

# Netzwerktechnologien

## 3 VO

Univ.-Prof. Dr. Helmut Hlavacs  
[helmut.hlavacs@univie.ac.at](mailto:helmut.hlavacs@univie.ac.at)

Dr. Ivan Gojmerac  
[gojmerac@ftw.at](mailto:gojmerac@ftw.at)

Bachelorstudium Medieninformatik  
SS 2012

# Kapitel 4 – Netzwerkschicht

4.1 Die Netzwerkschicht

4.2 Virtuelle Leitungen und Datagrammnetzwerke

4.3 Was steckt in einem Router?

4.4 Das Internetprotokoll (IP): Weiterleiten und Adressieren im Internet

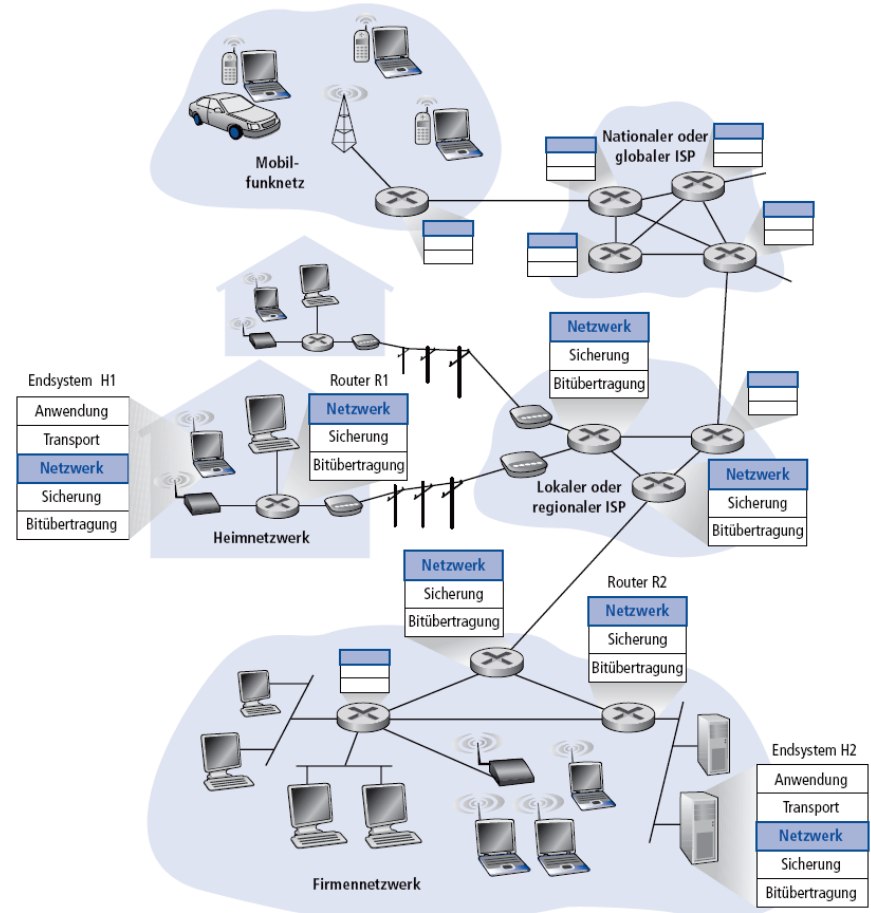
4.5 Routing-Algorithmen

4.6 Routing im Internet

4.7 Broadcast- und Multicast-Routing

## 4.1 Die Netzwerkschicht

- Auch: **Vermittlungsschicht** oder **Network Layer**
- Daten von der nächsthöheren Schicht (Transportschicht) des Senders entgegennehmen
- In Datagramme verpacken
- Durch das Netzwerk leiten
- Auspacken des Vermittlungspakets beim Empfänger
- Ausliefern der Daten an die nächsthöhere Schicht (Transportschicht) des Empfängers
- Netzwerkschicht existiert in jedem Host und Router!

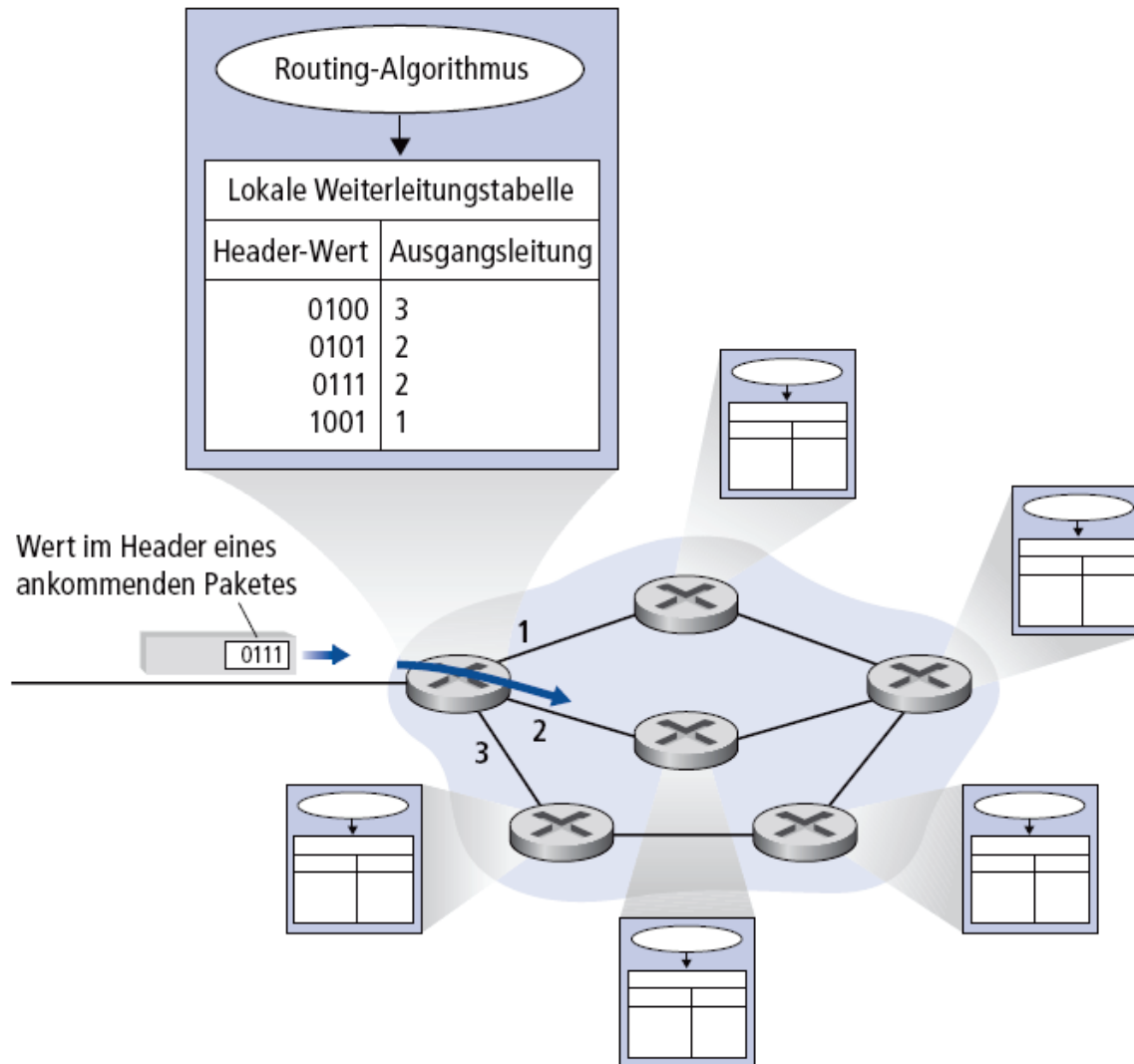


## 4.1.1 Funktionen der Netzwerkschicht

- Weiterleiten von Paketen (Forwarding):
  - Router nimmt Paket auf einer Eingangsleitung entgegen
  - Router bestimmt die Ausgangsleitung anhand lokaler Informationen (z.B. Routing-Tabelle)
  - Router legt das Paket auf die Ausgangsleitung
- Wegewahl (Routing):
  - Router kommunizieren miteinander, um geeignete Wege durch das Netzwerk zu bestimmen
  - Als Ergebnis erhalten sie Informationen, wie welches System im Netzwerk zu erreichen ist (z.B. wird eine Routingtabelle mit Einträgen gefüllt)
- Metapher:
  - Routing = Planen einer Strecke für eine Autofahrt
  - Weiterleitung = Verhalten an einer Autobahnkreuzung



# 4.1.1 Zusammenspiel von Routing und Forwarding



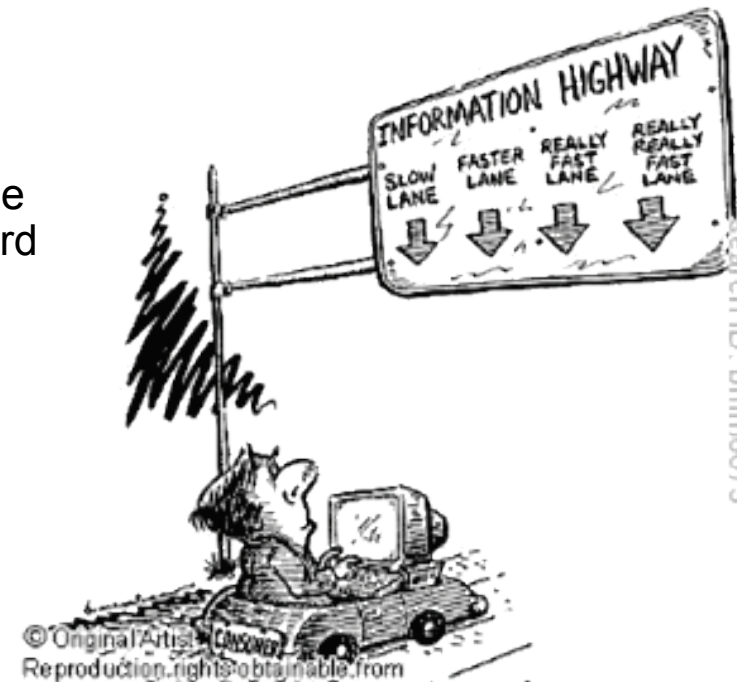
## 4.1.1 Verbindungsauf- und Verbindungsabbau

- Dritte wichtige Funktion in einigen Netzwerkarchitekturen:
  - ATM (Asynchronous Transfer Mode), Frame Relay, X.25
- Bevor Daten übertragen werden können, wird eine virtuelle Verbindung vom Sender zum Empfänger aufgebaut
  - Nach der Übertragung wird diese Verbindung wieder abgebaut
- Unterschied Verbindungen auf der Transportschicht und der Netzwerkschicht:
  - *Netzwerkschicht*: zwischen zwei Hosts (dazwischenliegende Router können involviert sein)
  - *Transportschicht*: zwischen zwei Prozessen

## 4.1.2 Dienstmodelle der Netzwerkschicht

Unterschiedliche Anforderungen verlangen unterschiedliche Dienstmodelle:

- Beispiele für einzelne Datagramme:
  - Garantierte Zustellung
  - Garantierte Zustellung in weniger als 40 ms
- Beispiele für einen Fluss von Datagrammen:
  - Reihenfolgeerhaltende Auslieferung
  - Garantierte minimale Datenrate
  - Beschränkung der Schwankungen in der Zeit, die für den Transport von Datagrammen benötigt wird



## 4.1.2 Dienstmodelle der Netzwerkschicht

Netzwerk- architektur	Dienst- modell	Band- breiten- garantien	Garantie der Ver- lustfreiheit	Reihen- folge	Zeit- garantien	Hinweis auf Überlast
Internet	Best Effort	keine	nein	beliebige möglich	nicht unterstützt	keiner
ATM	CBR	garantiert eine kon- stante Rate	ja	in korrekter Reihenfolge	unterstützt	Überlast tritt nicht auf
ATM	ABR	garantier- tes Mini- mum	nein	in korrekter Reihenfolge	nicht unterstützt	Überlasthin- weise werden verwendet



## 4.2 Virtuelle Leitungen und Datagrammnetzwerke

### Verbindungsorientierter und verbindungsloser Netzwerkschichtdienst

- Ein Datagrammnetzwerk verwendet eine verbindungslose Netzwerkschicht
  - Ein Netzwerk mit virtuellen Leitungen verwendet eine verbindungsorientierte Netzwerkschicht
  - Analog zur Transportschicht. **Aber:**
    - ! **Dienst: Host-zu-Host (nicht Prozess-zu-Prozess)**
    - ! **Keine Wahlmöglichkeit: Ein Netzwerk bietet das eine oder das andere an**
    - ! **Implementierung: im Inneren des Netzwerkes**
- Achtung! In höheren Schichten kann immer noch ein verbindungsorientierter Dienst (z.B. TCP) über eine verbindungslose Vermittlungsschicht (z.B. IP) realisiert werden!

## 4.2.1 Netzwerke mit virtuellen Leitungen

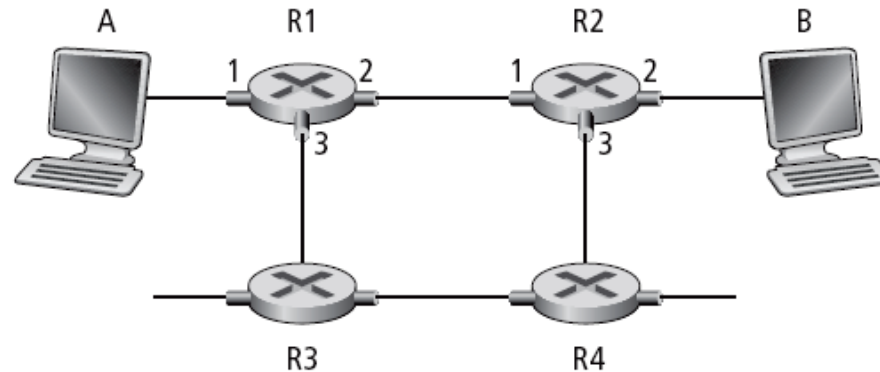
- Der Pfad zwischen Sender und Empfänger verhält sich wie eine Telefonleitung
- Aufbau einer Verbindung, bevor Daten transportiert werden können  
Danach: Abbau der Verbindung
  - Jedes Paket beinhaltet einen VC-Identifizier (Virtual Channel = Virtuelle Leitung) und keine Zieladresse
  - *Jeder Router* auf dem Pfad vom Sender zum Empfänger verwaltet einen Zustand für diese Verbindung
  - Ressourcen von Links und des Routers können einer virtuellen Leitung zugeordnet sein (zugeordnete Ressourcen = vorhersagbare Dienstgüte)

## 4.2.1 Implementierung virtueller Leitungen

Eine virtuelle Leitung besteht aus:

1. Einem Pfad vom Sender zum Empfänger
  2. VC-Identifizier, ein Identifizier für jeden Link entlang des Pfades
  3. Einträgen in den Weiterleitungstabellen der Router auf dem Pfad
- Pakete, die zu einem VC gehören, sind durch einen VC-Identifizier (nicht durch die Zieladresse!) gekennzeichnet
  - Der VC-Identifizier desselben Paketes kann von Link zu Link unterschiedlich sein
    - Die neuen VC-Identifizier stehen in der Weiterleitungstabelle der Router

## 4.2.1 Implementierung virtueller Leitungen

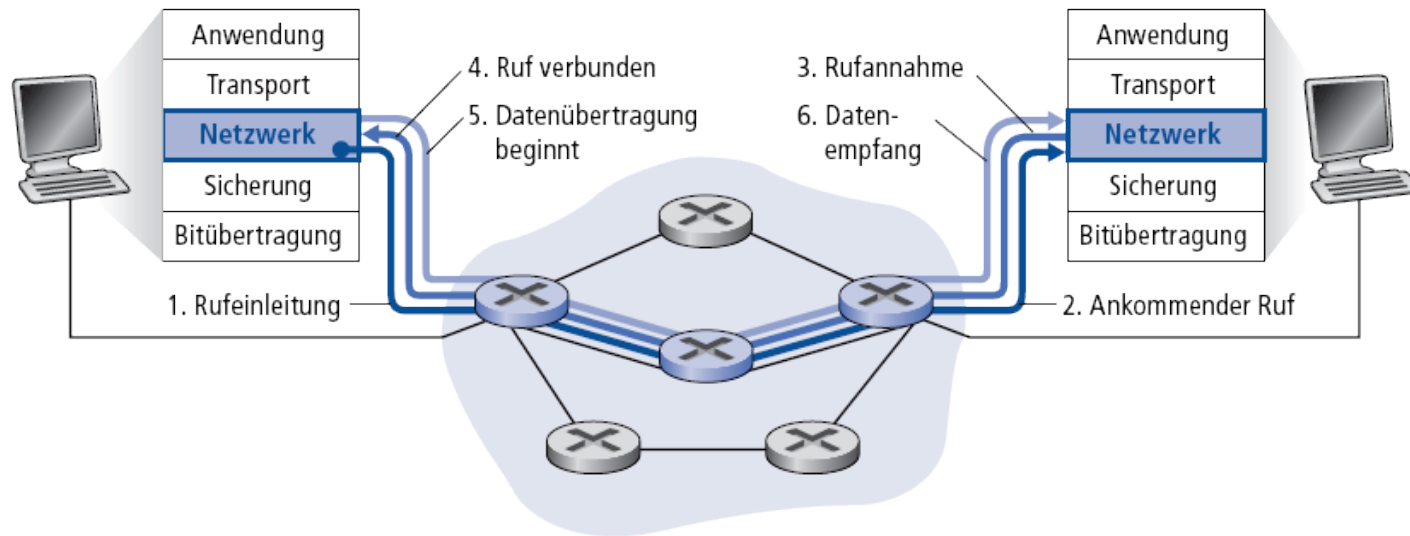


Weiterleitungstabelle in R1:

Eingehende Schnittstelle	Eingehende VC-Nummer	Ausgehende Schnittstelle	Ausgehende VC-Nummer
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

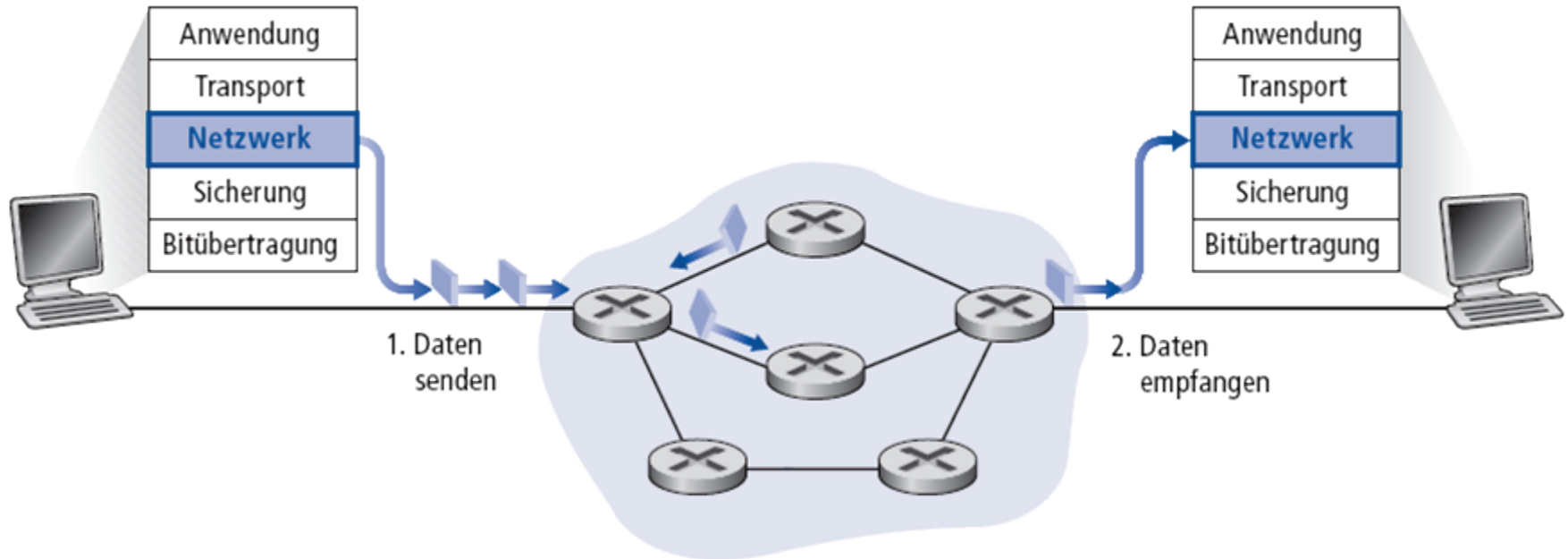
→ Router verwalten Zustand für jede Verbindung!

## 4.2.1 Virtuelle Leitungen: Signalisierungsprotokolle



- Verwendet zum Aufbau, Aufrechterhalten und Abbau von virtuellen Leitungen
- Verwendet in ATM, Frame Relay und X.25
- Nicht im Internet!

## 4.2.2 Datagrammnetzwerke



- Kein Verbindungsaufbau auf der Netzwerkschicht
- Router halten keinen Zustand für Ende-zu-Ende-Verbindungen
  - Auf Netzwerkebene gibt es das Konzept einer “Verbindung” nicht!
- Pakete werden unter Verwendung einer Zieladresse weitergeleitet
  - Pakete für dasselbe Sender-Empfänger-Paar können je nach Konfiguration in unterschiedlichen Richtungen einen unterschiedlichen Pfad nehmen

## 4.2.2 Weiterleitungstabelle eines Routers

Zieladressbereich	Schnittstelle
11001000 00010111 00010000 00000000	
bis	0
11001000 00010111 00010111 11111111	
11001000 00010111 00011000 00000000	
bis	1
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000	
bis	2
11001000 00010111 00011111 11111111	
sonst	3

## 4.2.2 Longest Prefix Matching

Passender Präfix	Schnittstelle
11001000 00010111 00010	0
11001000 0 0010111 00011000	1
11001000 00010111 00011	2
sonst	3

### Beispiele

Adresse: 11001000 00010111 00010110 10100001 → **Schnittstelle 0**

Passt zum 3. Eintrag! → Schnittstelle 2

Adresse: 11001000 00010111 00011000 10101010

**Passt zum 2. Eintrag! → Schnittstelle 1**

Gibt es mehrere Treffer, verwendet der Router die **Longest-Prefix-Matching-Regel** (längstes übereinstimmendes Präfix).



