



# Inhalt der Vorlesung „Rechnerkommunikation“

- ✓ Einführung
- ✓ Anwendungsschicht
- ✓ Transportschicht
- **Netzwerkschicht**
- Sicherungsschicht
- Physikalische Schicht

# Netzwerkschicht

- Einführung
- IP
- Aufbau eines Routers
- Routing
- IPsec
- Netzvirtualisierung

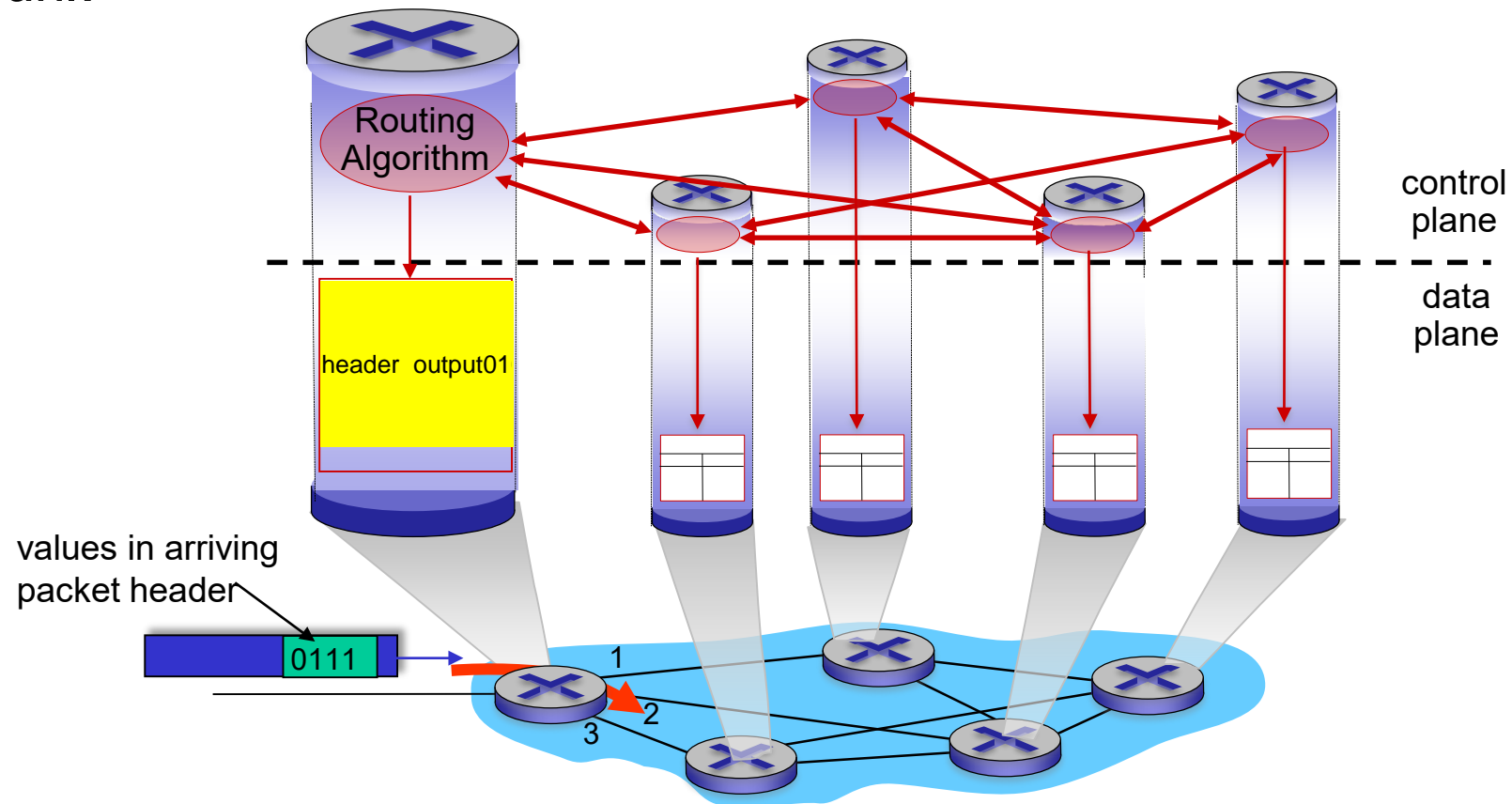
# Einführung

- Aufgabe der Netzwerkschicht: Kommunikation zwischen Endsystemen (Hosts), die über verschiedene Netze verbunden sind
  - Endsysteme senden und empfangen, Vermittlungseinheiten leiten weiter
  - **Weiterleitung (Forwarding)**: Vermittlungseinheit empfängt Dateneinheiten auf einer Schnittstelle (Interface) und leitet sie auf einer anderen weiter 
  - **Wegewahl (Routing)**: Verfahren, mit denen Vermittlungseinheiten entscheiden, über welchen Weg Dateneinheiten gesendet werden sollen 

# Einführung

## ■ Zusammenspiel von Weiterleitung und Wegewahl:

Quelle: Kurose, Ross.  
*Computer Networking:  
A Top-Down Approach*,  
7th Ed., Pearson  
Education, 2020.



# Einführung

## ■ Datagramm-basierte Paketvermittlung

- jedes Datagramm trägt eine **globale Adresse**, die von Routern zur **Weiterleitung** verwendet wird
- die Vermittlungseinheit bei IP: Router es gibt auch andere, die dies tun (aber sich im internet nicht durchgesetzt haben)
- **keine** Bereitstellung von Dienstmerkmalen wie
  - Fehlerkontrolle
  - Bewahrung der Reihenfolge
  - Unterscheidung verbindungslos/verbindungsorientiert
  - Fluss- und Überlastkontrolle
  - Garantien für Dienstgüte (z.B. Bitrate, Verzögerung, Verlust)
  - Informationssicherheit (Security), es sei denn, IPSec wird verwendet

# Einführung

## ■ Virtuelle Leitungsvermittlung

- jede Dateneinheit (für die in verschiedenen Protokollen unterschiedliche Namen verwendet werden) erhält eine lokale Kennung
- beim Weiterleiten wird diese Kennung von jeder Vermittlungseinheit verändert effizienter und garantien möglich
- dies ermöglicht den Aufbau einer virtuellen Leitung und gegebenenfalls die Bereitstellung von Dienstgütemerkmalen
- wird z.B. bei MPLS (Multiprotocol Label Switching) verwendet

## ■ hier

- zunächst Betrachtung von IP
- später MPLS als Beispiel für virtuelle Leitungsvermittlung

# Netzwerkschicht

## ✓ Einführung

### ■ IP

- Datagrammformat und klassenbasierte Adressierung
- Subnetze und klassenlose Adressierung
- Fragmentierung
- ICMP
- Autokonfiguration mit DHCP
- NAT
- IPv6

### ■ Aufbau eines Routers

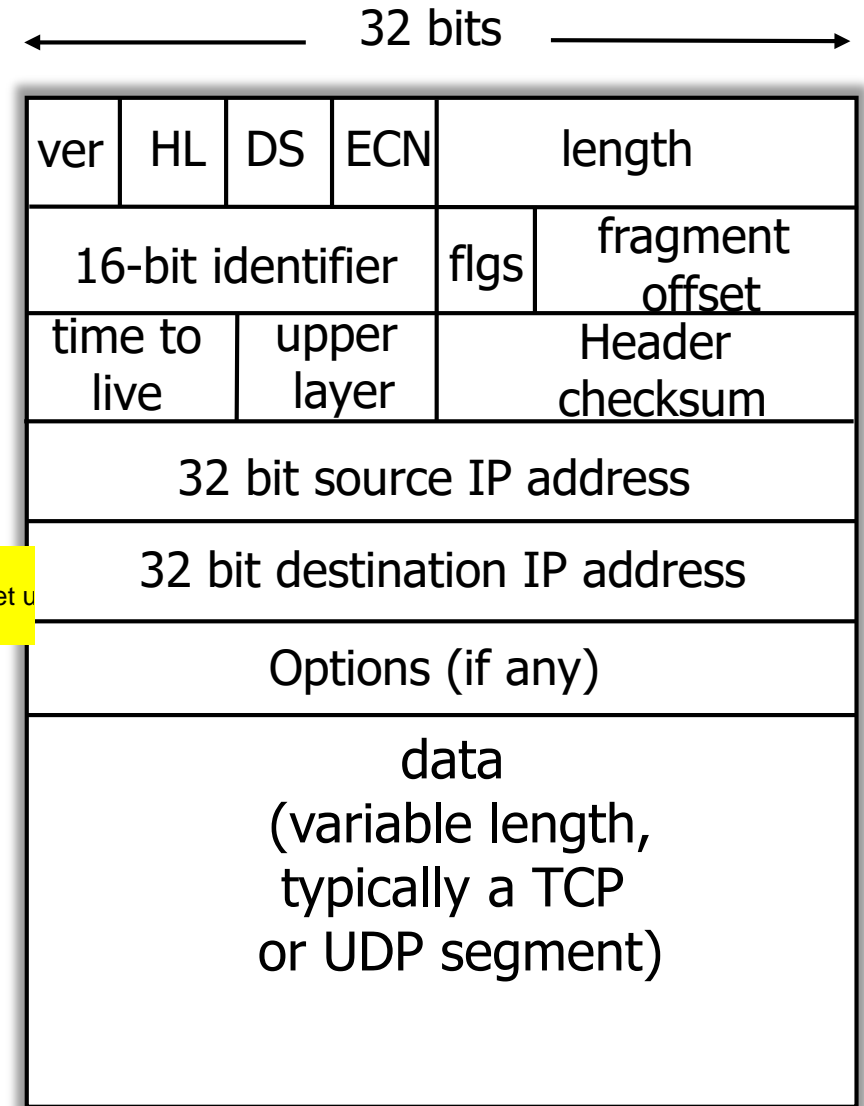
### ■ Routing

### ■ IPsec

### ■ Netzvirtualisierung

# IP: Datagrammformat

- ver (Version): 4 Bits, 4 für IPv4
- HL (Header Length): 4 Bits, Header-Länge in 32-Bit-Worten
- DS (Differentiated Services): 6 Bits, Priorisierung für Dienstgüte\* priorität
- ECN (Explicit Congestion Notification): 2 Bits\* wie letzte woche, weil
- length: Gesamtlänge in Bytes (max.  $2^{16} = 65.535$  Bytes)
- identifier, flgs, fragment offset: für Fragmentierung große paket u
- time to live: Hoplimit, jeder Router dekrementiert
- upper layer: höheres Protokoll e.g. tcp
- Options: z.B. Zeitstempelung, Weg aufzeichnen
- ohne Options 20 Bytes



\*TOS (Type of Service): ehemaliges DS/ECN-Feld

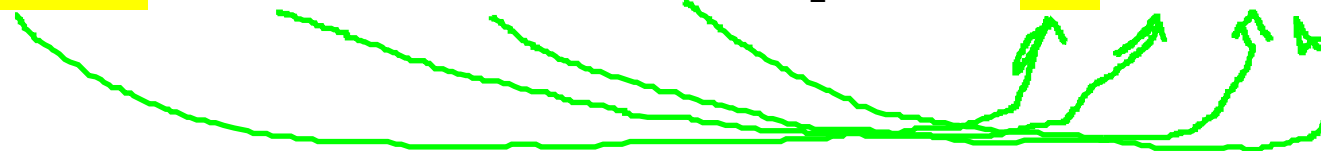


# IP: Klassenbasierte Adressierung

## ■ IPv4-Adresse

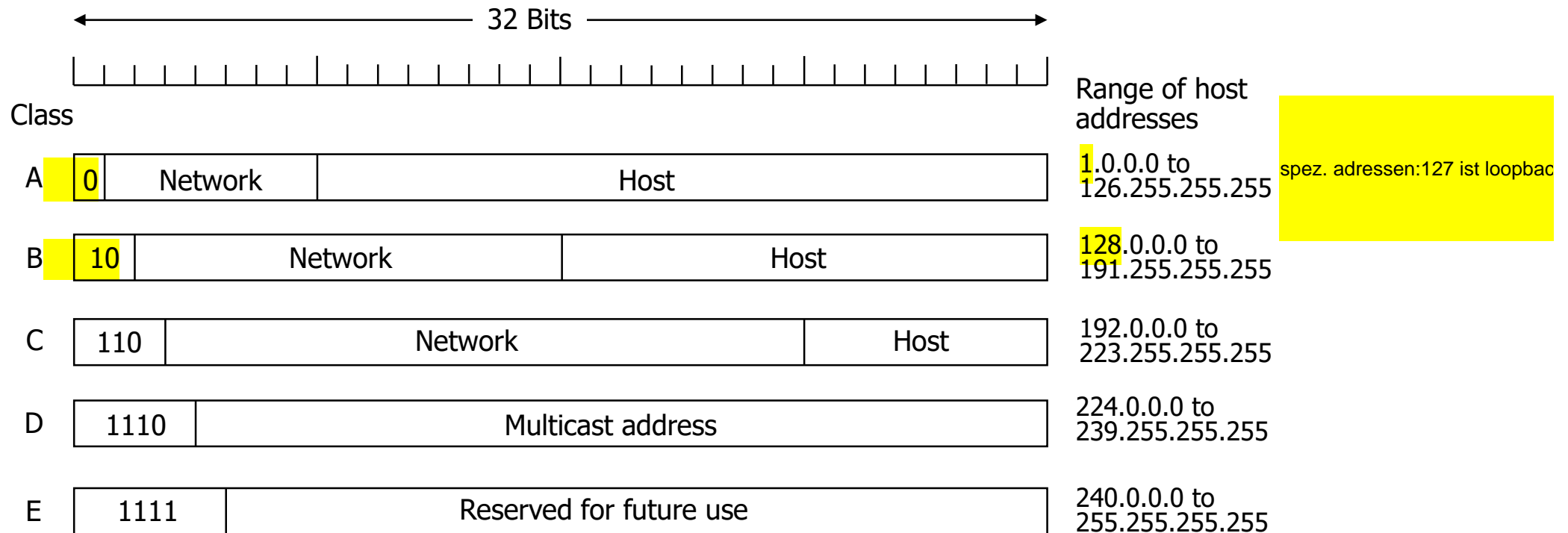
- kennzeichnet eine Schnittstelle eines Hosts oder Routers
- Hosts mit mehreren Schnittstellen (multi-homed) und Router benötigen mehrere IP-Adressen
- 32 Bits, 4 Bytes, Netzwerkteil und Hostteil
- zentrale Kontrolle durch
  - Internet Corporation for Assigned Names and Numbers (ICANN)
- autorisierte Address Registries vergeben an ISPs, diese wieder weiter
  - American Registry for Internet Numbers (ARIN),  
Reseaux IP Europeens (RIPE), ...
- Dotted-Decimal-Schreibweise
  - $d_1.d_2.d_3.d_4$  mit  $d_j$  = Dezimaldarstellung des j-ten Bytes
  - Bsp.: 10000000 10000111 01000100 00000101<sub>2</sub> wird als 128.135.68.5 notiert

IPv4 sind alle vergeben. Bei RIPE gibt es eine Wartelist, falls eine nicht mehr b



# IP: Klassenbasierte Adressierung

## ■ Klassenbasierte Adressen: früher



Quelle: Tanenbaum. Computer Networks. 5th Ed., Prentice Hall, 2011.

# IP: Klassenbasierte Adressierung

## ■ Besondere Adressen:

0 0
---

This host lokalhost

0 0	...	0 0	Host
-----	-----	-----	------

A host on this network host im lokalen netz

1 1
---

Broadcast on the local network an alle

Network	1 1 1 1	...	1 1 1 1
---------	---------	-----	---------

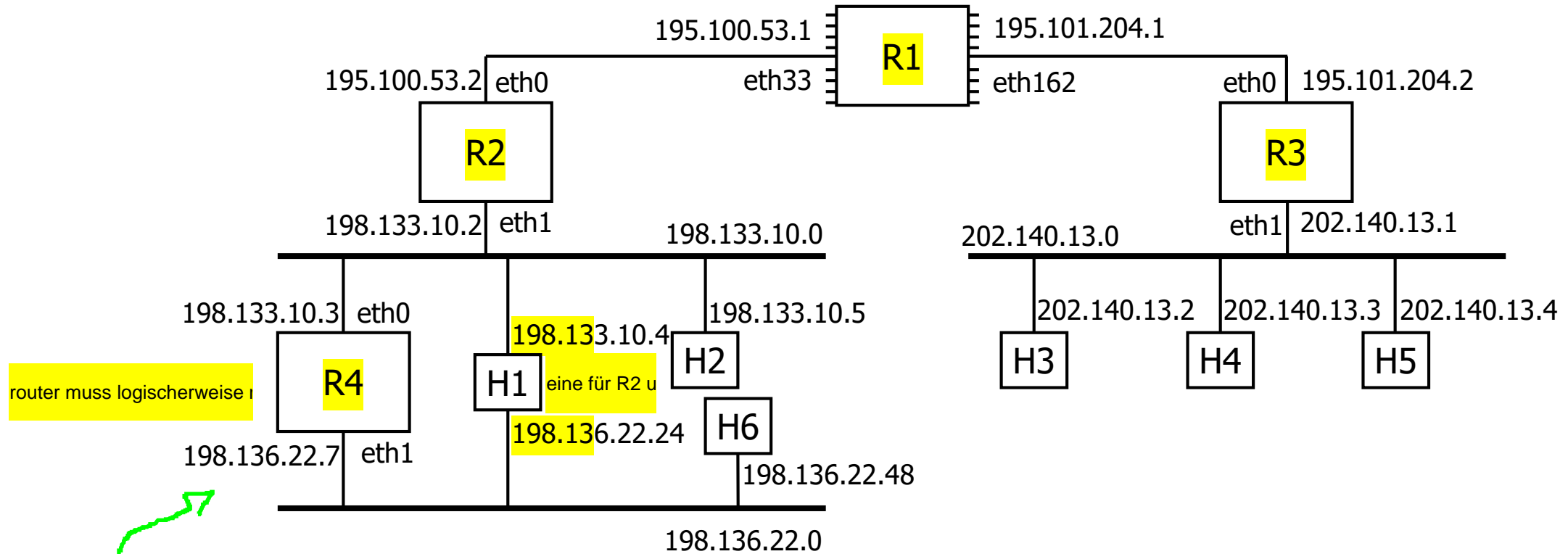
Broadcast on a distant network broadcast an alle im Network zielt

127	(Anything)
-----	------------

Loopback wird also direkt wieder zurück gesendet (out u

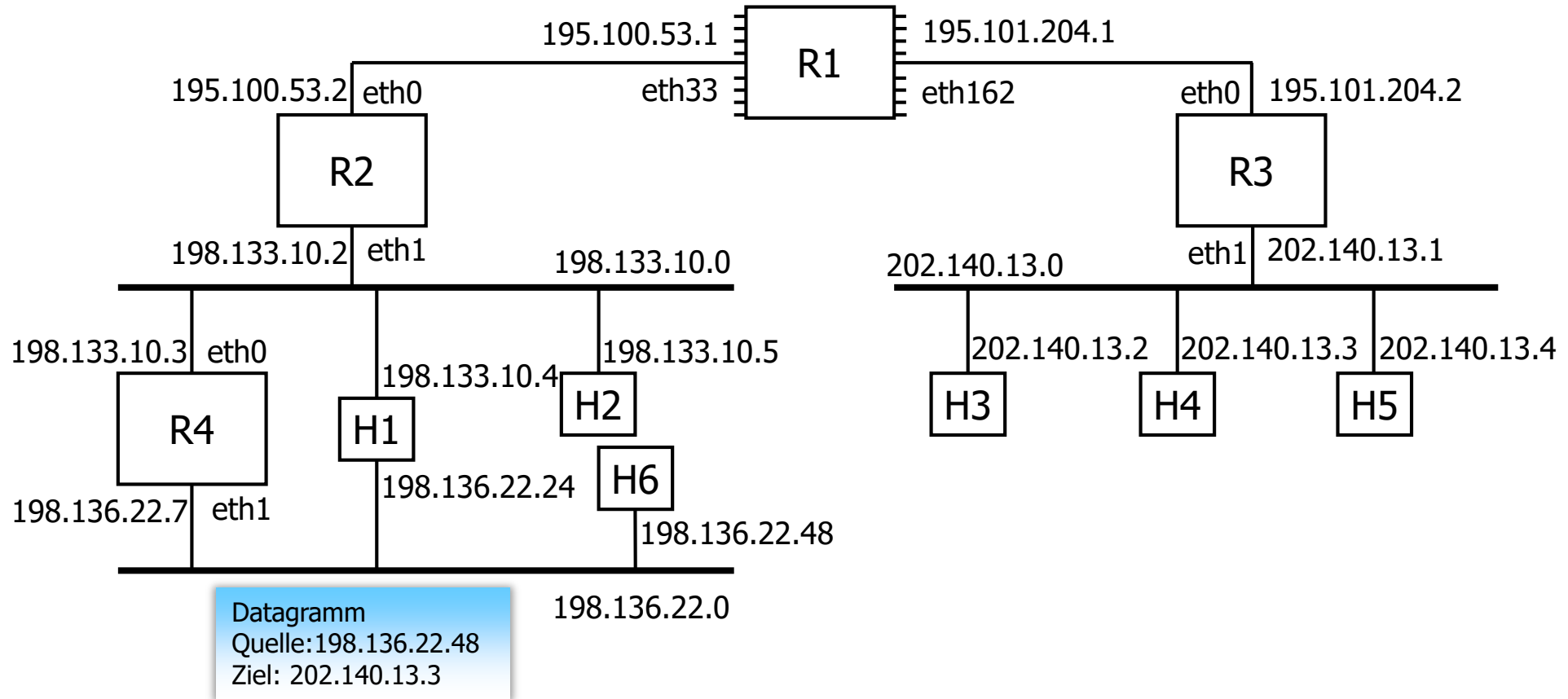
Quelle: Tanenbaum. Computer Networks. 5th Ed., Prentice Hall, 2011.

# IP: Klassenbasierte Adressierung



- Router R1, ..., R4, Hosts H1, ..., H6, Schnittstellen eth0, eth1, ...
- **Class-C-Netzwerke**: 198.133.10.0, 198.136.22.0, 202.140.13.0
- jedes Netzwerk ohne Router z.B. realisiert durch Ethernet, Switch-Struktur (früher Busleitung), wird als Balken abstrahiert
- wie findet ein IP-Datagramm von H6 nach H4?

# IP: Klassenbasierte Adressierung



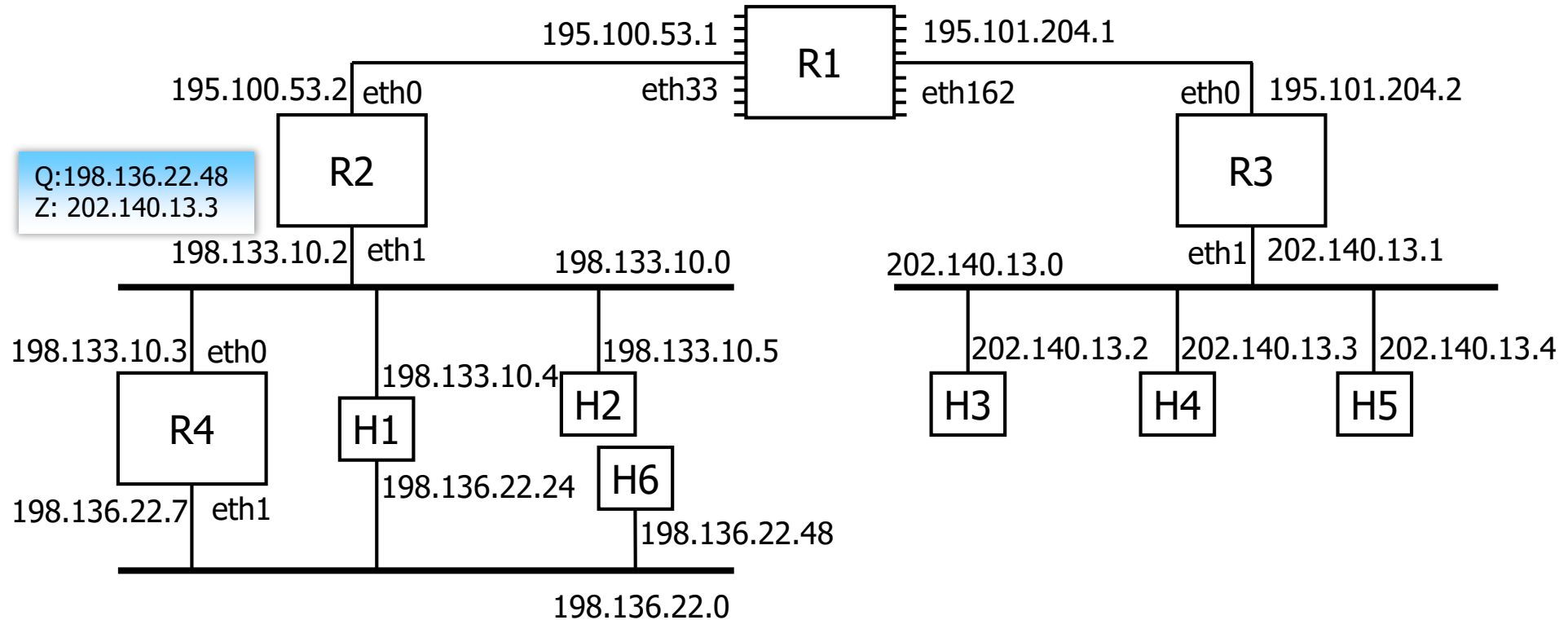
Weiterleitungstabelle von H6:

localhost

sonst "außenwelt"

Ziel	nächster Hop	Flag	Schnittstelle
198.136.22.0	-		eth0
default	198.136.22.7	G	eth0

# IP: Klassenbasierte Adressierung

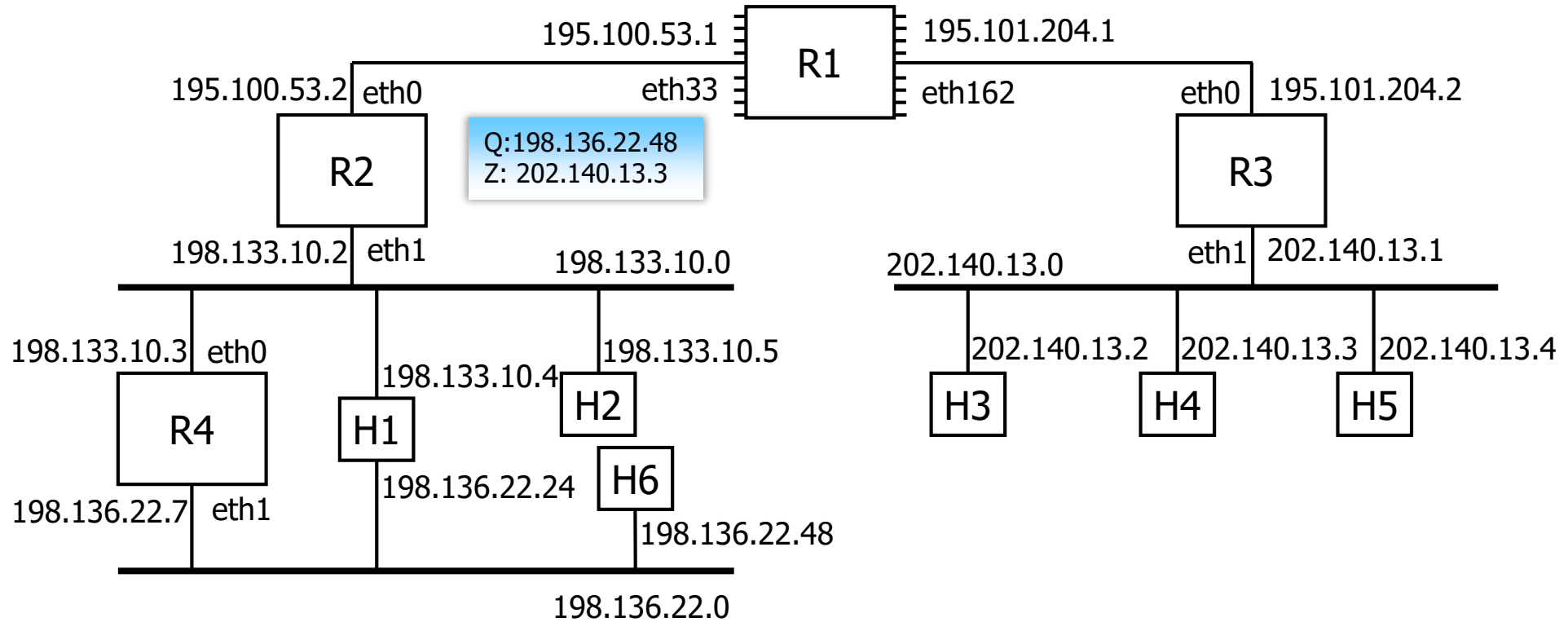


Weiterleitungstabelle von R4:

