

---

# Rechnernetze I

SoSe 2025

Roland Wismüller  
Universität Siegen  
roland.wismueller@uni-siegen.de  
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 1. April 2025

---

# Rechnernetze I

SoSe 2025

## 6 Routing

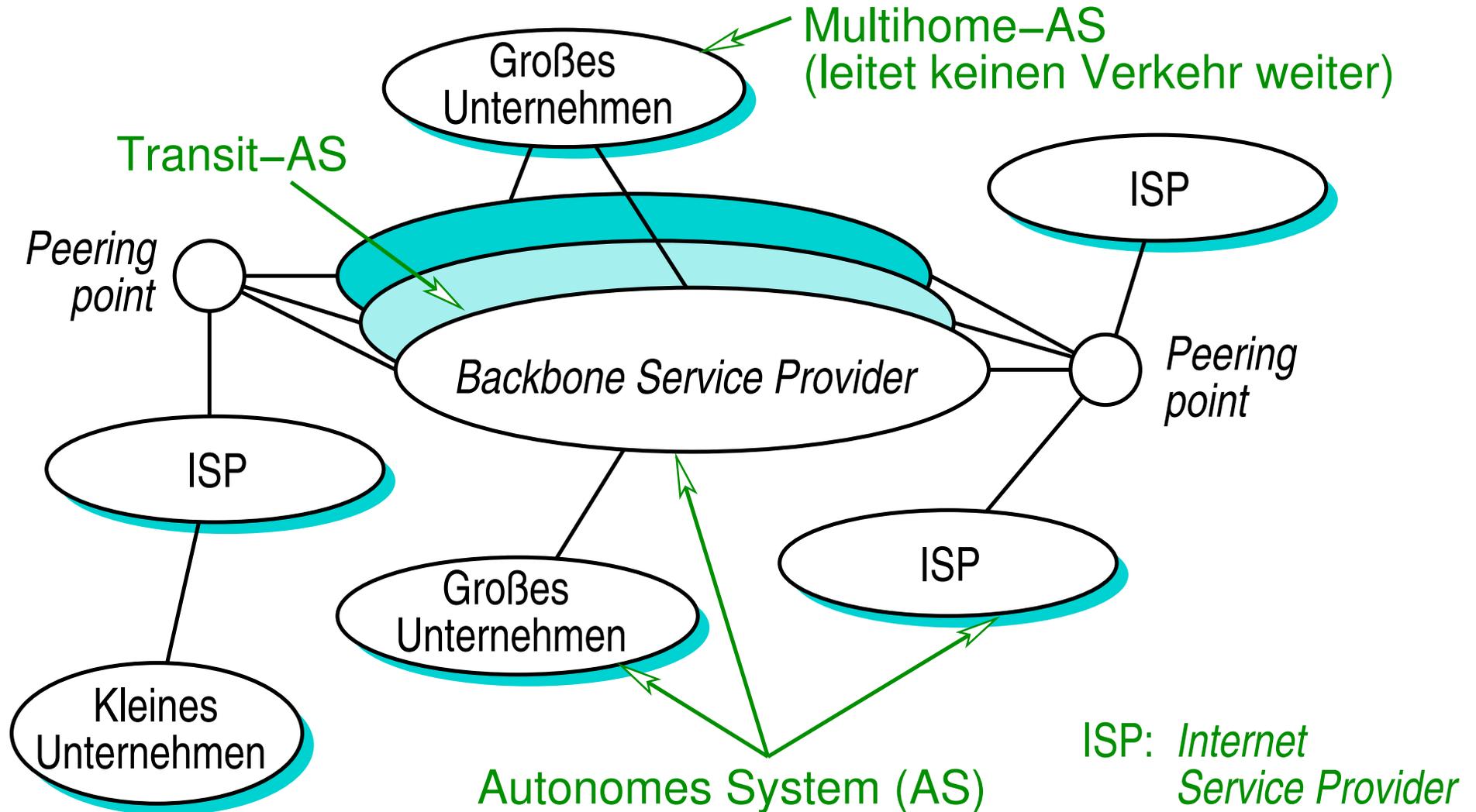


## Inhalt

- ➔ Einführung
- ➔ Routing innerhalb einer Domain
  - ➔ *Distance Vector Routing* (RIP, EIGRP)
  - ➔ *Link State Routing* (OSPF)
- ➔ Interdomain-Routing
  - ➔ *Border Gateway Protocol* (BGP)
  
- ➔ Peterson, Kap. 4.2.1 – 4.2.4



## Heutige Struktur des Internet





## Routing im Internet

- ➔ Heute über 76.000 autonome Systeme (AS)
  - ➔ Backbones, Provider, Endbenutzer
  - ➔ Netzwerk-Adressen werden an AS zugewiesen
- ➔ Probleme: Skalierbarkeit, heterogene Administration
- ➔ Hierarchischer Ansatz: **Routing Domains**
  - ➔ Routing innerhalb eines administrativen Bereichs (z.B. Campus, Unternehmen, Provider)
    - ➔ *Interior Gateway Protocols* (IGP)
      - ➔ z.B. RIP, OSPF, EIGRP
  - ➔ Interdomain Routing (zwischen Teilnetzen des Internet)
    - ➔ *Exterior Gateway Protocols* (EGP)
      - ➔ z.B. BGP (*Border Gateway Protocol*)



### Routing als Graph-Problem

➔ Knoten:

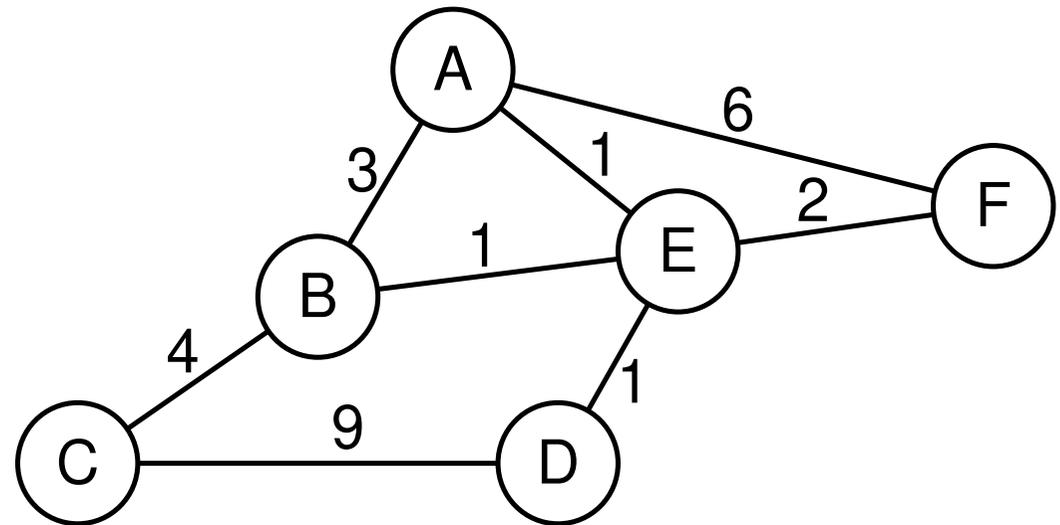
➔ Router

➔ (Hosts)

➔ (Netzwerke)

➔ Kanten: Verbindungen

➔ mit Kosten (Metrik)  
(symmetrisch oder asymmetrisch)



➔ Aufgabe des Routings:

➔ finde Pfade mit geringsten Kosten zwischen allen Paaren von Knoten



### Statisches Routing

- ➔ Pfade werden manuell bestimmt und in Tabellen eingetragen
- ➔ Probleme:
  - ➔ Ausfall von Knoten / Verbindungen
  - ➔ neue Knoten / Verbindungen
  - ➔ dynamische Änderung der Verbindungskosten (Last)

