

Rechnernetze I

SoSe 2024

Roland Wismüller
Universität Siegen
roland.wismueller@uni-siegen.de
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 13. Februar 2024

Rechnernetze I

SoSe 2024

5 Internetworking



Inhalt

➔ IP

➔ Grundlagen, Adressierung und Weiterleitung, Aufbau eines IP-Pakets, Fragmentierung / Reassembly

➔ ICMP

➔ Adreßübersetzung: APR, NDP

➔ Automatische IP-Konfiguration: DHCP

➔ NAT

➔ Tunneling

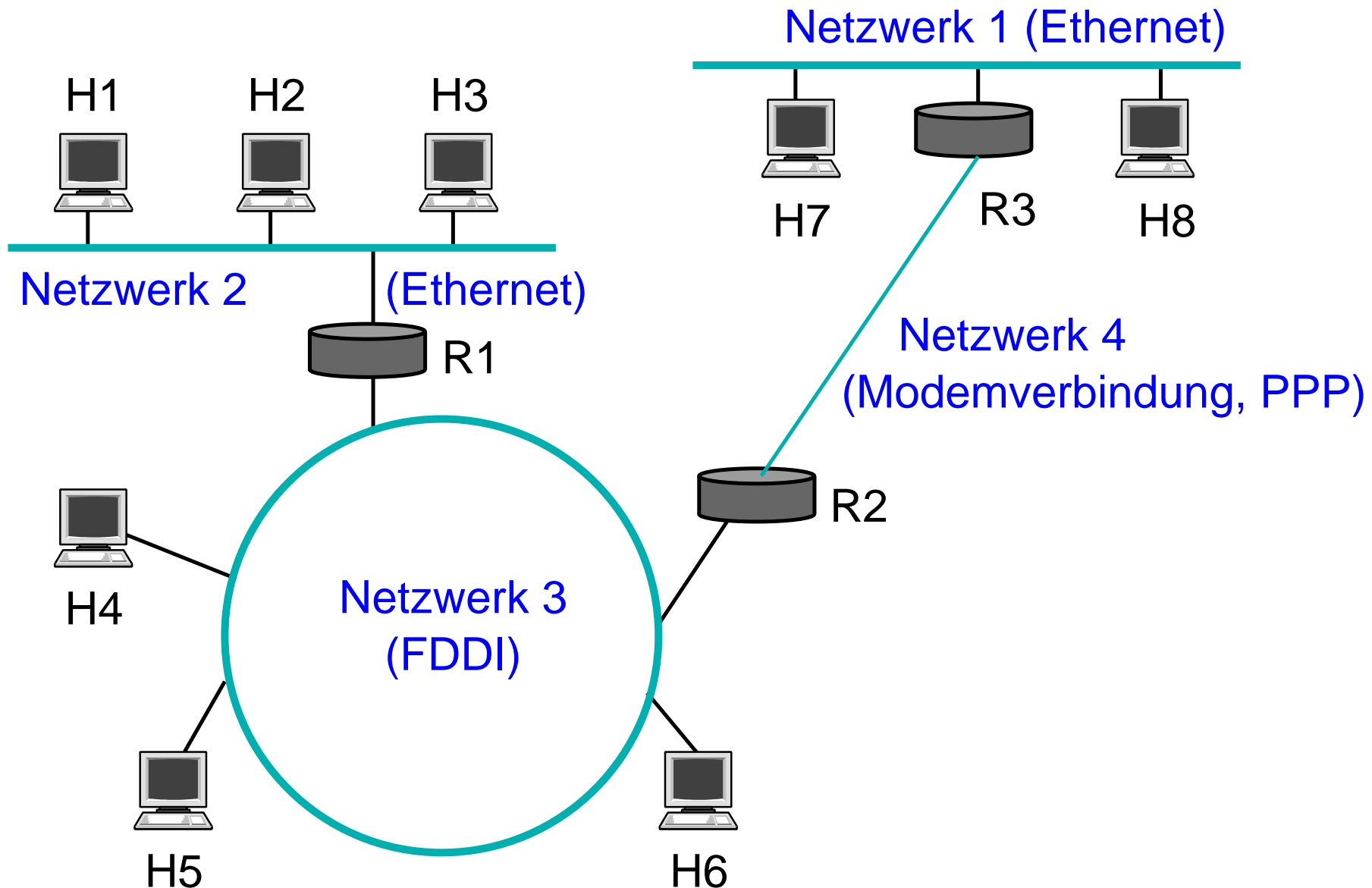
➔ Übergang von IPv4 auf IPv6

➔ Peterson, Kap. 4.1

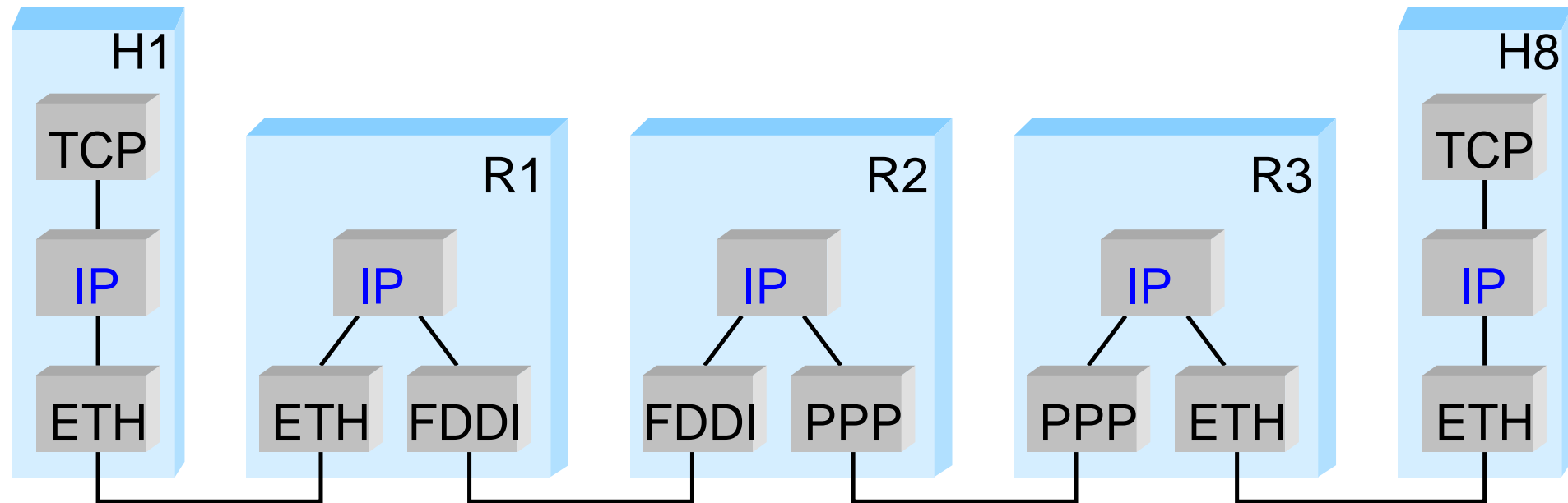
➔ CCNA, Kap. 6, 7, 8, 5.3

- ➔ Was ist ein Internetwork?
 - ➔ Zusammenschluß von einzelnen (physischen) Netzen über Router zu einem logischen Netz
 - ➔ Netz von Netzen
- ➔ Was macht ein Internetwork aus?
 - ➔ **Heterogenität**
 - ➔ Verbindung unterschiedlichster Netzwerktypen (auch zukünftiger!)
 - ➔ **Skalierung**
 - ➔ Integration von sehr vielen Rechnern und Netzen
 - ➔ Internet: ≥ 1 Milliarde Rechner
- ➔ Internetwork \neq Internet

Beispiel für ein Internetwork



IP (*Internet Protocol*) als Internetwork-Protokoll



- ➡ Auf jedem Rechner und jedem Router läuft IP
- ➡ IP kann auf unterschiedlichsten Netztechnologien aufsetzen

IP Dienstmodell (was bietet IP?)

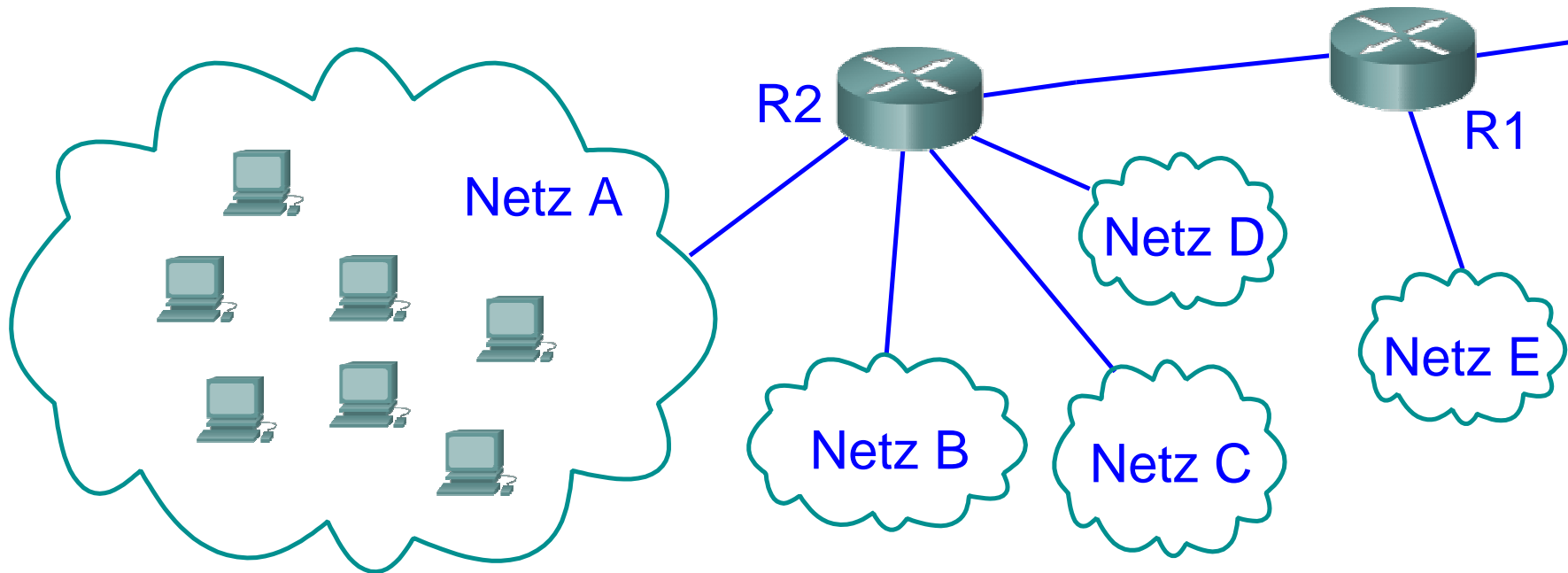
- ➔ Adressierungsschema
- ➔ Datagramm-Zustellung
- ➔ Um Heterogenität und Skalierbarkeit zu unterstützen:
kleinster gemeinsamer Nenner
 - ➔ IP bietet nur das, was mit jeder Netzwerktechnologie realisiert werden kann
 - ➔ „*run over everything*“
 - ➔ „*Best Effort*“-Modell:
 - ➔ IP „bemüht sich“, gibt aber keinerlei Garantien
 - ➔ Verlust, Duplikate, Vertauschung von Paketen möglich
 - ➔ höhere Schichten bieten bessere Dienste

IP-Versionen: IPv4 und IPv6

- ➔ IPv4 ist seit 1980 standardisiert
- ➔ Motivation für IPv6: Wachstum des Internets
 - ➔ größerer Adreßraum für IP-Adressen notwendig
- ➔ Arbeit an IPv6 seit ca. 1991
 - ➔ längere IP-Adressen \Rightarrow neuer IP-Header \Rightarrow Anpassung aller IP-Software
 - ➔ daher: auch andere Probleme von IPv4 adressiert, u.a.
 - ➔ Unterstützung von Dienstgüte-Garantien
 - ➔ Sicherheit
 - ➔ automatische Konfiguration
 - ➔ erweitertes Routing (z.B. mobile Hosts)

Ziel: Effiziente Weiterleitung von IP-Paketen

➔ Typische Situation:



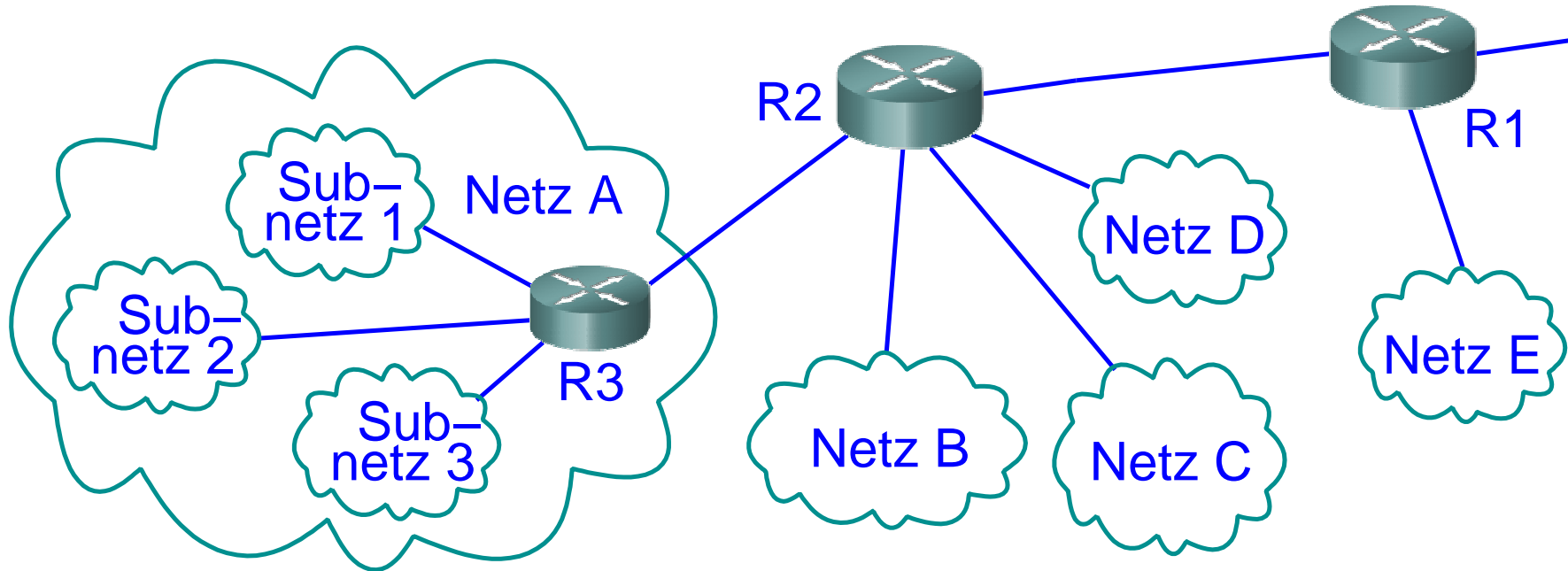
- ➔ Router müssen Pakete nur in das richtige Netz weiterleiten
- ➔ Weiterleitungstabellen sollten nur Information über Netze enthalten, nicht über einzelne Hosts
- ➔ Router muss aus IP-Adresse zugehöriges Netz bestimmen

