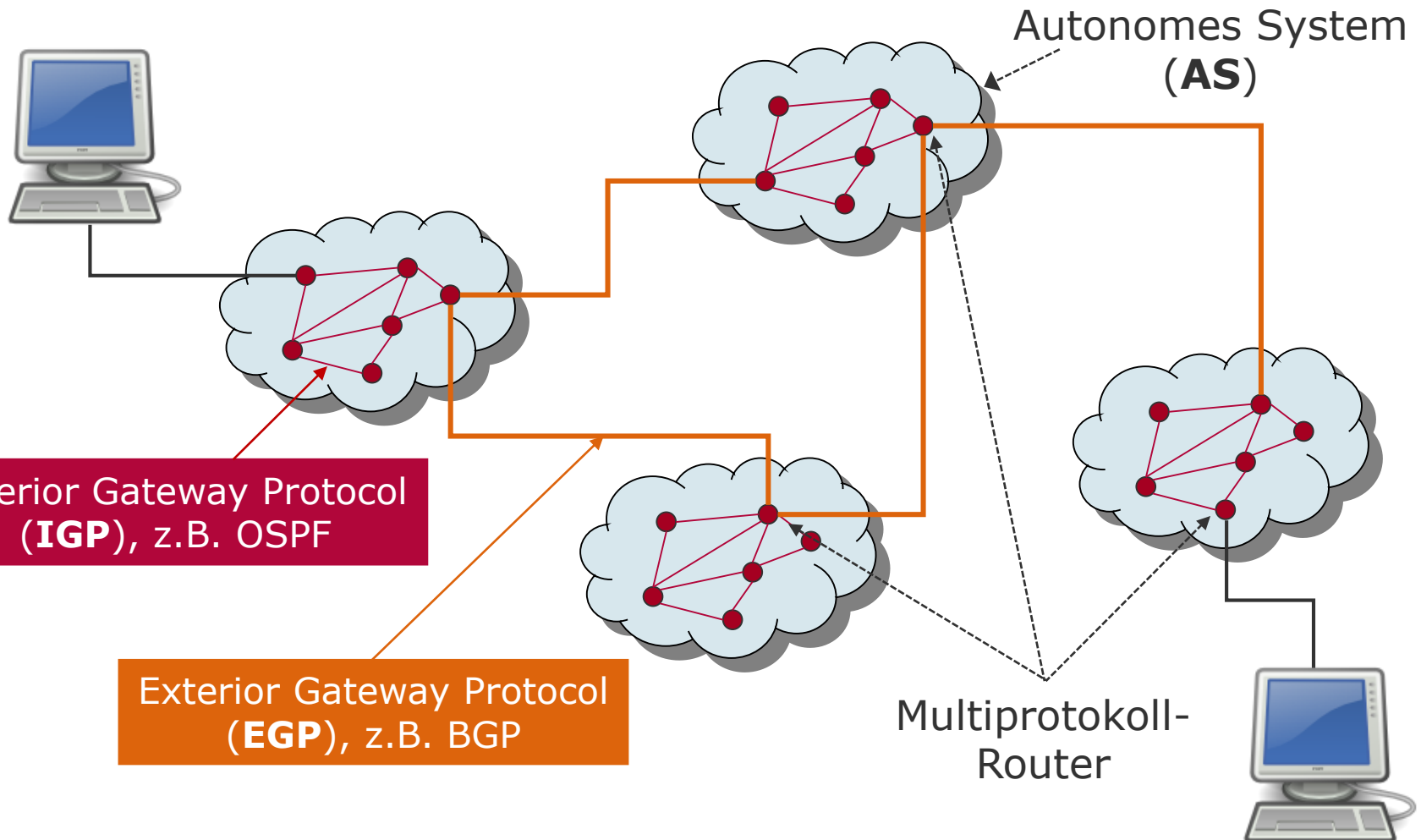




openHPI-Kurs: Wie funktioniert das Internet?

