Netzwerktechnologien
3 VO

Dr. Ivan Gojmerac
ivan.gojmerac@univie.ac.at

8. Vorlesungseinheit, 15. Mai 2013

Bachelorstudium Medieninformatik
SS 2013

4.4.2 Subnetzmasken variabler Länge

- Problem: Gegeben sei ein Klasse-C-Netzwerk, welches in zwei Subnetze mit 50 Endsystemen und ein Subnetz mit 100 Endsystemen unterteilt werden soll.

- Das funktioniert nicht mit einer einzelnen Subnetzmaske!
  - `255.255.255.128`: zwei Netze mit je 128 hostids
  - `255.255.255.192`: vier Netze mit je 64 hostids

- Lösung: Subnetzmasken variabler Länge
  - Unterteile den Adressraum zunächst mit der kürzeren Subnetzmaske (1 Bit im Beispiel)
  - Unterteile eine Hälfte davon weiter mit der längeren Subnetzmaske (2 Bit im Beispiel)
  - Resultat: Subnetze verschiedener Größe

4.4.2 Subnetzmasken variabler Länge

Beispiel:

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

4.4.2 Klassenbasierte Adressierung

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

4.4.2 Klassenlose Adressierung – Classless Inter-Domain Routing (CIDR)

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

4.4.2 Adressvergabe bei Hosts - DHCP

- Bei Hosts erfolgt die Adressvergabe durch die Konfiguration von:
  - IP-Adresse
  - Subnetzmaske
  - Weiteren Parametern

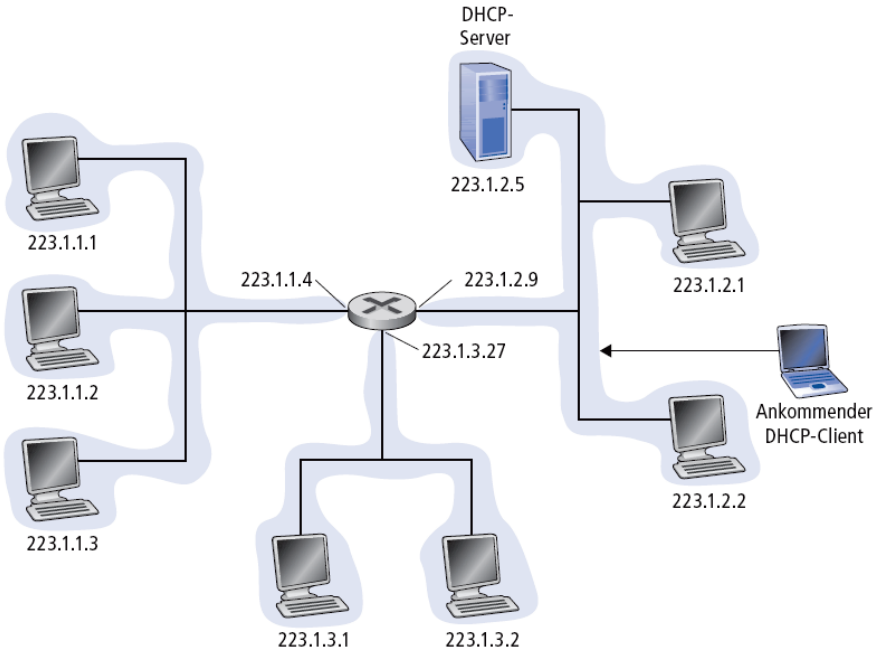
- Problem: Wie bekommen Endsysteme ihre IP-Adresse (und andere Parameter der Netzwerkschicht, wie z.B. die Subnetzmaske)?
  - Manuelle Konfiguration ist fehleranfällig
  - → Lösung: Automatische Zuweisung mithilfe von DHCP [RFC 2131]

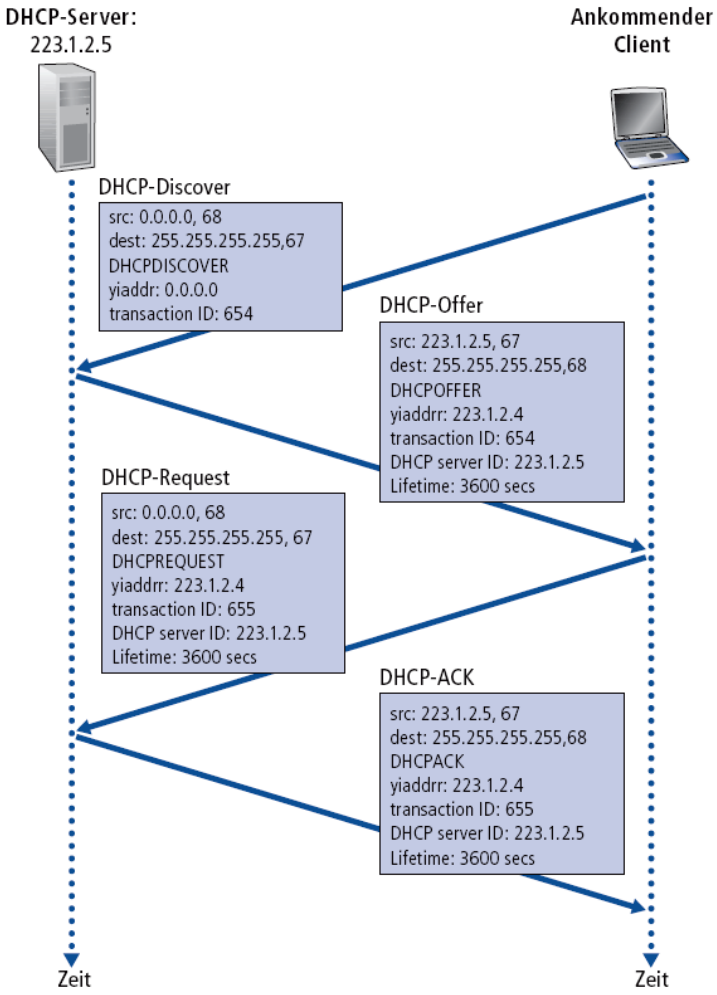
→ **DHCP**: Dynamic Host Configuration Protocol – dynamisches Beziehen der Adresse von einem Server
  - "Plug-and-Play"

