Tutoriumsblatt 7 mit Musterlösung

Aufgabe 7.1: Routing allgemein

a) Warum wird bei *Inter-AS-Routingprotokollen* eine Variante des *Distance-Vector-Routings* verwendet, statt Link-State-Routing einzusetzen?

b) Diskutieren Sie, warum eine *hierarchische Struktur* des Internets eine *gute Skalierbarkeit* zur Folge hat, so dass Millionen von Nutzern es nutzen können.

c) Vergleichen Sie den *Algorithmus von Link-State-Routing mit dem von Distance-Vector-Routing* und stellen Sie die Unterschiede heraus.

d) Meist wird der sehr abstrakte Begriff „Kosten“ gewählt, um die Nützlichkeit einer Verbindung zu kennzeichnen. *Was könnten Routing-Protokolle konkret betrachten, um Kosten zu modellieren?*

Lösung 7.1

Grundbegriffe:

- Routing-Protokoll: Protokoll zum Austausch von Routing-Information
- Routing-Algorithmus: Ermittlung der günstigsten Wege im Netz, Erzeugung von Einträgen für Routing-Tabelle
- Routing-Tabelle: Halten der Wegdaten

1.a) Zwischen autonomen Systemen hat man eine geringere Auswahl an Verbindungspunkten und damit möglichen Wegen, über die Daten ausgetauscht werden könnten. Die Pfade bleiben daher relativ lange unverändert und man spart den Aufwand des Floodings, der bei Link-State-Protokollen entsteht. Außerdem gibt es oft Verträge (Policies) zwischen den Betreibern autonomer Systeme, was und wie viel Daten ausgetauscht werden dürfen. (Und manchmal möchte man aus politischen oder finanziellen Gründen auch bestimmte Autonome Systeme meiden.) Daher werden üblicherweise statische Routen festgelegt, die den Policies genügen. Dennoch sollen Ausfälle kompensiert werden, so dass auch eine dynamische Komponente benötigt wird.

Distance-Vector-Algorithmen verursachen geringen Overhead beim Austausch von Routing-Informationen und werden so implementiert, dass ganze Pfade ausgetauscht werden, so dass der Betreiber entscheiden kann, welche Routing-Information er berücksichtigen will. Und der Austausch kompletter Pfade löst auch den Bouncing-Effect und das Count-to-Infinity-Problem.

Zusammengefasst:

- geringer Overhead,
- Kenntnis des gesamten Pfades hin zum Ziel,
- normalerweise stabile Routen, also auch keine Konvergenz-Probleme (langsame Konvergenz ist ein Nachteil von Distance-Vector-Algorithmen),
- kein Bouncing-Effekt, da vollständige Pfadinformationen übertragen werden und Schleifen in den Pfaden erkannt werden können.

1.b) Das Internet ist in autonome Systeme (AS) aufgeteilt, die jeweils eingenständige administrative Domänen mit lokaler Kontrolle darstellen. In einem AS verwenden alle Router dasselbe Routing-Protokoll. Die einzelnen AS sind mittels Gateways verbunden, welche Inter-AS-Routingprotokolle beherrschen, die den besten Pfad durch mehrere ASs bestimmen.

Das Skalierungsproblem wird also dadurch gelöst, dass die Intra-AS-Router lediglich die anderen Router im selben AS und die Gateways kennen müssen. Je mehr Netze installiert werden, desto weiter kann (sollte) man seine ASs aufsplitten, um dafür zu sorgen, dass eine überschaubare Anzahl an Einträgen in einer Weiterleitungstabelle gehalten werden muss. Durch eine Anordnung in Hierarchien bleibt auch für die Inter-AS-Router die Arbeit halbwegs überschaubar.

1.c) **Link-State**-Algorithmen benutzen *globales* und vollständiges Wissen über das gesamte Netzwerk, um den kostengünstigsten Pfad zwischen der Quelle und dem Ziel zu berechnen. (In den Protokoll-Implementierungen ermittelt jeder Router sein lokales Wissen und sendet es allen anderen Routern, damit alle ein globales Bild bekommen.)

