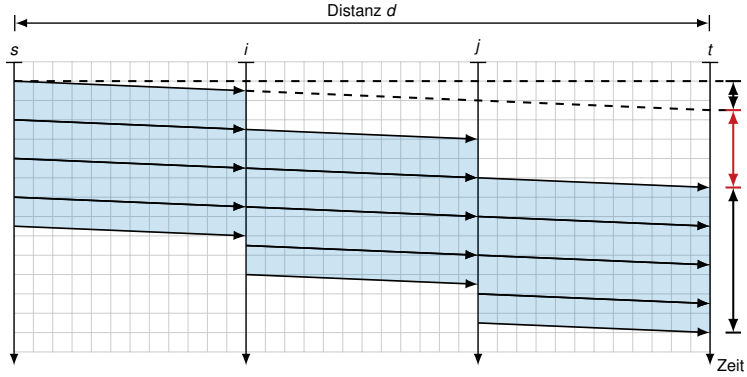
- Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.
- Die Nachrichtenlänge L muss kein Vielfaches von $p_{\max}$ sein.



$$T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c_0} + n \cdot \frac{L_h + p_{\max}}{r}$$

Übertragungszeit bei Paketvermittlung

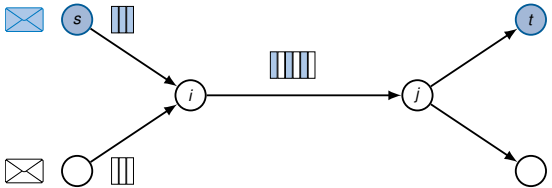
- Erinnerung: Hier $n = 2$ Vermittlungsknoten i und j.
- Die Nachrichtenlänge L muss kein Vielfaches von $p_{\max}$ sein.



$$T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c_0} + n \cdot \frac{L_h + p_{\max}}{r}$$

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

$$\text{Leitungsvermittlung:} \quad T_{LV} = \frac{L}{r} + 4\frac{d}{\nu c_0}$$

$$\text{Nachrichtenvermittlung:} \quad T_{NV} = (n+1)\frac{L_h + L}{r} + \frac{d}{\nu c_0}$$

$$\text{Paketvermittlung:} \quad T_{PV} = \frac{1}{r}\left(\left\lceil \frac{L}{p_{\max}} \right\rceil \cdot L_h + L\right) + \frac{d}{\nu c_0} + n \cdot \frac{L_h + p_{\max}}{r}$$

Zahlenbeispiel:

- Die Distanz zwischen Sender und Empfänger beträgt $d = 1000\,\text{km}$
- Für Glasfaserleitungen beträgt $\nu = 0.7$
- Es kommen $n = 2$ Vermittlungsknoten zum Einsatz
- Die Datenrate beträgt auf allen Teilstrecken $r = 777\,\text{kbit/s}$
- Die Länge der zu sendenden Nachricht beträgt $L = 5\,\text{MiB}$
- Die maximale Nutzlastgröße betrage $p_{\max} = 1480\,\text{B}$
- Die Headergröße pro Nachricht / Paket betrage $L_h = 20\,\text{B}$

Es ergeben sich folgende Zahlenwerte: $T_{LV} \approx 54\,\text{s}$, $T_{NV} \approx 162\,\text{s}$ und $T_{PV} \approx 55\,\text{s}$.

Paketvermittlung

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 4 aufwärts), z. B. nachrichtenorientierte Transportprotokolle wie UDP oder SCTP oder Anwendungsprotolle wie SMTP (Simple Mail Transfer Protocol)

Paketvermittlung:

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

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

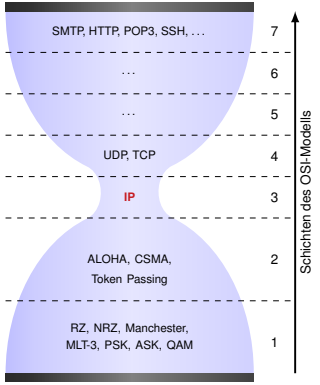
- Möglichkeiten zur global eindeutigen und strukturierten / logischen Adressierung sowie
- Verfahren zur Bestimmung von (möglichst) optimalen Pfaden.

Wir beschränken uns in diesem Teilkapitel auf die Betrachtung von

- IPv4 (Internet Protocol v4, 1981) bzw.
- seinem Nachfolger IPv6 (1998).

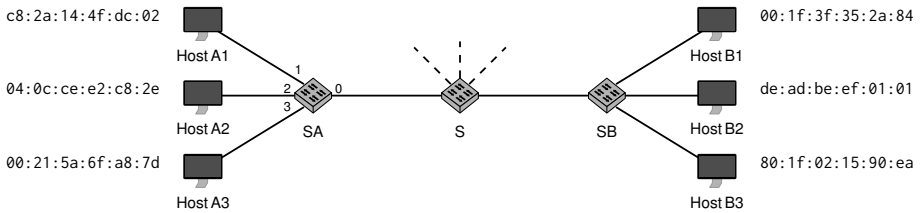
Beispiele für alternative Protokolle der Netzwerkschicht:

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

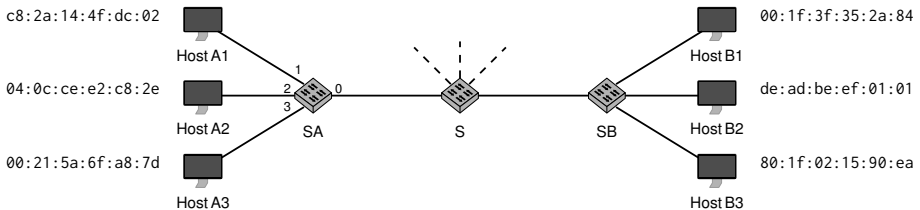


Internet Protocol version 4 (IPv4) [14]

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



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



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

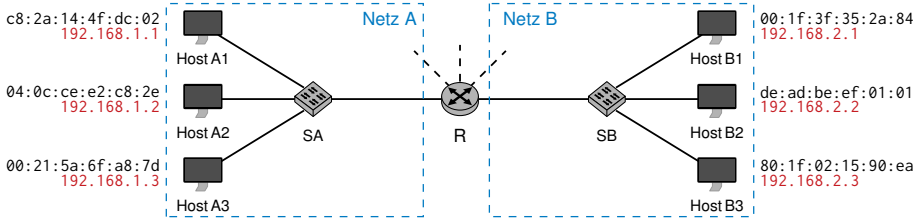
- Genau einen Eintrag für jeden bekannten Host
- Die meisten MAC-Adressen werden auf Port 0 abgebildet
- Eine Zusammenfassung von Einträgen ist i. A. nicht möglich, da MAC-Adressen nicht die Position eines Knotens innerhalb eines Netzwerks wiederspiegeln

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

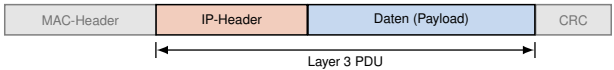
Internet Protocol version 4 (IPv4) [14]

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:



Internet Protocol version 4 (IPv4) [14]

IPv4-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.

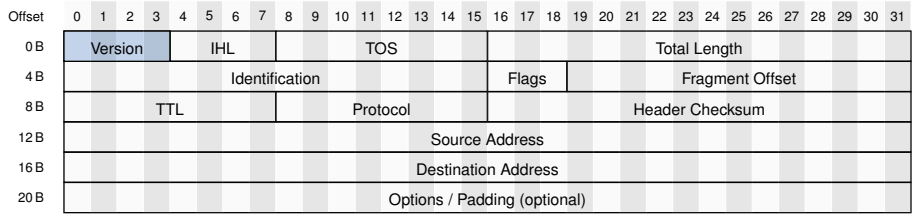


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

Version

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

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

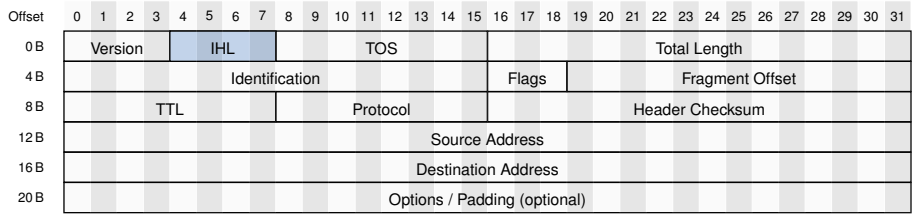


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

IHL (Internet 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.

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

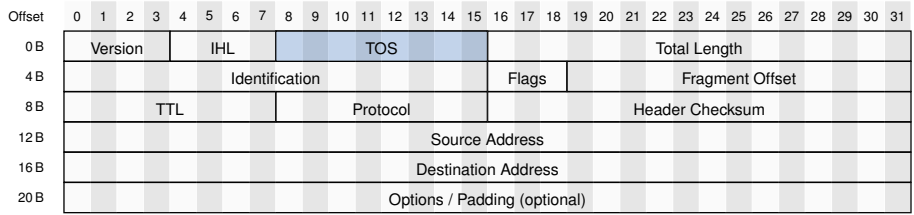


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

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).

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

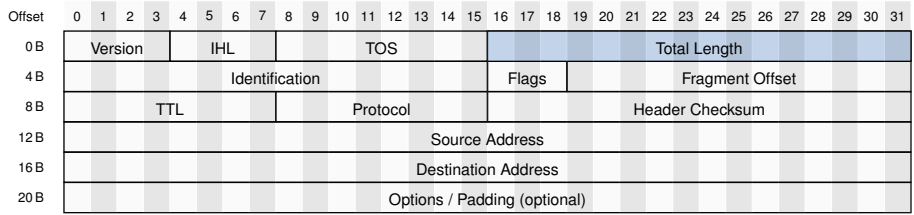


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

Total Length

- Gibt die Gesamtlänge des IP-Pakets (Header + Daten) in Bytes an.
- Die Maximallänge eines IP-Pakets beträgt damit 65 535 B.
- 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 B.

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

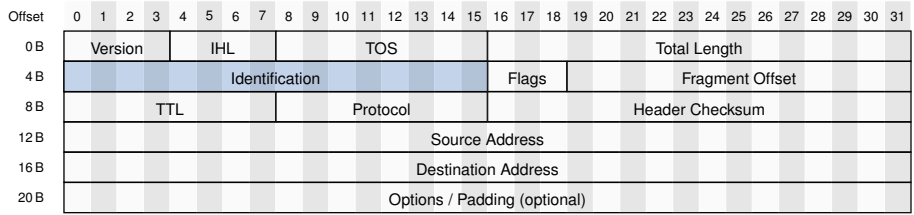


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

Identification

- Für jedes IP-Paket (zufällig) gewählter 16 bit langer Wert.
- Dient der Identifikation zusammengehörender Fragmente (IP-Fragmentierung → später).

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

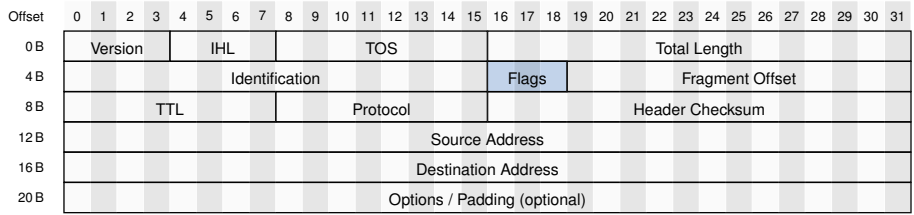


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

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.

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

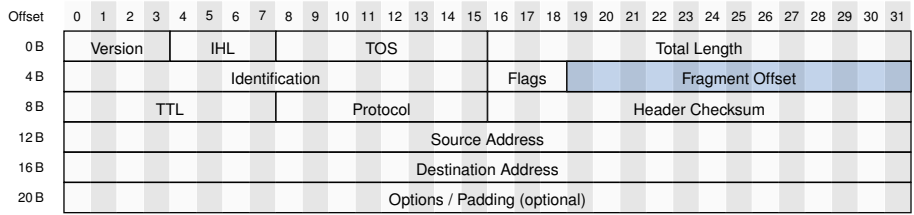


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

Fragment Offset

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

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

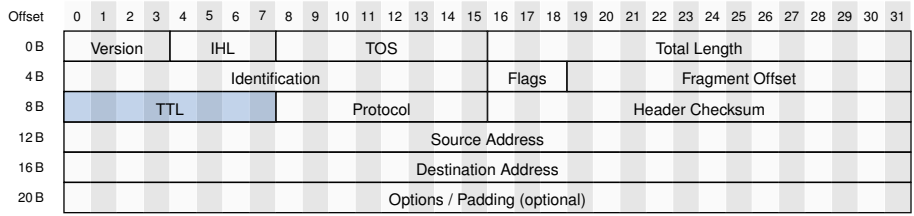


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

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.

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

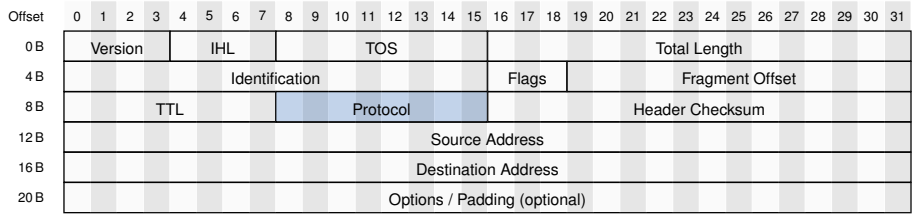


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

Protocol

- Identifiziert das Protokoll auf Schicht 4, welches in der Payload (Datenteil) 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 `0x11` (UDP).

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

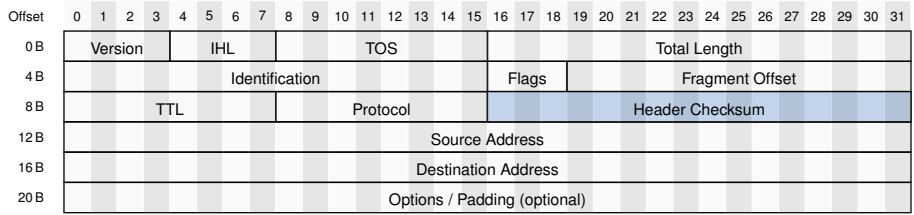


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

Header Checksum

- Einfache, auf Geschwindigkeit optimierte Prüfsumme, welche nur den IP-Header (ohne Daten) schützt.
- Die Prüfsumme ist so ausgelegt, dass die Dekrementierung des TTL-Felds einer Inkrementierung der Prüfsumme entspricht. Es ist also keine komplette Neuberechnung der Prüfsumme bei der Weiterleitung von Paketen notwendig, lediglich eine Inkrementierung +1.
- Es ist lediglich Fehlererkennung aber keine Korrektur möglich.

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

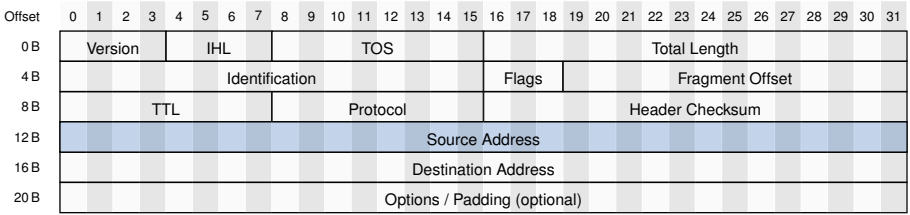


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

Source Address

- IP-Adresse des Absenders.

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

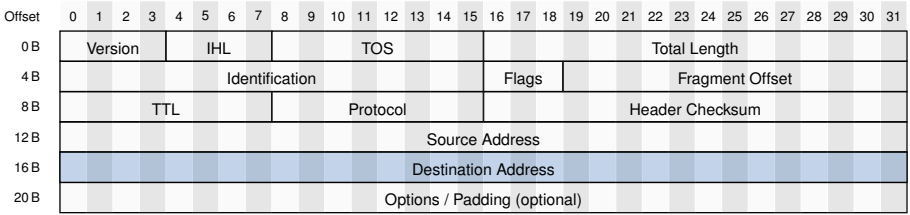


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

Destination Address

- IP-Adresse des Empfängers.

Internet Protocol version 4 (IPv4) [14]

IPv4-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.

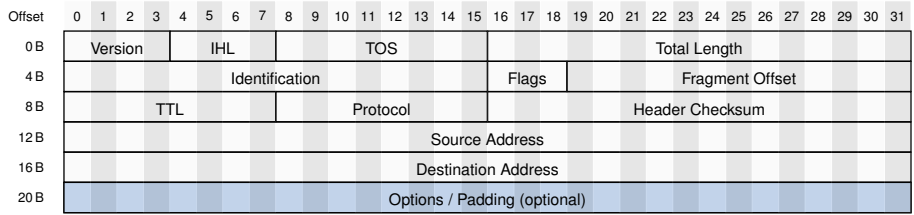


Abbildung 1: IPv4-Header (minimale Länge: 20 B)

Options / Padding

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

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

Es gibt zwei unterschiedle Byte-Orders:

1. Big Endian: „Niederwertigstes Byte steht an höchstwertigster Adresse“
   Intuitive 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`.

Netzwerkprotokolle haben zum Ziel, Kommunikation zwischen Hosts unabhängig von deren Byte-Order zu ermöglichen, und schreiben deshalb eine Byte-Order vor, die Network Byte Order (Big Endian).

x86-kompatible Computer verwenden intern Little Endian. Bei der Kommunikation mit anderen Computern ist deswegen eine Konvertierung in Network Byte Order erforderlich.

Network Byte Order
Vor dem Versenden 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.

Konvertierung von Host-Byte-Order und Network-Byte-Order

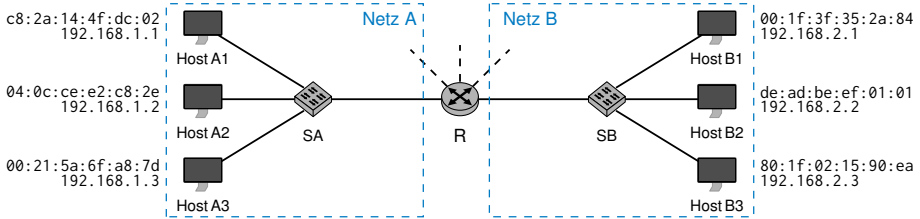
- Dies betrifft z. B. die Felder Total Length, Identification, Flags+Fragment Offset und Header Checksum aus dem IP-Header.
- Nicht betroffen sind 1 B lange Felder. Quell- und Zieladresse liegen gewöhnlich als Array von 1 B langen Werten vor, weswegen auch diese kein Problem darstellen.

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“

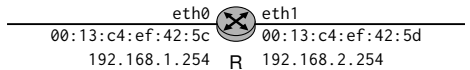
Für 32 bit lange Werte gibt es analog `htonl()` und `ntohl()` („`l`“ für den Datentyp `long`).