Routing im Internet

Prof. Dr. Christoph Meinel
Hasso-Plattner-Institut
Universität Potsdam

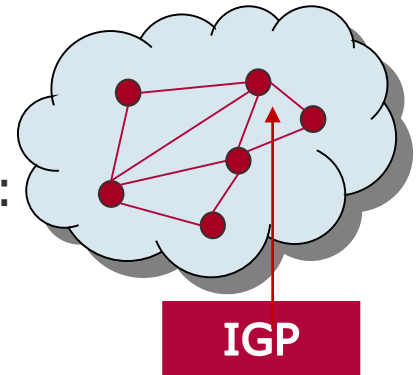


OSPF – Open Shortest Path First

Für das Routing innerhalb Autonomer Systeme ist das **Interior Gateway Protokoll – IGP** – zuständig

IGP basiert auf dem **OSPF – Open Shortest Path First**:

- Standardisiert in RFC 2328, ursprünglich schon 1989 eingeführt als RFC 1131
- **Link-State-Routing**, basiert auf **Dijkstra-Algorithmus**
- Erlaubt
 - schnelle, dynamische Anpassung an Topologieveränderungen
 - hierarchisches Routing
 - Routing im LAN (Broadcast-Netzwerk) und WAN
- unterstützt unterschiedliche Metriken im Internet (kürzeste Routen, billigste Routen, ...) und
- herstellerunabhängig



OSPF – Dijkstra-Algorithmus (1/2)

Netz wird als Graph G mit gewichteten Kanten modelliert

Ziel: Finde kürzeste Wege von Startknoten A zu allen Knoten von G

Vorgehensweise (1/2):

- Weise allen Knoten die beiden Eigenschaften **Distanz** und **Vorgänger** zu,
- führe zwei zunächst leere Knoten-Listen „**besucht**“ und „**gefunden**“ und
- initialisiere Distanz im Startknoten A mit 0 und in allen anderen Knoten mit ∞

OSPF – Dijkstra-Algorithmus (2/2)

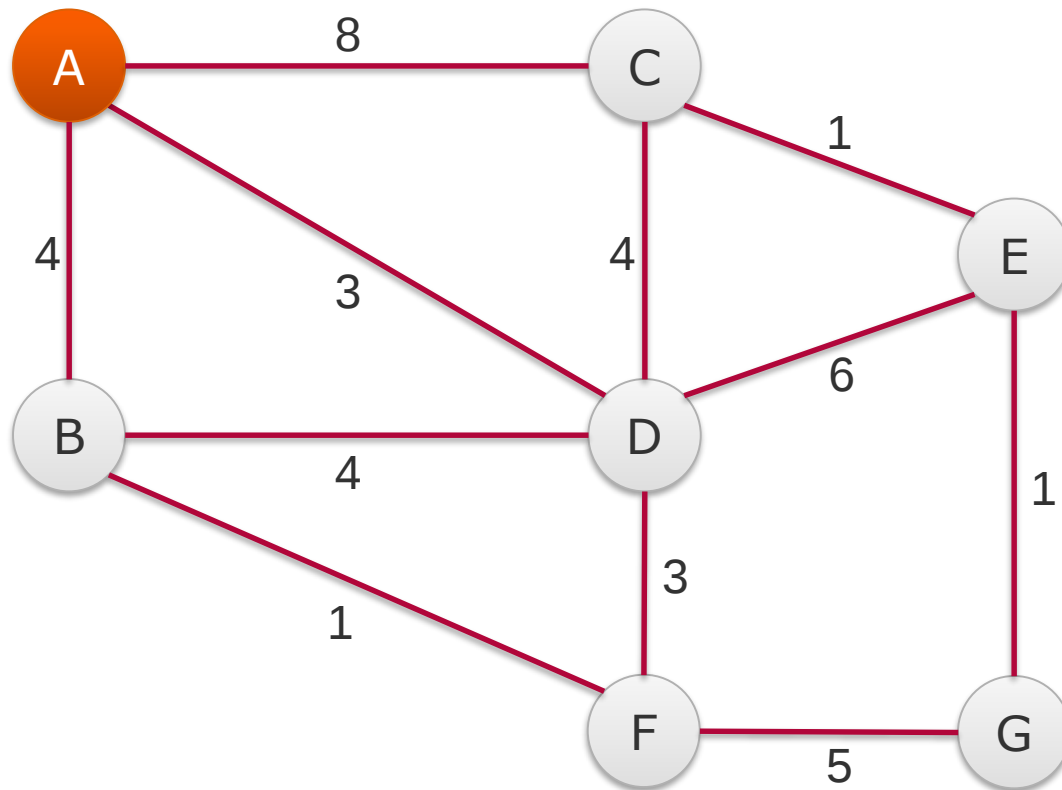
Ziel: Finde kürzeste Wege von Startknoten A zu allen Knoten von G.

