

Rechnernetze

Kapitel 4: Die Internetschicht

Hochschule Ulm
Prof. Dr. F. Steiper



Rechnernetze, INF2, 2022

- *Urheberrechte*

- ▶ *Die Vorlesungsmaterialien und Vorlesungsaufzeichnungen zum Kurs „Rechnernetze (INF2)“ dürfen nur für private Zwecke im Rahmen Ihres Studiums an der Technischen Hochschule Ulm genutzt werden.*
- ▶ *Eine Vervielfältigung und Weitergabe dieser Materialien in jeglicher Form an andere Personen ist untersagt.*
- ▶ *© Copyright. Frank Steiper. 2022. All rights reserved*

4.1 Aufgaben und Dienste

- *Aufgaben der Internetschicht*

[Ref 2] Kapitel 5, Seite 413-417

- ▶ *Wegfindung (Routing)*

- *Ende-zu-Ende-Kommunikation*
 - *Weiterleitung der Pakete, auch über verschiedene Teilnetze hinweg*
- *Bereitstellung eines globalen Adressierungsschemas*
 - *Unabhängig von spezifischen Adressierungsmethoden in physikalischen Teilnetzen*
- *Ermittlung eines optimalen Pfads zwischen Quell- und Ziel-Rechner*

- ▶ *Abstraktion*

- *Dienste der Vermittlungsschicht verbergen Eigenschaften physikalischer Teilnetze vor der Transportschicht*
 - *Anzahl, Art, Topologie...*

- ▶ *Überlastkontrolle (→ hier nicht weiter behandelt)*

- *Die Anzahl der von Quellrechnern generierten Pakete kann größer als die Übertragungskapazität eines Routers/einer Verbindung sein*

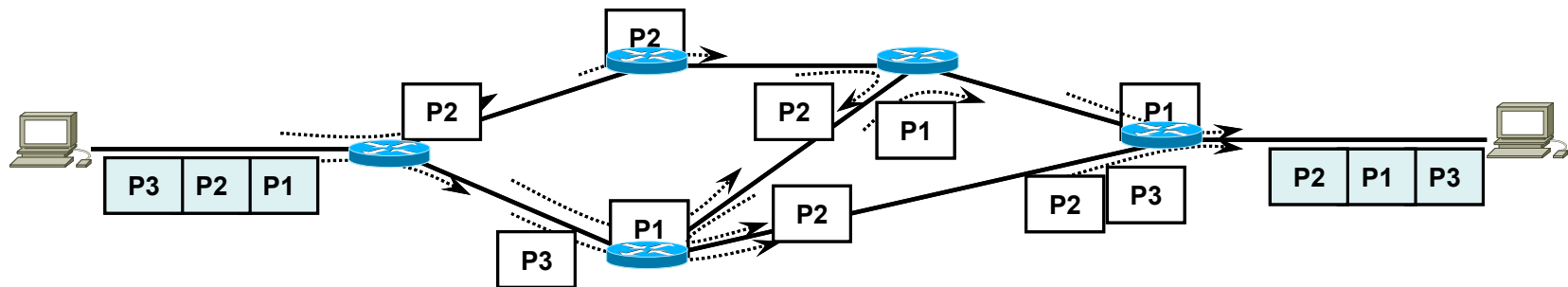
4.1 Aufgaben und Dienste

- *Dienstmodelle*

[Ref 1] Kapitel 4, Seite 346-360

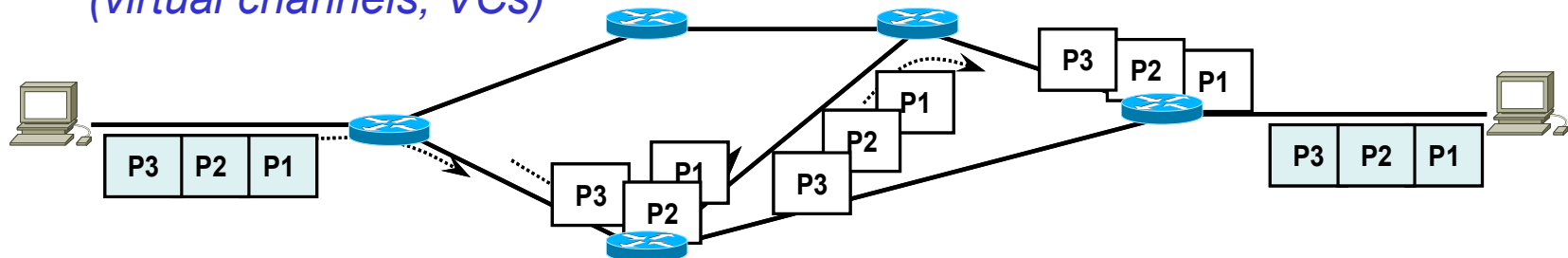
- *Verbindungslose Dienste*

- *Individuelle Weiterleitung* von Einzelpaketen auf Grund der Zieladresse



- *Verbindungsorientierte Dienste*

- Konfiguration, Aufbau und Abbau von *virtuellen Kanälen* (virtual channels, VCs)



4.1 Aufgaben und Dienste

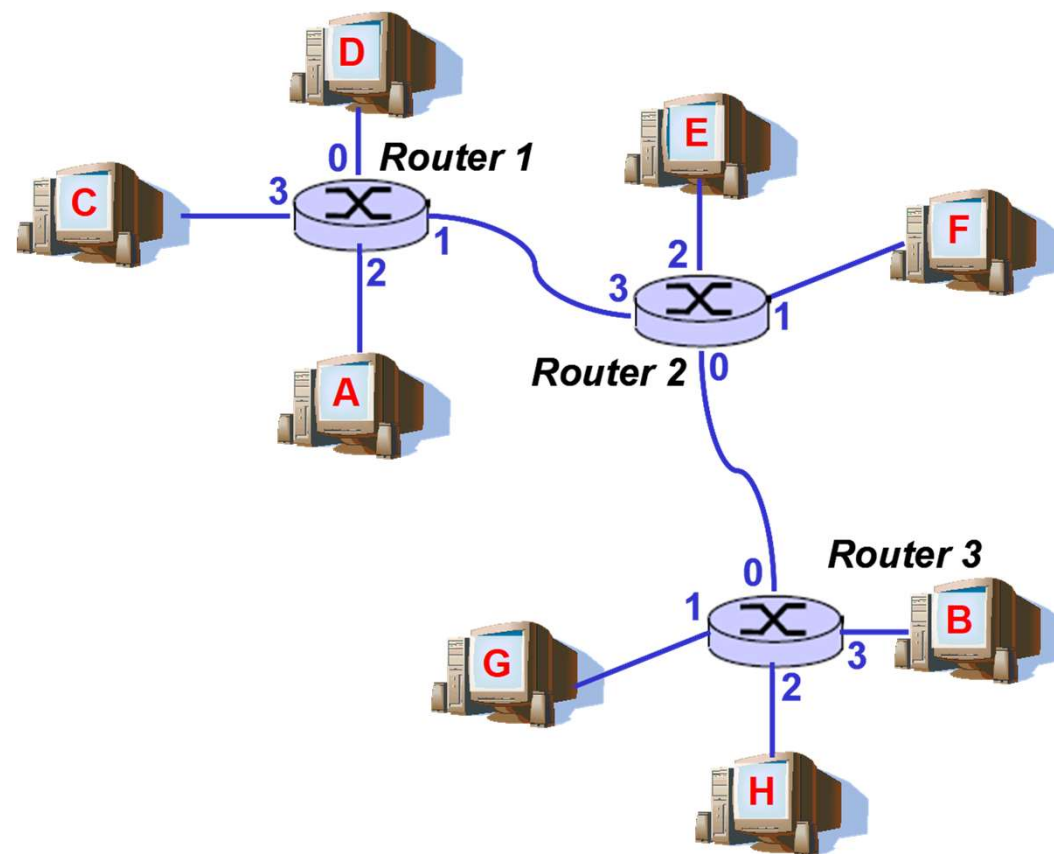
- *Verbindungslose Dienste (Datagram Services)*
 - *Minimalistischer Ansatz: unzuverlässiger Dienst*
 - *Jedes Einzelpaket enthält die gesamte Adressinformation um es über Vermittlungsknoten an das Ziel leiten zu können*
 - *Im Internet vorherrschend: IP implementiert verbindungslosen Dienst!*
 - *Merkmale*
 - *Ein Quellrechner versieht jedes Paket mit der Zieladresse*
 - *Der Quellrechner hat keine Möglichkeit festzustellen, ob das Paket zugestellt wurde oder der Zielrechner in Betrieb ist*
 - *Zwei aufeinander folgende Pakete mit gleichem Ziel können unterschiedliche Wege durchlaufen*
 - *Der Ausfall eines Vermittlungsknotens hat keine schwerwiegenden Konsequenzen, so lange redundante Pfade existieren*

4.1 Aufgaben und Dienste

- *Verbindungslose Dienste (Datagram Services)*

*Weiterleitungstabelle
für Router 2:*

Ziel	Port
A	3
B	0
C	3
D	3
E	2
F	1
G	0
H	0



4.2 Routing-Algorithmen

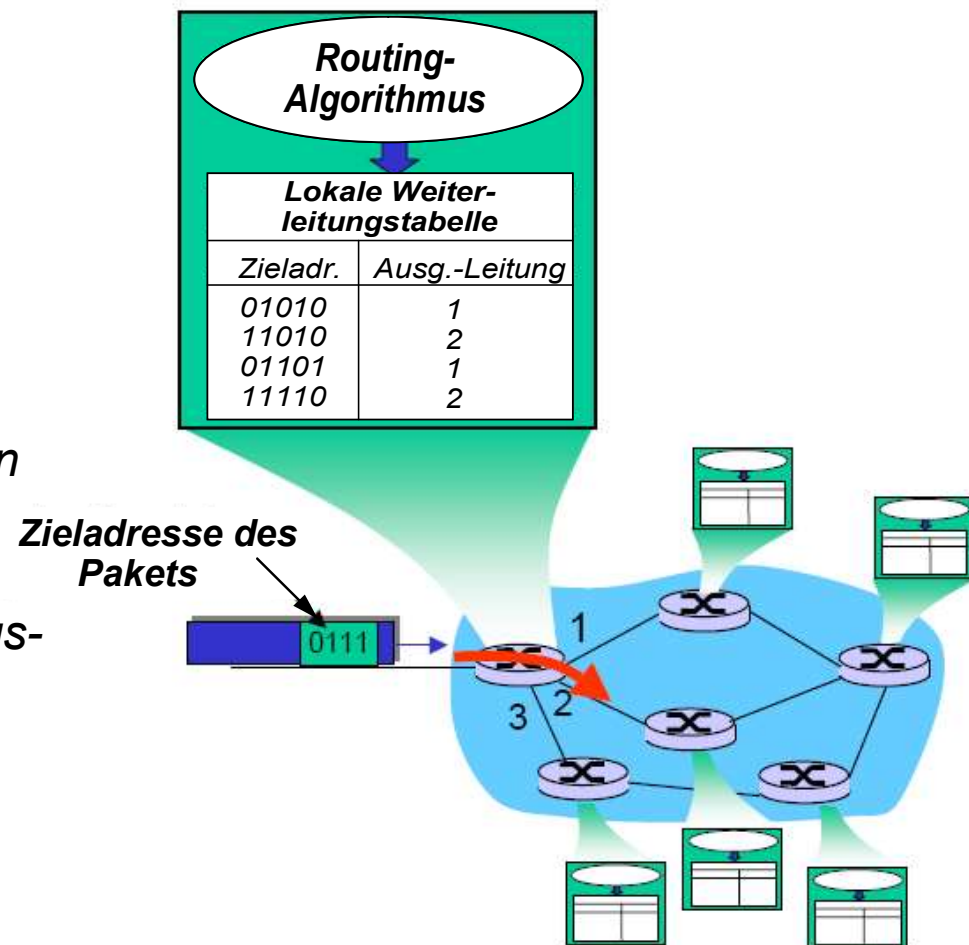
- *Routing vs. Weiterleitung*

- *Routing*

- Prozess, durch den in einem Router die Inhalte der lokalen Weiterleitungstabelle erzeugt werden

- *Weiterleitung*

- Prozess, bei dem durch den Vergleich der Zieladresse eines Pakets mit den Einträgen der Weiterleitungstabelle ermittelt wird, über welche Verbindungsleitung ein Paket weiter geleitet wird



