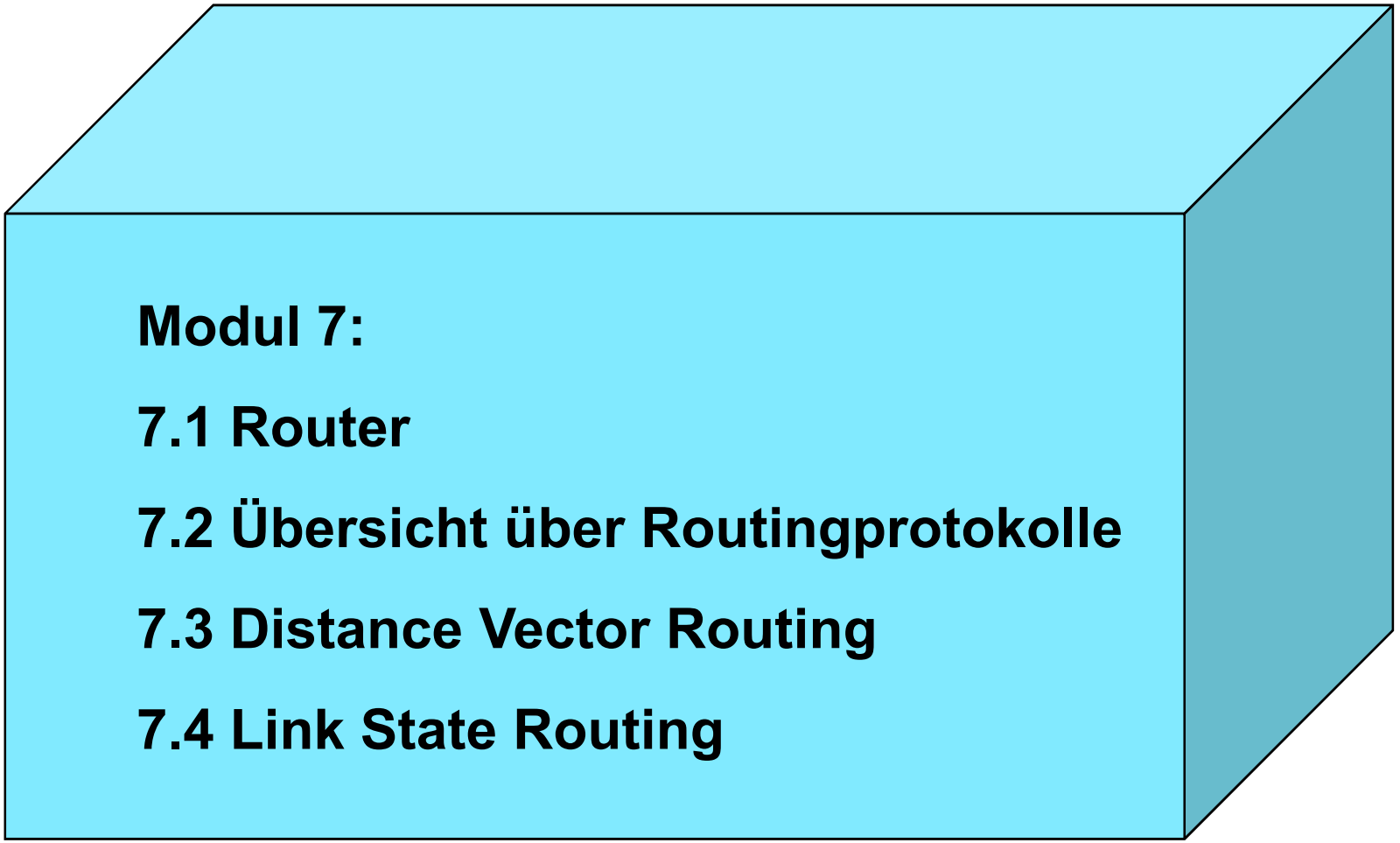
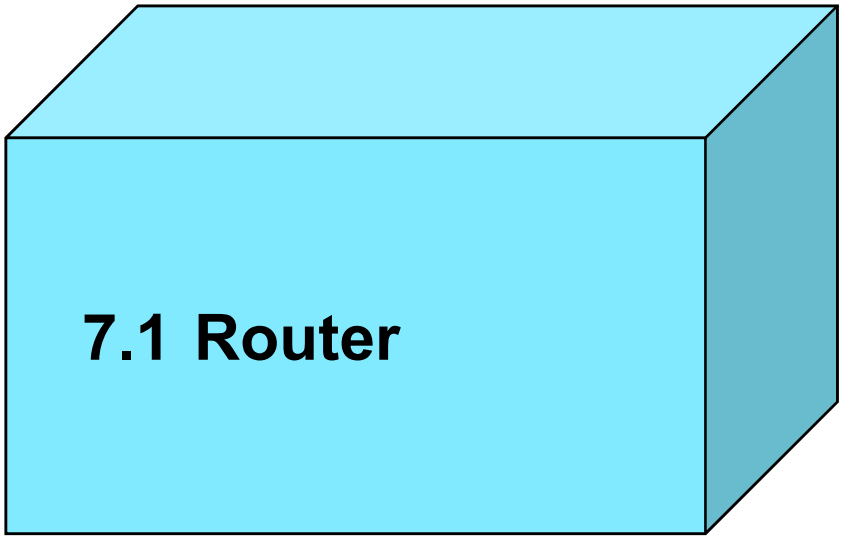
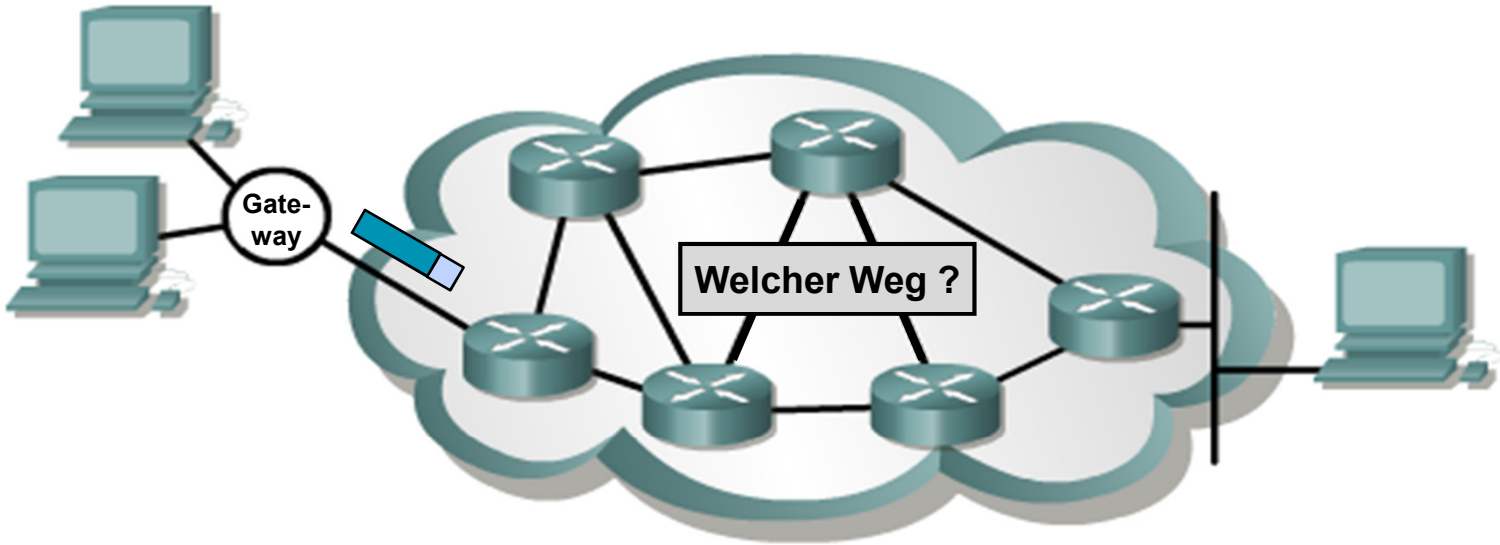

Modul 7:
7.1 Router
7.2 Übersicht über Routingprotokolle
7.3 Distance Vector Routing
7.4 Link State Routing


7.1 Router

Der Router als klassische Schicht 3 – Komponente (1)



Quelle: Schulungsunterlagen Cisco Academy

- Ein Router arbeitet auf Schicht 3 und wertet die IP-Protokollinformationen aus
- Seine wichtigste Aufgabe ist die Wegewahl innerhalb des Netzes
- Wie ist ein Router aufgebaut? Was ist bei der Konfiguration einer Routers zu tun?