Distance-Vector-Algorithmen berechnen den günstigsten Pfad iterativ und verteilt. Es wird lediglich *lokales* (und eventuell veraltetes) Wissen verwendet: Wissen über die eigenen Nachbarn und deren Entferungen zu anderen Routern. (Die Implementierungen tauschen also lokal ihr globales Wissen aus.)

1.d)

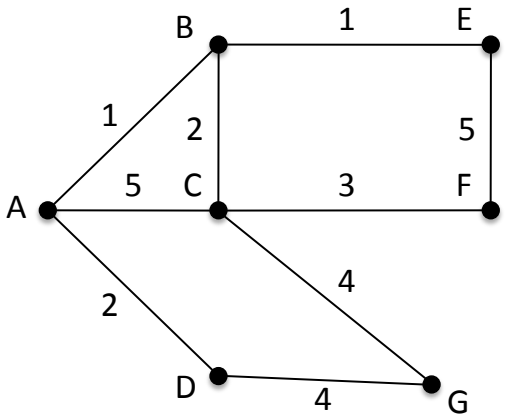
Hier gibt es viele Antwortmöglichkeiten:

- Zuverlässigkeit – welche Fehlerrate hat eine Leitung?
- Datenrate – welche Schrittgeschwindigkeit erlaubt mein Medium, was für Codes/Modulation werden eingesetzt?
- Aktuelle Auslastung einer Leitung
- Entfernung bzw. Latenz – wie lange brauchen meine Daten zwischen zwei Hops?
- Anzahl der Router: minimiere Zahl der Stationen, die ein Paket weiterleiten müssen (Hop Count). Dies ist zwar nichts, was man als Kosten mit einer Verbindung assoziieren kann, da diese immer zwischen genau zwei Routern existiert, aber man kann dieses Maß implementieren, indem die Kosten aller Verbindungen gleich gewählt werden (Hop-Count: Verbindungskosten = 1).
- Eingesetztes Medium: Glasfaser, Kupfer, Funk, ... – bevorzuge Glasfaser als bestes Medium, da es gleich viele weitere Kostenaspekte optimiert.
- Verbindungsart: halbduplex / vollduplex – erlaubt mein Medium eine Kommunikation in beide Richtungen gleichzeitig? Kann ich also die mögliche Datenrate tatsächlich erreichen?
- Netzbetreiber – gehören die Leitungen einem öffentlichen oder privaten Betreiber? Und welchem? Je nachdem muss ich vielleicht tatsächlich unterschiedliche monetäre Kosten aufwenden, um Daten weiterleiten zu dürfen.
- Politische Kosten – vielleicht möchte ich gerne einige Netze umgehen und andere bevorzugen (tatsächlich gibt es Provider in China, die Datenverkehr nach Europa lieber über die USA rooten).
- ... keine Garantie auf Vollständigkeit...

Aufgabe 7.2: Distance-Vector-Routing

Distance-Vector-Routing war das erste im Internet verwendete dynamische Routing-Verfahren. Jeder Knoten (Router) tauscht mit den Nachbarn Informationen über die eigene Routing-Tabelle aus, indem periodisch *Abstandsvektoren* der Form `(Ziel, Kosten)` aus der Routing-Tabelle generiert werden. Empfängt ein Knoten den Abstandsvektor eines Nachbarn, prüft er, ob er mittels der darin enthaltenen Informationen seine eigene Routing-Tabelle aktualisieren kann.