Ziel	nächster Hop	Flag	Schnittstelle
198.136.22.0	-		eth1
198.133.10.0	-		eth0
default	198.133.10.2	G	eth0

broadcast in höh

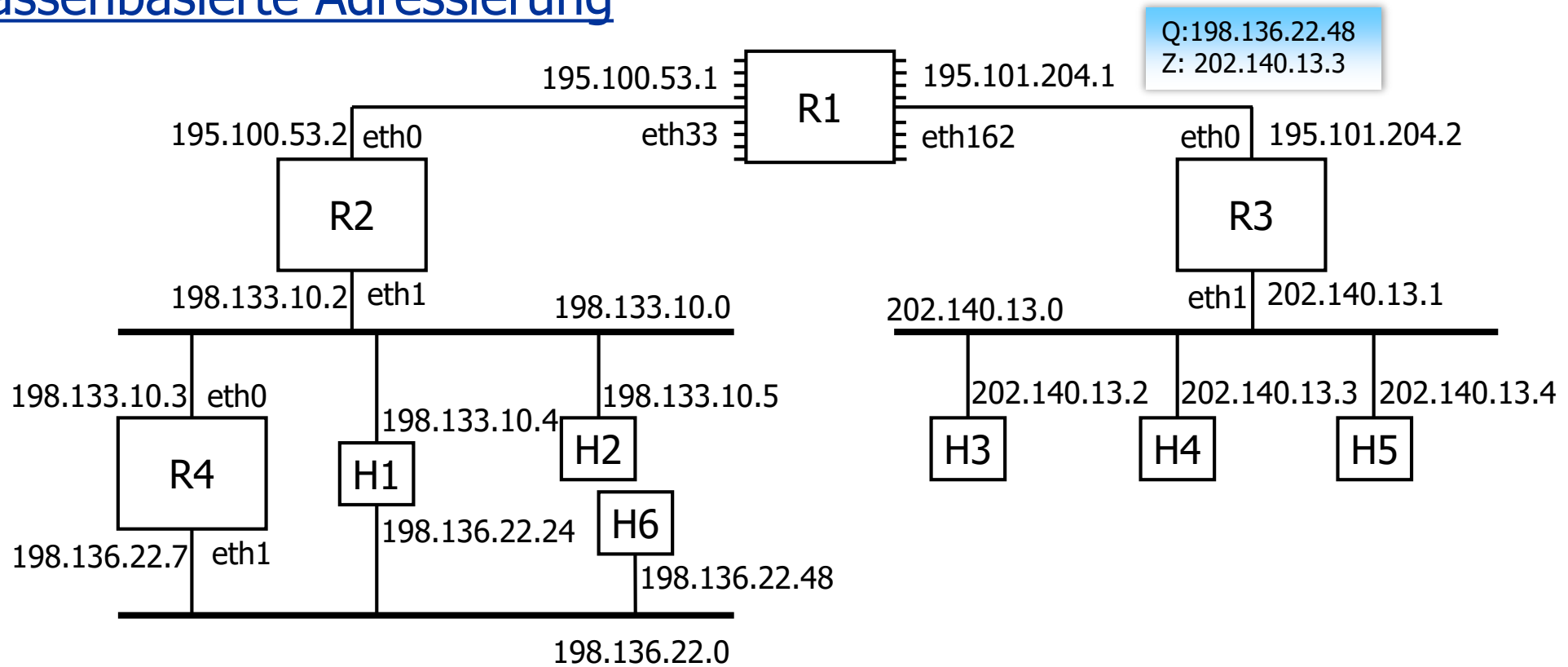
## IP: Klassenbasierte Adressierung



Weiterleitungstabelle von R2:

Ziel	nächster Hop	Flag	Schnittstelle
195.100.53.0	-		eth0
198.133.10.0	-		eth1
198.136.22.0	198.133.10.3	G	eth1
default	195.100.53.1	G	eth0

## IP: Klassenbasierte Adressierung

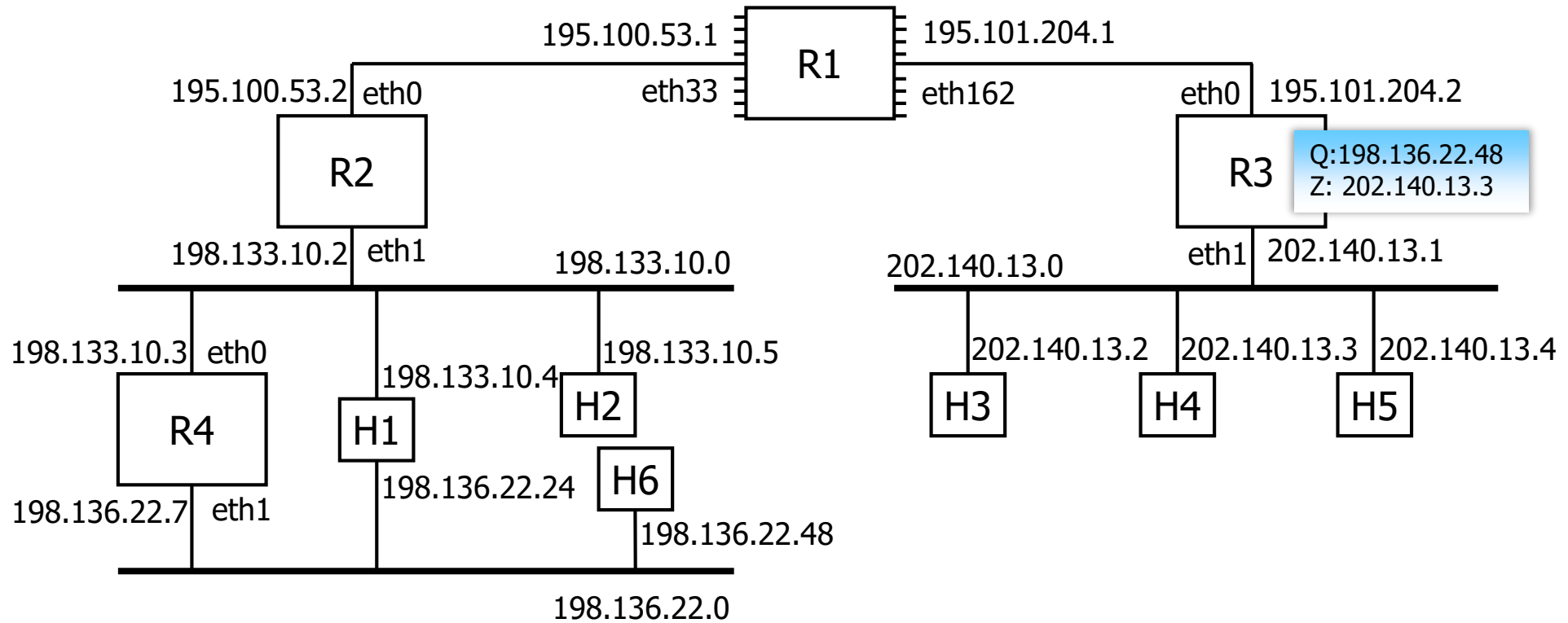


Weiterleitungstabelle von R1:

Ziel	nächster Hop	Flag	Schnittstelle
198.136.22.0	195.100.53.2	G	eth33
198.133.10.0	195.100.53.2	G	eth33
195.100.53.0	-		eth33
195.101.204.0	-		eth162
202.140.13.0	195.101.204.2	G	eth162



## IP: Klassenbasierte Adressierung



Weiterleitungstabelle von R3:

Ziel	nächster Hop	Flag	Schnittstelle
195.101.204.0	-		eth0
202.140.13.0	-		eth1
default	195.101.204.1	G	eth0

# IP: Klassenbasierte Adressierung

## ■ Vorteil der klassenbasierten Adressierung

- **selbstidentifizierende Adressen**: an den ersten Bits wird erkannt, um welche Klasse es sich handelt
- Weiterleitungstabelle benötigt nur Netzwerkteil der Adresse und kann klein gehalten werden

## ■ Nachteile der klassenbasierten Adressierung

- feste Zuordnung von Netzwerken, wenn ein Rechner in ein anderes Netzwerk zieht, muss seine IP-Adresse angepasst werden
- C-Netze erlauben nur wenige Hosts (8 Bits), B-Netze erlauben sehr viele Hosts (16 Bits)
- Verschwendung: größere Organisationen bemühen sich um B-Netz und nutzen Adressen nur teilweise
- Anfang der 90er Jahre war absehbar, dass der Adressraum für weiteres Wachstum nicht ausreicht
- 2011 hat ICANN letzte verbleibende IPv4-Adressen ausgegeben

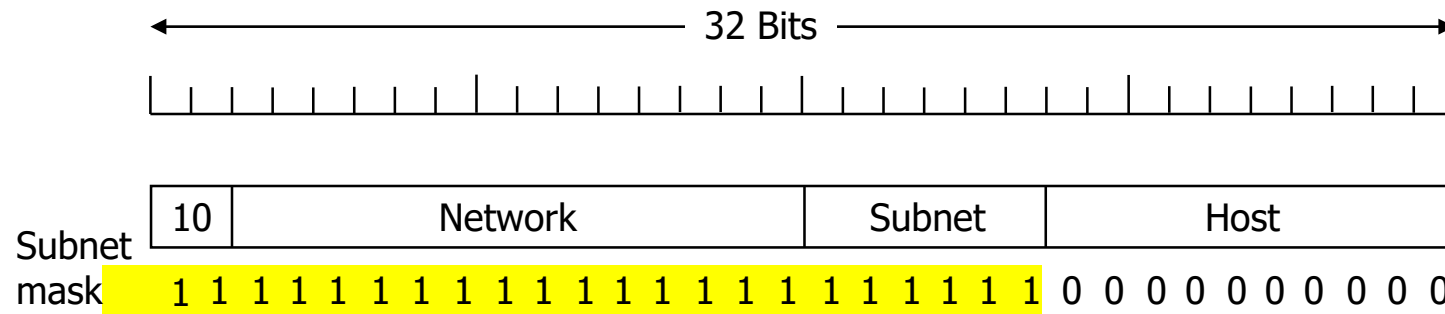
# Netzwerkschicht

- ✓ Einführung
- ✓ IP
  - ✓ Datagrammformat und klassenbasierte Adressierung
  - Subnetze und klassenlose Adressierung
  - Fragmentierung
  - ICMP
  - Autokonfiguration mit DHCP
  - NAT
  - IPv6
- Aufbau eines Routers
- Routing
- IPsec
- Netzvirtualisierung

# IP: Subnetze und klassenlose Adressierung

## ■ Subnetze

- Hostanteil wird weiter unterteilt in Subnetz (variable Länge) und Host
- entfernte Router benötigen nur den weiter klassenbasierten Netzwerkteil
- Maske: führende Bits vom Wert 1, „Verundung“ mit Adresse ergibt Netzwerkteil und Subnetzteil
- Notation: IP-Adresse und Subnetzmaske, z.B.: 150.100.12.176 und 255.255.255.128



Quelle: Tanenbaum. Computer Networks. 5th Ed., Prentice Hall, 2011.

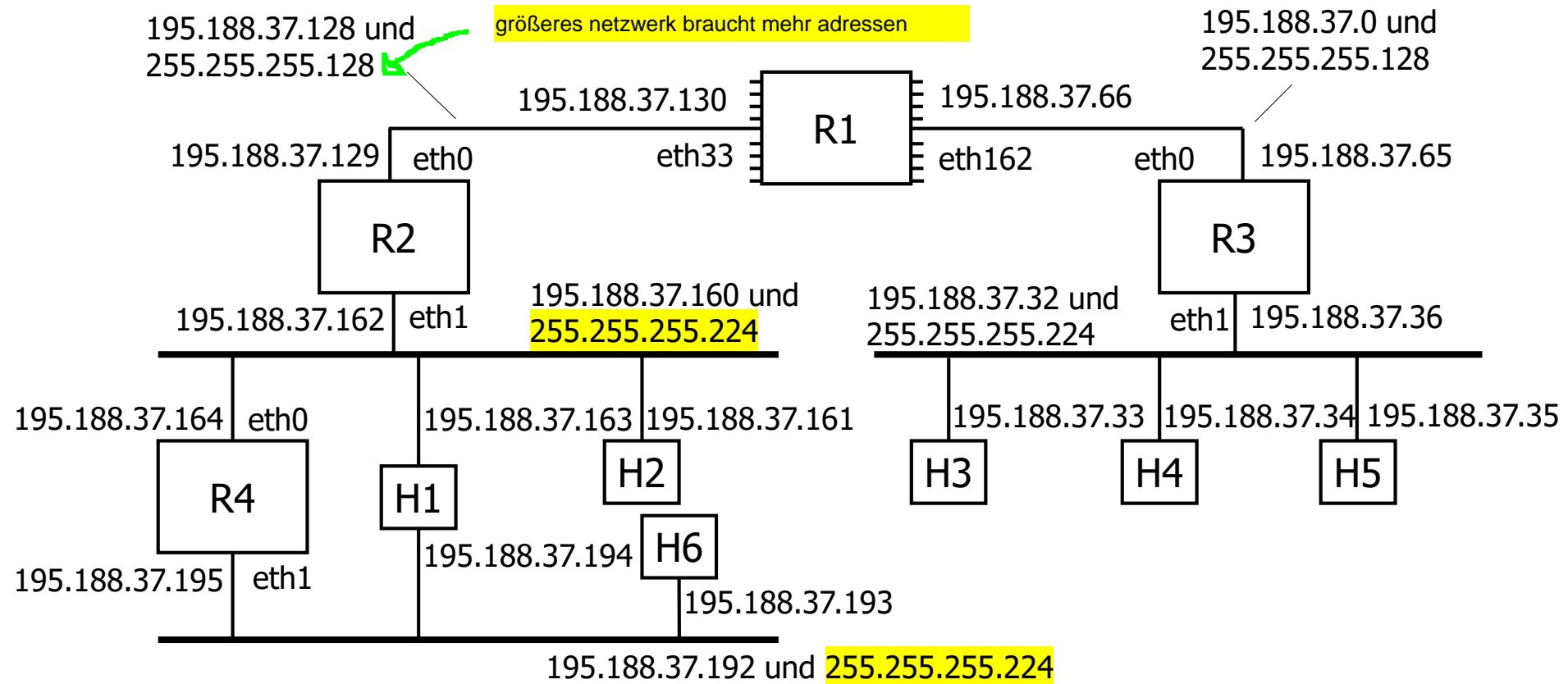
# IP: Subnetze und klassenlose Adressierung

## ■ Beispiel für Subnetz

- Organisation hat Klasse-B-Adresse (also 16 Bits Hostteil) mit Netzwerkteil: 150.100.0.0
- es werden Subnetze mit jeweils maximal 100 Hosts erzeugt
- hierfür reichen jeweils 7 Bits
- also  $16-7=9$  Bits für Subnetzteil
- z.B. IP-Adresse 150.100.12.176 und Maske 255.255.255.128
- binär = 10010110 01100100 00001100 10110000
- Maske = 11111111 11111111 11111111 10000000
- AND = 10010110 01100100 00001100 10000000
- Subnetzadresse = 150.100.12.128

16+7 = 25 bits

## IP: Subnetze und klassenlose Adressierung



- das vorherige klassenbasierte Beispiel könnte mit Subnetzen so realisiert werden
- es wird nur ein Class-C-Netzwerk verwendet: 195.188.37.0

# IP: Subnetze und klassenlose Adressierung

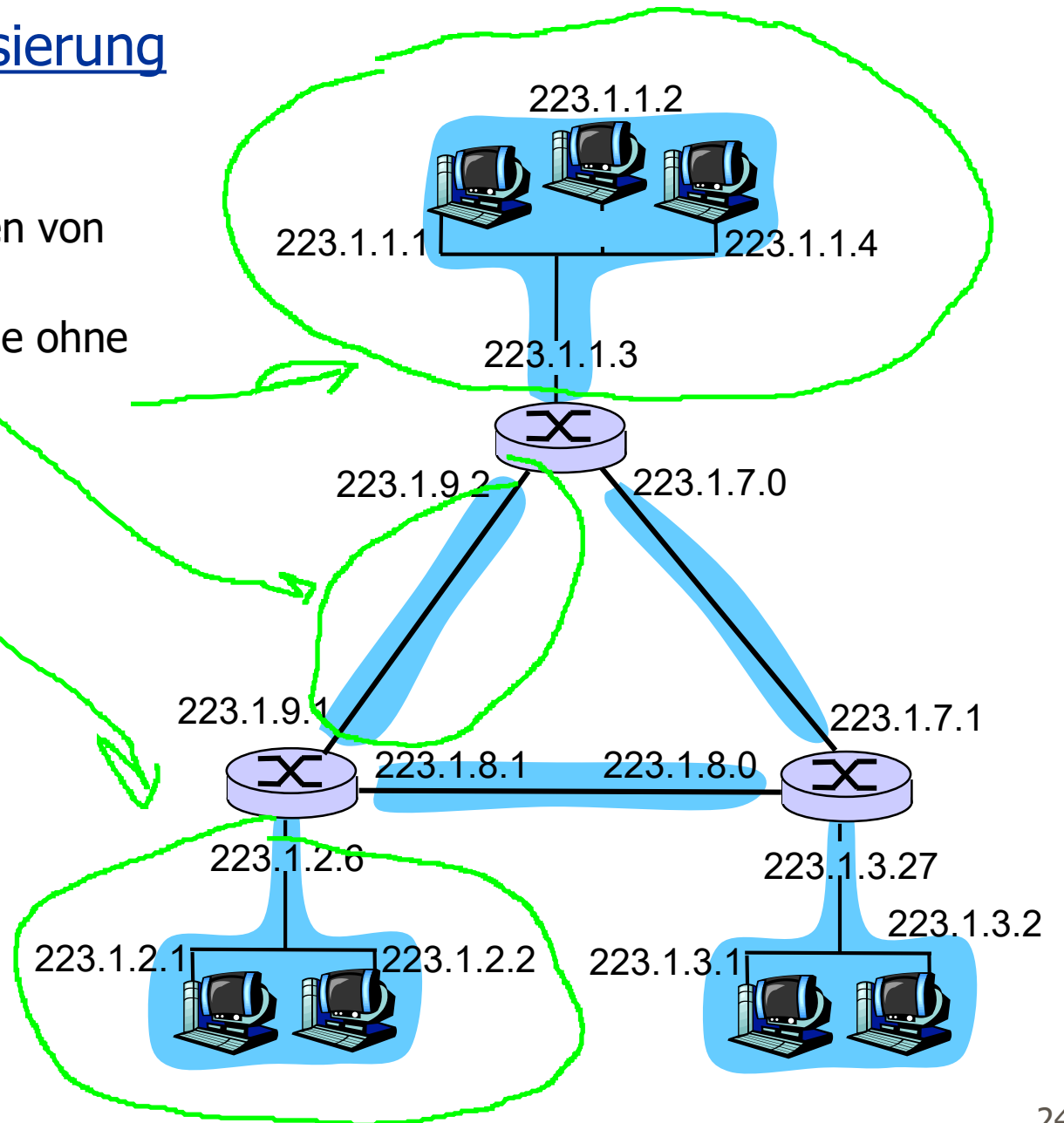
## ■ CIDR (Classless Inter-Domain Routing)

- neben Subnetting auch Supernetting: auch Zusammenfassen von aufeinanderfolgenden klassenbasierten Netzen
- Notation: Prefix/X mit Prefix als IP-Adressteil und X Prefix-Länge, z.B.:
  - 150.100.12.176/25 (= 150.100.12.176 und 255.255.255.128) erste 25 auf 1 gesetzt
  - 198.136.22/24 (= 198.136.22.0) normale klass-c adresse, kleinste einheit die vor
  - 192.4.16/20 (= 192.4.16.0 ... 192.4.31.255)
- im Ergebnis ist Subnetz durch Prefix und Prefix-Länge gegeben, diese werden als Einträge in die Weiterleitungstabelle geschrieben
- die starre Zuordnung von klassenbasierten Adressen ist aufgelöst, aber größere Tabellen
- Router müssen Longest-Prefix-Match durchführen: sie vergleichen die Adresse mit Übereinstimmungen in der Weiterleitungstabelle und wählen diejenige, bei denen die meisten Bits übereinstimmen
- spezielle Datenstrukturen (Varianten von Binärbaumen) oder spezielle Hardware für effizienten Zugriff

# IP: Subnetze und klassenlose Adressierung

## ■ Subnetze topologisch

- gedankliche Abtrennung der Schnittstellen von Routern
- Subnetze = zusammenhängende Bereiche ohne Router



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.



# Netzwerkschicht

- ✓ Einführung
- ✓ IP
  - ✓ Datagrammformat und klassenbasierte Adressierung
  - ✓ Subnetze und klassenlose Adressierung
  - Fragmentierung
  - ICMP
  - Autokonfiguration mit DHCP
  - NAT
  - IPv6
- Aufbau eines Routers
- Routing
- IPsec
- Netzvirtualisierung

# IP: Fragmentierung

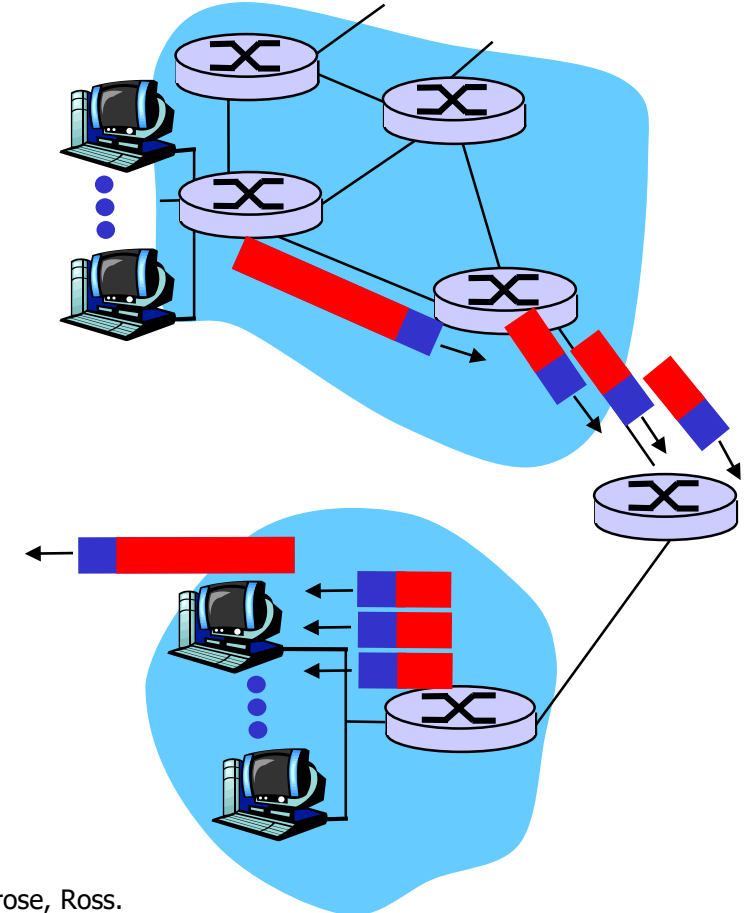
## ■ Fragmentierung

- wenn Verbindungen unterwegs eine kleinere **MTU (Maximum Transmission Unit)** erfordern, wird das Datagramm in Fragmente zerlegt und als kleinere Datagramme weitergeleitet
- dies kann sich u.U. mehrmals wiederholen
- IP-Header-Felder
  - **identifizier**: Kennzeichnung zusammengehöriger Fragmente
  - **flag**: Fragment folgt
  - **offset**: Position der Daten des Fragments bei  $\text{offset} \cdot 8$

## ■ Reassemblierung

- erst am Ziel wird wieder zusammengesetzt

wie mss nur das header mitbetrachtet wird



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

# IP: Fragmentierung

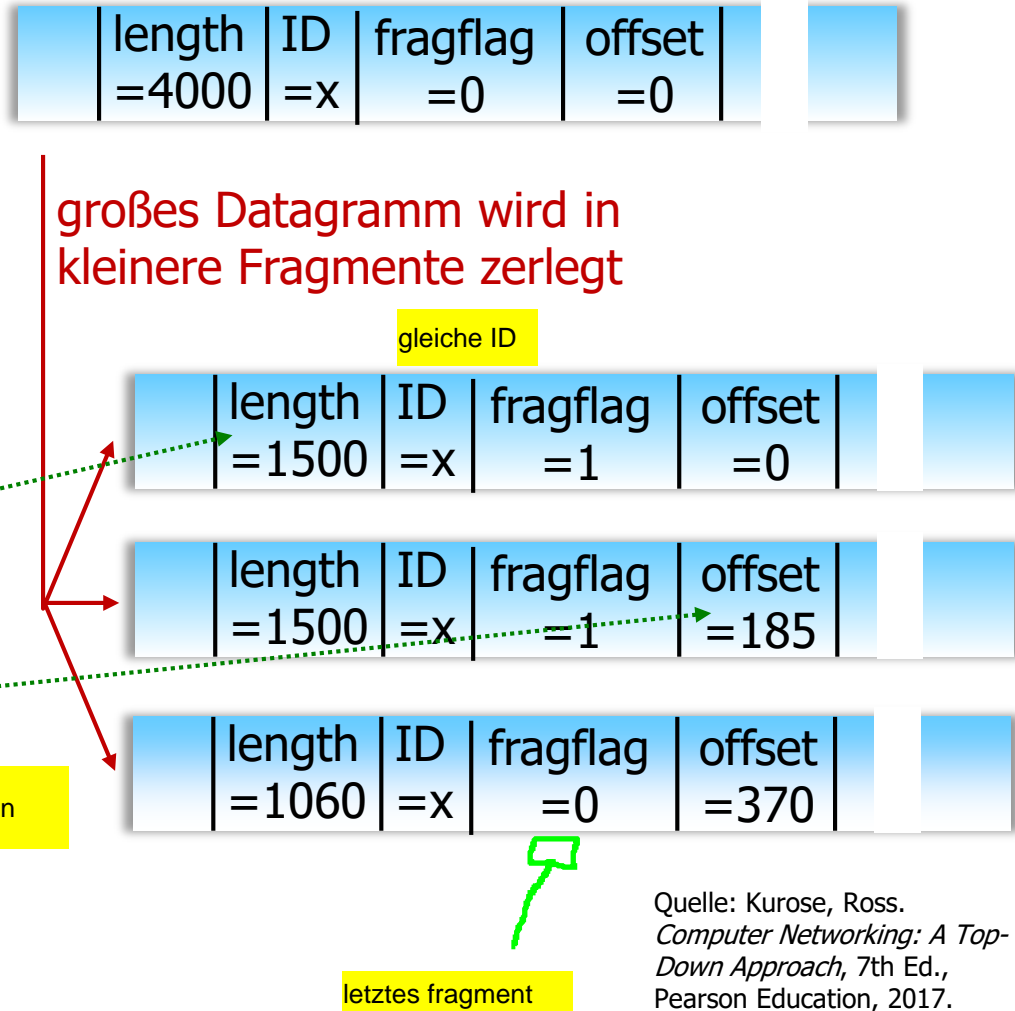
## ■ Beispiel

- 4000 Byte Datagram
- MTU = 1500 Bytes

1480 Bytes im  
Datenfeld

Offset =  
 $1480/8$

also in byte nicht bit angeben



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
  - ✓ Datagrammformat und klassenbasierte Adressierung
  - ✓ Subnetze und klassenlose Adressierung
  - ✓ Fragmentierung
  - ICMP
  - Autokonfiguration mit DHCP
  - NAT
  - IPv6
- Aufbau eines Routers
- Routing
- IPsec
- Netzvirtualisierung

# IP: ICMP

## ■ ICMP: Internet Control Message Protocol

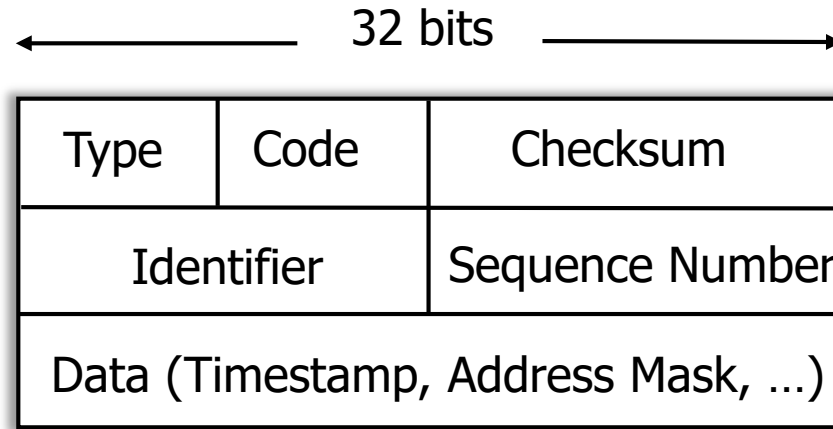
- Kontrollnachrichten von Routern an andere Router und Hosts
- z.B. Benachrichtigung über Fehler (unerreichbare Adresse, maximale Zahl von Hops erreicht, ...), ping, ...
- z.B. für Traceroute:  
IP-Pakete mit TTL = 1, 2, ...,  
Router verwerfen und antworten

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

Type	Code	description
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)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

## IP: ICMP

### ■ Format:



- 2. Zeile und Datenteil abhängig von Typ
- werden in IP-Datagrammen befördert
- bei Bezug auf ein IP-Paket enthält der Datenteil den IP-Header und die ersten 64 Bits des Pakets

direkt über IP ohne transportprotokoll

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
  - ✓ Datagrammformat und klassenbasierte Adressierung
  - ✓ Subnetze und klassenlose Adressierung
  - ✓ Fragmentierung
  - ✓ ICMP
    - Autokonfiguration mit DHCP
    - NAT
    - IPv6
- Aufbau eines Routers
- Routing
- IPsec
- Netzvirtualisierung