## 4.2.2 Datagramme und virtuelle Verbindungen

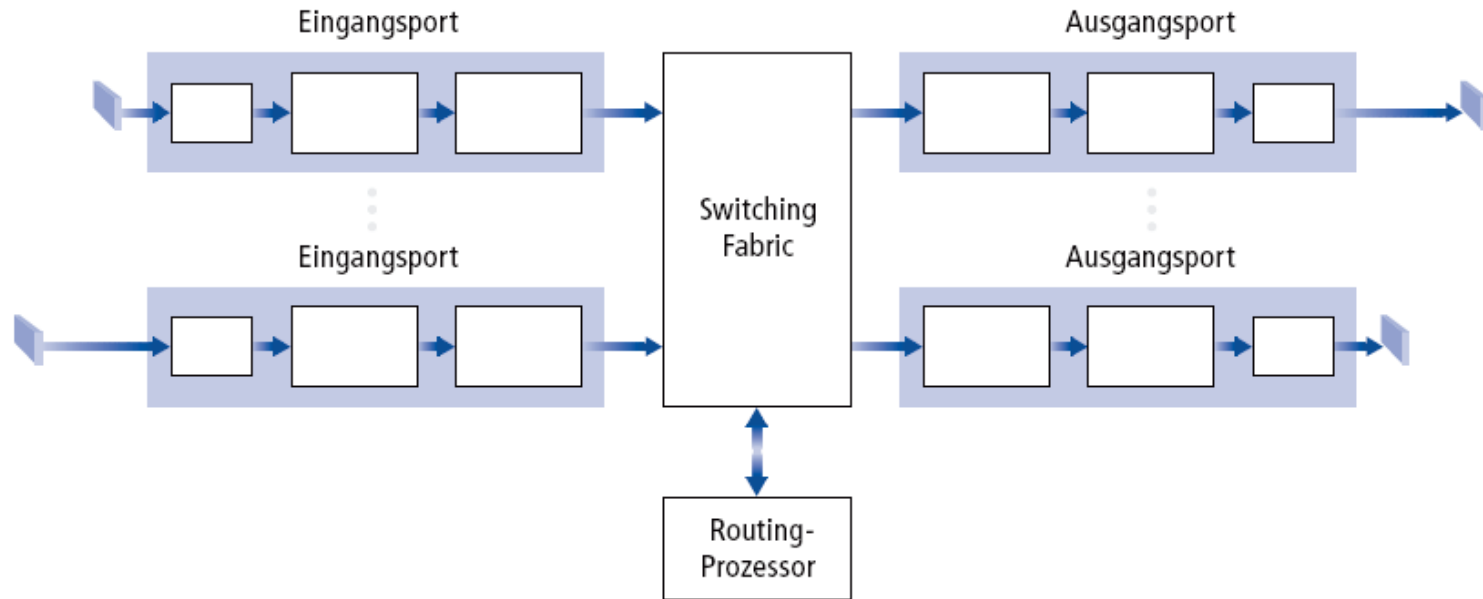
### Internet

- Datenaustausch zwischen Computern
  - Keine Echtzeitanforderungen
- Mächtige Endsysteme
  - Können sich anpassen und Fehler beheben
  - Konsequenz: einfaches Netzwerk, Komplexität in den Endsystemen
- Vielzahl verschiedener Links
  - Unterschiedliche Charakteristika
  - Einheitlicher Dienst schwierig zu realisieren

### ATM (Asynchronous Transfer Mode)

- Stammt von der klassischen Telefontechnologie ab
- Menschliche Kommunikation
  - Hohe Anforderungen an Echtzeit und Zuverlässigkeit
  - Dienstgarantien sind notwendig
- Einfache Endsysteme
  - Telefone
  - Konsequenz: einfache Endsysteme, Komplexität im Netzwerk

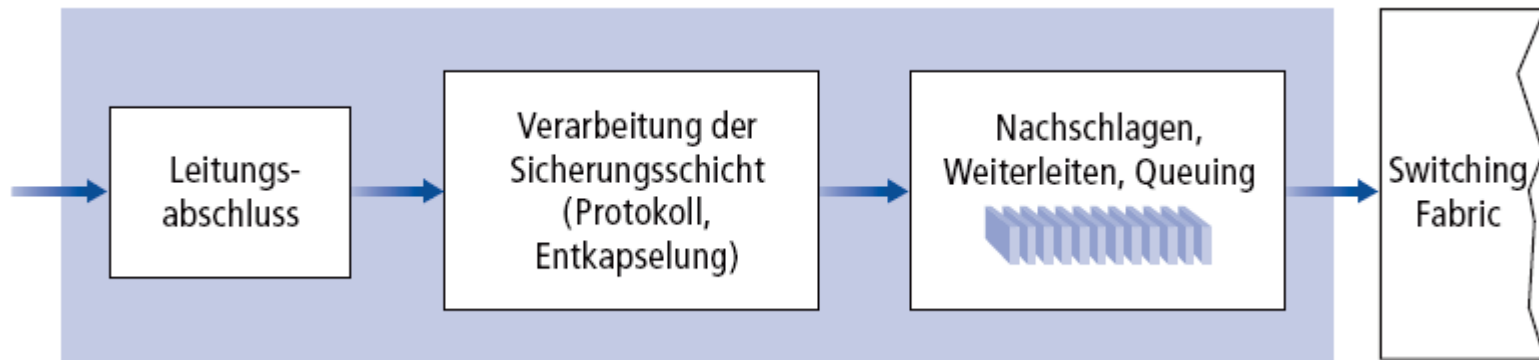
## 4.3 Router



Zwei wichtige Aufgaben eines Routers:

- Ausführen von Routing-Algorithmen und -Protokollen
  - RIP, OSPF, BGP
- Weiterleiten von Datagrammen von einem eingehenden zu einem ausgehenden Link

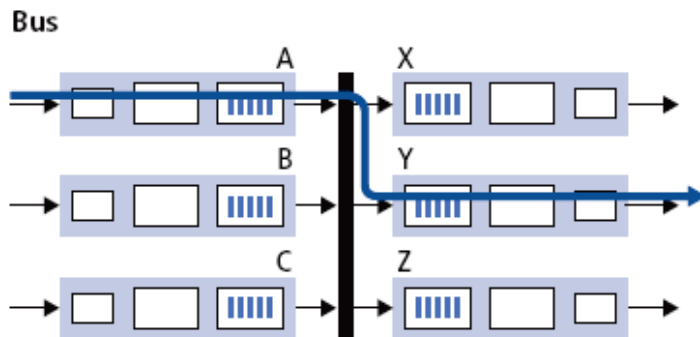
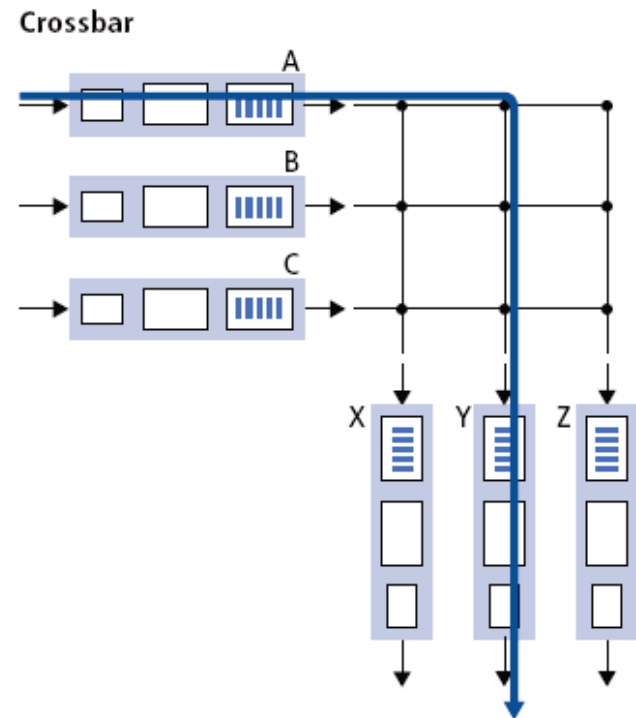
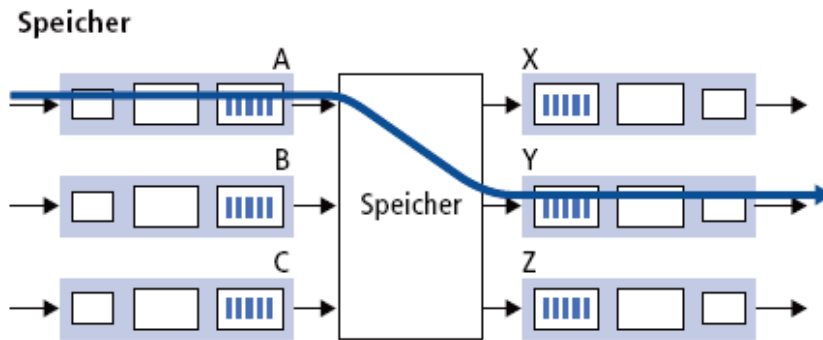
## 4.3.1 Eingangsports



- Leitungsabschluss: physikalische Schicht, Bits empfangen
- Sicherungsschicht: z.B. Ethernet (s. Kapitel 5)
- Nachschlagen, Weiterleiten, Queuing:
  - Suche nach einem geeigneten Ausgangsport
  - Dezentral, Kopie der Routing-Tabelle (oder Teile davon) notwendig
  - Ziel: Behandlung der Pakete mit „line speed“, also mit der Geschwindigkeit der Eingangsleitung des Ports
  - Puffern von Paketen, wenn die Switching Fabric belegt ist

# 4.3.2 Das Switching Fabric

## Drei Switching-Techniken:

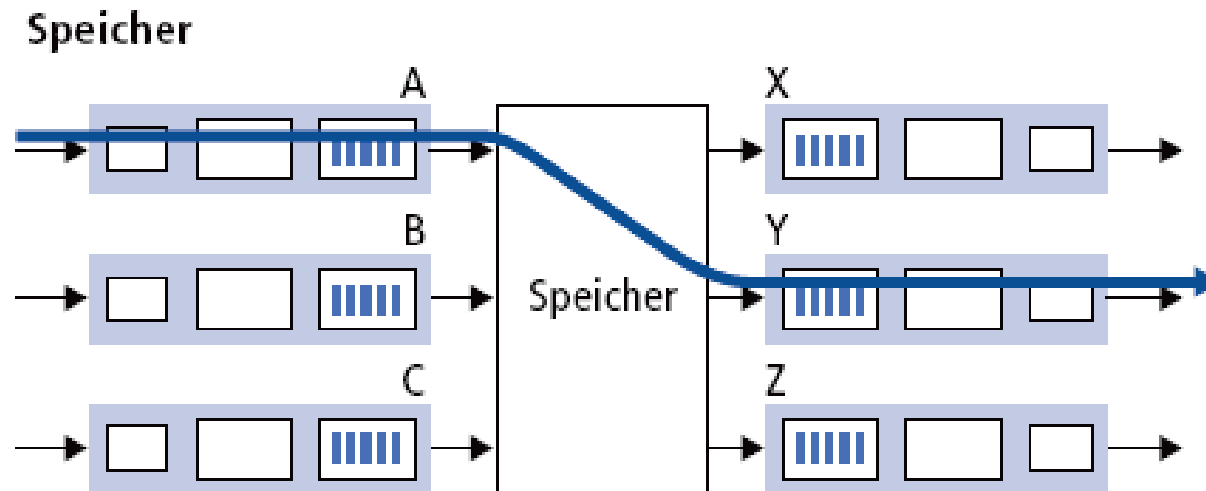


Legende:



## 4.3.2 Das Switching Fabric

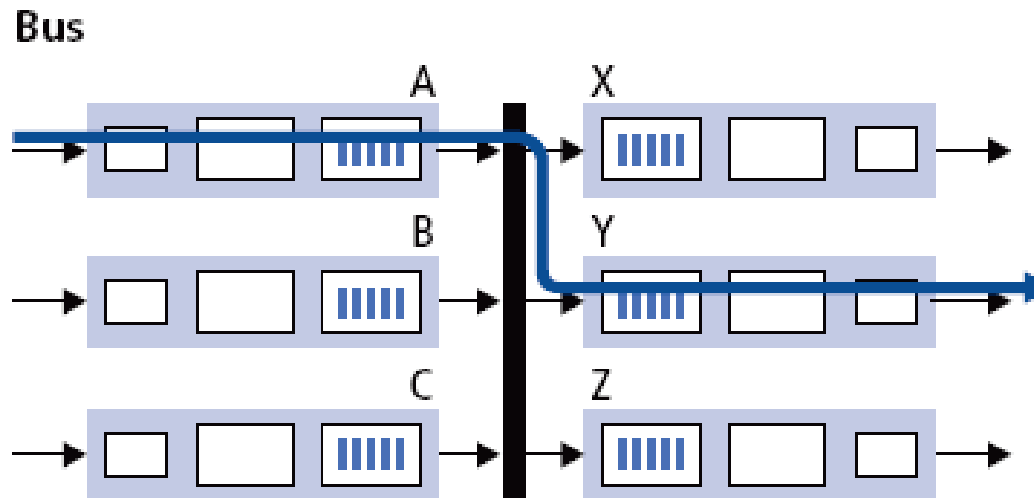
### 1. Switching über den Speicher:



- Erste Routergeneration:
  - „Normale“ Rechner, Switching wird über die CPU durchgeführt
  - Paket von Eingangsport in den Hauptspeicher kopieren
  - Paket vom Hauptspeicher in den Ausgangsport kopieren
  - Geschwindigkeit durch Speicherbus beschränkt!
    - Zwei Speicherzugriffe: einer zum Schreiben, einer zum Lesen

## 4.3.2 Das Switching Fabric

### 2. Switching über einen Bus:

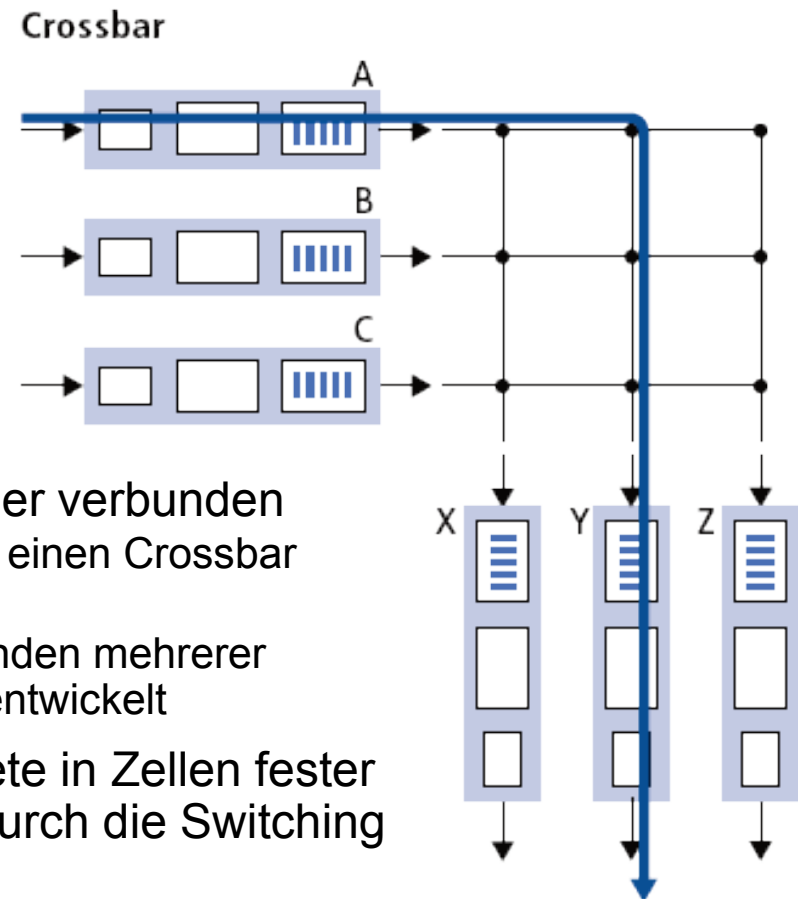


- Alle Ports teilen sich einen gemeinsamen Bus
- **Bus Contention:** Die gesamte Kommunikation erfolgt über den Bus, dieser beschränkt die Bandbreite des Routers
  - Aber: nur eine Busoperation (nicht zwei!)
  - *Beispiel:* 32-Gbps-Bus, Cisco 5600, ausreichend für Zugangsrouten und Router für Firmennetze (nicht geeignet im Backbone!)

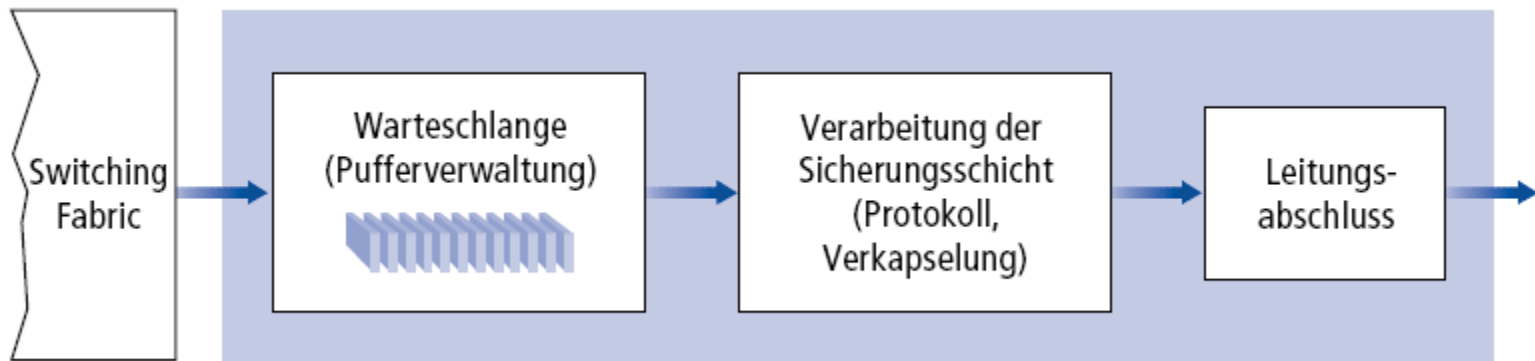
## 4.3.2 Das Switching Fabric

### 3. Switching über ein Spezialnetz:

- Ports sind über ein Netzwerk miteinander verbunden
  - Beispielsweise alle Eingangsports über einen Crossbar mit allen Ausgangsports
  - Technologie ursprünglich für das Verbinden mehrerer Prozessoren in einem Parallelrechner entwickelt
- Weitere Fortschritte: Zerlegen der Pakete in Zellen fester Größe, Zellen können dann schneller durch die Switching Fabric geleitet werden
  - *Beispiel:* Cisco 12000, Switching von 60 Gbps durch das interne Netz



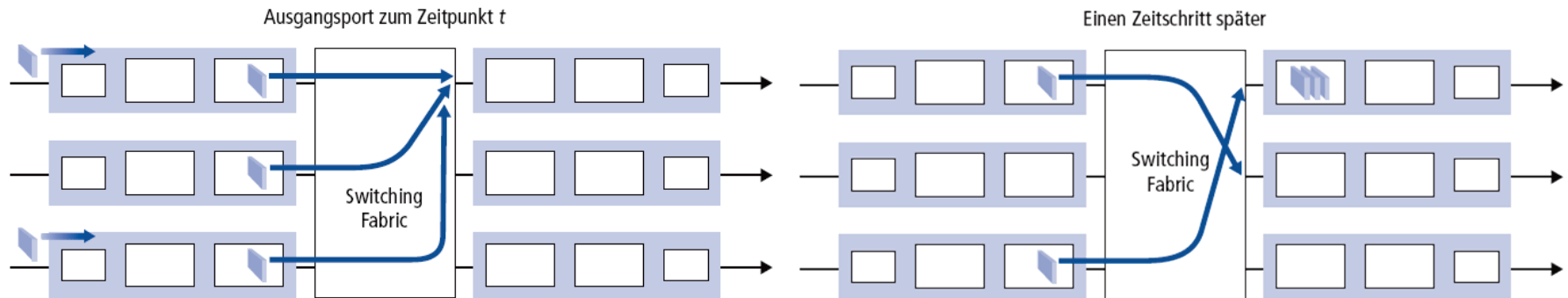
## 4.3.3 Ausgangsports



- Prinzipiell: analog zum Eingangsport!
- Einfacher, da die Entscheidung über die Weiterleitung schon getroffen ist



## 4.3.4 Puffern im Ausgangsport



- Puffern von Paketen, wenn sie schneller aus der Switching Fabric kommen, als sie auf die Leitung gelegt werden können
- Auswirkungen:
  - Gepufferte Pakete werden verzögert
  - Wenn der Puffer überläuft, müssen Pakete verworfen werden
- „**Scheduling Discipline**“: bestimmt die Reihenfolge, in der gepufferte Pakete auf die Leitung gelegt werden

## 4.3.4 Puffern im Ausgangsport

Wie groß sollten die Puffer sein?

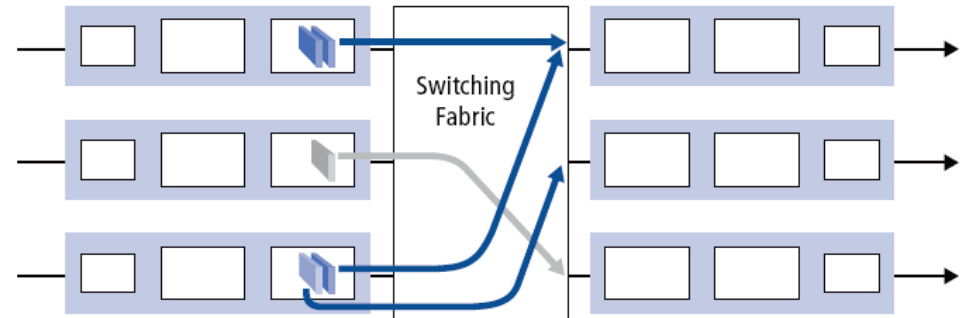
- [RFC 3439](#) beschreibt folgende **Faustregel**: Die Größe des Puffers sollte der Rundlaufzeit (RTT, z.B. 250 ms) multipliziert mit der Datenrate des Links entsprechen
  - Bei einem 10 Gps Link, 250 ms RTT, ergibt das 2,5 Gbit Puffer
- Neuere Empfehlungen: bei N Datenflüssen und Link-Datenrate C:

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

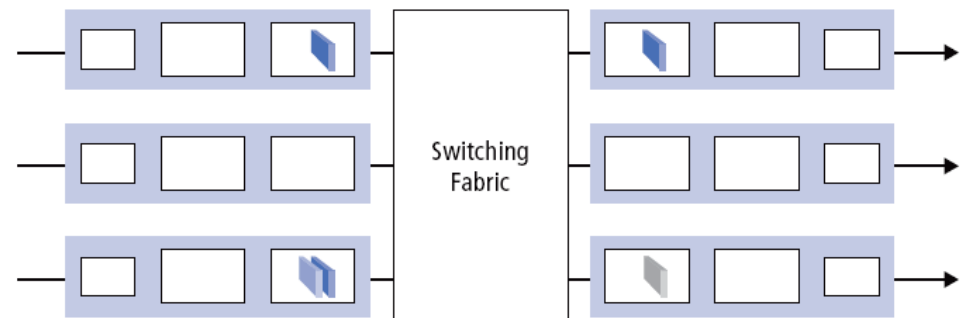
## 4.3.4 Puffern im Inputport

- Wenn die Switching Fabric ein Paket nicht direkt weiterleiten kann, muss dieses im Eingangsport gepuffert werden
- Dort kann es ein Paket blockieren, welches eigentlich bereits durch die Switching Fabric geleitet werden könnte → **Head-of-Line (HOL) Blocking**

Wettbewerb um den Ausgangsport zum Zeitpunkt  $t$  — ein dunkles Paket kann übertragen werden



Hellblaues Paket erfährt HOL-Blockade



Legende:

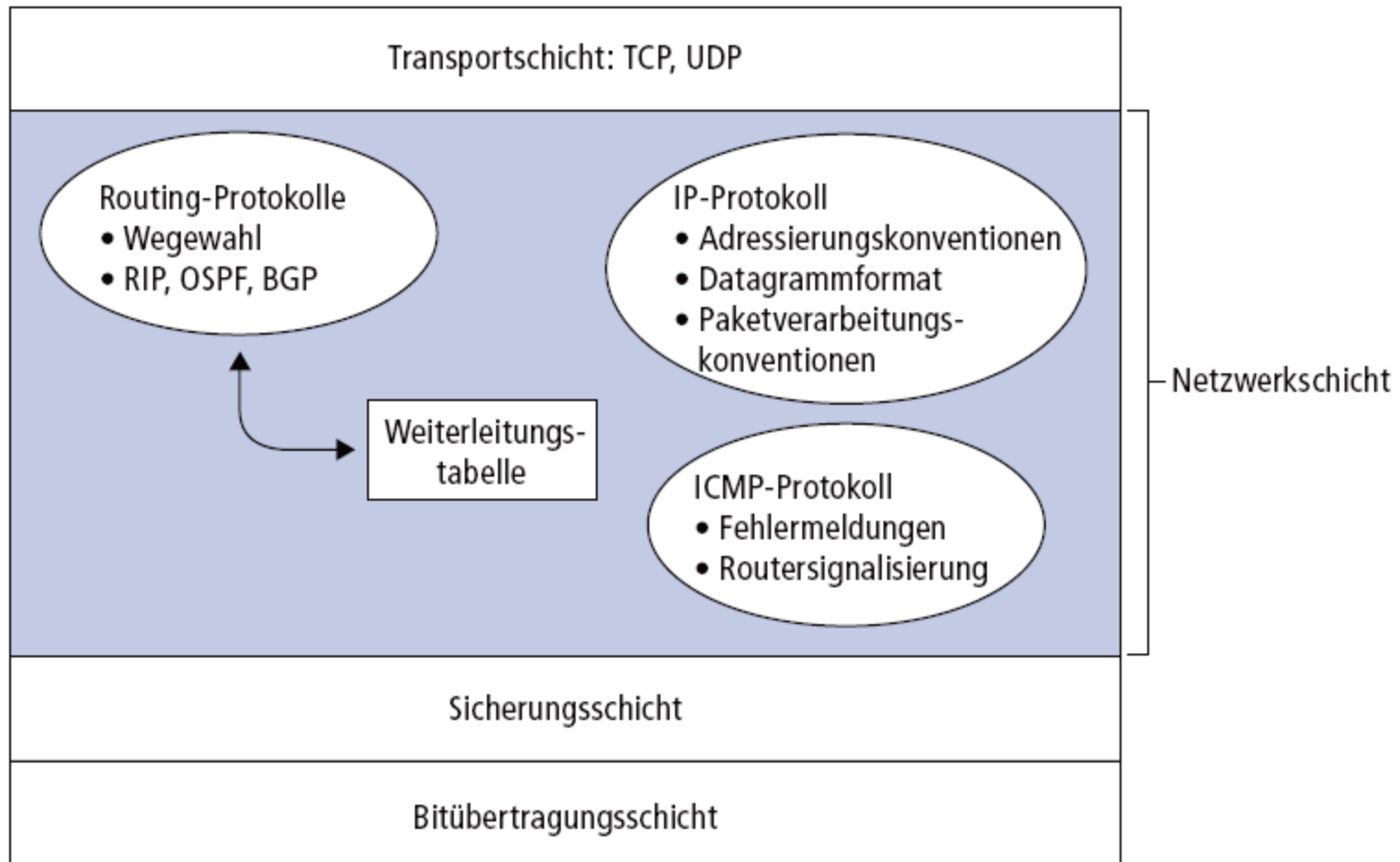
 Bestimmungsort ist der obere Ausgangsport

