Rechnernetze:
(6) Netzwerkschicht

Prof. Dr. Klaus-Peter Kossakowski

Gliederung der Vorlesung

- **Einführung und Historie des Internets**
- **Schichtenmodell**
- **Netzwerk als Infrastruktur**
- **Layer 7: Anwendungsschicht**
- **Layer 7/4: Socket-Programmierung**
- **Layer 4: Transportschicht**
- **Layer 3: Netzwerkschicht**
  - IPv4, ICMP, IPv6
  - Routing, Management und Firewalls
- **Layer 2: Sicherungsschicht**

Inhalte dieses Kapitels

In diesem Kapitel behandeln wir die beiden im Einsatz befindlichen Versionen des Internet-protokolls (IP).

Es werden die Protokollheader und die darin verwendeten Daten besprochen.

Ergänzend werden die zugeordneten Kontrollprotokolle (ICMP) angesprochen.

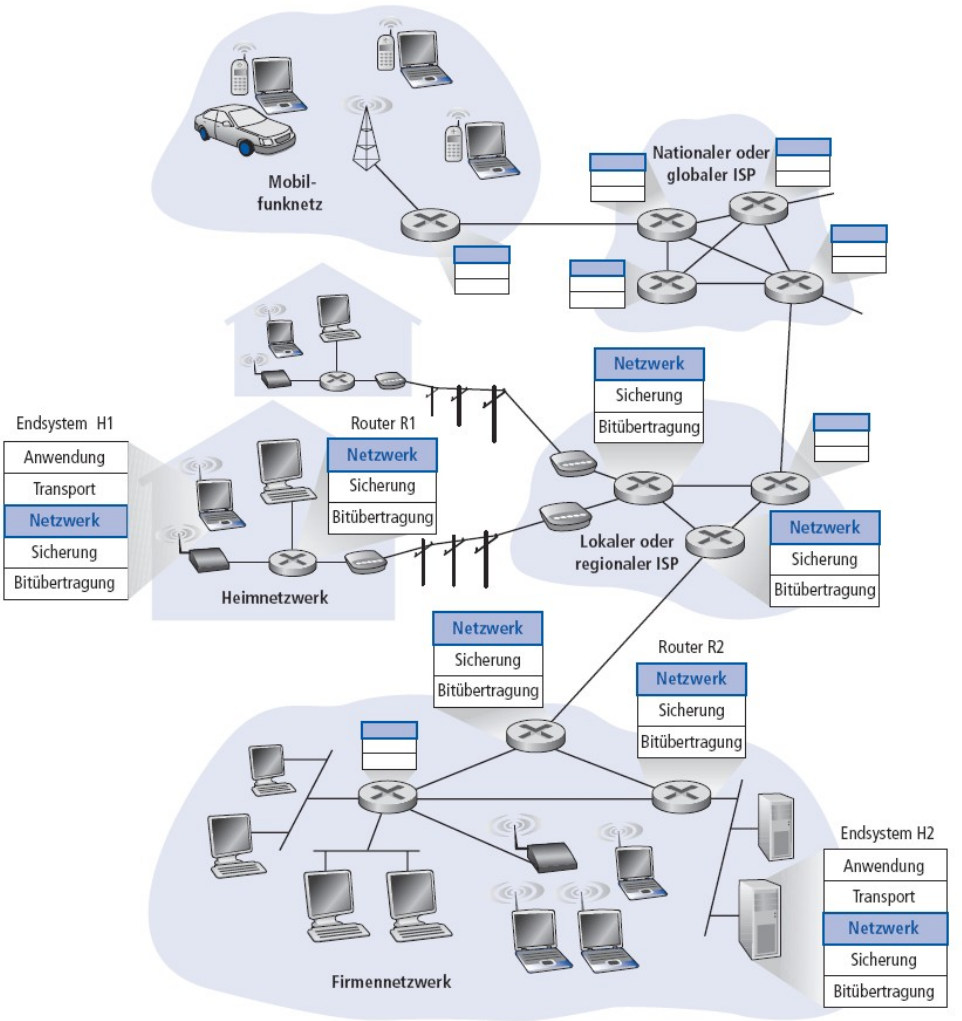
Konzepte der Netzwerkschicht wie Network Address Translation (NAT) werden erläutert.

Die Probleme eines gemischten Betriebs von IPv4 und IPv6 im heutigen Internet werden angesprochen.

Ziele dieses Kapitels

- Sie kennen die IP-Protokollheader und typische Funktionen von ICMP.
- Sie können den protokollkonformen Ablauf eines Austauschs von IP-Datagrammen erläutern und welche Probleme durch eine Fragmentierung gelöst werden.
- Sie können das Konzept der NAT bei IPv4 erklären.
- Sie können die Auswirkungen eines gemischten Betriebs von IPv4 und IPv6 für die Entwicklung von Anwendungen erklären und kennen mögliche Betriebsformen.

… und was macht nochmal eine Netzwerkschicht?

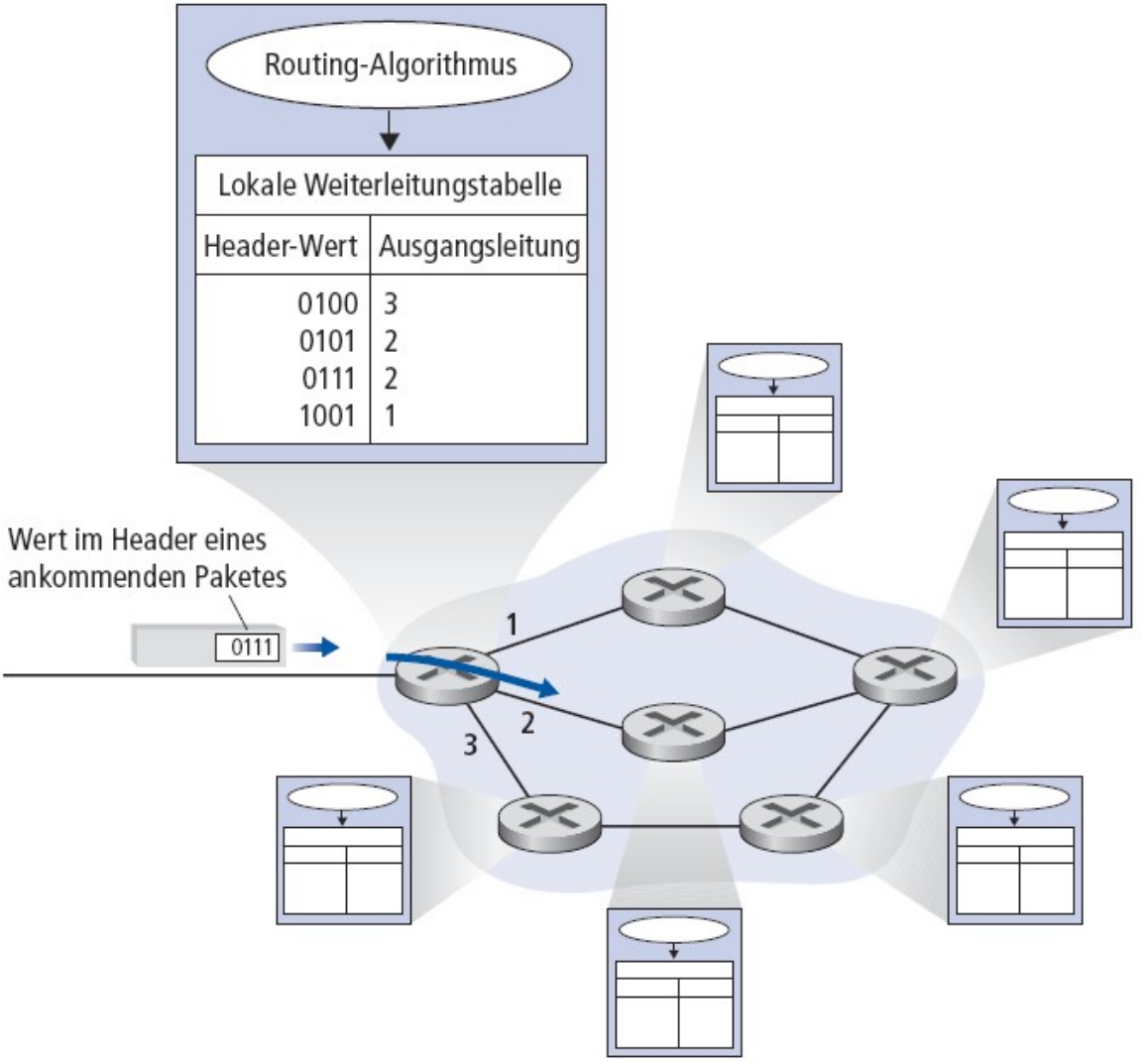

Mobil-
funknetz
Nationaler oder
globaler ISP
Netzwerk
Sicherung
Bitübertragung
Endsystem H1
Anwendung
Transport
Netzwerk
Sicherung
Bitübertragung
Router R1
Netzwerk
Sicherung
Bitübertragung
Heimnetzwerk
Lokaler oder
regionaler ISP
Netzwerk
Sicherung
Bitübertragung
Netzwerk
Sicherung
Bitübertragung
Router R2
Netzwerk
Sicherung
Bitübertragung
Endsystem H2
Anwendung
Transport
Netzwerk
Sicherung
Bitübertragung
Firmennetzwerk

Funktion der Netzwerkschicht

**Transport von Paketen,
vom sendenden zum empfangenden Host.**

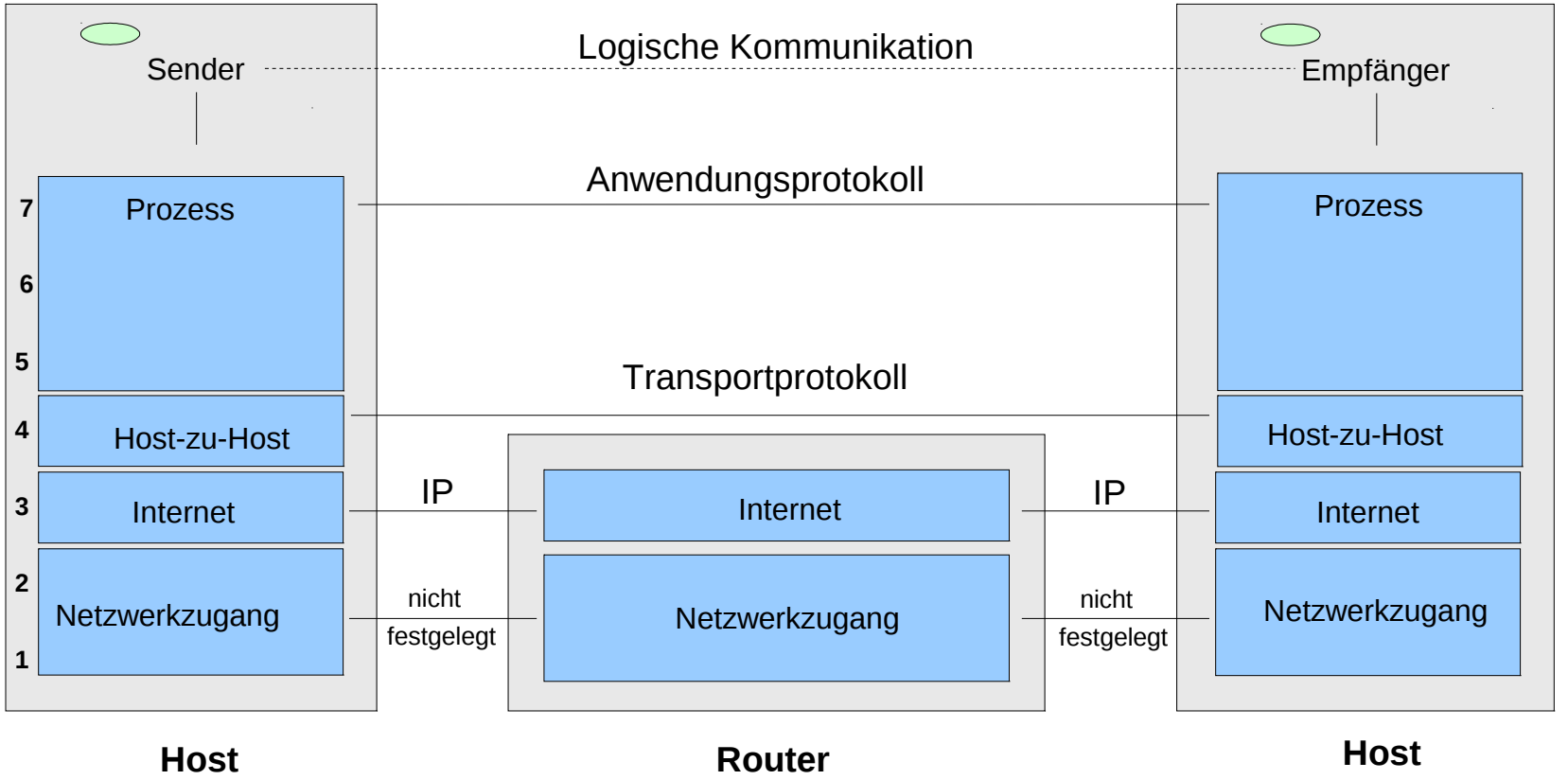
Um das zu können, bedarf es der Router:

