# Kapitel 4
## Netzwerkschicht & Routing

1. Einleitung und Netzwerkdienstmodelle
2. Aufbau eines Routers
3. Das Internet-Protokoll (IPv4)
4. Paketfilterung (Firewalls)
5. Routing-Algorithmen
6. Routing-Protokolle im Internet
7. NAT vs. IPv6
8. Mobile IP

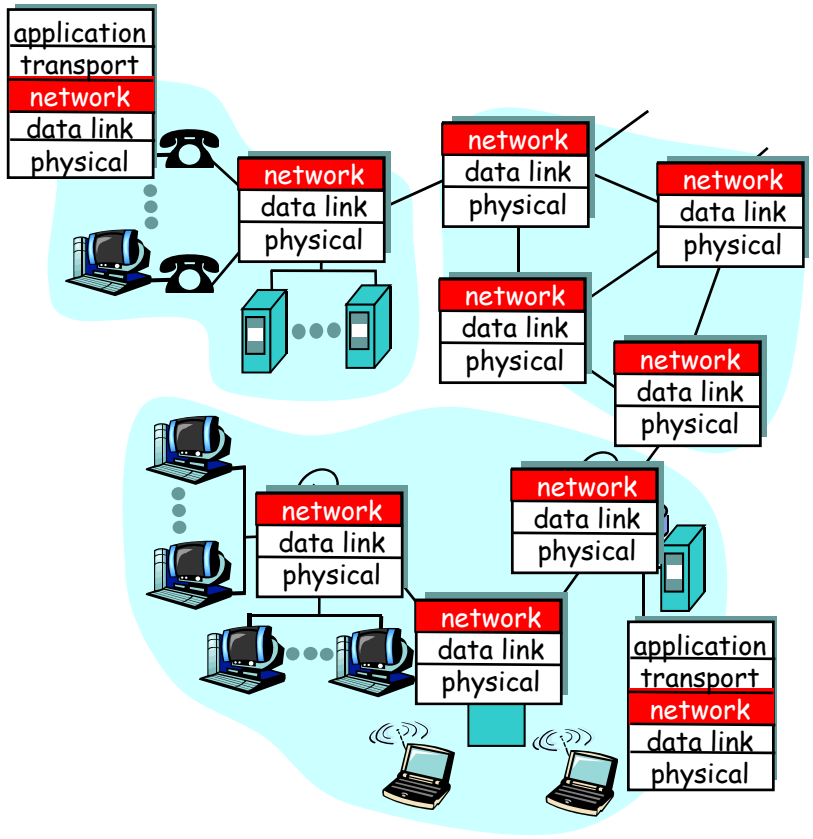
# Funktionen der Netzwerkschicht

- Transport der Pakete vom sendenden zum empf. Host
- Netzwerkschichtprotokolle laufen in *jedem* Router und Host

Drei wichtige Aufgaben:

- *Pfadbestimmung:* Bestimme den Weg (Route), den die Pakete von der Quelle zum Ziel laufen

  → *Routing-Algorithmen*
- *Switching:* Transportiere Pakete vom Eingang des Routers zum richtigen Ausgang
- *Verbindungsaufbau:* Einige Netzwerkarchitekturen benötigen die “Einrichtung” eines Pfades durch die Router vor dem Datenfluss



# Dienstmodell der Netzwerkschicht

Abstraktion des Dienstes

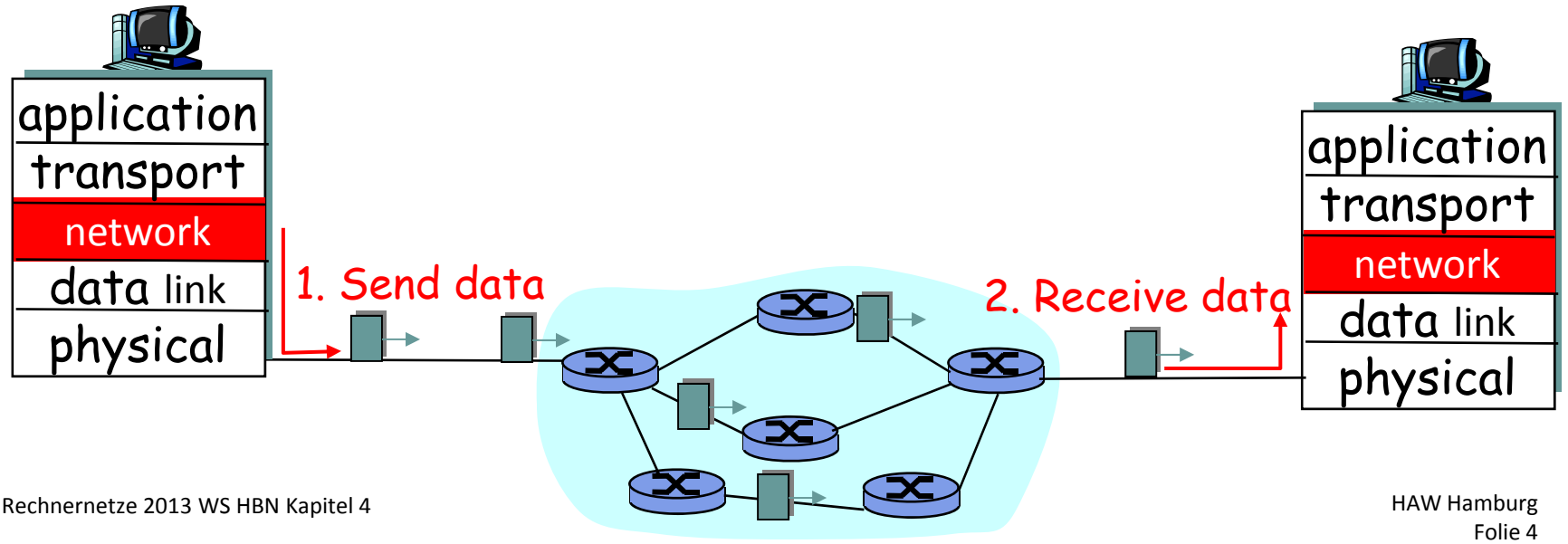
Q: Welches ***Dienstmodell*** gibt es für den "Kanal", durch den Pakete vom Sender zum Empfänger transportiert werden ?

- garantierte Bandbreite?
- Erhaltung des zeitlichen Abstandes zwischen den Paketen?
- Verlustfreier Transport?
- Reihenfolgeerhaltender Transport?
- Stau-Rückmeldung an den Sender?

Die wichtigste Abstraktion, die durch die Netzwerkschicht bereitgestellt wird:

# Datagramm-Netzwerke: das klassische Internet-Modell

- **Verbindungslos**: Kein Verbindungsaufbau auf der Netzwerkebene
- Router: kein "Zustand" über die Ende-zu-Ende-Verbindungen
  - kein "Verbindungskonzept" auf der Netzwerkebene
- Pakete werden typischerweise durch Verwendung der Zieladresse geroutet
  - Pakete zwischen dem gleichen Quelle-Ziel-Paar können unterschiedliche Wege durchs Netz laufen
- Beispiele: IPv4, IPv6 (teilweise)



# Virtuelle Kanäle ("Virtual Circuits" – VC-Netzwerke)

"Quelle-Ziel-Pfad verhält sich wie eine klassische Telefonleitung":

- In Bezug auf die Performanz
- und auf die Netzwerkaktionen auf dem Pfad von der Quelle zum Ziel

- **Verbindungsorientiert**: Verbindungsaufbau für jede Verbindung *vor* dem Transport der Daten (und Verbindungsabbau hinterher)
- Jedes Paket trägt die ID des virtuellen Kanals (VC) (nicht die Adresse des Zielhosts)
- *Jeder* Router auf dem Quelle-Ziel-Pfad speichert einen "Zustand" für jede durch ihn laufende Verbindung
  - Transportschicht-Verbindungen sind nur in den Endgeräten existent
- Resourcen der Verbindung (Übertragungskapazität, Puffer) können für den VC *reserviert* werden
  - um ein Verhalten zu erhalten, das dem einer festen Leitung entspricht
- Beispiele: IPv6 (teilweise), ATM, MPLS (zwischen Schicht 2 und 3 → Kap. 5)

# Vergleich Datagramm- / VC-Netzwerke

| | Datagramm-Netzwerk | VC-Netzwerk |
|---|---|---|
| *Verbindungsaufbau* | Nicht erforderlich | Erforderlich |
| *Adressierung* | Jedes Paket enthält die volle Quell- und Zieladresse | Jedes Paket enthält eine kurze VC-Nummer |
| *Zustandsinformation* | Router führen keine Zustands-informationen | Für jede virtuelle Verbindung ist ein Tabelleneintrag erforderlich |
| *Routing* | Jedes Paket wird unabhängig befördert | Die Route wird beim Aufbau der virtuellen Verbindung gewählt; alle Pakete folgen dieser Route |
| *Wirkung von Routerfehlern* | Nur Verlust einzelner Pakete | Alle virtuellen Verbindungen über den ausgefallenen Router werden beendet |
| *Dienstgüte-Garantie* | Schwierig | Einfach, wenn ausreichende Ressourcen reserviert sind |
| *Überlastkontrolle* | Schwierig | Einfach, wenn ausreichende Ressourcen reserviert sind |
| *Flexibilität* | sehr hoch | gering (hoher Verwaltungs- und Abstimmungsaufwand) |

nach [ATN]

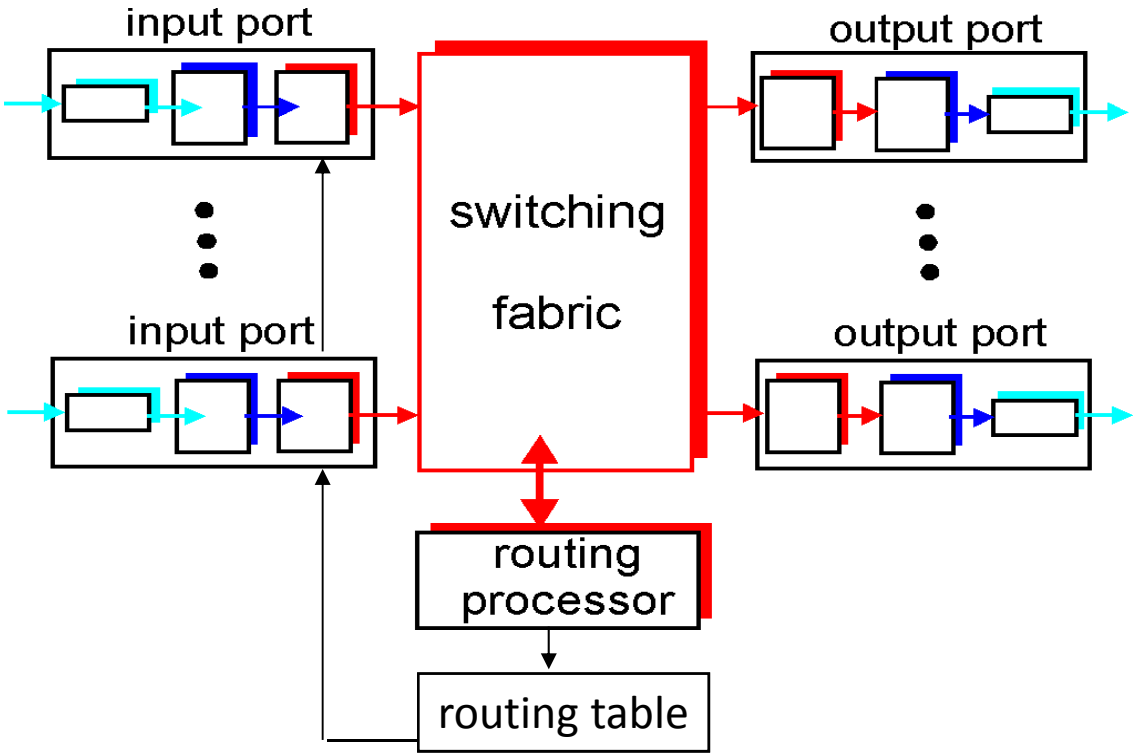
# Kapitel 4
## Netzwerkschicht & Routing

1. Einleitung und Netzwerkdienstmodelle
2. Aufbau eines Routers
3. Das Internet-Protokoll (IPv4)
4. Paketfilterung (Firewalls)
5. Routing-Algorithmen
6. Routing-Protokolle im Internet
7. NAT vs. IPv6
8. Mobile IP

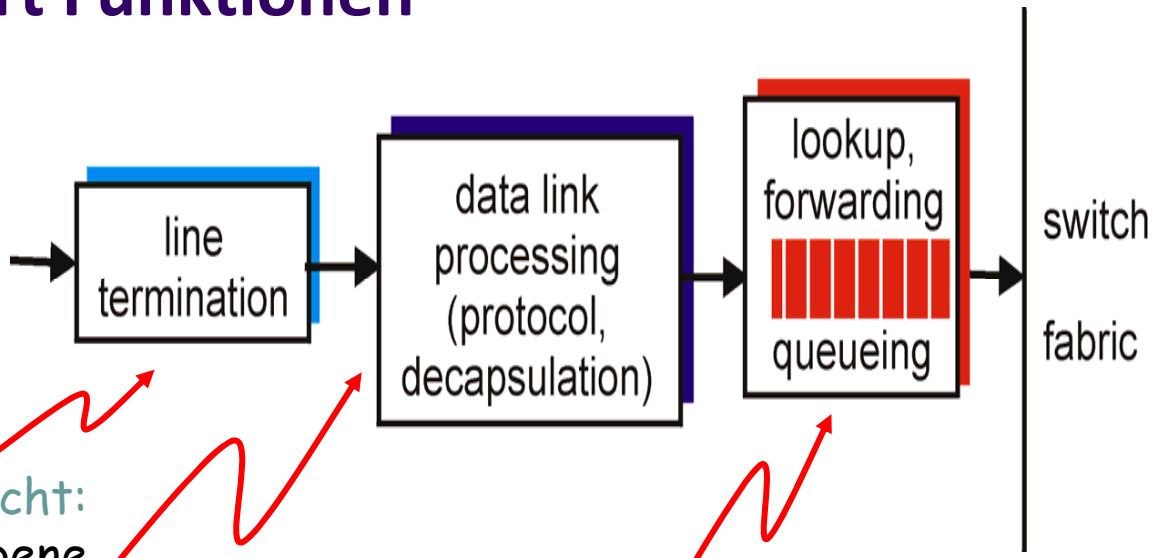
# Router-Architektur: Überblick

Hauptfunktionen:

- *Pfadermittlung (“routing”)*: Aktualisierung der “Routing-Tabelle”
  - Def. der Abbildung: Zieladresse → Ausgangsleitung
- *Weiterleitung (“forwarding”)* von Paketen (Datagrammen) von einer Eingangs- zu einer Ausgangsleitung



# Input Port Funktionen



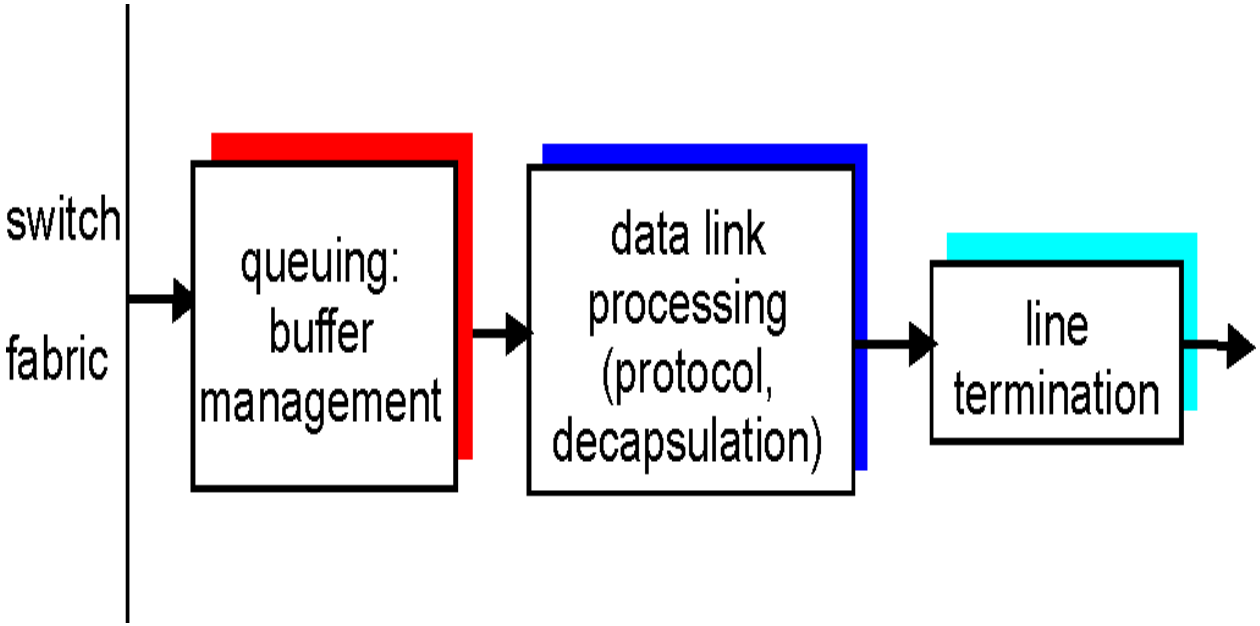

Physikalische Schicht:
Empfang auf Bitebene

Sicherungsschicht:
z.B. Ethernet

**Dezentralisierte Entscheidungen**

- Bestimme aufgrund der Routingtabelle die Ausgangsleitung
- Ziel: Komplette Verarbeitung eines Datagramms innerhalb der Empfangszeit!
- Warteschlange ("queuing"): Nötig, wenn Datagramme schneller ankommen als sie in das Schaltnetz ("switch fabric") eingestellt werden können

# Output Ports



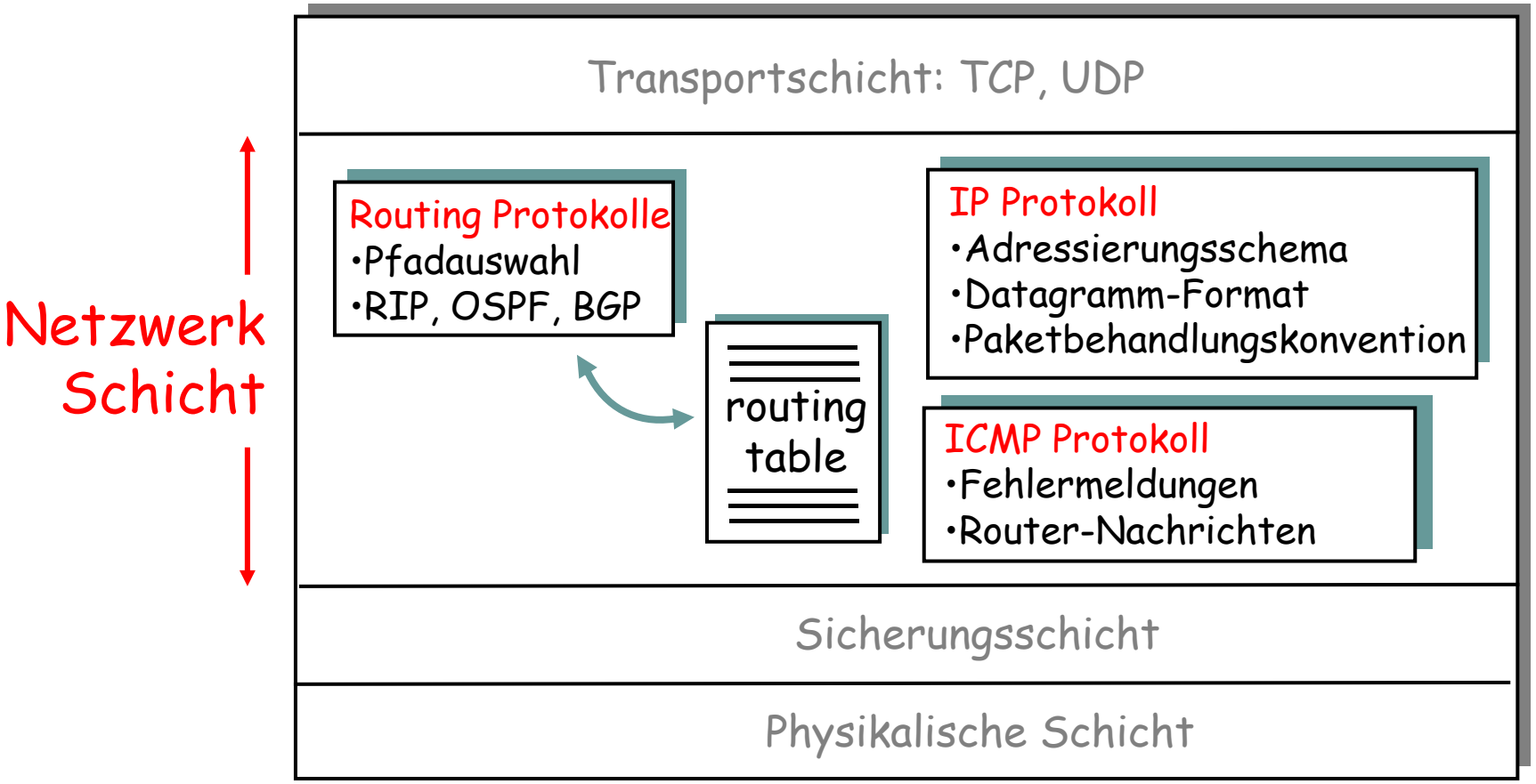

- Pufferung ("queuing"): Nötig, wenn Datagramme schneller aus dem Schaltnetz ("switch fabric") ankommen als sie übertragen werden können
- Über eine Scheduling-Strategie muss das nächste zu übertragende Datagramm aus dem Puffer gewählt werden (FCFS, Prioritäten, …)