Der Router als klassische Schicht 3 – Komponente (2)

- Die Entscheidung, an welches Interface ein Paket geleitet wird, trifft der Router auf der Basis von Routing-Tabellen
- Eine IP-Routing-Tabelle enthält IP-Adressen, denen bestimmte Interfaces zugeordnet sind
- Pakete mit Adressen, die nicht in der Tabelle vorhanden sind, werden in der Regel an ein „Standard Gateway“ geleitet

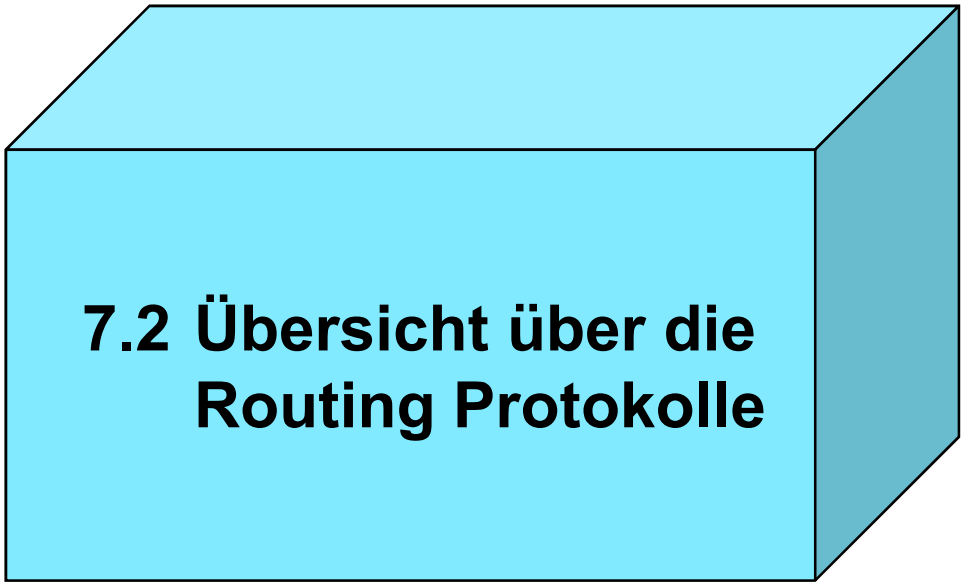
  Meistens bildet das Standard Gateway den Zugang zum Internet
- Router begrenzen Broadcastdomänen d.h. alle Broadcasts innerhalb eines Netzes gehen bis zum nächsten Router

Interne Komponenten eines Routers

- Ein Router ähnelt einem Computer, aber ohne Festplatte, Monitor und Tastatur
- Er enthält einen Prozessor, einen CPU Bus, einen System Bus, eine Leistungsversorgung ...
- Er verfügt über verschiedene Schnittstellen und verschiedene, interne Speicherbausteine
- Der System Bus verbindet die verschiedenen Schnittstellen mit der CPU, der CPU Bus verbindet die verschiedenen Speicherbausteine mit der CPU
- Wichtige Speicherbausteine im Router
  - (Boot) ROM
  - „Flash Memory“ (enthält i.d.R. das Betriebssystem des Routers)
  - NVRAM (Non Volatile RAM, behält den Inhalt beim Ausschalten und enthält z.B. das „Start-up“ Konfigurationsfile)
  - RAM (enthält die „Running-config“ d.h. das aktuelle Konfigurationsfile, die Routingtabelle und den schnellen Switching-Zwischenspeicher und verliert beim Ausschalten den Dateninhalt)

Wesentliche Schritte bei der Konfiguration eines Routers

- **Konfiguration der Interfaces:**
  - IP-Adresse des Interfaces
  - IP-Netzadresse mit Netzmaske des angeschlossenen Netzes
  - (Routing-) Protokolle, die über das Interface ausgetauscht werden sollen
  - Schicht 2 Protokoll des Interfaces (ev. mit Parametern; Ethernet, Frame Relay, ISDN ...)
- **Zusätzlich möglich:**
  - Zugangsbeschränkungen für das Interface (z.B.: wichtige Anwendungen haben einen festen Port, wenn auf einem Interface dieser Port „gesperrt“, ist damit auch die Anwendung gesperrt; Sperren bestimmter IP Adressbereiche ...)
- **Konfiguration der verwendeten Routingprotokolle (welches Routingprotokoll, welche Metrikwerte ...)**
- **Konfiguration zusätzlicher Protokolle wie DHCP oder NAT/ NAPT (nebst Parametern ...)**


7.2 Übersicht über die
Routing Protokolle

Autonomous System

- Im Internet werden oft Netze zu größeren Einheiten zusammengeschlossen
- Ein Autonomous System ist eine administrative Einheit unter der Kontrolle einer einzigen Instanz bzw. Firma
- Internes Routing bedeutet Routing innerhalb eines Autonomous Systems
  - Internes Routing kann statisch mit Tabellen oder dynamisch auf der Basis von Routingprotokollen realisiert werden
- Externes Routing bedeutet Routing zwischen verschiedenen Autonomous Systems
  - Externes Routing erfolgt immer auf der Basis von Routing Protokollen
  - in den Grenzgebieten werden über Protokolle Routing-Daten (Erreichbarkeitsinformationen) ausgetauscht
  - wichtige Protokolle: Exterior Gateway Protocol (EGP), Border Gateway Protocol (BGP)

Statisches Routing

- Die einfachste Möglichkeit für den Aufbau einer Routing-Tabelle:

⇒ **statisches Routing:**

Die Einträge der Routing-Tabelle werden manuell eingeben, d.h. das Routing basiert auf fest vorgegebenen Informationen

- Einsatzgebiet: in kleineren Netzen mit wenigen Routern
- Änderungen müssen von Hand eingegeben werden d.h. die Tabellen müssen „gepflegt“ werden
- Bestimmte Adressen werden immer an bestimmte, festgelegte Interfaces geleitet
- Unbekannte Adressen werden an ein „Standard-Gateway“ geleitet, das in der Regel den Zugang zum Internet bildet

Dynamisches Routing

- **Zweite Möglichkeit für den Aufbau einer Routing-Tabelle:**

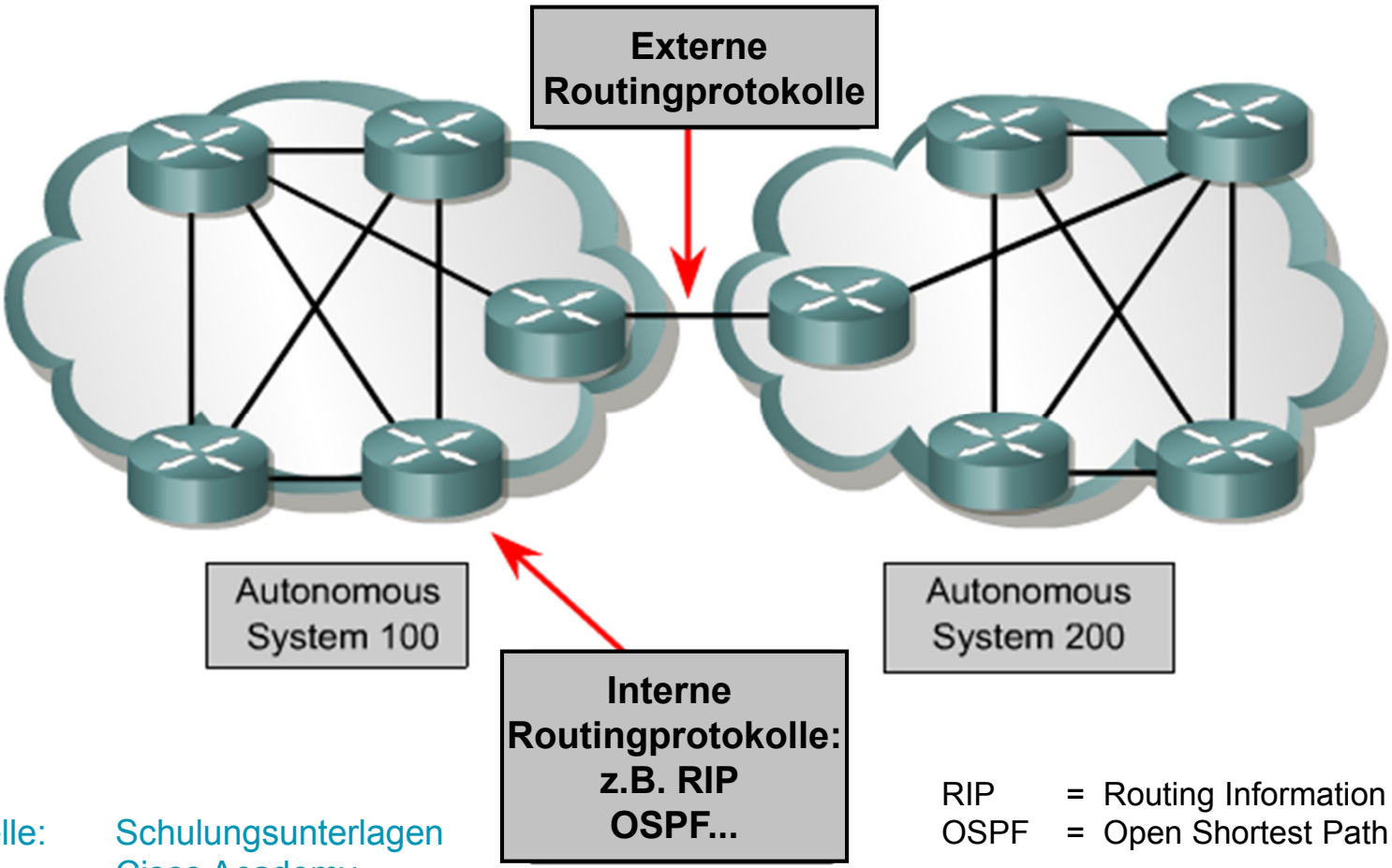
⇒ **dynamisches Routing d.h. Routing auf der Basis von Routing-Protokollen**

- **Einsatzgebiet: in größeren Netzen d.h. in Netzen mit mehreren/ vielen Routern**
- **Topologieänderungen werden automatisch erfasst, aber es gibt eine System-/ Netzwerkbelastung durch die Routing-Protokolle**
- **wichtige Protokolle:** Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Interior Gateway Routing Protocol (IGRP)

Wann und wo werden die Routing-Protokollinformationen übertragen?

