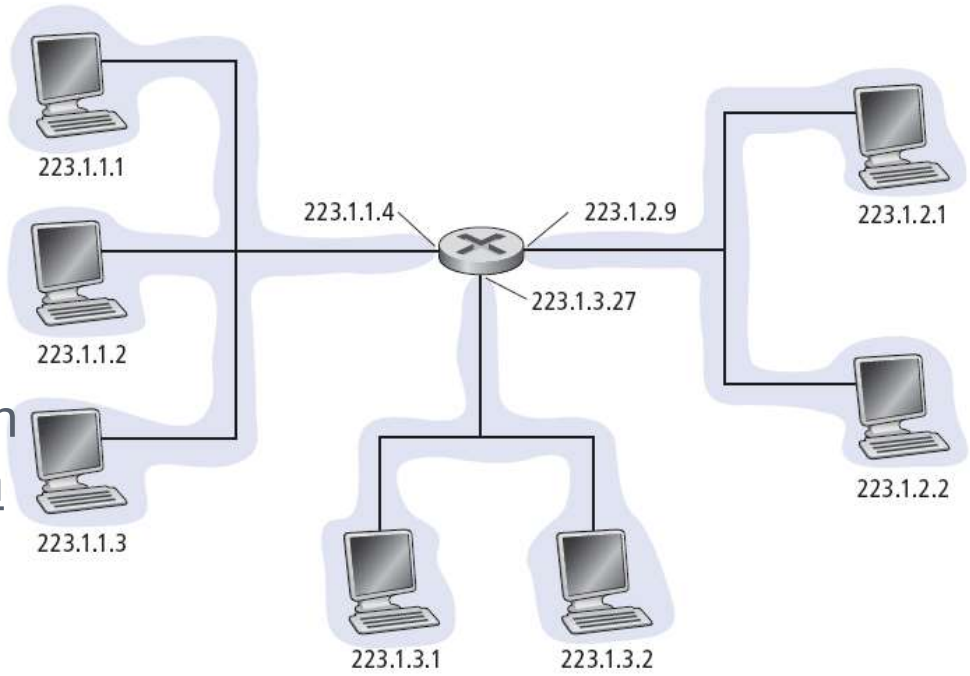
Betriebssysteme und Netzwerke Vorlesung N06

Artur Andrzejak

Netzwerkschicht:
Das Internetprotokoll (IP) – Details der Adressierung (Fortsetzung)

IP-Adressierung – Netzwerke (Wiederholung)

- IP-Adressen haben zwei Bestandteile:
  - **Netid: Netzwerkteil**: Die oberen Bits der Adresse, identifizieren ein Netzwerk
  - **Hostid: - Hostteil**: Die unteren Bits der Adresse, identifizieren ein Interface innerhalb des Netzwerks
- Wenn der Adressenbereich eines Netzwerks 2^k Adressen umfasst, dann
  - Hostid hat die unteren k Bits
  - Netid hat die oberen 32-k Bits



- Z.B. das untere Netzwerk hat 256 IP-Adressen:
  - 8 (untere) Bits als Hostteil
  - 24 obere Bits (223.1.3.*) als Netzwerkteil

Wozu überhaupt Netid und Hostid?

- Da alle Interfaces in einer Firma / Uni / Organization X (i.A.) im gleichen Netzwerk sind, haben sie gleiche Netid (d.h. gleiche obere Bits in Adressen)
  - Die Router außerhalb von X müssen nur die Netid (von X) betrachten, um die Pakete korrekt weiterzuleiten
  - Ähnlich wie Vorwahlnummern bei Telefonen
- Das hat einige Vorteile – welche?

1. Die Daten in den Routertabellen sind kleiner, da es viel weniger Netids als IP-Adressen gibt
2. Jede Organization kann intern beliebig die Hostids zu Interfaces zuordnen und braucht das nach „draußen“ nicht mitzuteilen

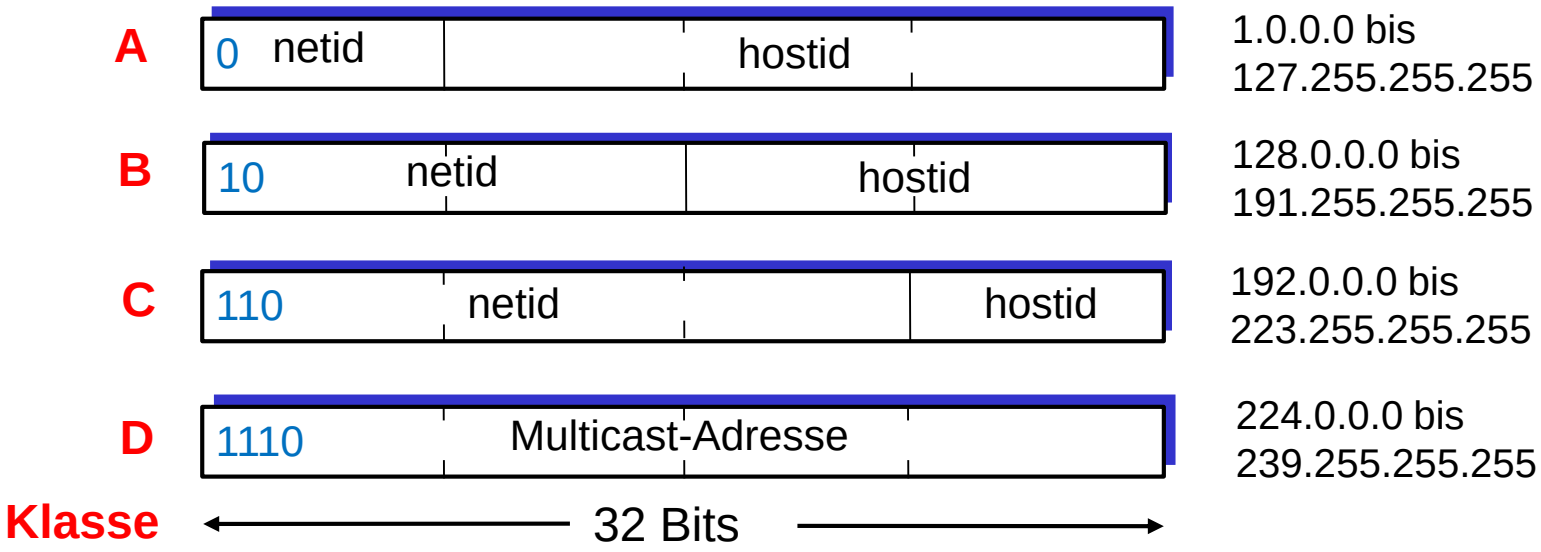
Details der Adressierung

Wie groß soll Netid (und damit Hostid) sein?

- Längeres Netid:
  - Mehr Netzwerke, aber jedes hat weniger Adressen
- Kürzeres Netid:
  - Wenige Netzwerke, aber jedes mit sehr viel Adressen
- Offenbar ist es sinnvoll, mehrere Längen von Netids zu haben – abhängig von der Größe des Netzwerkes
- Man hat (zunächst) nur drei Längen (Klassen) von Netids gehabt – **klassenbasierte Adressierung**
- Ab 1993 wurden diese drei Klassen auf viele (ca. 30) wesentlich kleinere aufgeteilt - **Classless Inter-Domain Routing**

(Alte) **Klassenbasierte Adressierung**

- IP-Adressen sind aufgeteilt in **Adressklassen**
- Die Klasse bestimmt #Bits von netid (=> auch #Bits hostid)
- Dies nennt man „**classfull**“ **addressing** oder **klassenbasierte Adressierung**



Video: IP Addressing and How it Works,
https://www.youtube.com/watch?v=KFooN7Mu0IM

Diskussion Klassenbasierte Adressierung

- Verteilung der Adressen:
  - Durch zentrale Organisationen (z.B. IANA oder ICANN: Internet Corporation for Assigned Names and Numbers)
  - Netzweise (also z.B. Klasse-B-Netz für ein Unternehmen)
  - Relativ chaotisch: Zuteilung der numerisch nächsten netid an den nächsten Nachfrager

