10. Vorlesung Verteilte Architekturen

Dr. Christian Baun

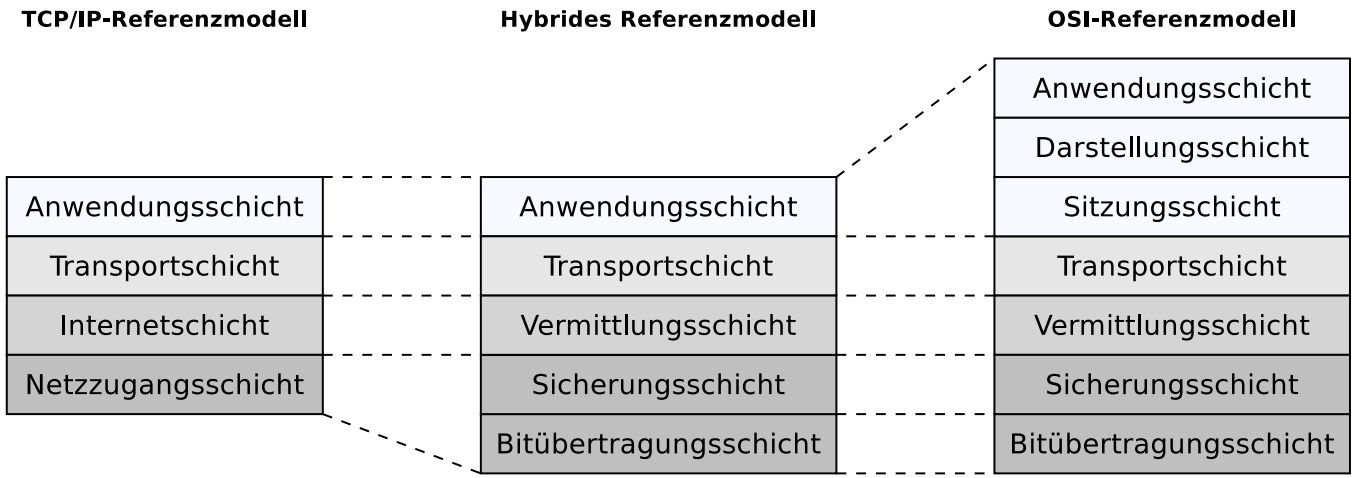
Hochschule Mannheim
Fakultät für Informatik
wolkenrechnen@gmail.com

Heute

- Vermittlungsschicht (Teil 2)
  - Weiterleitung und Wegbestimmung
    - Distanzvektorprotokolle
    - Link-State-Routing-Protokolle
  - Diagnose und Fehlermeldungen mit ICMP

Vermittlungsschicht

- Aufgaben der Vermittlungsschicht (Network Layer):
  - Segmente der Transportschicht in Pakete unterteilen
  - Logische Adressen (IP-Adressen) bereitstellen
  - Routing: Ermittlung des besten Weges
  - Forwarding: Weiterleitung der Pakete zwischen logischen Netzen, also über physische Übertragungsabschnitte hinweg



- Geräte: Router, Layer-3-Switch (Router und Bridge in einem)
- Protokolle: IPv4, IPv6, ICMP, IPX/SPX, DECnet

Weiterleitung und Wegbestimmung

- Primäre Aufgabe der Router: **Weiterleitung (Forwarding)** der IP-Pakete
- Um diese Aufgabe zu erfüllen, müssen die Router für jedes eintreffende Paket die korrekte Schnittstelle (Port) ermitteln
- Jeder Router verwaltet eine lokale **Routing-Tabelle**
  - Die Routing-Tabelle enthält die ihm bekannten **logischen Netze**
  - Aus der Routing-Tabelle geht hervor, über welchen **Port** welches logische Netz erreichbar ist
- Ein Router muss die IP-Pakete also nur in die Richtung versenden, die die Routing-Tabelle vorgibt

Wegbestimmung

- Die **Wegbestimmung** (**Routing**) ist der Prozess, bei dem die Weiterleitungstabellen mit Hilfe verteilter Algorithmen erstellt werden
  - Die Weiterleitungstabellen sind nötig, damit der die Bestimmung des besten Weges, also zu den niedrigsten Kosten, zum Ziel möglich ist
- Das Routing wird durch **Routing-Protokolle** realisiert
  - Diese Routing-Protokolle werden zwischen den Routern ausgeführt
- Routing-Protokolle basieren auf verteilten Algorithmen
  - Grund: Skalierbarkeit
- Es existieren zwei Hauptklassen von Routing-Protokollen:
  - **Distanzvektorprotokolle**, die den Bellman-Ford-Algorithmus verwenden
  - **Link-State-Routing-Protokolle**, die den (Dijkstra-Algorithmus) verwenden

Distanzvektorprotokolle (1/2)

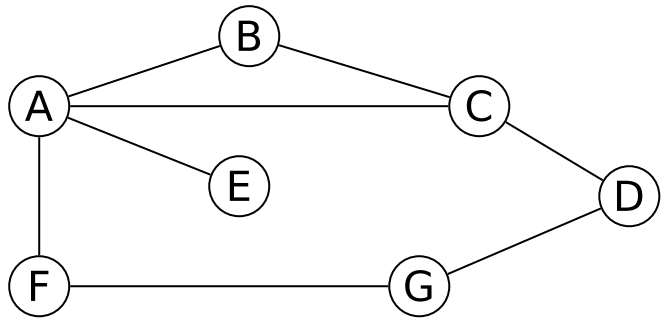
- Verwenden den *Bellman-Ford-Algorithmus*
- Ein Beispiel für ein Distanzvektorprotokoll ist das **Routing Information Protocol** (RIP)
- Arbeitsweise von RIP:
  - Jeder Router erkennt beim Start die direkt mit ihm verbundenen Netze
  - Die übrigen Router, die mit diesen Netzen direkt verbunden sind, sind die Nachbarn des neuen Routers
  - Alle 30 Sekunden sendet jeder Router seine Routing-Tabelle, die in diesem Kontext auch **Kostenvektor** heißt, über das verbindungslose Transportprotokoll UDP an seine direkten Nachbarn
    - Diese regelmäßige Nachricht heißt **Advertisement**
  - Empfängt ein Router einen Kostenvektor, überprüft er, ob Einträge darin besser sind, als die bislang von ihm gespeicherten
    - Enthält der empfangene Vektor günstigere Wege, aktualisiert der Router die entsprechenden Einträge in seiner lokalen Routing-Tabelle

Distanzvektorprotokolle (2/2)

- Arbeitsweise von RIP (Fortsetzung):
  - Zusätzlich zur periodischen Aktualisierungsnachricht sendet ein Router immer dann seinen Kostenvektor an die direkten Nachbarn, wenn er eine Veränderung in seiner Routing-Tabelle vorgenommen hat
  - Die Wegkosten zum Zielnetz hängen bei IP ausschließlich von der Anzahl der Router ab, die auf dem Weg passiert werden müssen
    - Die Anzahl der Router wird in **Hops** angegeben
  - Jeder Router erhöht den Wert der Hops um eins
  - Bei RIP kennt jeder Router nur den Inhalt seiner eigenen Routing-Tabelle
    - Die einzelnen Router haben keinen Überblick über das vollständige Netzwerk

- Weil kein Router einen Überblick über das komplette Netzwerk hat, implementiert das Protokoll einen **verteilten Algorithmus**
  - Nur so erreicht man eine gute Skalierbarkeit

Distanzvektorprotokoll – Beispiel (1/3)



- Am Anfang setzt jeder Router die Entfernung seiner direkten Nachbarn mit '1' an und die zu allen anderen mit unendlich (∞)

Im Router gespeicherte Informationen	Entfernung zu Router						
	A	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Distanzvektorprotokoll – Beispiel (2/3)