5.2 IP: Adressierung und Weiterleitung



Ziel: Effiziente Weiterleitung von IP-Paketen

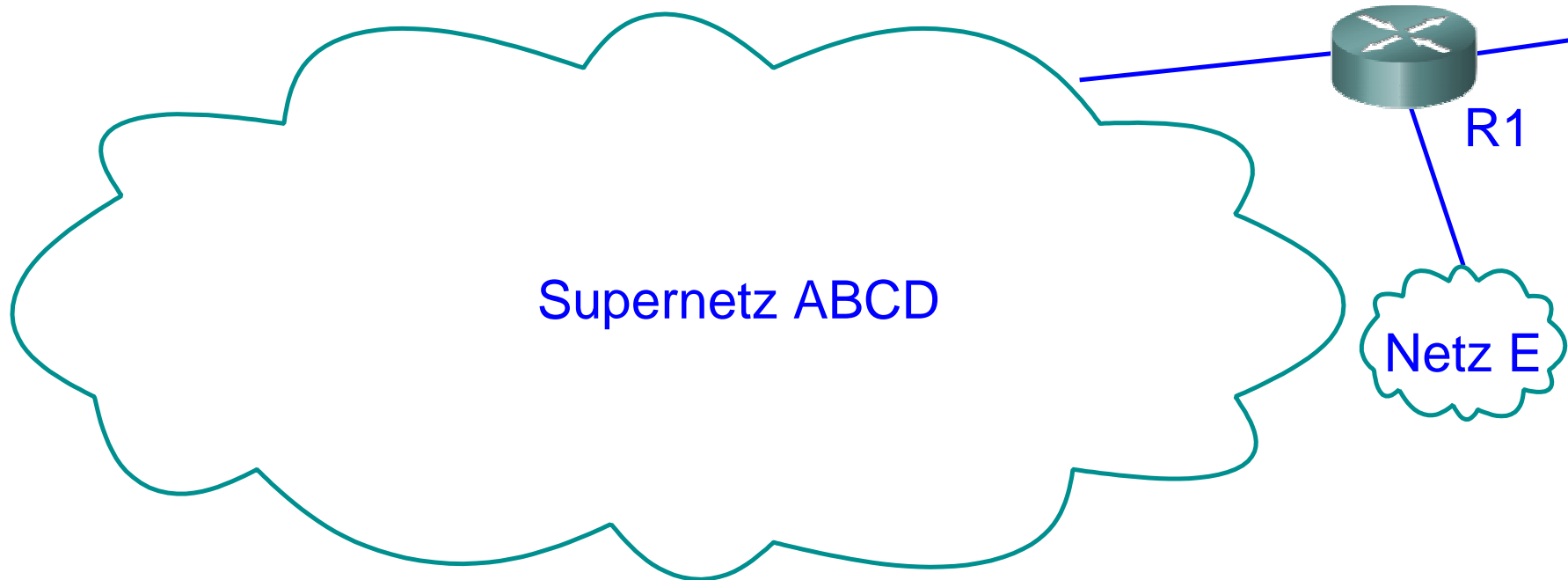
➔ Typische Situation:



- ➔ Router müssen Pakete nur in das richtige Netz weiterleiten
- ➔ Weiterleitungstabellen sollten nur Information über Netze enthalten, nicht über einzelne Hosts
- ➔ Router muss aus IP-Adresse zugehöriges Netz bestimmen

Ziel: Effiziente Weiterleitung von IP-Paketen

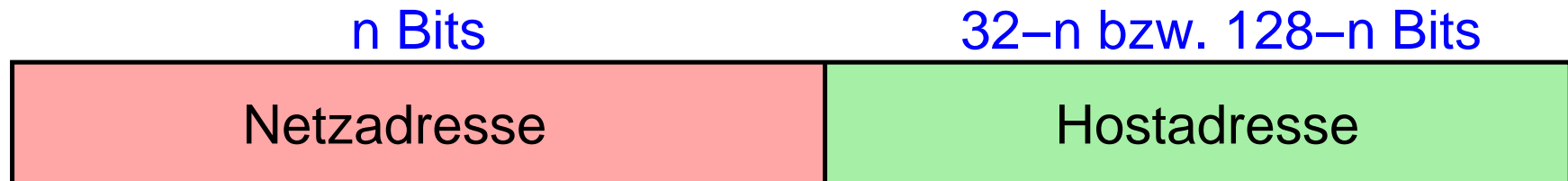
➔ Typische Situation:



- ➔ Router müssen Pakete nur in das richtige Netz weiterleiten
 - ➔ Weiterleitungstabellen sollten nur Information über Netze enthalten, nicht über einzelne Hosts
 - ➔ Router muss aus IP-Adresse zugehöriges Netz bestimmen

Aufgaben bei der Adressierung

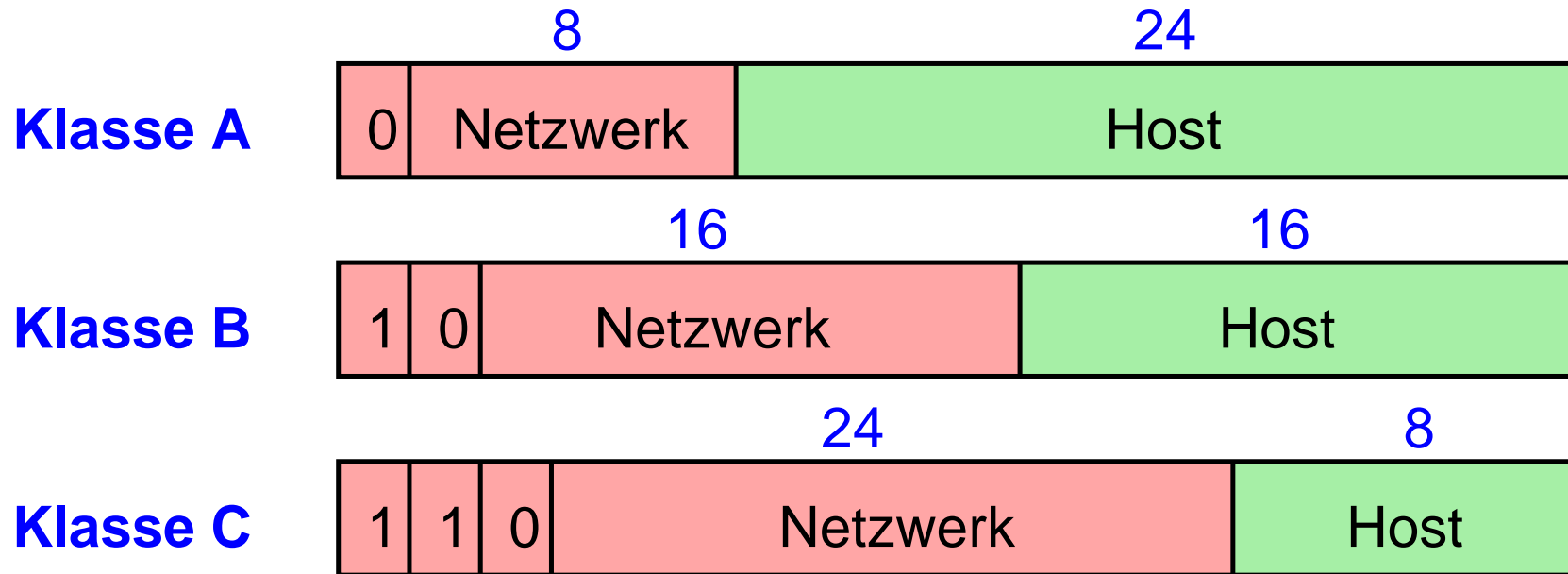
- ➔ Identifikation von Hosts
 - ➔ durch numerische Adresse (IPv4: 32 Bit, IPv6: 128 Bit)
 - ➔ hierarchischer Aufbau:



- ➔ Identifikation von Netzen
 - ➔ durch Netzadresse und Präfixlänge n
 - ➔ in IPv4 ursprünglich:
 - ➔ n geht aus der Adresse eindeutig hervor (Adressklassen)
 - ➔ heute in IPv4 und IPv6: explizite Angabe von n (klassenlose Adressierung, CIDR)

IPv4 Adressen

➔ Adressklassen:



➔ Schreibweise: byteweise dezimal, durch Punkt getrennt

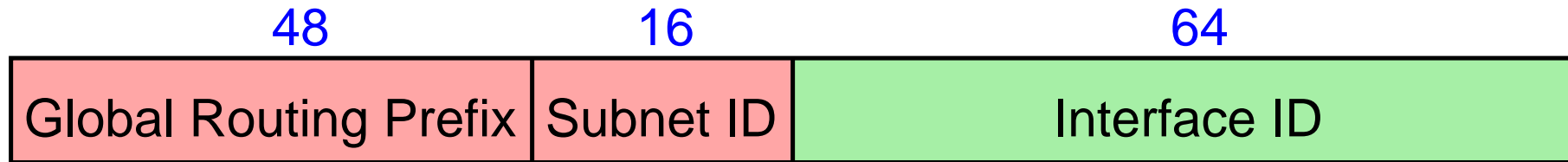
➔ z.B. 131.159.31.17

➔ Bei klassenloser Adressierung ggf. mit Angabe der Präfixlänge

➔ z.B. 131.159.31.17/16 oder 131.159.30.0/24

IPv6 Adressen

➔ Typische Struktur:



➔ Schreibweise:

➔ 16-Bit-Teile hexadezimal, getrennt durch ':'

➔ z.B. 47CD:0000:0000:0000:0000:1234:A456:0124

➔ Kurzform

➔ ohne führende Nullen, eine Nullfolge durch '::' ersetzt

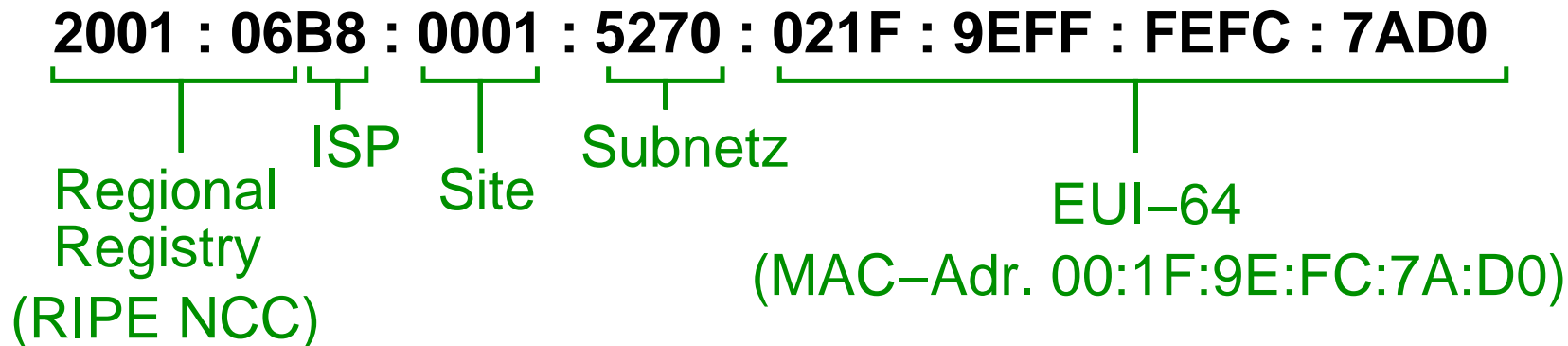
➔ z.B. 47CD::1234:A456:124

➔ Ggf. mit Angabe der Präfixlänge, z.B. 2000::/3

Adreßvergabe in IPv6

- ➔ Routing-Präfix: hierarchisch, i.W. nach Regionen
 - ➔ IANA vergibt Präfixe an *Regional Registries*, diese vergeben längere Präfixe an Provider, diese noch längere an Kunden
 - ➔ ermöglicht Aggregation von Routing-Information
- ➔ Interface-ID: manuell vergeben oder EUI-64 (~ MAC-Adresse)
- ➔ Beispielstruktur einer IPv6-Adresse:

2001 : 06B8 : 0001 : 5270 : 021F : 9EFF : FEFC : 7AD0



Regional Registry (RIPE NCC) ISP Site Subnetz EUI-64 (MAC-Adr. 00:1F:9E:FC:7A:D0)

Gültigkeitsbereiche von IP-Adressen

➔ Global

- ➔ weltweit eindeutig
- ➔ Pakete mit diesen Adressen werden im globalen Internet weitergeleitet

➔ Link Local

- ➔ nur innerhalb eines physischen LANs eindeutig
- ➔ Router leiten Pakete mit diesen Adressen nicht weiter

➔ Private (IPv4) bzw. Unique Local (IPv6)

- ➔ nur innerhalb eines privaten Netzes eindeutig
 - ➔ z.B. eine Organisation
- ➔ Pakete mit diesen Adressen werden nicht im globalen Internet weitergeleitet

Arten von IP-Adressen

- ➔ **Unicast**: Adresse für genau eine Netzwerk-Schnittstelle
 - ➔ aber: in IPv6 hat eine Schnittstelle i.d.R. mehrere Adressen
- ➔ **Multicast**: Adresse für eine Gruppe von Empfängern
- ➔ **Broadcast** (nur IPv4): alle Schnittstellen innerhalb eines Netzes
 - ➔ Adresse 255.255.255.255: Broadcast im lokalen Netz
- ➔ **Anycast**: nächstgelegene Schnittstelle aus einer Menge
 - ➔ mehrere Hosts mit identischer Adresse und gleicher Funktion
 - ➔ Router leiten Pakete zum nächstgelegenen Host weiter
 - ➔ Anwendung z.B. für DNS-Server

Spezielle Adreß-Bereiche in IP

Bedeutung	IPv4	IPv6
Global Unicast	0.0.0.0 - 223.255.255.255	2000::/3
Link Local Unicast	169.254.0.0/16	FE80::/10
Unique Local Unicast		FC00::/7
Private Adressen	10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	
Multicast	224.0.0.0/4	FF00::/8
Loopback	127.0.0.0/8	::1

Grundlagen

- ➔ Jedes IP-Datagramm enthält IP-Adresse des Ziels
 - ➔ Netzadresse kennzeichnet das physisches Netz des Ziels
- ➔ Hosts mit gleicher Netzadresse kommunizieren direkt über ihr lokales Netz
- ➔ Jedes physische Netz, das Teil des Internets ist, ist mit mindestens einem **Router** verbunden
 - ➔ Router hat mehrere Netzwerk-Schnittstellen
 - ➔ jede Schnittstelle hat ihre eigene IP-Adresse
 - ➔ **Gateway**: Schnittstelle eines Routers im lokalen Netz
- ➔ Aufgabe bei der Weiterleitung:
 - ➔ an welche Schnittstelle muß ein IP-Paket mit gegebener Zieladresse weitergeleitet werden?

Routing-Tabelle (Weiterleitungstabelle)

- ➔ Weiterleitung wird durch Routing-Tabelle gesteuert
 - ➔ in Routern und auch in normalen Hosts

- ➔ Prinzipieller Aufbau der Tabelle:

<i>Netzadresse₁</i>	<i>Präfixlänge₁</i>	<i>Next Hop₁</i>
<i>Netzadresse₂</i>	<i>Präfixlänge₂</i>	<i>Next Hop₂</i>
...

