



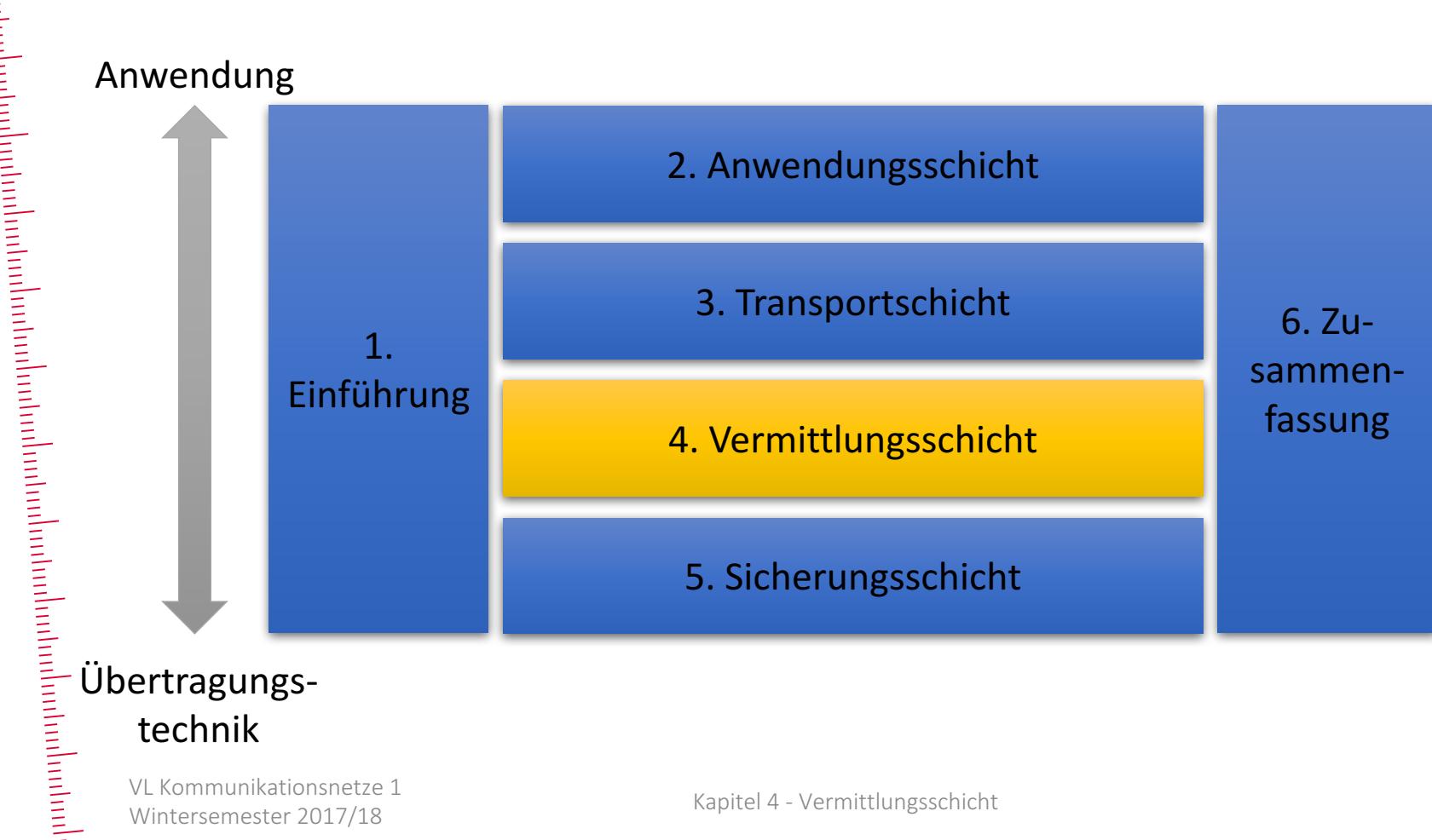
Kapitel 4  
**Die Vermittlungsschicht**

Vorlesung Kommunikationsnetze 1  
Wintersemester 2017/18

Oliver P. Waldhorst

(Basierend auf Materialien von J. Kurose und K. Ross © 1996-2017)

# Gliederung der Vorlesung



# Ziele dieses Kapitels

Verstehen der Prinzipien der Vermittlungsschicht

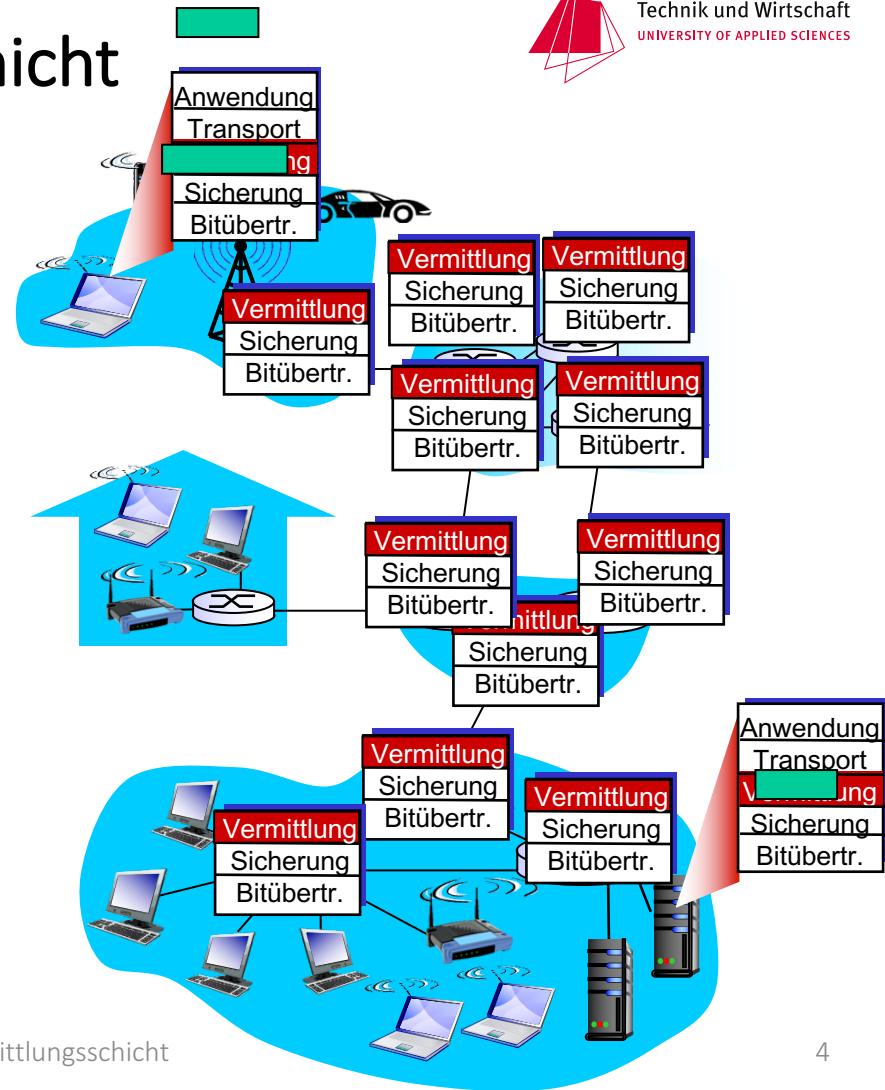
- Weiterleitung (Forwarding) und Wegbestimmung (Routing)
- Router: Aufbau, Pufferung, Weiterleitung
- Adressierung
- Routing-Algorithmen

... und deren Implementierung im Internet

- IPv4, IPv6, DHCP, OSPF, RIP, BGP, ICMP

# Die Internet-Vermittlungsschicht

- Bereitgestellter Dienst: Übertragung von Segmenten von sendenden zu empfangenden Endsystemen
  - Keine Garantien bzgl.. Verlusten, Reihenfolge, Übertragungszeit!
- Beim Sender: (Transportschicht-) Segmente in (Vermittlungsschicht-) **Datagramme** verpacken
- Beim Empfänger: Segmente entpacken, an Transportschicht übergeben
- Vermittlungsschichtprotokolle laufen auf jedem Endsystem und Router!
- Router verarbeiten die Header aller Datagramme, die durch sie laufen



# Schlüsselfunktionen der Vermittlungsschicht

## Schlüsselfunktionen:

- Weiterleitung (Forwarding): Bewegen von Paketen im Router von Eingangsschnittstelle (**Port**) zur richtigen Ausgangsschnittstelle
  - Weiterleitungstabelle (Forwarding Table)
- Wegbestimmung (Routing): Bestimmung des gesamten Wegs, den ein Paket von Quelle zum Ziel nehmen soll
  - Dezentraler Routing-Algorithmus bestimmt die Weiterleitungstabellen

## Analogie: Eine Autoreise

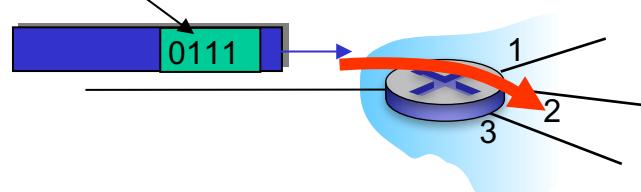
- Weiterleitung: Richtiges passieren eines einzelnen Autobahnkreuzes
- Wegbestimmung: Planung des Weges vom Start zum Ziel

# Daten- und Steuerungsebene

## Datenebene (Data plane)

- Lokale Funktion auf jedem Router
- Entscheidet, wie ein Datagramm von einem Eingangsport zu einem Ausgangsport weitergeleitet wird
- Weiterleitungsfunktion

Header-Einträge in ankommenden Paketen

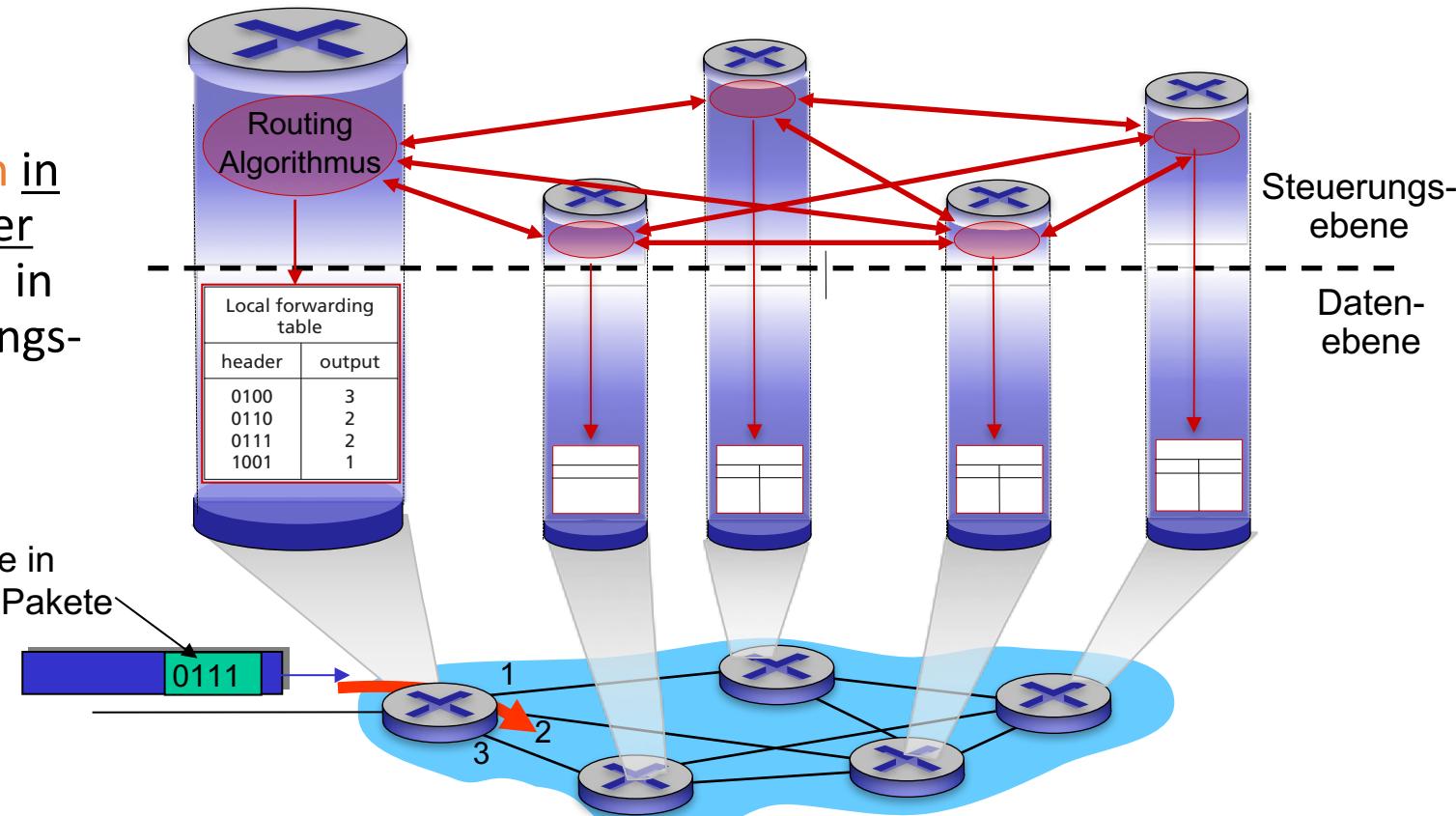


## Steuerungsebene (Control plane)

- Netzweite Logik
- Entscheidet welchen Weg ein Datagramm von Quelle zum Ziel nimmt
- Zwei Ansätze
  - Traditionelle **Routing-Algorithmen**, in Routern implementiert
  - **Software-definierte Netze (Software Defined Networking, SDN)**: Zentral implementiert in entfernten Servern (hier nicht weiter betrachtet)

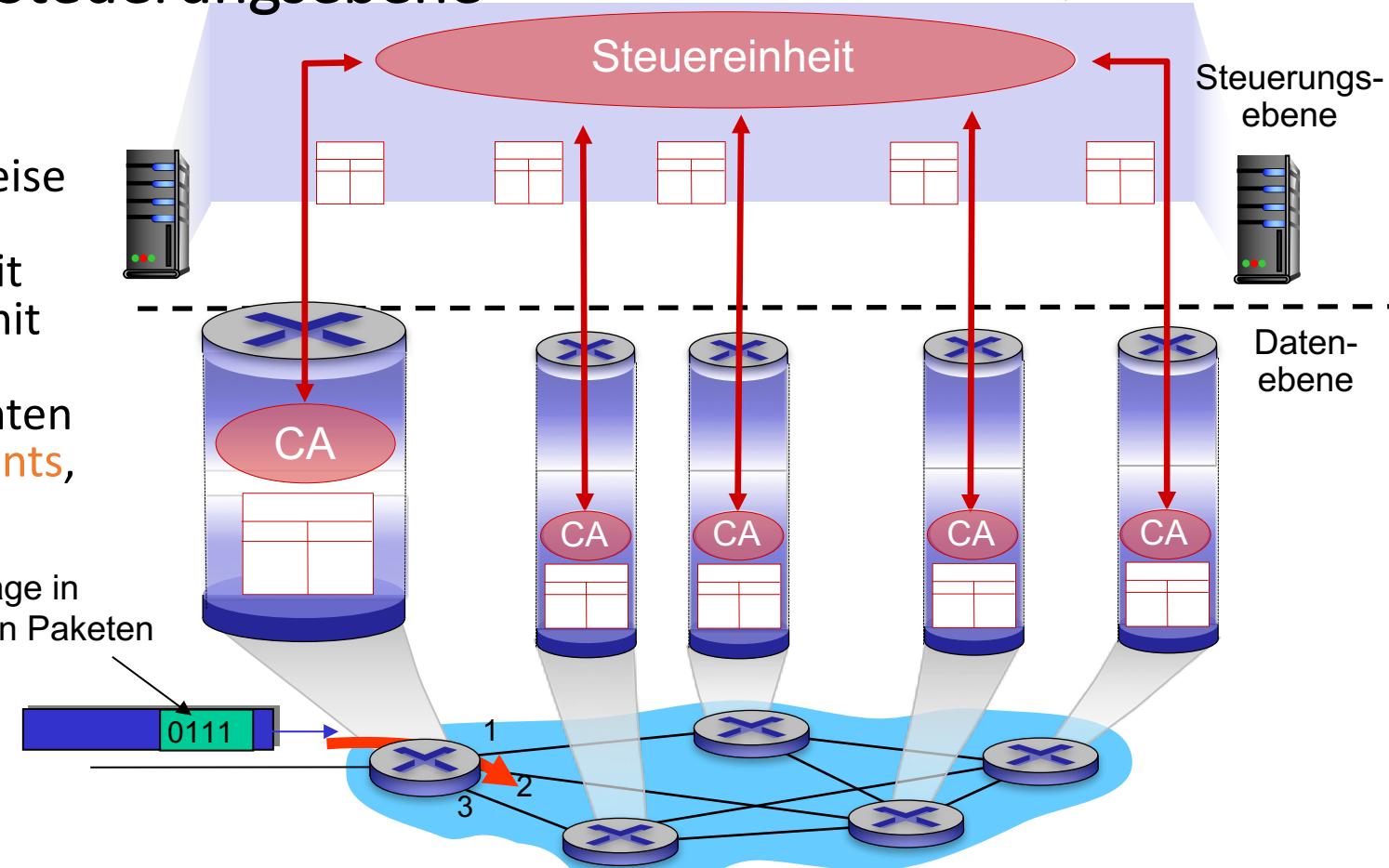
# Steuerungsebene pro Router

Einzelne  
Routing-  
Algorithmen in  
jedem Router  
interagieren in  
der Steuerungs-  
ebene