### Dynamisches Routing

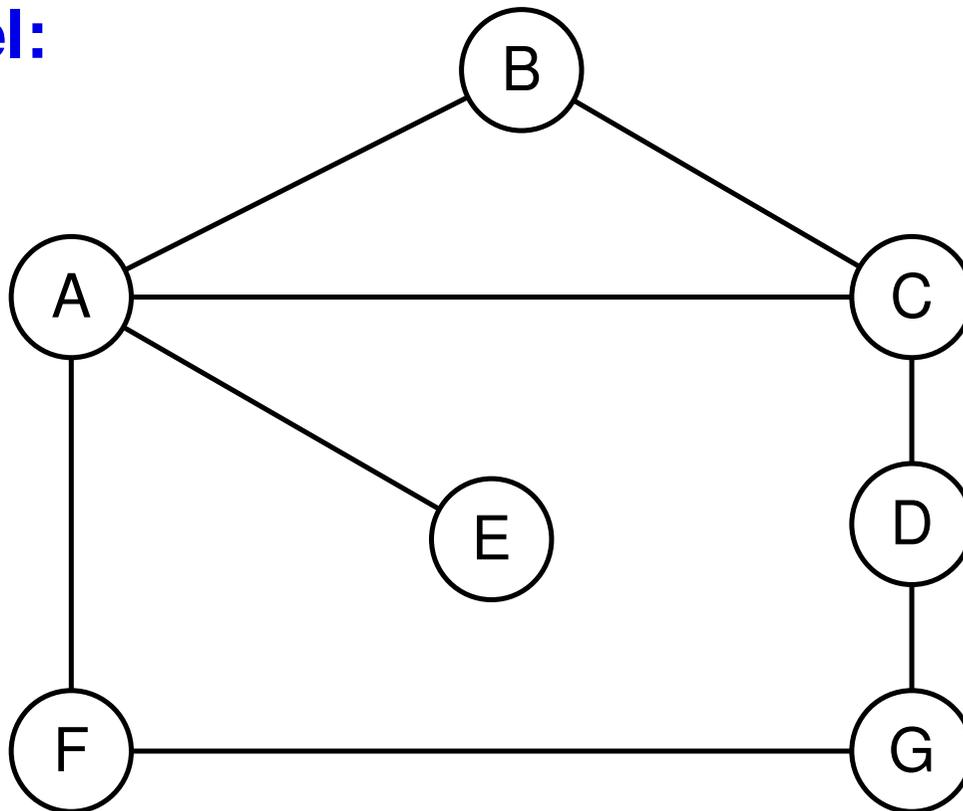
- ➔ Verteilte Algorithmen zum Aufbau der Tabellen
- ➔ Anforderungen:
  - ➔ schnelle Konvergenz / Skalierbarkeit
  - ➔ einfache Administration

## 6.2.1 Distance Vector Routing



- ➔ Router kennen nur Distanz und „Richtung“ (*Next Hop*) zum Ziel
  - ➔ nur diese Informationen werden (mit Nachbarn) ausgetauscht
- ➔ Verteilter Algorithmus zur Erstellung der Routing-Tabellen
  - ➔ typisch: Bellman-Ford-Algorithmus

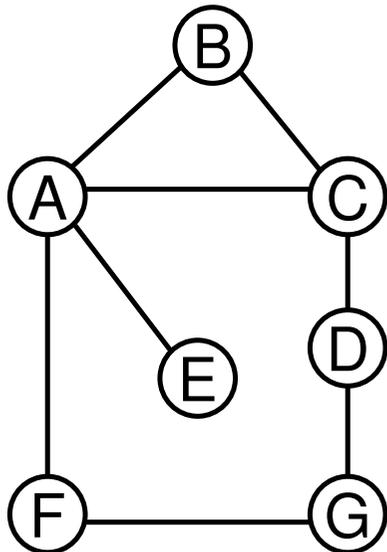
### Beispiel:



Vereinfachung:  
alle Link-Kosten  
seien 1

## Beispiel: ...

➔ Initial besitzen die Router Information über die folgenden Distanzvektoren (verteilt!)



		Distanz zu Knoten						
Information bei	A	B	C	D	E	F	G	
A	0	1	1	$\infty$	1	1	$\infty$	
B	1	0	1	$\infty$	$\infty$	$\infty$	$\infty$	
C	1	1	0	1	$\infty$	$\infty$	$\infty$	
D	$\infty$	$\infty$	1	0	$\infty$	$\infty$	1	
E	1	$\infty$	$\infty$	$\infty$	0	$\infty$	$\infty$	
F	1	$\infty$	$\infty$	$\infty$	$\infty$	0	1	
G	$\infty$	$\infty$	$\infty$	1	$\infty$	1	0	

### Beispiel: ...

➔ Initiale Routing-Tabelle von A:

➔ A kennt nur seine direkten Nachbarn

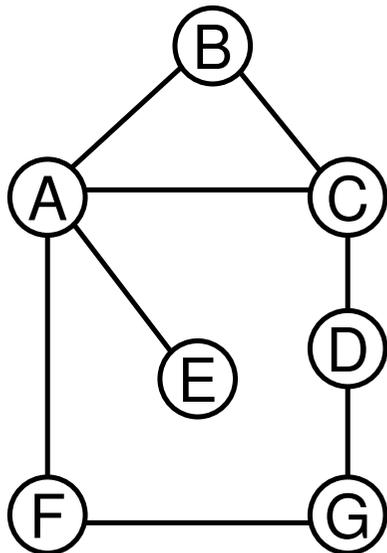
Ziel	Kosten	NextHop
B	1	B
C	1	C
E	1	E
F	1	F

### Vorgehensweise

- ➔ Nachrichtenaustausch zur Erstellung der Routing-Tabellen:
  - ➔ Router senden ihren Distanzvektor an alle direkten Nachbarn
  - ➔ bessere Routen werden in den eigenen Distanzvektor (und die Routing-Tabelle) übernommen
    - ➔ aber auch schlechtere vom Next Hop
- ➔ Nach mehreren Runden des Nachrichtenaustauschs konvergieren die Distanzvektoren ... jedenfalls meistens !!
- ➔ Der Nachrichtenaustausch erfolgt
  - ➔ periodisch (typ. alle 30 s)
  - ➔ bei Änderung des Distanzvektors eines Knotens
    - ➔ z.B. Ausfall einer Verbindung, neue Verbindung
  - ➔ auf Anfrage (z.B. bei Neustart eines Routers)

### Beispiel ...

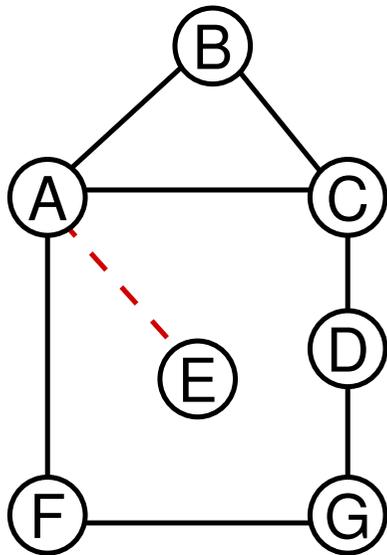
➔ Distanzvektoren nach dem Konvergieren des Verfahrens:



	Distanz zu Knoten						
Information bei	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

### Problem des Algorithmus: *Count-to-Infinity*

➔ Beispiel: A erkennt Ausfall der Verbindung zu E



Zeitliche Entwicklung der Distanz-Vektoren zu E:

A:  $(\infty, -)$

B:  $(2, A)$

C:  $(2, A)$

Distanz zu E

Next Hop

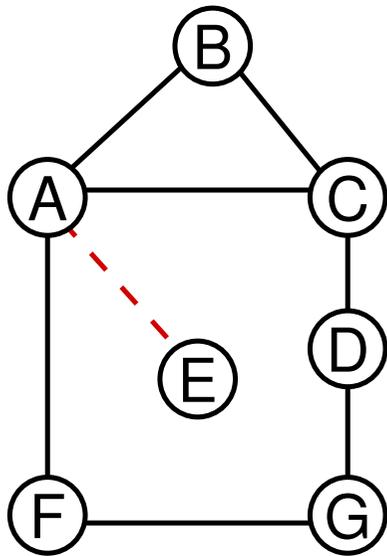
Zeit →

➔ Keine wirklich gute, allgemeine Lösung des Problems

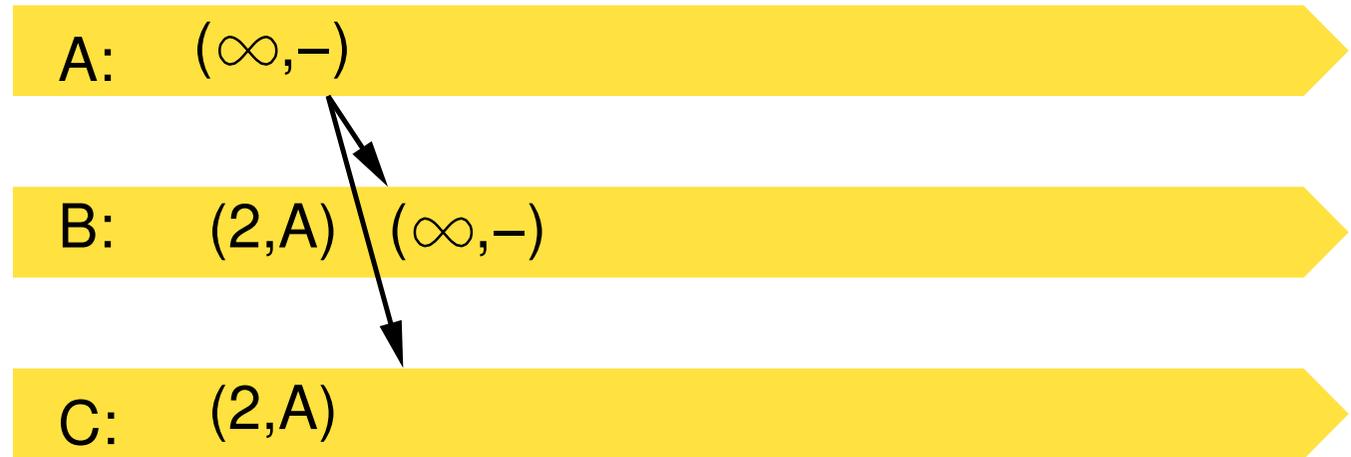
➔ pragmatisch: beschränke  $\infty$  auf den Wert 16

### Problem des Algorithmus: *Count-to-Infinity*

➔ Beispiel: A erkennt Ausfall der Verbindung zu E



Zeitliche Entwicklung der Distanz-Vektoren zu E:



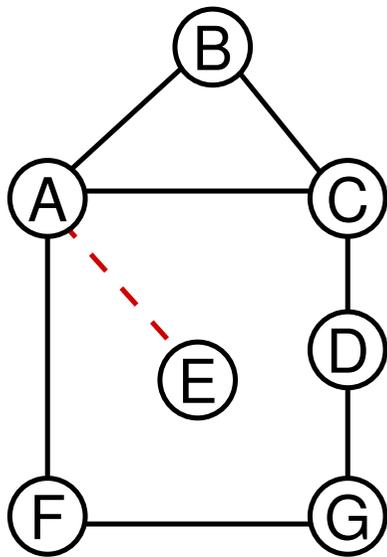
Zeit →

➔ Keine wirklich gute, allgemeine Lösung des Problems

➔ pragmatisch: beschränke  $\infty$  auf den Wert 16

### Problem des Algorithmus: *Count-to-Infinity*

➔ Beispiel: A erkennt Ausfall der Verbindung zu E



Zeitliche Entwicklung der Distanz-Vektoren zu E:

A:  $(\infty, -)$

B:  $(2, A)$   $(\infty, -)$   $(3, C)$

C:  $(2, A)$

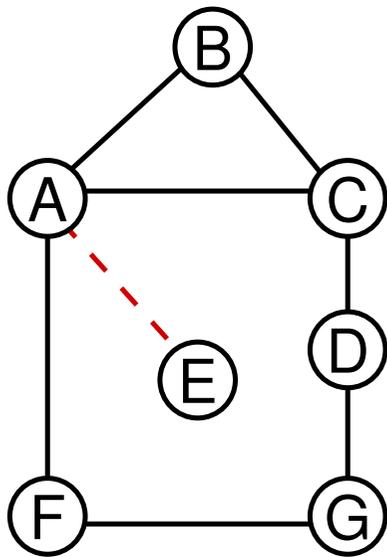
Zeit →

➔ Keine wirklich gute, allgemeine Lösung des Problems

➔ pragmatisch: beschränke  $\infty$  auf den Wert 16

### Problem des Algorithmus: *Count-to-Infinity*

➔ Beispiel: A erkennt Ausfall der Verbindung zu E



Zeitliche Entwicklung der Distanz-Vektoren zu E:

A:  $(\infty, -)$

B:  $(2, A)$   $(\infty, -)$   $(3, C)$

C:  $(2, A)$   $(\infty, -)$

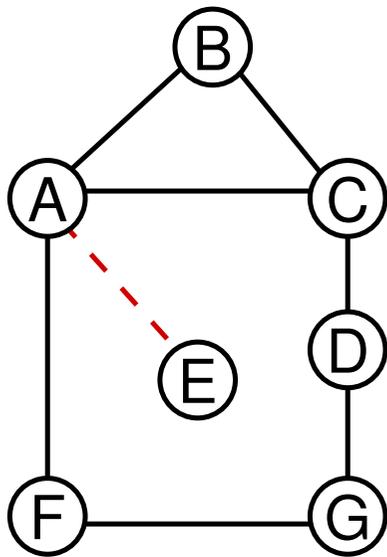
Zeit →

➔ Keine wirklich gute, allgemeine Lösung des Problems

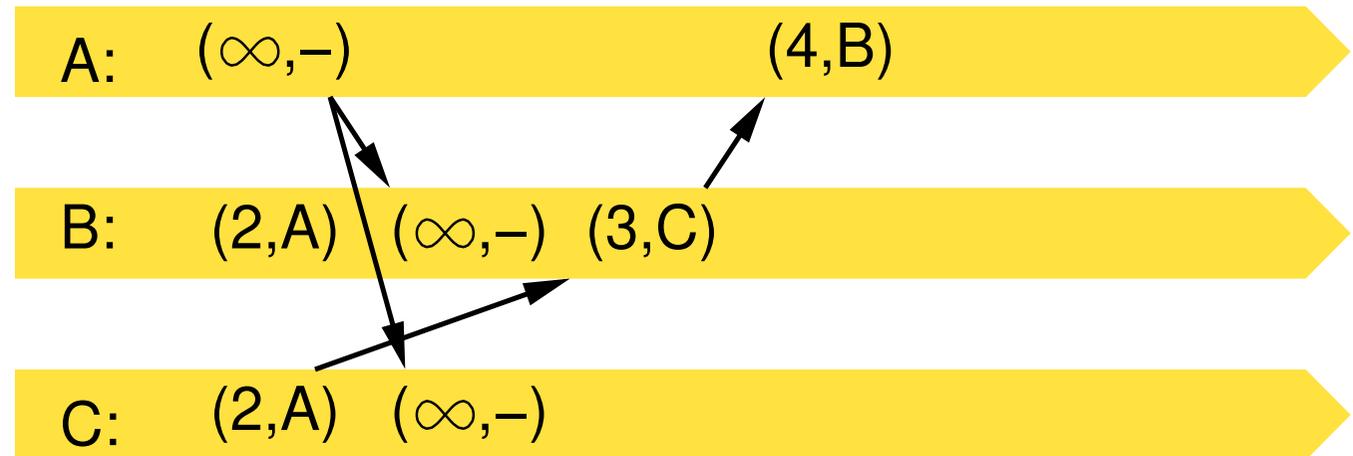
➔ pragmatisch: beschränke  $\infty$  auf den Wert 16

### Problem des Algorithmus: *Count-to-Infinity*

➔ Beispiel: A erkennt Ausfall der Verbindung zu E



Zeitliche Entwicklung der Distanz-Vektoren zu E:



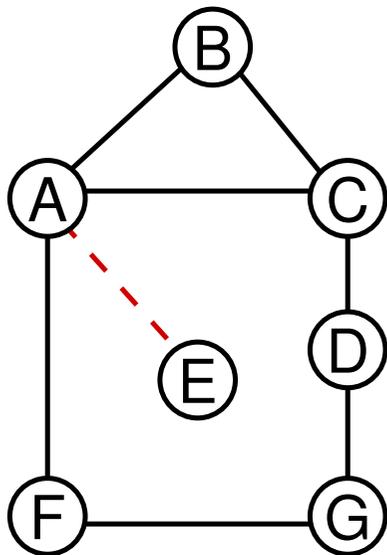
Zeit →

➔ Keine wirklich gute, allgemeine Lösung des Problems

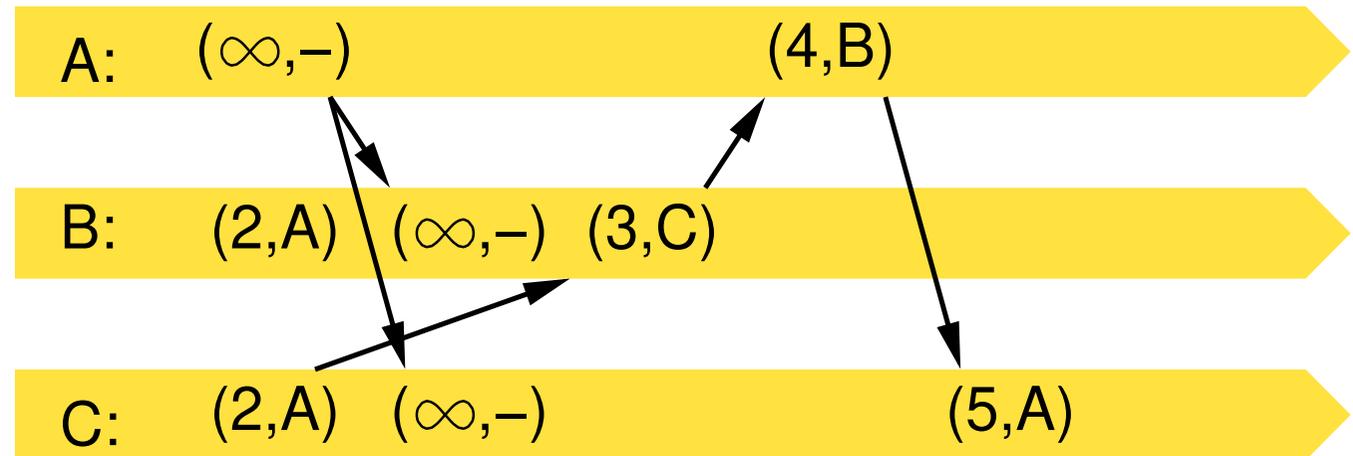
➔ pragmatisch: beschränke  $\infty$  auf den Wert 16

### Problem des Algorithmus: *Count-to-Infinity*

➔ Beispiel: A erkennt Ausfall der Verbindung zu E



Zeitliche Entwicklung der Distanz-Vektoren zu E:



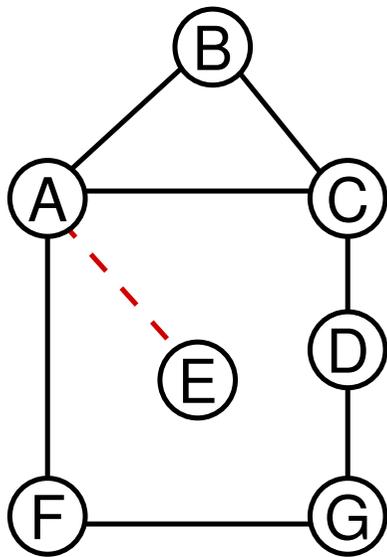
Zeit →

➔ Keine wirklich gute, allgemeine Lösung des Problems

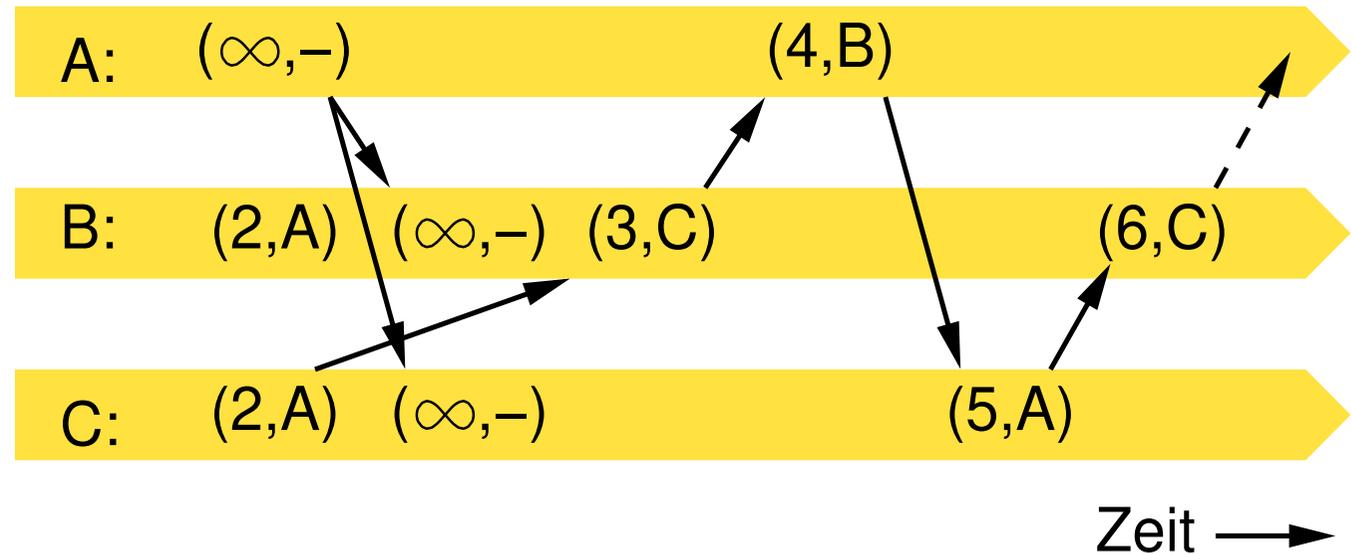
➔ pragmatisch: beschränke  $\infty$  auf den Wert 16

### Problem des Algorithmus: *Count-to-Infinity*

➔ Beispiel: A erkennt Ausfall der Verbindung zu E



Zeitliche Entwicklung der Distanz-Vektoren zu E:

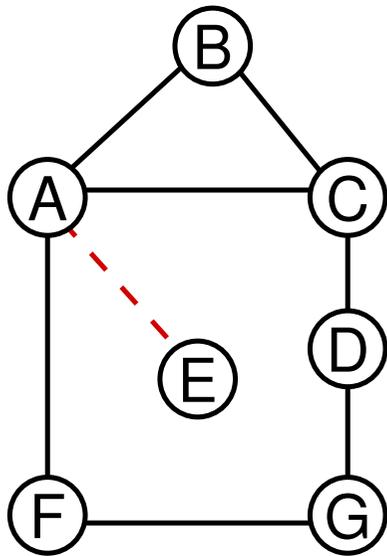


➔ Keine wirklich gute, allgemeine Lösung des Problems

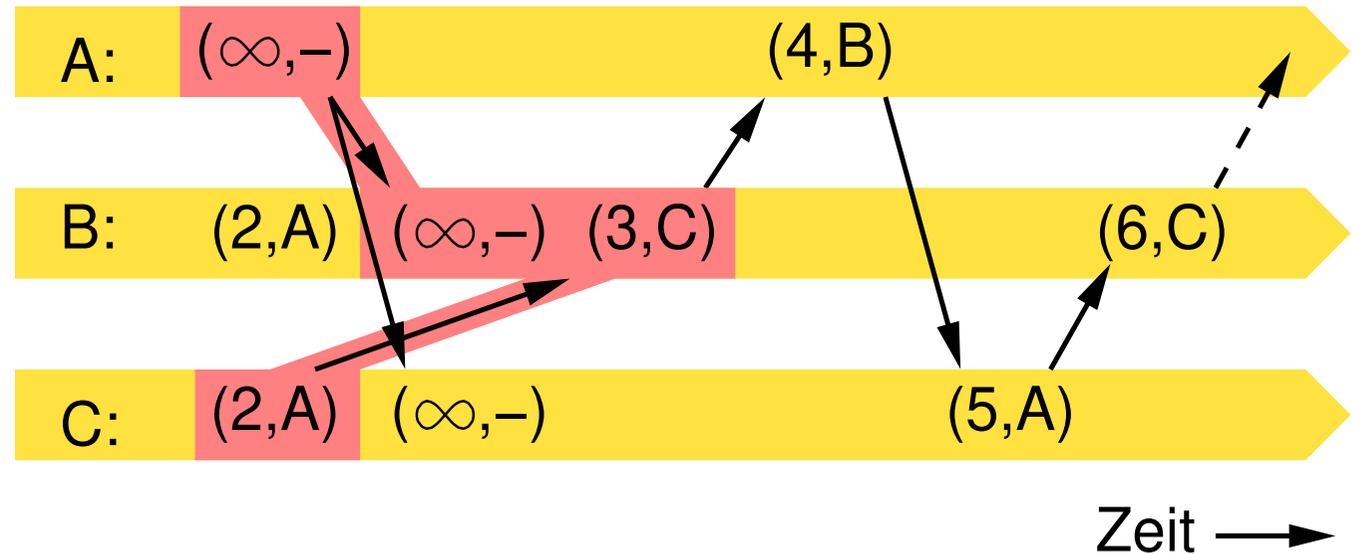
➔ pragmatisch: beschränke  $\infty$  auf den Wert 16