- ➔ Netzadresse und Präfixlänge zusammen identifizieren ein Netz
 - ➔ bei IPv4 statt Präfixlänge ggf. auch Subnetzmaske (☞ **S. 187**)
- ➔ *Next Hop* = Router bzw. Schnittstelle, an den/die das Paket weitergegeben werden soll, falls Ziel im angegebenen Netz liegt

Vorgehensweise bei der Weiterleitung

➔ Algorithmus:

- ➔ suche Eintrag i mit größter $Präfixlänge_i$, für den gilt:
 - ➔ Zieladresse und $Netzadresse_i$ stimmen in den ersten $Präfixlänge_i$ Bits überein
 - ➔ d.h. Zieladresse liegt in dem durch $Netzadresse_i$ und $Präfixlänge_i$ gegebenen Netz
- ➔ falls Eintrag gefunden: Weiterleiten an $NextHop_i$
- ➔ sonst: Verwerfen des Pakets

➔ Typisch: zusätzlicher Tabellen-Eintrag für Default-Route

$0.0.0.0$	0	$Next Hop_{\text{default}}$
-----------	-----	-----------------------------

- ➔ dieser Eintrag „paßt“ auf jede Zieladresse

Beispiel

	Netzadresse	Subnetzmaske	Next Hop
0	0.0.0.0	0.0.0.0	10.0.0.1
1	141.99.0.0	255.255.0.0	10.1.0.1
2	141.99.179.0	255.255.255.0	Interface 1
3	141.99.128.0	255.255.192.0	10.3.4.1

➡ IP-Paket mit Ziel 141.99.178.6

➡ 0: 141.99.178.6 AND 0.0.0.0 = **0.0.0.0** ($n = 0$)

➡ 1: 141.99.178.6 AND 255.255.0.0 = **141.99.0.0** ($n = 16$)

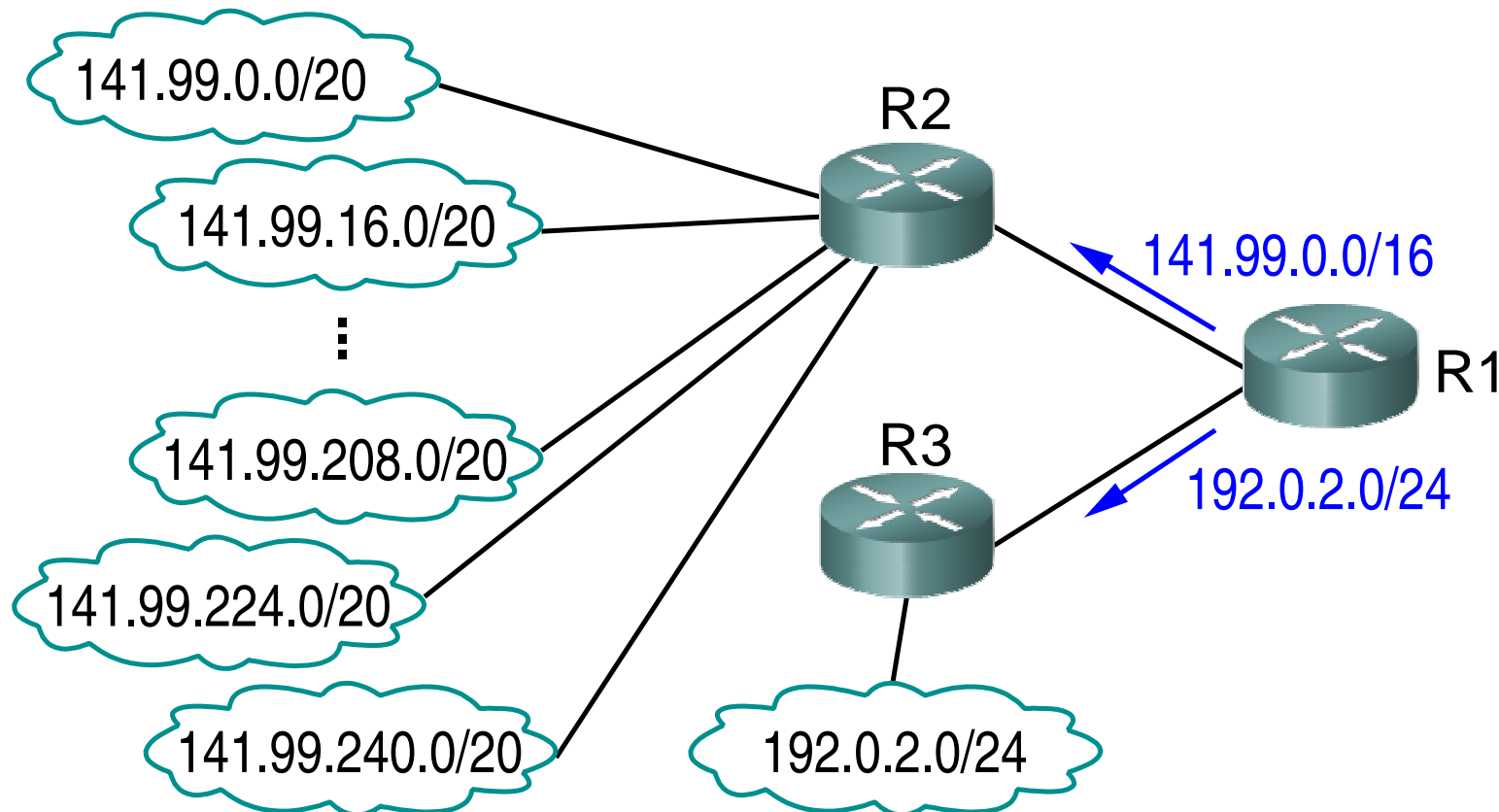
➡ 2: 141.99.178.6 AND 255.255.255.0 = **141.99.178.0**

➡ 3: 141.99.178.6 AND 255.255.192.0 = **141.99.128.0** ($n = 18$)

➡ Weiterleitung an Router 10.3.4.1!

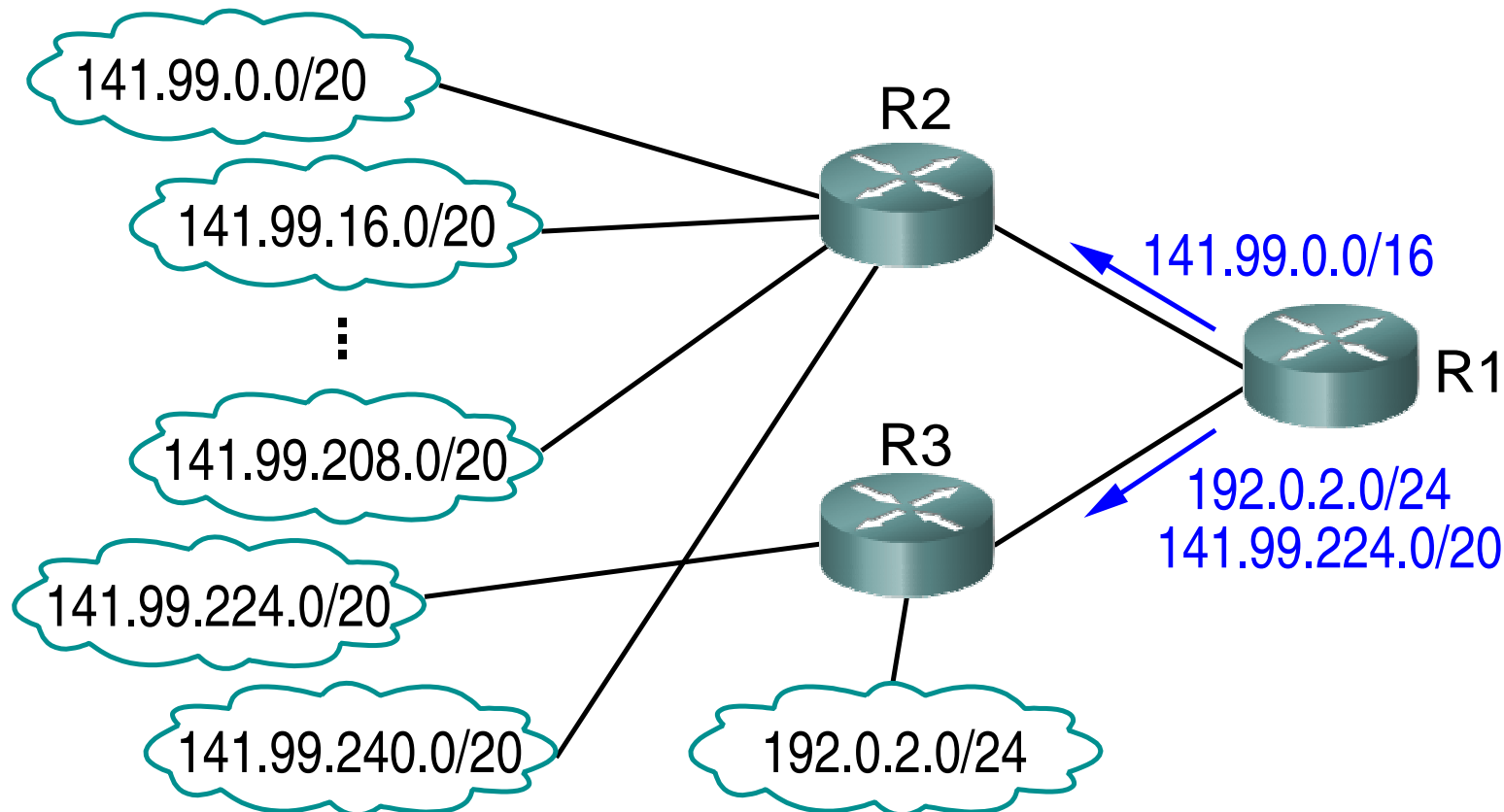
Motivation für die Suche nach dem längsten Präfix

- ➔ Erlaubt überlappende Einträge in der Routing-Tabelle
- ➔ Damit auch Zusammenfassung „nicht zusammenhängender“ Adreßbereiche möglich:



Motivation für die Suche nach dem längsten Präfix

- ➔ Erlaubt überlappende Einträge in der Routing-Tabelle
- ➔ Damit auch Zusammenfassung „nicht zusammenhängender“ Adreßbereiche möglich:



Bildung von Subnetzen

- ➔ Motivation: Unterteilung eines großen Netzes (z.B. Firmennetz) in mehrere kleinere Netze (z.B. Abteilungsnetze)
 - ➔ nach „ausen“ hin ist nur das Gesamtnetz sichtbar

- ➔ IPv4: ein Teil der Host-Bits wird für die Subnetz-ID „geborgt“

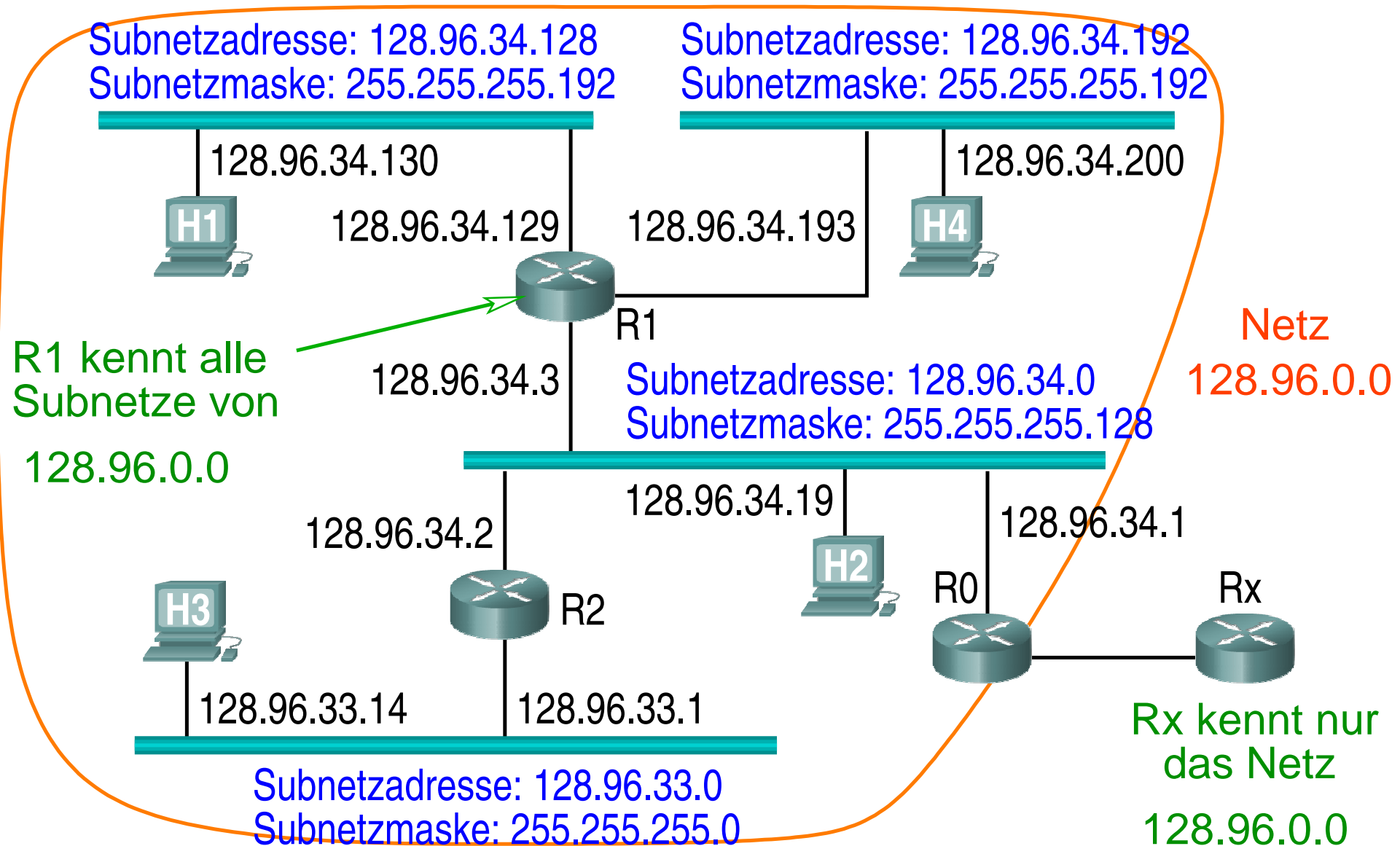
Netzwerk-Adresse	Host-Adresse	Klasse-B-Adresse (/16)
11111111 11111111 11111111	00000000	Subnetz-Maske (255.255.255.0 bzw. /24)
Netzwerk-Adresse	Subnetz-ID	Host-Adr.
Adresse mit Subnetz		

- ➔ i.a. werden Subnetze unterschiedlicher Größe erzeugt
- ➔ Subnetzmaske legt Präfixlänge für das Subnetz fest
- ➔ IPv6: Subnetz-ID in eigenem 16-Bit-Feld (👉 **S. 177**)

5.2.3 Subnetting ...



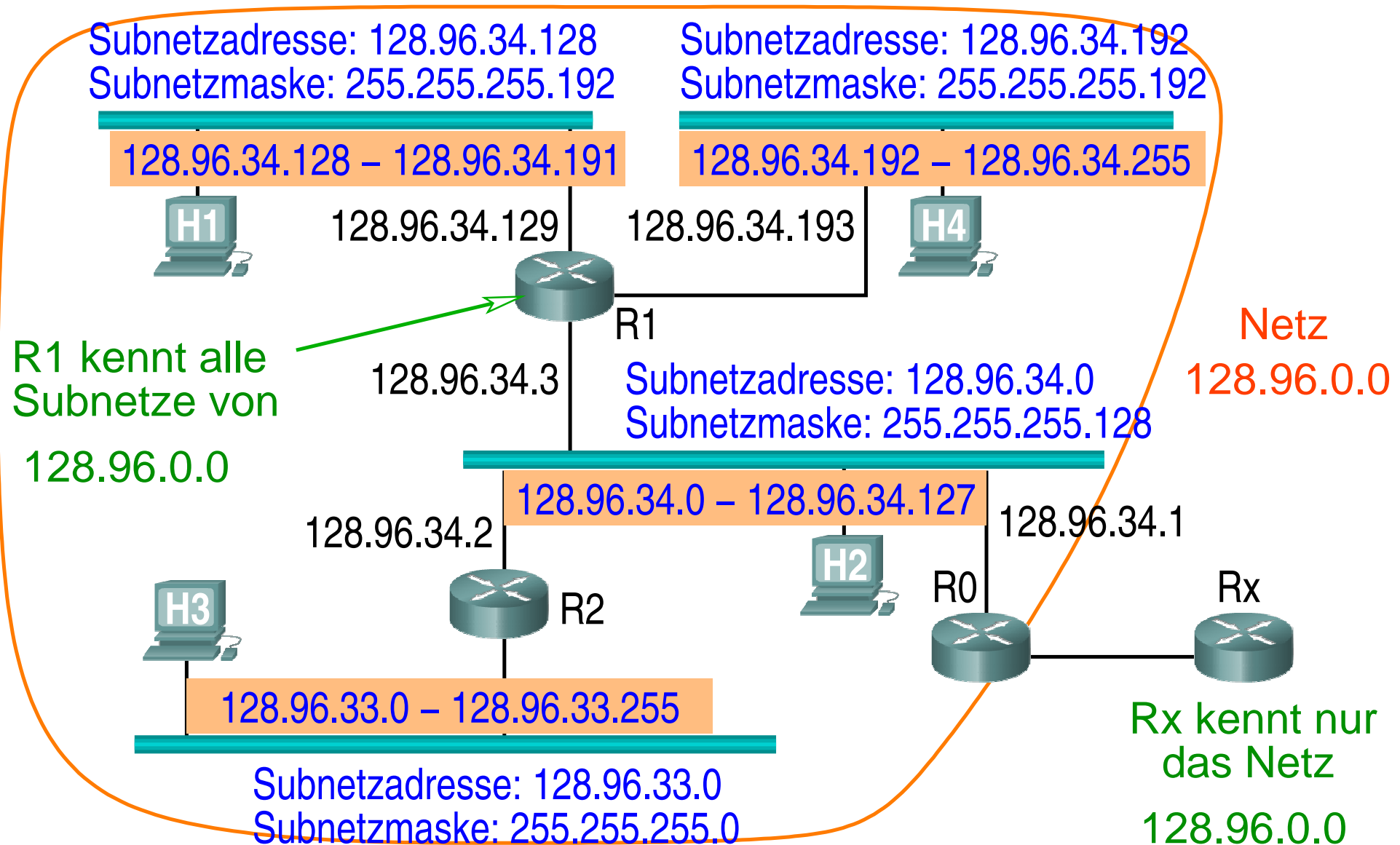
Ein Netz mit Subnetzen (IPv4):