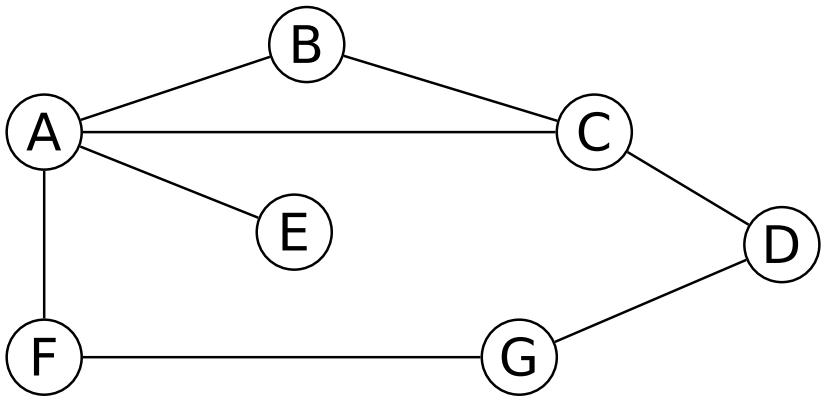
- Router A glaubt am Anfang, dass er B mit einem *Hop* und die Router D und G gar nicht erreichen kann
- Anfangs würde die Routing-Tabelle von Router A wie folgt aussehen

Ziel	Nächster Hop	Kosten
B	B	1
C	C	1
D	—	∞
E	E	1
F	F	1
G	—	∞

- Anschließend sendet jeder Router eine Nachricht mit seiner lokalen Routing-Tabelle (Kostenvektor) an die direkt mit ihm verbundenen Router
- Ein Beispiel:
  - Router A weiß, dass er Router F zu den Kosten 1 erreichen kann
  - Router F teilt Router A mit, dass er Router G mit Kosten von 1 erreichen kann
  - Dadurch weiß Router A, dass er Router G über F mit Kosten 2 erreichen kann und da $2 \leq \infty$ aktualisiert Router A seinen Kostenvektor

Distanzvektorprotokoll – Beispiel (3/3)

- Aktualisierte Routing-Tabelle in Router A



Ziel	Nächster Hop	Kosten
B	B	1
C	C	1
D	C	2
E	E	1
F	F	1
G	F	2

Im Router gespeicherte Informationen	Entfernung zu Router						
	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

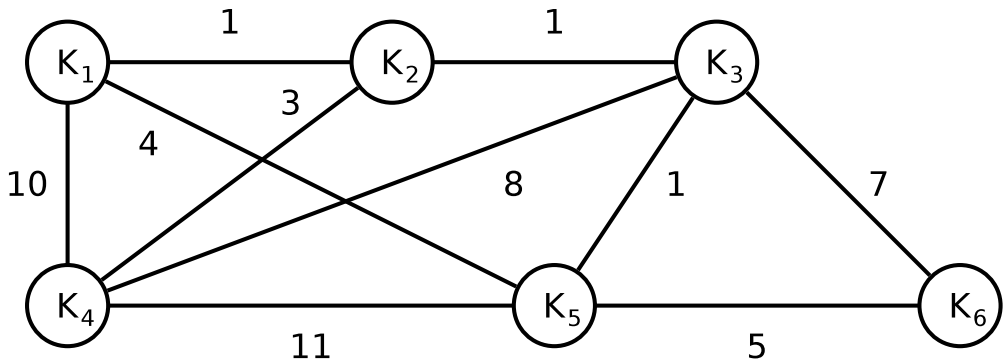
Das Beispiel war viel zu einfach

Distanzvektorprotokoll – größeres Beispiel (1/5)

Ziel	Hop	Distanz
K_1	K_1	0
K_2	?	∞
K_3	?	∞
K_4	?	∞
K_5	?	∞
K_6	?	∞

Ziel	Hop	Distanz
K_1	?	∞
K_2	K_2	0
K_3	?	∞
K_4	?	∞
K_5	?	∞
K_6	?	∞

Ziel	Hop	Distanz
K_1	?	∞
K_2	?	∞
K_3	K_3	0
K_4	?	∞
K_5	?	∞
K_6	?	∞



Ziel	Hop	Distanz
K_1	?	∞
K_2	?	∞
K_3	?	∞
K_4	K_4	0
K_5	?	∞
K_6	?	∞

Ziel	Hop	Distanz
K_1	?	∞
K_2	?	∞
K_3	?	∞
K_4	?	∞
K_5	K_5	0
K_6	?	∞

Ziel	Hop	Distanz
K_1	?	∞
K_2	?	∞
K_3	?	∞
K_4	?	∞
K_5	?	∞
K_6	K_6	0

- Initialisierung der Tabellen mit $\text{Hop}_{ij} \longleftarrow$? und $\text{Distanz}_{ij} \longleftarrow \infty$ für $i \neq j$ sowie $\text{Hop}_{ij} \longleftarrow K_i$ und $\text{Distanz}_{ij} \longleftarrow 0$ für $i = j$
- Für jeden direkten Nachbarn K_j von K_i wird eingetragen: $\text{Hop}_{ij} \longleftarrow K_j$ und $\text{Distanz}_{ij} \longleftarrow \text{Abstand}(K_i, K_j)$
- Jeder direkte Nachbar K_j von K_i sendet seine Routing-Tabelle an K_i
- Für einen Tabelleneintrag zu K_k wird überprüft, ob $\text{Distanz}_{ij} + \text{Distanz}_{ik} < \text{Distanz}_{ik}$
- Wenn das gilt, erfolgen diese Zuweisungen: $\text{Hop}_{ik} \longleftarrow K_j$ und $\text{Distanz}_{ik} \longleftarrow \text{Distanz}_{ij} + \text{Distanz}_{jk}$

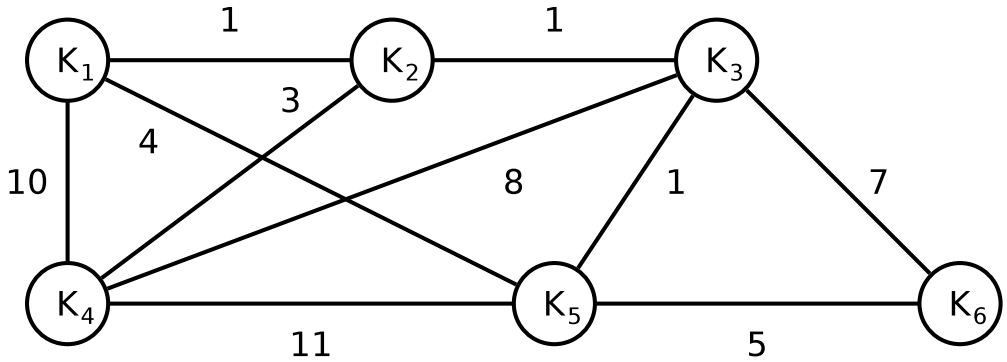
Quelle: Jörg Roth. **Prüfungstrainer Rechnernetze: Aufgaben und Lösungen**. Vieweg (2010)

Distanzvektorprotokoll – größeres Beispiel (2/5)

Ziel	Hop	Distanz
K_1	K_1	0
K_2	K_2	1
K_3	?	∞
K_4	K_4	10
K_5	K_5	4
K_6	?	∞

Ziel	Hop	Distanz
K_1	K_1	1
K_2	K_2	0
K_3	K_3	1
K_4	K_4	3
K_5	?	∞
K_6	?	∞

Ziel	Hop	Distanz
K_1	?	∞
K_2	K_2	1
K_3	K_3	0
K_4	K_4	8
K_5	K_5	1
K_6	K_6	7



Ziel	Hop	Distanz
K_1	K_1	10
K_2	K_2	3
K_3	K_3	8
K_4	K_4	0
K_5	K_5	11
K_6	?	∞

Ziel	Hop	Distanz
K_1	K_1	4
K_2	?	∞
K_3	K_3	1
K_4	K_4	11
K_5	K_5	0
K_6	K_6	5

Ziel	Hop	Distanz
K_1	?	∞
K_2	?	∞
K_3	K_3	7
K_4	?	∞
K_5	K_5	5
K_6	K_6	0

- Distanzen zu den direkten Nachbarn eingetragen

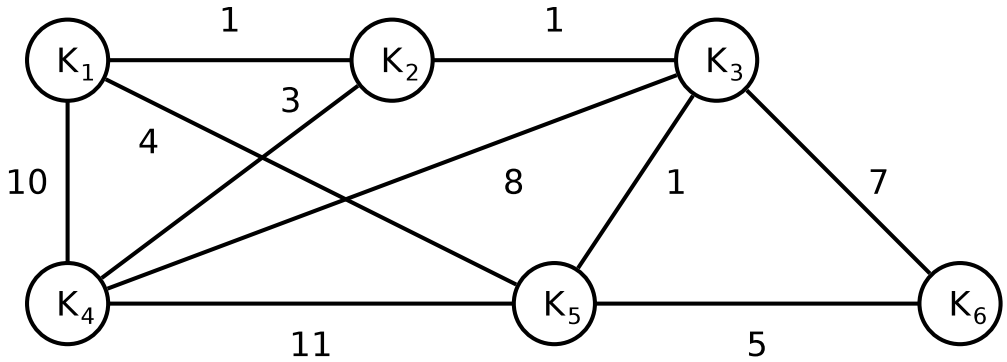
Quelle: Jörg Roth. **Prüfungstrainer Rechnernetze: Aufgaben und Lösungen**. Vieweg (2010)