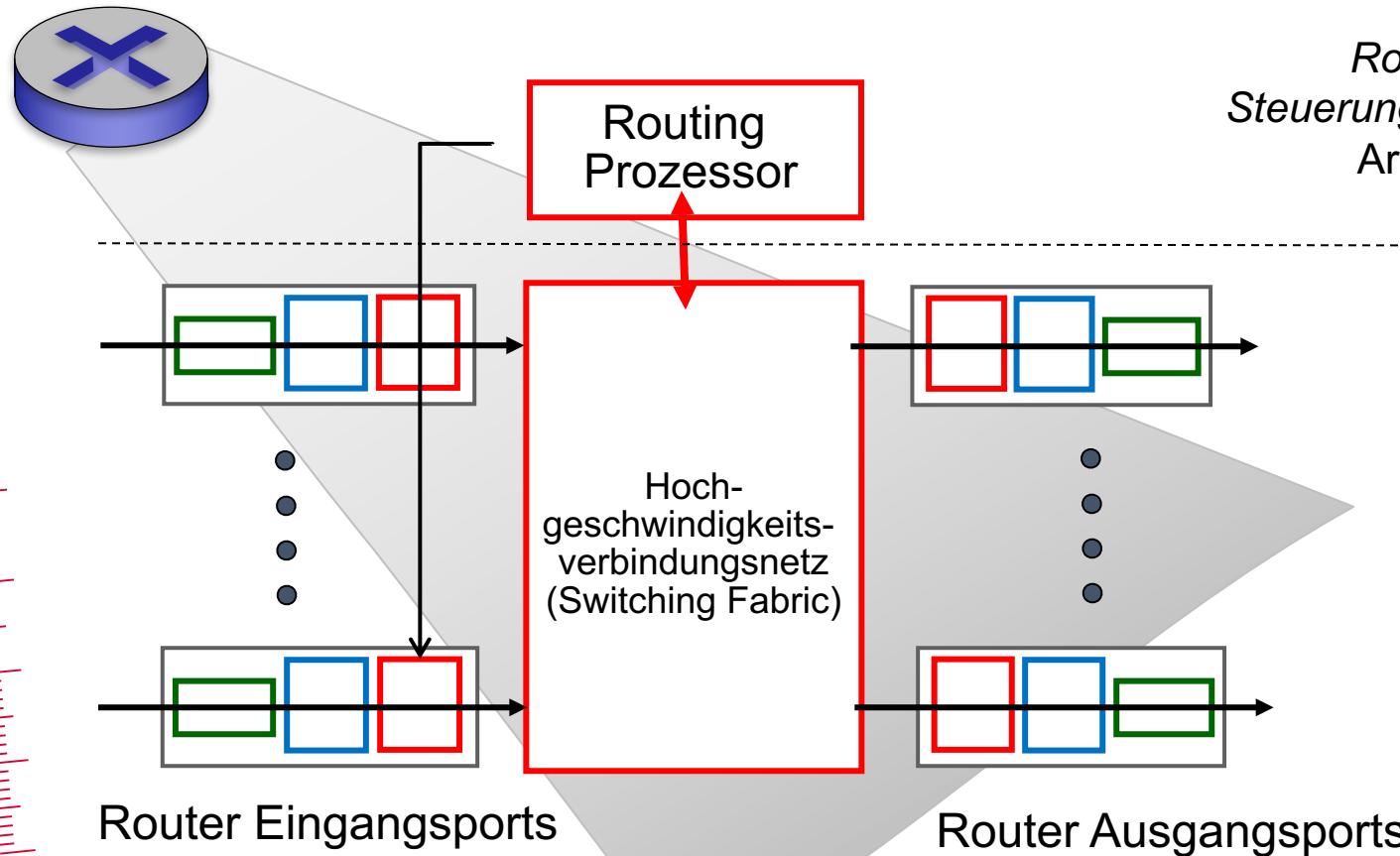


# Zentrale Steuerungsebene

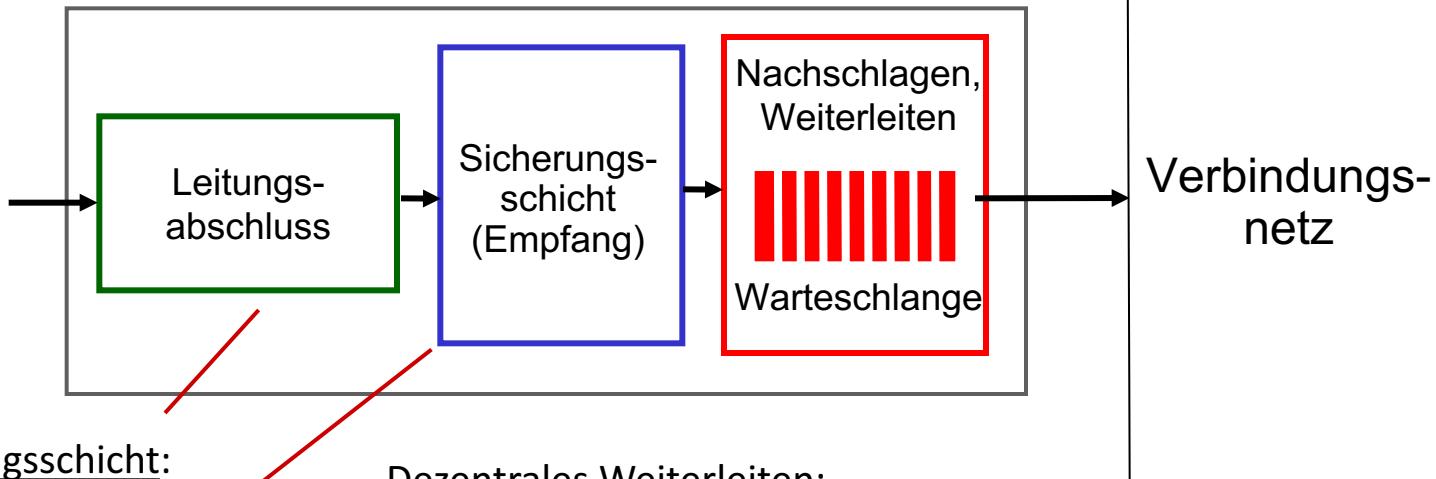
Eine einzige (typischerweise entfernte) Steuereinheit interagiert mit lokalen Kontrollagenten (Control Agents, CA)



# High-Lever Router-Architektur



# Eingangsports - Funktionen



Bitübertragungsschicht:  
Übertragung von Bits

Sicherungsschicht:  
Z.B. Ethernet,  
s. Kapitel 5

## Dezentrales Weiterleiten:

- Jeder Eingangsport nutzt Header-Felder zum Nachschlagen des Ausgangsports in der Weiterleitungstabelle im Speicher
  - Basierend auf Zieladresse (**Destination Based Forwarding**) oder zusätzlich weiteren Header-Feldern (**Generalized Forwarding**)
- Ziel: Verarbeitung in Übertragungsgeschwindigkeit („Line speed“)
- Queuing: Wenn Datagramme schneller als Verbindungsnetz speichern in Warteschlange

# Weiterleitung anhand der Zieladresse (Destination-based Forwarding)

*Weiterleitungstabelle*

Zieladressbereich	Ausgangsport
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

Was passiert,  
wenn die  
Adressbereiche  
sich nicht so  
schön  
aufteilen?

# Weiterleitung nach längstem Präfix (Longest Prefix Matching)

Bei der Suche in der Weiterleitungstabelle die Adresse auswählen, die das längste übereinstimmende Präfix mit der Zieladresse hat

Zieladressbereich	Ausgangsport
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
sonst	3



Übung:

ZA: 11001000 00010111 00010110 10100001

Welcher Ausgangsport?

ZA: 11001000 00010111 00011000 10101010

Welcher Ausgangsport?

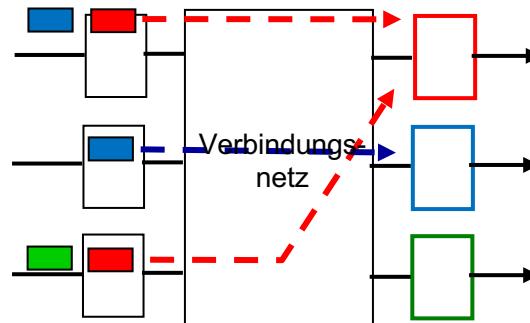


# Weiterleitung nach längstem Präfix (Longest Prefix Matching)

- Warum man längste Präfixe verwendet sehen wir später in diesem Kapitel
- Longest Prefix Matching wird häufig mit Hilfe von **Ternary Content Addressable Memory (TCAM)** realisiert
  - Erlaubt die Adressierung mit mit „*Don't Care*“ oder „X“-Bits zusätzlich zu „0“- und „1“-Bits in der Adresse
  - Eintrag der Weiterleitungstabelle wird in konstanter Zeit zurückgeliefert
  - Cisco Catalyst Router können im TCAM bis zu 1M Einträge der Weiterleitungstabelle halten

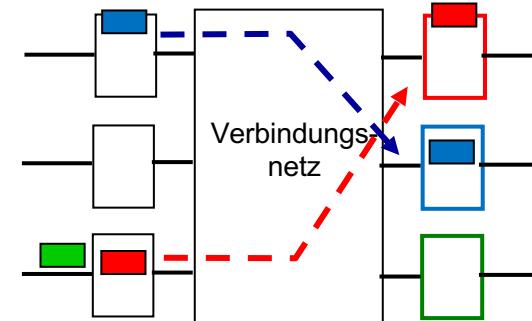
# Warteschlangen an den Eingangsports

- Wenn Verbindungsnetz langsamer ist als die Eingangsports müssen Datagramme zwischengespeichert werden
  - Verzögerungen und Verluste wegen Überläufen!
- **Head-of-Line (HOL)-Blockade:** Gespeichertes Datagramm an Position 1 verhindert Weiterleitung anderer Datagramme auf hinteren Positionen



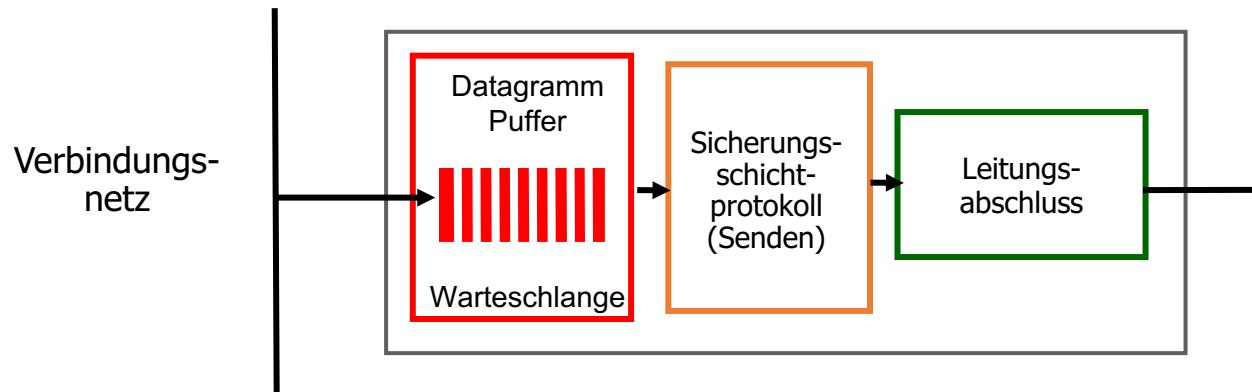
Wettbewerb um Ausgangsport:  
Nur ein rotes Datagramm  
kann übertragen werden.

*Das untere rote Datagramm wird blockiert*



Eine Übertragungszeit  
später: HOL-Blockade  
für das grüne Packet

# Ausgangsports



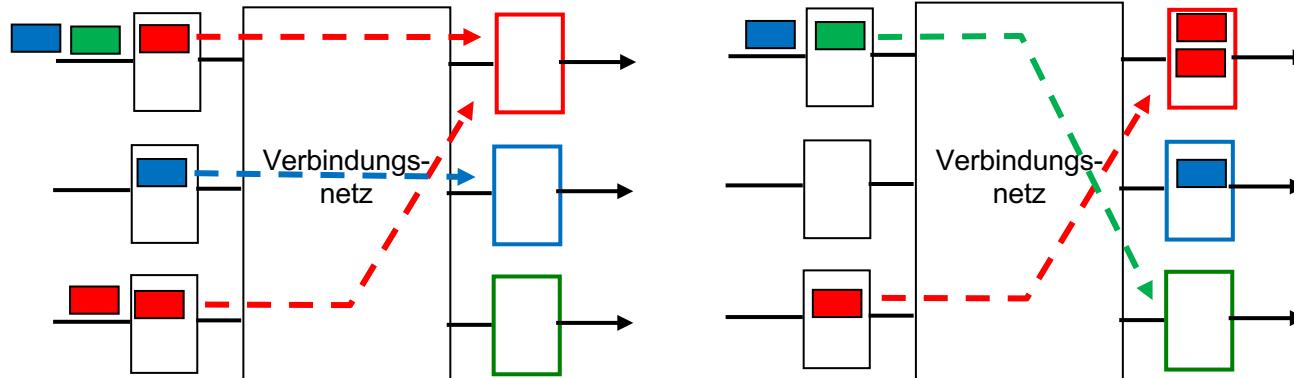
Warteschlange notwendig wenn Pakete schneller vom Verbindungsnetz ankommen als Übertragungsrate des Port

- Verlust von Datagrammen bei Überlast!

Scheduling-Verfahren für zur Auswahl von übertragenen Datagrammen

- Wer hat höchste Priorität? Netzneutralität?

# Warteschlangen an den Ausgangsport



Zum Zeitpunkt t  
mehrere Pakete zu  
einem Ausgangsport

Eine  
Übertragungszeit  
später

- Puffern wenn Ankunftsrate vom Verbindungsnetz die Senderate des Ausgangs übersteigt
- Verzögerungen und Verluste durch Pufferüberlauf!



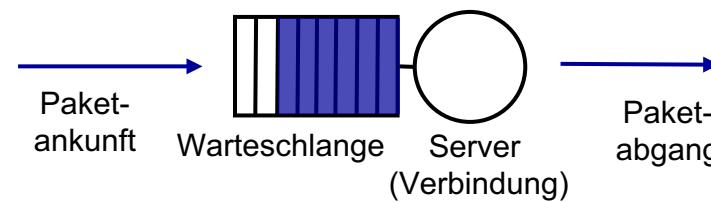
# Dimensionierung der Puffer

- Daumenregel: Durchschnittliche Puffergröße entspricht „typischer“ RTT (z.B. 250ms) mal Verbindungskapazität  $C$  [RFC 3439]
  - Z.B.  $C = 10 \text{ Gbps}$  Verbindung: 2,5 Gb Puffer
- Aktuelle Empfehlung: Mit  $N$  TCP-Verbindungen Puffergröße

$$\frac{RTT \cdot C}{\sqrt{N}}$$

# Scheduling-Verfahren

Auswahl des nächsten Datagramms, dass über eine Verbindung am Ausgangsport gesendet wird



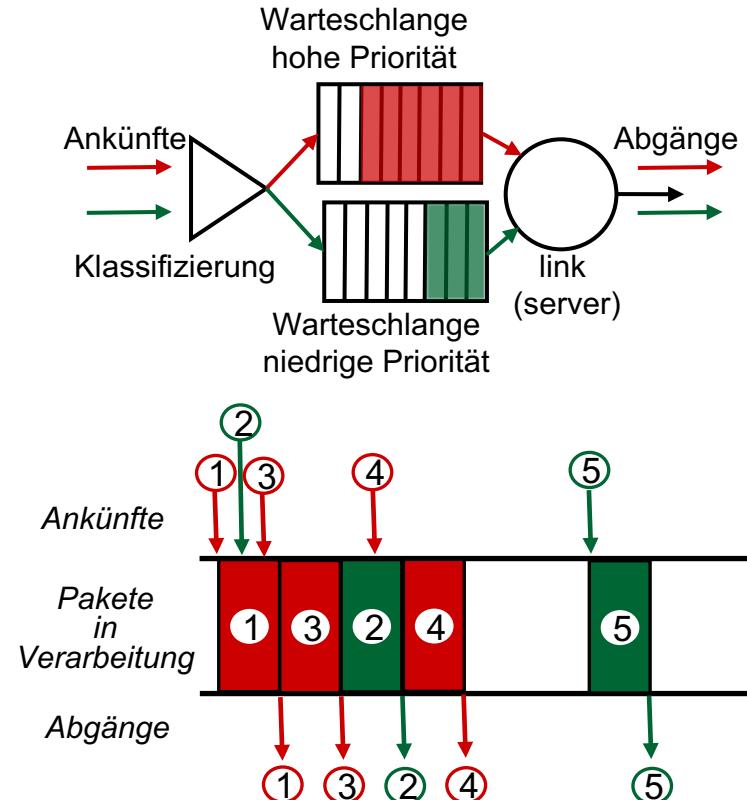
**FIFO (First-in-first-out) Scheduling:** In Ankunftsreihenfolge senden

- Beispiel aus dem Leben?
- Verwerfungsstrategie: Was passiert, wenn Datagramm bei voller Warteschlange eintrifft?
  - Tail Drop: Das eintreffende Paket wird verworfen
  - Priorität: Verwerfen anhand von Prioritäten
  - Zufällig: Ein zufälliges Paket wird verworfen

# Priority Scheduling

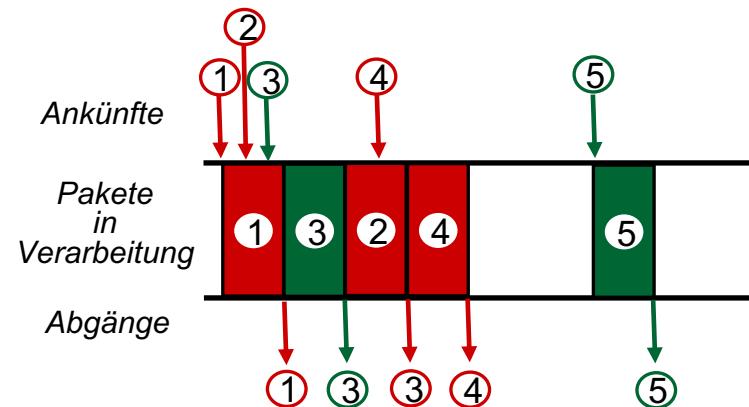
Sende Datagramm mit höchster Priorität

- Mehrere Klassen mit verschiedenen Prioritäten
  - Klassen können von Paket-Markierungen oder IP-Adressen von Quelle / Ziel, Port-Nummern usw. abhängen
  - Beispiel aus dem Leben?