 Bestimmungsort ist der mittlere Ausgangsport

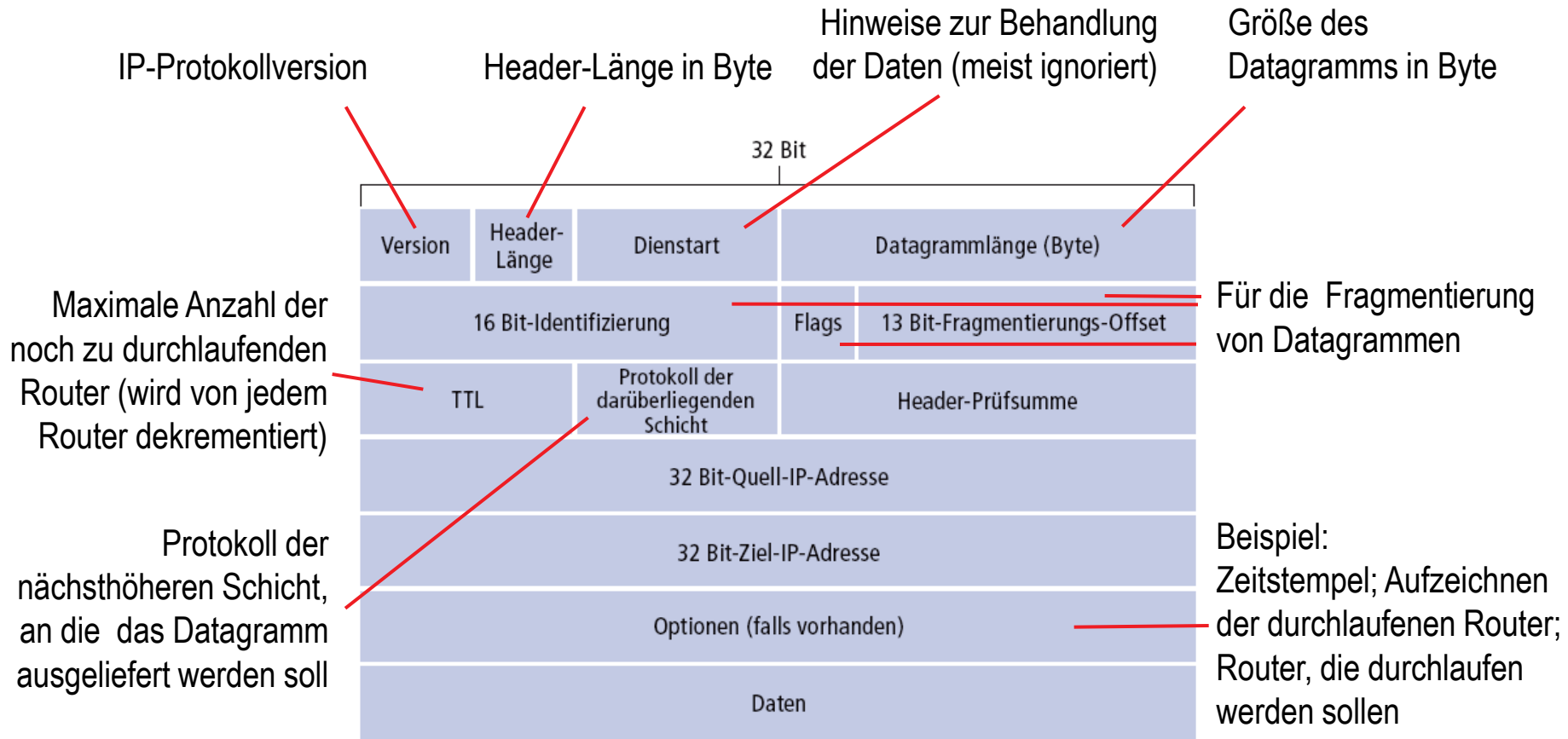
 Bestimmungsort ist der untere Ausgangsport

## 4.4 IP: Internetprotokoll

Die Netzwerkschicht des Internets:



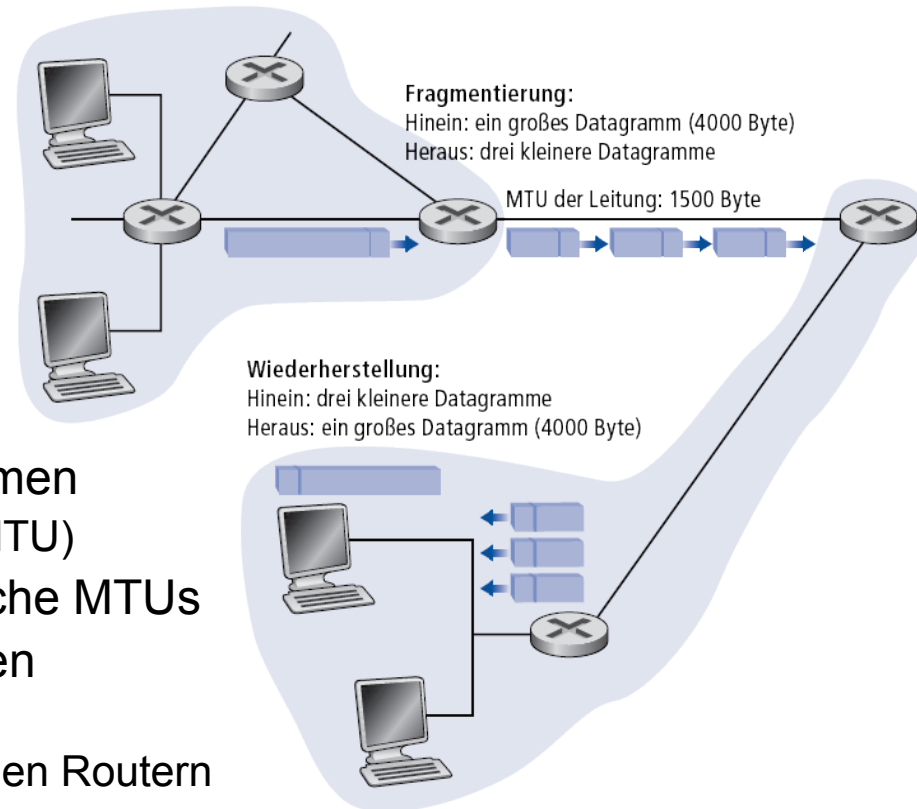
# 4.4.1 IP-Datagrammformat



**Wie viel Overhead entsteht bei Verwendung von TCP?**

→ 20 Byte für den TCP-Header, 20 Byte für den IP-Header= 40 Byte + Overhead auf der Anwendungsschicht

## 4.4.1 IP-Datagramm-Fragmentierung



- Links haben eine Maximalgröße für Rahmen
  - Genannt Maximum Transmission Unit (MTU)
- Verschiedene Links haben unterschiedliche MTUs
- IP-Datagramme müssen unter Umständen aufgeteilt werden
  - Aufteilung (Fragmentierung) erfolgt in den Routern
  - Zusammensetzen (Reassembly) erfolgt beim Empfänger
  - IP-Header enthält die notwendigen Informationen hierzu

## 4.4.1 IP-Datagramm-Fragmentierung

Beispiel: IP-Datagramm mit 4000 Byte (inklusive 20 Byte IP-Header),  
MTU des nächsten Links = 1500 Byte

Fragment	Bytes	ID	Offset	Flag
1. Fragment	1.480 Byte im Datenfeld des IP-Datagramms	Identifizierung = 777	Offset = 0 (d.h., die Daten sollten beginnend bei Byte 0 eingefügt werden)	Flag = 1 (d.h., da kommt noch mehr)
2. Fragment	1.480 Datenbytes	Identifizierung = 777	Offset = 185 (d.h., die Daten sollten bei Byte 1.480 beginnend eingefügt werden; beachten Sie, dass $185 \cdot 8 = 1.480$ )	Flag = 1 (d.h., da kommt noch mehr)
3. Fragment	1.020 Datenbytes (= $3.980 - 1.480 - 1.480$ )	Identifizierung = 777	Offset = 370 (d.h., die Daten sollten beginnend bei Byte 2.960 eingefügt werden; beachten Sie, dass $370 \cdot 8 = 2.960$ )	Flag = 0 (d.h., es ist das letzte Fragment)

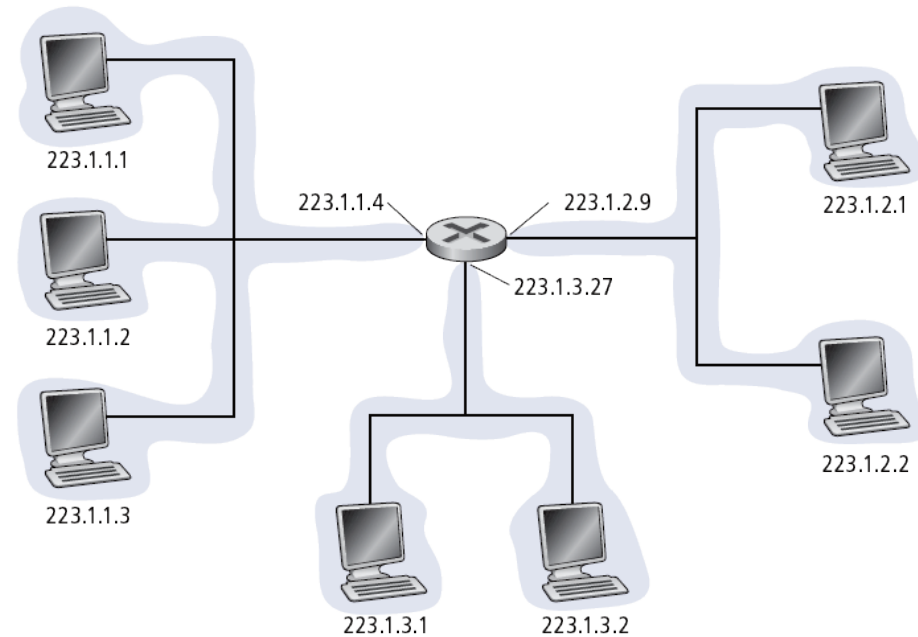
## 4.4.1 IP-Datagramm-Fragmentierung

- Praktisch:
  - Endsystem/Anwendung muss sich keine Gedanken über die Größe von MTUs verschiedener Links auf dem Weg vom Sender zum Empfänger machen
- Entspricht dem Prinzip einer geschichteten Architektur
- Aber:
  - Aufwand in den Routern
  - Wenn ein Fragment verloren geht, ist das ganze Datagramm verloren
- Daher: *Fragmentation considered harmful!*
- Lösung: Bestimmen der kleinsten MTU des Weges (Path MTU)
  - Setze **DF (Don't-Fragment-Bit)** im Header des IP-Paketes
  - Wenn fragmentiert werden soll, wird das Paket verworfen und der Sender per ICMP benachrichtigt
  - Sender wählt dann kleinere MTU
  - Wiederholen, bis akzeptable MTU gefunden wurde



## 4.4.2 IPv4-Adressierung

- IP-Adresse: 32-Bit-Kennung für das Interface (Schnittstelle) eines Endsystems oder eines Routers
- Interface: Verbindung zwischen dem System und dem Link
  - Wird normalerweise durch eine Netzwerkkarte bereitgestellt
  - Router haben typischerweise mehrere Interfaces
  - Endsysteme können ebenfalls mehrere Interfaces haben
  - Jedes Interface besitzt eine IP-Adresse

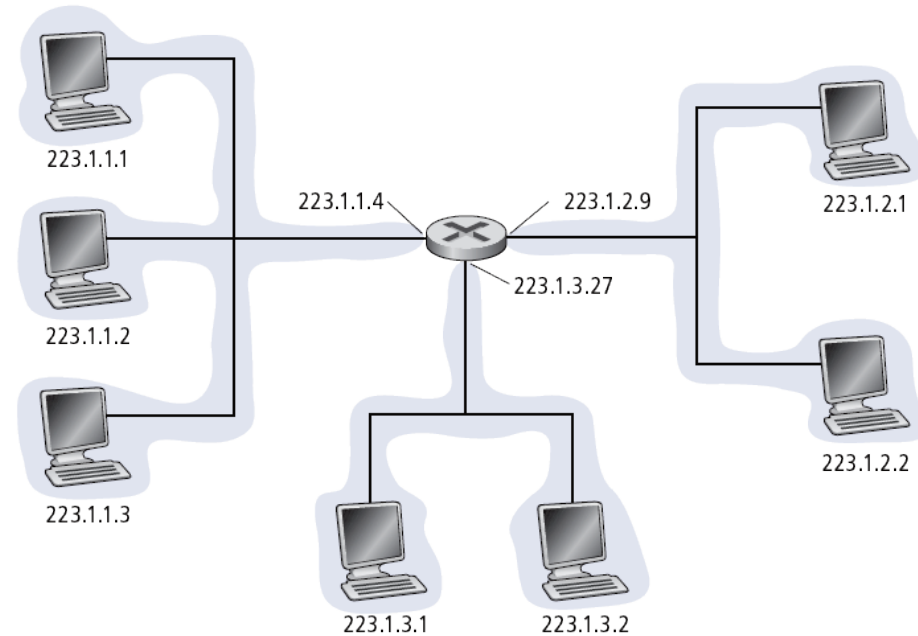


$$223.1.1.1 = \underline{11011111}, \underline{00000001}, \underline{00000001}, \underline{00000001}$$

223            1            1            1

## 4.4.2 IPv4-Adressierung

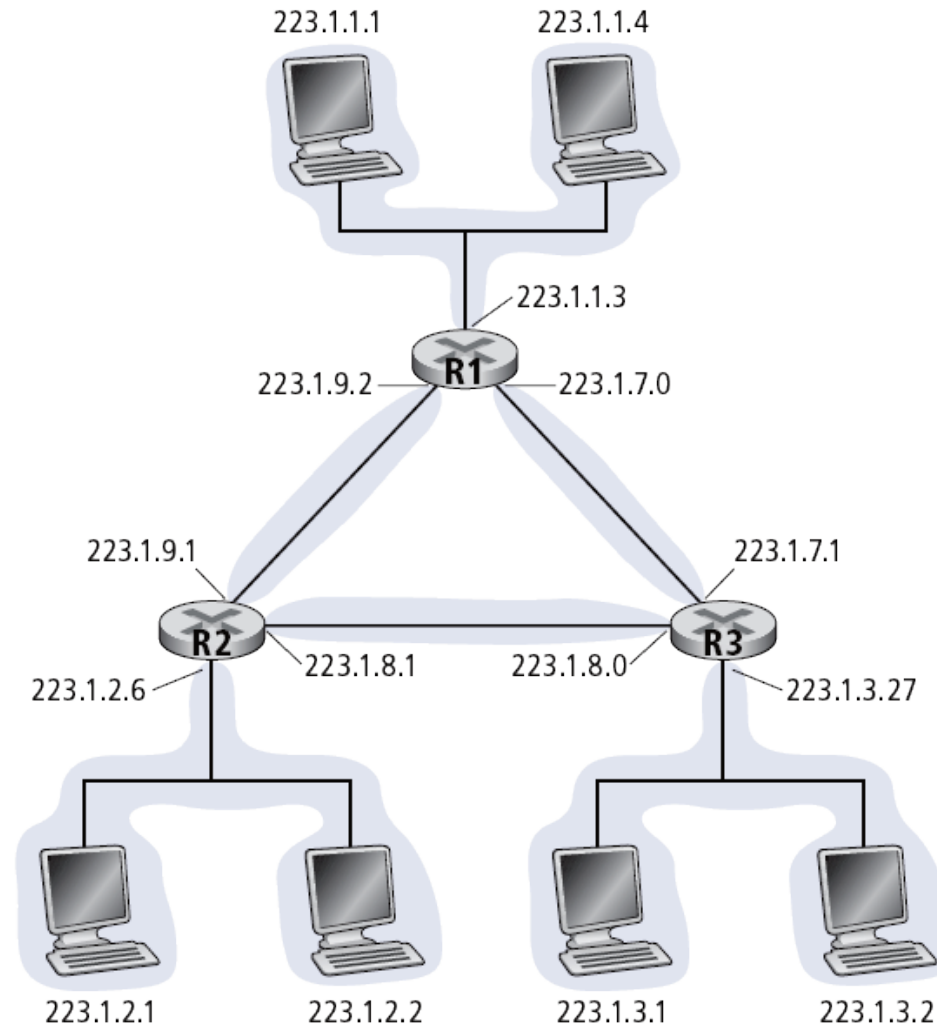
- IP-Adresse:
  - Zwei Bestandteile:
    - **netid**: die oberen Bits der Adresse, identifiziert ein Netzwerk
    - **hostid**: die unteren Bits der Adresse, identifiziert ein Interface eines Systems
- Was ist ein (Sub-)Netzwerk?
  - Alle Interfaces mit derselben netid formen ein Netzwerk
  - Alle Interfaces eines Netzwerkes können sich direkt (ohne einen Router zu durchqueren) erreichen



Drei IP-Netzwerke, die mit einem Router verbunden sind. Die netid steht hier in den oberen 24 Bit.

Subnetzmaske: /24 oder 255.255.255.0

## 4.4.2 IPv4-Adressierung

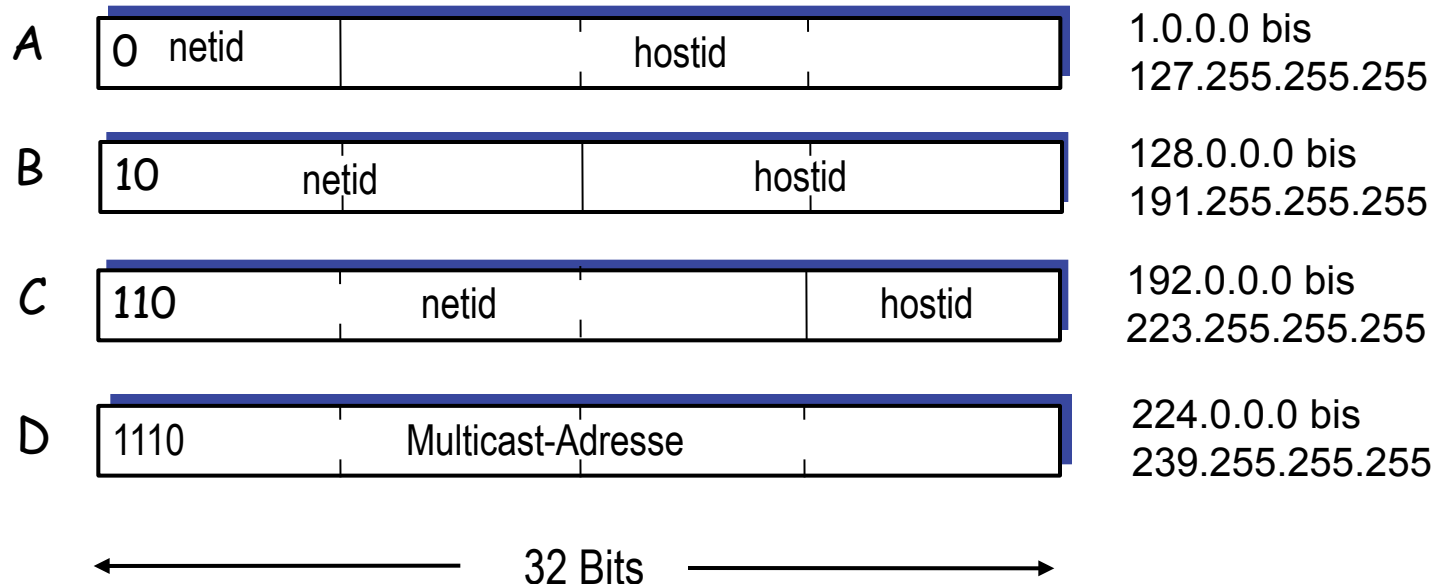


Drei Router verbinden sechs Subnetzwerke untereinander.

