Routing

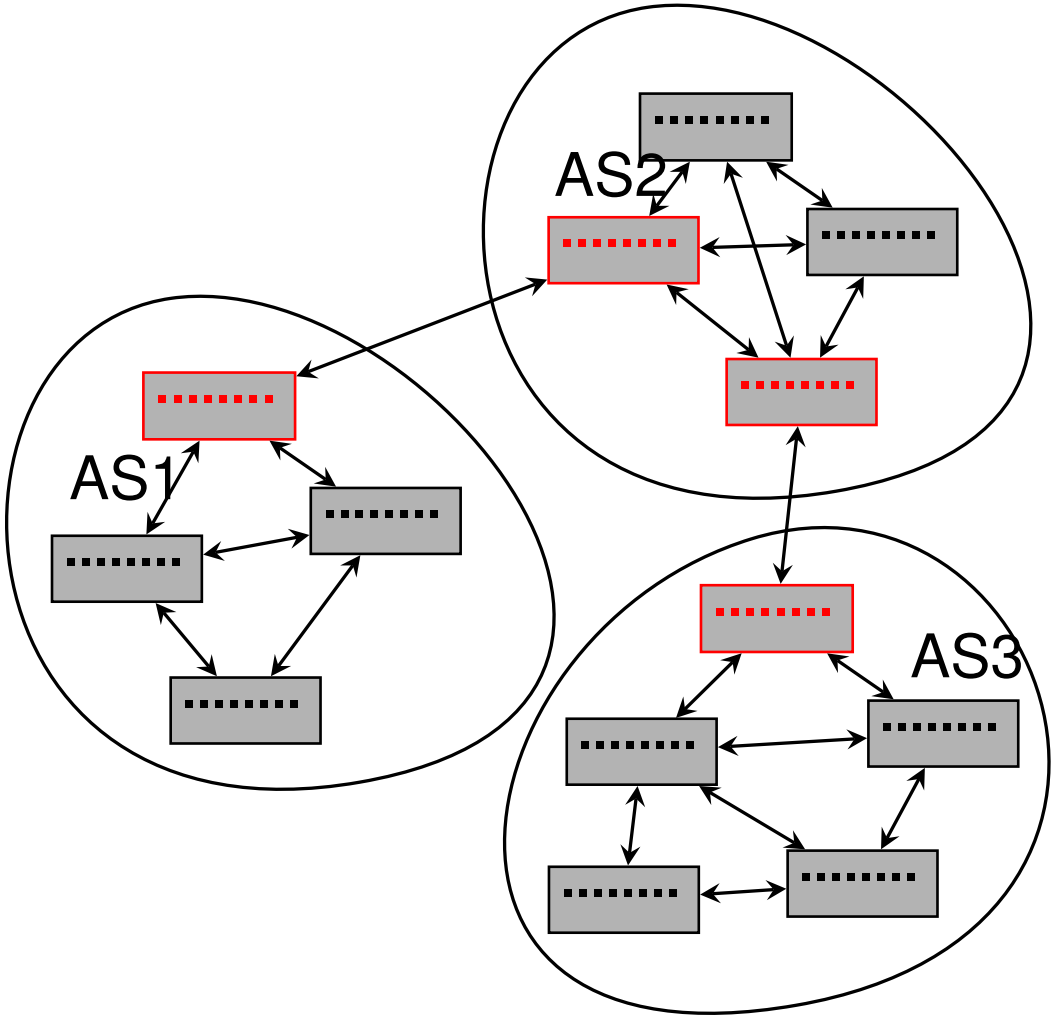
Grundlage

- Das Internet gliedert sich in Bereiche unterschiedlicher administrativer Verantwortung (z.B. Veratwortung eines ISPs), sog. **autonome Systeme** (**AS**).
- Es muß unterschieden werden zwischen Routing innerhalb eines AS (intra-AS) und zwischen verschiedenen AS (inter-AS), da hier unterschiedliche Anforderung an den Austausch von Routinginformation gestellt werden.
- Die Knoten an den Verbindungsstellen zwischen AS heißen **Gateway Router**.