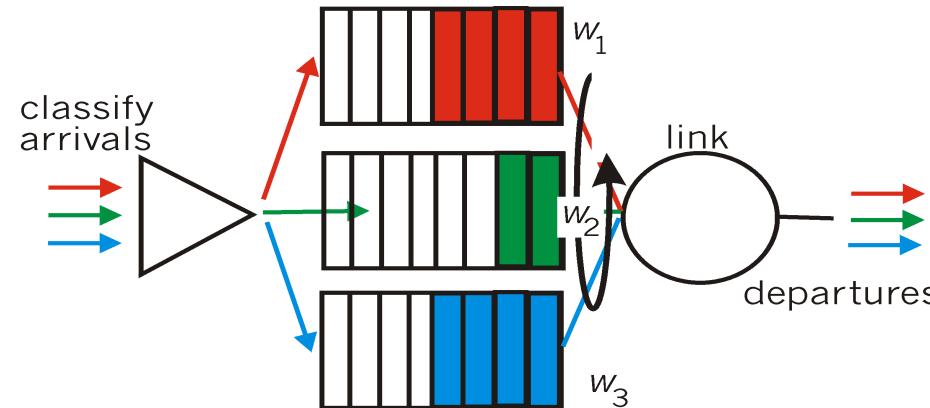
# Round-Robin Scheduling

- Mehrere Klassen
- Zyklische Betrachtung aller Klassen, jeweils ein komplettes Datagramm aus jeder Klasse senden (falls vorhanden)
- Beispiel aus dem Leben?

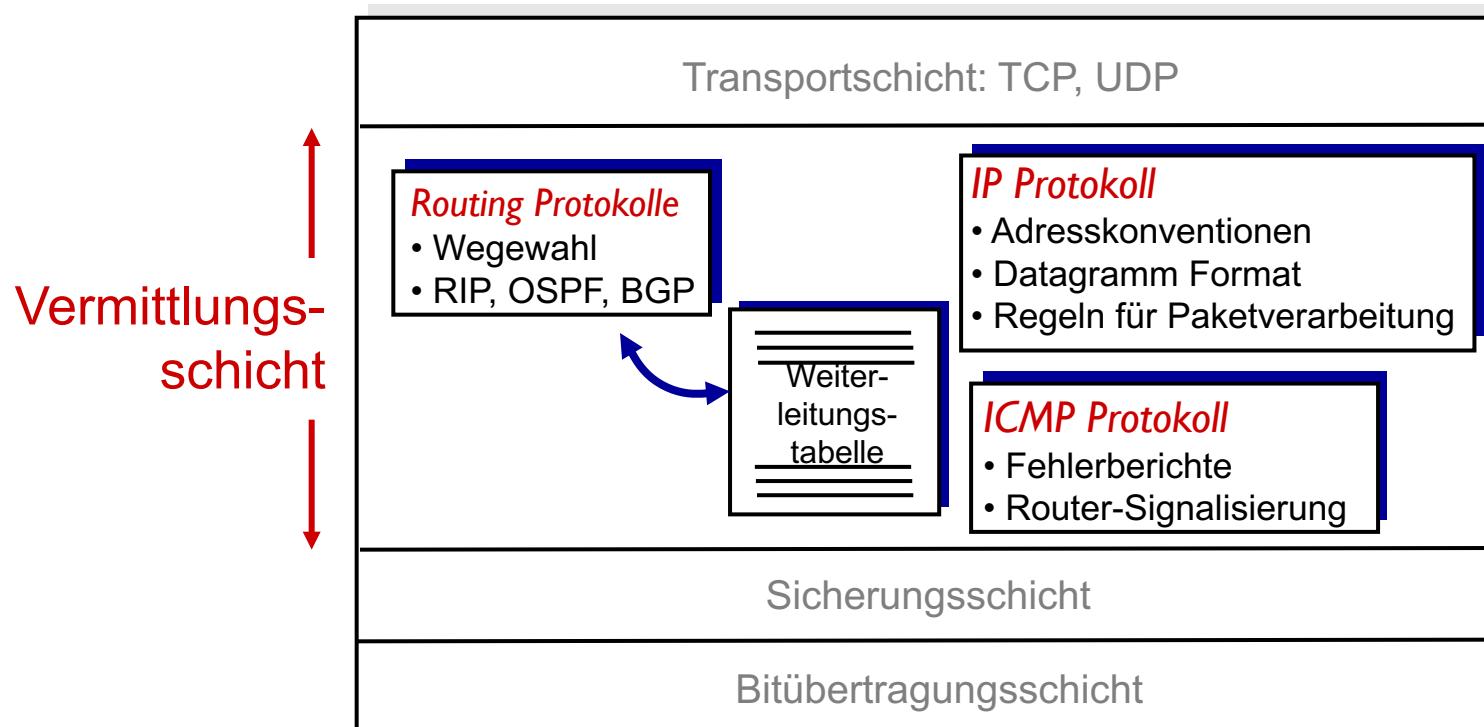


# Weighted Fair Queuing (WFQ)

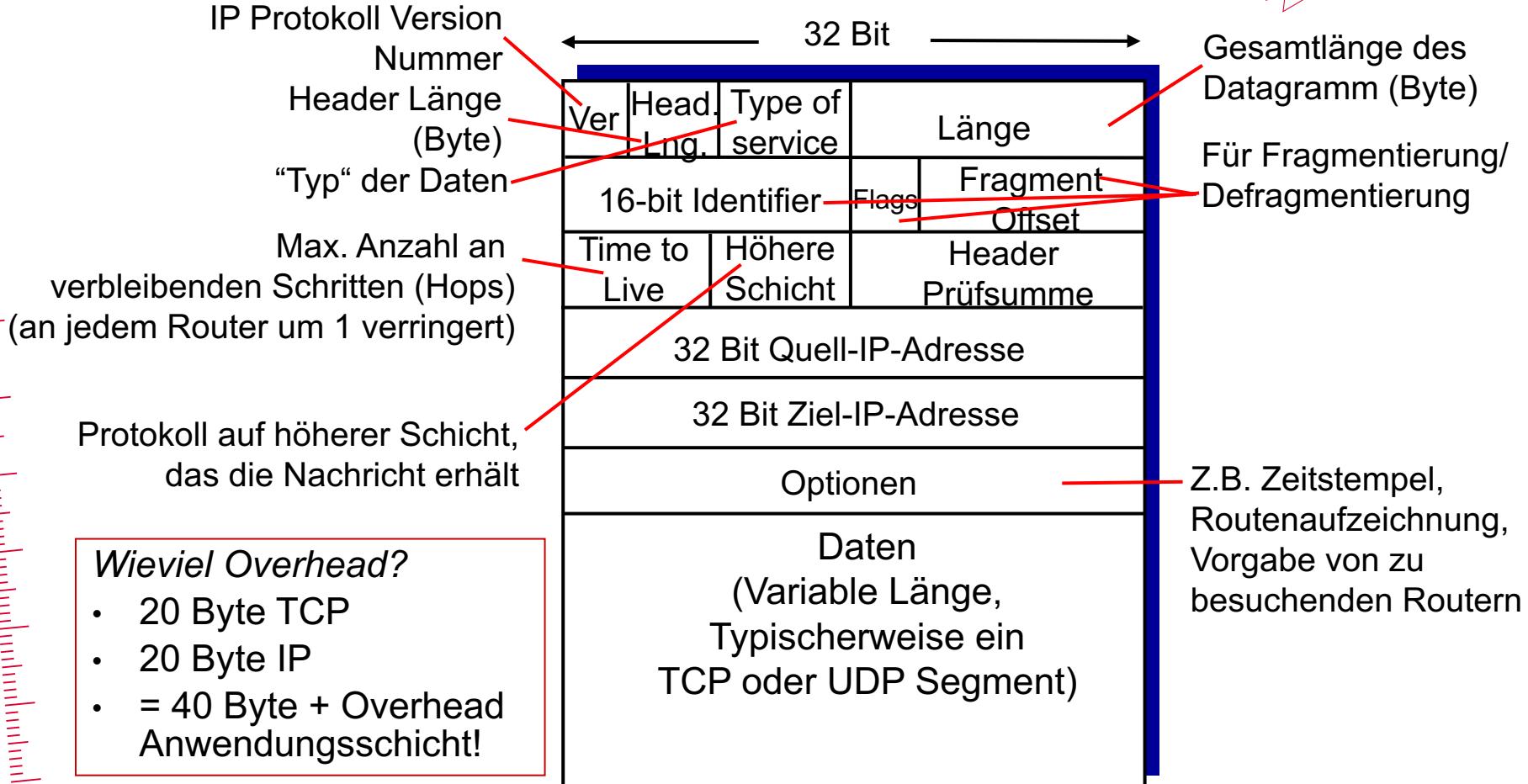
- Verallgemeinerung von Round-Robin
- Jede Klasse bekommt eine Gewichtung, die die Senderate bestimmt
- Beispiel aus dem Leben?



# Die Internet-Vermittlungsschicht

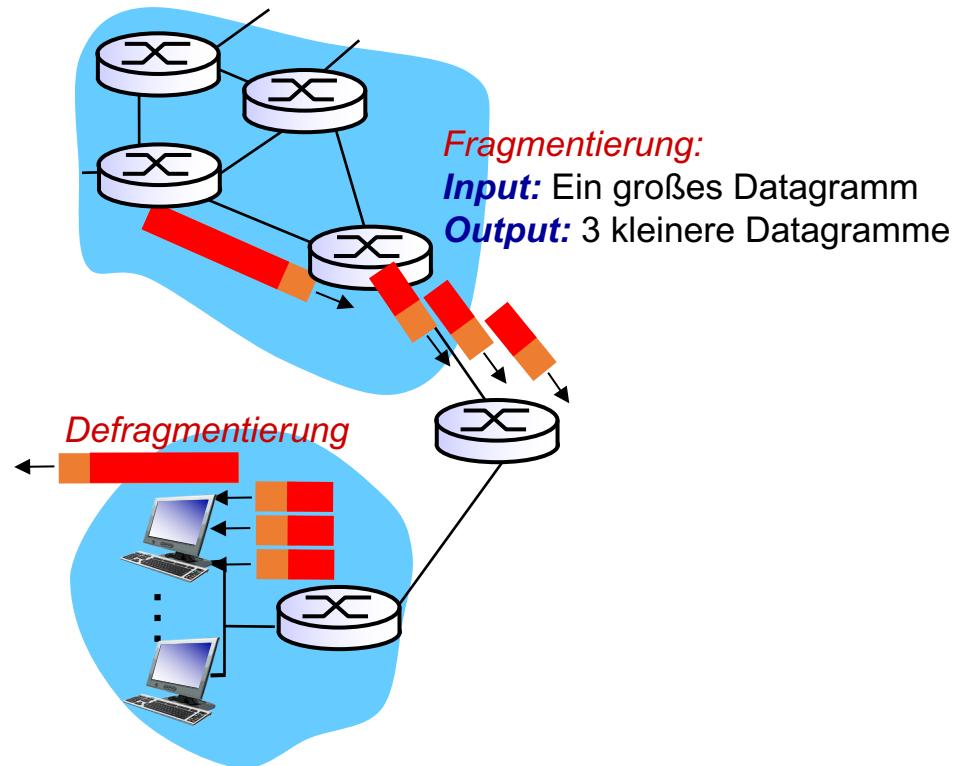


# IPv4 Datagramm Format



# IPv4 Fragmentierung/Defragmentierung

- Netzverbindungen haben begrenzte Größe für einen Sicherungsschichtrahmen (**Maximum Transfer Unit, MTU**)
  - Verschiedene Verbindungstypen, verschiedene MTUs
- Große IP Datagramme müssen während der Übertragung geteilt („**fragmentiert**“) werden
  - Aus einem Datagramm werden mehrere
  - Zusammensetzen („**Defragmentierung**“) nur beim Empfänger
- IP Header Bits zur Identifizierung und zum Ordnen zusammengehörender Fragmente genutzt



# IPv4 Fragmentierung/Defragmentierung

## [RFC 791]

### Beispiel:

- 4000 Byte Datagramm
- MTU = 1500 Byte

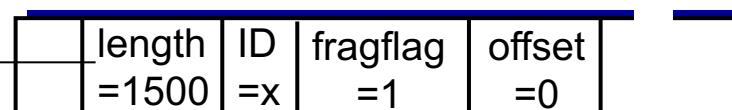


Länge Daten:

$$\text{length} - \text{Header} = 3980 \text{ Byte}$$

*Aus einem großen Datagramm werden mehrere kleinere*

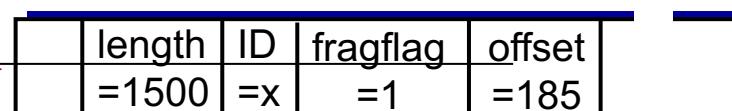
1480 Bytes in Datenfeld



Länge Daten:

$$\text{length} - \text{Header} = 1480 \text{ Byte}$$

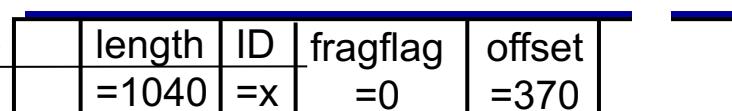
Offset =  
 $1480/8$



Länge Daten:

$$\text{length} - \text{Header} = 1480 \text{ Byte}$$

*fragflag* in letztem Fragment von ursprünglichem Datagramm übernommen; in allen übrigen Fragmenten *fragflag* = 1



Länge Daten:

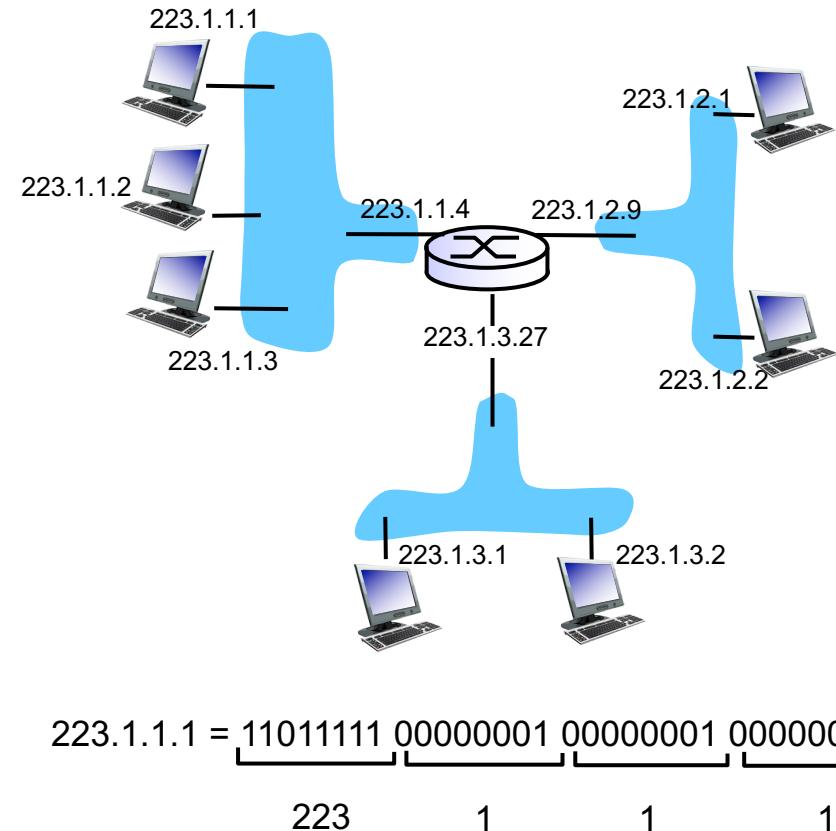
$$\text{length} - \text{Header} = 1020 \text{ Byte}$$

Gesamt-Datenmenge in Fragmenten:

$$1480 \text{ Byte} + 1480 \text{ Byte} + 1020 \text{ Byte} = 3980 \text{ Byte}$$

# Einführung IPv4-Adressierung

- **IP-Adresse:** 32-Bit Identifier für Endsystem- oder Router-Schnittstelle
- **Schnittstelle (Interface):** Verbindung zwischen Endsystem/Router und physischer Verbindung
  - Router haben typischerweise mehrere Schnittstellen
  - Endsysteme haben typischerweise ein oder zwei Schnittstellen (z.B. Ethernet, WLAN)
- Jede Schnittstelle hat zugeordnete IP-Adressen



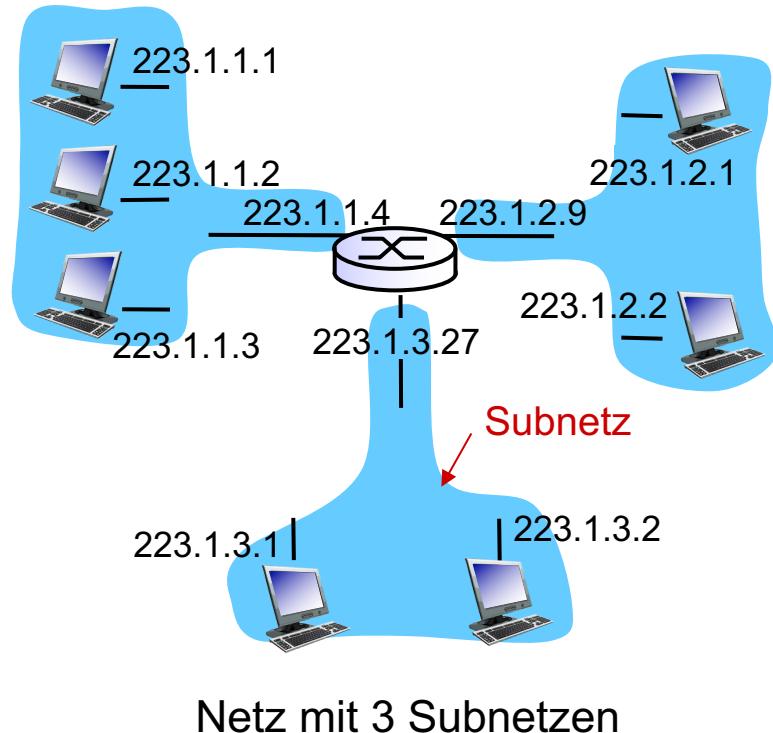
# Subnetze [RFC 950]