## 4.4.2 IP Adressierung - Adressklassen

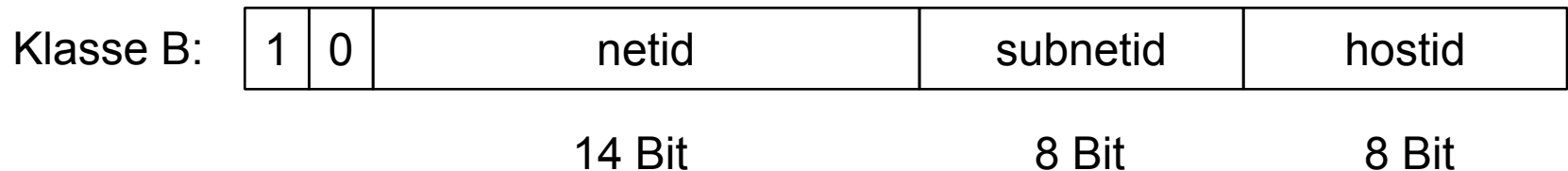
- Früher wurden IP-Adressen in Adressklassen aufgeteilt
- Die Klasse bestimmte das Verhältnis der Längen netid/hostid
- Dies nennt man „classfull“ addressing (klassenbasierte Adressierung)

Klasse



## 4.4.2 Adressierung von Subnetzen

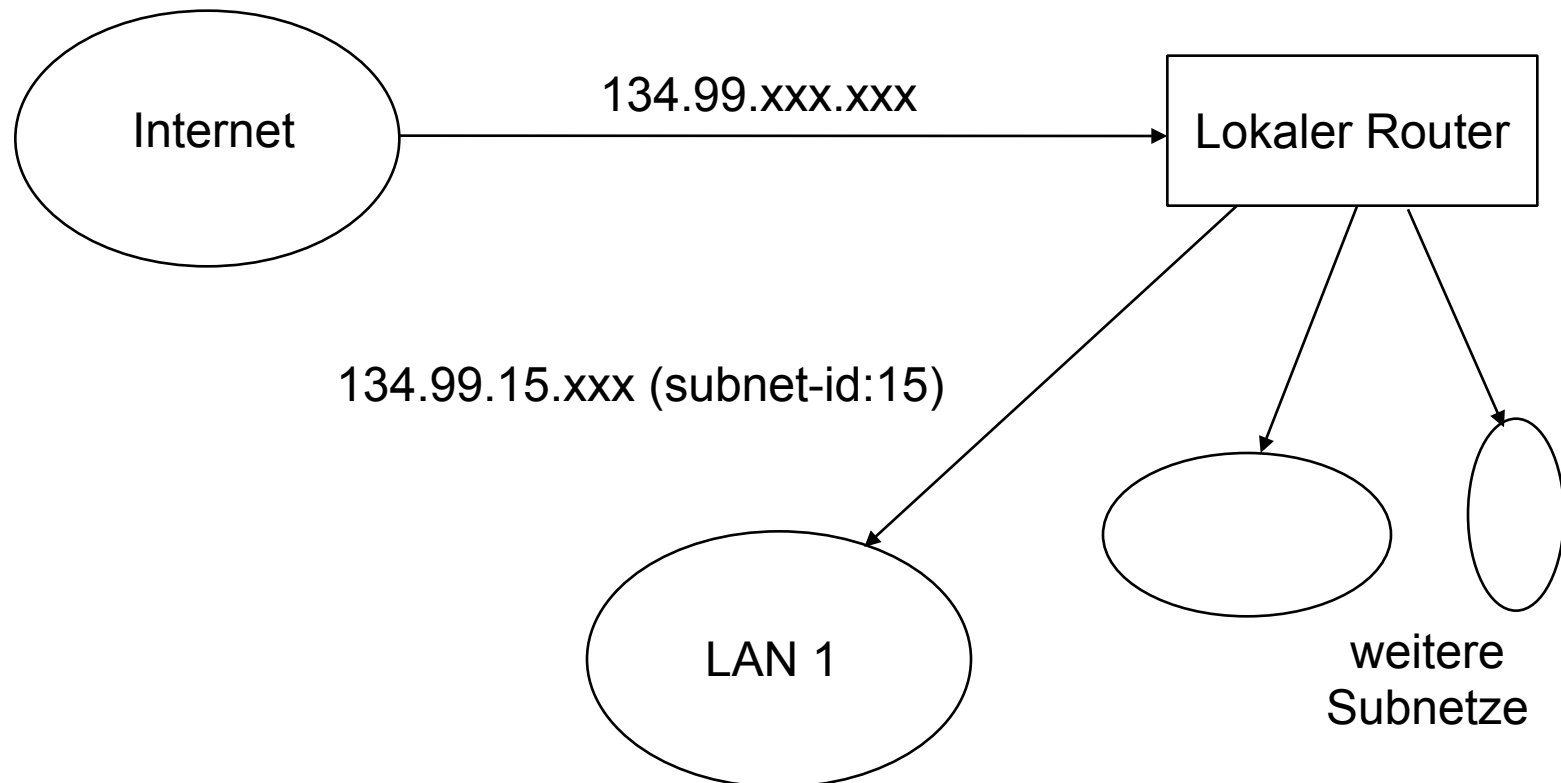
- Klasse-A- und -B-Adressen haben Platz für mehr Endsysteme, als man in einem Netzwerk sinnvoll unterbringen kann
- Daher teilt man die hostid weiter auf, z.B. so:



- Die Unterteilung (subnetid, hostid) ist eine lokale Entscheidung und wird von der Organisation vorgenommen, der die netid zugeordnet wurde

## 4.4.2 Adressierung von Subnetzen

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



## 4.4.2 Adressierung von Subnetzen

- Subnetzmaske (subnet mask)
  - Wird für jede IP-Adresse eines Systems im System gespeichert
  - Sie identifiziert, welcher Teil der Adresse zur subnetid und welcher zur hostid gehört

	16 Bit	8 Bit	8 Bit
Beispiel	1111111111111111	11111111	00000000

subnet mask: 0xfffff00=255.255.255.0  
oder auch /24

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

## 4.4.2 Beispiel für die Verwendung von Subnetzmasken

- Gegeben:
  - Eigene IP-Adresse: 134.155.48.10
  - Subnetzmaske: 255.255.255.0
  - Adresse A: 134.155.48.96, Adresse B: 134.155.55.96
- Überprüfen der beiden Adressen:
  - $134.155.48.10 \ \& \ 255.255.255.0 = 134.155.48.0$
  - $134.155.48.96 \ \& \ 255.255.255.0 = 134.155.48.0 \rightarrow$  identisch, gleiches Subnetz
  - $134.155.55.96 \ \& \ 255.255.255.0 = 134.155.55.0 \rightarrow$  verschieden, anderes Subnetz



## 4.4.2 Subnetzmasken variabler Länge

- Problem: Gegeben sei ein Klasse-C-Netzwerk, welches in zwei Subnetze mit 50 Endsystemen und ein Subnetz mit 100 Endsystemen unterteilt werden soll.
- Das funktioniert nicht mit einer einzelnen Subnetzmaske!
  - 255.255.255.128: zwei Netze mit je 128 hostids
  - 255.255.255.192: vier Netze mit je 64 hostids
- Lösung: Subnetzmasken variabler Länge
  - Unterteile den Adressraum zunächst mit der kürzeren Subnetzmaske (1 Bit im Beispiel)
  - Unterteile eine Hälfte davon weiter mit der längeren Subnetzmaske (2 Bit im Beispiel)
  - Resultat: Subnetze verschiedener Größe

## 4.4.2 Subnetzmasken variabler Länge

### Beispiel:

- Klasse-C-Netzwerk: 193.43.55.x
- Subnetzmaske für das Subnetz mit der ID 0 (100 Endsysteme):
  - 255.255.255.128
  - Adressen in diesem Subnetz: 193.43.55.0–127
- Subnetzmaske für die Subnetze mit den IDs 2 und 3 (je 50 Endsysteme):
  - 255.255.255.192
  - Adressen im Subnetz 2: 193.43.55.128–191
  - Adressen im Subnetz 3: 193.43.55.192–255

## 4.4.2 Klassenbasierte Adressierung

- Verteilung der Adressen:
  - Durch zentrale Organisationen (z.B. IANA)
  - Netzweise (also z.B. Klasse-B-Netz für ein Unternehmen)
  - Relativ chaotisch:
    - Zuteilung der numerisch nächsten netid an den nächsten Nachfrager
- Probleme:
  - Verschwendung von IP-Adressen:
    - Das Unternehmen könnte  $2^{16}$ , also mehr als 65.000 Adressen vergeben
    - Nur ein Bruchteil davon wird genutzt
  - *Classful* Routing-Tabellen werden schnell sehr groß:
    - Ein separater Eintrag für eine jede netid der Klasse C, auch wenn mehrere Klasse C Netze zusammengefasst werden könnten!
      - Routing-Tabellen müssen mit hoher Frequenz aufgefrischt werden
        - » Immer, wenn ein Klasse C Netzwerk hinzukommt, wegfällt oder sich verändert, muss dies im ganzen Internet bekanntgegeben werden

## 4.4.2 Klassenlose Adressierung

- Idee:
  - Die Aufteilung nach netid/hostid wird immer explizit per Subnetzmaske durchgeführt
  - Keine explizite Unterscheidung zwischen netid und subnetid
- Schreibweise:
  - a.b.c.d/x, wobei x die Länge der netid (Prefix) bestimmt
  - Alternative Schreibweise zur Subnetzmaske
  - Beispiel:
    - 192.48.96.0/23
    - 11000000.00110000.01100000.00000000 (netidhostid)

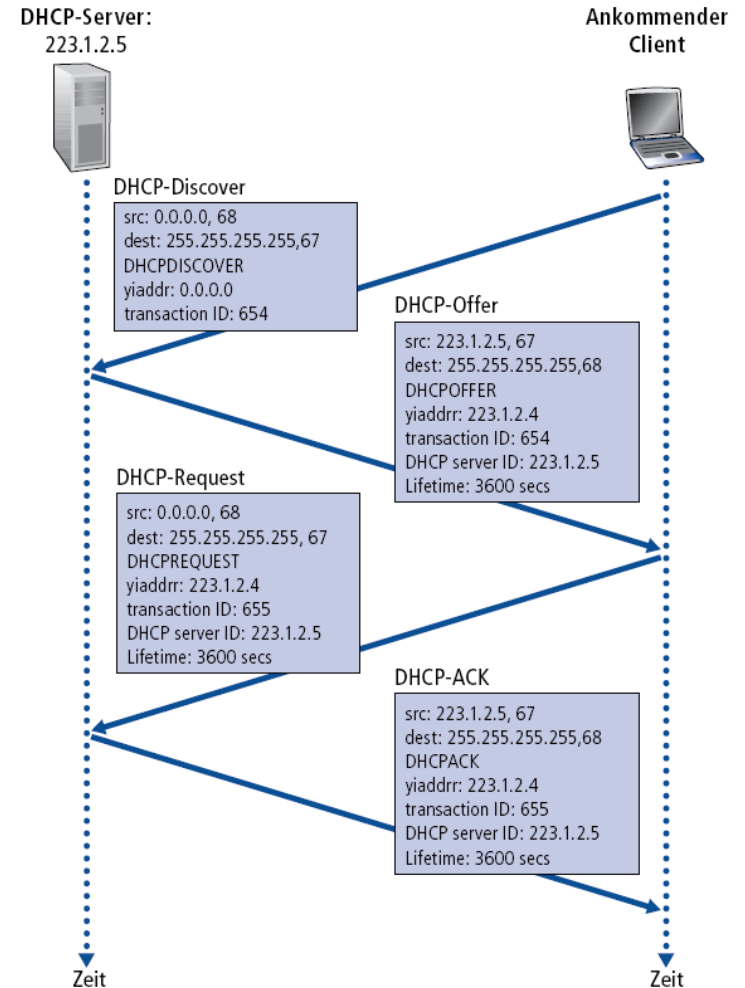
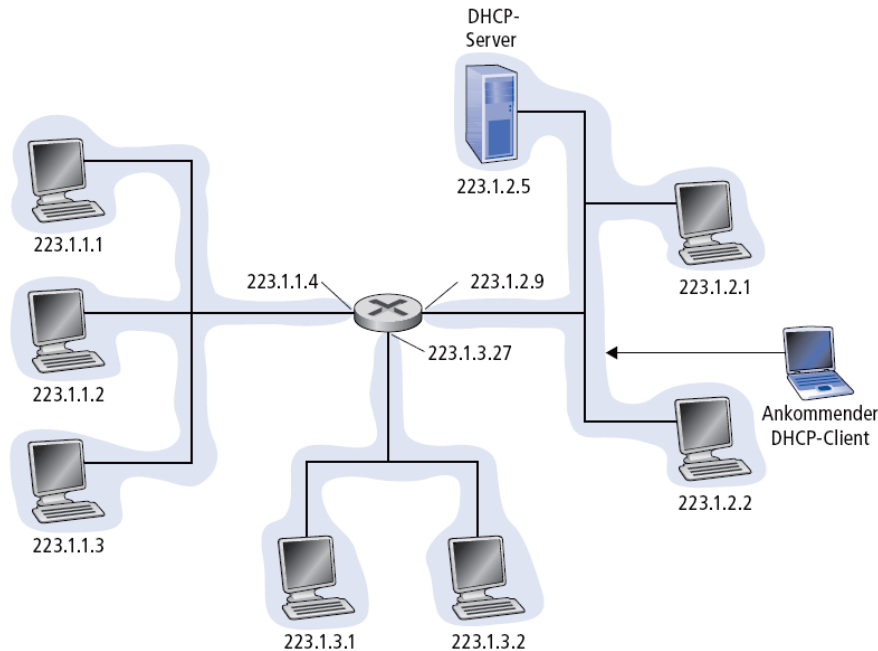
## 4.4.2 Adressvergabe bei Hosts - DHCP

- Bei Hosts erfolgt die Adressvergabe durch die Konfiguration von:
    - IP-Adresse
    - Subnetzmaske
    - Weiteren Parametern
  - Problem: Wie bekommen Endsysteme ihre IP-Adresse (und andere Parameter der Netzwerkschicht, wie z.B. die Subnetzmaske)?
    - Manuelle Konfiguration ist fehleranfällig
    - Lösung: Automatische Zuweisung mithilfe von DHCP [[RFC 2131](#)]
- **DHCP**: Dynamic Host Configuration Protocol: dynamisches Beziehen der Adresse von einem Server
- “Plug-and-Play”

## 4.4.2 DHCP

- Ziele von DHCP:
  - Automatische Vergabe von Adressen und Parametern
  - Keine Konfiguration der Endsysteme notwendig
  - Unterstützung von „nomadischen“ Benutzern
- Prinzipieller Ablauf
  - Endsystem schickt eine **DHCP-Discover-Nachricht** per IP-Broadcast (Adresse 255.255.255.255)
  - DHCP-Server antwortet mit einer **DHCP-Offer-Nachricht**
  - Endsystem beantragt eine IP-Adresse: **DHCP-Request-Nachricht**
  - DHCP-Server vergibt Adresse: **DHCP-Ack-Nachricht**
- Funktionen von DHCP [definiert in [RFC 2131](#)]:
  - Verschiedene Arten der Adresszuweisung (manuell, permanent, temporär)
  - Verlängerung und Rückgabe der Adresse durch den Client
  - Konfigurieren von Parametern

## 4.4.2 DHCP Szenario



- DHCP verwendet UDP
- DHCP-Nachrichten werden an die MAC-Broadcast-Adresse geschickt
- Es gibt ein Feld, in dem eine eindeutige Kennung des Clients verpackt ist → dies ist meist die MAC-Adresse.

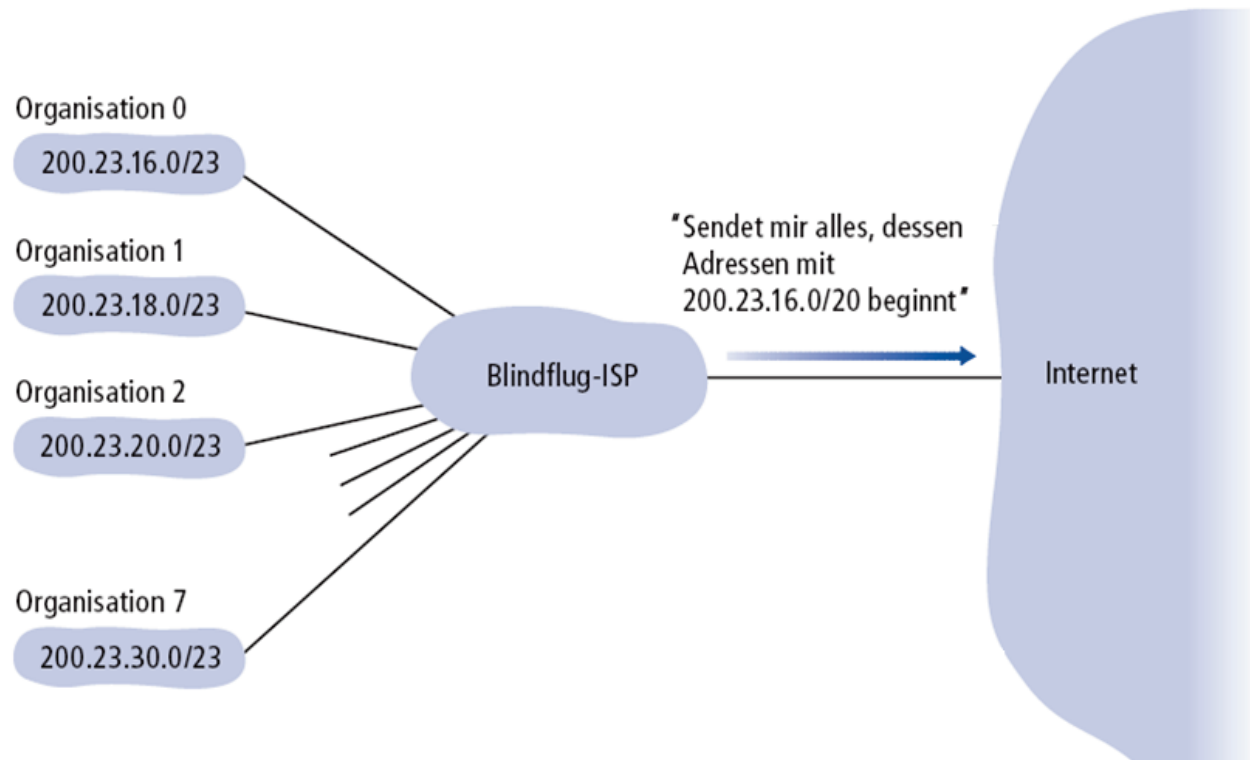
## 4.4.2 Adressvergabe bei Netzwerken

- Bei der klassenlosen Adressierung (CIDR) werden zusammenhängende Adressbereiche von der **Internet Assigned Numbers Authority (IANA)** an die **Regional Internet Registries (RIR)** vergeben
  - APNIC (Asia Pacific Network Information Centre) - Asien/Pazifik
  - ARIN (American Registry for Internet Numbers) - Nordamerika und Afrika südlich der Sahara
  - LACNIC (Regional Latin-American and Caribbean IP Address Registry) – Lateinamerika und einige karibische Inseln
  - RIPE NCC (Réseaux IP Européens) - Europa, Mittlerer Osten, Zentralasien und afrikanische Länder nördlich des Äquators
- Diese vergeben zusammenhängende Adressbereiche an ISPs
- ISPs vergeben zusammenhängende Adressbereiche an ihre Kunden
  - Dadurch: Aggregation in Routing-Tabellen teilweise möglich



## 4.4.2 Adressvergabe bei Netzwerken

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



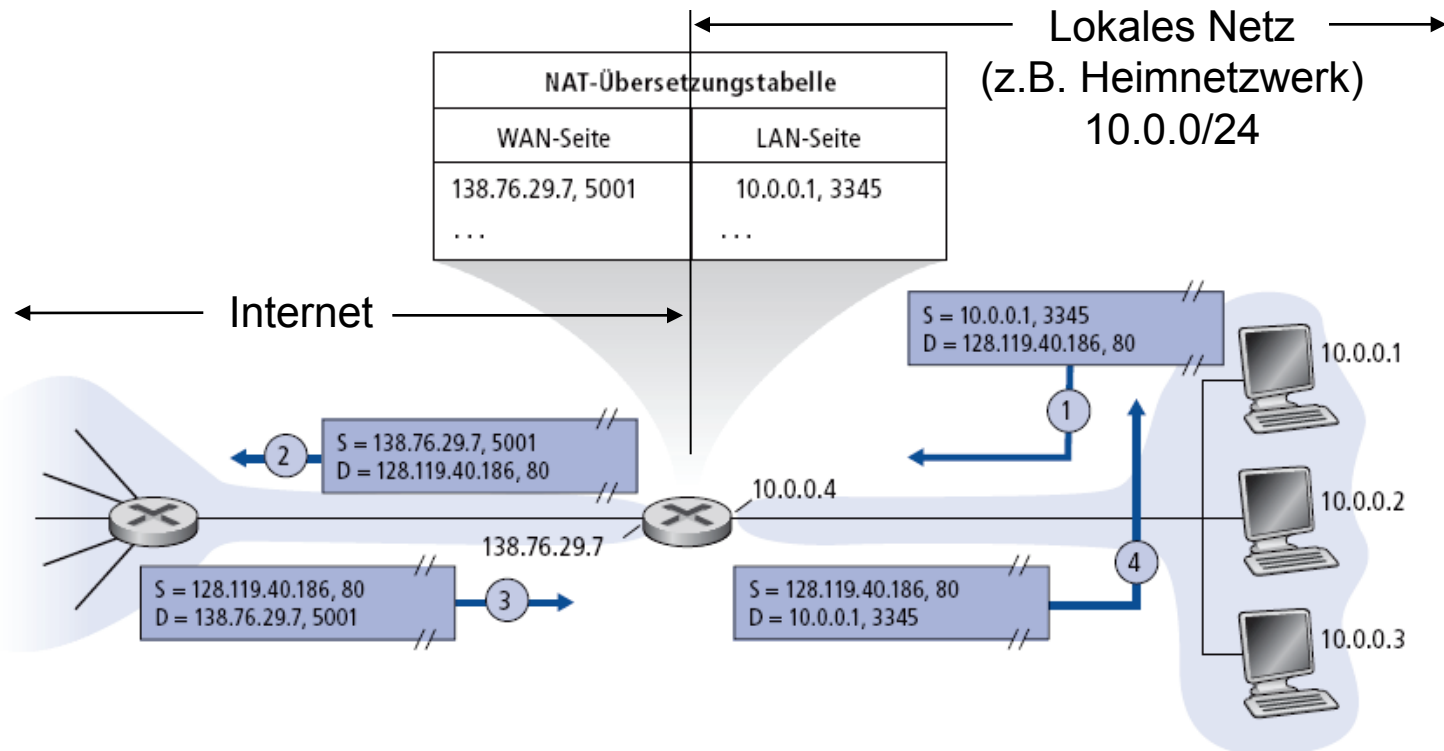
## 4.4.2 Adressvergabe bei Netzwerken

Address Block	Present Use	Reference
0.0.0.0/8	"This" Network	[RFC1700, page 4]
10.0.0.0/8	Private-Use Networks	[RFC1918]
14.0.0.0/8	Public-Data Networks	[RFC1700, page 181]
24.0.0.0/8	Cable Television Networks	---
39.0.0.0/8	Reserved but subject to allocation	[RFC1797]
127.0.0.0/8	Loopback	[RFC1700, page 5]
128.0.0.0/16	Reserved but subject to allocation	---
169.254.0.0/16	Link Local	---
172.16.0.0/12	Private-Use Networks	[RFC1918]
191.255.0.0/16	Reserved but subject to allocation	---
192.0.0.0/24	Reserved but subject to allocation	---
192.0.2.0/24	Test-Net	---
192.88.99.0/24	6 to 4 Relay Anycast	[RFC3068]
192.168.0.0/16	Private-Use Networks	[RFC1918]
198.18.0.0/15	Network Interconnect Device Benchmark Testing	[RFC2544]
223.255.255.0/24	Reserved but subject to allocation	---
224.0.0.0/4	Multicast	[RFC3171]
240.0.0.0/4	Reserved for Future Use	[RFC1700, page 4]

## 4.4.2 Network Address Translation (NAT)

- Motivation:
  - Häufig hat man nur eine IP-Adresse, aber mehrere Endsysteme
  - Diese ist meist nur temporär (per DHCP) zugewiesen
  - Man möchte bei einem Provider-Wechsel nicht die IP-Adressen der Endsysteme verändern
  - IP-Adressen im eigenen Netzwerk sollen aus Sicherheitsgründen nicht vom Internet aus sichtbar sein
  - Interne IP-Adressen sollen veränderbar sein, ohne dass der Rest des Internets darüber informiert werden muss
- Idee:
  - Vergebe lokale (weltweit nicht eindeutige) Adressen an die Systeme im eigenen Netzwerk
  - Router zur Anbindung an das Internet übersetzt diese Adressen in eine gültige, weltweit eindeutige IP-Adresse
  - Dazu wird die Adressierung auf der Transportschicht gebraucht: Ports

## 4.4.2 Network Address Translation (NAT)



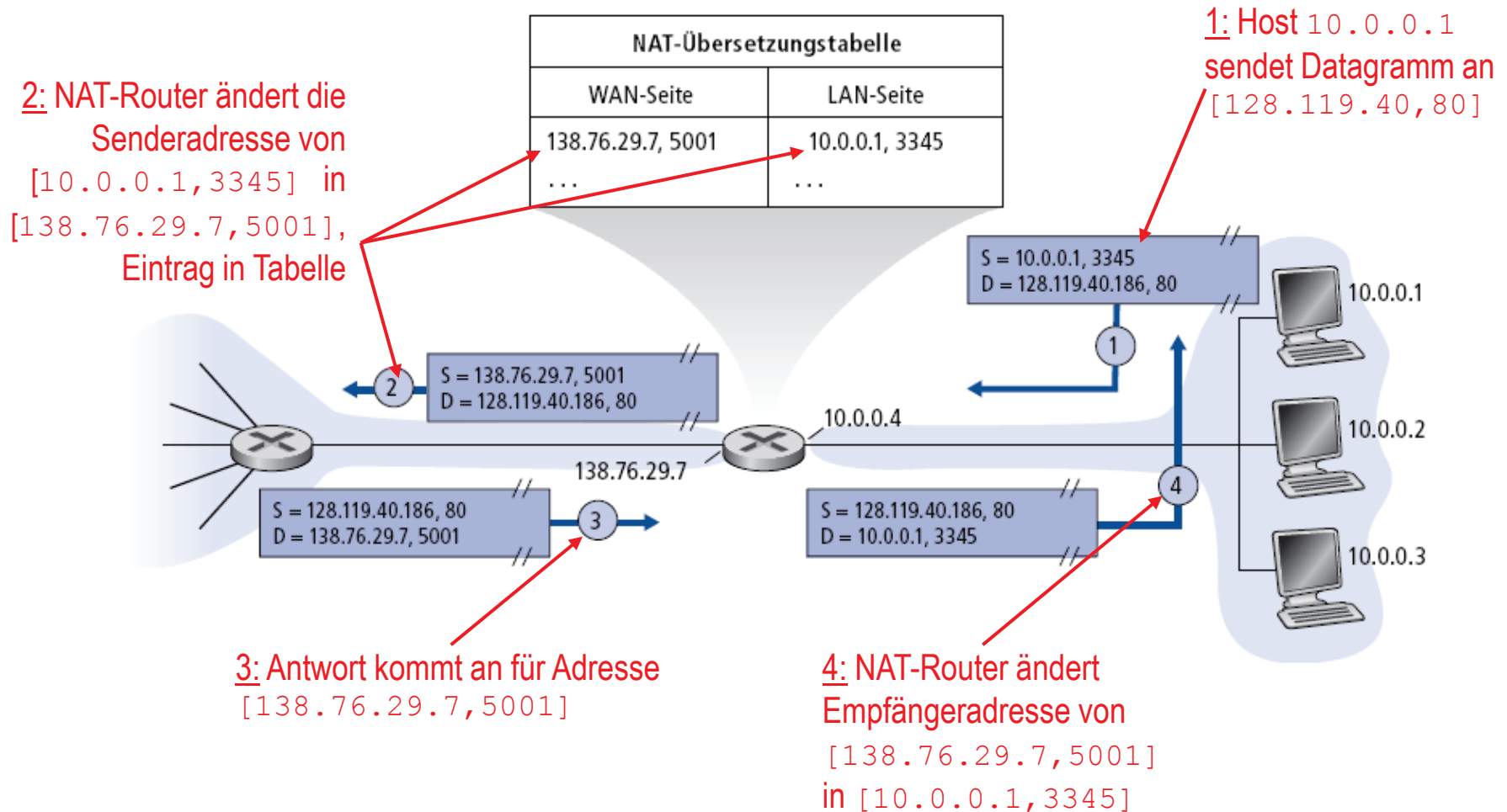
Alle Datagramme die das lokale Netz *verlassen*, haben die gleiche NAT-IP-Adresse als Absender: 138.76.29.7, unterschieden werden sie über die Portnummern.