4.4.2 DHCP

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

4.4.2 DHCP Szenario





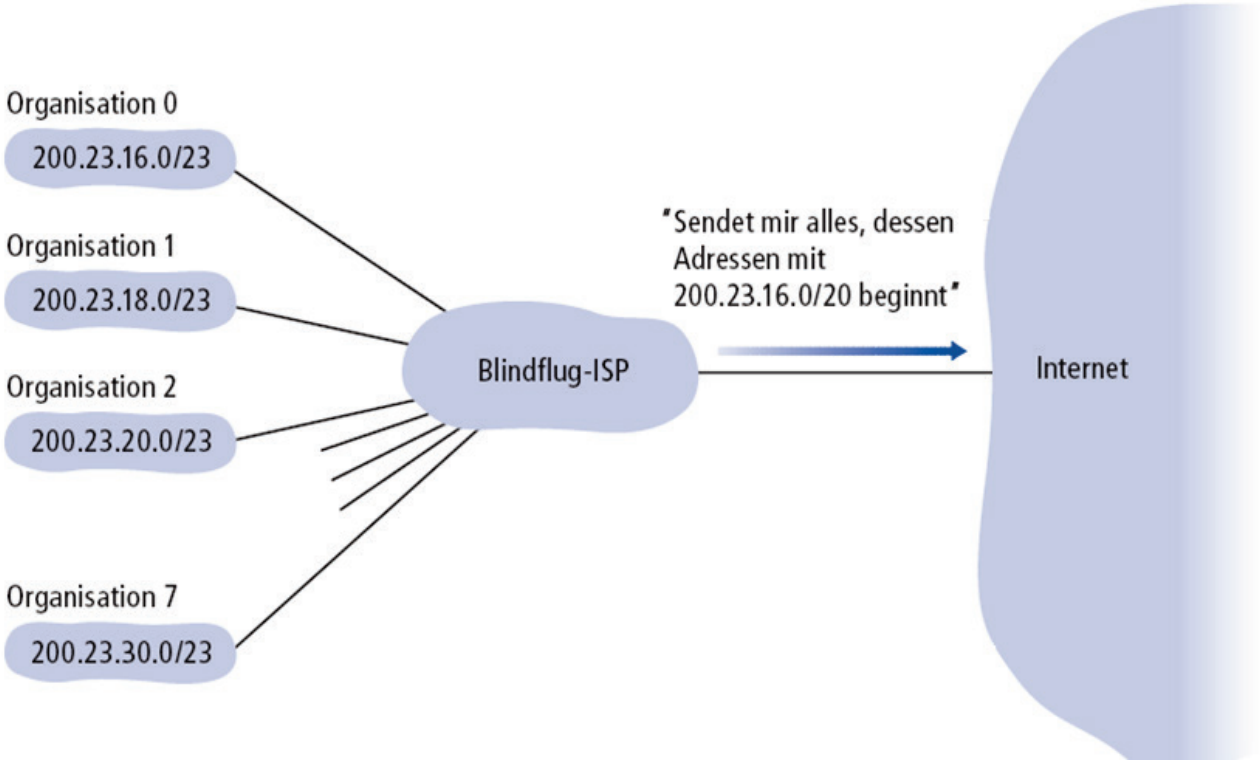
- DHCP verwendet UDP
- DHCP-Nachrichten werden an die MAC-Broadcast-Adresse geschickt
- Es gibt ein Feld, in dem eine eindeutige Kennung des Clients verpackt ist → dies ist meist die MAC-Adresse.