Betrachten Sie das folgende Netzwerk, in dem *Distance-Vector-Routing* eingesetzt wird. Die Knoten seien Router, die Kanten Leitungen zwischen den Routern. Die Beschriftungen der Kanten stellen ein Maß für die Kosten der Übertragung auf der entsprechenden Leitung dar. Alle Router seien gleichzeitig eingeschaltet worden und zeitlich synchronisiert, d.h. sie senden ihre Abstandsvektoren zum gleichen Zeitpunkt aus – und alle Abstandsvektoren kommen auch gleichzeitig an und werden zeitgleich verarbeitet. Zunächst ermitteln alle Router die Kosten zu ihren direkten Nachbarn (Initialisierung). Beispielsweise weiß Router A, dass er Router B mit Kosten 1 erreichen kann, Router C mit Kosten 5 und Router D mit Kosten 2.



a) Wie sehen die *Weiterleitungstabellen nach der Initialisierung* aus? Ergänzen Sie dazu die nachfolgende Tabelle. Als Beispiel ist hier die Routing-Tabelle von Router A schon eingetragen. Ist kein Wissen über ein Ziel vorhanden, tragen Sie ∞ in die entsprechende Zelle ein.

	A	B	C	D	E	F	G
A	-	B,1	C,5	D,2	∞	∞	∞
B							
C							
D							
E							
F							
G							

b) Nun versenden die Router zum ersten mal ihre Abstandsvektoren. *Wie sehen die Weiterleitungstabellen nun aus*? Tragen Sie die Werte in die folgende Tabelle ein. Als Beispiel ist hier die Routing-Tabelle von Router A schon eingetragen (nach der Initialisierung und dem Empfang der ersten Abstandsvektoren). Der Eintrag $(B, 2)$ in Zeile A und Spalte E bedeutet, dass ein Paket mit Quelle A und Ziel E mit Kosten 2 über Router B geleitet wird. Ist kein Wissen über ein Ziel vorhanden, tragen Sie auch hier wieder ∞ in die entsprechende Zelle ein.

	A	B	C	D	E	F	G
A	-	B,1	B,3	D,2	B,2	C,8	D,6
B							
C							
D							
E							
F							
G							

c) Die Router versenden nun ein weiteres mal ihre Abstandvektoren. *Wie sehen die Weiterleitungstabellen nun aus*? Sind die Tabellen nach diesem Update *stabil* oder würde ein weiteres Update noch Einträge verändern?

d) Was würde bei einem *Zusammenbruch der Leitung* $C - F$ passieren?

Lösung 7.2

Zwar müssen hier ganz, ganz viele Tabellen ausgefüllt werden – aber hinterher kann man schön sehen: jeder Router muss am Anfang nur seine eigene Umgebung kennen, und dieses Wissen wird nach und nach durchs Netz propagiert. Wie man bei den ganzen Schritten sehen kann, dauert es seeehr lange, bis Informationen propagiert sind. In Teil c) sieht man schön: es ist eine vierte Interation nötig, aber danach ist alles fertig berechnet. Und warum vier Iterationen? Weil der beste Pfad zwischen D und F über vier Hops geht. Je größer das Netz und je länger damit die besten Pfade sind, desto länger dauert es, bis alle optimalen Pfade gefunden sind.

2.a) Für die Initialisierung der Tabellen ermitteln die Router zunächst die Kosten zur ihren direkten Nachbarn (Kosten der Kanten). Damit haben die Router nach der Intialisierung folgende Einträge in ihren Routing-Tabellen:

	A	B	C	D	E	F	G
A	-	B,1	C,5	D,2	∞	∞	∞
B	A,1	-	C,2	∞	E,1	∞	∞
C	A,5	B,2	-	∞	∞	F,3	G,4
D	A,2	∞	∞	-	∞	∞	G,4
E	∞	B,1	∞	∞	-	F,5	∞
F	∞	∞	C,3	∞	E,5	-	∞
G	∞	∞	C,4	D,4	∞	∞	-

2.b) Routing-Tabellen nach dem ersten Update:

	A	B	C	D	E	F	G
A	-	B,1	**B,3**	D,2	**B,2**	**C,8**	**D,6**
B	A,1	-	C,2	**A,3**	E,1	**C,5**	**C,6**
C	**B,3**	B,2	-	**A,7**	**B,3**	F,3	G,4
D	A,2	**A,3**	**A,7**	-	∞	∞	G,4
E	**B,2**	B,1	**B,3**	∞	-	F,5	∞
F	**C,8**	**C,5**	C,3	∞	E,5	-	**C,7**
G	**D,6**	**C,6**	C,4	D,4	∞	**C,7**	-

Um die Tabelle auszufüllen, hilft es, wie folgt vorzugehen (hier beispielhaft für A):

- Welche Nachbarn hat A? Das sind B, C und D.
- Schreibe auf, welche Abstandvektoren man von den jeweiligen Nachbarn (hier B, C, D) bekommt. Betrachte jeden Knoten (A, B, ...) und addiere zu dem empfangenen Wert in jeder Zeile die vorne stehenden Leistungskosten auf. Wähle das Minimum und trage es zusammen mit dem Buchstaben des sendenden Knoten ein.

Nachbar	Leitungskosten	Empfangener Abstandsvektor						
		A	B	C	D	E	F	G
B	+1	(A,1)	(B,0)	(C,2)	∞	(E,1)	∞	∞
C	+5	(A,5)	(B,2)	(C,0)	∞	∞	(F,3)	(G,4)
D	+2	(A,2)	∞	∞	(D,0)	∞	∞	(G,4)
		Ergebnis Addition der Leitungskosten:						
B	+1	(A,2)	(B,1)	(C,3)	∞	(E,2)	∞	∞
C	+5	(A,10)	(B,7)	(C,5)	∞	∞	(F,8)	(G,9)
D	+2	(A,4)	∞	∞	(D,2)	∞	∞	(G,6)
	Minimum:	(A,0)	(B,1)	(B,3)	(D,2)	(B,2)	(C,8)	(D,6)

2.c) Die Einträge in den Routing-Tabellen nach dem zweiten Update, d.h. nachdem die Router ihre Abstandsvektoren ein zweites mal an ihre Nachbarn gesendet haben:

	A	B	C	D	E	F	G
A	-	B,1	B,3	D,2	B,2	**B,6**	D,6
B	A,1	-	C,2	A,3	E,1	C,5	C,6
C	B,3	B,2	-	**B,5**	B,3	F,3	G,4
D	A,2	A,3	**A,5**	-	**A,4**	**A,10**	G,4
E	B,2	B,1	B,3	**B,4**	-	F,5	**B,7**
F	**C,6**	C,5	C,3	**C,10**	E,5	-	C,7
G	D,6	C,6	C,4	D,4	**C,7**	C,7	-

Die Tabellen sind noch nicht stabil, wie durch scharfes Hinsehen zu erkennen ist. (Der Weg vom Knoten D zu F würde sich beim nächsten Update nochmal ändern und in den betroffenen Routingtabellen (D, F) aktualisiert. Die entsprechenden Einträge sind oben rot gekennzeichnet.)

2.d) Da die Update-Nachrichten zeitgleich ausgetauscht werden, ist die Reihenfolge der Abarbeitung auf den Knoten nicht festgelegt (race condition). Darüber hinaus kann es zu Ungenauigkeiten in der Synchronisation kommen. So kann es passieren, dass C zwar die Route $C \rightarrow F$ mit Kosten ∞ in seine Tabelle einträgt, aber von B ein Update bekommt, bevor es seine eigenen Updates rausschickt. Es kommt dann zu folgendem Ablauf:

Schritt	Knoten	Erhält von	Nachricht	Folge
1	C	B	B → F: 5	C → F: (B,7)
2	B	C	C → F: 7	
	B	E	E → F: 5	B → F: (E,6)
	G	C	C → F: 7	G → F: (C,11)
3	C	B	B → F: 6	C → F: (B,8)
	A	B	B → F: 6	A → F: (B,7)
4	D	A	A → F: 7	D → F: (A,9)
	G	C	C → F: 8	G → F: (C,12)