## Struktur der IP-Adresse

- **Subnetzteil:** Höherwertige Bits
- **Host-Teil:** Niederwertige Bits

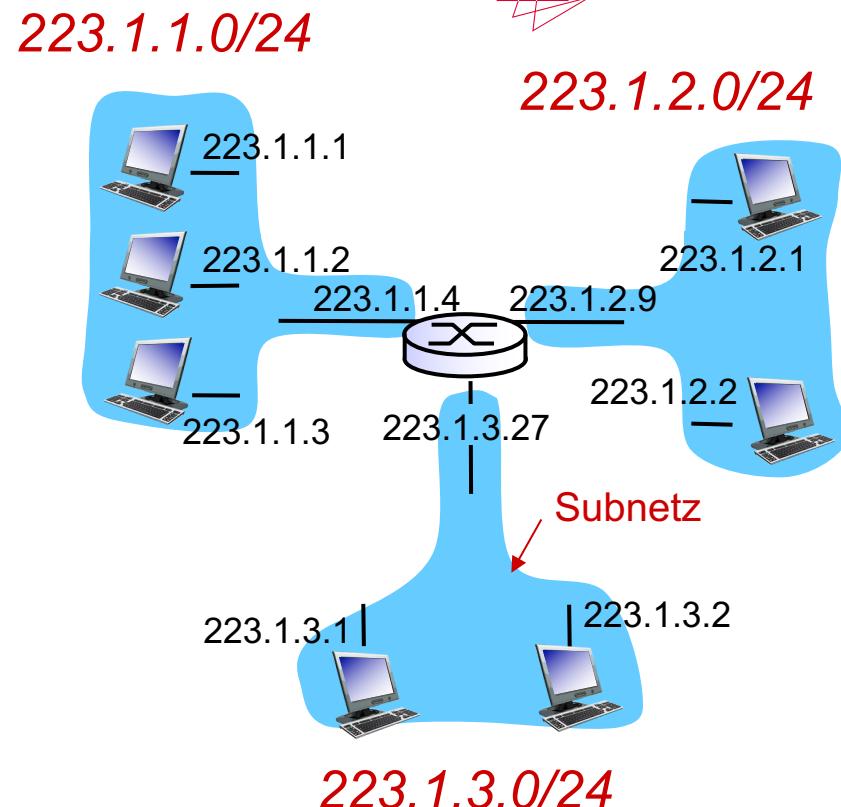
## Was ist ein Subnetz?

- Schnittstellen mit dem selben Subnetzteil in der IP-Adresse
- Können physisch miteinander kommunizieren ohne dazwischenliegenden Router



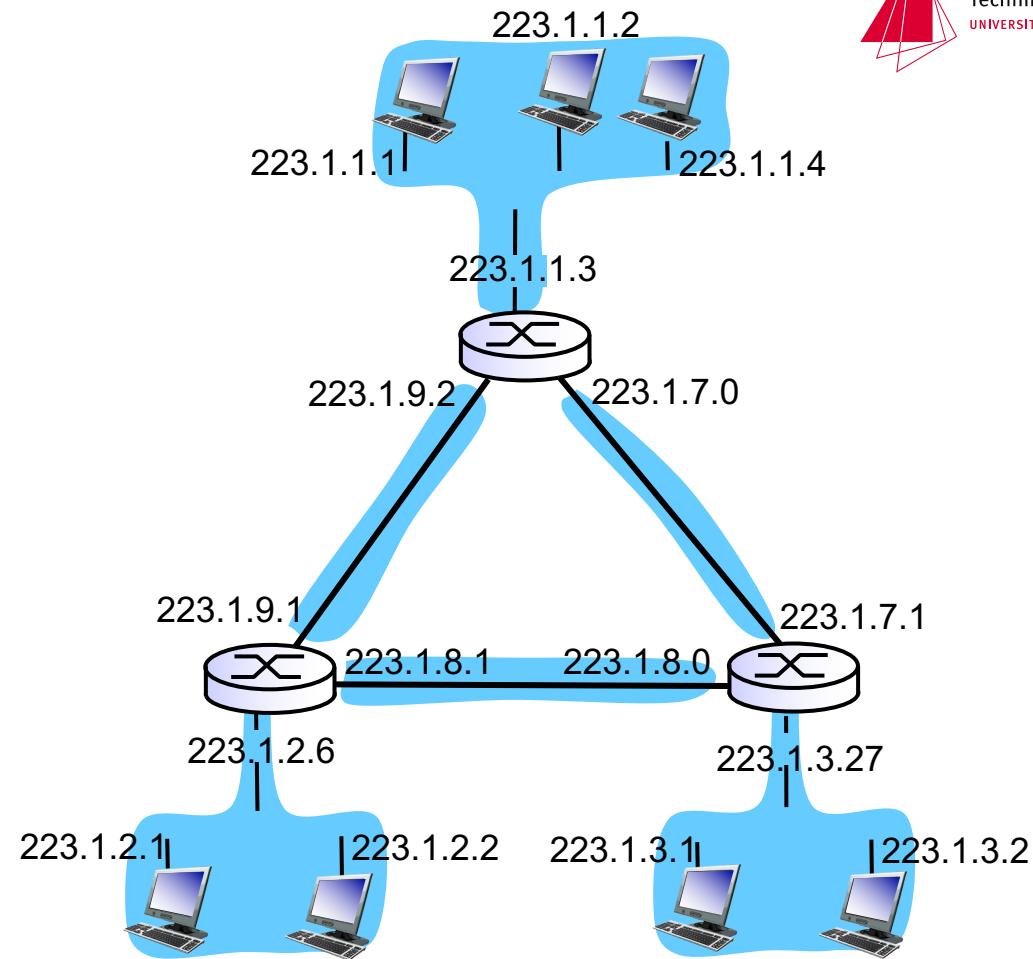
# Subnetz-Bestimmung

- Zur Bestimmung des Subnetz jede Schnittstelle von ihrem Endsystem / Router trennen
    - so entstehen „Inseln“ von isolierten Netzen
  - Jedes isolierte Netz ist ein Subnetz



## Subnetzmaske: /24

# ! Subnetze - wie viele?



# IP-Adressierung: CIDR

## [RFC 1518, RFC 1519, RFC 4632]

### Classless InterDomain Routing (CIDR)

- Subnetzteil der IP-Adresse hat beliebige Länge
- Adressformat: a.b.c.d/x, mit x ist Anzahl der Bits in Subnetzteil der Adresse



# Woher kommt die IP-Adresse?

Entweder vom Administrator statisch eingestellt in einer Konfigurationsdatei

- Windows: Systemeinstellungen → Netzwerk → Konfiguration → TCP/IP → Eigenschaften
- Mac: Systemeinstellungen → Netzwerk
- UNIX: /etc/rc.config

Oder **Dynamic Host Configuration Protocol (DHCP)**

- Dynamische Zuweisung einer IP-Adresse von einem Server
- „Plug-and-Play“

# Dynamic Host Configuration Protocol (DHCP)

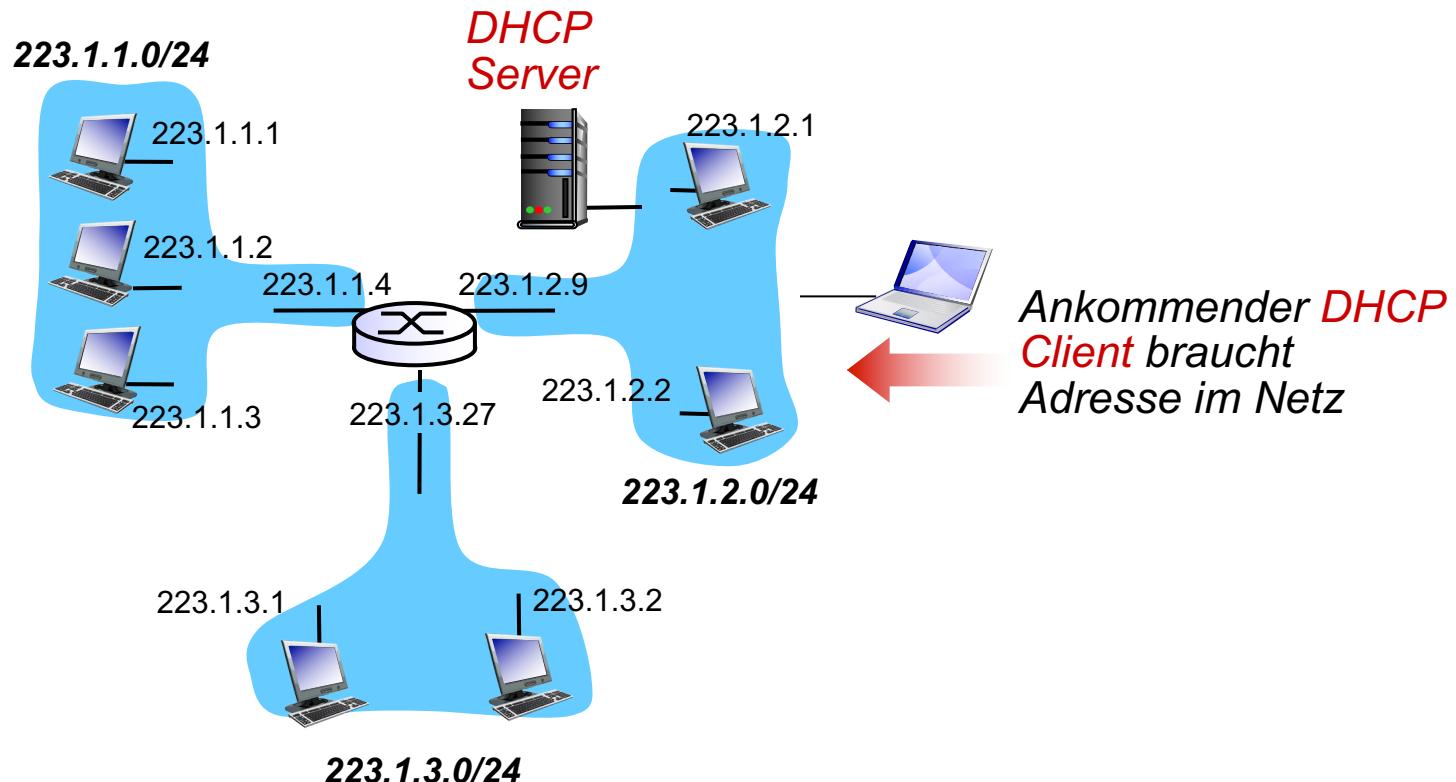
Ziel: Ermögliche Endgerät bei Verbindung mit Netz dynamisch IP-Adresse von einem Server zu beziehen

- Endgerät kann die Gültigkeit einer verwendeten Adresse verlängern
- Wiederverwertung von Adressen
- Unterstützung mobiler Nutzer eines Netzes

## DHCP Überblick:

- Endgerät sendet **DHCP Discover** Nachricht (optional)
- DHCP-Server antwortet mit **DHCP Offer** Nachricht (optional)
- Endgerät fordert IP-Adresse an: **DHCP Request** Nachricht
- DHCP-Server sendet Adresse: **DHCP ACK** Nachricht

# DHCP-Beispiel



# DHCP-Beispiel

DHCP Server: 223.1.2.5



Broadcast: Ich bin  
DHCP-Server! Hier  
ist eine IP-Adresse  
für Dich

DHCP Discover

src : 0.0.0.0, 68  
dest.: 255.255.255.255,67  
yiaddr: 0.0.0.0  
transaction ID: 654

Neues  
Endgerät



Broadcast: Gibt es  
hier einen DHCP-  
Server?

DHCP Offer

src: 223.1.2.5, 67  
dest: 255.255.255.255, 68  
yiaddr: 223.1.2.4  
transaction ID: 654  
lifetime: 3600 secs

DHCP Request

src: 0.0.0.0, 68  
dest:: 255.255.255.255, 67  
yiaddr: 223.1.2.4  
transaction ID: 655  
lifetime: 3600 secs

Broadcast: OK.  
Ich nehme diese  
IP-Adresse!

DHCP ACK

src: 223.1.2.5, 67  
dest: 255.255.255.255, 68  
yiaddr: 223.1.2.4  
transaction ID: 655  
lifetime: 3600 secs

Broadcast: OK.  
Du hast die IP-  
Adresse!



# DHCP: Mehr als nur IP-Adressen

DHCP kann mehr als nur IP-Adressen im Subnetz zuweisen

- Adresse des ersten Routers für den Client
- Name und IP-Adresse des DNS-Servers
- Netzmaske (Anzeigen des Subnetz- und Hostteils der IP-Adresse)



# DHCP Wireshark-Ausgabe

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**

**3 = Router; 6 = Domain Name Server**

44 = NetBIOS over TCP/IP Name Server

.....

## Request

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."

## Reply

# Woher kommt die IP-Adresse (zum Zweiten)?

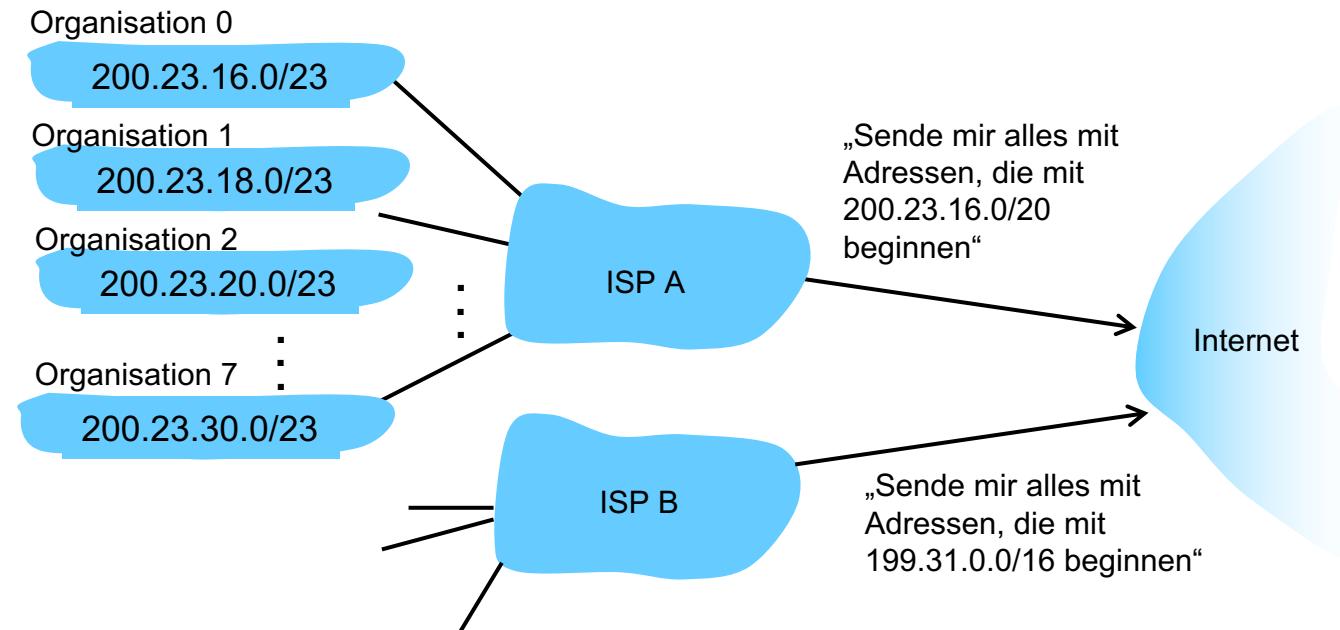
Wie kommt das Netz an den Subnetzteil der IP-Adresse?