4.4.2 Adressvergabe bei Netzwerken

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

4.4.2 Adressvergabe bei Netzwerken

ISP	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



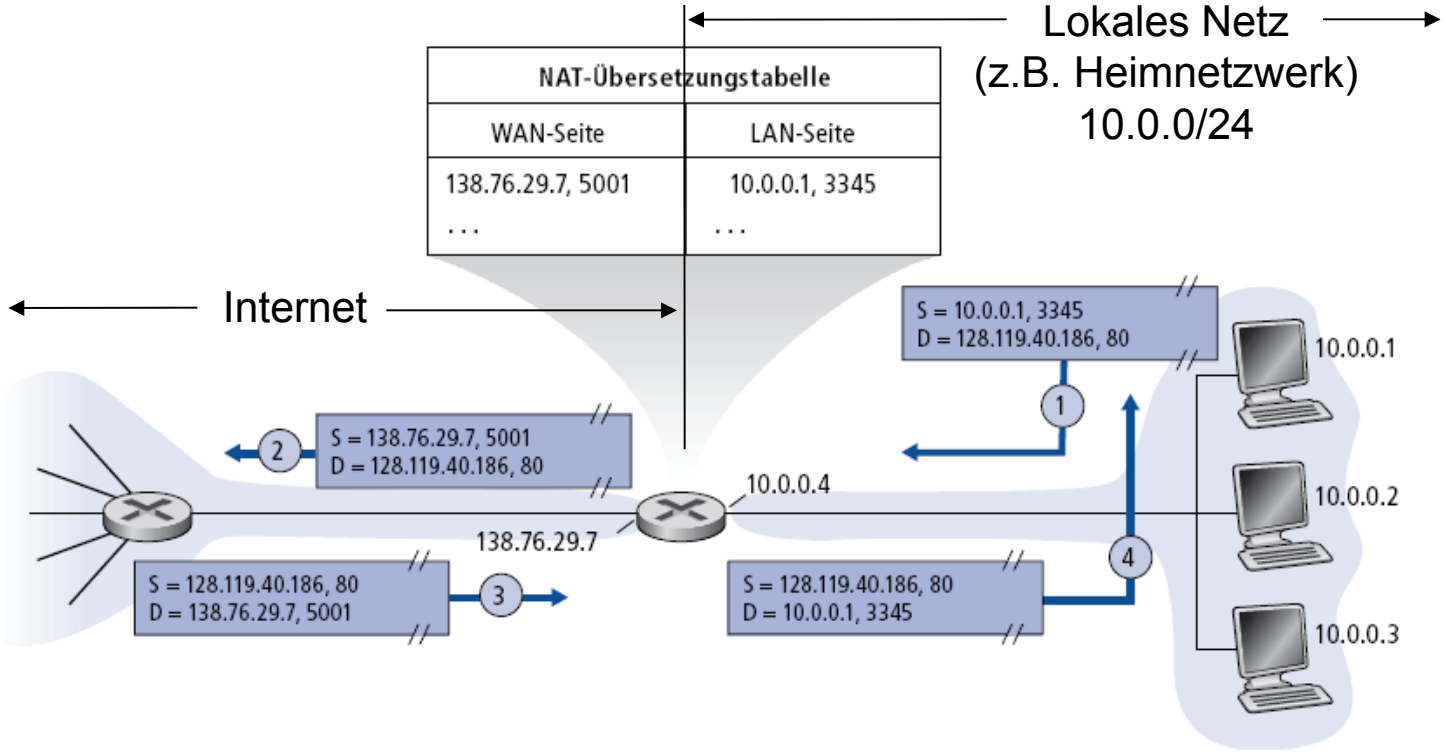
4.4.2 Adressvergabe bei Netzwerken

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

4.4.2 Network Adress Translation (NAT)

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

4.4.2 Network Adress Translation (NAT)



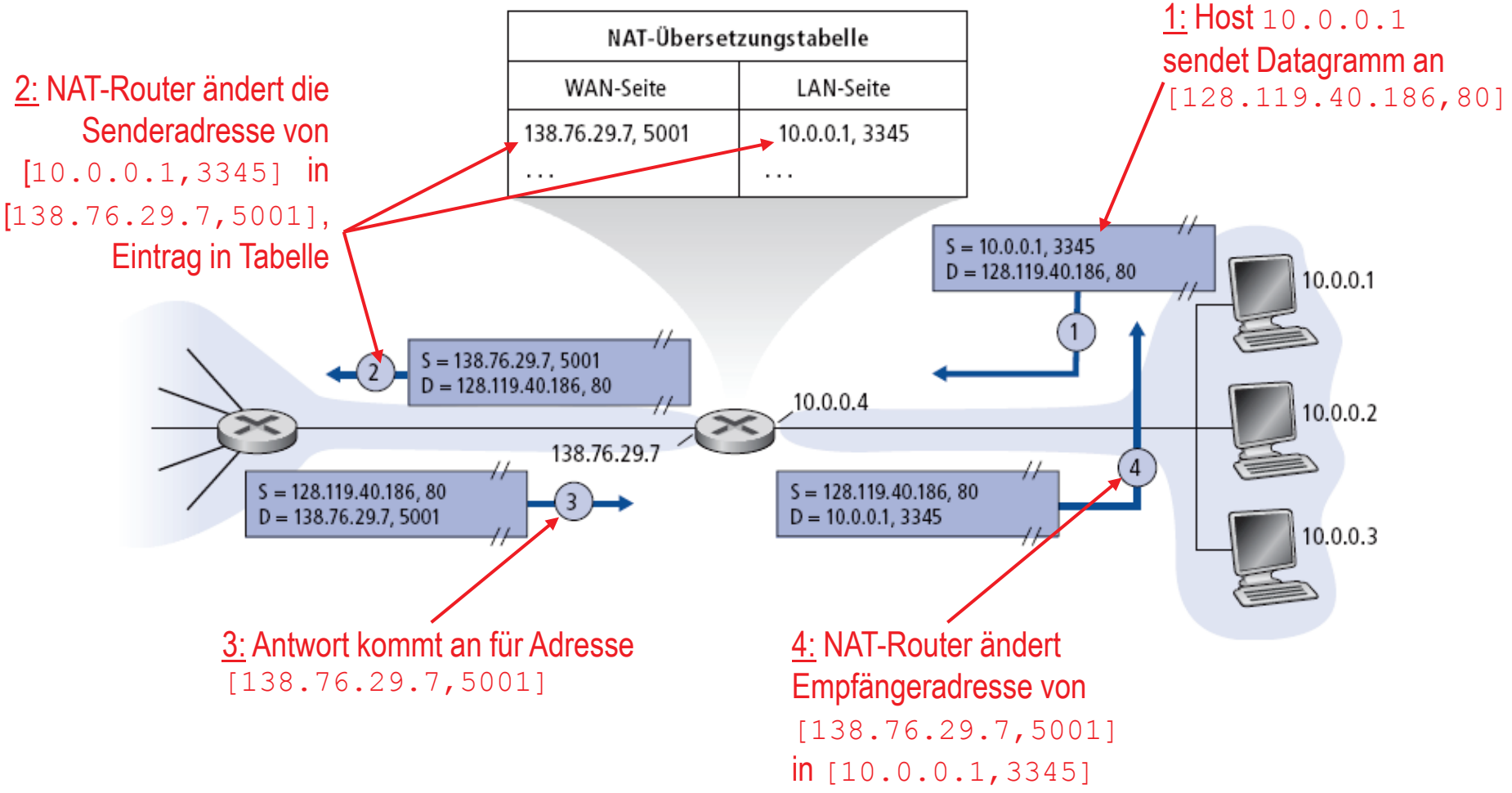
Alle Datagramme, die das lokale Netz *verlassen*, haben die *gleiche* NAT-IP-Adresse als Absender: 138.76.29.7, unterschieden werden sie über die Portnummern.

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

4.4.2 Network Adress Translation (NAT)

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

4.4.2 Network Adress Translation (NAT)



4.4.2 Network Adress Translation (NAT)

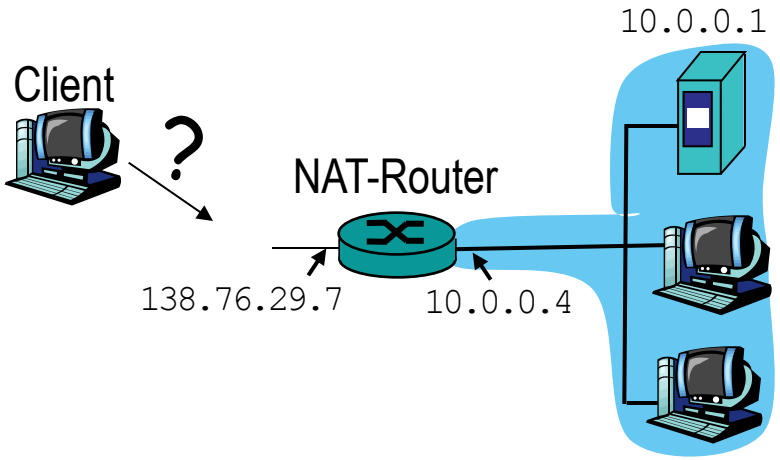
- 16-Bit-Port Number-Feld:
  - Mehr als 60.000 gleichzeitige Verbindungen mit einer IP-Adresse sehr einfach möglich
- NAT ist nicht unumstritten:
  - Router sollten im Idealfall nur Informationen der Schicht 3 verwenden
  - Verletzung des End-to-End Principle (Ende-zu-Ende Prinzip):
    - Transparente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

4.4.2 NAT-Traversal

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

Angenommene Situation:

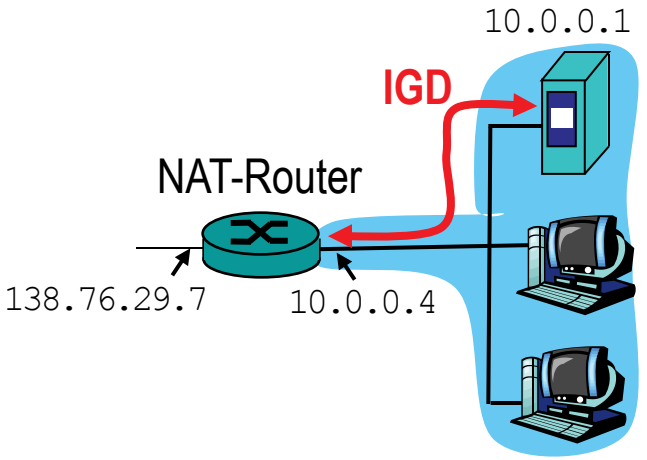
- Der Client möchte den Server mit der Adresse 10.0.0.1 kontaktieren
  - Die Adresse 10.0.0.1 ist eine lokale Adresse und kann nicht als Adresse im globalen Internet verwendet werden
  - Die einzige nach außen sichtbare Adresse ist: 138.76.29.7
- **Lösung 1**: Statische Konfiguration von NAT, so dass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