Die Routinginformationen werden in regelmäßigen zeitlichen Abständen über die gleichen Interfaces und Netzverbindungen übertragen wie die Nutzdaten, quasi im Hintergrund

Routing innerhalb und zwischen von Autonomous Systems



RIP = Routing Information Protocol
OSPF = Open Shortest Path First

Quelle: Schulungsunterlagen Cisco Academy

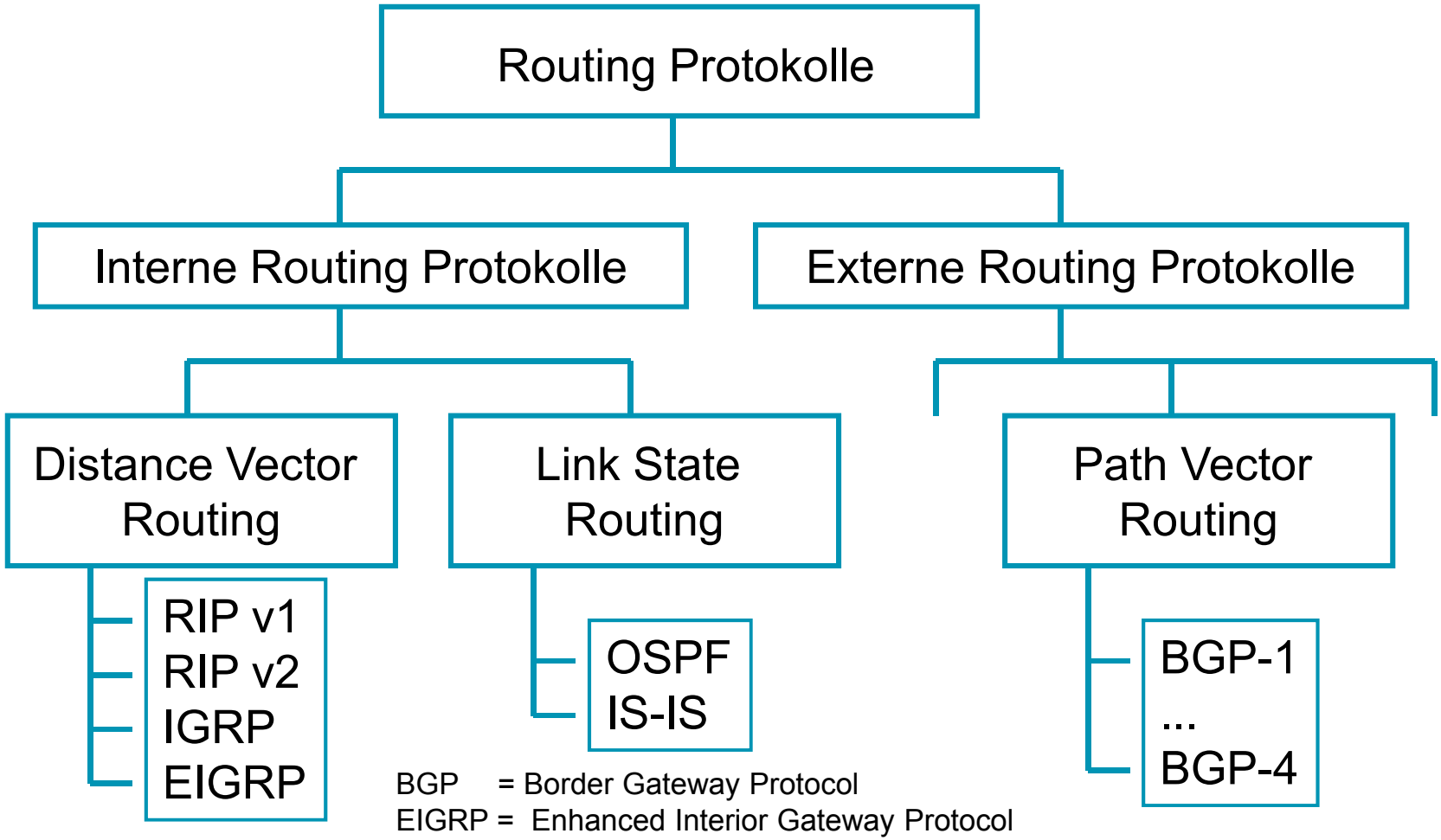
Wie sieht eine Routingtabelle aus?

Wie können die Routingtabellen aufgebaut werden?

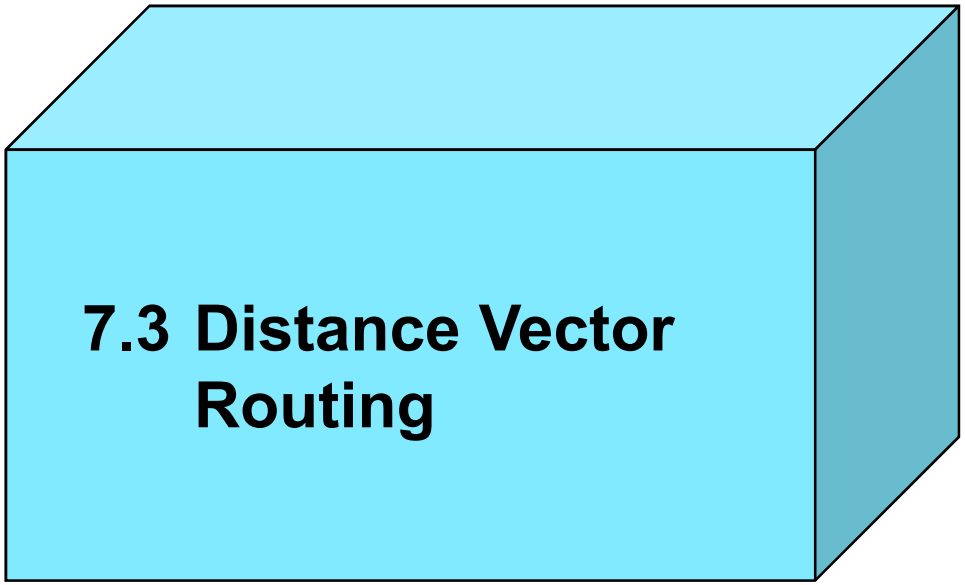
Metrik

- Allgemeine Bewertungskriterien verschiedener Pfade bei Routing-Protokollen werden „Metrik“ genannt
- Diese Werte werden benutzt, um unterschiedliche Wege zu bewerten und miteinander vergleichen zu können
- Mögliche Parameter:
  - Hop Count
  - Bandbreite
  - Verzögerung
  - Belastung
  - Zuverlässigkeit
  - Kosten

Übersicht über Routingprotokolle

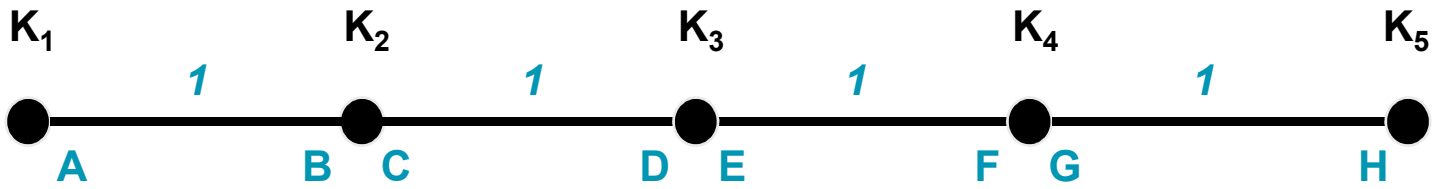


BGP = Border Gateway Protocol
EIGRP = Enhanced Interior Gateway Protocol
IS-IS = Intermediate System to Intermediate System Protocol


7.3 Distance Vector
Routing

Distance Vector Routing: Router tauschen ihre „Routing Tabellen“ aus

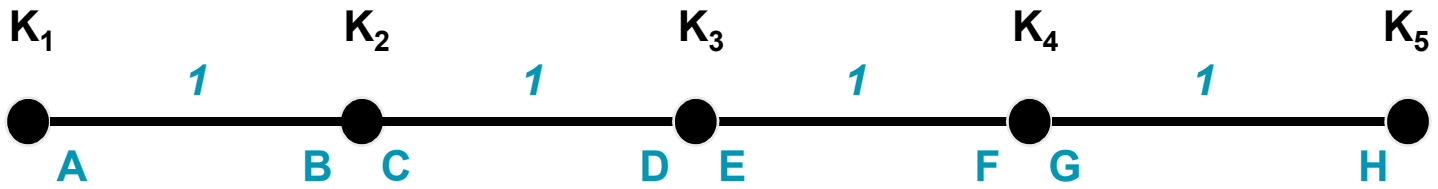
Aufbau einer Routing Tabelle (1)



Routing Tabellen aller Router zu Beginn ($t = t_0$)

K_1	K_2	K_3	K_4	K_5
(K_1, 0) local	(K_2, 0) local	(K_3, 0) local	(K_4, 0) local	(K_5, 0) local

Distance Vector Routing: Aufbau der Routing Tabelle (2)



Routing Tabellen nach dem 1. Update ($t = t_1$)

K_1
(K_1, 0) local
(K_2, 1) via A

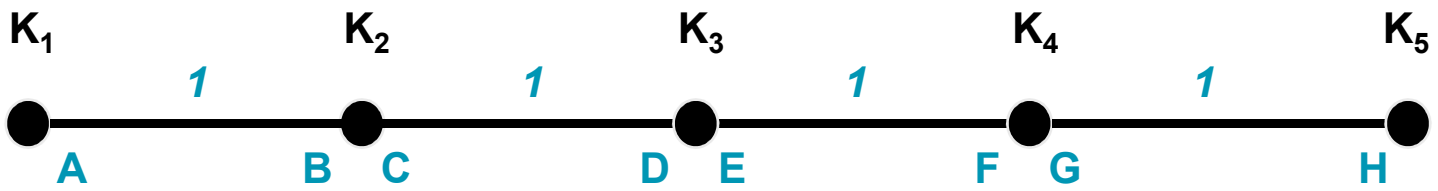
K_2
(K_1, 1) via B
(K_2, 0) local
(K_3, 1) via C

K_3
(K_2, 1) via D
(K_3, 0) local
(K_4, 1) via E

K_4
(K_3, 1) via F
(K_4, 0) local
(K_5, 1) via G

K_5
(K_4, 1) via H
(K_5, 0) local

Distance Vector Routing: Aufbau der Routing Tabelle (3)