So ähnlich wie FIFA des Internets …

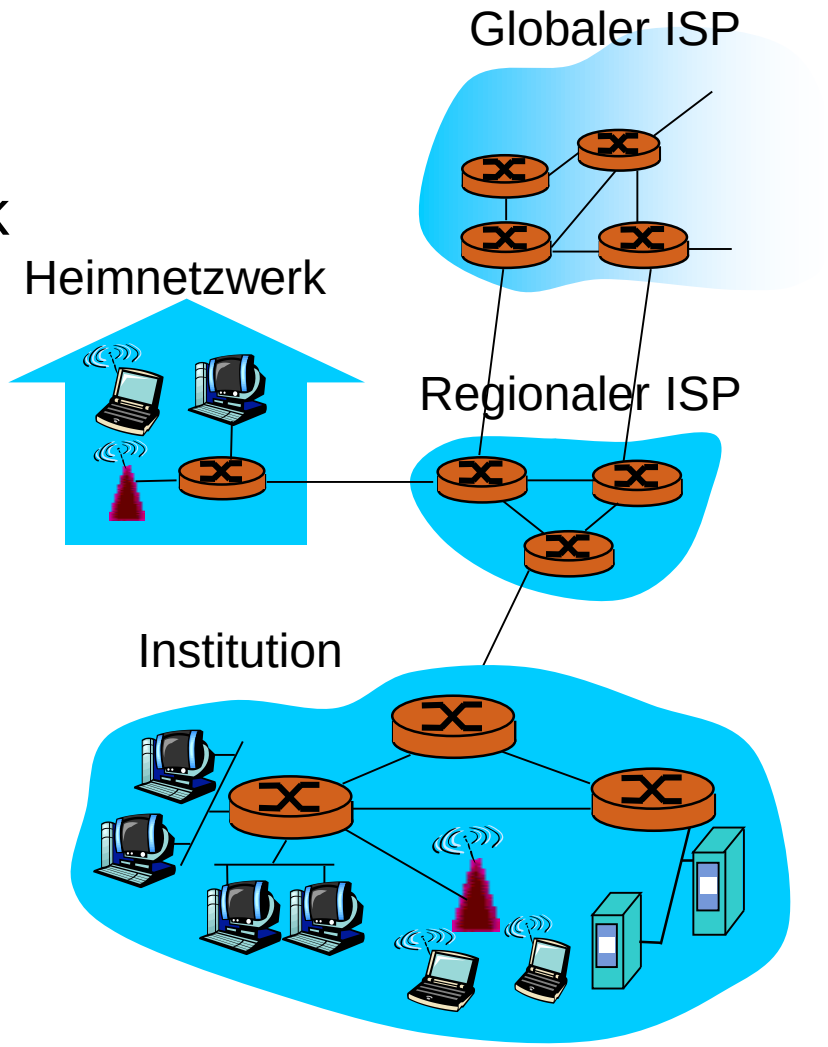
- Probleme?
  - Verschwendung von IP-Adressen, z.B. Class-B-Adressen
    - Besitzer könnte 2^{16} (>= 65.000) Adressen vergeben
    - Nur ein Bruchteil davon wird vielleicht genutzt
  - Routing-Tabellen werden schnell sehr groß
    - Ein Eintrag für jede netid
    - Keine Möglichkeit, Einträge zusammenzufassen
  - Routing-Tabellen müssen mit hoher Frequenz aufgefrischt werden
    - Wenn ein Netzwerk hinzukommt, wegfällt oder sich verändert, muss dies im ganzen Internet bekanntgegeben werden

Klassenlose Adressierung

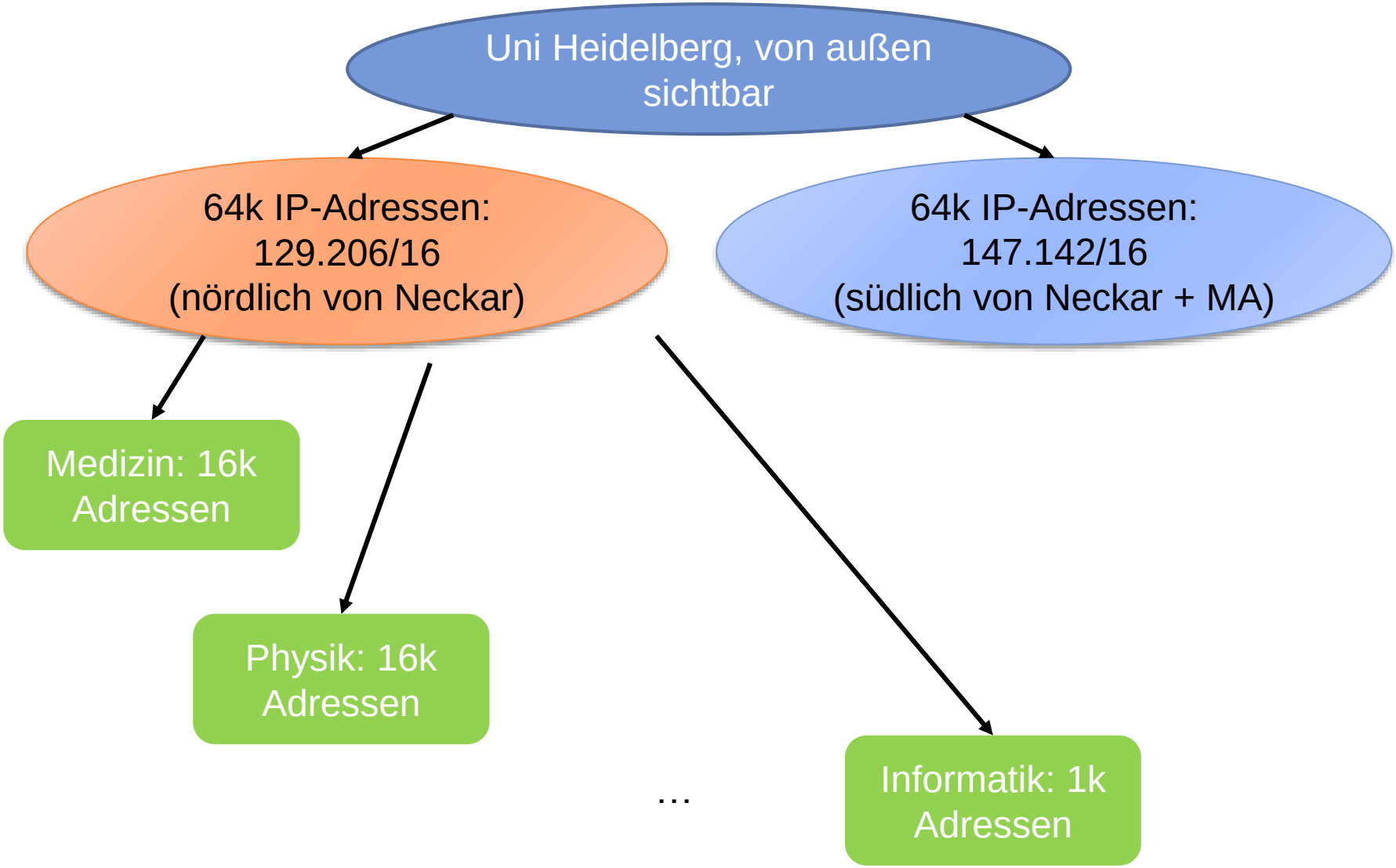
- Ab 1993 erlaubt man beliebige Längen von Netid
- Das Schema heißt **CIDR**: **C**lassless **I**nter**D**omain **R**outing
- Schreibweise:
  - a.b.c.d/**x**, wobei **x** die Länge der netid (hier auch Präfix) bestimmt
  - Angabe wie a.b.c.d/x legt eindeutig den Bereich von IP-Adressen fest, die einer Organisation gehören
- Welche Netzwerkgrößen (= Anzahl Adressen pro Netzwerk) sind damit möglich?
- Theoretisch jede Zweierpotenz 2^{32-x} für x = 1 bis 31

Router verschiedener Hierarchien

- Die Router der ISPs betrachten nur die Netids
- Der Router im Heimnetzwerk kennt (i) die Hostids der Geräte im Netzwerk UND (ii) die IP-Adresse des Routers "nach außen"
- Institutionen wollen oft ihre Adressenbereiche aufteilen
- Deshalb hat man oft innerhalb der Institutionen weitere **Subnetze**

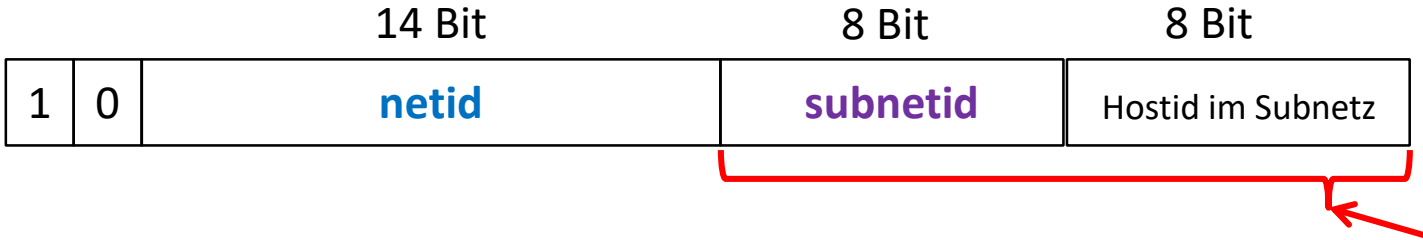


Beispiel Subnetze



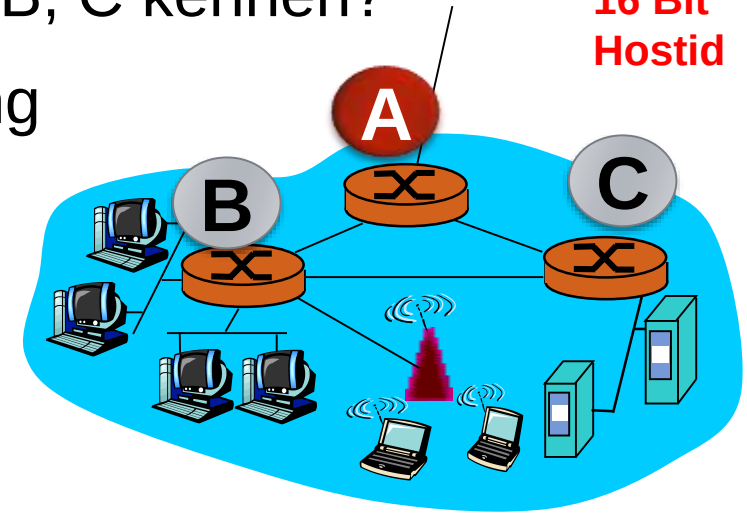
Adressierung von Subnetzen /1

- Institutionen wollen oft ihre Adressenbereiche aufteilen
- Daher teilt man die hostid weiter auf, z.B. so:



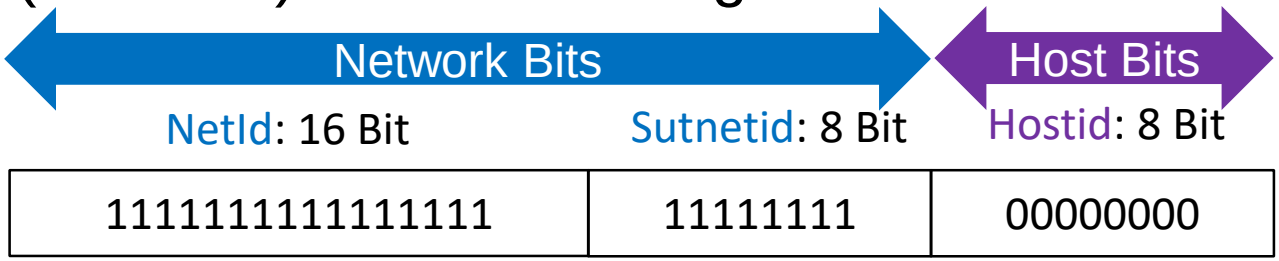
Welche Daten müssen Router A, B, C kennen?