4.2 Routing-Algorithmen

- *Routing als graphentheoretisches Problem* [Ref 1] Kapitel 4, Seite 404-407

- ▶ *Graph $G=(N,E)$*

- *N ist eine Menge von Routern $\{u,v, \dots\}$*
- *E ist eine Menge von Verbindungsleitungen $\{(u,v), (u,x), \dots\}$*

- ▶ *Pfad*

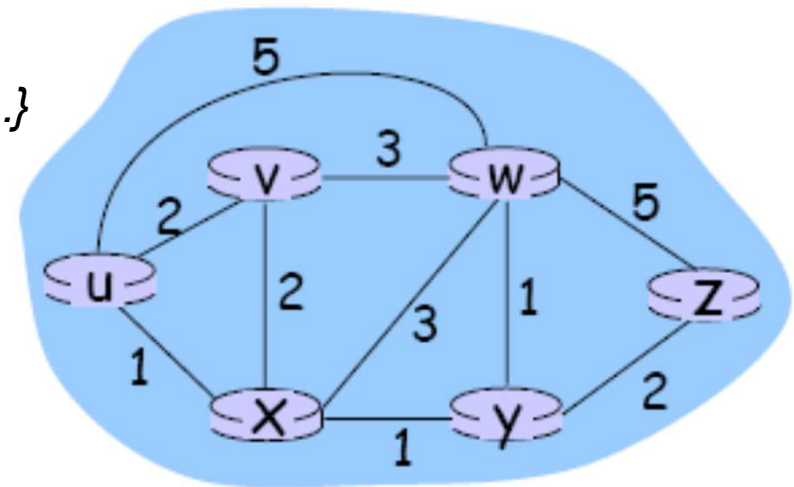
- *Ein Pfad ist eine Sequenz von Verbindungsleitungen*

- ▶ *Kosten*

- *$C(X,X')$ sind die Kosten einer Verbindungsleitung zwischen den Routern X und X'*

- ▶ *Least Cost Path:*

- *Routing-Algorithmen **minimieren die Gesamtkosten** für den Pfad zwischen Quelle und Ziel*



4.2 Routing-Algorithmen

- *Klassifizierung von Routing-Algorithmen (RA)*
 - *Globaler RA*
 - Benötigt vorab die *vollständige Information* über das Gesamtnetzwerk
 - Die Routenberechnung selbst könnte irgendwo im Netz an zentraler Stelle durchgeführt werden
 - In der Praxis werden diese Protokolle oft als „*Link State*“-Algorithmen bezeichnet
 - *Dezentraler RA*
 - Berechnung des „optimalen“ Pfades mit *verteiletem* Algorithmus
 - *Kein Knoten verfügt über die vollständige Information* des Gesamtnetzes
 - Jeder Knoten kennt nur die Kosten für die direkt angeschlossenen Verbindungsleitungen
 - Direkt benachbarte Router tauschen ihre aktuellen Routing-Informationen aus
 - „*Distance-Vector*“-Algorithmen gehören zu den dezentralen RA

4.2.1 LSA, Link-State Algorithmus

- *Ein Link-State-Algorithmus: Der Dijkstra-Algorithmus*

- ▶ *Voraussetzung*

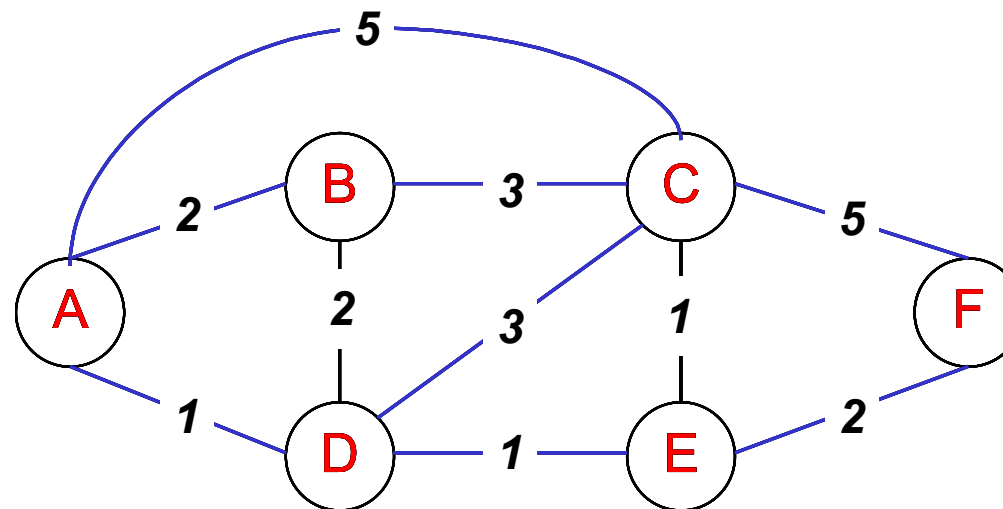
[Ref 1] Kapitel 4, Seite 407-412

- *Die Kosten aller Verbindungsleitungen des Netzwerks sind bekannt*

- ▶ *Ergebnis*

- *Berechnet iterativ die optimalen Pfade von einer Quelle zu allen anderen Knoten im Netz*

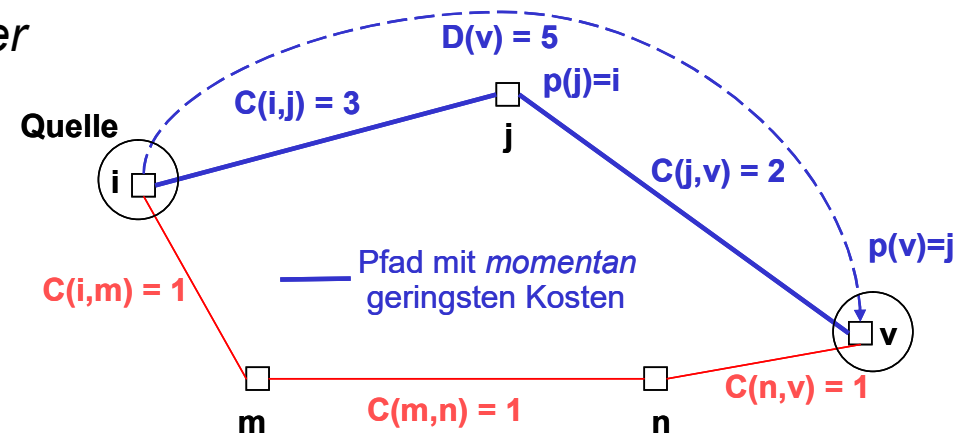
- ▶ *Beispiel-Graph:*



4.2.1 LSA, Link-State Algorithmus

• Definitionen

- $c(i,j)$ Verbindungskosten von Knoten i zu Knoten j ;
Wenn keine Verbindung zwischen i und j besteht,
wird $c(i,j)=\infty$ gesetzt
- $D(v)$ Kosten des Pfads vom Quell-Knoten zum Ziel v , der *momentan*
die geringsten Kosten
besitzt (vom Stand der
Iteration abhängig)




- $p(v)$ Vorheriger Knoten
(Nachbar von v) auf dem
momentan optimalen Pfad von der Quelle zum Knoten v
- N' Menge der Knoten, zu denen der optimale Pfad mit den
geringsten Kosten bereits bekannt ist

4.2.1 LSA, Link-State Algorithmus

- *Der Algorithmus*

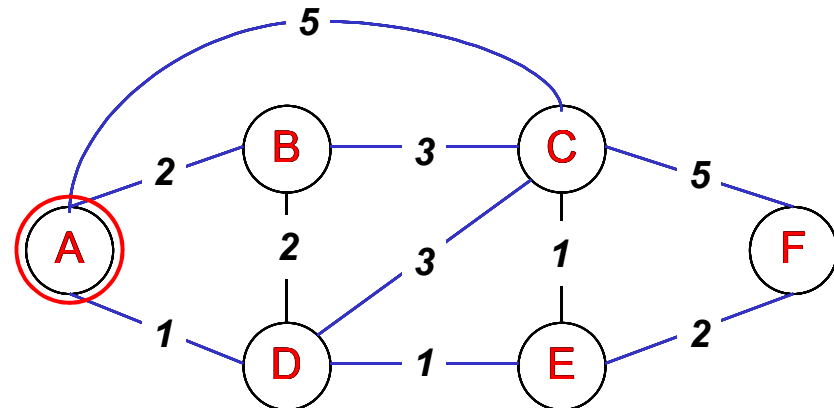
```
1      Initialization /* for source node u */
2       $N' = \{u\}$ 
3      for all nodes  $v$ 
4          if  $v$  adjacent to  $u$ 
5              then  $D(v) = c(u,v)$ 
6          else     $D(v) = \infty$ 
7      Loop
8          find  $w$  not in  $N'$  such that  $D(w)$  is a minimum
9          add  $w$  to  $N'$ 
10         update  $D(v)$  for all  $v$  adjacent to  $w$  and not in  $N'$ :
11          $D(v) = \min(D(v), D(w) + c(w,v))$ 
12         /* new cost to  $v$  is either old cost to  $v$  or
13            known shortest path cost to  $w$  + cost from  $w$  to  $v$  */
14     until all nodes  $n \in N'$ 
```



4.2.1 LSA, Link-State Algorithmus

- *Initialisierungsschritt*

- Annahme: *Der Quellknoten sei A*
- $D(v)$ wird initialisiert
 - Die Kosten momentan bekannter Pfade mit geringsten Kosten von A zu direkten Nachbarn C,B,D werden auf 5,2,1 initialisiert
- Es wird $N'=\{A\}$ gesetzt
- Es ist
 - $D(B) = c(A,B) = 2$
 - $D(C) = c(A,C) = 5$
 - $D(D) = c(A,D) = 1$
 - $D(E) = c(A,E) = \infty$
 - $D(F) = c(A,F) = \infty$

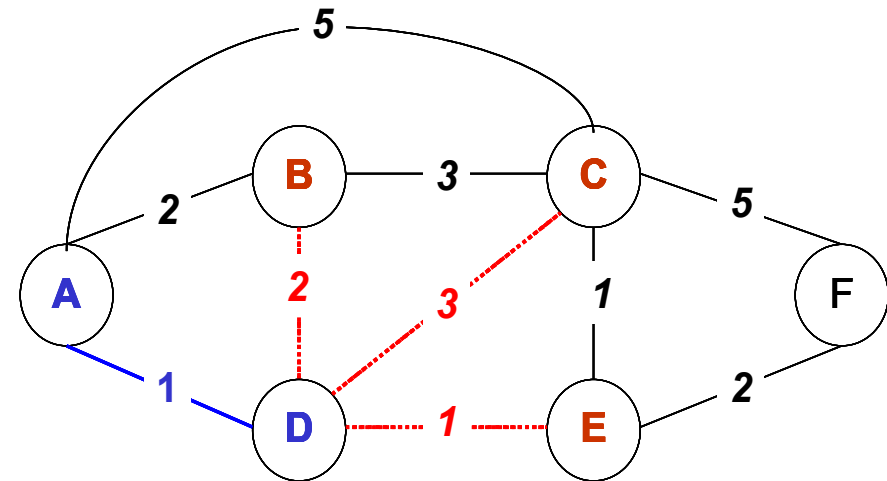


4.2.1 LSA, Link-State Algorithmus

• 1. Iterationsschritt

- Suche nach jenen Knoten, die noch nicht zur Menge N' gehören
 (→ also **B,D,C,E,F**)
- Ermittle daraus Knoten mit geringsten Kosten bei vorheriger Iteration
 (→ also **D** mit $D(D)=1$)
- Dieser Knoten wird zur Menge N hinzugefügt:
 (→ also $N'=\{A,D\}$)
- $D(v)$ wird jetzt für alle Nachbarn von D (die nicht zu N' gehören) nach folgender Vorschrift aktualisiert:

$$D(v) = \min (D(v), D(w) + c(w,v))$$
 - w ist oben ermittelter Knoten
 (→ also $w=D$)
 - Dabei wird Vorgängerknoten $p(v)$ mit geringsten Kosten zum Ziel v gespeichert!
 - Es werden **alle direkten Nachbarn** von D betrachtet



4.2.1 LSA, Link-State Algorithmus

- *Ergebnis des 1. Iterationsschrittes*
 - Für $D(v)$ ergibt sich:
 $D(B) = \min(2, D(D) + c(D,B)) = \min(2,3) = 2$
 $D(C) = \min(5, D(D) + c(D,C)) = \min(5,4) = 4$
 $D(E) = \min(\infty, D(D) + c(D,E)) = \min(\infty,2) = 2$
 - In Knoten A wird folgende Tabelle aufgebaut:

Iter.- Schritt	Menge N'	$D(B), p(B)$	$D(C), p(C)$	$D(D), p(D)$	$D(E), p(E)$	$D(F), p(F)$
0	A	2,A	5,A	1,A	$\infty, ?$	$\infty, ?$
1	A,D	2,A	4,D	----	2,D	$\infty, ?$
2

4.2.1 LSA, Link-State Algorithmus

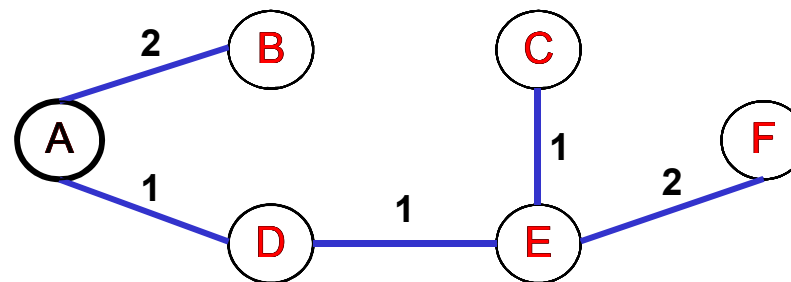
- *Endergebnis beim 5. Iterationsschritt für Quelle A*

<i>Iter.- Schritt</i>	<i>Menge N'</i>	<i>D(B), p(B)</i>	<i>D(C), p(C)</i>	<i>D(D), p(D)</i>	<i>D(E), p(E)</i>	<i>D(F), p(F)</i>
0	A	2,A	5,A	1,A	$\infty, ?$	$\infty, ?$
1	A,D	2,A	4,D	----	2,D	$\infty, ?$
2						
3						
4						
...						

- *Wenn LSA endet,*
 - *ist für jeden Knoten sein Vorläufer auf dem Pfad mit den geringsten Kosten bekannt*
 - *Für jeden Vorläufer ist dessen Vorläufer bekannt usw.*

4.2.1 LSA, Link-State Algorithmus

- *Endergebnis für den Quellrechner A*
 - *“Least-cost path”- Baum*



- Weiterleitungstabelle des Rechners A:

destination	link
B	(A,B)
D	(A,D)
E	(A,D)
C	(A,D)
F	(A,D)

4.2.1 LSA, Link-State Algorithmus

- *Zuverlässiges Fluten*

- ▶ Für LSA ist vorab die *vollständige Information* über das Gesamtnetz nötig
 - Diese wird durch „zuverlässiges Fluten“ ermittelt
- ▶ Prinzip
 - Jeder Knoten sendet LS (Link State)-Pakete zu direkten Nachbarn
 - Diese übertragen die LS-Pakete wiederum an ihre nächsten Nachbarn, nur nicht über die Leitung, wo die Information her kam
- ▶ LS-Pakete enthalten
 1. Die Kennung des Knotens, der LS-Paket erzeugt hat
 2. Die Liste der direkten Nachbarn inkl. zugehöriger Verbindungskosten
 3. Eine Sequenznummer
 4. Eine Lebensdauer für das LS-Paket

4.2.1 LSA, Link-State Algorithmus

- *Beispiel-Ablauf: Zuverlässiges Fluten*

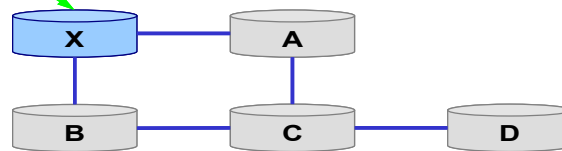
a) Ein LS-Paket (LSP) kommt bei Knoten X an

b) X flutet das LSP nach A und B

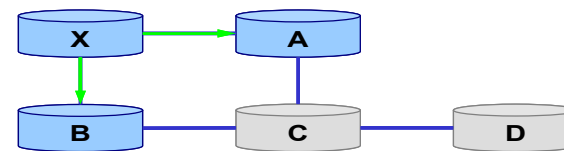
c) A und B fluten das LSP zu C, aber nicht mehr zu X

d) C flutet das LSP zu D; *Wie wird das Fluten nach B und A verhindert?*

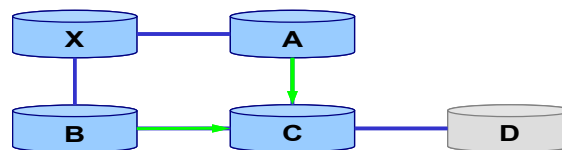
LS-Paket



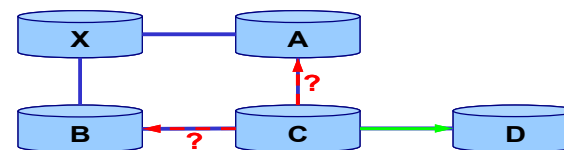
(a)



(b)



(c)



(d)

4.2.2 DVA, Distance-Vector Algorithmus

- *Ein Distance-Vector Algorithmus: Bellman-Ford-Algorithmus*

- ▶ *Verteilter Routing-Algorithmus*

[Ref 1] Kapitel 4, Seite 412-419

[Ref 2] Kapitel 5, Seite 428-430

- *Jeder Router kennt zu Beginn nur die Kosten der Verbindungsleitungen zu direkten Nachbarn*
 - *Einer ausgefallenen Verbindung werden unendlich hohe Kosten zugeordnet*
 - *Jeder Router speichert einen Distanz-Vektor, der die momentan bekannten, besten Pfade zu allen übrigen Routern enthält*
 - *Der Distanz-Vektor wird nur zwischen direkten Nachbarn ausgetauscht*
 - *Aufgrund ausgetauschter Informationen aktualisieren Router ihre Weiterleitungstabellen*

- ▶ *Kein Knoten hat die globale Kenntnis der Verbindungen und Kosten im Gesamtnetzwerk*

- *Prinzip: „Gerüchteverbreitung“*

4.2.2 DVA, Distance-Vector Algorithmus

- *Bellmann-Ford-Gleichung*

‣ $d_X(Y)$ bezeichne die Kosten auf dem Least-Cost-Pfad von X nach Y

Dann gilt: $d_X(Y) = \min_V \{ c(X,V) + d_V(Y) \}$

– Das Minimum wird über alle *direkten Nachbarn* V vom Knoten X bestimmt

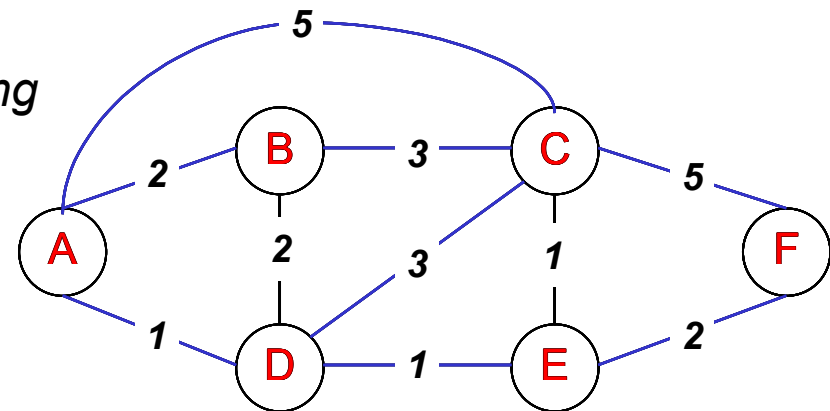
‣ *Beispiel:*

– Aus der Zeichnung folgt:

$$d_B(F)=5, d_D(F)=3, d_C(F)=3$$

– Aus der Bellmann-Ford-Gleichung folgt:

$$\begin{aligned} d_A(F) &= \min \{ c(A,B) + d_B(F), \\ &\quad c(A,D) + d_D(F), \\ &\quad c(A,C) + d_C(F) \} \\ &= \min \{ 2+5, 1+3, 5+3 \} \\ &= 4 \end{aligned}$$



4.2.2 DVA, Distance-Vector Algorithmus

- *Definitionen*

- $N = \{X, Y, \dots\}$ sei die Menge aller Knoten im betrachteten Netz
- $D_X(Y)$ sind die *momentan bekannten* geringsten Kosten von X nach Y
- Der *Distanz-Vektor* $D_X = [D_X(Y): Y \in N]$ wird von jedem Knoten X beim Empfang von Distanz-Vektoren aktualisiert
- Für jeden Nachbarn V speichert X auch den Vektor $D_V = [D_V(Y): Y \in N]$ (aktueller Distanz-Vektor, den X vom Nachbarn V erhalten hat)

- *Methode*

- Jeder Knoten sendet seinen Distanz-Vektor an direkte Nachbarn
- Wenn ein Knoten X einen Distanz-Vektor D_V vom Knoten V erhält, wird sein eigener Distanz-Vektor D_X gemäß der B-F-Gleichung aktualisiert:

$$D_X(Y) = \min_v \{ c(X, V) + D_V(Y) \} \text{ für jeden Knoten } Y \in N$$

↑ Minimum über alle direkten Nachbarn v

4.2.2 DVA, Distance-Vector Algorithmus

- DVA (1)

At each node, x :

```
1      Initialization
2      for all destinations  $y$  in  $N$ :
3           $D_x(y) = \infty$       if  $y$  is not a neighbor
4           $D_x(y) = c(x,y)$     if  $y$  is a neighbor
5      for each neighbor  $w$ 
6           $D_w(y) = \infty$  for all destinations  $y$  in  $N$ 
7      for each neighbor  $w$ 
8          send distance vector  $D_x = [D_x(y): y \text{ in } N]$  to  $w$ 
```

4.2.2 DVA, Distance-Vector Algorithmus

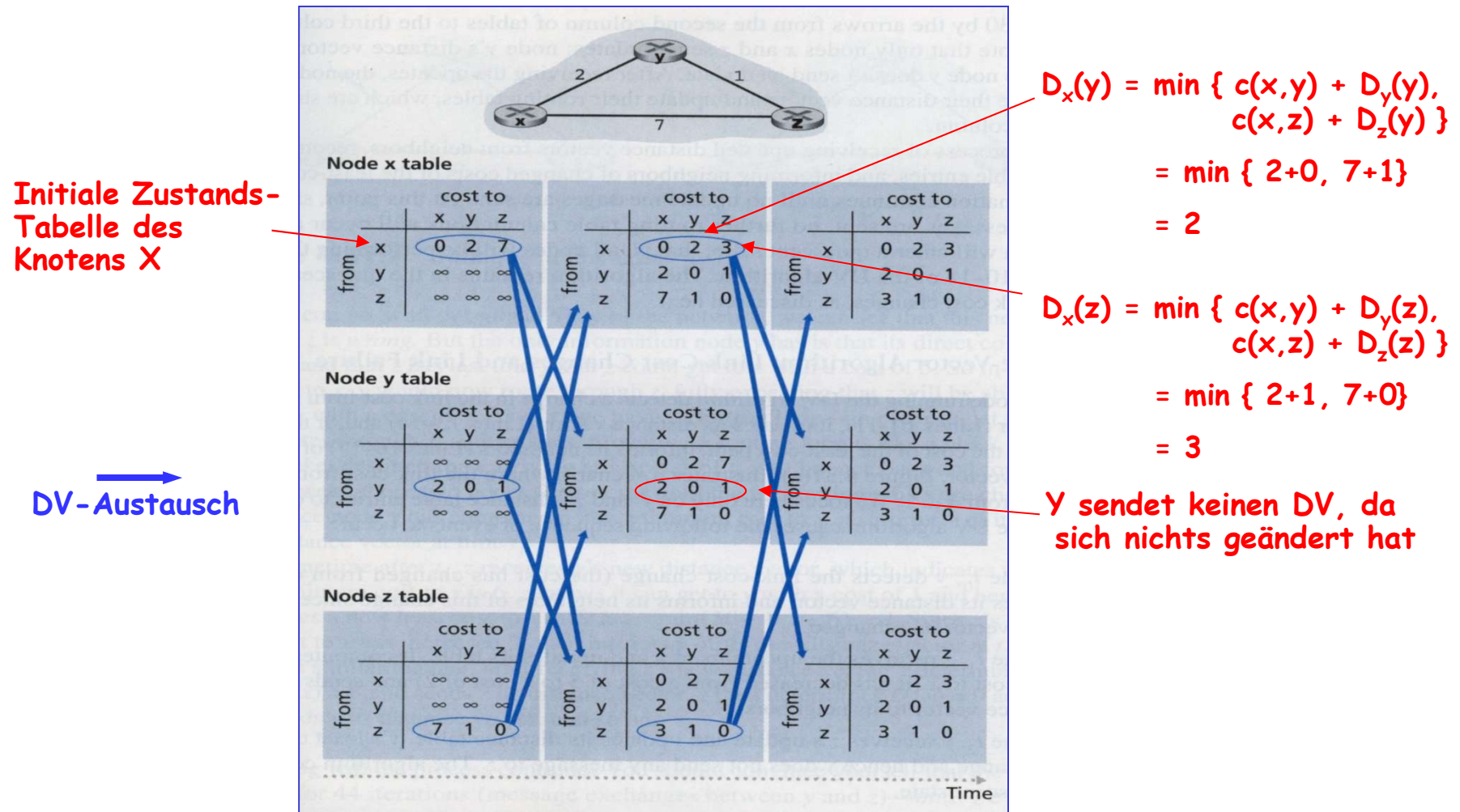
- DVA (2)