- Vom ISP wird ein Teil von dessen Adressraum zugewiesen

ISP Adressraum	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	<u>00000000</u>	200.23.16.0/20
Organisation 0	11001000	00010111	00010000	00000000	200.23.16.0/23
Organisation 1	<u>11001000</u>	<u>00010111</u>	<u>00010010</u>	<u>00000000</u>	200.23.18.0/23
Organisation 2	11001000	00010111	00010100	00000000	200.23.20.0/23
...	.....	.....	....	....	....
Organisation 7	<u>11001000</u>	<u>00010111</u>	<u>00011110</u>	<u>00000000</u>	200.23.30.0/23

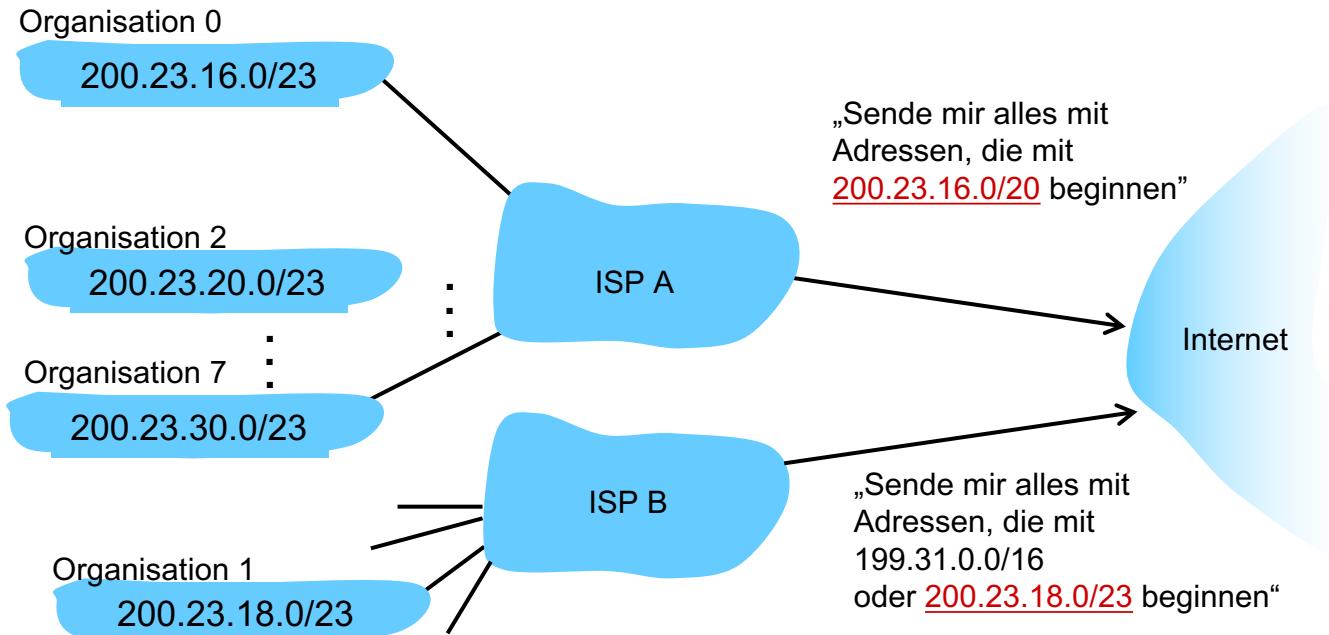
# Hierarchische Adressierung: Routenaggregation

Über hierarchische Adressierung können Routeninformationen effizient verteilt werden:



# Hierarchische Adressierung: Spezifischere Routen

ISP B hat spezifischere Rote für Organisation 1





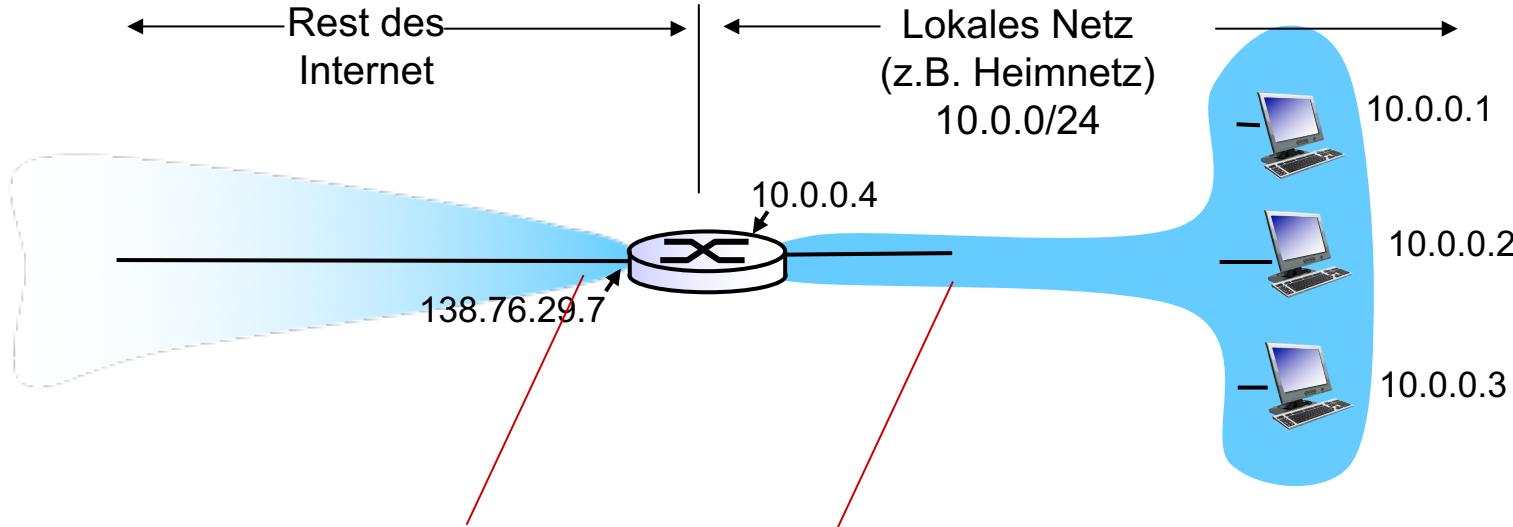
# Woher kommt die IP-Adresse (zum Dritten)

## Wie bekommt der ISP einen Adressblock?

- Internet Corporation for Assigned Names and Numbers (ICANN) bzw. deren Abteilung Internet Assigned Numbers Authority (IANA)
  - Allokiert Adressen [RFC 2050]
  - Verwaltet DNS-Root-Server
  - Weist Domainnamen zu, löst Streitigkeiten auf
  - <http://www.icann.org/>

# Network Address Translation (NAT)

[RFC 2663, RFC 3022]



Alle Datagramme, die das lokale Netz verlassen haben die gleiche Quell-(NAT)-IP Adresse 138.76.29.7, aber verschiedene Quellportnummern

Datagramme Quelle oder Ziel in diesem Netz haben 10.0.0/24 Adresse für Quelle bzw. Ziel (wie üblich)

# Network Address Translation (NAT) - Motivation

Lokales Netz benutzt zur Kommunikation mit Außenwelt nur eine einzige IP-Adresse

- Es wird kein Adressbereich vom ISP benötigt: nur eine Adresse für alle Geräte
- Adressen im lokalen Netz können geändert werden ohne dies der Außenwelt mitzuteilen
- ISP kann gewechselt werden, ohne lokale Geräte neu konfigurieren zu müssen
- Geräte im lokalen Netz können nicht von außen erreicht werden (Sicherheit!)

# Network Address Translation (NAT)

NAT-Router müssen:

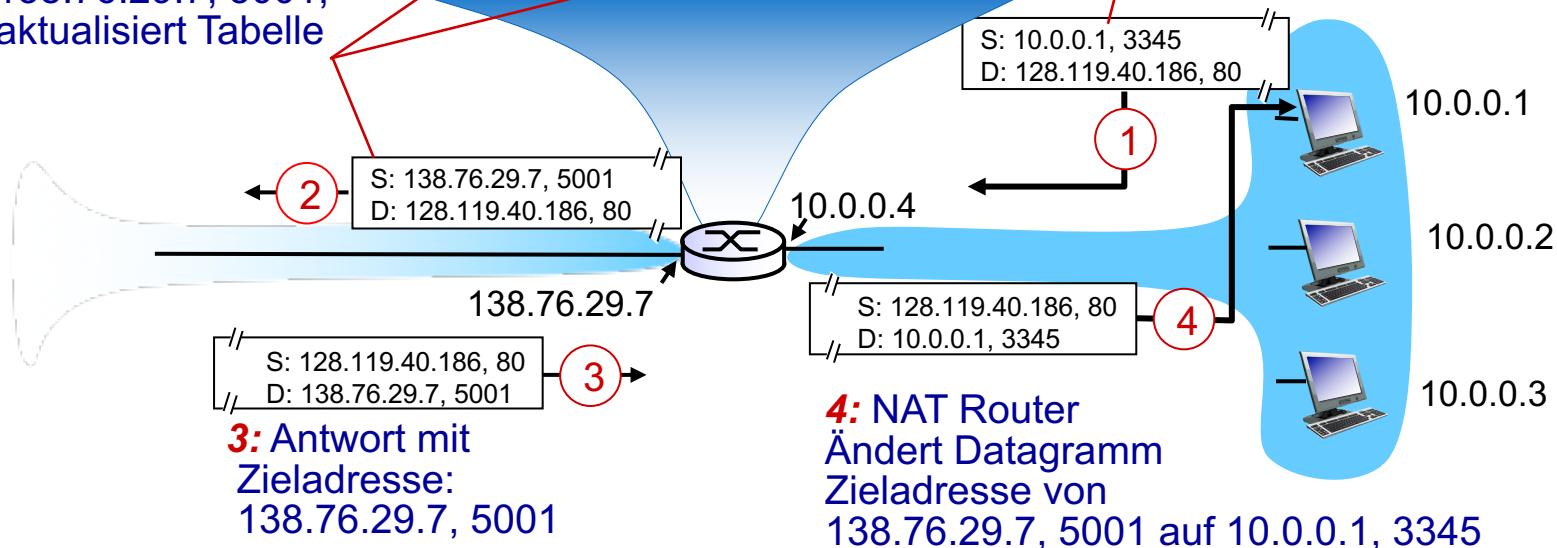
- Für ausgehende Datagramme: Ersetzen von  $(\text{Quell-IP-Adresse}, \text{Portnummer})$  gegen  $(\text{NAT IP Adresse}, \text{neue Portnummer})$  für jedes Datagramm
  - Clients werden an  $(\text{NAT IP Adresse}, \text{neue Portnummer})$  antworten
- Speichern der Abbildung  $(\text{Quell-IP-Adresse}, \text{Portnummer})$  zu  $(\text{NAT IP Adresse}, \text{neue Portnummer})$  in NAT-Übersetzungstabelle
- Für eingehende Datagramme: Ersetzen von  $(\text{NAT IP Adresse}, \text{neue Portnummer})$  durch  $(\text{Quell-IP-Adresse}, \text{Portnummer})$  gemäß NAT-Übersetzungstabelle für jedes Datagramm

# Network Address Translation (NAT)

**2:** NAT Router ändert Datagramm Quelladresse von 10.0.0.1, 3345 auf 138.76.29.7, 5001, aktualisiert Tabelle

NAT-Übersetzungstabelle	
WAN-seitige Adr.	LAN-seitige Adr.
138.76.29.7, 5001	10.0.0.1, 3345
.....	.....

**1:** Host 10.0.0.1 sendet Datagramm an 128.119.40.186, 80



# Network Address Translation (NAT)

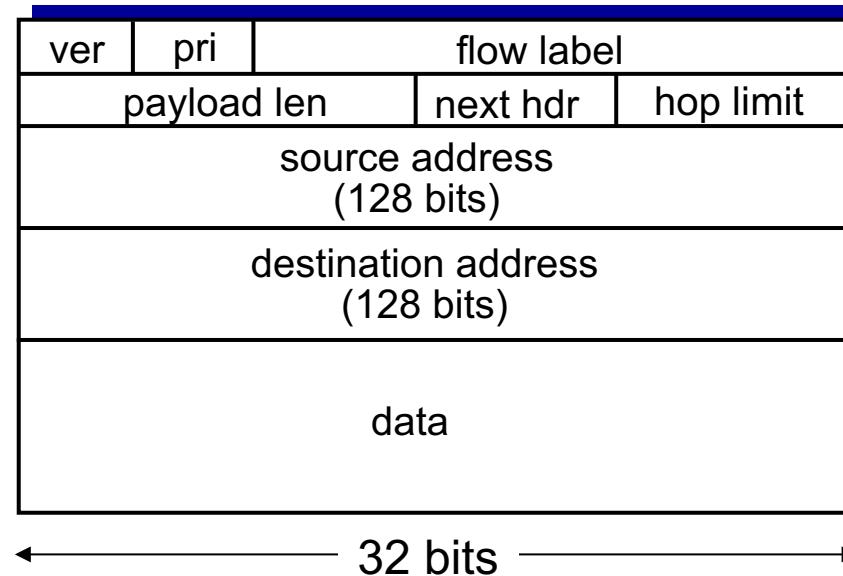
- 16-Bit Portnummer-Feld
  - >60.000 gleichzeitige Verbindungen mit einer einzigen IP-Adresse
- NAT wird kontrovers diskutiert
  - Router sollen nur bis zur Vermittlungsschicht agieren
  - IP-Adressknappheit soll von IPv6 gelöst werden
  - Verletzt Ende-zu-Ende Argument
    - NAT muss von Anwendungsentwicklern berücksichtigt werden, z.B. P2P
  - NAT Traversal: Was ist, wenn ein Client einen Server hinter NAT kontaktieren möchte?

# IPv6: Motivation

- Ursprungsgedanke: 32-bit Adressraum ist bereits zum großen Teil vergeben (Quelle: Wikipedia)
  - Blöcke für Vergabe von IANA an **Regional Internet Registries (RIR)** seit 31.01.2011
  - Blöcke von RIR Asien/Pazifik seit 15.04.2011
  - Blöcke von RIR Europa/Russland/Mittlerer Osten/Zentralasien seit 14.09.2012
  - Blöcke von RIR Lateinamerika/Karibik seit 10.06.2012
  - Blöcke von RIR Nordamerika 24.09.2015
  - In Afrika sind noch Blöcke zu haben!
- Weitere Motivation:
  - Neue Header-Formate beschleunigen Verarbeitung und Weiterleitung
  - Anycast-Adressen
  - Ermöglicht Dienstgütegarantien

# IPv6 Datagramm Format

- Feste Header-Länge (40 Byte), keine Fragmentierung erlaubt
- **Priority**: Priorität des Pakets im Datenfluss
- **Flow Label**: Identifiziert Ströme von zusammengehörigen Paketen (Flow-Konzept nicht gut ausgearbeitet)
- **Next Header**: Spezifiziert das Transportschichtprotokoll, dem die Daten übergeben werden



# Weitere Änderungen gegenüber IPv4

## Prüfsumme & Fragmentierung

- Wurden entfernt, um Verarbeitungszeit pro Router zu reduzieren

## Optionen

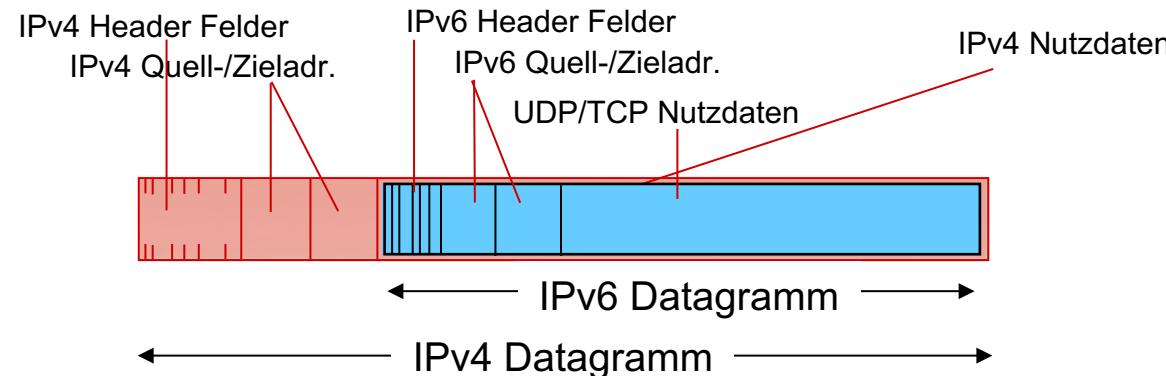
- Werden unterstützt, allerdings außerhalb des Headers, angezeigt durch „Next Header“ Feld

## ICMPv6: Neue Version von ICMP (s.u.)

- Zusätzliche Nachrichtentypen, z.B. „Packet Too Big“
- Funktionen für Management von Multicast-Gruppen

# Übergang von IPv4 zu IPv6

- Es können nicht alle Router gleichzeitig aktualisiert werden
  - Kein Internet-weiter „Flag-Day“!
  - Wie kann ein Mischbetrieb von IPv4 und IPv6 funktionieren?
- **Tunneln (Tunneling):** IPv6 Datagramme werden zwischen IPv4 Routern als Daten in IPv4 Datagrammen übertragen

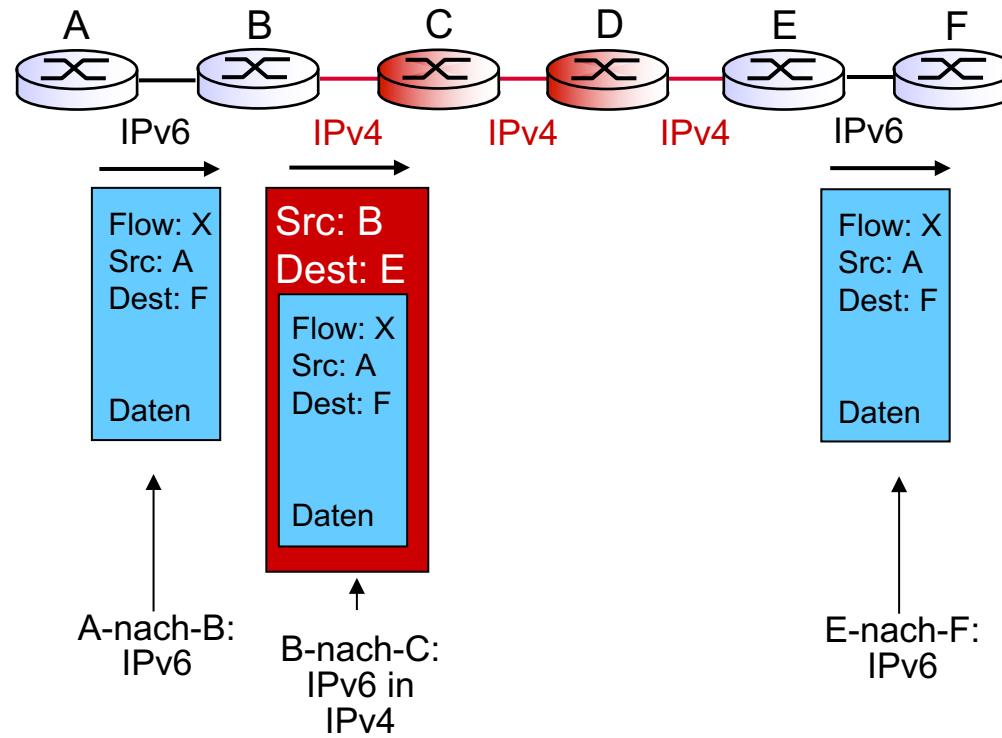


# Tunneln

Logische Sicht:



Physische Sicht:



# Adaption von IPv6

Google berichtet, dass aktuell 17,8% der Zugriffe über IPv6 erfolgen

- Vgl. <https://www.google.com/intl/en/ipv6/statistics.html>

Sehr langsame Adaption!