Vorgehensweise (2/2):

Solange es Knoten gibt, die noch nicht gefunden sind,

- wähle einen Knoten mit **minimaler Distanz** aus und
 - speichere ihn Liste „**gefunden**“ und
 - berechne für alle seine Nachbarknoten, zu denen die kürzeste Distanz noch nicht gefunden wurde, die Summe des jeweiligen Kantengewichtes und der aktuellen Distanz
 - ist Knoten noch nicht in besucht-Liste, wird er eingefügt
 - Ansonsten prüfe, ob gerade berechneter Wert kleiner ist als die in der besucht-Liste gespeicherte Distanz
 - aktualisiere Distanz und setze den aktuellen Knoten als Vorgänger (Update)

Beispiel: Dijkstra-Algorithmus (1/8)



Ziel: Kürzeste Routen von A zu allen anderen Knoten finden

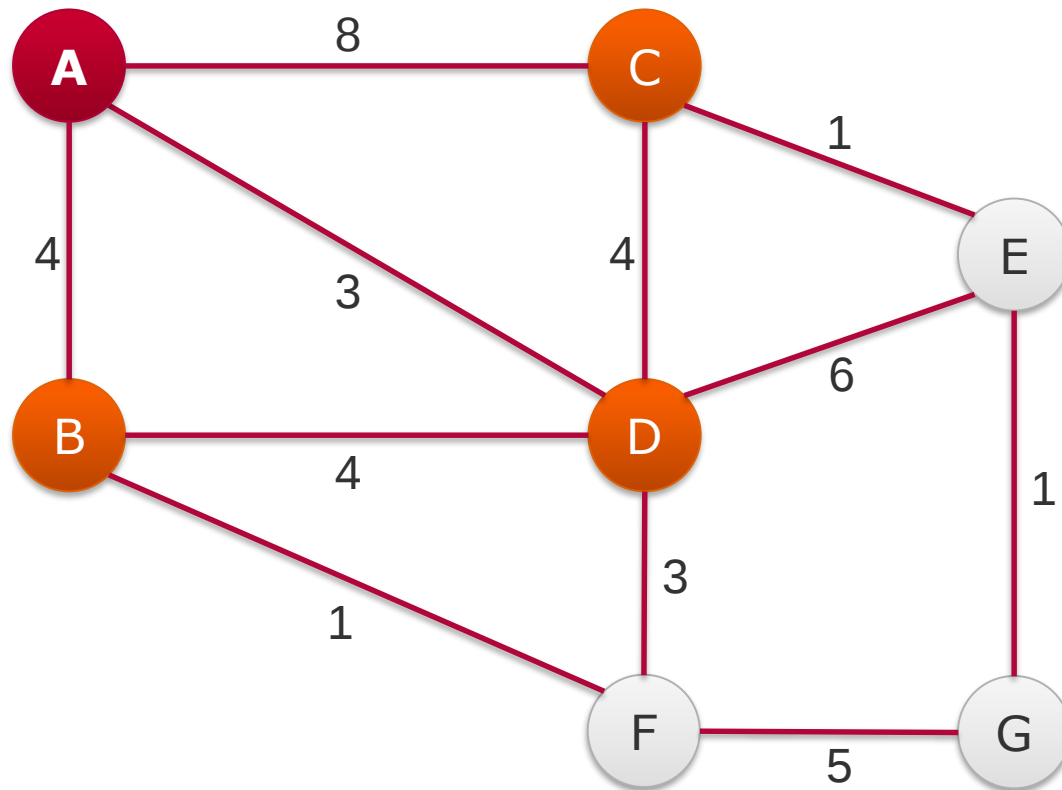
Wähle **Knoten A** aus besucht-Liste

Besucht

A:0

Kürzester Weg gefunden

Beispiel: Dijkstra-Algorithmus (2/8)