# IP: Autokonfiguration mit DHCP

## ■ Dynamic Host Configuration Protocol (DHCP)

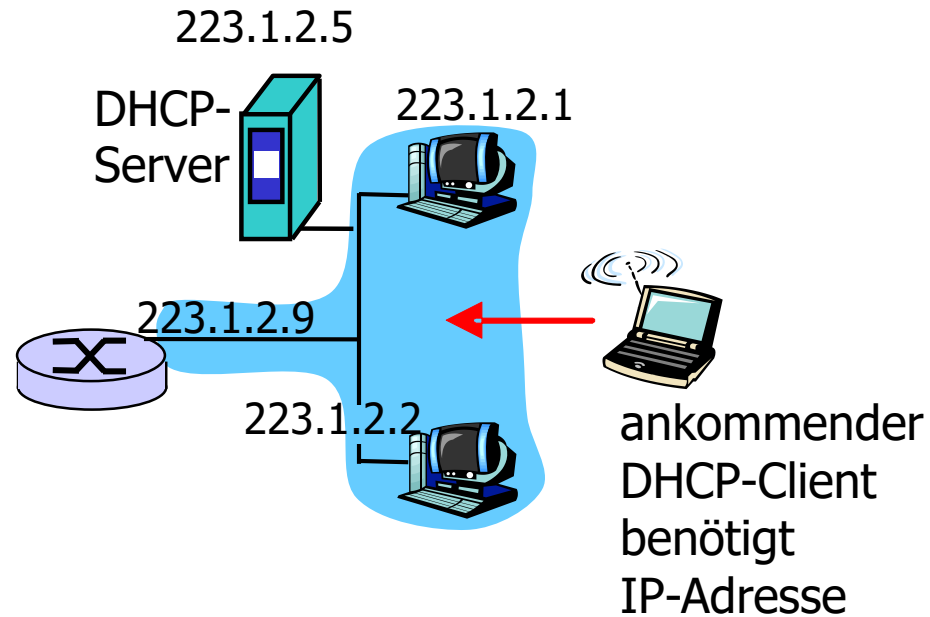
- Client-Server-Protokoll zum automatischen Bezug einer IP-Adresse
- auch für weitere Parameter wie z.B. Router und DNS-Server
- **DHCP-Server** im Subnetz (oder DHCP-Relay, der DHCP-Server kennt) wenn im subnetz kein DHCP server existiert
- 4 Schritte mit UDP-Paketen
  - **DHCP server discovery:** von 0.0.0.0 an 255.255.255.255 broadcast der Suche
  - **DHCP server offers:** von IP-Adresse des DHCP-Servers an 255.255.255.255 (evtl. von mehreren DHCP-Servern) also broadcast der DHCP server adressen
  - **DHCP request:** von 0.0.0.0 an IP-Adresse des ausgewählten DHCP-Servers einer der u.u mehreren wird ausgewählt durch
  - **DHCP Ack:** vom ausgewählten DHCP-Server an 255.255.255.255, enthält die IP-Adresse **yiaddr** und ggfs. weitere Parameter antwort mit eigener IP vom DHCP
- bis auf DHCP request werden **alle per Broadcast im Subnetz auf Ebene der Sicherungsschicht gesendet**
- ggfs. noch Absicherung durch **ARP-Request (Address Resolution Protocol)**, siehe nächstes Kapitel

ip zu physikalische adresse



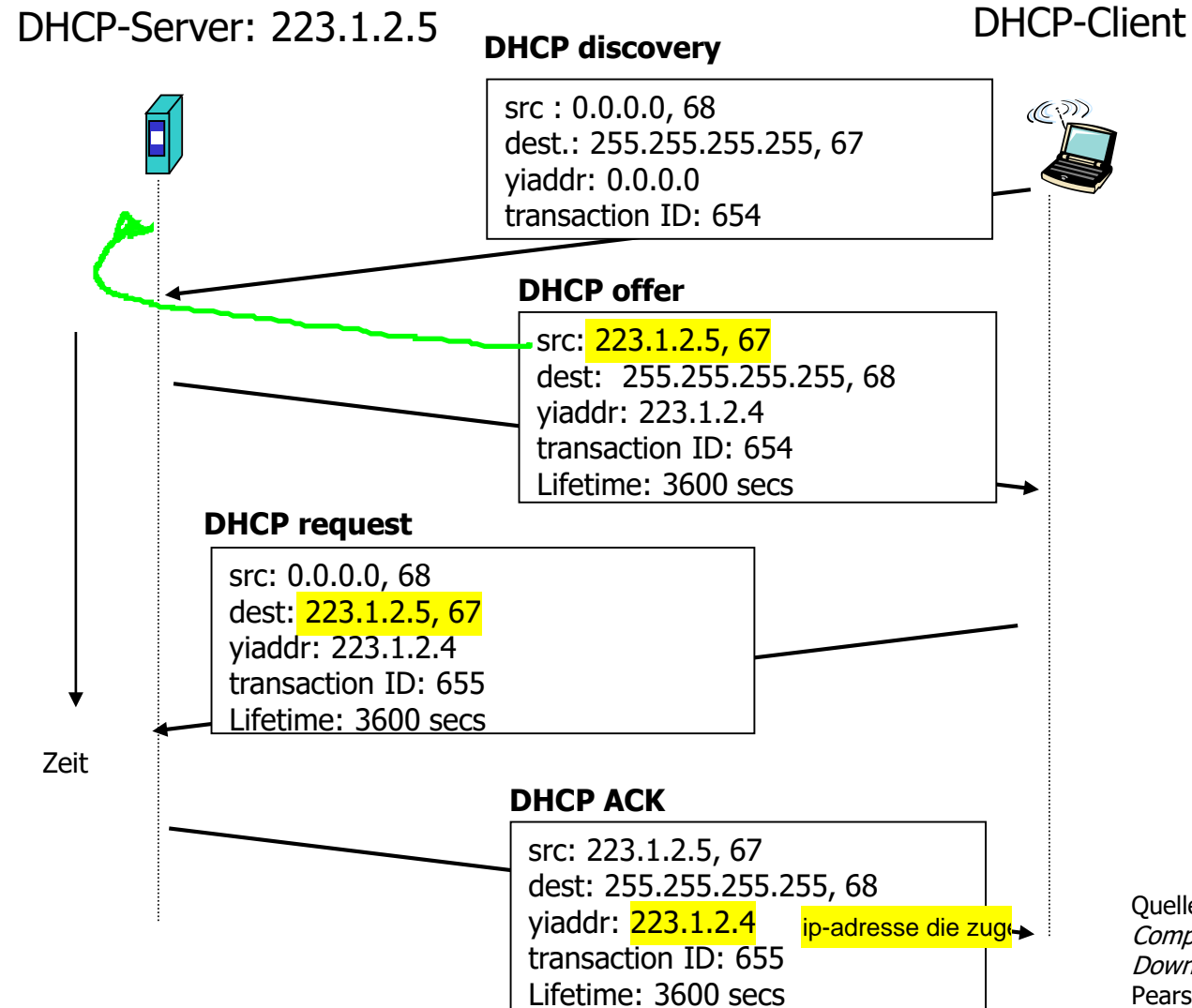
# IP: Autokonfiguration mit DHCP

## ■ Beispielkonfiguration für DHCP:



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

# IP: Autokonfiguration mit DHCP



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

# Netzwerkschicht

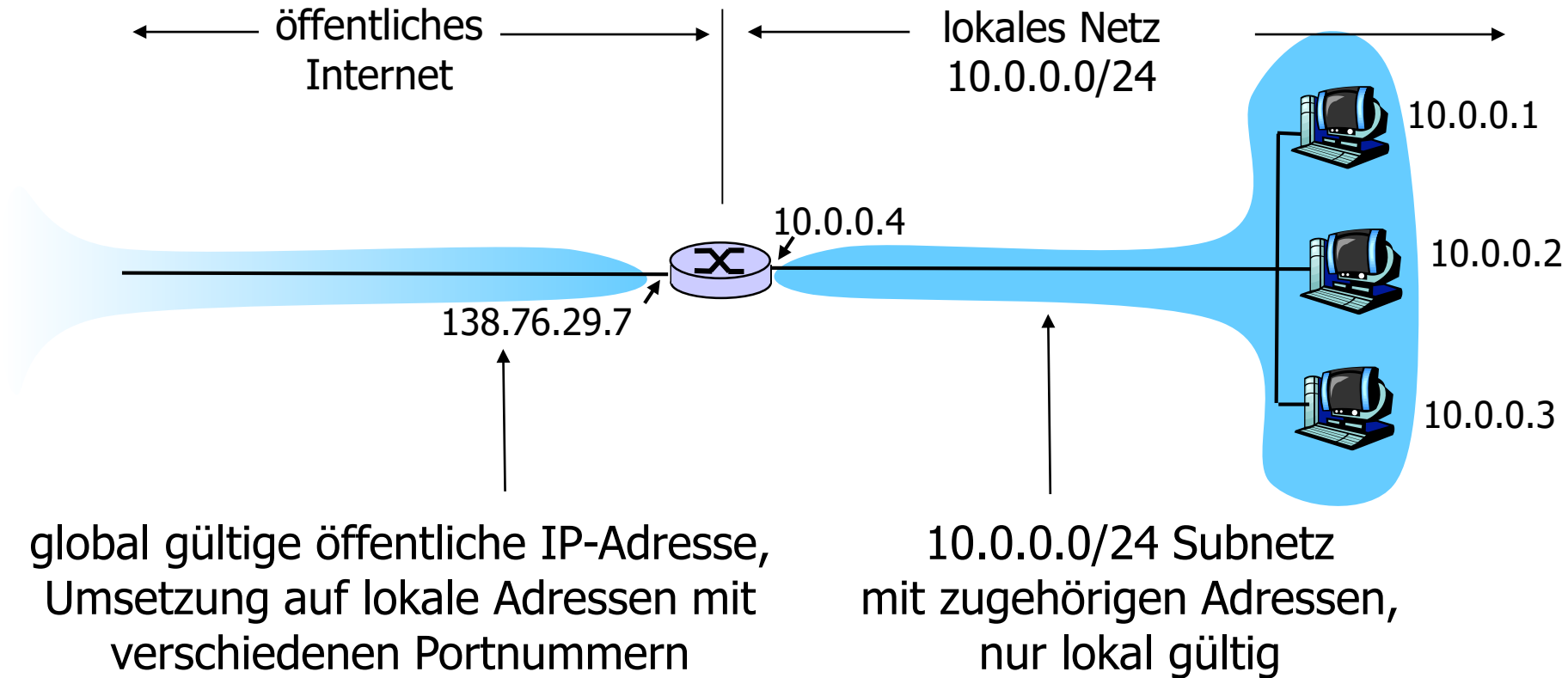
- ✓ Einführung
- ✓ IP
  - ✓ Datagrammformat und klassenbasierte Adressierung
  - ✓ Subnetze und klassenlose Adressierung
  - ✓ Fragmentierung
  - ✓ ICMP
  - ✓ Autokonfiguration mit DHCP
  - NAT
  - IPv6
- Aufbau eines Routers
- Routing
- IPsec
- Netzvirtualisierung

# IP: NAT

## ■ NAT (Network Address Translation)

- Umgehung der Adressknappheit man kann noch nicht auf IPv6 umstellen (NACH 24 JAHREN!!!)
- Netzwerk verwendet intern global ungültige Adressen, z.B. 10.0.0.0/24
- Netzwerk besitzt eine global gültige IP-Adresse
- Verbindungen zu internen Hosts werden auf Paare abgebildet, die aus dieser Adresse und einem Port bestehen
- NAT-Router muss die Abbildung ausführen, er besitzt Tabelle hierfür und überschreibt Adressen und Ports in den IP-Datagrammen
- Größe durch Anzahl von Portnummern begrenzt
- Nachteile: Verletzung des Schichtenprinzips (Router beschäftigt sich mit Hosts, Ports sind für Dienste zwischen Transport- und Anwendungsschicht gedacht), Eingriff in die Ende-zu-Ende-Verbindung, Hosts hinter NAT können nicht als Server auftreten (NAT-Traversal benötigt Tricks)
- Vorteil: interne Änderungen ohne externe Auswirkung, bessere Abschirmung
- IPv6 wäre eigentlich besser, NAT ist dennoch sehr verbreitet

# IP: NAT



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

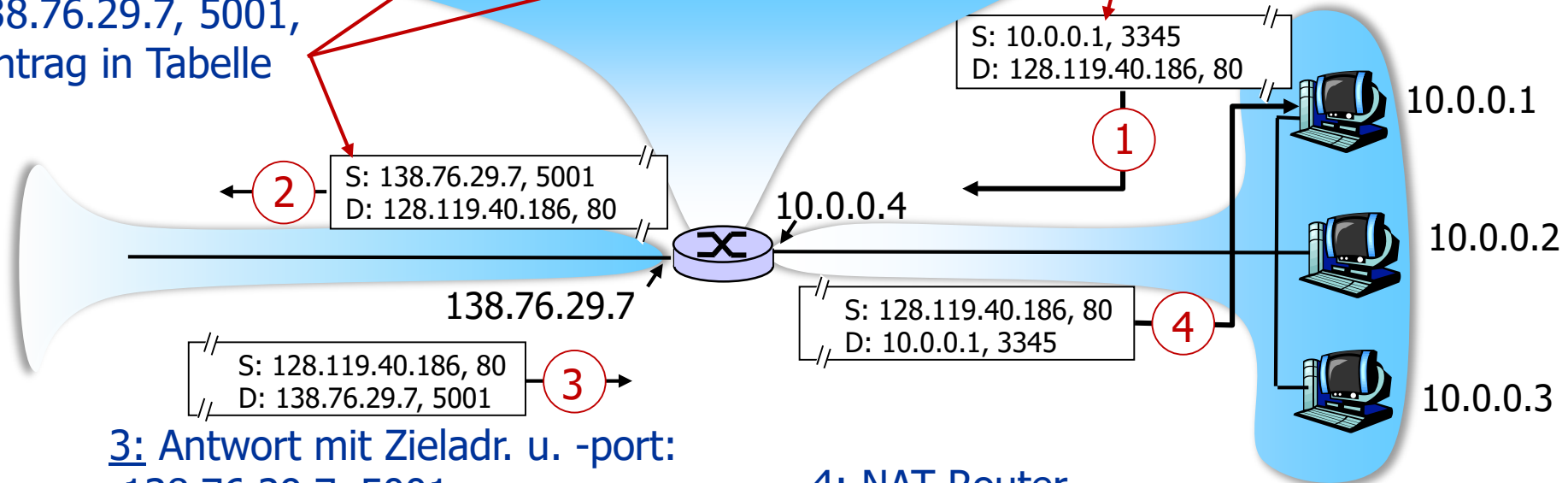
# IP: NAT

spricht die eigentliche anfrage wird vom Router gestellt, der das dann i

**2: NAT Router  
überschreibt  
Quelladr. u. -port  
10.0.0.1, 3345 mit  
138.76.29.7, 5001,  
Eintrag in Tabelle**

NAT-Tabelle	
Internet	LAN
138.76.29.7, 5001	10.0.0.1, 3345
.....	.....

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



**3: Antwort mit Zieladr. u. -port:  
138.76.29.7, 5001**

**4: NAT Router  
überschreibt Zieladr. u. -port  
138.76.29.7, 5001 mit 10.0.0.1, 3345**

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
  - ✓ Datagrammformat und klassenbasierte Adressierung
  - ✓ Subnetze und klassenlose Adressierung
  - ✓ Fragmentierung
  - ✓ ICMP
  - ✓ Autokonfiguration mit DHCP
  - ✓ NAT
  - IPv6
- Aufbau eines Routers
- Routing
- IPsec
- Netzvirtualisierung

# IP: IPv6

## ■ IPv6 (IP version 6)

- ursprünglicher Name: Internet Protocol Next Generation (IPng)
- IPv5 war experimentelles Protokoll
- IETF-Standardisierung initiiert wegen Adressraumproblemen von IPv4, dabei gleich Lösung weiterer Probleme
  - Header mit fester Länge (für schnelles Weiterleiten)
  - keine Fragmentierung (wird einfach verworfen falls > MTU)
  - keine Prüfsumme (Fehlererkennung in höheren Schichten)
  - zusätzliche Optionen außerhalb als nächster Header also header konstant mit einem pointer" next header "
  - zustandslose Autokonfiguration
  - Dienstgüte (neben DS und ECN auch Flow-Label)
  - Informationssicherheit ipsec
- anfängliche Probleme bei der Durchsetzung, inzwischen „kann“ IPv4 vieles auch, inzwischen beschleunigte Verbreitung



# IP: IPv6

## ■ Adresskategorien

- Unicast: an ein Ziel
- Multicast: an mehrere Ziele
- Anycast: an ein Ziel in einer Gruppe

## ■ Notation

- Gruppen von 16 Bits, repräsentiert durch 4 hexadezimale Ziffern, getrennt durch Doppelpunkt
- z.B. 4BF5:AA12:0216:FEBC:BA5F:039A:BE9A:2176
- Kurzformen:
  - 4BF5:0000:0000:0000:BA5F:039A:000A:2176
  - Nullen kompakter: 4BF5:0:0:0:BA5F:39A:A:2176
  - Nullen weglassen: 4BF5::BA5F:39A:A:2176 (nur einmal)
- gemischte Notation:
  - die letzten 32 Bits sind oft IPv4-Adresse, Darstellung durch ::FFFF:128.155.12.198

128bits



ab jetzt IPv4

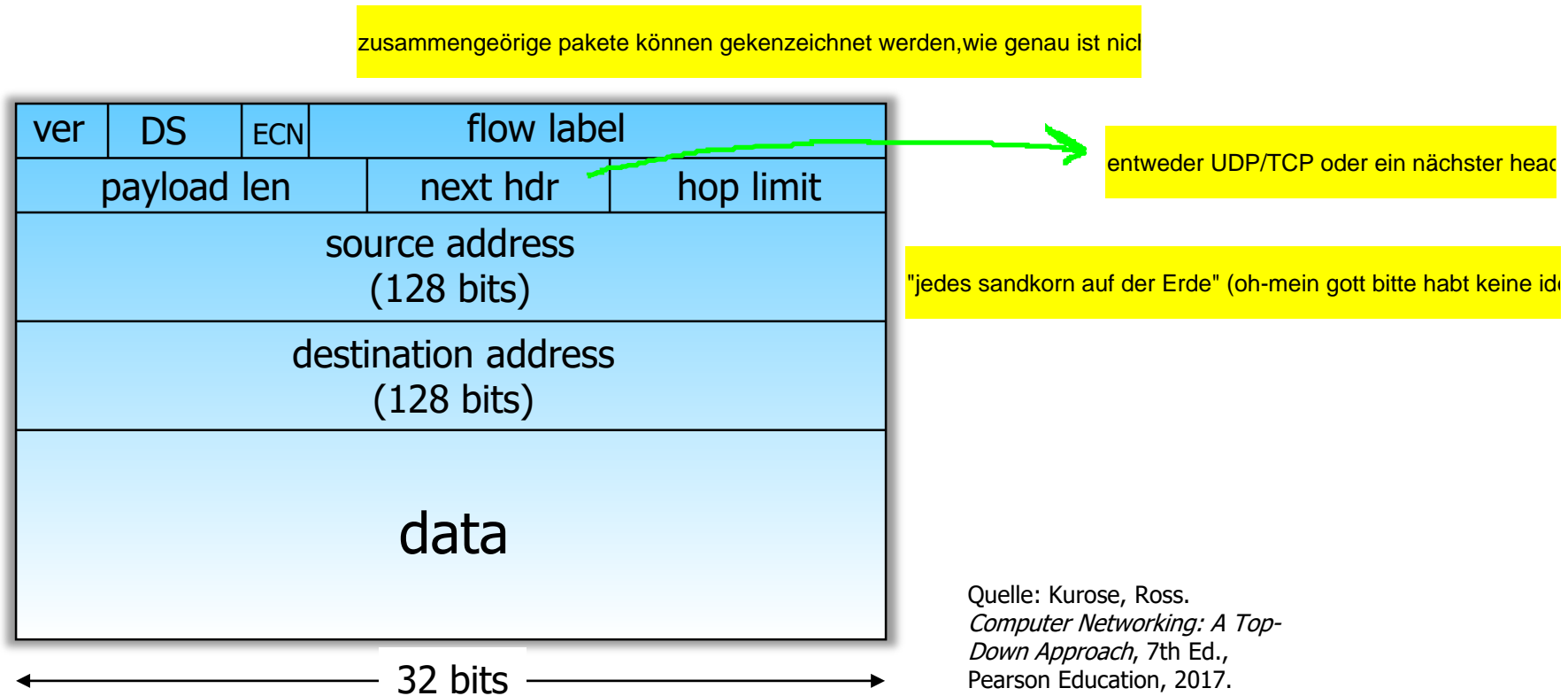


## ■ DNS: AAAA-Einträge für IPv6-Adressen, längere Präfixe

# IP: IPv6

## ■ Datagrammformat

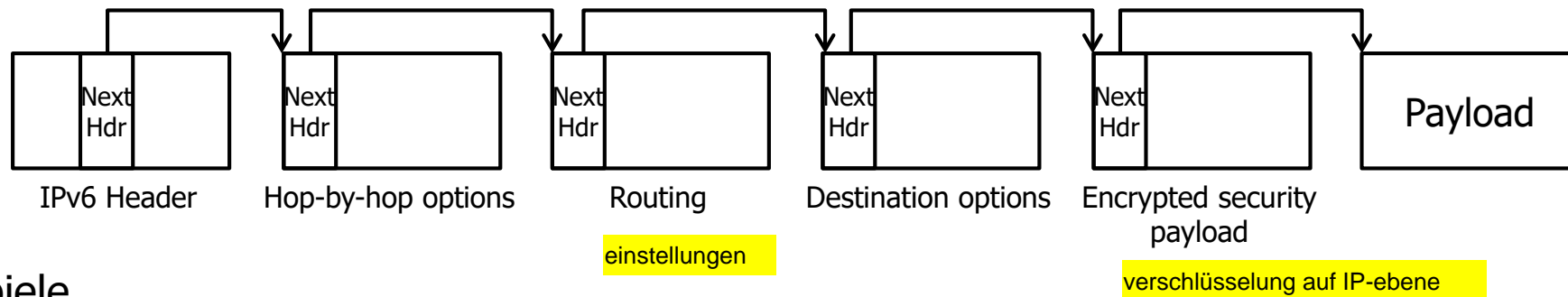
- Adressen sind 128 Bits lang, „das sollte eine Weile reichen“



# IP: IPv6

## ■ Header Konzept

- für jede Aufgabe ein eigener „Extension Header“ fester Größe
- schnelle Verarbeitung, jedes System sieht nur in die benötigten bzw. bekannten Header




### • Beispiele

- TCP (6)
- UDP (17)
- IPsec ESP (50), IPsec AH (51)
- ICMPv6 (58)
- SCTP (132)

MTU als einzige theoretische obergrenze

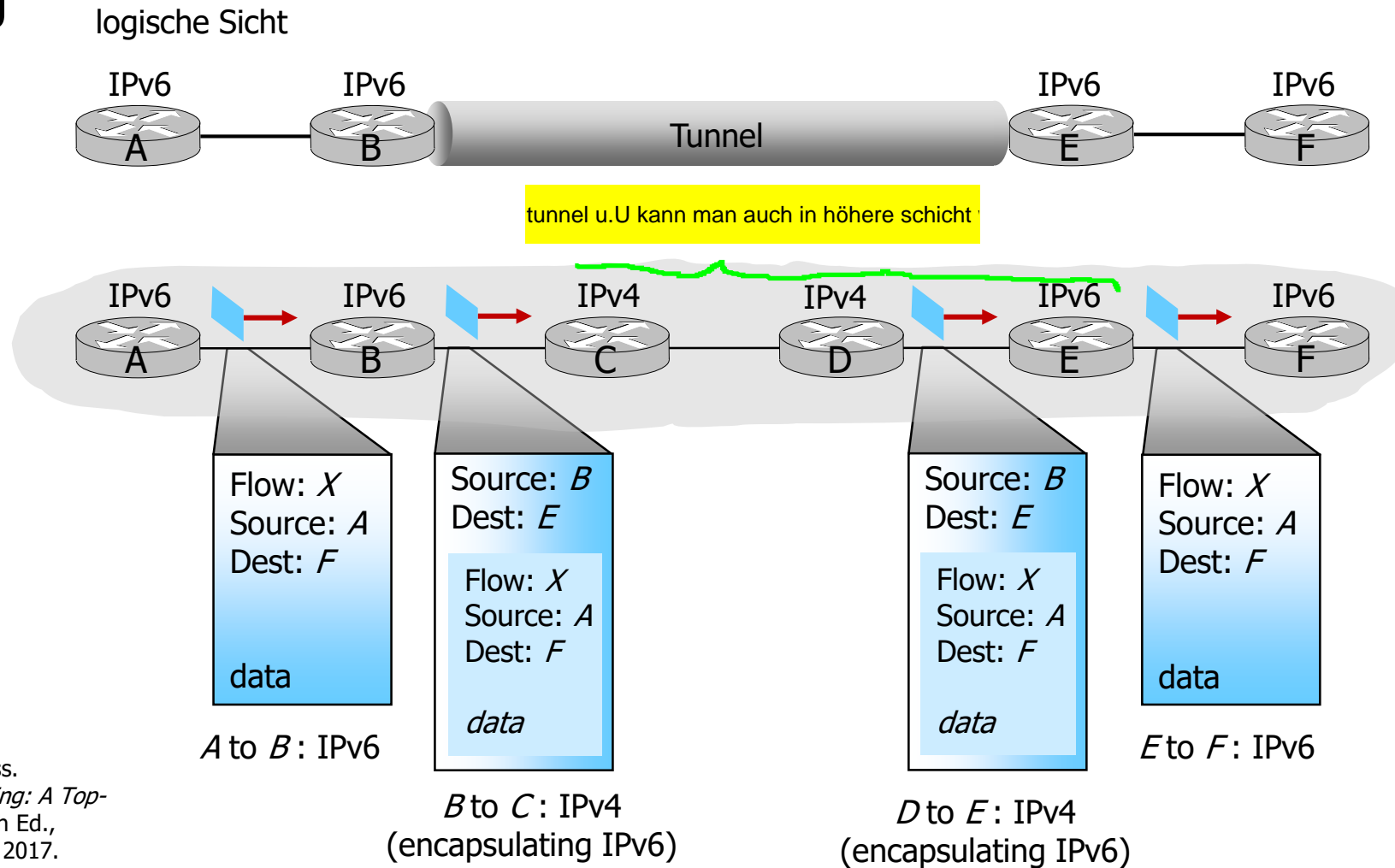
# IP: IPv6

## ■ Übergang von IPv4 zu IPv6

- gleichzeitige Umstellung unmöglich
- Ansätze für den Übergang
  - **Dual Stack**: Endsystem mit IPV6 und IPv4-Implementierung, abhängig vom Ziel (liefert DNS) wird IPv6 oder IPv4 verwendet, zusätzliche IPv6-Information wie z.B. Flow-Label geht dabei verloren, problematisch z.B. wenn Router nur mit IPv4 weiterleiten  der hat auch keinen Dualstack
  - **Tunneling**: das IPv6-Datagramm wird in ein IPv4-Datagramm eingebettet, diverse Tunnelmechanismen (z.B. 6to4, 6in4, 6over4, ISATAP), wrapping in ipv6
  - **NAT**: Datagramm mit privater IPv4-Adresse in IPv6-Paket einbetten und bei NAT-Übersetzung in ein IPv4-Paket mit öffentlicher Adresse wandeln, Einsatz z.B. bei Dual-Stack-Lite

# IP: IPv6

## ■ Tunneling



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

# Netzwerkschicht

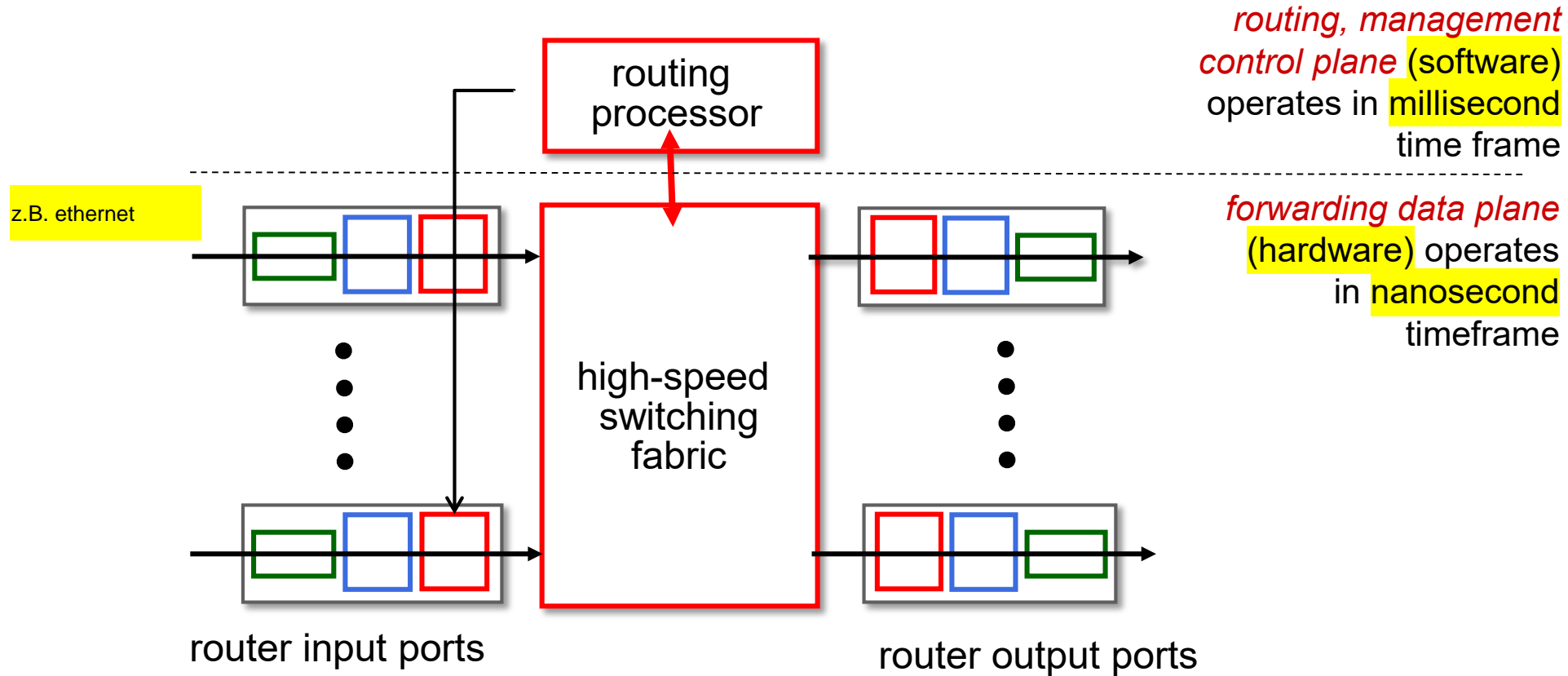
- ✓ Einführung
- ✓ IP
- Aufbau eines Routers
- Routing
- IPsec
- Netzvirtualisierung

# Aufbau eines Routers

Quelle: Kurose, Ross.  
*Computer Networking:  
A Top-Down Approach*,  
7th Ed., Pearson  
Education, 2020.