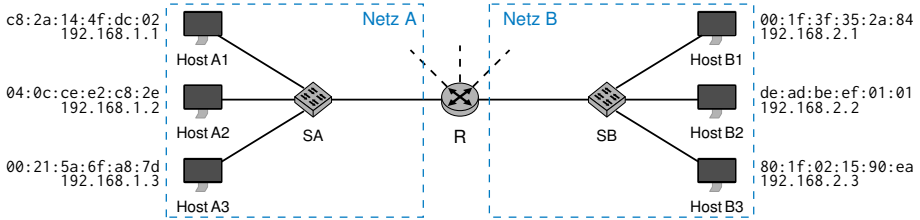
Zurück zu unserem Beispiel:



- Innerhalb von Netz A erreichen sich alle Hosts weiterhin direkt über SA.
- Will Host A1 nun ein Paket an Host B2 schicken, so geht das nicht mehr direkt:
  - Die Weiterleitung von A nach B erfolgt über R **auf Basis von IP-Adressen**
  - Host A1 muss also dafür sorgen, dass R das Paket erhält
  - Dazu verwendet Host A1 als Ziel-MAC-Adresse die des lokalen Interfaces von R, als Ziel-IP-Adresse wird die IP-Adresse von Host B2 verwendet

⇒ 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:





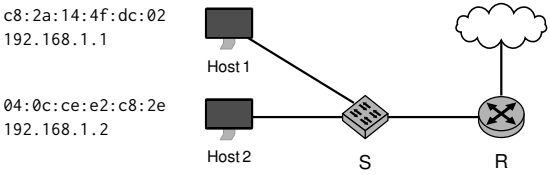
Offene Probleme:

- Angenommen der Sender kennt nur die IP-Adresse des Ziels. Wie kann die MAC-Adresse des Next-Hops bestimmt werden? (Adressauflösung)
- Woher weiß Host A1, 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? (Genauer: Den richtigen Next Hop?) (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? (Classful Routing, Classless Routing, Subnetting)
- Woher kennt der Sender überhaupt die IP-Adresse des Ziels? (DNS → Schicht 7)

Internet Protocol version 4 (IPv4) [14]

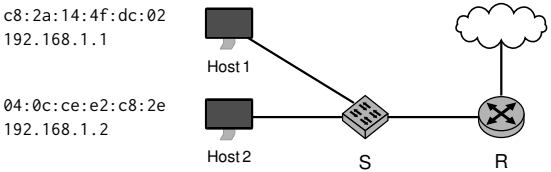
Adressauflösung [15]

- Host 1 will eine Nachricht an Host 2 senden
- Die IP-Adresse von Host 2 (192.168.1.2) sei ihm bereits bekannt
- Wie erhält Host1 die zugehörige MAC-Adresse?



Adressauflösung [15]

- Host 1 will eine Nachricht an Host 2 senden
- Die IP-Adresse von Host 2 (192.168.1.2) sei ihm bereits bekannt
- Wie erhält Host1 die zugehörige MAC-Adresse?



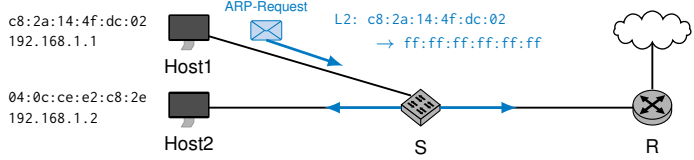
Address Resolution Protocol (ARP)

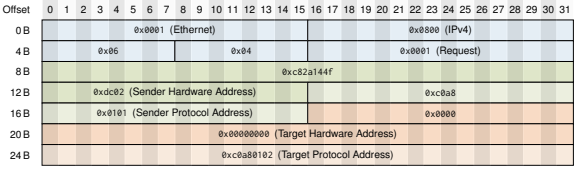
1. Host1 sendet einen ARP-Request: „Who has 192.168.1.2? Tell 192.168.1.1 at `c8:2a:14:4f:dc:02`“
2. Host2 antwortet mit einem ARP-Reply: „192.168.1.2 is at `04:0c:ce:e2:c8:2e`“

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 2: ARP-Paket für IPv4 über Ethernet

Beispiel: (`L2: xx:xx:xx:xx:xx:xx` → `yy:yy:yy:yy:yy:yy` stellt Quell- und Ziel-MAC-Adresse auf Schicht 2 dar.)

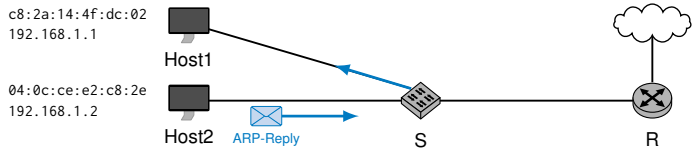


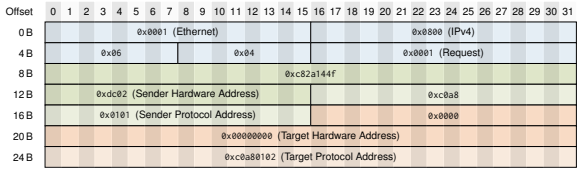


(a) ARP Request

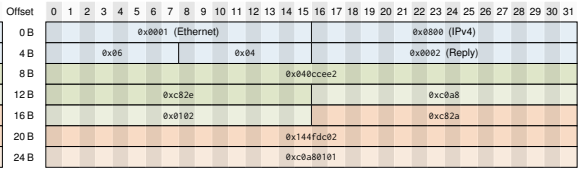
- Der ARP-Request wird an die MAC-Broadcast-Adresse `ff:ff:ff:ff:ff:ff` geschickt, weswegen der Switch S den Rahmen an alle angeschlossenen Hosts weiterleitet.

Beispiel: (`L2: xx:xx:xx:xx:xx:xx → yy:yy:yy:yy:yy:yy` stellt Quell- und Ziel-MAC-Adresse auf Schicht 2 dar.)





(c) ARP Request



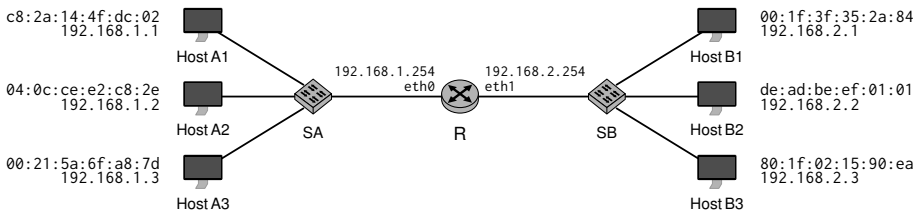
(d) ARP Reply

- Der ARP-Request wird an die MAC-Broadcast-Adresse `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).

Internet Protocol version 4 (IPv4) [14]

Was ist nun, wenn das Ziel nicht im selben Netz liegt (z. B. Host A1 an Host B2)?

- Jeder Host sollte einen Router zum Internet, das sog. Default Gateway, kennen, an das er alle Pakete schickt, deren Zieladressen nicht im eigenen Netz liegen, und für die in seiner Routing-Tabelle nicht ein spezifisches Gateway eingetragen ist.
- Ob eine Zieladresse zum eigenen Netz gehört erkennt ein Host durch Vergleich der Zieladresse mit der eigenen Netzadresse.
- 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.

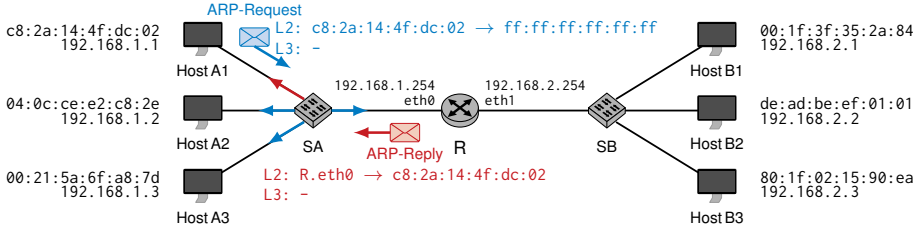


1. Host A1 erkennt, dass 192.168.2.2 nicht im eigenen Netz liegt. Sein Default-Gateway ist 192.168.1.254.

Internet Protocol version 4 (IPv4) [14]

Was ist nun, wenn das Ziel nicht im selben Netz liegt (z. B. Host A1 an Host B2)?

- Jeder Host sollte einen Router zum Internet, das sog. Default Gateway, kennen, an das er alle Pakete schickt, deren Zieladressen nicht im eigenen Netz liegen, und für die in seiner Routing-Tabelle nicht ein spezifisches Gateway eingetragen ist.
- Ob eine Zieladresse zum eigenen Netz gehört erkennt ein Host durch Vergleich der Zieladresse mit der eigenen Netzadresse.
- 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.

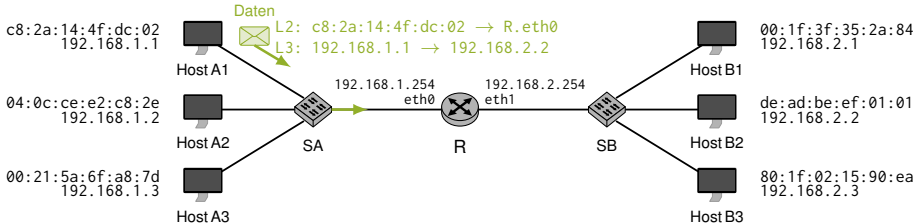


1. Host A1 erkennt, dass 192.168.2.2 nicht im eigenen Netz liegt. Sein Default-Gateway ist 192.168.1.254.
2. Host A1 löst die MAC-Adresse zu 192.168.1.254 auf.

Internet Protocol version 4 (IPv4) [14]

Was ist nun, wenn das Ziel nicht im selben Netz liegt (z. B. Host A1 an Host B2)?

- Jeder Host sollte einen Router zum Internet, das sog. Default Gateway, kennen, an das er alle Pakete schickt, deren Zieladressen nicht im eigenen Netz liegen, und für die in seiner Routing-Tabelle nicht ein spezifisches Gateway eingetragen ist.
- Ob eine Zieladresse zum eigenen Netz gehört erkennt ein Host durch Vergleich der Zieladresse mit der eigenen Netzadresse.
- 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.

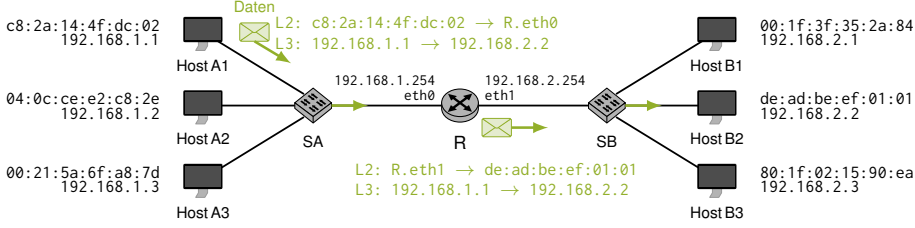


1. Host A1 erkennt, dass 192.168.2.2 nicht im eigenen Netz liegt. Sein Default-Gateway ist 192.168.1.254.
2. Host A1 löst die MAC-Adresse zu 192.168.1.254 auf.
3. Host A1 sendet das Datenpaket an R: Dabei adressiert er R mittels der eben bestimmten MAC-Adresse (Schicht 2). Als Ziel-IP-Adresse (Schicht 3) verwendet er die IP-Adresse von Host B2.

Internet Protocol version 4 (IPv4) [14]

Was ist nun, wenn das Ziel nicht im selben Netz liegt (z. B. Host A1 an Host B2)?

- Jeder Host sollte einen Router zum Internet, das sog. Default Gateway, kennen, an das er alle Pakete schickt, deren Zieladressen nicht im eigenen Netz liegen, und für die in seiner Routing-Tabelle nicht ein spezifisches Gateway eingetragen ist.
- Ob eine Zieladresse zum eigenen Netz gehört erkennt ein Host durch Vergleich der Zieladresse mit der eigenen Netzadresse.
- 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. Host A1 erkennt, dass 192.168.2.2 nicht im eigenen Netz liegt. Sein Default-Gateway ist 192.168.1.254.
2. Host A1 löst die MAC-Adresse zu 192.168.1.254 auf.
3. Host A1 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 Host B2.
4. R akzeptiert das Paket, bestimmt das ausgehende Interface und leitet das Paket weiter an Host B2. Dabei adressiert R wiederum Host B2 anhand seiner MAC-Adresse (erfordert ggf. einen weiteren ARP-Schritt).

Merke:

- MAC-Adressen dienen zur Adressierung innerhalb eines (Direktverbindungs-)Netzes und werden beim Forwarding durch einen Router verändert.
- IP-Adressen dienen der End-zu-End-Adressierung zwischen mehreren (Direktverbindungs-)Netzen und werden beim Forwarding durch einen Router 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 (z.B. 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 Protocol version 4 (IPv4) [14]

Internet Control Message Protocol (ICMP) [16]

Das IP-Protokoll unterstützt das Senden von Paketen an Ziel-Hosts. Dabei kann es zu Fehlern kommen, z. B.

- ein Paket gerät in 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).

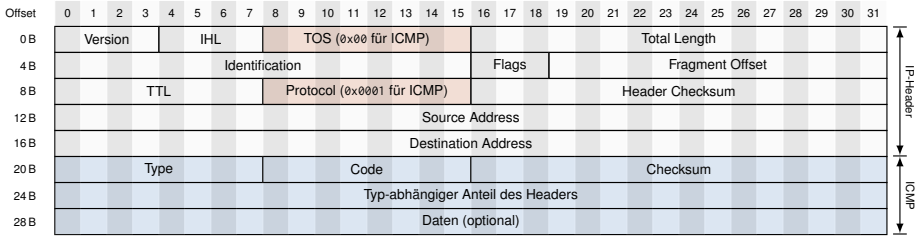
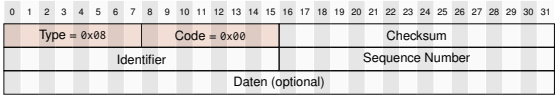
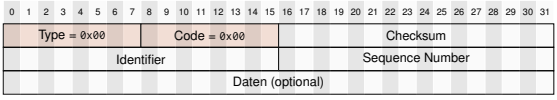


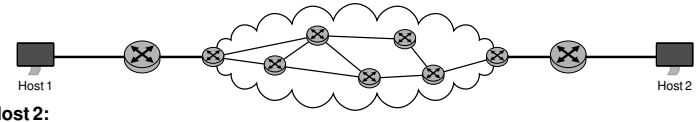
Abbildung 3: Allgemeiner ICMP-Header mit vorangestelltem IP-Header



(a) ICMP Echo Request



(b) ICMP Echo Reply



„Ping“ von Host 1 nach Host 2:

- Host 1 wählt einen zufälligen Identifier (16 bit), die Sequenznummer wird für jeden gesendeten Echo-Request um eins inkrementiert.
- Der Echo Request wird von Routern wie jedes IP-Paket weitergeleitet.
- Erhält Host 2 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 Host 2 fehlschlagen, so wird eine ICMP-Nachricht mit dem entsprechenden Fehlercode an Host 1 zurückgeschickt.

Frage: Wozu dient der Identifier?

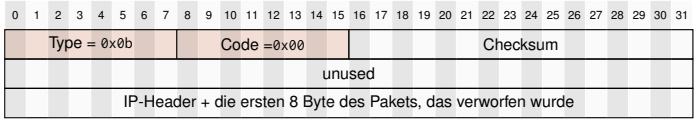
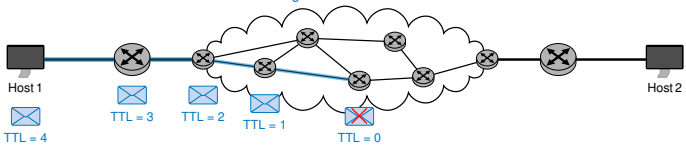


Abbildung 4: ICMP TTL-Exceeded



- Der IP-Header besitzt ein TTL-Feld, welches bei der Weiterleitung eines Pakets durch den jeweiligen Router um 1 dekrementiert wird.
- Erreicht es den Wert 0, so wird das betreffende Paket verworfen.

Internet Control Message Protocol (ICMP) [16] – ICMP Time Exceeded

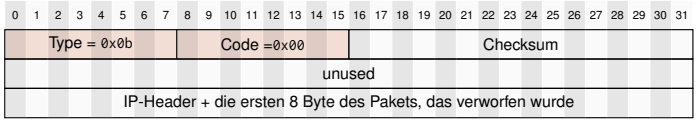
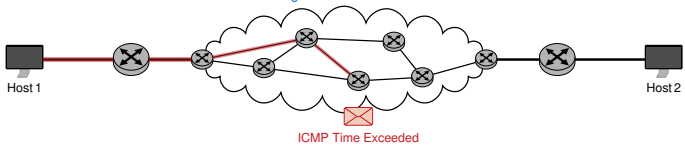


Abbildung 4: ICMP TTL-Exceeded



- Der IP-Header besitzt ein TTL-Feld, welches bei der Weiterleitung eines Pakets durch den jeweiligen Router um 1 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.

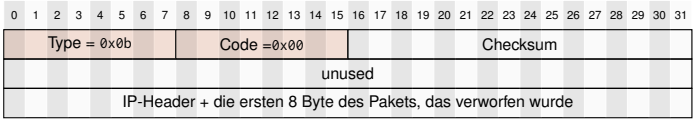
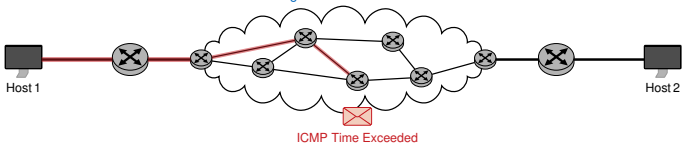


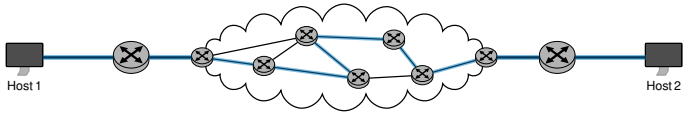
Abbildung 4: ICMP TTL-Exceeded



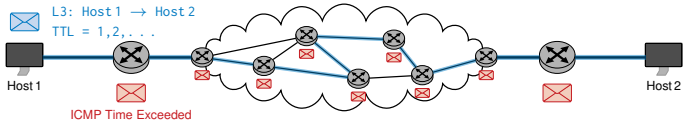
- Der IP-Header besitzt ein TTL-Feld, welches bei der Weiterleitung eines Pakets durch den jeweiligen Router um 1 dekrementiert wird.
- Erreicht es den Wert 0, so wird das betreffende Paket verworfen.
- Der Router generiert ein ICMP Time Exceeded und schickt es an den Absender des verworfenen Pakets zurück.

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

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



- Obwohl Pakete zwischen Host 1 und Host 2 (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 Host 1 nach Host 2?