4.4.2 NAT-Traversal

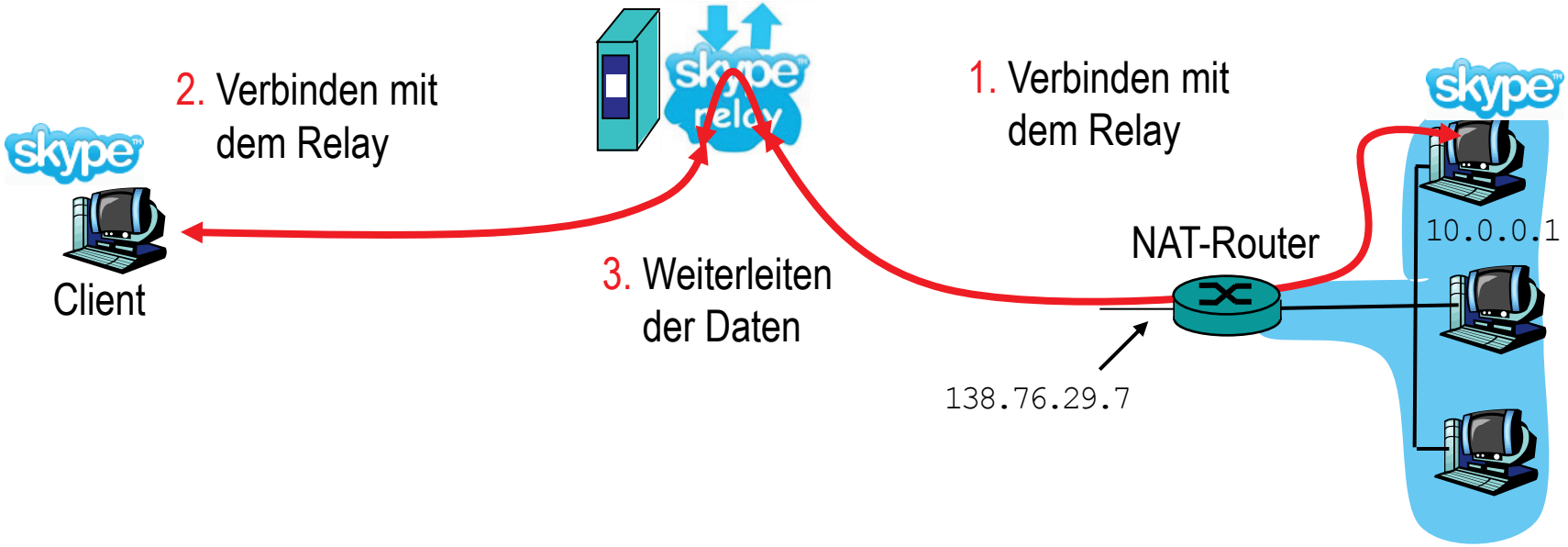
- **Lösung 2**: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Dies ermöglicht dem Host hinter dem NAT Folgendes:
  - Herausfinden der öffentlichen IP-Adresse des NAT-Routers (138.76.29.7)
  - Auffinden existierender Abbildungen in der NAT-Tabelle
  - Einträge in die NAT-Tabelle einfügen oder aus ihr löschen

  → *De facto* eine automatische Konfiguration von statischen NAT-Einträgen!



4.4.2 NAT-Traversal

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



4.4.3 Internet Control Message Protocol (ICMP)

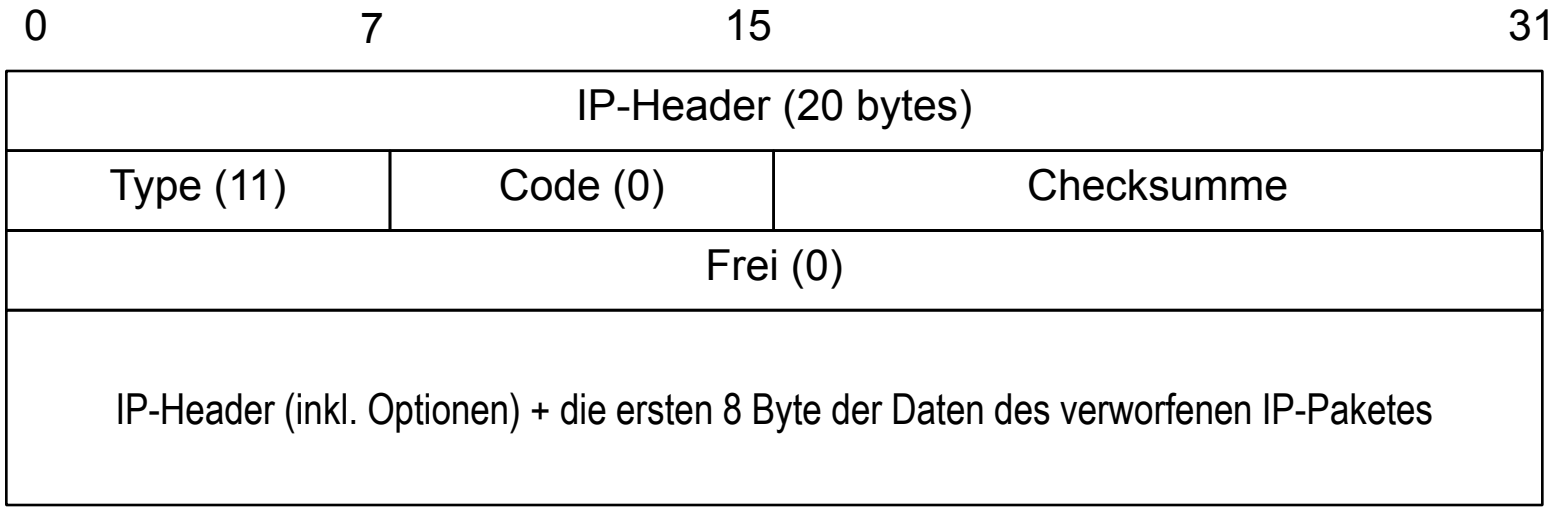
- Wird von Hosts und Routern verwendet, um Informationen über das Netzwerk zu verbreiten
  - Fehlermeldungen: Host, Netzwerk, Port, Protokoll nicht erreichbar
  - Echo-Anforderung und Antwort (von *Ping* genutzt)
- Gehört zur Netzwerkschicht, wird aber in IP-Datagrammen transportiert
- **ICMP-Nachricht**: Type, Code und die ersten 8 Byte des IP-Datagramms, welches die Nachricht ausgelöst hat