Distanzvektorprotokoll – größeres Beispiel (3/5)

Ziel	Hop	Distanz
K_1	K_1	0
K_2	K_2	1
K_3	K_2	2
K_4	K_2	4
K_5	K_5	4
K_6	K_5	9

Ziel	Hop	Distanz
K_1	K_1	1
K_2	K_2	0
K_3	K_3	1
K_4	K_4	3
K_5	K_3	2
K_6	K_3	8

Ziel	Hop	Distanz
K_1	K_2	2
K_2	K_2	1
K_3	K_3	0
K_4	K_2	4
K_5	K_5	1
K_6	K_5	6



Ziel	Hop	Distanz
K_1	K_2	4
K_2	K_2	3
K_3	K_2	4
K_4	K_4	0
K_5	K_3	9
K_6	K_3	15

Ziel	Hop	Distanz
K_1	K_1	4
K_2	K_3	2
K_3	K_3	1
K_4	K_3	9
K_5	K_5	0
K_6	K_6	5

Ziel	Hop	Distanz
K_1	K_5	9
K_2	K_3	8
K_3	K_5	6
K_4	K_3	15
K_5	K_5	5
K_6	K_6	0

- Jeden Eintrag in den Routing-Tabelle mit den Tabellen der direkten Nachbarn inklusive der Wegekosten vergleichen und gegebenenfalls anpassen

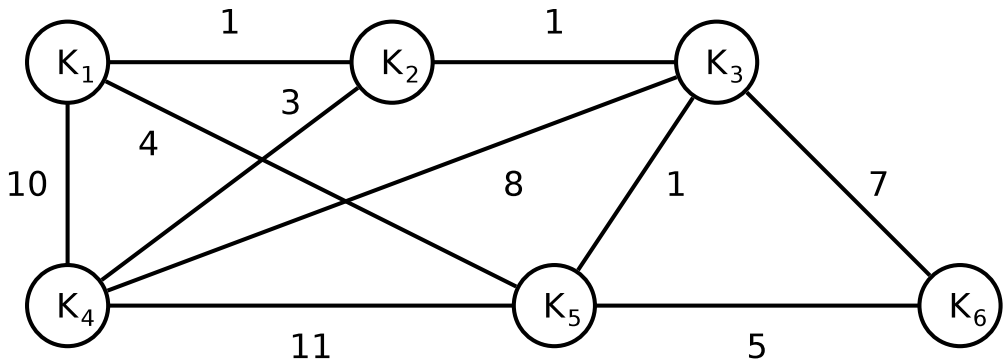
Quelle: Jörg Roth. **Prüfungstrainer Rechnernetze: Aufgaben und Lösungen**. Vieweg (2010)

Distanzvektorprotokoll – größeres Beispiel (4/5)

Ziel	Hop	Distanz
K_1	K_1	0
K_2	K_2	1
K_3	K_2	2
K_4	K_2	4
K_5	K_2	3
K_6	K_5	9

Ziel	Hop	Distanz
K_1	K_1	1
K_2	K_2	0
K_3	K_3	1
K_4	K_4	3
K_5	K_3	2
K_6	K_3	7

Ziel	Hop	Distanz
K_1	K_1	2
K_2	K_2	1
K_3	K_3	0
K_4	K_2	4
K_5	K_5	1
K_6	K_5	6



Ziel	Hop	Distanz
K_1	K_2	4
K_2	K_2	3
K_3	K_2	4
K_4	K_4	0
K_5	K_2	5
K_6	K_2	11

Ziel	Hop	Distanz
K_1	K_3	3
K_2	K_3	2
K_3	K_3	1
K_4	K_3	5
K_5	K_5	0
K_6	K_6	5

Ziel	Hop	Distanz
K_1	K_5	9
K_2	K_5	7
K_3	K_5	6
K_4	K_2	11
K_5	K_5	5
K_6	K_6	0

- Jeden Eintrag in den Routing-Tabelle mit den Tabellen der direkten Nachbarn inklusive der Wegekosten vergleichen und gegebenenfalls anpassen

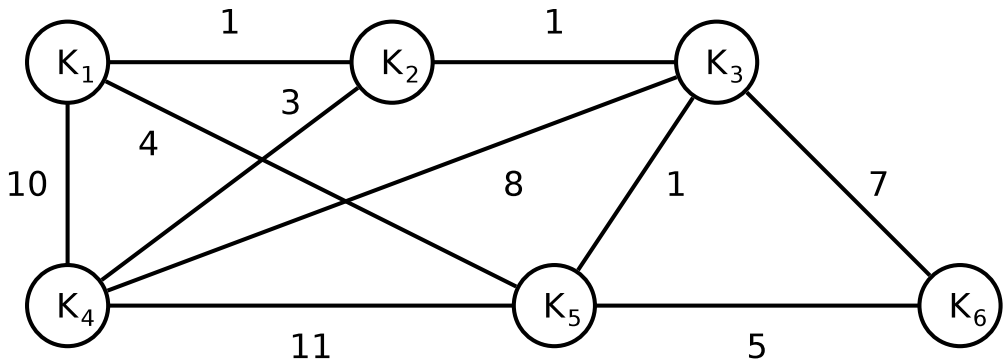
Quelle: Jörg Roth. **Prüfungstrainer Rechnernetze: Aufgaben und Lösungen**. Vieweg (2010)

Distanzvektorprotokoll – größeres Beispiel (5/5)

Ziel	Hop	Distanz
K_1	K_1	0
K_2	K_2	1
K_3	K_2	2
K_4	K_2	4
K_5	K_2	3
K_6	K_2	8

Ziel	Hop	Distanz
K_1	K_1	1
K_2	K_2	0
K_3	K_3	1
K_4	K_4	3
K_5	K_3	2
K_6	K_3	7

Ziel	Hop	Distanz
K_1	K_1	2
K_2	K_2	1
K_3	K_3	0
K_4	K_2	4
K_5	K_5	1
K_6	K_5	6



Ziel	Hop	Distanz
K_1	K_2	4
K_2	K_2	3
K_3	K_2	4
K_4	K_4	0
K_5	K_2	5
K_6	K_2	10

Ziel	Hop	Distanz
K_1	K_3	3
K_2	K_3	2
K_3	K_3	1
K_4	K_3	5
K_5	K_5	0
K_6	K_6	5

Ziel	Hop	Distanz
K_1	K_5	8
K_2	K_5	7
K_3	K_5	6
K_4	K_5	10
K_5	K_5	5
K_6	K_6	0

- Jeden Eintrag in den Routing-Tabelle mit den Tabellen der direkten Nachbarn inklusive der Wegekosten vergleichen und gegebenenfalls anpassen

Quelle: Jörg Roth. **Prüfungstrainer Rechnernetze: Aufgaben und Lösungen**. Vieweg (2010)

Begriffe des Routing Information Protocol (RIP)