# Kapitel 4
## Netzwerkschicht & Routing

1. Einleitung und Netzwerkdienstmodelle
2. Aufbau eines Routers
3. Das Internet-Protokoll (IPv4)
4. Paketfilterung (Firewalls)
5. Routing-Algorithmen
6. Routing-Protokolle im Internet
7. NAT vs. IPv6
8. Mobile IP

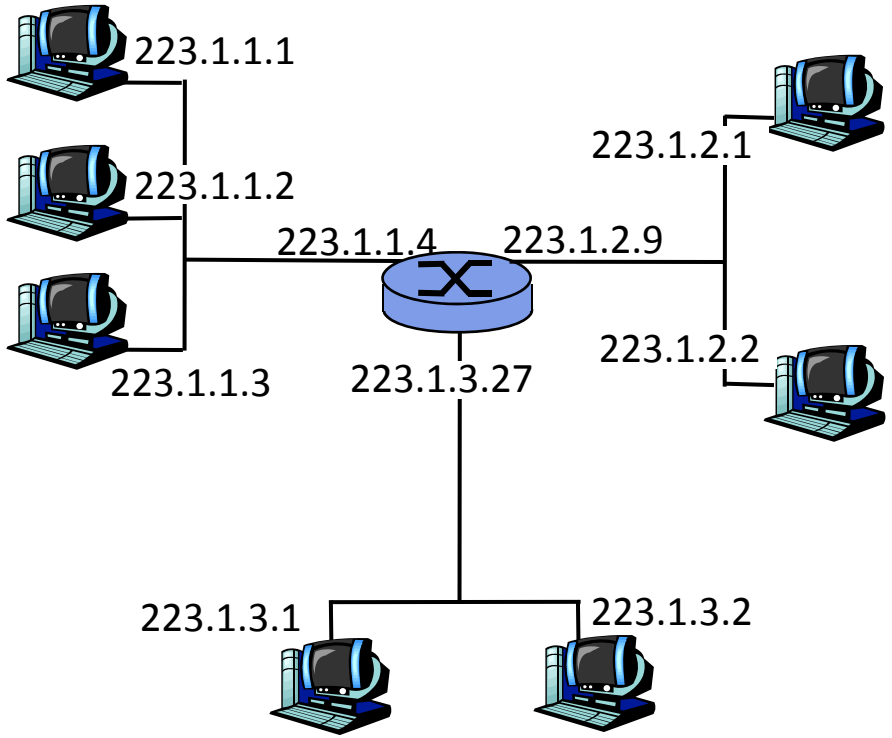
# Die Internet-Netzwerkschicht (IPv4)

Netzwerkschicht-Funktionen von Hosts und Routern:



# IPv4 Adressierung: Einführung

- IP-Adresse: 32-bit ID für Host- und Router-*Interface*
- *Interface:* Schnittstelle zwischen Host/Router und physikalischer Verbindungsleitung
  - Router haben viele Interfaces
  - Hosts können mehrere Interfaces haben
  - IP-Adressen werden einem **Interface** (nicht Host oder Router) nach Bedarf zugewiesen

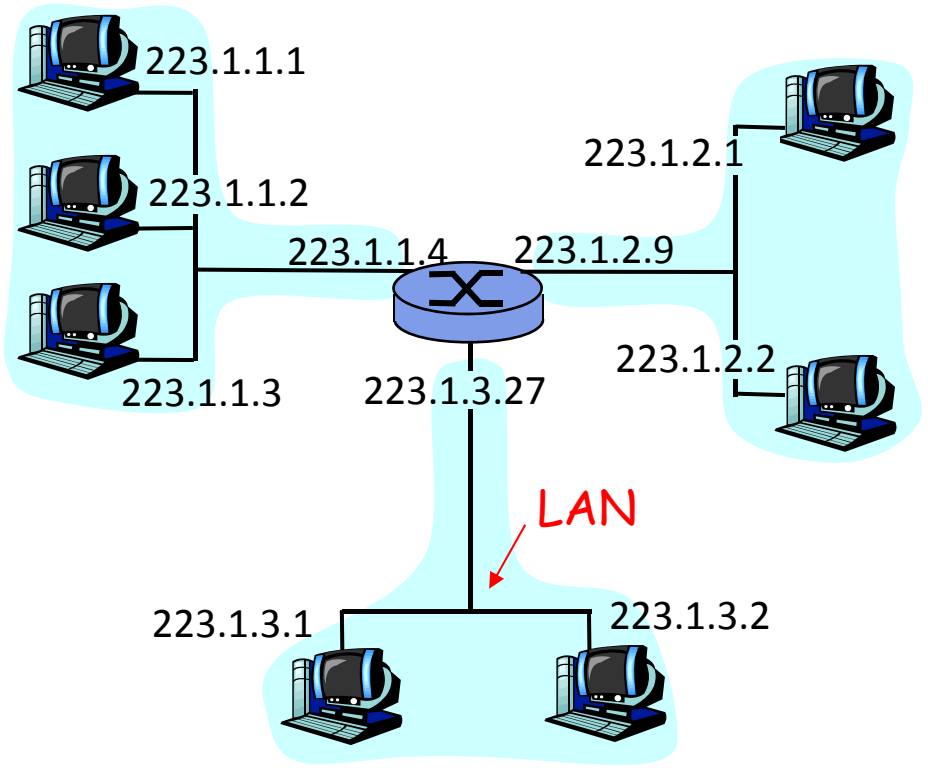



223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

# IPv4 Adressierung: IP-Netzwerke

- IP-Adresse:
  - Netzwerk-Teil (high order bits)
  - Host-Teil (low order bits)
- *Was ist ein Netzwerk ?* (aus IP-Perspektive)
  - Interfaces mit identischem Netzwerk-Teil der IP-Adresse,
  - die sich physikalisch gegenseitig ohne Inanspruchnahme eines Routers erreichen können

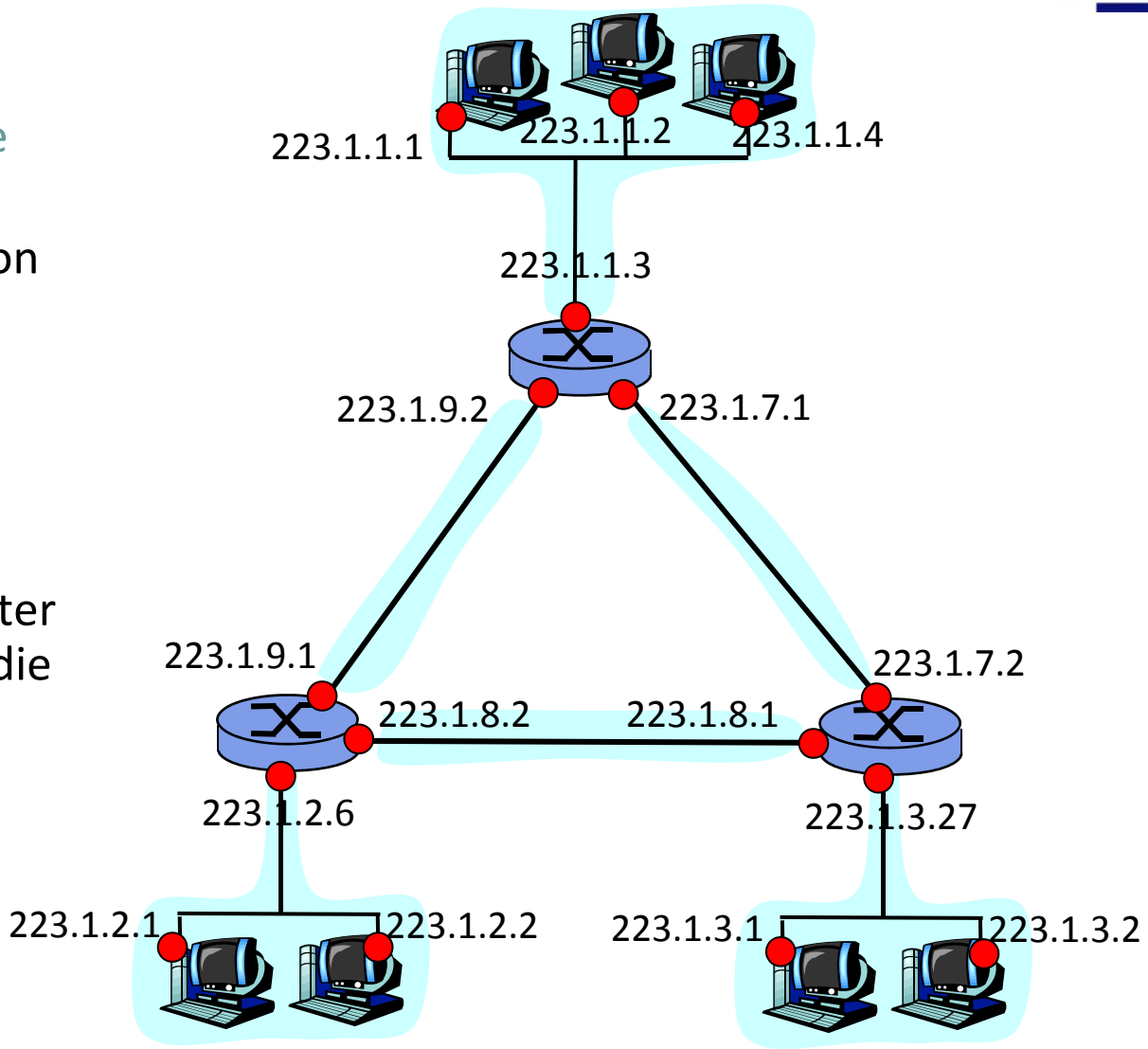



Netzwerk bestehend aus 3 IP-Netzwerken (die ersten 24 Bit einer IP-Adresse sind hier der Netzwerk-Teil)

# Definition von IP-Netzwerken

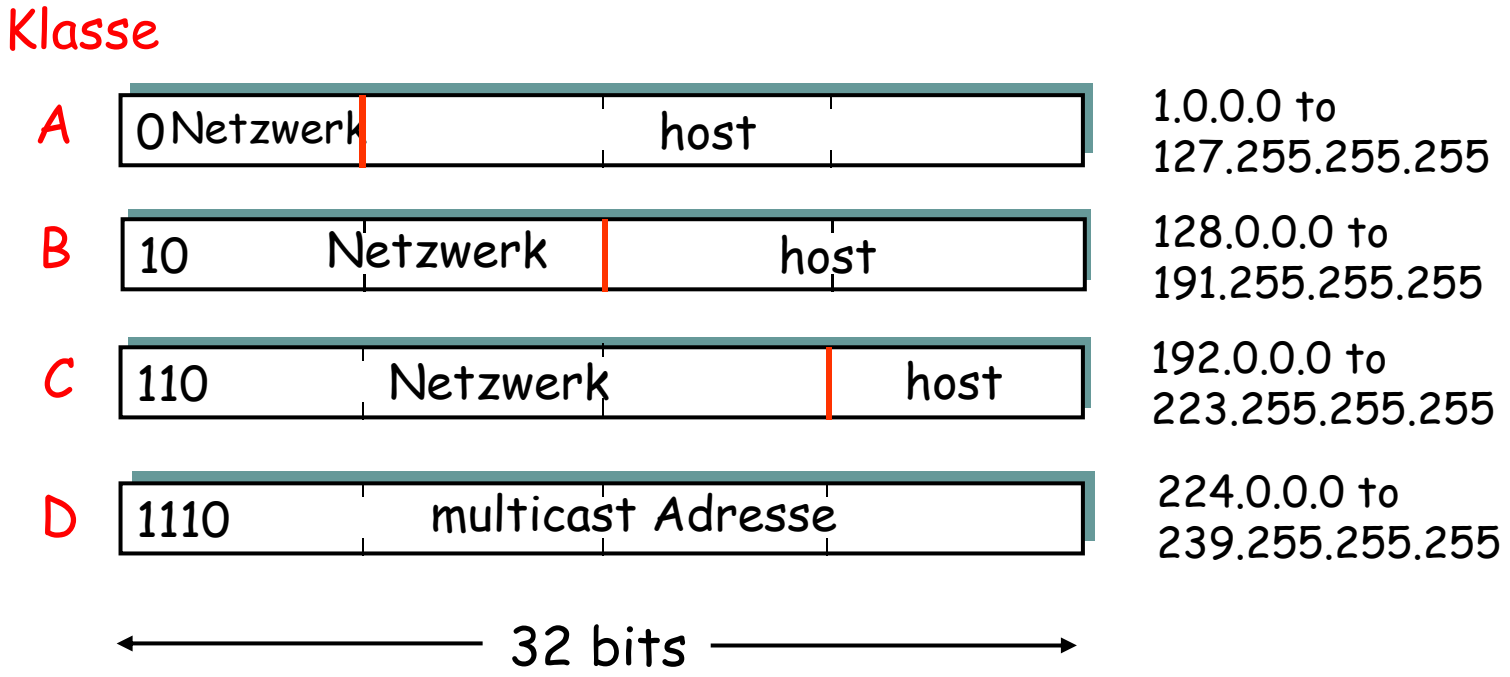
Wie werden IP-Netzwerke definiert?

- Jedes Interface wird von seinem Router/Host abgekoppelt
- So entstehen “Inseln” mit isolierten IP-Netzwerken
- Die Interfaces der Router sind Endpunkte, über die ein IP-Netzwerk mit anderen verbunden werden kann




Verbundsystem aus 6 Netzwerken

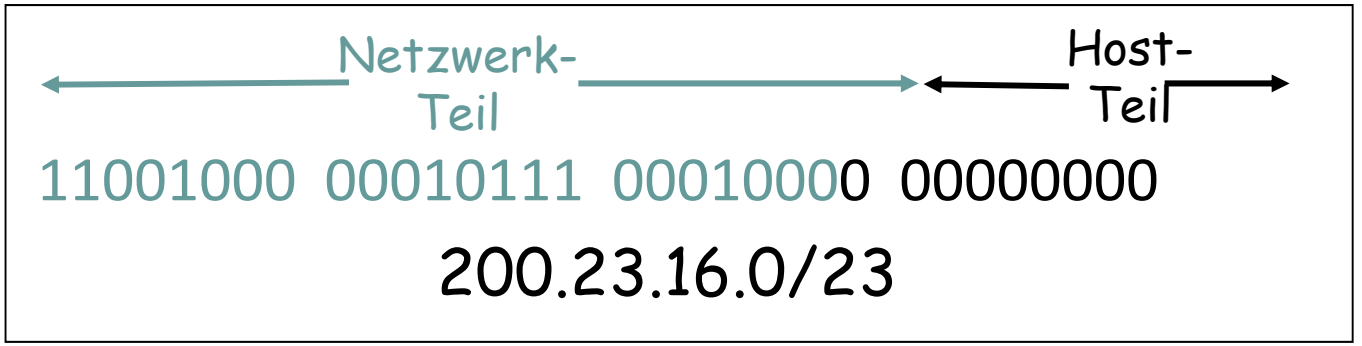
# Strukturierung des IP-Adressraums: Adressklassen *(klassisch)*



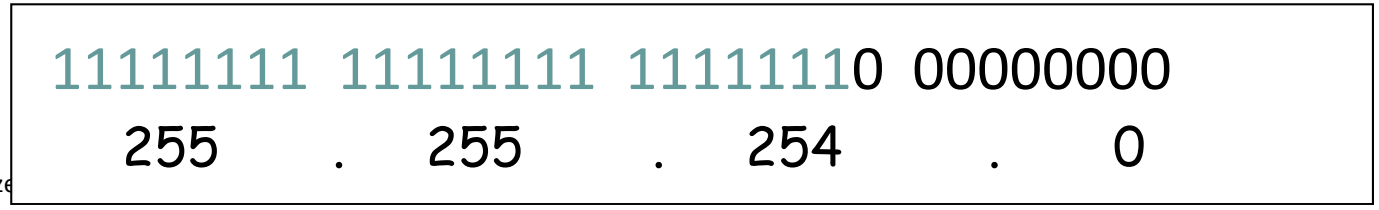

- Nachteile der IP-Adressklassen:
  - Ineffiziente Nutzung des Adressraums!
  - Beispiel: Ein Klasse-B-Netz belegt 65.536 Adressen, auch wenn in einer Firma nur 2.000 genutzt werden

# Strukturierung des IP-Adressraums: CIDR *(neu)*

- CIDR: Classless InterDomain Routing
  - Netzwerk-Teil einer IP-Adresse kann von *beliebiger* Länge sein
  - Adressformat: **a.b.c.d/x**, wobei x die Anzahl der Bits im Netzwerk-Teil der Adresse darstellt




  - Aufteilung Netzwerkteil/Hostteil wird auch über eine "Subnetzmaske" angegeben:



# Vergabe von IP-Adressen

- ICANN: Internet Corporation for Assigned Names and Numbers ("Politische" Oberorganisation)
  - Vergabe von IP-Adressbereichen (→ Netzwerkteil) an ISPs und große Organisationen
  - Verwaltung von DNS-Top-Level-Domains und Betrieb der DNS-Root Server

  Delegation der technischen Durchführung:

  - IANA: Internet Assigned Numbers Authority ("Technische" Zentralorganisation)

    Delegation der regionalen Zuständigkeit:
    - ARIN (American Registry for Internet Numbers)
    - RIPE (Réseaux IP Européens)
    - APNIC (Asia Pacific Network Information Centre)