- > 20 Jahre und kein Ende in Sicht!
- Vgl. mit Änderungen auf Anwendungsschicht (WWW, Facebook, Multimedia-Streaming, Skype, ...)
- Warum?



# Funktionen der Vermittlungsschicht (Wdh.)

## Zwei Schlüsselfunktionen

- Weiterleitung (Forwarding): Transport von Paketen von Eingangsschnittstelle eines Routers zur passenden Ausgangsschnittstelle → Datenebene
- Wegbestimmung (Routing): Bestimmung des Wegs eines Pakets von der Quelle zum Ziel → Steuerungsebene

## Zwei Ansätze zur Strukturierung der Steuerungsebene

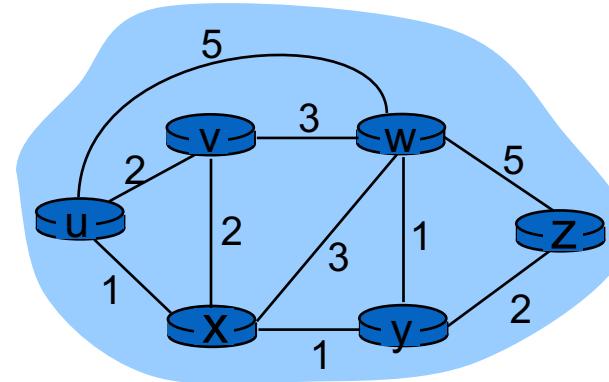
- Steuerung pro Router (traditionell, hier betrachtet)
- Logisch zentralisierte Steuerung (Software Defined Networking)

# Routing Protokolle

Ziel: Bestimmung eines „guten“ Wegs (**Pfad**, **Route**) vom Sender zum Empfänger durch das Netz der dazwischenliegenden Router

- **Pfad**: Sequenz von Routern die das Paket durchlaufen wird
- „**gut**“: niedrigste „Kosten“, „am schnellsten“, „am wenigsten ausgelastet“, ...
- Routing ist eine der größten Herausforderungen im Fachgebiet Kommunikationsnetze!

# Abstraktion des Netzes als Graph



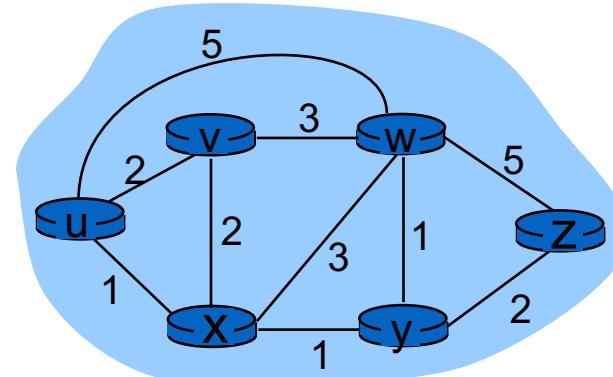
Ungerichteter Graph:  $G = (N, E)$

- Knoten-Menge  $N$  = Menge der Router =  $\{u, v, w, x, y, z\}$
- Kanten-Menge  $E$  = Menge der Verbindungen  
 $= \{\{u,v\}, \{u,w\}, \{u,x\}, \{v,x\}, \{v,w\}, \{x,w\}, \{x,y\}, \{w,y\}, \{w,z\}, \{y,z\}\}$

Graph-Abstraktion häufig auf verschiedenen Schichten verwendet

- Peer-to-Peer (Kanten sind TCP-Verbindungen), Mobile Ad Hoc Netze (Kanten sind Funkverbindungen), ...

# Graph-Abstraktion: Kosten



$c(x, x')$  = Kosten der Verbindung  $(x, x')$ , z.B.  $c(x, w) = 3$

- Kosten können immer 1 sein, oder umgekehrt proportional zu Bandbreite, Auslastung, ...

Kosten eines Pfads:  $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

Schlüssefrage: Was ist der Pfad mit den geringsten Kosten zwischen  $u$  und  $z$ ?

**Routing-Algorithmus:** Findet den Pfad mit den geringsten Kosten

# Klassifizierung von Routing-Algorithmen

## Globale oder dezentrale Informationen?

### Global:

- Alle Router haben komplettes Abbild von Topologie und Kosten
- **Link State**-Algorithmen

### Lokal / Dezentral:

- Router kennen direkt verbundene Nachbarn sowie Kosten dieser Verbindungen
- Iterativer Berechnungsprozess, Informationsaustausch mit Nachbarn
- **Distanzvektor**-Algorithmen

## Statisch oder dynamisch?

### Statisch:

- Routen ändern sich sehr langsam

### Dynamisch:

- Routen Ändern sich häufig
  - Periodische Updates
  - Reaktion auf Kostenänderungen

# Link-State: Dijkstra-Algorithmus

Topologie und Verbindungskosten sind allen Knoten bekannt

- Wird über „Link State Broadcast“ sichergestellt

Berechnung des „kürzesten“ Pfades (d.h. mit den geringsten Kosten) von ausgewähltem Knoten (Quelle) zu allen anderen Knoten

- Ergibt die Weiterleitungstabelle dieses Knotens

Iterativ: Nach  $k$  Iterationen sind kürzeste Pfade zu  $k$  Knoten bekannt

# Dijkstra-Algorithmus: Notation

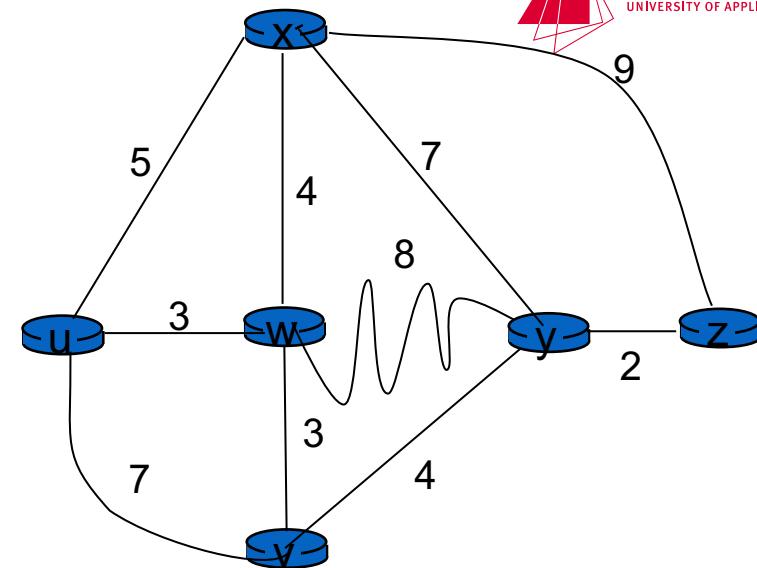
- $c(x, y)$ : Verbindungskosten von Knoten  $x$  zu Knoten  $y$ 
  - $c(x, y) = \infty$  für nicht benachbarte Knoten  $x$  und  $y$
- $D(v)$ : Aktuell berechnete Kosten für Pfad von Quelle zu Knoten  $v$  (muss noch nicht der kürzeste Pfad sein!)
- $p(v)$ : Vorgänger (Predecessor) von  $v$  auf dem Pfad von der Quelle zu  $v$
- $N'$ : Menge von Knoten, für die der kürzeste Pfad bereits berechnet wurde (pro Runde kommt ein Knoten hinzu)

# Dijkstra-Algorithmus

- 1 **Initialisierung:**
- 2  $N' = \{u\}$
- 3 Für alle Knoten  $v$
- 4 Wenn  $v$  Nachbar von  $u$
- 5 Dann  $D(v) = c(u, v)$
- 6 Sonst  $D(v) = \infty$
- 7

**8 Schleife**

- 9 Finde  $w$  nicht in  $N'$  mit  $D(w)$  minimal
- 10 Füge  $w$  zu  $N'$  hinzu
- 11 Aktualisiere  $D(v)$  für alle Nachbarn  $v$  von  $w$  und nicht in  $N'$  :
- 12  $D(v) = \min(D(v), D(w) + c(w, v))$
- 13 /\* Neue Kosten zu  $v$  sind entweder alte Kosten zu  $v$  oder  
Kosten des kürzesten Pfades zu  $w$  plus Kosten von  $w$  nach  $v$  \*/
- 15 Wenn  $D(v)$  aktualisiert wurde, setze  $p(v) = w$
- 16 **Bis alle Knoten in  $N'$  sind**

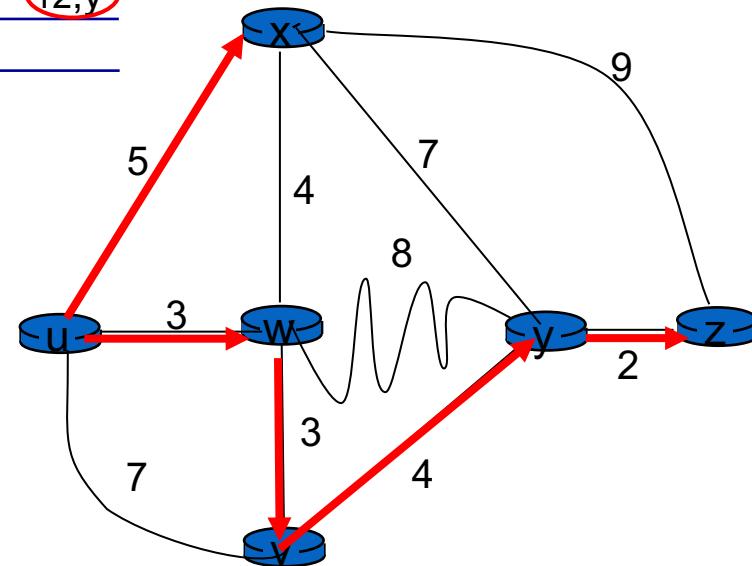


# Dijkstra-Algorithmus: Beispiel

Step	$N'$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
		$p(v)$	$p(w)$	$p(x)$	$p(y)$	$p(z)$
0	u	7,u	3,u	5,u	$\infty$	$\infty$
1	uw	6,w		5,u	11,w	$\infty$
2	uwx	6,w		11,w	14,x	
3	uwxv			10,v	14,x	
4	uwxvy				12,y	
5	uwxvyz					

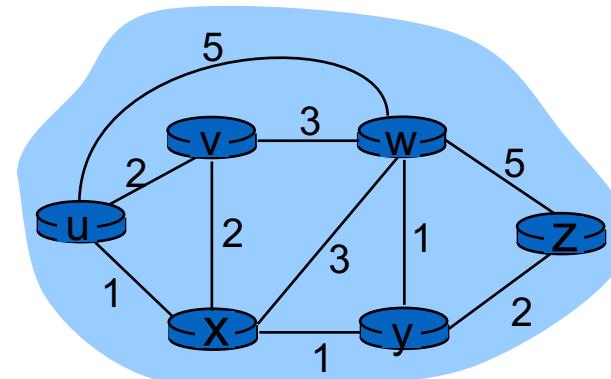
Die kürzesten Pfade ergeben sich aus den Vorgängerknoten

- Bei gleichen Kosten kann beliebig gewählt werden



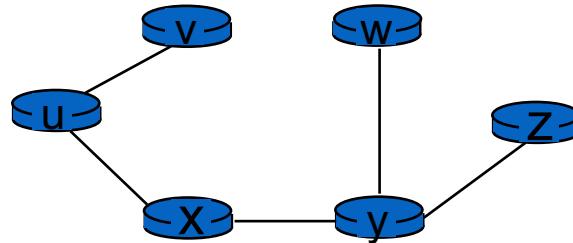
# ! Übung: Dijkstra-Algorithmus

Step	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0						
1						
2						
3						
4						
5						



# Dijkstra-Algorithmus: Zweites Beispiel

Resultierende kürzeste Pfade



Weiterleitungstabelle von  $u$

Ziel	Verbindung
$v$	$(u, v)$
$x$	$(u, x)$
$y$	$(u, x)$
$w$	$(u, x)$
$z$	$(u, x)$

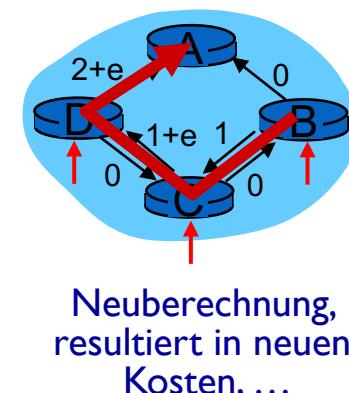
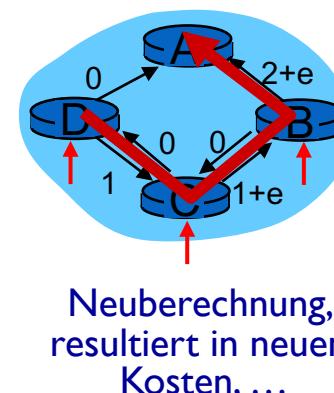
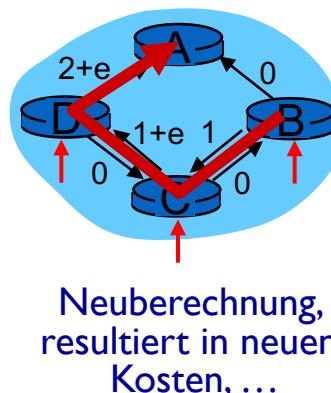
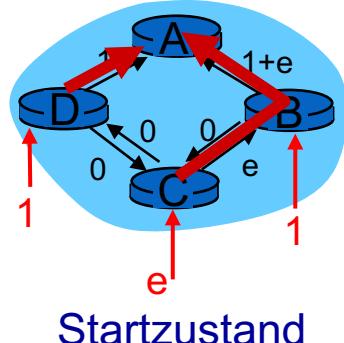
# Dijkstra-Algorithmus – Diskussion

## Komplexität

- $n$  Knoten
- In jeder Runde müssen alle Knoten geprüft werden, die noch nicht in  $N'$  sind
- $n(n+1)/2$  Vergleiche, Komplexität  $O(n^2)$ 
  - Komplexe Implementierung mit Heap erreicht  $O(n \log n)$

## Oszillieren ist möglich

- Z.B. wenn Link-Kosten von aktueller Last abhängen, Knoten synchronisiert



# Distanzvektor-Algorithmen

## Grundlage: Bellman-Ford-Gleichung

Für

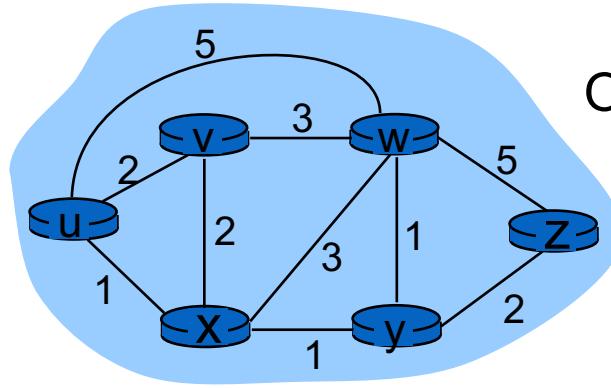
$d_x(y) :=$  Kosten des kürzesten Weges von  $x$  nach  $y$

Ist

$$d_x(y) = \min_v \{c(x, v) + d_v(y)\}$$

|||  
Kosten von Nachbar  $v$  zu Ziel  $y$   
Kosten zu Nachbar  $v$   
**Minimum über alle Nachbarn  $v$  von  $x$**

# Bellman-Ford Beispiel



Offensichtlich ist  $d_v(z) = 5$ ,  $d_x(z) = 3$ ,  $d_w(z) = 3$

B-F-Gleichung besagt:

$$\begin{aligned}
 d_u(z) &= \min \{ c(u,v) + d_v(z), \\
 &\quad c(u,x) + d_x(z), \\
 &\quad c(u,w) + d_w(z) \} \\
 &= \min \{ 2 + 5, \\
 &\quad 1 + 3, \\
 &\quad 5 + 3 \} = 4
 \end{aligned}$$

Knoten mit Minimum wird als Next Hop in Weiterleitungstabelle verwendet

# Distanzvektor-Algorithmen

$D_x(y)$  = Schätzung der Kosten von  $x$  nach  $y$

- $x$  speichert seinen **Distanzvektor**  $\mathbf{D}_x = [D_x(y): y \in N]$
- Distanzvektoren werden unter Nachbarn ausgetauscht

Sicht von Knoten  $x$ :

- Kennt die Kosten zu jedem Nachbarn  $v$ :  $c(x,v)$
- Kennt die Distanzvektoren seiner Nachbarn, d.h. für jeden Nachbarn  $v$  kennt  $x$ :

$$\mathbf{D}_v = [D_v(y): y \in N]$$

# Distanzvektor-Algorithmen

## Idee:

- Von Zeit zu Zeit sendet jeder Knoten seine(n) eigene Distanzvektor(-Schätzung) an seine Nachbarn
- Wenn  $x$  neuen DV-Update von Nachbarn empfängt, wird Distanzvektor von  $x$  mit Hilfe von Bellman-Ford-Gleichung aktualisiert:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \text{ für jeden Knoten } y \in N$$

- Unter „normalen“ Bedingungen konvergieren die Schätzungen  $D_x(y)$  gegen die echten Kosten  $d_x(y)$  der kürzesten Pfade