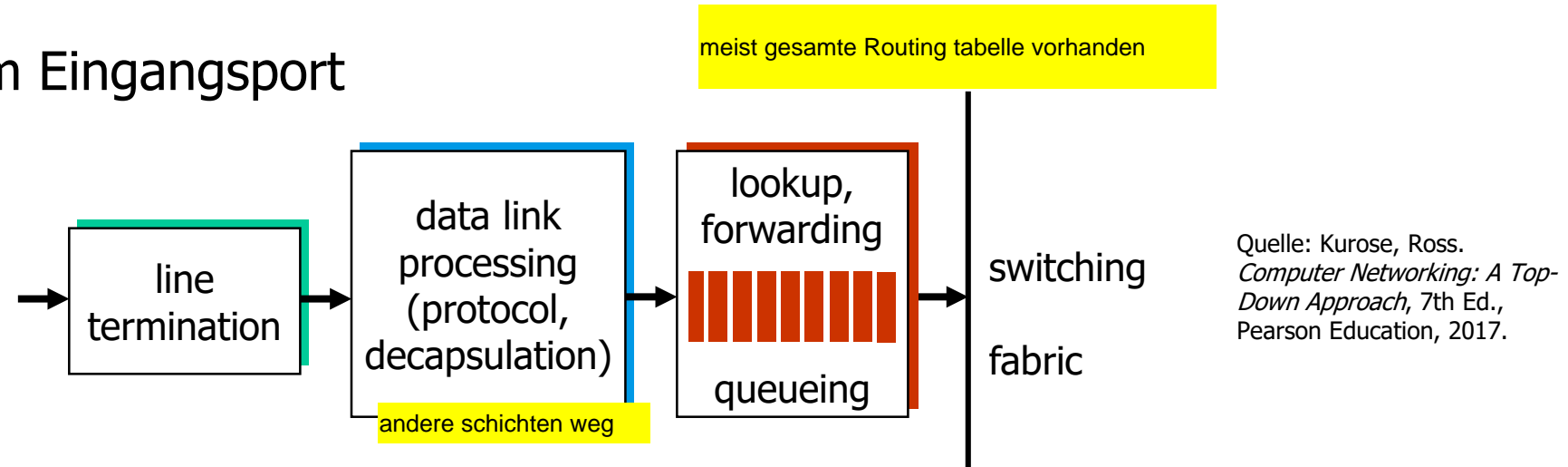
## ■ Aufgaben

- Weiterleitung, Ausführung von Routingprotokollen
- prinzipieller Aufbau:



# Aufbau eines Routers

## ■ Funktionen am Eingangsport



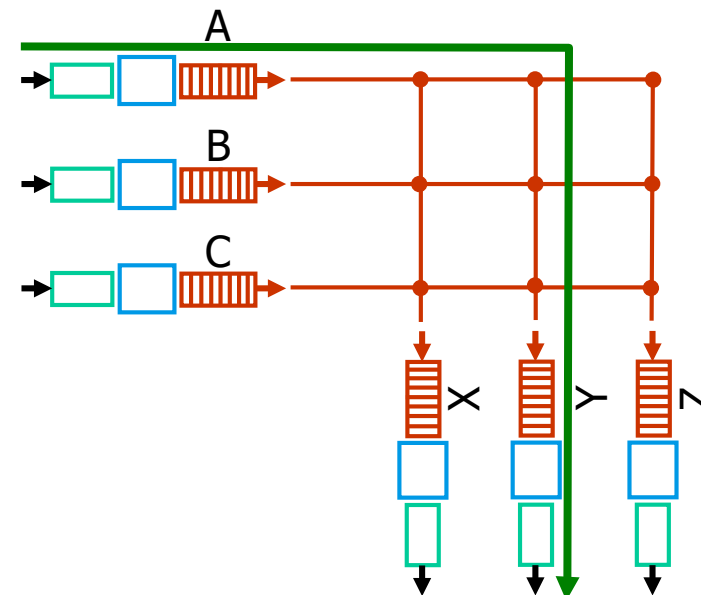
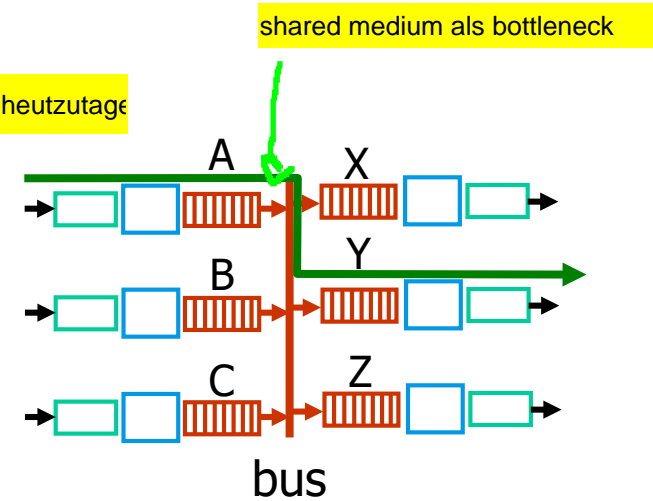
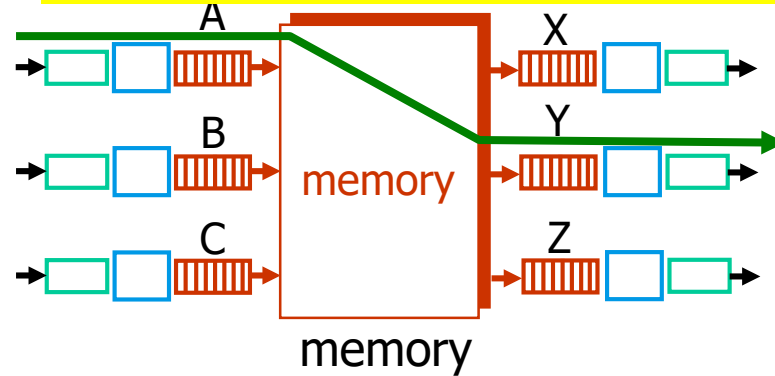
- **Pufferung**, falls Pakete schneller von der Leitung kommen, als sie weitergegeben werden können
- Paketverlust, wenn Puffer überläuft
- verteiltes Weiterleiten: Port besitzt Kopie der Weiterleitungstabelle
- effiziente Datenstrukturen für schnelle Suche
- Empfang eines Pakets mit 64 Byte bei 10 Gbps dauert 51,2 ns, in dieser Zeit routen
- inhaltsadressierbarer Speicher (Content Addressable Memory, CAM) konstante Zeit



# Aufbau eines Routers

## ■ Möglichkeiten für die Switching Fabric

speicher kopieren, ziel rauskopieren (früher üblich, performancegrößen) heutzutage



über mesh, aber schaltungsaufwand wächst quadratisch in aus/eingangs

interconnection network,  
e.g., crossbar

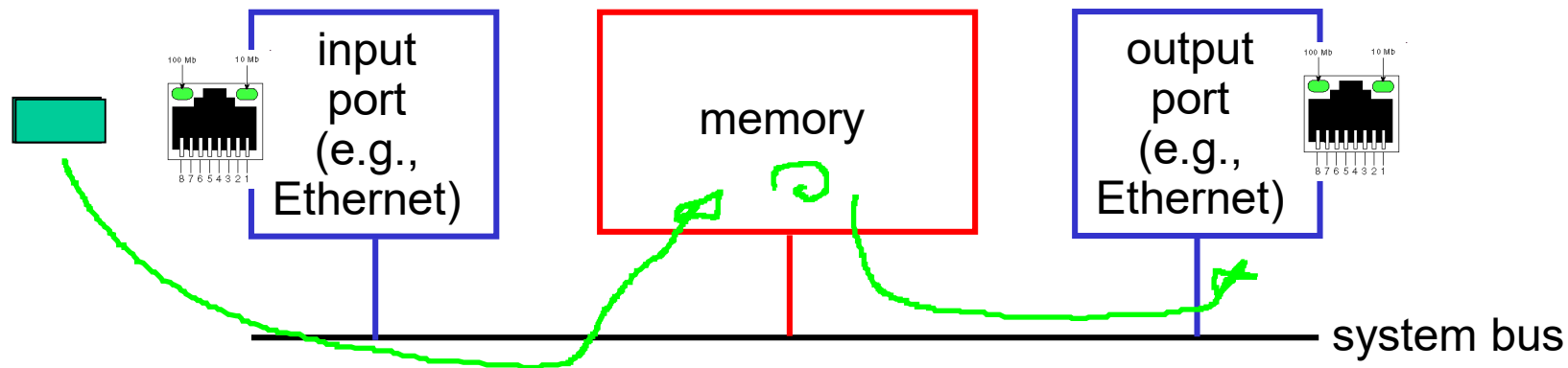
Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

# Aufbau eines Routers

## ■ Möglichkeiten für die Switching Fabric

### ● Speicher

- frühe Router waren einfache Rechner: CPU kopiert Paket von Eingangsport in Hauptspeicher, führt Weiterleitungsentscheidung durch und kopiert Paket zum Ausgangsport, heute auch möglich
- 2 mal internen Bus benutzen, Begrenzung der Leistungsfähigkeit
- in modernen Routern Direct Memory Access durch Eingangsports



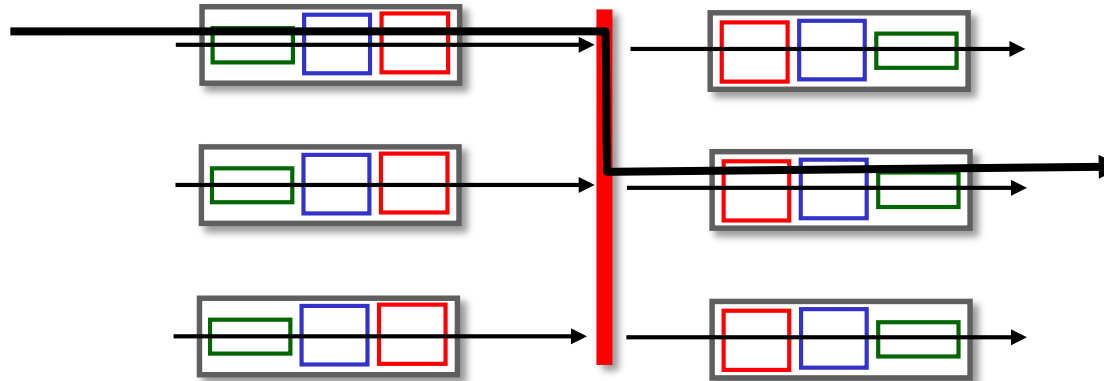
Quelle: Kurose, Ross.  
*Computer Networking:  
A Top-Down Approach*,  
7th Ed., Pearson  
Education, 2020.

# Aufbau eines Routers

## ■ Möglichkeiten für die Switching Fabric

### ● Bus

- ein Bus verbindet alle Ports, Eingangsport versieht Paket mit Markierung und sendet sie über den Bus per Broadcast an alle Ausgangsports, kann nur jeweils für einen Transfer benutzt werden, Wettbewerb
- üblich für kleine bis mittlere Router, z.B. 32 Gbps Bus



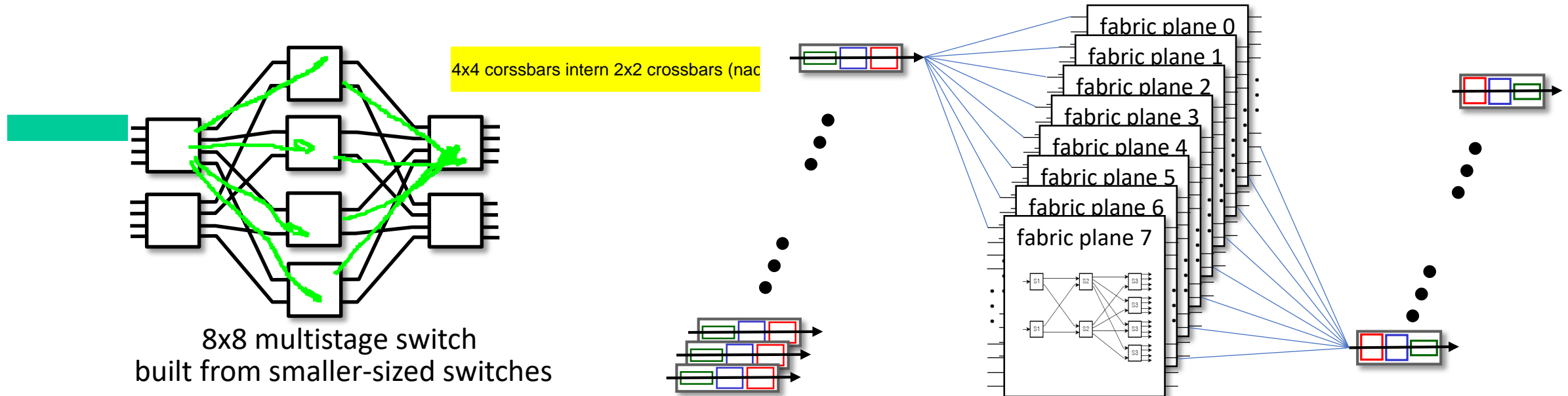
Quelle: Kurose, Ross.  
*Computer Networking:  
A Top-Down Approach*,  
7th Ed., Pearson  
Education, 2020.

# Aufbau eines Routers

Quelle: Kurose, Ross.  
*Computer Networking:  
A Top-Down Approach*,  
7th Ed., Pearson  
Education, 2020.

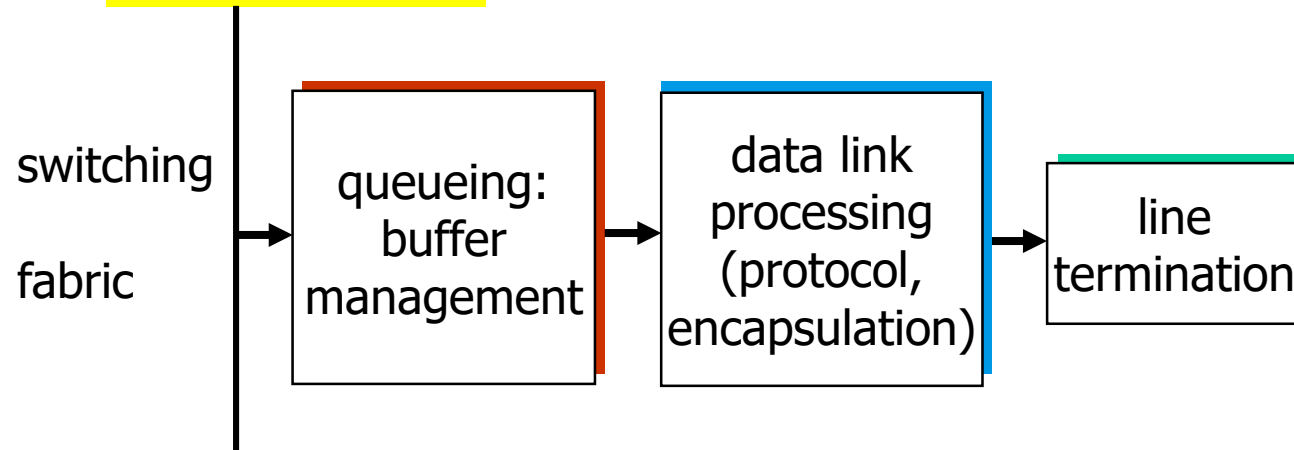
## ● Verbindungsnetzwerk

- bekannt aus der Verbindung von Prozessoren in Parallelrechnern
- z.B. **Crossbar**: jeder Port kann direkt mit jedem anderen verbunden werden (quadratischer Schaltungsaufwand)
- auch mehrstufige Anordnungen, z.B. **Banyan-Netze**
- ermöglicht nebenläufige Weiterleitung
- und mit parallelen Planes zu mehreren 100 Tbps



# Aufbau eines Routers

## ■ Funktionen des Ausgangsports



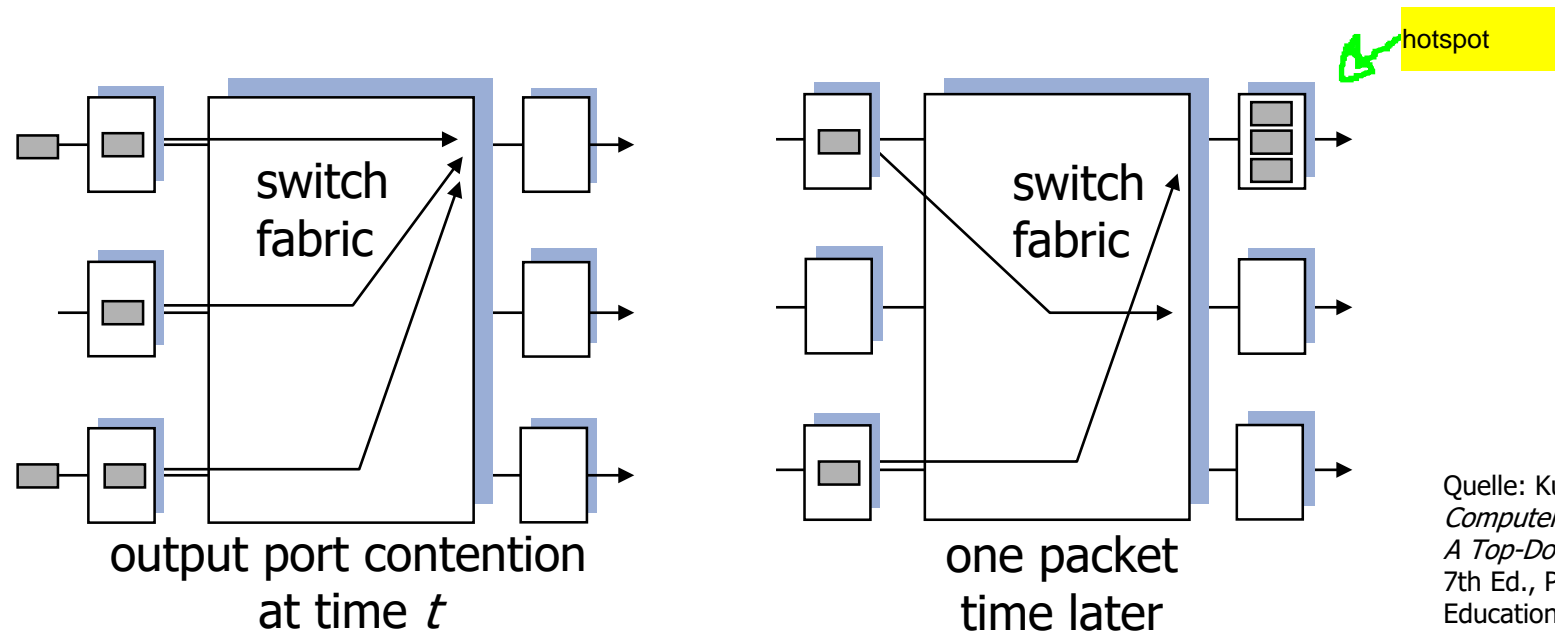
Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

- **Pufferung**, wenn Switching Fabric schneller liefert als Pakete auf die Leitung gegeben werden können
- **Paketverlust**, wenn Puffer überläuft, z.B. ankommendes Paket (Tail-Drop)
- **Active Queue Management**: proaktive Entscheidung, wann Pakete verworfen werden, z.B. mit Random Early Detect (RED)
- **Scheduling**: wenn mehrere Pakete gepuffert sind, kann entschieden werden, welches als nächstes gesendet wird, z.B. FIFO oder Weighted Fair Queuing (WFQ), Deficit Round Robin (DRR), ermöglicht Dienstgüte

# Aufbau eines Routers

## ■ Pufferungs- und Verlusteffekte

- wenn Switching Fabric schneller als Anzahl Ports x Leitungsgeschwindigkeit, normalerweise keine Pufferung bei Eingangsports notwendig
- wenn mehrere gleichzeitig zu einem Ausgangsport schicken, ist dort Pufferung notwendig:



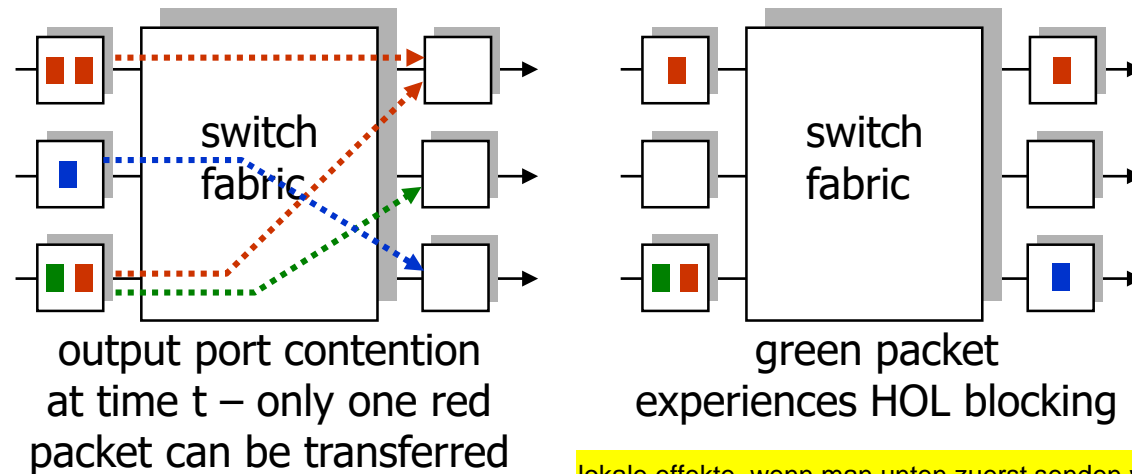
Quelle: Kurose, Ross.  
*Computer Networking:  
A Top-Down Approach*,  
7th Ed., Pearson  
Education, 2017.

# Aufbau eines Routers

## ■ Pufferungs- und Verlusteffekte

### ● Head-of-the-Line (HOL) Blocking

- falls Switching Fabric nicht schneller als Anzahl Ports x Leitungsgeschwindigkeit
- mehrere Eingangsports wollen auf gleichen Ausgangsport
- manche müssen warten, die dahinter werden blockiert, obwohl ihr Ausgangsport frei wäre



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

lokale effekte, wenn man unten zuerst senden würde, gäbe es keine congestion

- noch weitere Effekte, Auslegung durch Simulation und Analyse

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers

## ■ Routing

- Graphen
- Link-State-Routing
- Distanzvektor-Routing
- Interdomain-Routing

intra-network

## ■ IPsec

## ■ Netzvirtualisierung



# Routing

- **Routing:** Verfahren, mit denen Router entscheiden, über welchen Weg Pakete gesendet werden sollen
  - **Intradomain:** innerhalb einer **Routing-Domäne** (= unter einer administrativen Instanz), hier können Verfahren verwendet werden, die nicht für sehr große Netze skalieren, zwei Varianten
    - **Link-State:** jeder Router besitzt vollständige Information über die gesamte Routing-Domäne, jeder Router berechnet kürzeste Pfade von ihm zu allen anderen Routern mit dem Dijkstra-Verfahren, Beispiele: OSPF, IS-IS
    - **Distanzvektor:** jeder Router kennt nur Kosten zu direkten Nachbarn und die von ihm erreichbaren Ziele, die kürzesten Pfade werden verteilt mit dem Bellman-Ford-Verfahren berechnet, Beispiel: RIP, EIGRP
  - **Interdomain:** zwischen Routing-Domänen
    - ausgetauschte Routing-Informationen enthalten ganze Pfade, Auswahl durch Regeln, Beispiel: BGP

# Routing

## ■ Unicast-Routing (Punkt-zu-Punkt)

- **proaktiv**: Information über Netztopologie wird ausgetauscht, aktuell gehalten, mit Graph-basierten Verfahren werden Pfade zu allen Zielen bestimmt, bei Sendewunsch werden diese genutzt
- auf diesen Fall beschränken wir uns hier

also vorher shortest path

## ■ Multicast-Routing (Punkt-zu-Mehrpunkt)

- Router und Links sollen effizient genutzt werden
- Erweiterung von Unicast-Routing
- ebenfalls proaktiv

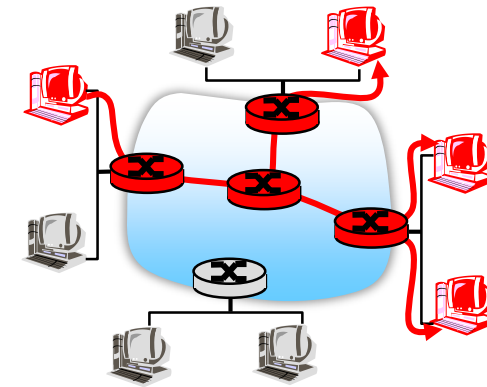
nicht teil von RK

## ■ Ad-Hoc-Routing

- dynamische Netztopologie: Pfade veralten schnell
- Erweiterung von proaktiven Verfahren
- auch **reaktive** Verfahren: erst bei Sendewunsch wird Pfad bestimmt

## ■ Datenzentrische Verfahren

- adresslos, Nachrichten werden aufgrund ihres Inhalts weitergeleitet, z.B. in Sensornetzen



Quelle: Kurose, Ross.  
*Computer Networking:  
A Top-Down Approach*,  
7th Ed., Pearson  
Education, 2017.

# Netzwerkschicht

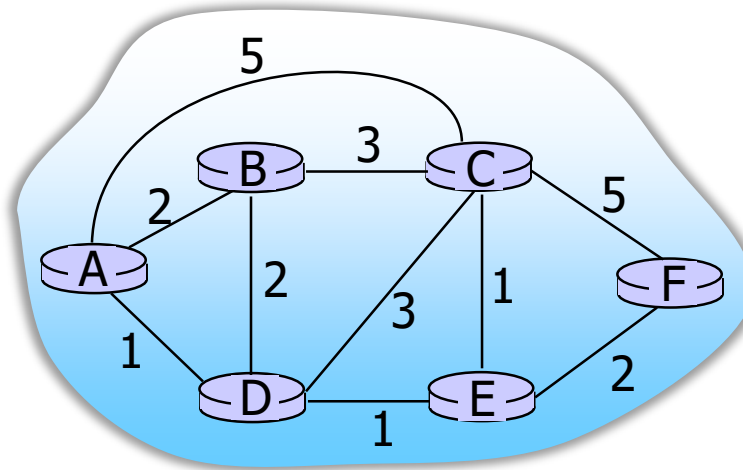
- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers
- ✓ Routing
  - Graphen
  - Link-State-Routing
  - Distanzvektor-Routing
  - Interdomain-Routing
- IPsec
- Netzvirtualisierung

## Routing: Graphen

- Graphen sind eine übliche Abstraktion für Netze
- Anwendung von Suchverfahren aus Graphentheorie
- einige Grundbegriffe:
  - **Graph**  $G = (N, E)$
  - **Knoten**  $N$  (nodes)
  - **Kanten**  $E \subseteq N \times N$  (edges),  
eine Kante ist ein Paar  $(v, w) \in E$ ,  $v$  und  $w$  heißen **Nachbarn**
  - ein Graph ist **ungerichtet** wenn alle Kanten symmetrisch sind:  
 $(v, w) \in E \Rightarrow (w, v) \in E$   
wir betrachten im Folgenden nur ungerichtete Graphen
  - **Kosten** sind eine Funktion  $c: E \rightarrow K$  auf eine geeignete Menge  $K$ , verkürzte Schreibweise  $c(v, w)$

## Routing: Graphen

- Beispiel für ungerichteten Graphen:



- $N = \{A, B, C, D, E, F\}$
- $E = \{(A,B), (A,C), (A,D), (B,A), (B,C), (B,D), (C,A), (C,B), (C,D), (C,E), (C,F), (D,A), (D,B), (D,C), (D,E), (E,C), (E,D), (E,F), (F,C), (F,E)\}$
- $c(A,B) = c(B,A) = 2, c(A,C) = c(C,A) = 5, \dots$

# Routing: Graphen

## ■ einige weitere Begriffe

also folgende Knoten

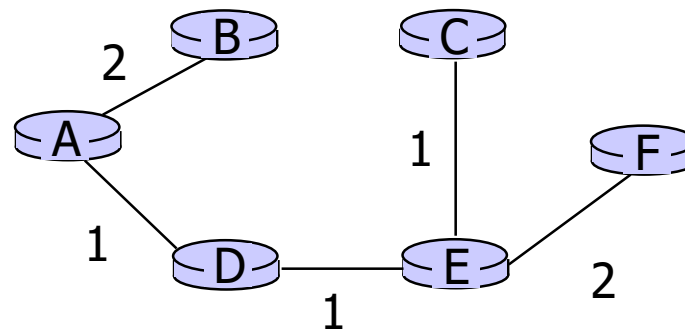
- ein **Pfad** ist eine Sequenz  $(v_1, v_2, \dots, v_n)$ , so dass alle Paare  $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n) \in E$
- die Kosten eines Pfads betragen  $c(v_1, v_2) + c(v_2, v_3) + \dots + c(v_{n-1}, v_n)$
- ein Pfad zwischen zwei Knoten  $v_1$  und  $v_n$  heißt **kürzester Pfad**, wenn es zwischen diesen Knoten keinen Pfad mit geringeren Kosten gibt (es kann mehrere kürzeste Pfade zwischen zwei Knoten geben), seine Kosten heißen Distanz  $D(v_1, v_n)$  zwischen  $v_1$  und  $v_n$
- ein **Zyklus** ist ein Pfad mit Anfangsknoten = Endknoten
- ein Graph ist **zusammenhängend**, wenn es einen Pfad zwischen jedem Knotenpaar gibt

# Routing: Graphen

## ■ und noch ein paar ...

- ein **Baum** ist ein zusammenhängender Graph, der **keine Zyklen enthält**
- ein **aufspannender Baum** eines Graphen  $G = (N, E)$  ist ein Baum  $B = (N, E')$  mit  $E' \subseteq E$  **alle kanten rausnehmen, sodass nur die für einen baum notwendigen vorhanden sind**
- ein **minimaler aufspannender Baum** eines Graphen  $G$  **für einen Startknoten  $v$**  ist ein aufspannender Baum von  $G$ , der für jedes Knotenpaar mit  $v$  als Startknoten einen **kürzesten Pfad aus  $G$**  enthält  
**(single-source shortest paths spanning tree, 1-SPST)**

Beispiel: minimaler aufspannender Baum für A:

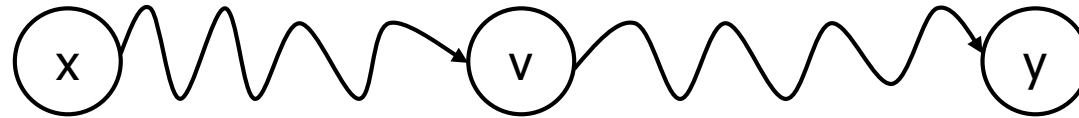


das ist genau das Routingproblem

## Routing: Graphen

### ■ Eigenschaft der kürzesten Pfade

- wenn  $v$  ein Teil des kürzesten Pfades von  $x$  nach  $y$  ist, dann setzt sich der kürzeste Pfad von  $x$  nach  $y$  aus den kürzesten Pfaden von  $x$  nach  $v$  und  $v$  nach  $y$  zusammen



- führt zu Rekursionsschema:  $D(x,y) = \min_v \{D(x,v) + D(v,y)\}$
- dies wird von den Verfahren zur Suche kürzester Pfade ausgenutzt