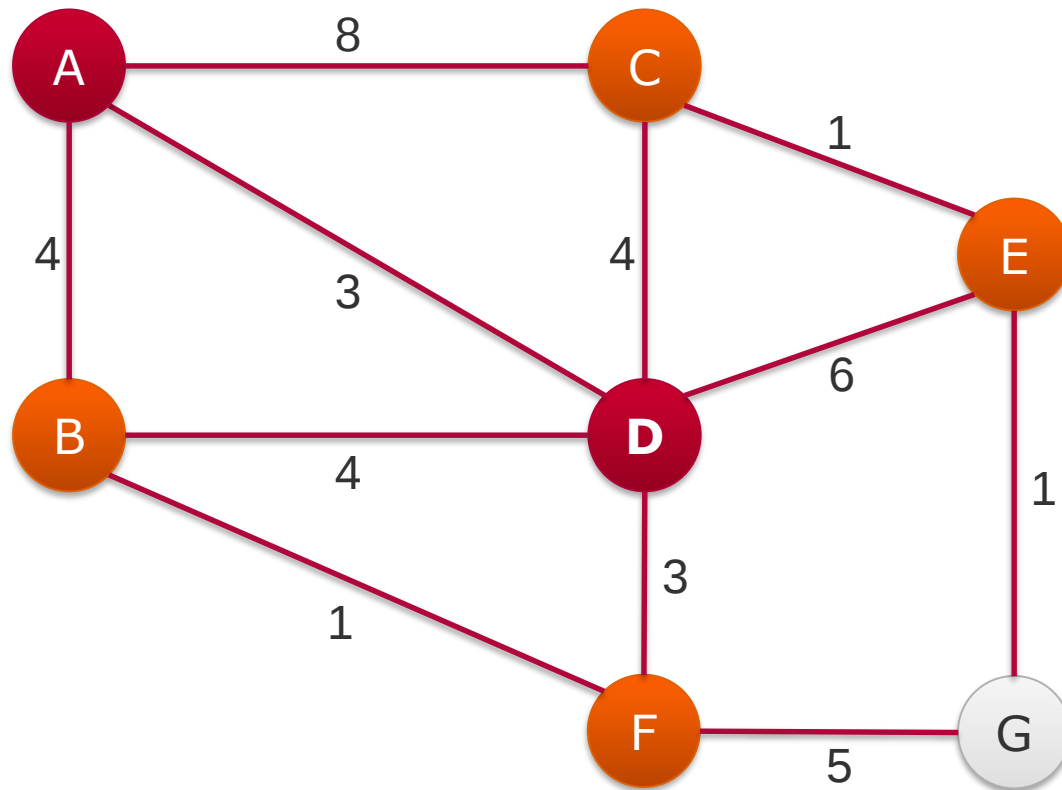


Ziel: Kürzeste Routen von A zu allen anderen Knoten finden

Wähle **Knoten D** aus besucht-Liste

Besucht	B:4, C:8, D:3
Kürzester Weg gefunden	A:0

Beispiel: Dijkstra-Algorithmus (3/8)



Ziel: Kürzeste Routen von A zu allen anderen Knoten finden

Update Distanz C (8 auf 7)