- **Maximale Metrik**
  - Die **Metrik** (= **Kosten**) sind der Aufwand, um ein Netz zu erreichen
  - Beim Protokoll IP wird dazu ausschließlich der Hop Count verwendet
    - Dieser bezeichnet die Anzahl der Router, die entlang eines Pfades bis zum Zielnetz durchlaufen werden müssen
  - Die Unerreichbarkeit eines Ziels gibt RIP mit dem Hop-Count 16 an
    - RIP erlaubt also nur Netze mit einer maximalen Länge von 15 Routern
- **Counting to Infinity**
  - Damit Pakete nicht unendlich lange kreise, gibt es den Infinite-Wert
    - Bei RIP gilt der Hopcount-Wert 16 als Infinite-Wert
  - Dieser zeigt an, dass eine Route nicht erreichbar ist
  - Ist der Infinite-Wert noch nicht erreicht, kreisen IP-Pakete im Netz bis die Time to Live (TTL) abgelaufen ist
- **Route Invalidation Timer**
  - Nötig, damit alte Routing-Einträge gelöscht werden
  - Ansonsten würden falsche Routen dauerhaft bestehen bleiben

Routing Information Protocol (RIP)

Konvergenzzeit

- Zeitspanne, die für die Berechnung der besten Pfade für alle Router benötigt wird
- Die Dauer der Konvergenzzeit bei Distanzvektorprotokollen ist *lang*, weil sich Aktualisierungen nur langsam *fortpflanzen*

- Durch welche Maßnahmen kann man Routing-Schleifen bei RIP (und allg. bei Distanzvektorprotokollen) verhindern und die Konvergenzzeit verkürzen?
  - **Maximale Metrik**
  - **Split-Horizon**
  - **Poisoned Reverse Updates** (Route Poisoning)
  - **Triggered Updates**
  - **Holddown Timer**

Quelle: Vorlesungsfolien von Prof. Dr. Michael Massoth

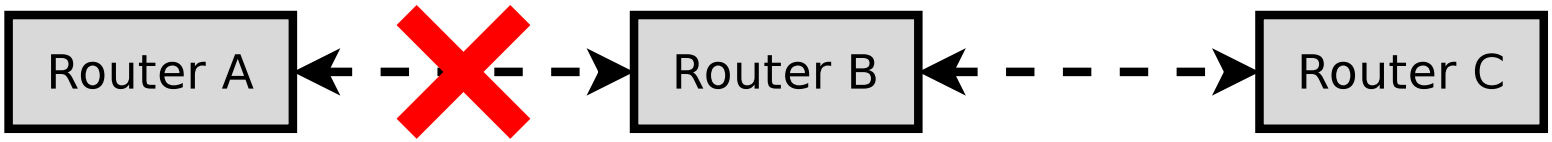
Begriffe des Routing Information Protocol (RIP)

- **Split Horizon**
  - Ein Router sendet die über eine seiner Schnittstellen empfangenen Routinginformationen zwar über alle anderen Schnittstellen weiter, aber nicht über die empfangende Schnittstelle zurück
    - Ein Router wird also daran gehindert eine Route zu einem bestimmten Ziel zurück an den Router zu übermitteln, von dem er diese Route gelernt hat
  - Kurzfassung (kann man sich gut merken):
    - „*Sende kein Routing-Update zu der Schnittstelle hinaus, von der du es bekommen hast*“
  - Grund: Verhindert Routing-Schleifen mit direkt benachbarten Routern

Beispiel zu Split Horizon

- Router C weiß von Router B, das Netzwerk 0 über Router A erreichbar ist

- Szenario: Router A und Netzwerk 0 sind nicht zu erreichen



- Auswirkung von Split Horizon:
  - Router B sendet beim nächsten Update an Router C, dass Router A nicht erreichbar ist
  - Router C passt seine Routing-Tabelle an, sendet aber die erhaltene Information nicht wieder an Router B zurück

Begriffe des Routing Information Protocol (RIP)

- **Poisoned Reverse Updates**
  - Poisoned Reverse = **blockierte Rückroute**
  - Alle über eine Schnittstelle gelernten und empfangenen Routen werden als „nicht erreichbar“ gekennzeichnet und zurückgesendet
    - Dafür wird die Anzahl der Hops direkt auf den Hopcount-Wert 16 (Infinite) gesetzt
  - Deutlicher ausgedrückt:
    - Ein Router propagiert eine gelernte Route über alle Schnittstellen weiter
    - Nur über diese Schnittstelle, über die er die Route gelernt hat, propagiert er diese Route mit dem mit Hopcount-Wert 16 (Infinite $\Longrightarrow$ „Netz ist nicht erreichbar“)
  - Kurzfassung (kann man sich gut merken):
    - *„Sende Routing-Update mit Hopcount-Wert 16 (Infinite) $\Longrightarrow$ „Netz ist nicht erreichbar“) zu der Schnittstelle hinaus, von der du es bekommen hast“*
  - Grund: Verhindert größere Routing-Schleifen

Begriffe des Routing Information Protocol (RIP)

- **Triggered Updates**
  - Normalerweise sendet jeder Router in einem festen Zeitintervall (typisch z.B. 30 Sekunden) alle ihm bekannten Routinginformationen an seine Nachbar-Router
    - **Periodische Aktualisierungsnachricht**
    - Wird auch dann verschickt, wenn sich nichts ändert
  - Bei eingeschalteter Option **Triggered Updates** sendet ein Router zusätzlich Informationen, wenn er selbst ein Update von seinen Nachbar-Routern bekommen hat
  - Ein Triggered Update wird sofort nach einer Netzwerktopologieänderung gesendet
    - Es ist unabhängig vom Update-Timer

Timer beim Routing Information Protocol (RIP)

- **Update Timer**: 30s
  - Periodische Aktualisierungsnachricht
- **Timeout**, **Expiration Timer** oder **Invalid Timer** (Cisco): 180s
  - Die Metrik (Hopcount-Wert) für eine Route wird auf 16 (Infinite) gesetzt, wenn innerhalb dieser Zeit kein Update für die Route ankommt
  - Die Route wird noch nicht aus der Routing-Tabelle gelöscht
- **Holddown Timer**: 180s (existiert nur bei Cisco)
  - Fällt ein Netz aus, wird es nicht sofort aus der Routing-Tabelle gelöscht
    - Während der Holddown-Zeit akzeptiert der Router keine Route mit besserer Metrik als die zuvor als nicht erreichbar markierte Route
    - So können sich andere Router darauf einstellen und das Netzwerk kommt schneller wieder in einen stabilen Zustand (Konvergenzzeit wird verkürzt)
- **Flush Timer** oder **Garbage Collection**: 60s (Cisco) oder 120s
  - Nach dieser Zeit wird die Route aus der Routing-Tabelle gelöscht

Link-State-Routing-Protokolle

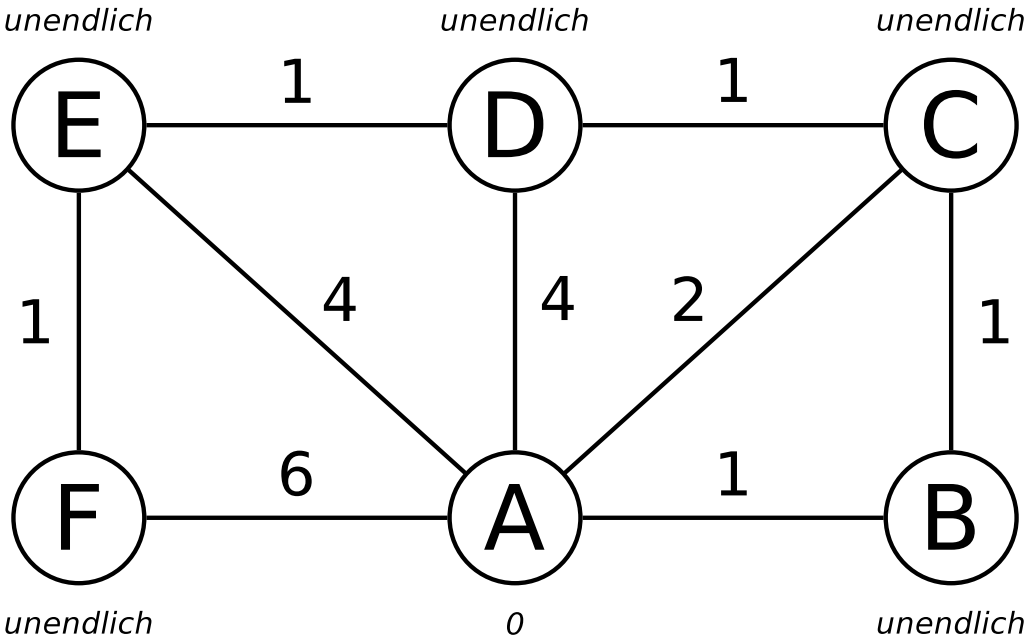
- **Link-State-Routing-Protokolle** verwenden den **Dijkstra-Algorithmus**
  - Link-State-Routing-Protokolle ermöglichen die Berechnung des kürzesten Weges zwischen einem Startknoten und allen anderen Knoten in einem kantengewichteten Graphen
- Ein Beispiel für ein Link-State-Routing-Protokoll ist **Open Shortest Path First** (OSPF)
  - Bei OSPF kann jeder Router den Zustand der Verbindung zu seinen Nachbarn und die Kosten dahin ermitteln
  - Die Information, die ein Router hat, gibt er an alle anderen Router weiter
  - Jeder Router erstellt sich eine **vollständige Übersicht** mit Topologie-Informationen über das Netzwerk
  - Es finden regelmäßige Link-State-Aktualisierungen durch Fluten (*Flooding*) statt
    - Dadurch reagiert das Protokoll rascher auf Topologieänderungen und Knotenausfälle
    - Nachteil: Overhead durch das Fluten, weil alle Router Informationen über die Topologie des vollständigen Netzwerks lokal speichern