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

4.4.3 Traceroute

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

4.4.3 ICMP-Time-Exceeded-Nachricht



4.4.3 Traceroute und ICMP

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

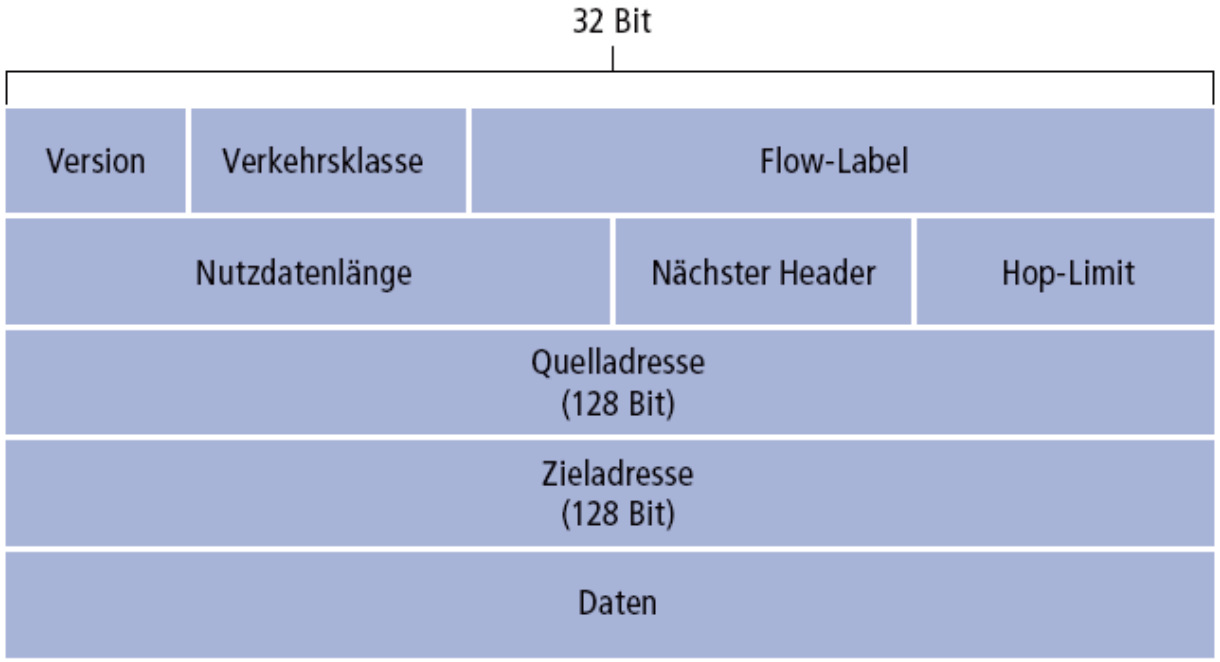
- Traceroute sendet UDP-Pakete an einen Port, der wahrscheinlich nicht verwendet wird, und erwartet eine ICMP-Port-Unreachable-Nachricht vom Empfänger!
- Demo:
  - `traceroute <host>`

→ So lernt das Quellsystem Anzahl und Identitäten der Router kennen, die zwischen ihm und dem Zielhost liegen und es kann die RTT zwischen den beiden Hosts messen.

4.4.4 IPv6

IPv6 Header

- *Verkehrsklasse:* Zur potentiellen Priorisierung von Datagrammen
- *Flow Label:* Identifikation von zusammengehörigen Flüssen von Datagrammen (z.B. ein Voice-over-IP-Telefonat)
- *Nächster Header:* An welches Protokoll sollen die Daten im Datenteil übergeben werden? Beispiel: TCP!