At each node, x :

```
9  loop
10      wait (until I see a link cost change to some neighbor w
11           or I receive a distance vector from some neighbor w)
12
13      for each y in N:
14           $D_x(y) = \min_v \{ c(x,v) + D_v(y) \}$ 
15
16      if  $D_x(y)$  changed for any destination y
17          send dist. vector  $D_x = [D_x(y): y \text{ in } N]$  to all neighbors
18
19  forever
```


4.2.2 DVA, Distance-Vector Algorithmus

- Ablaufbeispiel



4.2.2 DVA, Distance-Vector Algorithmus

- *Wann werden Nachrichten ausgetauscht?*
 - *Getriggerte Aktualisierung*
 - *Nachrichten werden immer dann gesendet, wenn sich der lokale Distanz-Vektor ändert*
 - *Periodische Nachrichtenmeldungen*
 - *Knoten senden periodisch Nachrichten an Nachbarn, auch wenn sich nichts geändert hat*
 - *Dadurch wird auch signalisiert: Sendeknoten ist aktiv!*
 - *Stellt sicher, dass ein Knoten gültige Info erhält, wenn er nach einem Ausfall seine Weiterleitungstabelle neu aufbauen muss!*
 - *Typische Wiederholrate: mehrere Minuten*

4.2.3 Hierarchisches Routing

- *Problemstellungen*

[Ref 1] Kapitel 4, Seite 421-424

[Ref 2] Kapitel 5, Seite 437-438

- *Skalierung:*

- *Mit wachsender Anzahl von Routern steigen Aufwände für Berechnung, Speicherung und Übermittlung von Routing-Informationen*
 - *Das Internet beinhaltet Millionen von Routern*
 - *Die bisher besprochenen Routing-Algorithmen würden nie konvergieren*

- *Administrative Autonomie:*

- *Eine Organisation sollte in der Lage sein, ihr eigenes Netz nach eigenem Ermessen zu verwalten und zu betreiben*
 - *Trotzdem muss es möglich sein, diese „autonomen“ Teilnetze wieder optimal miteinander zu verbinden*

- *Lösung:*

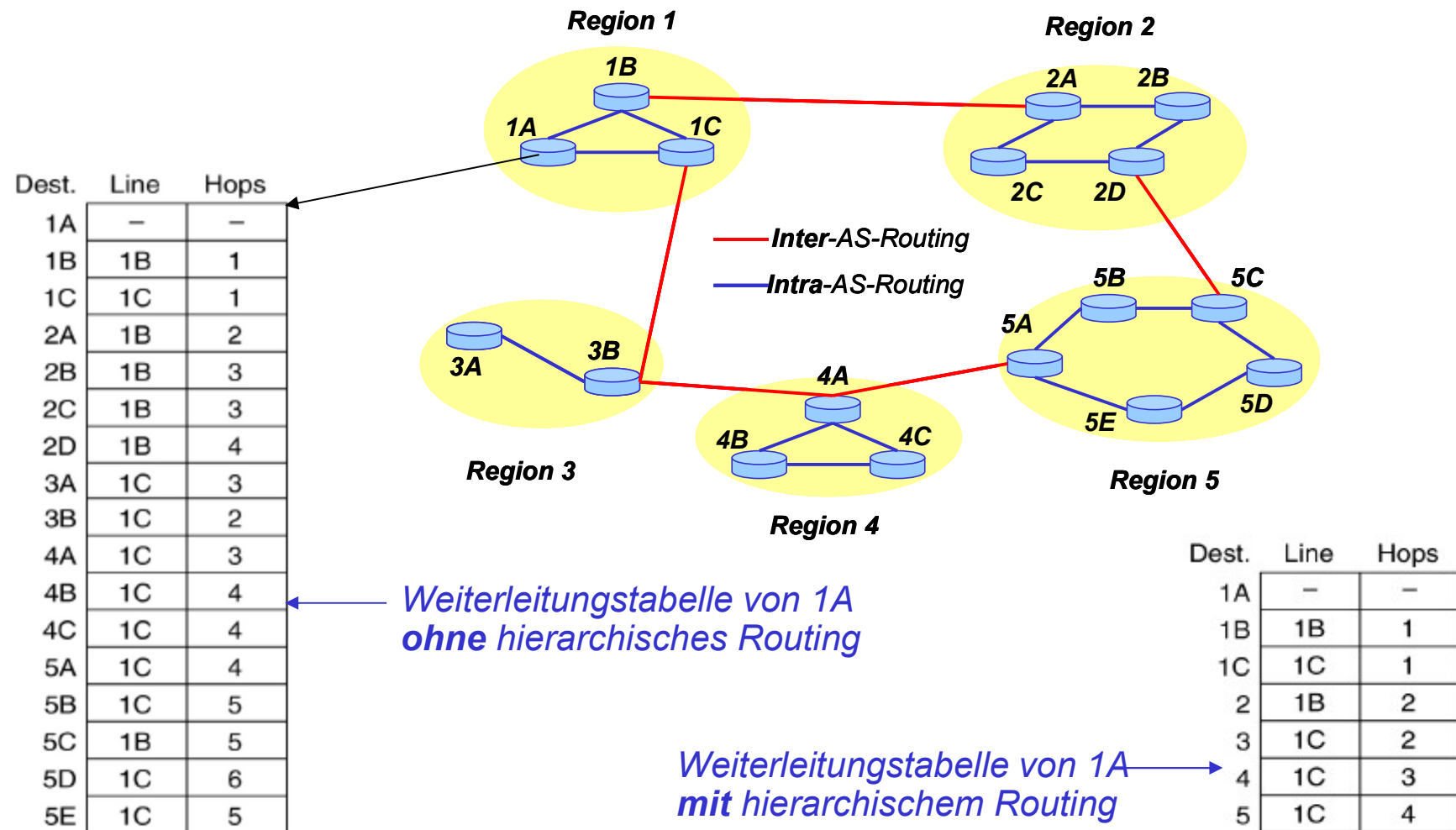
- *Hierarchisches Routing*

4.2.3 Hierarchisches Routing

- *Autonome Systeme (AS)*
 - *Router, die zu einer administrativen Gruppe gehören, werden in einem **Autonomen System (AS)** zusammengefasst*
 - *Router innerhalb eines AS verwenden den gleichen Routing-Algorithmus (z.B. LS- oder DV-Algorithmus); Das im AS verwendete Routing-Protokoll wird **Intra-AS-Routing-Protokoll** genannt*
 - *Mehrere AS können miteinander durch Gateway-Router (GR) verbunden werden*
 - *GR sind ausgewählte Router eines AS, die für das Weiterleiten von Paketen außerhalb der AS zuständig sind*
 - *Routen zu Zielen außerhalb eines AS sind nur den GR bekannt*
 - *Die Gateway-Router der verschiedenen AS kommunizieren über das **Inter-AS-Routing-Protokoll***