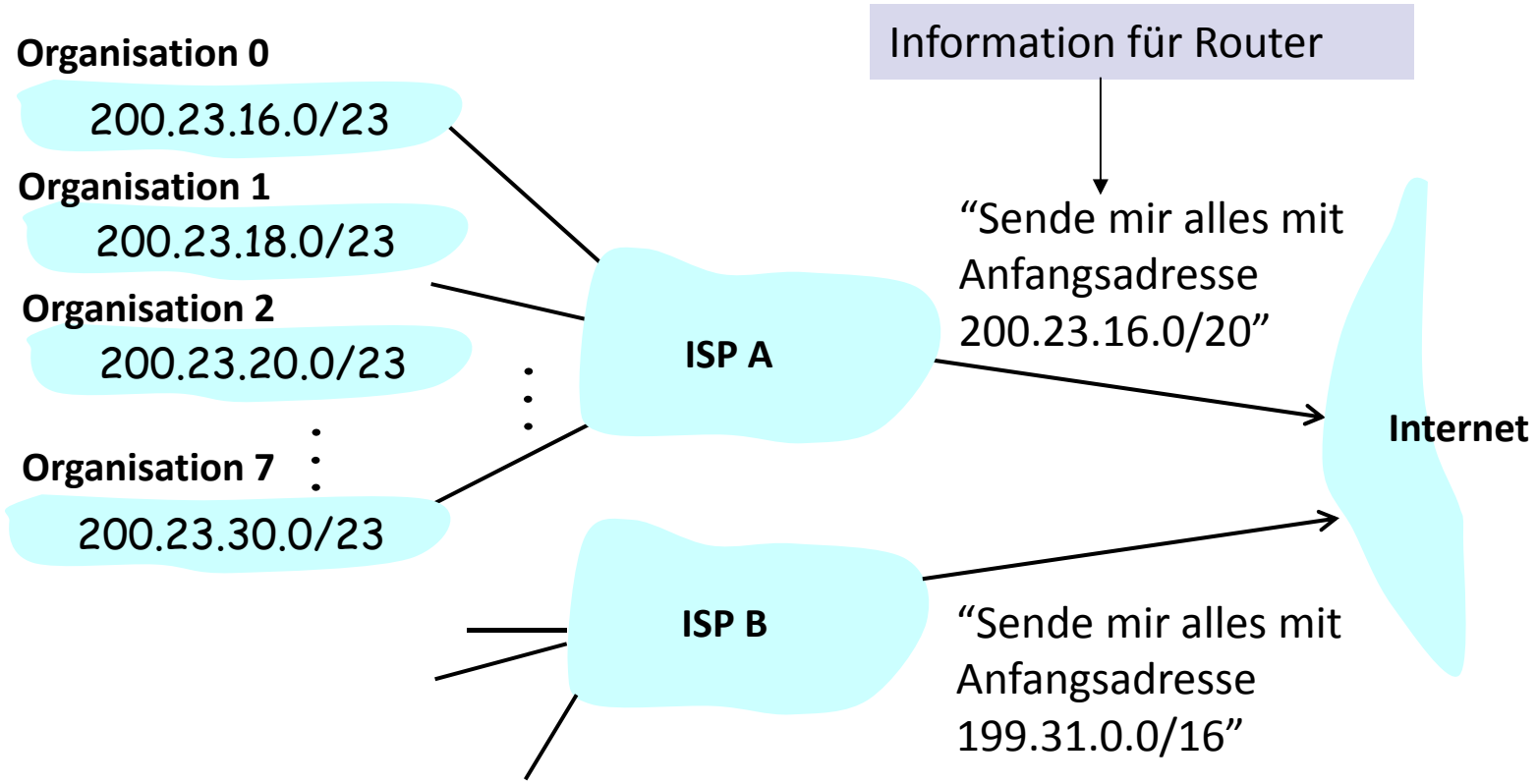
# Weitergabe von IP-Adressen durch ISPs

Ein ISP kann seinen zugewiesenen Adressbereich untergliedern (indem er den Netzwerk-Teil erweitert) und damit Subnetze an Organisationen weitergeben ("Subnetting" RFC 950)

| | | | |
|---|---|---|---|
| ISP-Adressblock | 11001000 00010111 00010000 | 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 |

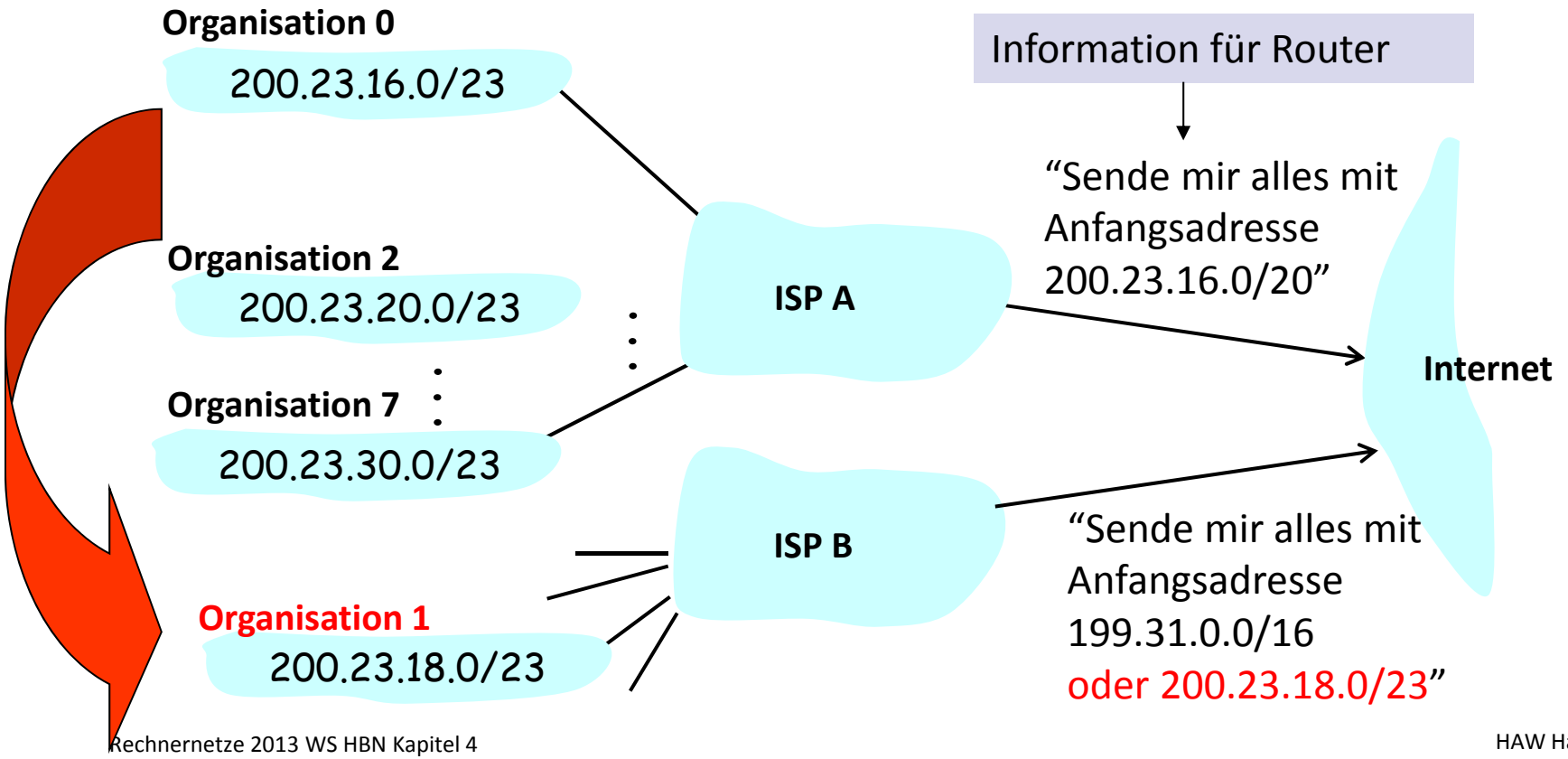
# Hierarchische Adressierung: Routenaggregation

Ziel: Vereinfachung von Routingtabellen durch Zusammenfassung von Subnetzen



# Hierarchische Adressierung: “Longest Prefix Matching”

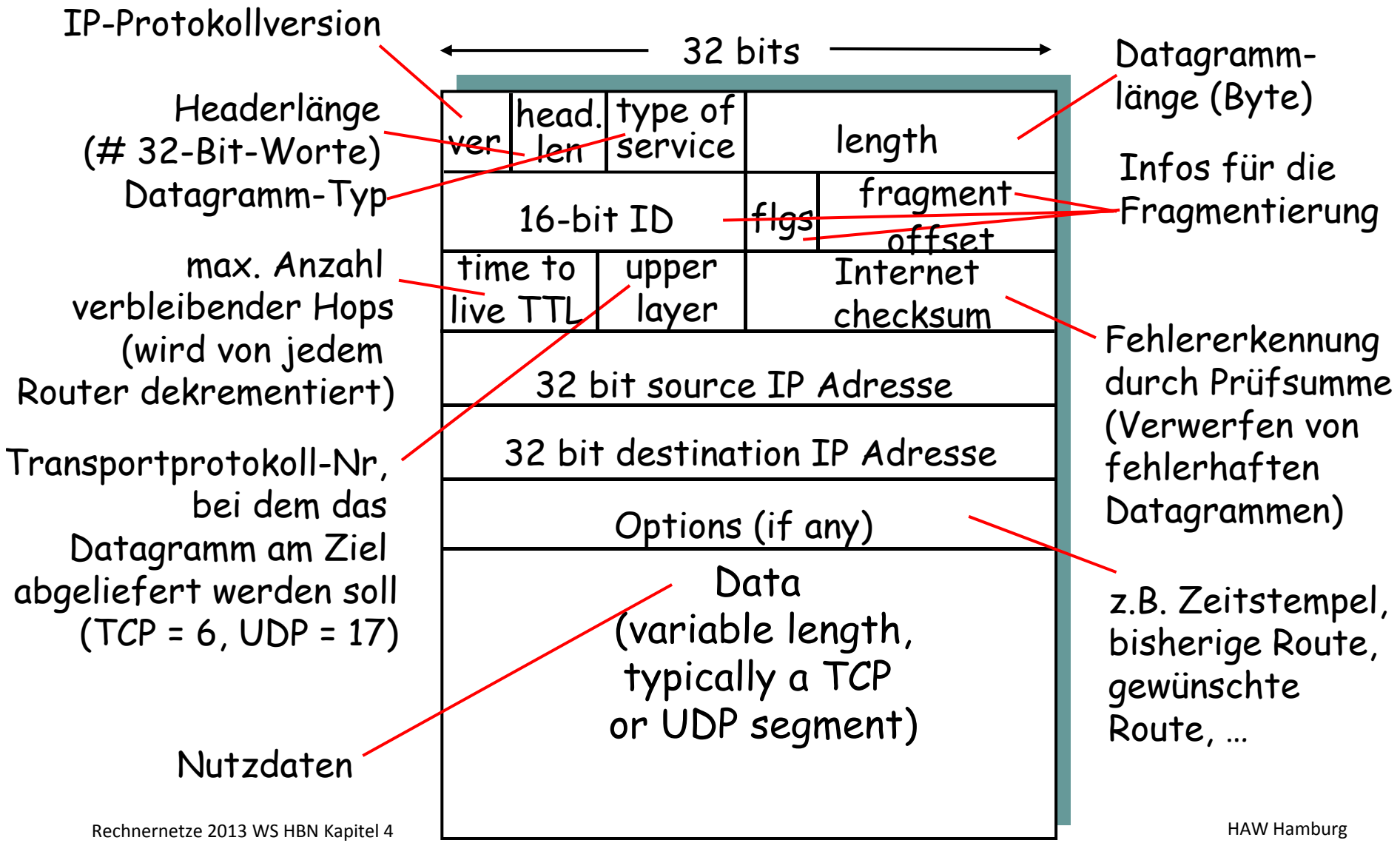
- Organisation 1 ist zu ISP B gewechselt und hat seinen IP-Adressbereich mitgenommen
- Pakete für Organisation 1 werden aus dem Internet an ISP B gesendet, wenn die ersten 23 Bit der Adresse übereinstimmen!



# Zuweisung von IP-Adressen an Hosts

- Eintrag in eine Systemdatei (von Hand durch Administrator)
- DHCP: Dynamic Host Configuration Protocol [RFC 2131]
  Dynamische Adresszuweisung: “plug-and-play”
  - Protokollablauf
    - Host sendet “DHCP discover” als Broadcast-UDP-Message
      (IP-Zieladresse: 255.255.255.255, Quelladresse: 0.0.0.0)
    - DHCP-Server antwortet mit “DHCP offer” (Broadcast oder direkt an LAN-Adresse → Schicht 2!)
    - Host fragt nach IP-Adresse: “DHCP request”
    - DHCP-Server sendet IP-Adresse: “DHCP ack”
  - Möglichkeiten der Zuordnung Host – IP-Adresse:
    - Statisch (z.B. in Firmennetzen)
      → “feste” Zuordnung (über LAN-Adresse)
    - Dynamisch (z.B. bei ISPs)
      → “zufällige” Zuordnung (IP-Adresse aus Pool wird “vermietet”)

# Format eines IPv4-Datagramms



# Der Weg eines Datagramms von der Quelle zum Ziel: Szenario

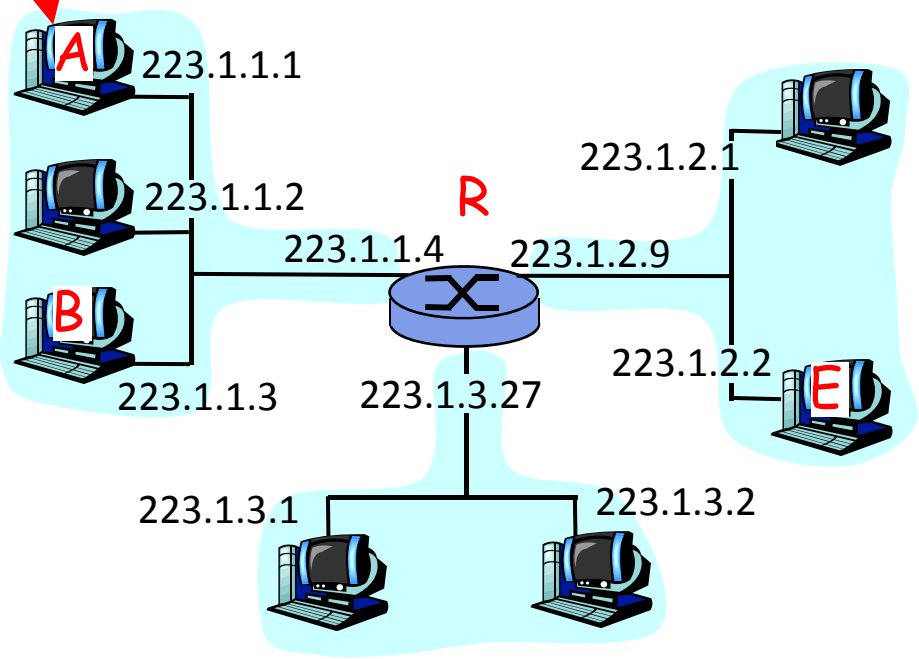
IP Datagramm:

| ... | source IP addr | dest IP addr | data |
|---|---|---|---|

- Ein Datagramm wird auf dem Weg von einer Quelle (A) zu einem Ziel (B oder E) in den IP-Adressfeldern nicht verändert!
- Zusätzlich benötigt: "LAN-Adresse" auf Schicht 2 → kommt später genauer!

Routingtabelle in A

| Zielnetz | Nächster Router | Interface |
|---|---|---|
| 223.1.1.0/24 | – | 223.1.1.1 |
| 223.1.2.0/24 | 223.1.1.4 | 223.1.1.1 |
| 223.1.3.0/24 | 223.1.1.4 | 223.1.1.1 |



# Der Weg eines Datagramms von der Quelle zum Ziel: Beispiel 1

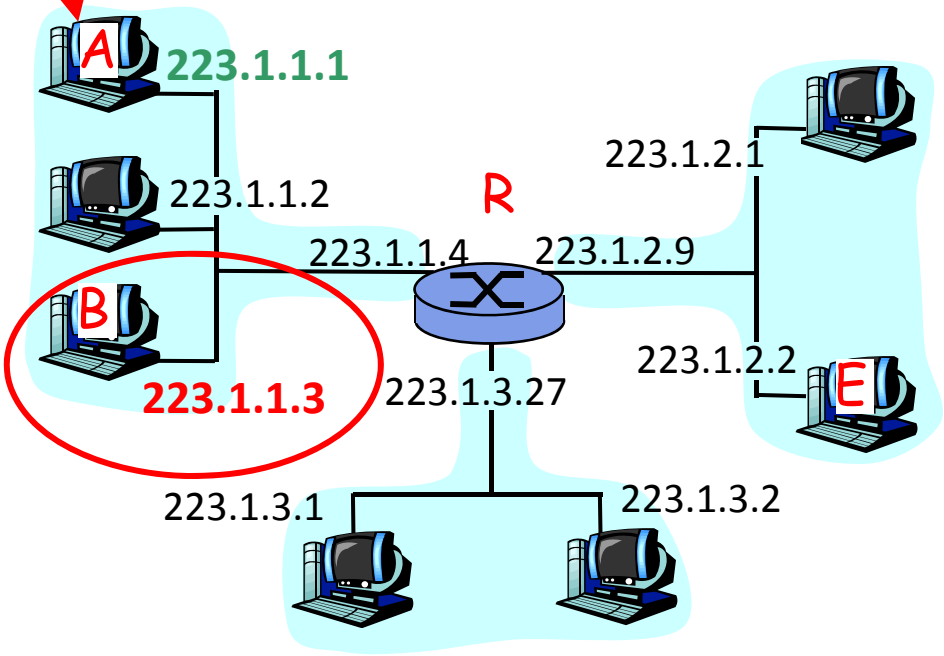
| ... | 223.1.1.1 | 223.1.1.3 | data |
|---|---|---|---|

**Quelle A sendet ein IP-Datagramm zum Ziel B**

1. A: Ermittle Netzwerkadresse von B = Zielnetz
2. A: Suche das Zielnetz in der Routingtabelle: Eigenes Netz!
3. A: Sende das Datagramm über die Sicherungsschicht ins eigene LAN mit LAN-Adresse von B (A und B sind direkt miteinander verbunden!)

Routingtabelle in A

| Zielnetz | Nächster Router | Interface |
|---|---|---|
| 223.1.1.0/24 | – | 223.1.1.1 |
| 223.1.2.0/24 | 223.1.1.4 | 223.1.1.1 |
| 223.1.3.0/24 | 223.1.1.4 | 223.1.1.1 |



# Der Weg eines Datagramms von der Quelle zum Ziel: Beispiel 2 (I)

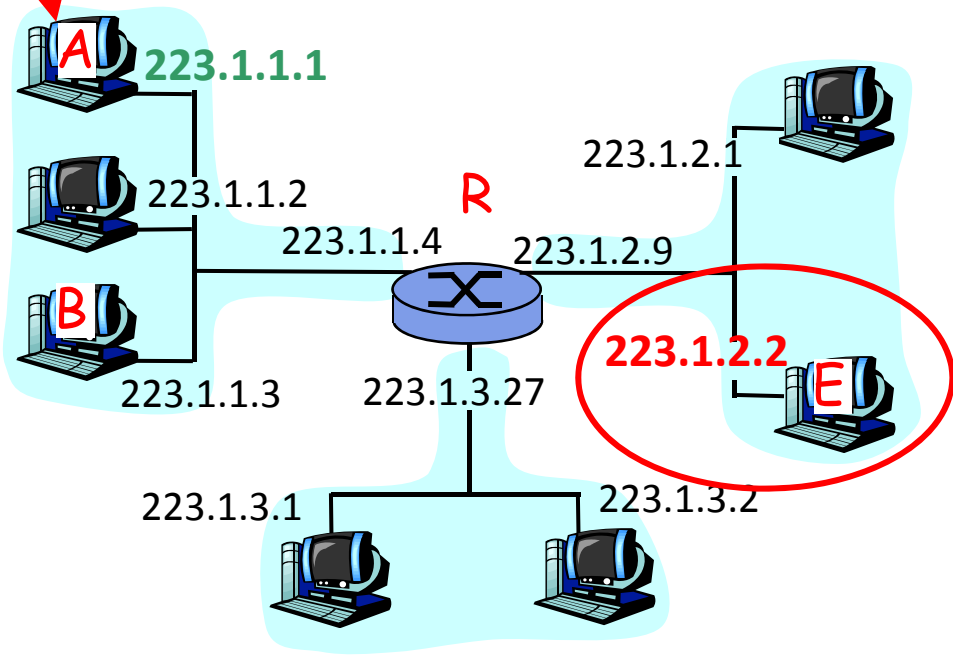
| ... | 223.1.1.1 | 223.1.2.2 | data |
|---|---|---|---|

**Quelle A sendet ein IP-Datagramm zum Ziel E**

1. A: Ermittle Netzwerkadresse von E = Zielnetz
2. A: Suche das Zielnetz in der Routingtabelle: → E ist in anderem Netzwerk! Nächster Router R hat die Adresse 223.1.1.4 (R ist in eigenem Netzwerk)
2. A: Sende das Datagramm über die Sicherungsschicht ins eigene LAN mit LAN-Adresse des Routers R