Anforderungen an Routingprotokolle

- **Skalierbakeit:** Besonders ein inter-AS Routingprotokoll muß mit wachsender Anzahl AS skalieren können, da es keine Möglichkeit gibt, die Anzahl zu kontrollieren/reduzieren.
- **Leistungsfähigkeit:** Das Protokoll sollte in der Lage sein, schnell auf Änderungen in der Netzstruktur zu reagieren, administrative Vorgaben in die Routenplanung zu integrieren und gute Routen zu generieren.
- **Flexibilität:** Das Protokoll muß in der Lage sein, administrative Vorgaben in die Planung einzubauen. Besonders wichtig ist das beim inter-AS Routing, da z.B. keine/unterschiedliche vertragliche Beziehung zwischen den Betreibern einzelner AS bestehen.

Beispielnetz



Bezeichnungen aus der Graphenthorie

- **Graph:** Ein 2-Tupel $G = (V, E)$ ist ein (ungerichteter) Graph, falls $V \neq \emptyset$ und $E = \{ \{v_1, v_2\} \mid v_1, v_2 \in V\}$.
- **Gewichteter Graph:** Ein Graph $G = (V, E)$ mit einer Funktion $c : E \rightarrow \mathbb{R}$ heißt gewichteter Graph.
- **adjazent:** Zwei Ecken $v_1, v_2 \in V$ heißen adjazent, falls $\{v_1, v_2\} \in E$.
- **zusammenhängend:** Ein Graph heißt zusammenhängend, falls es zu jedem Paar $w_1, w_2 \in V$ eine Folge $v_1, \ldots, v_n \in V$ gibt, so daß $v_i, v_{i+1}\ \forall i$ und w_1, v_1 bzw. v_n, w_2 adjazent sind.

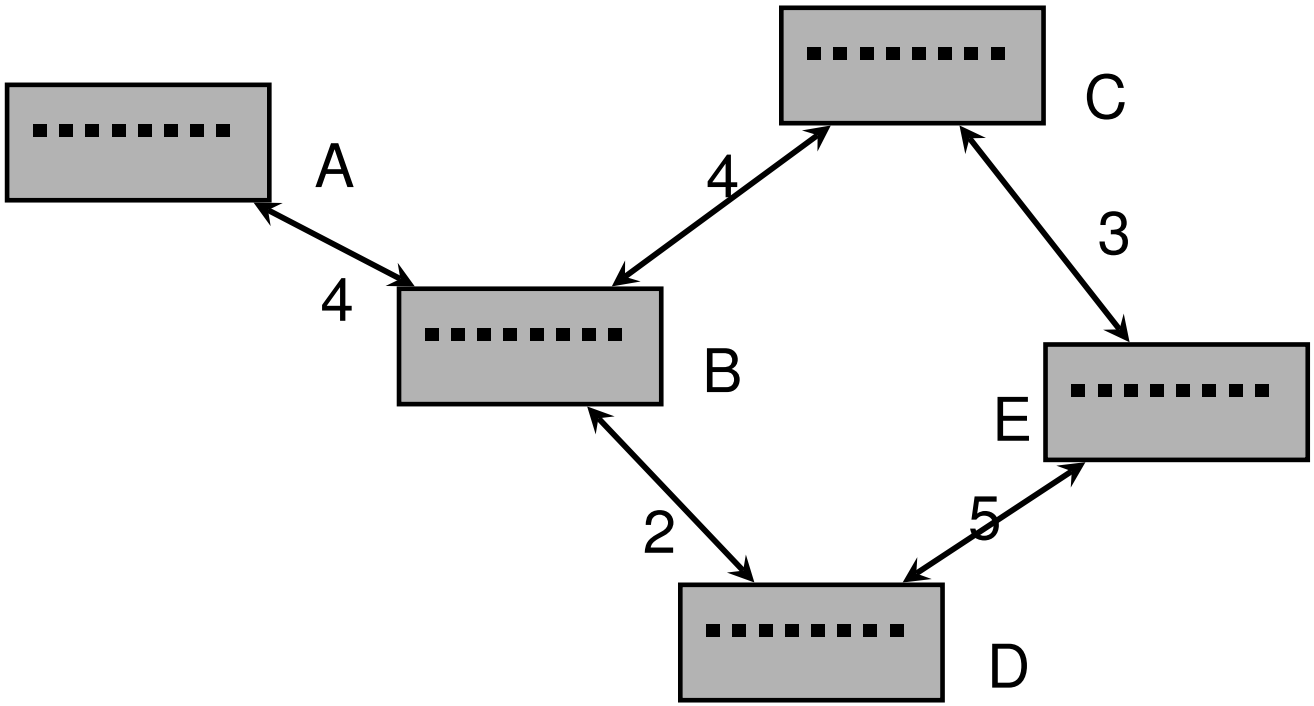
V ist die Menge der Ecken (Vertices) eines Graphen, E die Menge der Kanten. Die Funktion c ordnet jeder Kante $e \in E$ ein Gewicht zu. Oft läßt sich das Gewicht als Kosten oder Distanz interpretieren.