4.2.3 Hierarchisches Routing

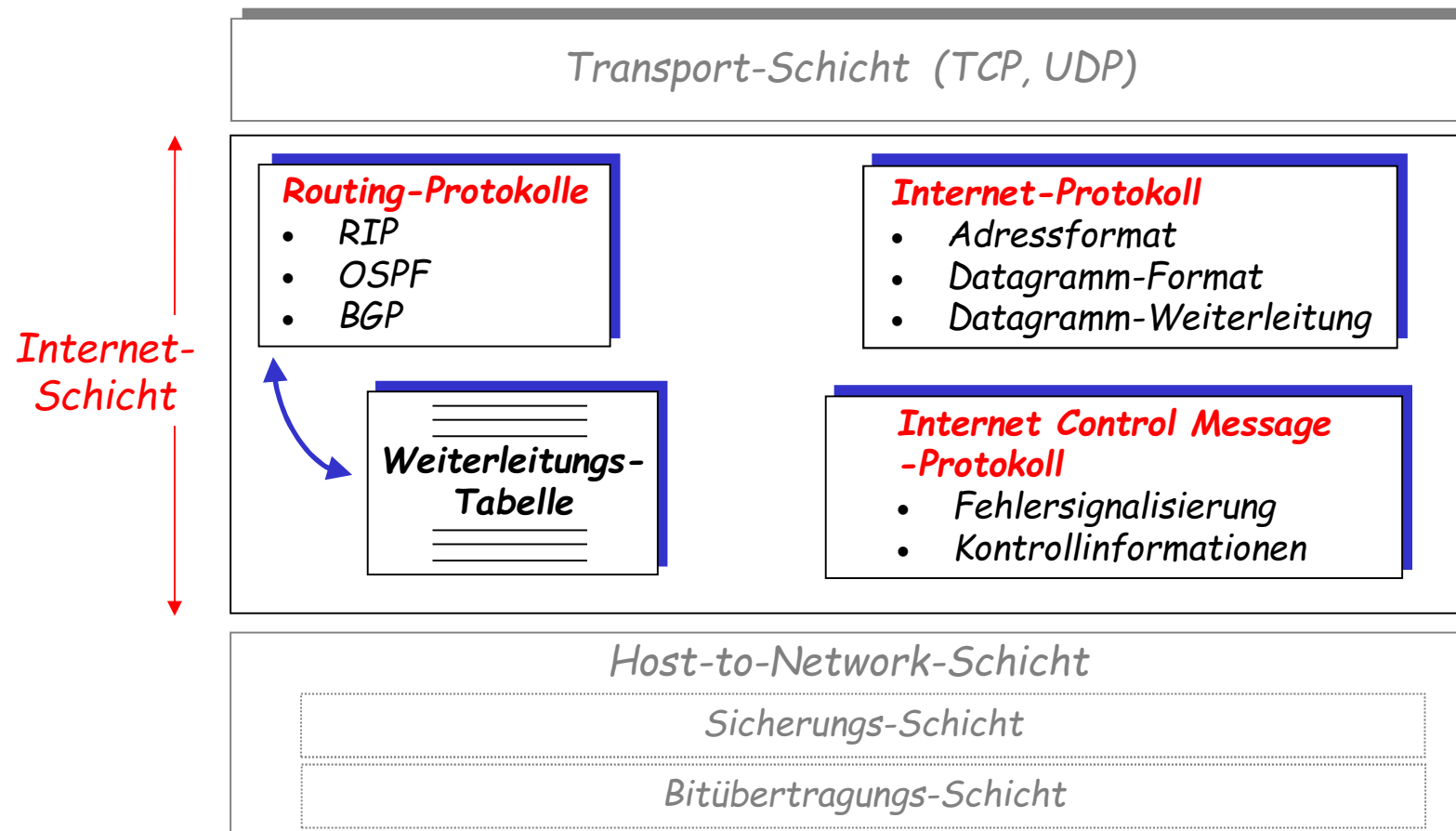
- *Beispiel: Hierarchisches Routing*



4.3 Internet Protocol (IP)

- Die Vermittlungsschicht im Internet

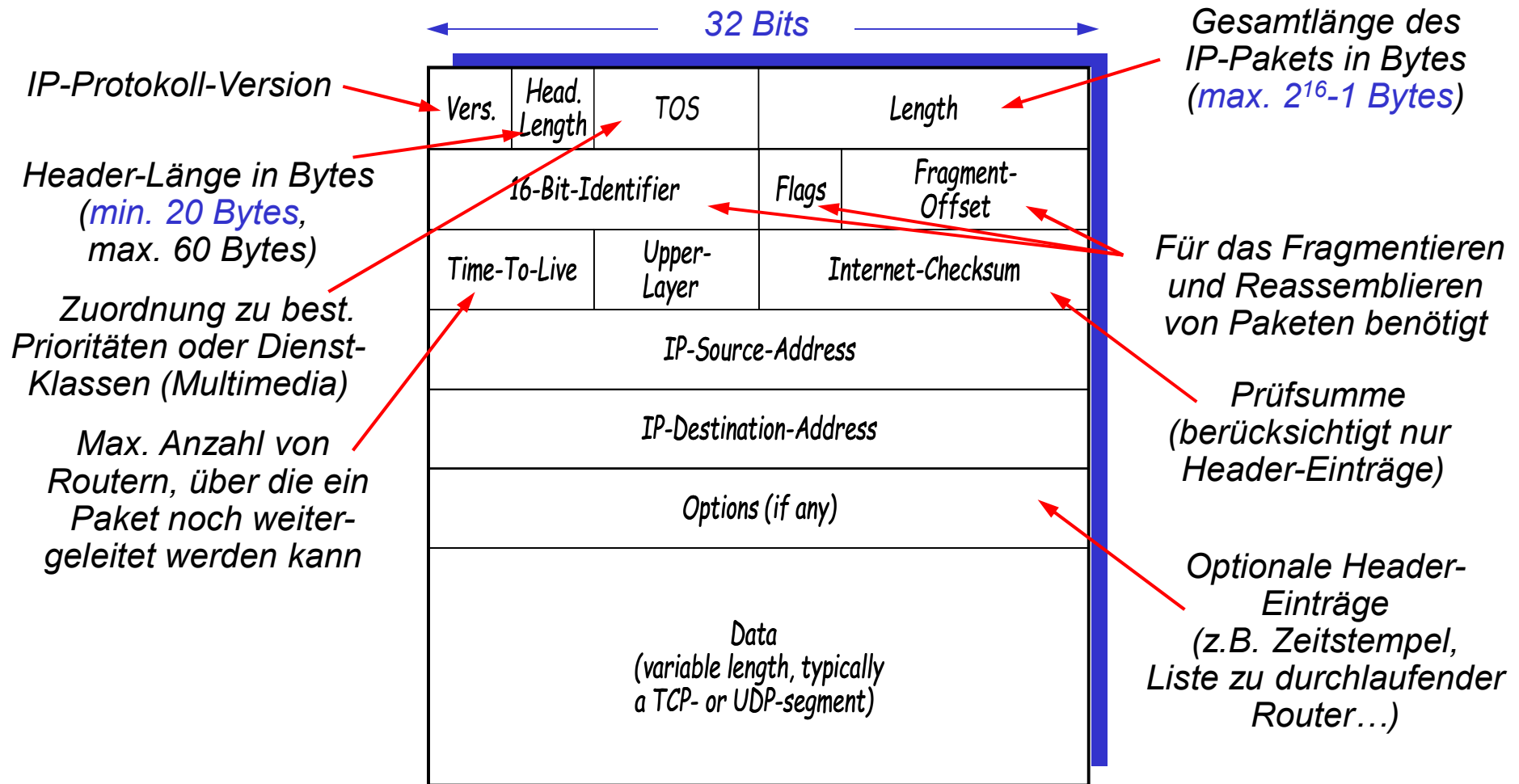
[Ref 1] Kapitel 4, Seite 371-372



4.3.1 Internet Protocol Version 4 (IPv4)

- IPv4-Paket-Format

[Ref 1] Kapitel 4, Seite 372-375



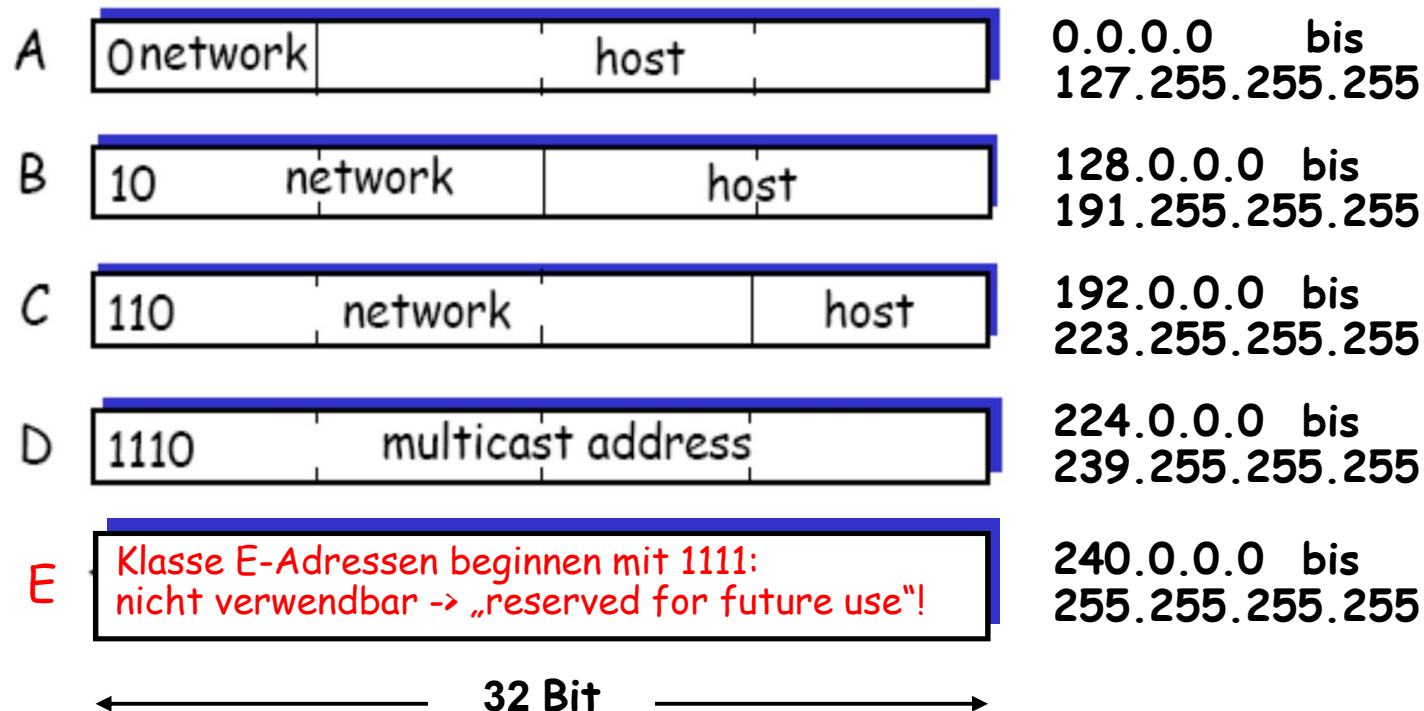
4.3.1 Internet Protocol Version 4 (IPv4)

- *IP-Adressen und Netzwerk-Klassen*

[Ref 1] Kapitel 4, Seite 378-382

[Ref 2] Kapitel 5, Seite 507-516

- ▶ *IP-Adressen wurden in Klassen aufgeteilt*
- ▶ *Die Klassenzugehörigkeit einer Internet-Adresse **hängt von dem Wert der ersten 4 Bits ab.***



4.3.1 Internet Protocol Version 4 (IPv4)

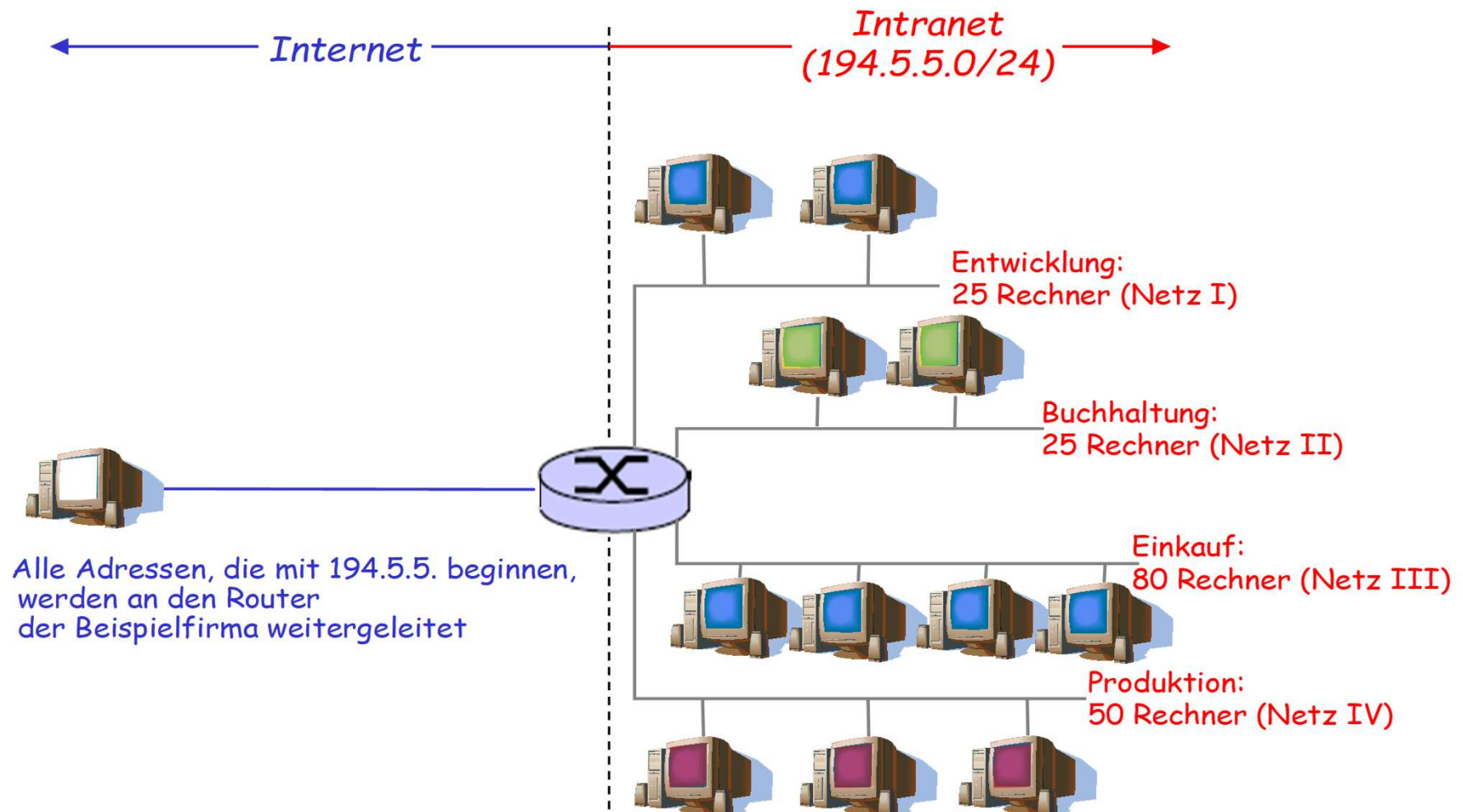
- *IP-Subnetting*

- ▶ *Eine Beispiel-Firma hat das Klasse-C-Netz „194.5.5“ zugeordnet bekommen*
 - *Intern ist das Netz in die logischen Netze „Buchhaltung“, „Entwicklung“, „Verkauf“ usw. strukturiert*
 - *Die Kommunikation zwischen logischen Netzen soll geregelt werden*
 - *Diese Subnetzbildung ist im Internet nicht sichtbar*
- ▶ *Problem*
 - *Einteilung in Netzklassen zu starr und unflexibel*
- ▶ *Lösung*
 - *Grenze zwischen Bits für Netz- und Host-Anteil auf Kosten des Host-Anteils nach rechts verschieben*
- ▶ *Folgeproblem*
 - *Woher kennt ein Rechner jetzt die Länge von Netz- und Host-Anteil einer IP-Adresse?*

4.3.1 Internet Protocol Version 4 (IPv4)

- IP-Subnetting ...