Routingtabelle in A

| Zielnetz | Nächster Router | Interface |
|---|---|---|
| 223.1.1.0/24 | – | 223.1.1.1 |
| 223.1.2.0/24 | 223.1.1.4 | 223.1.1.1 |
| 223.1.3.0/24 | 223.1.1.4 | 223.1.1.1 |



# Der Weg eines Datagramms von der Quelle zum Ziel: Beispiel 2 (II)

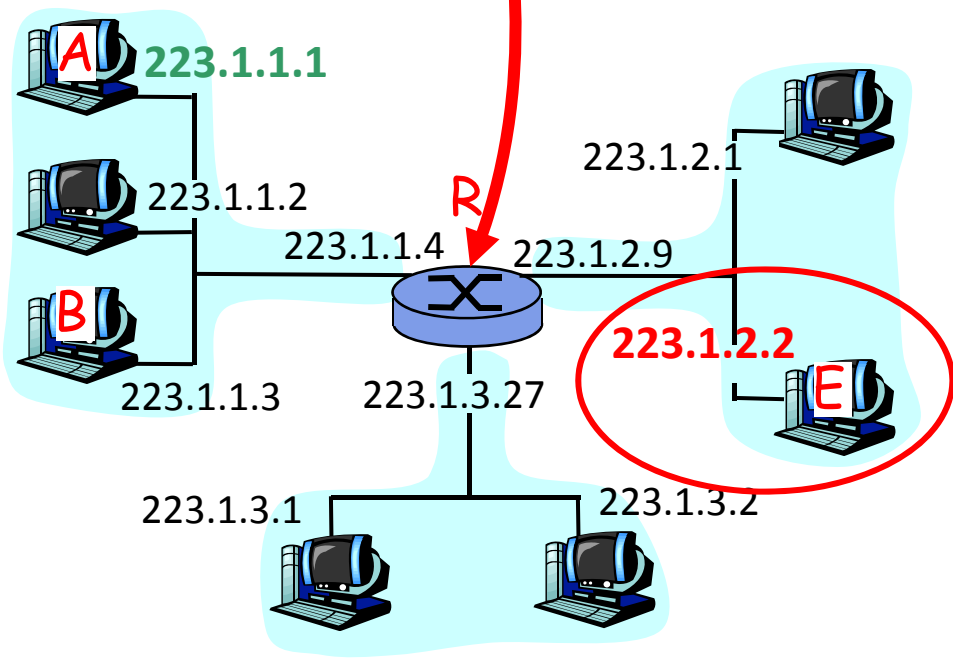
| ... | 223.1.1.1 | 223.1.2.2 | data |
|---|---|---|---|

Routingtabelle im Router R

| Ziel-netz | nächst. Router | Interface |
|---|---|---|
| 223.1.1.0/24 | - | 223.1.1.4 |
| 223.1.2.0/24 | - | 223.1.2.9 |
| 223.1.3.0/24 | - | 223.1.3.27 |

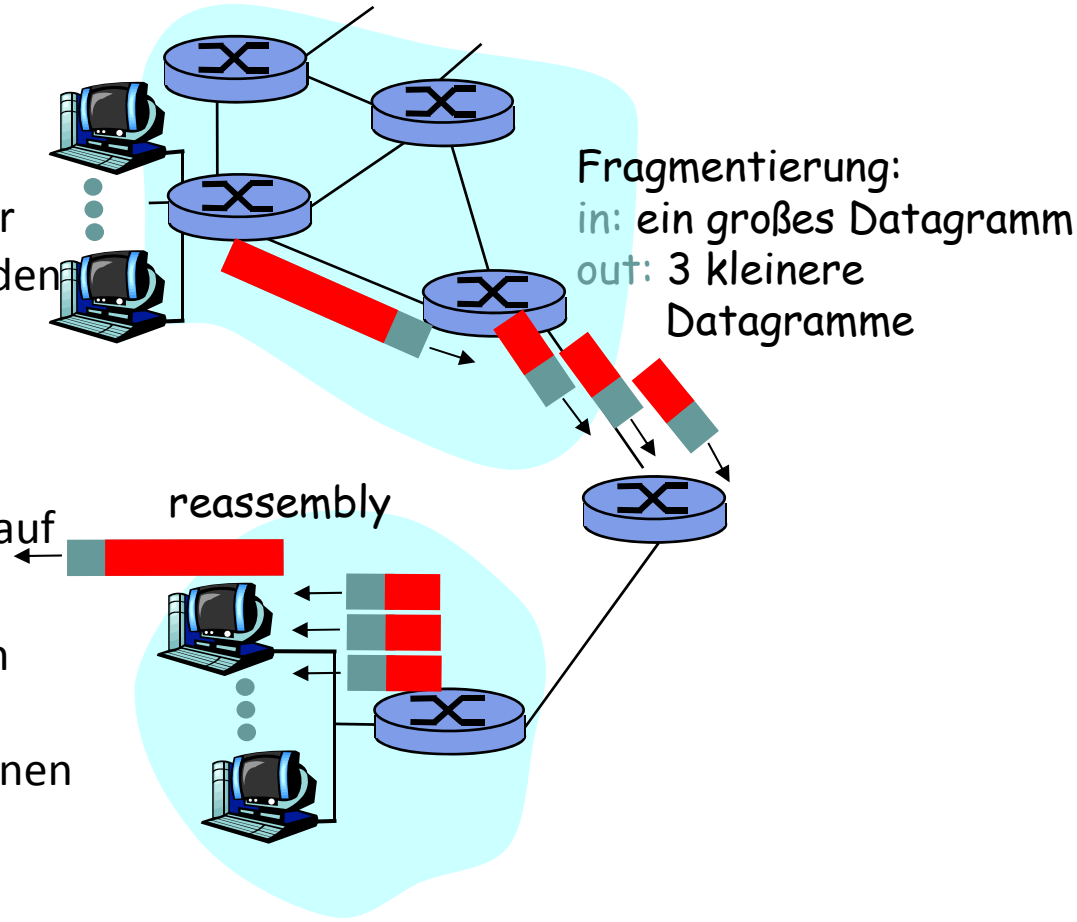
Router R erhält das Datagramm auf Interface 223.1.1.4 und muss es an **E** weiterleiten

1. R: Ermittle Netzwerkadresse von E = Zielnetz
2. R: Suche das Zielnetz in der Routingtabelle: E ist im selben Netzwerk wie Interface 223.1.2.9
3. R: Sende das Datagramm über Interface 223.1.2.9 und die entspr. Sicherungsschicht ins LAN mit LAN-Adresse von E

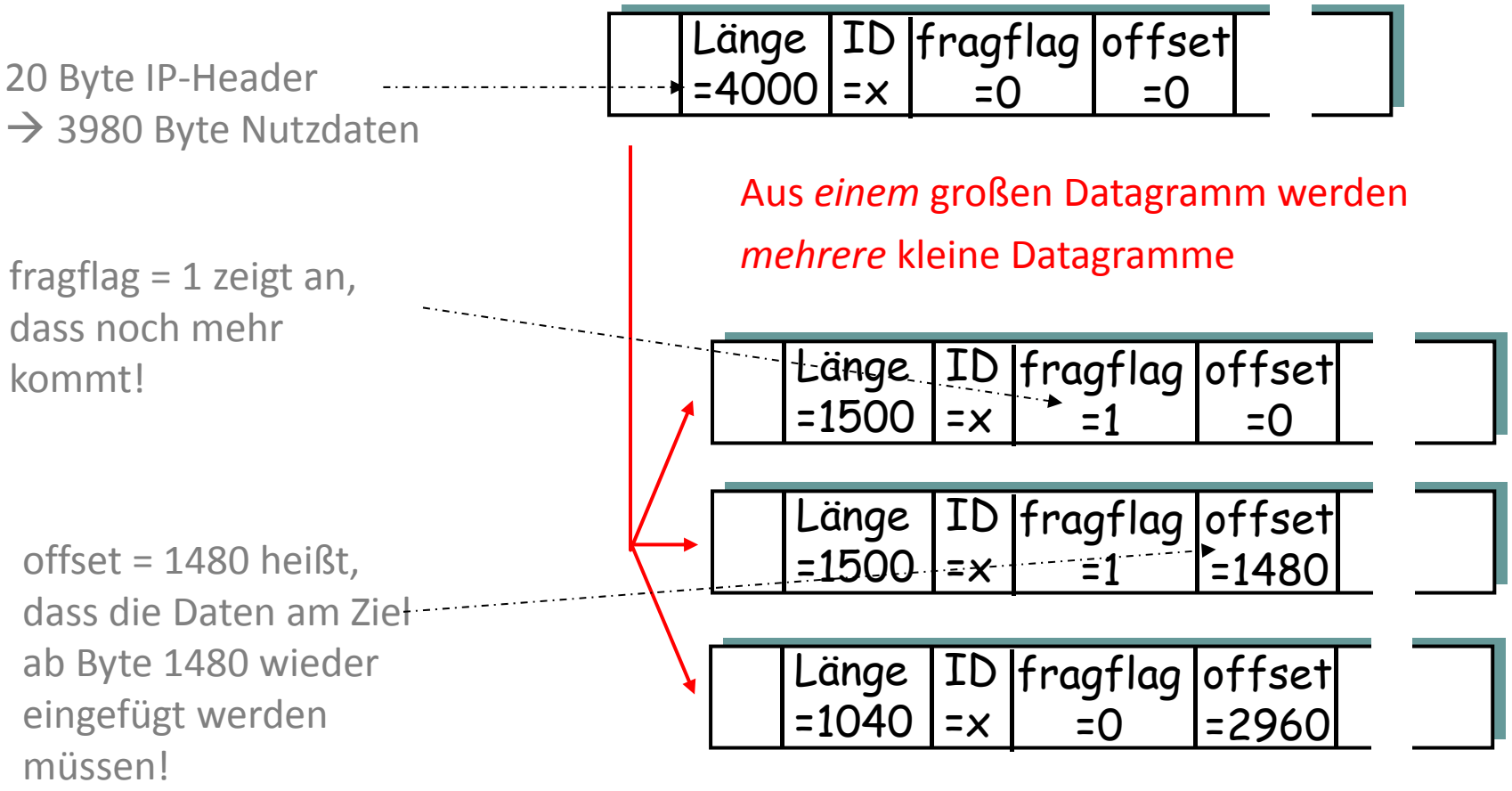


# IPv4-Fragmentierung und Reassemblierung

- Sicherungsschicht-Pakete haben eine MTU (Max. Transfer Unit) = größte mögliche Paketlänge
- MTU ist abhängig vom Protokoll, Hardware, Betriebssystem, …
- Große IP-Datagramme müssen daher evtl. aufgeteilt ("fragmentiert") werden
  - aus *einem* Datagramm werden *mehrere* Datagramme
  - Zusammensetzen ("Reassemblierung") findet *nur* auf dem Zielhost statt!
  - IP-Headerinformationen werden zur Identifikation und Reihenfolgeerhaltung der einzelnen Fragmente benötigt



# IPv4-Fragmentierung und Reassemblierung: Beispiel



# ICMP: Internet Control Message Protocol

- Benutzt von Hosts und Routern, um Steuerungsinformationen auf Netzwerkebene auszutauschen
  - Fehlermeldungen (z.B. “dest host unreachable”)
  - Statusmeldungen (z.B. “echo request/reply” → ping)
- ICMP-Nachrichten werden in IP-Datagrammen transportiert!
- ICMP-Nachrichtenformat:
  - Typ
  - Code
  - erste 8 Byte des IP-Datagramms, das den Fehler verursacht hat

| 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 |
| … | … | |

# Kapitel 4
## Netzwerkschicht & Routing

1. Einleitung und Netzwerkdienstmodelle
2. Aufbau eines Routers
3. Das Internet-Protokoll (IPv4)
4. Paketfilterung (Firewalls)
5. Routing-Algorithmen
6. Routing-Protokolle im Internet
7. NAT vs. IPv6
8. Mobile IP

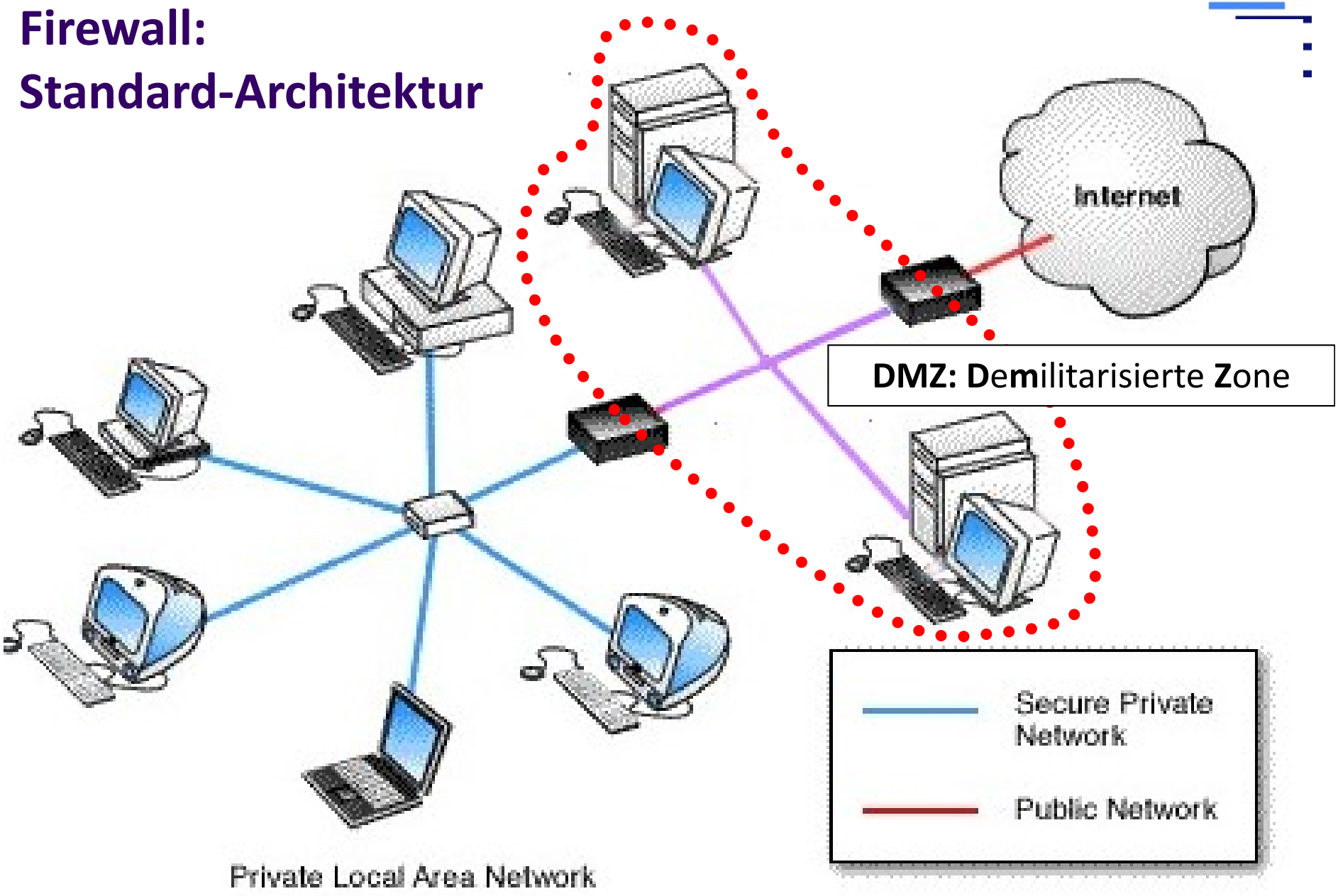
# Firewall

- Vergleich mit Burgtor / Burggraben einer mittelalterlichen Burg:
  - Erlaubt Eintritt und Verlassen nur an einem bewachten Punkt
  - Verhindert, dass Angreifer an weitere Verteidigungsanlagen herankommen
- Grenze zwischen unsicherem und vertrauenswürdigem Netz
- Technische Umsetzung einer Sicherheitspolitik:
  - Durchlassen von akzeptablem Netzverkehr
  - Sperren von nicht akzeptablem Netzverkehr (Verwerfen, Protokollieren, ggf. weitergehende Analyse: Entdeckung von Angriffen/Störungen)

# Firewall: Komponenten

- Eine (professionelle) Firewall kann aus den folgenden Komponenten bestehen:
  - **Paketfilter**: Anwendungsunabhängige Filterung der Datenpakete
  - **Applikationsfilter (= Proxy Server)**: Anwendungsspezifische Filterung der Datenpakete
- Ein im Internet „sichtbarer“ Rechner heißt **Bastion Host**
- Beim Einsatz mehrerer Firewall-Komponenten heißt der Netzbereich zwischen der ersten Firewall-Komponente vor dem Internet und der letzten Firewall-Komponente vor dem internen Netz **„Demilitarisierte Zone“ (DMZ)**

# Firewall: Standard-Architektur



# Paketfilter

- Filterung von Datenpaketen aufgrund von Informationen auf ISO/OSI-Ebenen 3 und 4:
  - Quell-IP-Adresse
  - Ziel-IP-Addresse
  - TCP/UDP Quell- und Ziel-Portnummern
  - ICMP Nachrichtentyp
  - TCP SYN und ACK Flags
- In der Regel zustandslos!
- Spezifikation über Filterregeln / -tabellen

- Beispiel 1: Blockiere ein- und ausgehende Pakete, bei denen im IP-Protokollfeld (“upper layer”) 17 steht oder deren Quell- oder Zielport 23 ist
  - Alle UDP-Pakete und Telnet-Verbindungen sind gesperrt!
- Beispiel 2: Blockiere eingehende TCP-Segmente mit ACK=0
  - Verhindert, dass externe Rechner TCP-Verbindungen mit einem internen Rechner aufbauen, erlaubt aber umgekehrt allen internen Clients Verbindungen nach außen.

# Beispiel für eine Filtertabelle einer Paketfilter-Firewall

| Aktion | Quelladresse | Quellport | Zieladresse | Zielport |
|---|---|---|---|---|
| erlauben | extern | > 1023 | intern | 25 |
| blockieren | extern | > 1023 | intern | !=25 |
| erlauben | intern | 25 | extern | > 1023 |
| blockieren | intern | !=25 | extern | > 1023 |

- Nur Anfragen bzgl. Mailverbindungen (Port 25) von externen Clients sowie alle entsprechenden Serverantworten werden zugelassen
- Anwendung der Regeln (übliches Verfahren)
  - Sequentielles zeilenweises Durchlaufen der Tabelle
  - Die Aktion in der ersten Zeile, in der alle Bedingungen erfüllt sind, wird ausgeführt!