- Der Router A kennt die Zuordnung der Subnetid zu den Routern „unter ihm“ (wie B, C)
- Die Router B, C, …, gehören zu je einem Subnetz und kennen die IP-Adressen aller Interfaces darin sowie die netid+subnetid der lokalen Router (A, B, C)



Subnetzmaske /1

- **Subnetzmaske** (**subnet mask**) identifiziert, welcher Teil der Adresse uns zu dem Subnetz führt (Netid + Subnetz-Id), welcher (restliche) Teil zu Hostid gehört

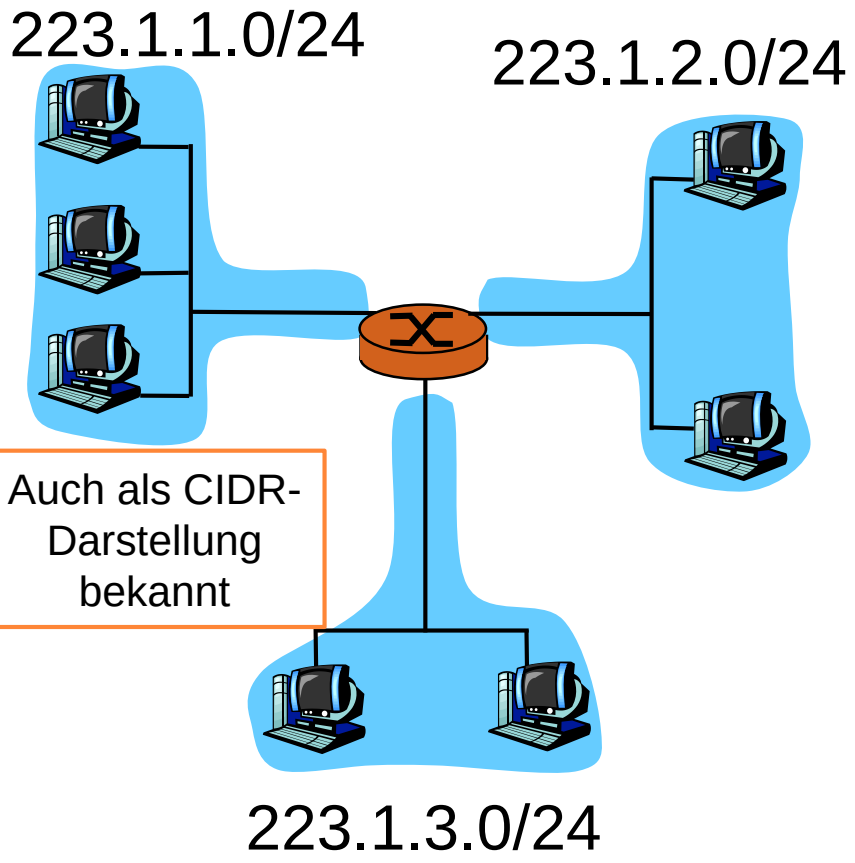


subnet mask: 0xffffff00=**255.255.255.0** oder auch **/24**

- Die eigene IP-Adresse in Verbindung mit der Subnetzmaske erlaubt Rückschlüsse darüber, wo sich eine andere IP-Adresse befindet:
  - Im selben Subnetz (also direkt erreichbar)
  - Im selben Netzwerk, aber in einem anderen Subnetz
  - In einem anderen Netzwerk

Subnetzmaske /2

- Die **Subnetzmaske** eines Subnetzes gibt die Länge von netid PLUS Subnetz-Teil an
- **Längenschreibweise**: **/x** – die linken **x** Bits gehören zu dem Netid+Subnetid-Teil der Adresse
  - z.B. 223.1.3.0/**24**
- **Maskenschreibweise**: wie eine IP-Adresse, die für den Netid+Subnetid-Teil 1-en hat (sonst 0-en)
  - z.B. 255.255.255.**0** (entspricht /24)



200.23.16.0/24

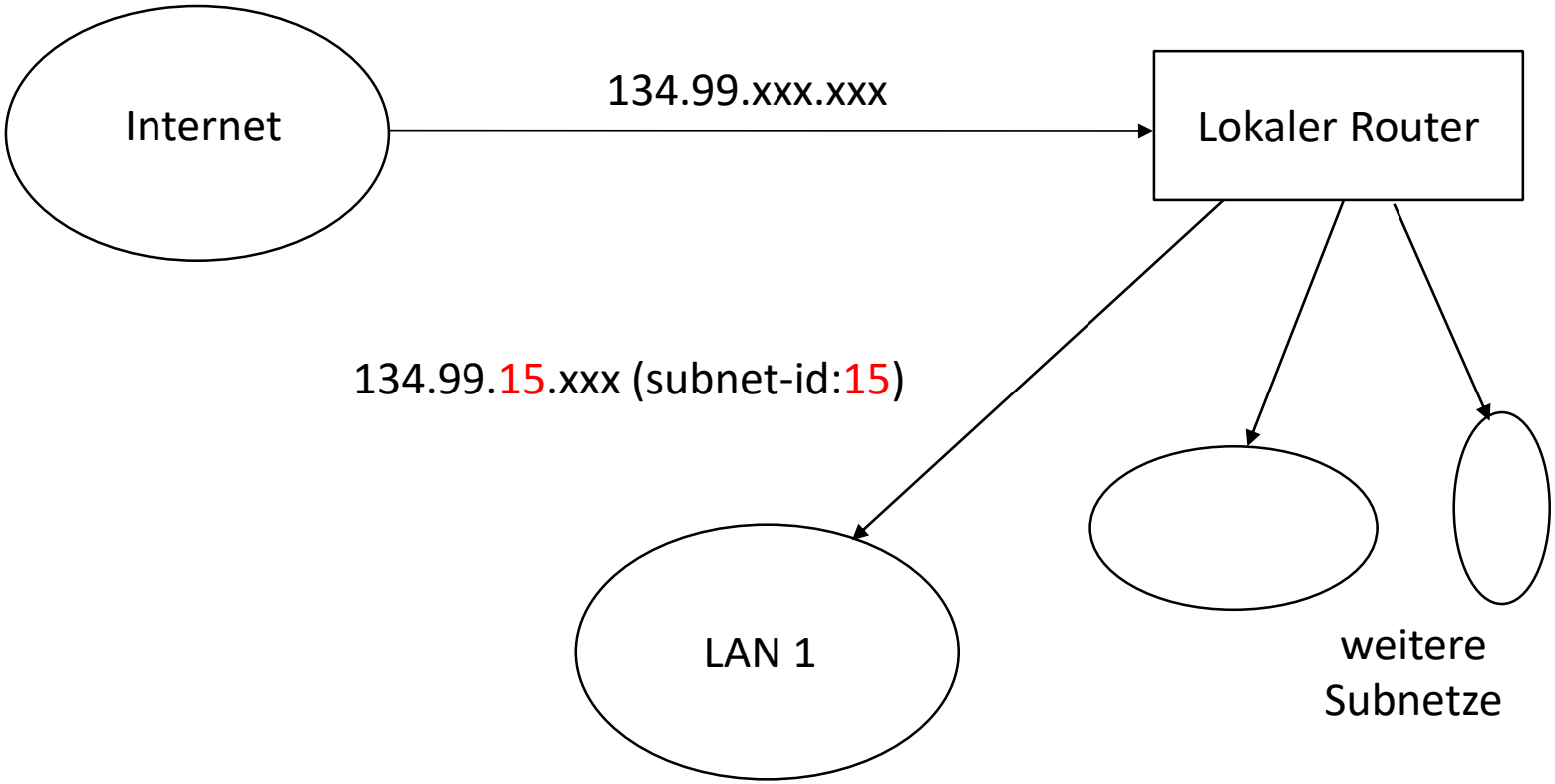
Subnetzmaske: /24
oder 255.255.255.0

Beispiel für die Verwendung von Subnetzmasken

- Gegeben:
  - Eigene IP-Adresse: 134.155.48.10
  - Subnetzmaske: 255.255.255.0
  - Fremde IP-Adressen
    - A: 134.155.48.96, B: 134.155.55.96
- Überprüfen der beiden Adressen – welche Subnetze?
  - Subnetz der eigenen IP-Adresse: 134.155.48.10 & 255.255.255.0 = 134.155.48.0
  - fremde Adr. A: 134.155.48.96 & 255.255.255.0 = 134.155.48.0 => gleiches Subnetz
  - fremde Adr. B: 134.155.55.96 & 255.255.255.0 = 134.155.55.0 => verschieden, anderes Subnetz

Adressierung von Subnetzen /2

- Die subnetid ist außerhalb des Netzwerkes, für das sie verwendet wird, nicht sichtbar:



Video - Subnetzmasken

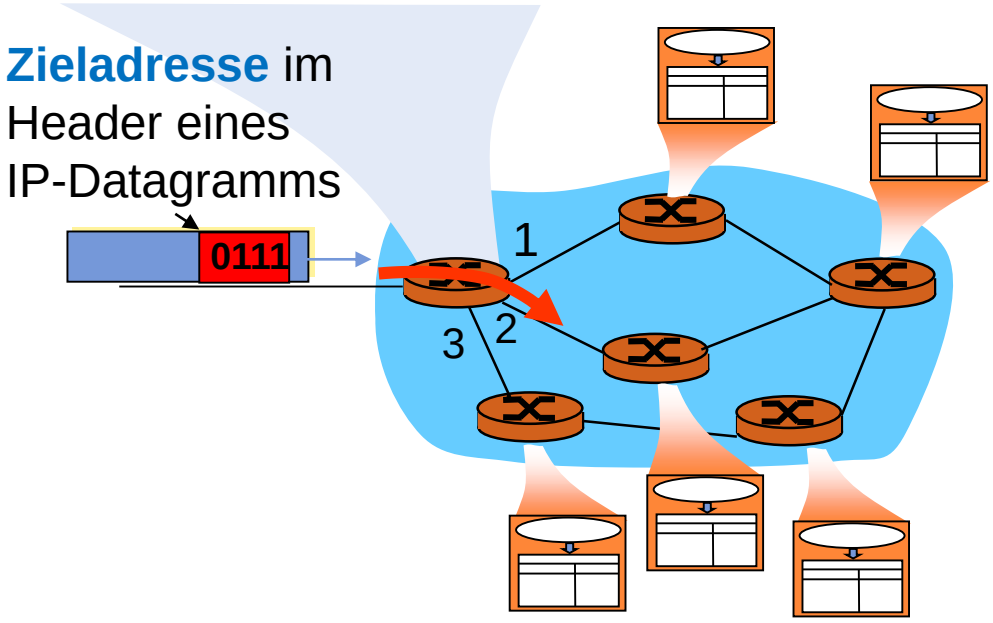
- Understanding an IP Address: Cisco Router Training 101
  - https://www.youtube.com/watch?v=LxNgWsseE0w
  - Ab 16:30 (min:sec) [06a]

Router

Hauptaufgabe des Routers: Weiterleitung

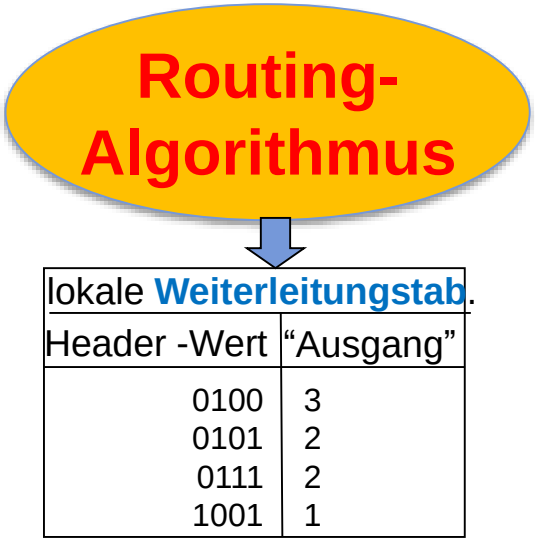
- **Weiterleitung** (**Forwarding**)
  1. Router liest die **Zieladresse** des Pakets ab
  2. Findet einen passenden Eintrag in der **Weiterleitungs-tabelle** (**forwarding table**)
  3. Leitet das Paket über den gefundenen Ausgang weiter

lokale **Weiterleitungstab.**	
Header -Wert	"Ausgang"
0100	3
0101	2
0111	2
1001	1

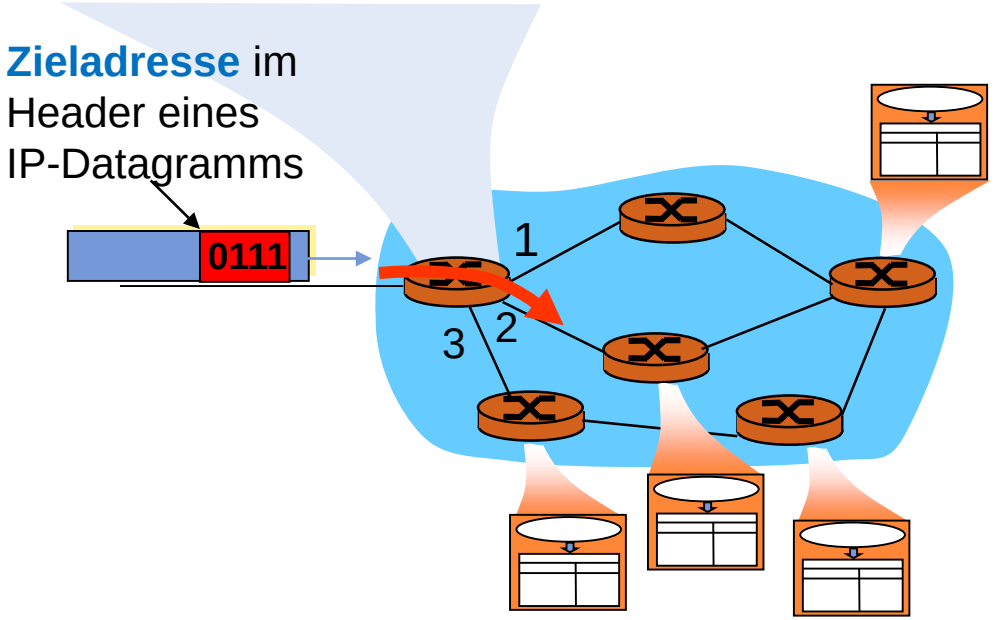


Forwarding und Routing

- Ein Router macht eine lokale Entscheidung: Paket an einen Ausgang schicken
- Warum kommen die Pakete damit im Ziel an?
- D.h. woher kommt die „Weitsicht“?



Antwort:
Die (größeren) Router führen ab und zu einen **Routing-Algorithmus** aus, um die Weiterleitungstabelle zu aktualisieren



Weiterleitungstabelle

Leite diesen Bereich der IP-Adressen an Ausgang 0

Intervall der Zieladressen	Interface (Ausgang)
11001000 00010111 00010000 00000000 bis 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 bis 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 bis 11001000 00010111 00011111 11111111	2
Sonst	3

Leite diesen Bereich der IP-Adressen an Ausgang 2

Weiterleitungstabelle – Kompakt /1

	Intervall der Zieladressen	Ausgang
A	11001000 00010111 00010000 00000000 bis 11001000 00010111 00010111 11111111	0
B	11001000 00010111 00011000 00000000 bis 11001000 00010111 00011000 11111111	1

- In (älteren) Routern musste man Speicherplatz sparen
- Wie können wir die linke Spalte kompakt kodieren?
- Wie können wir z.B. bei B die (Adressen-Untergrenze, Adressen-Obergrenze) durch einen Eintrag ersetzen?

Weiterleitungstabelle – Kompakt /2

	Intervall der Zieladressen	Ausgang
A	11001000 00010111 00010000 00000000 bis 11001000 00010111 00010111 11111111	0
B	**11001000 00010111 00011000** 00000000 bis **11001000 00010111 00011000** 11111111	1

- Wie kann man mit nur <u>einem</u> Eintrag auskommen?
- Bei B: die „blauen“ Bits sind für Weiterleitung irrelevant
- => Behalte für B nur <u>einen</u> Eintrag in der Form:
  - **11001000 00010111 00011000** ********

Weiterleitungstabelle – Kompakt/3: Präfixe

- Wir kodieren also Adress-Intervalle durch **Präfixe**
- **Präfix** = k obere Stellen einer IP-Adresse, die in der unteren und oberen Adr.-Grenzen übereinstimmen

Bereich der Zieladressen	Präfix?
11001000 00010111 00010000 00000000 bis 11001000 00010111 00010111 11111111	**11001000 00010111 00010**
11001000 00010111 00011000 00000000 bis 11001000 00010111 00011000 11111111	**11001000 00010111 00011000**
11001000 00010111 00011001 00000000 bis 11001000 00010111 00011111 11111111	**11001000 00010111 00011**

Weiterleitungstabelle – Kompakt/4: Präfixe

- **Präfix** = k obere Stellen einer IP-Adresse, die in der unteren und oberen Adr.-Grenzen übereinstimmen
- Unsere Weiterleitungstabelle sieht dann so aus:

Präfix	Ausgang
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
sonst	3

Präfixlänge und Subnetzgröße

- Die Weiterleitungstabelle besteht aus solchen Präfixen und den zugehörigen Ausgängen
- Job des Routes: für jede Zieladresse wird ein passender Präfix gefunden und das Paket an den zugehörigen Ausgang weitergeleitet
- Wenn Präfix P die Länge k hat, wie viele IP-Adressen leitet der Router an den Ausgang zu P?
- Router leitet $N = 2^{(32-k)}$ verschiedene Adressen an diesen Ausgang
  - Davon sind i.A. nur N-2 benutzbar
- Aber muss N eine Zweierpotenz sein? Gleich …

Longest-Prefix-Matching

Präfix	Ausgang
11001000 00010111 00010	0
11001000 00010111 00011000	A1
11001000 00010111 00011	A2
sonst	3

- Der Präfix zu A2 ist in Präfix zu A1 „enthalten“
- Die Regel **Longest Prefix Matching** (**LPM**) sagt:
  - *Passen mehrere Präfixe zu einer Zieladresse, nehmen wir den Ausgang mit der längsten Prefix-Übereinstimmung*
- Beispiel –Ausgang A1 oder A2?