# Distanzvektor-Algorithmen

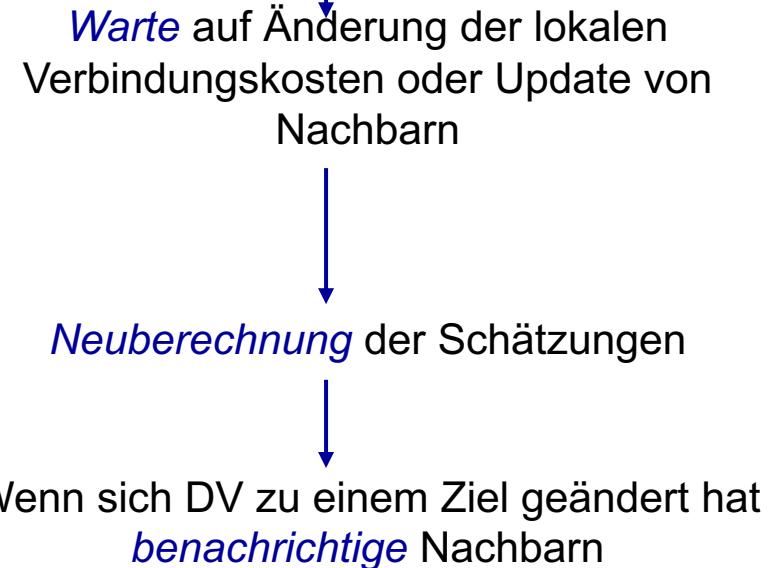
## Iterativ, asynchron

- Jede lokale Iteration ausgelöst von Änderung der Verbindungskosten oder DV-Update von einem Nachbarn

## Verteilt

- Jeder Knoten benachrichtigt Nachbarn nur wenn sich DV ändert
- Nachbarn benachrichtigen dann ihre Nachbarn falls nötig

## Ablauf pro Knoten:



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$



**Knoten x**

**Tabelle**

	x	y	z	
x	0	2	7	
von	y	$\infty$	$\infty$	$\infty$
z	$\infty$	$\infty$	$\infty$	

*nach*

	x	y	z	
x	0	2	3	
von	y	2	0	1
z	7	1	0	

*nach*

**Knoten y**

**Tabelle**

	x	y	z	
x	$\infty$	$\infty$	$\infty$	
von	y	2	0	1
z	$\infty$	$\infty$	$\infty$	

*nach*

	x	y	z	
x	$\infty$	$\infty$	$\infty$	
von	y	$\infty$	$\infty$	$\infty$
z	$\infty$	$\infty$	$\infty$	

	x	y	z	
x	$\infty$	$\infty$	$\infty$	
von	y	$\infty$	$\infty$	$\infty$
z	$\infty$	$\infty$	$\infty$	

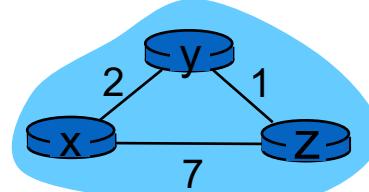
	x	y	z	
x	$\infty$	$\infty$	$\infty$	
von	y	$\infty$	$\infty$	$\infty$
z	$\infty$	$\infty$	$\infty$	

	x	y	z	
x	$\infty$	$\infty$	$\infty$	
von	y	$\infty$	$\infty$	$\infty$
z	$\infty$	$\infty$	$\infty$	

	x	y	z	
x	$\infty$	$\infty$	$\infty$	
von	y	$\infty$	$\infty$	$\infty$
z	$\infty$	$\infty$	$\infty$	

	x	y	z	
x	$\infty$	$\infty$	$\infty$	
von	y	$\infty$	$\infty$	$\infty$
z	$\infty$	$\infty$	$\infty$	

	x	y	z	
x	$\infty$	$\infty$	$\infty$	
von	y	$\infty$	$\infty$	$\infty$
z	$\infty$	$\infty$	$\infty$	



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$



**Knoten x**

**Tabelle**

	x	y	z
x	0	2	7
von			
y	$\infty$	$\infty$	$\infty$
z	$\infty$	$\infty$	$\infty$

*nach*

x y z

	x	y	z
x	0	2	3
von			
y	2	0	1
z	7	1	0

*nach*

x y z

**Tabelle**

	x	y	z
x	0	2	3
von			
y	2	0	1
z	3	1	0

**Knoten y**

**Tabelle**

	x	y	z
x	$\infty$	$\infty$	$\infty$
von			
y	2	0	1
z	$\infty$	$\infty$	$\infty$

*nach*

x y z

**Tabelle**

	x	y	z
x	0	2	3
von			
y	2	0	1
z	3	1	0

**Knoten z**

**Tabelle**

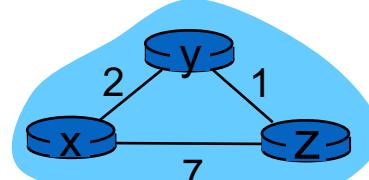
	x	y	z
x	$\infty$	$\infty$	$\infty$
von			
y	$\infty$	$\infty$	$\infty$
z	7	1	0

*nach*

x y z

**Tabelle**

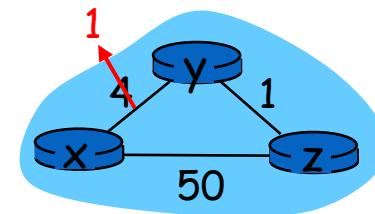
	x	y	z
x	0	2	3
von			
y	2	0	1
z	3	1	0



Zeit

# DV: Änderung von Verbindungskosten

- Knoten bemerkt lokale Kostenänderung
- Aktualisiert Routing-Info, berechnet DV neu
- Wenn sich DV ändert, Nachbarn benachrichtigen



! Übung: Berechnen Sie die neuen DV-Tabellen der Knoten!

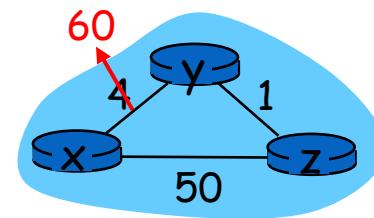
- “*Good news travel fast*”
  - $t_0$ :  $y$  bemerkt Kostenänderung, aktualisiert DV, benachrichtigt die Nachbarn
  - $t_1$ :  $z$  empfängt Aktualisierung von  $y$ , aktualisiert DV-Tabelle, berechnet neue Kosten zu  $x$ , sendet DV an Nachbarn
  - $t_2$ :  $y$  empfängt  $z$ 's Update, aktualisiert DV-Tabelle.  $y$ 's Kosten ändern sich nicht, daher kein Update an  $z$

# DV: Änderung von Verbindungskosten

! Übung: Berechnen Sie die neuen DV-Tabellen der Knoten!

*“Bad news travel slow”*

- Count-to-infinity-Problem



Lösungsansatz Poisoned Reverse:

- Wenn  $z$  durch  $y$  routet, um zu  $x$  zu gelangen:  
 $z$  meldet  $y$  Distanz  $d_z(x) = \infty$
- Löst dies das Problem vollständig?

# Vergleich Link State – Distanzvektor

## Nachrichtenkomplexität

- LS: Mit  $n$  Knoten  $e$  Kanten  $O(ne)$  Nachrichten
- DV: Austausch nur zwischen Nachbarn (Anzahl variiert)

## Konvergenz

- LS:  $O(n^2)$  mit  $O(ne)$  Nachrichten
- DV: Konvergenzzeit variiert, Schleifen in Pfaden, Count-to-infinity Problem

## Robustheit: Was passiert bei Router-Fehlfunktion?

- LS: Knoten kann falsche Verbindungskosten bekanntmachen, aber jeder Knoten berechnet nur eigene Tabelle
- DV: Knoten kann falsche Pfadkosten bekanntmachen, Tabelle wird von anderen benutzt -> Fehler propagieren durch Netz

# Skalierbares Internet-Routing (1)

Routing-Betrachtungen waren bis hier stark idealisiert

- Alle Router identisch, flaches Netz, ...

Anders in der Praxis:

- Größe: Milliarden von Zielen
  - Nicht alle in Routing-Tabelle speicherbar!
  - Austausch der Routing-Tabellen würde Verbindungen überlasten!
- Administrative Autonomie
  - Internet = Netz von Netzen
  - Jeder Netz-Administrator möchte Routing in „seinem“ Netz kontrollieren

# Skalierbares Internet-Routing (2)

Zusammenfassen von Routern zu **Autonomen Systemen (AS)** oder **Domains**

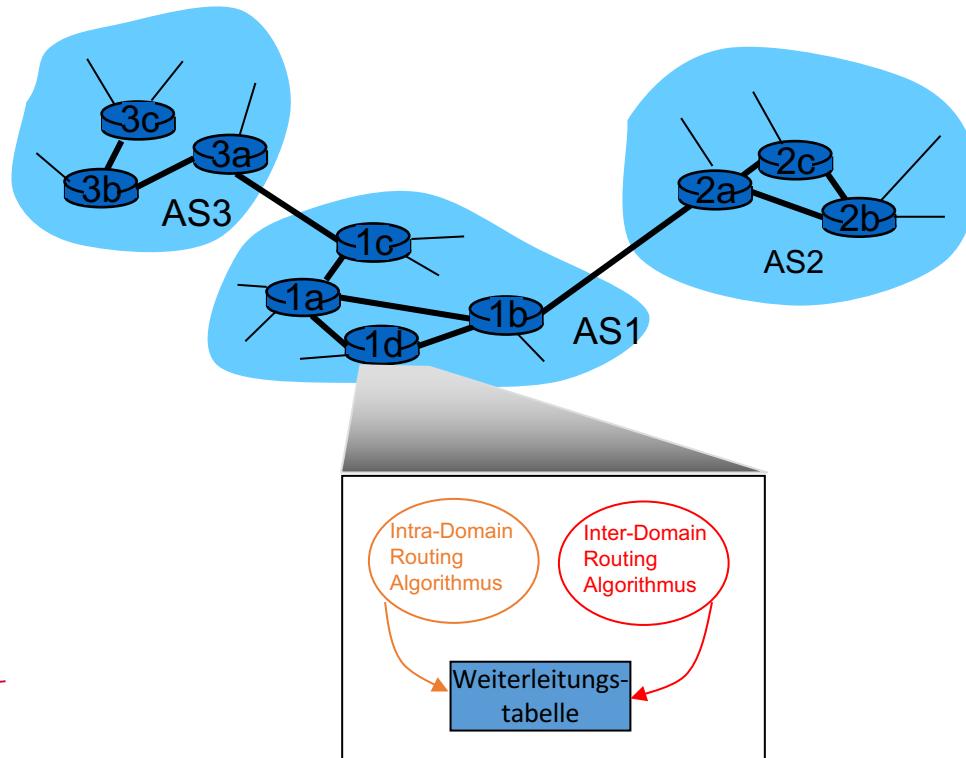
## Intra-AS Routing / Intra-Domain-Routing

- Routing zwischen Routern im gleichen AS („Netz“)
- Alle Router in einem AS nutzen das selbe Intra-Domain-Protokoll
- Router in verschiedenen AS können verschiedene Intra-Domain-Protokolle nutzen
- Gateway-Router am „Rand“ eines AS hat Verbindungen zu Routern in anderen AS

## Inter-AS Routing / Inter-Domain-Routing

- Routing zwischen AS
- Gateway-Router betreiben sowohl Inter- als auch Intra-Domain-Routing

# Verbindung zwischen Autonomen Systemen



Weiterleitungstabellen werden sowohl von Inter- als auch von Intra-Domain-Routing konfiguriert

- Intra-Domain-Routing legt Einträge für Ziele im eigenen AS fest
- Inter- & Intra-Domain-Routing legt Einträge für externe Ziele fest

# Intra-Domain Routing

- Auch bekannt als **Interior Gateway Protocols (IGP)**
- Gebräuchlichste Protokolle:
  - **Routing Information Protocol (RIP)** [RFC 1058, RFC 2454]
    - Distanzvektor-Protokoll mit Pfadkosten = Anzahl durchquerter Subnetze (max. 15)
      - Austausch der kompletten Routing-Tabellen zwischen Nachbarn per UDP (Port 580) alle 30s,
      - Timeout für Nachbarn nach 180s
    - **Open Shortest Path First (OSPF)** [RFC 2328]
      - „Open“ = frei verfügbar
      - Links-State-Protokoll mit konfigurierbaren Kantenkosten
        - Verteilung der Verbindungsinformationen (Link State)
        - Komplette Netztopologie (beschränkt auf AS) in jedem Router
        - Routen-Berechnung mit Dijkstra-Algorithmus
      - Router „flutet“ Link State Informationen an alle Router im gesamten AS mind. alle 30 Minuten
      - OSPF setzt direkt auf IP auf anstatt auf TCP / UDP (Protokoll-ID 89)
      - Bietet Sicherheit, Mehrfachpfade, Unicast-/Multicast-Unterstützung, Hierarchien, ...

# Internet Inter-Domain-Routing

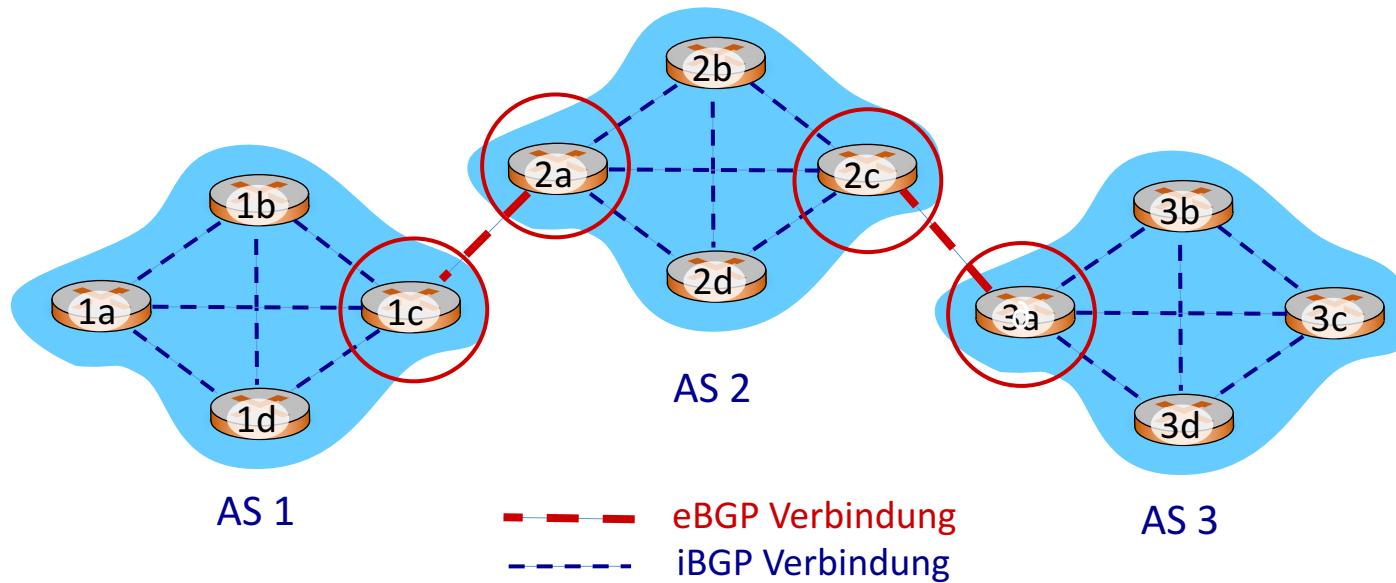
## Border Gateway Protocol (BGP) [RFC 4271]

- Standard Inter-Domain Routing Protokoll  
*„Klebstoff, der das Internet zusammen hält“*
- Ermöglicht Existenz eines Subnetz im Internet bekannt zu machen
- Ermöglicht einem AS zu erfahren, welche Subnetze über benachbartes AS erreichbar sind

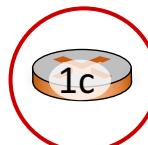
BGP bietet AS Mechanismen für:

- **eBGP**: Informationen über Subnetz-Erreichbarkeit von benachbarten AS
- **iBGP**: Verteilung von Erreichbarkeitsinformationen an alle internen Router
- Auswahl von „guten“ Routen basierend auf Erreichbarkeit und Strategien („**Policy**“)

# eBGP, iBGP Verbindungen



Gateway Routers mit sowohl eBGP als auch iBGP



# BGP Grundlagen

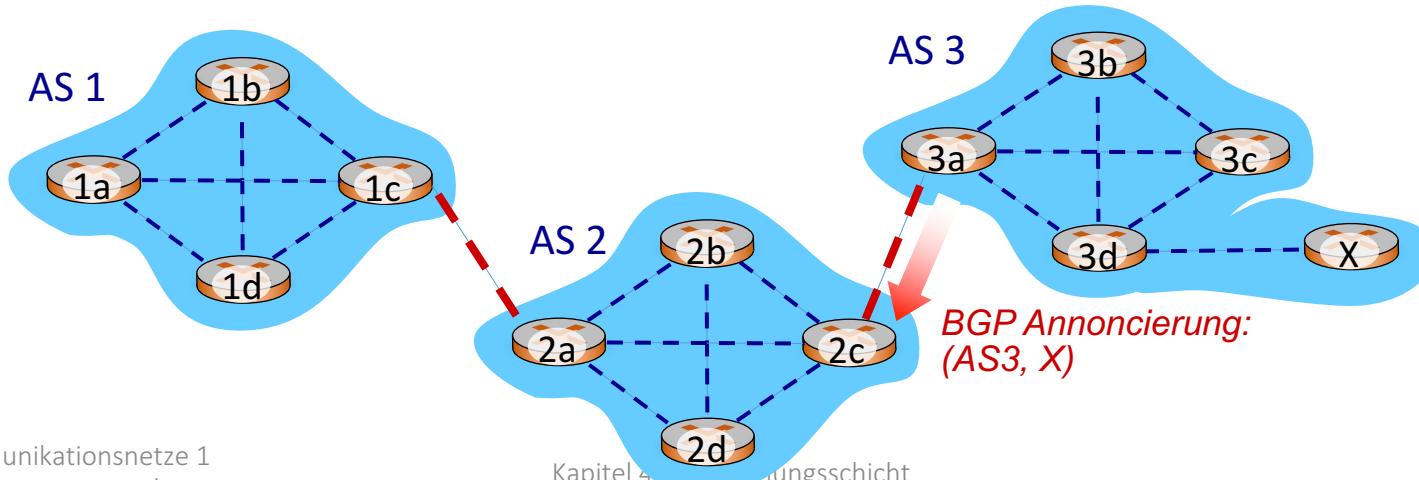
BGP Sitzung: BGP Router („Peers“) tauschen BGP-Nachrichten über TCP-Verbindungen (Port 179) aus