# Einige Leitlinien zur Aufstellung von Filterregeln/-tabellen

- Möglichst frühe Filterung eingehender Pakete: Abwehr von Spoofing-Angriffen
- Reihenfolge der Regeln beachten (meist sequentielle Abarbeitung)!
- Blockieren von Paketen unbekannter Protokolle!
- Blockieren von Paketen problematischer Dienste bzw. Protokolle, z.B. tftp, sunrpc, rlogin, UDP-Pakete bei zustandslosen Filtern
- **Grundsätzlich**: **Alles sperren**, außer wohlbekannten und benötigten Protokollen/Diensten (→ Ports)

# Paketfilter: Was bringen sie?

- **Vorteile**
  - Zugriff auf Netzdienste geschieht völlig transparent
  - Die meisten Router unterstützen die Angabe von Filterregeln, so dass keine teure Zusatzhardware nötig ist
- **Nachteile**
  - Konfiguration kann sehr komplex werden
  - Spoofing kann nicht erkannt werden
  - Filterung von *unsicheren* Nutzdaten (z.B. Viren, sensible Daten in Emails, ...) und variablen Protokollparametern (z.B. RPC Portmapper) nicht zuverlässig möglich
- **Aber**: neuere Paketfilter-Konzepte können
  - **zustandsbehaftet** arbeiten (z.B. FTP-DATA nur nach FTP)
  - bei gewissen Ereignissen **dynamisch die Regeln verändern**
  - Konsequenz: **noch komplexer**

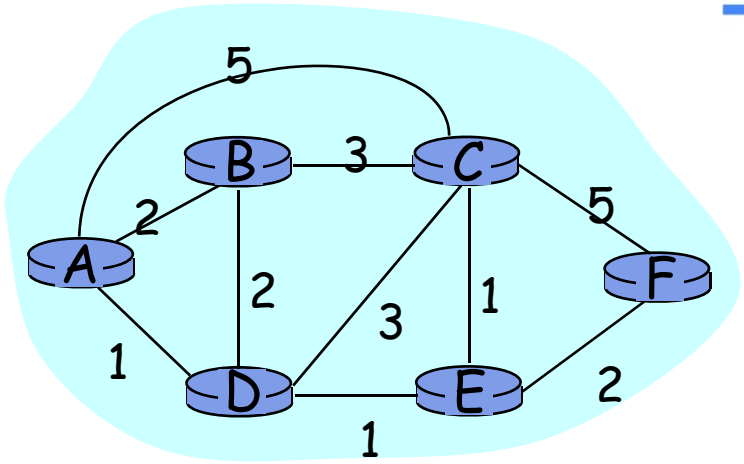
# Kapitel 4
## Netzwerkschicht & Routing

1. Einleitung und Netzwerkdienstmodelle
2. Aufbau eines Routers
3. Das Internet-Protokoll (IPv4)
4. Paketfilterung (Firewalls)
5. Routing-Algorithmen
6. Routing-Protokolle im Internet
7. NAT vs. IPv6
8. Mobile IP

# Routing

**Routingalgorithmus**

Ziel: finde "optimalen" Pfad (Folge von Routern) durch das Netzwerk



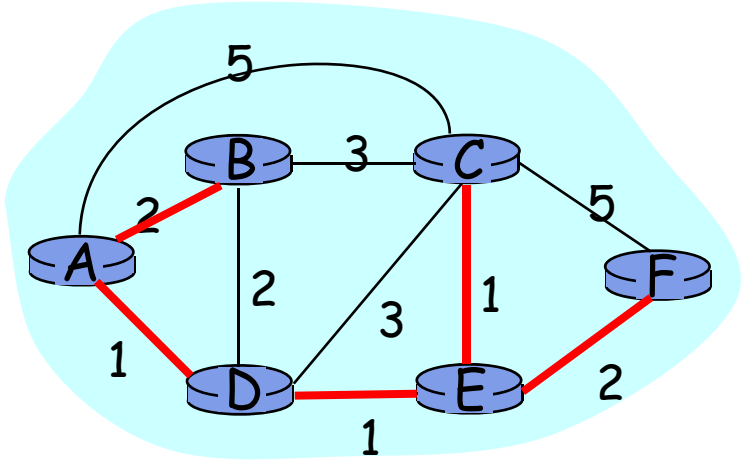

Graph-Abstraktion für Routingalgorithmen:

- Knoten im Graph sind Router
- Graph-Kanten sind die physikalischen Verbindungen ("Links")
  - Verbindungskosten: Verzögerung, € Kosten, Staugefahr, …

- "optimaler" Pfad:
  - heißt meist minimale Kosten
  - andere Definitionen sind möglich
  - Wichtig: Nur Kostenwerte $\geq 0$ sind erlaubt!

# Eigenschaft optimaler Pfade

- Wenn Router D auf dem optimalen Pfad r von Router A zu Router F liegt,
- dann ist der optimale Pfad von D zu F ein Teil von r
- Beweis?
- Folgerung:
  - Die optimalen Pfade von A zu allen möglichen Zielen bilden einen Baum mit Wurzel A



# Klassifizierung von Routing-Algorithmen

## Globale oder dezentrale Information?

Global:

- Alle Router kennen die komplette Topologie / Verbindungskosten
- “Link state”-Algorithmen

Dezentral:

- Jeder Router kennt die physikalisch direkt verbundenen Nachbarn mit den entsprechenden Verbindungskosten
- Iterativer Berechnungsprozess, Austausch der Information mit den direkten Nachbarn
- “Distanz-Vektor”-Algorithmen

## Statisch oder dynamisch?

Statisch:

- Routen ändern sich langsam mit der Zeit

Dynamisch:

- Routen ändern sich häufig
  - periodische Updates
  - in Reaktion auf die Änderung der Verbindungskosten

# Ein Link-State Routing Algorithmus

## Dijkstra's Algorithmus

- Netz-Topologie, Verbindungkosten in allen Knoten bekannt
  - Erreicht durch "Link state broadcast": Rundsenden der Identitäts- und Kosteninformationen
  - Alle Knoten haben die gleiche Information
- Berechnet die kürzesten Pfade von einem Knoten ('Quelle') zu allen anderen
  - Ergibt Routing-Tabelle für diesen Knoten

## Notation:

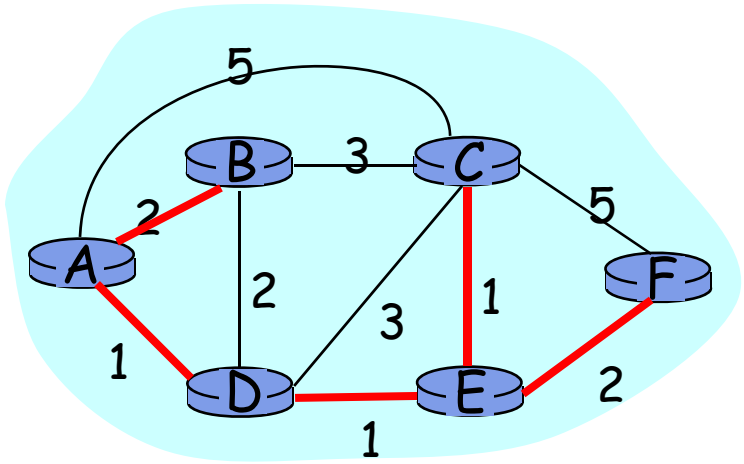
- A: Quelle
- c(i,j): Verbindungskosten von Knoten i nach j, Kosten ∞, wenn nicht direkter Nachbar
- D(v): Aktueller minimaler Wert der Pfadkosten von der Quelle zu Knoten v
- p(v): Vorgängerknoten von v auf dem momentan besten Weg von der Quelle nach v
- N: Menge der Knoten, für die die minimalen Pfadkosten bereits feststehen (fertig!)

# Dijkstra's Algorithmus

```
1 /* Initialisierung: */
2    N = {A}
3    for alle Knoten v ∉ N
4      if v direkter Nachbar von A
5         then D(v) = c(A,v)
6         else D(v) = ∞
7  Loop
8    Finde den Knoten w ∉ N mit: D(w) ist minimal
9    Füge w zu N hinzu /*Kürzester Pfad zu w steht fest*/
10      for alle Knoten v ∉ N ,
                  die direkte Nachbarn von w sind:
11         D(v) = min( D(v), D(w) + c(w,v) )
12    /* Die neuen Kosten der direkten Nachbarn v sind
      entweder die alten Kosten oder die bekannten
      Kosten des kürzesten Pfades zu w zuzüglich der
      Kosten von w zu v */
14 until alle Knoten sind in N
```

# Dijkstra's Algorithmus: Beispiel

| Iteration | N | *B* D(B),p(B) | *C* D(C),p(C) | *D* D(D),p(D) | *E* D(E),p(E) | *F* D(F),p(F) |
|---|---|---|---|---|---|---|
| → 0 | A | 2,A | 5,A | 1,A | ∞ | ∞ |
| → 1 | AD | 2,A | 4,D | | 2,D | ∞ |
| → 2 | ADE | 2,A | 3,E | | | 4,E |
| → 3 | ADEB | | 3,E | | | 4,E |
| → 4 | ADEBC | | | | | 4,E |
| → 5 | ADEBCF | | | | | |



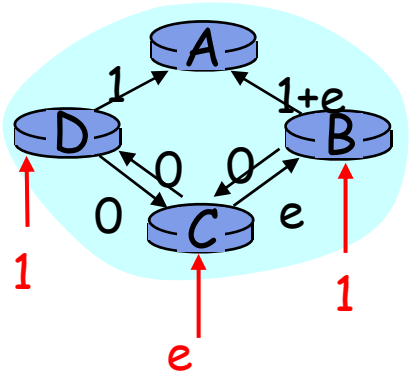
# Dijkstra's Algorithmus: Diskussion

Komplexität des Algorithus: n Knoten

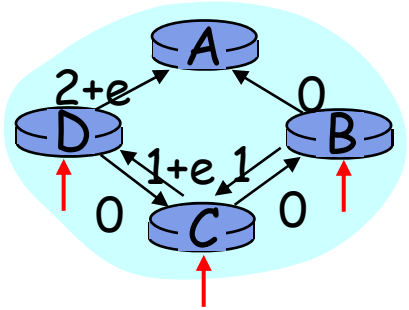
- jede Iteration: alle Knoten, die nicht in N sind, müssen geprüft werden
- n*(n+1)/2 Vergleiche: O(n**2)
- effektivere Implementierungen erreichen: O(n * log n)

Oszillationen sind möglich:

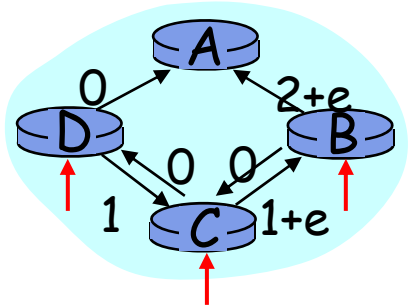
- z.B. wenn Verbindungskosten = Höhe der aktuellen Verkehrslast auf der Verbindung



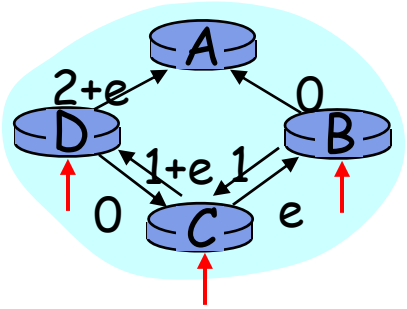

Anfangsrouting
→ B,C,D erzeugen
Last für A




B,C erkennen
besseren
Pfad zu A
im Uhrzeigersinn




B,C,D erkennen
besseren
Pfad zu A
entgegen dem
Uhrzeigersinn




B,C,D erkennen
besseren
Pfad zu A
im Uhrzeigersinn

# Distanz-Vektor Routing: Überblick

Verteilt:

- jeder Knoten kommuniziert *nur* mit seinen *direkten* Nachbarn

Iterativ:

- stoppt, wenn kein Knoten mehr Infos austauscht
- *Selbstterminierend:* kein “Stop”-Signal notwendig

Asynchron:

- Austausch muss nicht synchron getaktet sein!

# Distanz-Vektor Routing Algorithmus

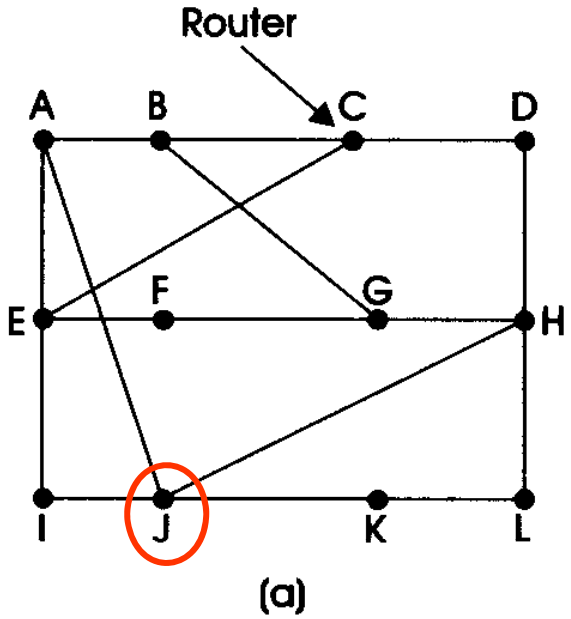
## Datenstrukturen pro Knoten

- Liste mit aktuellen Kosten zu jedem direkten Nachbarn (direkte Verbindungskosten) → laufende Messung
- "Distanz-Tabelle" mit den minimalen Kosten aller direkten Nachbarn zu allen möglichen Zielen
  - eine Zeile für jedes mögliche Ziel
  - eine Spalte für die minimalen Kosten jedes direkten Nachbarn des Knoten
- Routing-Tabelle
  - Ziel
  - Kosten
  - Ausgangsleitung

## Update der Routing-Tabelle

1. Berechne für jedes Ziel und jeden direkten Nachbarn:
   Aktuelle Kosten zum Nachbarn + dessen minimale Kosten zum Ziel
2. Minimale Kosten zu Ziel = Minimum aus Schritt 1
   → Wähle entsprechenden Nachbarn als Ausgangsleitung!

# Distanz-Vektor Routing: Beispiel




(a)

Minimale Kosten von J zu Ziel

| Ziel | A | I | H | K | Leitung |
|---|---|---|---|---|---|
| A | 0 | 24 | 20 | 21 | 8 A |
| B | 12 | 36 | 31 | 28 | 20 A |
| C | 25 | 18 | 19 | 36 | 28 I |
| D | 40 | 27 | 8 | 24 | 20 H |
| E | 14 | 7 | 30 | 22 | 17 I |
| F | 23 | 20 | 19 | 40 | 30 I |
| G | 18 | 31 | 6 | 31 | 18 H |
| H | 17 | 20 | 0 | 19 | 12 H |
| I | 21 | 0 | 14 | 22 | 10 I |
| J | 9 | 11 | 7 | 10 | 0 - |
| K | 24 | 22 | 22 | 0 | 6 K |
| L | 29 | 33 | 9 | 9 | 15 K |
| | JA Zeit ist | JI Zeit ist | JH Zeit ist | JK Zeit ist | |
| Aktuelle Kosten zu den direkten Nachbarn (direkte Verbindungskosten) → | 8 | 10 | 12 | 6 | |

Empfangene Vektoren von J's vier Nachbarn

Distanztabelle von J

Neue Routing-Tabelle für J

[AT]

# Distanz-Vektor Routing: Anwendung

Iterativ, asynchron:

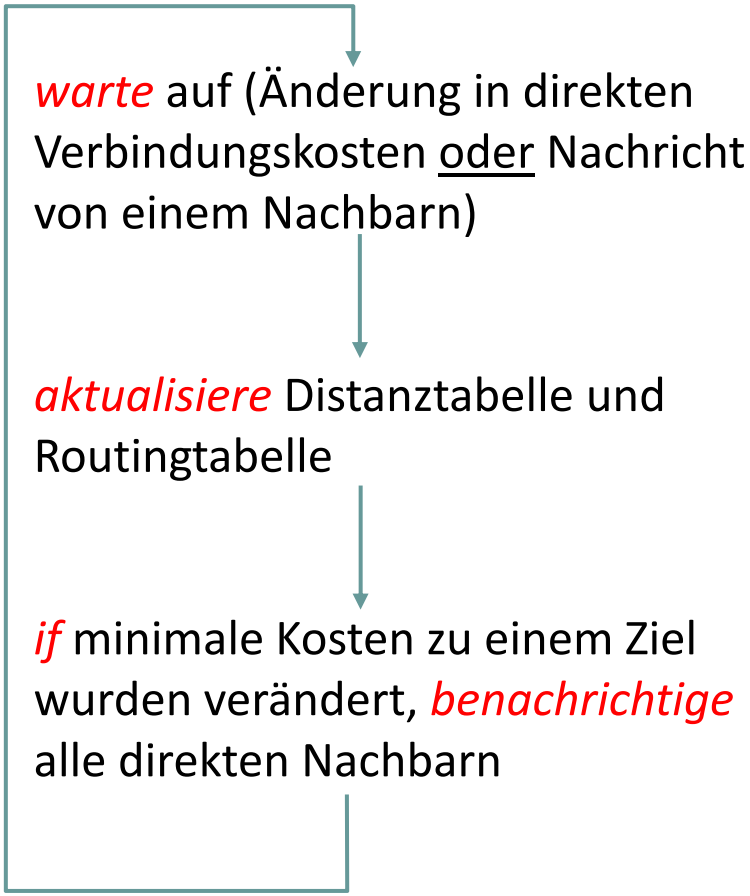
jede lokale Iteration wird verursacht durch:

- Änderung der direkten Verbindungskosten oder
- Nachricht vom Nachbarn: seine Pfade der minimalen Kosten haben sich geändert

Verteilt:

- Jeder Knoten benachrichtigt die Nachbarn nur, wenn sich ein Pfad mit minimalen Kosten geändert hat
  - Die Nachbarn benachrichtigen dann ihre Nachbarn, wenn notwendig

Jeder Knoten:



# Distanz-Vektor Routing: Eigenschaften

- Was passiert, wenn ein neuer Knoten (A) auftaucht?


| | | | | |
|---|---|---|---|---|
| ∞ | ∞ | ∞ | ∞ | Anfangs |
| 1 | ∞ | ∞ | ∞ | Nach einem Austausch |
| 1 | 2 | ∞ | ∞ | Nach 2 Austauschen |
| 1 | 2 | 3 | ∞ | Nach 3 Austauschen |
| 1 | 2 | 3 | 4 | Nach 4 Austauschen |

Eintrag in jeweilige Distanztabelle: Kosten für den Weg nach A

- Gute Nachrichten verbreiten sich schnell!

# Distanz-Vektor Routing: Problem

- Was passiert, wenn ein Knoten (A) ausfällt?


| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | Anfangs |
| 3 | 2 | 3 | 4 | Nach einem Austausch |
| 3 | 4 | 3 | 4 | Nach 2 Austauschen |
| 5 | 4 | 5 | 4 | Nach 3 Austauschen |
| 5 | 6 | 5 | 6 | Nach 4 Austauschen |
| 7 | 6 | 7 | 6 | Nach 5 Austauschen |
| 7 | 8 | 7 | 8 | Nach 6 Austauschen |
| | ... | | | |
| ∞ | ∞ | ∞ | ∞ | |

Eintrag in jeweilige Distanztabelle: Kosten für den Weg nach A

**Count-to-Infinity-Problem**

- Nicht generell lösbar!

# Kapitel 4
## Netzwerkschicht & Routing

1. Einleitung und Netzwerkdienstmodelle
2. Aufbau eines Routers
3. Das Internet-Protokoll (IPv4)
4. Paketfilterung (Firewalls)
5. Routing-Algorithmen
6. Routing-Protokolle im Internet
7. NAT vs. IPv6
8. Mobile IP

# Routing: Praxisprobleme

Unsere Routing-Diskussion war bisher vereinfacht → Idealisierung

- Alle Router waren identisch
- Netzwerk war “flach”

*… nicht Realität !*

Größe: mit > 100 Millionen Zieladressen:

- Nicht alle Ziele können in einer Routing-Tabelle gespeichert werden!
- Austausch von Routing-Informationen würde das Netz überlasten!