Dijkstra-Algorithmus

- Berechnung des kürzesten Weges zwischen einem Startknoten und allen anderen Knoten in einem kantengewichteten Graphen
  - Kantengewichte dürfen nicht negativ sein
  - Ist man nur am Weg zu einem bestimmten Knoten interessiert, kann man in Schritt 2 abbrechen, wenn der gesuchte Knoten der aktive ist

1. Weise allen Knoten die Eigenschaften **Distanz** und **Vorgänger** zu
   - Initialisiere die Distanz im Startknoten mit 0 und in allen anderen Knoten mit ∞
2. Solange es noch nicht besuchte Knoten gibt, wähle darunter denjenigen mit minimaler Distanz aus
   - Speichere, dass dieser Knoten schon besucht wurde
   - Berechne für alle noch nicht besuchten Nachbarknoten die Summe des jeweiligen Kantengewichtes und der Distanz im aktuellen Knoten
   - Ist dieser Wert für einen Knoten kleiner als die dort gespeicherte Distanz, aktualisiere sie und setze den aktuellen Knoten als Vorgänger

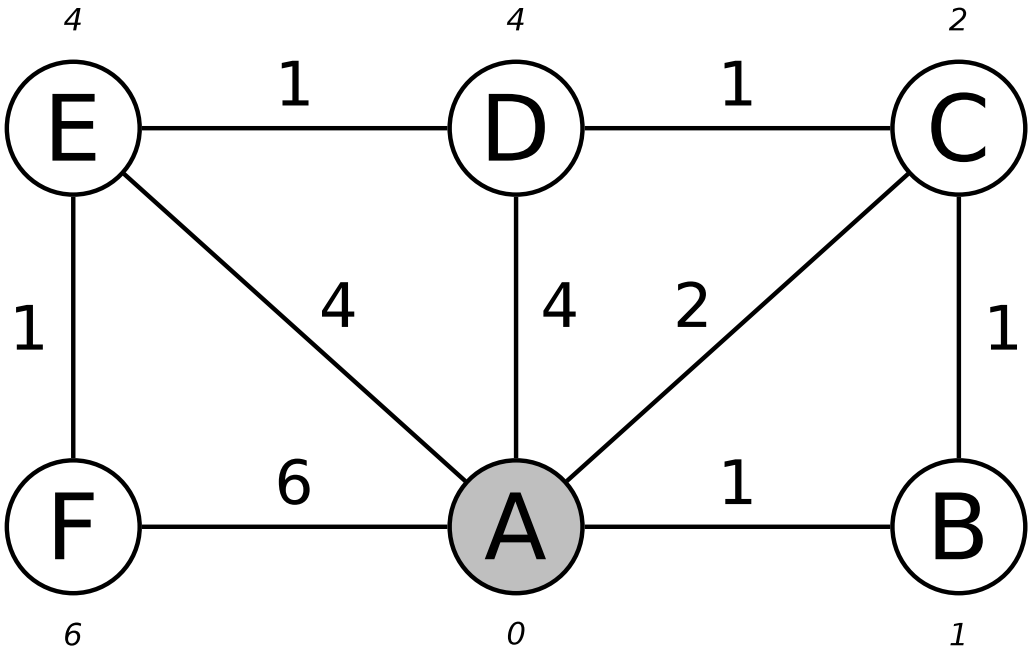
Dijkstra-Algorithmus – Beispiel (1/7)



Distanzwerte	
$d_A = 0$	
$d_B = \infty$	
$d_C = \infty$	
$d_D = \infty$	
$d_E = \infty$	
$d_F = \infty$	

- Schritt 1: Initialisiere mit 0 und ∞
  - Sei A der Startknoten
  - A hat die minimale Distanz
- Besuchte Knoten $= \{\}$
- Quellbaum $= \{\}$

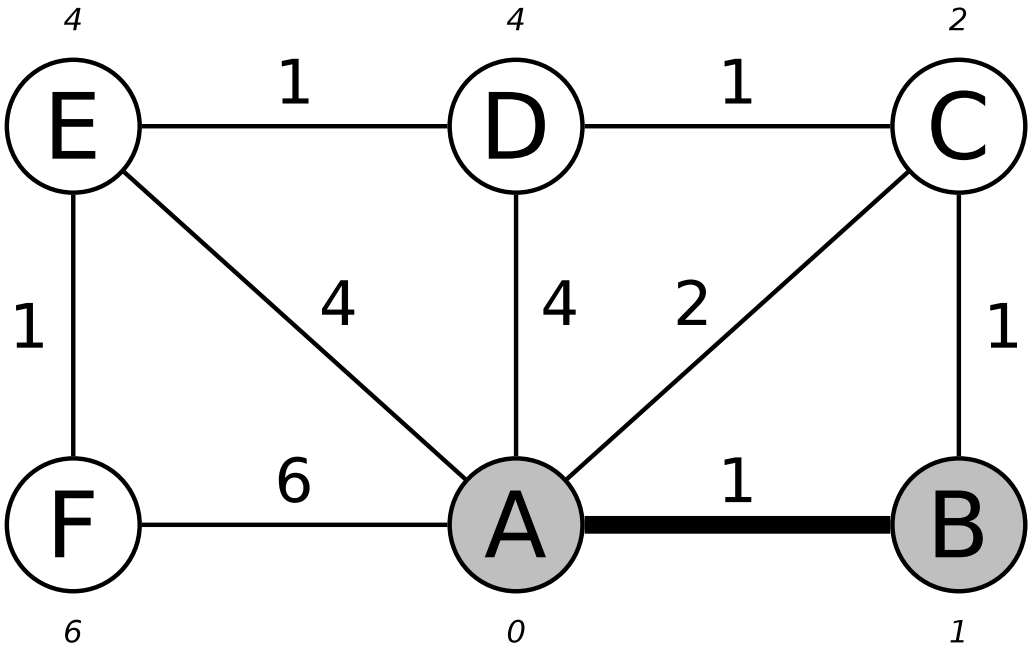
Dijkstra-Algorithmus – Beispiel (2/7)



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	⟵ minimale Distanz
$d_C = 2$	
$d_D = 4$	
$d_E = 4$	
$d_F = 6$	

- Schritt 2: Summe der Kantengewichte berechnen
  - B hat die minimale Distanz
- Besuchte Knoten $= \{A\}$
- Quellbaum $= \{A\}$