- **Pfadbestimmung:**
  - Entscheidung über den Weg (Route), den die Pakete von der Quelle zum Ziel nehmen
- **Portfestlegung:**
  - Pakete vom Eingang des Routers müssen zum richtigen Ausgang hinaus

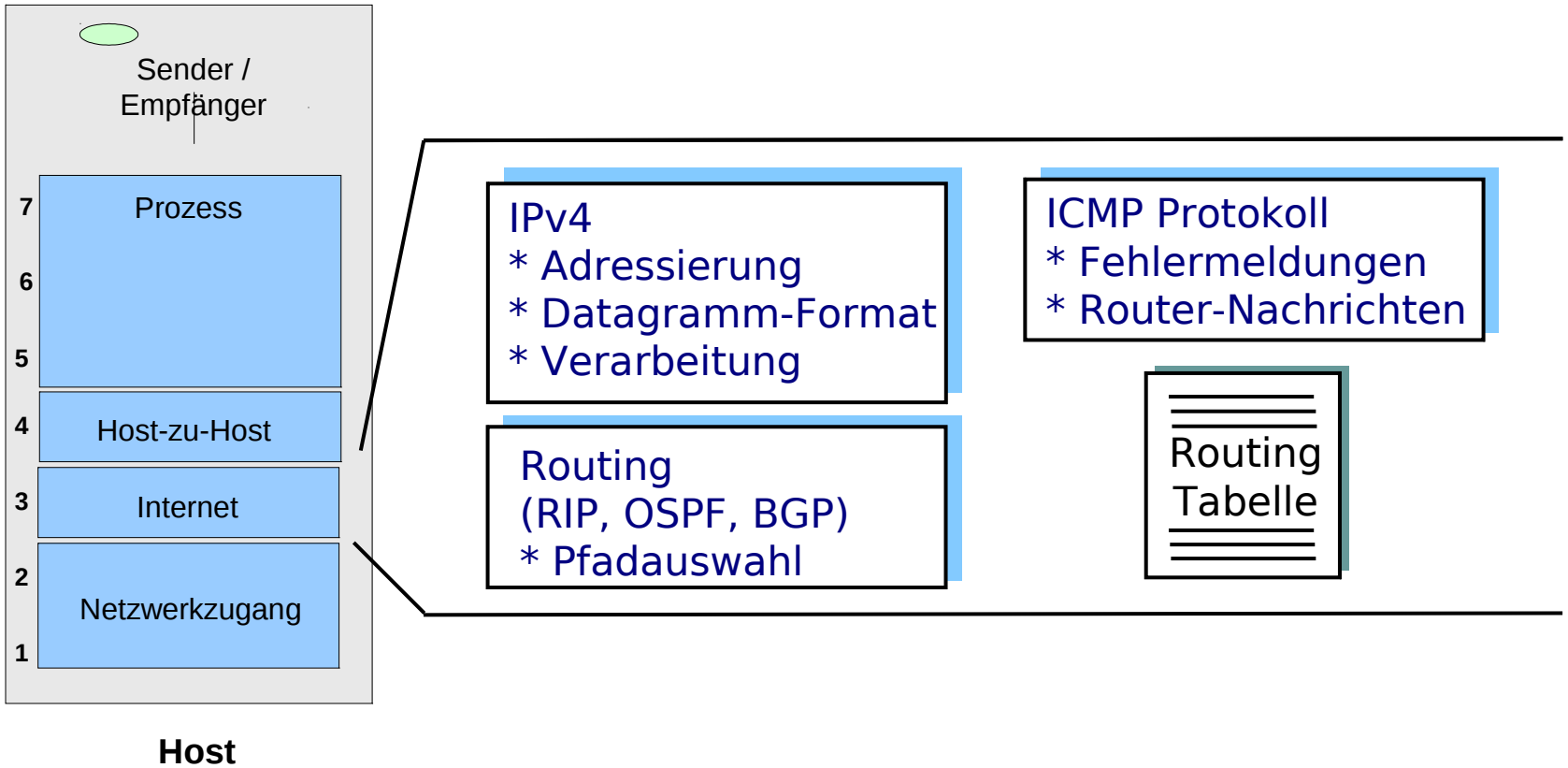

Routing-Algorithmus
Lokale Weiterleitungstabelle
Header-Wert
Ausgangsleitung
0100 3
0101 2
0111 2
1001 1
Wert im Header eines ankommenden Paketes
0111
1
2
3

IPv4

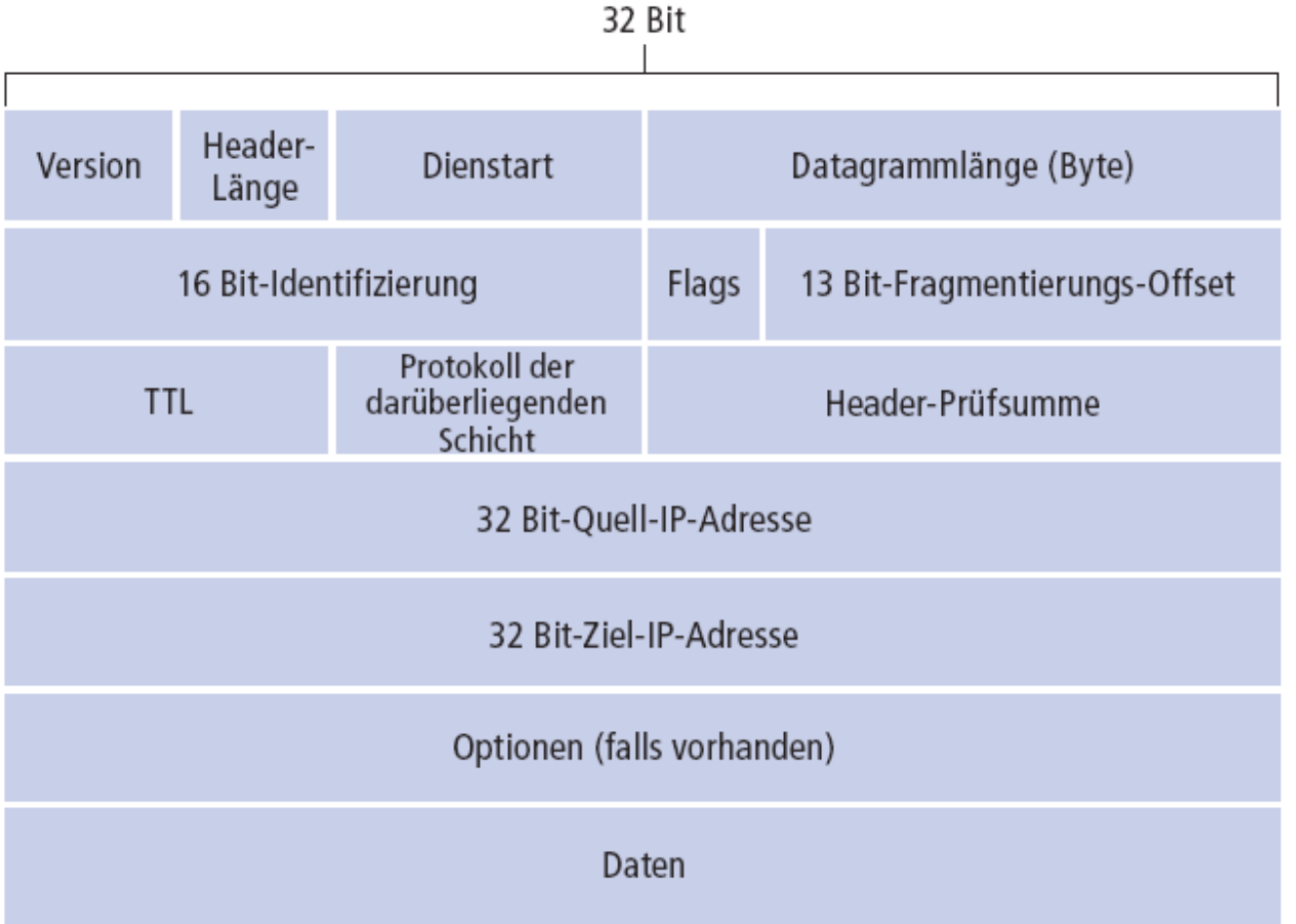
Das Internet-Modell



IPv4 als Teil des Internet-Modells



IPv4-Header gemäß [RFC 791]



Felder des IPv4-Header

Version (version, 4 bit)

- Version des IP-Protokolls
- Zur Zeit existieren nur Versionen 4 und 6

Headerlänge (hlength, 4 bit)

- Gibt die Länge des IP-Headers – jeweils in 32 bit Einheiten – an
- Erlaubt Variable Anzahl von Optionen

Dienstart (type of service, 8 bit)

- Definition der Behandlung eines IP-Datagramms → siehe [RFC 1349]

Felder des IPv4-Header: Dienstart / Type of Service

Datagramme werden gemäß des „best effort“ -Prinzips ausgeliefert. Durch ToS werden Datenflüsse priorisiert, nichts garantiert!

Application	Low delay	High throughput	High reliability	Low cost
Telnet	⊗			
ftp (data)		⊗		
SNMP			⊗	
NNTP				⊗

Felder des IPv4-Header (2)

Datagrammlänge (total length, 16 bit, >= 0x14)

- Gesamtlänge des Datagramms in Oktetts (=Bytes)
- Incl. der Länge des IP-Headers, etwaige Optionen und der Nutzdaten (Payload)
- Obergrenze: 65.535 Oktetts
  - wird diese Länge von einem Netzwerk nicht unterstützt, muss fragmentiert werden
  - Bei Einzelfragmenten wird immer die Länge des vorliegenden Fragments im Header eingetragen!