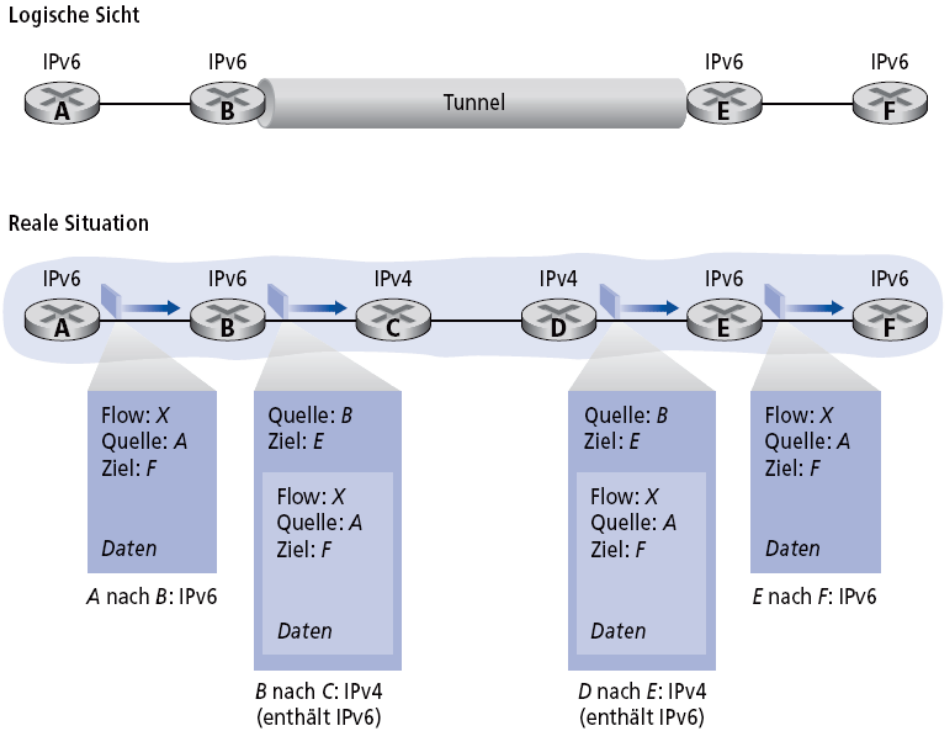


4.4.4 Veränderungen in IPv6 gegenüber IPv4

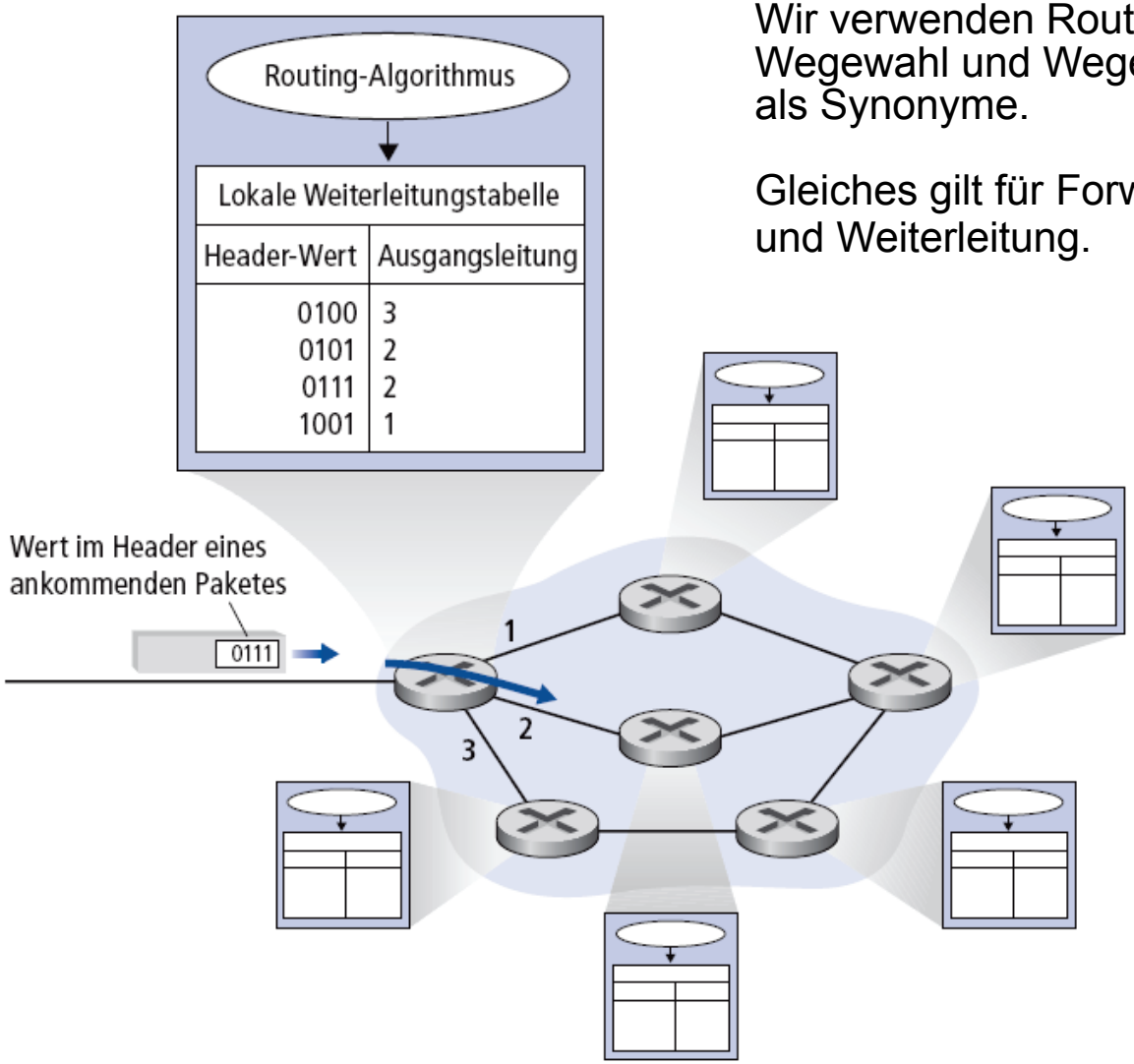
- *Checksumme*: entfernt, um die Verarbeitung in den Routern zu erleichtern
- *Optionen:* als separate Header, die auf den IP-Header folgen
  - Werden durch das “Nächster Header”-Feld angezeigt
  - Einfachere Behandlung in Hosts und Routern
- *ICMPv6:* neue Version von ICMP
  - Zusätzliche Pakettypen,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

4.4.4 Übergang von IPv4 zu IPv6

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



4.5 Routing-Algorithmen



Wir verwenden Routing, Wegewahl und Wegelenkung als Synonyme.

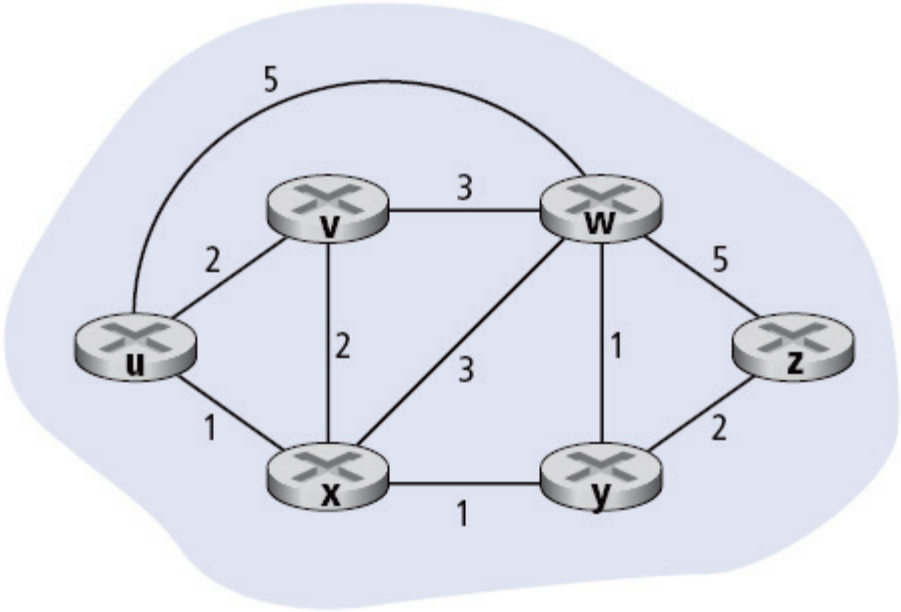
Gleiches gilt für Forwarding und Weiterleitung.