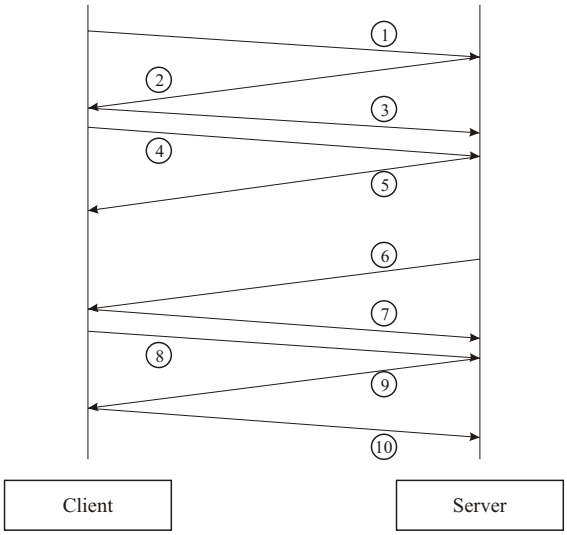
Wie man sieht, führen die Update-Nachrichten kurzfristig zu einem Loop zwischen C und B (Bouncing Effect), der aber von B in Schritt 2 durch die geringen Kosten des Routings über E durchbrochen wird. Es ist wiederum zu erkennen, dass nach der Änderung der Netzwerktopologie das DVR-Verfahren nur langsam konvergiert (4 Update-Iterationen notwendig).

Aufgabe 7.3: TCP-Verbindung

a) Eine TCP-Instanz setzt in den versendeten Segmenten üblicherweise das `ACK`-Flag auch dann, wenn es eigentlich nichts zu bestätigen gibt – im Prinzip entspricht dies der Wiederholung einer vorherigen Quittung. *Warum wird dies wohl gemacht?*

b) Welche *grundlegenden Aufgaben* hat der *Verbindungsaufbau* bei TCP? Welche der *Verbindungsabbau*?

c) Im Folgenden soll eine TCP-Verbindung zwischen einem Client und einem Server nachgebildet werden. In der folgenden Abbildung ist der zeitliche Ablauf der Übertragung skizziert. Gehen Sie davon aus, dass im Laufe dieser Kommunikation der Client genau eine Anfrage der Länge 20 Bytes stellt und der Server eine Antwort der Länge 1000 Bytes sendet. Des Weiteren initialisiert der Client seine Sequenznummer mit 432 und der Server seine mit 987.



Erstellen Sie eine Tabelle, in der Sie für die Segmente 1 bis 10 jeweils angeben, welche Flags gesetzt sind und welche Werte die Sequenz- und Bestätigungsnummern haben.

Lösung 7.3

3.a) Eine vorherige Quittung könnte verloren gegangen sein. Da es nichts kostet, in einem Segment, welches sowieso versendet werden muss, eben noch das ACK-Flag zu setzen und eine Quittungsnummer einzutragen, wird daher dauerhaft die Quittung neu übertragen. Der Empfang von lauter Duplikaten (DUP-ACKS) wird beim Empfänger der Quittungen in diesem Fall keine Aktion auslösen, da keine weiteren Daten gesendet wurden. DUP-ACKS werden nur dann als negative Quittung aufgefasst, wenn weitere, unbestätigte Daten unterwegs sind.

3.b) Ein Verbindungsaufbau hat allein schon die wichtige Aufgabe, die Empfangsbereitschaft des Kommunikationspartners zu testen; ohne diesen Test würde man eventuell Daten versenden, die der Empfänger gar nicht entgegennehmen kann oder will.