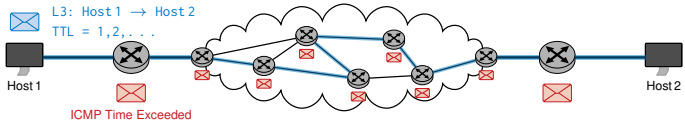


- Obwohl Pakete zwischen Host 1 und Host 2 (in Hin- und Rückrichtung) jeweils unterschiedliche Wege nehmen können, werden in der Praxis meist nur einer oder sehr wenige genutzt.
- Welchen Pfad nehmen Pakete von Host 1 nach Host 2?

Traceroute: Host 1 sendet z. B. ICMP Echo Requests an Host 2

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

⇒ Router entlang des Pfads von Host 1 nach Host 2 werden schrittweise die Pakete verwerfen und jeweils ein TTL Exceeded an Host 1 zurücksenden. Anhand der IP-Quelladresse dieser Fehlernachrichten kann Host 1 den Pfad hin zu Host 2 nachvollziehen.



- Obwohl Pakete zwischen Host 1 und Host 2 (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 Host 1 nach Host 2?

Traceroute: Host 1 sendet z. B. ICMP Echo Requests an Host 2

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

⇒ Router entlang des Pfads von Host 1 nach Host 2 werden schrittweise die Pakete verwerfen und jeweils ein TTL Exceeded an Host 1 zurücksenden. Anhand der IP-Quelladresse dieser Fehlernachrichten kann Host 1 den Pfad hin zu Host 2 nachvollziehen.

⇒ 3. Programmieraufgabe: „Implementiere Traceroute“

Beispiel:

```
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:

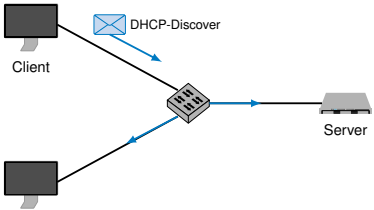
- Ein Router hat mehrere IP-Adressen. Welche wählt er als Absender-Adresse für die Fehlerbenachrichtigungen? Wählt er immer dieselbe Adresse?
- Was ist, wenn es tatsächlich mehrere gleichzeitig genutzte Pfade oder Pfadabschnitte gibt?
- Müssen Router überhaupt Fehlerbenachrichtigungen versenden?
- Angenommen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

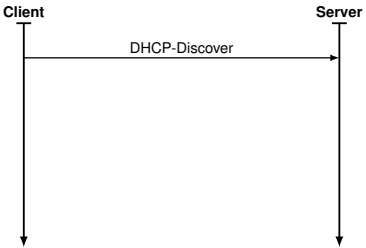
Woher bekommen Hosts eigentlich ihre IP-Adresse?

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

Ablauf:

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



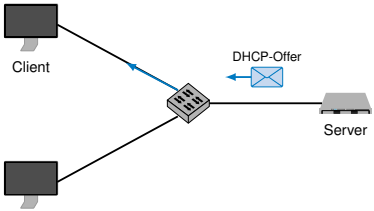


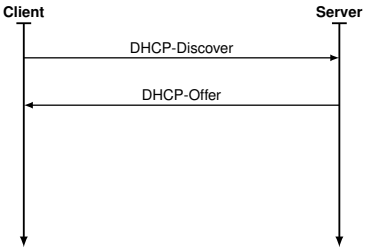
Woher bekommen Hosts eigentlich ihre IP-Adresse?

- 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



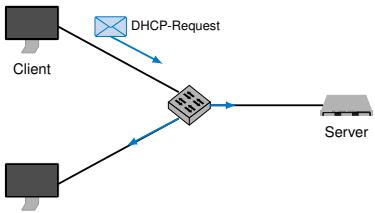


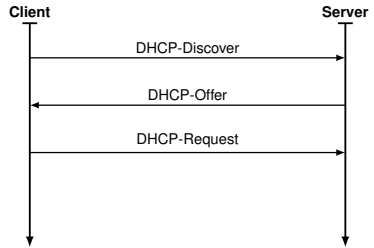
Woher bekommen Hosts eigentlich ihre IP-Adresse?

- 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



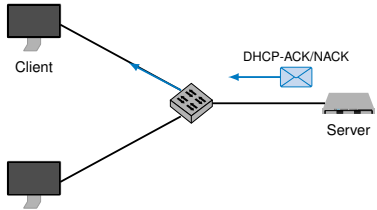


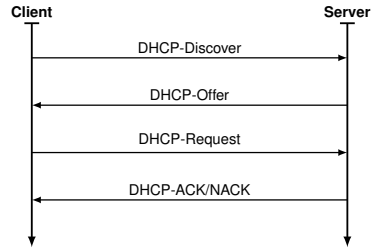
Woher bekommen Hosts eigentlich ihre IP-Adresse?

- 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 DHCP-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 weitere Optionen ausliefern:
  - DNS-Resolver zur Namensauflösung (→ Kapitel 5)
  - Hostname und Domänen-Suffix (→ Kapitel 5)
  - 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

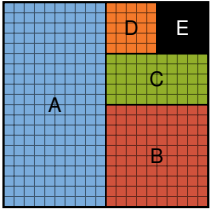
- die ersten drei Oktette einer IP-Adresse das Netz identifizieren und
- das vierte Oktett Hosts innerhalb eines Netzes adressiert.

Diese Aufteilung war 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 [2] (selbe Farbkodierung).

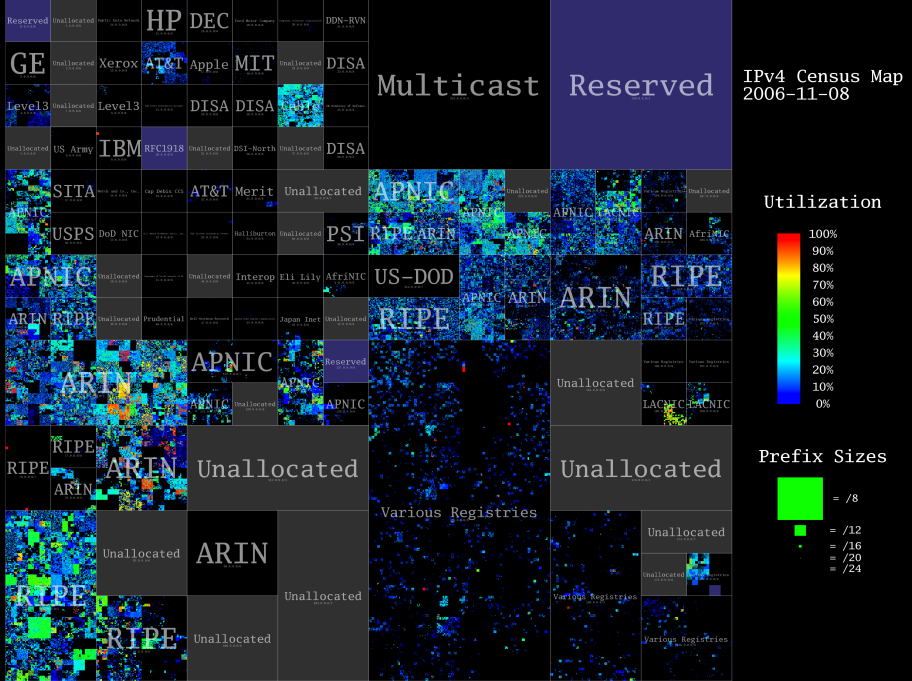
IPv4 Census Map
2006-11-08

Utilization

100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%

Prefix Sizes

= /8
= /12
= /16
= /20
= /24



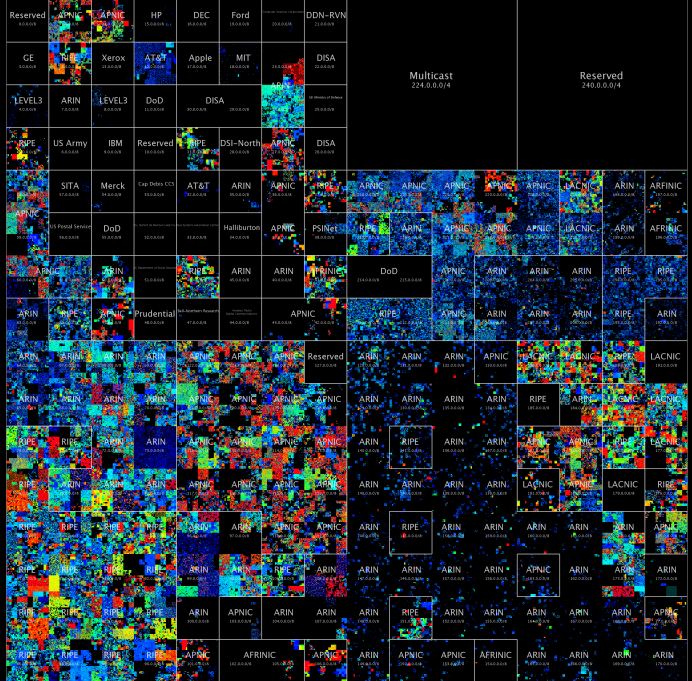
IPv4 Census Map
2006-11-08

Utilization

100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%

Prefix Sizes

= /8
= /12
= /16
= /20
= /24


Reserved
APNIC
APNIC
HP
DEC
Ford
DDN-RVN
GE
RIPE
Xerox
AT&T
Apple
MIT
DISA
Multicast
224.0.0.0/4
Reserved
240.0.0.0/4
LEVEL3
ARIN
LEVEL3
DoD
DISA
RIPE
US Army
IBM
Reserved
RIPE
DSI-North
APNIC
DISA
SITA
Merck
Cap Debis CCS
AT&T
ARIN
APNIC
RIPE
APNIC
APNIC
APNIC
APNIC
APNIC
LACNIC
ARIN
AFRINIC
US Postal Service
DoD
Halliburton
APNIC
PSINet
RIPE
ARIN
APNIC
APNIC
APNIC
LACNIC
ARIN
AFRINIC
APNIC
ARIN
RIPE
ARIN
APNIC
DoD
APNIC
ARIN
ARIN
ARIN
RIPE
RIPE
ARIN
RIPE
APNIC
Prudential
APNIC
RIPE
APNIC
ARIN
ARIN
ARIN
RIPE
ARIN
ARIN
ARIN
ARIN
ARIN
APNIC
APNIC
APNIC
Reserved
ARIN
ARIN
ARIN
APNIC
LACNIC
APNIC
RIPE
LACNIC
ARIN
ARIN
ARIN
ARIN
APNIC
APNIC
APNIC
ARIN
ARIN
ARIN
ARIN
RIPE
ARIN
LACNIC
LACNIC
RIPE
RIPE
ARIN
ARIN
APNIC
APNIC
APNIC
APNIC
ARIN
RIPE
ARIN
ARIN
APNIC
APNIC
RIPE
LACNIC
RIPE
RIPE
ARIN
ARIN
APNIC
APNIC
APNIC
ARIN
ARIN
ARIN
ARIN
ARIN
LACNIC
APNIC
LACNIC
RIPE
RIPE
RIPE
RIPE
ARIN
ARIN
APNIC
APNIC
ARIN
RIPE
ARIN
ARIN
ARIN
ARIN
ARIN
APNIC
RIPE
RIPE
RIPE
RIPE
ARIN
ARIN
RIPE
ARIN
ARIN
ARIN
ARIN
APNIC
ARIN
ARIN
ARIN
ARIN
RIPE
RIPE
RIPE
RIPE
ARIN
APNIC
ARIN
ARIN
ARIN
RIPE
ARIN
ARIN
ARIN
ARIN
ARIN
APNIC
RIPE
RIPE
RIPE
RIPE
APNIC
AFRINIC
APNIC
ARIN
APNIC
APNIC
AFRINIC
ARIN
ARIN
ARIN
ARIN
IPv4 Census Map
June - October 2012
Utilization
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Prefix Sizes
= /8
= /12
= /16
= /20
= /24
420 Million hosts that responded to ICMP Ping at least 2 times between June and October 2012
Source: Carna Botnet

Verschwendung des Adressraums

- Als IPv4 1981 eingeführt wurde, konnte man sich nicht vorstellen, dass $\sim 2^{32}$ 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[3] an eine der fünf Regional Internet Registries (RIRs), das APNIC[4]
- IPv4-Adressraum praktisch aufgebraucht

Ein aktuelles Beispiel:

- Die Rechnerbetriebsgruppe (RBG) verwaltet ein /16 Subnetz (131.159.0.0 / 16)
- Davon verfügen wir derzeit über ein /23 Subnetz zur Bereitstellung von TUMexam

[3] Internet Assigned Numbers Authority

[4] Asia-Pacific Network Information Center (APNIC)

Bereits 1993 wurde mit CIDR[5] 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
	Netz / Host	
Netzadresse	11000000 . 10101000 . 00000000 . 00000000	192.168.0.0
Broadcastadresse	11000000 . 10101000 . 00000000 . 11111111	192.168.0.255

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

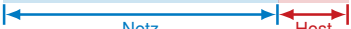
[5] Classless Inter-Domain Routing

Bereits 1993 wurde mit CIDR[5] 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
Netzadresse	11000000 . 10101000 . 00000000 . 10000000	192.168.0.128
Broadcastadresse	11000000 . 10101000 . 00000000 . 11111111	192.168.0.255



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

[5] Classless Inter-Domain Routing

Bereits 1993 wurde mit CIDR[5] 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
Netzadresse	11000000 . 10101000 . 00000000 . 10000000	192.168.0.128
Broadcastadresse	11000000 . 10101000 . 00000000 . 10111111	192.168.0.191

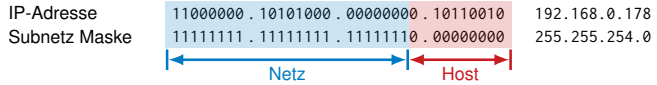
- 26 bit Netzanteil, 6 bit Hostanteil $\Rightarrow 2^6 = 64$ Adressen
- Nutzbare Adressen für Hosts: $2^6 - 2 = 62$
- Netzadresse: 192.168.0.128
- Broadcast-Adresse: 192.168.0.191

[5] Classless Inter-Domain Routing

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:



- Netzadresse: 192.168.0.0
- Broadcast-Adresse: 192.168.1.255
- 23 bit Netzanteil, 9 bit Hostanteil $\Rightarrow 2^9 = 512$ Adressen
- 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.

Internet Protocol version 4 (IPv4) [14]

Besondere Adressbereiche

Einige Adressbereiche sind für bestimmte Zwecke reserviert:

1. 0.0.0.0 / 8: Hosts in diesem Netzwerk
   - Verwendung z.B. als Quelladresse bei DHCP, wenn ein Client noch keine IP-Adresse besitzt.
   - Serveranwendungen erwarten eingehende Verbindungen auf der Adresse 0.0.0.0, was soviel bedeutet wie „jede Adresse, die verfügbar ist“.
   - Andere Adressen innerhalb dieses Bereichs dürfen als Zieladresse für bestimmte Hosts innerhalb des lokalen Netzes verwendet werden.
   - Adressen aus diesem Bereich werden grundsätzlich nicht geroutet.
2. 127.0.0.0 / 8: „Loopback-Adressen“
   - Adressen in diesem Bereich identifizieren den lokalen Rechner (Localhost), z. B. 127.0.0.1.
   - Diese Adressen werden grundsätzlich nicht geroutet.
   - Pakete mit dieser Zieladresse würden niemals gesendet sondern vor dem Sendevorgang „geloopt“.
3. 10.0.0.0 / 8, 172.16.0.0 / 12, 192.168.0.0 / 16: Private Adressbereiche
   - Dürfen innerhalb lokaler Netzwerke ohne Registrierung durch die IANA verwendet werden.
   - Adressen innerhalb dieser Bereiche dürfen zwischen privaten Netzen geroutet werden.
   - Mehrfache Vergabe in unterschiedlichen privaten Netzen möglich.
   - Dürfen nicht in öffentliche Netze geroutet werden.
4. 169.254.0.0 / 16: Automatic Private IP Adressing (APIPA)
   - Block zur automatischen Adressvergabe (APIPA).
   - Routing wie bei den privaten Adressbereichen.
5. 255.255.255.255 / 32: Global Broadcast
   - Identifiziert im Prinzip alle Hosts.
   - Wird niemals geroutet.

IPv6 wurde Ende 1995 [6] als Nachfolger für IPv4 vorgeschlagen und mit RFC 2460 [7] 1998 standardisiert.

- Damit dürfte es älter als die Mehrzahl der Vorlesungsteilnehmer sein . . .
- . . . und hat IPv4 noch lange nicht obsolet gemacht.

Hauptgrund dafür ist, dass

- in erster Linie die Adressknappheit treibende Kraft für die Verbreitung von IPv6 ist und
- diese erst in den vergangenen Jahren zum Problem wurde.

[6] 2^{128} = 340.282.366.920.938.463.463.374.607.431.768.211.456 Adressen

Internet Protocol version 6 (IPv6)

IPv6 wurde Ende 1995 [6] als Nachfolger für IPv4 vorgeschlagen und mit RFC 2460 [7] 1998 standardisiert.

- Damit dürfte es älter als die Mehrzahl der Vorlesungsteilnehmer sein ...
- ... und hat IPv4 noch lange nicht obsolet gemacht.

Hauptgrund dafür ist, dass

- in erster Linie die Adressknappheit treibende Kraft für die Verbreitung von IPv6 ist und
- diese erst in den vergangenen Jahren zum Problem wurde.

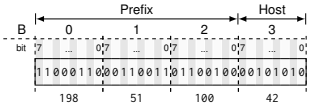
Die wesentlichen Änderungen gegenüber IPv4 umfassen:

- Vergrößerung des Adressraums von 2^{32} auf 2^{128}. [6]
- Vereinfachung des Headerformats (effizientere Verarbeitung auf Routern).
- Änderungen bei der IP-Fragmentierung.
- Flexibilität durch sog. Extension Header bei gleichzeitiger Vereinfachung des Headerformats.
- Stateless Address Autoconfiguration (SLAAC) mittels ICMPv6.
- Möglichkeit für Stateful Autoconfiguration durch DHCPv6.
- Nativer Einsatz von Multicast, beispielsweise um alle Router in einem Segment zu adressieren.

[6] 2^{128} = 340.282.366.920.938.463.463.374.607.431.768.211.456 Adressen

Adressformat

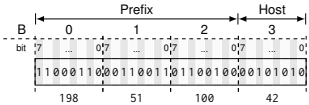
- IPv4-Adressen werden üblicherweise in Dotted-Decimal-Notation dargestellt, d. h. je eine 8 bit lange Gruppe wird als Dezimalzahl im Bereich 0–255 getrennt durch Punkte notiert.



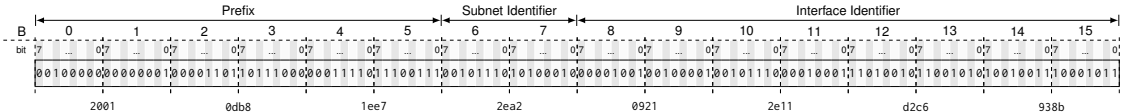
- Diese Schreibweise würde bei IPv6 insgesamt 16 Gruppen ergeben – das wäre unübersichtlich und schwer zu merken.

Adressformat

- IPv4-Adressen werden üblicherweise in Dotted-Decimal-Notation dargestellt, d. h. je eine 8 bit lange Gruppe wird als Dezimalzahl im Bereich 0–255 getrennt durch Punkte notiert.

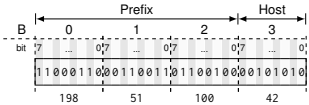


- Diese Schreibweise würde bei IPv6 insgesamt 16 Gruppen ergeben – das wäre unübersichtlich und schwer zu merken.
- IPv6-Adressen werden in 8 Gruppen zu je 16 bit getrennt durch Doppelpunkte (colon-separated) in hexadezimaler Schreibweise dargestellt.

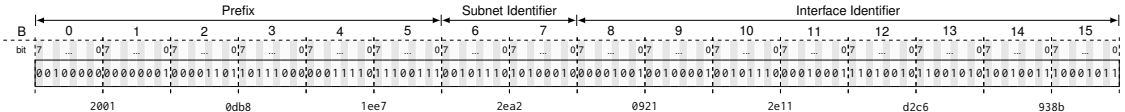


Adressformat

- IPv4-Adressen werden üblicherweise in Dotted-Decimal-Notation dargestellt, d. h. je eine 8 bit lange Gruppe wird als Dezimalzahl im Bereich 0–255 getrennt durch Punkte notiert.



- Diese Schreibweise würde bei IPv6 insgesamt 16 Gruppen ergeben – das wäre unübersichtlich und schwer zu merken.
- IPv6-Adressen werden in 8 Gruppen zu je 16 bit getrennt durch Doppelpunkte (colon-separated) in hexadezimaler Schreibweise dargestellt.



Achtung:

- Bitte IPv6-Adressen nicht mit MAC-Adressen verwechseln, nur weil beide hexadezimal notiert werden.
- Bitte das Präfix, das es auch bei IPv4 gibt, und den Subnet Identifier auseinander halten.

Hinweise zur Notation [10][1]

Wir betrachten die folgende Adresse:

```
2001:0db8:0000:0000:0001:0000:0000:0001
```

[1] Weitere Details finden sich in RFC 5952.

Hinweise zur Notation [10][1]

Wir betrachten die folgende Adresse:

`2001:0db8:0000:0000:0001:0000:0000:0001`

1. Führende Nullen in den einzelnen Blöcken werden weggelassen:
   ✓ `2001:db8:0:0:1:0:0:1`

[1] Weitere Details finden sich in RFC 5952.

Hinweise zur Notation [10][1]

Wir betrachten die folgende Adresse:

`2001:0db8:0000:0000:0001:0000:0000:0001`

1. Führende Nullen in den einzelnen Blöcken werden weggelassen:
   ✓ `2001:db8:0:0:1:0:0:1`
2. Höchstens eine Gruppe konsekutiver Blöcke, die nur aus Nullen bestehen, darf wie folgt abgekürzt werden:
   ✓ `2001:db8::1:0:0:1`

[1] Weitere Details finden sich in RFC 5952.

Hinweise zur Notation [10][1]

Wir betrachten die folgende Adresse:

`2001:0db8:0000:0000:0001:0000:0000:0001`

1. Führende Nullen in den einzelnen Blöcken werden weggelassen:
   ✓ `2001:db8:0:0:1:0:0:1`
2. Höchstens eine Gruppe konsekutiver Blöcke, die nur aus Nullen bestehen, darf wie folgt abgekürzt werden:
   ✓ `2001:db8::1:0:0:1`
3. Gibt es mehrere Möglichkeiten für Fall 2), so wählt man die längste Sequenz von Nullen. Bei mehreren gleich langen Sequenzen wählt man die erste Möglichkeit. Falsch wäre also:
   × `2001:db8:0:0:1::1`
   × `2001:db8::1::1`

[1] Weitere Details finden sich in RFC 5952.

Hinweise zur Notation [10][1]

Wir betrachten die folgende Adresse:

`2001:0db8:0000:0000:0001:0000:0000:0001`

1. Führende Nullen in den einzelnen Blöcken werden weggelassen:
   ✓ `2001:db8:0:0:1:0:0:1`
2. Höchstens eine Gruppe konsekutiver Blöcke, die nur aus Nullen bestehen, darf wie folgt abgekürzt werden:
   ✓ `2001:db8::1:0:0:1`
3. Gibt es mehrere Möglichkeiten für Fall 2), so wählt man die längste Sequenz von Nullen. Bei mehreren gleich langen Sequenzen wählt man die erste Möglichkeit. Falsch wäre also:
   × `2001:db8:0:0:1::1`
   × `2001:db8::1::1`
4. Ein einzelner 0 Block darf nicht mit `::` abgekürzt werden:
   ✓ `2001:db8:0:1:1:1:1:1`
   × `2001:db8::1:1:1:1:1`

Anderes Beispiel: Die Loopback-Adresse (vgl. 127.0.0.1 bei IPv4) lässt sich wie folgt kürzen:

`0000:0000:0000:0000:0000:0000:0000:0001/128`

[1] Weitere Details finden sich in RFC 5952.

Hinweise zur Notation [10][1]

Wir betrachten die folgende Adresse:

`2001:0db8:0000:0000:0001:0000:0000:0001`

1. Führende Nullen in den einzelnen Blöcken werden weggelassen:
   ✓ `2001:db8:0:0:1:0:0:1`
2. Höchstens eine Gruppe konsekutiver Blöcke, die nur aus Nullen bestehen, darf wie folgt abgekürzt werden:
   ✓ `2001:db8::1:0:0:1`
3. Gibt es mehrere Möglichkeiten für Fall 2), so wählt man die längste Sequenz von Nullen. Bei mehreren gleich langen Sequenzen wählt man die erste Möglichkeit. Falsch wäre also:
   × `2001:db8:0:0:1::1`
   × `2001:db8::1::1`
4. Ein einzelner 0 Block darf nicht mit `::` abgekürzt werden:
   ✓ `2001:db8:0:1:1:1:1:1`
   × `2001:db8::1:1:1:1:1`

Anderes Beispiel: Die Loopback-Adresse (vgl. 127.0.0.1 bei IPv4) lässt sich wie folgt kürzen:

`0000:0000:0000:0000:0000:0000:0000:0001/128` ↦ `::1/128`

[1] Weitere Details finden sich in RFC 5952.

Internet Protocol version 6 (IPv6)

Headerformat

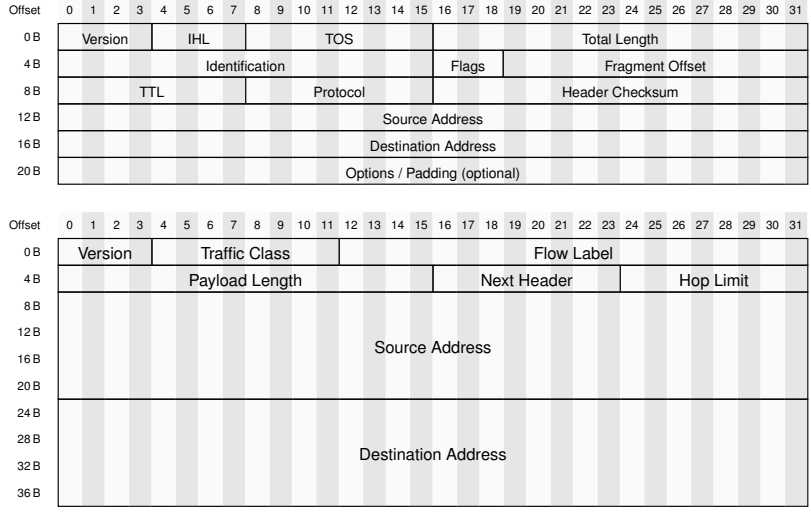


Abbildung 5: IPv4-Header (oben) und IPv6-Header (unten) im Vergleich

Internet Protocol version 6 (IPv6)

Headerformat

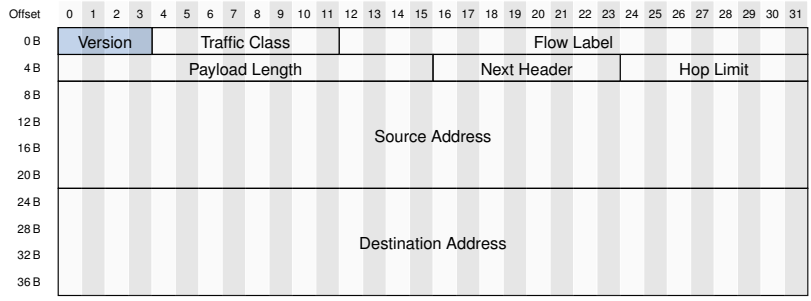


Abbildung 6: IPv6-Header (minimale Länge: 40 B, vgl. 20 B bei IPv4)

Version

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

Internet Protocol version 6 (IPv6)

Headerformat

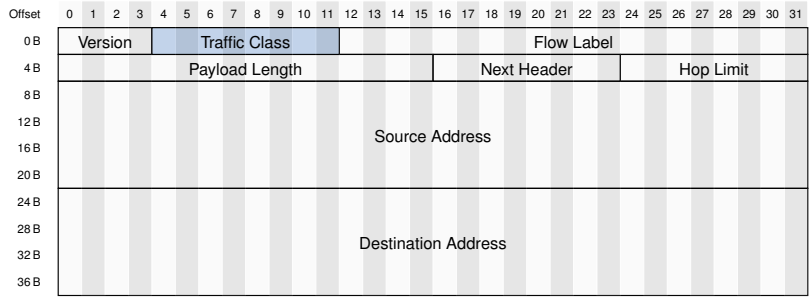


Abbildung 6: IPv6-Header (minimale Länge: 40 B, vgl. 20 B bei IPv4)

Traffic Class

- Äquivalent zum TOS-Feld des IPv4-Headers.
- Wird zur Verkehrspriorisierung / Quality of Service (QoS) verwendet.

Internet Protocol version 6 (IPv6)

Headerformat

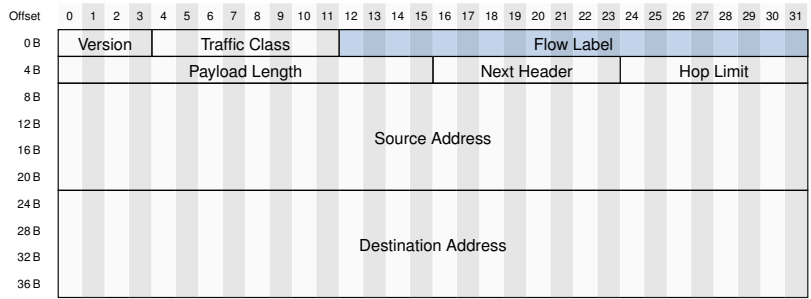


Abbildung 6: IPv6-Header (minimale Länge: 40 B, vgl. 20 B bei IPv4)

Flow Label

- Ursprünglich vorgesehen für Echtzeitanwendungen.
- Wird heute in erster Linie von Routern verwendet, um zusammengehörende Pakete (Flows) auf Schicht 3 zu erkennen.
- Pakete, die zum selben Flow gehören sollen ggf. gleich behandelt werden, z. B. im Fall mehrerer möglicher Pfade zum Ziel alle über denselben Pfad geroutet werden.

Internet Protocol version 6 (IPv6)

Headerformat

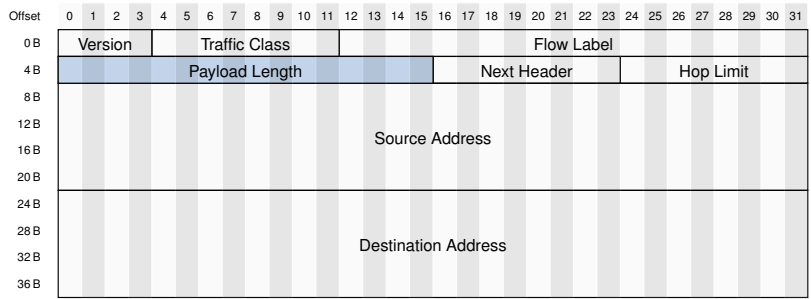


Abbildung 6: IPv6-Header (minimale Länge: 40 B, vgl. 20 B bei IPv4)

Payload Length

- Gibt die Länge der auf den IPv6-Header folgenden Daten (inkl. Extension Header, mehr dazu gleich) an.
- Angabe in Vielfachen von 1 B.
- Der IPv6-Header inkl. seiner Extension Header muss immer ein Vielfaches von 8 B sein.

Headerformat

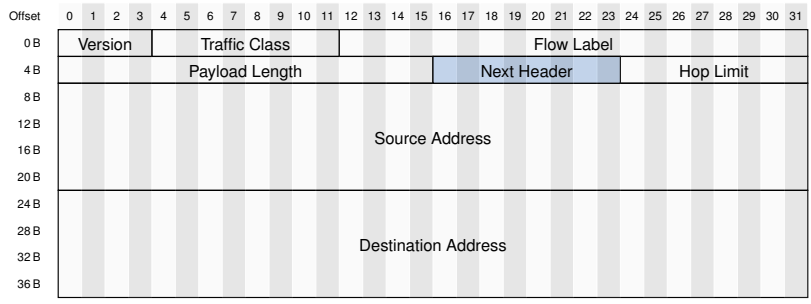


Abbildung 6: IPv6-Header (minimale Länge: 40 B, vgl. 20 B bei IPv4)

Next Header

- Gibt den Typ des nächsten Headers an, der am Ende des IPv6-Headers folgt.
- Dies kann entweder ein L4-Header (z. B. TCP oder UDP), ein ICMPv6-Header oder ein sog. IPv6 Extension Header sein (mehr dazu gleich).

Internet Protocol version 6 (IPv6)

Headerformat

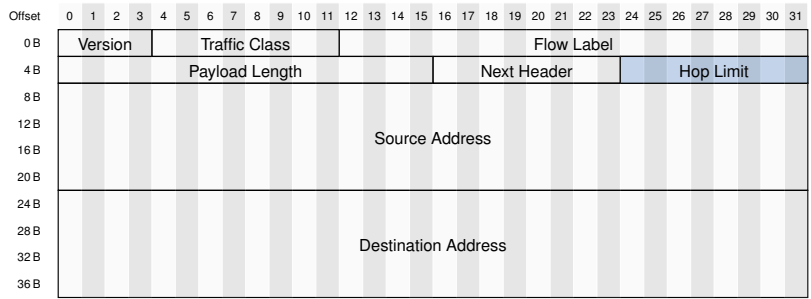


Abbildung 6: IPv6-Header (minimale Länge: 40 B, vgl. 20 B bei IPv4)

Hop Limit

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

Internet Protocol version 6 (IPv6)

Headerformat

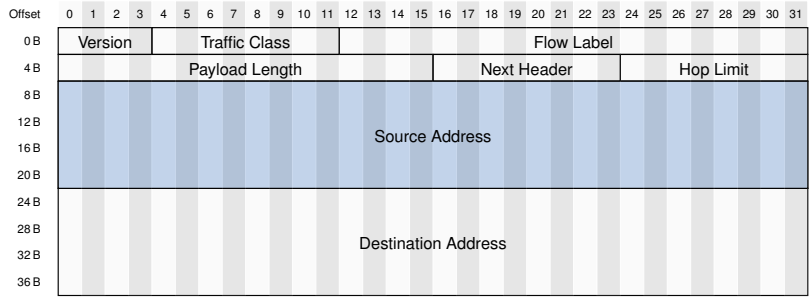


Abbildung 6: IPv6-Header (minimale Länge: 40 B, vgl. 20 B bei IPv4)

Source Address

- 128 bit lange IPv6-Quelladresse.

Internet Protocol version 6 (IPv6)

Headerformat

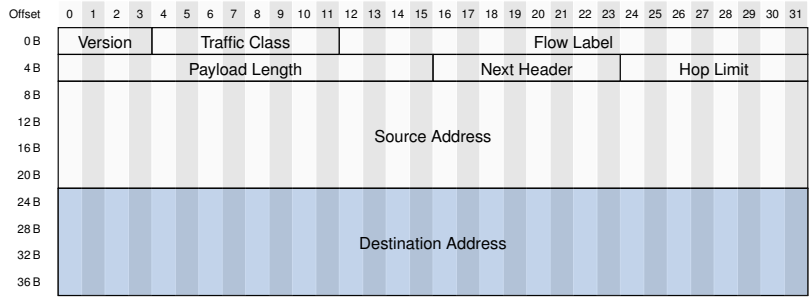


Abbildung 6: IPv6-Header (minimale Länge: 40 B, vgl. 20 B bei IPv4)

Destination Address

- 128 bit lange IPv6-Zieladresse.

IPv6 Extension Header [7]

- Extension Header erlauben es zusätzliche Layer 3 Informationen in einem IPv6 Paket anzufügen.
- Das jeweilige Next Header Feld gibt den jeweils nächsten Extension Header oder das L4 Protokoll an.
- Die Next Header Felder des IPv6 Pakets bzw. der Extension Header bilden hierbei eine Kette, z. B.
  - IPv6 Header, Next Header: Routing
  - Routing Header, Next Header: Fragment
  - Fragment Header, Next Header: TCP
  - TCP Payload
- Das Format hängt vom jeweiligen Header ab.
- Mit Ausnahme des Hop-by-Hop Options Header und des Routing Header werden Extension Header nur vom Empfänger ausgewertet.
  - Unbekannte Extension Header können vom Empfänger nicht verarbeitet werden
    $\rightarrow$ Paket wird mit ICMP Error Message verworfen

Im Folgenden betrachten wir exemplarisch den Fragment Header.

Beispiel:

- Rahmen haben auf Schicht 2 eine maximale Größe, z. B. 1514 B für die L2-PDU (ohne CRC-Checksumme) bei IEEE 802.3u (100 Mbit/s Ethernet).
- Diese gibt auch die maximale Größe einer L3-PDU vor, welche als Maximum Transmission Unit (MTU) bezeichnet wird.
- Überschreitet eine L3-PDU diese Grösse, muss die L3-SDU fragmentiert und in Form unabhängiger Pakete versendet werden.
- Der Empfänger muss die einzelnen Fragmente im Anschluss reassemblieren.

Beispiel:

- Rahmen haben auf Schicht 2 eine maximale Größe, z. B. 1514 B für die L2-PDU (ohne CRC-Checksumme) bei IEEE 802.3u (100 Mbit/s Ethernet).
- Diese gibt auch die maximale Größe einer L3-PDU vor, welche als Maximum Transmission Unit (MTU) bezeichnet wird.
- Überschreitet eine L3-PDU diese Grösse, muss die L3-SDU fragmentiert und in Form unabhängiger Pakete versendet werden.
- Der Empfänger muss die einzelnen Fragmente im Anschluss reassemblieren.

IPv6 verfügt zu diesem Zweck über einen eigenen Extension Header, den Fragment Header:

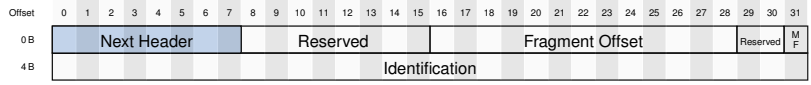


Abbildung 7: Fragment Header

Next Header

- Gibt den nächsten Extension Header oder das verwendete L4 Protokoll an

Beispiel:

- Rahmen haben auf Schicht 2 eine maximale Größe, z. B. 1514 B für die L2-PDU (ohne CRC-Checksumme) bei IEEE 802.3u (100 Mbit/s Ethernet).
- Diese gibt auch die maximale Größe einer L3-PDU vor, welche als Maximum Transmission Unit (MTU) bezeichnet wird.
- Überschreitet eine L3-PDU diese Grösse, muss die L3-SDU fragmentiert und in Form unabhängiger Pakete versendet werden.
- Der Empfänger muss die einzelnen Fragmente im Anschluss reassemblieren.

IPv6 verfügt zu diesem Zweck über einen eigenen Extension Header, den Fragment Header:

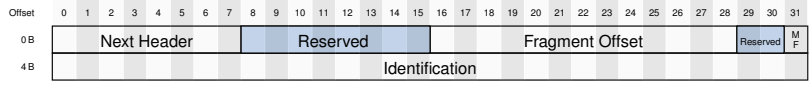


Abbildung 7: Fragment Header

Reserved

- Hat derzeit beim Fragmentation Header keine Verwendung.
- Bei den meisten anderen Extension Headers gibt dieses Feld die Länge des jeweiligen Extension Headers an.

Beispiel:

- Rahmen haben auf Schicht 2 eine maximale Größe, z. B. 1514 B für die L2-PDU (ohne CRC-Checksumme) bei IEEE 802.3u (100 Mbit/s Ethernet).
- Diese gibt auch die maximale Größe einer L3-PDU vor, welche als Maximum Transmission Unit (MTU) bezeichnet wird.
- Überschreitet eine L3-PDU diese Grösse, muss die L3-SDU fragmentiert und in Form unabhängiger Pakete versendet werden.
- Der Empfänger muss die einzelnen Fragmente im Anschluss reassemblieren.

IPv6 verfügt zu diesem Zweck über einen eigenen Extension Header, den Fragment Header:

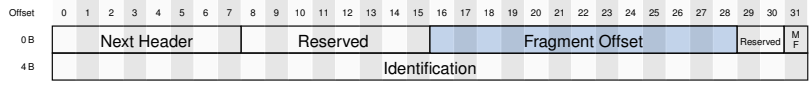


Abbildung 7: Fragment Header

Fragment Offset

- Offset der fragmentierten L3-SDU in Vielfachen von 8 B.
- Bei IPv6 erfolgt die Fragmentierung *ausschließlich* am Sender.
  Frage: Woher kennt der Sender die maximale MTU auf dem Weg von sich selbst bis hin zum Ziel?
- Bei IPv4 können Pakete, falls nicht explizit über das DF-Bit untersagt, bei Bedarf auch von Routern fragmentiert werden.
  Frage: Können Fragmente noch einmal fragmentiert werden?

Beispiel:

- Rahmen haben auf Schicht 2 eine maximale Größe, z. B. 1514 B für die L2-PDU (ohne CRC-Checksumme) bei IEEE 802.3u (100 Mbit/s Ethernet).
- Diese gibt auch die maximale Größe einer L3-PDU vor, welche als Maximum Transmission Unit (MTU) bezeichnet wird.
- Überschreitet eine L3-PDU diese Grösse, muss die L3-SDU fragmentiert und in Form unabhängiger Pakete versendet werden.
- Der Empfänger muss die einzelnen Fragmente im Anschluss reassemblieren.

IPv6 verfügt zu diesem Zweck über einen eigenen Extension Header, den Fragment Header:

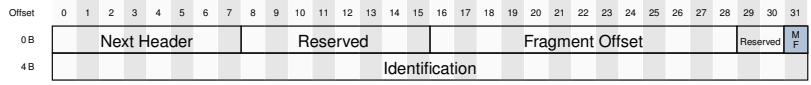


Abbildung 7: Fragment Header

More Fragments (MF)

- Gibt an, ob auf das aktuelle Paket weitere Fragmente folgen oder ob es sich um das letzte Fragment (gemäß des Fragment Offsets) handelt.

Beispiel:

- Rahmen haben auf Schicht 2 eine maximale Größe, z. B. 1514 B für die L2-PDU (ohne CRC-Checksumme) bei IEEE 802.3u (100 Mbit/s Ethernet).
- Diese gibt auch die maximale Größe einer L3-PDU vor, welche als Maximum Transmission Unit (MTU) bezeichnet wird.
- Überschreitet eine L3-PDU diese Grösse, muss die L3-SDU fragmentiert und in Form unabhängiger Pakete versendet werden.
- Der Empfänger muss die einzelnen Fragmente im Anschluss reassemblieren.

IPv6 verfügt zu diesem Zweck über einen eigenen Extension Header, den Fragment Header:

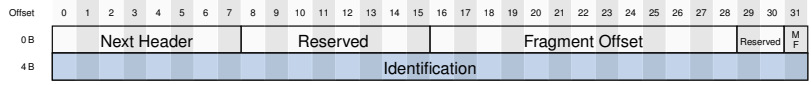


Abbildung 7: Fragment Header

Identification

- 32 bit langer, vom Sender zufällig gewählter Wert, welcher alle Fragmente identifiziert, die zu einer L3-SDU reassembliert werden sollen.

Frage: Warum wird **nicht** das Flow Label des IPv6 Headers verwendet?

IPv6-Adressen zur besonderen Verwendung [4]

`::1/128` – Loopback-Adresse

- Adressiert den Localhost, d.h. Pakete mit dieser Zieladresse verlassen den lokalen Rechner gar nicht erst, sondern werden über das sog. Loopback-Interface sofort wieder zugestellt (vgl. 127.0.0.1 bei IPv4).
- Werden nicht geroutet.

`::/128` – nicht-spezifizierte Adresse

- Analogon zu 0.0.0.0 bei IPv4.
- Wird nicht geroutet.

`fe80::/10` – Link-Local Adressen

- Jedes IPv6 Interface benötigt eine Link-Local-Adresse.
- Die Link-Local-Adresse wird aus dem Interface-Identifier generiert.
- Hat nur innerhalb des lokalen Links Gültigkeit und wird daher nicht geroutet.

`fc00::/7` – Unique-Local Unicast-Adressen

- Adressen, die nur für lokale Kommunikation (z. B. innerhalb eines Firmennetzes) vorgesehen sind.
- Dürfen wie private IPv4-Adressen lokal aber nicht im Internet geroutet werden.

`ff00::/8` – Multicast-Adressen

- Adressen, die eine bestimmte Gruppe von Hosts adressieren (mehr dazu gleich).
- Werden geroutet.

(fast) der ganze Rest – Globale Adressen

- Global eindeutige Adressen, die für den Einsatz in öffentlichen Netzen vorgesehen sind.
- Werden geroutet.

IPv6-Multicast

Grundsätzlich unterscheidet man sowohl auf Schicht 3/2 die vier folgenden Adressierungsarten:

1. Unicast
   - Pakete (Rahmen), die an ein einzelnes Ziel adressiert sind.
   - Alle anderen Knoten im Netzwerk verwerfen derartige Pakete (Rahmen) bzw. leiten sie lediglich zum Ziel weiter.
2. Broadcast
   - Pakete (Rahmen), die an alle Stationen im Netzwerk adressiert sind.
   - Adressierung erfolgt mittels spezieller Broadcast-Adressen.