Routing Tabellen nach dem 2. Update ($t = t_2$)

K_1

(K_1, 0) local
(K_2, 1) via A
(K_3, 2) via A

K_2

(K_1, 1) via B
(K_2, 0) local
(K_3, 1) via C
(K_4, 2) via C

K_3

(K_1, 2) via D
(K_2, 1) via D
(K_3, 0) local
(K_4, 1) via E
(K_5, 2) via E

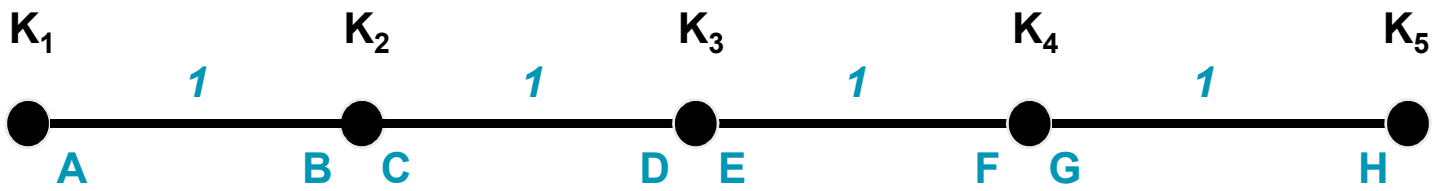
K_4

(K_2, 2) via F
(K_3, 1) via F
(K_4, 0) local
(K_5, 1) via G

K_5

(K_3, 2) via H
(K_4, 1) via H
(K_5, 0) local

Distance Vector Routing: Aufbau der Routing Tabelle (4)



Routing Tabellen nach dem 3. Update ($t = t_3$)

K_1

(K_1, 0) local
(K_2, 1) via A
(K_3, 2) via A
(K_4, 3) via A

K_2

(K_1, 1) via B
(K_2, 0) local
(K_3, 1) via C
(K_4, 2) via C
(K_5, 3) via C

K_3

(K_1, 2) via D
(K_2, 1) via D
(K_3, 0) local
(K_4, 1) via E
(K_5, 2) via E

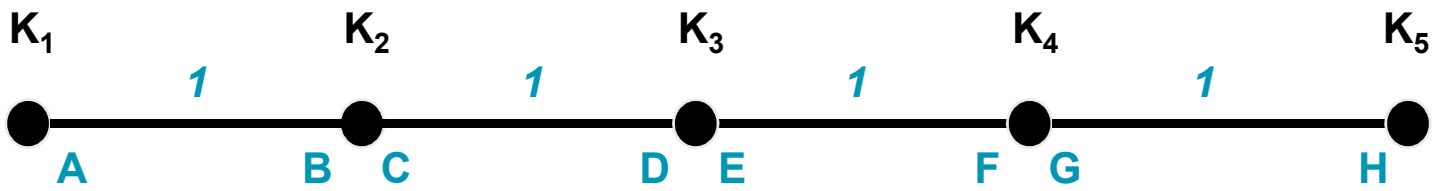
K_4

(K_1, 3) via F
(K_2, 2) via F
(K_3, 1) via F
(K_4, 0) local
(K_5, 1) via G

K_5

(K_2, 3) via H
(K_3, 2) via H
(K_4, 1) via H
(K_5, 0) local

Distance Vector Routing: Aufbau der Routing Tabelle (5)



Routing Tabellen nach dem 4. Update ($t = t_4$)

K_1	K_2	K_3	K_4	K_5
(K_1, 0) local	(K_1, 1) via B	(K_1, 2) via D	(K_1, 3) via F	(K_1, 4) via H
(K_2, 1) via A	(K_2, 0) local	(K_2, 1) via D	(K_2, 2) via F	(K_2, 3) via H
(K_3, 2) via A	(K_3, 1) via C	(K_3, 0) local	(K_3, 1) via F	(K_3, 2) via H
(K_4, 3) via A	(K_4, 2) via C	(K_4, 1) via E	(K_4, 0) local	(K_4, 1) via H
(K_5, 4) via A	(K_5, 3) via C	(K_5, 2) via E	(K_5, 1) via G	(K_5, 0) local

Distance Vector Routing (1)

- Beim Distance Vector Routing berechnet jeder Router, zu welchen Kosten er jeden Knoten im Netz erreichen kann

  Das ergibt eine Tabelle mit drei Einträgen:
  dem jeweiligen Knoten, den dazugehörigen Kosten und dem Routerport, der zu dem entsprechenden Knoten führt

  Die Kosten basieren auf der Metrik des Routing Protokolls. Häufig wird als Metrik nur der „Hop-Count“ verwendet

  Weitere Informationen berechnet bzw. speichert der Router nicht!

- Als „Distance-Vector“ wird die Tabelle aller Netzknoten mit den dazugehörigen Kosten bezeichnet. Diese Tabelle überträgt der Router an seine direkten Nachbarn

- Der Austausch der Distance Vectoren zwischen benachbarten Routern ist die grundlegende Aktion beim Distance Vector Routing!

Distance Vector Routing (2)

- Auf der Basis der von den Nachbarn erhaltenen Distanzvektoren berechnet bzw. aktualisiert der Router seine Routing Tabelle

  Die Berechnung basiert in der Regel auf dem Bellman-Ford-Algorithmus

- Jeder Router kennt nur einen einzigen Weg zu jedem Knoten und hat außer Kosten und dem Ausgangsport keine weiteren Informationen!!

- Die Reaktion auf „gute Nachrichten“ geht schnell: „good news travel fast“. Mit jedem Update wird die Information, z.B. dass eine Verbindung wieder verfügbar ist, weitergegeben

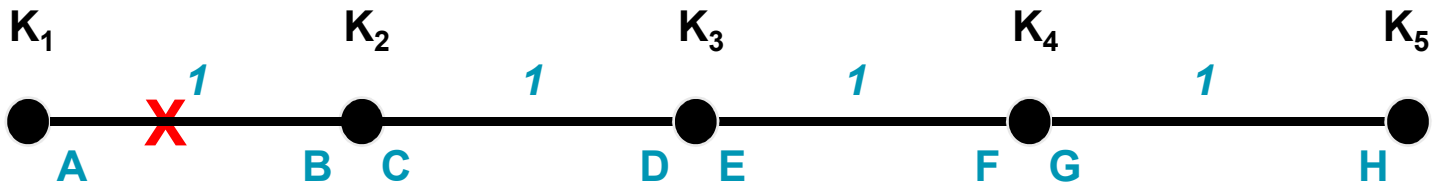
- Aber die Reaktion auf „schlechte Nachrichten“ (z.B. bei Ausfall einer Leitung) ist sehr langsam => „Count-to-Infinity-Problem“

  Das liegt daran, dass aus dem Distanzvektor nicht der „Weg“ sichtbar ist. Für einen Knoten ist so nicht feststellbar, ob der Weg z.B. über ihn selbst führt ...

- RIP (= Routing Information Protocol) gehört zu den Distance Vector Routing Protokollen

Distance Vector Routing: Count to Infinity Problem (1)

Was passiert beim Ausfall einer Leitung?



Routing Tabellen von K_2 und K_3 vor dem Ausfall ($t = t_4$)

K_2

(K_1, 1) via B
(K_2, 0) local
(K_3, 1) via C
(K_4, 2) via C
(K_5, 3) via C

K_3

(K_1, 2) via D
(K_2, 1) via D
(K_3, 0) local
(K_4, 1) via E
(K_5, 2) via E

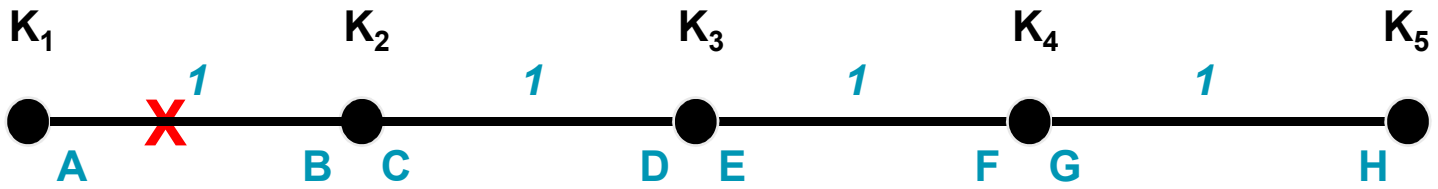
K_2 weiß **nicht** sofort, dass er K_1 nicht mehr erreichen kann!

Bei $t = t_5$ sendet K_3 seine aktuelle Routingtabelle an K_2 , in der K_1 noch enthalten ist.

Die Information über welches Interface K_1 für K_3 erreichbar ist, wird **nicht** mit übertragen!

Distance Vector Routing: Count to Infinity Problem (2)

Wie sieht die neue Routing Tabelle aus?