5.2.3 Subnetting ...



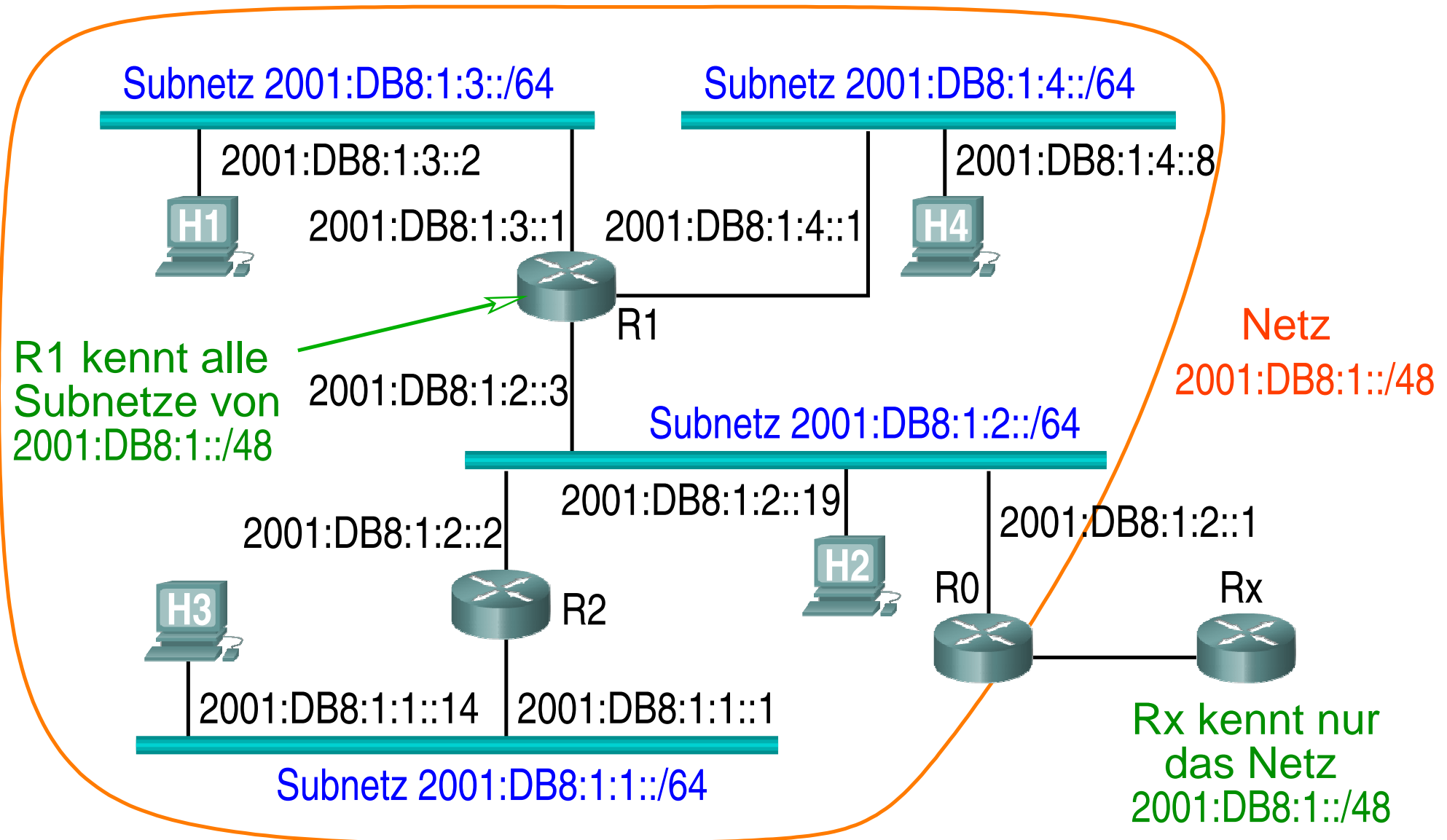
Ein Netz mit Subnetzen (IPv4):



5.2.3 Subnetting ...



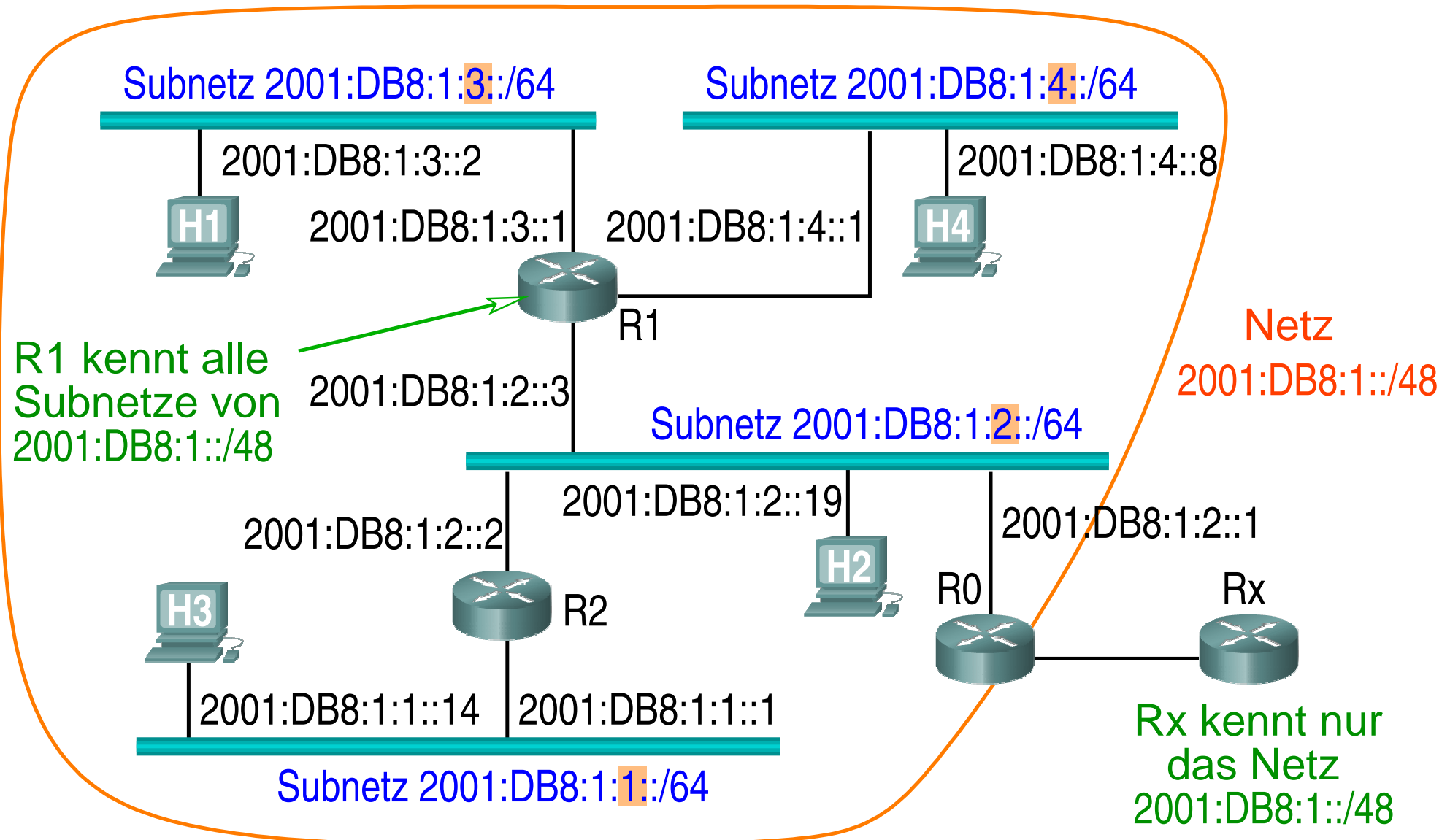
Ein Netz mit Subnetzen (IPv6):



5.2.3 Subnetting ...



Ein Netz mit Subnetzen (IPv6):



Bestimmung der (Sub-)Netzzugehörigkeit

- ➔ Wie bestimmt ein Host bzw. Router, ob eine IP-Adresse xyz in einem gegebenen (Sub-)Netz liegt?
- ➔ Gegeben: (Sub-)Netzadresse und Präfixlänge n
 - ➔ stimmen xyz und Netzadresse in den ersten n Bits überein?
 - ➔ (vgl. Weiterleitungsalgorithmus auf S. 184)
- ➔ Gegeben: (Sub-)Netzadresse und (Sub-)Netzmaske
 - ➔ gilt $xyz \text{ AND Netzmaske} = \text{Netzadresse}$?
- ➔ Beispiel: $128.96.34.19 \text{ AND } 255.255.255.128 = 128.96.34.0$

Erstellung von Subnetzen in IPv4

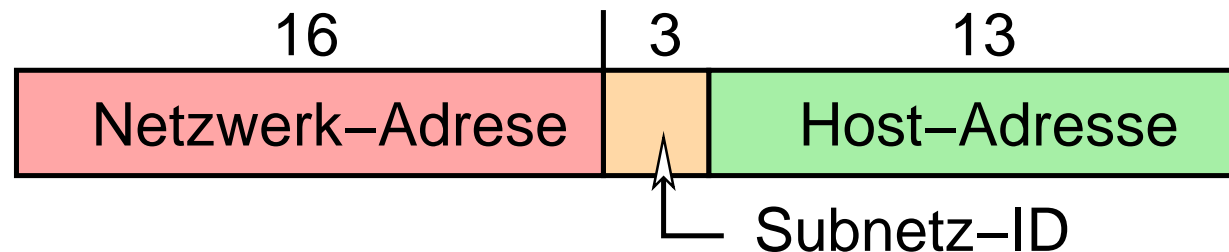
- ➔ Einschränkungen durch Realisierung:
 - ➔ Zahl der Hostadressen in einem Subnetz ist immer eine Zweierpotenz
 - ➔ bei Präfixlänge l des Subnetzes also 2^{32-l}
 - ➔ zwei Hostadressen sind reserviert:
 - ➔ $0...0_2$: Adresse des Netzwerks selbst
 - ➔ $1...1_2$: Broadcastadresse
 - ➔ damit $2^{32-l} - 2$ Hosts möglich
 - ➔ Subnetz mit 2^k Hostadressen kann nur an einer durch 2^k teilbaren Adresse beginnen
 - ➔ z.B. Adressbereich 141.99.179.64 - 141.99.179.191 ist nicht möglich
- ➔ Im folgenden: ursprüngliche Präfixlänge sei n

Mögliche Vorgehensweisen

- ➔ Einfaches Subnetting
 - ➔ feste Anzahl von k Bits des (bisherigen) Hostanteils ($32 - n$ Bits) wird für die Subnetz-ID „geborgt“
 - ➔ identische Subnetzmaske für alle Subnetze
 - ➔ ergibt 2^k Subnetze mit identischer Größe $2^{32-n-k} - 2$
- ➔ Hierarchisches Subnetting
 - ➔ bei Bedarf werden einzelne Subnetze weiter unterteilt
 - ➔ dabei entstehen längere Subnetzmasken
- ➔ VLSM (*Variable Length Subnet Mask*)
 - ➔ Subnetze werden von Anfang an entsprechend ihrer Größe durch Subnetzmasken unterschiedlicher Länge realisiert

Beispiel: Einfaches Subnetting

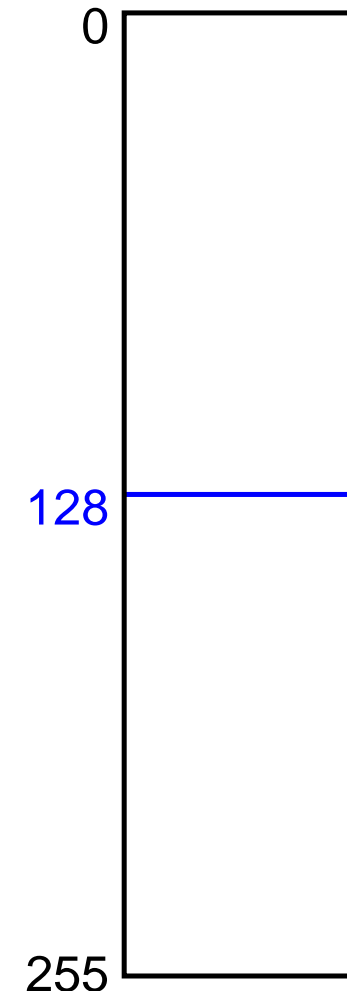
- ➔ Im Netzwerk 141.99.0.0/16 sollen 6 Subnetze realisiert werden
- ➔ Für 6 Subnetze müssen von den 16 Host-Bits 3 Bits „geborgt“ werden ($6 \leq 2^3$):



- ➔ Es entstehen 8 Subnetze für jeweils $2^{13} - 2 = 8190$ Hosts
 - ➔ 141.99.0.0 - 141.99.31.255; 141.99.32.0 - 141.99.63.255;
...; ...; 141.99.224.0 - 141.99.255.255
 - ➔ Subnetzmaske 255.255.224.0, Präfixlänge: 19
- ➔ Nachteil:
 - ➔ Subnetze unterschiedlicher Größe sind nicht möglich

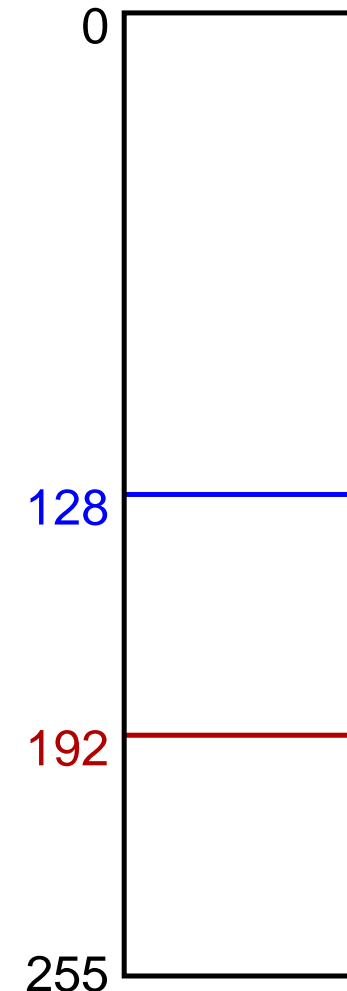
Beispiel: Hierarchisches Subnetting

- ➔ Im Netzwerk 1.2.3.0/24 sollen 4 Subnetze realisiert werden, die 100, 50, 25 und 5 Hosts aufnehmen können
- ➔ Durch Borgen eines Bits entstehen 2 Subnetze:
 - ➔ 1.2.3.0 - 1.2.3.127; 1.2.3.128 - 1.2.3.255
 - ➔ Netzmaske 255.255.255.128, Präfixlänge: 25