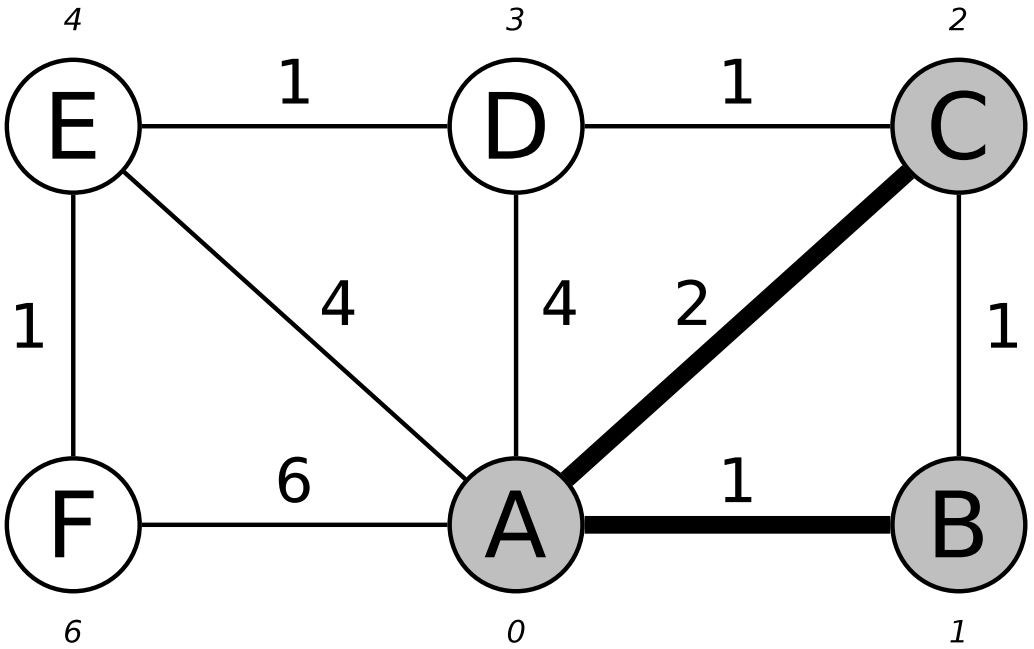
Dijkstra-Algorithmus – Beispiel (3/7)



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	$\longleftarrow$ minimale Distanz
$d_D = 4$	
$d_E = 4$	
$d_F = 6$	

- Schritt 3: Knoten B besuchen
  - Keine Veränderung zu C
  - C hat die minimale Distanz
- Besuchte Knoten = {A, B}
- Quellbaum = {A, A$\longrightarrow$B}

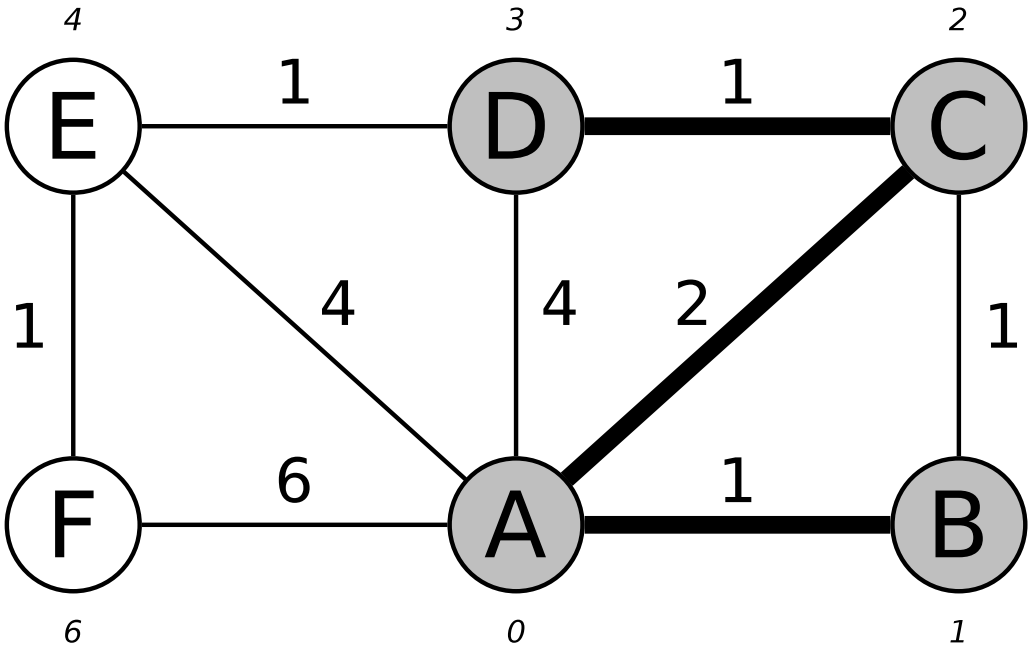
Dijkstra-Algorithmus – Beispiel (4/7)



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	besucht
$d_D = \mathbf{3}$	$\longleftarrow$ minimale Distanz
$d_E = 4$	
$d_F = 6$	

- Schritt 4: Knoten C besuchen
  - Keine Veränderung zu B
  - Veränderung zu D (Weg über C ist kürzer als der direkte Weg)
  - D hat die minimale Distanz
- Besuchte Knoten = {A, B, C}
- Quellbaum = {A, A$\longrightarrow$B, A$\longrightarrow$C}

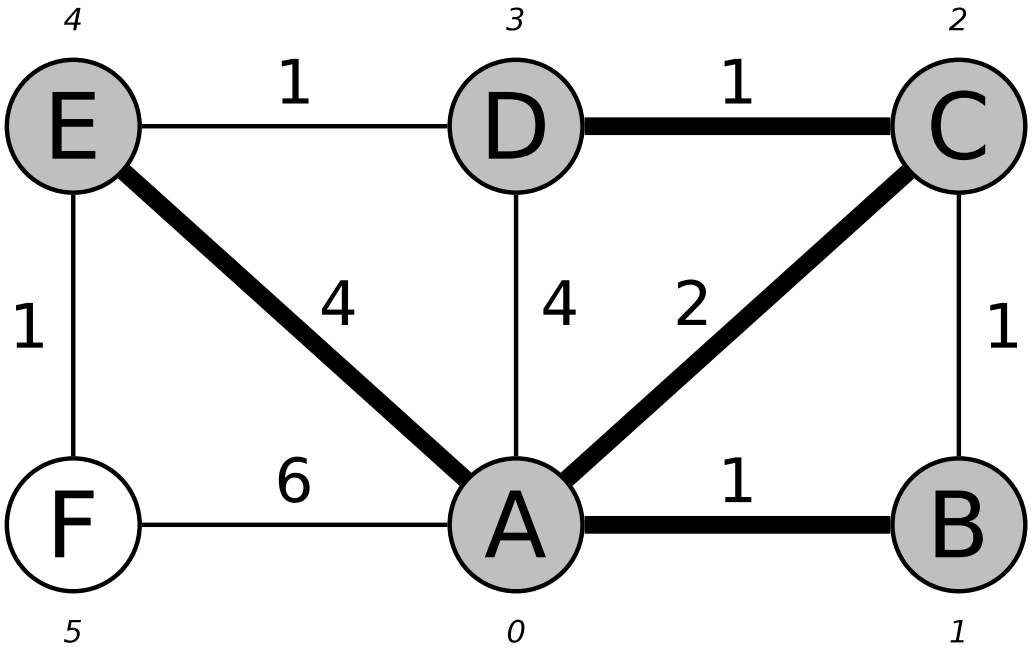
Dijkstra-Algorithmus – Beispiel (5/7)



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	besucht
$d_D = 3$	besucht
$d_E = 4$	⟵ minimale Distanz
$d_F = 6$	

- Schritt 5: Knoten D besuchen
  - Keine Veränderung zu C
  - Keine Veränderung zu E
  - E hat die minimale Distanz
- Besuchte Knoten = {A, B, C, D}
- Quellbaum = {A, A⟶B, A⟶C, C⟶D}