Ziel P: 11001000 00010111 00011110 10100001 — an A2

Ziel Q: 11001000 00010111 00011000 10101010 — an A1 (LPM)

Longest-Prefix-Matching: Verwendung

Präfix mit Länge k	Ausgang	k	Anz. Adressen
11001000 00010111 00011000	A1	24	$2^{32-24} = 2^8$
11001000 00010111 00011	A2	21	?

- Ein Teil der IP-Adressen zum A2 geht wegen der LPM-Regel an den A1
- Wir können damit aus dem Adressenbereich zu A2 einen Teil „herausschneiden“
  - #Adressen zu A2 ist keine 2er-Potenz mehr!
- Wie viele Adressen hat A2?

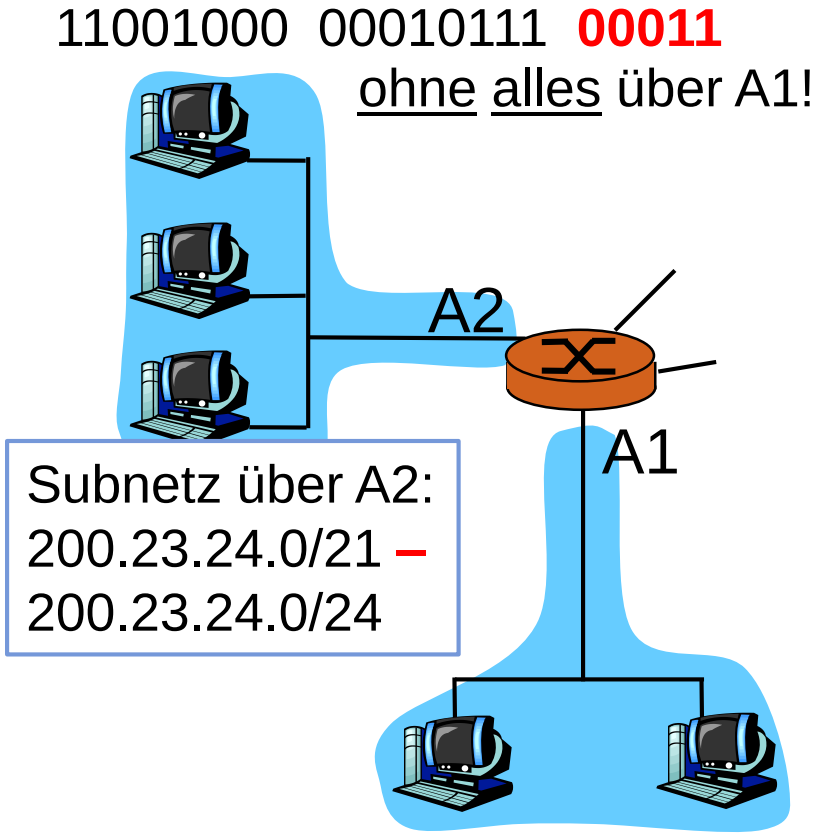
#Adressen zu A2:
$2^{32-k} - 2^{32-j}$

#Adressen zu A1:
2^{32-j}

$2^{32-21} - 2^8 = 2^{11} - 2^8$

Beschreibung der Subnetze

- Zur Erinnerung: Angabe der Adresse eines Subnetzes
  - **Längenschreibweise**: z.B. 223.1.3.0/24
  - **Maskenschreibweise**: z.B. 255.255.255.0 (entspricht /24)
- Problem bei Ausgang A2: wir haben einen Teil der Adressen herausgenommen!
- Lösung: schreibe das in der Form **a.b.c.d/x – e.f.g.h/y**
- D.h. „großer Bereich“ **–** „Teilmenge“
- Beispiel für Ausgang A2?

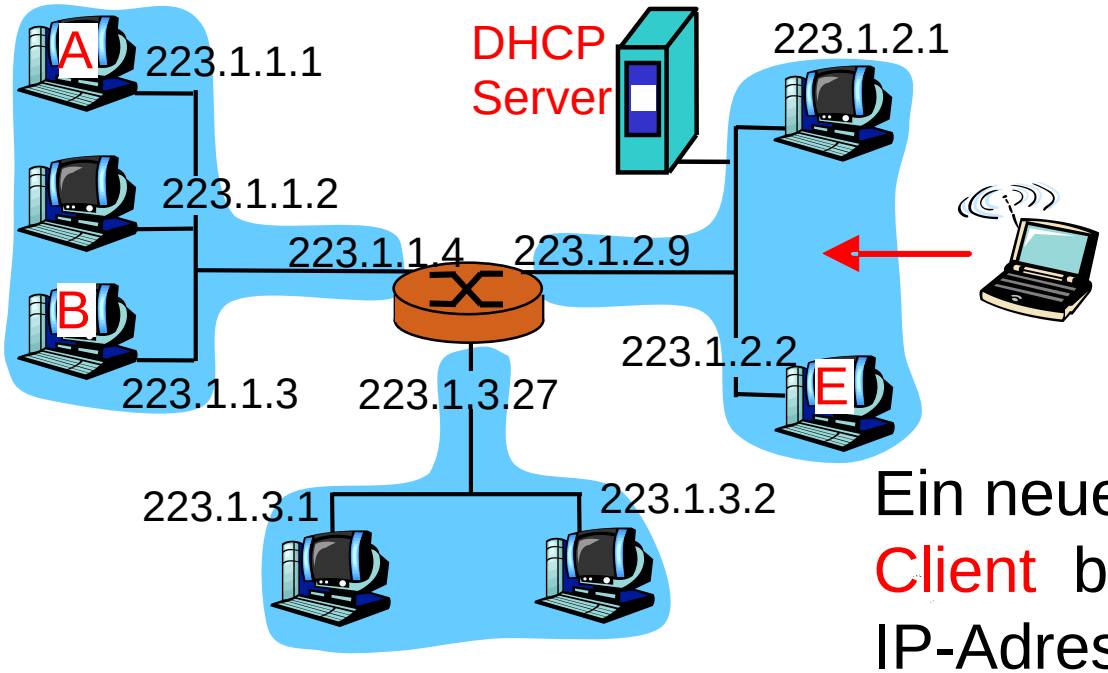


Das Internetprotokoll (IPv4) – DHCP, NAT, ICMP

Wie bekommt ein Host seine IP-Adresse?

- Manuell festgelegt durch den Systemverwalter
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- Oder durch den **DHCP**: Dynamic Host Configuration Protocol
  - Einer oder mehrere **Server** und ein **Protokoll** – RFC 2131
  - Server vergibt dynamisch den Hosts eine IP-Adresse
  - Diese Adressen-Zuordnung heißt **Lease** und gilt nur für gewisse Zeit
- Aufgaben des DHCP-Systems
  - Teilt den Hosts die IP-Adressen dynamisch zu, sobald sie dem Netzwerk (d.h. dem Subnetz) beitreten
  - Ermöglicht das Erneuern der Adressen-Zuordnung
  - Ermöglicht das “Recycling” von Adressen
    - Hosts werden diese entzogen, wenn sie nicht online sind

DHCP Client-Server Szenario

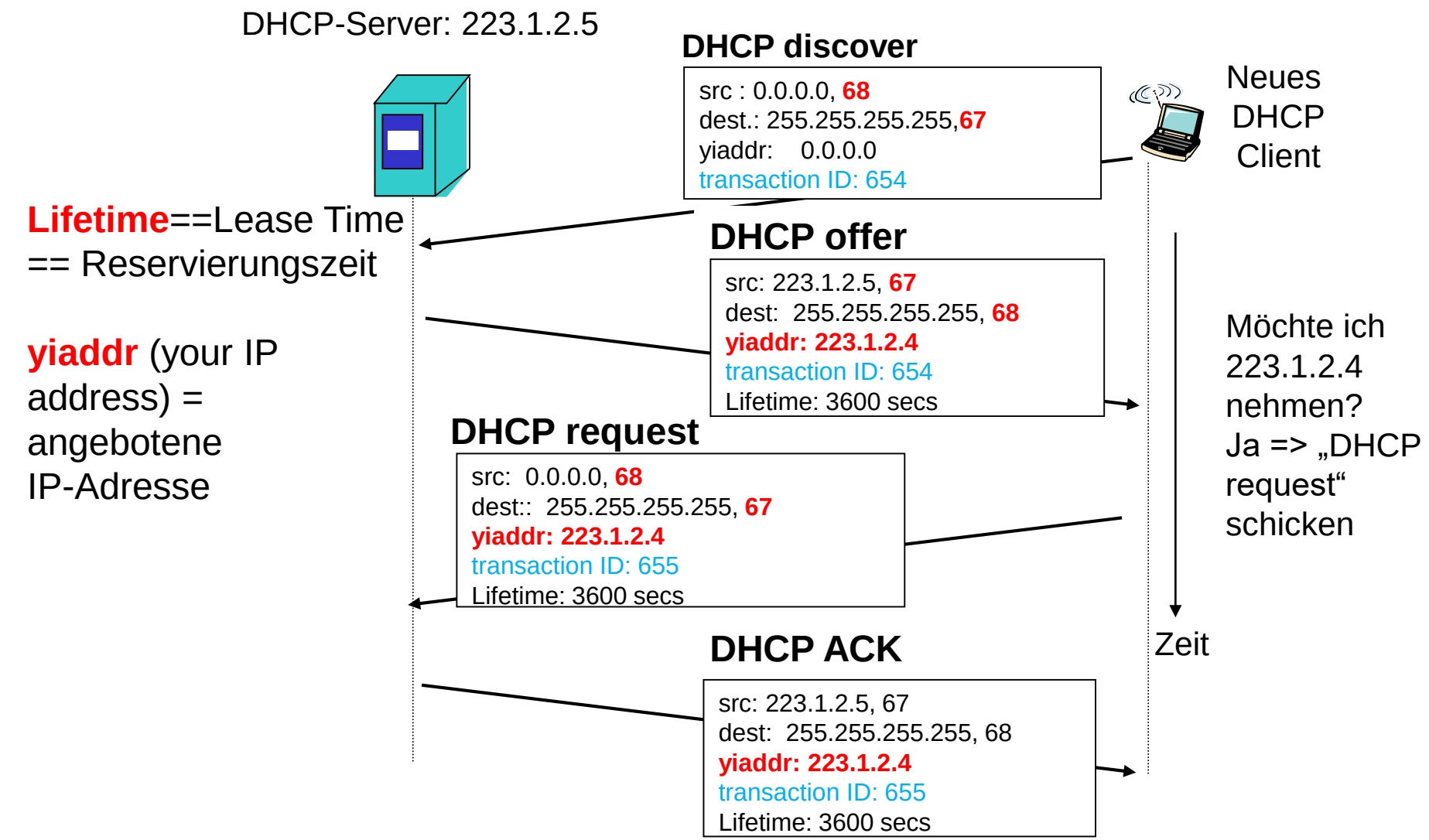


Ein neues DHCP Client braucht eine IP-Adresse in diesem Netzwerk; dafür wird das DHCP-Protokoll verwendet