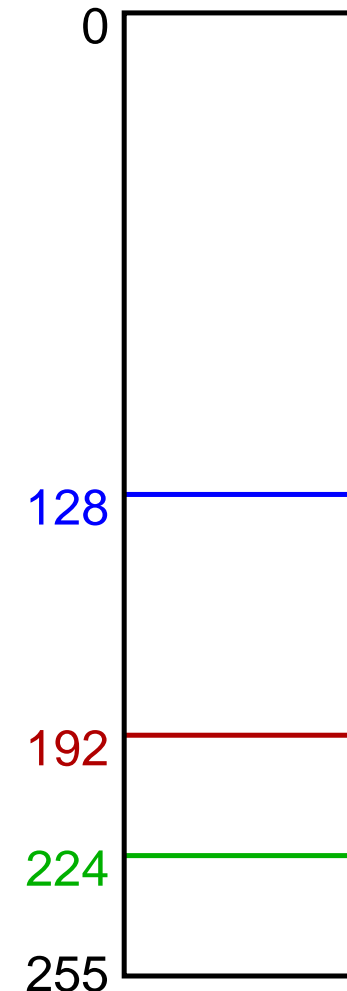
Beispiel: Hierarchisches Subnetting

- ➔ Im Netzwerk 1.2.3.0/24 sollen 4 Subnetze realisiert werden, die 100, 50, 25 und 5 Hosts aufnehmen können
- ➔ Durch Borgen eines Bits entstehen 2 Subnetze:
 - ➔ 1.2.3.0 - 1.2.3.127; 1.2.3.128 - 1.2.3.255
 - ➔ Netzmaske 255.255.255.128, Präfixlänge: 25
- ➔ Weitere Unterteilung des zweiten Subnetzes:
 - ➔ 1.2.3.128 - 1.2.3.191; 1.2.3.192 - 1.2.3.255
 - ➔ Netzmaske 255.255.255.192, Präfixlänge: 26



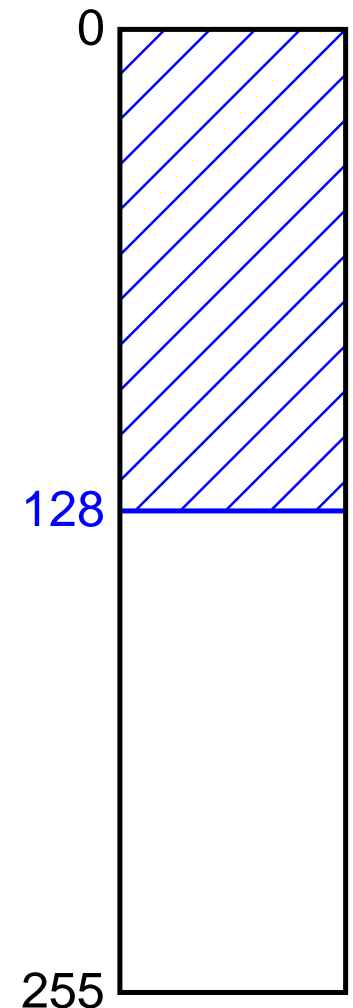
Beispiel: Hierarchisches Subnetting

- ➔ Im Netzwerk 1.2.3.0/24 sollen 4 Subnetze realisiert werden, die 100, 50, 25 und 5 Hosts aufnehmen können
- ➔ Durch Borgen eines Bits entstehen 2 Subnetze:
 - ➔ 1.2.3.0 - 1.2.3.127; 1.2.3.128 - 1.2.3.255
 - ➔ Netzmaske 255.255.255.128, Präfixlänge: 25
- ➔ Weitere Unterteilung des zweiten Subnetzes:
 - ➔ 1.2.3.128 - 1.2.3.191; 1.2.3.192 - 1.2.3.255
 - ➔ Netzmaske 255.255.255.192, Präfixlänge: 26
- ➔ Nochmal Unterteilung des zweiten Subnetzes:
 - ➔ 1.2.3.192 - 1.2.3.223; 1.2.3.224 - 1.2.3.255
 - ➔ Netzmaske 255.255.255.224, Präfixlänge: 27



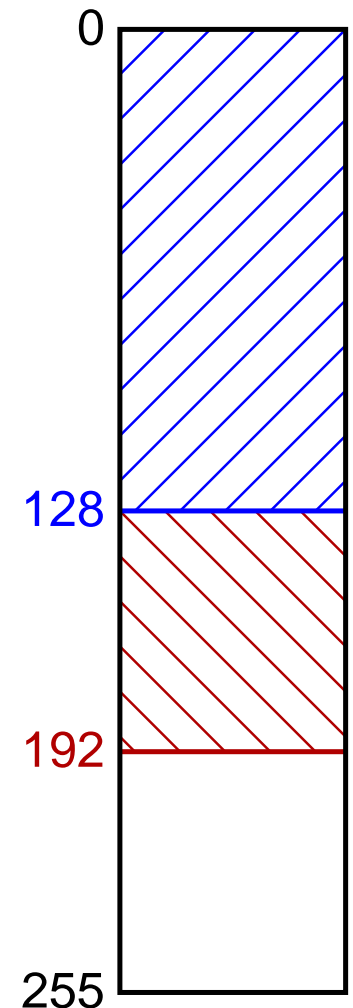
Beispiel: VLSM

- ➔ Im Netzwerk 1.2.3.0/24 sollen 4 Subnetze realisiert werden, die 100, 50, 25 und 5 Hosts aufnehmen können
- ➔ Teile möglichst kleine Subnetze in **absteigender** Reihenfolge der Größe zu:
 - ➔ 100 Hosts: Größe $128 = 2^7$, P.länge $32 - 7 = 25$
 - ➔ 1.2.3.0 - 1.2.3.127
 - ➔ Netzadr. 1.2.3.0, N.maske 255.255.255.128



Beispiel: VLSM

- ➔ Im Netzwerk 1.2.3.0/24 sollen 4 Subnetze realisiert werden, die 100, 50, 25 und 5 Hosts aufnehmen können
- ➔ Teile möglichst kleine Subnetze in **absteigender** Reihenfolge der Größe zu:
 - ➔ 100 Hosts: Größe $128 = 2^7$, P.länge $32 - 7 = 25$
 - ➔ 1.2.3.0 - 1.2.3.127
 - ➔ Netzadr. 1.2.3.0, N.maske 255.255.255.128
 - ➔ 50 Hosts: Größe $64 = 2^6$, P.länge $32 - 6 = 26$
 - ➔ 1.2.3.128 - 1.2.3.191
 - ➔ Netzadr. 1.2.3.128, N.maske 255.255.255.192



Beispiel: VLSM

- ➔ Im Netzwerk 1.2.3.0/24 sollen 4 Subnetze realisiert werden, die 100, 50, 25 und 5 Hosts aufnehmen können
- ➔ Teile möglichst kleine Subnetze in **absteigender** Reihenfolge der Größe zu:

- ➔ 100 Hosts: Größe $128 = 2^7$, P.länge $32 - 7 = 25$

- ➔ Netzadr. 1.2.3.0, N.maske 255.255.255.128

- ➔ 50 Hosts: Größe $64 = 2^6$, P.länge $32 - 6 = 26$

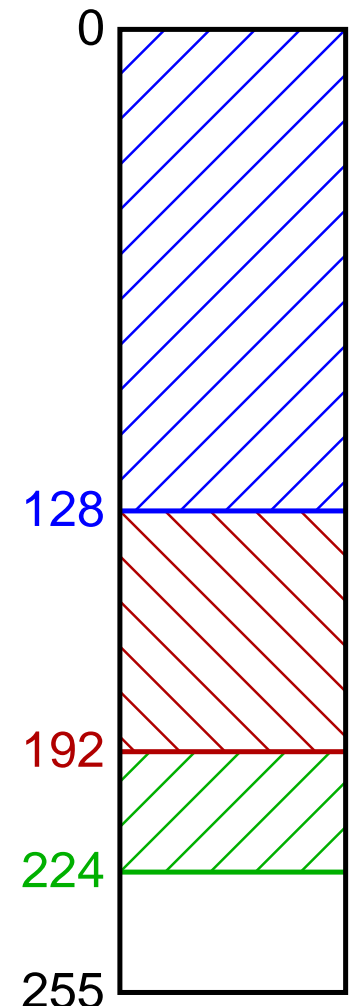
- ➔ 1.2.3.128 - 1.2.3.191

- ➔ Netzadr. 1.2.3.128, N.maske 255.255.255.192

- ➔ 25 Hosts: Größe $32 = 2^5$, P.länge $32 - 5 = 27$

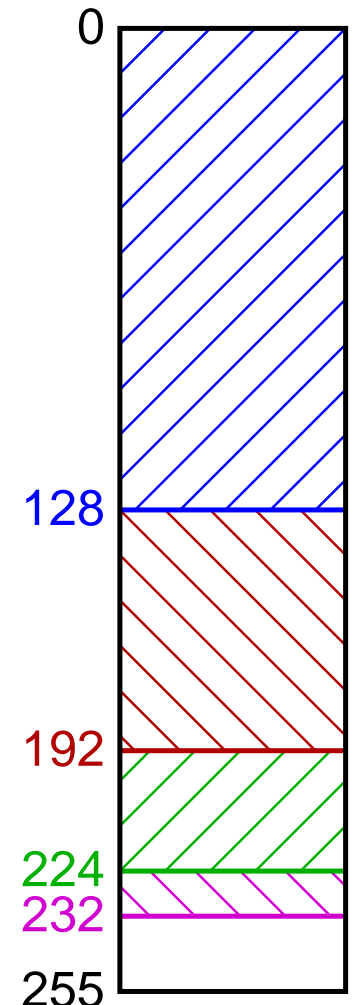
- ➔ 1.2.3.192 - 1.2.3.223

- ➔ Netzadr. 1.2.3.192, N.maske 255.255.255.224



Beispiel: VLSM

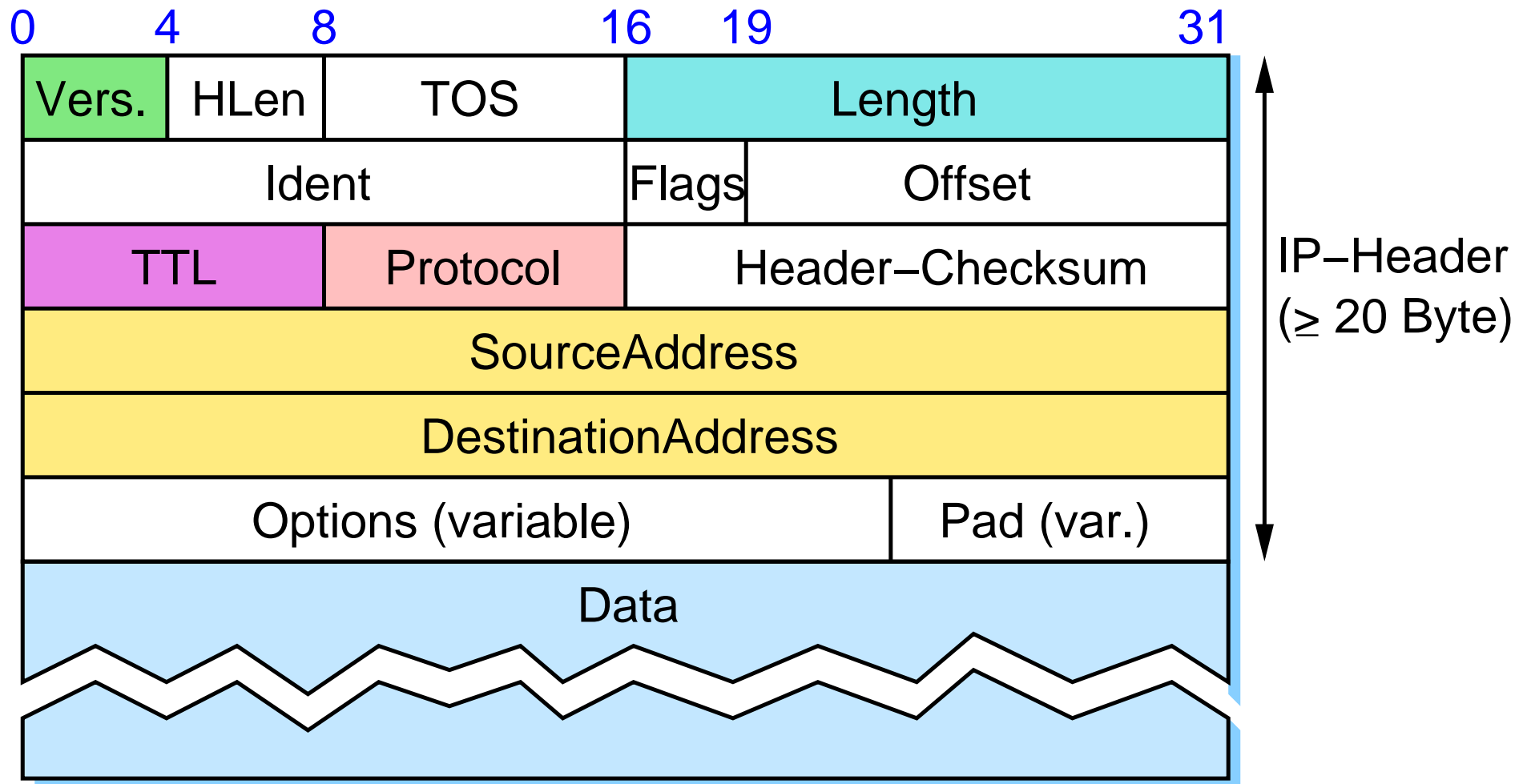
- ➔ Im Netzwerk 1.2.3.0/24 sollen 4 Subnetze realisiert werden, die 100, 50, 25 und 5 Hosts aufnehmen können
- ➔ Teile möglichst kleine Subnetze in **absteigender** Reihenfolge der Größe zu:
 - ➔ 100 Hosts: Größe $128 = 2^7$, P.länge $32 - 7 = 25$
 - ➔ Netzadr. 1.2.3.0, N.maske 255.255.255.128
 - ➔ 50 Hosts: Größe $64 = 2^6$, P.länge $32 - 6 = 26$
 - ➔ Netzadr. 1.2.3.128, N.maske 255.255.255.192
 - ➔ 25 Hosts: Größe $32 = 2^5$, P.länge $32 - 5 = 27$
 - ➔ Netzadr. 1.2.3.192, N.maske 255.255.255.224
 - ➔ 5 Hosts: Größe $8 = 2^3$, P.länge $32 - 3 = 29$
 - ➔ Netzadr. 1.2.3.224, N.maske 255.255.255.248