Übersicht über Routingverfahren

- **Distance Vector Algorithms:** Benutzt den “distributed Bellman Ford Algorithmus”, hat Probleme in großen Netzen. Kommunikation ist nur mit unmittelbaren Nachbarn notwendig.
- **Link State Algorithms:** Jeder Router bildet einen gewichteten Graph des gesamten Netzes. Die Routingtabelle ergibt sich durch den Shortest Path Algorithm. Das Verfahren hat Probleme in großen Netzen und erzeugt hohe Netz- und CPU-Last.
- **Path Vector Protocol:** Ähnlich den Distance Vector Algorithms, aber es werden nur ausgewählte Hosts (Speaker Nodes) einbezogen.

Beispiel eines AS

- 5 Router innerhalb des autonomen Systems
- 5 Links mit unterschiedlichen Kosten



Bellmann-Ford Algorithmus: Initialisierung

Wir betrachten ungerichteten Graphen $G = (V, E)$ mit Gewicht c. Ziel ist, zu jeder Ecke v die minimale Distanz $D_v(w)$ zu jeder anderen Ecke w zu bestimmen.

Initialisierung:

$$D_v(w) := \begin{cases} 0 & \text{falls } v = w \\ c(\{v, w\}) & \text{falls } v, w \text{ adjazent} \\ \infty & \text{sonst} \end{cases}$$

$D_w(y) := \infty$ für alle $w \in V,\ v, w$ adjazent. (Initialisierung der möglich Distanzvektoren der Nachbarn von v)

Sende Distanzvektor $\mathbf{D}_v := (D_v(y),\ y \in V)$ an alle Nachbarn von v.

Bellmann-Ford Algorithmus: Hauptschleife

Der Algorithmus besteht aus drei Schritten:

1. Empfange Distanzvektor(en) $\mathbf{D}_w$ von Nachbar(n) w.
2. Für alle $y \in V$ setzte
$$D_v(y) := \min_{w \in V} \left\{ c(\{v, w\}) + D_w(y) \right\}$$
3. Falls $\mathbf{D}_v$ verändert wurde, sende $\mathbf{D}_v$ an alle Nachbarn.

Wann immer ein Distanzvektor von einem Nachbarn w empfangen wird, prüfe, ob sich dadurch ein besserer Weg zu einem der Knoten im Netz ergibt, d.h. ist die Summe

- Weg nach w
- Weg von w zum Ziel

günstiger als der bisher bekannte Weg.

Bellmann-Ford im Beispiel AS

	A	B	C	D	E
A	0	**4**	∞	∞	∞
B	**4**	0	**4**	**2**	∞
C	∞	**4**	0	∞	**3**
D	∞	**2**	∞	0	**5**
E	∞	∞	**3**	**5**	0

C sendet Distanzvektor an B und E:

	A	B	C	D	E
A	0	**4**	∞	∞	∞
B	**4**	0	**4**	**2**	$7(C)$
C	∞	**4**	0	∞	**3**
D	∞	**2**	∞	0	**5**
E	∞	7(C)	**3**	**5**	0

Bellmann-Ford im Beispiel AS

B sendet Distanzvektor an A,C,D:

	A	B	C	D	E
A	0	**4**	8(B)	6(B)	11(B)
B	**4**	0	**4**	**2**	7(*C*)
C	8(B)	**4**	0	6(B)	**3**
D	6(B)	**2**	6(B)	0	**5**
E	∞	7(C)	**3**	**5**	0

D sendet Distanzvektor an B,E:

	A	B	C	D	E
A	0	**4**	8(B)	6(B)	11(B)
B	**4**	0	**4**	**2**	7(*C*)
C	8(B)	**4**	0	6(B)	**3**
D	6(B)	**2**	6(B)	0	**5**
E	11(D)	7(C)	**3**	**5**	0

Bellmann-Ford Algorithmus: Count-to-Infinity Problem

Distance Vector Algorithmen können sehr langsam konvergieren.