Felder des IPv4-Header (3)

Identifizierung (identification,16 bit)

- Kennzeichnung, um Einzelfragmente wieder zu einem Datagramm zusammenführen zu können
- Geschieht erst beim Zielrechner, d.h. Router führen keine De-Fragmentierung durch

Fragmentierungs-Offset (13 bit)

- Position der Daten – in Bytes – des Fragments im Verhältnis zum ursprünglichen Datagramm

Felder des IPv4-Header (4)

Flags (3 bit)

- 010: Don't fragment
  - Kann das Paket nicht übertragen werden, wird eine Fehlermeldung geschieht
- 001: More fragments
  - Es folgt noch ein Fragment
- Routing für Fragment bleibt unabhängig von der Übertragung anderer Fragmente
  - Destination- und Source- Address werden beibehalten
  - Total Length, Identification, Flags, Fragment Offset und Header Checksum ändern sich

Felder des IPv4-Header (5)

TTL (time to live, 8 bit)

- Anzahl der Hops, die das Datagramm noch „überlebt“
- Jeder Router dekrementiert den Zähler des Pakets auf dem Weg durch das Internet
  - Erreicht der Zähler den Wert „0“, wird das Paket entfernt

Protokoll (8 bit)

Protocol No.	Protocol	Kommentar
0	IP	Internet Protocol Version 4
1	ICMP	Internet Control Message Protocol
6	TCP	Transmission Control Protocol
17	UDP	User Datagramm Protocol
41	IPv6	Internet Protocol Version 6

Größe von Datagrammen

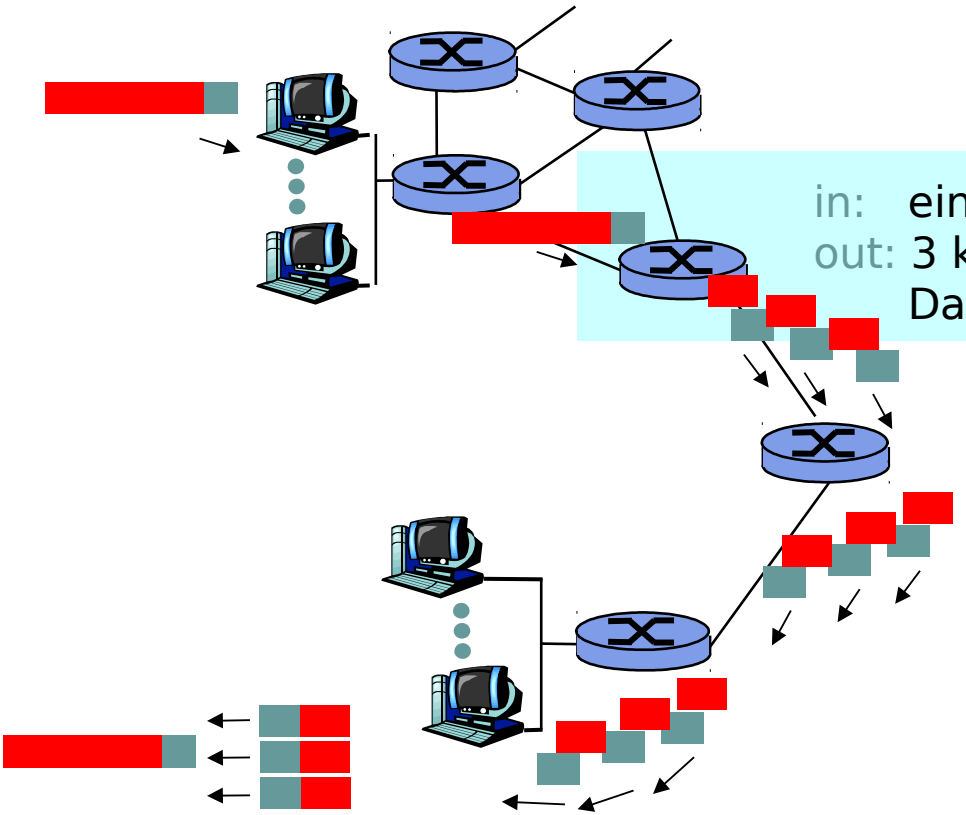
Die maximale Größe von Datagrammen ist abhängig von dem Medium der Sicherungsschicht (Level 2)

Maximum Transfer Unit (MTU) in Bytes:

- FDDI 4.500
- Ethernet 1.500
- IEEE 802.3 1.492

Große MTUs nutzen das Medium besser aus, aber Störungen verursachen hohe Verluste.

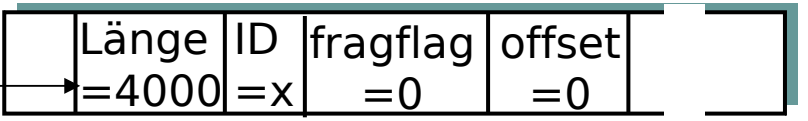
Wie sieht das im Netz aus?



in: ein großes Datagramm
out: 3 kleinere Datagramme

IPv4-Fragmentierung

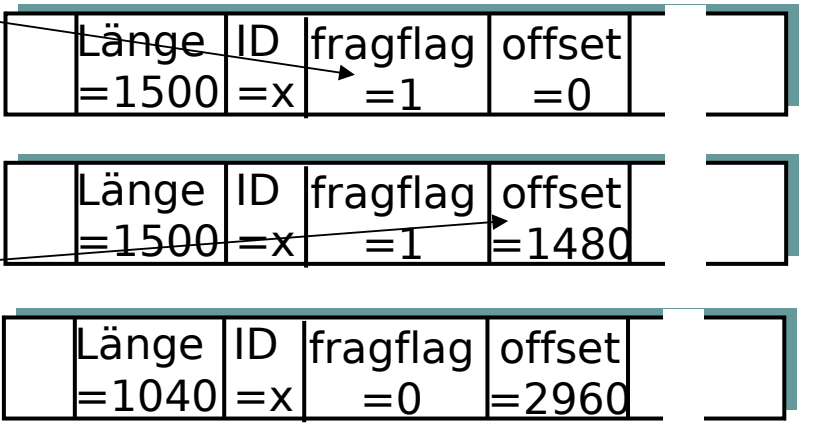
20 Byte IP-Header
= 3980 Byte
Nutzdaten



Aus *einem* großen Datagramm werden mehrere kleine Datagramme

flag = 1 zeigt an, dass noch mehr kommt!

offset = 1480 heißt, dass die Daten am Ziel ab Byte 1480 wieder eingefügt werden müssen!



Defragmentierung am Ziel

- Trifft das erste Fragment (MF=1) ein, wird ein Timer gestartet (Werte unterschiedlich abhängig von Implementierung : 15-30 s)
- Ist beim Ablauf des Timers die Nachricht noch unvollständig, MF=0 identifiziert letztes Paket, wird sie komplett verworfen
- Innerhalb des IPv4-Protokolls erfolgt keine Detektion, eine Sendewiederholung muss durch die Transportschicht (Layer 4) initiiert werden

Network Address Translation

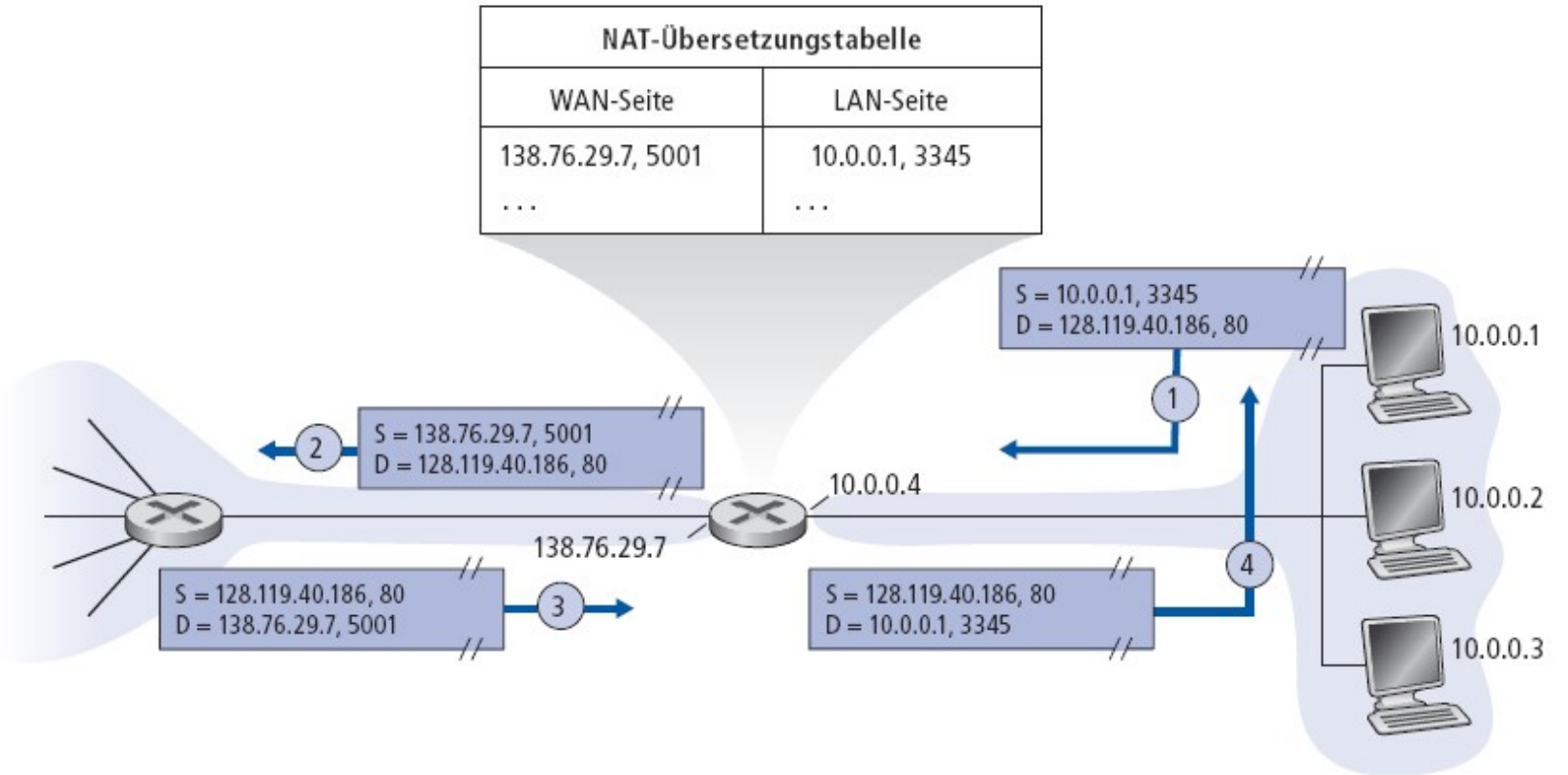
Es gibt zu wenig Adressen

- **Häufig hat man nur eine IP-Adresse, aber mehrere Endsysteme**
  - Z.B. vom ISP, temporär z.B. per DHCP
- **Interne IP-Adressen sollen**
  - Nicht verändert werden müssen, wenn man den ISP wechselt
  - Nicht vom Internet aus ermittelt werden können
  - Jederzeit änderbar sein, ohne jemanden fragen zu müssen

Idee für NAT

- **Vergabe „eigener“ lokaler Adressen an die Systeme im eigenen Netzwerk**
  - Das diese weltweit nicht eindeutig sind, macht nichts, bleibt ja intern
- **Der NAT-fähige Router zur Anbindung an das Internet übersetzt diese lokalen Adressen in eine gemeinsame, weltweit dann jedoch eindeutige, IP-Adresse**
  - dazu wird die Adressierung auf der Transportschicht „missbraucht“

Ports differenzieren Verbindungen

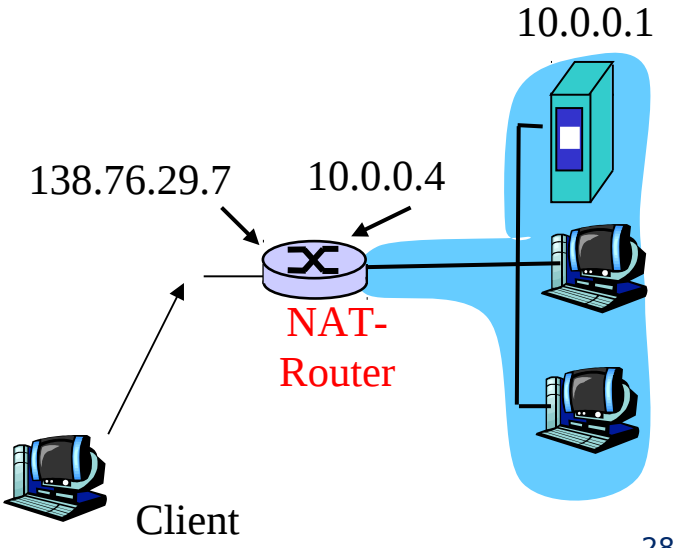


Bewertung von NAT

- **Ports sind 16-Bit lang**
  - Nur etwas mehr als 65.000 gleichzeitige TCP-Verbindungen mit einer IP-Adresse
- **Router arbeiten doch nur auf Layer 3?**
  - Verletzung des sogenannten Ende-zu-Ende-Prinzips (end-to-end principle):
    - Transparente Kommunikation von Endsystem zu Endsystem, im Inneren des Netzes wird nicht an den Daten „herumgepfuscht“
  - Anwendungsentwickler müssen die Präsenz von NAT-Routern berücksichtigen.