4.5 Ein Netzwerk als Graph

Graph: **G = (N, E)**

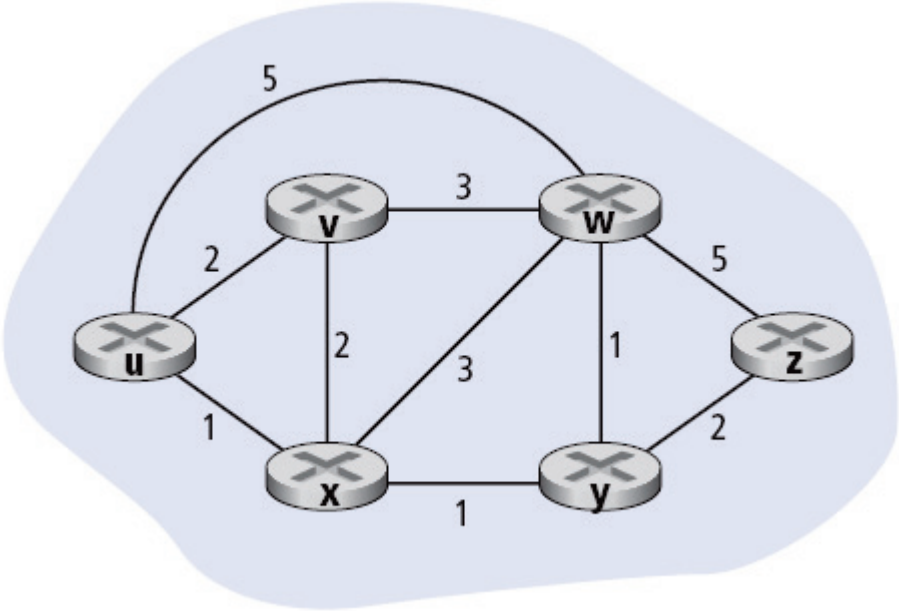
N = Menge von Routern:
{ u, v, w, x, y, z }

E = Menge von (in diesem Fall ungerichteten) Links:
{ (u, v), (u, x), (u, w), (v, x), (v, w), (x, w), (x, y), (w, y), (w, z), (y, z) }



Anmerkung:	Graphen können auch in anderen Zusammenhängen in Computernetzwerken betrachtet werden.
Beispiel:	P2P, dann ist **N** die **Menge der Peers** und **E** die **Menge der Layer 4-Verbindungen** zwischen den Peers.

4.5 Kosten



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

Kosten eines Pfades (x1, x2, x3,…, xp) = c(x1, x2) + c(x2, x3) + … + c(xp-1, xp)

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

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

4.5 Klassifikation von Routing-Algorithmen

Statisch oder dynamisch?

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

Globale oder dezentrale Informationen?