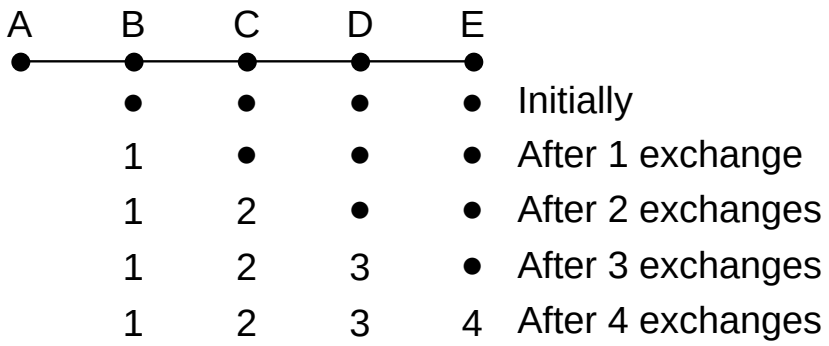
- Gute Nachrichten breiten sich schnell aus
- Schlechte Nachrichten breiten sich langsam aus

Beispiel:
5 Knoten A,B,C,D,E linear vernetzt
Distanz jeweils 1

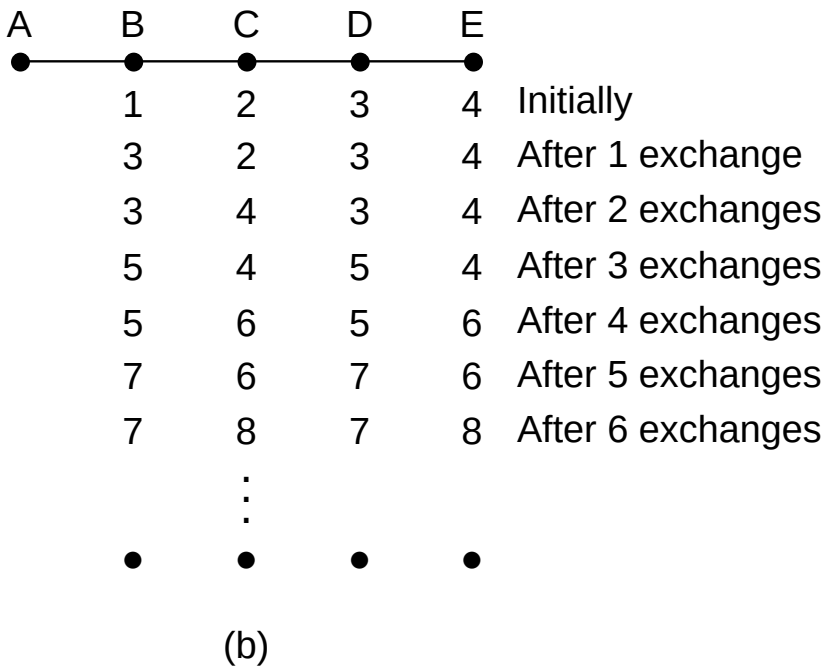
- a) Knoten A wird eingeschaltet
- b) Knoten A wird ausgeschaltet

Bellmann-Ford Algorithmus: Count-to-Infinity Problem

Angegeben ist die Distanz zu Knoten A.



(a)



(b)

Mögliche Lösung:
Abbruch wenn Distanz größer als längster Pfad + 1.

Dijkstra Algorithmus

Seien G und c wie im Bellmann-Ford Algorithmus. Sei ferner $u \in V$ eine Ecke des Graphen.