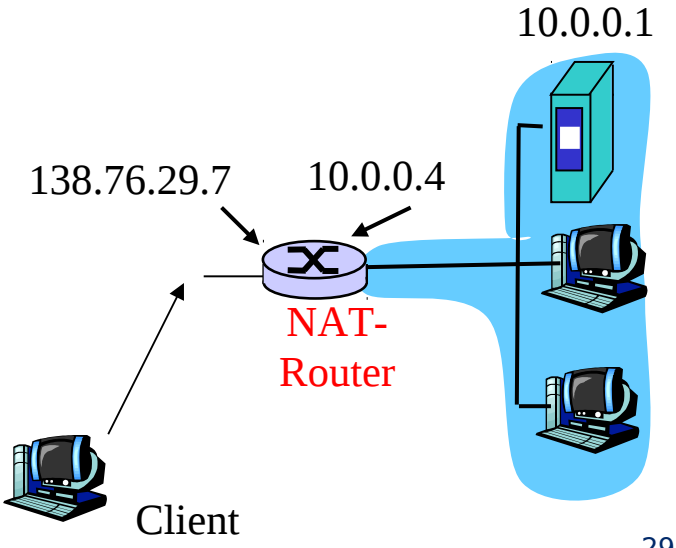
Durchqueren von NAT

- **Internet-Client möchte einen lokalen Server kontaktieren**
  - Die einzige nach außen sichtbare Adresse ist: 138.76.29.7



Durchqueren von NAT

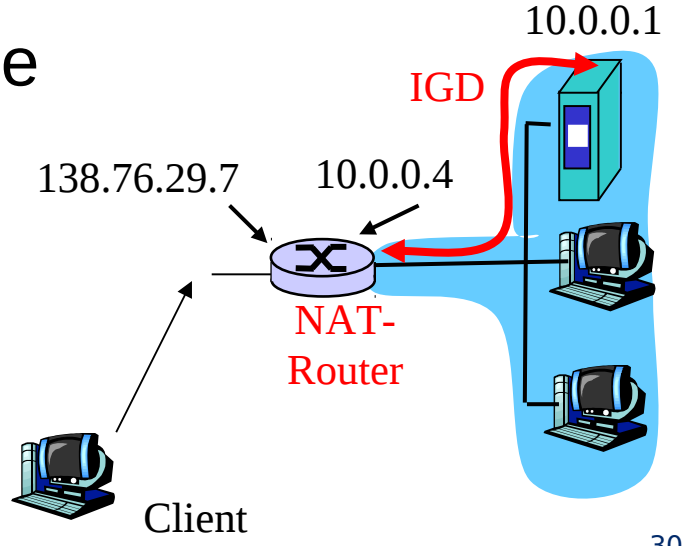
- **Internet-Client möchte einen lokalen Server kontaktieren**
  - Die einzige nach außen sichtbare Adresse ist: 138.76.29.7
- **Lösung 1: (138.76.29.7 und Port 2500) wird immer an 10.0.0.1, Port 25000 weitergeleitet**



Durchqueren von NAT - 2

Lösung 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol.

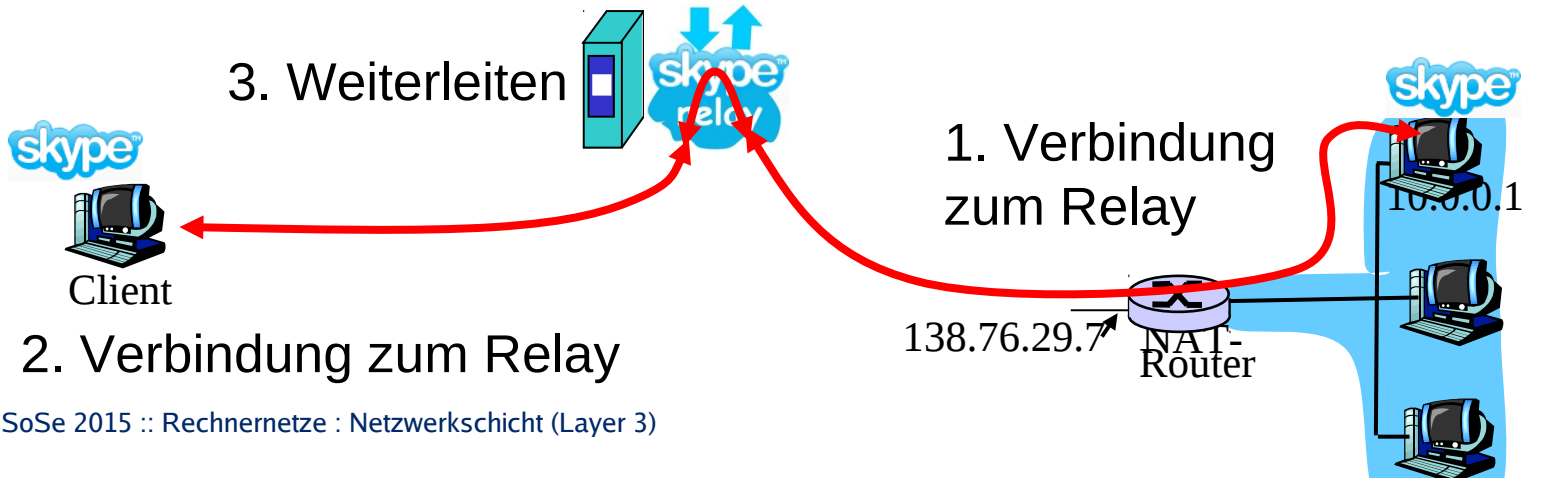
- **Dies ermöglicht dem Host hinter dem NAT Folgendes:**
  - Herausfinden der öffentlichen IP-Adresse
  - Laden der NAT-Tabelle
  - Einträge einfügen oder löschen
  - → **automatische Konfiguration**



Durchqueren von NAT - 3

Lösung 3: Relaying, z.B. bei Skype

- Server hinter einem NAT-Router baut eine Verbindung zu einem Relay im Internet auf
- Client baut eine Verbindung zum Relay auf
- Relay leitet die Pakete vom Client zum Server – und umgekehrt – weiter



ICMP

Kontrollfunktionen beim Versenden von Datagrammen

Die Netzwerkschicht bietet nur einen ungesicherten Datagramm-Dienst, d.h.

- Datagramme können verloren gehen
- Datagramme können dupliziert werden
- Datagramme können in ungeordneter Reihenfolge eintreffen
- Datagramme können verändert ankommen

Auf der IP-Ebene benachrichtigt das ICMP - Internet Control Message Protocol über Fehler

Nachrichten des ICMP

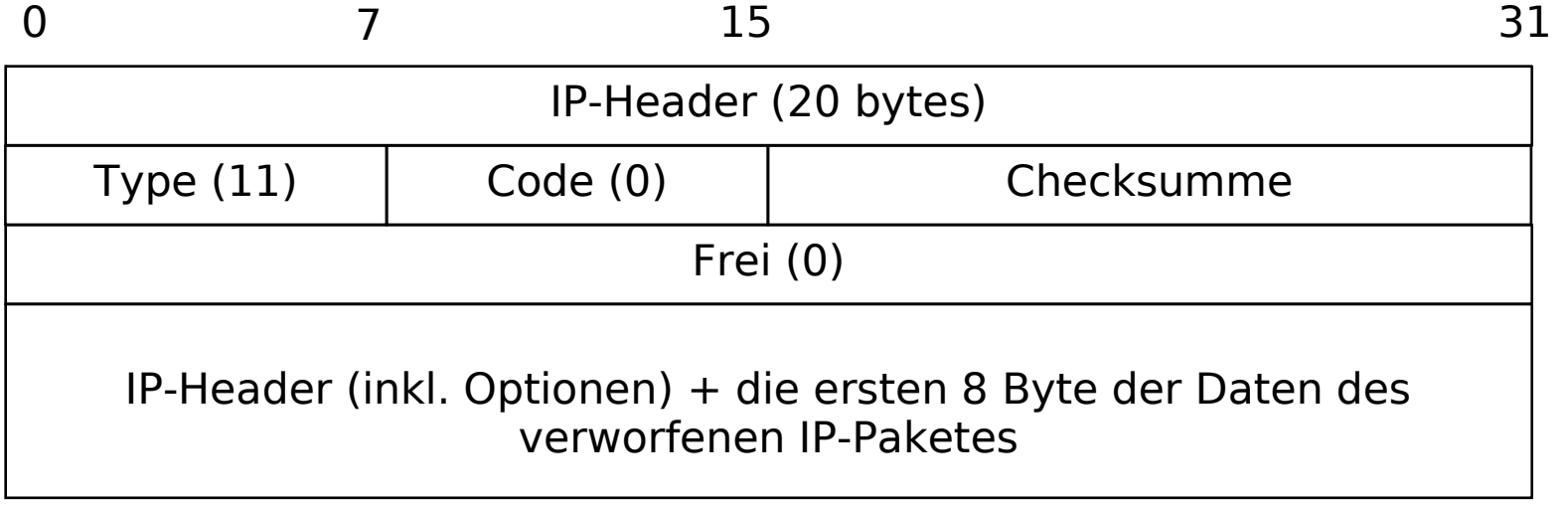
- **Echo Request - Echo Reply**
  - Überprüfen der Betriebsbereitschaft oder Performance (ping)
- **Destination unreachable**
  - Netzwerk, Rechner, Protokoll oder Port sind nicht erreichbar
- **Source Quench**
  - Empfänger hat keine Puffer mehr frei

Nachrichten des ICMP (2)

- **Redirect**
  - Gateway teilt die IP-Adresse eines besser geeígneten Gateways mit
- **Time Exceed**
  - Benachrichtigung über vernichtetes Datagramm (TTL = 0)
- **Bad IP Header**
  - Fehler im IP-Header
- **Timestamp Request / Reply**
  - Angabe der lokalen Zeit

Aufbau von ICMP-Paketen [RFC792]

- Jedes ICMP-Paket ist in IP-Paket eingeschlossen
- Nur der Grundaufbau des ICMP-Headers gleich, die Bedeutung der einzelnen Felder im ICMP-Header ändert sich jedoch.