Routing Tabellen von K_2 und K_3 nach dem Ausfall ($t = t_6$)

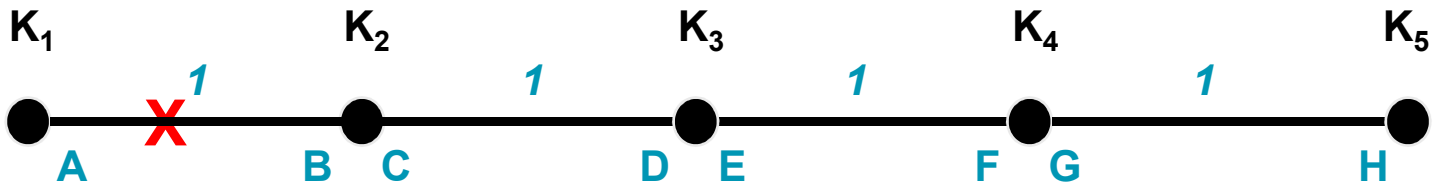
K_2

(K_1, 3) via C
(K_2, 0) local
(K_3, 1) via C
(K_4, 2) via C
(K_5, 3) via C

K_3

(K_1, 2) via D
(K_2, 1) via D
(K_3, 0) local
(K_4, 1) via E
(K_5, 2) via E

Distance Vector Routing: Count to Infinity Problem (3)



Routing Tabellen von K_2 und K_3 nach dem Ausfall ($t = t_7$)

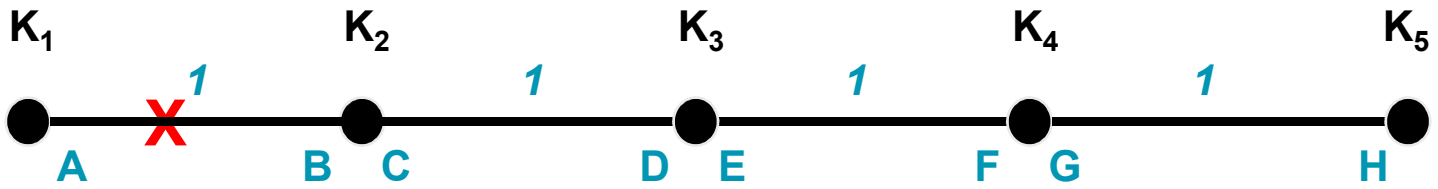
K_2

(K_1, 3) via C
(K_2, 0) local
(K_3, 1) via C
(K_4, 2) via C
(K_5, 3) via C

K_3

(K_1, 4) via D
(K_2, 1) via D
(K_3, 0) local
(K_4, 1) via E
(K_5, 2) via E

Distance Vector Routing: Count to Infinity Problem (4)



Routing Tabellen von K_2 und K_3 nach dem Ausfall ($t = t_8$)

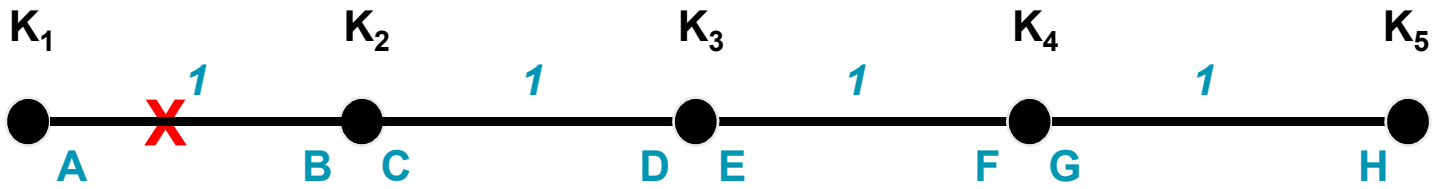
K_2

(K_1, 5) via C
(K_2, 0) local
(K_3, 1) via C
(K_4, 2) via C
(K_5, 3) via C

K_3

(K_1, 4) via D
(K_2, 1) via D
(K_3, 0) local
(K_4, 1) via E
(K_5, 2) via E

Distance Vector Routing: Count to Infinity Problem (5)



Routing Tabellen von K_2 und K_3 nach dem Ausfall ($t = t_9$)

K_2

(K_1, 5) via C
(K_2, 0) local
(K_3, 1) via C
(K_4, 2) via C
(K_5, 3) via C

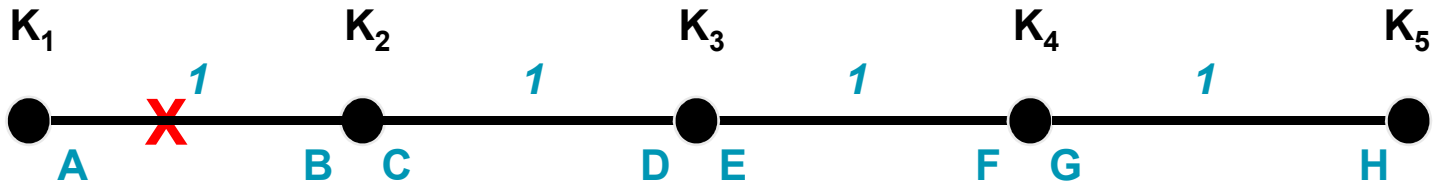
K_3

(K_1, 6) via D
(K_2, 1) via D
(K_3, 0) local
(K_4, 1) via E
(K_5, 2) via E

... das geht so weiter, bis der max. Metrikwert erreicht ist

=> "Count to Infinity Problem"

Split Horizon: als Abhilfe für das Count to Infinity Problem



Grundidee bei Split Horizon: ein Router versendet keine Routing Updates an Interfaces, über die er eine Route gelernt hat

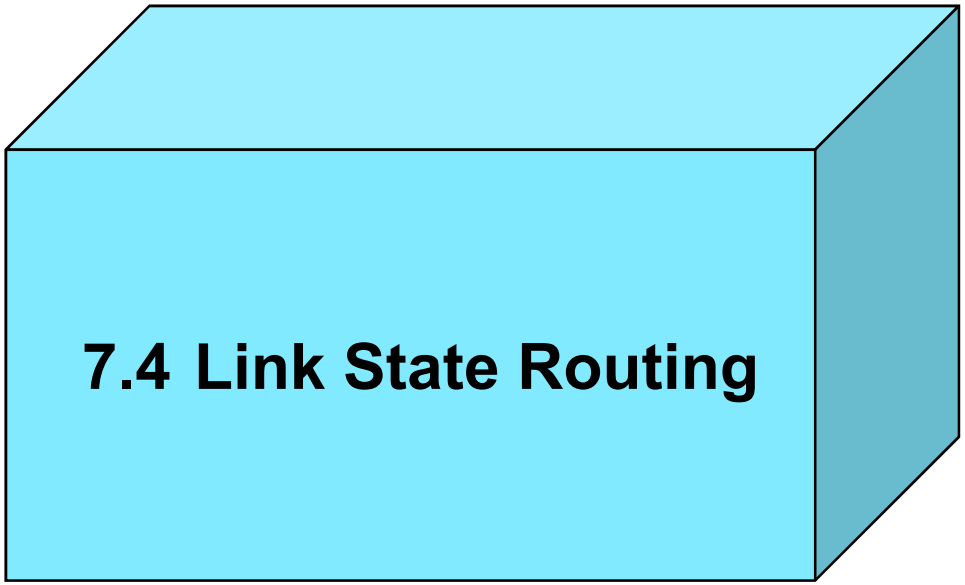
=> die Routingtabelle, die K_3 an K_2 sendet, enthält keinen Eintrag bezüglich der Erreichbarkeit von K_1

K_3 an K_2

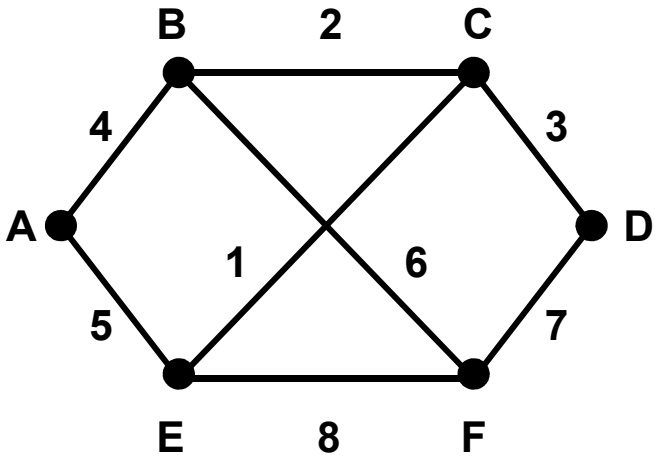
~~(K_1, 6)~~
(K_2, 1)
(K_3, 0)
(K_4, 1)
(K_5, 2)

Weitere Abhilfe mit Triggered Update:

=> Ein Router sendet seine Routing Tabelle nach einer Änderung erneut/ außer der Reihe


7.4 Link State Routing

Netz-Beispiel (1): Inhalt von Link State Paketen



- Link State Pakete von allen Knoten in jedem Router
- Berechnung der Routing Tabelle mit Dijkstras Algorithmus

A	
Seq. Nr.	
Timest.	
B	4
E	5

B	
Seq. Nr.	
Timest.	
A	4
C	2
F	6

C	
Seq. Nr.	
Timest.	
B	2
D	3
E	1

D	
Seq. Nr.	
Timest.	
C	3
F	7

E	
Seq. Nr.	
Timest.	
A	5
C	1
F	8

F	
Seq. Nr.	
Timest.	
B	6
D	7
E	8

Link State Routing

- Kernstück des Link State Routing ist die „Nachbarschaftserkundung“: Benachbarte Router tauschen in regelmäßigen Abständen Keepalives, sogenannte Hello Nachrichten aus. Durch das Ausbleiben einer Hello Nachricht können die Router den Ausfall von benachbarten Routern feststellen.
- Die Informationen über den Status und die Kosten der Verbindungen zu seinen direkten Nachbarn fasst jeder Router in Link State Advertisements (= LSA) zusammen. Diese LSAs werden an an alle Router im Netz verteilt.
- Ein Link State Advertisement enthält:
  - die Quelladresse d.h. die Adresse des Routers, der das Paket erzeugt hat
  - eine Sequenznummer
  - eine Zeitmarke
  - die Liste der direkten Nachbarn und Informationen über die Links zu ihnen
- Wichtiger Vertreter der Link State Routing Protokolle : OSPF (Open Shortest Path First)