   - Auf Schicht 3 sind Broadcasts zumeist auf das lokale Netzsegment begrenzt (vgl. Broadcast-Domain).
3. Multicast
   - Pakete (Rahmen), die an eine bestimmte Gruppe von Knoten adressiert sind.
   - Adressierung erfolgt mittels spezieller Multicast-Adressen.
   - Auf Schicht 2 werden Multicasts von Switche häufig wie Broadcasts behandelt.
   - Auf Schicht 3 gibt es spezielle Protokolle[1], die Multicast-Adressierung auch über das lokale Netzsegment hinaus ermöglichen.
4. Anycast[2]
   - Pakete, die an eine beliebige Station einer bestimmten Gruppe adressiert sind.
   - Beispiel: Die wichtigsten Nameserver (→ Kapitel 5).

[1] Protocol Independent Multicast (PIM) ermöglicht das Routing von Multicast-Pakete sowohl für IPv4 als auch IPv6. Details sind Inhalt weiterführender Veranstaltungen.

[2] Wird im Rahmen der Veranstaltung nicht weiter behandelt.

IPv6-Multicast

Multicast ist elementarer Bestandteil von IPv6. Die folgenden Adressen bzw. Präfixe sind für bestimmte Multicast-Gruppen reserviert:

`ff02::1` – All Nodes

- Adressiert alle Knoten auf dem lokalen Link.

`ff02::2` – All Routers

- Adressiert alle Router auf dem lokalen Link.

`ff02::1:2` – All DHCP-Agents

- Adressiert alle DHCP-Server auf dem lokalen Link.

`ff02::1:ff00:0/104` – Solicited-Node Address

- Die Solicited-Node Adresse wird im Neighbor Discovery Protocol (mehr dazu gleich) verwendet, welches u. a. zur Adressauflösung dient.
- Die Solicited-Node Adresse zu einer IPv6 Adresse wird aus dem `ff02::1:ff00:0/104` und den letzten 24 bit der ursprünglichen IPv6 Adresse generiert.
- Die Solicited-Node Adresse für `2001:0db8:1ee7:2ea2:0921:2e11:d2c6:938b` ist somit `ff02::1:ffc6:938b`.
- IPv6-Multicasts werden auch auf Schicht 2 mittels Multicast-Adressen versendet.
  - Switche müssen Multicast-Rahmen nur an die Ports weiterleiten, an denen ein Mitglied der entsprechenden Multicast-Gruppe angeschlossen ist.
  - In großen L2-Netzen können so unnötige Broadcasts vermieden werden.
  - Knoten, für die eine Nachricht nicht von Interesse ist, bekommen diese somit erst gar nicht.

Mapping von Multicast IPv6 Adressen auf MAC-Adressen [5]

- IPv6-Pakete mit einer Zieladresse aus dem Präfix `ff00::/8` werden mit der zugehörigen Multicast-Adresse auf Schicht 2 (Ethernet) versendet.
- Um Multicasts auf Schicht 3 auch auf Schicht 2 abbilden zu können, muss es einen Zusammenhang zwischen den verwendeten Adressen beider Schichten geben.
- Die ersten 2 Oktette der MAC-Adresse werden auf `33:33` gesetzt.
  - letztes Bit des ersten Oktetts ist gesetzt → Multicast
  - vorletztes Bit des ersten Oktetts ist gesetzt → locally administered
  - siehe Kapitel 2
- Die letzten 4 Oktette der Ethernetadresse werden die letzten 4 Oktette der IPv6 Multicastadresse.

Mapping von Multicast IPv6 Adressen auf MAC-Adressen [5]

- IPv6-Pakete mit einer Zieladresse aus dem Präfix `ff00::/8` werden mit der zugehörigen Multicast-Adresse auf Schicht 2 (Ethernet) versendet.
- Um Multicasts auf Schicht 3 auch auf Schicht 2 abbilden zu können, muss es einen Zusammenhang zwischen den verwendeten Adressen beider Schichten geben.
- Die ersten 2 Oktette der MAC-Adresse werden auf `33:33` gesetzt.
  - letztes Bit des ersten Oktetts ist gesetzt → Multicast
  - vorletztes Bit des ersten Oktetts ist gesetzt → locally administered
  - siehe Kapitel 2
- Die letzten 4 Oktette der Ethernetadresse werden die letzten 4 Oktette der IPv6 Multicastadresse.

Beispiel:

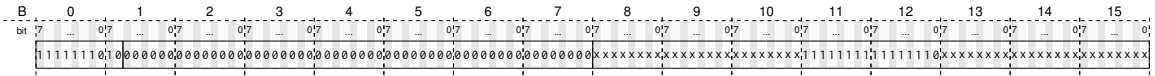
`ff02::1:ffc6:938b` ↦ `33:33:ff:c6:93:8b`

RFC 7042, Abschnitt 2.3.1:

> *"(Historical note: It was the custom during IPv6 design to use '3' for unknown or example values, and 3333 Coyote Hill Road, Palo Alto, California, is the address of PARC (Palo Alto Research Center, formerly 'Xerox PARC'). Ethernet was originally specified by the Digital Equipment Corporation, Intel Corporation, and Xerox Corporation. The pre-IEEE [802.3] Ethernet protocol has sometimes been known as 'DIX' Ethernet from the first letters of the names of these companies.)"*

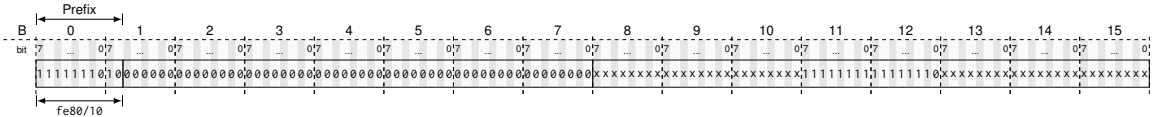
Stateless Address Autoconfiguration (SLAAC) [17]

IPv6 erlaubt eine automatische Konfiguration von Hosts innerhalb eines einzelnen Subnetzes. Ein Host generiert sich die für ein Interface benötigte link-local IPv6-Adresse wie folgt:



Stateless Address Autoconfiguration (SLAAC) [17]

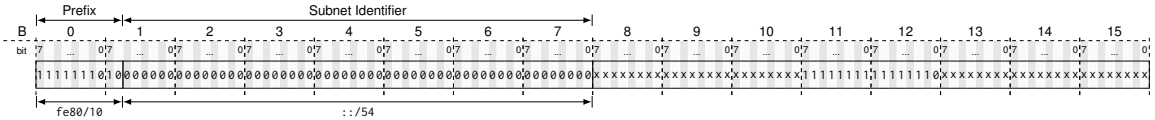
IPv6 erlaubt eine automatische Konfiguration von Hosts innerhalb eines einzelnen Subnetzes. Ein Host generiert sich die für ein Interface benötigte link-local IPv6-Adresse wie folgt:



- Das Präfix ist `fe80::/10`.

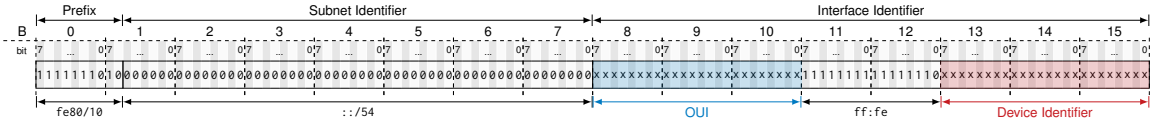
Stateless Address Autoconfiguration (SLAAC) [17]

IPv6 erlaubt eine automatische Konfiguration von Hosts innerhalb eines einzelnen Subnetzes. Ein Host generiert sich die für ein Interface benötigte link-local IPv6-Adresse wie folgt:



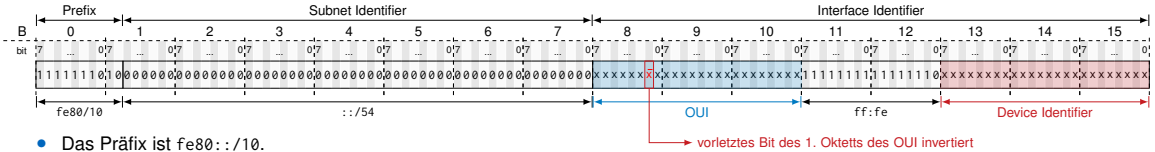
- Das Präfix ist `fe80::/10`.
- Der Subnet Identifier (die folgenden 54 bit) werden auf `0` gesetzt.

IPv6 erlaubt eine automatische Konfiguration von Hosts innerhalb eines einzelnen Subnetzes. Ein Host generiert sich die für ein Interface benötigte link-local IPv6-Adresse wie folgt:



- Das Präfix ist `fe80::/10`.
- Der Subnet Identifier (die folgenden 54 bit) werden auf `0` gesetzt.
- Die verbleibenden 64 bit stellen den Interface Identifier dar, welcher aus der MAC-Adresse des jeweiligen Interfaces als modifizierter EUI-64 Identifier generiert wird:
  - Die ersten 24 bit sind der OUI der MAC-Adresse.
  - Die nachfolgenden 16 bit werden mit `ff:fe` „gestopft“.
  - Die restlichen 24 bit werden mit dem Device Identifier der MAC-Adresse aufgefüllt.

IPv6 erlaubt eine automatische Konfiguration von Hosts innerhalb eines einzelnen Subnetzes. Ein Host generiert sich die für ein Interface benötigte link-local IPv6-Adresse wie folgt:



- Das Präfix ist `fe80::/10`.
- Der Subnet Identifier (die folgenden 54 bit) werden auf `0` gesetzt.
- Die verbleibenden 64 bit stellen den Interface Identifier dar, welcher aus der MAC-Adresse des jeweiligen Interfaces als modifizierter EUI-64 Identifier generiert wird:
  - Die ersten 24 bit sind der OUI der MAC-Adresse.
  - Die nachfolgenden 16 bit werden mit `ff:fe` „gestopft“.
  - Die restlichen 24 bit werden mit dem Device Identifier der MAC-Adresse aufgefüllt.
- Dabei ist das vorletzte Bit des ersten Oktett des OUI (global/local-Bit) invertiert:
  - Bei MAC-Adressen bedeutet eine `0` an dieser Bitstelle eine global eindeutige und eine `1` eine lokal administrierte Adresse (siehe Kapitel 2).
  - Bei IPv6 ist es genau andersrum: Durch die Invertierung wird erreicht, dass eine manuell konfigurierte IPv6-Adresse wie `2001:db8::1` nicht einen Interface Identfier enthält, der auf eine global eindeutige MAC- Adresse hinweist.
  - Andernfalls müsste man von Hand Adressen wie `2001:db8::200:0:0:1` vergeben ...

Stateless Address Autoconfiguration (SLAAC) [17]

Auch globale Adressen können über SLAAC konfiguriert werden:

- Um eine globale Adresse konfigurieren zu können, muss der Host zunächst wissen, welche IPv6 Präfixe von den lokalen Routern bedient werden.
- Präfix Informationen können von den Routern über das Neighbor Discovery Protocol (später) in Form von Router Advertisements versendet werden.
- Der Host kann sich über das /64 Präfix und dem modifizierten EUI-64 Identifier selbstständig eine Adresse erzeugen.
- SLAAC heißt stateless, da die Adressen nicht von einem Server vergeben werden
  - Globale Adressen können auch über DHCPv6 vergeben werden

Stateless Address Autoconfiguration (SLAAC) [17]

Auch globale Adressen können über SLAAC konfiguriert werden:

- Um eine globale Adresse konfigurieren zu können, muss der Host zunächst wissen, welche IPv6 Präfixe von den lokalen Routern bedient werden.
- Präfix Informationen können von den Routern über das Neighbor Discovery Protocol (später) in Form von Router Advertisements versendet werden.
- Der Host kann sich über das /64 Präfix und dem modifizierten EUI-64 Identifier selbstständig eine Adresse erzeugen.
- SLAAC heißt stateless, da die Adressen nicht von einem Server vergeben werden
  - Globale Adressen können auch über DHCPv6 vergeben werden

Frage: Welche Konsequenzen hat die Erzeugung des Interface Identifiers aus der MAC-Adresse einer Netzwerkkarte hinsichtlich Privacy?

Stateless Address Autoconfiguration (SLAAC) [17]

Auch globale Adressen können über SLAAC konfiguriert werden:

- Um eine globale Adresse konfigurieren zu können, muss der Host zunächst wissen, welche IPv6 Präfixe von den lokalen Routern bedient werden.
- Präfix Informationen können von den Routern über das Neighbor Discovery Protocol (später) in Form von Router Advertisements versendet werden.
- Der Host kann sich über das /64 Präfix und dem modifizierten EUI-64 Identifier selbstständig eine Adresse erzeugen.
- SLAAC heißt stateless, da die Adressen nicht von einem Server vergeben werden
  - Globale Adressen können auch über DHCPv6 vergeben werden

Frage: Welche Konsequenzen hat die Erzeugung des Interface Identifiers aus der MAC-Adresse einer Netzwerkkarte hinsichtlich Privacy?

- Da MAC-Adressen i. d. R. eindeutig sind, kann ein Host durch die in seine IPv6-Adresse eingebettete MAC-Adresse unabhängig von Standard, Anschluss oder Provider verfolgt werden.
- Abhilfe schaffen die IPv6 Privacy Extensions [12]:
  - Erzeugung und regelmäßige Erneuerung von zufälligen Device Identifiern und damit einhergehende Wechsel der globalen IPv6 Adresse.
  - Das vorletzte Bit des ersten Oktett des „OUI“ wird auf `0` gesetzt.

Genereller Aufbau einer ICMPv6-Nachricht:

- Die ersten 32 bit sind immer vorhanden.
- Gesamtlänge der Nachricht hängt von Type und Code der ICMPv6-Nachricht ab.
- Die Länge, innerhalb der für IPv6 geltenden Grenzen, auf 1 B beliebig.

- Type gibt die Art der ICMP-Nachricht an, z.B. Echo Request, Echo Reply, Time Exceeded usw. . . .

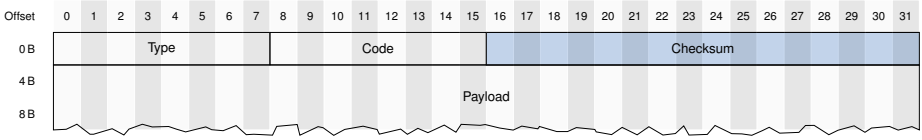
Genereller Aufbau einer ICMPv6-Nachricht:

- Die ersten 32 bit sind immer vorhanden.
- Gesamtlänge der Nachricht hängt von Type und Code der ICMPv6-Nachricht ab.
- Die Länge, innerhalb der für IPv6 geltenden Grenzen, auf 1 B beliebig.

- Code präzisiert die Art der Nachricht, z.B. den Grund einer Time Exceeded Nachricht (0 - Hop limit exceeded; 1 - Fragment reassembly time exceeded).

Genereller Aufbau einer ICMPv6-Nachricht:

- Die ersten 32 bit sind immer vorhanden.
- Gesamtlänge der Nachricht hängt von Type und Code der ICMPv6-Nachricht ab.
- Die Länge, innerhalb der für IPv6 geltenden Grenzen, auf 1 B beliebig.



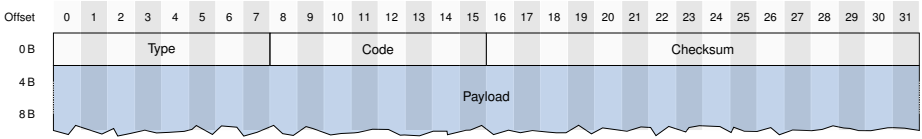
- Checksum ist eine (vergleichsweise einfache) fehlererkennende Checksumme:
  - Die Checksumme berücksichtigt das gesamte ICMPv6-Paket (inkl. Payload).
  - Zur Berechnung wird ein sog. IPv6-Pseudo-Header verwendet.
  - Der Pseudo-Header enthält die Absender- und Ziel-IP sowie das Next-Header-Feld.

Warum ein „Pseudo-Header“?

- Wie wir aus Kapitel 1/2 wissen, können Kanalkodierung und Checksummen nicht garantieren, dass eine Nachricht selbst bei gültigen Checksummen korrekt übertragen wurde.
- Weitere Checksummen auf höheren Schichten können dies ebenfalls nicht, aber die Wahrscheinlichkeit für einen unbemerkten Fehler sinkt hier selbst bei einfachen Checksummen drastisch.
- Gleichzeitig würde eine Checksumme über Headerfelder, welche sich von Hop zu Hop ändern, eine gewisse Belastung für Router (oder auf Schicht 2 für Switches) darstellen.
- Daher wird in diesem Fall die Checksumme nur über den Teil der Nachricht gebildet, welcher sich gewöhnlich nicht (oder nur sehr selten) auf dem Weg vom Sender zum Empfänger ändert.
- Beachten Sie beispielsweise den Ausschluss des Hopcounts, welcher sich ja bei jedem Hop ändert.

Genereller Aufbau einer ICMPv6-Nachricht:

- Die ersten 32 bit sind immer vorhanden.
- Gesamtlänge der Nachricht hängt von Type und Code der ICMPv6-Nachricht ab.
- Die Länge, innerhalb der für IPv6 geltenden Grenzen, auf 1 B beliebig.



- Payload der ICMPv6-Nachricht (Beispiele folgen).

Neighbor Discovery Protocol (NDP) [13]

IPv6 bietet mit seiner Neighbor Discovery, welche Bestandteil von ICMPv6 ist, eine Reihe von Funktionalitäten, für die bei IPv4 nicht oder nur unvollständig standardisiert sind und eigene Protokolle notwendig gemacht haben. Funktionen der Neighbor Discovery sind insbesondere:

- Adressauflösung, Duplicate Address Detection und Neighbor Unreachability Detection: Neighbor Solicitations und Advertisements.
- Automatisches Auffinden von Routern innerhalb des lokalen Netzsegments, Adress-Präfixen und Parameter Konfiguration: Router Discovery und Router Advertisements.
- Umleitung zu anderen Gateways: Redirects.

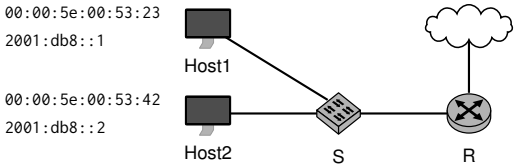