Wähle **Knoten B** aus besucht-Liste

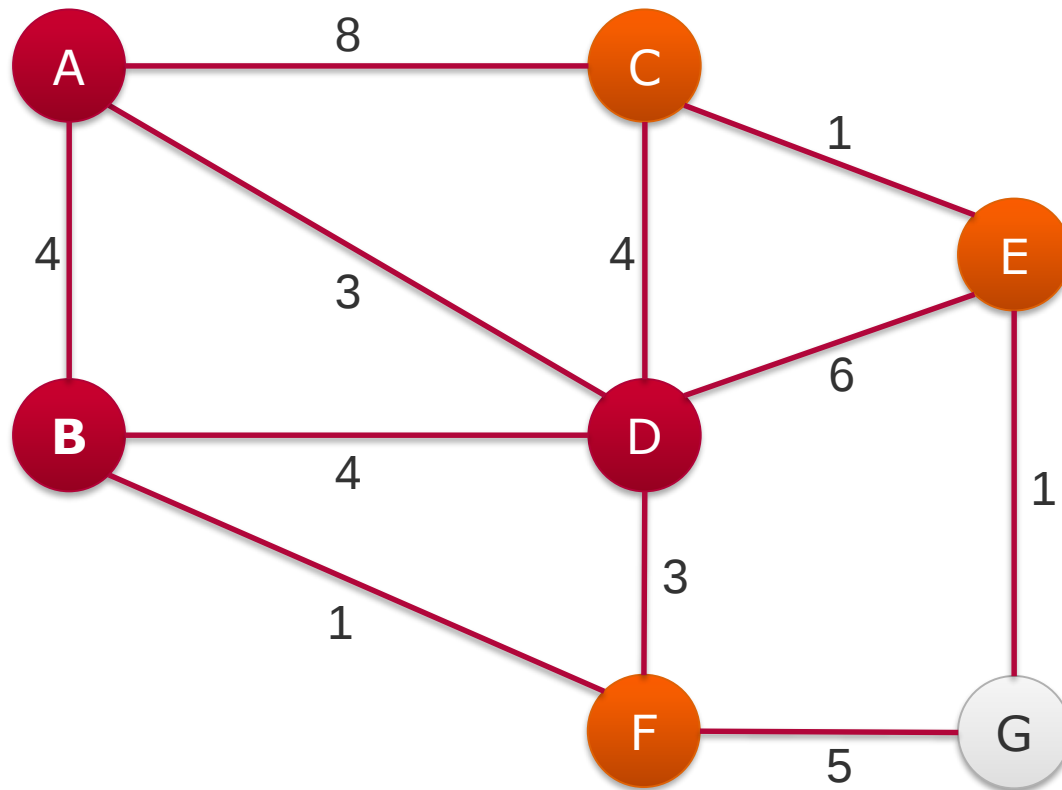
Besucht

B:4, C:7, E:9, F:6

Kürzester Weg gefunden

A:0, D:3

Beispiel: Dijkstra-Algorithmus (4/8)



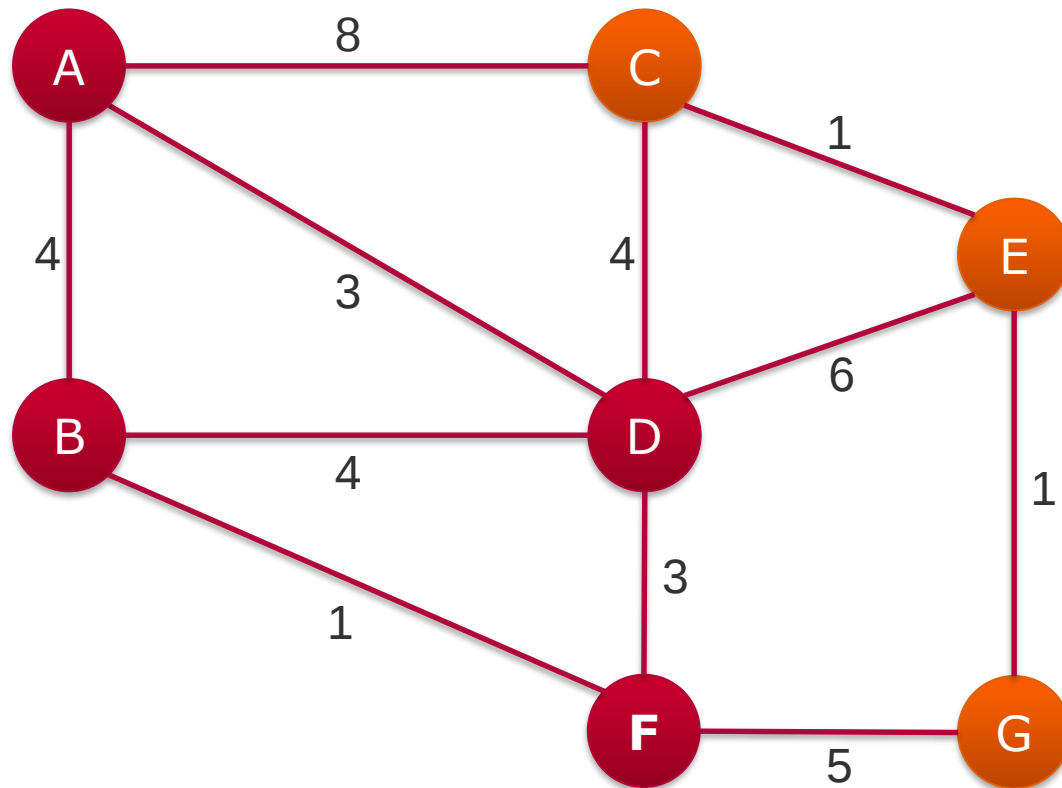
Ziel: Kürzeste Routen von A zu allen anderen Knoten finden

Update Distanz F (6 auf 5)

Wähle **Knoten F** aus besucht-Liste

Besucht	C:7, E:9, F:5
Kürzester Weg gefunden	A:0, D:3, B:4

Beispiel: Dijkstra-Algorithmus (5/8)