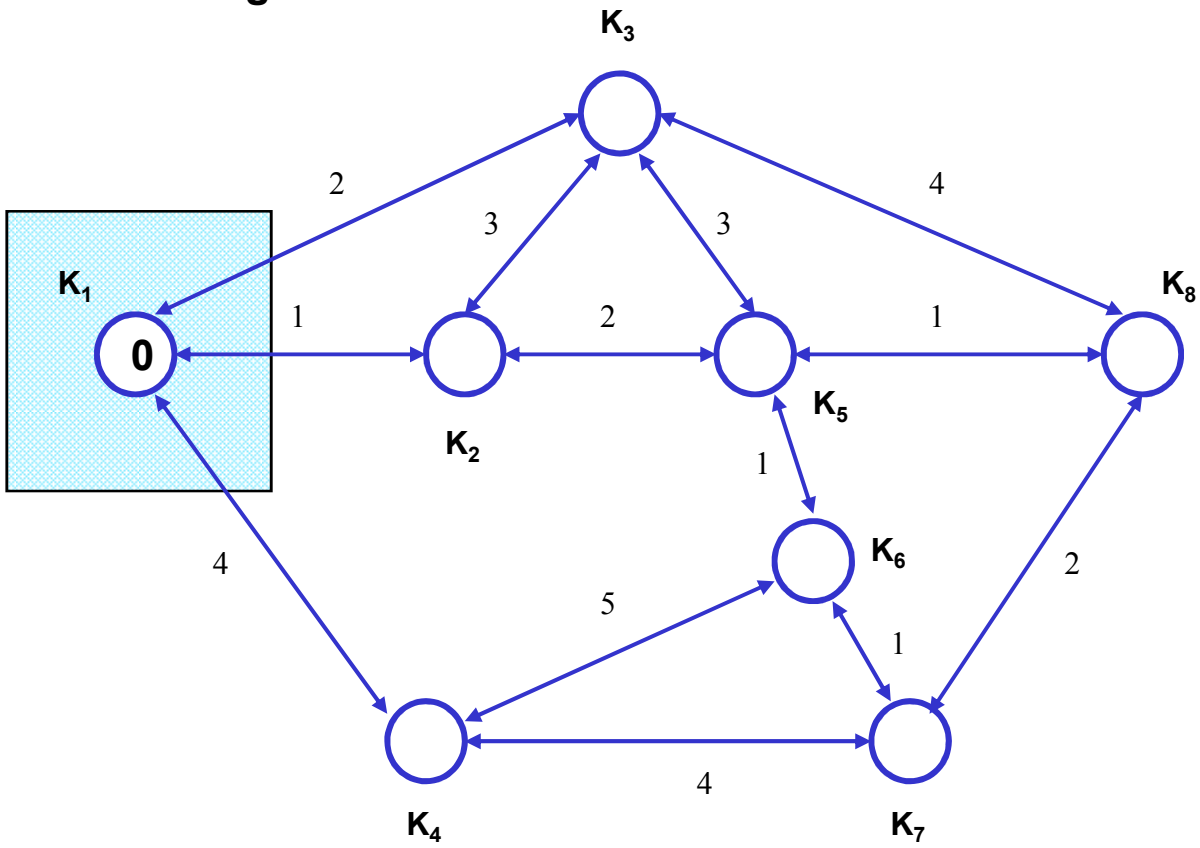
Ablauf des Link State Routing

- Ermittlung der Nachbarn (unter Verwendung von „Hello-Paketen“/ Keepalives)
- Bestimmung der Übertragungsbandbreite/ Kosten zu den Nachbarn
- LSA zusammenstellen und an alle Router verteilen
- Verteilung der LSAs über „Flooding“: ein Router verteilt ein neues, empfangenes LSA an alle Ausgänge (außer dem Empfangs-Link).
- LSAs, die ein Router schon erhalten/ gespeichert hat, werden nicht weiterverteilt
- ein LSA wird bei Netz-Änderungen generiert
- Nach Empfang der LSAs unter Verwendung des Dijkstras Algorithmus den kürzesten Pfad zu allen anderen Routern berechnen
- Jeder Router erhält ein komplettes Abbild des Netzes d.h. alle Wege sind sichtbar. Konvergenzprobleme gibt es nicht

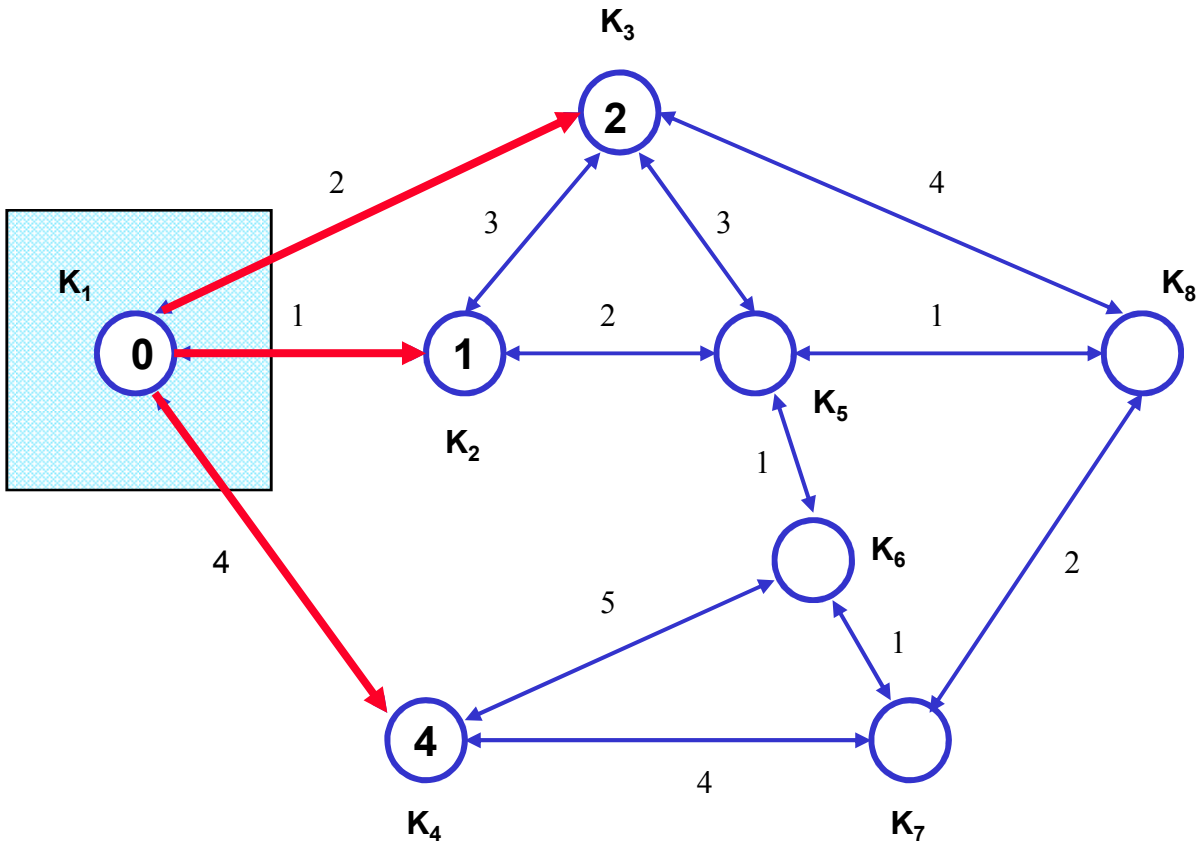
⇒ Protokoll ist sehr robust, aber es hat einen höheren Speicherbedarf als Distance Vector Routing Protokolle und benötigt mehr Router- Rechenleistung

Dijkstra Algorithmus (1):

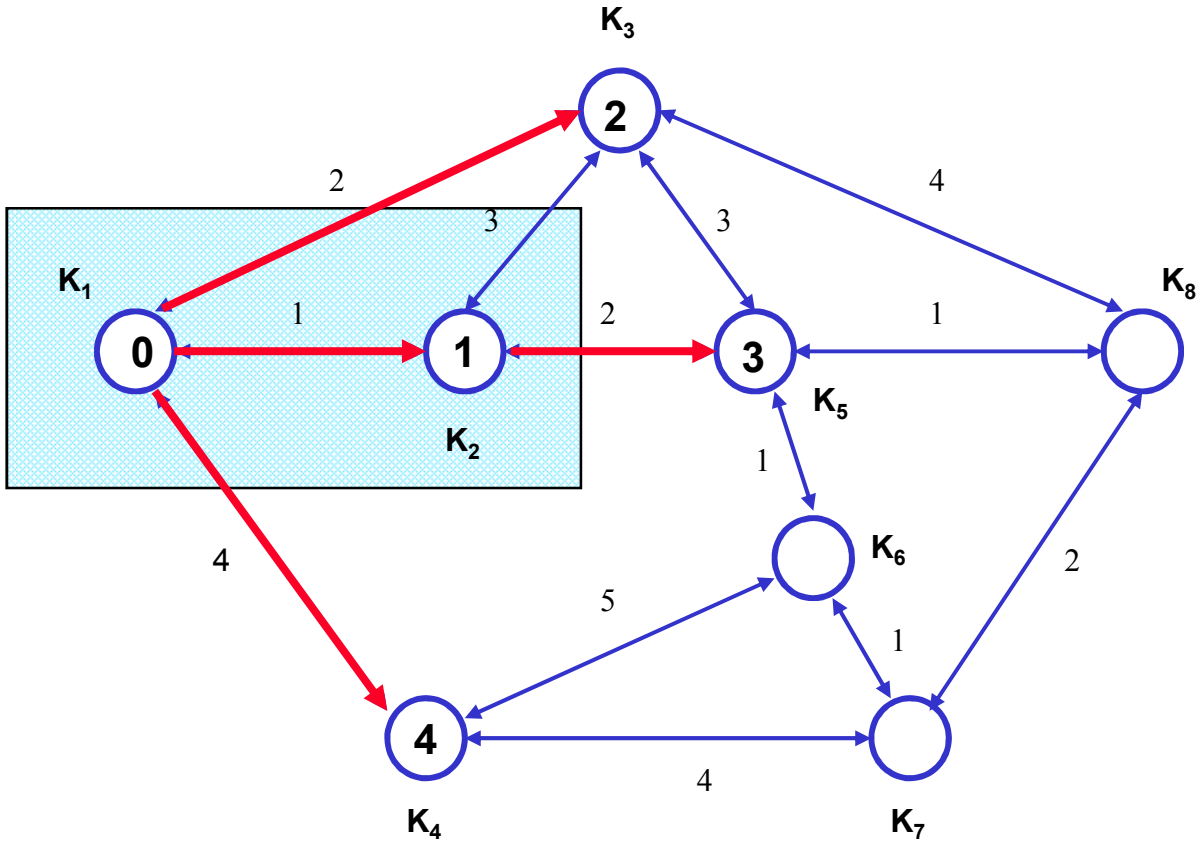
Schrittweiser Aufbau der Routing-Tabelle, der jeweils „günstigste“ Knoten wird weiterverfolgt