Traceroute als „Hack“ des ICMP

- **Traceroute bestimmt Informationen über alle Router, die auf dem Weg zu einer IP-Adresse liegen**
  - Dabei wird jeweils auch die Round-Trip-Zeit ermittelt

Traceroute als „Hack“ des ICMP

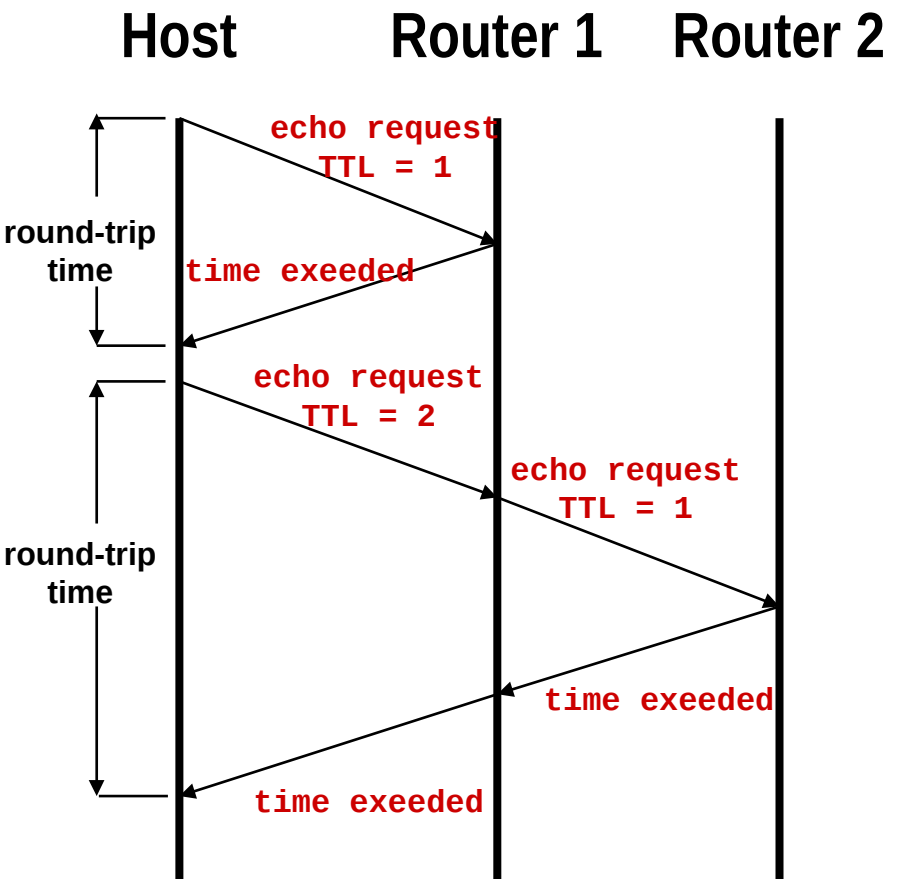
- **Traceroute bestimmt Informationen über alle Router, die auf dem Weg zu einer IP-Adresse liegen**
  - Dabei wird jeweils auch die Round-Trip-Zeit ermittelt

  sofern die Router mitspielen!

Traceroute als „Hack“ des ICMP

- **Funktionsweise:**
  - Traceroute schickt ein UDP-Paket an die Adresse, für die der Weg untersucht werden soll; TTL im IP-Header wird auf 1 gesetzt
    - Hoffentlich ist der UDP-Port gerade frei ...
  - Der erste Router verwirft das IP-Paket (TTL = 1!) und schickt eine ICMP-Time-Exceeded-Fehlermeldung an den Absender
  - Traceroute wiederholt dies mit TTL = 2 etc.

Traceroute: ein Beispiel



… und darf ein Router oder Endgerät auf ein ICMP-Paket mit einem weiteren ICMP-Paket antworten?

… und darf ein Router oder Endgerät auf ein ICMP-Paket mit einem weiteren ICMP-Paket antworten?

Immer?

Keine ICMP-Nachrichten, falls:

- Reaktion auf andere ICMP-Nachrichten
- Reaktion auf Pakete, deren Zieladressen IP Broadcast oder IP Multicast Adressen sind
- Reaktion auf Pakete, die mittels Link Layer Multicast bzw. Broadcast versandt wurden
- die Ursache ein Paket war, dessen Ziel kein eindeutiger Host war (NULL-, Loopback-, Broadcast- oder Multicast-Adresse)

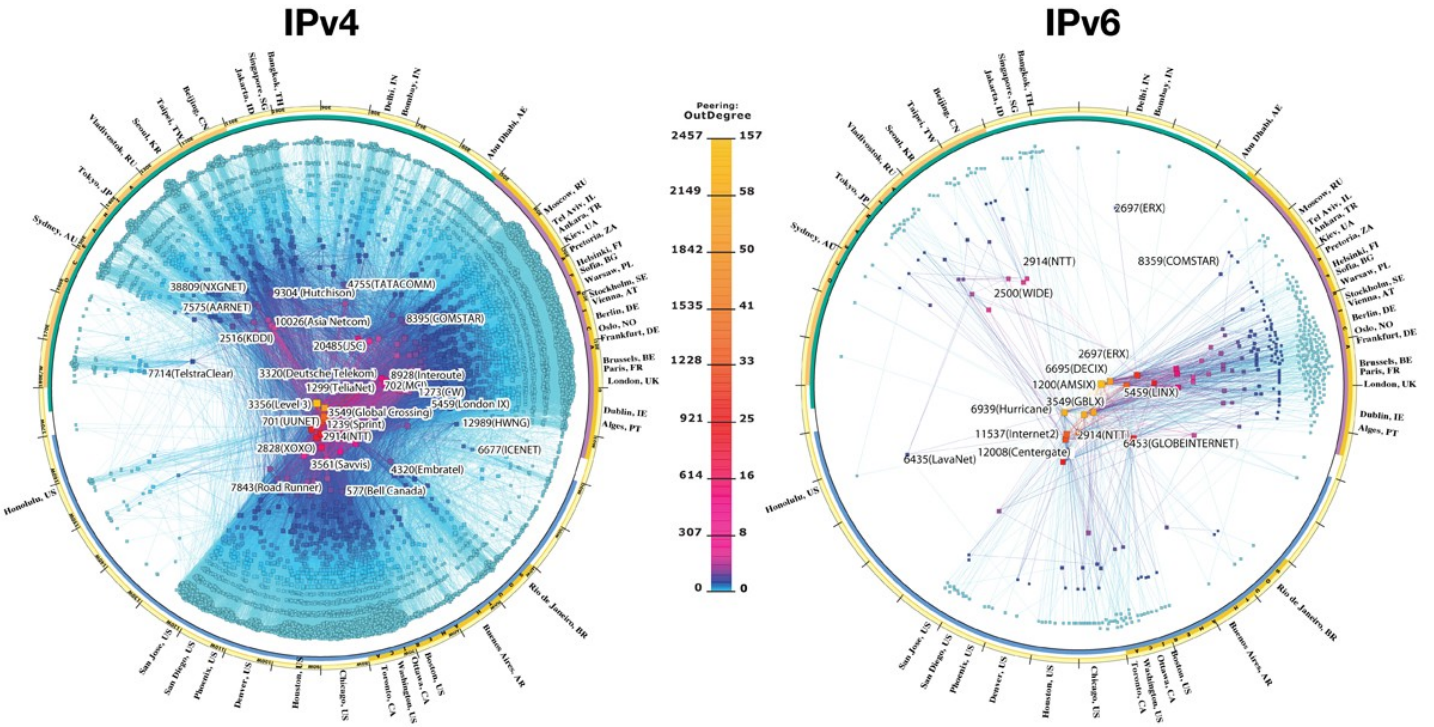
IPv6

IPv6: Die Grenzen von IPv4

- **Grunddesign etwa 30 Jahre alt**
  - Veraltetes Paketformat
  - Stark verbesserte Hardwareentwicklung
- **Adressraum erschöpft**
  - „Normales“ Internetwachstum nicht mehr normal
  - Neue Arten von vernetzten Geräten (Handys, intelligente Komponenten, „Internet of Things“, ...)
- **Unterstützung neuer Services „mühsam“**