→ Datagramme mit Sender oder Empfänger in diesem Netzwerk haben 10.0.0/24 als Adresse für diesen Sender/Empfänger.

## 4.4.2 Network Address Translation (NAT)

- Implementierung:  
Ein NAT-Router muss Folgendes tun:
  - *Ausgehende Datagramme:*  
Ersetze [Sender-IP-Adresse, Portnummer] im Absenderfeld für jedes ins Internet geleitete Datagramm durch [NAT-IP-Adresse, neue Portnummer]  
→ Kommunikationspartner wird die Antworten an [NAT-IP-Adresse, neue Portnummer] schicken
  - Speichere in einer NAT-Tabelle die Abbildung zwischen [Sender-IP-Adresse, Portnummer] und [NAT-IP-Adresse, neue Portnummer]
  - *Ankommende Datagramme:*  
Ersetze [NAT-IP-Adresse, neue Portnummer] im Empfängerfeld durch [Sender-IP-Adresse, Portnummer] aus der NAT-Tabelle

## 4.4.2 Network Address Translation (NAT)



## 4.4.2 Network Address Translation (NAT)

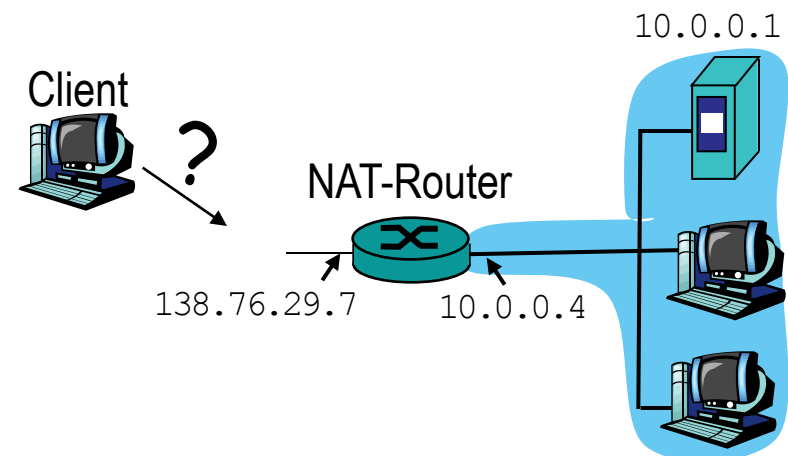
- 16-Bit-Port Number-Feld:
  - Mehr als 60.000 gleichzeitige Verbindungen mit einer IP-Adresse sehr einfach möglich
- NAT ist nicht unumstritten:
  - Router sollten nur Informationen der Schicht 3 verwenden
  - Verletzung des End-to-End Principle (Ende-zu-Ende Prinzip):
    - Transparente Kommunikation von Endsystem zu Endsystem nicht möglich, da im Netz die Adressen durch die NATs übersetzt werden!
    - Der Anwendungsentwickler muss die Präsenz von NAT-Routern berücksichtigen
      - Beispiel: Verwenden der Host-IP-Adresse als weltweit eindeutige Nummer nicht möglich!
  - NAT dient hauptsächlich der Bekämpfung der Adressknappheit im Internet. Dies sollte besser über IPv6 (mehr dazu später) erfolgen!

## 4.4.2 NAT-Traversal

→ NAT-Traversal nennt man das Durchqueren von Routern, die NAT einsetzen.

Angenommene Situation:

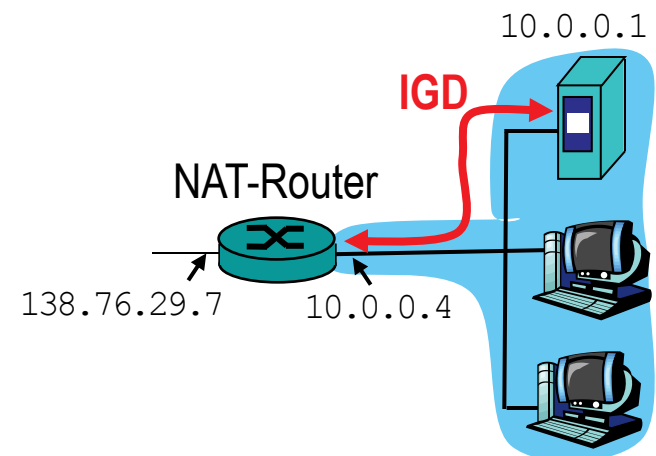
- Der Client möchte den Server mit der Adresse 10.0.0.1 kontaktieren
  - Die Adresse 10.0.0.1 ist eine lokale Adresse und kann nicht als Adresse im globalen Internet verwendet werden
  - Die einzige nach außen sichtbare Adresse ist: 138.76.29.7
- **Lösung 1:** Statische Konfiguration von NAT, so dass eingehende Anfragen angemessen weitergeleitet werden
  - Beispiel: [123.76.29.7, Port 2500] wird immer an [10.0.0.1, Port 25000] weitergeleitet





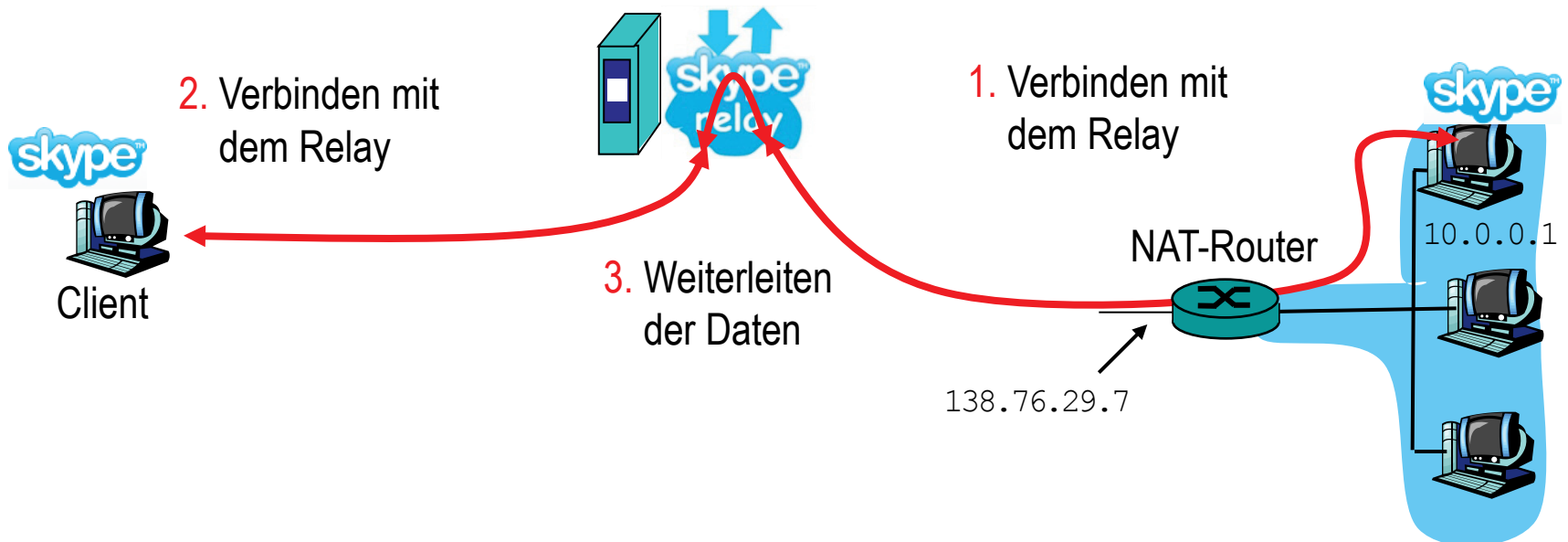
## 4.4.2 NAT-Traversal

- **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 des NAT-Routers (138.76.29.7)
    - Auffinden existierender Abbildungen in der NAT-Tabelle
    - Einträge in die NAT-Tabelle einfügen oder aus ihr löschen
- Das heißt automatische Konfiguration von statischen NAT-Einträgen!



## 4.4.2 NAT-Traversal

- **Lösung 3: Relaying (von Skype verwendet)**
  - Server hinter einem NAT-Router baut eine Verbindung zu einem Relay auf (welches nicht hinter einem NAT-Router liegt)
  - Client baut eine Verbindung zum Relay auf
  - Relay leitet die Pakete vom Client zum Server und umgekehrt weiter



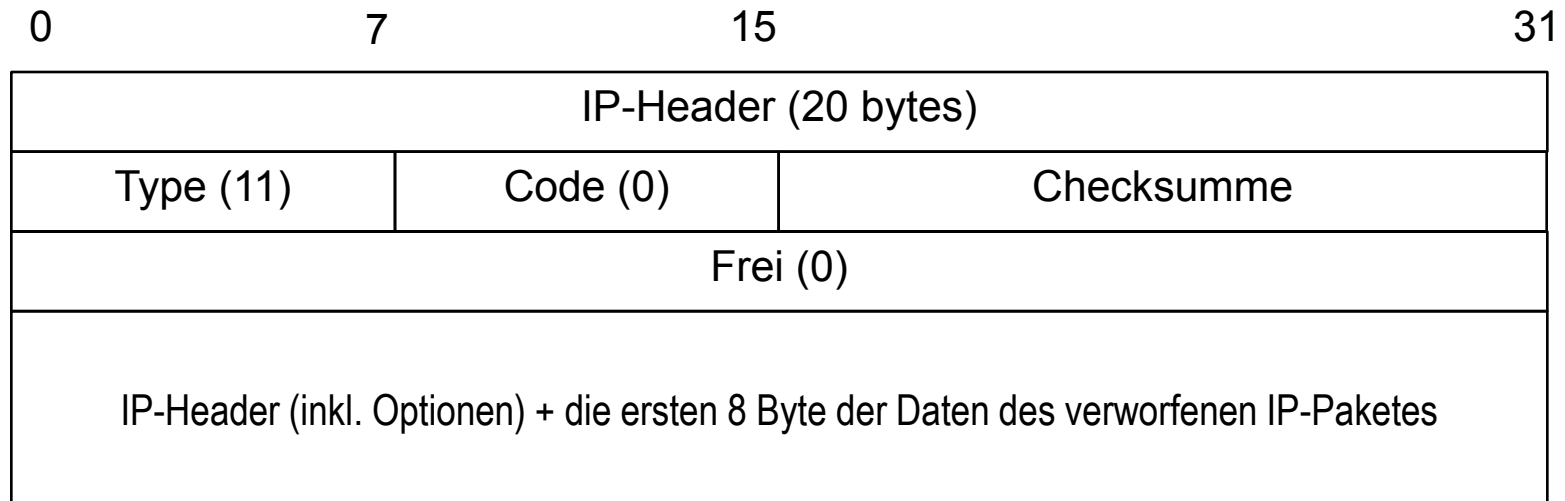
## 4.4.3 Internet Control Message Protocol (ICMP)

	<u>Type</u>	<u>Code</u>	<u>Beschreibung</u>
• Wird von Hosts und Routern verwendet, um Informationen über das Netzwerk selbst zu verbreiten	0	0	echo reply (ping)
– Fehlermeldungen: Host, Netzwerk, Port, Protokoll nicht erreichbar	3	0	dest. network unreachable
– Echo-Anforderung und Antwort (von ping genutzt)	3	1	dest host unreachable
	3	2	dest protocol unreachable
	3	3	dest port unreachable
	3	6	dest network unknown
• Gehört zur Netzwerkschicht, wird aber in IP-Datagrammen transportiert	3	7	dest host unknown
	4	0	source quench (congestion control - not used)
• <b>ICMP-Nachricht:</b> Type, Code und die ersten 8 Byte des IP-Datagramms, welches die Nachricht ausgelöst hat	8	0	echo request (ping)
	9	0	route advertisement
	10	0	router discovery
	11	0	TTL expired
	12	0	bad IP header

## 4.4.3 Traceroute

- Aufgabe von Traceroute:
  - Traceroute bestimmt Informationen über alle Router, die auf dem Weg zu einer IP-Adresse liegen
  - Dabei wird auch die RTT zu jedem Router bestimmt
- Funktionsweise von Traceroute:
  1. Traceroute schickt ein UDP-Paket an die Adresse, für die der Weg untersucht werden soll; TTL im IP-Header wird auf 1 gesetzt
  2. Der erste Router verwirft das IP-Paket (TTL = 1!) und schickt eine ICMP-Time-Exceeded-Fehlermeldung an den Absender
  3. Traceroute wiederholt dies mit TTL = 2 etc.

## 4.4.3 ICMP-Time-Exceeded-Nachricht



## 4.4.3 Traceroute und ICMP

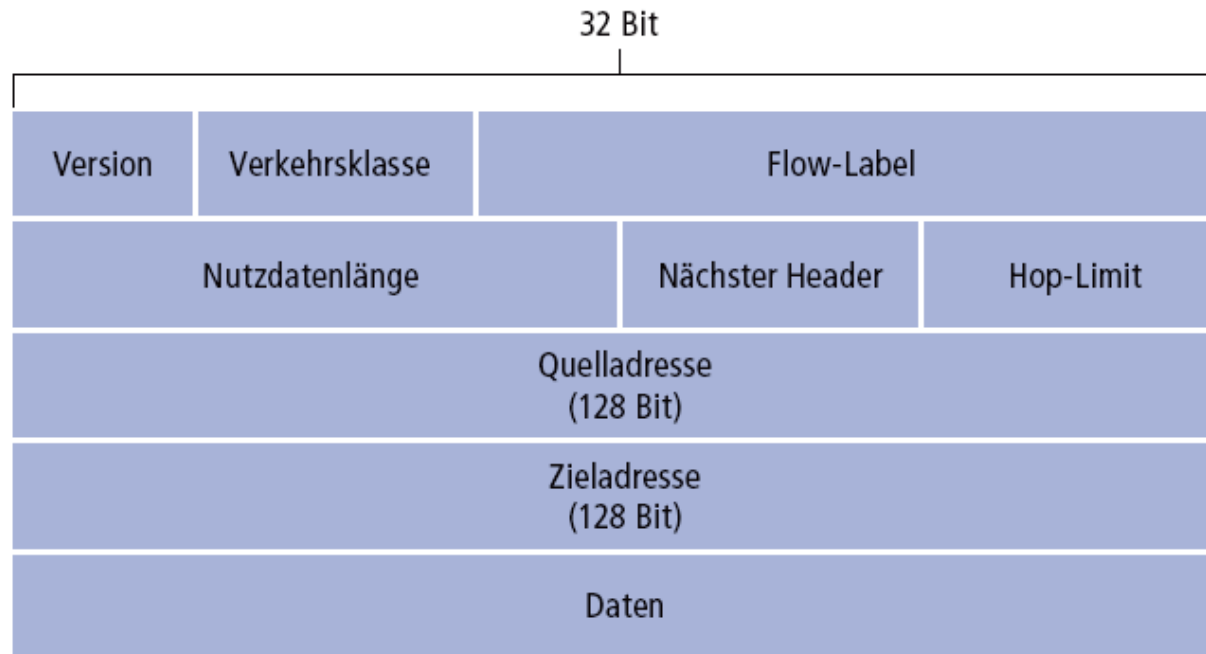
Wie erkennt man, ob das Paket schließlich beim Empfänger angekommen ist?

- Traceroute sendet UDP-Pakete an einen Port, der **wahrscheinlich** nicht verwendet wird, und erwartet eine ICMP-Port-Unreachable-Nachricht vom Empfänger!
  - Demo:
    - `traceroute <host>`
- So lernt das Quellsystem Anzahl und Identitäten der Router kennen, die zwischen ihm und dem Zielhost liegen und es kann die RTT zwischen den beiden Hosts messen.

## 4.4.4 IPv6

### IPv6 Header

- *Verkehrsklasse*: 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? Beispiel: TCP!



## 4.4.4 Veränderungen in IPv6 gegenüber IPv4

- *Checksumme*: entfernt, um die Verarbeitung in den Routern zu erleichtern
- *Optionen*: als separate Header, die auf den IP-Header folgen
  - Werden durch das “Nächster Header”-Feld angezeigt
  - Einfachere Behandlung in Hosts und Routern
- *ICMPv6*: neue Version von ICMP
  - Zusätzliche Pakettypen, z.B. “Packet Too Big”
  - Funktionen zur Verwaltung von Multicast-Gruppen (später mehr)



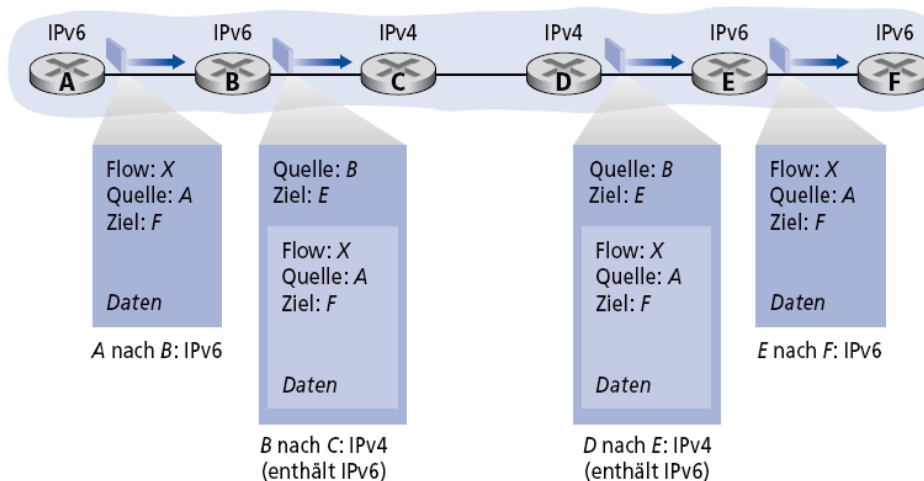
## 4.4.4 Übergang von IPv4 zu IPv6

- Es können nicht alle Router gleichzeitig umgestellt werden
  - Wie kann ein Netzwerk funktionieren, in dem sowohl IPv4- als auch IPv6-Router vorhanden sind?
- *Tunneling*: IPv6 wird im Datenteil von IPv4-Datagrammen durch das klassische IPv4-Netzwerk transportiert

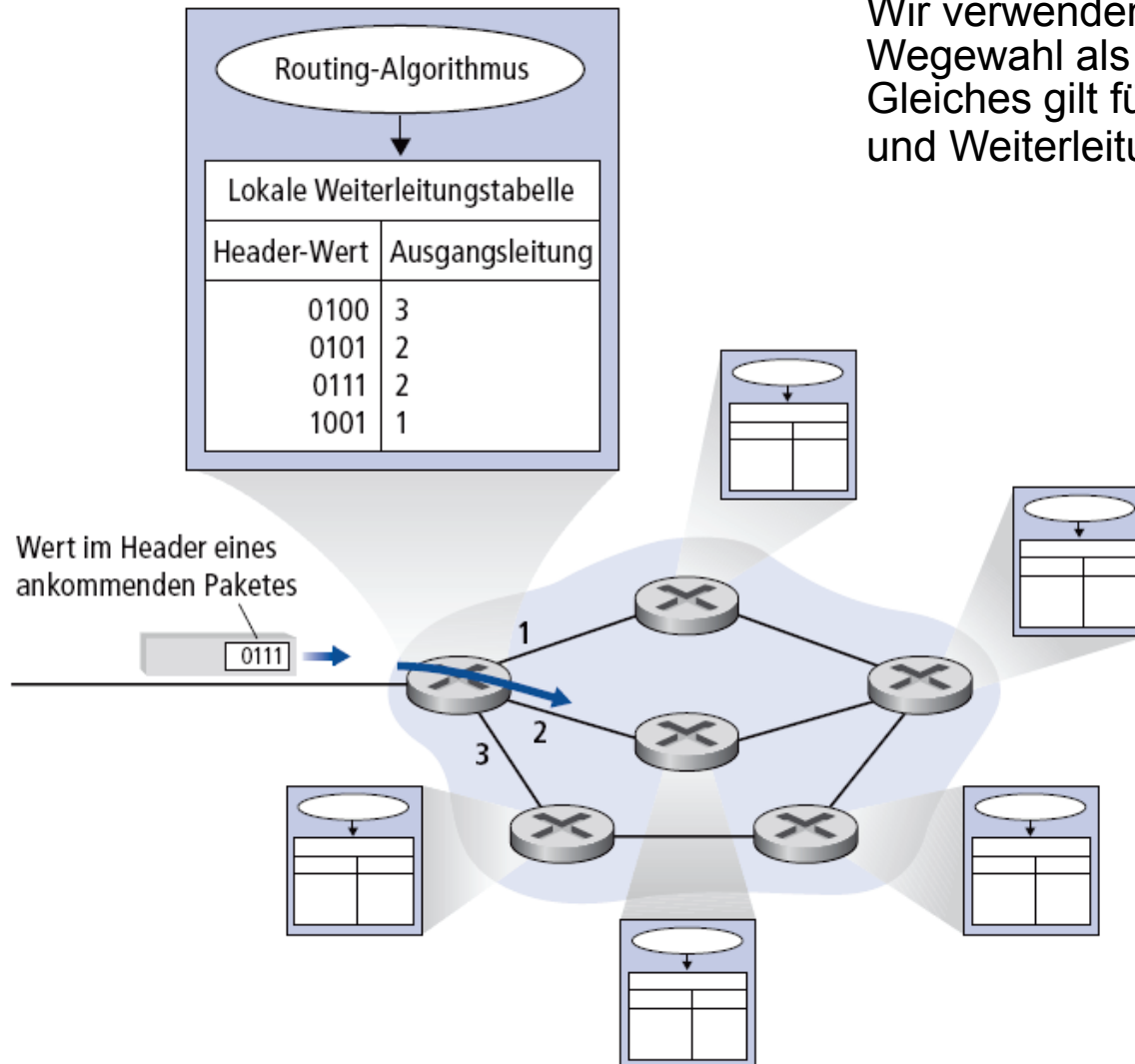
Logische Sicht



Reale Situation



# 4.5 Routing-Algorithmen



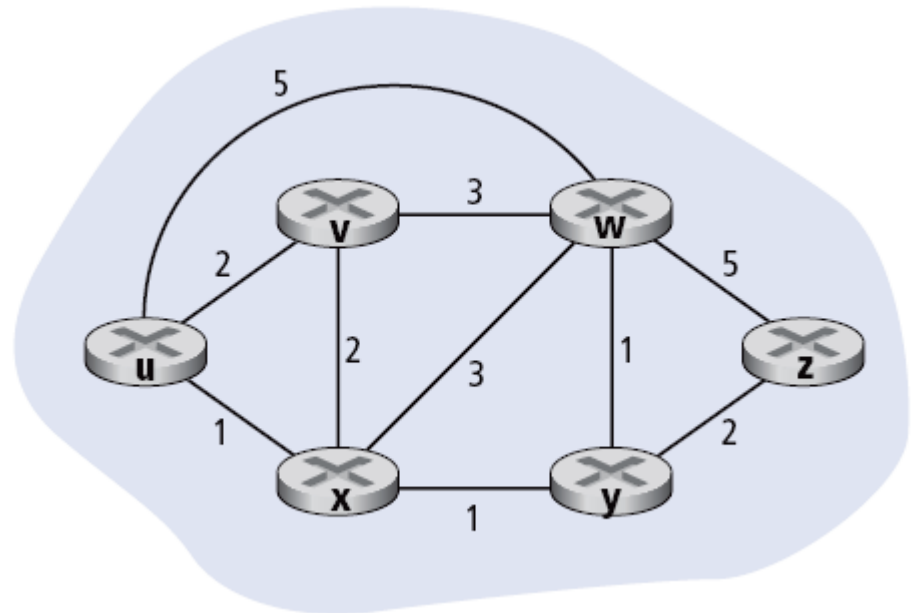
Wir verwenden Routing und Wegewahl als Synonyme. Gleiches gilt für Forwarding und Weiterleitung.