Initialisierung:

Setze $N := \{u\}$ und

$$D(v) := \begin{cases} 0 & \text{falls } u = v, \\ c(\{u, v\}) & \text{falls } u, v \text{ adjazent}, \\ \infty & (\textit{sonst}) \end{cases}$$

Dijkstra Algorithmus

do
finde $w \in V \setminus N$, so daß $D(w) \leq D(w') \; \forall \; w' \in V \setminus N$
$N := N \cup \{w\}$
Für alle v mit $v \in V \setminus N$, v, w adjazent bilde
$D(v) := \min\{D(v), D(w) + c(\{v, w\})\}$
while $N \neq V$

- Während des Ablaufes gibt $D(v)$ stets für alle $v \in V$ die beste bis dahin gefundene Distanz zwischen u und v an.
- N enthält die Ecken, für die das Ergebnis schon feststeht.
- Initial ist $N = \{u\}$ mit Distanz $D(u) = 0$.
- In jedem Schritt wird die nächstgelegene Ecke außerhalb N betrachtet und geprüft, ob Routing über diese Ecke den Weg zu anderen Ecken verkürzt.

Dijkstra Algorithmus im Beispiel AS

Wir betrachten Ecke A als Ausgangspunkt, in der Tabelle sind für jeden Schritt die aktuellen Werte von $D(v)$ für $v \in \{ABCDE\}$ aufgetragen.

Schritt	A	B	C	D	E	N
1	0	4(B)	∞	∞	∞	$\{A\}$
2	0	4(B)	8(B)	6(B)	∞	$\{A, B\}$
3	0	4(B)	8(B)	6(B)	11(B)	$\{A, B, D\}$
4	0	4(B)	8(B)	6(B)	11(B)	$\{A, B, C, D\}$
5	0	4(B)	8(B)	6(B)	11(B)	$\{A, B, C, D, E\}$

Dijkstra Algorithmus im Beispiel AS

Entsprechend für Ecke C als Ausgangspunkt:

Schritt	A	B	C	D	E	N
1	∞	4(B)	0	∞	3(E)	$\{C\}$
2	∞	4(B)	0	8(E)	3(E)	$\{C, E\}$
3	8(B)	4(B)	0	6(B)	3(E)	$\{B, C, E\}$
4	8(B)	4(B)	0	6(B)	3(E)	$\{B, C, D, E\}$
5	8(B)	4(B)	0	6(B)	3(E)	$\{A, B, C, D, E\}$

Broadcast Routing

Broadcast:
Übertragen einer Nachricht an alle Empfänger simultan

Mögliche Broadcast Routing Verfahren:

- Übertragen getrennter Nachrichten an jeden Empfänger
  - sehr einfach
  - Verschwendung von Bandbreite
  - alle Adressen müsen bekannt sein
- **Flodding:** Jeder Router sendet ein ankommendes Paket an alle Nachbarn, außer dem vorherigen Sender
  - Flodding generiert sehr viele Duplikate
  - Maßnahmen zur Begrenzung notwendig
  - sehr robust
  - benutzt immer den kürzesten Weg (mögliche Referenz)