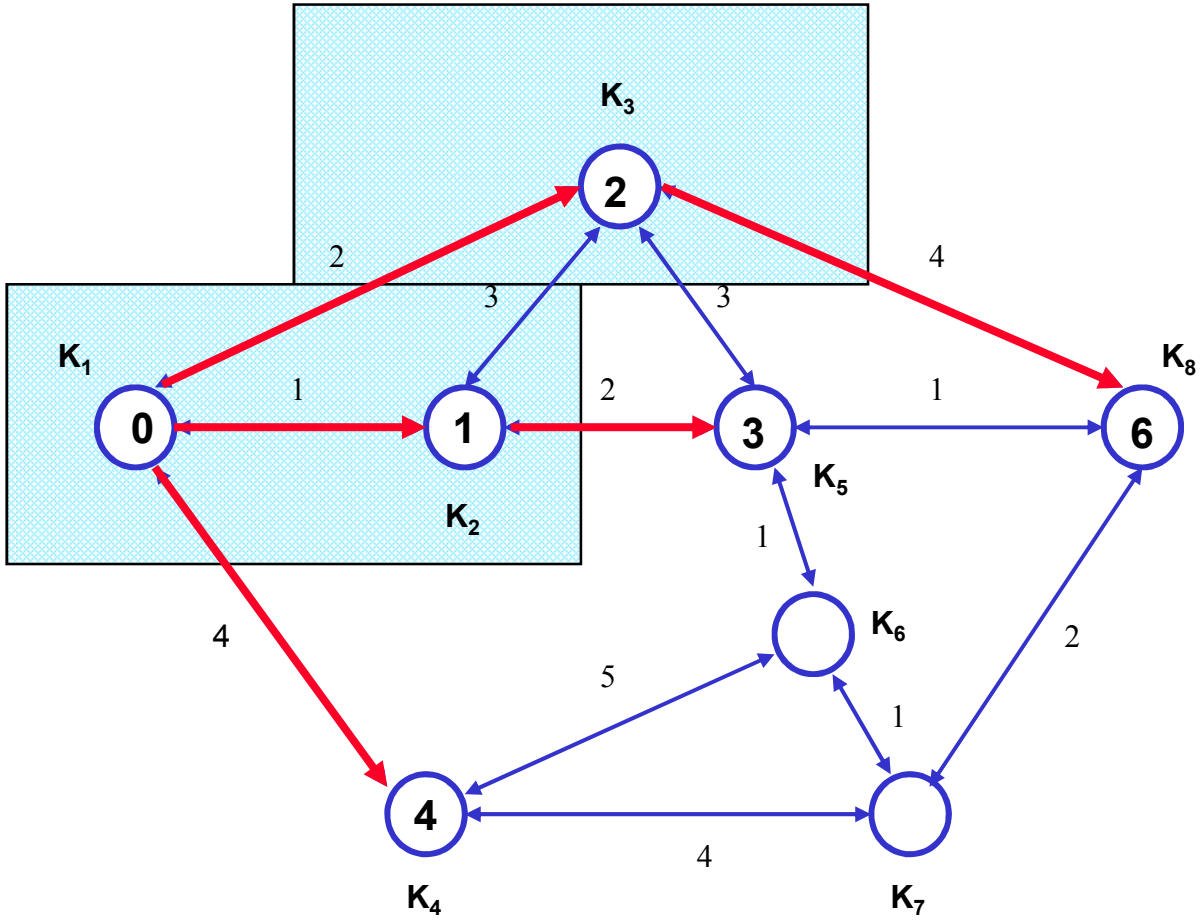
Dijkstra Algorithmus (2)



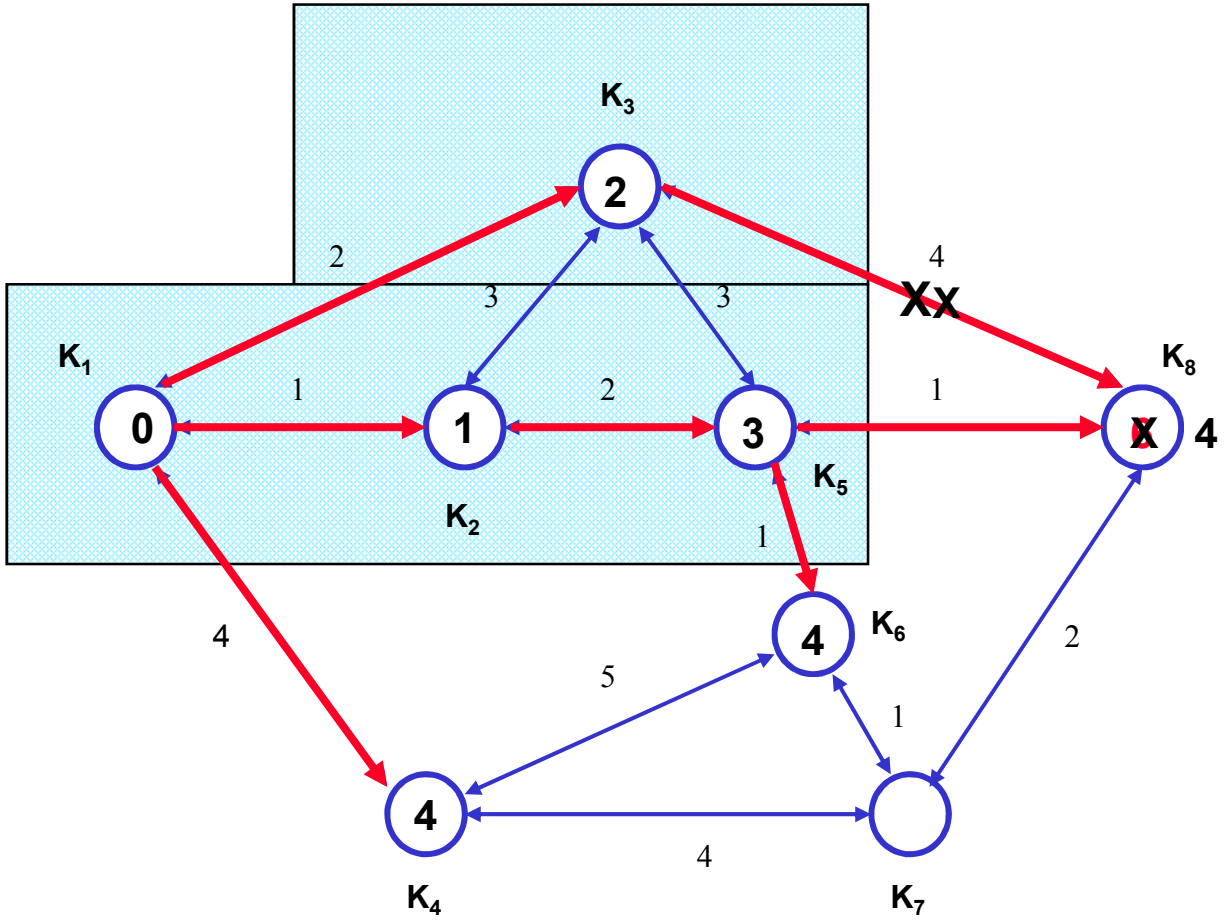
Dijkstra Algorithmus (3)