DHCP Übersicht (RFC-Link hier)

- Host schickt die Nachricht **DHCP discover** (Broadcast)
  - UDP-Paket an Port 67
  - Broadcast-IP-Zieladresse 255.255.255.255, Quelladresse: 0-en
- DHCP-Server antwortet mit **DHCP offer** (Broadcast)
  - UDP-Paket an Port 68
  - „Each server may respond with a DHCPOFFER message that includes an available network address in the **yiaddr** field (and other configuration parameters in DHCP options)" (RFC-Punkt 2)
- Host wählt einen DHCP-Server aus und schickt ihm die Nachricht **DHCP request**
  - Nicht ausgewählte Server interpretieren das als Ablehnung
- DHCP-Server schickt dem Client **DHCP ack**
- Vorzeitige Aufgabe: Client schickt **DHCP release**

DHCP Client-Server Austausch



DHCP: Request und ACK in Wireshark

Request

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) **Requested IP Address = 192.168.1.101**
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
    Length: 11; Value: 010F03062C2E2F1F21F92B
    1 = Subnet Mask; 15 = Domain Name

The 'requested IP address' option MUST be set to the value of 'yiaddr' in the DHCPOFFER message from the server. (RFC, Punkt 3)

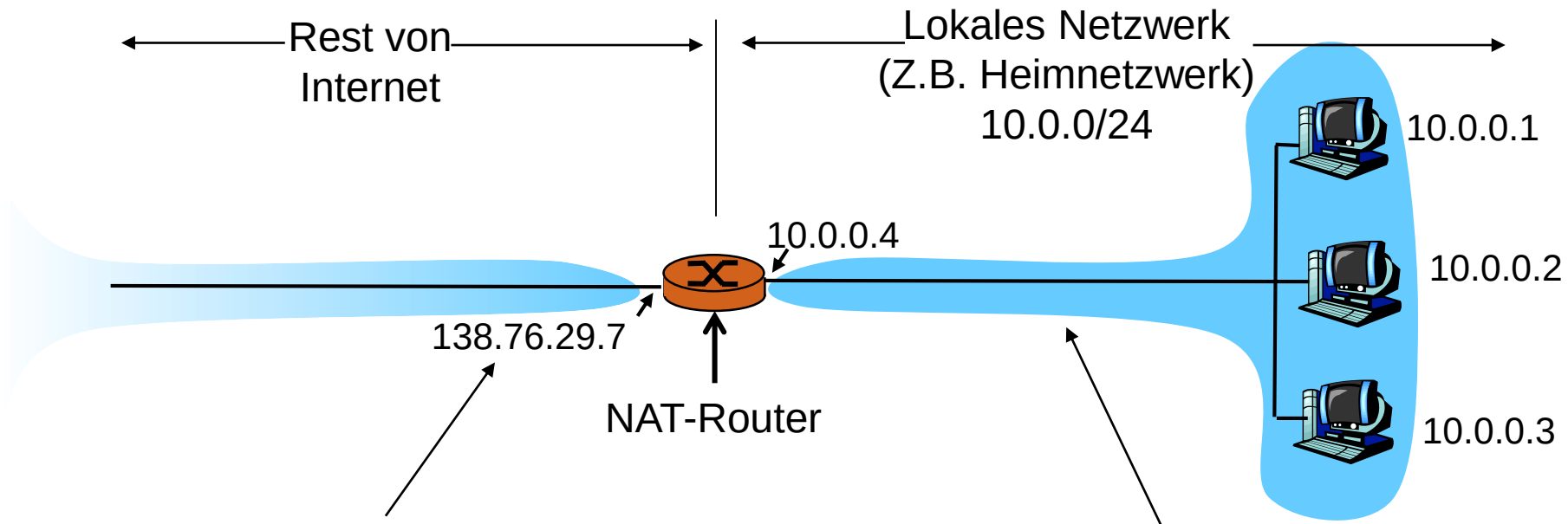
ACK

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
    Length: 12; Value: 445747E2445749F244574092;
    IP Address: 68.87.71.226;
    IP Address: 68.87.73.242;
    IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

NAT: Network Address Translation

- **Network Address Translation** (**NAT**) sind Verfahren, die automatisiert IP-Adressen durch andere ersetzen, um verschiedene Netze zu verbinden
  - Für Anwender ist **Source-NAT** interessant: Es wird die Adresse des Computers umgeschrieben, der die Verbindung aufbaut
- Motivation: Lokales Netzwerk (LN) besitzt für die Außenwelt nur eine einzige IP-Adresse
  - => Kompensiert die Knappheit öffentlicher IPv4-Adressen
  - => Man kann die Adressen im LN ändern, ohne die Außenwelt informieren zu müssen
  - Bietet zusätzliche Sicherheit, da die internen Eigenschaften / Hosts des LNs nach außen unsichtbar bleiben

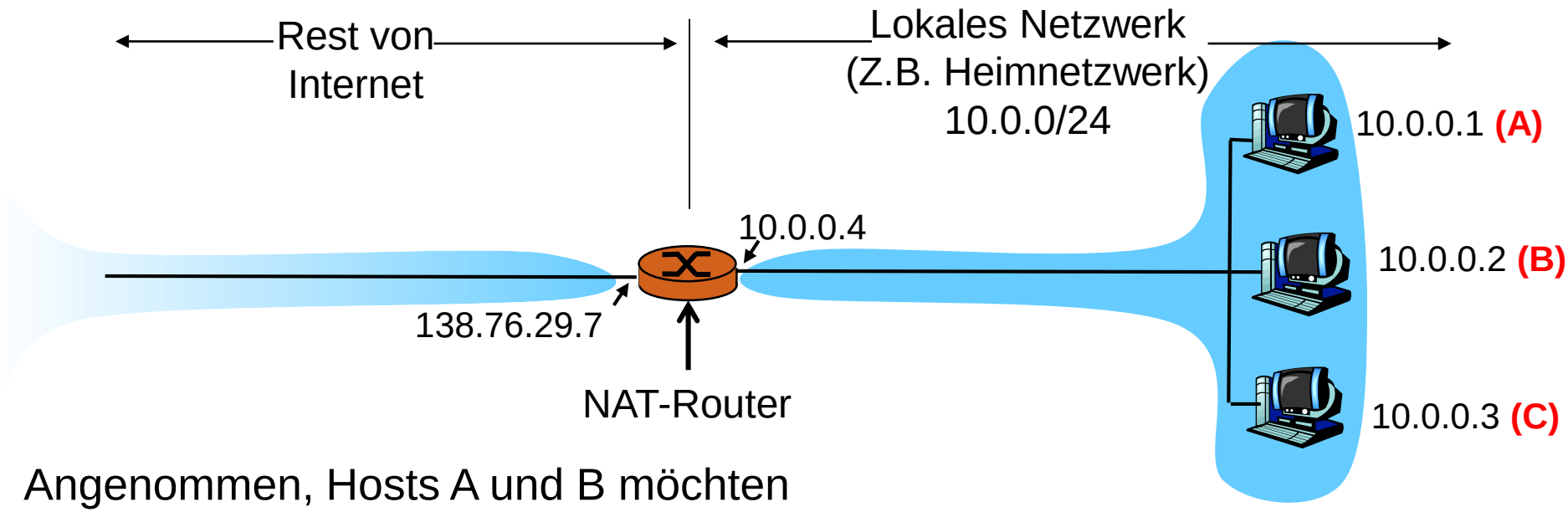
Source-NAT – Übersicht



Alle Datagramme, die den NAT-Router verlassen, haben dieselbe Quell-IP-Adresse: 138.76.29.7

Datagramme, die in diesem Netzwerk erzeugt wurden, haben 10.0.0/24 als Bereich der Quell-IP-Adressen

Source-NAT - Prinzip



Angenommen, Hosts A und B möchten je eine TCP-Verbindung zu einem Web-Server eröffnen, und haben jeweils den lokalem Port 5678 gewählt

- Falls das geht, wie kann der NAT-Router die Antworten des Webservers für A und für B unterscheiden?