<http://www.netplanet.org/adressierung/subnetting.shtml>
Applet: <http://www.subnet-calculator.com/>



4.3.1 Internet Protocol Version 4 (IPv4)

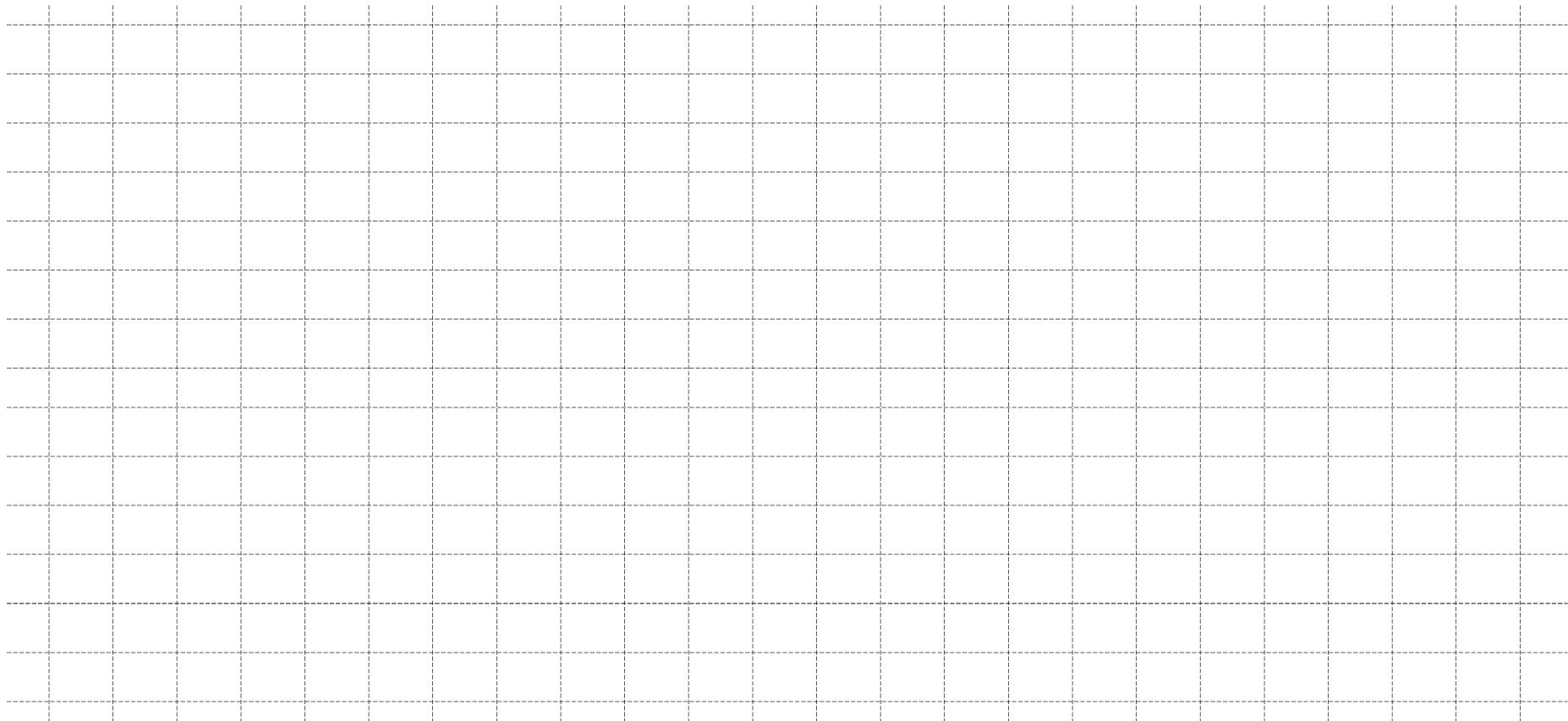
- *Netzmasken*

- *Eingeführt, damit ein Rechner erkennen kann, wie lang der Netzanteil seiner IP-Adresse ist*
 - *Eine 32 Bit lange Netzmaske wird durch bitweises AND mit der IP-Adresse verknüpft*
- *Ist ein Bit der Netzmaske gesetzt:
entsprechendes Bit der IP-Adresse gehört zu Netzadresse*
- *Ist ein Bit der Netzmaske nicht gesetzt:
entsprechendes Bit der IP-Adresse gehört zu Host-Adresse*

4.3.1 Internet Protocol Version 4 (IPv4)

- *Subnetz-Bildung im Detail*

- ▶ *Klasse C Netz: 192.5.5.X (X=0,1,...,255): 254 nutzbare Adressen*
 - *Reservierte Adressen: alle Host-Bits=1 → ger. Broadcast: 192.5.5.255*
alle Host-Bits=0 → Netzadresse: 192.5.5.0



4.3.1 Internet Protocol Version 4 (IPv4)

- *Ableitung der Subnetz-Adresse aus einer IP-Adresse*
 - *Beispiel für Klasse-C-Netz:*
192.147.1.0 bzw. „11000000.10010011.00000001.00000000“
 - Das Netz wird in 4 Subnetze aufgeteilt
 - $4 = 2^2 \rightarrow$ Es werden 2 zusätzliche „Netz-Bits“ gebraucht
 - Die Netzmaske ist damit:
255.255.255.192 bzw. „11111111.11111111.11111111.11000000“
 - *In welchem Teilnetz liegt die IP-Adresse 192.147.1.129?*

IP-Adresse:	11000000.10010011.00000001.10000001	} Bitw. AND
Netzmaske:	11111111.11111111.11111111.11000000	
⇒ Subnetz	11000000.10010011.00000001.10000000	(=192.147.1.128)

- *Antwort: Im 3. Subnetz!*

4.3.1 Internet Protocol Version 4 (IPv4)

Reservierte IP-Adressen und IP-Adressen mit besonderer Bedeutung

- *Netzadresse*

- *Netzadressen dienen in Verbindung mit Netzmasken ausschließlich zu Routing-Zwecken*
 - *Sie werden von Routern/Rechnern in Weiterleitungstabellen benötigt*
- *Die Netzadresse wird gebildet*
 - *in dem alle Host-Bits auf 0 gesetzt werden und*
 - *der Adress-Präfix voran gestellt wird*

- *Gerichtete Broadcast-Adresse*

- *Ein IP-Paket mit der gerichteten Broadcast-Adresse als Ziel reist so lange als Einzelpaket durchs Internet, bis das Zielnetz erreicht wird.*
 - *Erst im Zielnetz wird das Paket von allen Endgeräten gelesen*
- *Die gerichtete Broadcast-Adresse wird gebildet*
 - *in dem alle Host-Bits auf 1 gesetzt werden und*
 - *der Adress-Präfix voran gestellt wird*

4.3.1 Internet Protocol Version 4 (IPv4)

- *Beispiele: Netzadressen und gerichtete Broadcast-Adressen*

<i>Präfix</i>	<i>Klasse</i>	<i>Netzmaske</i>	<i>Netzadresse</i>	<i>Ger. BC-Adresse</i>
194.95.60	C	255.255.255.0	194.95.60.0	194.95.60.255
129.247.0	B	255.255.192.0	129.247.0.0	129.247.63.255
129.247.64	B	„	129.247.64.0	129.247.127.255
129.247.128	B	„	129.247.128.0	129.247.191.255
129.247.192	B	„	129.247.192.0	129.247.255.255

4.3.1 Internet Protocol Version 4 (IPv4)

- *Lokale Broadcast-Adresse*

- Besteht nur aus 1-Bits: **255.255.255.255**
- *Gebraucht für Versendung eines Broadcast-Pakets, wenn eigene IP-Adresse des Rechners noch nicht bekannt ist (z.B. benötigt in der Startphase eines Rechners)*

- *„This Computer“-Adresse*

- Besteht nur aus 0-Bits: **0.0.0.0**
- *Gebraucht als Quell-Adresse für Rechner, die ihre eigene IP-Adresse noch nicht kennen*

- *„Loopback“-Adresse*

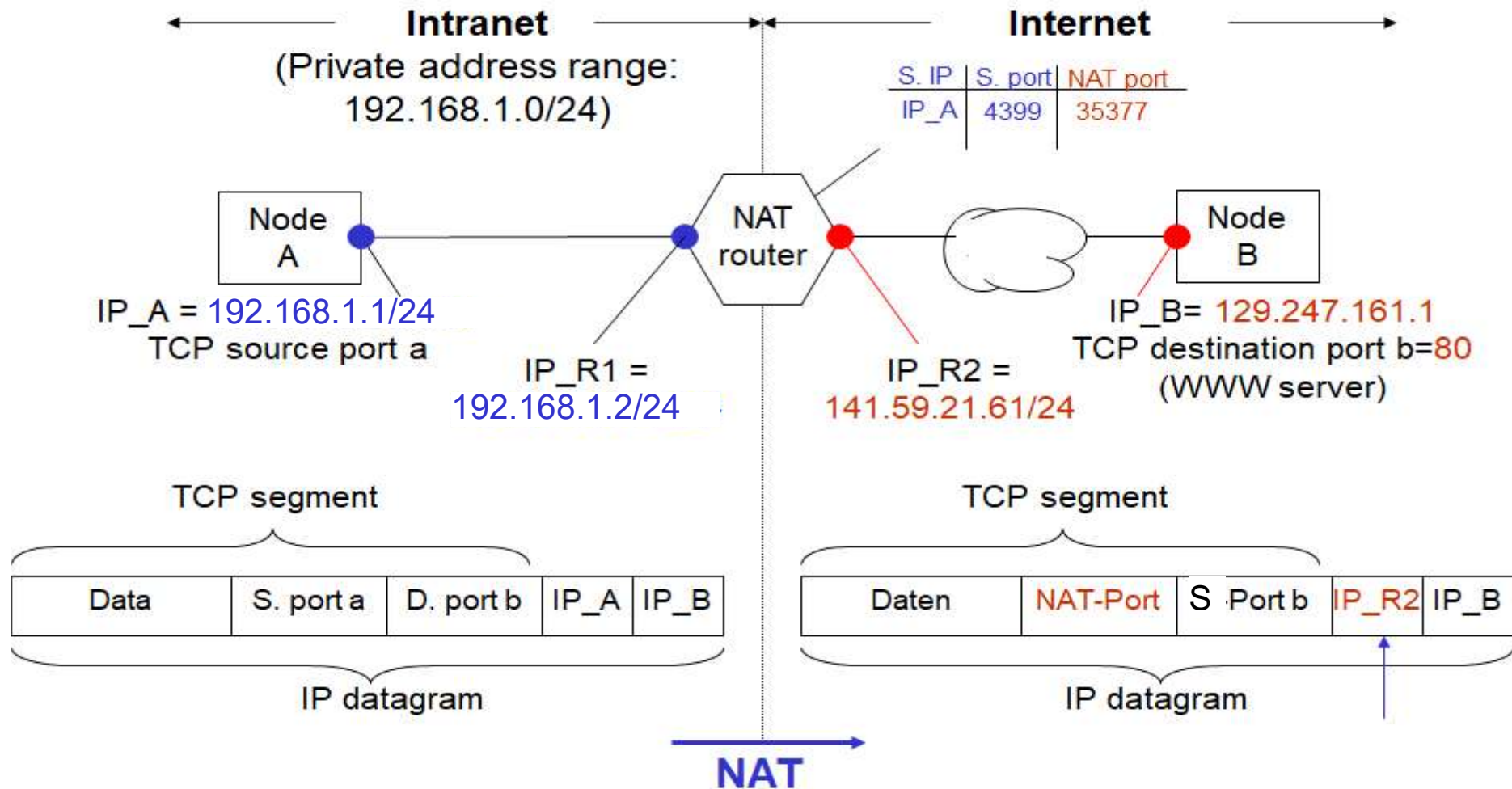
- *Auch ohne phys. Netzwerk-Interfaces sind in Rechnern interne Netzschleifen aktiv.*
- *IP-Standard reserviert dazu Netzpräfix 127 der Klasse A. Konvention ist die Verwendung der Schleifenadr.: **127.0.0.1** (→ 1. Intf, 127.0.0.2 → 2. Intf....)*

4.3.1 Internet Protocol Version 4 (IPv4)

- *Private Adressbereiche*
 - *IP-Pakete mit IP-Adressen aus den privaten Adressbereichen*
 - *werden nicht in das Internet weitergeleitet!*
 - *dürfen nur innerhalb eines privaten Netzes genutzt werden*
 - *Vorteil*
 - *Jede Einrichtung kann intern private Adressbereiche nutzen, die Kommunikation zur Rechnern im Internet ist damit jedoch nicht möglich*
 - *Festgelegte private Adressbereiche sind:*
 - *Klasse A: 10.0.0.0 – 10.255.255.255*
 - *Klasse B: 172.16.0.0 – 172.31.255.255*
 - *Klasse C: 192.168.0.0 – 192.168.255.255*