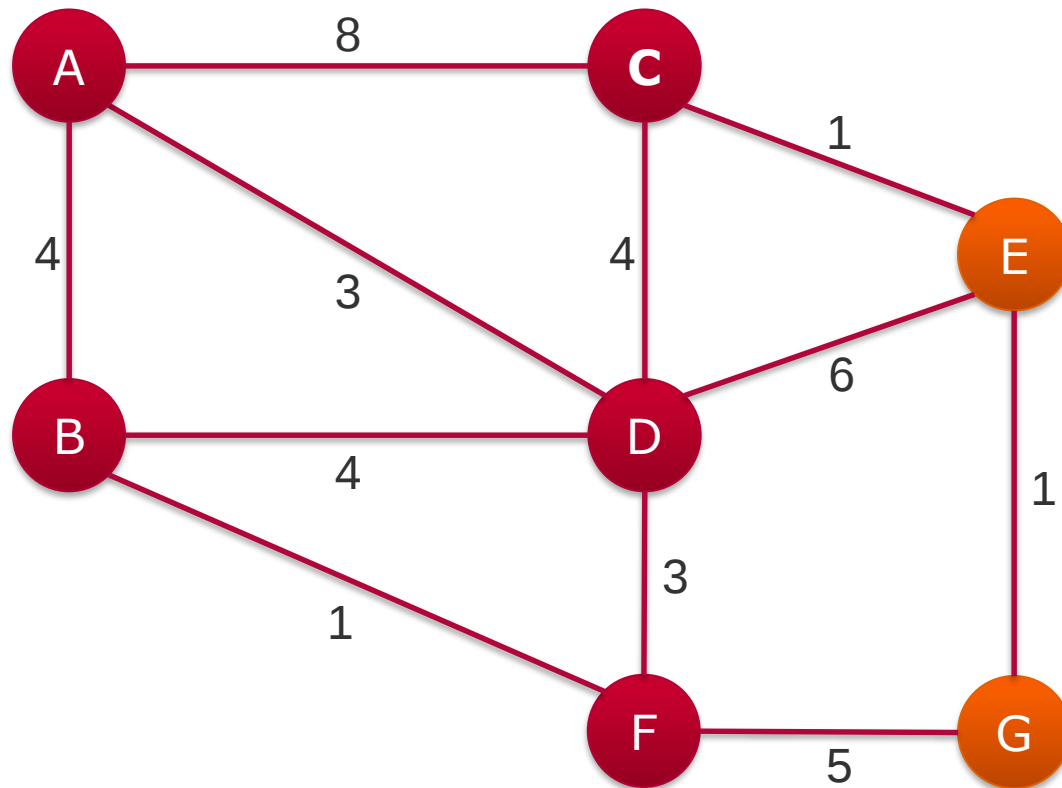


Ziel: Kürzeste Routen von A zu allen anderen Knoten finden

Wähle **Knoten C** aus besucht-Liste

Besucht	C:7 , E:9, G:10
Kürzester Weg gefunden	A:0, D:3 , B:4, F:5

Beispiel: Dijkstra-Algorithmus (6/8)



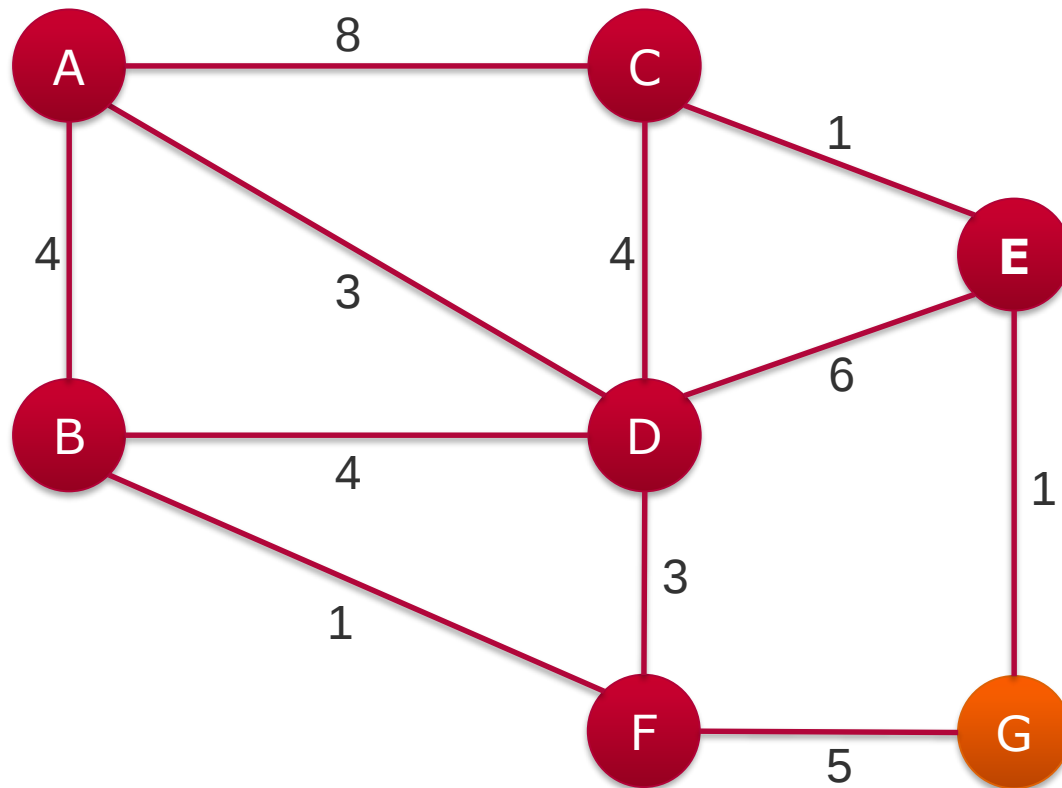
Ziel: Kürzeste Routen von A zu allen anderen Knoten finden