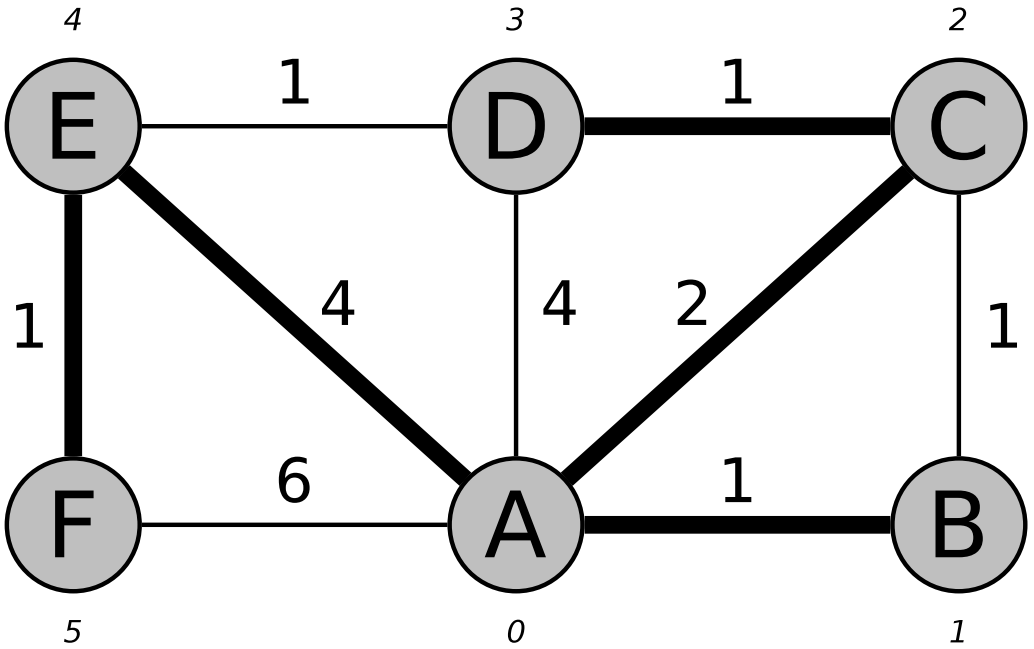
Dijkstra-Algorithmus – Beispiel (6/7)



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	besucht
$d_D = 3$	besucht
$d_E = 4$	besucht
$d_F = \mathbf{5}$	$\longleftarrow$ minimale Distanz

- Schritt 6: Knoten E besuchen
  - Keine Veränderung zu D
  - Veränderung zu F (Weg über E ist kürzer als der direkte Weg)
  - F hat die minimale Distanz
- Besuchte Knoten = {A, B, C, D, E}
- Quellbaum = {A, A$\longrightarrow$B, A$\longrightarrow$C, C$\longrightarrow$D, A$\longrightarrow$E}

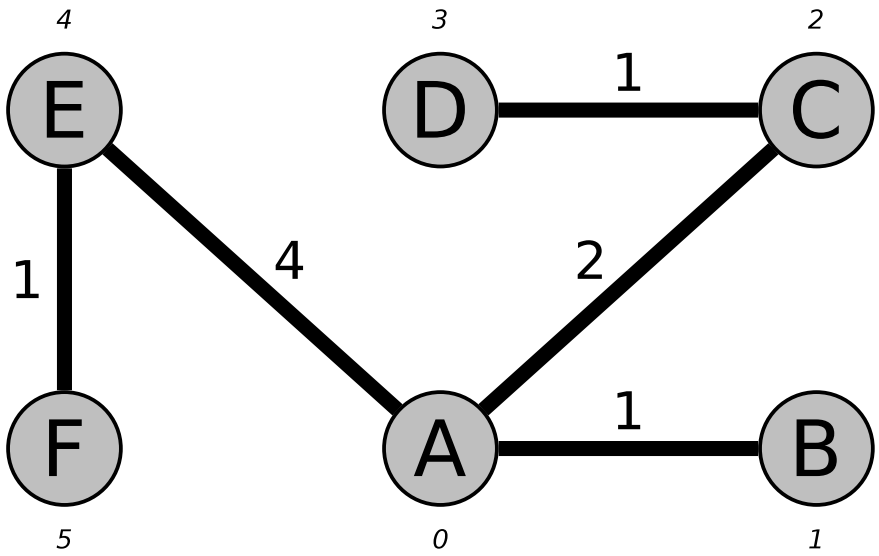
Dijkstra-Algorithmus – Beispiel (7/7)



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	besucht
$d_D = 3$	besucht
$d_E = 4$	besucht
$d_F = 5$	besucht

- Schritt 7: Knoten F besuchen
  - Keine Veränderung zu E
- Besuchte Knoten = {A, B, C, D, E, F}
- Quellbaum = {A, A$\longrightarrow$B, A$\longrightarrow$C, C$\longrightarrow$D, A$\longrightarrow$E, E$\longrightarrow$F}

Dijkstra-Algorithmus – Beispiel (Ergebnis)



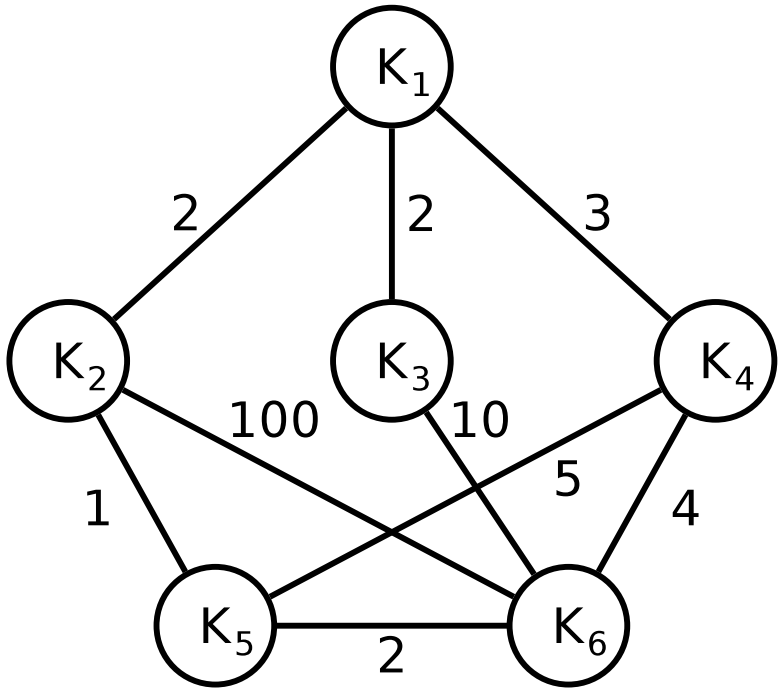
- Ergebnis

Distanzvektorprotokolle vs. Link-State-Routing-Protokolle

- Distanzvektorprotokolle (Bellman-Ford)
  - Jeder Knoten kommuniziert nur mit seinen **direkten** Nachbarn
  - **Keine Kenntnis über die komplette Netzwerk-Topologie**
- Link-State-Routing-Protokolle (Dijkstra)
  - **Alle** Knoten kommunizieren untereinander $\Longrightarrow$ Netzwerk wird geflutet
  - Baut eine **komplexe Datenbank mit Topologie-Informationen** auf

Übung

- Gegeben sei folgendes Netzwerk



1. Bestimmen Sie mit Hilfe das Link-State-Routing-Protokolls (Dijkstra-Algorithmus) den Quellbaum von Knoten K_5

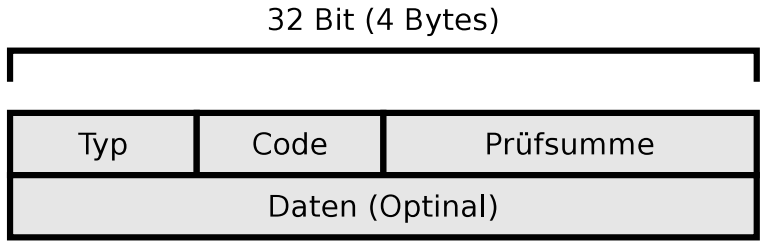
Quelle: Jörg Roth. **Prüfungstrainer Rechnernetze: Aufgaben und Lösungen**. Vieweg (2010)

Diagnose und Fehlermeldungen mit ICMP

- Der Austausch von Informations- und Fehlermeldungen ist über das **Internet Control Message Protocol** (ICMP) möglich
- ICMP ist ein Bestandteil (*Partnerprotokoll*) von IPv4, wird aber wie ein eigenständiges Protokoll behandelt
  - Für IPv6 existiert mit ICMPv6 ein ähnliches Protokoll
- Alle Router und Endgeräte können mit ICMP umgehen
- Typische Situationen, wo ICMP zum Einsatz kommt:
  - Ein Router verwirft ein IP-Paket, weil er nicht weiß, wie er es weiterleiten kann
  - Nur ein Fragment eines IP-Pakets kommt am Ziel an
  - Das Ziel eines IP-Pakets ist unerreichbar, weil die Time To Live (TTL) abgelaufen ist