5.3 Aufbau eines IP-Pakets



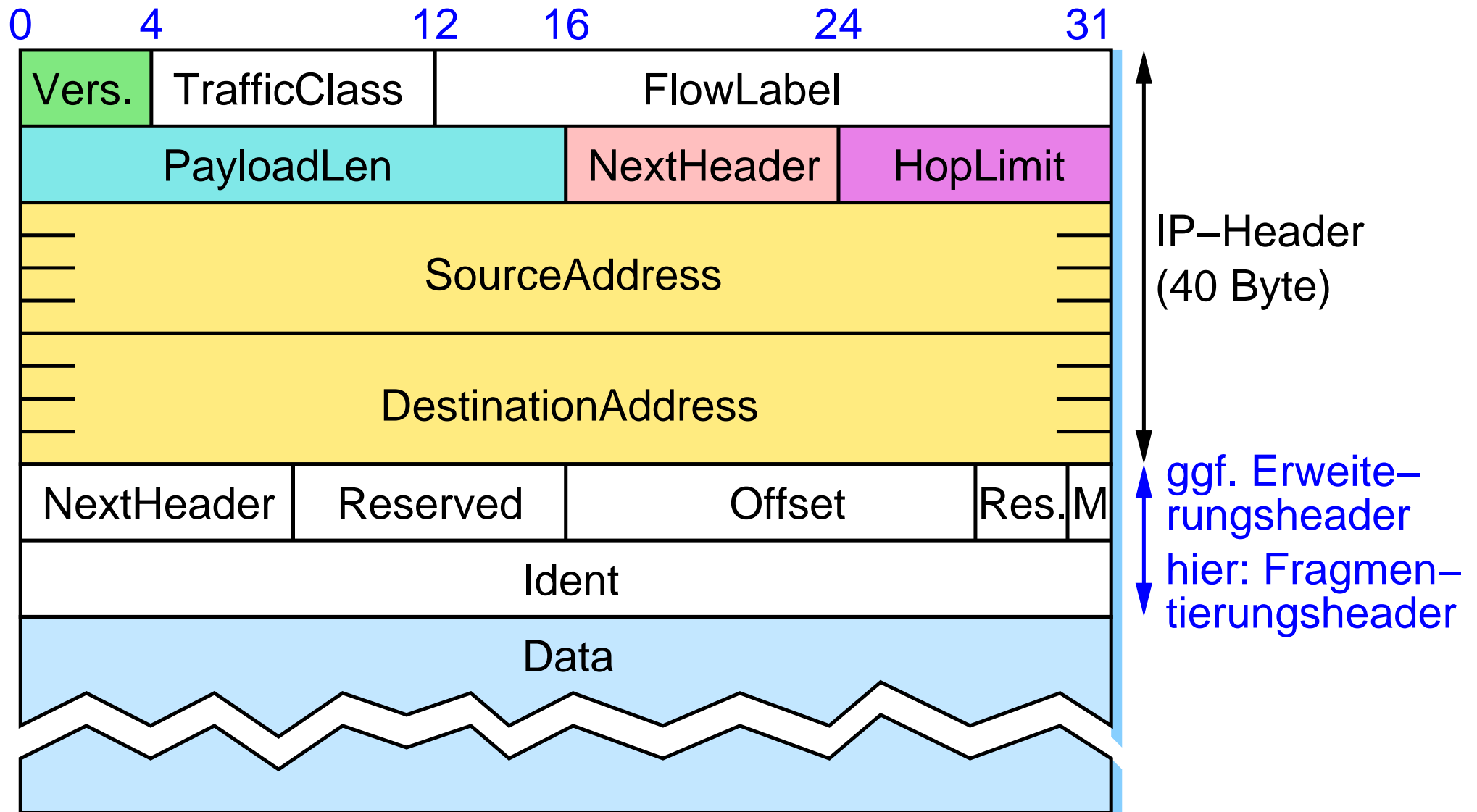
IPv4



5.3 Aufbau eines IP-Pakets ...



IPv6



Bedeutung der wichtigsten Felder

- ➔ **HLen**: Länge des Headers in 32-Bit Worten
- ➔ **TOS / TrafficClass / FlowLabel**: für *Quality of Service*
- ➔ **Length**: Gesamtlänge des Pakets inkl. IP-Header in Bytes
PayloadLen: Länge des Pakets ohne Basis-IP-Header
 - ➔ maximal 65535 Bytes
- ➔ **Ident / Flags (M) / Offset**: für Fragmentierung / Reassembly
- ➔ **TTL / HopLimit**: zur Erkennung endlos kreisender Pakete
 - ➔ wird von jedem Router heruntergezählt, bei 0 wird das Paket verworfen
- ➔ **Protocol / NextHeader**: kennzeichnet das im Datenteil versendete Protokoll (z.B. TCP, UDP; für Demultiplexing) oder (bei IPv6) ggf. den Typ des folgenden Erweiterungsheaders

Problem für IP

- ➔ Jedes lokale Netzwerk definiert eine (unterschiedliche) maximale Framegröße (**MTU: *Maximum Transmission Unit***)

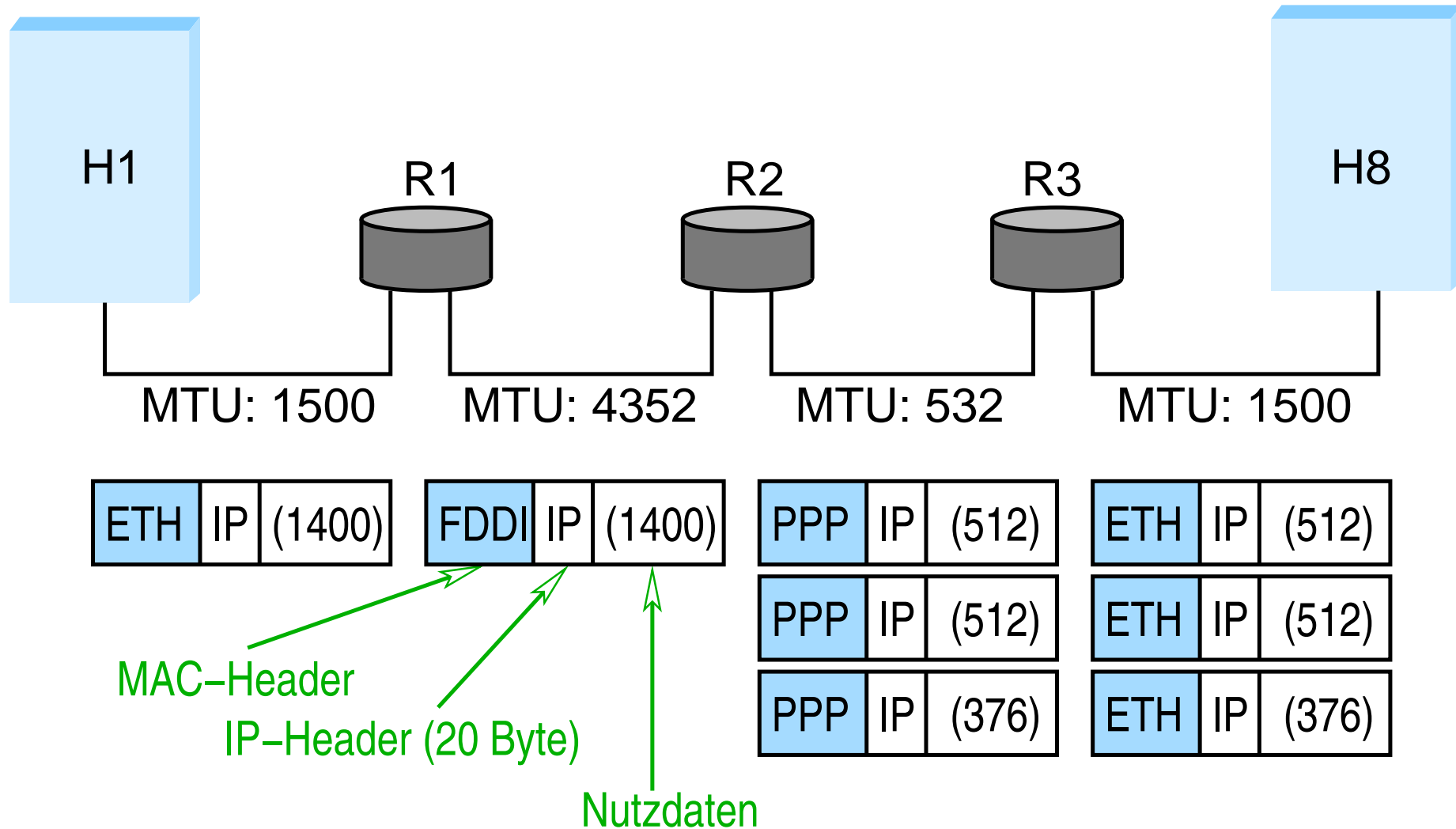
Netzwerk	MTU [Byte]
Ethernet	1500
FDDI	4352
4 Mbits/sec Token-Ring (IEEE 802.5)	4464
16 Mbits/sec Token-Ring (IBM)	17914

- ➔ Alternativen für IP:
 - ➔ max. Paketgröße = minimale MTU (welcher Netze??)
 - ➔ Möglichkeit der Fragmentierung von IP-Paketen

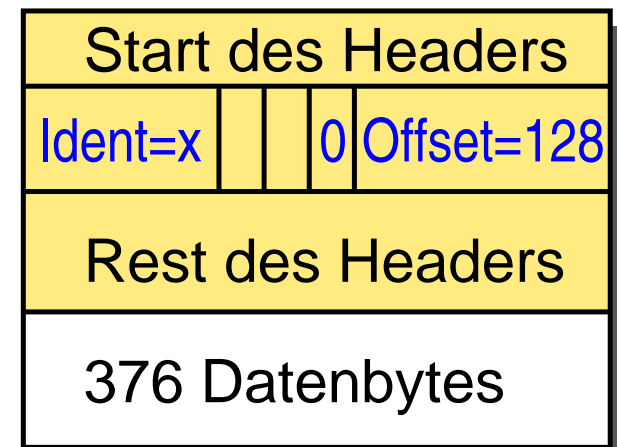
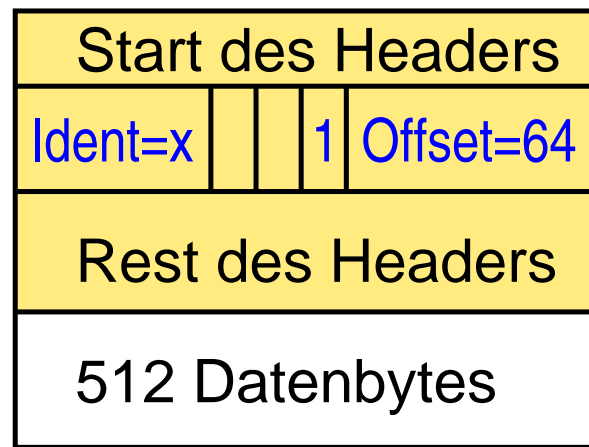
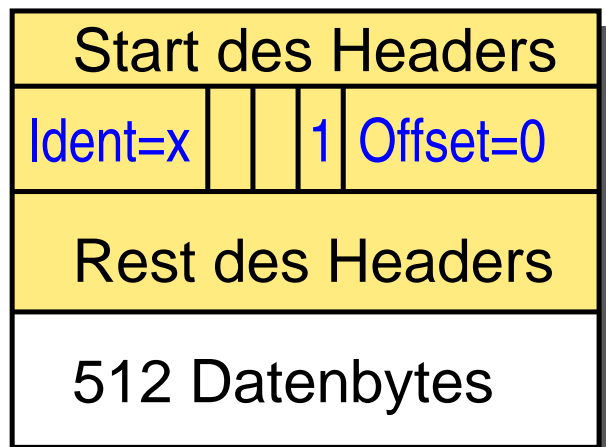
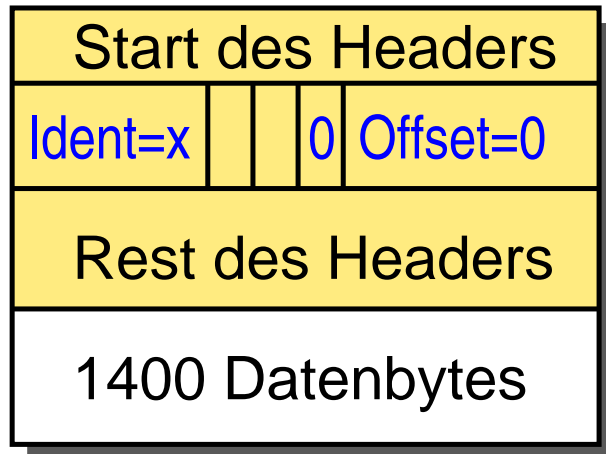
5.4 IP: Fragmentierung/Reassembly ...



Beispiel zur Fragmentierung (IPv4)



Fragmentierung im Detail (IPv4)



- ➔ **Ident**-Feld kennzeichnet zusammengehörige Fragmente
- ➔ **M**-Bit (bei IPv4 im **Flags**-Feld): „*more fragments*“
- ➔ **Offset** zählt in 8-Byte-Schritten!
- ➔ Analog auch in IPv6

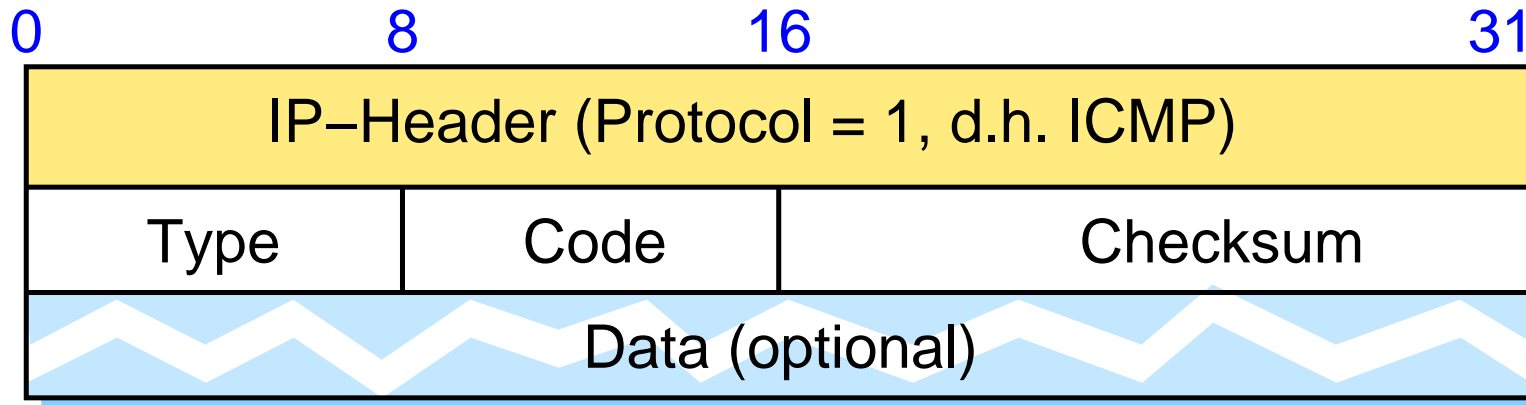
Fragmentierung im Detail ...

- ➔ Fragmentierung geschieht bei Bedarf im Sender bzw. bei IPv4 auch in den Routern
- ➔ Jedes Fragment ist ein eigenständiges IP-Datagramm
 - ➔ IPv4: ein Fragment kann ggf. nochmals fragmentiert werden
- ➔ Empfänger baut alle Fragmente wieder zusammen
 - ➔ falls ein Fragment nicht ankommt, werden alle anderen zugehörigen Fragmente verworfen
- ➔ Bei IPv6 und meist auch bei IPv4: „*Path MTU Discovery*“
 - ➔ Fragmentierung im Router verboten (IPv4: **DF**-Flag im Header)
⇒ ggf. Fehlermeldung an Sender (über ICMP, siehe später)
 - ➔ Sender kann minimale MTU ermitteln

ICMP: *Internet Control Message Protocol*

- ➔ Datagramme für Fehler- und Verwaltungsmeldungen:
 - ➔ Ziel nicht erreichbar
 - ➔ Reassembly fehlgeschlagen
 - ➔ Fragmentierung nicht erlaubt, aber erforderlich
 - ➔ TTL wurde 0
 - ➔ *Redirect*: besserer Router für das Ziel
 - ➔ *Echo Request / Reply*: z.B. für ping und traceroute
 - ➔ *Router Solicitation / Advertisement* (nur IPv6): Suche nach / Bekanntgabe von lokalen Routern
 - ➔ *Neighbor Solicitation / Advertisement* (nur IPv6): Adreßübersetzung (siehe später)
 - ➔ ...

Aufbau eines ICMP-Pakets (siehe auch Wireshark-Aufzeichnung)



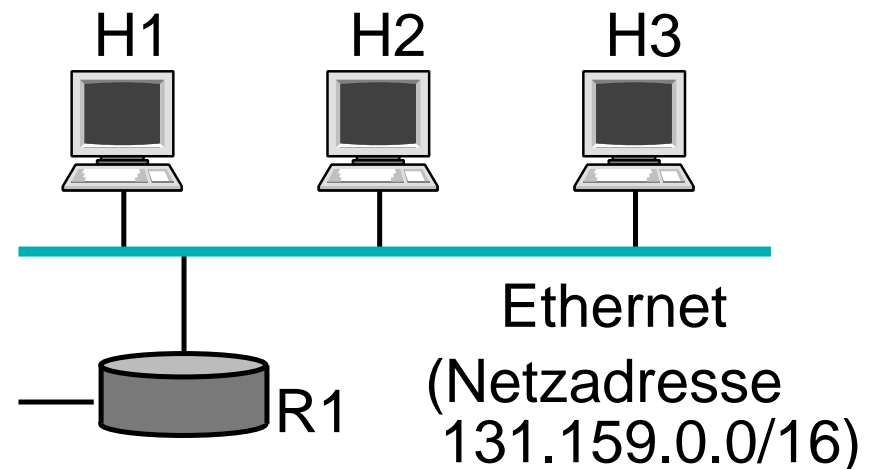
- ➔ *Echo / Echo Reply* (Typ = 8 / 0):
 - ➔ Data: Identifikator + Sequenznummer
- ➔ Ziel nicht erreichbar (Typ = 3):
 - ➔ Code: z.B. 0 $\hat{=}$ Netz, 1 $\hat{=}$ Host, 3 $\hat{=}$ Port nicht erreichbar
 - ➔ Data: IP-Header + erste 64 Datenbytes des unzustellbaren Pakets

Motivation

- ➔ IP-Weiterleitung bringt ein Paket in das richtige LAN
- ➔ Wie funktioniert IP-Kommunikation **innerhalb** eines LANs?