Administrative Autonomie

- Internet = Netzwerk von Netzwerken
- Jeder Netzwerk-Admin möchte Routing im eigenen Subnetz beeinflussen können

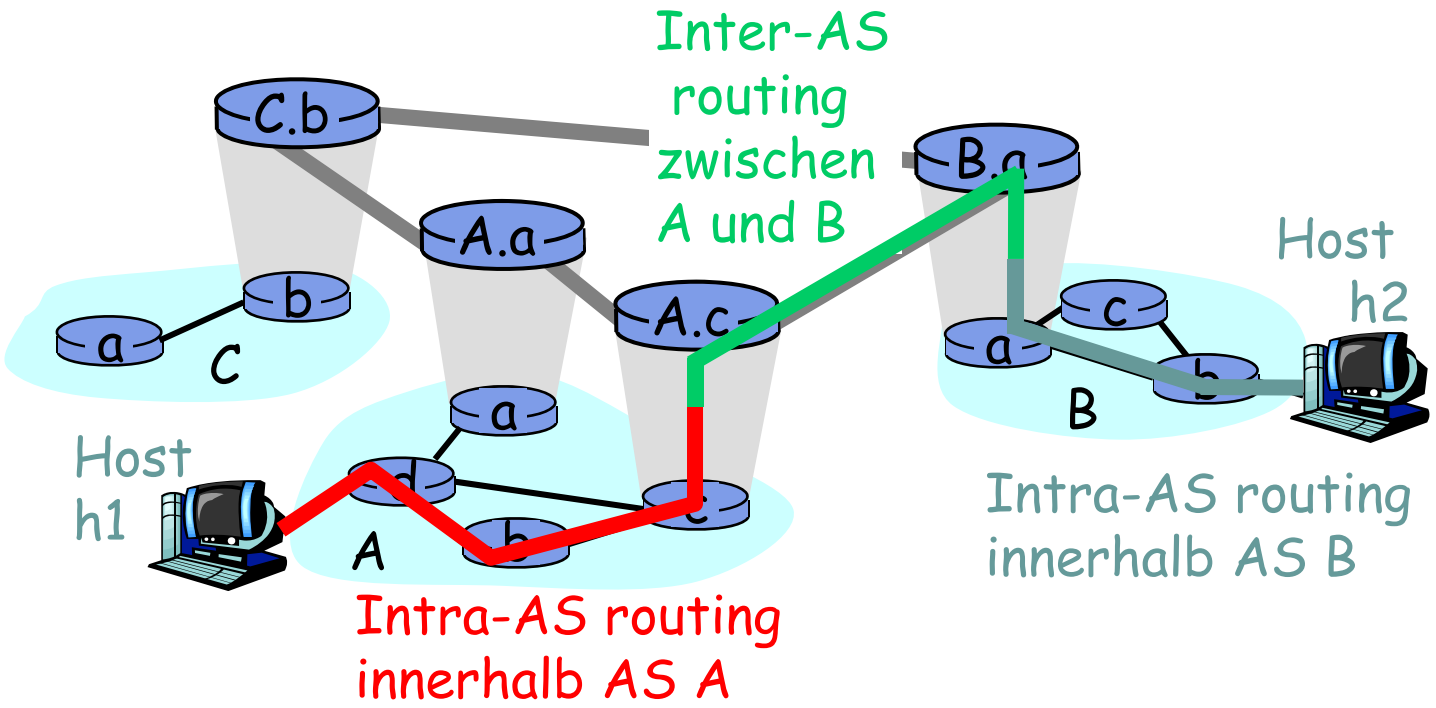
# Hierarchisches Routing: Idee

- Zusammenschluss von Routern zu Regionen "Autonome Systeme" (AS)
- Router in demselben AS benutzen das gleiche Routing-Protokoll
  - "Intra-AS"-Routing-Protokoll
  - Router in unterschiedlichen AS können unterschiedliche Intra-AS-Routing-Protokolle benutzen

## Gateway Router

- Spezielle Router im AS
- "Sprechen" Intra-AS-Routing-Protokoll mit allen anderen Routern im AS
- zusätzlich verantwortlich für Routing zu Zielen außerhalb des AS
  - Nutzen *Inter-AS-Routing-*Protokoll mit anderen Gateway-Routern

# Intra-AS und Inter-AS Routing




- Wir schauen uns spezifische Inter- und Intra-AS- Routing-Protokolle im Internet an

# Intra-AS Routing

- Auch genannt Interior Gateway Protocols (IGP)
- Wichtigste Protokolle:
  - RIP: Routing Information Protocol (RFC 1058)
  - OSPF: Open Shortest Path First (RFC 2178)
  - IGRP: Interior Gateway Routing Protocol
    (proprietäres Cisco-Protokoll)

# Beispiel: Routingtabelle eines PC mit 3 Netzwerkschnittstellen (Auszug)

| Destination | Gateway | Interface |
|---|---|---|
| 127.0.0.1 | 127.0.0.1 | lo |
| 192.168.2. | 192.168.2.5 | fa0 |
| 193.55.114. | 193.55.114.6 | le0 |
| 192.168.3. | 192.168.3.5 | qaa0 |
| 224.0.0.0 | 193.55.114.6 | le0 |
| default | 193.55.114.129 | le0 |

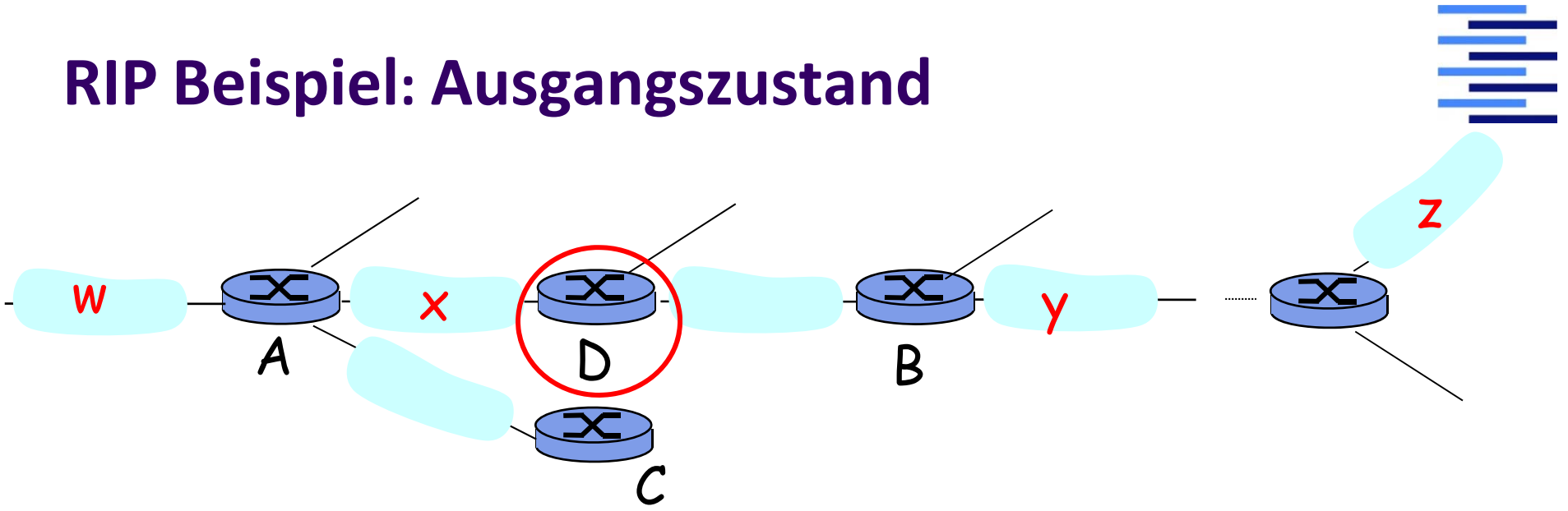
- Drei angeschlossene Klasse-C-Netzwerke (LANs) im AS
- Der Default-Router 193.55.114.129 wird benutzt für alle Zielnetze, die nicht angeschlossen sind (im AS liegen) → Weg “nach oben”
- Multicast-Adresse: 224.0.0.0
- Loopback-Interface 127.0.0.1: Gesendetes Paket wird sofort zurückgeschleift (wird wie ein angekommenes Paket behandelt)

# RIP (Routing Information Protocol)

- Distanzvektor-Algorithmus
- Pfadkosten: Anzahl Hops
  - Kosten jeder Verbindung = 1
  - Maximal 15 Hops erlaubt! *(Warum??)*
- Routinginformationen werden regelmäßig alle 30 Sekunden mittels einer “RIP Response Message” (auch RIP-Advertisement genannt) ausgetauscht
- Jedes Advertisement enthält die Routingtabelle des Senders für bis zu 25 Zielnetzwerke innerhalb des AS

- Enthalten in Standard-UNIX Distributionen
- CIDR-Unterstützung erst ab Version 2
- Wird in Routern nur noch selten benutzt

# RIP Beispiel: Ausgangszustand




**Routingtabelle von D**

| Destination Network | Next Router | Num. of hops to dest. |
|---|---|---|
| w | A | 1 |
| y | B | 1 |
| z | B | 6 |
| x | D | 0 |
| …. | …. | …. |

# RIP Beispiel: Advertisement von A erreicht D

**Advertisement = Routingtabelle von A**

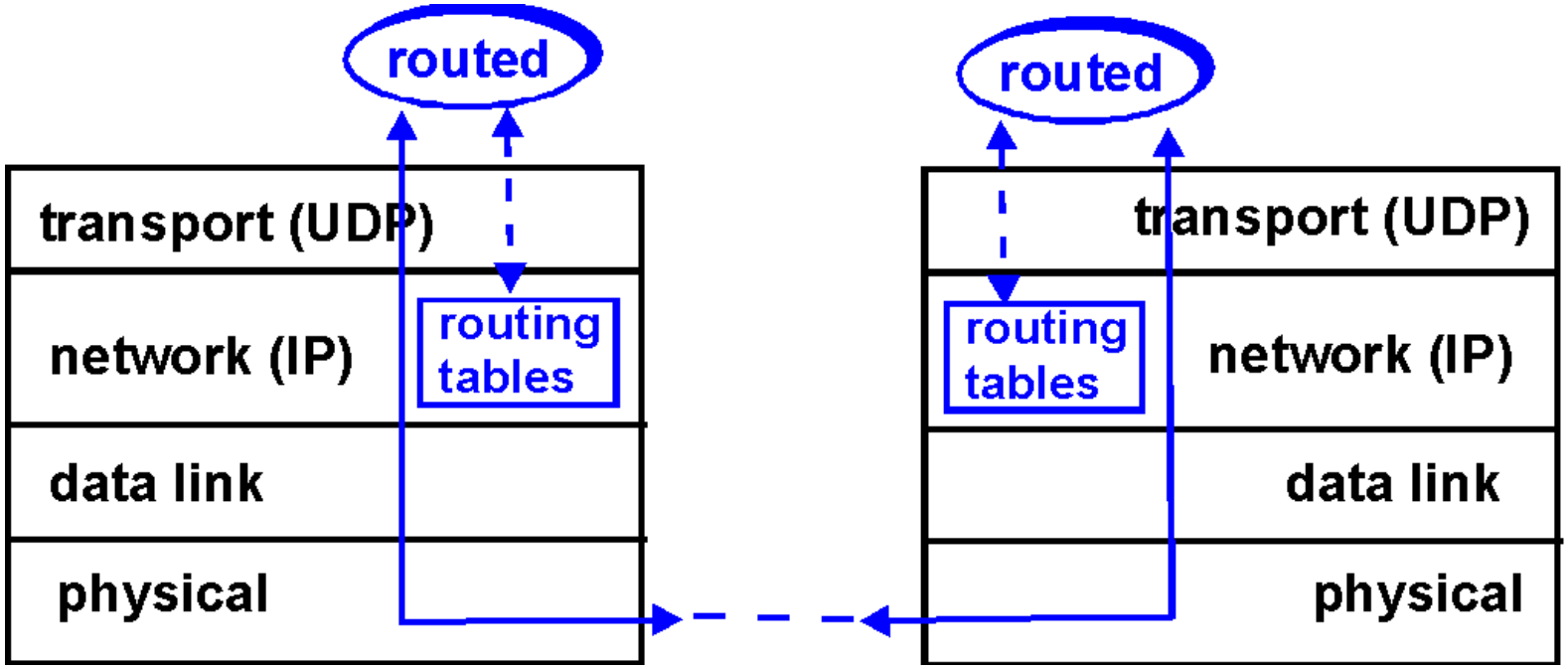
| Destination Network | Next Router | Num. of hops to dest. |
|---|---|---|
| z | C | 3 |
| w | A | 0 |
| x | A | 0 |
| …. | …. | …. |

**→ Neue Routingtabelle von D**

| Destination Network | Next Router | Num. of hops to dest. |
|---|---|---|
| w | A | 1 |
| y | B | 1 |
| z | A | 4 |
| x | D | 0 |
| …. | …. | …. |

# RIP-Implementierung in UNIX

- RIP ist in Unix als ein Hintergrund-Anwendungsprozess namens routed implementiert (“route-daemon”)
- Der routed-Prozess darf trotzdem die Routingtabellen im Kernel aktualisieren!
- Advertisements werden in UDP-Paketen versendet



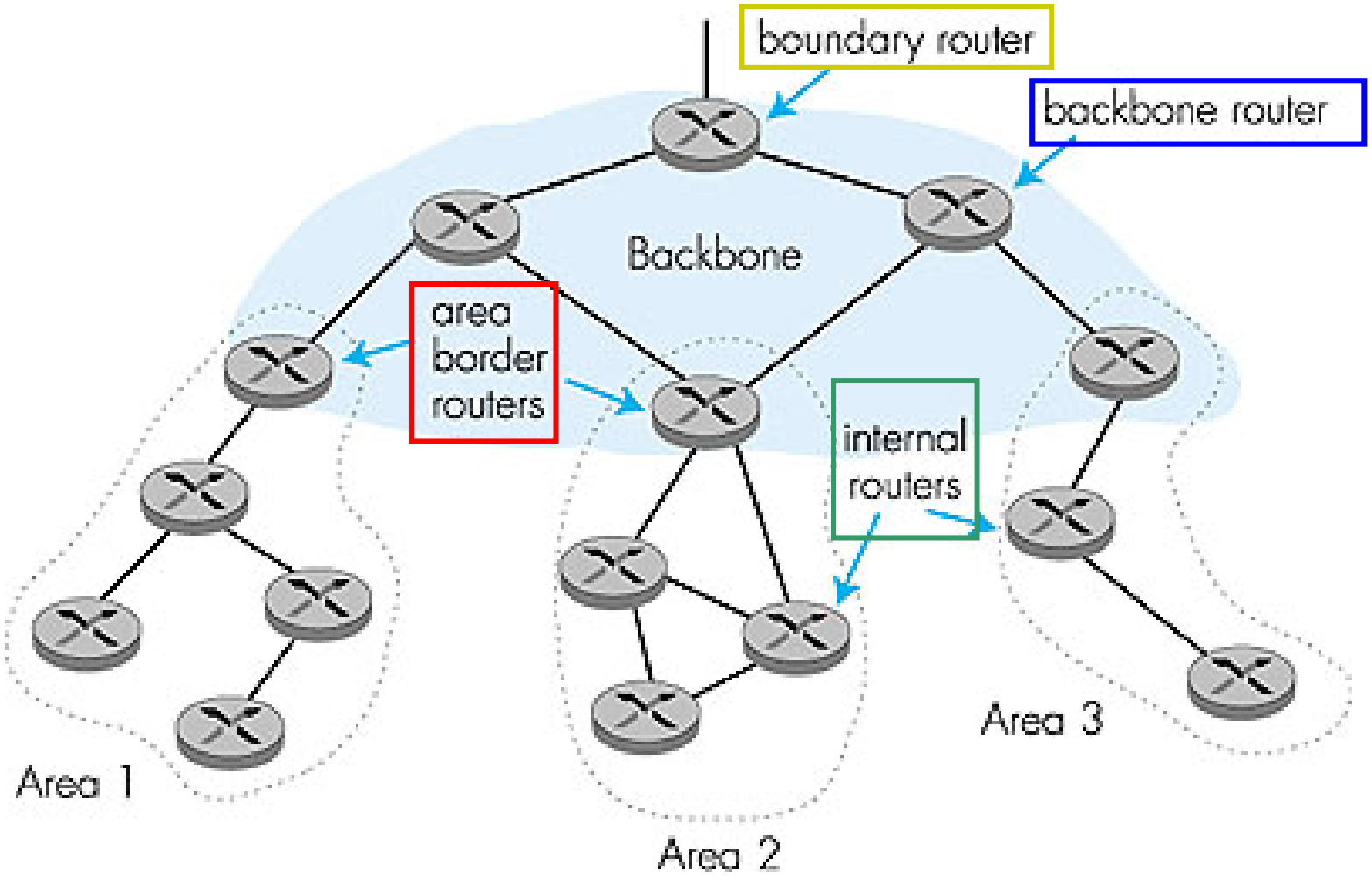
# OSPF (Open Shortest Path First)

- "Offener" Standard seit 1990 (RFC 2178)
- Nachfolger von RIP
- Benutzt "Link State" Algorithmus
  - Zustandsinformationen werden an alle Router über periodische "Link state broadcast" – Nachrichten verbreitet
  - Ein OSPF-Advertisement enthält nur die aktuellen Verbindungskosten zu den direkten Nachbarn
  - Jeder Router kennt die gesamte Topologie des AS inkl. Verbindungskosten
  - Routenberechnung gemäß Dijkstra-Algorithmus

# OSPF: Zusätzliche Eigenschaften

- Sicherheit: Alle ausgetauschten OSPF-Nachrichten werden authentifiziert und über TCP-Verbindungen gesendet
- Mehrere Pfade mit denselben Kosten können parallel verwendet werden (in RIP nur ein Pfad) → *Lastverteilung!*
- Für eine Verbindung zwischen zwei Routern können verschiedene Verbindungskosten in Abhängigkeit vom TOS-Wert (Type of Service) im IP-Datagramm definiert werden
- Integrierte Unterstützung von Unicast- und Multicast-Routing
- Hierarchische Strukturierung großer Autonomer Systeme
  - Aufteilung in Bereiche (“Areas”), innerhalb derer ein eigener Link-State-Algorithmus angewendet wird

# OSPF-Hierarchie: Beispiel



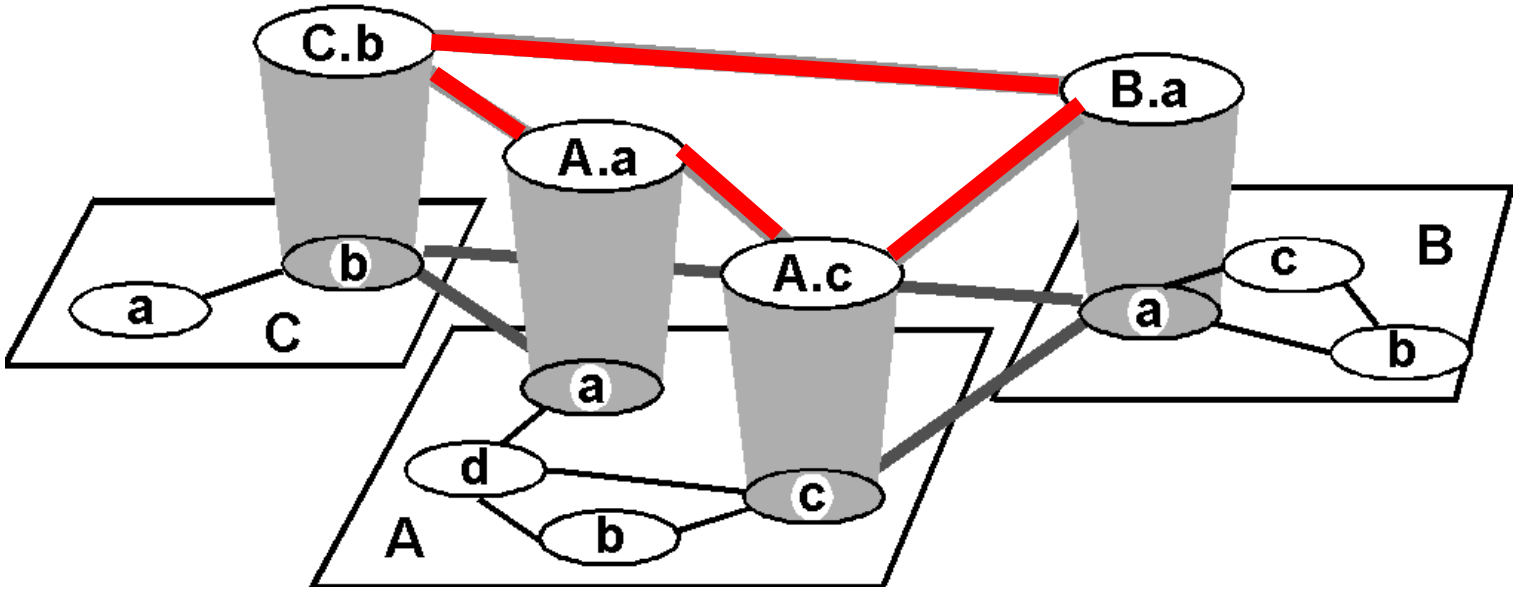
# OSPF-Hierarchie

- Zwei-Ebenen-Hierarchie: **Mehrere "lokale" Bereiche (Areas) und zusätzlich ein "Backbone"-Bereich**
  - Link-State-Informationsaustausch nur innerhalb eines Bereichs
  - Der Backbone-Bereich dient nur zur Weiterleitung zwischen den lokalen Bereichen oder für den Verkehr nach "Außen"
- Typen von OSPF-Routern:
  - **Interne Router**: Router für den Intra-AS-Verkehr innerhalb eines lokalen Bereichs
  - **Area Border Router:** Gehören zu einem lokalen Bereich und zum Backbone → dienen als Gateway für den bereichsübergreifenden Verkehr
  - **Backbone Router:** Gehören zum Backbone und führen das Routing innerhalb des Backbone durch
  - **Boundary Router:** Gehören zum Backbone und sind mit Routern aus anderen AS verbunden → dienen als Gateway für den externen Verkehr mit anderen autonomen Systemen

# IGRP (Interior Gateway Routing Protocol)

- Proprietäres CISCO-Protokoll
- Ebenfalls als RIP-Nachfolger entwickelt
- Distanz-Vektor-Protokoll (wie RIP)
- Konfigurierbare Kostenmetriken (z.B. Verzögerung, Übertragungskapazität, Verfügbarkeit, Last, ..)
- Verwendet  TCP statt UDP
- Updateinformationen werden nur bei Veränderungen ausgetauscht (nicht periodisch wie bei RIP)
- Berechnung schleifenfreier Routingpfade aufgrund des “Distributed Updating Algorithmus” (DUAL)

# Inter-AS-Routing



# Inter-AS-Routing im Internet: BGP

- BGP (Border Gateway Protocol):
  *Der* de-facto-Standard
- Keine Routenberechnung (Vorgabe von Pfaden durch Administratoren)
- Hauptfunktion: Verteilung von Pfadinformationen
  - ➔ "Pfad-Vektor" - Protokoll:
  - Algorithmus ähnlich dem Distanz-Vektor-Protokoll
  - Jedes Gateway (Boundary-Router) versendet seine Informationen über den kompletten Pfad (Folge von AS) zu einem Ziel an alle seine direkten Nachbarn ("Peers")

# BGP: Beispiel

- *Annahme:* Gateway X sendet seinen Pfad $X,Y_1,Y_2,Y_3,\ldots,Z$ an Peer-Gateway W
- W kann den angebotenen Pfad ignorieren!
  Mögliche Gründe:
  - Günstigerer Pfad vorhanden, Schleifenbildung vermeiden,
    AS $Y_2$ soll vermieden werden, …
- Wenn W den Pfad akzeptiert, fügt er einen neuen Eintrag in seiner Routingtabelle hinzu:

$$\text{Pfad } (W,Z) = W, X,Y_1,Y_2,Y_3,\ldots,Z$$

- Bemerkung: X kann den eingehenden Verkehr mittels der Pfadinformationen, die er an seine Peers weitergibt, steuern!
  - Bsp.: Verkehr nach Z geht nicht über X, wenn X keine Pfade X,…,Z an seine Peers weitergibt!