## 4.5 Ein Netzwerk als Graph

Graph:  $G = (N, E)$

$N$  = Menge von Routern =  
 $\{ u, v, w, x, y, z \}$

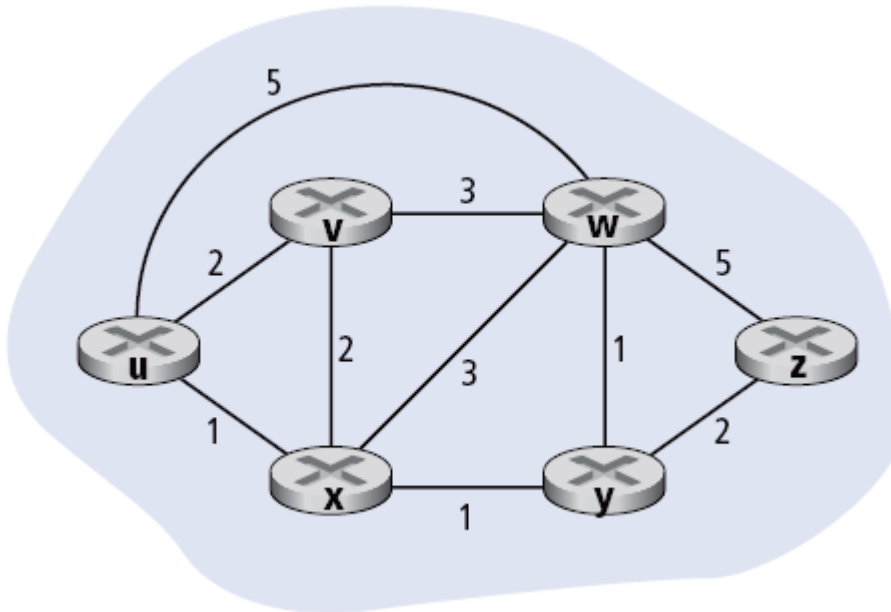
$E$  = Menge von Links =  
 $\{ (u,v), (u,x), (u,w), (v,x), (v,w),$   
 $(x,w), (x,y), (w,y), (w,z), (y,z) \}$



**Anmerkung:** Graphen können auch in anderen Zusammenhängen in Computernetzwerken betrachtet werden.

Beispiel: P2P, dann ist  $N$  die Menge der Peers und  $E$  die Menge der TCP-Verbindungen zwischen den Peers.

## 4.5 Kosten



- $c(x,x')$  = Kosten von Link  $(x,x')$ 
  - z.B.  $c(w,z) = 5$
- Kosten können:
  - immer 1 sein (= *Hop-Count*)
  - invers proportional zur Linkkapazität sein
  - proportional zur Ausbreitungsverzögerung sein
  - ...

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

Frage: Was ist der günstigste Weg von u nach z?

**Routing-Algorithmus:** Algorithmus, der den günstigsten Weg findet

## 4.5 Klassifikation von Routing-Algorithmen

### Globale oder dezentrale Informationen?

- Globale Informationen:
  - Alle Router kennen die vollständige Topologie des Graphen (alle Knoten, alle Kanten, alle Kosten)
  - Link-State-Routing
- Dezentrale Informationen:
  - Router kennt die Kanten und Kosten zu seinen direkten Nachbarn
  - Router tauschen iterativ Informationen mit ihren Nachbarn aus
  - Distance-Vector- oder Distanzvektor-Routing

### Statisch oder dynamisch?

- Statisch:
  - Routen ändern sich langsam/selten
  - Einmaliges Ausführen des Algorithmus OK
  - Manuelle Konfiguration (manchmal) OK
- Dynamisch:
  - Routen ändern sich schnell/ständig
  - Ausführen des Algorithmus notwendig als Reaktion auf Änderungen von Links
  - Manuelle Konfiguration nicht möglich

## 4.5 Link-State-Routing-Algorithmus

### Dijkstras Algorithmus

- Alle Kanten, Knoten und Kosten sind in allen Knoten bekannt
  - Durch Fluten von Link-State-Informationen im Netz
  - Alle Knoten haben das gleiche Wissen
- Berechnet die günstigsten Pfade von einem Knoten (der Quelle) zu allen anderen Knoten
  - Bestimmt damit die Weiterleitungstabelle für die Quelle
- Iterativ: Nach  $k$  Iterationen sind die günstigsten Pfade zu den  $k$  am günstigsten zu erreichenden Zielen bekannt

## 4.5 Dijkstra's Algorithmus

```
1 initialisiere:
2   N' = {u}    // u = Quelle
3   für alle Knoten v aus N
4     wenn v ein Nachbar von u ist
5       dann  $D(v) = c(u,v)$ 
6     sonst  $D(v) = \infty$ 
7
8 wiederhole:
9   finde ein w aus N welches nicht in N' ist, so dass D(w) minimal ist
10  füge w zu N' hinzu
11  Berechne D(v) neu für jeden Nachbarn v von w der nicht in N' ist:
12     $D(v) = \min(D(v), D(w) + c(w,v))$ 
13  /* die neuen Kosten nach v sind entweder die alten Kosten
14  oder die Kosten nach w plus die Kosten von w nach v */
15 bis N' = N
```

N: Menge aller Knoten

N': Menge der Knoten, für die der günstigste Pfad definitiv feststeht

$c(x,y)$ : Kosten des Links von x nach y; wird auf  $\infty$  gesetzt, wenn sie keine Nachbarn sind

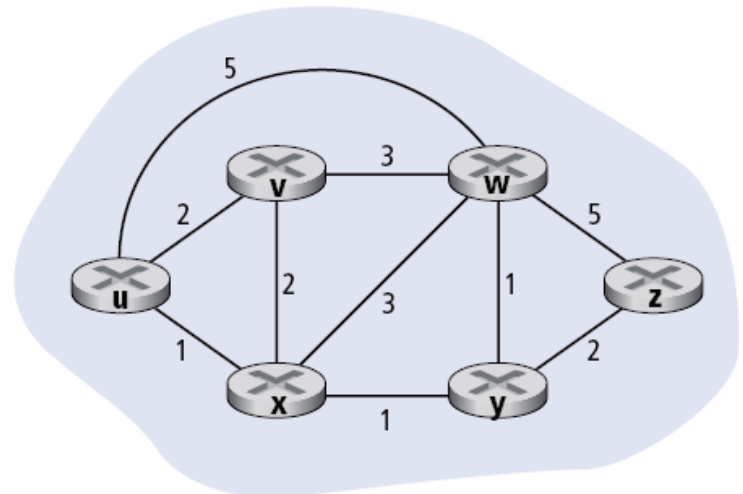
$D(v)$ : Kosten des günstigsten derzeit bekannten Pfades von der Quelle zu v

$p(v)$ : Vorgängerknoten entlang des Pfades von der Quelle nach v

# 4.5 Dijkstra's Algorithmus

Beispiel:

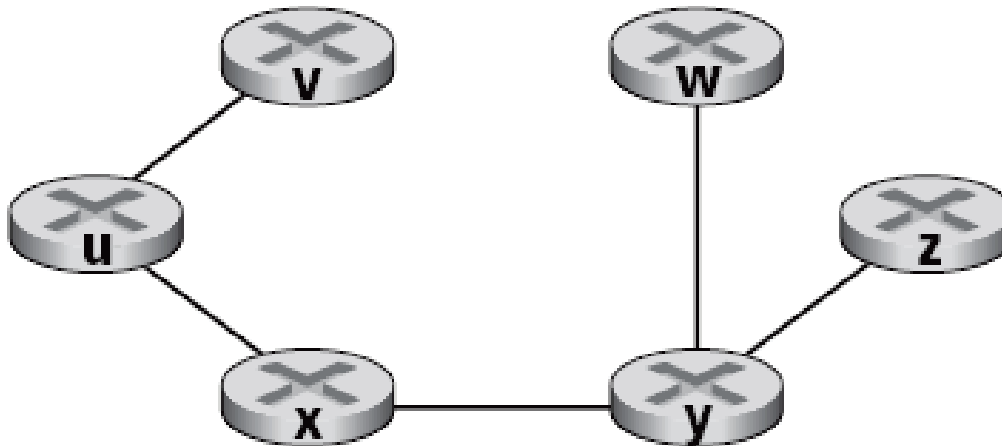
Schritt	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	$u$	$2, u$	$5, u$	$1, u$	$\infty$	$\infty$
1	$ux$	$2, u$	$4, x$		$2, x$	$\infty$
2	$uxy$	$2, u$	$3, y$			$4, y$
3	$uxyv$		$3, y$			$4, y$
4	$uxyvw$					$4, y$
5	$uxyvwz$					





## 4.5 Dijkstra's Algorithmus

Beispiel:



Weiterleitungstabelle in u:

Ziel	Leitung
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Rechenkomplexität: n Knoten

Jede Iteration: alle Knoten, die nicht in N' sind, überprüfen

→  $n(n+1)/2$  Vergleich:  $O(n^2)$

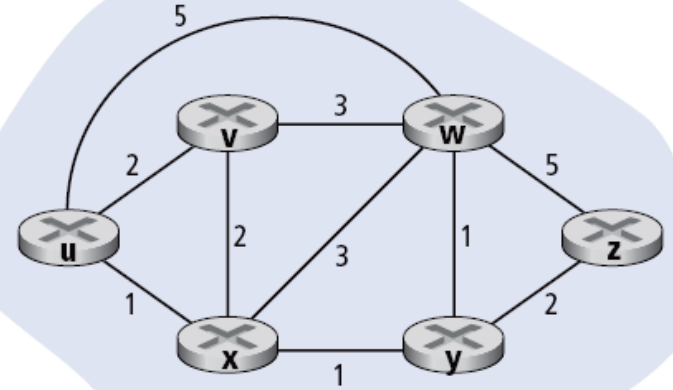
- Effizientere Implementierung möglich:  $O(n \log n)$

## 4.5 Distance-Vector-Routing

- Prinzipielle Idee: **Bellman-Ford-Gleichung**
- Sei  $d_x(y)$  der billigste Pfad von  $x$  nach  $y$
- Dann gilt:
  - $d_x(y) = \min \{c(x,v) + d_v(y) \}$
- Wobei das Minimum über alle Nachbarn  $v$  von  $x$  gebildet wird

## 4.5 Distance-Vector-Routing

Beispiel zu Bellman-Ford:



- Es gilt:
  - $d_v(z)=5$ ,  $d_x(z) = 3$ ,  $d_w(z) = 3$
- Aussage der Bellman-Ford-Gleichung ist dann:
 
$$d_u(z) = \min \{c(u,v) + d_v(z), c(u,x) + d_x(z), c(u,w) + d_w(z)\}$$

$$= \min \{ \quad 2 + 5, \quad \quad 1 + 3, \quad \quad 5 + 3 \}$$

$$= 4$$
- Der Nachbar, über den das Minimum erreicht wird, wird für das betrachtete Ziel in die Routing-Tabelle eingetragen
  - Hier: **z** wird von **u** aus am besten über **x** erreicht

## 4.5 Der Distance-Vector-Algorithmus

- $D_x(\mathbf{y})$  schätzt die günstigsten Kosten für einen Pfad von  $x$  nach  $y$
- Knoten  $x$  kennt die Kosten zu jedem Nachbarn  $v$ :  $\mathbf{c}(x,v)$
- Knoten  $x$  führt einen Distanzvektor:  $\mathbf{D}_x = [D_x(\mathbf{y}): \mathbf{y} \in \mathbf{N}]$
- Knoten  $x$  merkt sich die Distanzvektoren der Nachbarn
  - Für jeden Nachbarn  $v$  merkt sich  $x$ :  $\mathbf{D}_v = [D_v(\mathbf{y}): \mathbf{y} \in \mathbf{N}]$

## 4.5 Distance-Vector-Routing

### Prinzipielles Vorgehen:

- Wenn ein Knoten hochgefahren wird, dann sendet ein Knoten seinen eigenen Distanzvektor an alle seine Nachbarn
- Immer wenn sich der Distanzvektor eines Knotens ändert, sendet er diesen an alle seine Nachbarn

Wenn ein Knoten einen neuen Distanzvektor von einem Nachbarn erhält, überprüft er seinen eigenen Distanzvektor nach der Bellman-Ford-Gleichung:

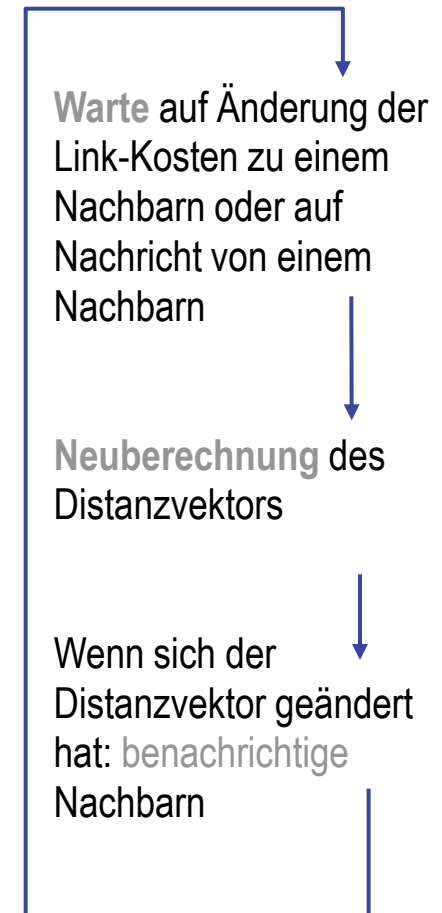
$d_x(y) = \min \{c(x,v) + d_v(y)\}$ , Minimum wird über alle Nachbarn  $v$  gebildet

→ Unter realistischen Annahmen konvergiert dieser Algorithmus zu einer Situation, in der jeder Knoten den Nachbarn auf dem günstigsten Weg zu jedem anderen Knoten kennt.

## 4.5 Distance-Vector-Routing

- Iterativ und asynchron:
  - Jede Iteration wird ausgelöst durch:
    - Veränderung der Kosten eines Links zu einem direkten Nachbarn
    - Neuer Distanzvektor von einem Nachbarn
- Verteilt:
  - Keine globale Kenntnis der vollständigen Netzwerktopologie notwendig
  - Stattdessen: lokales Verbreiten von Informationen

Ablauf in  
jedem Knoten:

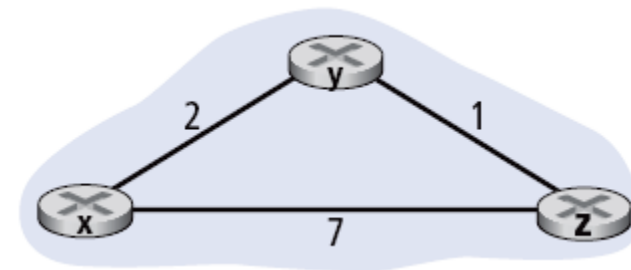


# 4.5 Distance-Vector-Routing

**Tabelle von y**

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

Distanzvektor von Nachbar x  
 Eigener Distanzvektor (von y)  
 Distanzvektor von Nachbar z



# 4.5 Distance-Vector-Routing

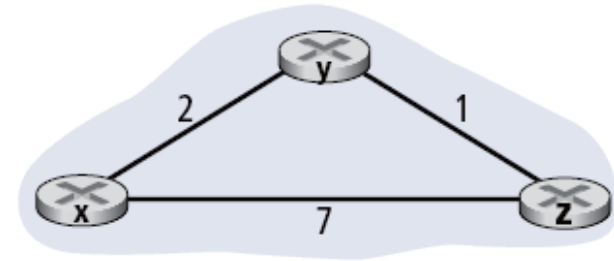


Tabelle von x

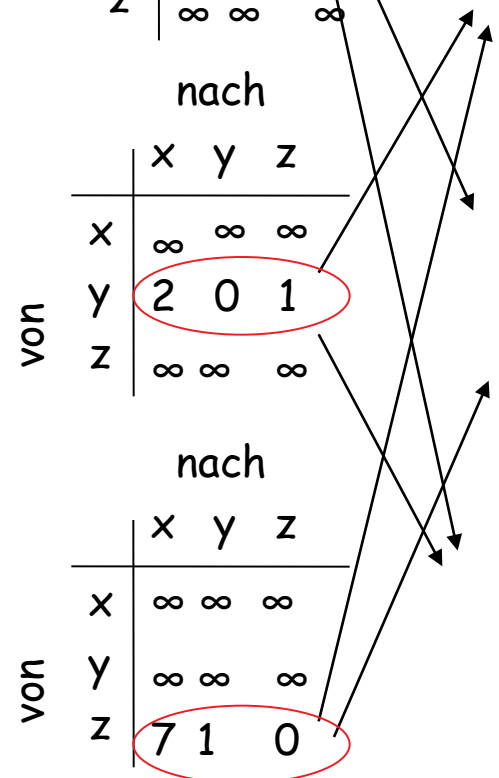
		nach		
		x	y	z
von	x	0	2	7
	y	$\infty$	$\infty$	$\infty$
	z	$\infty$	$\infty$	$\infty$

Tabelle von y

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

Tabelle von z

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



Zeit



# 4.5 Distance-Vector-Routing

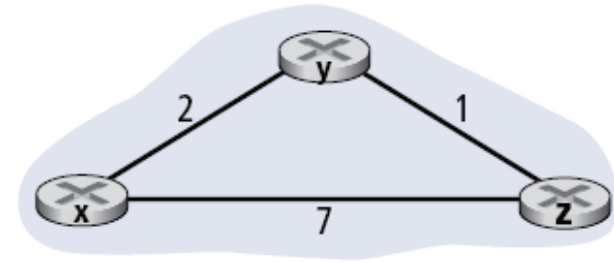


Tabelle von x

		nach		
		x	y	z
von	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

Tabelle von y

		nach		
		x	y	z
von	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

Tabelle von z

		nach		
		x	y	z
von	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

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

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

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

$$D_x(y) = \min\{c(x,y)+D_y(y), c(x,z)+D_z(y)\} = \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y)+D_y(z), c(x,z)+D_z(z)\} = \min\{2+1, 7+0\} = 3$$

Zeit

# 4.5 Distance-Vector-Routing

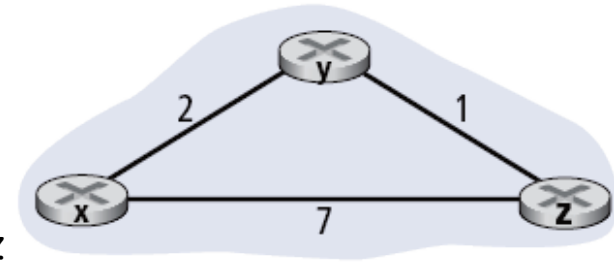


Tabelle von x

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

Tabelle von y

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

Tabelle von z

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

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

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

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

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

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

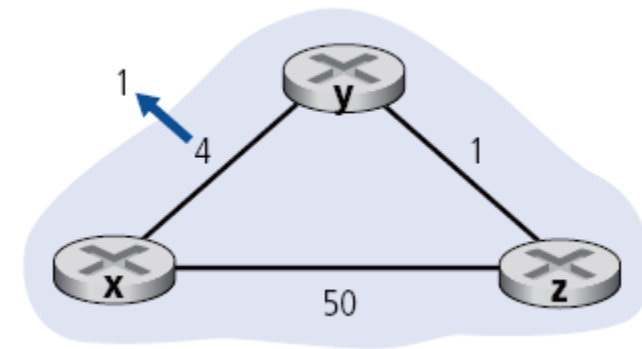
		nach		
		x	y	z
von	x	0	2	3
	y	2	0	1
	z	3	1	0

Zeit

## 4.5 Distance-Vector: Änderungen der Link-Kosten

Link-Kosten verringern sich:

- Knoten erkennt lokale Änderung
- Berechnet seine Distanzvektoren neu
- Wenn sich sein DV geändert hat, dann werden die Nachbarn informiert



Entwicklung des Distanzvektors nach x:

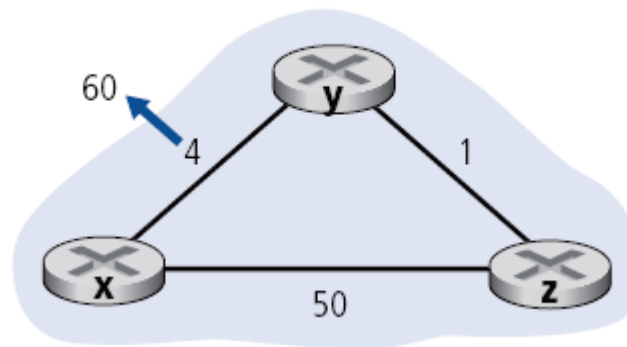
1. Zum **Zeitpunkt  $t_0$** , erkennt y die Verringerung der Kosten, sein DV ändert sich und er informiert seine Nachbarn
2. Zum **Zeitpunkt  $t_1$** , empfängt z den neuen DV von y, sein DV ändert sich, er informiert alle seine Nachbarn

“good news  
travels fast”

## 4.5 Distance-Vector: Änderungen der Link-Kosten

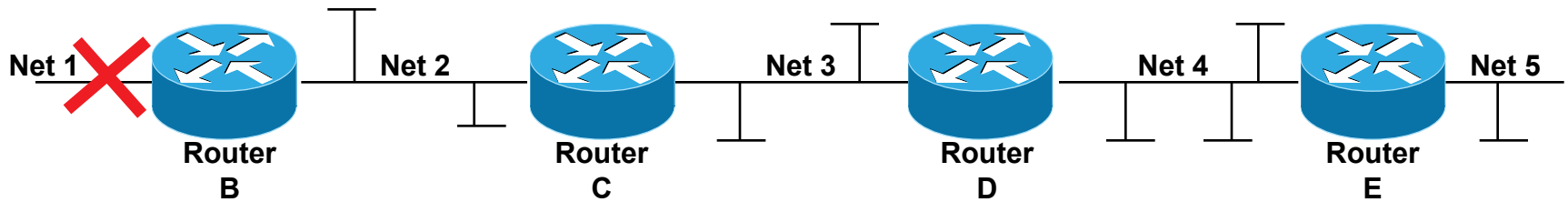
Link-Kosten erhöhen sich:

- Verringerung der Link-Kosten: *Good news travels fast*
- Erhöhung der Link-Kosten: **Bad news travels slow** - “*Count to Infinity*”-Problem!
- 44 Iterationen, bis der Algorithmus zur Ruhe kommt im Beispiel auf dieser Folie



- Ausführlicheres Beispiel mit anderer Topologie auf der nächsten Folie!