4.3.1 Internet Protocol Version 4 (IPv4)

- Einschub: Network Address Translation (NAT)*

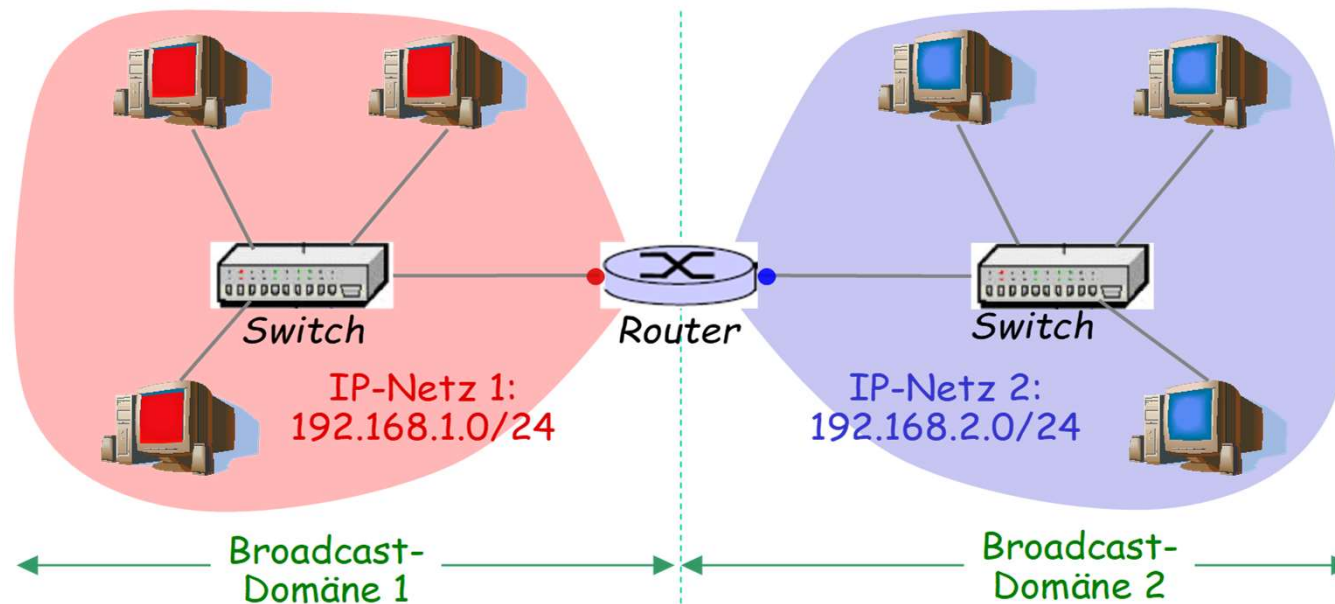


4.3.1 Internet Protocol Version 4 (IPv4)

- *Klassenlose IP-Adressierung*
 - ▶ *Bisher: Klassenbasierte IP-Adressierung + „Subnetting“-Ansatz*
 - *In IP-Netzen der Klasse A/B/C werden Subnetz-Kennungen definiert*
 - *Erweitertes Netzwerkpräfix und Einführung von Subnetzmasken*
 - *Innerhalb eines Netzes nur gleich große Subnetze möglich*
 - ▶ *Jetzt: Klassenlose IP-Adressierung*
 - *Verallgemeinerung: Grenze zwischen Netzwerk- und Host-Anteil wird variabel definierbar, unabhängig von der Netzklasse*
 - *Verwendung klassenloser IP-Adressierung im Intranet:*
Variable Length Subnet Masks (VLSM)
 - *Verwendung klassenloser IP-Adressierung im Internet:*
Classless Interdomain Routing (CIDR)
→ früher „Supernetting“
 - ▶ *Notation einer CIDR-Netzadresse*
 - *a.b.c.d/x , wobei x die Anzahl der führenden Bits angibt, die den Netzanteil darstellen*

4.3.1 Internet Protocol Version 4 (IPv4)

- *Zuordnung und Kopplung von IP-Netzen*
 - *Grundregeln*
 - *Rechner in unterschiedlichen IP-(Sub)Netzen können nur über Router hinweg kommunizieren*
 - *In verbindungslosen LANs ist ein IP-(Sub)Netz deckungsgleich mit (oder eine Untermenge von) einer Broadcast-Domäne*
 - *Beispiel: Kopplung zweier Ethernet-LANs*

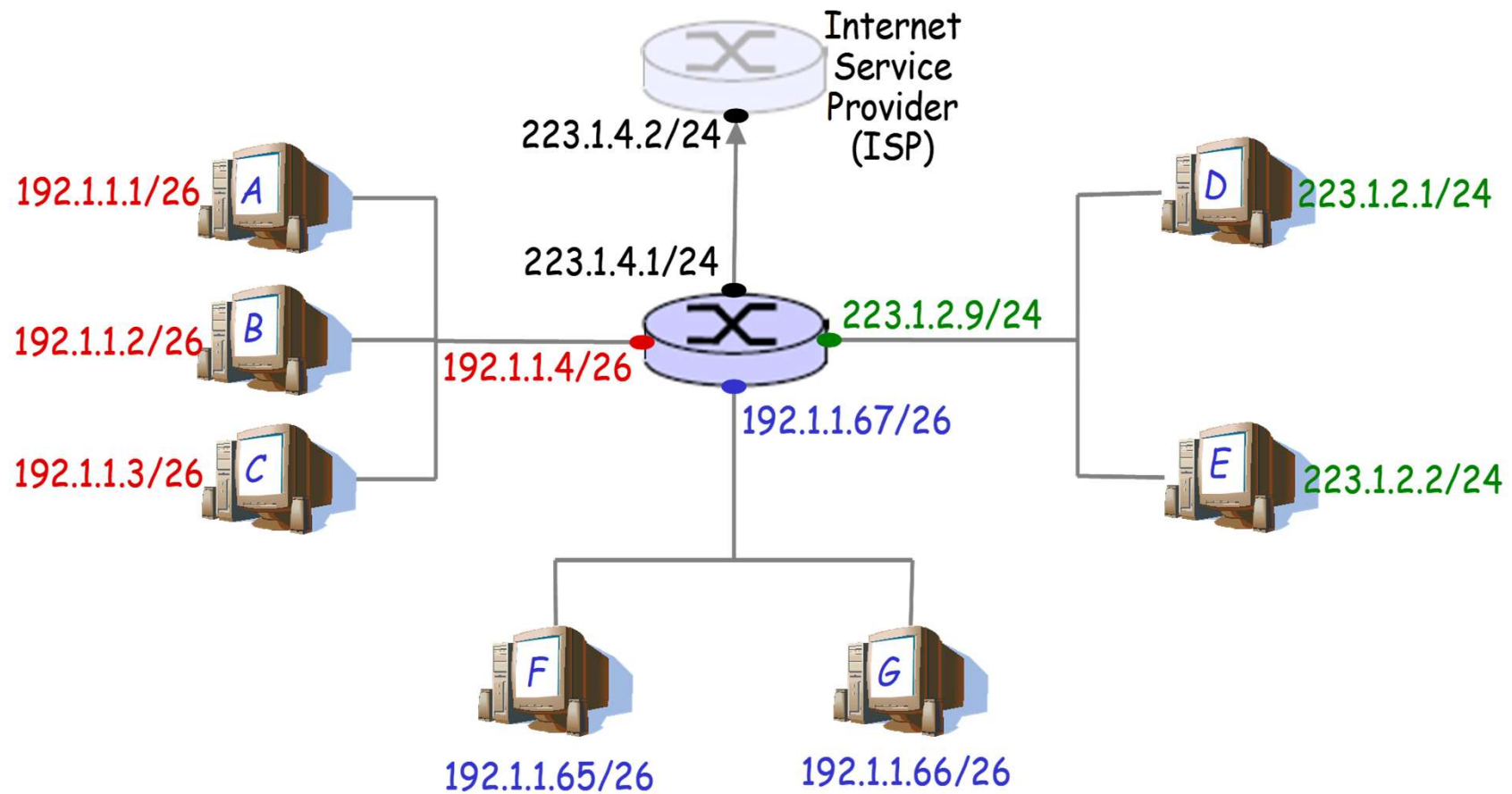


4.3.1 Internet Protocol Version 4 (IPv4)

- *Struktur von Weiterleitungstabellen*
 - ▶ *Zielnetz*: repräsentiert Ziel des Routen-Eintrags;
kann ein IP-Netz, IP-Subnetz (→ „Netz/Subnetz-Route“) oder ein einzelner Host sein (→ „Host-Route“)
 - ▶ *Netzwerk-Maske*: Netzmaske bzw. Länge der Netzmaske bei Verwendung von CIDR/VLSM
Bei Host-Routen immer: 255.255.255.255
 - ▶ *Gateway*: IP-Adresse des nächsten Routers auf dem Weg zum Netzziel
 - ▶ *Schnittstelle*: Angabe der Router-Schnittstelle (entweder logische Bezeichnung oder IP-Adresse der Schnittstelle)
 - ▶ *Metrik*: Enthält die „Kosten“ einer Route;
Dient zu „Bewertung“ von Routen-Einträgen, die zum gleichen Ziel führen

4.3.1 Internet Protocol Version 4 (IPv4)

- *Beispiel-Netz zu Weiterleitungstabellen*



4.3.1 Internet Protocol Version 4 (IPv4)

- *Weiterleitungstabellen zum Beispielnetz auf Seite 43*

- *Weiterleitungstabelle von Rechner A*

<i>Zielnetz</i>	<i>Netzmaske</i>	<i>Gateway</i>	<i>Schnittstelle</i>	<i>Metrik</i>	
192.1.1.0	255.255.255.192	-	192.1.1.1	1	(lokales Netz)
0.0.0.0	0.0.0.0	192.1.1.4	192.1.1.1	2	(Default-Route)

- *Weiterleitungstabelle des Routers*

<i>Zielnetz</i>	<i>Netzmaske</i>	<i>Gateway</i>	<i>Schnittstelle</i>	<i>Metrik</i>	
192.1.1.0	255.255.255.192	-	192.1.1.4	1	
192.1.1.64	255.255.255.192	-	192.1.1.67	1	
223.1.2.0	255.255.255.0	-	223.1.2.9	1	
223.1.4.0	255.255.255.0	-	223.1.4.1	1	
0.0.0.0	0.0.0.0	223.1.4.2	223.1.4.1	2	

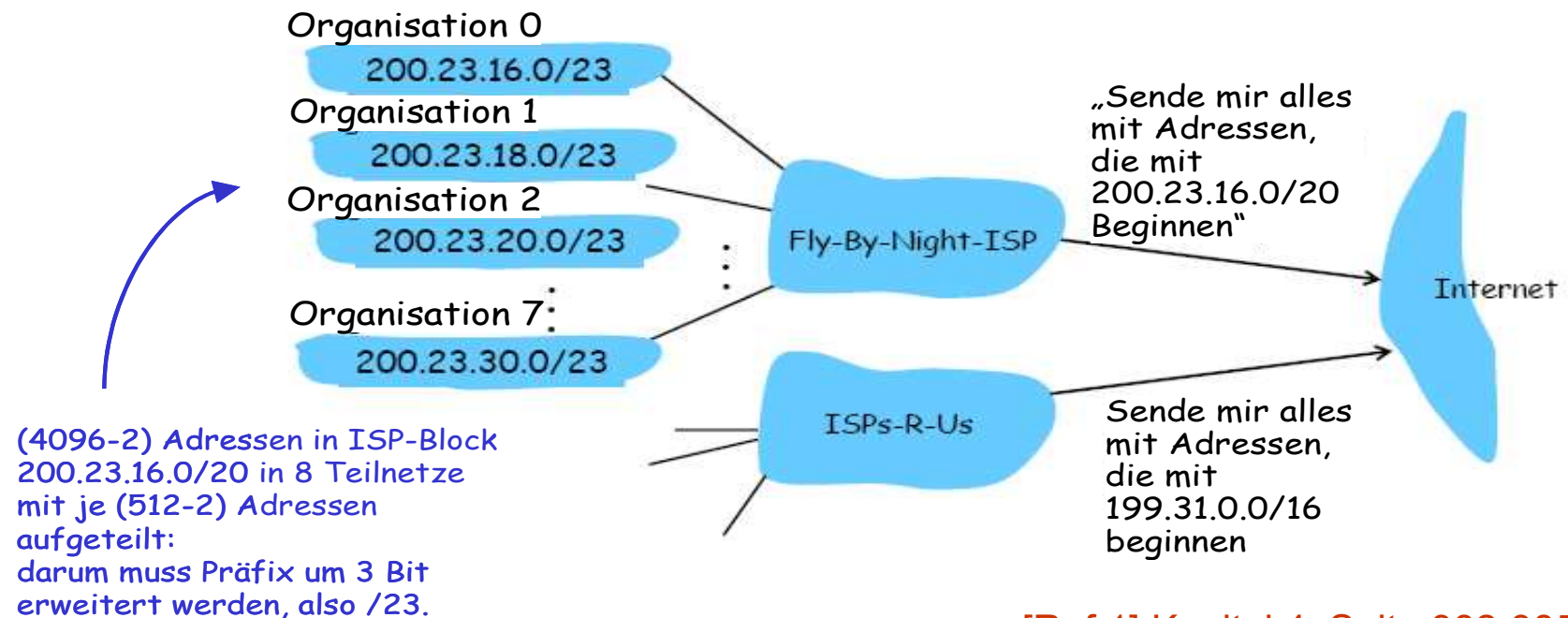
4.3.1 Internet Protocol Version 4 (IPv4)