Neighbor Discovery Protocol (NDP) [13]

IPv6 bietet mit seiner Neighbor Discovery, welche Bestandteil von ICMPv6 ist, eine Reihe von Funktionalitäten, für die bei IPv4 nicht oder nur unvollständig standardisiert sind und eigene Protokolle notwendig gemacht haben. Funktionen der Neighbor Discovery sind insbesondere:

- Adressauflösung, Duplicate Address Detection und Neighbor Unreachability Detection: Neighbor Solicitations und Advertisements.
- Automatisches Auffinden von Routern innerhalb des lokalen Netzsegments, Adress-Präfixen und Parameter Konfiguration: Router Discovery und Router Advertisements.
- Umleitung zu anderen Gateways: Redirects.

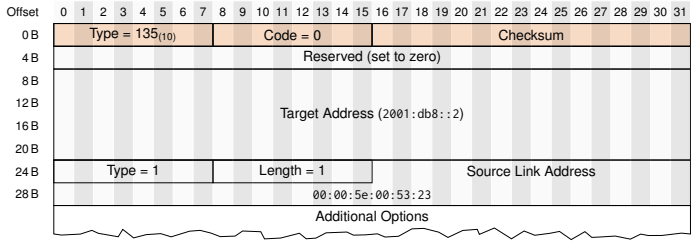
Beispiel: Adressauflösung bei IPv6[3]

- Host1 will eine Nachricht an Host2 senden
- Die IP-Adresse von Host2 (`2001:db8::2`) sei ihm bereits bekannt
- Wie erhält Host1 die zugehörige MAC-Adresse?



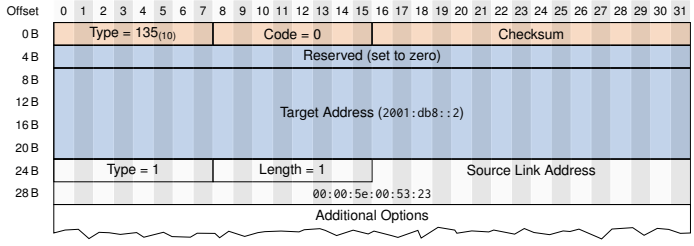
[3] Wir betrachten nur den Fall der Adressauflösung (vgl. ARP bei IPv4). Beim Einsatz von Neighbor Solicitations und Neighbor Advertisements zu anderen Zwecken sind i.A. andere Adressen und Optionen erforderlich.

Neighbor Discovery Protocol (NDP) [13] – Neighbor Solicitation (Request)



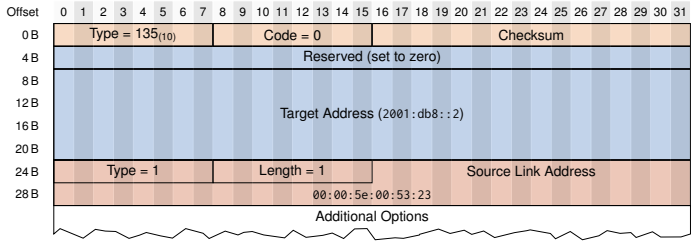
- ICMPv6 Header
  - ICMPv6 Type und Code (`0x87` und `0x00` für eine Neighbor Solicitation Nachricht) sowie die ICMPv6 Checksumme.

Neighbor Discovery Protocol (NDP) [13] – Neighbor Solicitation (Request)



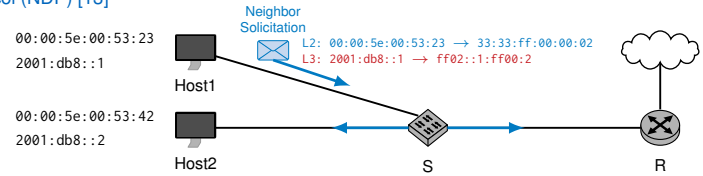
- ICMPv6 Header
  - ICMPv6 Type und Code (`0x87` und `0x00` für eine Neighbor Solicitation Nachricht) sowie die ICMPv6 Checksumme.
- Neighbor Discovery Body
  - Die ersten 32 bit sind reserviert, so dass die Nachricht insgesamt wieder ein Vielfaches von 8 B lang wird.
  - Im Anschluss folgt die Ziel-IPv6-Adresse, zu der die entsprechende MAC-Adresse gesucht wird.

Neighbor Discovery Protocol (NDP) [13] – Neighbor Solicitation (Request)

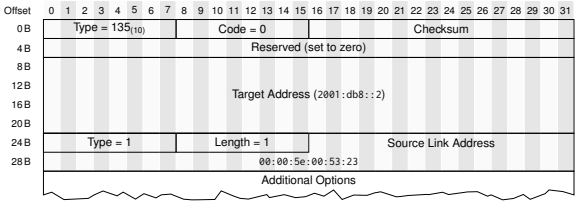


- ICMPv6 Header
  - ICMPv6 Type und Code (`0x87` und `0x00` für eine Neighbor Solicitation Nachricht) sowie die ICMPv6 Checksumme.
- Neighbor Discovery Body
  - Die ersten 32 bit sind reserviert, so dass die Nachricht insgesamt wieder ein Vielfaches von 8 B lang wird.
  - Im Anschluss folgt die Ziel-IPv6-Adresse, zu der die entsprechende MAC-Adresse gesucht wird.
- Neighbor Discovery Options
  - Neighbor Discovery Pakete können selbst wiederum Optionen enthalten.
  - Type und Length geben den Typ (1 für Source Link Layer Address) und Gesamtlänge der Option in Vielfachen von 8 B an.
  - Im Fall eines Neighbor Solicitation Pakets folgt L2-Adresse des anfragenden Knotens (Source Link Address).
  - Je nach Typ des NDP-Pakets können weitere Optionen folgen, wobei Empfänger unbekannte Optionen ignorieren müssen.

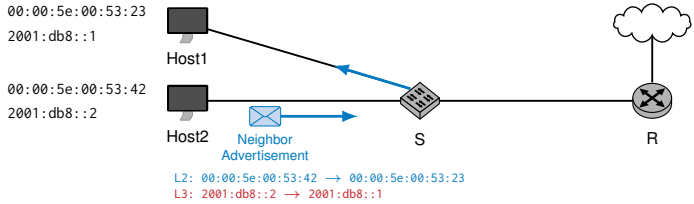
Beispiel:



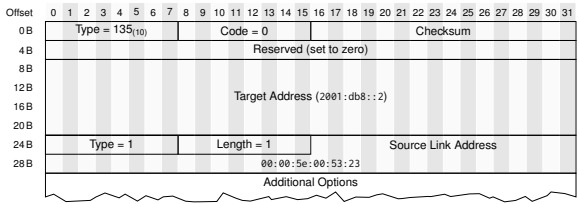
- Host 1 sendet eine „Neighbor Solicitation for `2001:db8::2` from `00:00:5e:00:53:23`“ an die zur bekannten IPv6 Adresse gehörende Solicited-Node Adresse (Multicast).

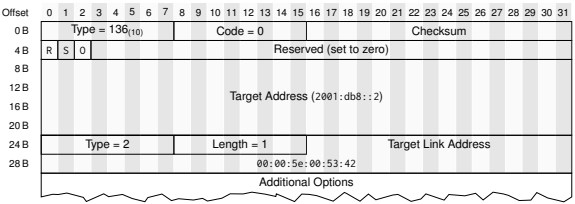


Beispiel:

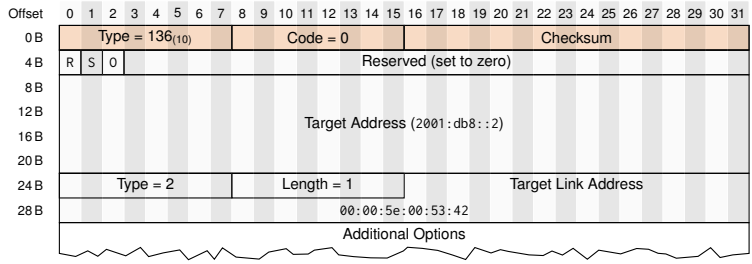


- Host 1 sendet eine „Neighbor Solicitation for `2001:db8::2` from `00:00:5e:00:53:23`“ an die zur bekannten IPv6 Adresse gehörende Solicited-Node Adresse (Multicast).
- Host 2 empfängt diese Nachricht und antwortet mit einer „Neighbor Advertisement `2001:db8::2` (`sol`, `ovr`) is at `00:00:5e:00:53:42`“ Nachricht (Unicast).



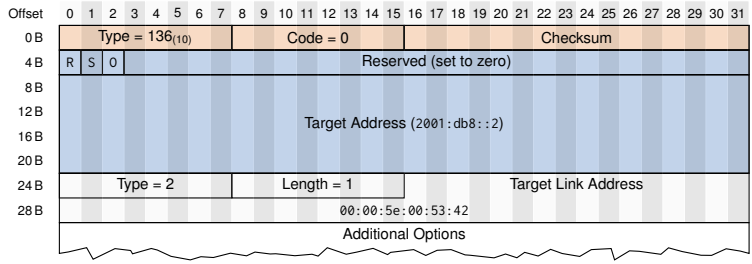


Neighbor Discovery Protocol (NDP) [13] – Neighbor Advertisement (Reply)



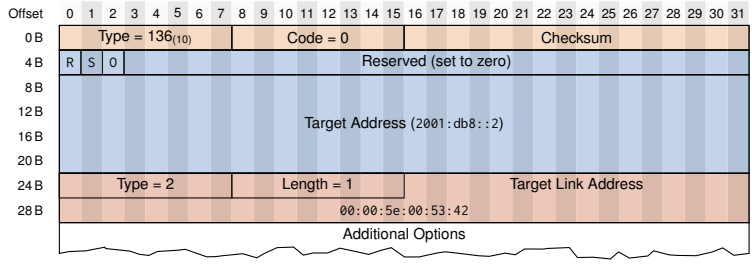
- ICMPv6 Header
  - ICMPv6 Type und Code (`0x88` und `0x00` für ein Neighbor Advertisement) sowie die ICMPv6 Checksumme.

Neighbor Discovery Protocol (NDP) [13] – Neighbor Advertisement (Reply)



- ICMPv6 Header
  - ICMPv6 Type und Code (`0x88` und `0x00` für ein Neighbor Advertisement) sowie die ICMPv6 Checksumme.
- Neighbor Discovery Body
  - Die drei höchstwertigen Bit des erste Oktetts haben folgende Bedeutungen:
    - Router-Flag `R` wird gesetzt, wenn der antwortende Knoten ein Router ist.
    - Solicited Flag `S` gibt an, ob das Advertisement infolge einer Solicitation geschickt wird.
    - Override Flag `O` wird gesetzt, wenn das Advertisement eine möglicherweise gecached Link-Layer Adresse beim Empfänger aktualisieren soll.

Neighbor Discovery Protocol (NDP) [13] – Neighbor Advertisement (Reply)



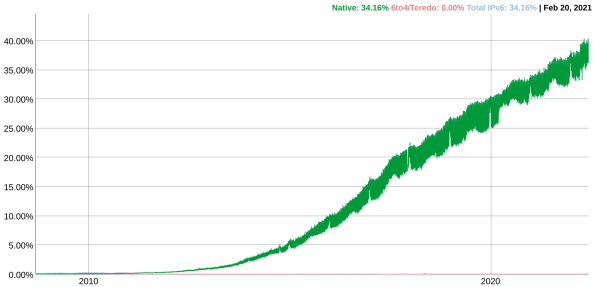
- ICMPv6 Header
  - ICMPv6 Type und Code (`0x88` und `0x00` für ein Neighbor Advertisement) sowie die ICMPv6 Checksumme.
- Neighbor Discovery Body
  - Die drei höchstwertigen Bit des erste Oktetts haben folgende Bedeutungen:
    - Router-Flag `R` wird gesetzt, wenn der antwortende Knoten ein Router ist.
    - Solicited Flag `S` gibt an, ob das Advertisement infolge einer Solicitation geschickt wird.
    - Override Flag `O` wird gesetzt, wenn das Advertisement eine möglicherweise gecached Link-Layer Adresse beim Empfänger aktualisieren soll.
- Neighbor Discovery Options
  - Type und Length geben den Typ (`2` für Target Link Layer Address) und Gesamtlänge der Option in Vielfachen von 8 B an.
  - Im Fall eines Neighbor Advertisements folgt die L2-Adresse des angefragten Knotens (Target Link Address).
  - Je nach Typ des NDP-Pakets können weitere Optionen folgen, wobei Empfänger unbekannte Optionen ignorieren müssen.

Kompatibilität

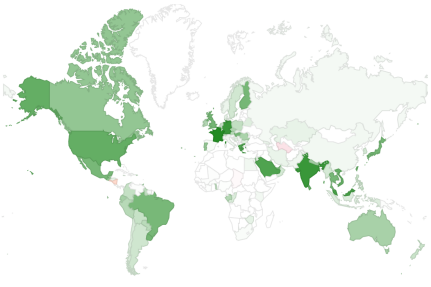
- IPv4 und IPv6 sind nicht kompatibel, können aber nebeneinander existieren.
- Man verwendet heute häufig IPv6 parallel zu IPv4 (Dual Stack).
- Router müssen für beide Protokollversionen Routinginformationen getrennt voneinander austauschen und verarbeiten.

Stand bis 2022:

- Obwohl IPv6 bereits seit 1998 standardisiert ist (RFC 8200), ist die Umstellung auf IPv6 noch lange nicht abgeschlossen.
- Rund 60 % des weltweiten Datenverkehrs ist noch immer IPv4:

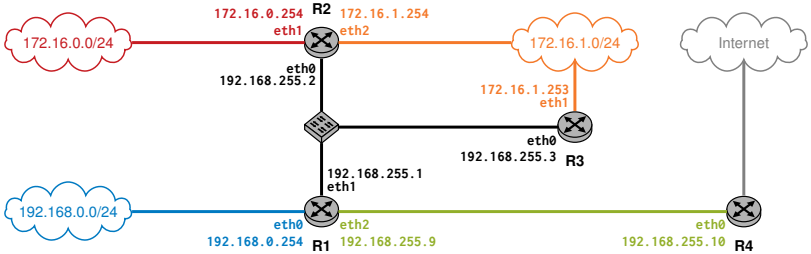


(a) IPv6 Traffic [1]



(b) IPv6 pro Land [1]

Wir betrachten im Folgenden das unten abgebildete Beispielnetzwerk:

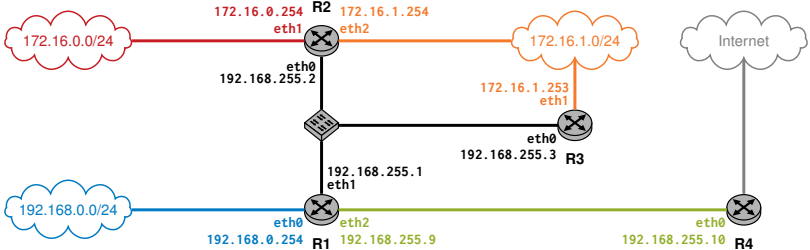


- Die Farben der Links und Interface-Adressen verdeutlichen die einzelnen Subnetze
- Das Netzwerk 192.168.255.0/29 (schwarz) verfügt über 6 nutzbare Hostadressen
- Das Netzwerk 192.168.255.8/30 (grün) ist ein Transportnetz mit nur 2 nutzbaren Hostadressen
- Die übrigen Netze sind /24 Netze mit jeweils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

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

Statisches Routing

Routing Table



Routing Table

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

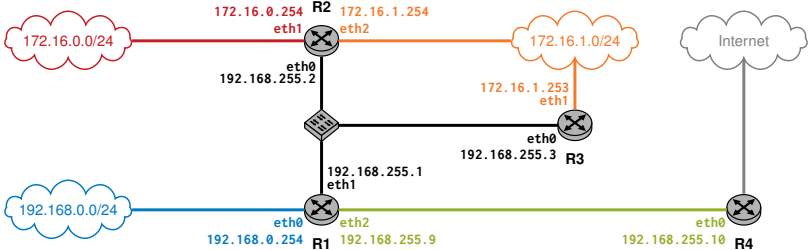
- die Netzadresse eines Ziels,
- die Länge des Präfixes,
- den zugehörigen Next-Hop (auch Gateway genannt),
- das Interface, über welches dieser Next-Hop erreichbar ist, und
- die Kosten bis zum Ziel.

Hinweise:

- Bei IPv4 wird häufig anstatt der Präfixlänge die Subnetzmaske (Linux-Terminologie: Genmask) angegeben. Bei einem Präfix von *N* bit handelt es sich bei IPv4 um eine IP-Adresse eines Netzes, bei dem die ersten *N* bit die Netzwerkadresse darstellen, und die restlichen bit 0 sind.
- Kosten sind von der Metrik zu unterscheiden. Kosten beziehen sich auf eine Metrik (z. B. Hop-Count, Bandbreite, Verzögerung etc.).

Statisches Routing

Routing Table



Beispiel: Routing-Tabelle für R1

Destination	NextHop	Costs	Iface
192.168.255.8/30	0.0.0.0	0	eth2
192.168.255.0/29	0.0.0.0	0	eth1
192.168.0.0/24	0.0.0.0	0	eth0
172.16.1.0/24	192.168.255.3	1	eth1
172.16.0.0/23	192.168.255.2	1	eth1
0.0.0.0/0	192.168.255.10	1	eth2

- Die Netze 172.16.{0,1}.0/24 wurden zusammengefasst
- Die Route 0.0.0.0/0 wird auch als Default Route bezeichnet
- Interessant: R1 kennt zwei (eigentlich sogar drei) Routen zum Netz 172.16.1.0/24 !

1. R1 berechnet das logische AND aus der Zieladresse des Pakets und den Subnetzmasken (welche aus der Präfixlänge hervorgehen) in seiner Routingtabelle, in absteigender Präfixlänge.
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 der Zieladresse 172.16.1.23.

Destination	NextHop	Costs	Iface
192.168.255.8/30	0.0.0.0	0	eth2
192.168.255.0/29	0.0.0.0	0	eth1
192.168.0.0/24	0.0.0.0	0	eth0
172.16.1.0/24	192.168.255.3	1	eth1
172.16.0.0/23	192.168.255.2	1	eth1
0.0.0.0/0	192.168.255.10	1	eth2

1. R1 berechnet das logische AND aus der Zieladresse des Pakets und den Subnetzmasken (welche aus der Präfixlänge hervorgehen) in seiner Routingtabelle, in absteigender Präfixlänge.
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 der Zieladresse 172.16.1.23.

	Destination	NextHop	Costs	Iface
→	192.168.255.8/30	0.0.0.0	0	eth2
	192.168.255.0/29	0.0.0.0	0	eth1
	192.168.0.0/24	0.0.0.0	0	eth0
	172.16.1.0/24	192.168.255.3	1	eth1
	172.16.0.0/23	192.168.255.2	1	eth1
	0.0.0.0/0	192.168.255.10	1	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

1. R1 berechnet das logische AND aus der Zieladresse des Pakets und den Subnetzmasken (welche aus der Präfixlänge hervorgehen) in seiner Routingtabelle, in absteigender Präfixlänge.
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 der Zieladresse 172.16.1.23.

	Destination	NextHop	Costs	Iface
	192.168.255.8/30	0.0.0.0	0	eth2
→	192.168.255.0/29	0.0.0.0	0	eth1
	192.168.0.0/24	0.0.0.0	0	eth0
	172.16.1.0/24	192.168.255.3	1	eth1
	172.16.0.0/23	192.168.255.2	1	eth1
	0.0.0.0/0	192.168.255.10	1	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

1. R1 berechnet das logische AND aus der Zieladresse des Pakets und den Subnetzmasken (welche aus der Präfixlänge hervorgehen) in seiner Routingtabelle, in absteigender Präfixlänge.
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 der Zieladresse 172.16.1.23.

	Destination	NextHop	Costs	Iface
	192.168.255.8/30	0.0.0.0	0	eth2
	192.168.255.0/29	0.0.0.0	0	eth1
→	192.168.0.0/24	0.0.0.0	0	eth0
	172.16.1.0/24	192.168.255.3	1	eth1
	172.16.0.0/23	192.168.255.2	1	eth1
	0.0.0.0/0	192.168.255.10	1	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

1. R1 berechnet das logische AND aus der Zieladresse des Pakets und den Subnetzmasken (welche aus der Präfixlänge hervorgehen) in seiner Routingtabelle, in absteigender Präfixlänge.
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 der Zieladresse 172.16.1.23.

	Destination	NextHop	Costs	Iface
	192.168.255.8/30	0.0.0.0	0	eth2
	192.168.255.0/29	0.0.0.0	0	eth1
	192.168.0.0/24	0.0.0.0	0	eth0
→	172.16.1.0/24	192.168.255.3	1	eth1
	172.16.0.0/23	192.168.255.2	1	eth1
	0.0.0.0/0	192.168.255.10	1	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

⇒ Match, da 172.16.1.0 = 172.16.1.0 ⇒ Gateway ist 192.168.255.3

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.

Destination	NextHop	Costs	Iface
192.168.255.8/30	0.0.0.0	0	eth2
192.168.255.0/29	0.0.0.0	0	eth1
192.168.0.0/24	0.0.0.0	0	eth0
172.16.1.0/24	192.168.255.3	1	eth1
172.16.0.0/23	192.168.255.2	1	eth1
0.0.0.0/0	192.168.255.10	1	eth2

- Der Eintrag für 172.16.0.0/23 liefert ebenfalls einen Match, ist aber weniger spezifisch als der für 172.16.1.0/24 (1 bit kürzeres Präfix).
- Die Default Route 0.0.0.0/0 liefert immer einen Match(logisches AND mit der Maske 0.0.0.0).
- Es ist nicht garantiert, dass das Gateway/Next-Hop der Default Route („Gateway of last resort“) eine Route zum Ziel kennt (→ ICMP Destination Unreachable / Host Unreachable).
- Routen zu direkt verbundenen Netzen (also solchen, zu denen ein Router selbst gehört) können automatisch erzeugt werden. Der NextHop ist in diesem Fall die unspezifizierte Adresse.
- 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

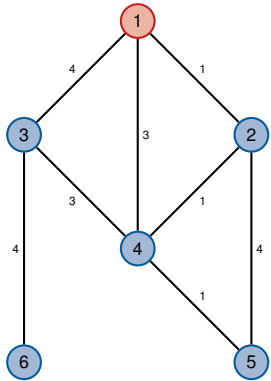
- Router kennen nur Richtung (NextHop) und Entfernung (Kosten) zu einem Ziel
  (vgl. Straßenschild mit Richtungs- und Entfernungsangabe).
- Router haben keine Information über die Netzwerktopologie.
- Router tauschen untereinander lediglich kumulierte Kosten aus
  (z. B. den Inhalt Ihrer Routingtabellen).
- Funktionsprinzip basiert auf dem Algorithmus von Bellman-Ford, der kürzeste Wege ausgehend von einem Startknoten ermittelt und sich leicht verteilt implementieren lässt.

Link-State-Protokolle

- Router informieren einander zusätzlich zu den Kosten auch darüber, wie ein Ziel erreichbar ist.
- Häufig komplexe Nachbarschaftsbeziehungen und Update-Nachrichten.
- Router erhalten so vollständige Topologieinformationen.
- Basierend auf den Topologieinformationen bestimmt jeder Router kürzeste Pfade,
  z. B. mittels Dijkstras Algorithmus.

Algorithmus von Bellman-Ford

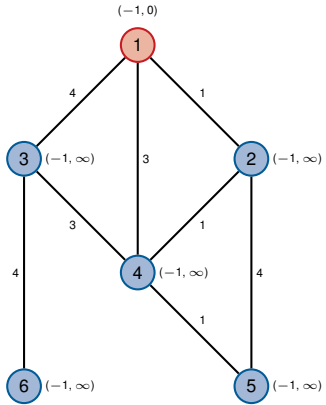
- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i



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

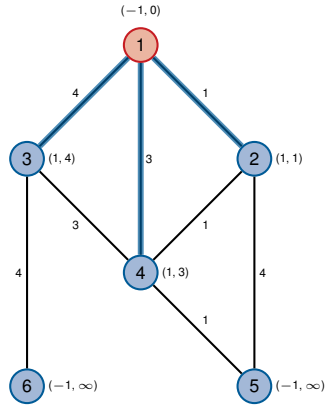


Algorithmus von Bellman-Ford

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