# Netzwerkschicht

- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers
- ✓ Routing
  - ✓ Graphen
  - Link-State-Routing
  - Distanzvektor-Routing
  - Interdomain-Routing
- IPsec
- Netzvirtualisierung

# Routing: Link-State-Routing

## ■ Link-State-Routing

- alle Knoten besitzen vollständige Kenntnis der Netztopologie
- dies wird durch **Fluten** erreicht
- jeder Knoten berechnet die kürzesten Pfade zu allen anderen Knoten
- hierfür wird das **Dijkstra-Verfahren** verwendet
- bei **Änderungen in der Netztopologie** (kann durch Austausch von Hello-Nachrichten zwischen benachbarten Routern erkannt werden), erfolgt **erneutes Fluten** und Neuberechnung der kürzesten Wege

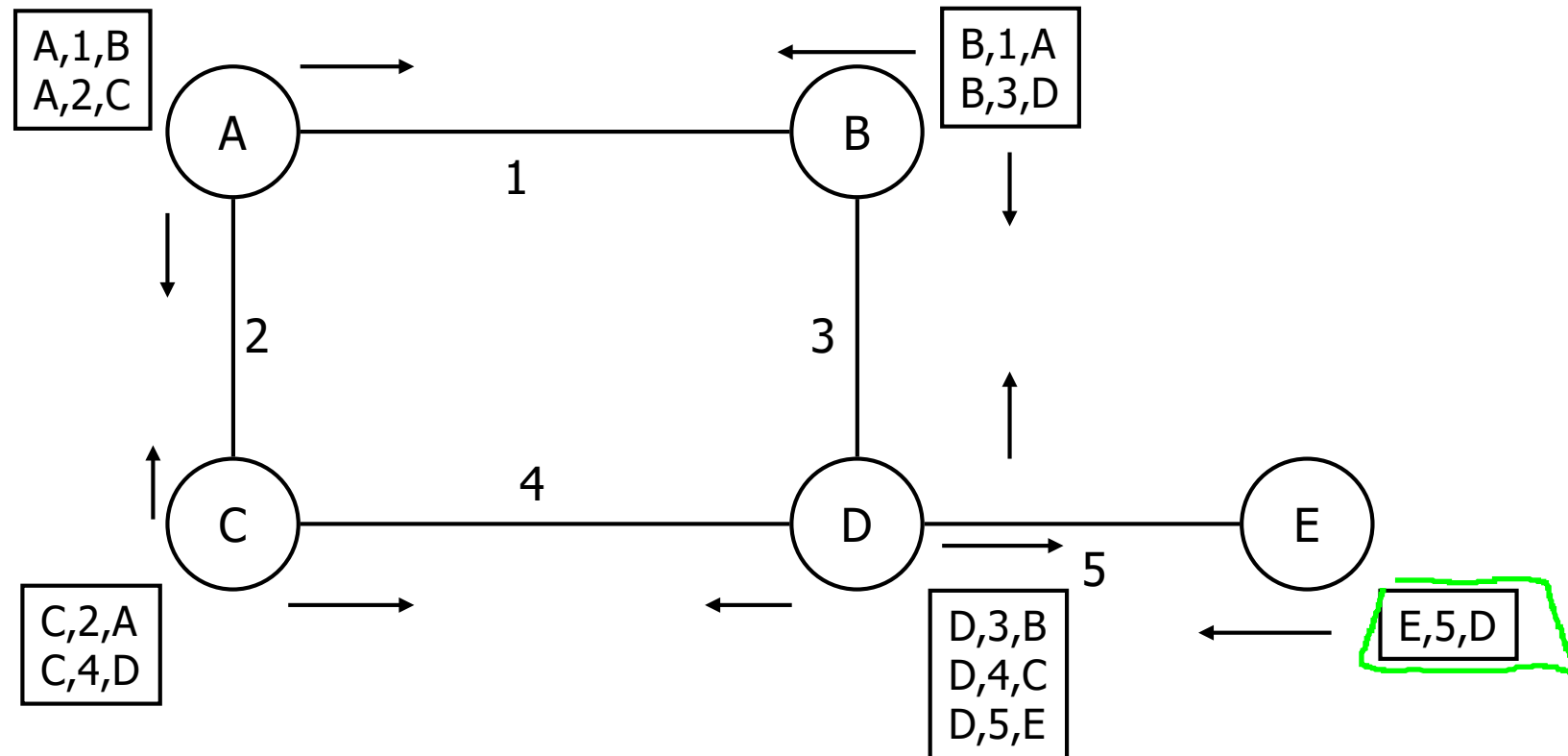
# Routing: Link-State-Routing

## ■ Fluten

- Link-State-Advertisements (LSAs) mit
  - Kennung des Knotens, der LSA erzeugt
  - Kosten zu Nachbarn und dessen Kennung
  - Sequenznummer
  - Lebensdauer
- jeder Knoten erzeugt LSAs mit den ihm bekannten Informationen über die Verbindungen zu den Nachbarn und sendet sie an alle Nachbarn
- neue von Nachbarn erhaltene LSAs werden an alle Nachbarn weitergeleitet, aber nicht an den Nachbarn, von dem das LSA kam
- zur Erzielung von Zuverlässigkeit auch Bestätigungen und Sendewiederholungen zwischen Nachbarn sowie Sequenznummern und Lebensdauer

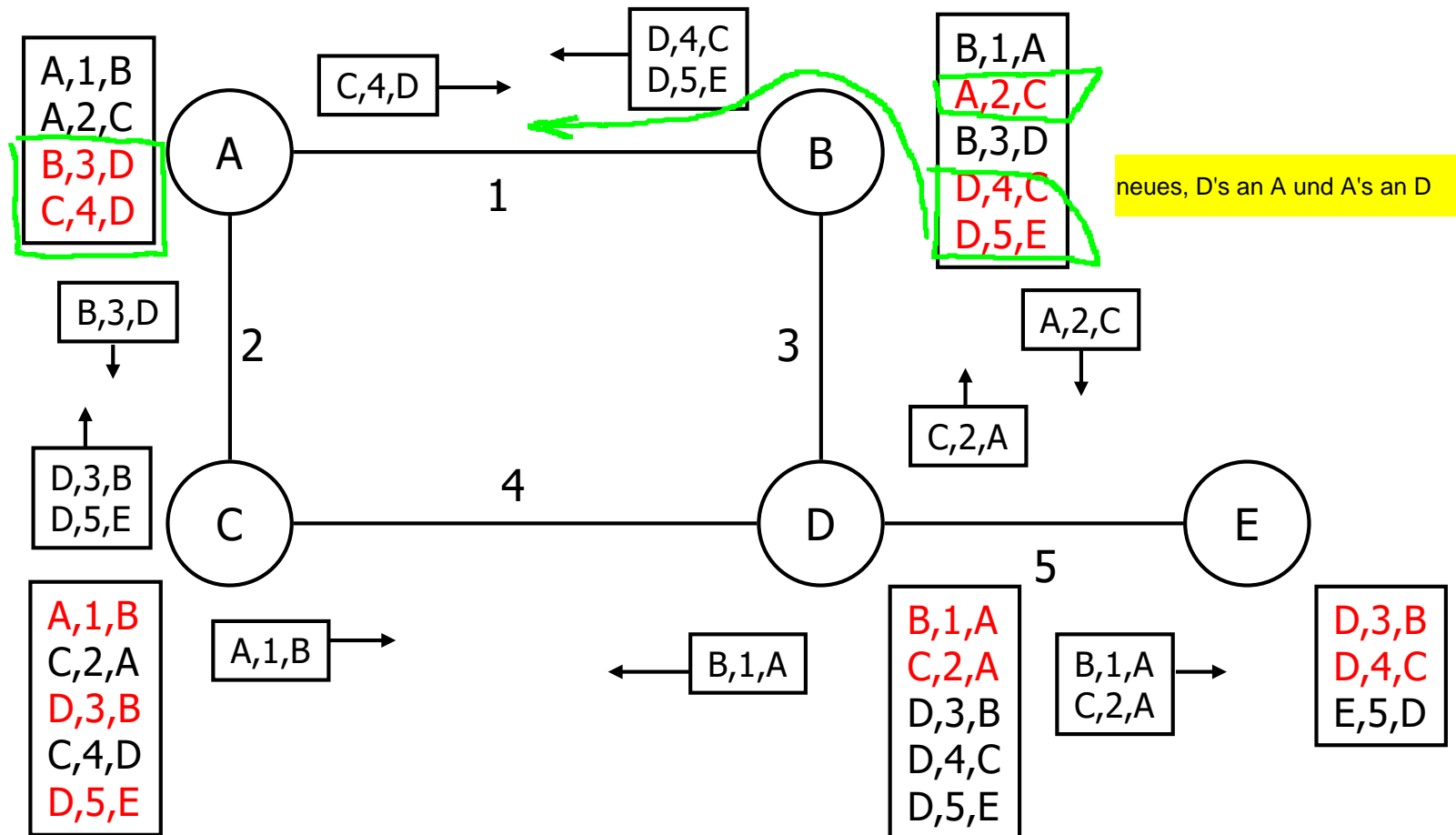
## Routing: Link-State-Routing

- Bsp. für Fluten: initial sind jedem Knoten die Kosten zu den Nachbarn bekannt, jeder schickt diese zu seinen Nachbarn:



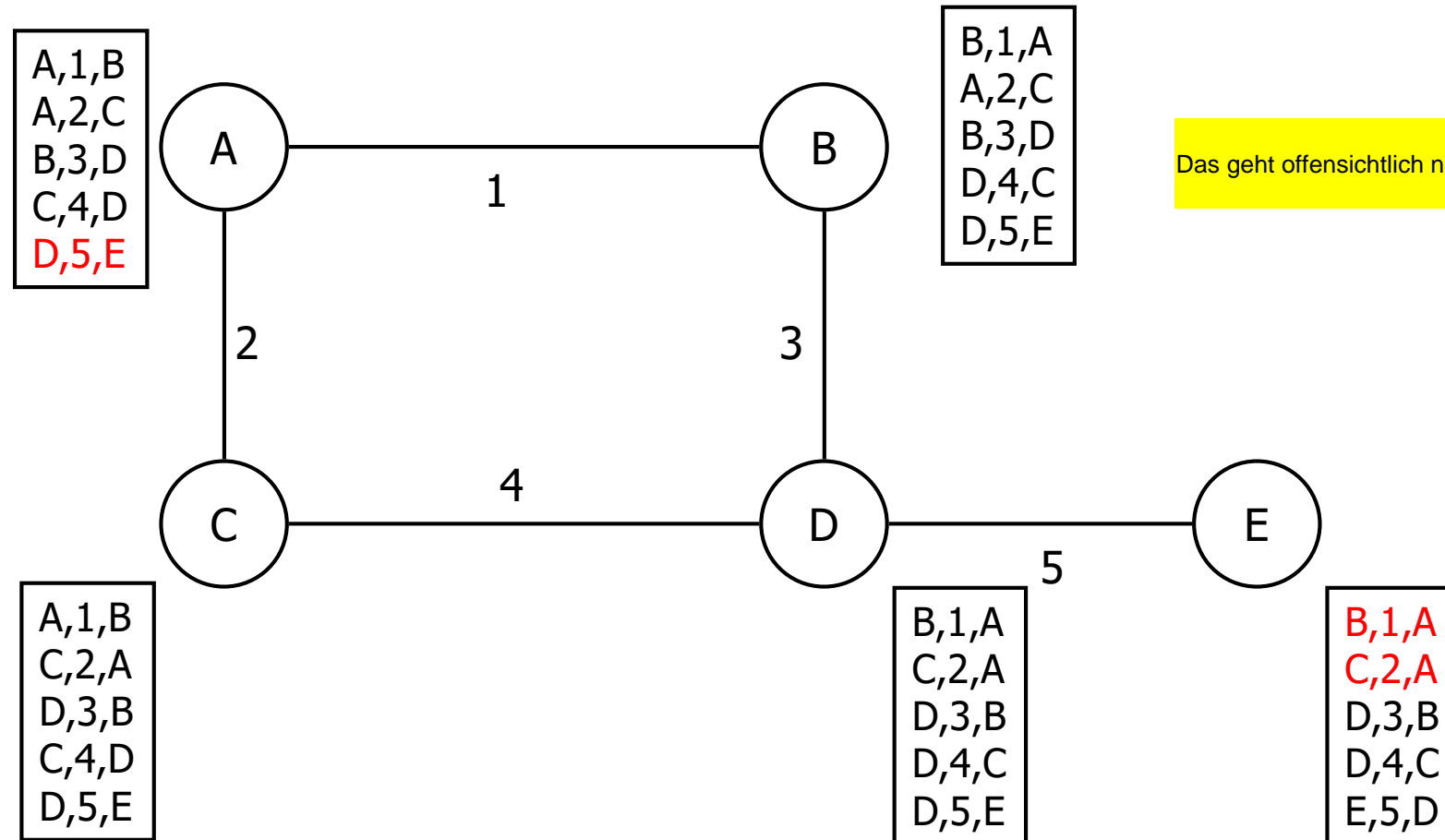
## Routing: Link-State-Routing

- Bsp. für Fluten: nach Empfang werden die Tabellen jedes Knoten aktualisiert und danach die neuen Informationen weitergereicht:



## Routing: Link-State-Routing

- Bsp. für Fluten: nach Empfang werden die Tabellen jedes Knoten nochmals aktualisiert und enthalten daraufhin die Information über den gesamten Graphen:



# Routing: Link-State-Routing

## ■ Idee des Dijkstra-Verfahrens

- der minimale aufspannende Baum wird iterativ aufgebaut
- eine Knotenmenge  $N'$  enthält immer die Knoten, für die die kürzesten Pfade bereits bekannt sind
- $N'$  wird mit dem **Startknoten  $u$**  initialisiert
- es wird jeweils ein Nachbarknoten  $v$  zugefügt, der über alle aktuellen Knoten  $w$  von  $N'$  und die jeweilige Kante zu  $v$  den kürzesten Pfad besitzt:  $D(u,v) = \min_{w \in N'} \{D(u,w) + c(w,v)\}$
- dies ist eine Spezialisierung des allgemeinen Rekursionsschemas  $D(u,v) = \min_w \{D(u,w) + D(w,v)\}$ , bei dem für den 2. Teil statt eines Pfads eine Kante verwendet wird
- der Algorithmus endet, wenn  $N = N'$

Iterativ den kürzesten Knoten und deren Nachfolger wählen: Zuerst R

## Routing: Link-State-Routing

### ■ Datenstrukturen für das Dijkstra-Verfahren

- Knotenmenge  $N$ , Startknoten  $u$
- $N'$  ist die Menge von Knoten, zu denen kürzeste Pfade von  $u$  aus bereits bekannt sind
- Kosten  $c(v,w)$  zwischen zwei Knoten  $v$  und  $w$ 
  - positive Verbindungskosten, wenn  $v$  und  $w$  Nachbarn sind
  - 0 für  $v = w$
  - $\infty$  sonst
- Distanz  $D(v)$  gibt Kosten des aktuell bekannten kürzesten Pfads von  $u$  nach  $v$
- $p(v)$  gibt Vorgänger von  $v$  auf dem aktuell bekannten kürzesten Pfad von  $u$  nach  $v$



# Routing: Link-State-Routing

## ■ Dijkstra-Verfahren

- Initialisierung

$N' = \{u\};$

für alle  $v \in N$ :  $D(v) = c(u, v);$

- Wiederhole bis  $N' = N$

finde  $w \in N \setminus N'$ , so dass  $\forall v \in N \setminus N': D(w) \leq D(v);$

/\*  $w$  ist ein Nachbar von Knoten aus  $N'$  mit minimalen Kosten \*/

$N' = N' \cup \{w\};$

für alle  $v \in N \setminus N'$

falls  $D(w) + c(w, v) < D(v)$ , dann

$D(v) = D(w) + c(w, v)$

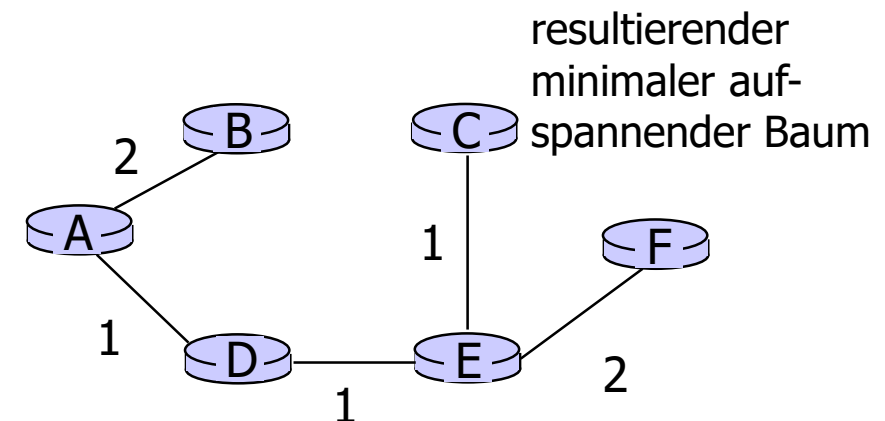
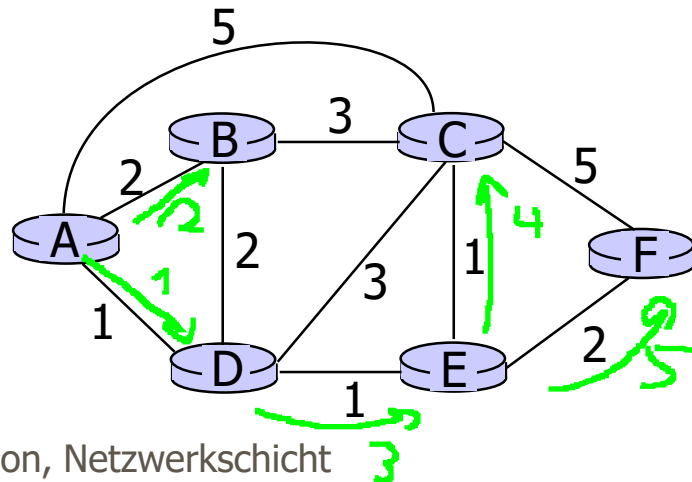
$p(v) = w$

/\* wenn ein Knoten über  $w$  günstiger zu erreichen ist,  
werden seine Kosten und sein Vorgänger angepasst \*/

# Beispiel für den Ablauf des Dijkstra-Verfahrens:

		jeweils D( $\cdot$ ), p( $\cdot$ )				
Schritt	N'	B	C	D	E	F
0	A	2,A	5,A	1,A	$\infty$ , -	$\infty$ , -
1	A,D	2,A	4,D	"	2,D	$\infty$ , -
2	A,D,B	"	4,D	"	2,D	$\infty$ , -
3	A,D,B,E	"	3,E	"	"	4,E
4	A,D,B,E,C	"	"	"	"	4,E
5	A,D,B,E,C,F	"	"	"	"	"

(Einträge mit " können sich nicht mehr verändern, da Knoten in N')



# Routing: Link-State-Routing

## ■ Ermittlung der Routing-Tabelle

- Routing-Tabelle enthält für jedes Ziel  $v$  den nächsten Hop auf dem kürzesten Weg
- wie kann er aus dem Vektor  $p(v)$  mit Vorgänger-Knoten ermittelt werden?
- dies liefert die rekursive Funktion  
$$\text{nexthop}(v) = \text{if } p(v)=u \text{ then } v \text{ else } \text{nexthop}(p(v))$$

man geht rekursiv vom Ziel die Vorgä

## Routing: Link-State-Routing

### ■ praktische Umsetzung des Dijkstra-Verfahrens

- in der Praxis sammelt jeder Knoten die LSAs und berechnet daraus direkt die Routing-Tabelle
- hierfür wird die als **Forward-Search-Algorithmus** bekannte Variante benutzt
- es werden Einträge der **Form (Ziel, Kosten, nächster Hop)** in **2 Listen verwaltet**
  - bestätigteListe (entspricht  $N'$ )  
man berechnet den nextHop direkt bei Dijkstra mit
  - vorläufigeListe (entspricht den Nachbarn von Knoten aus  $N'$ )
- $\text{nexthop}(v)$  ist der nächste Hop, um einen Knoten  $v$  vom Startknoten  $u$  aus zu erreichen
- die Werte für  $c(w,v)$  werden aus den LSAs gelesen, die Werte für  $D(w)$  und  $\text{nexthop}(w)$  werden aus den Einträgen in den 2 Listen gelesen

# Routing: Link-State-Routing

## ■ Forward-Search-Algorithmus

- Initialisierung

bestätigteListe =  $\langle (u, 0, -) \rangle$ , vorläufigeListe =  $\langle \rangle$ ;

kein prevNode

- Wiederhole

w = letzter in bestätigteListe eingetragener Knoten

für alle Nachbarn v von w

falls v weder in bestätigteListe noch in vorläufigeListe

füge  $(v, D(w)+c(w,v), \text{nexthop}(w))$  vorläufigeListe zu

D's sind distanzen vom Ursprung aus

falls v in vorläufigeListe und  $D(w)+c(w,v) < D(v)$

ersetze in vorläufigeListe den Eintrag für v durch

falls v besser geworden ist, update

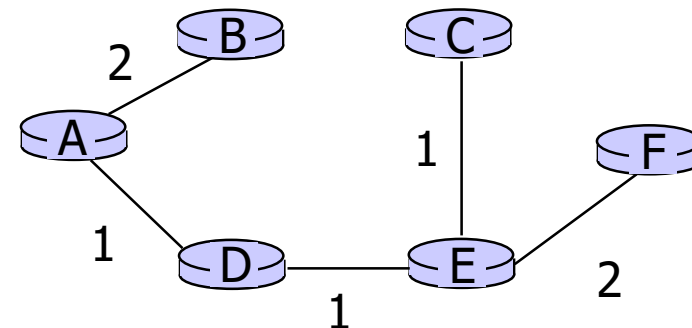
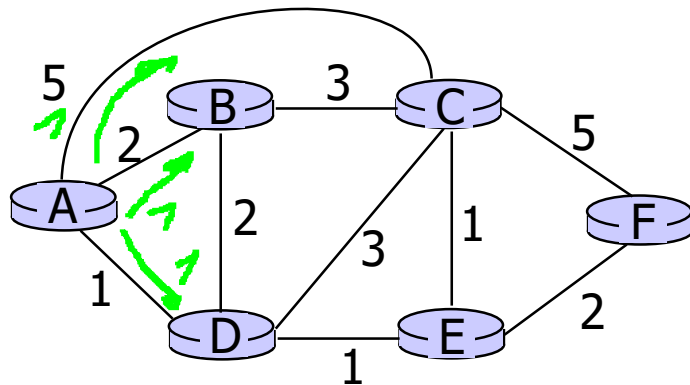
$(v, D(w)+c(w,v), \text{nexthop}(w))$

verschiebe Eintrag mit geringsten Kosten von vorläufigeListe in bestätigteListe

bis vorläufigeListe leer

# Routing: Link-State-Routing

Schritt	bestätigteListe	vorläufigeListe
0	(A,0,-)	Wenn - als next hop vom Ursprung ist, dann setzt man den next hop auf self
1	(A,0,-)	(B,2,B), (C,5,C), (D,1,D)
2	(A,0,-), (D,1,D)	(B,2,B), (C,4,D), (E,2,D)
3	(A,0,-), (D,1,D), (B,2,B)	(C,4,D), (E,2,D) B auf verbesserung untersuchen
4	(A,0,-), (D,1,D), (B,2,B), (E,2,D)	(C,3,D), (F,4,D) E auf verb...
5	(A,0,-), (D,1,D), (B,2,B), (E,2,D), (C,3,D)	(F,4,D) Das ist eine Verbesserung, der Next hop är
6	(A,0,-), (D,1,D), (B,2,B), (E,2,D), (C,3,D), (F,4,D)	



## Routing: Link-State-Routing

### ■ OSPF (Open Shortest Path First)

- verbreitetes Intradomain-Routing-Protokoll, OSPFv2 in RFC 2328, OSPFv3 für IPv6 in RFC 5340
- **Neighbor Discovery:** Router erkundet seine Nachbarn durch Austausch von Hello-Nachrichten (z.B. alle 10 s), Point-to-Point bzw. Broadcast bei Anschluss an Multi-Access-Netzwerk (z.B. Ethernet), Kennzeichnung von Router durch eindeutige ID (z.B. Loopback-IP-Adresse)
- **Synchronizing Database State:** Austausch von Database-Description- und Link-State-Request-Nachrichten zwischen Nachbarn
- **Advertizing Link State:** Versenden von Link-State-Update-Nachrichten mit LSAs an Nachbarn (periodisches Fluten, z.B. alle 30 min.)
- **Designated Routers:** bei Anschluss an Multi-Access-Netzwerk Auswahl eines Routers basierend auf Priorität und ID zur Verringerung der Kommunikationsbeziehungen
- kürzeste Wege mit Dijkstra-Verfahren

# Routing: Link-State-Routing

## ■ weitere Mechanismen von OSPF

- Authentifizierung (in OSPFv3 entfernt, auf IPv6 verlagert)
- **OSPF Areas:** 2 Hierarchieebenen für große Domänen
  - Area Zero ist Backbone, Area 1 bis N darunter
  - in jeder Area eigenes Routing (Aufbau Link State, Dijkstra)
- **Type-of-Service (TOS)**
  - verschiedene Kostenmetriken sind möglich, Standard ist Hop-Count, weiter möglich ist z.B. max. Durchsatz, min. Verzögerung
  - werden manuell gesetzt, für jede Metrik wird minimaler aufspannender Baum berechnet, ermöglicht zentrale Ermittlung und Verteilung von Gewichten
  - wenn in IP-Paket im TOS-Feld dieser Wert gesetzt ist, wird der entsprechende Baum verwandt
  - kaum genutzt
- **Equal Cost Multiple Path (ECMP)**
  - mehrere kürzeste Wege gleicher Länge mit geändertem Dijkstra-Verfahren bestimmen, Verkehr über diese Wege verteilen



# Routing: Link-State-Routing

## ■ Repräsentation eines Netzwerks als Graph in OSPF

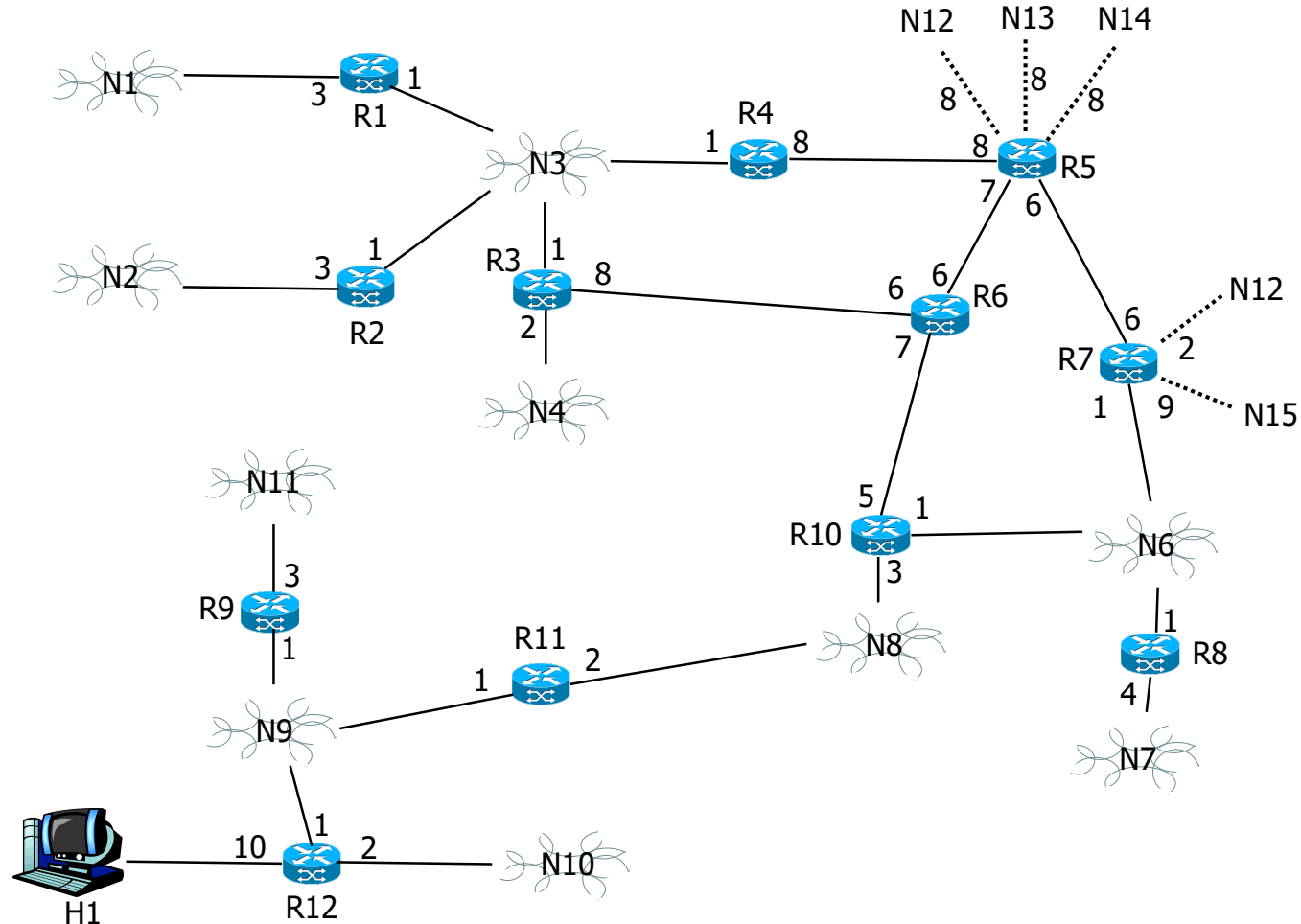
- Knoten
  - Router
  - Multi-Access-Netzwerk: Transit (mit Durchgangsverkehr), Stub (sonst)
- Kanten
  - direkte Verbindungen zwischen Routern (Punkt-zu-Punkt)
  - Verbindungen zwischen Routern und Netzwerken (Multi-Access)
- Kosten nur für Ausgangsports der Router, Kanten von Multi-Access-Netzwerken zu Routern besitzen keine Kosten (Kosten = 0)

## ■ Bsp.: Netzwerk mit

- Punkt-zu-Punkt-Verbindungen: z.B. zwischen R6 und R10
- Multi-Access-Verbindungen: z.B. R1, R2, R3, R4 an N3
- Stub-Netzwerk nur an einem Router: z.B. N7
- Host direkt an Router: H1
- Verbindungen zu anderen Netzwerken: N12 – N15

# Routing: Link-State-Routing

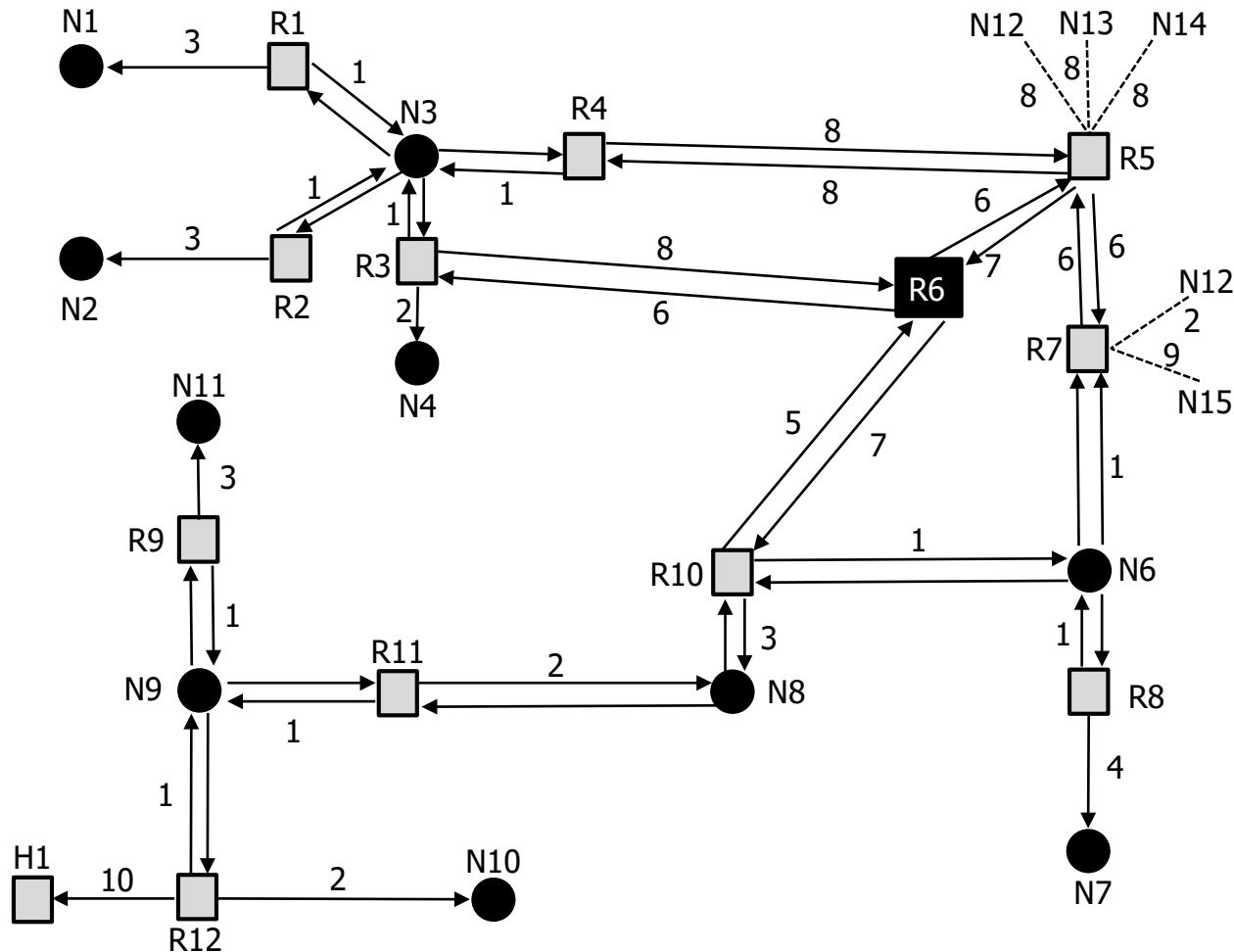
## ■ Bsp.: Netzwerk



Quelle: W. Stallings.  
*Data and Computer  
Communications*,  
10th Ed., Pearson  
Education, 2014.