- ➔ Beispiel:

- ➔ R1 empfängt Paket für Rechner 131.159.32.12
- ➔ R1 muß Paket über Ethernet an diesen Rechner weiterleiten
- ➔ Woher weiß R1 die MAC-Adresse des Rechners mit IP-Adresse 131.159.32.12?



- ➔ Anmerkung: dasselbe Problem tritt auch auf, wenn z.B. H1 ein Paket an IP-Adresse 131.159.32.12 senden will



Motivation ...

- ➔ Problem: Umsetzung von IP-Adressen auf MAC-Adressen im lokalen Netz
 - ➔ allgemein: auf Sicherungsschicht-Adresse
- ➔ Lösungs-Alternativen:
 - ➔ MAC-Adresse in IP-Adresse kodieren?
 - ➔ bei IPv4 nicht realisierbar (Ethernet: 48 Bit MAC-Adresse!)
 - ➔ Manuell verwaltete Tabellen?
 - ➔ Verwaltungsaufwand!
 - ➔ Automatisches (dynamisches) Erstellen der Tabellen!

ARP: Address Resolution Protocol (IPv4)

- ➔ Annahme: ein Rechner H will ein Paket an IP-Adresse xyz senden, xyz ist im lokalen Netz
- ➔ H sucht in seinem ARP-Cache nach der zu xyz gehörigen MAC-Adresse
- ➔ Falls gefunden: Paket an diese MAC-Adresse senden
- ➔ Sonst:
 - ➔ H sendet Anfrage (*ARP-Request*) per Broadcast in das LAN: „wer hat IP-Adresse xyz?“
 - ➔ Der betroffene Rechner sendet Antwort (*ARP Reply*) mit seiner IP- und MAC-Adresse zurück
 - ➔ H trägt Zuordnung in sein ARP-Cache ein
 - ➔ automatische Löschung nach bestimmter Zeit ohne Nutzung

Aufbau eines ARP-Pakets (siehe auch Wireshark-Aufzeichnung)

0	8	16	31
Hardwareadrestyp		Protokolladrestyp	
HA-Länge	PA-Länge	Operation	
Sender-Hardware-Adresse (z.B. MAC-Adresse)			
Sender-HW-Ad. (MAC-A.)		Sender-Protokoll-Adr. (IP)	
Sender-Protokoll-Adr. (IP)		Ziel-HW-Ad. (MAC-A.)	
Ziel-Hardware-Adresse (z.B. MAC-Adresse)			
Ziel-Protokoll-Adresse (z.B. IP-Adresse)			

- ➔ ARP wird direkt über das Layer-2-Protokoll übertragen
- ➔ Typ und Länge der Sender- und Zieladressen sind frei wählbar
- ➔ Operation: *request* (1), *reply* (2)

Spezielle Verwendungen von ARP

➔ ARP *probe*

- ➔ zur Prüfung von Konflikten nach Konfiguration der IP-Adresse
- ➔ Host sendet ARP-Anfrage nach seiner gewählten IP-Adresse

➔ *Gratuitous* APR

- ➔ Host sendet ARP-Anfrage mit seiner eigenen IP-Adresse als Sender- und Zieladresse
- ➔ alle anderen Hosts aktualisieren ihren ARP-Cache
- ➔ z.B. beim Booten oder bei Umschaltung auf Backup-Server

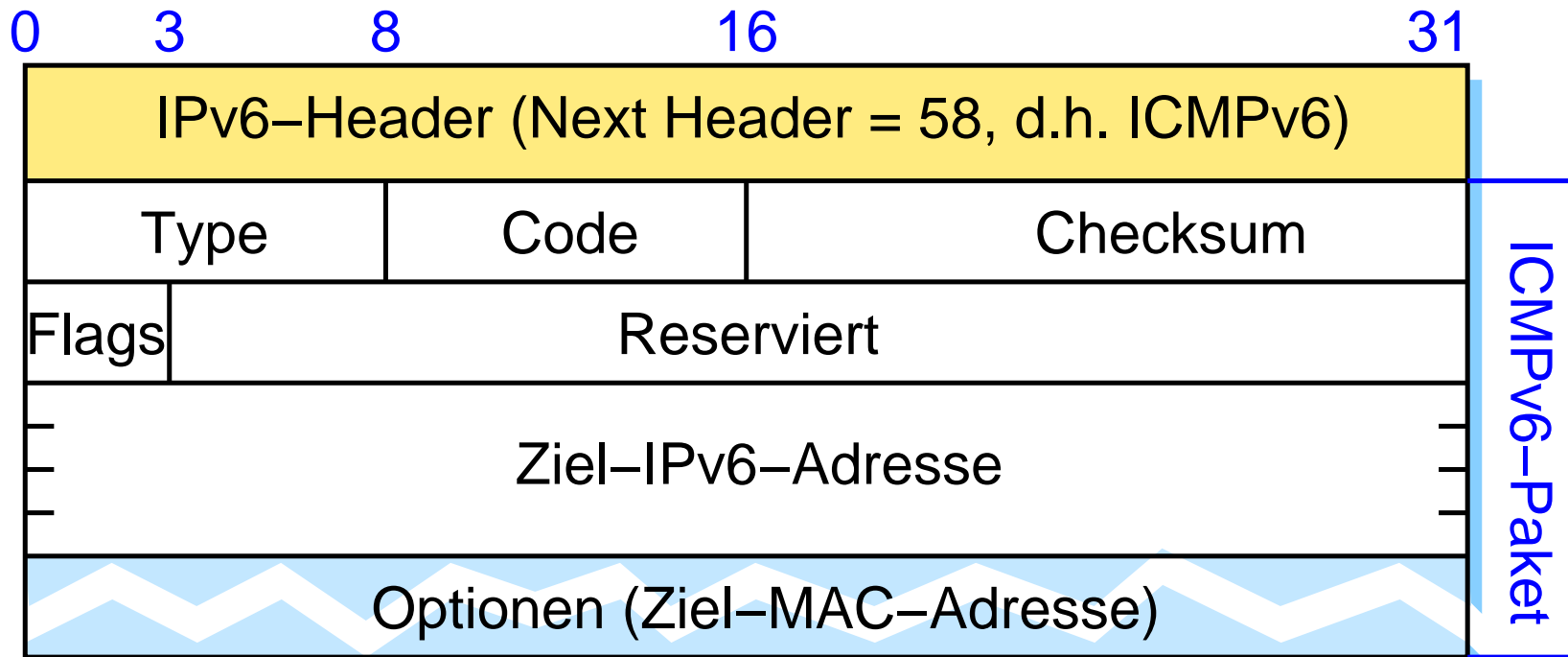
➔ *Proxy* ARP

- ➔ Router sendet ARP-Antwort für Anfrage nach einem Host in einem andern Netz
- ➔ anfragender Host schickt IP-Paket dann an Router

NDP: *Neighbor Discovery Protocol* (IPv6)

- ➔ NDP ist Teilprotokoll von ICMPv6
 - ➔ *Neighbor Solicitation/Advertisement*
- ➔ Funktionsweise der Adreßumsetzung analog zu ARP
- ➔ Unterschiede:
 - ➔ Nutzung von ICMP-Paketen statt ARP-Protokoll
 - ➔ Anfrage: *Neighbor Solicitation*
 - ➔ Antwort: *Neighbor Advertisement*
 - ➔ Anfrage nicht per Broadcast sondern per Multicast
 - ➔ an die zugehörige *Solicited Nodes* Multicast-Gruppe
 - ➔ Gruppe wird aus letzten 24 Bits der Ziel-IP-Adresse bestimmt

Aufbau eines NDP-Pakets (siehe auch Wireshark-Aufzeichnung)



- ➔ *Neighbor Solicitation* (Type = 135)
 - ➔ Multicast an ff02::1:ff00:0/104 + letzte 24 Bits der Zieladresse
- ➔ *Neighbor Advertisement* (Type = 136)
 - ➔ Multicast an alle IPv6 Hosts (ff02::1)
 - ➔ Options-Feld beinhaltet MAC-Adresse des Ziels

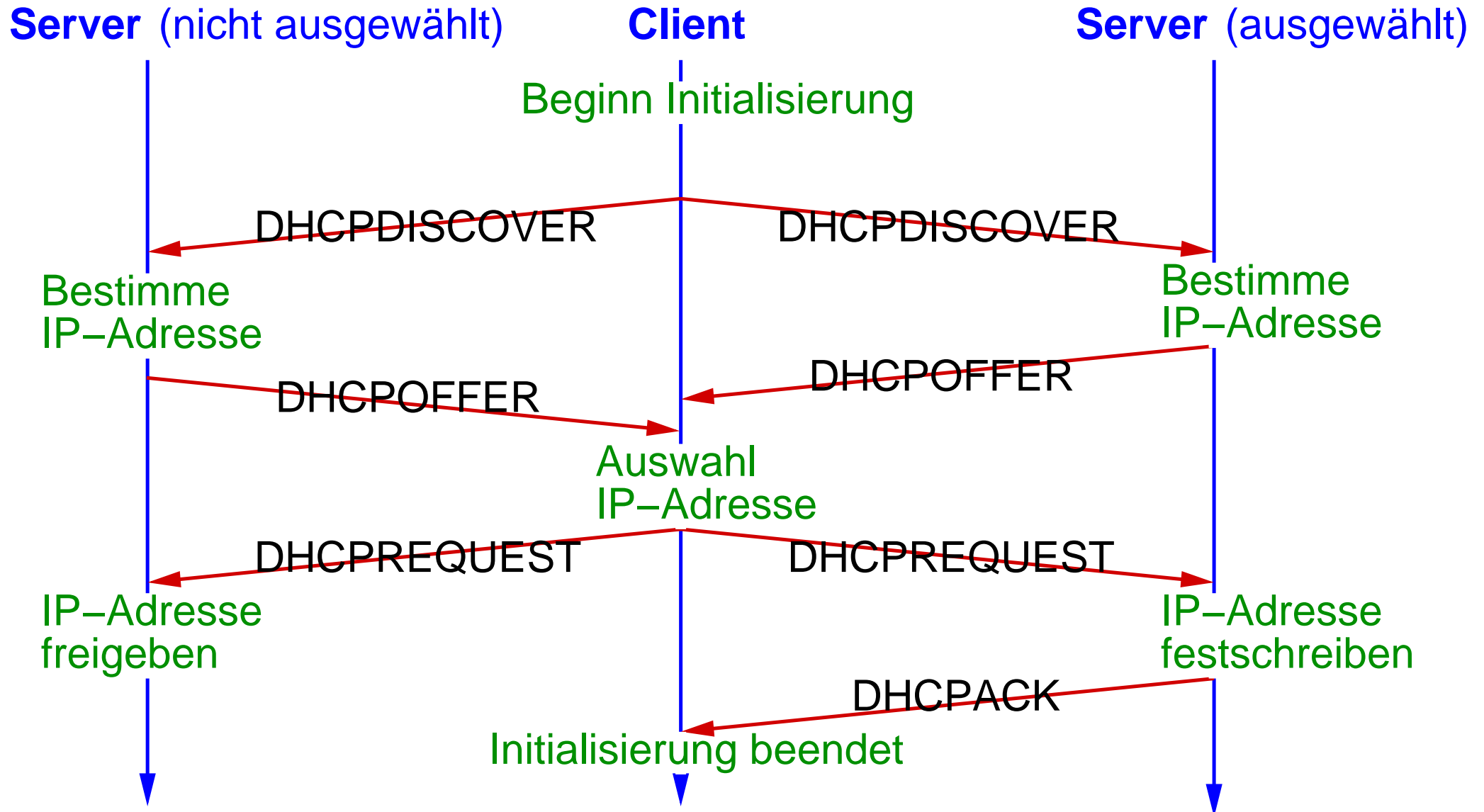
DHCP: *Dynamic Host Configuration Protocol*

- ➔ Automatisiert u.a. die Vergabe von IP-Adressen (IPv4 und IPv6)
- ➔ Vorgehensweise:
 1. Rechner sendet Broadcast-Anfrage (DHCPDISCOVER)
 - ➔ Verbreitung nur im lokalen Netz
 2. DHCP-Server sendet DHCPOFFER: Angebot für IP-Adresse
 - ➔ Zuordnung statisch oder dynamisch, als Schlüssel dient MAC-Adresse des Rechners
 - ➔ ggf. auch weitere Optionen (Hostname, DNS-Server, ...)
 3. Rechner fordert angebotene Adresse vom DHCP-Server an (DHCPREQUEST)
 4. DHCP-Server bestätigt (DHCPACK)
- ➔ IP-Adresse wird nur für eine bestimmte Zeit „gemietet“
 - ➔ periodische Wiederholung der Schritte 3 und 4

5.7 Automatische IP-Konfiguration ...



Ablauf





Aufbau eines DHCP-Pakets (siehe auch Wireshark-Aufzeichnung)

- ➔ Felder u.a. für:
 - ➔ Pakettyp (*request*, *reply*)
 - ➔ angefragte Client-IP-Adresse
 - ➔ im DHCPREQUEST
 - ➔ angebotene Client-IP-Adresse
 - ➔ in DHCPOFFER und DHCPACK
 - ➔ Hardware-Adresse des Clients
 - ➔ i.a. MAC-Adresse
 - ➔ Name einer Boot-Datei
 - ➔ wird vom Server an Client per SFTP übertragen
 - ➔ Optionen
 - ➔ z.B. Anfrage / Zuweisung von Subnetzmaske, Hostname, Default-Gateway, DNS-Server, NTP-Server

Zustandslose IPv6 Autokonfiguration

➡ Vorgehensweise eines Hosts:

1. bilde *link-local* Adresse

➡ Präfix FE80::/10 und z.B. EUI-64 oder Zufallszahl

2. prüfe Adresse mit NDP: *Neighbor Solicitation*

➡ falls keine Antwort: Adresse im LAN eindeutig

3. warte auf *Router Advertisement*

➡ sende ggf. explizit *Router Solicitation* wg. Wartezeit

4. *Router Advertisement* teilt mit:

➡ Adresse und/oder andere Optionen über DHCP holen?