Update Distanz E (9 auf 8)

Wähle **Knoten E** aus Besucht-Liste

Besucht	E:8, G:10
Kürzester Weg gefunden	A:0, D:3, B:4, F:5, C:7

Beispiel: Dijkstra-Algorithmus (7/8)



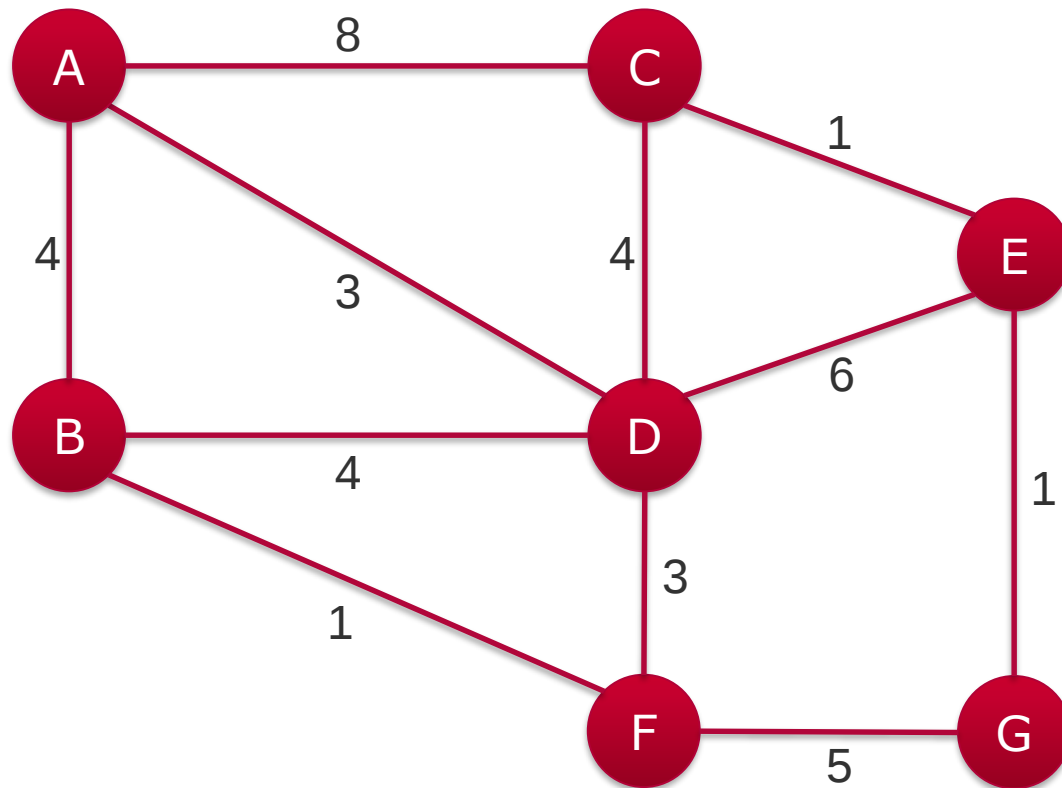
Ziel: Kürzeste Routen von A zu allen anderen Knoten finden

Update Distanz G (10 auf 9)

Wähle **Knoten G** aus besucht-Liste

Besucht	G:9
Kürzester Weg gefunden	A:0, D:3, B:4, F:5, C:7, E:8

Beispiel: Dijkstra-Algorithmus (8/8)