Der Router ersetzt die Quell-Port-Nummern (in den ausgehenden Paketen) mit verschiedenen Quell-Port-Nummer für A und für B; Router übersetzt damit die Antworten

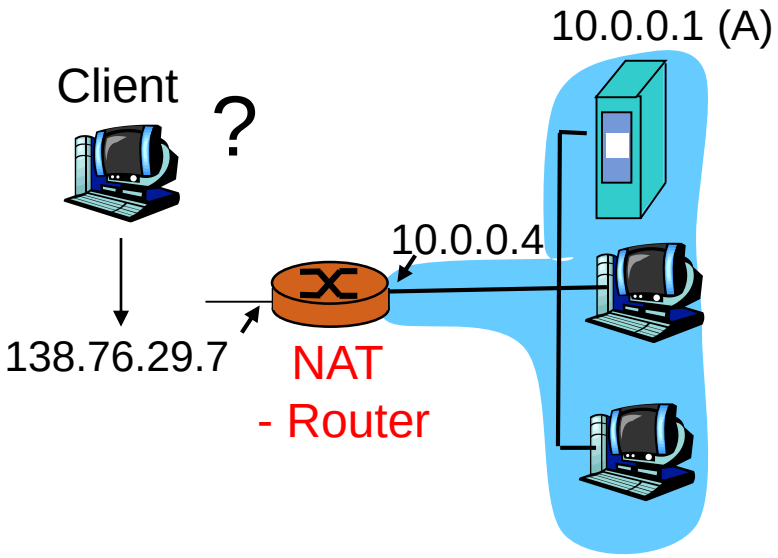
Source-NAT: Implementierung

- Prinzip: die Quell-Port# wird „missbraucht“, um nicht nur zwischen den Anwendungen, sondern auch zw. den Hosts (im lokalen Netzwerk) zu unterscheiden

- Ausgehende Datagramme:
  - Ersetze (Quell-IP-Adresse, Quell-Port#) durch (öffentliche-IP-Adresse, neue Quell-Port#)
  - Speichere in einer **NAT-Übersetzungstabelle** (**NAT translation table**) jede Übersetzung:
    - (Quell-IP-Adresse, Quell-Port #) <=> (öffentliche-IP-Adresse, neue Quell-Port#)
- Eingehende Datagramme:
  - Ersetze (öffentliche-IP-Adresse, neue Port#) in den Ziel-Feldern der Datagramme durch die (Quell-IP-Adresse, Port #) gemäß der Übersetzungstabelle

NAT-Traversal-Problem

- Angenommen, Client möchte sich zum Server A mit Adresse 10.0.0.1 verbinden
  - Aber nach außen ist nur die Adresse 138.76.29.7 sichtbar, die von vielen Hosts verwendet wird
  - Die Adresse 10.0.0.1 ist nur im LAN bekannt / verwendbar
- Eine Lösung: konfiguriere den NAT-Router so, dass alle Verbindungsanfragen bei einem bestimmten Port immer an den Host A weitergeleitet werden
  - Z.B. Eingehende Datagramme mit Ziel (138.76.29.7, port 5900) werden immer zu 10.0.0.1, Port 5910 weitergeleitet



Video: How Network Address Translation Works, https://www.youtube.com/watch?v=QBqPzHEDzvo
ab ca. 3:55 (min:sec)

ICMP: Internet Control Message Protocol

- ICMP wird von Hosts und Routern verwendet, um Netzwerkschichtinformationen miteinander auszutauschen
  - Versand von Fehlermeldungen: z.B. "Destination network unreachable"
  - Ping-Funktionalität
- ICMP-Nachrichten werden in IP-Datagrammen gesendet
- ICMP-Nachrichten haben ein **Typ-** und ein **Code-**Feld
  - Enthalten den Header und die ersten 8 Byte des IP-Datagramms, der die Nachricht verursacht hat

- ICMP-Nachrichtentypen

Typ	**Code**	**Beschreibung**
0	0	Echo-Antwort (Ping)
3	0	Zielnetz unerreichbar
3	1	Zielhost unerreichbar
3	2	Zielprotokoll unerreichbar
3	3	Zielport unerreichbar
3	6	Zielnetz unbekannt
3	7	Zielhost unbekannt
4	0	Source Quench (Überlastkontrolle)
8	0	Echo-Anforderung (Ping)
9	0	Routerbekanntmachung
10	0	Routersuche
11	0	TTL abgelaufen
12	0	IP-Header fehlerhaft

Traceroute und ICMP

- Quelle schickt eine Folge von UDP-Segmenten zum Ziel
  - 1. hat TTL = 1
  - 2. hat TTL = 2, usw.
  - Zielport möglichst unbenutzt
- Wenn das n-te Datagramm beim n-ten Router ankommt:
  - Router verwirft das Datagramm
  - … und sendet eine ICMP-Warnmeldung an die Quelle (Typ 11, Code 0)
  - Diese Warnmeldung enthält die IP-Adresse des Routers
  - Wenn diese ICMP Nachricht bei der Quelle ankommt, kann diese aus dem laufenden Timer die RTT (Round Trip Time) zum n-ten Router ablesen

Stopp-Kriterium:

- UDP-Segment erreicht irgenwann den Zielhost
- Der Zielhost antwortet mit ICMP-Nachricht “Port nicht erreichbar” (Typ 3, Code 3)
- Sobald das Quellsystem diese besondere ICMP-Nachricht erhält, weiß es, dass es keine weiteren Testpakete absenden muss

---

- Traceroute sendet immer Gruppen von drei Paketen mit derselben TTL
  - Warum?

Zusammenfassung

– Netzwerkschicht –

- Das Internetprotokoll (IP) – Grundlagen, Adressierung
- Das Internetprotokoll (IPv4) – DHCP, NAT, ICMP
- Zusatzfolien: Das Internetprotokoll IPv6

- Quellen:
  - Kurose / Ross Kapitel 4, Wikipedia

Danke.

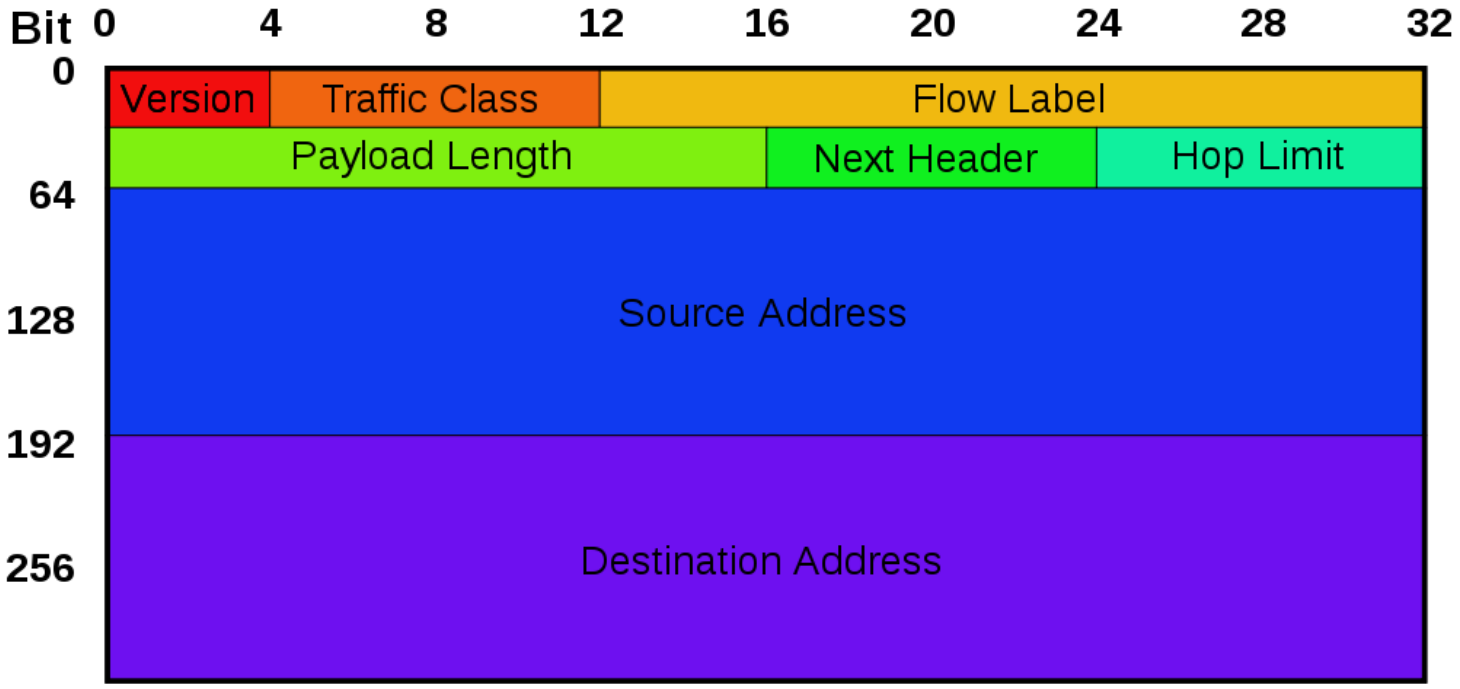
Zusätzliche Folien:
Das Internetprotokoll IPv6

IPv6

- Ursprungliche Motivation: Die 32-Bit IP-Adressenraum wird bald voll sein
- Zusätzliche Motivation:
  - Vereinfachtes Format des Headers beschleunigt die Verarbeitung / Weiterleitung von Datagrammen
  - Header-Änderungen erleichtern verschiedene Stufen der **Dienstgüte** (**Quality of Service**, **QoS**)
    - z.B. beschleunigte Behandlung von Audio- / Videoströmen
- Datagramm-Format
  - Eine fixe Länge des Headers von 40 Bytes
  - Keine Fragmentierung mehr möglich
    - Wenn ein Paket zu groß ist, wird es verworfen, und es erfolgt eine ICMP-Warnung (vom Router an die Quelle)