### Problem des Algorithmus: *Count-to-Infinity*

➔ Beispiel: A erkennt Ausfall der Verbindung zu E



Zeitliche Entwicklung der Distanz-Vektoren zu E:



➔ Keine wirklich gute, allgemeine Lösung des Problems

➔ pragmatisch: beschränke  $\infty$  auf den Wert 16

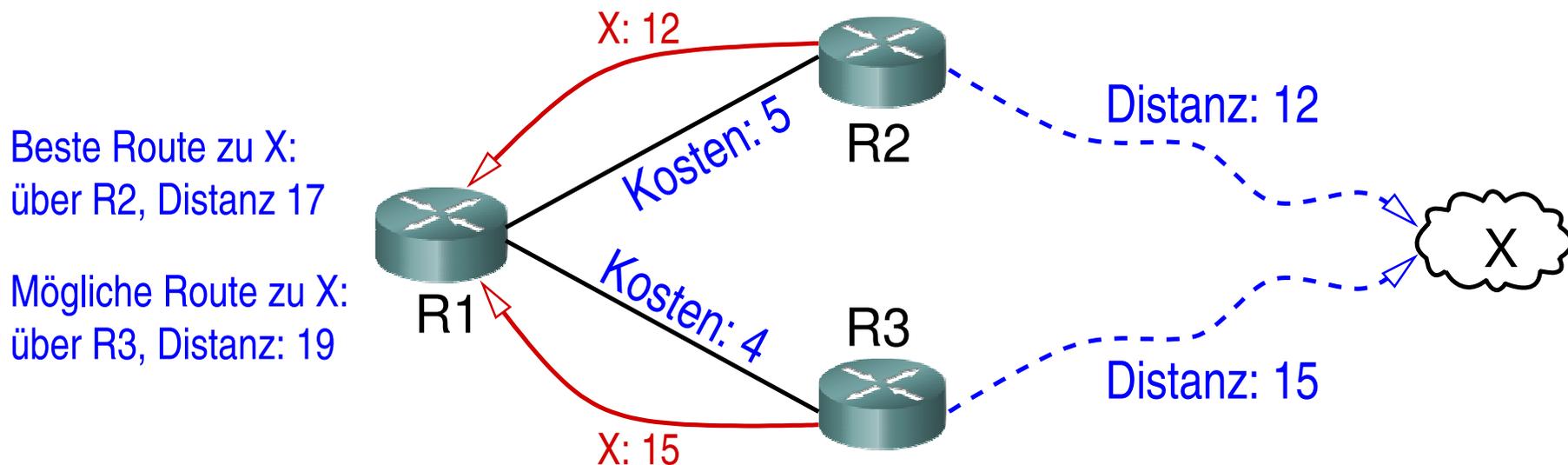


### **RIP (*Routing Information Protocol*)**

- ➔ Einfaches *Distance Vector Routing* Protokoll
- ➔ Internet-Standard
- ➔ Alle Link-Kosten sind 1, d.h. Distanz = *Hop Count*
- ➔ Drei Versionen:
  - ➔ RIPv1: leitet keine Präfixlänge weiter
    - ➔ Subnetting nur möglich, wenn alle Subnetze des klassen-behafteten Netzes dieselbe Größe haben
    - ➔ d.h. Subnetzmaske ist global
  - ➔ RIPv2: ermöglicht klassenloses Routing
  - ➔ RIPv6: unterstützt IPv6

### EIGRP (*Enhanced Interior Gateway Routing Protocol*)

- ➔ Erweitertes *Distance Vector Routing* Protokoll
- ➔ Cisco-proprietär, seit 2013 offener Internet-Standard
- ➔ Link-Kosten berücksichtigen Bandbreite und Latenz
- ➔ Unterstützt IPv4, IPv6 und andere Schicht-3-Protokolle
- ➔ Updates nur bei Änderungen, kein *Count-to-Infinity*
- ➔ Behält alle Routen, nicht nur die beste



- ➔ Grund„problem“ beim *Distance Vector Routing*:
  - ➔ Router haben ausschließlich lokale Information
- ➔ ***Link State Routing***:
  - ➔ Router erhalten Information über die Struktur des gesamten Netzwerks
- ➔ Vorgehensweise:
  - ➔ Kennenlernen der direkten Nachbarn
    - ➔ einschließlich der Link-Kosten
  - ➔ Versenden von Link State Paketen an **alle** anderen Router (***Reliable Flooding***)
  - ➔ Berechnung der kürzesten Wege mit Dijkstra-Algorithmus

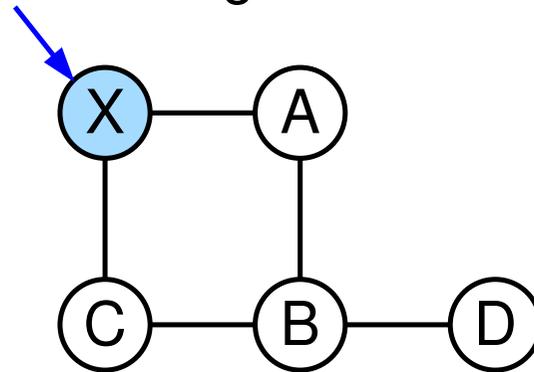
### Link State Pakete

- ➔ Inhalt eines Link State Pakets:
  - ➔ ID des erzeugenden Routers
  - ➔ Liste der direkten Nachbarn mit Link-Kosten
  - ➔ Sequenznummer
    - ➔ Paket nur weitergeleitet, wenn die Sequenznummer größer als die des letzten weitergeleiteten Pakets ist
  - ➔ *Time-to-Live* (TTL)
    - ➔ jeder Router dekrementiert TTL
    - ➔ bei TTL = 0 wird das Paket gelöscht
- ➔ Versenden der Link State Pakete
  - ➔ Periodisch ( $\sim$  Stunden) oder bei Topologie-Änderungen

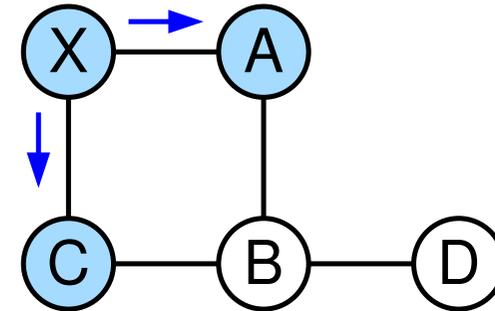
### Reliable Flooding

➔ Flooding mit ACK und ggf. wiederholter Übertragung

(1) X erzeugt Paket

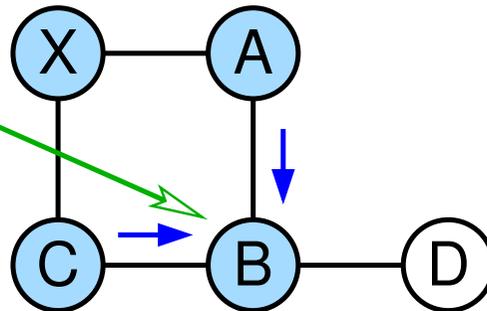


(2) X sendet an A und C

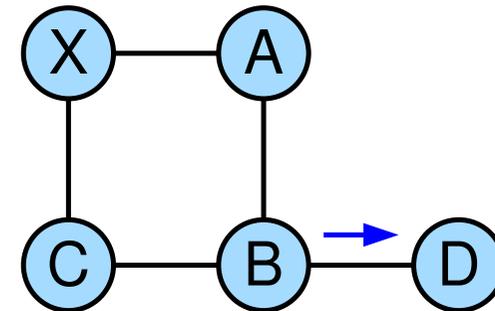


Sequenznummer verhindert, daß B Paket zweimal an D sendet

(3) A und C senden an B



(4) B sendet an D



### Dijkstra-Algorithmus zur Bestimmung kürzester Wege

- ➔ Eingabe:  $N$ : Knotenmenge,  $l(i, j)$ : Link-Kosten,  $s$ : Startknoten
- ➔ Ausgabe:  $C(n)$ : Pfad-Kosten von  $s$  zu  $n$
- ➔ Algorithmus:

$$M = \{s\}$$

**Für alle**  $n \in N - \{s\}$  :  $C(n) = l(s, n)$

**Solange**  $M \neq N$  :

Wähle  $w \in N - M$  so, daß  $C(w)$  minimal ist

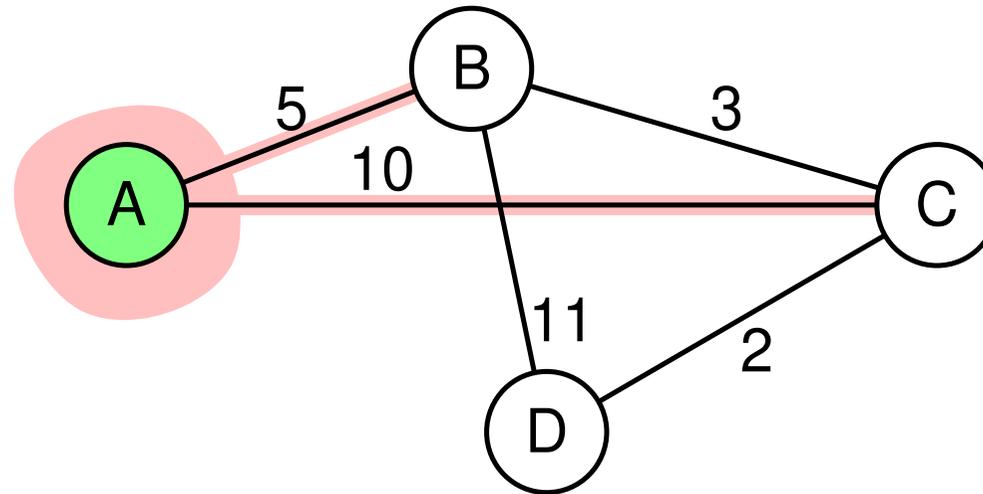
$$M = M \cup \{w\}$$

**Für alle**  $n \in N - M$  :

$$C(n) = \min( C(n), C(w) + l(w, n) )$$

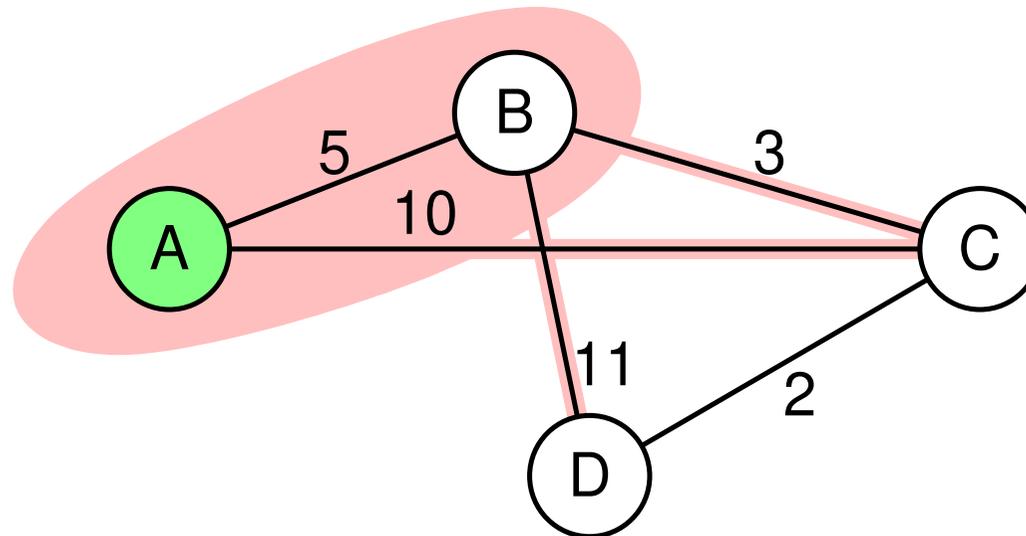


## Beispiel



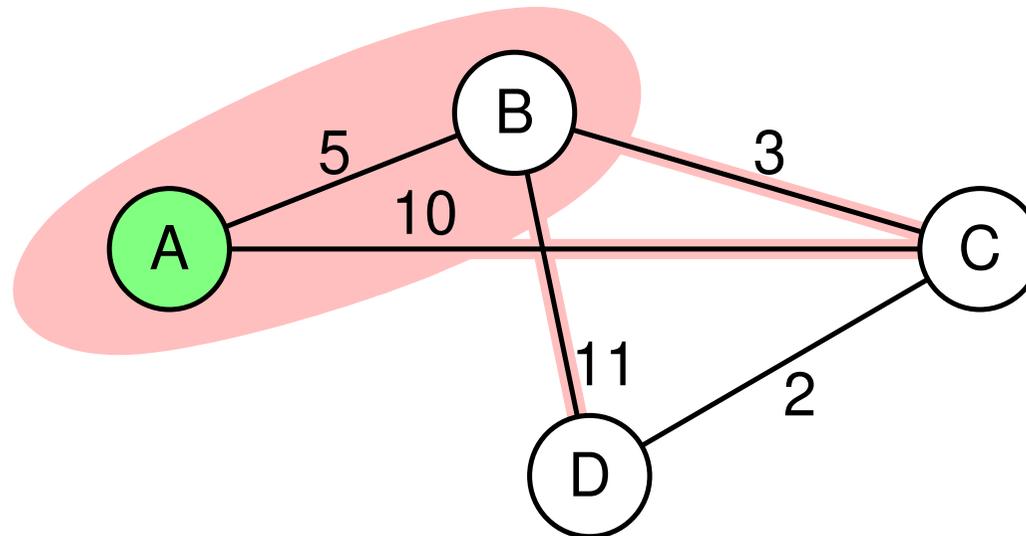
Schritt	$w$	$M$	$C(B)$	$C(C)$	$C(D)$
Initial		{A}	5	10	$\infty$

### Beispiel



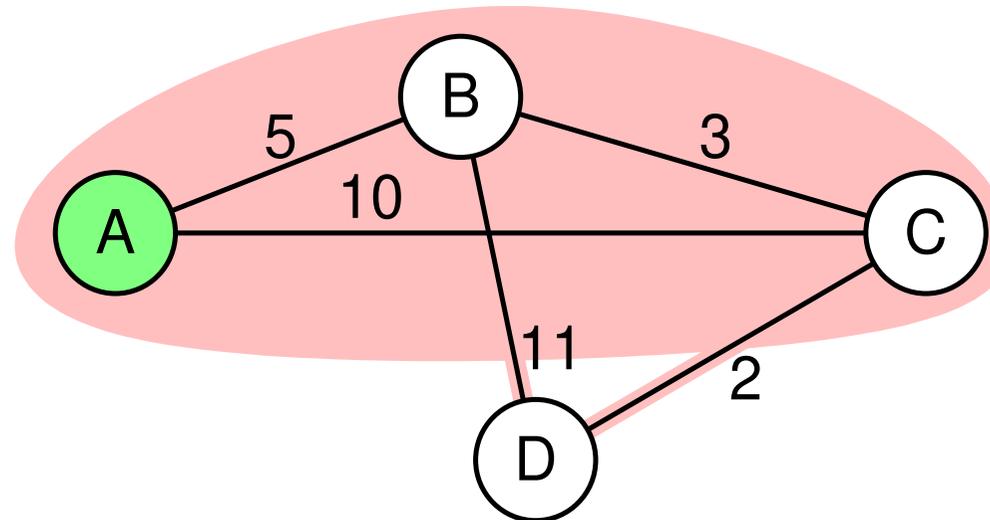
Schritt	$w$	$M$	$C(B)$	$C(C)$	$C(D)$
Initial		{A}	5	10	$\infty$
1	B	{A,B}	5		