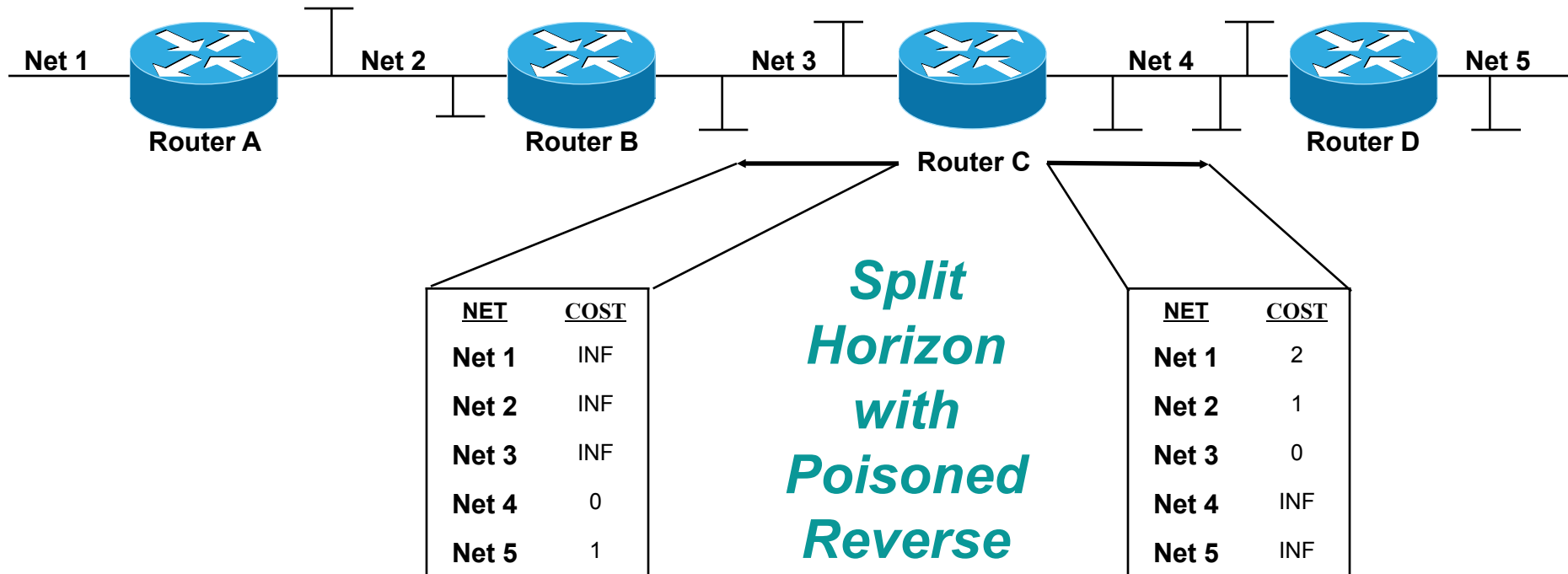
# 4.5 Distance-Vector: „Count-to-Infinity“-Problem



Pfad-Kosten (Distanz) zu Net 1 in allen Routing Tabellen. Ausfall passiert vor Schritt 1.

Schritt	Distanz zu Net 1	Distanz zu Net 1	Distanz zu Net 1	Distanz zu Net 1
0	1	2	3	4
1	3	2	3	4
2	3	4	3	4
3	5	4	5	4
4	5	6	5	6
5	7	6	7	6
6	7	8	7	8
.	.	.	.	.
.	.	.	.	.
X	INF	INF	INF	INF

# 4.5 Distance-Vector: Split Horizon with Poisoned Reverse



Lösungsansatz → *Split Horizon with Poisoned Reverse*:

- Prinzip: Zum Knoten, über welchen X erreicht wird, eigene Entfernung zu X als unendlich ankündigen!

Wichtig: *Split Horizon with Poisoned Reverse* löst das Problem nicht in allen Fällen!

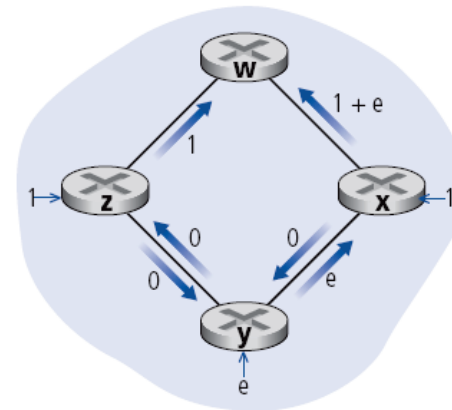
## 4.5 Link-State- vs. Distance-Vector-Routing

- **Nachrichtenaustausch**
  - LS: Nachrichten werden im ganzen Netz geflutet
  - DV: Nachrichten werden nur mit Nachbarn ausgetauscht
- **Geschwindigkeit der Konvergenz**
  - LS: Schnelle Konvergenz! 😊
    - Fluten der Zustände der Links
  - DV: variiert stark 😞
    - Temporäre Routing-Schleifen sind möglich 😞
    - Count-to-Infinity-Problem 😞
- **Robustheit: *Was passiert, wenn ein Router fehlerhaft ist?***
  - LS:
    - Knoten kann falsche Kosten für einen Link fluten
    - Pfade möglicherweise nicht mehr optimal, das Netzwerk bleibt aber schleifenfrei und verbunden! 😊
  - DV:
    - Router kann falsche Kosten für einen ganzen Pfad ankündigen
    - Fehler propagiert durch das ganze Netzwerk  
→ u.U. sehr schädlich 😞

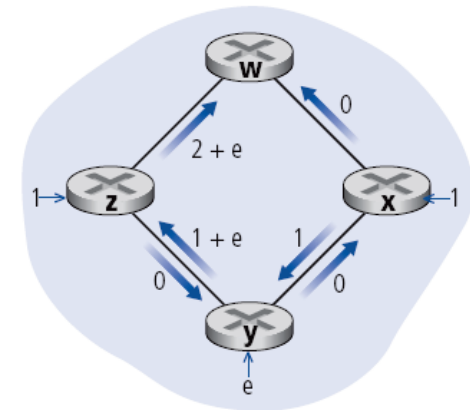
# 4.5 Oszillationen bei lastabhängigen Linkgewichten

→ Oszillationen sind möglich wenn die Metrik für die Kosten der Links von der Netzwerklast abhängt

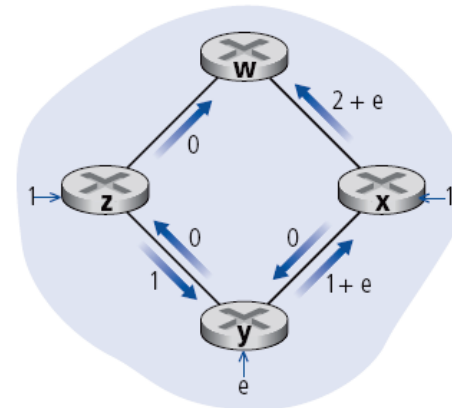
→ Daher ist es sehr ratsam, verkehrsabhängige Link-Metriken strikt zu meiden!



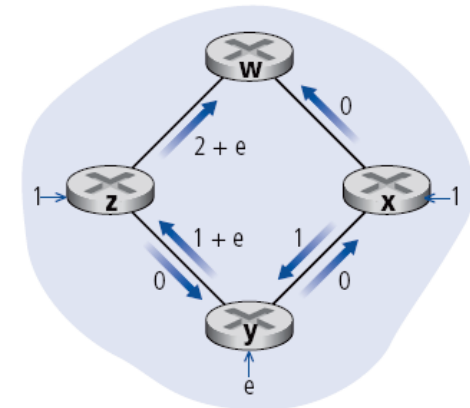
a Anfängliches Routing



b x, y entdecken den im Uhrzeigersinn verlaufenden besseren Pfad nach w



c x, y, z entdecken den gegen den Uhrzeigersinn verlaufenden besseren Pfad nach w



d x, y, z entdecken den im Uhrzeigersinn verlaufenden besseren Pfad nach w



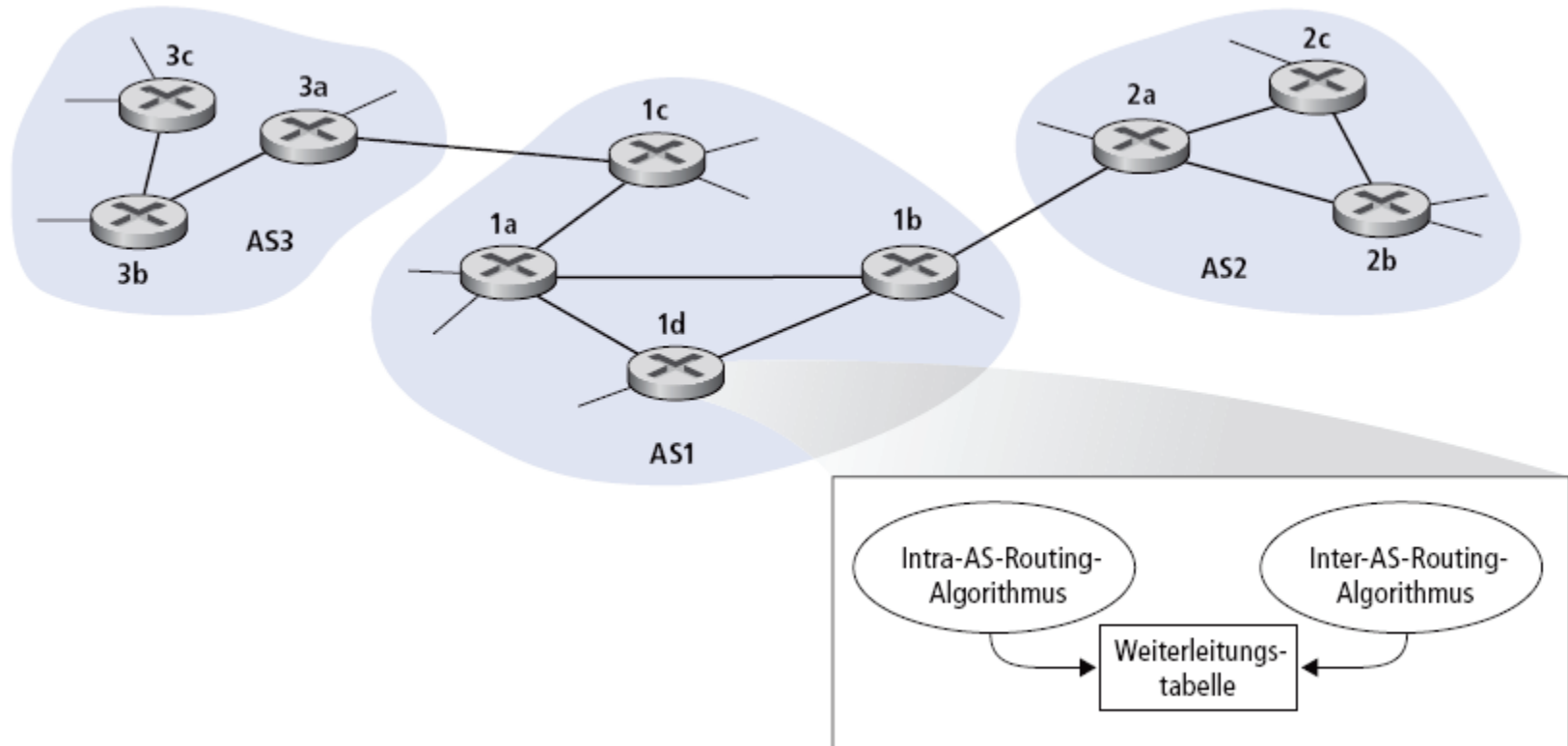
## 4.5 Hierarchisches Routing - Problem

- Bisher:
  - Alle Router sind gleich
  - Das Netzwerk ist „flach“ und besitzt keine Hierarchie
  - Entspricht nicht der Realität
- Eine Frage der Administration:
  - Internet = Netzwerk von Netzwerken
  - Jede Organisation hat eigene Politiken und Präferenzen bezüglich ihres Netzwerkes
- Eine Frage der Größenordnung:
  - Viele Zielnetzwerke!
  - Es können nicht alle Netzinternen Links weltweit berücksichtigt werden, da die Routing-Protokolle (d.h. der Austausch von Routing-Informationen) alle Links im Internet überlasten würden

## 4.5 Hierarchisches Routing - Idee

- Router werden zu Regionen zusammengefasst, diese nennt man **Autonome Systeme (AS)**
- Router innerhalb eines AS verwenden ein Routing-Protokoll
  - “Intra-AS”-Routing-Protokoll
  - Router in verschiedenen AS können verschiedene Intra-AS-Routing-Protokolle verwenden
- Manchmal gilt:
  - Eine Organisation = ein AS
  - Es gibt aber auch Organisationen (z.B. einige ISPs), die aus mehreren AS bestehen
- Gateway-Router:
  - Ein Router in einem AS, der eine Verbindung zu einem Router in einem anderen AS hat
- **Routing zwischen AS**
  - „Inter-AS“-Routing-Protokoll

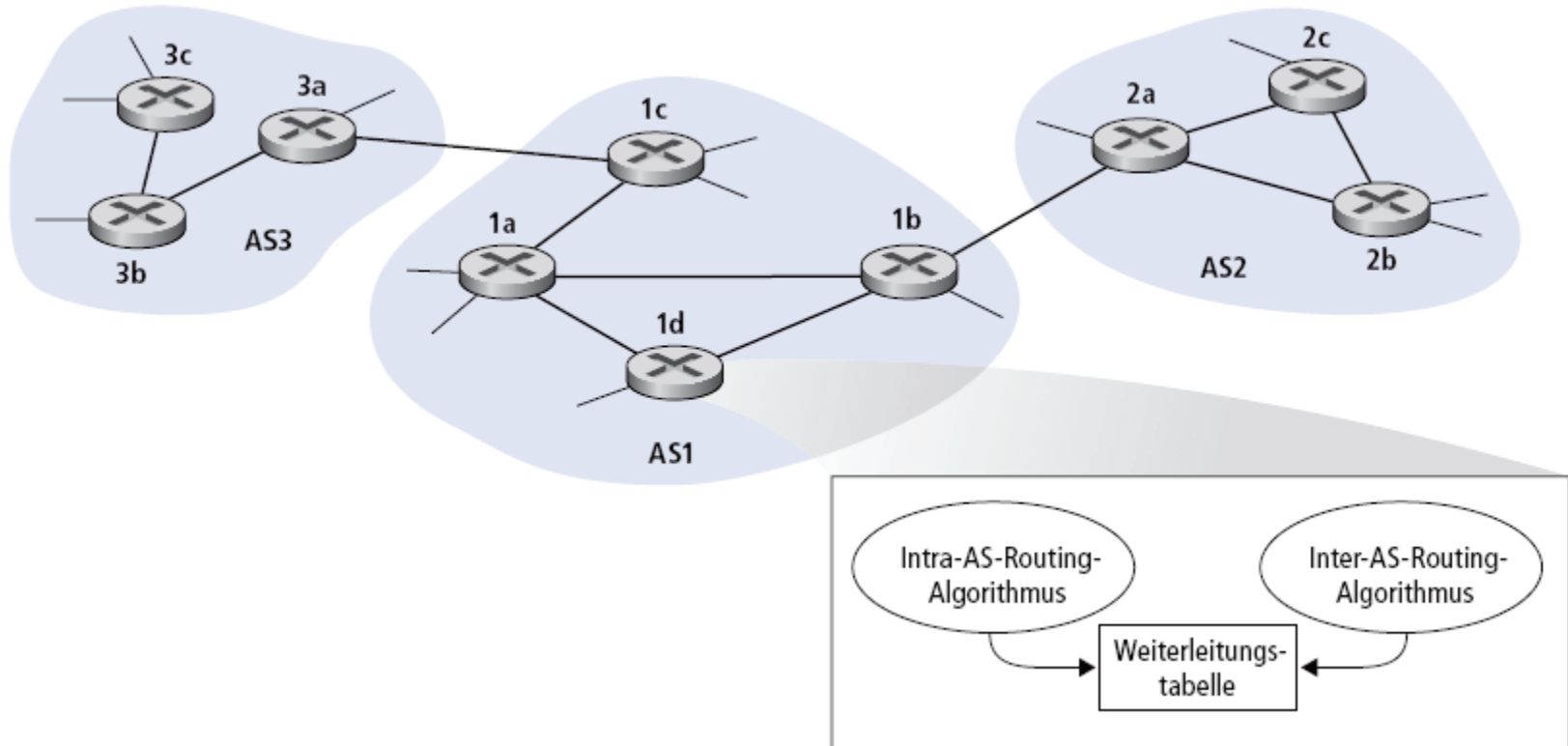
## 4.5 Verbundene Autonome Systeme



Routing-Tabelle wird durch **Intra-AS-** und **Inter-AS-Routing-Algorithmen** gefüllt:

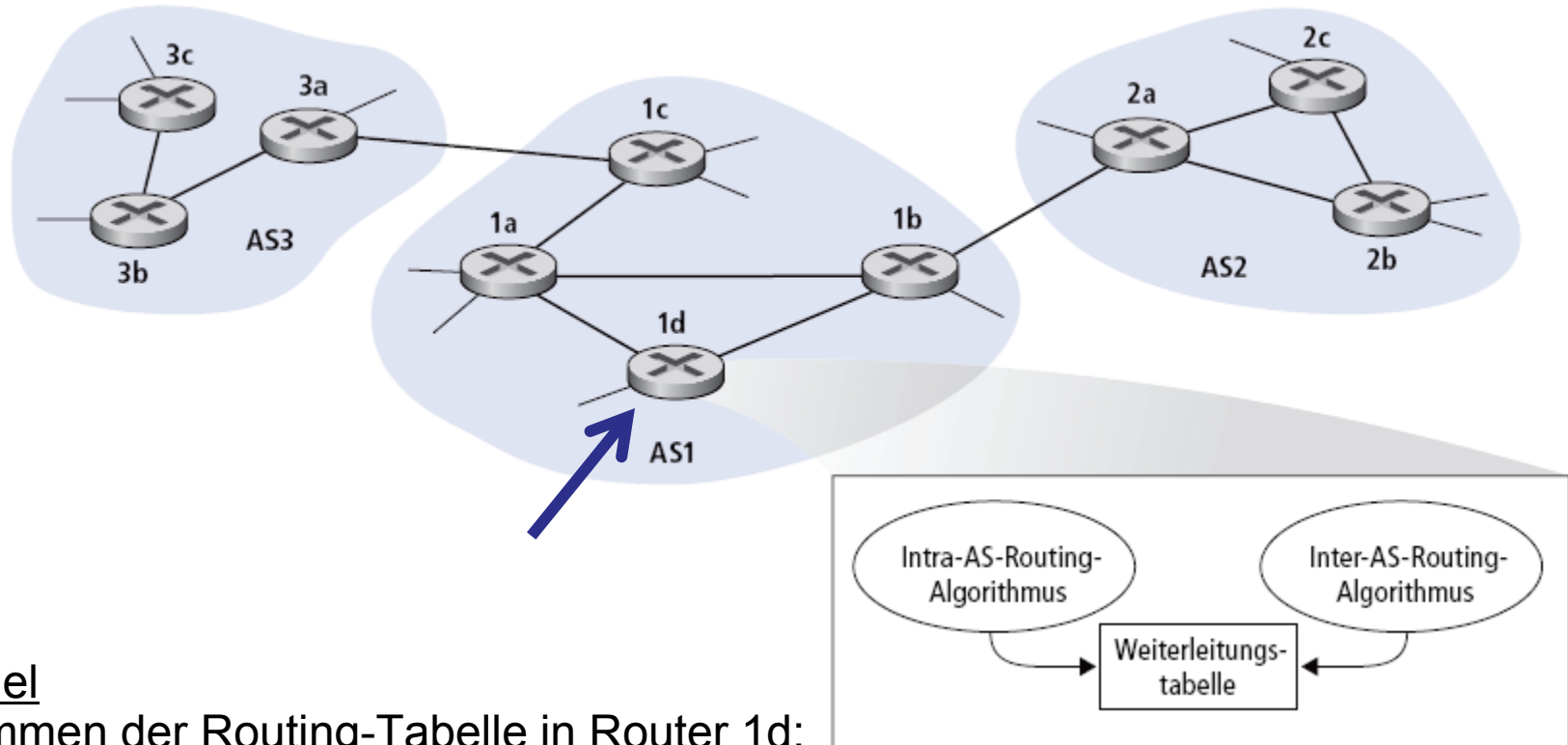
- Intra-AS-Einträge für interne Ziele
- Inter-AS- & Intra-AS-Einträge für externe Ziele

## 4.5 Aufgaben des Inter-AS-Routing



- Wenn ein Router in AS1 ein Paket für ein Ziel außerhalb von AS1 erhält:
  - Router sollte das Paket zu einem der Gateway-Router in AS1 weiterleiten
  - Aber zu welchem?
- AS1 muss mit Hilfe von Inter-AS-Routing Folgendes tun:
  - Lernen, welche Ziele über die Autonomen Systeme AS2 und AS3 erreichbar sind
  - Verteilen dieser Informationen an alle Backbone-Router in AS1

## 4.5 Aufgaben des Inter-AS-Routing



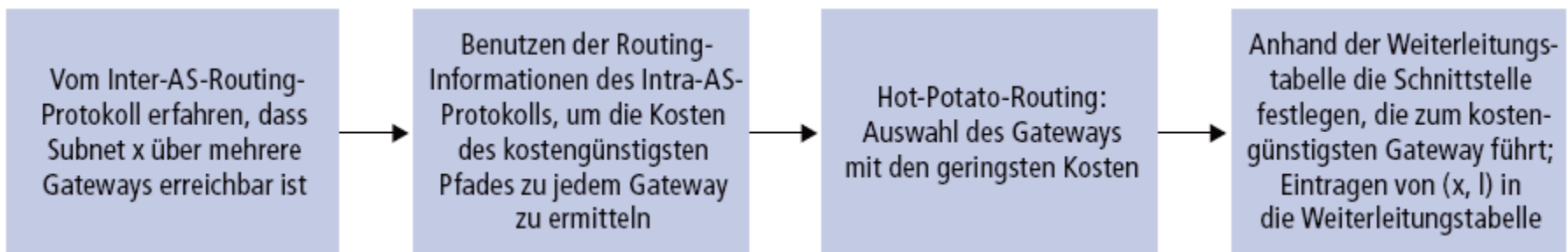
### Beispiel

Bestimmen der Routing-Tabelle in Router 1d:

1. Angenommen, AS1 lernt durch das Inter-AS-Routing-Protokoll, dass Netzwerk x von AS3 (Gateway-Router 1c), aber nicht von AS 2 aus erreicht werden kann
2. Inter-AS-Routing-Protokoll propagiert diese Information zu allen internen Backbone-Routern
3. Backbone-Router 1d bestimmt durch das Intra-AS-Routing-Protokoll, dass sein Interface I auf dem kürzesten Pfad zu 1c liegt
4. Router 1d nimmt einen Eintrag (x,I) in der Routing-Tabelle vor

## 4.5 Beispiel: Alternative Routen

- Angenommen AS1 lernt durch das Inter-AS-Routing-Protokoll, dass Netzwerk X sowohl über AS2 als auch über AS3 zu erreichen ist
- Für den Eintrag in die Routing-Tabelle muss Router 1d sich für einen Pfad entscheiden
- Ebenfalls Aufgabe des Inter-AS-Routing-Protokolls
- Eine Möglichkeit:
  - **Hot Potato-Routing:** Schicke das Paket an den nächsten Gateway-Router, der es in ein anderes AS weiterleiten kann



## 4.6 Intra-AS-Routing

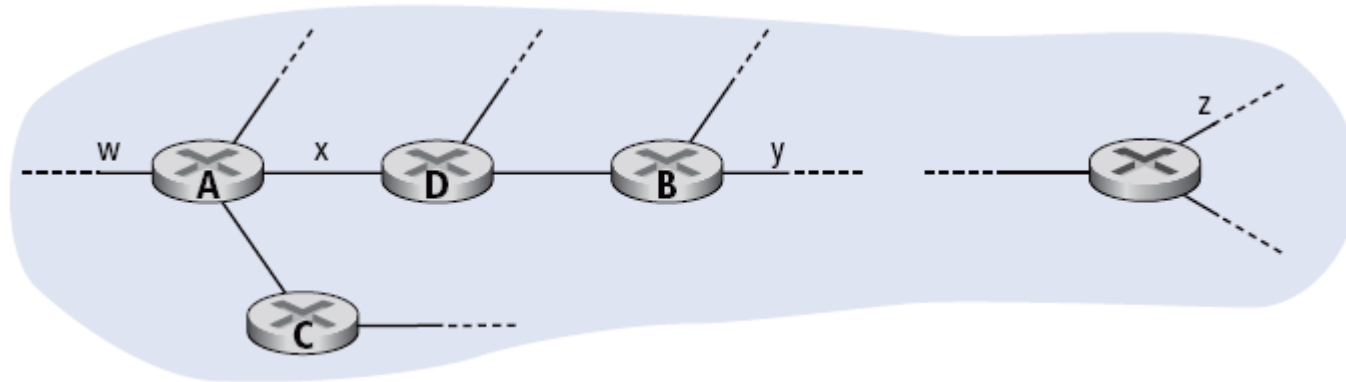
- Auch **Interior Gateway Protocol (IGP)**
- Die bekanntesten Intra-AS-Routing-Protokolle:
  - **RIP**: Routing Information Protocol → *Distance Vector*
  - **IGRP**: Interior Gateway Routing Protocol → *Distance Vector* (Proprietär: Cisco)
  - **OSPF**: Open Shortest Path First → Link State
  - **IS-IS**: Intermediate System to Intermediate System → Link State

## 4.6 Routing Information Protocol (RIP)

- Version 1 spezifiziert in RFC 1058
- Version 2 (kompatibel mit Version 1) spezifiziert in [RFC 2453](#)
- Distance-Vector-Algorithmus
- War bereits in der BSD-UNIX-Distribution von 1982 enthalten
- Metrik: Anzahl der Hops (Maximum = 15 Hops)
  
- Distanzvektoren werden zwischen den Nachbarn alle 30 Sekunden per **RIP-Advertisement** ausgetauscht
- Jedes Advertisement enthält eine Liste von bis zu 25 Zielnetzwerken im Inneren des Autonomen Systems



## 4.6 RIP Beispiel



Routing-Tabelle in D:

Zielsubnetz	Nächster Router	Anzahl von Hops zur Zieladresse
w	A	2
y	B	2
z	B	7
x	–	1
...	...	...