```
// Initialisierung
for i ∈ 𝒩 do
    p[i] = −1
    d[i] = { 0   i = s
           { ∞   sonst
end for
T = {s} // Menge der erreichbaren Knoten

// Berechnung der Pfade
while d[j] changes for some j ∈ 𝒩 do
    S = { } // Aktualisierte Knoten
    for i ∈ T do
        // Für alle Nachbarn von i:
        for ∀j : (i,j) ∈ ℰ do
            if d[i] + c_ij < d[j] then
                p[j] = i
                d[j] = d[i] + c_ij
                S = S ∪ {j}
            end if
        end for
    end for
    T = T ∪ S
end while
```

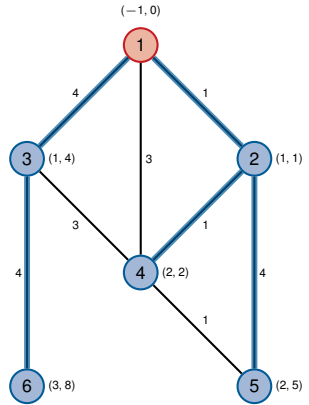


Algorithmus von Bellman-Ford

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

```
// Initialisierung
for i ∈ 𝒩 do
    p[i] = −1
    d[i] = { 0   i = s
           { ∞   sonst
end for
T = {s} // Menge der erreichbaren Knoten

// Berechnung der Pfade
while d[j] changes for some j ∈ 𝒩 do
    S = { } // Aktualisierte Knoten
    for i ∈ T do
        // Für alle Nachbarn von i:
        for ∀j : (i,j) ∈ ℰ do
            if d[i] + c_ij < d[j] then
                p[j] = i
                d[j] = d[i] + c_ij
                S = S ∪ {j}
            end if
        end for
    end for
    T = T ∪ S
end while
```

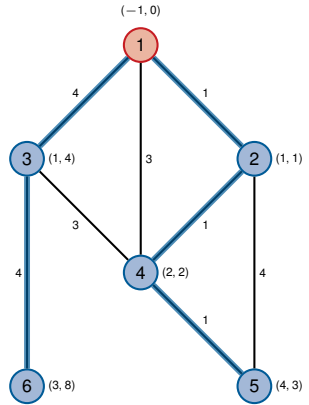


Algorithmus von Bellman-Ford

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

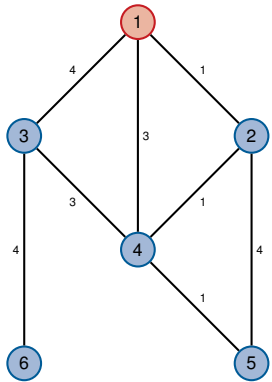
```
// Initialisierung
for i ∈ 𝒩 do
    p[i] = −1
    d[i] = { 0   i = s
           { ∞   sonst
end for
T = {s} // Menge der erreichbaren Knoten

// Berechnung der Pfade
while d[j] changes for some j ∈ 𝒩 do
    S = { } // Aktualisierte Knoten
    for i ∈ T do
        // Für alle Nachbarn von i:
        for ∀j : (i,j) ∈ ℰ do
            if d[i] + c_ij < d[j] then
                p[j] = i
                d[j] = d[i] + c_ij
                S = S ∪ {j}
            end if
        end for
    end for
    T = T ∪ S
end while
```



Dijkstras Algorithmus

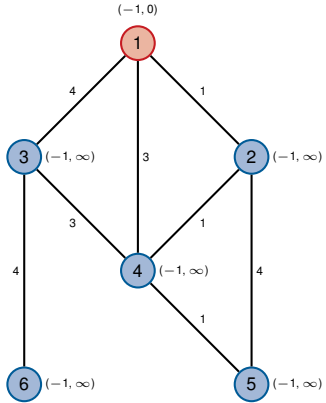
- $p[i]$: Vorgänger von Knoten i im Graphen
- $d[i]$: Distanz von der Wurzel zu Knoten i
- Priority Queue Q, welche Elemente nach Schlüsseln sortiert ausgibt



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 bearbeiteten Knoten

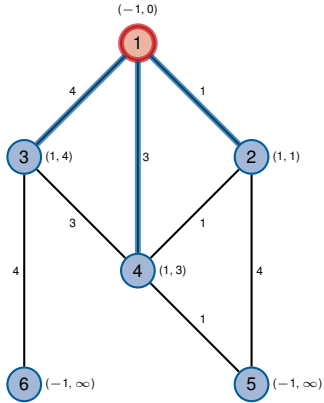


Dijkstras Algorithmus

- $p[i]$: 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 ∈ 𝒩 do
    p[i] = −1
    d[i] = { 0    i = s
           { ∞    sonst
    Q.enqeue(i, d[i])
end for
T = { } // Menge der bearbeiteten Knoten

// Berechnung der Pfade
while T ≠ 𝒩 do
    i = Q.dequeue()
    T = T ∪ {i}
    // Für alle Nachbarn von i:
    for ∀j : (i,j) ∈ ℰ do
        if d[i] + c_ij < d[j] then
            Q.decreaseKey(j, d[i] + c_ij)
            d[j] = d[i] + c_ij
            p[j] = i
        end if
    end for
end while
```

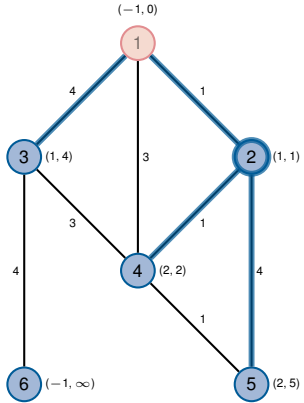


Dijkstras Algorithmus

- $p[i]$: 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 ∈ 𝒩 do
    p[i] = −1
    d[i] = { 0    i = s
           { ∞    sonst
    Q.enqeue(i, d[i])
end for
T = { } // Menge der bearbeiteten Knoten