➡ ggf. globales Routing-Präfix und Präfixlänge

5. bilde Adresse aus Präfix und z.B. EUI-64 oder Zufallszahl

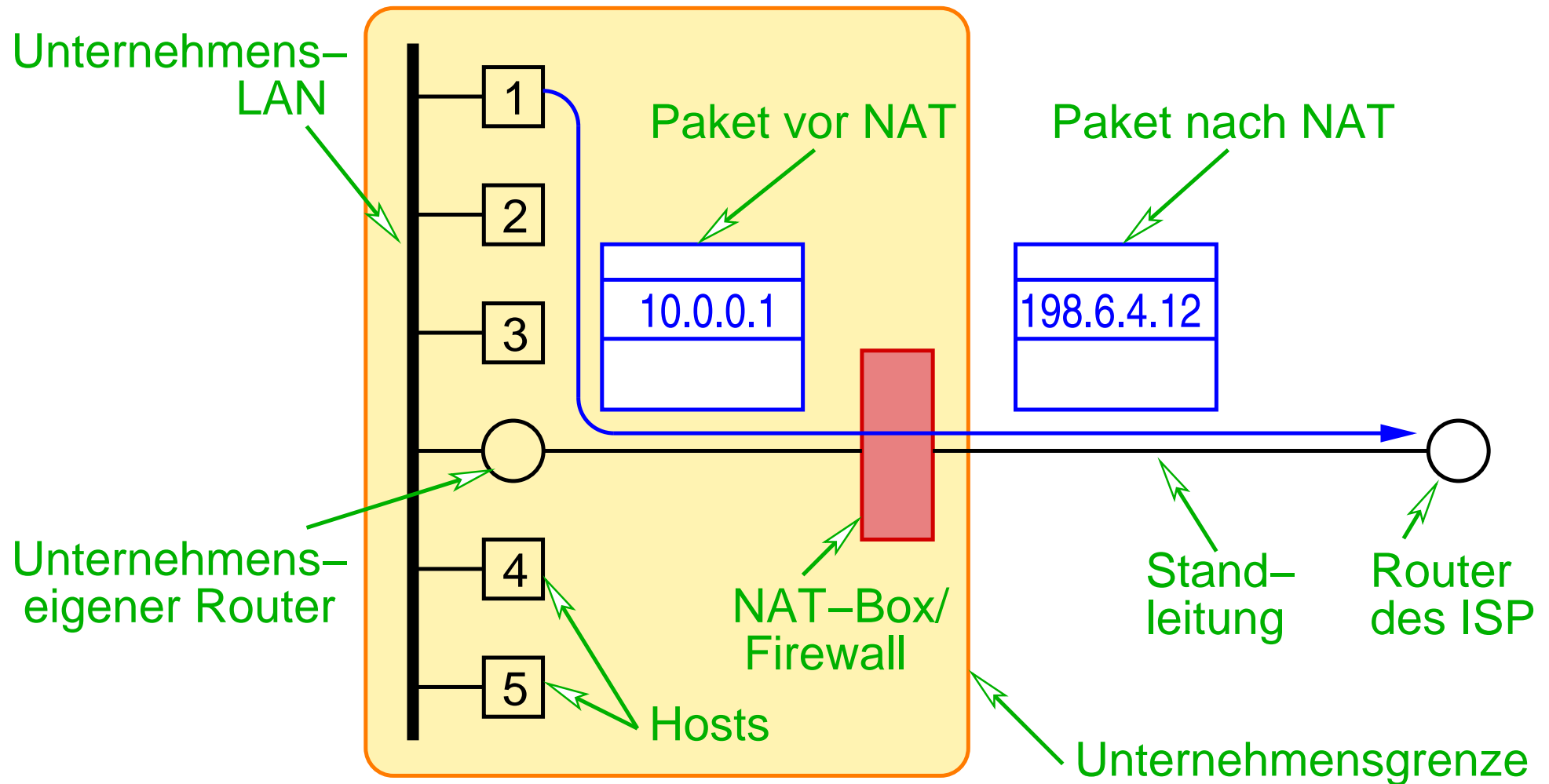
6. prüfe Adresse mit NDP: *Neighbor Solicitation*

➡ Adresse ist i.d.R. nur begrenzte Zeit gültig

NAT, speziell NAPT (*Network Address Port Translation*)

- ➔ Ziel: Einsparung von IP-Adressen
 - ➔ mehrere Rechner eines Netzes werden auf dieselbe, nach außen sichtbare IP-Adresse gemultiplext
- ➔ Prinzip:
 - ➔ jeder Rechner des Netzes bekommt **private** IP-Adresse
 - ➔ drei reservierte Adreßbereiche:
 - ➔ 10.0.0.0/8 (10.0.0.0 - 10.255.255.255)
 - ➔ 172.16.0.0/12 (172.16.0.0 - 172.31.255.255)
 - ➔ 192.168.0.0/16 (192.168.0.0 - 192.168.255.255)
 - ➔ das gesamte Netz erhält **eine** „echte“ IP-Adresse
 - ➔ ausgehende Pakete durchqueren NAT-Box
 - ➔ NAT-Box ersetzt private IP-Adresse durch echte

Prinzipielle Funktionsweise von NAT

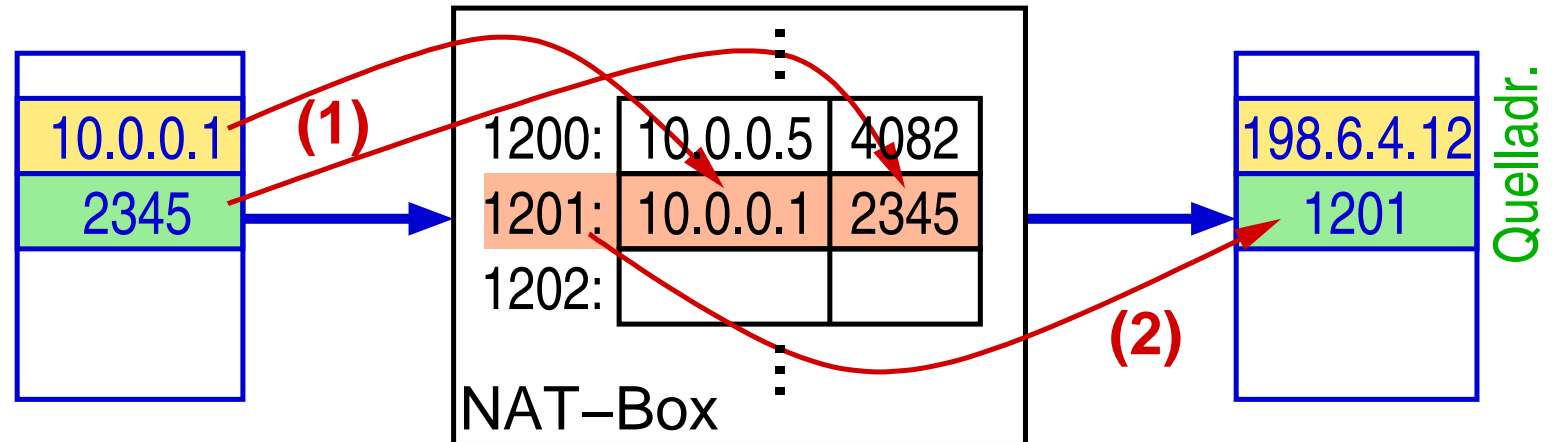


Problem: Zustellung von Antwort-Paketen!?

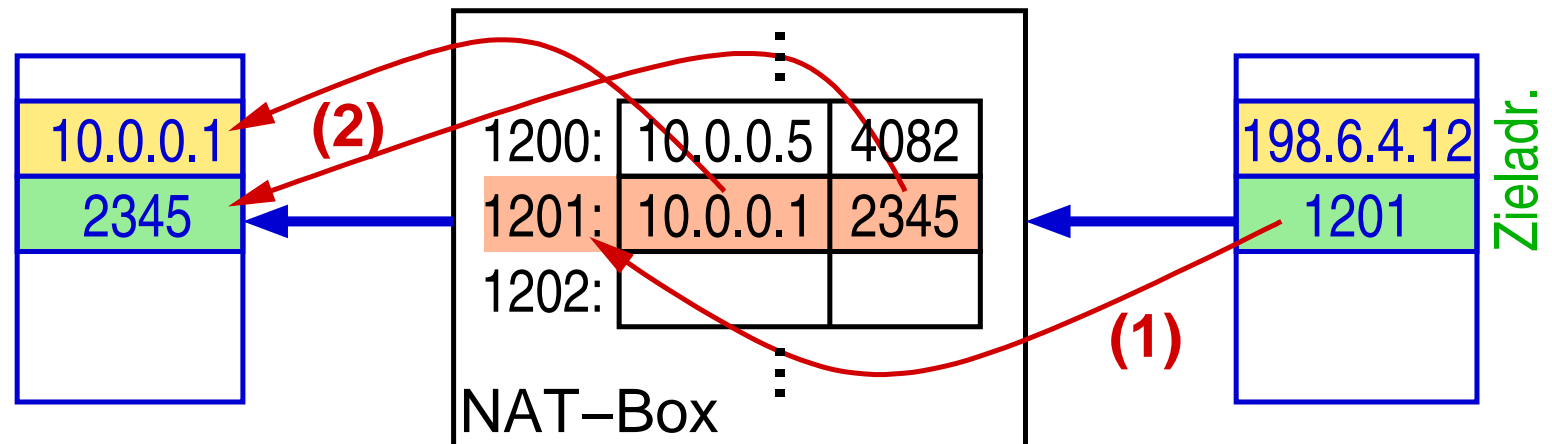
- ➔ Alle eingehende Pakete sind an dieselbe Adresse gerichtet
 - ➔ woher weiß die NAT-Box, an wen das Paket gehen soll?
- ➔ Lösung (schmutzig!):
 - ➔ praktisch der gesamte IP-Verkehr ist UDP oder TCP
 - ➔ UDP- und TCP-Header enthalten Felder zur Adressierung von Quell- und Zielprozess (Quell-/Ziel-Port)
 - ➔ NAT-Box verwendet Quell-Port-Feld des UDP/TCP-Headers zum Speichern eines Demultiplex-Schlüssels
 - ➔ bei ausgehendem Paket: speichere Quell-IP-Adresse und Quell-Port (\Rightarrow Schlüssel)
 - ➔ ersetze IP-Adresse, ersetze Quell-Port durch Schlüssel
 - ➔ Antwortpaket enthält Schlüssel als Ziel-Port
 - ➔ ersetze Ziel-Adresse/Port durch gespeicherte Adresse/Port

Genaue Funktionsweise von NAT

A) Paket
wird
gesendet



B) Antwort-
paket
kommt an



IP-Adresse im IP-Header

Port im TCP/UDP-Header



Kritik:

- ➔ IP-Adressen nicht mehr weltweit eindeutig
- ➔ Macht aus Internet quasi verbindungsorientiertes Netz
 - ➔ Ausfall der NAT-Box: Zerstörung aller Kommunikationsbeziehungen (TCP und UDP)!
- ➔ NAT (Schicht 3) macht Annahmen über Schicht 4!
 - ➔ neue TCP/UDP-Version, andere Protokolle über IP !?
- ➔ Nutzdaten können IP-Adressen enthalten (z.B. FTP, H.323)

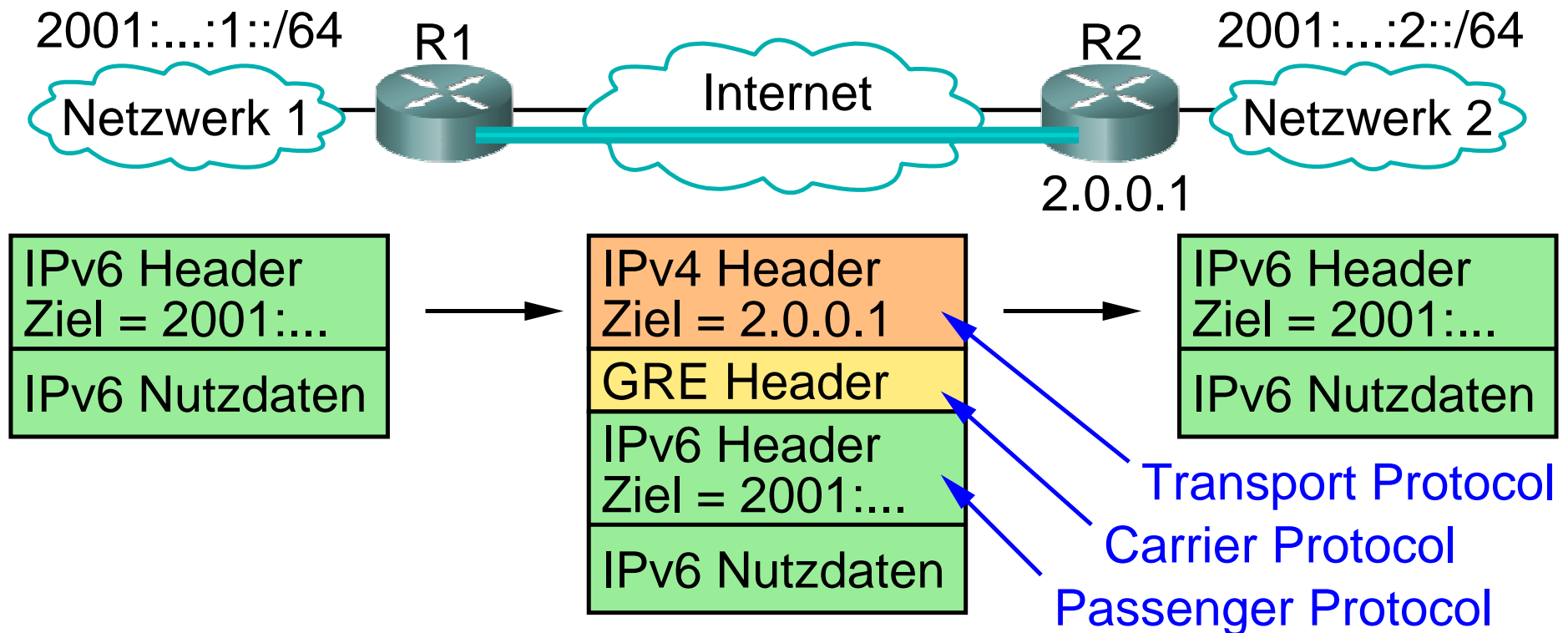
Vorteile:

- ➔ Einsparung von IP-Adressen
- ➔ Sicherheit: Hosts sind von außen nicht erreichbar, Struktur des lokalen Netzes nach außen verborgen

5.9 Tunneling



- ➔ Übertragung eines Protokolls über ein Protokoll derselben oder einer höheren OSI-Schicht
- ➔ z.B. Übertragung von IPv6-Paketen über IPv4





- ➔ Aufgaben des *Carrier*-Protokolls:
 - ➔ z.B. Multiplexing, Reihenfolgeerhaltung, Authentifizierung, ...
 - ➔ ggf. kann das Carrier-Protokoll auch fehlen
- ➔ Einsatz von Tunneln:
 - ➔ Kommunikation zwischen Routern mit speziellen Fähigkeiten
 - ➔ z.B. Multicast, IPv6
 - ➔ Kopplung von nicht-IP Netzen (z.B. Ethernet) über das Internet
 - ➔ VPNs: Authentifizierung und Verschlüsselung im Tunnel
 - ➔ „Durchtunneln“ von Firewalls
 - ➔ oft HTTP als Transportprotokoll

Verschiedene Ansätze

- ➔ Dual-Stack-Ansatz
 - ➔ Netzknoten implementieren sowohl IPv6 als auch IPv4
 - ➔ Fähigkeiten des Zielknotens über DNS zu ermitteln
 - ➔ IPv6-Adresse für IPv6-fähigen Knoten, IPv4-Adresse sonst
 - ➔ Router müssen ggf. IP-Header der Pakete anpassen
 - ➔ wenn zwischenliegende Router nur IPv4 unterstützen
 - ➔ Informationsverlust möglich (z.B. FlowLabel nur in IPv6)
- ➔ Tunneling
 - ➔ Aufbau eines IPv4-Tunnels zwischen IPv6-Knoten
 - ➔ d.h. IPv6-Pakete werden in IPv4-Pakete eingepackt
- ➔ Adreßübersetzung analog zu NAT (6to4)



Stand

- ➔ Praktisch alle neuen Features von IPv6 inzwischen auch in IPv4 verfügbar
 - ➔ Hauptmotivation: größerer Adreßbereich
- ➔ Moderne Betriebssysteme beinhalten IPv4 und IPv6
 - ➔ z.B. Linux, Windows (ab XP)
- ➔ Bis Juni 2006: Internet-Testbett 6bone
- ➔ IPv6 inzwischen im „Produktionseinsatz“
 - ➔ einige Dienste nur noch mit IPv6 angeboten
- ➔ Zeitrahmen für die Ablösung von IPv4 aber unklar



- ➡ Internetwork: Netz von Netzen
- ➡ IP-Protokoll: Best-Effort, run over everything
 - ➡ Hierarchische Adressen: Routing nur zwischen Netzen
 - ➡ Bessere Skalierbarkeit durch CIDR und Subnetze
- ➡ Hilfsprotokolle: ICMP, ARP, NDP, DHCP
- ➡ NAT: Umsetzung privater auf globale IP-Adressen

Nächste Lektion:

- ➡ Routing