Broadcast Routing

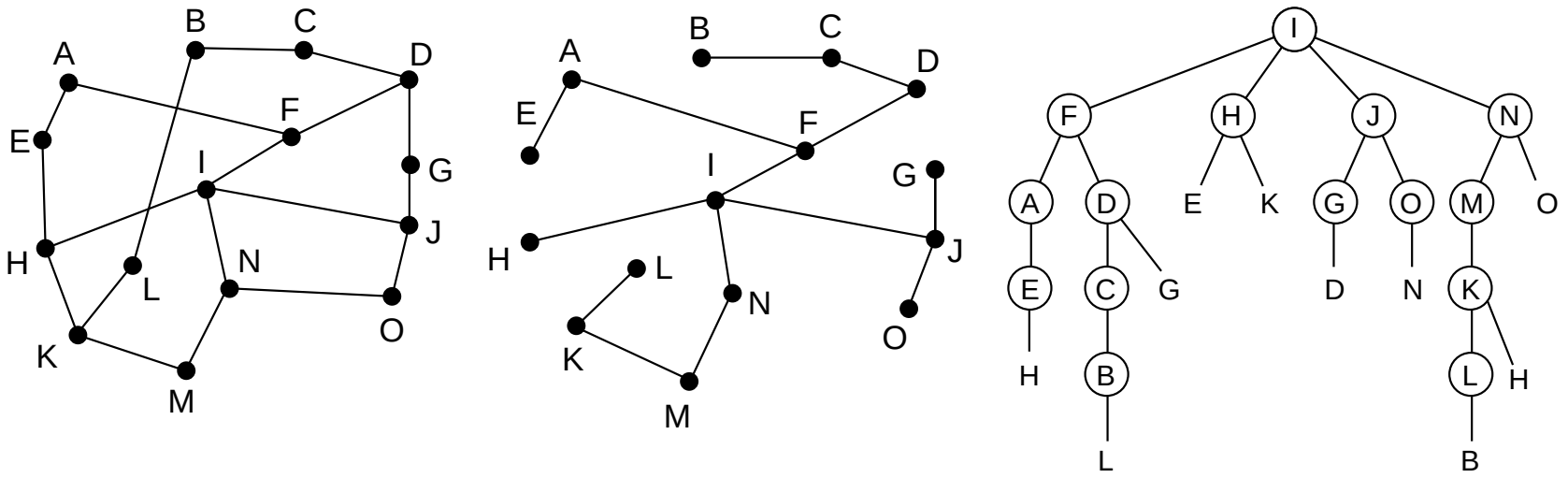
Mögliche Broadcast Routing Verfahren (Fortsetzung):

- **multidestination routing:** Jedes Paket enthält eine Liste mit seinen Zielen. Jeder Router prüft diese Liste und versendet Kopien mit angepaßter Liste an die notwendigen Nachbarn.
  - Wie “separates Senden”, nur das parallele Pakete zusammengefaßt werden
- **reverse path forwarding:** Ein Router prüft für ein ankommendes Paket ob es auf der Leitung ankommt, die üblicherweise zum Senden an die Quelladresse des Broadcast benutzt wird.
  - ankommende Leitung verschieden: Paket verwerfen
  - ankommende Leitung stimmt überein: Paket weiterleiten

Broadcast Routing: Reverse Path Forwarding

Beispiel:

- a) Gesamtes (Sub-)Netz mit Knoten A...O
- b) Baum von Knoten I
- c) Baum von Knoten I für Reverse Path Forwarding



(c) Tanenbaum, Computer Networks

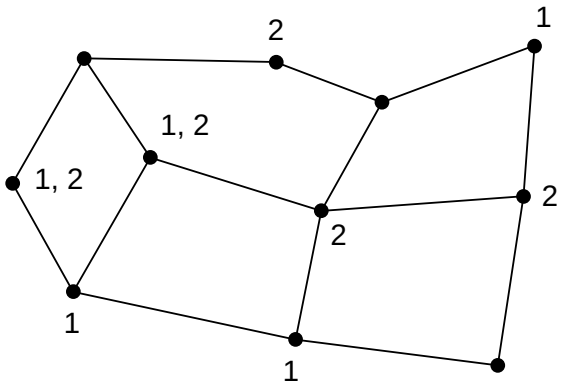
Multicast Routing

Multicast: Senden einer Nachricht an eine definierte Gruppe

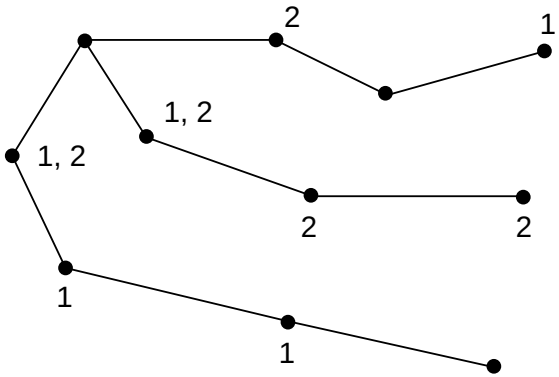
- sehr kleine Gruppe $\rightarrow$ einzelnes Routing für jeden Empfänger
- sehr große Gruppe $\rightarrow$ Broadcast Routing