- *Bestimmung des besten Routen-Eintrags*
 1. *Für jede Zeile in Routing-Tabelle wird Bitweises_AND zwischen Ziel-IP-Adresse des IP-Pakets und der angegebenen Netzwerk-Maske ausgeführt:*
 - *Das Ergebnis wird mit „Zielnetz“ dieser Zeile verglichen.*
 - *Stimmt Ergebnis mit „Zielnetz“ überein, ist entsprechender Eintrag eine mögliche Route.*
 2. *Eine Liste der möglichen Routen wird erstellt und ausgewertet:*
 - *Die Route mit längstem Netzpräfix wird ausgewählt (also der Eintrag mit längster, spezifischster Netzmaske)*
 - *Falls immer noch mehrere mögliche Routen existieren, wird der Eintrag „Metrik“ ausgewertet.
Der Eintrag mit dem niedrigsten Wert wird verwendet.*
 - *Falls immer noch keine eindeutig „beste“ Route gefunden ist, kann der Router eine der übrig gebliebenen Routen beliebig auswählen.*

4.3.1 Internet Protocol Version 4 (IPv4)

- Routen-
Aggregation*

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



[Ref 1] Kapitel 4, Seite 382-385

4.3.2 Internet Protocol Version 6 (IPv6)

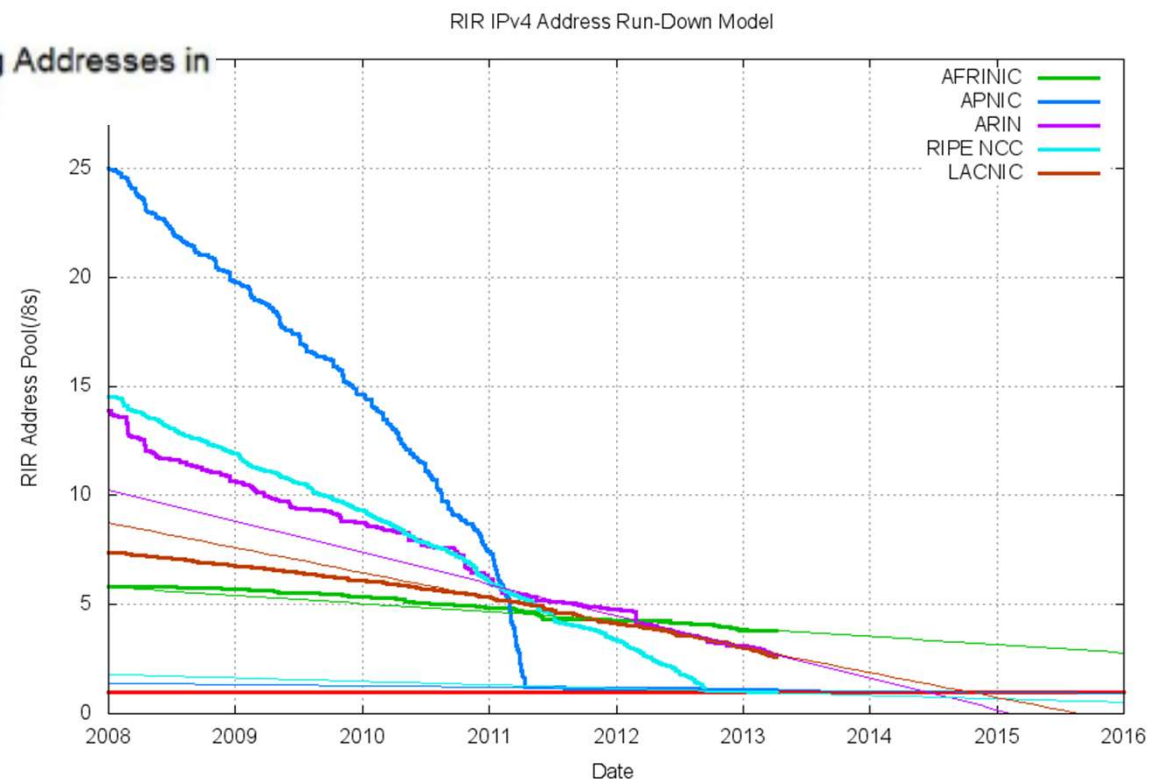
- *Das IPv4 Adress-Dilemma*

[Ref 1] Kapitel 4, Seite 396-402

[Ref 2] Kapitel 5, Seite 520-530

- *Verfügbaren IPv4-Adressräume von regionalen Internet-Registries:*

RIR	Projected Exhaustion Date	Remaining Addresses in Pool (/8s)
APNIC:	19-Apr-2011	1.2009
RIPENCC:	18-Jul-2012	3.4513
ARIN:	26-Jun-2013	5.7016
LACNIC:	01-Feb-2014	4.0902
AFRINIC:	22-Sep-2014	4.3621



4.3.2 Internet Protocol Version 6 (IPv6)

- *Ziele der Einführung von IPv6*

- ▶ *Im Juli 1994 wurde IPv6 von der IETF als Nachfolger von IPv4 ausgewählt*

- *IETF = Internet Engineering Task Force*

- ▶ *Ziele von IPv6*

- *Vergrößerung des IP-Adressraums* → *128 Bit Adresslänge statt 32 Bit*
 - *Effizientere Routing-Möglichkeiten* → *Routen-Aggregation*
→ *Vereinfachtes Header-Format*
→ *„Extension Headers“*
 - *„Quality of Service“-Unterstützung* → *„Traffic Class“, Flow Labels“*
 - *Verbesserte Security-Unterstützung* → *„Privacy Extensions“, IPsec ...*
 - *Bessere Unterstützung von Autokonfiguration und Mobilität* → *„Stateless Autoconfiguration“*
 - *Schrittweisen Migration des IPv4-basierten Internets nach IPv6* → *„Dual Stack“-Ansatz, Tunneling...*

4.3.2 Internet Protocol Version 6 (IPv6)

- *Neues IP-Header-Format*

IPv4-Header

 = aus Basisheader entfernt

IPv6-Header

 = neues Feld

fast alle anderen Header-Felder
umbenannt und ggf. inhaltlich
angepasst ...

Quelle: „IP version 6“, Guido Wessendorf, Westfälische Wilhelms-Universität Münster, Dez. 2011

+++++			
Version	IHL	Type of Service	Total Length
+++++			
	Identification	Flags	Fragment Offset
+++++			
	Time to Live	Protocol	Header Checksum
+++++			
	Source Address		
+++++			
	Destination Address		
+++++			
	Options		Padding
+++++			

+++++			
Version	Traffic Class	Flow Label	
+++++			
	Payload Length		Next Header Hop Limit
+++++			
	Source Address		
+++++			
	Destination Address		
+++++			

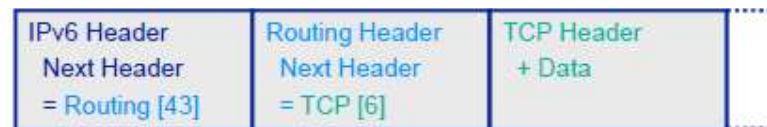
4.3.2 Internet Protocol Version 6 (IPv6)

- *Die IPv6-Erweiterungs-Header*
 - *Einem IPv6-Header können „Extension Header“ folgen*
 - *Mögliche Typen: Hop-by-Hop Options-, Routing-, Fragment-, Destination Options-, Authentication-, Encapsulation Security- und Mobility-Header*
 - *Jeder Erweiterungs Header besitzt „Next Header“-Feld als Verweis auf folgenden Header*

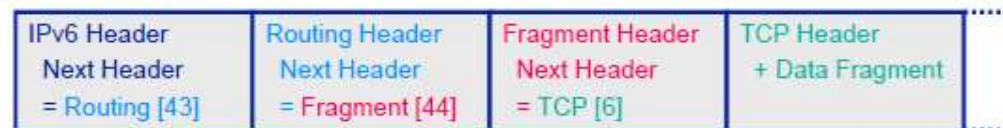
Minimaler IPv6 -Header ohne Extension Header



IPv6-Header, erweitert durch Routing-Header

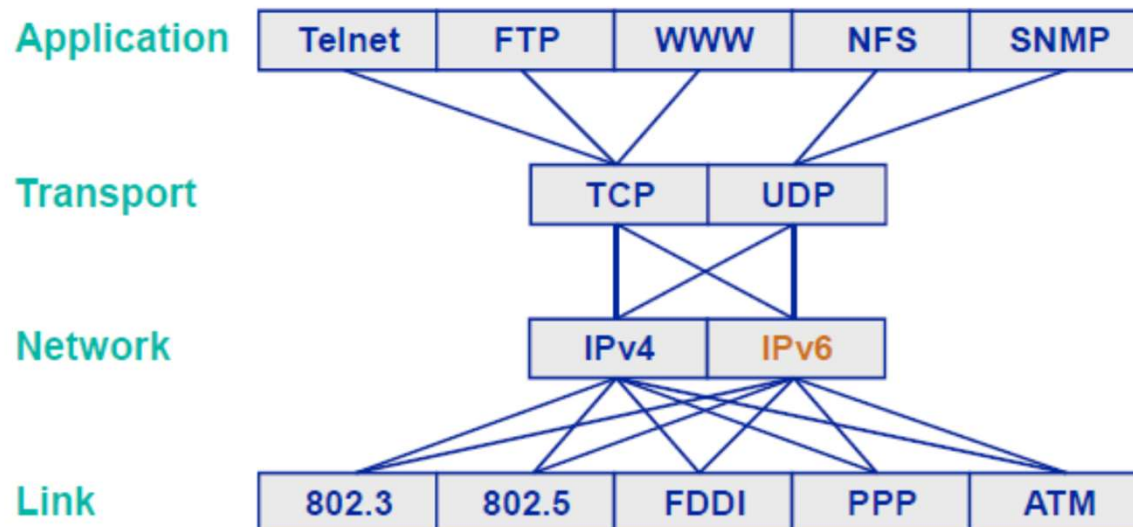


IPv6-Header, erweitert durch Routing- und Fragment-Header



4.3.2 Internet Protocol Version 6 (IPv6)

- *Migrationskonzept von IPv4 nach IPv6*
 - *Der „Dual IP-Stack“-Ansatz*
 - *IPv4-Hosts und -Router werden nach und nach um IPv6-Stack ergänzt*
 - *Software/Applikationen müssen ggf. angepasst werden!*



Quelle: „IP version 6“, Guido Wessendorf, Westfälische Wilhelms-Universität Münster, Dez. 2011

4.3.2 Internet Protocol Version 6 (IPv6)

- *Kopplung von IPv6-Netzen über existierende IPv4-Netze*
 - *Nutzung eines IPv4-Tunnels als Transitnetz*
 - *IPv6-Pakete werden in IPv4-Pakete eingebettet und als Nutzlast transportiert*
 - *Der Quell-Router (Q-R) muss eine Adressermittlungstabelle für die Zuordnung „Ziel-IPv6-Adresse \Leftrightarrow IPv4-Adresse des Tunnelendes“ erhalten*
 - *Alternativ: Nutzung von 6to4-Adressen zur automatischen Tunnel-Konfiguration auf Bedarf*