# BGP-Nachrichten

- BGP-Nachrichten werden über TCP (Port 179) ausgetauscht!
- Nachrichtentypen:
  - OPEN: Öffnet eine TCP-Verbindung zum Peer und authentifiziert den Sender
  - UPDATE: Aktualisiert eine Pfadinformation
  - KEEPALIVE Hält die TCP-Verbindung offen, falls keine Updateinformationen vorliegen
  - NOTIFICATION: Fehlermeldung oder Anzeige des Verbindungsendes

# Warum gibt es unterschiedliche Intra-und Inter-AS-Routing-Protokolle?

## Steuerung:

- Inter-AS: Gezielte Steuerung des Verkehrs nötig (Kosten, Sicherheit, Verfügbarkeit, Politik, …)
- Intra-AS: Einheitlicher Administrationsbereich, keine Notwendigkeit der Abgrenzung

## Skalierung:

- Eine Aufteilung in überschaubare Bereiche ist wichtig für die Anwendbarkeit der Routingalgorithmen
  (→ Hierarchisches Routing)

## Leistung:

- Intra-AS: Leistungsoptimierung i.d.R. oberstes Ziel
- Inter-AS: Steuerungsaspekte teilweise wichtiger als Leistung

# Kapitel 4
## Netzwerkschicht & Routing

1. Einleitung und Netzwerkdienstmodelle
2. Aufbau eines Routers
3. Das Internet-Protokoll (IPv4)
4. Paketfilterung (Firewalls)
5. Routing-Algorithmen
6. Routing-Protokolle im Internet
7. NAT vs. IPv6
8. Mobile IP

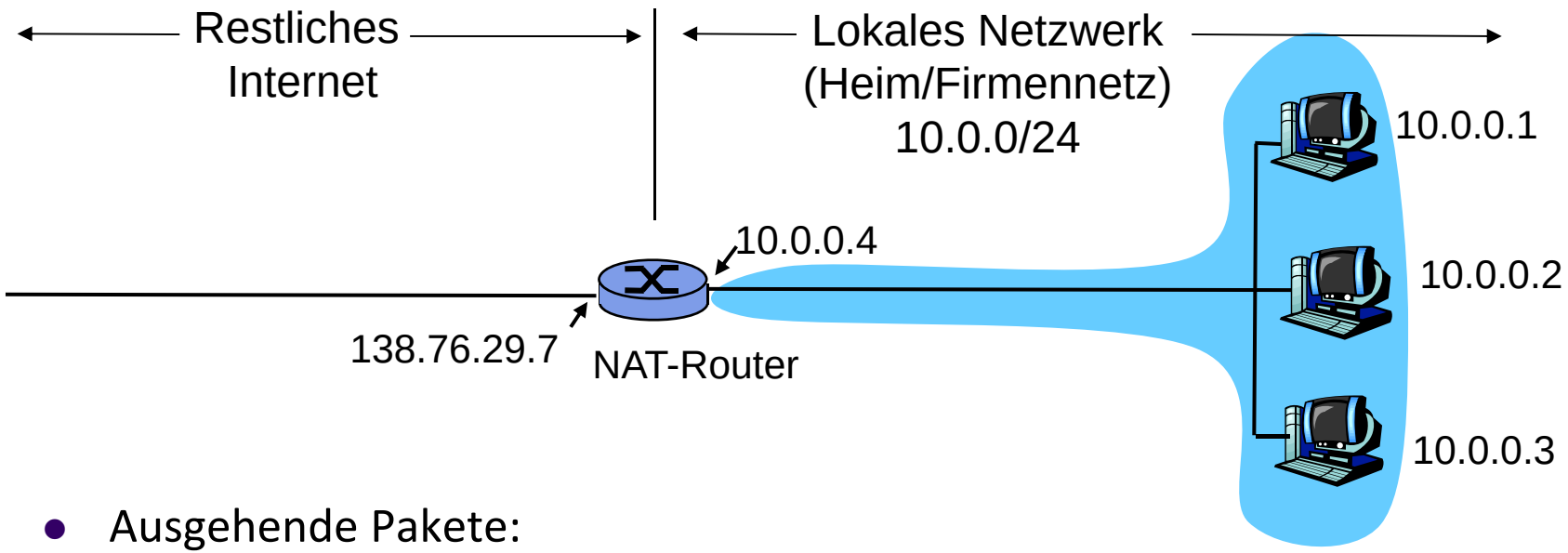
# Problemstellung

- Der IPv4-Adressraum (32 Bit) ist zu klein, um allen Internet-Hosts eine weltweit eindeutige IP-Adresse zu geben!
- Lösungswege:
  - CIDR √
  - DHCP √
  - **NAT** ("Network Address Translation"): Lösung für private Netze
  - **IPv6**: Einzige langfristige Lösung

# NAT (“Network Address Translation”)
## [RFC 3022]

- Folgende Adressen werden im Internet nicht geroutet:
  - 10.0.0.0/8 – 10.255.255.255/8
  - 172.16.0.0/12 – 172.31.255.255/12
  - 192.168.0.0/16 – 192.168.255.255/16
- Benutzung dieser „privaten“ IP-Adressen im Intranet (Heim- oder Firmennetz)
- Umsetzung der virtuellen IP-Adresse in eine „öffentliche“ IP-Adresse, wenn eine Verbindung zum Internet nötig ist (im NAT-Router)

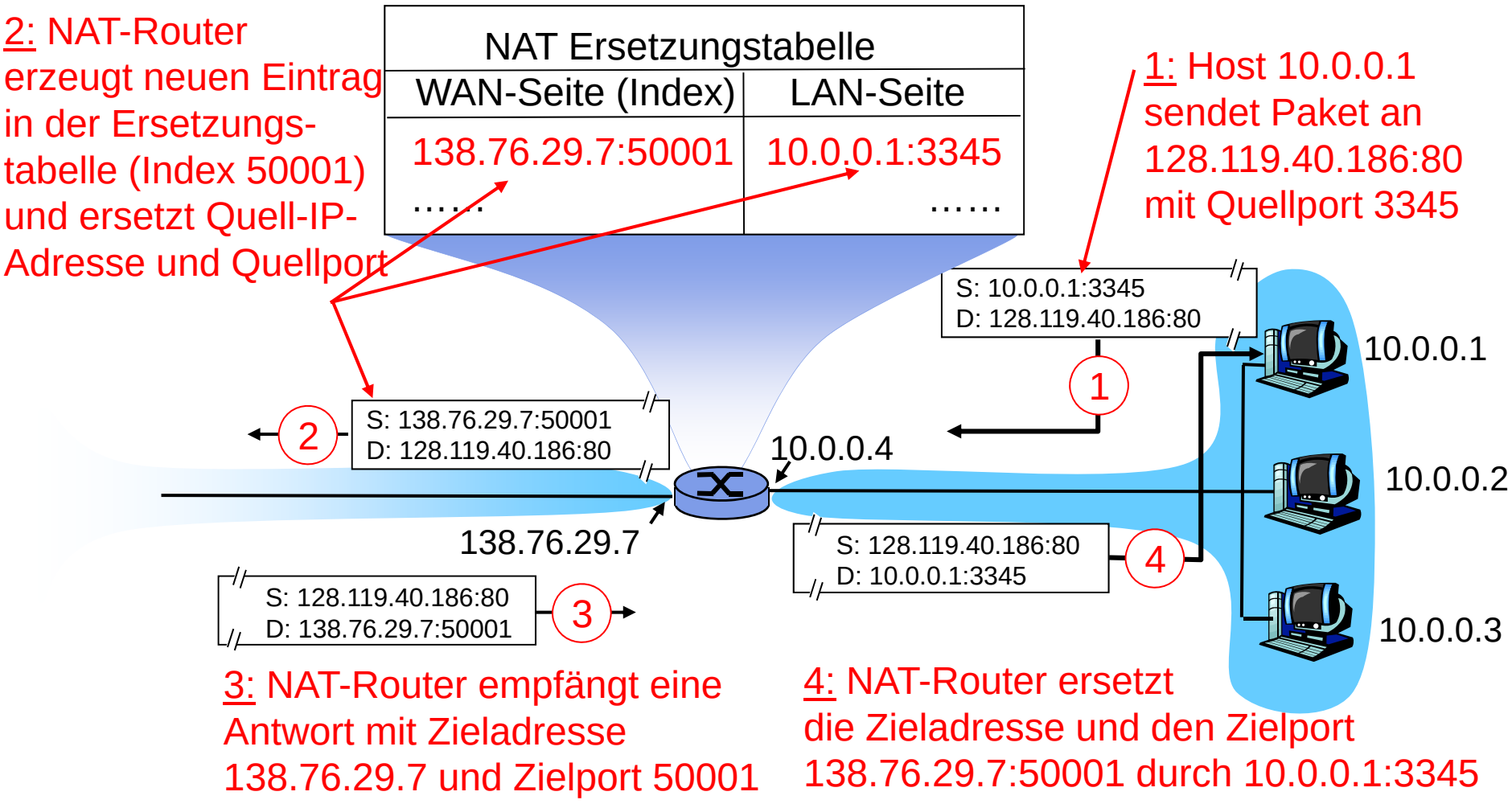
# NAT - Realisierung




- Ausgehende Pakete:
  - NAT-Router speichert private IP-Adresse und originalen TCP/UDP-Quellport in einer Ersetzungstabelle unter einem neu erzeugten Index
  - NAT-Router ersetzt im Paket Quell-IP-Adresse und TCP/UDP-Quellport

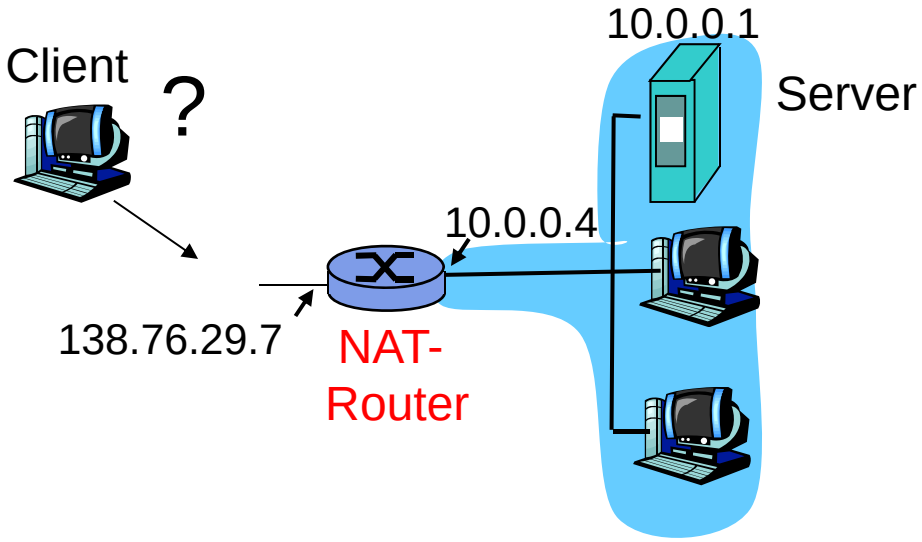
    Quell-IP-Adresse → Öffentliche IP-Adresse des NAT-Routers (hier 138.76.29.7)
    TCP/UDP Quellport → Index des neuen Ersetzungstabelleneintrags
- Eingehende Pakete (Antworten):
  - NAT-Router ersetzt Ziel-IP-Adresse (NAT-Routeradresse) und TCP/UDP-Zielport (Index) anhand der Ersetzungstabelle durch die gespeicherten Werte

# NAT - Beispiel



# NAT - Probleme

- 💣 Unzulässige Vermischung der Protokollschichten!
- Wie erreichen Client-Anfragen Server, die hinter einem NAT-Router in einem privaten Netz betrieben werden??

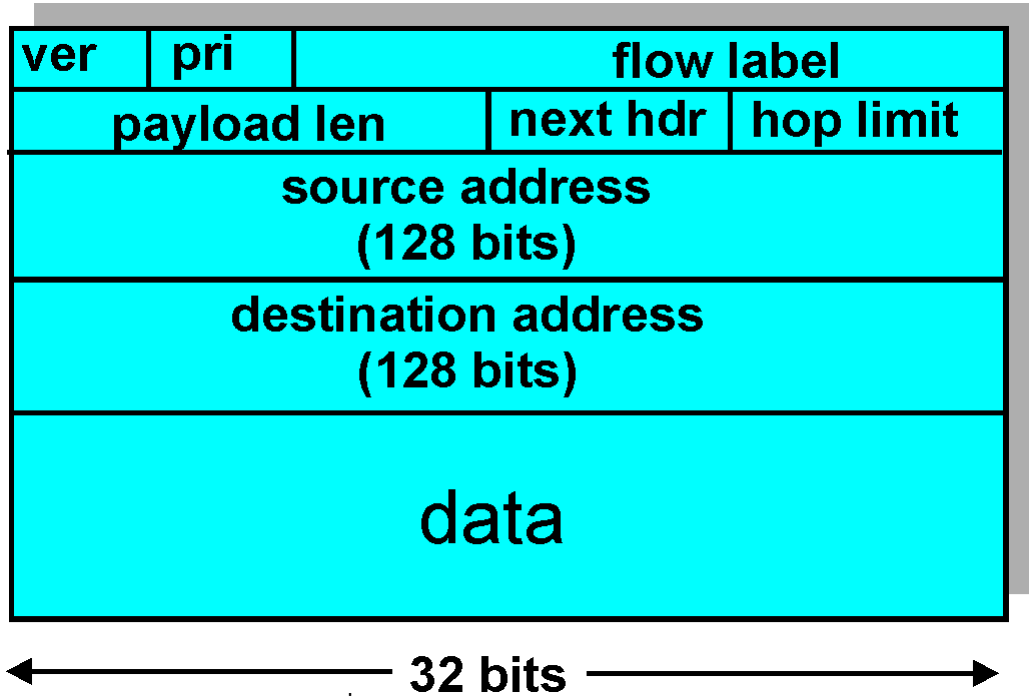


# IPv6 [RFC 2460 -2466]

- Designziele:
  - Vergabe einer weltweit eindeutigen Adresse an Milliarden von Rechnern (auch bei “Verschnitt”)
  - Routingtabellenverkleinerung
  - Vereinfachung des IP-Protokolls
  - Sicherheitsmechanismen
  - Verkehrsklassen über Prioritätenangabe (“Quality of Service”)
  - Intelligentes Multicasting für definierte Gruppen
  - Unterstützung mobiler Endgeräte
  - Zukünftige Erweiterungsmöglichkeiten
  - Koexistenz von IPv4 und IPv6 für Jahre

# IPv6-Header

*ver:* IP-Versionsnummer
*pri:* Verkehrsklasse (Priorität innerhalb eines “Flows”)
*flow label:* ID eines Flows
*payload len:* Länge des Nutzdatenfelds (ohne IPv6-Header)
*next hdr:* Code für die ersten Bytes des Nutzdatenfeldes: z.B. TCP/UDP-Header oder weiterer optionaler IPv6-Header
*hop limit:* analog TTL bei IPv4




*IPv6 Datagramm-Format:*

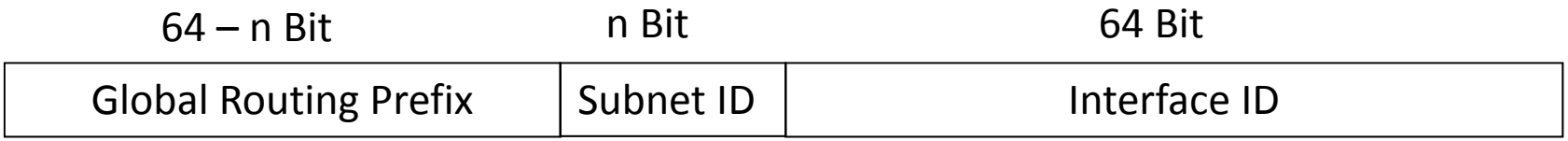
- Header hat eine feste Länge von 40 Byte
- Fragmentierung von Datagrammen nur durch den Sender erlaubt!

# IPv6-Adressen (I)

- Länge: 128 Bit (16 Byte)
  - Ergibt insgesamt ca. $3*10^{38}$ Adressen oder $7*10^{23}$ Adressen pro Quadratmeter (auf der gesamten Erde)
- Format:
  - Acht Gruppen von jeweils 4 Hex-Ziffern (mit je 4 Bit)
  - durch Doppelpunkt getrennt
  - Beispiel: 8000:0000:0000:0000:0123:4567:89AB:CDEF
  - Führende Nullen können weggelassen werden, eine oder mehrere aufeinander folgende Gruppen mit 16 Null-Bits können einmal durch :: ersetzt werden: 8000::123:4567:89AB:CDEF

# IPv6-Adressen (II)

- Typen:
  - **Unicast**: „normale“ Adresse eines Interfaces
  - **Multicast**: Gruppenadresse mit Zustellung an alle Mitglieder
  - **Anycast**: Gruppenadresse mit Zustellung an das „nächste“ Mitglied
- Struktur einer Unicast-Adresse:




  - Global Routing Prefix: ~ IPv4 Netzwerk-Teil
  - Subnet ID: Teilnetz innerhalb eines autonomen (End-)Netzes
  - Interface ID: ~ IPv4 Host-Teil, muss innerhalb des Subnetzes eindeutig sein!