IPv4 & IPv6
INTERNET TOPOLOGY MAP
JANUARY 2009

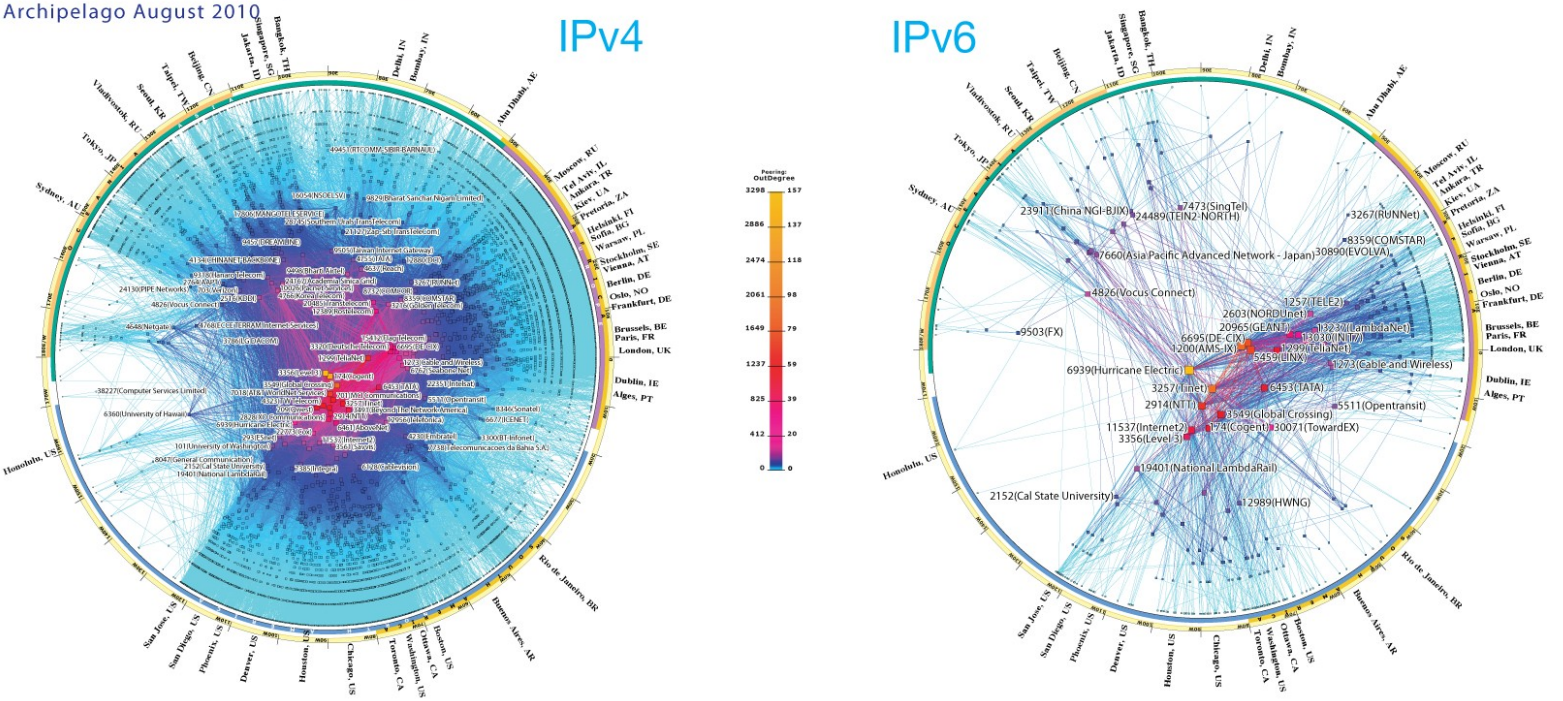
AS-level INTERNET GRAPH



Source: CAIDA
http://www.caida.org/research/topology/as_core_network/ipv6.xml

CAIDA's IPv4 & IPv6 AS Core AS-level INTERNET GRAPH

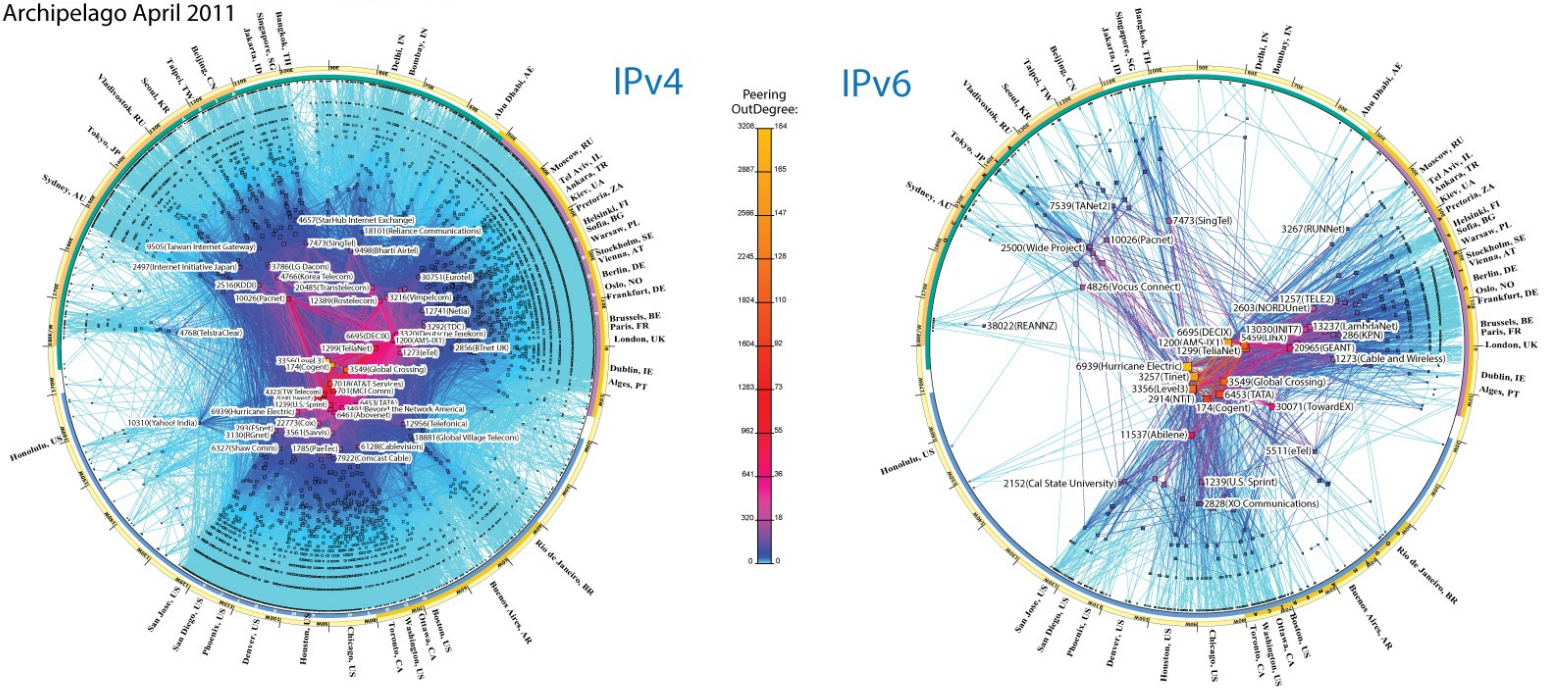
Archipelago August 2010



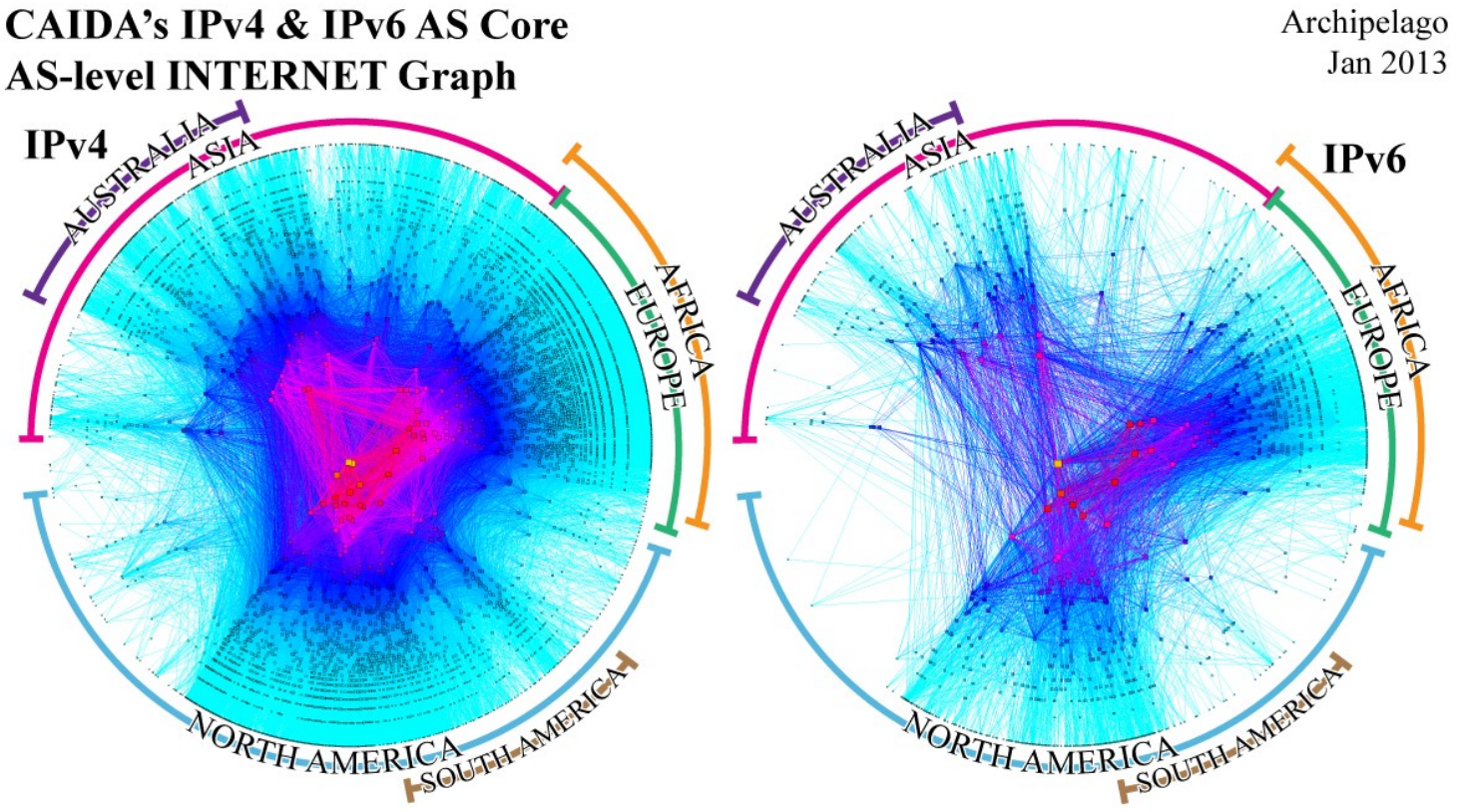
Source: CAIDA
http://www.caida.org/research/topology/as_core_network/ipv6.xml

CAIDA'S IPv4 & IPv6 AS Core AS-level INTERNET GRAPH

Archipelago April 2011



Source: CAIDA
http://www.caida.org/research/topology/as_core_network/ipv6.xml



Source: CAIDA
http://www.caida.org/research/topology/as_core_network/ipv6.xml

IETF WG IPng begann Anfang der 90er

- **Winter 1992:**
  7 Vorschläge zur Weiterentwicklung von IP
  - CNAT, IP Encaps, Nimrod, Simple CLNP, PIP, SIP, TP/IX
- **Herbst 1993:**
  Verschiedene Zusammschlüsse führen zu
  - „Simple Internet Protocol Plus“ (SIPP) und
  - „Common Architecture for the Internet“ CATNIP
- **Juli 1994: IPng Area Director empfiehlt Roadmap (RFC 1752) auf Basis von SIPP**

IETF WG IPng begann Anfang der 90er

- Dez. 1995: S. Deering, R. Hinden: „Internet Protocol, Version 6 (IPv6) Specification“ (RFC 1883, jetzt RFC 2460)
- 1999: Sub-TLAs erhältlich durch RIPE-NCC, APNIC sowie ARIN
- Mai 2007: ARIN ruft Internet Community zur Migration nach IPv6 auf

Gute Gründe für IPv6

- **Vereinfachtes Header-Format für eine schnellere Verarbeitung in den Routern**
  - Header soll Dienstgütemechanismen (Quality of Service, QoS) unterstützen
- **IPv6-Datagrammformat:**
  - Header fester Länge (40 Byte)
    - Dafür flexible Option-Header hintereinander
  - Verzicht auf Prüfsumme
  - keine Fragmentierung in den Routern
- **Verbesserte Multicast-, Anycast-, QoS und Mobile Services**

IPv6-Adressen

- **IPv6-Adressen sind 128-bit lang und variabel aufgebaut [RFC 1884, 4291]**
  - Automatische Adresskonfiguration
  - Globale Adresshierarchie von der Top-Level-Vergabe bis zur Interface-ID
  - 3 Bit Format-Präfix (FP) dient zur Identifikation des Adresstyps
- **„Aggregation-based Allocation“ zur Vereinfachung des weltweiten Routings**

IPv6-Adressen (2)

- **Standard Form:**
  - 8 x 16 bit Hexadezimal

    1080:0:FF:0:8:800:200C:417A
- **Verkürzte Form:**
  - Folgen von Nullen ersetzt durch „::“

    FF01:0:0:0:0:0:0:43 → FF01::43
- **IPv4 Kompatible Adressen:**
  - Beginnen mit „0:0:0:0:0:FFFF“

    0:0:0:0:0:FFFF:13.1.68.3 → ::FFFF:13.1.68.3
- **CIDR-Notation möglich (1080:645:FF::/48)**

Beispiel der FHTW Berlin

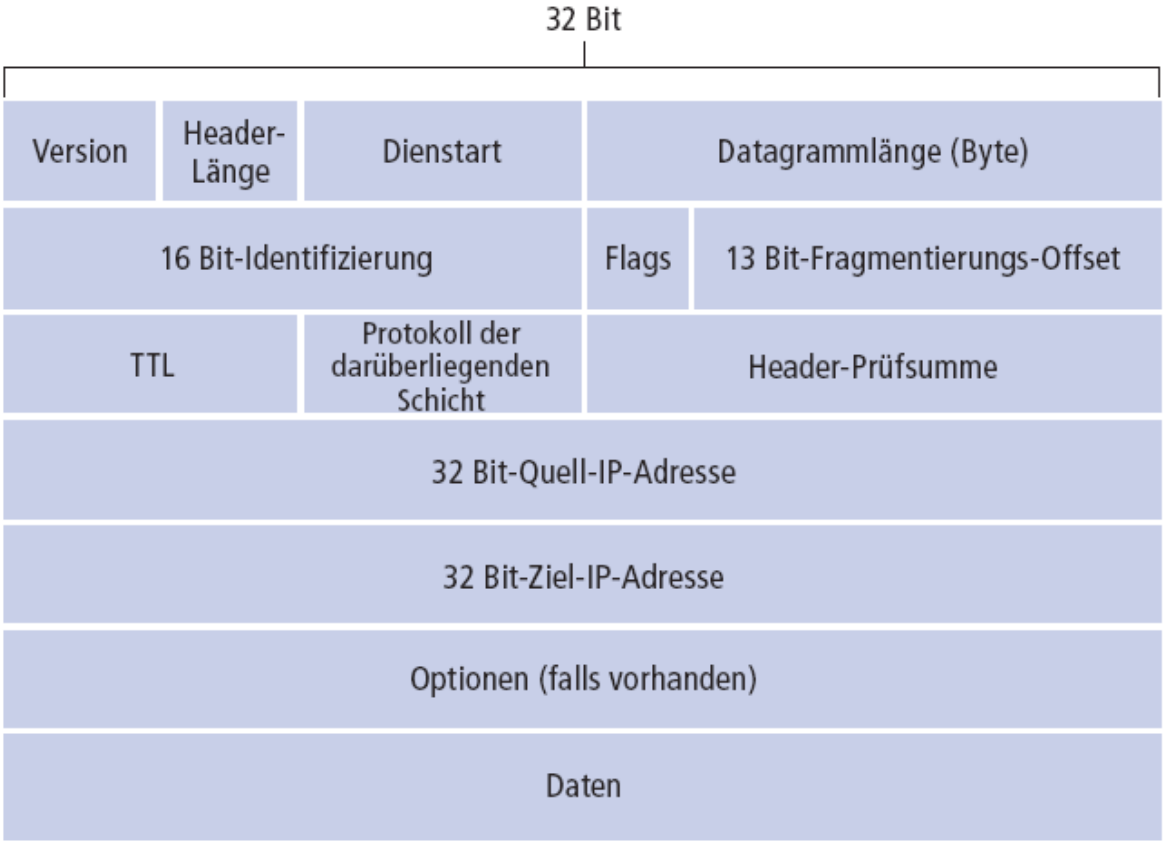
2001:: /16	Vorgabe
2001:0600:: /23	RIR Präfix (RIPE)
2001:0638:: /32	DFN Präfix
2001:0638:0801:: /48	FHTW-Netz
2001:0638:0801:0001:: /64	erstes FHTW Subnetz
2001:0638:0801:0001:0000:0000:0000:0001 /128	erster Rechner