### Beispiel



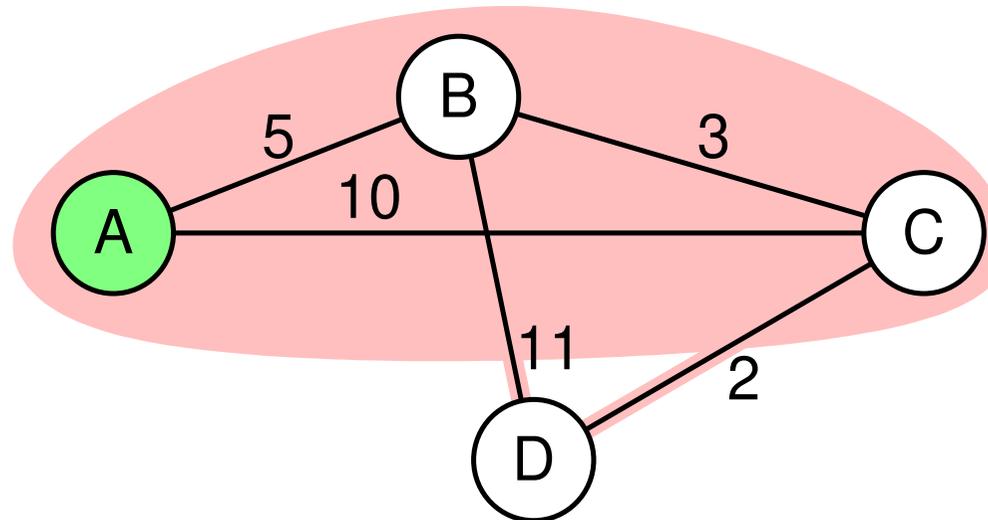
Schritt	$w$	$M$	$C(B)$	$C(C)$	$C(D)$
Initial		{A}	5	10	$\infty$
1	B	{A,B}	5	8	16

### Beispiel



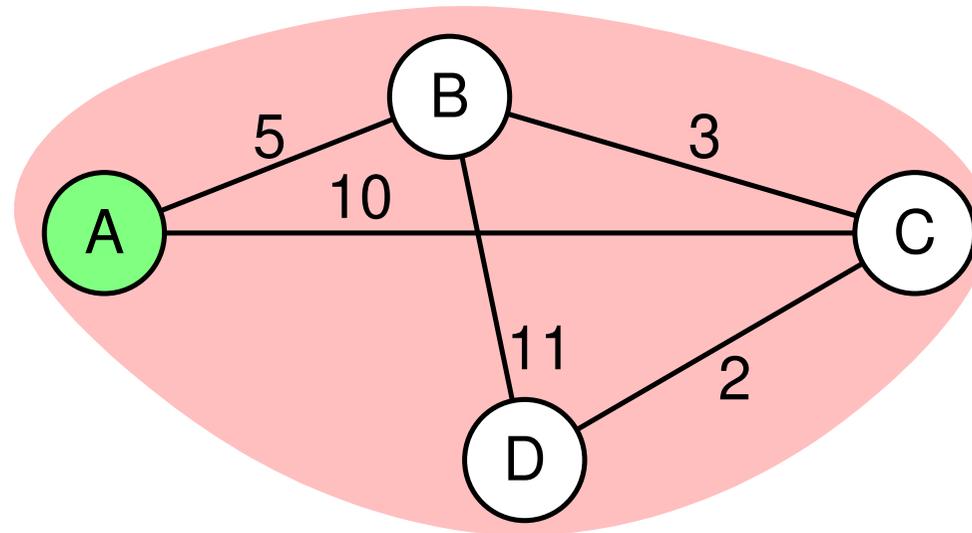
Schritt	$w$	$M$	$C(B)$	$C(C)$	$C(D)$
Initial		{A}	5	10	$\infty$
1	B	{A,B}	5	8	16
2	C	{A,B,C}	5	8	

### Beispiel



Schritt	$w$	$M$	$C(B)$	$C(C)$	$C(D)$
Initial		{A}	5	10	$\infty$
1	B	{A,B}	5	8	16
2	C	{A,B,C}	5	8	10

### Beispiel



Schritt	$w$	$M$	$C(B)$	$C(C)$	$C(D)$
Initial		{A}	5	10	$\infty$
1	B	{A,B}	5	8	16
2	C	{A,B,C}	5	8	10
3	D	{A,B,C,D}	5	8	10

### OSPF (*Open Shortest Path First*)

- ➔ Weit verbreitetes *Link State Routing* Protokoll
- ➔ Internet-Standard
- ➔ Link-Kosten vom Protokoll nicht festgelegt
  - ➔ in der Praxis meist Link-Bandbreite verwendet
- ➔ Versionen:
  - ➔ OSPFv2: für IPv4
  - ➔ OSPFv3: unterstützt IPv6