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

4.5 Link-State-Routing-Algorithmus

Dijkstras Algorithmus

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

4.5 Djikstras Algorithmus

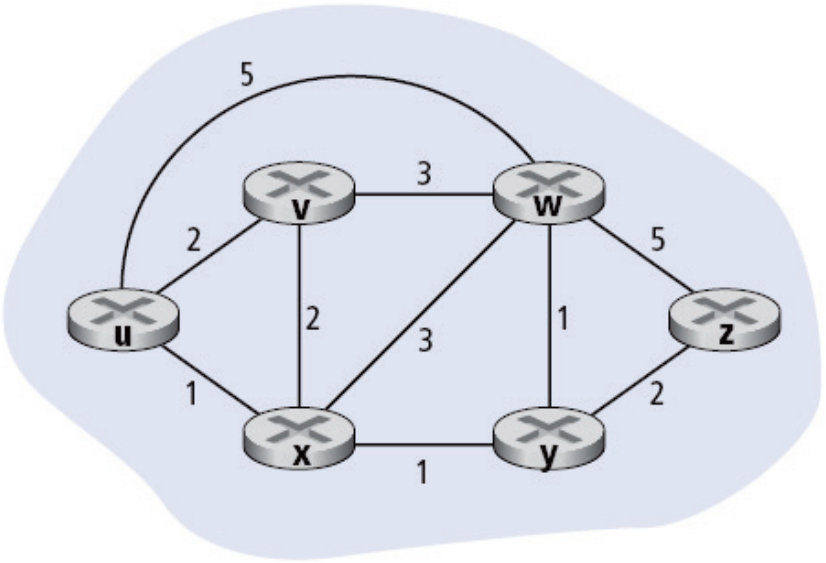
```
1  initialisiere:
2     N' = {u}     // u = Quelle
3     für alle Knoten v aus N
4        wenn v ein Nachbar von u ist
5           dann D(v) = c(u,v)
6        sonst D(v) = ∞
7
8  wiederhole:
9     finde ein w aus N welches nicht in N' ist, so dass D(w) minimal ist
10    füge w zu N' hinzu
11    Berechne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

N: Menge aller Knoten
N': Menge der Knoten, für die der günstigste Pfad definitiv feststeht
c(x, y): Kosten des Links von x nach y; wird auf ∞ gesetzt, wenn sie keine Nachbarn sind
D(v): Kosten des günstigsten derzeit bekannten Pfades von der Quelle zu v
p(v): Vorgängerknoten entlang des Pfades von der Quelle nach v

4.5 Djikstras Algorithmus

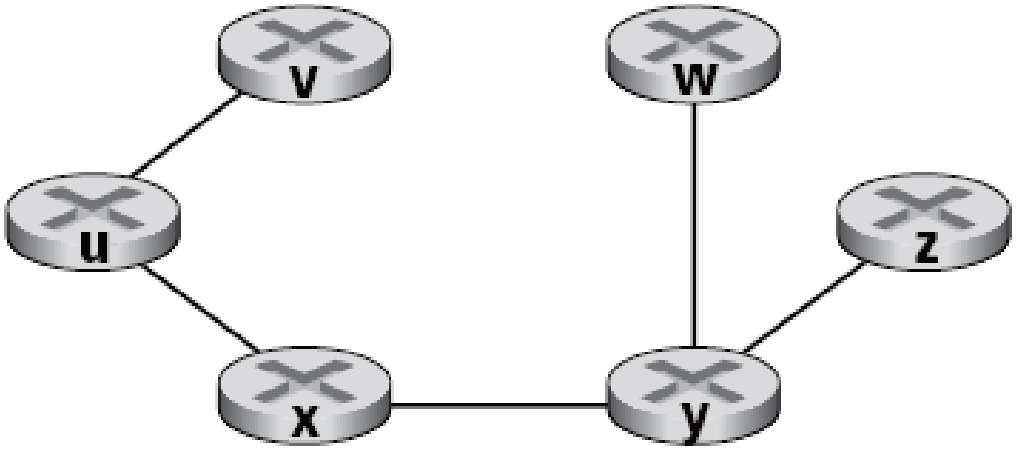
Beispiel:

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



4.5 Djikstras Algorithmus

Baum der kürzesten Pfade (Shortest Path Tree) von **u**:



Weiterleitungstabelle in u:

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

Rechenkomplexität: n Knoten

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

→ $n(n+1)/2$ Vergleich: $O(n^2)$

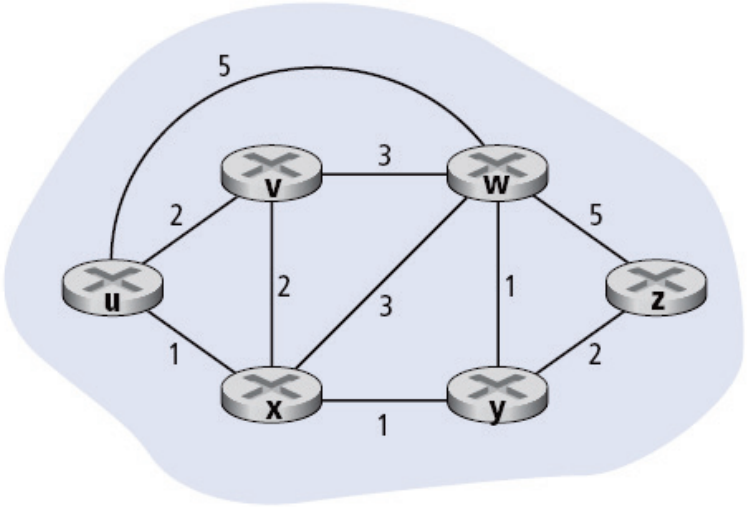
- Effizientere Implementierung möglich: O(n log n)

4.5 Distance-Vector-Routing

- Prinzipielle Idee: **Bellman-Ford-Gleichung**
- Sei $d_x(y)$ der billigste Pfad von x nach y
- Dann gilt:
  - $\mathbf{d_x(y) = \min \{c(x,v) + d_v(y) \}}$

- Wobei das Minimum über alle Nachbarn v von x gebildet wird

4.5 Distance-Vector-Routing



Beispiel zu Bellman-Ford:

- Es gilt:
  - $d_v(z)=5$, $d_x(z) = 3$, $d_w(z) = 3$

- Aussage der Bellman-Ford-Gleichung ist dann:
  $d_u(z) = \min \{c(u,v) + d_v(z), c(u,x) + d_x(z), c(u,w) + d_w(z) \}$
  $= \min \{ \quad 2 + 5, \quad 1 + 3, \quad 5 + 3\}$
  $= 4$
- Der Nachbar, über den das Minimum erreicht wird, wird für das betrachtete Ziel in die Routing-Tabelle eingetragen
  → Hier: **z** wird von **u** aus am besten über **x** erreicht

4.5 Der Distance-Vector-Algorithmus

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

4.5 Distance-Vector-Routing

Prinzipielles Vorgehen:

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

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

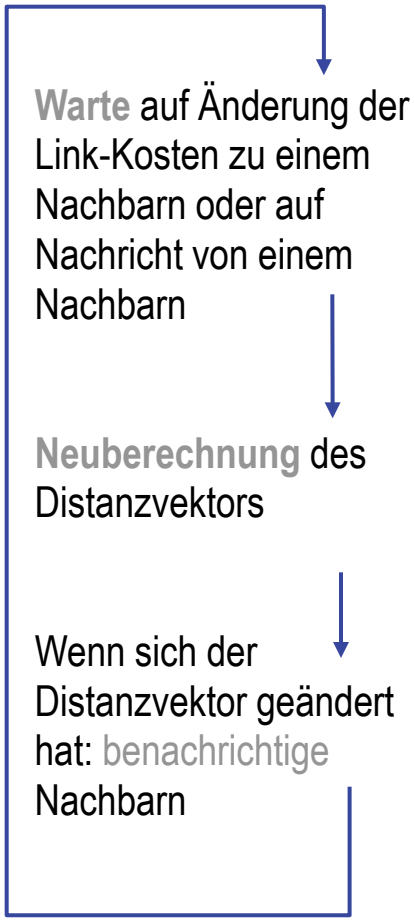
$d_x(y) = \min \{c(x,v) + d_v(y)\}$, Minimum wird über alle Nachbarn v gebildet

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

4.5 Distance-Vector-Routing

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

Ablauf in jedem Knoten:



4.5 Distance-Vector-Routing

Beispiel für eine Distanzvektor-Tabelle im Knoten **y**:

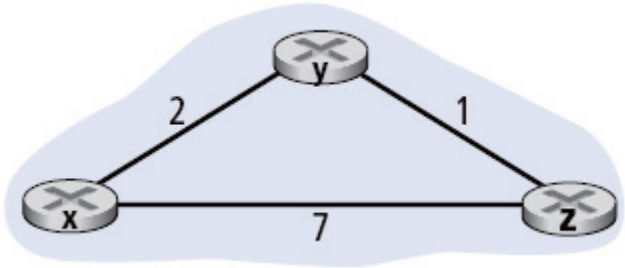
Tabelle von y

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

Distanzvektor von Nachbar x

Eigener Distanzvektor (von y)

Distanzvektor von Nachbar z



4.5 Distance-Vector-Routing

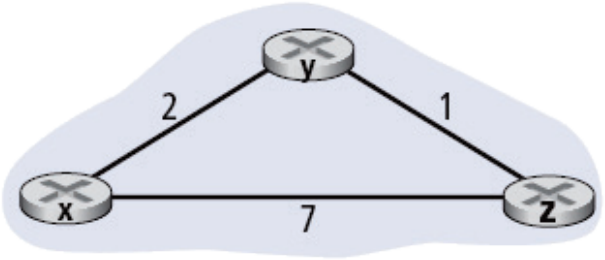


Tabelle von x

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

Tabelle von y

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

Tabelle von z

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

Zeit

4.5 Distance-Vector-Routing

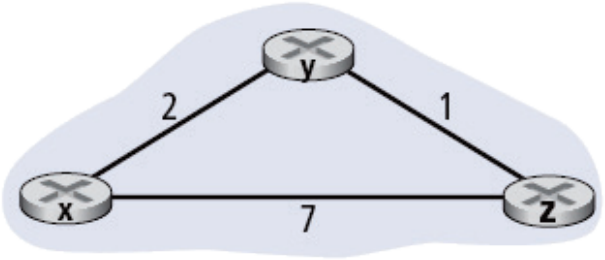


Tabelle von x

nach

von	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

nach

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

Tabelle von y

nach

von	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

nach

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

Tabelle von z

nach

von	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

nach

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

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

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

Zeit

4.5 Distance-Vector-Routing

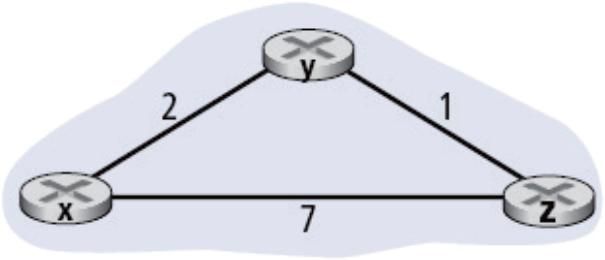


Tabelle von x

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

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

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

Tabelle von y

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

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

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

Tabelle von z

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

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

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

Zeit