Stateless Autokonfiguration

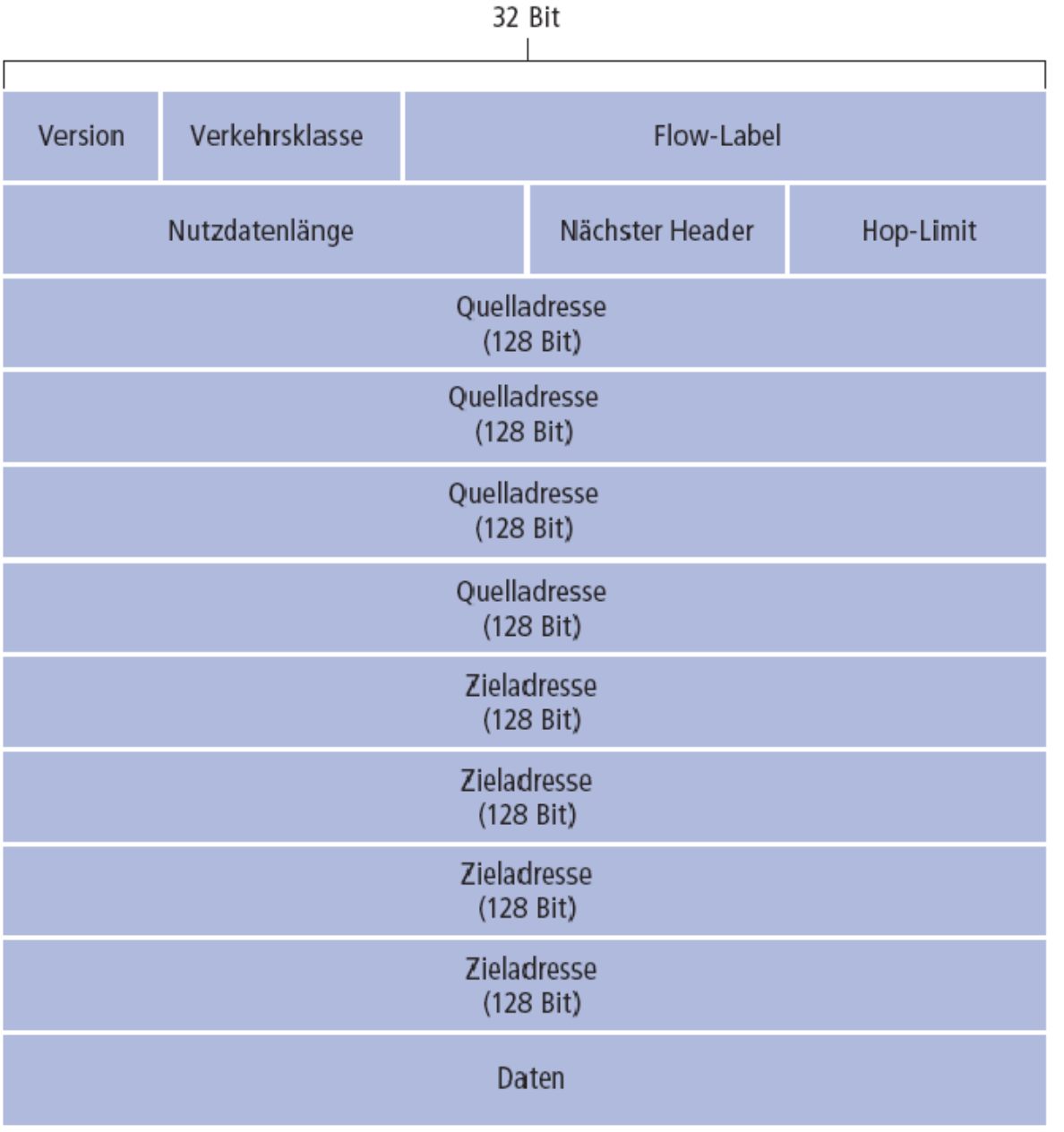
- **Interface erhält bei Aktivierung eine sogenannte „link-local“ Adresse**
  - z.B. aus der Hardwareadresse gebildet
- **Interface sendet eine „router solicitation“**
  - Router sendet Router-Advertisement (Präfix, Defaultgateway, …) als Reaktion
- **Das Interface bildet aus Präfix und „link-local“ Adresse eine globale Adresse.**
- **Verifikation der Eindeutigkeit durch ICMP „neighbor solicitation“**

… und wie sah noch mal der IPv4-Header aus?

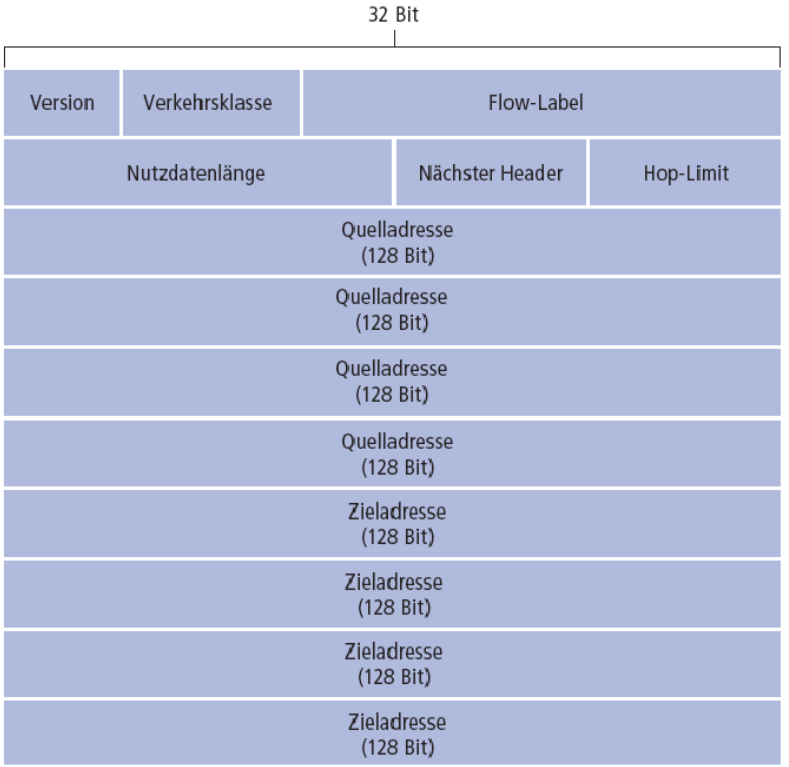
IPv4-Header gemäß [RFC 791]

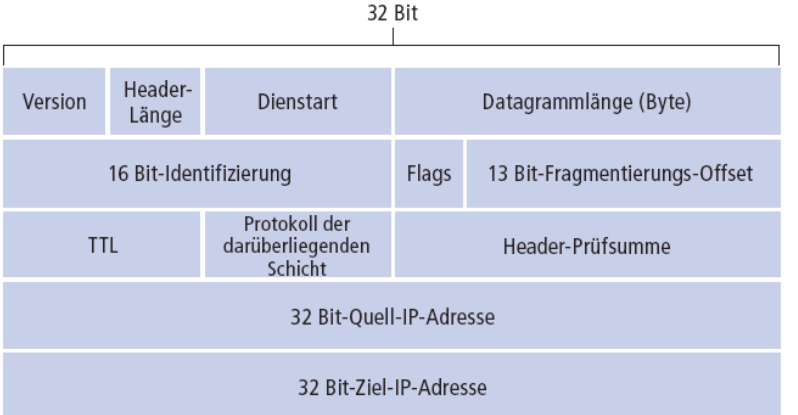


IPv6-Header


32 Bit
Version
Verkehrsklasse
Flow-Label
Nutzdatenlänge
Nächster Header
Hop-Limit
Quelladresse (128 Bit)
Quelladresse (128 Bit)
Quelladresse (128 Bit)
Quelladresse (128 Bit)
Zieladresse (128 Bit)
Zieladresse (128 Bit)
Zieladresse (128 Bit)
Zieladresse (128 Bit)
Daten

IPv4-Header vs. IPv6-Header





IPv6-Header

- **Verkehrsklasse (Traffic Class):**
  - Priorisierung von Datagrammen
- **Flow Label:**
  - Identifikation von zusammengehörigen Flüssen von Datagrammen (z.B. ein Voice-over-IP-Telefonat)
- **Nächster Header:**
  - An welches Protokoll sollen die Daten im Datenteil übergeben werden?
    - Zum Beispiel an ein Transportprotokoll wie TCP oder ein Sicherheitsprotokoll (AH, ESP, ...)

Weitere Veränderungen bei IPv6

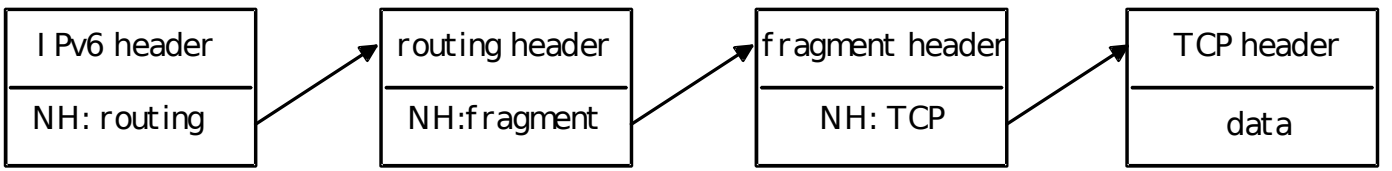
- **Optionen als separate Header, die auf den IP-Header folgen**
  - Werden durch das “Nächster Header”-Feld angezeigt
  - Einfachere Behandlung in Hosts und Routern
  - Keine feste Länge

Grundlegende Option-Header

- **Routing**
  - Erweiterte Routinginformationen
- **Fragmentation**
  - Fragmentierungs- bzw. Defragmentierungsinformationen
- **Authentication**
  - Authentizität und Integrität
- **Encapsulation**
  - für „Tunneling“, z.B. für vertrauliche Daten

Grundlegende Option-Header (2)

- **Hop-by-Hop Option**
  - Spezielle Optionen, die an jedem Router verarbeitet werden
- **Destination Option**
  - Informationen für den Empfänger-Host



Format der Option-Header

- **Erweiterter Optionsmechanismus**
  - Jeder Header verweist auf die eventuell nachfolgende Header –
    oder abschließend auf die Nutzdaten (Payload)
- **Keine Längenbeschränkung**
  - Padding auf 8 Oktetts
- **Keine Verarbeitung durch Router**
  - Ausnahme:
    Hop-by-Hop Option-Header