IPv6-Header

Traffic Class: bestimmt Priorität des Pakets (für QoS)
Flow Label: identifiziert den gleichen “Datenstrom”, z.B. eine Videoübertragung
Next Header: Typ des nächsten Kopfdatenbereiches, z. B. TCP (Typ 6) oder UDP (Typ 17)

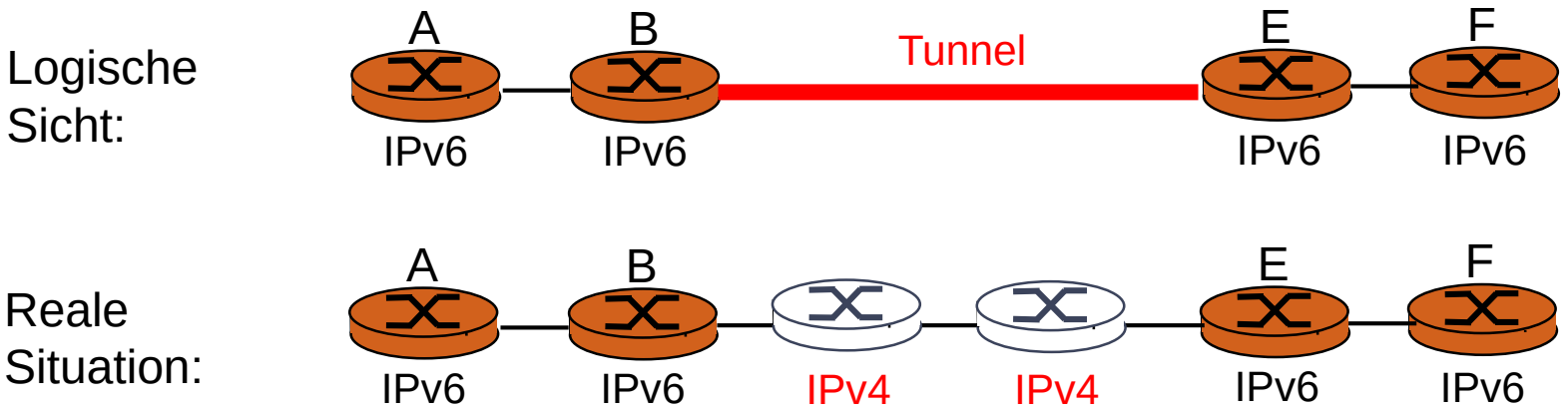


Andere Änderungen

- Prüfsumme: komplett entfernt, um die Verarbeitungszeit bei Routern zu reduzieren
  - TTL wird bei jedem Router verringert => Neuberechnung
- Optionen*:* erlaubt, aber außerhalb des Headers (d.h. im Datenbereich); angezeigt durch “Next Header”
- ICMPv6*:* neue Version von ICMP
  - Zusätzliche Nachrichten, z.B. “Packet Too Big”
- Sind so viele IP-Adressen gut?
  - Nicht für BitTorrent und anonymes Web-Surfen
  - Zeit-Artikel: “Das Internet-Protokoll 6 verändert die Spielregeln“ von Torsten Kleinz
    - **Link:** http://www.zeit.de/digital/datenschutz/2011-01/ipv6-vorratsdaten

Übergang von IPv4 zu IPv6

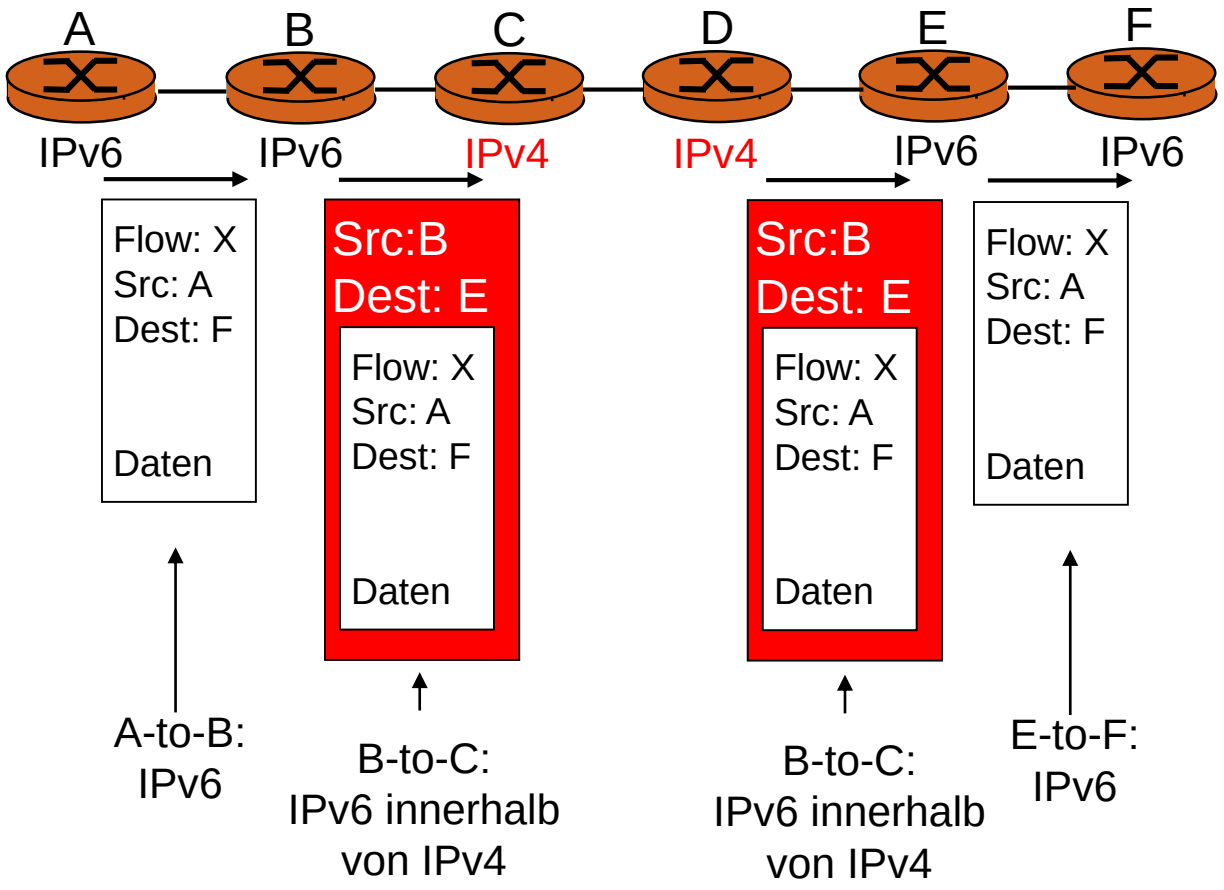
- Wie wird das Internet auf IPv6 umgestellt?
  - Umstellung an einem Stichtag ist wegen Millionen von Geräten unmöglich
    - Bei der Umstellung vor ~ 25 Jahren (NCP->TCP) ging das schon
- Lösungen: **Dual-Stack-Ansatz** und **Tunneling**
- Tunneling:
  - Der IPv6-Knoten auf der sendenden Seite des Tunnels (zum Beispiel B) plaziert das komplette IPv6-Datagramm im Nutzdatenfeld (Payload-Feld) eines IPv4-Datagramms



Tunneling

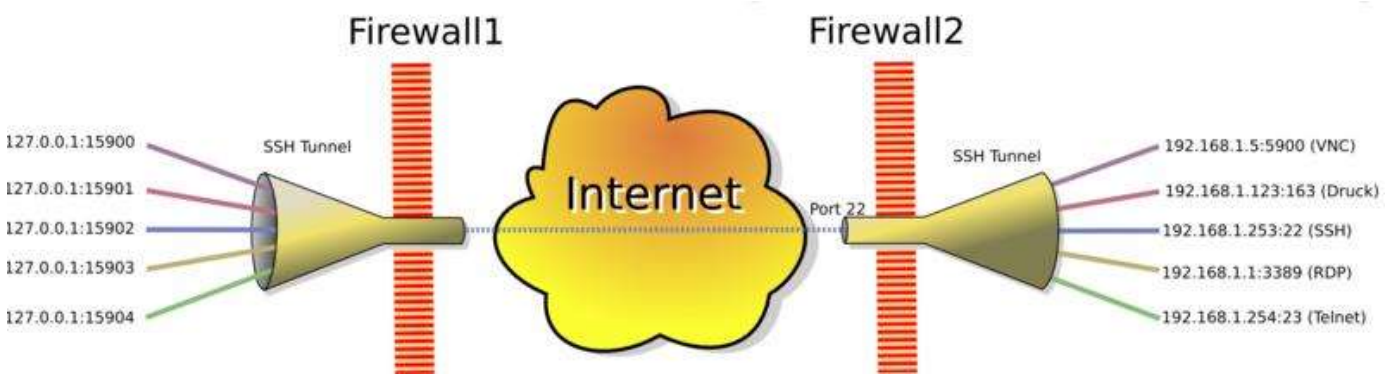
Logische Sicht:

Reale Situation:



Anderes Tunneling: **SSH-Tunneling**

- Hat nichts mit IPv6 zu tun, ist einfach nützlich!
- **Secure Shell** (ssh) ist ein Netzwerkprotokoll als auch Anwendungen, um eine sichere Netzwerkverbindung mit einem Host herzustellen
  - Ersatz für rlogin, telnet, rsh, aber auch ftp und rcp
- Ermöglicht auch SSH-Tunneling (Link)
  - Umgehen von Firewalls (z.B. für rdp); Tunneln von ungesicherten Protokollen (z.B. vnc, X11, telnet)



lokaler Host

entfernter (remote) Host