# Routing: Link-State-Routing

## ■ Bsp.: Abbildung des Netzwerks auf gerichteten Graph

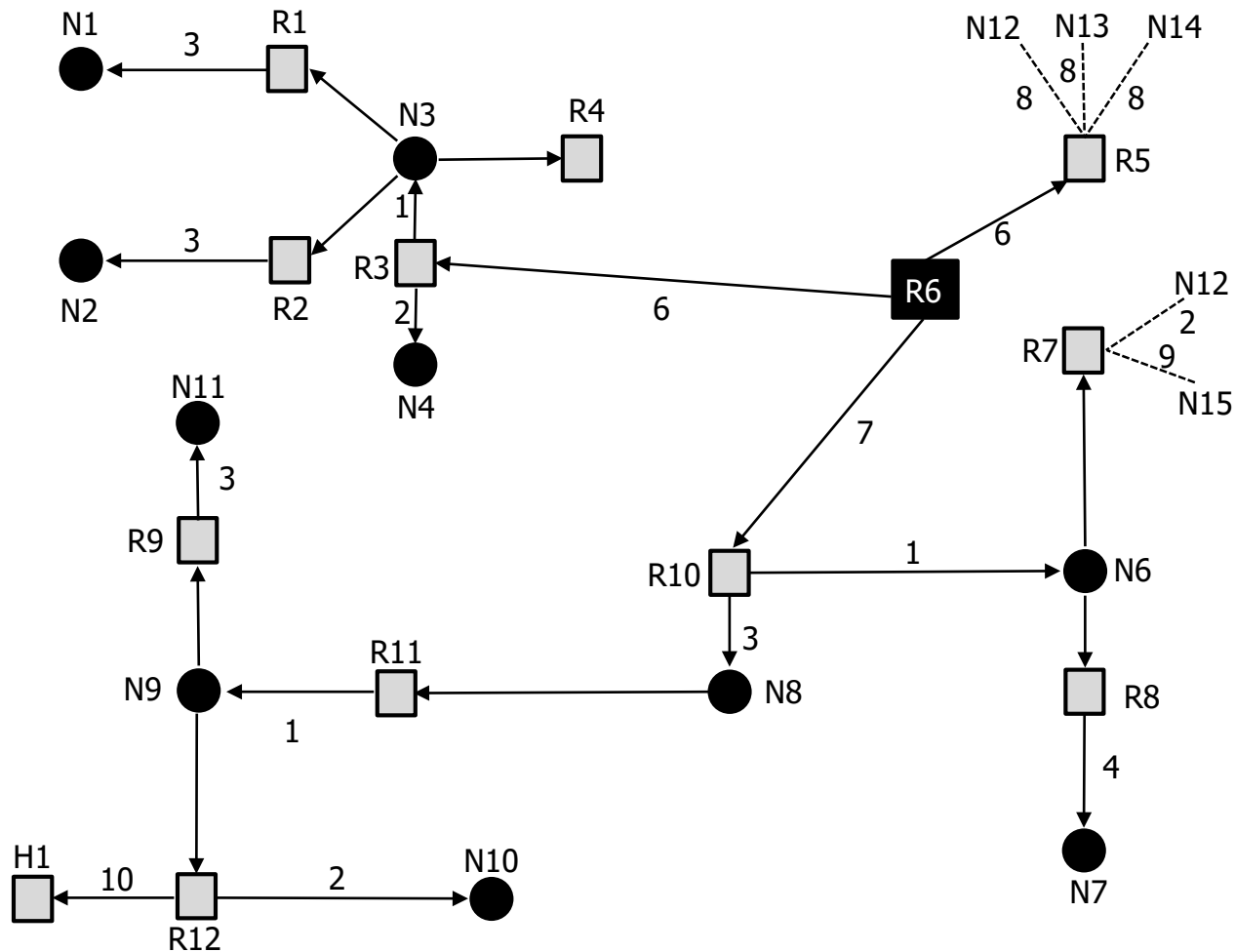


(Kanten ohne Kosten  
bedeuten Kosten = 0)

Quelle: W. Stallings.  
*Data and Computer  
Communications*,  
10th Ed., Pearson  
Education, 2014.

# Routing: Link-State-Routing

## ■ Bsp.: resultierender 1-SPST für R6



Quelle: W. Stallings.  
*Data and Computer  
Communications*,  
10th Ed., Pearson  
Education, 2014.

## Routing: Link-State-Routing

### ■ Bsp.: Routing-Tabelle für R6

Ziel	Next Hop	Distanz
N1	R3	10
N2	R3	10
N3	R3	7
N4	R3	8
N6	R10	8
N7	R10	12
N8	R10	10
N9	R10	11
N10	R10	13
N11	R10	14
H1	R10	21
R5	R5	6
R7	R10	8
N12	R10	10
N13	R5	14
N14	R5	14
N15	R10	17

Quelle: W. Stallings.  
*Data and Computer  
Communications*,  
10th Ed., Pearson  
Education, 2014.

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers
- ✓ Routing
  - ✓ Graphen
  - ✓ Link-State-Routing
  - Distanzvektor-Routing
  - Interdomain-Routing
- IPsec
- Netzvirtualisierung

## Routing: Distanzvektor-Routing

### ■ Idee des Bellman-Ford-Verfahrens

- die Suche nach kürzesten Wegen wird verteilt durch alle beteiligten Knoten durchgeführt
- jeder Knoten teilt seinen Nachbarn mit, mit welchen minimalen Kosten er andere Knoten erreichen kann
- anfangs können nur die Kosten zu den Nachbarn bekanntgegeben werden, mit jedem Austausch werden längere Pfade bekannt
- es wird wieder die Eigenschaft der kürzesten Pfade  $D(u,v) = \min_w \{D(u,w) + D(w,v)\}$  verwendet
- diesmal wird für den 1. Teil statt eines Pfads eine Kante verwendet:  
 $D(u,v) = \min_w \{c(u,w) + D(w,v)\}$
- wenn ein Knoten eine kürzere Entfernung ermittelt, sendet er diese Information an alle Nachbarn
- der Algorithmus endet, wenn sich keine Veränderung mehr ergibt (Konvergenz wurde erreicht)

## Routing: Distanzvektor-Routing

### ■ Datenstrukturen für das Bellman-Ford-Verfahren

- Kosten  $c(v,w)$  zwischen zwei Knoten  $v$  und  $w$ 
  - positive Verbindungskosten, wenn  $v$  und  $w$  Nachbarn sind
  - 0 für  $v = w$
  - $\infty$  sonst
- Distanz  $D_u(v)$ : bekannte kürzeste Entfernung von  $u$  nach  $v$
- Distanzvektor  $\mathbf{D}_u = (D_u(v), v \in N)$
- jeder Knoten  $u$  besitzt
  - Distanzen  $D_u(v)$  für alle Knoten  $v \in N$
  - lokale Kopien  $\underline{D}_w(v)$  der Distanzen  $D_w(v)$  von allen Nachbarn  $w$  zu allen Knoten  $v \in N$
  - nächster Hop  $\text{nexthop}_u(v)$ , um  $v$  von  $u$  aus zu erreichen für alle Knoten  $v \in N$



## Routing: Distanzvektor-Routing

### ■ Bellman-Ford-Verfahren (für jeden Knoten $u \in N$ )

- Initialisierung

für alle  $v \in N$ :

falls  $v$  Nachbar von  $u$ :  $D_u(v) = c(u, v)$ ;  $\text{nexthop}_u(v) = v$ ;

sonst  $D_u(v) = \infty$ ;  $\text{nexthop}_u(v) = -$ ;

für alle Nachbarn  $w$ :

für alle  $v \in N$ :  $\underline{D}_w(v) = \infty$ ;

sende  $\mathbf{D}_u$  an  $w$ ;

- Wiederhole

warte bis

Änderung der Kosten zu Nachbar  $w$  ( $\mathbf{D}_u$  ändert sich) oder

Erhalt eines Distanzvektors  $\mathbf{D}_w$  von Nachbar  $w$  ( $\underline{\mathbf{D}}_w$  ändert sich)

für alle  $v \in N$ :

$D_u(v) = \min_w \{c(u, w) + \underline{D}_w(v)\}$ ;

$\text{nexthop}_u(v) =$  der Knoten  $w$  aus dieser Minimumsbildung

wenn Änderung in  $\mathbf{D}_u$ : sende  $\mathbf{D}_u$  an alle direkten Nachbarn

Beispiel:  
nach  
Initialisierung

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	2	B
C	5	C
D	1	D
E	$\infty$	-
F	$\infty$	-

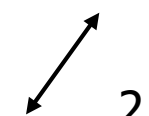
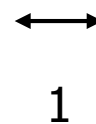
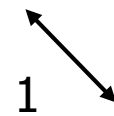
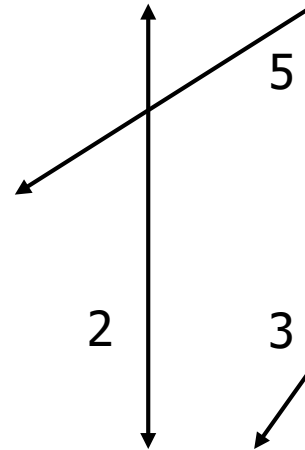
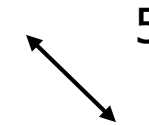
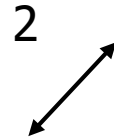
von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	2	A
B	0	-
C	3	C
D	2	D
E	$\infty$	-
F	$\infty$	-

von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	5	A
B	3	B
C	0	-
D	3	D
E	1	E
F	5	F

von <b>F</b> zu	$D_F(\cdot)$	$nh_F(\cdot)$
A	$\infty$	-
B	$\infty$	-
C	5	C
D	$\infty$	-
E	2	E
F	0	-

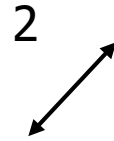
von <b>D</b> zu	$D_D(\cdot)$	$nh_D(\cdot)$
A	1	A
B	2	B
C	3	C
D	0	-
E	1	E
F	$\infty$	-

von <b>E</b> zu	$D_E(\cdot)$	$nh_E(\cdot)$
A	$\infty$	-
B	$\infty$	-
C	1	C
D	1	D
E	0	-
F	2	F



nach 1.  
Austausch

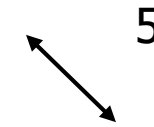
von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	2	B
C	4	D
D	1	D
E	2	D
F	10	C



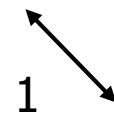
von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	2	A
B	0	-
C	3	C
D	2	D
E	3	D
F	8	C



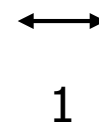
von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	4	D
B	3	B
C	0	-
D	2	E
E	1	E
F	3	E



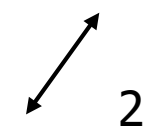
von <b>F</b> zu	$D_F(\cdot)$	$nh_F(\cdot)$
A	10	C
B	8	C
C	3	E
D	3	E
E	2	E
F	0	-



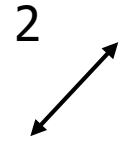
von <b>D</b> zu	$D_D(\cdot)$	$nh_D(\cdot)$
A	1	A
B	2	B
C	2	E
D	0	-
E	1	E
F	3	E



von <b>E</b> zu	$D_E(\cdot)$	$nh_E(\cdot)$
A	2	D
B	3	D
C	1	C
D	1	D
E	0	-
F	2	F



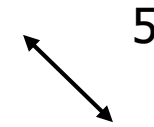
nach 2.  
Austausch



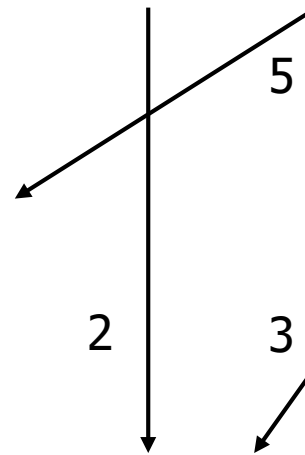
von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	2	A
B	0	-
C	3	C
D	2	D
E	3	D
F	5	D



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	3	E
B	3	B
C	0	-
D	2	E
E	1	E
F	3	E



von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	2	B
C	3	D
D	1	D
E	2	D
F	4	D



2

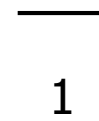
3

1

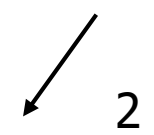
von <b>F</b> zu	$D_F(\cdot)$	$nh_F(\cdot)$
A	4	E
B	5	E
C	3	E
D	3	E
E	2	E
F	0	-



von <b>D</b> zu	$D_D(\cdot)$	$nh_D(\cdot)$
A	1	A
B	2	B
C	2	E
D	0	-
E	1	E
F	3	E



von <b>E</b> zu	$D_E(\cdot)$	$nh_E(\cdot)$
A	2	D
B	3	D
C	1	C
D	1	D
E	0	-
F	2	F



nach 3.  
Austausch  
Konvergenz

2

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	2	A
B	0	-
C	3	C
D	2	D
E	3	D
F	5	D

3

von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	3	E
B	3	B
C	0	-
D	2	E
E	1	E
F	3	E

5

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	2	B
C	3	D
D	1	D
E	2	D
F	4	D

2

5

3

1

von <b>F</b> zu	$D_F(\cdot)$	$nh_F(\cdot)$
A	4	E
B	5	E
C	3	E
D	3	E
E	2	E
F	0	-

1

von <b>D</b> zu	$D_D(\cdot)$	$nh_D(\cdot)$
A	1	A
B	2	B
C	2	E
D	0	-
E	1	E
F	3	E

1

von <b>E</b> zu	$D_E(\cdot)$	$nh_E(\cdot)$
A	2	D
B	3	D
C	1	C
D	1	D
E	0	-
F	2	F

2

## Routing: Distanzvektor-Routing

### ■ Verhalten bei Änderungen der Netztopologie

- der Algorithmus funktioniert auch bei Topologieänderungen und wenn die Informationen in asynchroner Weise ausgetauscht werden (also nicht jeweils gleichzeitig)
- bei Verkleinerung der Verbindungskosten konvergiert das Verfahren schnell: **good news travel fast**, siehe nächstes Beispiel
- bei Vergrößerung der Verbindungskosten können jedoch durch Zyklen in den Pfaden Probleme entstehen: **bad news travel slowly**, siehe übernächstes Beispiel

Verkleinerung,  
 $D_C(D)$  und  
 $D_D(C)$   
 werden  
 besser

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	2	A
B	0	-
C	3	C
D	2	D
E	3	D
F	5	D

von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	3	E
B	3	B
C	0	-
D	1	C
E	1	E
F	3	E

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	2	B
C	3	D
D	1	D
E	2	D
F	4	D

von <b>F</b> zu	$D_F(\cdot)$	$nh_F(\cdot)$
A	4	E
B	5	E
C	3	E
D	3	E
E	2	E
F	0	-

von <b>D</b> zu	$D_D(\cdot)$	$nh_D(\cdot)$
A	1	A
B	2	B
C	1	C
D	0	-
E	1	E
F	3	E

von <b>E</b> zu	$D_E(\cdot)$	$nh_E(\cdot)$
A	2	D
B	3	D
C	1	C
D	1	D
E	0	-
F	2	F

3

5

5

~~3~~ → 1

2

1

1

1

2

Verkleinerung,  
nach  
1. Austausch

2

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	2	A
B	0	-
C	3	C
D	2	D
E	3	D
F	5	D

3

von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	2	D
B	3	B
C	0	-
D	1	C
E	1	E
F	3	E

5

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	2	B
C	2	D
D	1	D
E	2	D
F	4	D

2

~~3~~ → 1

von <b>D</b> zu	$D_D(\cdot)$	$nh_D(\cdot)$
A	1	A
B	2	B
C	1	C
D	0	-
E	1	E
F	3	E

1

1

von <b>F</b> zu	$D_F(\cdot)$	$nh_F(\cdot)$
A	4	E
B	5	E
C	3	E
D	3	E
E	2	E
F	0	-

2

von <b>E</b> zu	$D_E(\cdot)$	$nh_E(\cdot)$
A	2	D
B	3	D
C	1	C
D	1	D
E	0	-
F	2	F



Verkleinerung,  
nach  
2. Austausch  
Konvergenz

2

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	2	A
B	0	-
C	3	C
D	2	D
E	3	D
F	5	D

3

von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	2	E
B	3	B
C	0	-
D	1	C
E	1	E
F	3	E

5

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	2	B
C	2	D
D	1	D
E	2	D
F	4	D

5

2

~~3~~ → 1

1

von <b>F</b> zu	$D_F(\cdot)$	$nh_F(\cdot)$
A	4	E
B	5	E
C	3	E
D	3	E
E	2	E
F	0	-

1

von <b>D</b> zu	$D_D(\cdot)$	$nh_D(\cdot)$
A	1	A
B	2	B
C	1	C
D	0	-
E	1	E
F	3	E

1

von <b>E</b> zu	$D_E(\cdot)$	$nh_E(\cdot)$
A	2	D
B	3	D
C	1	C
D	1	D
E	0	-
F	2	F

2

# Routing: Distanzvektor-Routing

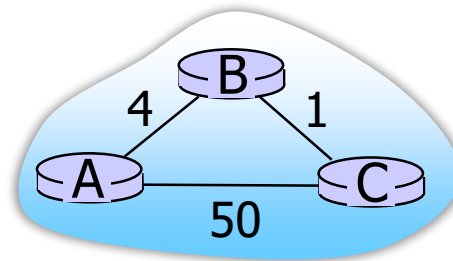
## ■ Beispiel für Vergrößerung

- Ausgangspunkt:

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	4	A
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	4	B
C	5	B



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	5	B
B	1	B
C	0	-

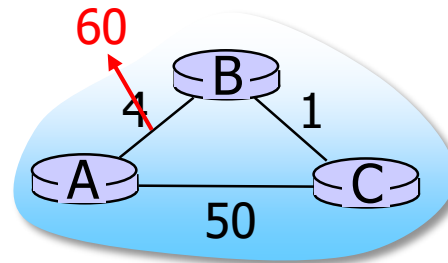
# Routing: Distanzvektor-Routing

- Beispiel für Vergrößerung
  - nach Vergrößerung der Kosten

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	6	C
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	5	B
B	1	B
C	0	-

# Routing: Distanzvektor-Routing

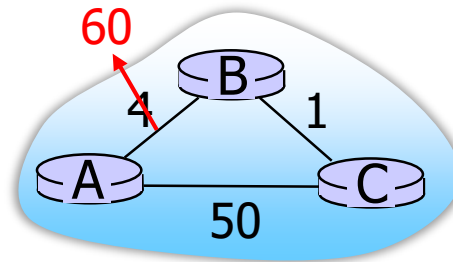
## ■ Beispiel für Vergrößerung

- nach 1. Austausch

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	6	C
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	7	B
B	1	B
C	0	-

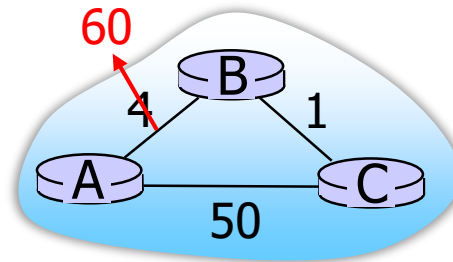
# Routing: Distanzvektor-Routing

- Beispiel für Vergrößerung
  - nach 2. Austausch

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	8	C
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	7	B
B	1	B
C	0	-

# Routing: Distanzvektor-Routing

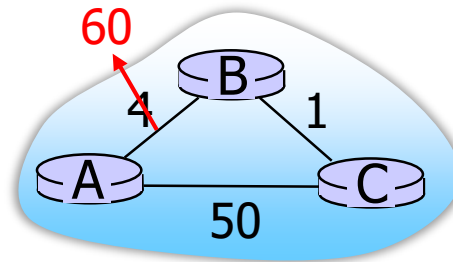
## ■ Beispiel für Vergrößerung

- nach 3. Austausch

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	8	C
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	9	B
B	1	B
C	0	-

- usw. ...

# Routing: Distanzvektor-Routing

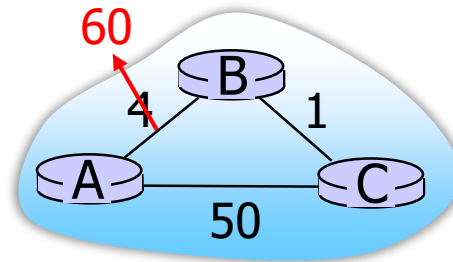
## ■ Beispiel für Vergrößerung

- ... nach 46. Austausch

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	50	C
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	49	B
B	1	B
C	0	-

# Routing: Distanzvektor-Routing

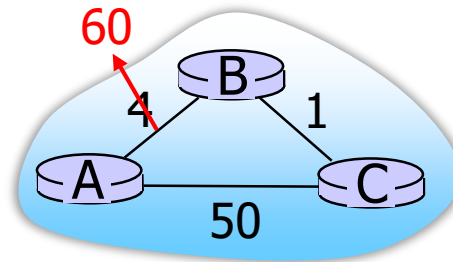
## ■ Beispiel für Vergrößerung

- nach 47. Austausch

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	50	C
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	50	A
B	1	B
C	0	-

- C erkennt, dass A direkt mit den geringsten Kosten erreichbar ist



# Routing: Distanzvektor-Routing

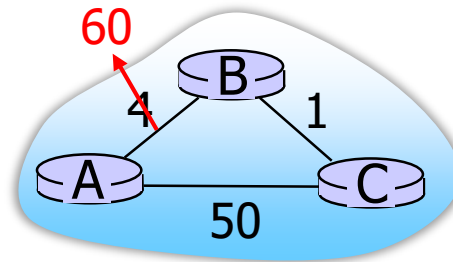
## ■ Beispiel für Vergrößerung

- nach 48. Austausch Konvergenz erreicht

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	51	C
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	50	A
B	1	B
C	0	-

# Routing: Distanzvektor-Routing

## ■ Count-to-infinity-Problem

- veraltete Information in den verteilten Routing-Tabellen enthält zyklischen Pfad
- die langsame Iteration endet erst, wenn die Kosten des alternativen Pfads erreicht sind
- Abhilfe
  - größten Kostenwert beschränken (z.B. 16)
  - **Poisoned Reverse**: wenn der kürzeste Weg von u nach v über den nächsten Hop w führt, sendet u an w die Kosten von unendlich für die Entfernung von u nach v
- mit Poisoned Reverse können Zyklen der Länge 2 vermieden werden, nicht jedoch längere Zyklen

# Routing: Distanzvektor-Routing

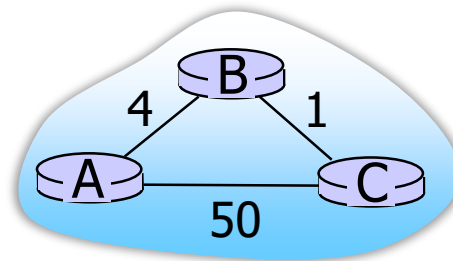
## ■ Beispiel für Vergrößerung mit Poisoned Reverse

- Ausgangspunkt:

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	4	A
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	4	B
C	5	B



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	5	B
B	1	B
C	0	-

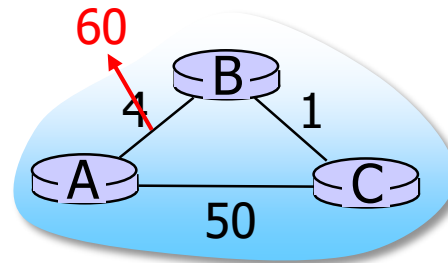
## Routing: Distanzvektor-Routing

- Beispiel für Vergrößerung mit Poisoned Reverse
  - nach Vergrößerung der Kosten

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	60	A
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	5	B
B	1	B
C	0	-

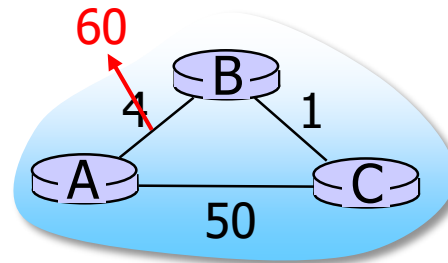
# Routing: Distanzvektor-Routing

- Beispiel für Vergrößerung mit Poisoned Reverse
  - nach 1. Austausch

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	60	A
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	50	A
B	1	B
C	0	-

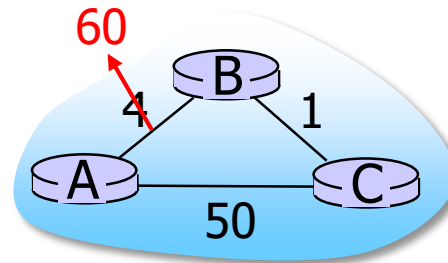
## Routing: Distanzvektor-Routing

- Beispiel für Vergrößerung mit Poisoned Reverse
  - nach 2. Austausch Konvergenz

von <b>B</b> zu	$D_B(\cdot)$	$nh_B(\cdot)$
A	51	C
B	0	-
C	1	C

Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

von <b>A</b> zu	$D_A(\cdot)$	$nh_A(\cdot)$
A	0	-
B	51	C
C	50	C



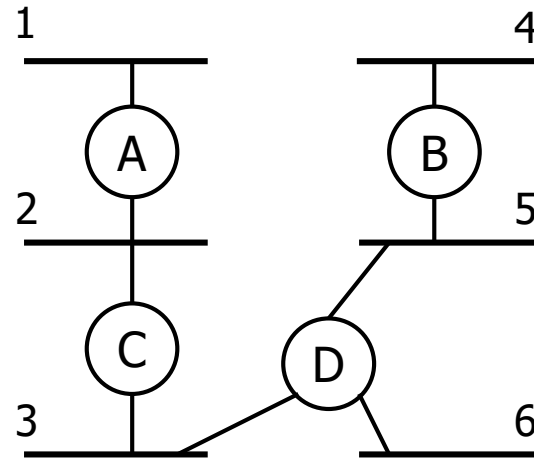
von <b>C</b> zu	$D_C(\cdot)$	$nh_C(\cdot)$
A	50	A
B	1	B
C	0	-

# Routing: Distanzvektor-Routing

## ■ RIP (Routing Information Protocol)

- frühes und verbreitetes Routing-Protokoll, Teil von BSD Unix
- Distanzvektor-Routing
- Router teilen in Advertisements über UDP den Nachbarn mit, welche Netzwerke sie mit welchen Kosten erreichen können
- Advertisements werden periodisch (alle 30 s) und bei Änderungen gesendet
- Kostenmetrik: Anzahl von Hops, maximal 15 (16 für Poisoned Reverse)

Quelle: Peterson, Davie.  
Computer Networks: A Systems  
Approach. Elsevier, 4th Ed.,  
2007.