Ziel: Kürzeste Routen von A zu allen anderen Knoten finden

besucht-Liste leer
→ **Algorithmus terminiert**

Besucht

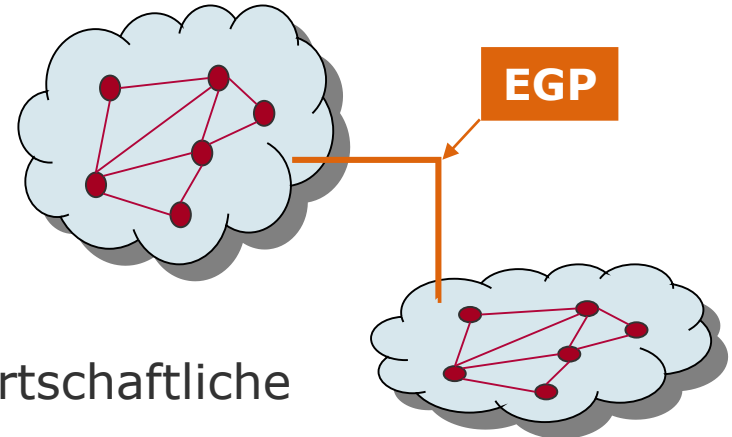
Kürzester Weg gefunden A:0, D:3, B:4, F:5, C:7, E:8, G:9

Nachrichtentypen für Kommunikation zwischen Routern

Hello	Wer sind meine Nachbarn?
Link State Update	Link-State-Aktualisierung an Nachbarn (Status, Metrik/Kosten)
Link State Ack	Bestätigung der Link-State-Aktualisierung
Database Description	Link-State-Pakete mit aktuellen Informationen des Senders
Link State Request	Link-State-Anforderung an Nachbarn zum Ermitteln der aktuellsten Verbindungsdaten

BGP – Border Gateway Protocol (1/2)

- Nur für AS-Grenz-Router(!), also für die Verbindung zwischen zwei Internet Service Providern
- Muss bestimmte Regelungen beachten
 - politische, sicherheitstechnische, wirtschaftliche
 - manuelle Konfiguration (Scripts)
 (bei IGP → geht es „nur“ um möglichst effiziente Paketzustellung)
- BGP-Router kommunizieren via TCP-Verbindung
 - zuverlässig, verbirgt Details der durchquerten Netze
- Grundsätzlich Distanzvektor-Routing
 - verwaltet Distanz UND zugehörigen Pfad
 - teilt Nachbarn den verwendeten Pfad mit

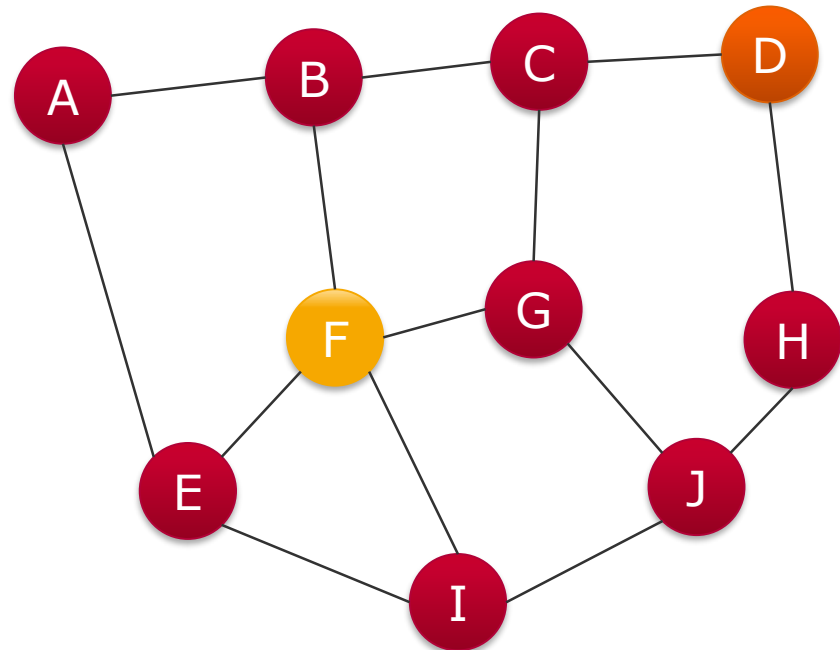


BGP – Border Gateway Protocol (2/2)

Beispiel:

Informationen, die **F** von seinen Nachbarn über den Router **D** erhält:

B:	B – C – D
G:	G – C – D
I:	I – J – H – D
E:	E – F – G – C – D



- F ermittelt z.B. kürzesten Pfad nach D,
endgültige Routingentscheidung trifft Administrator durch Router-Regeln