QoS: Verkehrsklasse + Flow Label

- **Verkehrsklasse**
  - analog zum IPv4 ToS-Feld
- **Flow Label**
  - 24-bit (zufällig bestimmt) markieren zusammenhängende Pakete
  - Definiert einen Zustand für 120 s Lebenszeit
  - Ziel: Beschleunigte und gleichförmige Behandlung der Paketströme durch Router
  - Headerinformationen für einen Flow einheitlich, erlaubt Caching im Router

Weitere Veränderungen bei IPv6

- **ICMPv6: neue Version von ICMP**
  - Zusätzliche Pakettypen
    - Packet Too Big
    - Parameter Problem
    - Unterstützung der neighbourhood solicitation
- **Funktionen zur Verwaltung von Multicast-Gruppen**

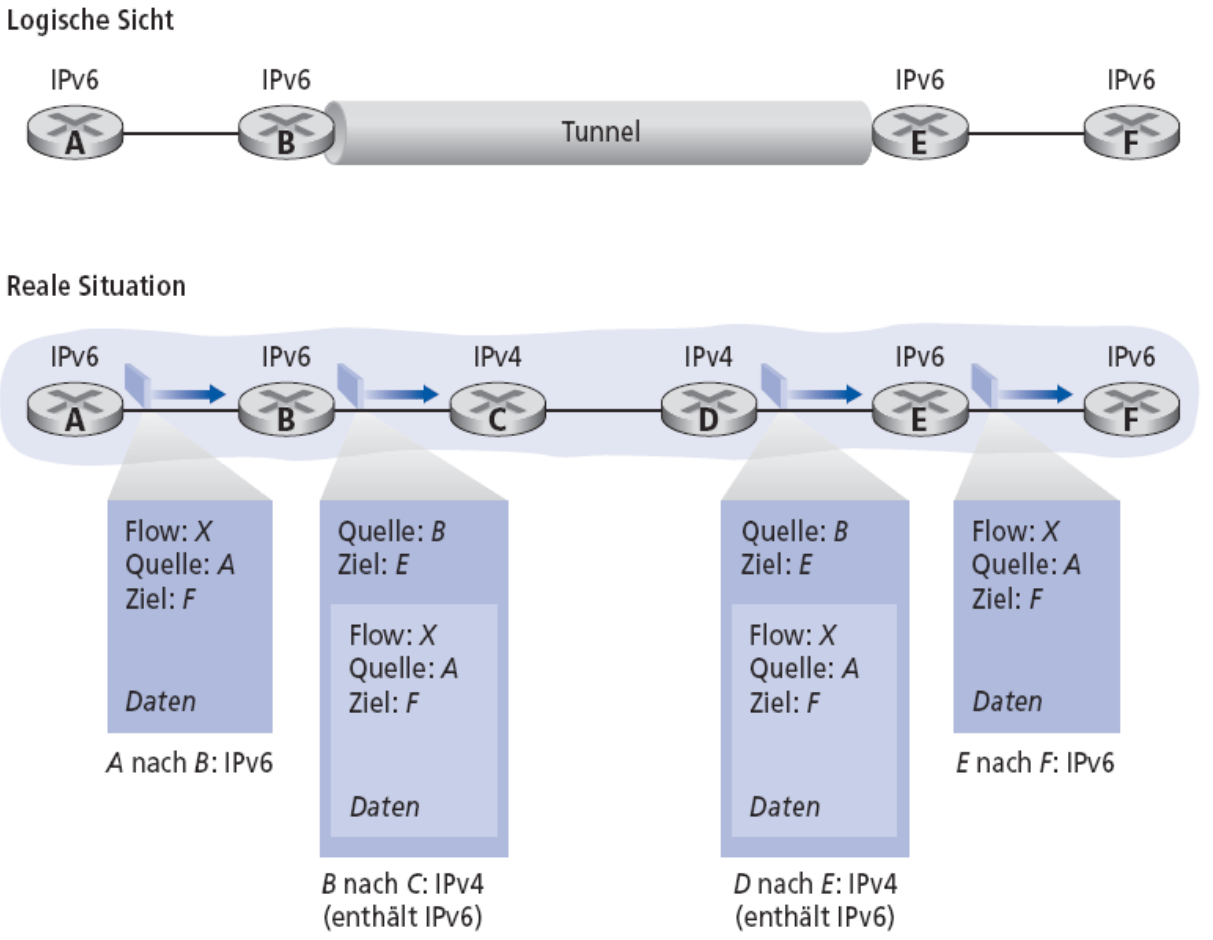
… und wann stellen wir jetzt alle um?

Übergang von IPv4 zu IPv6

- Es können nicht alle Router gleichzeitig umgestellt werden, also stellt sich die Frage anders:

  Wie kann ein Netzwerk funktionieren, in dem sowohl IPv4- als auch IPv6-Router vorhanden sind?

Tunneling von IPv6 durch IPv4

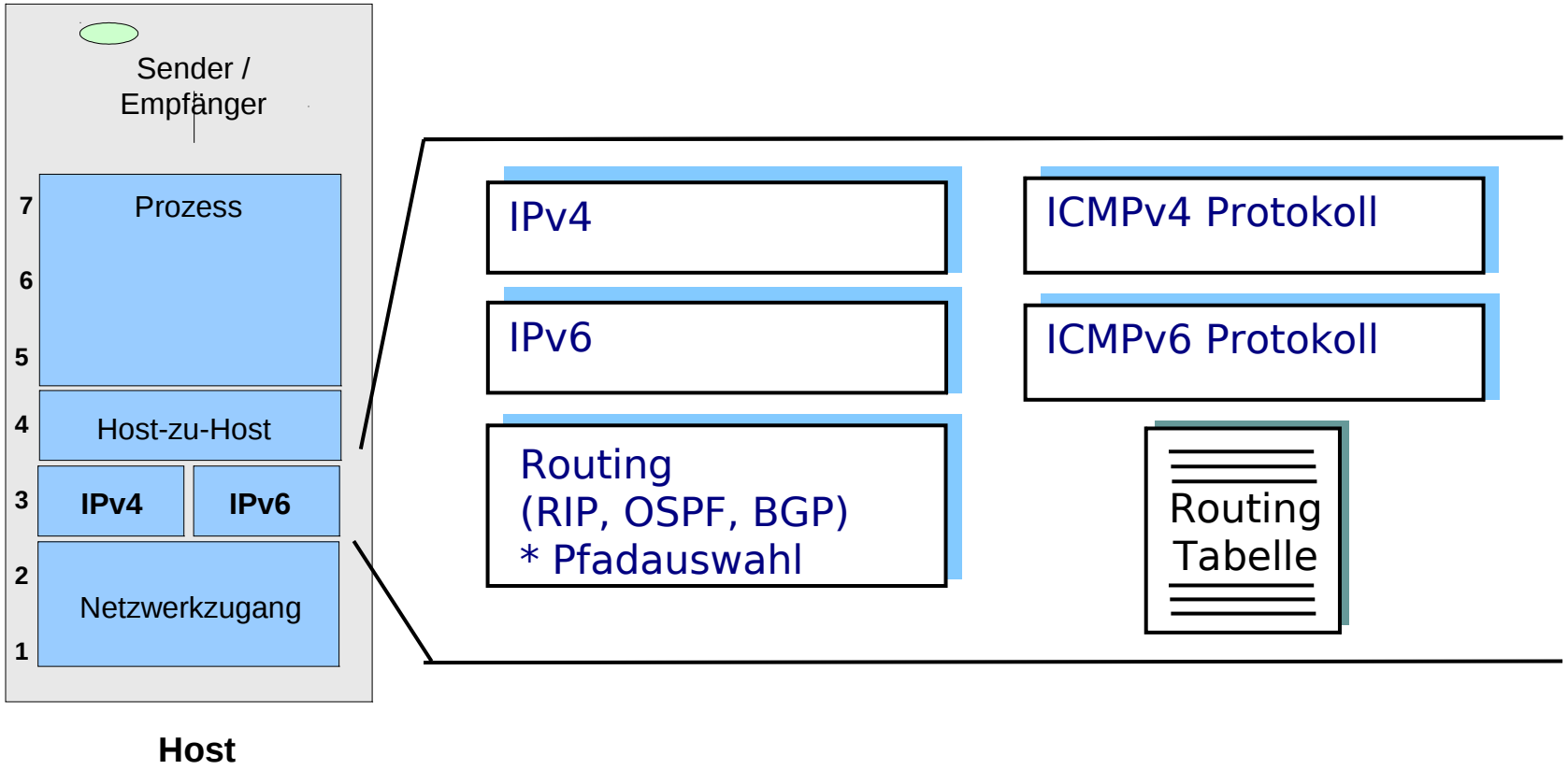


Übergang von IPv4 zu IPv6 (2)

Vielfältige Techniken zur Migration wurden konzipiert und implementiert nach folgenden Ansätzen:

- Tunnel wie gerade gesehen
- Protokollübersetzer (NATs), die IPv6-Geräte mit IPv4-Geräten sprechen lassen
- Dual-Stack-Techniken, die die Koexistenz von IPv4 und IPv6 auf demselben Gerät und in gleichen Netzen erlauben

Dual Stack aus IPv4 und IPv6 im Internet-Modell



Dual Stack

- **Multiprotokoll-Ansatz**
  - Beim Aktivieren von IPv6 kann der IPv4-Stack einfach weiterbetrieben werden
  - Sicherheitsprobleme, wenn Benutzer dies gar nicht mehr merken
- **Hosts behalten ihre Adressen**
  - 13.1.68.3
  - ::FFFF:13.1.68.3

Dual Stack (2)

- **Anwendungen bzw. eingebundene Bibliotheken wählen die IP-Version aus:**
  - Bei der Kontaktaufnahme in Abhängigkeit von der DNS-Antwort
  - Bei der Beantwortung in Abhängigkeit vom den eingegangen Paketen
- **Der Dual-Stack Betrieb kann prinzipiell unbeschränkt fortgeführt werden und erlaubt die schrittweise Portierung der Applikationen**

RCF 3493:
Versionsübergreifende Adress-API

Benutzung der Koexistenz-API

`sockaddr_in` und `sockaddr_in6` sind inkompatible Datenstrukturen!

Lösung:

- Eine Indirektionsstruktur `addrinfo` erlaubt den transparenten Zugriff auf die (versionsabhängigen) `sockaddr*` Strukturen
- Diese werden automatisch gefüllt durch getaddrinfo:
  - liefert als Ergebnis einen Pointer auf eine verkettete Liste von `addrinfo` Adressstrukturen

Benutzung der Koexistenz-API (2)

`sockaddr_in` und `sockaddr_in6` sind inkompatible Datenstrukturen!

Lösung (Fortsetzung):

- Um einen erfolgreichen Kommunikationsweg zu finden, müssen die Adressen der Liste ausprobiert werden.
- Zusätzlich ist `sockaddr_storage` eine versionsübergreifende Datenstruktur, die gemäß der gewünschten Version gecasted werden kann.

Versionsneutrale Programmierung

Client-seitig:

- **Der Client muss eine IP-Version wählen, die der Server versteht. Hierzu nutzt er**
  - DNS (`getaddrinfo`)
  - ggfs. Anwendereingaben bzw.
  - probiert die Versionen durch
- **In der Regel wählt der Client also zwischen mehreren Adressoptionen des Servers**

Versionsneutrale Programmierung

Server-seitig:

- **Der Server muss in allen IP-Versionen ansprechbar sein.**
  - Hierfür benötigt er eine Adressstruktur, die für alle Adressen geeignet ist (`sockaddr_storage`).
  - Gegenwärtig ist diese identisch mit `sockaddr_in6` und IPv4 wird eingebettet.
- **Der Server antwortet in derselben Version, in der er angesprochen wurde.**

Kontakt

Prof. Dr. Klaus-Peter Kossakowski

Email: klaus-peter.kossakowski@haw-hamburg.de

Mobil: +49 171 5767010

https://users.informatik.haw-hamburg.de/~kpk/