Beispiel:

Router C teilt A mit, dass er  
Netzwerk 2 und 3 mit Kosten 0,  
Netzwerk 5 und 6 mit Kosten 1,  
Netzwerk 4 mit Kosten 2  
erreichen kann

# Routing: Vergleich

## ■ Link-State-Routing

- zentrales Verfahren
- bei  $n$  Knoten Komplexität des Dijkstra-Verfahrens:  $O(n^2)$
- effiziente Implementierungen schaffen  $O(n \log n)$
- beschränkt Skalierbarkeit
- Nachrichtenaustausch:  $O(ne)$  bei  $e$  Kanten
- Robustheit: nur die weitergegebene Topologie kann fehlerhaft sein

## ■ Erfahrung aus Arpanet

- dynamische Metriken (die von aktueller Netzlast abhängen, z.B. wartende Pakete, Verzögerung, Auslastung), führen zu instabilem Verhalten, daher ggw. i.w. statische Metriken im Einsatz

## ■ Distanzvektor-Routing

- verteilter Algorithmus
- Konvergenzprobleme bei Zyklen
- beschränkt Skalierbarkeit
- Robustheit: Router können fehlerhafte Pfade weitergeben, Fehlerfortpflanzung möglich
- Fehlfunktion eines Routers wirkt sich auf andere aus



# Netzwerkschicht

- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers
- ✓ Routing
  - ✓ Graphen
  - ✓ Link-State-Routing
  - ✓ Distanzvektor-Routing
  - Interdomain-Routing
- IPsec
- Netzvirtualisierung

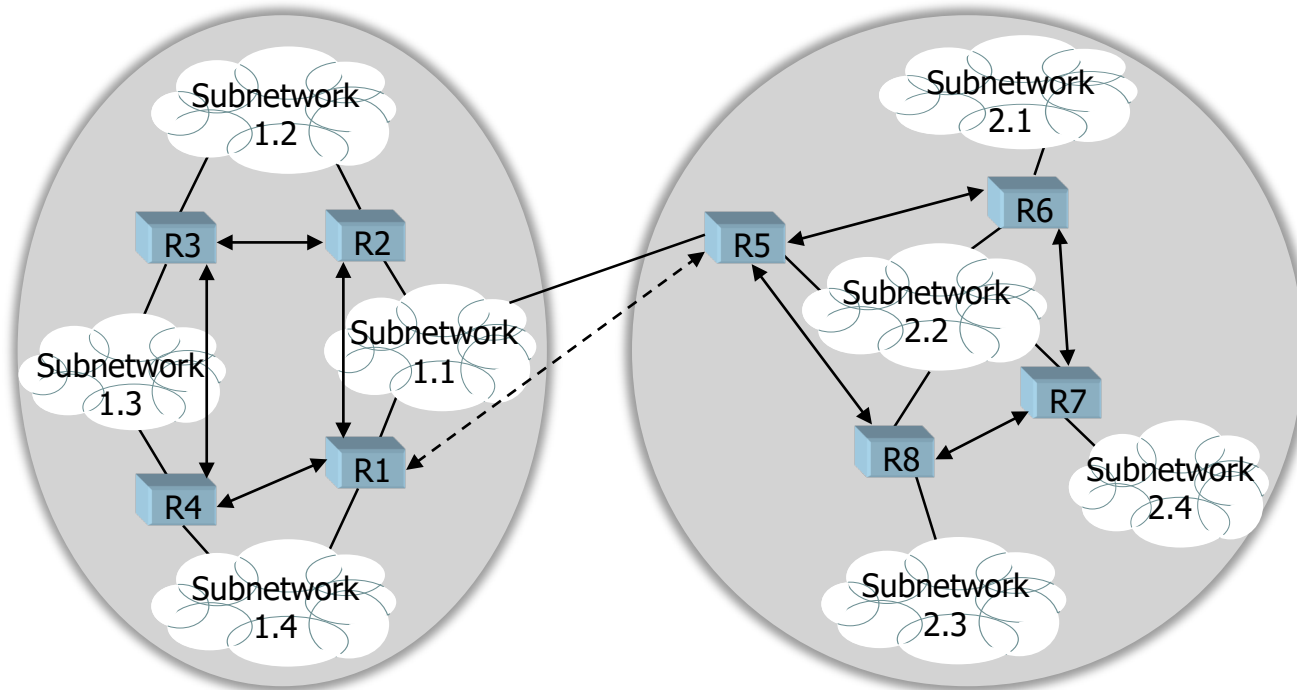
# Routing: Interdomain-Routing

## ■ Routinghierarchie

- innerhalb einer Routingdomäne
  - Intradomain Routing, **Interior Gateway Protokolle (IGP)**
- zwischen Routingdomänen
  - Interdomain Routing, **Exterior Gateway Protokoll (EGP)**, „das“ EGP ist **BGP**: Border Gateway Protocol, RFC 1771
  - Routingdomänen: **autonome Systeme (AS)**
    - **Stub AS**: hat nur eine Verbindung zu anderen AS
    - **Multihomed AS**: hat mehrere Verbindungen zu anderen AS, befördert aber keinen Durchgangsverkehr
    - **Transit AS**: hat mehrere Verbindungen zu anderen AS und befördert Durchgangsverkehr
  - Transit AS gekennzeichnet durch zentral vergebene AS-Nummer (16 Bits)
  - ein Router eines AS ist Gateway und führt EGP aus

# Routing: Interdomain-Routing

- Beispiel für 2 autonome Systeme
  - R1 und R5 sind Gateways



Autonomous System 1

Autonomous System 2

Interior router protocol  $\longleftrightarrow$   
Exterior router protocol  $\longleftrightarrow$

Quelle: W. Stallings.  
*Data and Computer  
Communications*,  
10th Ed., Pearson  
Education, 2014.

# Routing: Interdomain-Routing

## ■ Pfadbasiertes Routing

- Austausch von ganzen Pfaden
- keine Berücksichtigung von Kosten (wegen Größe der beteiligten Netze und uneinheitlicher Metriken nicht möglich)
- Router gibt nur die Pfade bekannt, die andere Router nutzen sollen
- Router sucht sich aus den bekannten Pfaden einen Pfad nach individuellen Regeln aus (z.B. Länge des Pfads, AS im Pfad)
- Zyklen können erkannt werden (ein Router zweimal im Pfad)
- Pfade können auch annulliert werden

# Routing: Interdomain-Routing

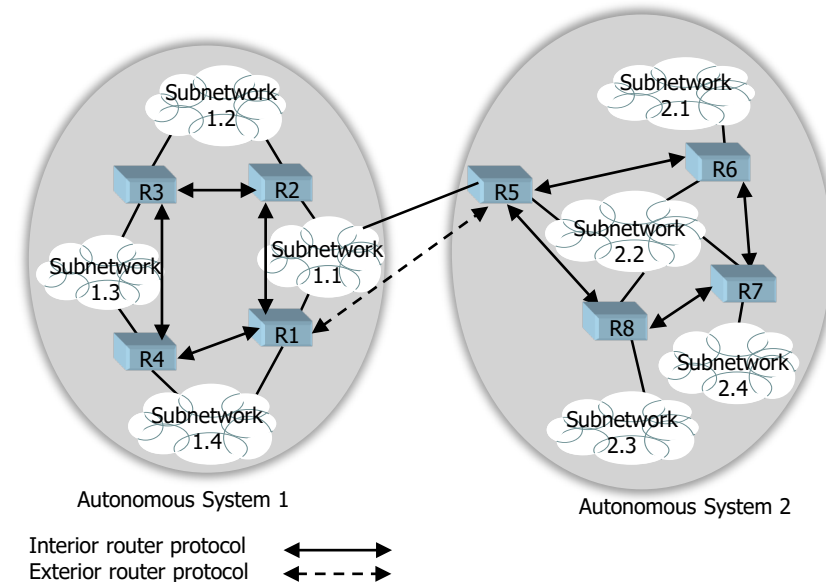
## ■ Funktion von BGP

- Gateways tauschen Advertisements in BGP-Sessions über TCP aus
- Advertisements enthalten Pfade zu Netzwerken, die den Gateways bekannt sind
- ein Pfad besteht aus
  - Liste von Netzwerken (Präfix/Präfixlänge, CIDR) in einem erreichbaren AS
  - Sequenz von AS-Nummern zu diesem AS
  - IP-Adresse des sendenden Gateways
- beim Weiterleiten eines Pfades fügt ein Router sein AS am Anfang des Pfades an und setzt sich als den nächsten Hop ein
- dadurch erlaubt er, dass Verkehr an die Liste von Netzwerken über ihn geleitet wird
- ein Pfad, in dem er selbst in der Sequenz auftaucht, wird verworfen

# Routing: Interdomain-Routing

## ■ Beispiel für den Ablauf von BGP

- R1 erzeugt Routingtabelle für AS1 mit IGP
- R1 schickt Advertisement an R5 (in AS2)
  - Netzwerkliste: alle Subnetze in AS1
  - Sequenz von AS-Nummern: {AS1}
  - Nächster Hop: IP-Adresse von R1
- Annahme: R5 ist Nachbar von R9 in AS3
- R5 leitet die Information von R1 an R9 in Advertisement weiter
  - Netzwerkliste: alle Subnetze in AS1
  - Sequenz von AS-Nummern: {AS2,AS1}
  - Nächster Hop: IP-Adresse von R5
- R9 entscheidet, dass dies sein bevorzugter Pfad zu den Subnetzen in AS1 ist und leitet an seinen Nachbarn den Pfad {AS3, AS2, AS1} weiter

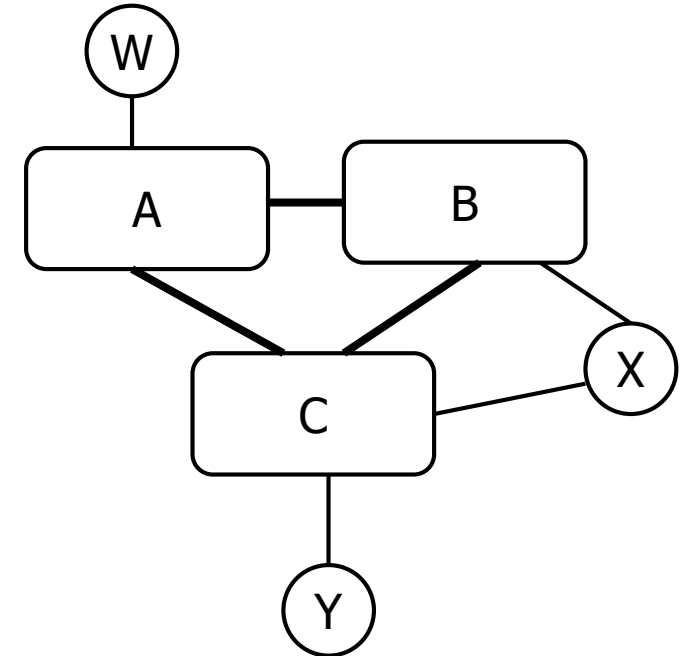


Quelle: W. Stallings. *Data and Computer Communications*, 10th Ed., Pearson Education, 2014.

# Routing: Interdomain-Routing

## ■ Beispielkonfiguration

- A, B, C: Providernetz, Transit AS
- X: Kundennetz, Multihomed AS
- W, Y: Kundennetze, Stub AS
- X möchte keinen Durchgangsverkehr, sendet also keine Advertisements (und kann dies auch gar nicht, da es keine AS-Nummer besitzt)
- A versendet den Pfad AW an B
- B versendet den Pfad BAW an X
- B versendet den Pfad BAW nicht an C, weil B es nicht möchte, dass C Verkehr durch B leitet



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers
- ✓ Routing
- **Ipsec**
- Netzvirtualisierung



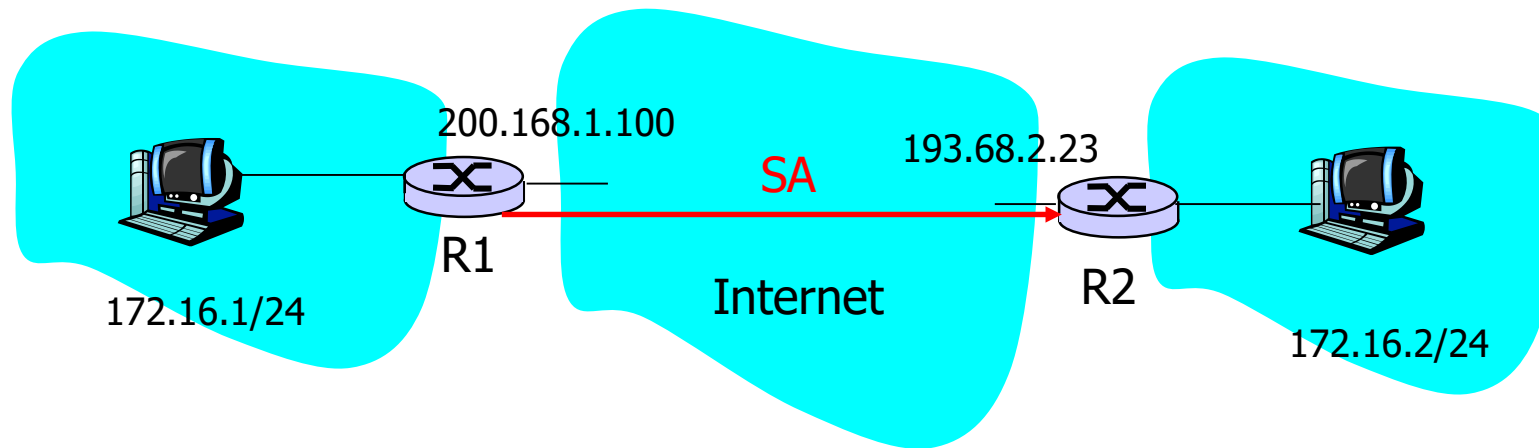
# IPsec

- IPsec: IP security protocol (u.a. RFCs 4301, 6071)
  - Sicherheit auf Netzwerkschicht
  - bietet **Authentizität** des Senders, **Datenintegrität** der Nachrichten, **Vertraulichkeit** des Inhalts und von Protokollinformationen, Schutz gegen Replay-Angriffe
  - ermöglicht ebenfalls **Virtuelle Private Netze (VPNs)**, siehe nächstes Unterkapitel
  - Varianten
    - **Authentication Header (AH)** Protokoll: ermöglicht Authentifikation, Integrität, aber keine Vertraulichkeit
    - **Encapsulation Security Payload (ESP)** Protokoll: ermöglicht Authentifikation, Integrität und Vertraulichkeit
    - **Transport Mode**: ohne Tunneling, kein neuer IP-Header
    - **Tunnel Mode**: mit Tunneling, neuer IP-Header
    - ESP in Tunnel Mode am stärksten verbreitet

# IPsec

## ■ IPsec erfordert die Einrichtung von Security Associations

- **Security Association (SA)**: logische unidirektionale Verbindung zwischen IPsec-Systemen
- **Security Parameter Index (SPI)**: 32-Bit-Identifizier für SA
- **Security Association Database (SAD)**: Datenbank auf IPsec-System für SAs mit IP-Quell- und -Zieladresse, Algorithmen und Schlüssel für Verschlüsselung und MAC
- **Security Policy Database (SPD)**: Datenbank auf IPsec-System für Behandlung von Paketen, z.B. Art der SA für IP-Ziel-Adresse



Quelle: Kurose, Ross.  
*Computer Networking:  
A Top-Down Approach*,  
7th Ed., Pearson  
Education, 2017.

# IPsec

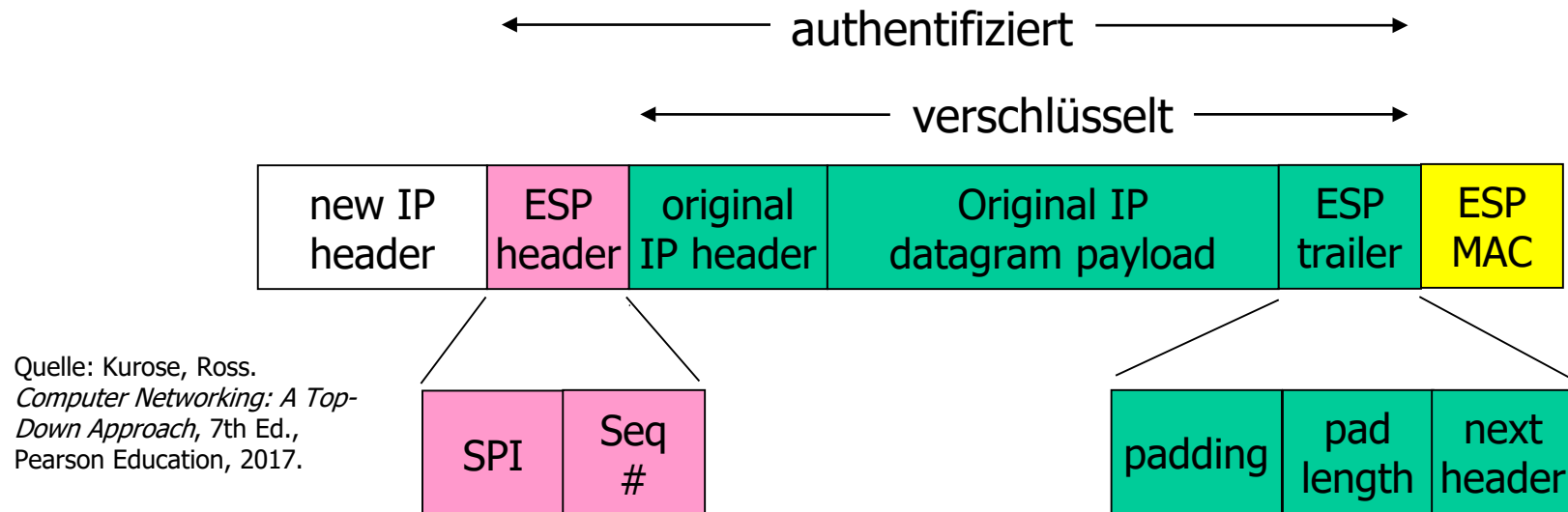
## ■ Schlüsselaustausch in IPsec

- Internet Key Exchange Version 2 (IKEv2) Protokoll, RFC 7296
- beide Kommunikationspartner besitzen Zertifikate für öffentliche Schlüssel
- 1. Phase
  - mit Diffie-Hellmann-Verfahren wird ein symmetrischer Schlüssel erzeugt
  - damit wird eine bidirektionale authentifizierte und vertrauliche IKE SA ( $\neq$  IPsec SAs) aufgebaut
  - außerdem wird ein Master-Secret ausgetauscht
- 2. Phase
  - benötigt nur symmetrische Verschlüsselung
  - für die einzelnen IPsec SAs werden Algorithmen und Parameter ausgehandelt
  - die Schlüssel werden mit Hilfe des Master-Secrets erzeugt

# IPsec

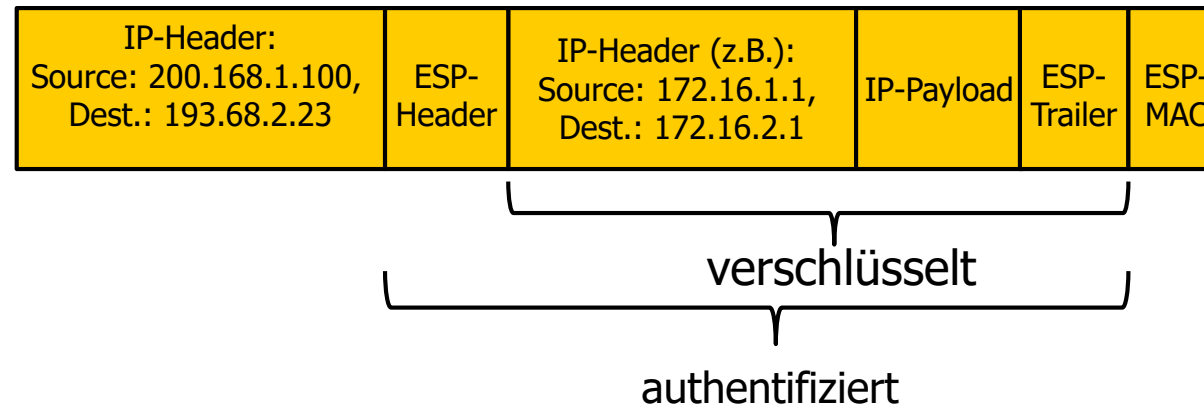
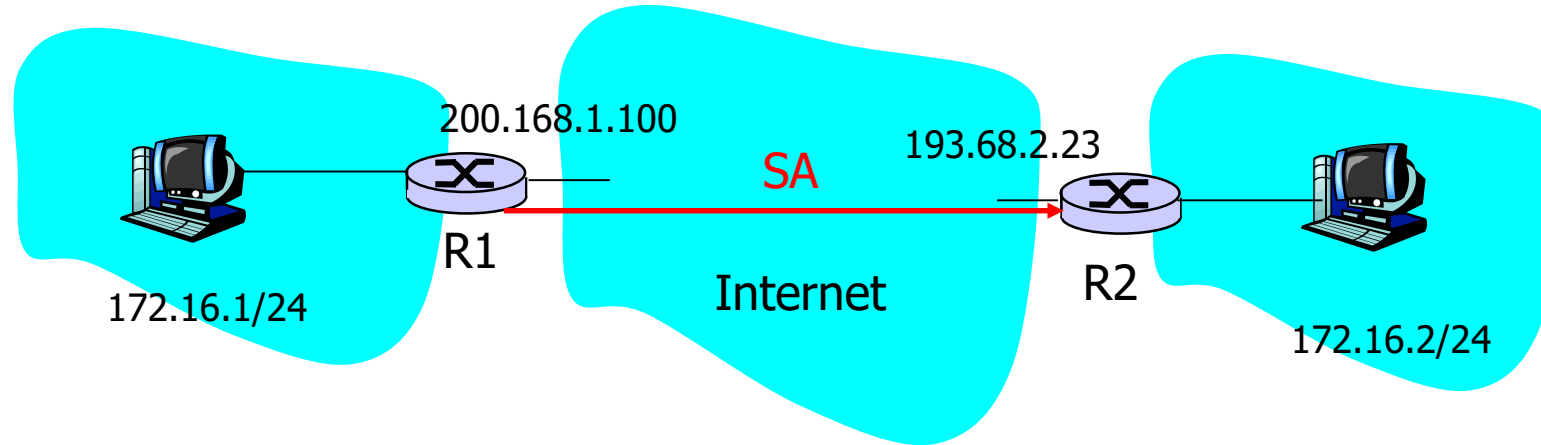
## ■ ESP-Paketformat im Tunnelmodus

- Original-IP-Datagramm wird mit ESP-Trailer auf die für die Verschlüsselung benötigte Länge aufgefüllt, dies wird verschlüsselt (gemäß Schlüssel für SA)
- ESP-Header enthält SPI (damit Empfänger SA in seiner SAD findet)
- dies wird mit MAC authentifiziert und in ESP MAC geschrieben
- dies erhält einen neuen IP-Header mit IP-Adressen der IPsec-Systeme und mit 50 für IPsec-ESP im Protokollfeld



# IPsec

## ■ Bsp.: ESP im Tunnelmodus:



Quelle: Kurose, Ross.  
*Computer Networking: A Top-Down Approach*, 7th Ed.,  
Pearson Education, 2017.

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers
- ✓ Routing
- ✓ IPsec
- **Netzvirtualisierung**
  - VPNs mittels IP-Tunneling
  - Multiprotocol Label Switching (MPLS)
  - Software-Defined Networking (SDN)

# Netzvirtualisierung

## ■ Private Netze

- Unternehmen kann durch Eigentum oder Miete (z.B. von Telekommunikationsunternehmen) von Leitungen zwischen Standorten sein eigenes in sich geschlossenes Netz betreiben
- Vorteile: Auslastung liegt in eigener Hand, hohe Sicherheit (getrennt von anderen Netzen, physikalisches Eindringen erforderlich)
- Nachteil: hohe Kosten

## ■ Virtual Private Networks (VPNs)

- auf öffentlichem Netz Overlay aufsetzen, das wie ein privates Netz erscheint, Varianten:
  - **End-to-Site**: virtuelle Integration eines Hosts in ein anderes Netz, z.B. Anbindung eines mobilen Hosts an ein Unternehmensnetz, erscheint wie ein Host im Netzwerk
  - **Site-to-Site**: Verbindung von Netzen, z.B. an unterschiedlichen Standorten
  - **End-to-End**: Verbindung von Hosts zu einem virtuellen Netz

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers
- ✓ Routing
- ✓ IPsec
- ✓ Netzvirtualisierung
  - VPNs mittels IP-Tunneling
  - Multiprotocol Label Switching (MPLS)
  - Software-Defined Networking (SDN)



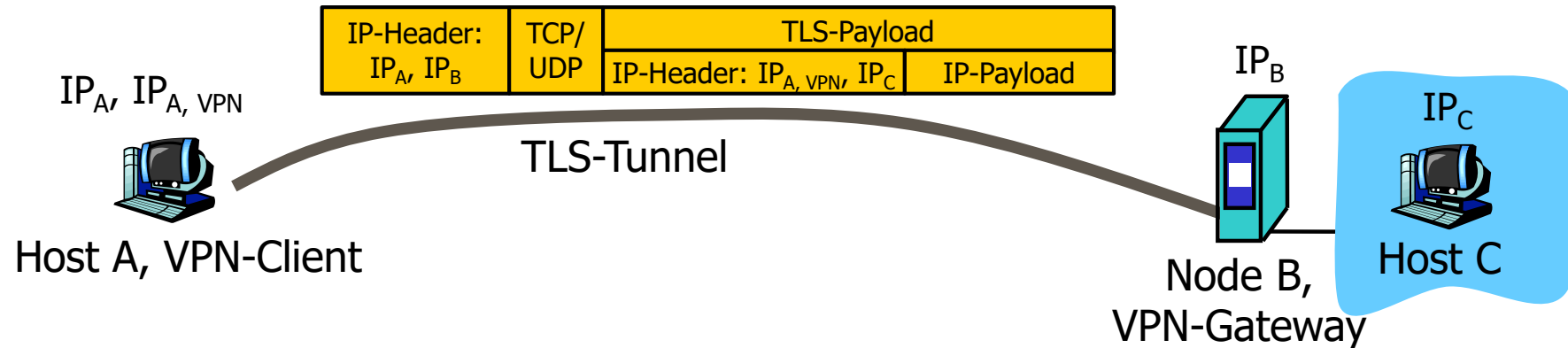
# VPNs mittels IP-Tunneling

## ■ IP-Tunneling

- **Tunneling**: Einbettung von Paketen eines Protokolls in Pakete eines Protokolls der gleichen oder einer höheren Schicht
- **TSL-VPN** (z.B. OpenVPN): Tunneling von IP in TLS
- **IPsec-VPN**: Tunneling von IP in IPsec
- wahlweise mit Authentifikation, Integrität und Verschlüsselung

# VPNs mittels IP-Tunneling

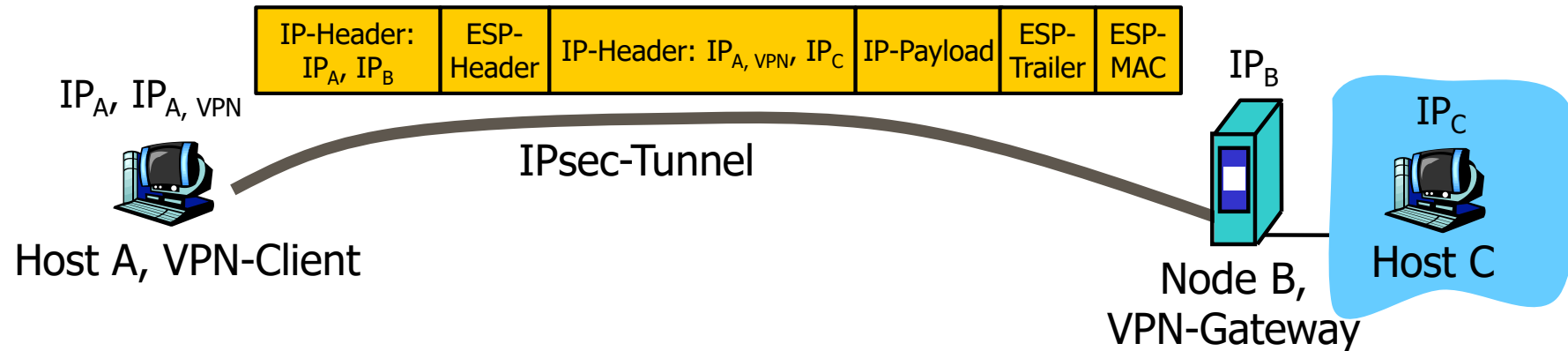
## ■ TLS-VPN, End-to-Site



- A baut sichere Transportverbindung zu B auf mit TLS: Aushandlung von Authentifizierung, Datenintegrität, Verschlüsselung
- Tunneling von DHCP durch TLS, um A eine zusätzliche im VPN gültige IP-Adresse  $IP_{A, VPN}$  zuzuweisen
- Tunneling von IP-Paketen durch TLS, A erscheint wie ein Teil des Netzwerks
- weitere Konfigurationsmöglichkeiten, z.B. ohne Verschlüsselung

# VPNs mittels IP-Tunneling

## ■ IPsec-VPN, End-to-Site



- A baut Security Association (SA) zu B auf:  
Aushandlung von Protokollversion, hier ESP, Zuweisung von IP-Adresse  $IP_{A, VPN}$
- Tunneling von IP-Paketen durch ESP, A erscheint wie ein Teil des Netzwerks
- ermöglicht zusätzlich Authentifizierung, Datenintegrität und Verschlüsselung, diverse Konfigurationsmöglichkeiten

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers
- ✓ Routing
- ✓ IPsec
- ✓ Netzvirtualisierung
  - ✓ Virtual Private Networks (VPNs)
  - Multiprotocol Label Switching (MPLS)
  - Software-Defined Networking (SDN)

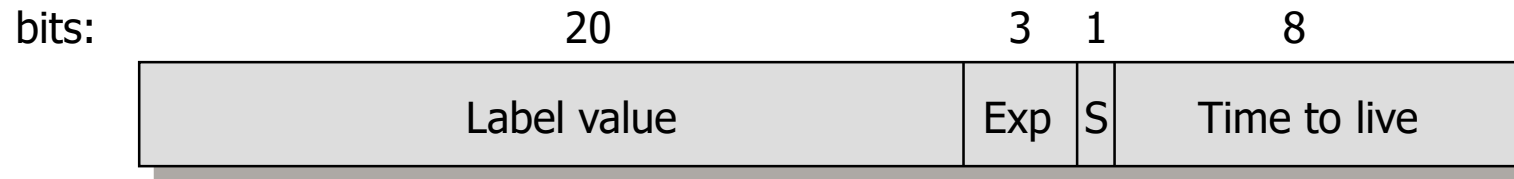
# Multiprotocol Label Switching (MPLS)

## ■ Multiprotocol Label Switching (MPLS)

- Einsatz bei fast jedem großen ISP
- Name: kann mit mehreren Schicht 2 (u.a. ATM, Ethernet, PPP) und Schicht 3 (u.a. IPv4, IPv6) Protokollen zusammenarbeiten
- virtuelle Leitungsvermittlung
  - Pakete erhalten **Label**
  - Weiterleitung mit **Label-Switching**: jeder Router erkennt Eingangsport und Label, sieht in Tabelle nach, welches das neue Label ist und auf welchem Ausgangsport das Paket weitergeleitet wird
  - Signalisierung zum Aufbau eines **Label-Switched Path (LSP)**
- Vorteile
  - **schnelleres Weiterleiten** mit Labeln statt mit IP-Adressen (vermeidet Longest Prefix Match in Routern)
  - **Traffic Engineering**: zur optimalen Verkehrsführung LSPs gezielt aufbauen
  - **Virtual Private Networks (VPNs)**: LSPs als Tunnel zwischen Nutzern
  - **Dienstgüteeigenschaften (QoS)** für LSPs möglich

# Multiprotocol Label Switching (MPLS)

## ■ MPLS Header ([Shim Header](#))



Quelle: Leon-Garcia,  
Widjaja: *Communication  
Networks: Fundamental  
Concepts and Key  
Architectures*, 2nd ed.,  
McGraw-Hill, 2004.