ICMP

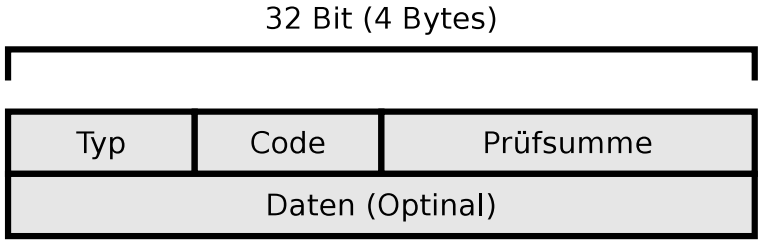
- Mit Ausnahme der Echo-Funktion kann ein ICMP-Paket niemals ein anderes ICMP-Paket auslösen
  - Kann ein ICMP-Paket nicht zugestellt werden, wird nichts weiter unternommen
- Eine Anwendung, die ICMP-Pakete versendet, ist das Programm Ping
- ICMP definiert verschiedene Informationsnachrichten, die ein Router zurücksenden kann



- ICMP-Nachrichten werden im Nutzdatenteil von IPv4-Paketen übertragen
  - Im Header des IPv4-Pakets steht dann im Datenfeld **Servicetyp** der Wert 0 und im Datenfeld **Protokollnummer** der Wert 1
  - Bei ICMPv6 ist die Protokollnummer 58

ICMP-Nachrichten

- Das Datenfeld **Typ** im ICMP-Header gibt den Nachrichtentyp an
  - Es gibt also die Klasse an, zu der die ICMP-Nachricht gehört

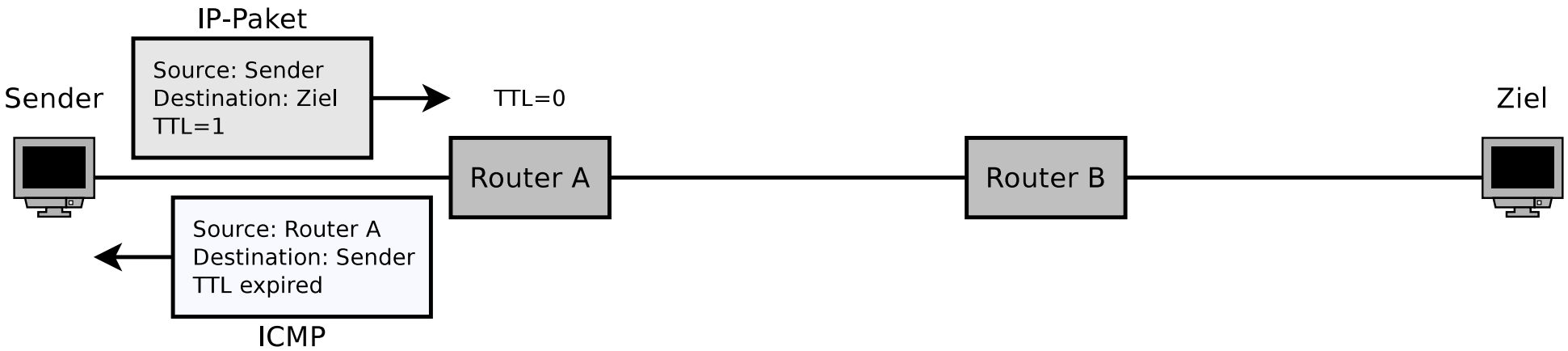


- Das Datenfeld **Code** spezifiziert die Art der Nachricht innerhalb eines Nachrichtentyps
- Die Tabelle enthält einige Nachrichtentyp-Code-Kombinationen

Typ	Typname	Code	Bedeutung
0	Echo-Antwort	0	Echo-Antwort (Antwort auf Ping)
3	Ziel nicht erreichbar	0	Netz unerreichbar
		1	Ziel unerreichbar
		2	Protokoll nicht verfügbar
		3	Port nicht verfügbar
		4	Fragmentierung nötig, aber im IP-Paket untersagt
		13	Firewall des Ziels blockt IP-Paket
4	Sender verlangsamen	0	Empfangspuffer voll, IP-Paket verworfen
8	Echo-Antwort	0	Echo-Anfrage (Ping)
11	Zeitlimit überschritten	0	TTL (Time To Live) abgelaufen
17	Address Mask Request	0	Anfrage nach der Anzahl der Bits in der Subnetzmaske
18	Address Mask Reply	0	Antwort auf Nachrichtentyp 18
		1	Zeitlimit während der Defragmentierung überschritten
30	Traceroute	0	Weg zum Ziel ermitteln

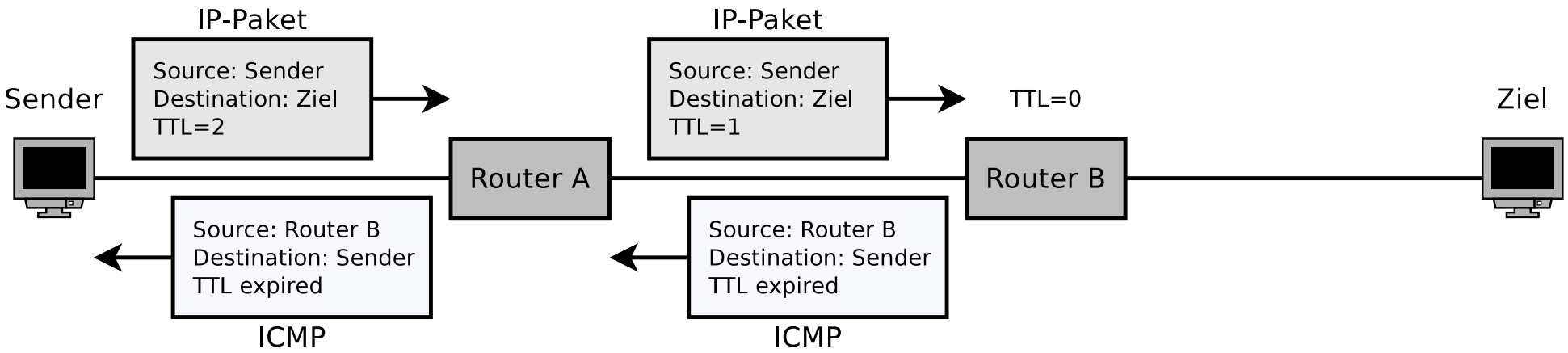
Anwendungsbeispiel für ICMP: `traceroute` (1/3)

- Ein Anwendungsbeispiel für ICMP ist das Werkzeug `traceroute`, das ermittelt, über welche Router Datenpakete bis zum Ziel vermittelt werden
  - Der Sender schickt ein IP-Paket an den Empfänger mit `TTL=1`
  - Router A empfängt das IP-Paket, setzt `TTL=0`, verwirft das IP-Paket und sendet eine ICMP-Nachricht vom Nachrichtentyp `11` und Code `0` an den Sender



Anwendungsbeispiel für ICMP: `traceroute` (2/3)

- `traceroute` (Fortsetzung):
  - Daraufhin schickt der Sender ein IP-Paket an den Empfänger mit `TTL=2`
  - Das IP-Paket wird von Router A weitergeleitet und dabei wird auch der Wert von `TTL` dekrementiert
  - Router B empfängt das IP-Paket, setzt `TTL=0`, verwirft das IP-Paket und sendet eine ICMP-Nachricht vom Nachrichtentyp `11` und Code `0` an den Sender



Anwendungsbeispiel für ICMP: `traceroute` (3/3)

- `traceroute` (Fortsetzung):
  - Sobald der Wert von `TTL` groß genug ist, dass der Empfänger erreicht wird, sendet dieser eine ICMP-Nachricht vom Nachrichtentyp 3 und Code 3 an den Sender
  - Auf dieses Art und Weise kann via ICMP der Weg vom Sender zum Empfänger nachvollzogen werden