Marek Kubica kubica@in.tum.de, Michael Kerscher kerschem@in.tum.de

Punkteverteilung:

16	17	18	Σ

Aufgabe (16)

(a)

	A	B	C	D	E
A	0	∞	∞	∞	∞
B	∞	0	∞	∞	∞
C	∞	∞	0	∞	∞
D	∞	∞	∞	0	∞
E	∞	∞	∞	∞	0

	A	B	C	D	E
A	0	1	4	∞	∞
B	1	0	1	1	∞
C	4	1	0	2	∞
D	∞	1	2	0	3
E	∞	∞	∞	3	0

	A	B	C	D	E
A	0	1	2	2	∞
B	1	0	1	1	4
C	2	1	0	2	5
D	2	1	2	0	3
E	5	4	5	3	0

	A	B	C	D	E
A	0	1	2	2	5
B	1	0	1	1	4
C	2	1	0	2	5
D	2	1	2	0	3
E	5	4	5	3	0

(b)

Ziel	Nächster Hop	Kosten
A	A	0
B	B	1
C	B	2
D	B	2
E	B	5

(c) Von A über B über C. Es wird angenommen, dass das Paket an jedem Knoten Verarbeitungszeit T_V benötigt.

$T_V + \frac{8L}{\frac{10}{1} \cdot 10^6 \frac{Bit}{s}} + T_A + T_V + \frac{8L}{\frac{10}{1} \cdot 10^6 \frac{Bit}{s}} + T_A + T_V$

$3T_V + 2(T_A + \frac{8L}{10^7 \frac{Bit}{s}})$

(d) $2T_V + \frac{8L}{\frac{10}{4} \cdot 10^6 \frac{Bit}{s}} + T_A$

(e) Von A nach C:

$2ms + \frac{8L}{\frac{10}{4} \cdot 10^6 \frac{Bit}{s}} + 1ms = 3ms + \frac{8L}{\frac{10}{4} \cdot 10^6 \frac{Bit}{s}} = 0.0003s + \frac{8L}{\frac{10}{4} \cdot 10^6 \frac{Bit}{s}}$

Von A über B nach C:

$5ms + \frac{16L}{10^7 \frac{Bit}{s}} = 0.0005s + \frac{16L}{10^7 \frac{Bit}{s}}$

Vergleich:

$0.0005 + \frac{16L}{10^7} < 0.0003 + \frac{8L}{2.5 \cdot 10^6}$

$0.0002 + \frac{16L}{10^7} < \frac{8L}{2.5 \cdot 10^6}$

$2000 + 16L < 32L$

$2000 < 16L$

$125 < L$

(f) Distanz, Leitungsmaterial, Leitungsauslastung, Zuverlässigkeit

(g)

	A	B	C	D	E
A	0	1	2	2	5
B	1	0	1	1	4
C	2	1	0	2	5
D	2	1	2	0	3
E	5	4	5	3	0

	A	B	C	D	E
A	0	1	2	2	5
B	1	0	1	1	4
C	2	1	0	2	5
D	2	1	2	0	∞
E	∞	∞	∞	∞	0

	A	B	C	D	E
A	0	1	2	2	5
B	1	0	1	1	∞
C	2	1	0	2	∞
D	2	1	2	0	∞
E	∞	∞	∞	∞	0

	A	B	C	D	E
A	0	1	2	2	∞
B	1	0	1	1	∞
C	2	1	0	2	∞
D	2	1	2	0	∞
E	∞	∞	∞	∞	0

(h) Bei *Split Horizon* wird zusätzlich gespeichert von welchem Router Informationen „erlernt“ wurden. Diese von einem Netzwerk erlernten Informationen werden an alle anderen Router verteilt bis auf das Netzwerk, von dem die ursprüngliche Information stammte.

Aufgabe (17)

(a)

Step	N	A	B	C	D	E
0	{ A }	d=0, p=null	d=inf, p=undef	d=inf, p=undef	d=inf, p=undef	d=inf, p=
1	{ B, D, E }	d=0, p=null	d=1, p=A	d=inf, p=undef	d=3, p=A	d=6, p=u
2	{ C, E }	d=0, p=null	d=1, p=A	d=2, p=B	d=3, p=A	d=4, p=B
3	{ E, F }	d=0, p=null	d=1, p=A	d=2, p=B	d=3, p=A	d=3, p=C
4	{ F }	d=0, p=null	d=1, p=A	d=2, p=B	d=3, p=A	d=3, p=C

(b) Der im folgenden aufgeführte Scheme-Code bestimmt die Routingtabelle, über die Datenstruktur p (als assoziative Liste implementiert).

```
(define p '(
            (A #f)
            (B A)
            (C B)
            (D A)
            (E C)
            (F E)))

(define is-a?
  (lambda (candidate)
    (eq? candidate 'A)))

(define get-next-to-a
  (lambda (entry)
    (let ((node (car entry))
          (prev (cadr entry)))
      (cond
        ((is-a? node) #f)
        ((is-a? prev) node)
        (else (get-next-to-a (assoc prev p)))))))

(define to-target
  (lambda (entry)
    (let ((node (car entry)))
      (list node (get-next-to-a entry)))))

(map to-target p)
```

Aufgabe (18)

Beim *Link-State*-Routing wird der beste Pfad durchs Netzwerk im Gegensatz zum *Distance-Vector*-Routing mithilfe der Kenntnis sämtlicher Pfade im Netzwerk berechnet. Jeder Router sendet hierbei seine gesamten lokalen Informationen an alle anderen per Broadcast. Mithilfe dieser Daten kann z.B. mit dem Dijkstra-Algorithmus der bestmögliche Pfad ermittelt werden.

Im Gegensatz dazu wird beim *Distance-Vector*-Routing Information nur mit direkten Nachbarn ausgetauscht. Zu Beginn kennt ein Router nur seine eigenen Nachbarn und kann die Kosten dorthin berechnen. Diese Information sendet der Router an die Nachbarn, die diese Informationen nutzen um ihre eigenen Daten zu verbessern. Kann ein Router eine günstigere Route durch die Infos des Nachbars generieren, so teilt er dies allen Nachbarn mit. Dies geschieht solange, bis ein stabiler Zustand erreicht wird.

Im Vergleich zeigt sich, dass Link-State bessere Konvergenzeigenschaften als Distance-Vector aufweist weil in ersterem nach dem ersten Durchlauf aller Router-Informationen die gesamte Topologie bekannt ist und sofort Pfade berechnet werden können. Das Distance-Vector-Verfahren muss jedoch solange Information austauschen, bis jeder Router die besten Pfade zu allen anderen Routern berechnet haben und sich diese Information nicht mehr verändert. Bei häufigen Veränderungen im Netz ist das Link-State-

Verfahren geeigneter, weil hierbei nur die Änderungen mit den anderen Routern ausgetauscht werden muss.

Der Rechenaufwand ist beim Link-State-Verfahren aufwändiger, weil die gesamte Topologie einbezogen und der Dijkstra-Algorithmus (oder ähnlicher) ausgeführt werden muss. Distance-Vector-Verfahren sind günstiger zu implementieren, da sie mit weitaus weniger komplexen Routing-Informationen arbeiten müssen.