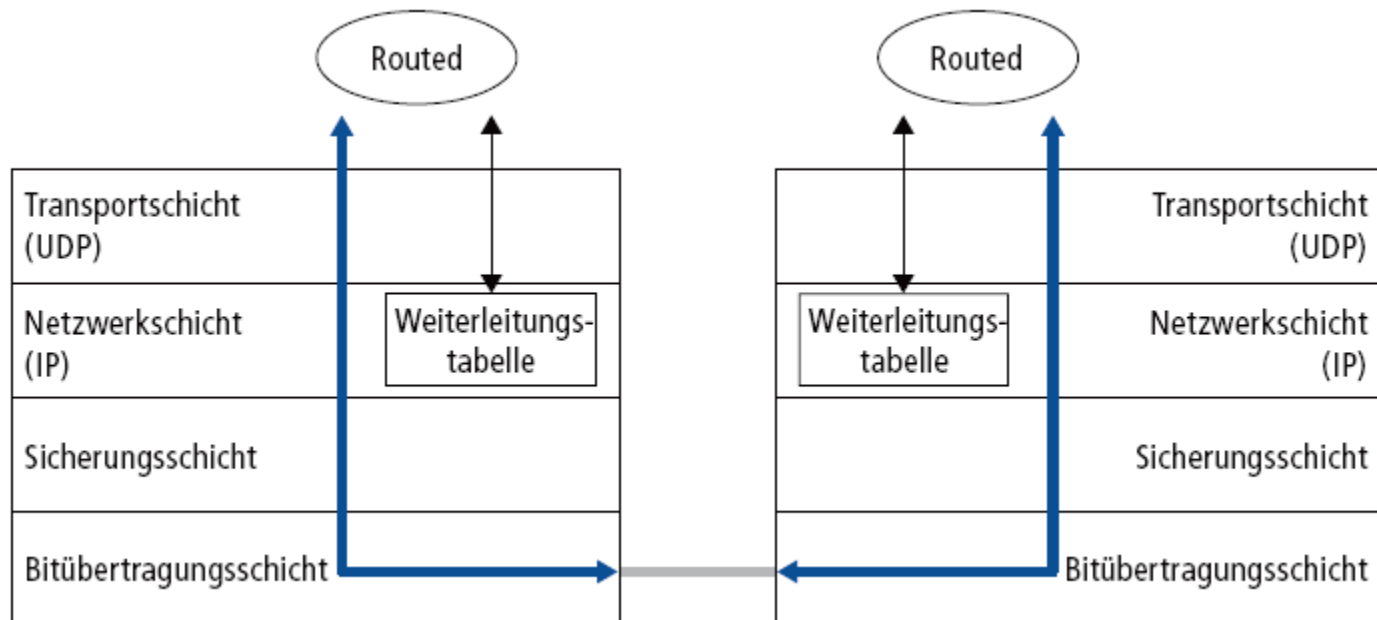
## 4.6 RIP - Brechen von Links

Wenn von einem Nachbarn 180 Sekunden lang kein Advertisement empfangen wurde, gilt der Nachbar als nicht mehr vorhanden

- Alle Routen über diesen Nachbarn werden ungültig
- Neuberechnung des lokalen Distanzvektors
- Verschicken des neuen Distanzvektors (wenn er sich verändert hat)
- Nachbarn bestimmen ihren Distanzvektor neu und verschicken ihn gegebenenfalls
- ...
- Die Information propagiert schnell durch das Netzwerk
- *Split Horizon with Poisoned Reverse* wird verwendet, um Routing-Schleifen zu vermeiden (unendlich ist hier 16 Hops!)

## 4.6 RIP-Architektur

- Die Routing-Tabelle kann von RIP in einem Prozess auf Anwendungsebene gepflegt werden: z.B. routed (für „route daemon“)
- Advertisements werden per UDP verschickt



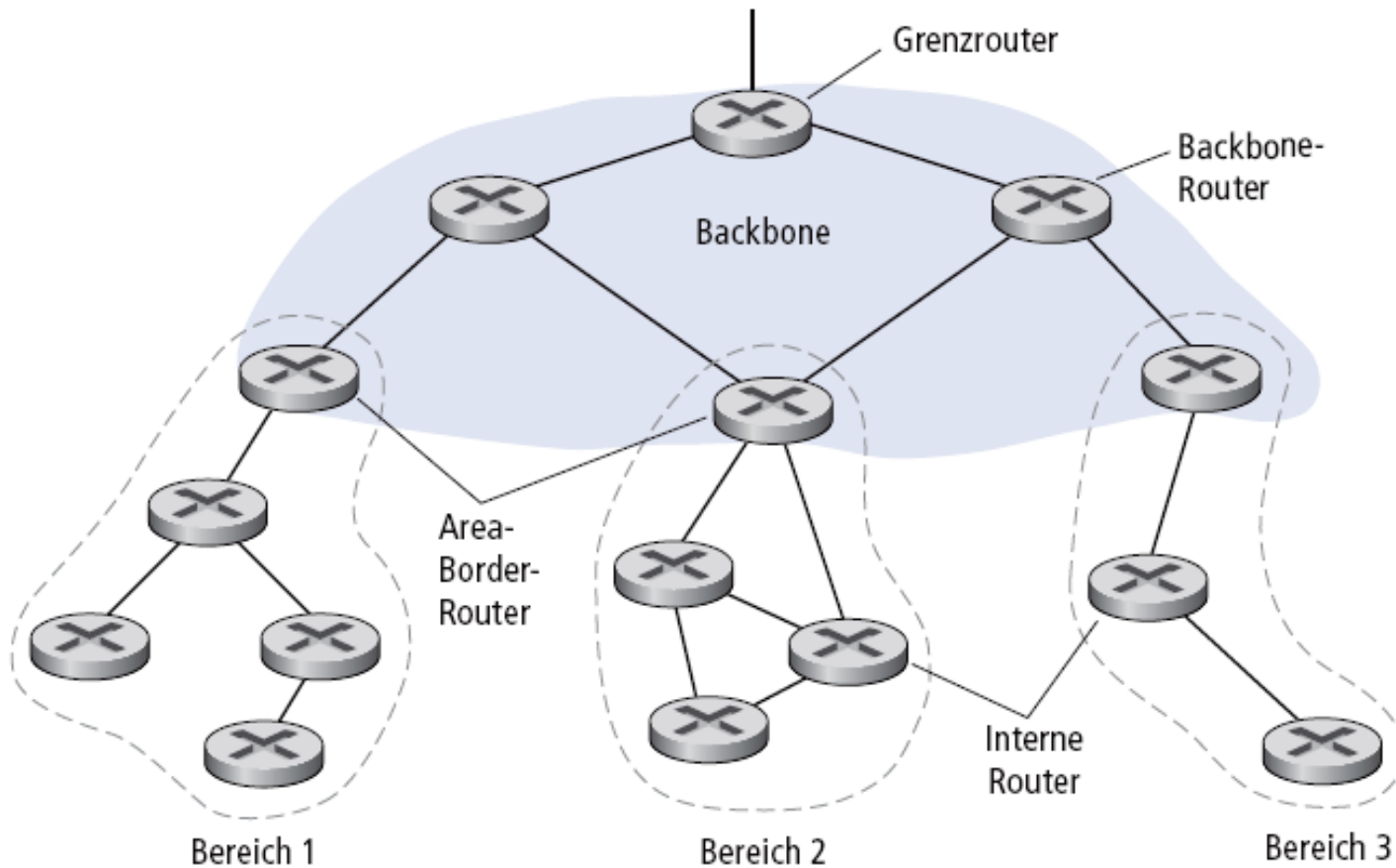
## 4.6 Open Shortest Path First (OSPF)

- Version 2 spezifiziert in [RFC 2328](#)
- “Open”: frei verfügbar
- „Shortest Path First“: verwendet **Link-State-Routing-Algorithmus**
  - Periodisches Fluten von Link-State-Paketen (LSAs – Link State Advertisements)
    - Jeder Router kündigt seine Links an
    - Diese Ankündigungen werden effizient geflutet
      - Ein Router schickt eine empfangene Ankündigung allen seinen Nachbarn, von denen er dieselbe Ankündigung noch nicht erhalten hat
    - OSPF-Pakete werden direkt in IP-Pakete eingepackt
  - Topologieänderungen (z.B. Linkausfälle) werden schnell mittels LSAs angekündigt
  - Topologie des Netzwerkes in jedem Router bekannt
    - Wird in einer sogenannten „Netzwerkkarte“ oder „Topology Map“ gespeichert
  - Routen werden mit Dijkstras Algorithmus berechnet

## 4.6 Eigenschaften von OSPF

- Schnell, effizient, schleifenfrei → besitzt alle guten Eigenschaften von Link-State-Routing!
- Sicherheit ist gegeben: Alle OSPF-Nachrichten können authentifiziert werden
- Hierarchisches OSPF in größeren autonomen Systemen

## 4.6 Hierarchisches OSPF



## 4.6 Hierarchisches OSPF

- Zweistufige Hierarchie: Local Area, Backbone
  - Ein Backbone pro AS
  - Jeder Router in einer Local Area kennt deren detaillierte Topologie und die Richtung zu den Netzwerken der anderen Local Areas
- Area-Border-Router:
  - Zusammenfassen der Distanzen zu den Netzwerken in der eigenen Local Area
  - Ankündigen dieser Zusammenfassung an die anderen Area-Border-Router (d.h. Backbone-Router)
  - Ankündigungen der Zusammenfassungen der anderen Area-Border-Router (d.h. Backbone-Router) in der Local Area
- Backbone-Router: führt OSPF-Routing und *Traffic Forwarding* im Backbone durch
- Boundary-Router: stellt eine Verbindung zu anderen Autonomen Systemen her

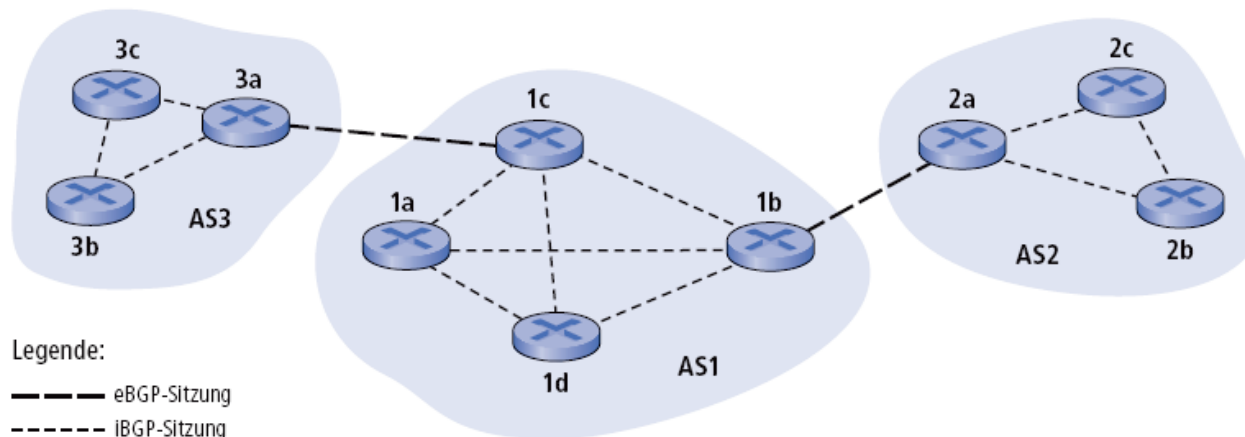
## 4.6 Border Gateway Protocol (BGP)

- Das Border Gateway Protocol, Version 4, ist der De-facto-Standard für Inter-AS-Routing im Internet
- Spezifiziert in [RFC 4271](#)
- BGP erlaubt es einem AS:
  - Informationen über die Erreichbarkeit von Netzen von seinen benachbarten Autonomen Systemen zu erhalten
  - Diese Informationen an die Router im Inneren des eigenen AS weiterzuleiten
  - „Gute“ Routen zu einem gewünschten Zielnetzwerk zu bestimmen, wobei die Qualität einer Route von den Informationen über die Erreichbarkeit und Politiken abhängig ist
- Außerdem ermöglicht BGP es einem AS, sein eigenes Netzwerk anzukündigen und so dessen Erreichbarkeit den anderen Autonomen Systemen im Internet mitzuteilen



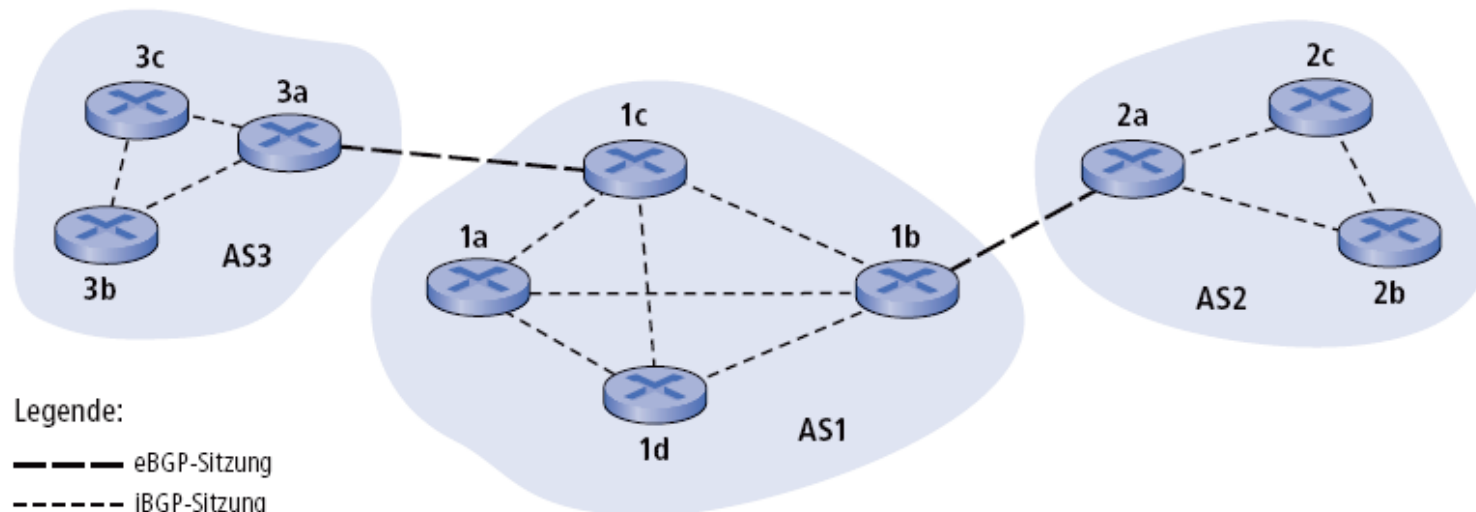
## 4.6 BGP-Grundlagen

- Paare von Routern (BGP-Peers) tauschen Routing-Informationen über semi-permanente TCP-Verbindungen aus: BGP-Sitzung (engl. BGP session)
  - Wenn beide Router im selben AS sind: **interne** BGP-Sitzung (iBGP)
  - Wenn beide Router in verschiedenen AS sind: **externe** BGP-Sitzung (eBGP)
- Wichtig: BGP-Sitzungen entsprechen nicht notwendigerweise physikalischen Links
- Wenn AS2 einen Präfix mitsamt Pfadattributen an AS1 meldet (z.B. 167.3/16), dann verspricht AS2, dass es alle Datagramme auf dem gemeldeten Pfad weiterleitet, deren Zieladressen zu diesem Präfix passen
  - AS2 kann Präfixe in seinen Ankündigungen aggregieren



## 4.6 Verbreiten der Erreichbarkeitsinformationen

- AS3 kündigt die Erreichbarkeit von Präfixen über die eBGP-Sitzung zwischen 3a and 1c an
- 1c kann iBGP verwenden, um diese Informationen im AS zu verbreiten
- 1b kann dann diese neuen Informationen über die eBGP-Sitzung zwischen 1b und 2a weitermelden und die Präfixe dem Autonomen System 2 ankündigen



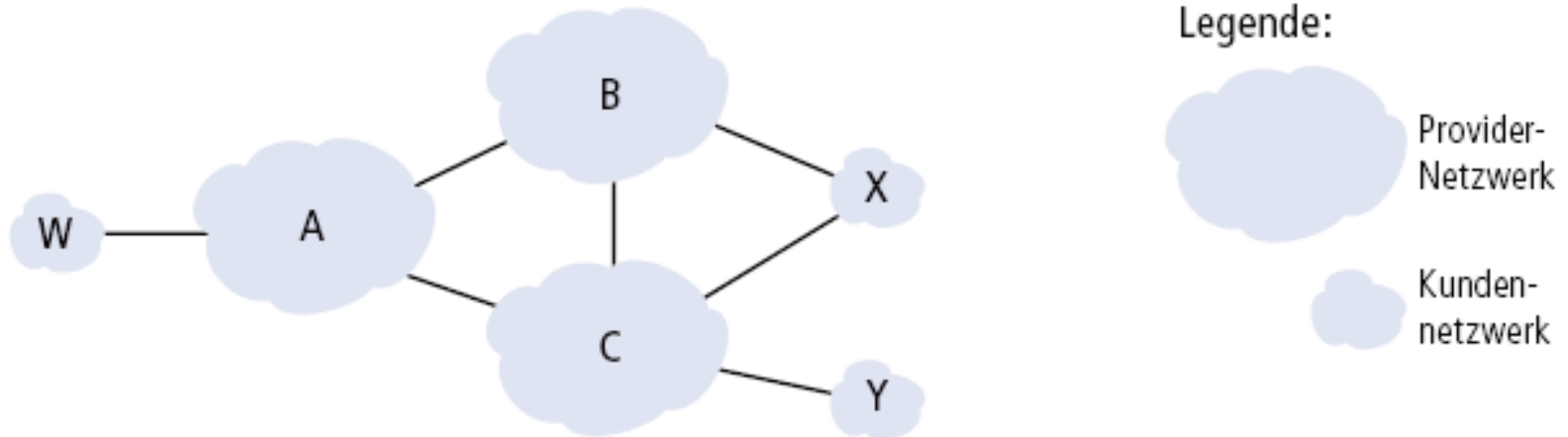
## 4.6 Pfadattribute und BGP-Routen

- Wenn ein Präfix angekündigt wird, dann beinhaltet diese Ankündigung sogenannte Pfadattribute
  - Präfix + Attribute = Route (in BGP-Terminologie)
- Zwei wichtige Attribute:
  - **AS-PATH**: eine Liste aller AS, durch welche die Ankündigung weitergeleitet wurde: AS 67, AS 17
  - **NEXT-HOP**: zwei Autonome Systeme können über mehr als ein Paar von Gateway-Routern in Kontakt stehen. Um die Ankündigung einem Router im anderen AS zuordnen zu können, schickt der ankündigende Router seine IP-Adresse als NEXT-HOP-Attribut mit
- Der empfangende Gateway-Router entscheidet durch konfigurierbare Politiken anhand der Attribute, ob eine Ankündigung angenommen werden soll bzw. welche Ankündigung bevorzugt wird

## 4.6 BGP-Routenwahl

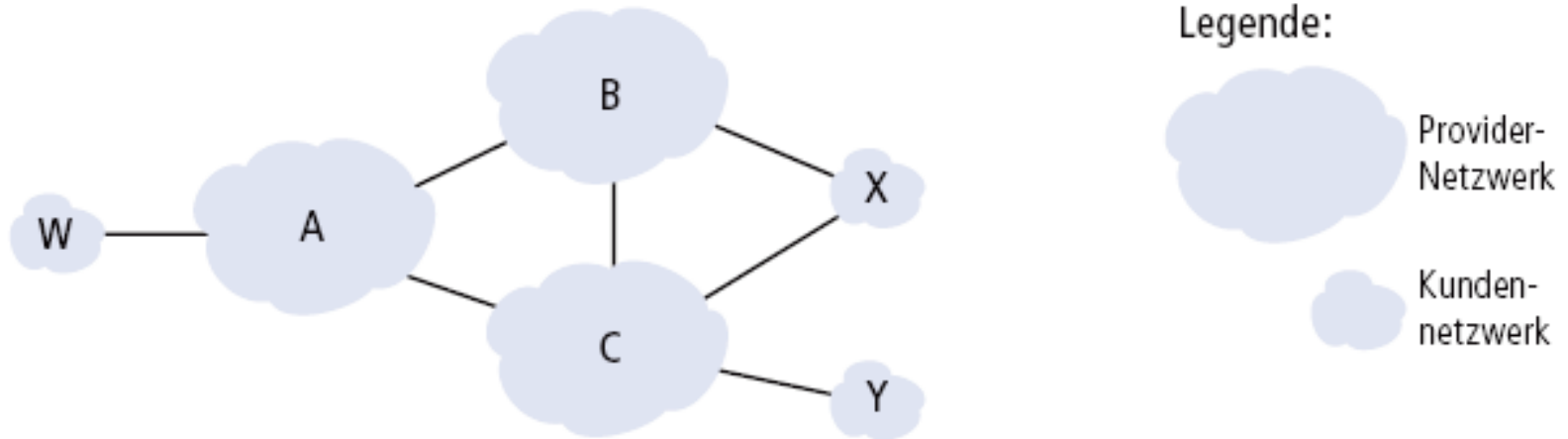
- Router können mehr als eine Route zum Ziel angeboten bekommen. Ein Router muss entscheiden, welche Route verwendet wird.
  - Regeln nach Priorität (nur ein Auszug!):
    1. Lokale Präferenz
    2. Kürzester AS-PATH
    3. Dichtester NEXT-HOP-Router: Hot Potato Routing
    4. Weitere Kriterien (häufigster Fall)
- Die BGP-Routenwahl ist eine firmenpolitische Entscheidung!

## 4.6 BGP-Routing-Politiken



- A,B,C sind Netzwerke von Providern
- X,W,Y sind Netzwerke von Kunden (der Provider)
- X ist „dual homed“: an zwei Provider angebunden
  - X möchte keine Daten von B nach C weiterleiten
  - ➔ daher wird X keine Route nach C an B ankündigen

## 4.6 BGP-Routing-Politiken



- A kündigt B den Pfad AW an (ein Kundenpfad wird angekündigt)
- B kündigt X den Pfad BAW an (alle Pfade werden immer den eigenen Kunden angekündigt)
- Sollte B den Pfad BAW auch C ankündigen?
  - **Nein!** B hätte nichts davon, da weder W noch C Kunden von B sind
    - B und C sind lediglich *Peers*, die nur den eigenen Verkehr und den der eigenen Kunden untereinander austauschen!
  - B möchte, dass C Datenpakete zu W über A (als den Provider von W) leitet
  - B möchte nur Verkehr an oder von seinen Kunden weiterleiten

## 4.6 BGP-Routing-Politiken

Warum verschiedene Protokolle für Intra-AS- und Inter-AS-Routing?

- Politiken:
  - Inter-AS: Eine Organisation möchte kontrollieren, wie (und ob) der Verkehr anderer Organisationen durch das eigene Netzwerk geleitet wird
  - Intra-AS: eigener Verkehr, eigene Administration, hier sind keine Politiken nötig
- Größenordnung:
  - Hierarchisches Routing reduziert die Größe der AS-internen Routing-Tabellen und reduziert den Netzwerkverkehr für Routing-Updates
    - ➔ Dringend notwendig für Inter-AS-Routing, nicht allzu wichtig in Intra-AS-Routing
- Performance:
  - Intra-AS: kann sich auf Performance konzentrieren
  - Inter-AS: Politiken können wichtiger sein als Performance