EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Vorlesung Grundlagen des Internets
SoSe 2024
Prof. Dr. M. Menth
F. Ihle

Übungsblatt 5

17. Mai 2024

Abgabe: 27. Mai 2024, 8:00. **elektronische Abgabe als PDF-Datei über Moodle!**

Aufgabe 5.1: *Routing-Algorithmen*

Routing-Algorithmen lassen sich verschiedenen Kategorien zuordnen. Man unterscheidet zwischen sogenannten *global routing algorithms* und *decentralized routing algorithms*. *Global routing algorithms* nutzen Kenntnisse über das gesamte Netzwerk, um Pfade zwischen den Knoten zu bestimmen. Bei *decentralized routing algorithms* haben Knoten keine Kenntnisse über das gesamte Netzwerk.

1. Ordnen Sie RIP und OSPF den oben genannten Kategorien von Routing-Algorithmen zu! Begründen Sie Ihre Antwort! 5 Punkte

2. Wie viele Iterationen (innere Schleifendurchläufe) des Dijkstra-Algorithmus müssen Sie in einem Netzwerk mit n Knoten durchführen? Geben Sie den Dijkstra-Algorithmus in *eigenem Pseudocode* wieder! 10 Punkte

3. Funktioniert der Dijkstra-Algorithmus mit negativen Kantengewichten? Begründen Sie Ihre Antwort! 5 Punkte

Gegeben sei das in Abbildung 1 dargestellte Netzwerk. Es besteht aus einer Reihe von Routern, die miteinander verbunden sind. Die Verbindungskanten geben als Parameter jeweils die Kosten der einzelnen Verbindungen an. Diese gelten in beide Richtungen.

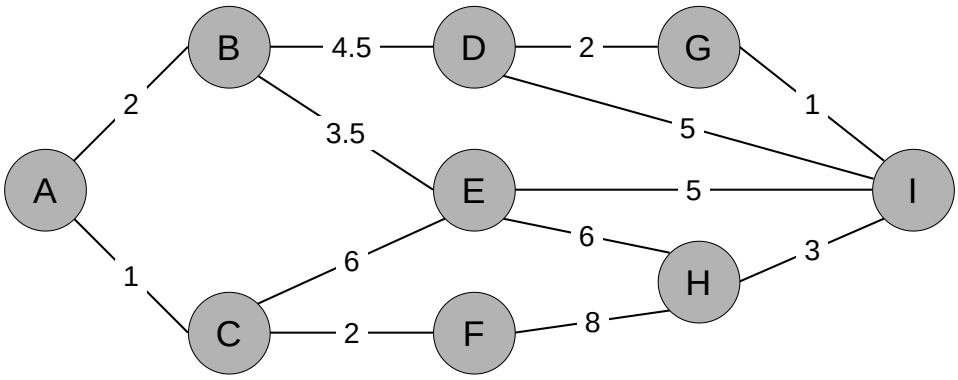

Abbildung 1: Netzwerk mit Link-Kosten.

4. Berechnen Sie mit Hilfe des Dijkstra-Algorithmus die kürzesten Wege von Knoten **A** zu allen anderen Knoten. Benutzen Sie zu diesem Zweck eine Tabelle in der folgenden Form: 35 Punkte

Schritt	Confirmed	Tentative
1	(A, 0, -)	{(C, 1, C), (B, 2, B)}
2		
...		

Hinweis:
Die Spalte *confirmed* enthält den Knoten, zu dem im jeweiligen Schritt der kürzeste Weg gefunden wurde. Die Liste *tentative* enthält alle noch zu untersuchenden Knoten, zu denen noch kein kürzester Weg gefunden wurde. Die Einträge der Listen *confirmed* und *tentative* sollen dabei die Form (Ziel, Kosten, *Next-Hop*) haben.
Der *Next-Hop* ist dabei der Nachbar von A, über den das Ziel mit den entsprechenden Kosten erreichbar ist. Der *Next-Hop* wird zum Aufbau der Routing-Tabelle verwendet.

Aufgabe 5.2: *Distance Vector Routing*

Gegeben ist ein Netzwerk, in dem ein *distance vector* Protokoll zum Aufbau der Routing-Tabellen verwendet wird.

1. Ein Knoten A habe von seinen Nachbarknoten B, C und D Distanzvektoren erhalten, die in Tabelle 1 dargestellt sind. Jede Zeile enthält einen Distanzvektor, der die Kosten angibt, zu denen der jeweilige Knoten die übrigen Knoten erreichen kann. Die Kosten für den Link von A nach B betragen 2, von A nach C 5, und von A nach D 1, d.h. die Kosten eines Links sind nicht symmetrisch und können sich je nach Richtung unterscheiden.

	A	B	C	D	E	F	G	H
B	3	0	3	8	1	7	3	2
C	2	1	0	3	4	2	2	1
D	1	4	2	0	6	4	3	6

Tabelle 1: Distanzvektoren, die A von seinen Nachbarn B, C und D erhalten hat.

Berechnen Sie für jeden Zielknoten B bis H die Kosten der Pfade, die entstehen, wenn Knoten A den Knoten B, C oder D als *Next-Hop* auswählt und suchen Sie aus diesen den günstigsten Pfad aus! Geben Sie die Rechnung in Tabellenform wie in Tabelle 2 an! Bauen Sie daraus die Routing-Tabelle des Knotens A auf, die aus den Spalten *Ziel*, *Kosten* und *Next-Hop* besteht. 30 Punkte

	B	...
B	2+0=2	...
C	5+1=6	...
D	1+4=5	...
Min	2,B	...

Tabelle 2: Beispieltabelle für Zielknoten B.

2. Betrachten Sie nun das Netzwerk in Abbildung 2, in dem die Kosten für alle Links gleich 1 seien! Die Routing-Tabellen in jedem Knoten seien korrekt aufgebaut und jeder Knoten hat zusätzlich den aktuellen Distanzvektor von seinen Nachbarn gespeichert. Nehmen Sie an, dass die Updates der Distanzvektoren zwischen den übrigen Knoten nicht synchron ausgetauscht werden, sondern nacheinander! Nach dem Empfangen eines Updates berechnet der Knoten erst seine neue Routing-Tabelle und schickt danach einen aktualisierten Distanzvektor an seine Nachbarn.

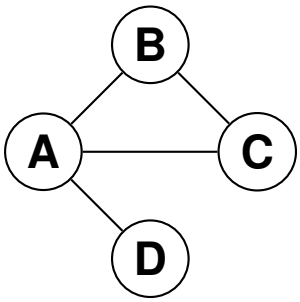

Abbildung 2: Netzwerk mit Link-Kosten.

Nun fällt der Link zwischen Knoten A und Knoten D aus. Daraufhin sendet Knoten A an Knoten B und C seinen neuen Distanzvektor. Nehmen Sie an, das Update von A erreicht den Knoten B früher als den Knoten C! B berechnet dann seine Routing-Tabelle neu und sendet den neuen Distanzvektor an C. Was passiert, wenn das Update von Knoten B den Knoten C früher erreicht als das Update, das A an C geschickt hat? Warum wird das als Count-To-Infinity-Problem bezeichnet? 15 Punkte

Gesamt: 100 Punkte