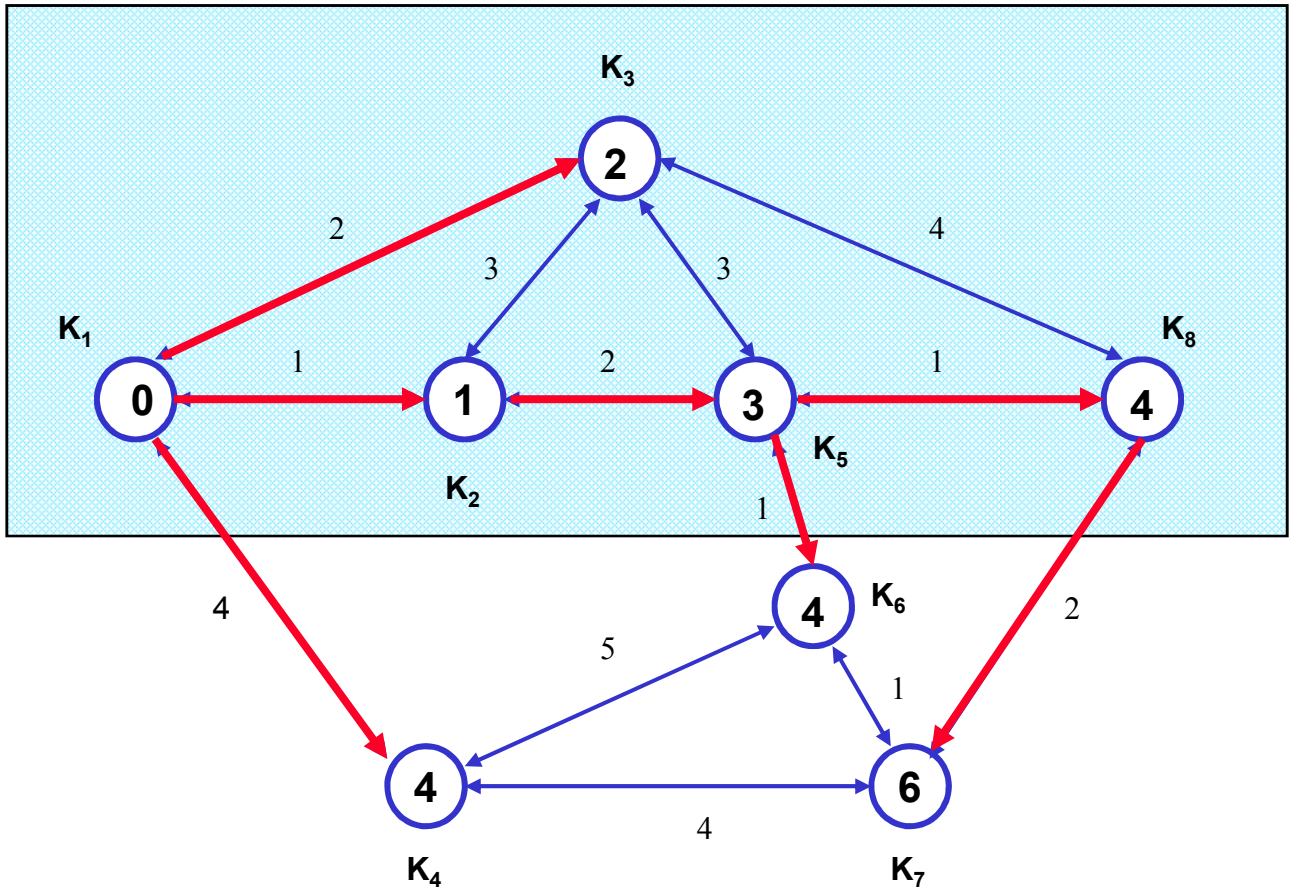
Dijkstra Algorithmus (4)



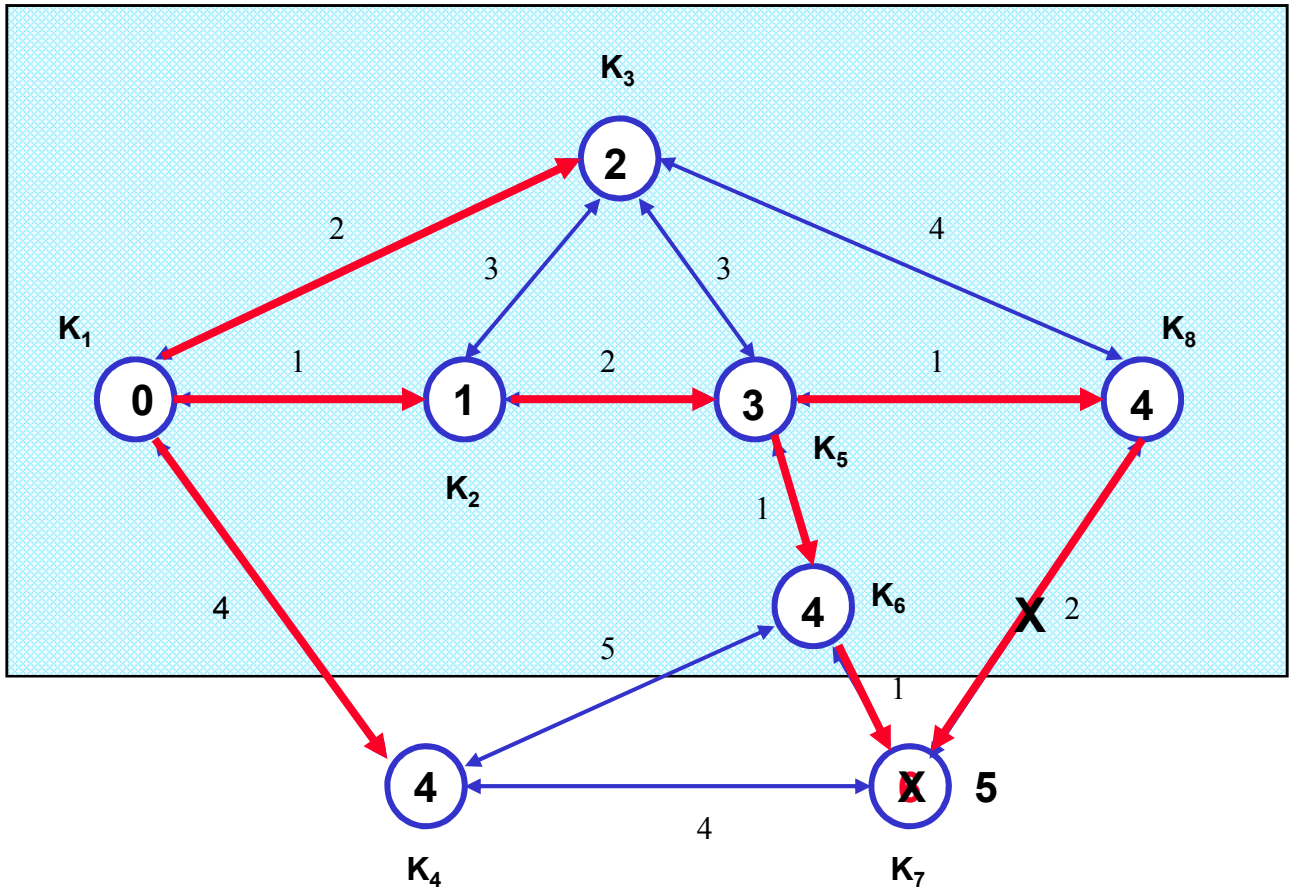
Dijkstra Algorithmus (5)



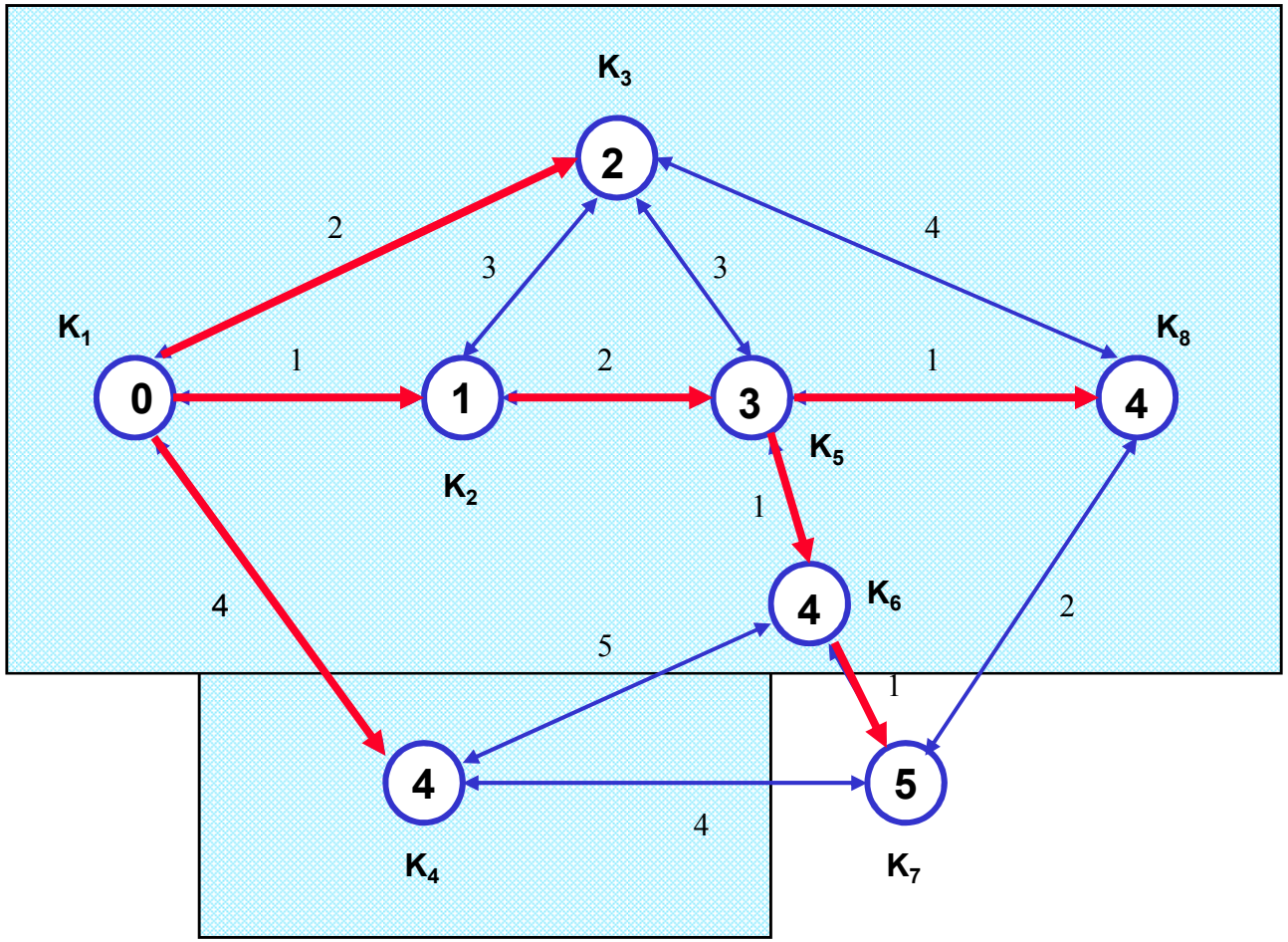
Dijkstra Algorithmus (6)



Dijkstra Algorithmus (7)



Dijkstra Algorithmus (8)



Dijkstra Algorithmus (9)