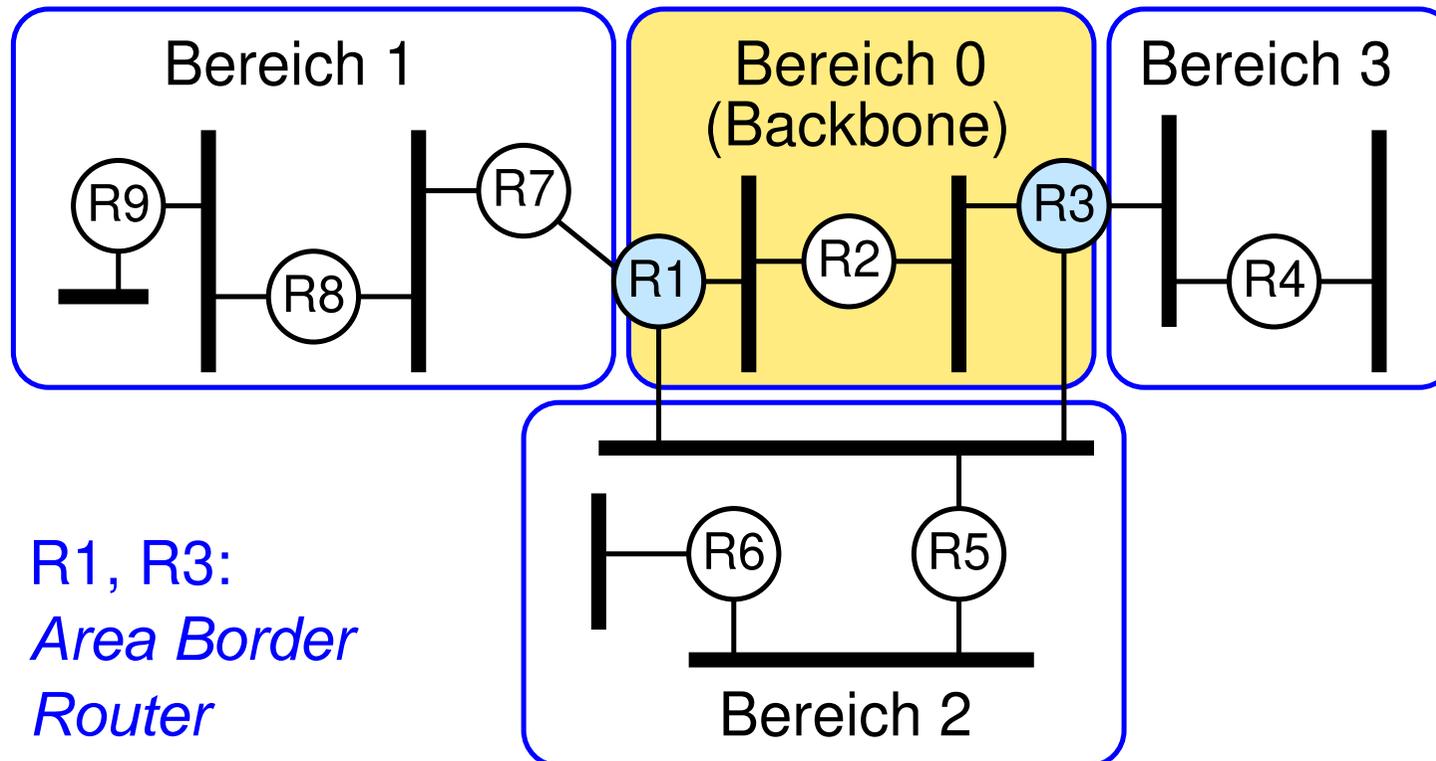


### OSPF (*Open Shortest Path First*) ...

- ➔ Wichtige Pakettypen:
  - ➔ *Hello*-Pakete: bauen Nachbarschaftsbeziehungen (Adjazenzen) auf und testen diese
  - ➔ *Link State Update*-Pakete enthalten *Link State Advertisements* (LSAs)
    - ➔ LSA beschreibt Verbindung zwischen zwei Routern
- ➔ In Mehrfachzugriffsnetzen:
  - ➔ Router wählen einen „designierten“ Router (DR) und einen Backup DR (BDR)
    - ➔ bei Ausfall DR: BDR wird neuer DR, Neuwahl BDR
  - ➔ jeder Router hat (nur) DR als Nachbarn
    - ➔ LSAs werden nur an DR gesendet und von diesem verteilt
    - ➔ verhindert exzessives Flooding im Netz

### Multi-Area OSPF

- ➔ Für große *Routing-Domains*:
  - ➔ OSPF erlaubt Einführung einer weiteren Hierarchieebene:  
**Routing-Bereiche (Areas)**
- ➔ Beispiel:





### Multi-Area OSPF ...

- ➔ Innerhalb jedes Bereichs: *Flooding* von Link-State-Paketen
- ➔ *Area Border Router* (ABR) geben diese nicht weiter
- ➔ Stattdessen: ABR sendet zusammengefaßte Information
  - ➔ nur ein Link-State-Paket für den gesamten Bereich
  - ➔ ABR spiegelt vor, daß alle Hosts in seinem Bereich **direkt** mit ihm verbunden sind
- ➔ Pakete zwischen Bereichen immer über ABR geleitet
  - ➔ bei mehreren ABR: automatische Auswahl über die Link-Kosten
- ➔ Damit: bessere Skalierbarkeit
  - ➔ kleinere Graphen in den einzelnen Routing-Bereichen
  - ➔ weniger Neuberechnungen der kürzesten Wege
  - ➔ evtl. aber suboptimale Routen



### Routing innerhalb und außerhalb von Domains

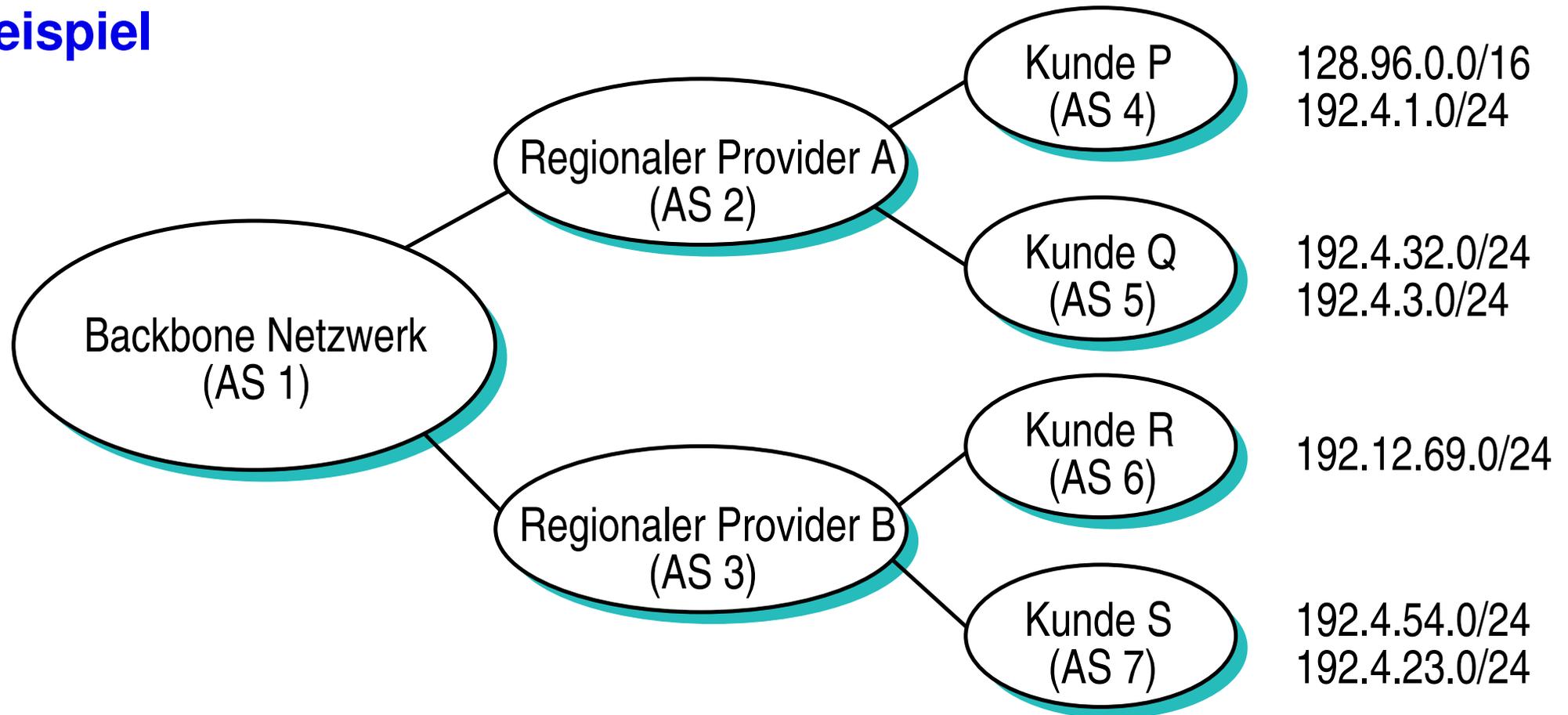
- ➔ Innerhalb einer Domain: RIP, OSPF
  - ➔ Bestimmung optimaler Routen
- ➔ Zwischen Domains: **Border Gateway Protocol (BGP)**
  - ➔ Autonome Systeme  $\Rightarrow$  keine gemeinsame Metrik
  - ➔ Routen werden „politisch“ bestimmt
    - ➔ „benutze Provider B für Adressen xyz“
    - ➔ „benutze Pfad über möglichst wenige AS“
  - ➔ Wichtigstes Ziel: Erreichbarkeit
    - ➔ „gute“ Routen sind sekundär

### Routing mit BGP

- ➔ Jedes Autonome System (AS) hat
  - ➔ ein oder mehrere *Border Router*
    - ➔ Verbindung zu anderen AS
  - ➔ einen BGP Sprecher, der bekanntgibt:
    - ➔ lokales Netzwerk
    - ➔ über dieses AS erreichbare Netzwerke  
(nur, wenn das AS Pakete an andere AS weiterleitet)
- ➔ Bekanntgegeben werden vollständige Pfade
  - ➔ zur Vermeidung von zyklischen Routen



### Beispiel



- ➔ AS 2 gibt (u.a.) bekannt: „ich kann AS 4, AS 5 direkt erreichen“
- ➔ Netze 128.96.0.0/16, 192.4.1.0/24, 192.4.32.0/24, 192.4.3.0/24
- ➔ AS 1 gibt (u.a.) bekannt: „ich kann AS 4, AS 5 über AS 2 erreichen“



- ➔ Routing „im kleinen“ (innerhalb einer Domain):
  - ➔ Suche optimale Pfade
  - ➔ *Distance Vector Routing* (nur mit lokaler Information)
  - ➔ *Link State Routing* (globale Information, *reliable Flooding*)
- ➔ Routing „im großen“ (zwischen Domains)
  - ➔ *Border Gateway Protocol*
    - ➔ Bekanntgabe der Erreichbarkeit
    - ➔ Routenwahl ist „politische“ Entscheidung

### Nächste Lektion:

- ➔ Ende-zu-Ende Protokolle: UDP, TCP