Multicast Routing:

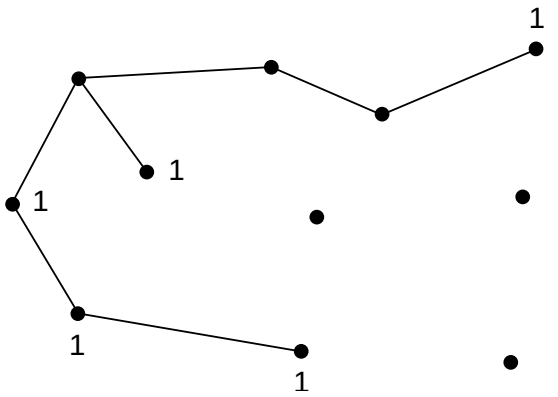
- erfordert Management der Gruppen durch die Router
- Router müssen über neue Gruppen informiert werden
- jeder Router berechnet Baum für jede Gruppe (hoher Speicheraufwand)
- jeder Router leitet Pakete nur an die Knoten weiter, die Teil des Baums der Gruppe sind (verkürzter Baum)

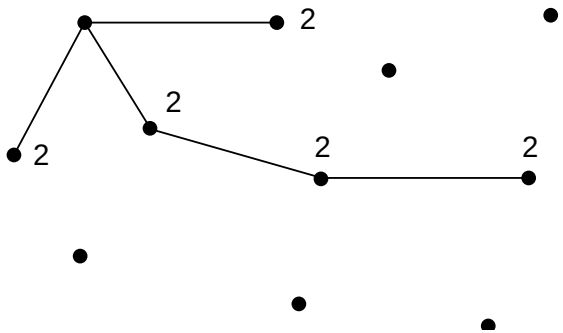


(a)



(b)





a) gesamte Netzwerk
b) Baum des linken Knoten
c) Baum für Multicast Gruppe 1
d) Baum für Multicast Gruppe 2

(c) Tanenbaum, Computer Networks

Beispiel: Routingtabelle des lokalen Rechners

Windows prompt `>route print`

```
===========================================================================
Schnittstellenliste
11 ...00 1c bf XY ZX YZ ...... Intel(R) PRO/Wireless 3945ABG Network Connec
10 ...00 16 d3 XY ZX YZ ...... Intel(R) 82566MM Gigabit Network Connection
1 ........................... Software Loopback Interface 1
21 ...00 00 00 00 00 00 00 e0 Microsoft-ISATAP-Adapter #2
23 ...00 00 00 00 00 00 00 e0 Microsoft-6zu4-Adapter
14 ...02 00 54 XY ZX YZ ...... Teredo Tunneling Pseudo-Interface
===========================================================================
IPv4-Routentabelle
===========================================================================
Aktive Routen:
     Netzwerkziel    Netzwerkmaske          Gateway    Schnittstelle  Metrik
          0.0.0.0          0.0.0.0    134.130.XX.YY    134.130.XX.YYY     10
          0.0.0.0          0.0.0.0      134.61.XX.Y       134.61.YY.Z     25
        127.0.0.0        255.0.0.0   Auf Verbindung         127.0.0.1    306
        127.0.0.1  255.255.255.255   Auf Verbindung         127.0.0.1    306
  127.255.255.255  255.255.255.255   Auf Verbindung         127.0.0.1    306
      134.61.XX.Z    255.255.248.0   Auf Verbindung       134.61.XX.Z    281
      134.61.XX.Z  255.255.255.255   Auf Verbindung       134.61.XX.Z    281
     134.61.YY.ZZZ  255.255.255.255   Auf Verbindung       134.61.XX.Z    281
     134.130.XX.XX  255.255.255.240   Auf Verbindung    134.130.XX.ZZZ    266
    134.130.XX.ZZZ  255.255.255.255   Auf Verbindung    134.130.XX.ZZZ    266
    134.130.XX.XYZ  255.255.255.255   Auf Verbindung    134.130.XX.ZZZ    266
        224.0.0.0        240.0.0.0   Auf Verbindung         127.0.0.1    306
...
  255.255.255.255  255.255.255.255   Auf Verbindung         127.0.0.1    306
```