- BGP ist ein **Pfad-Vektor-Protokoll**: Annonciert Pfade zu verschiedenen Netz-Präfixen (z.B. zu 193.196.64.0/18)
- Pfade bestehen aus AS-Nummern (ASN), eindeutig für jedes AS

Beispiel:

Wenn AS3 Gateway Router 3a einen Pfad (AS3, X) zu AS2 Gateway Router 2c annonciert:

- AS3 verspricht AS2, dass Datagramme in Richtung X weitergeleitet werden



# BGP Attribute und Routen

Annoncierte Präfixe enthalten BGP Attribute

- Präfix + Attribute = „Route“

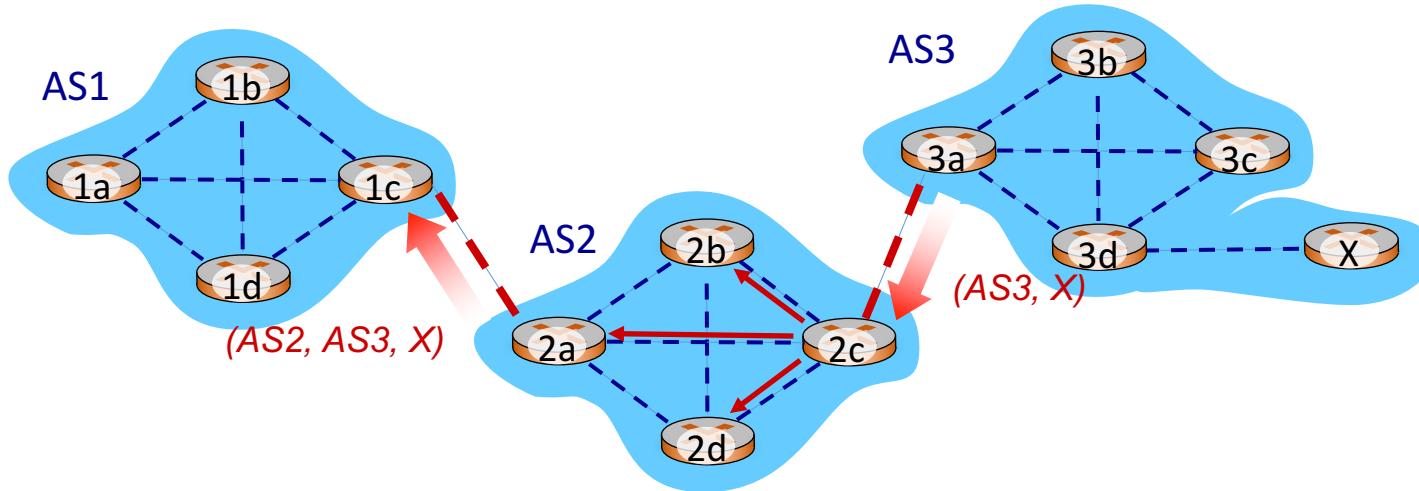
Zwei wichtige Attribute

- **AS-PATH:** AS-Liste, über die ein Präfix empfangen wurde
- **NEXT-HOP:** AS-interner Router auf dem Weg zu einem spezifischen AS

Strategie-basiertes (**Policy-based**) Routing

- Gateway kann Routen basierend auf **Import Policy** annehmen / ablehnen
  - z.B. niemals durch AS Y Routen, ...
  - AS Strategie legt auch fest, ob bestimmte Routen an benachbarte AS annonciert werden

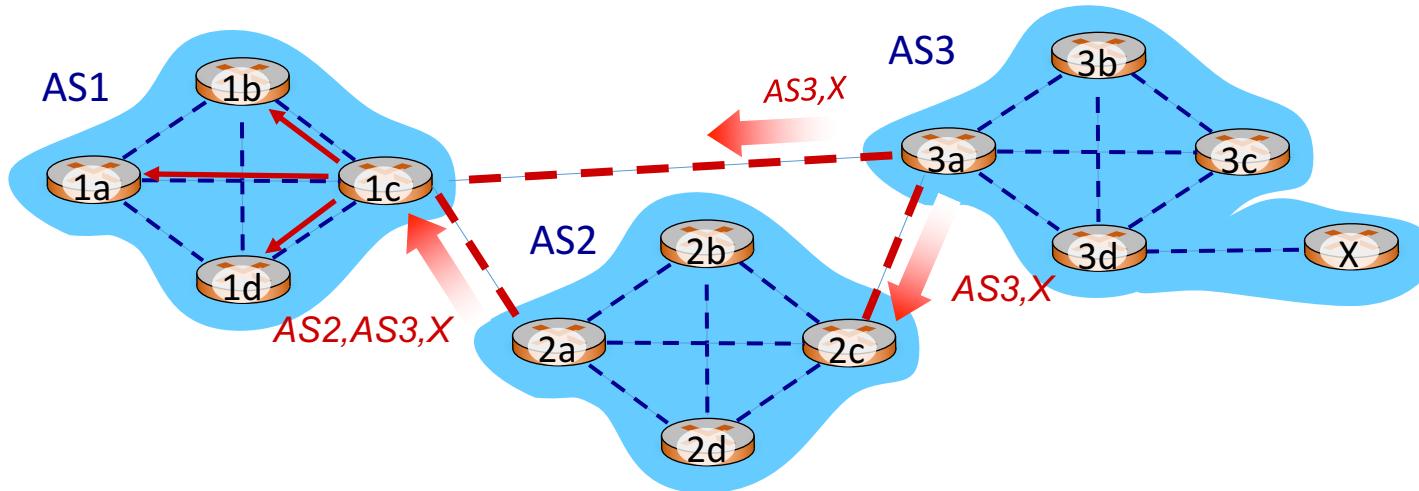
# BGP Pfad-Annoncierung (Path Advertisement)



## Beispiel:

- AS2 Router 2c erhält Pfad **(AS3, X)** (per eBGP) von AS3 Router 3a
- Gemäß AS2 Strategie akzeptiert AS2 Router 2c Pfad **(AS3, X)** und annonciert diesen (per iBGP) an alle AS2 Router
- Gemäß AS2 Strategie annonciert AS2 Router 2a (via eBGP) den Pfad **(AS2, AS3, X)** an AS1 Router 1c

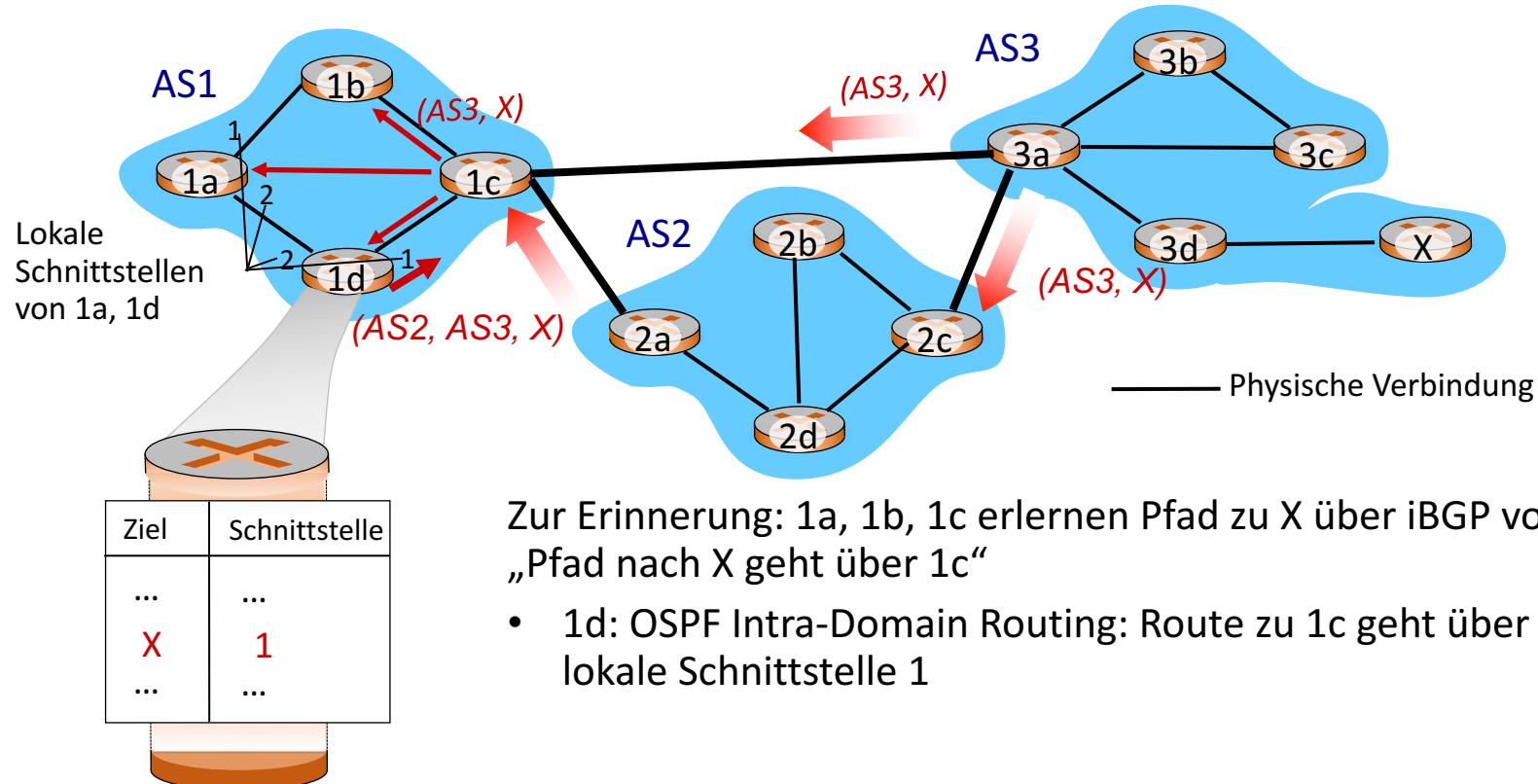
# BGP Pfad-Annoncierung (Path Advertisement)



Gateway Router können mehrere Pfade zu einem Ziel erlernen

- AS1 Gateway Router 1c lernt Pfad **(AS2, AS3, X)** (via eBGP) von 2a
- AS1 Gateway Router 1c lernt Pfad **(AS3, X)** (via eBGP) von 3a
- AS1 Strategie bestimmt, welcher Pfad von 1c (via iBGP) in AS1 annonciert wird

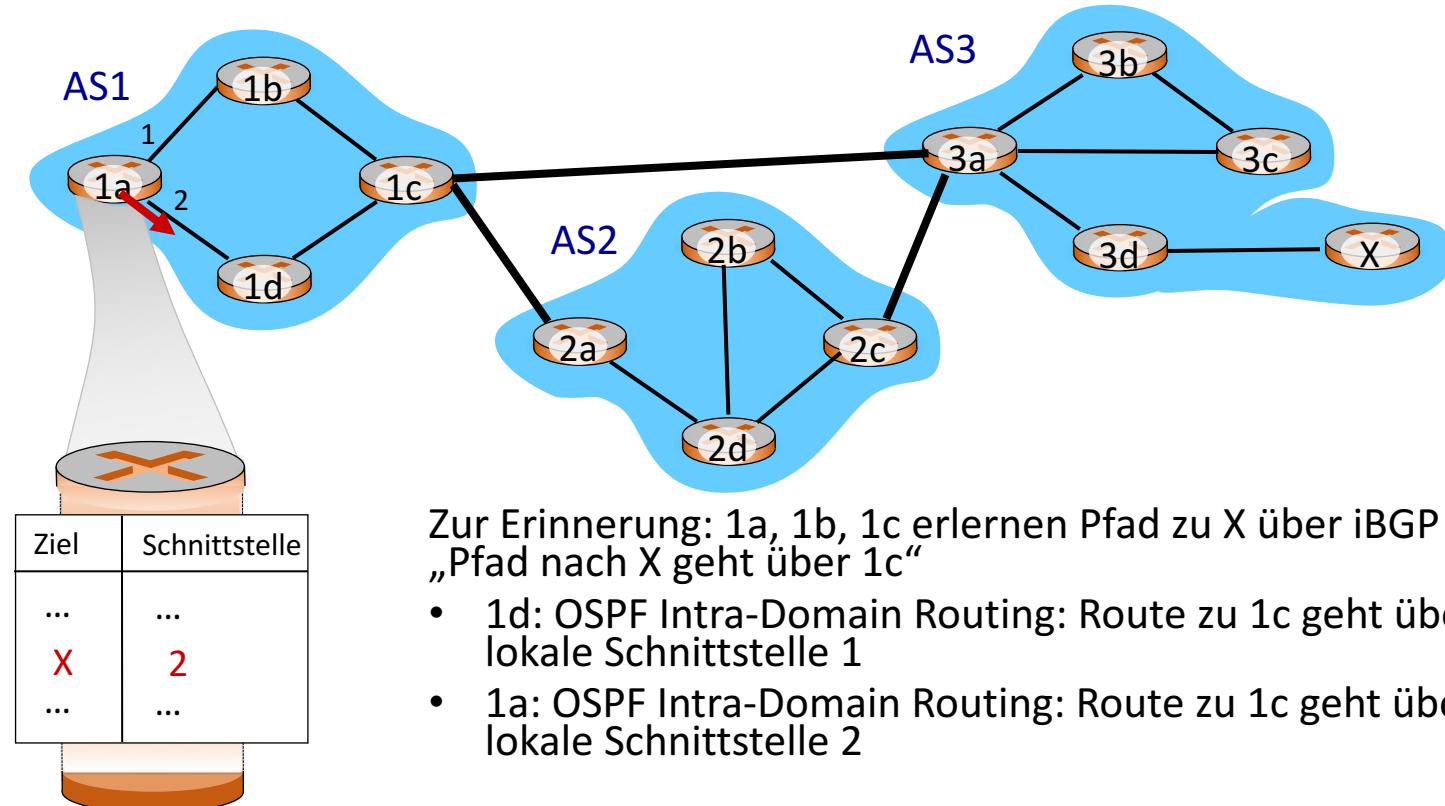
# BGP, OSPF und Weiterleitungstabelleneinträge



Zur Erinnerung: 1a, 1b, 1c erlernen Pfad zu X über iBGP von 1c:  
„Pfad nach X geht über 1c“

- 1d: OSPF Intra-Domain Routing: Route zu 1c geht über lokale Schnittstelle 1

# BGP, OSPF und Weiterleitungstabelleneinträge



Zur Erinnerung: 1a, 1b, 1c erlernen Pfad zu X über iBGP von 1c:  
„Pfad nach X geht über 1c“

- 1d: OSPF Intra-Domain Routing: Route zu 1c geht über lokale Schnittstelle 1
- 1a: OSPF Intra-Domain Routing: Route zu 1c geht über lokale Schnittstelle 2

# BGP Routen-Auswahl

Wenn Gateway Router mehrere Pfade zu einem Ziel erlernen passiert Auswahl basierend auf (in dieser Reihenfolge):

1. Strategie
2. Kürzester AS-PATH
3. Niedrigste AS-Interne Kosten zu NEXT-HOP Router („Hot Potato Routing“)
4. Weiteren Kriterien

# Internet Control Message Protocol (ICMP)

Genutzt von Endsystemen und Routern um Verbindungsschicht-Informationen auszutauschen, z.B.

- Fehlerberichte: Nicht erreichbare Hosts, Netze, Ports, Protokolle
- Echo Anfragen / Antworten (genutzt von ping)

In Vermittlungsschicht „über“ IP

- ICMP-Nachricht wird in IP Datagramm übertragen

ICMP-Nachricht: Typ, Code, erste 8 Bytes des IP Datagramms das Fehler verursacht hat

<u>Typ</u>	<u>Code</u>	<u>Beschreibung</u>
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest. host unreachable
3	2	dest. protocol unreachable
3	3	dest. port unreachable
3	6	dest. network unknown
3	7	dest. host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header



# Anwendung: Traceroute über ICMP

Quelle sendet Sequenz von UDP Segmenten zum Ziel

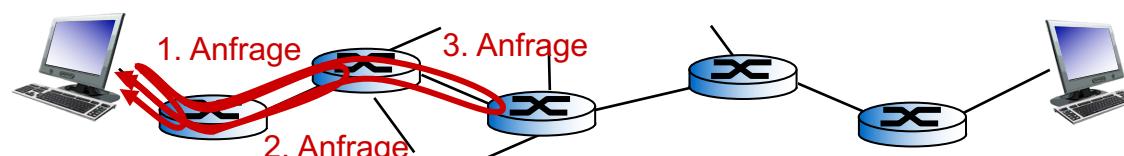
- Erstes hat TTL=1, zweites TTL=2, ...
- Portnummer frei erfunden

Wenn Segment n bei n-tem Router ankommt

- Router verwirft Datagramm und sendet ICMP-Nachricht an Quelle (Typ 0, Code 0 „TTL expired“)
- ICMP-Nachricht enthält Name & IP des Routers
- Daraus lässt sich Route rekonstruieren

Abbruchkriterium

- UDP Segment kommt beim Ziel an
- Ziel antwortet mit ICMP-Nachricht an Quelle (Typ 3, Code 3 „dest port unreachable“)
- Quelle stoppt Vorgang



# Zusammenfassung Kapitel 4

- Organisation in Daten- und Steuerungsebene
- Weiterleitung in einem Router
- Adressierung mit IPv4 & IPv6, NAT
- Routing-Algorithmen
  - Link State und Distanzvektor
  - Anwendung in RIP, OSPF & BGP
- ICMP