// Berechnung der Pfade
while T ≠ 𝒩 do
    i = Q.dequeue()
    T = T ∪ {i}
    // Für alle Nachbarn von i:
    for ∀j : (i,j) ∈ ℰ do
        if d[i] + c_ij < d[j] then
            Q.decreaseKey(j, d[i] + c_ij)
            d[j] = d[i] + c_ij
            p[j] = i
        end if
    end for
end while
```

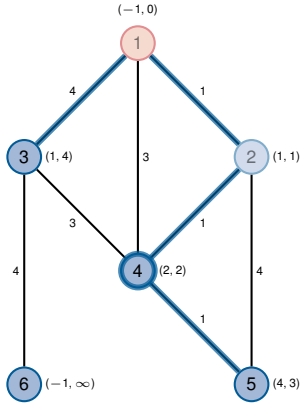


Dijkstras Algorithmus

- $p[i]$: 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 bearbeiteten 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**
            $Q.decreaseKey(j, d[i] + c_{ij})$
            $d[j] = d[i] + c_{ij}$
            $p[j] = i$
        end if
    end for
end while

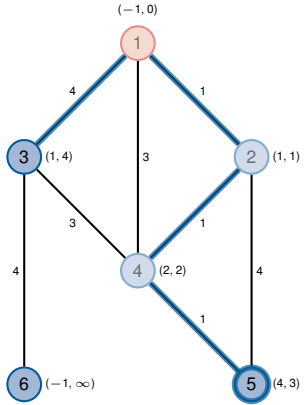


Dijkstras Algorithmus

- $p[i]$: 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 bearbeiteten 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**
      $Q.decreaseKey(j, d[i] + c_{ij})$
      $d[j] = d[i] + c_{ij}$
      $p[j] = i$
    end if
  end for
end while

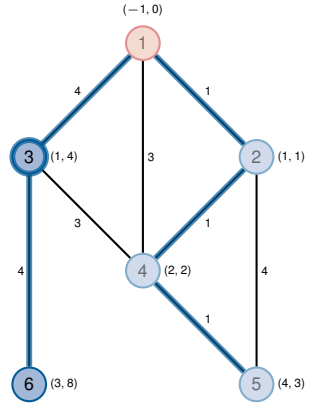


Dijkstras Algorithmus

- $p[i]$: 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 bearbeiteten 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**
            $Q.decreaseKey(j, d[i] + c_{ij})$
            $d[j] = d[i] + c_{ij}$
            $p[j] = i$
        end if
    end for
end while

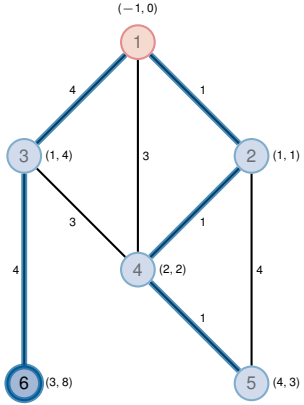


Dijkstras Algorithmus

- $p[i]$: 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 bearbeiteten 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**
            $Q.decreaseKey(j, d[i] + c_{ij})$
            $d[j] = d[i] + c_{ij}$
            $p[j] = i$
        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.
- Laufzeit in $\mathcal{O}(|N| \cdot |E|)$.

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 sind (Priority Queue).
- Vollständige Kenntnis der Netzwerktopologie erforderlich.
- Asymptotisch bessere Laufzeit.
- Laufzeit in $\mathcal{O}(|E| + |N| \log_2 |N|)$.

Routing Information Protocol (RIP) [8, 11]

Eigenschaften:

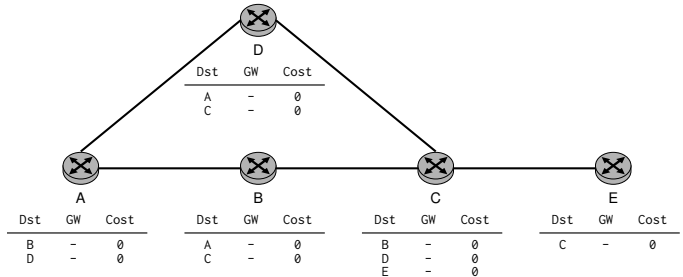
- Einfaches Distanz-Vektor-Protokoll
- RIPv1 standardisiert in RFC 1058 (1988)
- Unterstützung für CIDR in RIPv2 hinzugefügt (RFC 2453, 1998)
- Einzige Metrik: Hop Count (entspricht Bellman-Ford mit Kantengewicht 1 auf allen Kanten)
- Hop Count Limit von 15, weiter entfernte Ziele sind nicht erreichbar

Funktionsweise:

- Router senden in regelmäßigen Abständen (Standardwert 30 s) den Inhalt ihrer Routingtabelle an die Multicast-Adresse 224.0.0.9.
- Alle Geräte mit dieser Multicast-Adresse akzeptieren das Update.
- Jeder RIP-Router akzeptiert diese Update-Nachrichten, inkrementiert die Kosten 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.

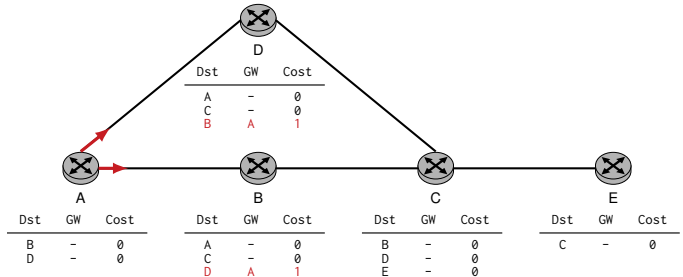
Beispiel mit vereinfachter Darstellung:

- 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, . . .).



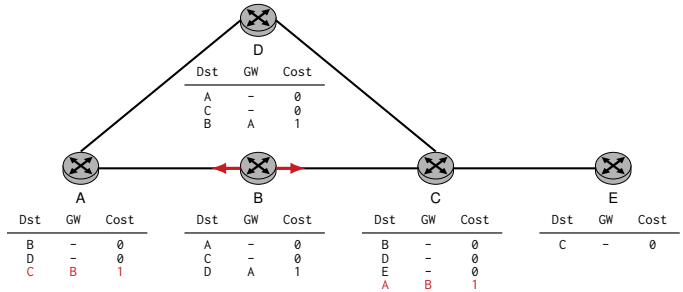
Beispiel mit vereinfachter Darstellung:

- 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, . . .).



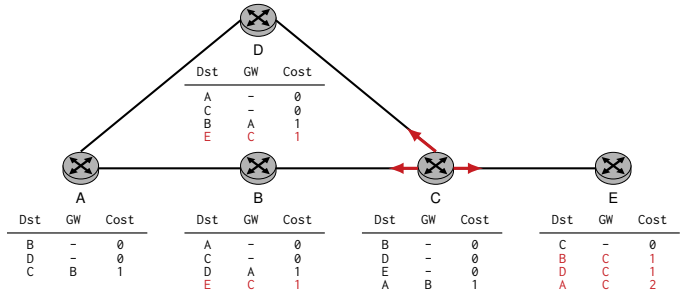
Beispiel mit vereinfachter Darstellung:

- 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, . . .).



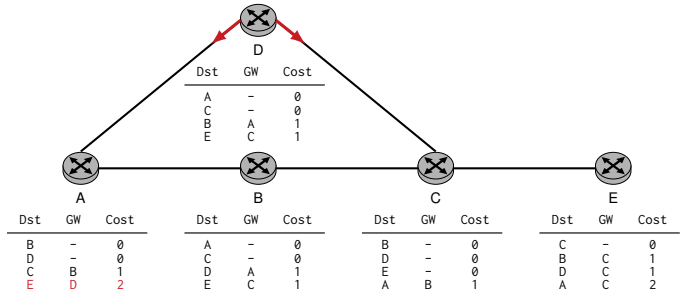
Beispiel mit vereinfachter Darstellung:

- 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, . . .).



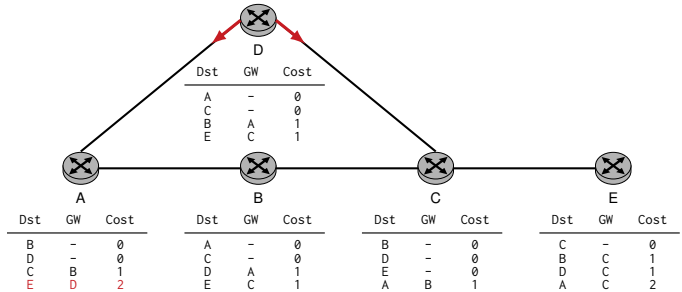
Beispiel mit vereinfachter Darstellung:

- 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, . . .).



- 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.

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.

Problem:

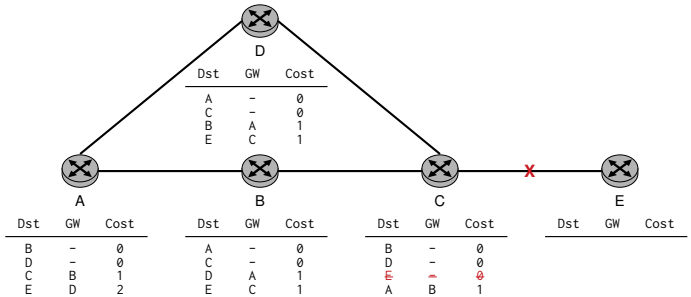
- Bis jeder Router den besten Next Hop bestimmen kann, dauert es ggf. mehrere „Runden“.
- Eine obere Schranke für die Anzahl der notwendigen Nachrichten, die jeder Router senden muss, ist die maximale Entfernung zwischen zwei Routern in Hops.
- Die maximale Entfernung beträgt bei RIP 15 Hops.
- Da Updates 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.

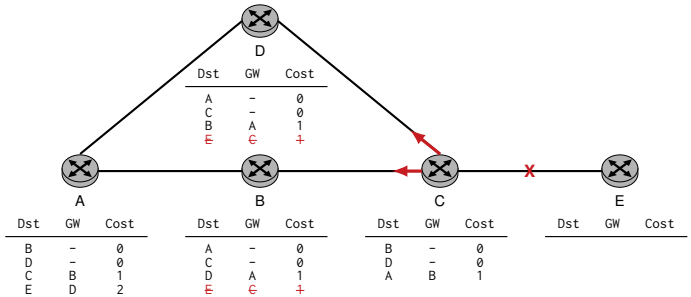
Anderes Problem: Count to infinity

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



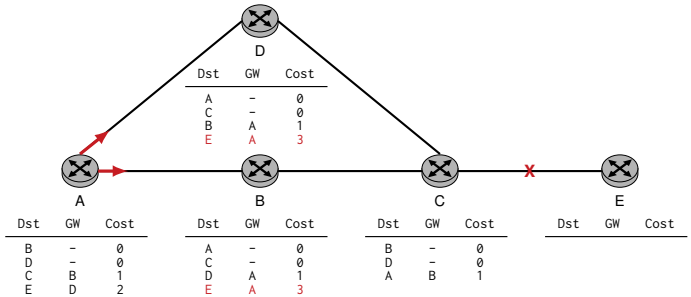
Anderes Problem: Count to infinity

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



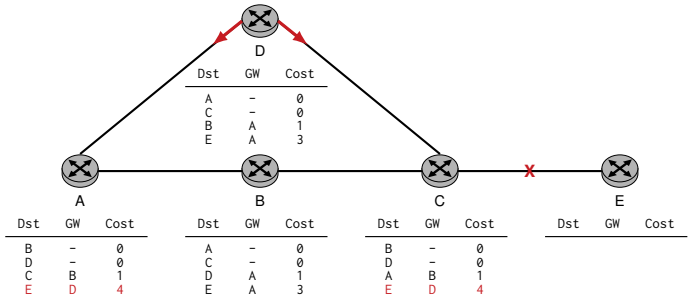
Anderes Problem: Count to infinity

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



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 inkrementiert
- bis schließlich das Hop Count Limit von 15 erreicht ist.

Diesen Vorgang bezeichnet man als Count to infinity.

Lösungsansätze für Count-to-Infinity:

- Split Horizon
  - „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 Protokollkomplexität.

Dynamisches Routing

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, da es sein eigenes L3-Protokoll mitbringt
- HWMP (Hybrid Wireless Mesh Protocol)
  Ermöglicht Routing in IEEE 802.11s (Wireless Mesh Networks), wobei Routing-Entscheidungen hier allerdings auf Basis von MAC-Adressen anstatt von IP-Adressen getroffen werden[4]

[4] MAC-based Routing ist ein Spezialfall für (kabellose) Meshnetzwerke, in dem sich nicht alle Knoten direkt erreichen aber dennoch eine gemeinsame Broadcast-Domain bilden.

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)

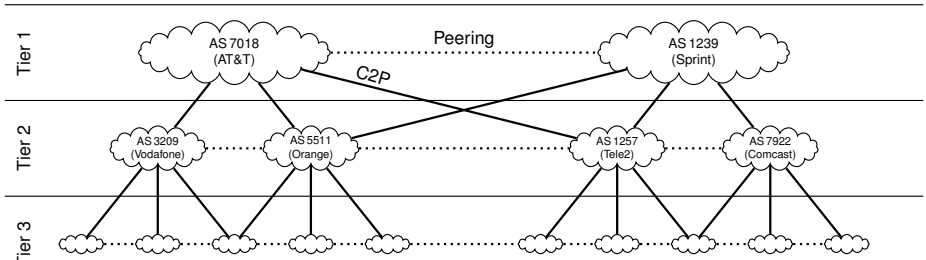
Routingentscheidungen auf Basis derartiger Kriterien bezeichnet man als Policy-Based Routing.

Autonomes System
Eine Menge von Netzwerken, die unter einheitlicher administrativer Kontrolle stehen, bezeichnet man als Autonomes System (AS). Ein AS wird durch einen 16 bit bzw. 32 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).

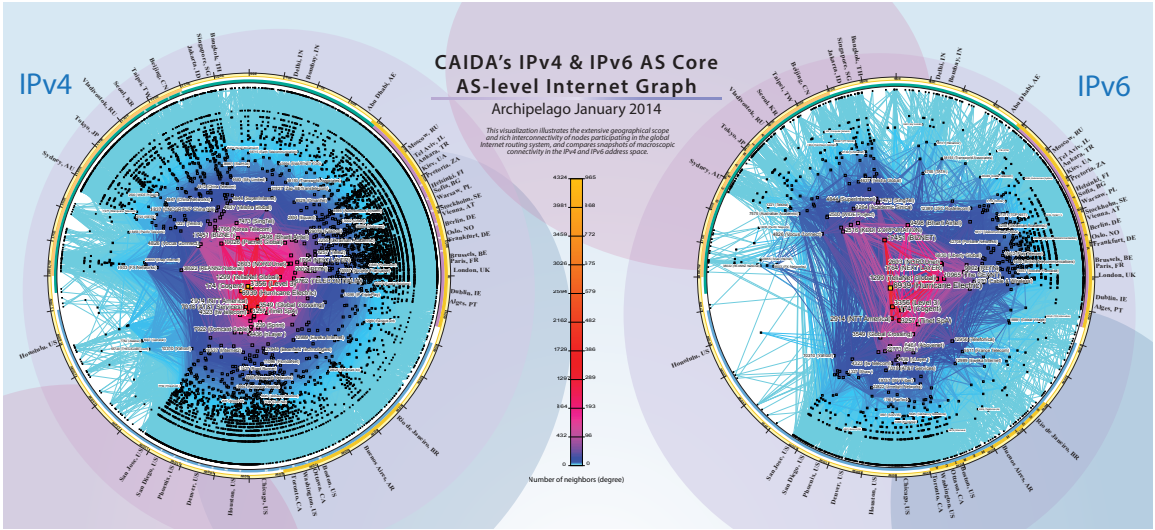
Das Internet: Stark vereinfachte schematische Darstellung



- 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 die eigenen Präfixe sowie die seiner Customer an seine Peerings und Upstream-Provider.
- AS 5511 würde die Netze von Vodafone nicht an Tele2 announcen, und auch nicht an Sprint oder AT&T.

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

Das Internet: Etwas weniger stark vereinfacht [9]



Kapitel 3: Vermittlungsschicht

Zusammenfassung

In diesem Kapitel haben wir

- die Vorteile von Paketvermittlung gegenüber Leitungs- und Nachrichtenvermittlung diskutiert,
- die Notwendigkeit logischer Adressen zur End-zu-End Adressierung erkannt,
- die beiden wichtigsten Protokolle für End-zu-End Adressierung im Internet kennen gelernt,
- Methoden zur weiteren logischen Unterteilung von Netzen in Subnetze und Präfixe behandelt und
- ein grundlegendes Verständnis bzgl. des Austauschs von Routinginformationen im Internet entwickelt.

Wir sollten nun wissen,

- was die Unterschiede zwischen Leitungs-, Nachrichten- und Paketvermittlung sind,
- worin der technische und logische Unterschied zwischen den Adressen auf Schicht 2 und 3 besteht,
- welche Möglichkeiten IPv4 und IPv6 zur logischen Strukturierung bieten und wie dies funktioniert,
- wie zu einer gegebenen Adresse auf Schicht 3 die zugehörige Link-Layer Adresse bestimmt wird,
- was eine Routing Tabelle ist,
- wie Router Weiterleitungsentscheidungen treffen,
- was der Unterschied zwischen Routing und Forwarding ist,
- wie Router untereinander Routen austauschen
- welche Arten von Routingprotokollen es gibt und
- wie diese funktionieren.

Literaturangaben

[1] Google IPv6 Statistiken, 2017.
https://www.google.de/ipv6/statistics.html#tab=ipv6-adoption&tab=ipv6-adoption.

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

[3] A. Conta and S. Deering.
Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, 2006.
http://tools.ietf.org/html/rfc4443.

[4] M. Cotton, L. Vegoda, R. Bonica, and B. Haberman.
Special-Purpose IP Address Registries, 2013.
http://tools.ietf.org/html/rfc6890.

[5] M. Crawford.
Transmission of IPv6 Packets over Ethernet Networks, 2007.
https://tools.ietf.org/html/rfc2464.

[6] S. Deering and R. Hinden.
Internet Protocol, Version 6 (IPv6) Specification, 1995.
http://tools.ietf.org/html/rfc1883.

[7] S. Deering and R. Hinden.
Internet Protocol, Version 6 (IPv6) Specification, 1998.
http://tools.ietf.org/html/rfc2460.

[8] C. Hedrick.
Routing Information Protocol, 1988.
http://tools.ietf.org/html/rfc1058.

[9] B. Huffaker, Y. Hyun, and M. Luckie.
IPv4 and IPv6 AS Core: Visualizing IPv4 and IPv6 Internet Topology at a Macroscopic Scale in 2014, 2014.
http://www.caida.org/research/topology/as_core_network/2014.

Literaturangaben

[10] S. Kawamura and M. Kawashima.
A Recommendation for IPv6 Address Text Representation, 2010.
http://tools.ietf.org/html/rfc5952.

[11] G. Malkin.
RIP Version 2, 1998.
http://tools.ietf.org/html/rfc2453.

[12] T. Narten, R. Draves, and S. Krishnan.
Privacy Extensions for Stateless Address Autoconfiguration in IPv6, 2007.
https://tools.ietf.org/html/rfc4941.

[13] E. Nordmark, W. Simpson, and H. Soliman.
Neighbor Discovery for IP version 6 (IPv6), 2007.
http://tools.ietf.org/html/rfc4861.

[14] L. L. Peterson and S. D. B.
Computer Networks – A System Approach, chapter Internetworking, pages 234–242.
Elsevier, 4. edition, 2007.

[15] L. L. Peterson and S. D. B.
Computer Networks – A System Approach, chapter Internetworking, pages 248 – 256.
Elsevier, 4. edition, 2007.

[16] L. L. Peterson and S. D. B.
Computer Networks – A System Approach, chapter Internetworking, pages 259 – 262.
Elsevier, 4. edition, 2007.

[17] S. Thomson, T. Narten, and T. Jinmei.
IPv6 Stateless Address Autoconfiguration, 2007.
http://tools.ietf.org/html/rfc2462.