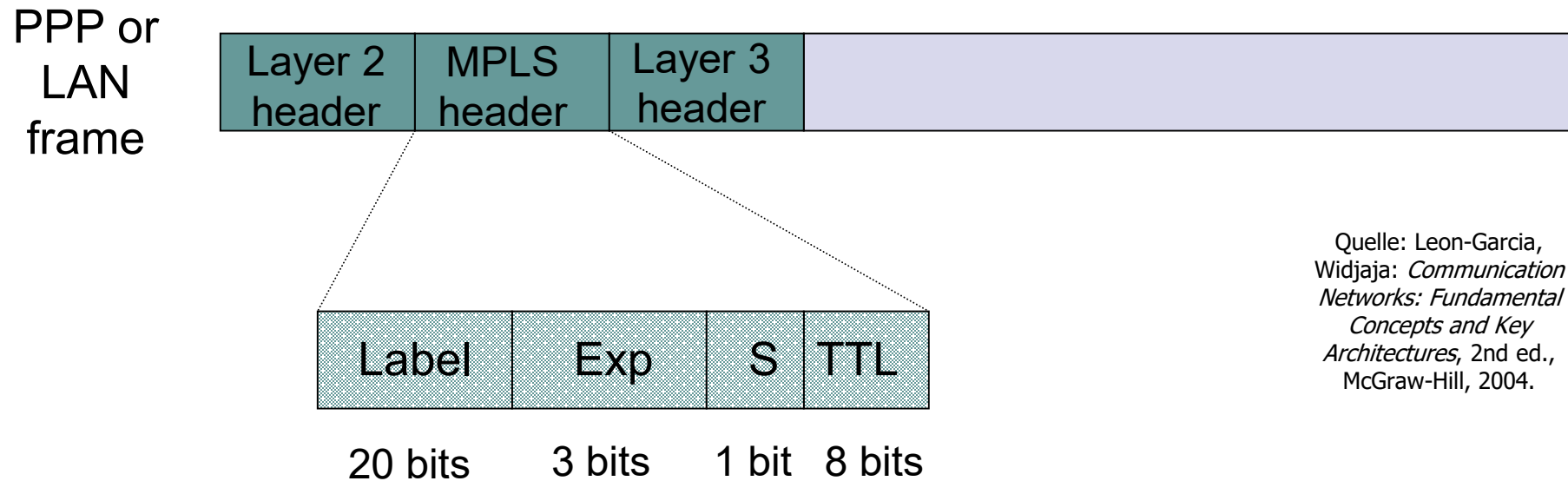
Exp = experimental  
S = bottom of stack bit

- Label Value: 20 Bits
- Exp: 3 Bits für experimentelle Nutzung, z.B. für DiffServ-Klassen
- S: MPLS Header können gestapelt werden ([Stack](#)), Weiterleitung erfolgt mit obersten Label (links), S=1 kennzeichnet untersten Wert (rechts, Bottom of Stack)
- TTL: Time-To-Live, Übernahme und Dekrementieren aus IP-Paket

# Multiprotocol Label Switching (MPLS)

## ■ Position des MPLS Headers

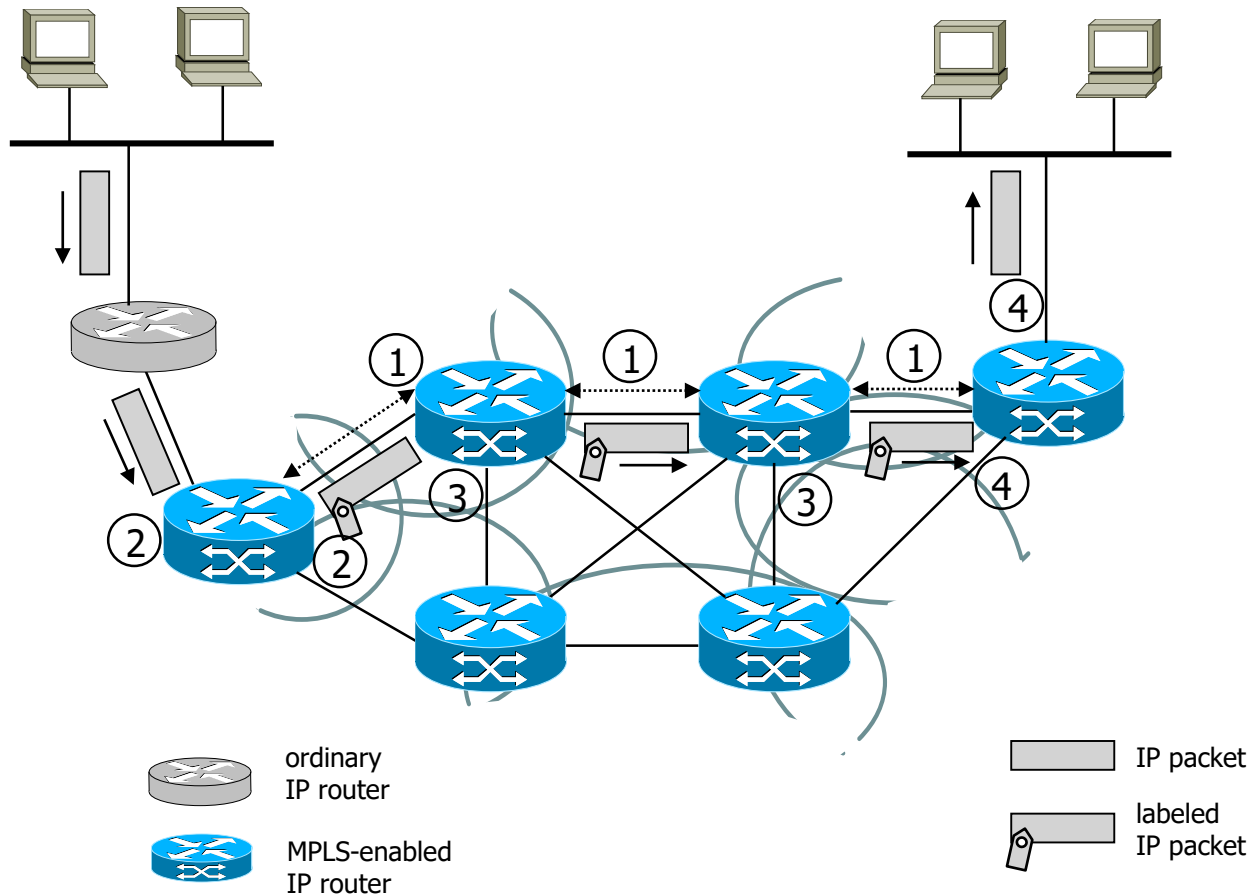
- zwischen Schicht 2 (z.B. Ethernet, PPP) und Schicht 3 (IP)



Quelle: Leon-Garcia,  
Widjaja: *Communication  
Networks: Fundamental  
Concepts and Key  
Architectures*, 2nd ed.,  
McGraw-Hill, 2004.

# Multiprotocol Label Switching (MPLS)

- Weiterleiten von IP-Paketen mit Labels in einer MPLS-Domain über einen LSP:

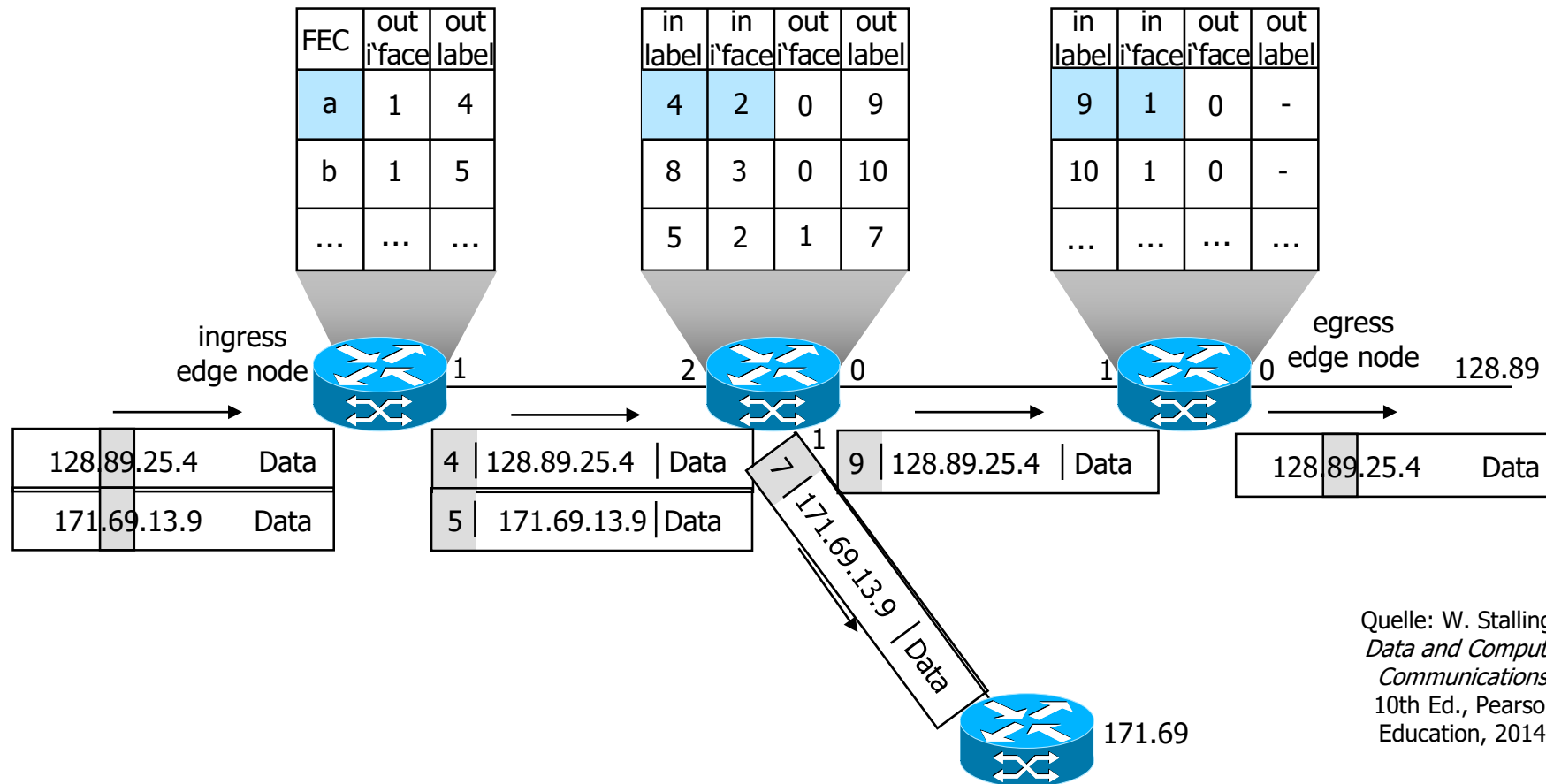


Quelle: W. Stallings.  
*Data and Computer  
Communications*,  
10th Ed., Pearson  
Education, 2014.



# Multiprotocol Label Switching (MPLS)

## ■ Weiterleitungstabellen in jedem Label-Switching Router (LSR):



Quelle: W. Stallings.  
*Data and Computer  
Communications*,  
10th Ed., Pearson  
Education, 2014.

# Netzwerkschicht

- ✓ Einführung
- ✓ IP
- ✓ Aufbau eines Routers
- ✓ Routing
- ✓ Netzvirtualisierung
  - ✓ Virtual Private Networks (VPNs)
  - ✓ Multiprotocol Label Switching (MPLS)
  - Software-Defined Networking (SDN)

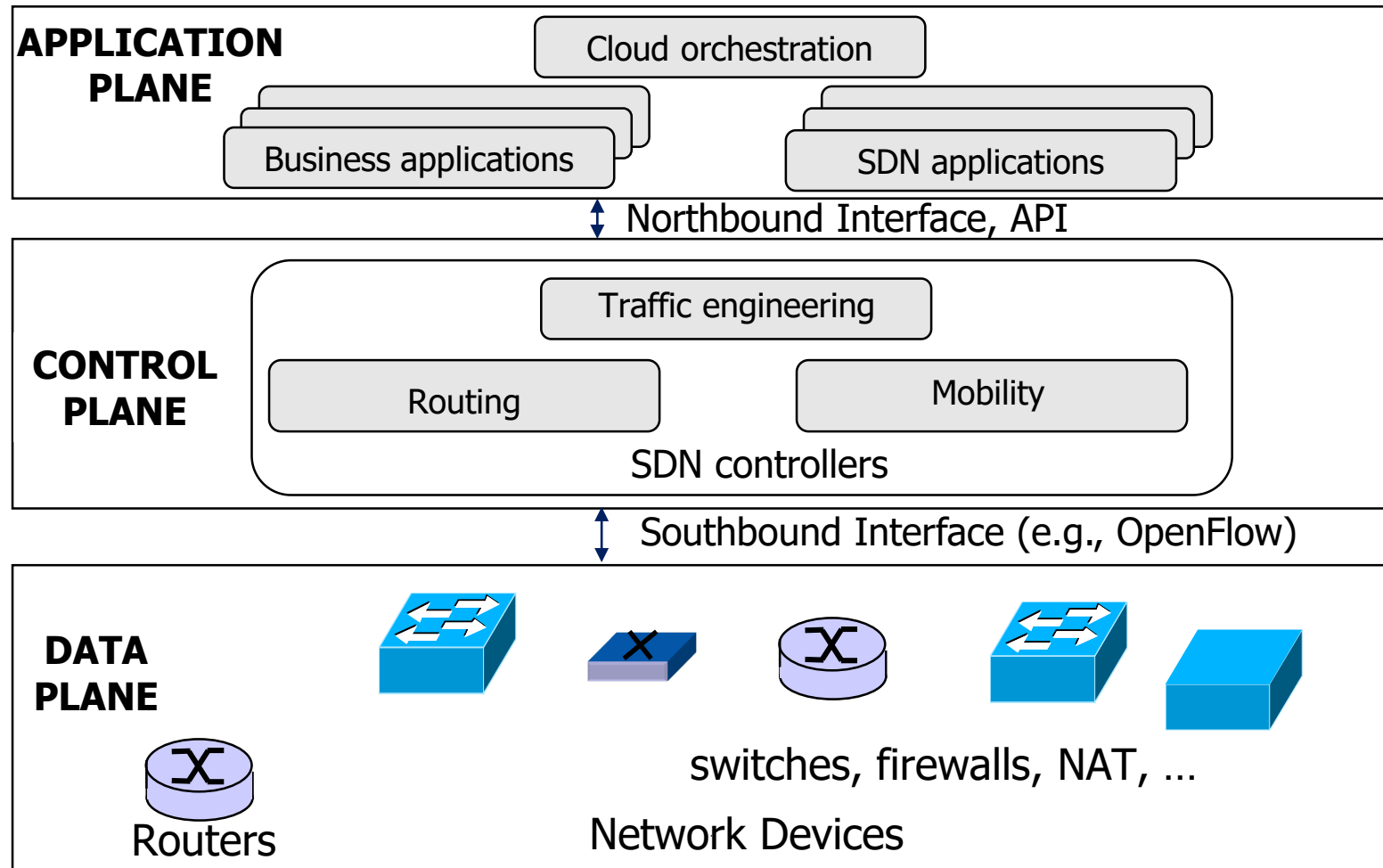
# Software-Defined Networking (SDN)

## ■ Software-Defined Networking (SDN)

- RFC 7426, Industriekonsortium: Open Networking Foundation (ONF)
- Trennung von **Data Plane** (zur Weiterleitung von Daten), **Control Plane** (für Steuerfunktionen) und **Application Plane**, bisher waren diese Funktionen in Netzwerkgeräten integriert und über die Geräte verteilt
- Protokoll zwischen Control und Data Plane z.B. **OpenFlow**
- **SDN-Controller** instruiert **OpenFlow-Switch**, wie Pakete aufgrund von Attributen (z.B. Header-Felder, MPLS-Label) und internen Variablen behandelt werden sollen (z.B. weiterleiten, verwerfen, zu SDN-Controller senden)
- über API kann SDN-Anwendung programmiert werden
- Vorteile: Einsatz von preiswerter HW auf Data Plane, Vermeidung der „Protokollsuppe“, zentrale Sicht auf Netzwerk, flexible Programmierung von Netzen, offene Schnittstellen, Open Source in Netzwerken, schnelle Reaktion auf Netzsituationen (z.B. Traffic Engineering), Bedürfnisse von Rechenzentren, ...

# Software-Defined Networking (SDN)

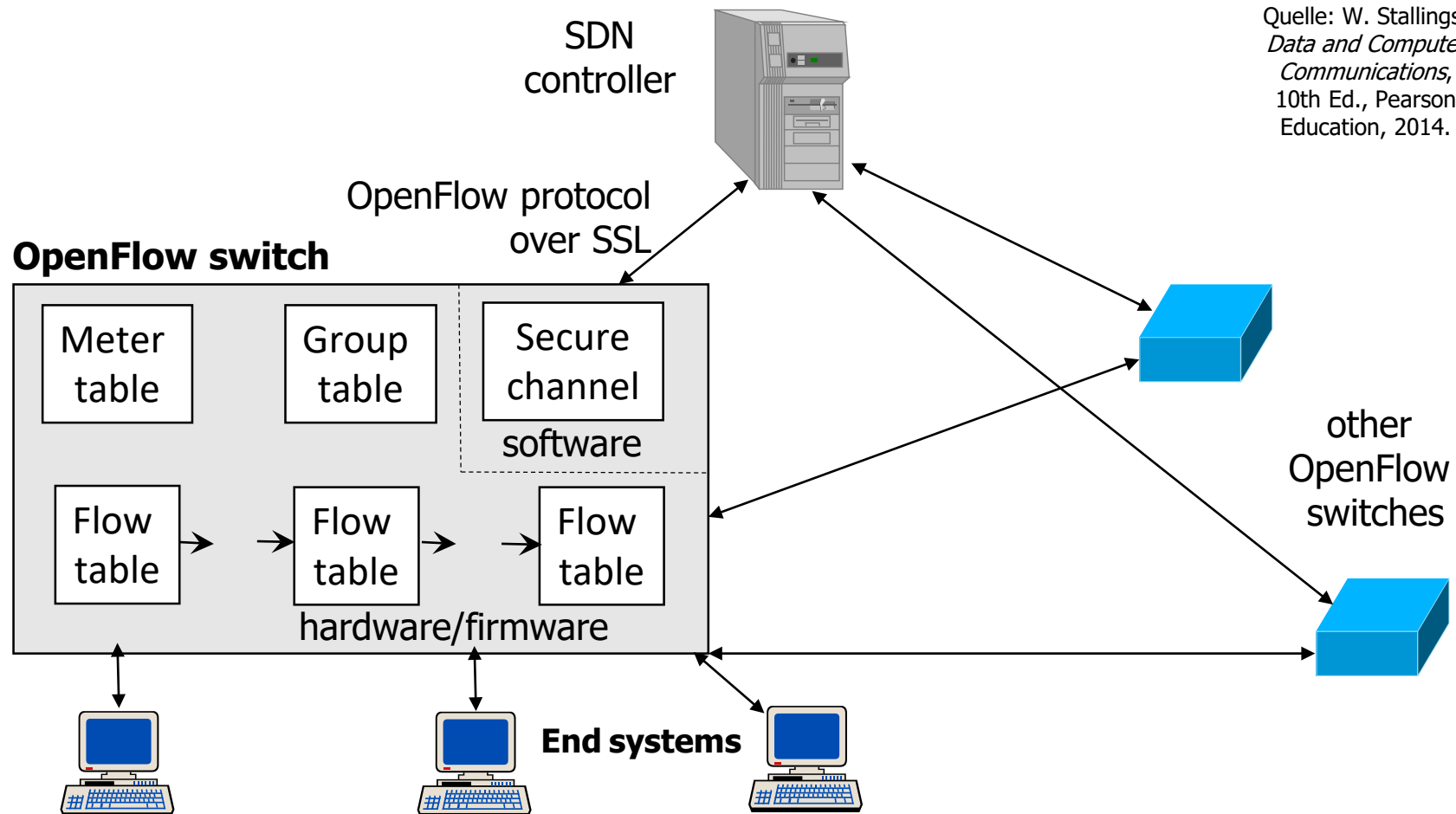
## ■ SDN-Architektur:



Quelle: W. Stallings.  
*Data and Computer  
Communications*,  
10th Ed., Pearson  
Education, 2014.

# Software-Defined Networking (SDN)

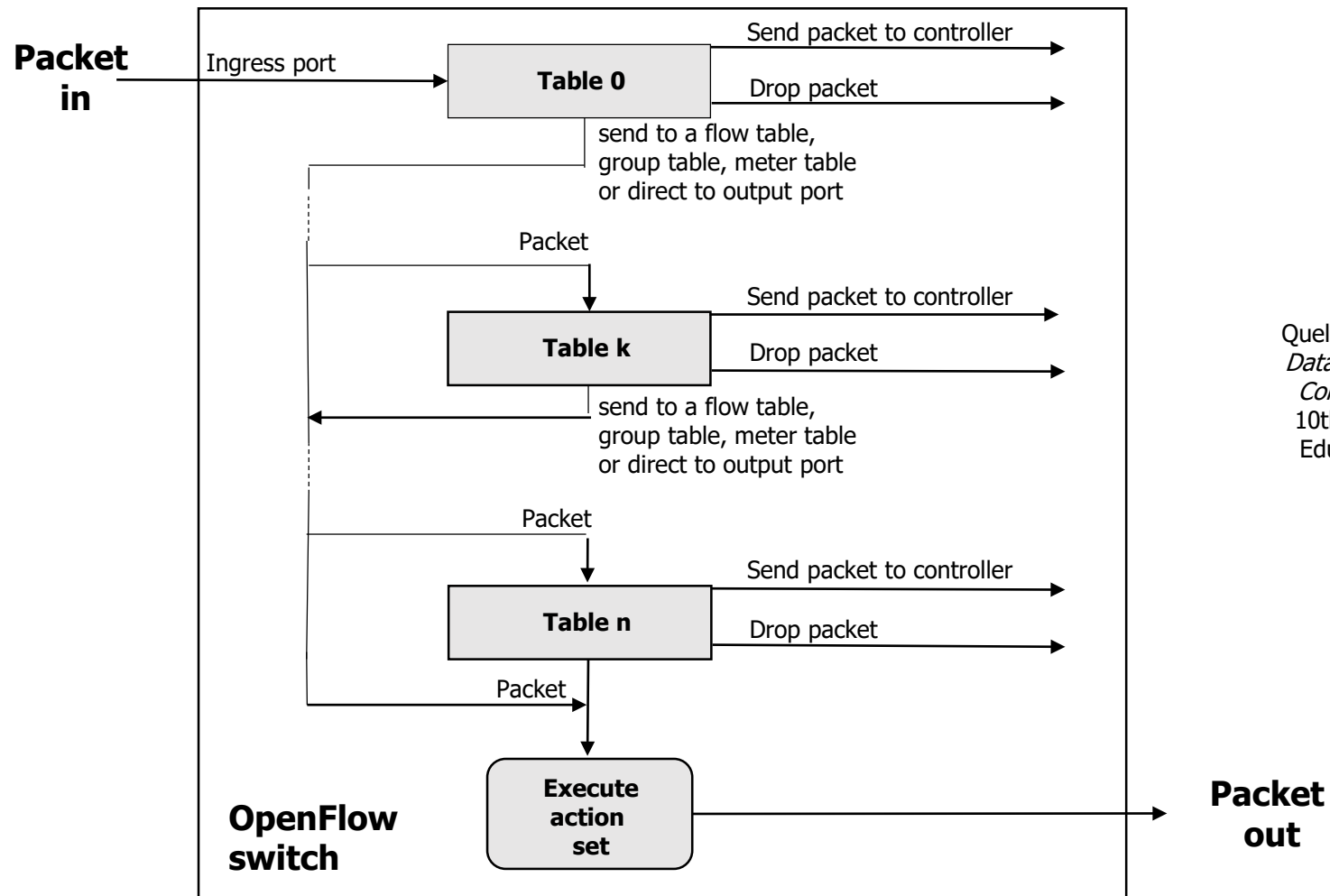
## ■ Architektur eines OpenFlow-Switch:



Quelle: W. Stallings.  
*Data and Computer Communications*,  
10th Ed., Pearson  
Education, 2014.

# Software-Defined Networking (SDN)

## ■ Behandlung eines Pakets in einem OpenFlow-Switch:

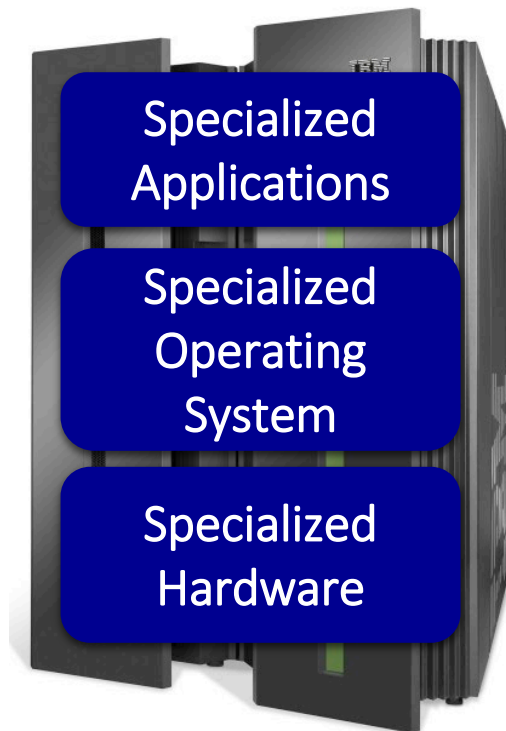


Quelle: W. Stallings.  
*Data and Computer Communications*,  
10th Ed., Pearson Education, 2014.

# Software-Defined Networking (SDN)

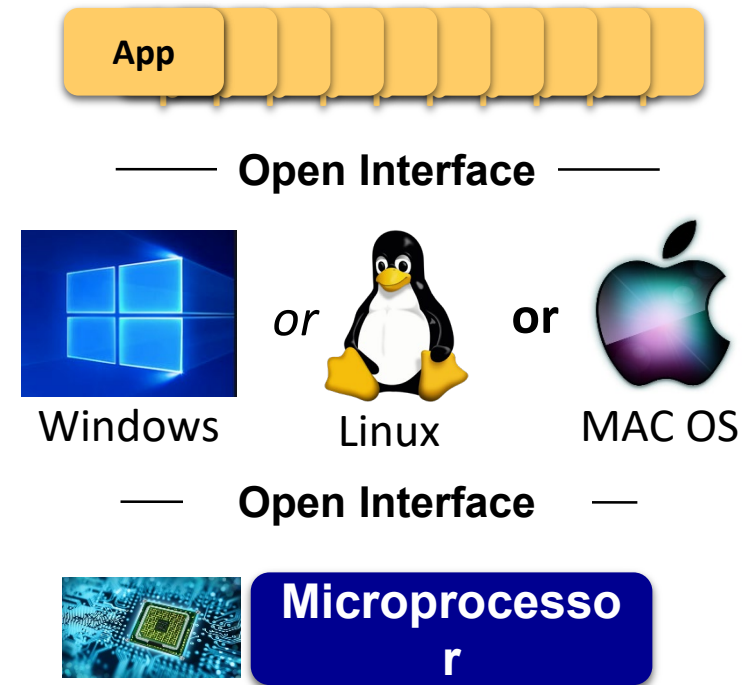
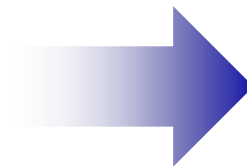
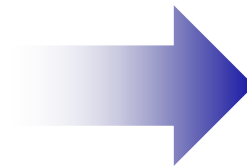
Quelle: Kurose, Ross.  
*Computer Networking:  
A Top-Down Approach*,  
7th Ed., Pearson  
Education, 2020.

- Analogie zu Entwicklung weg von Mainframes:



Vertically integrated  
Closed, proprietary  
Slow innovation  
Small industry

Rechnerkommunikation, Netzwerkschicht



Horizontal  
Open interfaces  
Rapid innovation  
Huge industry