# IPv6-Adressen (III)

- Datenschutz-Problem: Jedes Endgerät hinterlässt aufgrund der eindeutigen IP-Adresse nun eine "Spur" im Internet *("War NAT vielleicht doch besser ..?")*
- Lösung: "**Privacy Extensions**" (RFC 4941)
  - Nutzt aus, dass einem Interface **mehrere** IPv6-Adressen zugeordnet werden können
  - Verwendet eine eindeutige Netzwerkadresse (Prefix + Subnet-ID) und Interface-ID nur zum Empfangen von IPv6-Paketen
  - Nur zum Senden soll (i.a. täglich) vom ISP eine temporäre Netzwerkadresse zugewiesen werden, eine temporäre Interface-ID wird dann jeweils zufällig generiert

# Weitere Änderungen gegenüber IPv4

- *Checksumme*: komplett entfernt, um die Verarbeitungsgeschwindigkeit zu erhöhen
- *ICMPv6:* neue Version von ICMP
  - zusätzliche Nachrichtentypen, z.B. “Packet Too Big”
  - Multicast-Gruppenverwaltungsfunktionen
- *Optionen:*
  - erlaubt, aber nur außerhalb des festen 40 Byte-IPv6-Headers
  - Optionale Header stehen vor TCP/UDP-Header, falls benötigt
  - angezeigt durch Zahl im “Next Header”-Feld
    → “Header-Chaining”
  - Beispiel:

| **IPv6** header Next=43(Routing) | **Routing** header Next=44(Fragment) | **Fragment** header Next=6(TCP) | **TCP segment** |
|---|---|---|---|

# IPv6 Basis-Optionsheader

- *Hop-by-Hop Option (0)*
  Spezielle Optionen, die an jedem Router verarbeitet werden
- *Routing (43)*
  Erweiterte Routinginformationen (vorgegebene Route)
- *Fragmentation (44)*
  Fragmentierungs-/Defragmentierungsinformationen
- *Encapsulation (50)*
  Verschlüsselung, z.B. für ‚Tunneling' vertraulicher Daten (IPSec/ESP-Protokoll)
- *Authentication (51)*
  Sicherheitsinformationen: Authentizität und Integrität (IPSec/AH-Protokoll)
- *Destination Option (60)*
  Informationen für den Empfänger-Host
- *Mobility Header (62)*
  Informationen für das Mobile IP-Protokoll

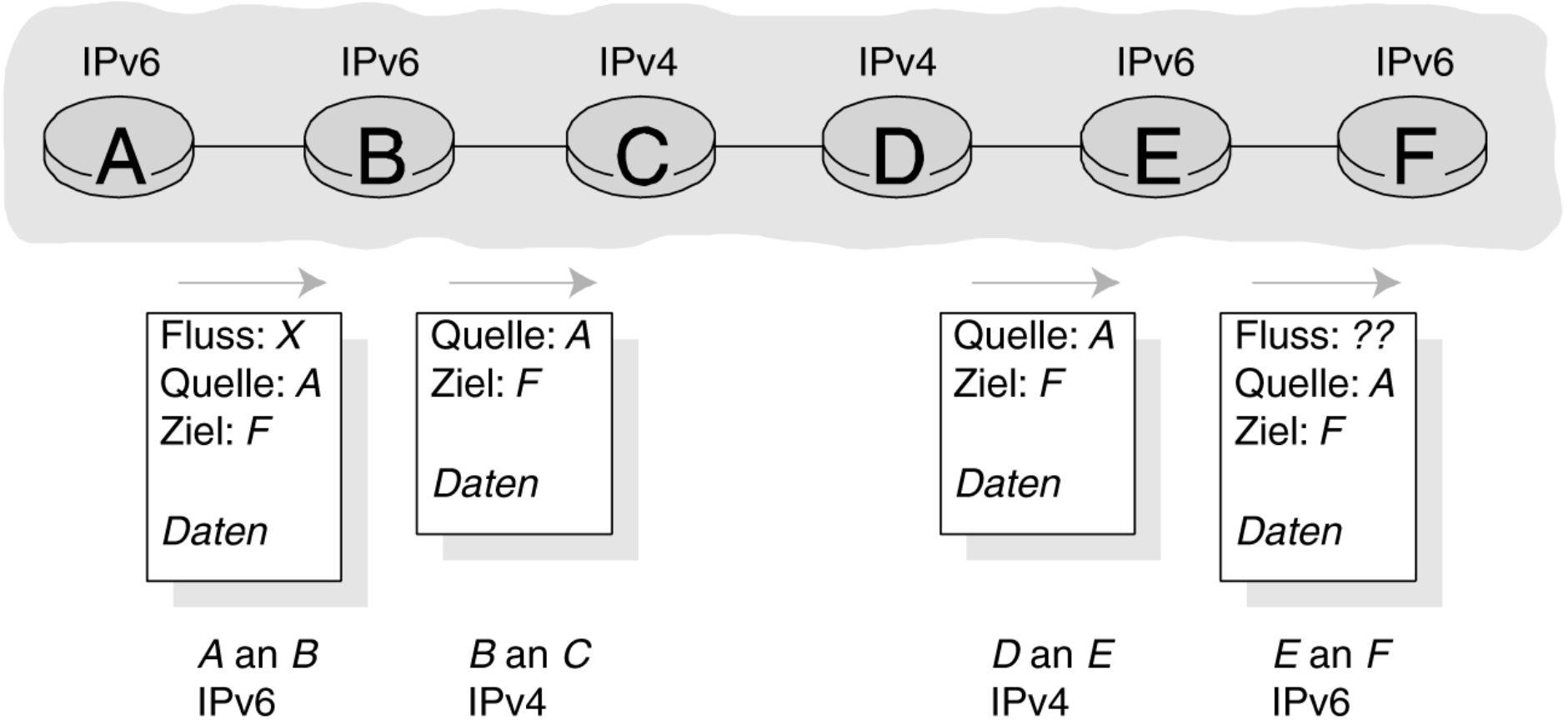
# Übergang von IPv4 auf IPv6

- Nicht alle Router können gleichzeitig aufgerüstet werden
  → Wie kann ein Netzwerk mit gemischten IPv4- und IPv6-Routern arbeiten?
- Spezielle IPv6-Router müssen mit IPv4-Routern IPv4-Datagramme austauschen können!

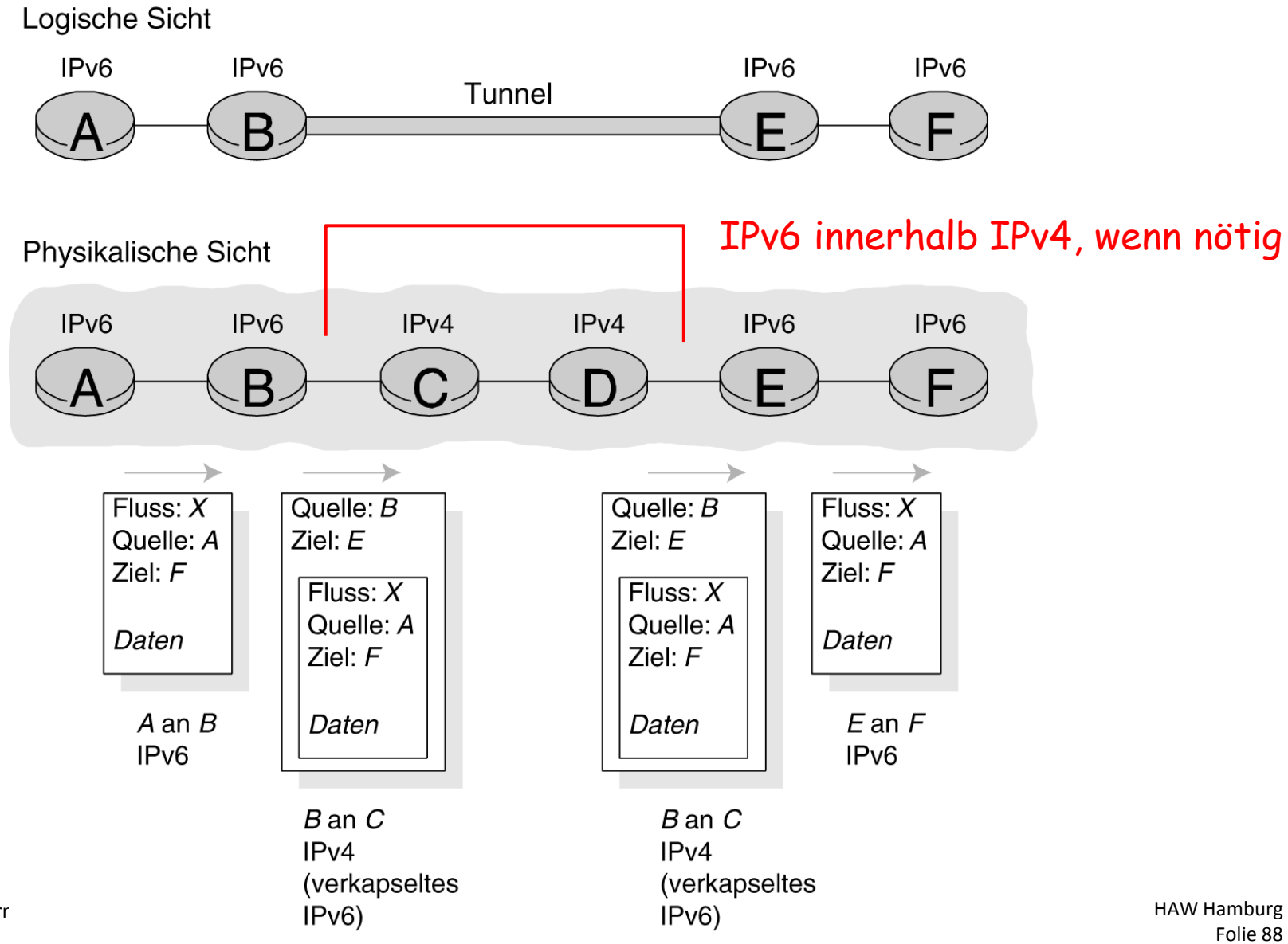
  Anwendungsmöglichkeiten:

  - *Dual Stack*: Zwischen den IP-Formaten findet eine "Übersetzung" statt
  - *Tunneling:* IPv6-Datagramme werden als Nutzdaten in IPv4-Datagrammen übertragen

# IPv6/IPv4: Dual Stack - Beispiel



# IPv6/IPv4: Tunneling - Beispiel



# Kapitel 4
## Netzwerkschicht & Routing

1. Einleitung und Netzwerkdienstmodelle
2. Aufbau eines Routers
3. Das Internet-Protokoll (IPv4)
4. Paketfilterung (Firewalls)
5. Routing-Algorithmen
6. Routing-Protokolle im Internet
7. NAT vs. IPv6
8. Mobile IP

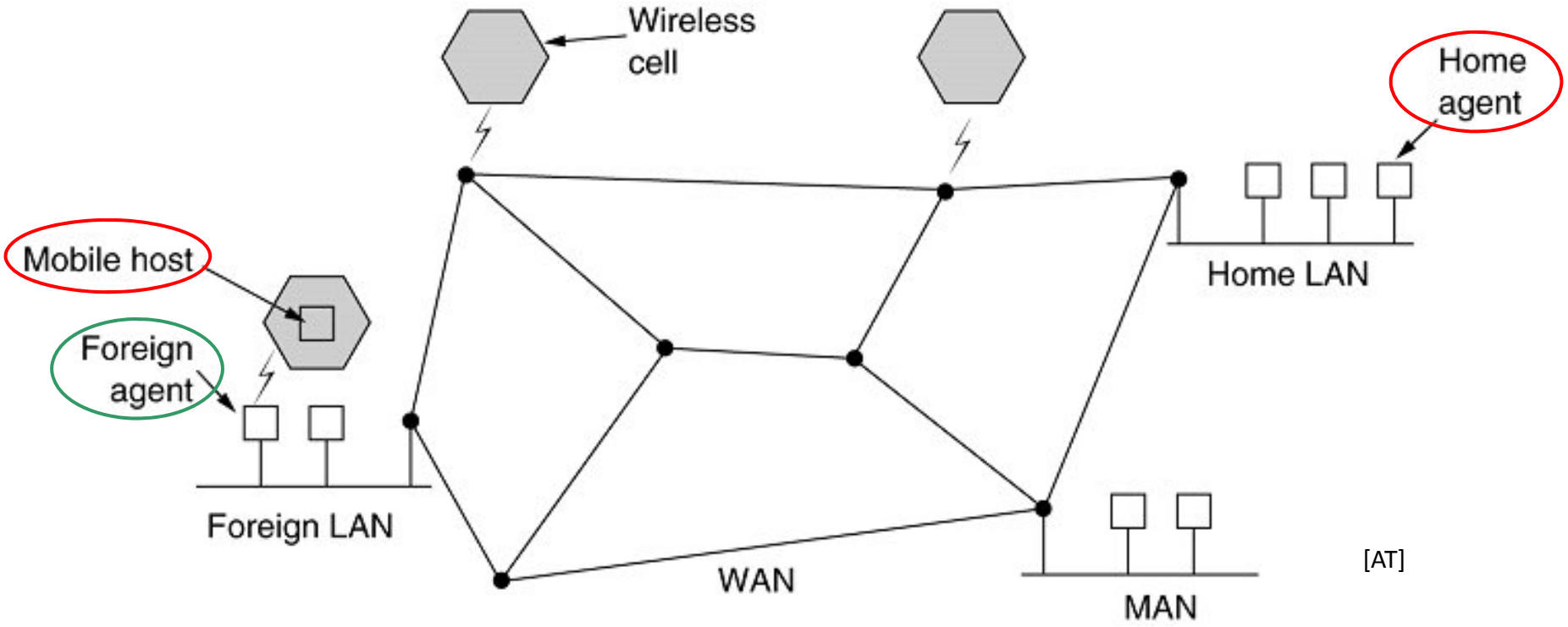
# Motivation für Mobile IP

- Problem:
  - Routing basiert auf IP-Zieladresse, Netzwerk-Präfix (z.B. 129.13.42) legt IP-Subnetz fest
  - Wird das Subnetz gewechselt so **muss** auch die IP-Adresse passend gewechselt werden (normales IP) oder ein spezieller Routing-Eintrag vorgenommen werden
- Mögliche Lösungen:
  - Spezifische Routen zum Endgerät?
    - Anpassen aller Routing-Einträge
    - Skaliert nicht mit Anzahl der mobilen Geräte und u.U. häufig wechselnden Aufenthaltsorten, Sicherheitsprobleme
  - Wechseln der IP-Adresse?
    - Je nach Lokation wird entsprechende IP-Adresse gewählt
    - Wie sollen Rechner nun gefunden werden - DNS-Aktualisierung dauert lange
    - TCP-Verbindungen brechen ab, Sicherheitsprobleme!

# Anforderungen an Mobile IP [RFC 3220]

- Transparenz
  - mobile Endgeräte **behalten** ihre IP-Adresse
  - Wiederaufnahme der Kommunikation nach Abtrennung möglich
  - Anschlusspunkt an das Netz kann gewechselt werden
- Kompatibilität
  - Unterstützung der gleichen Schicht 2-Protokolle wie IP
  - keine Änderungen an bisherigen Rechnern und Router
  - mobile Endgeräte können mit festen Endgeräten kommunizieren
- Sicherheit
  - alle Registrierungsnachrichten müssen authentifiziert werden
- Effizienz und Skalierbarkeit
  - möglichst wenige zusätzliche Daten zum mobilen Endgerät (diese ist ja evtl. über eine schmalbandige Funkstrecke angebunden)
  - eine große Anzahl mobiler Endgeräte soll Internet-weit unterstützt werden

# Mobile IP: Ausgangssituation

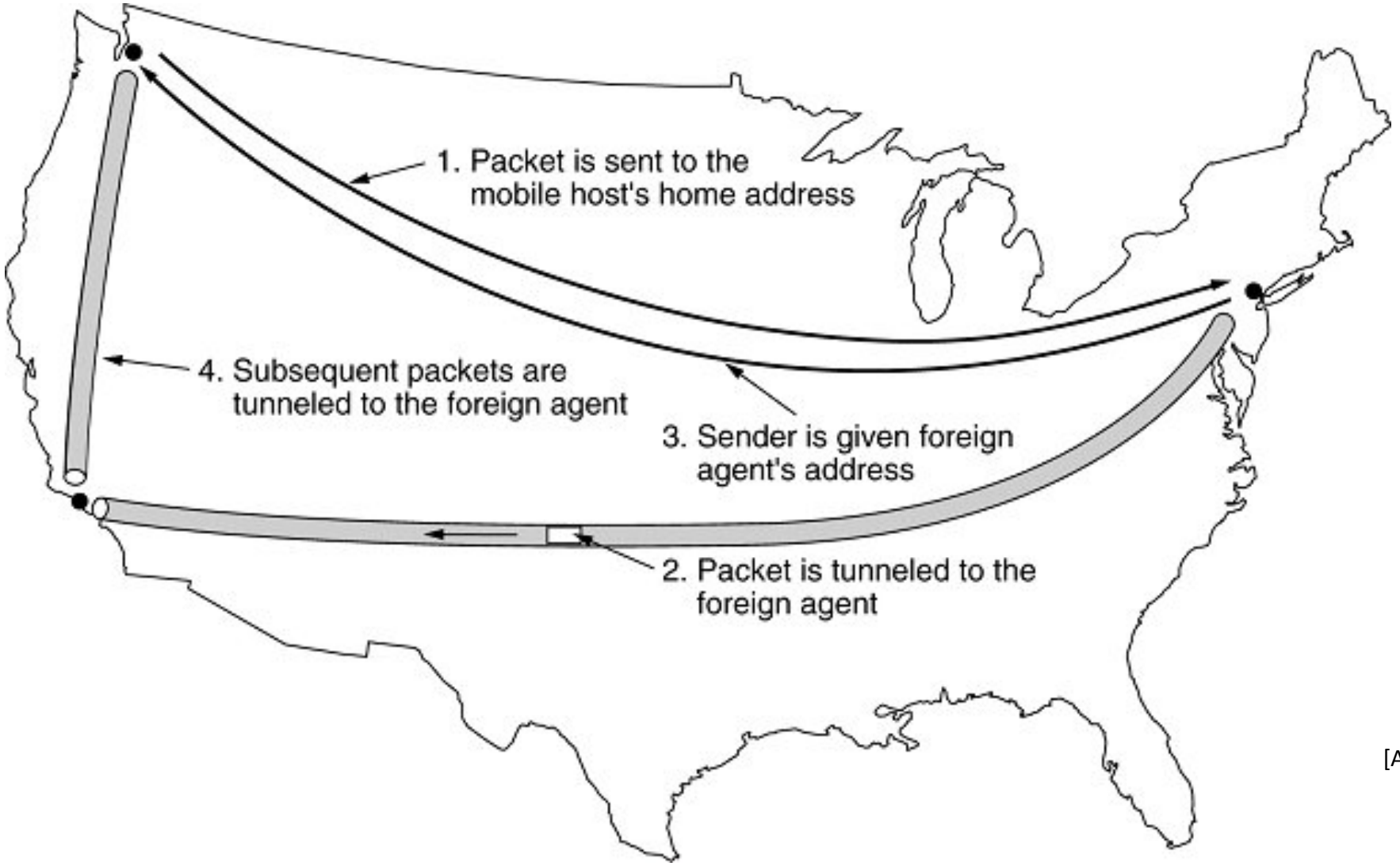



[AT]

# Registrierung eines Mobile Hosts bei einem Foreign Agent

- Jeder Foreign Agent sendet periodisch seine Kennung und Adresse als Broadcast-Nachricht ins LAN
- Ein neu hinzugekommer Mobile Host wartet auf eine Broadcast-Nachricht eines Foreign Agent und registriert sich anschließend durch Angabe seiner Heimatadresse, der aktuellen LAN-Adresse und Sicherheitsinformationen
- Der Foreign Agent informiert den Home Agent und übergibt dabei seine eigene Adresse sowie die Sicherheitsinformationen
- Der Home Agent prüft die Sicherheitsinformationen und bestätigt dem Foreign Agent die Verbindung
- Der Foreign Agent bestätigt dem Mobile Host die Registrierung

# Mobile IP: Anwendung




[AT]

# Ende des 4. Kapitels:
# Was haben wir geschafft?

## Netzwerkschicht & Routing

1. Einleitung und Netzwerkdienstmodelle
2. Aufbau eines Routers
3. Das Internet-Protokoll (IPv4)
4. Paketfilter (Firewalls)
5. Routing-Algorithmen
6. Routing-Protokolle im Internet
7. NAT vs. IPv6
8. Mobile IP