Zweite Aufgabe ist die Synchronisation: die Kommunikationspartner teilen sich mit, ab welcher Sequenznummer sie ihren Bytestrom durchnummerieren werden. Andernfalls wäre es nicht möglich, beim Empfang eines Segments sicher zu sein, dass es wirklich das erste ist und nicht bereits vorher etwas verloren gegangen ist. Darüber hinaus werden noch weitere Parameter für die Verbindung ausgehandelt – beispielsweise die MSS, um zu vermeiden, dass bereits in einem der LANs Fragmentierung vorgenommen werden muss, oder die Behandlungsstrategie für die Neuübertragung von Segmenten.

Drittens reservieren die Kommunikationspartner hier Bufferspeicher und teilen sich dessen Größe gegenseitig mit.

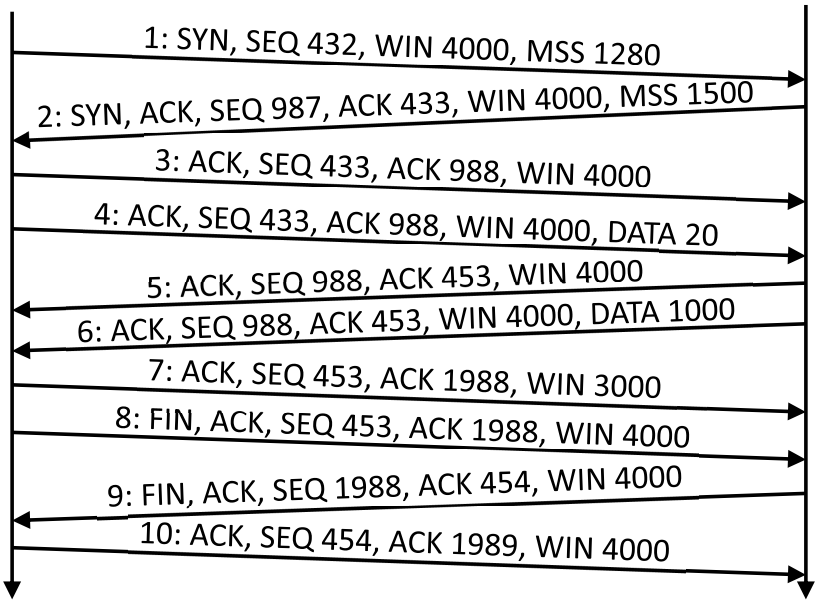
Der Verbindungsabbau dient der Freigabe von Ressourcen – die Kommunikationspartner geben den Bufferspeicher und die Einträge in der Verbindungsverwaltungstabelle wieder frei.

3.c)

Segment	Flags	Sequenznummer	Bestätigungsnummer
1	S	432	-
2	S,A	987	433
3	A	433	988
4	(A)	433	(988)
5	A	988	453
6	(A)	988	(453)
7	A	453	1988
8	F,(A)	453	(1988)
9	F,A	1988	454
10	A	454	1989

Achtung: ein (A) bedeutet, dass das ACK-Flag und die Ack-Nummer eigentlich nicht gesetzt werden müssten, da bereits vorher eine entsprechende Quittierung erfolgt ist, dass TCP sie aber in der Praxis doch setzen wird, siehe Aufgabenteil a).

Zwar nicht in der Aufgabe gefordert, aber hier das ganze noch mal als Diagramm und mit weiteren Angaben wie Window, um zu einer vollständigeren Ansicht zu kommen:



Hier nehmen wir an, dass die Daten direkt an die Anwendung weitergereicht werden und daher das Window immer gleich groß bleibt. Außer bei der siebten Nachricht – beim Senden des ACKs wurden die Daten noch nicht an die nächst höhere Schicht weitergereicht, beim anschließenden FIN (Nachricht 8) allerdings schon. (Man braucht sich hier nicht zu fragen, warum gerade an der einen Stelle das Window kleiner wird. Das war rein willkürlich so gewählt – es war ja nichts dazu in der Aufgabe vorgegeben.)