ROUTING

Problemstellung:

Wie werden Packete von Netzwerk zu Netzwerk geleitet?

Lernziel:

- Die Teilnehmer sollen dynamisches Routing erklären können.

Inhalt:

- Routing
- Distance Vector Routing (RIP)
- Link State Routing (OSPF)
- Link Metrics
- Routing für mobile Rechner

Routing

Zur Erinnerung:

- Unter **Forwarding** versteht man das aktuelle Weiterleiten von Packeten: Ein Router liest die Empfängeradresse, konsultiert seine Forwarding-Tabelle um den "next hop" auf dem Weg zum Empfänger zu finden, und sendet das Packet dort hin.
- Unter **Routing** versteht man den Aufbau der Forwarding-Tabellen.

Im allgemeinen Sprachgebrauch bedeuten die Begriffe *Forwarding-Tabelle* und *Routing-Tabelle* das selbe, nämlich die gerade erwähnte Tabelle zum Weiterleiten von Packeten. Manchmal wird aber zwischen den beiden Begriffen unterschieden, dann bezeichnet *Routing-Tabelle* eine Datenstruktur, die das Routing-Verfahren verwendet, um die *Forwarding-Tabellen* aufzubauen.

Ein weiterer Unterschied ist daß Forwarding lokal in einem Router stattfindet während am Routing alle Router eines Netzwerks gemeinsam beteiligt sind.

Beim Verbindungs-orientierten Packet Switching (Virtual Circuits) finden die Routing-Aktivitäten beim Verbindungsaufbau (Signaling) statt.

Routing Domain

Da beim Routing alle Router eines Netzwerks kommunizieren müssen, tritt sofort wieder das Problem der Skalierbarkeit (Anwendbarkeit auf große Netzwerke mit vielen Routern) auf. Die in diesem Kapitel diskutierten Routing-Verfahren sind zur Anwendung innerhalb einer administrativen Domain (z.B. dem Netzwerk einer Universität) gedacht. Sie werden deshalb auch "Interior Gateway Protocols" (IGPs) genannt. Interdomain Routing (z.B. im Internet) wird im Kapitel "Das Globale Internet" behandelt.

Routing als Graph-Problem

Routing-Verfahren lassen sich gut beschreiben, wenn die Netzwerke als Graphen betrachtet werden. Abb. ROU–1 zeigt ein Beispiel hierfür.

- Die Knoten des Graphen können dabei sowohl Rechner, als auch Router, oder ganze Netzwerke darstellen.
  - Dabei ist man natürlich daran interessiert, Routing auf der Basis von Netzwerken durchzuführen.
  - Zur Vereinfachung der folgenden Diskussion wird jedoch davon ausgegangen, daß die Knoten einzelne Router repräsentieren. Wie die Kombination der Information über Router und Netzwerke geschehen kann, wird am Beispiel der konkreten Routing-Verfahren (RIP und OSPF) gezeigt.
- Die Kanten (Links) sind mit Kosten versehen, die für die benötigten Übertragungszeiten stehen.
  - In realen Netzwerken sind die Link-Kosten asymmetrisch – eine Übertragung von A nach B kan schneller oder langsamer sein als in umgekehrter Richtung. Ebenfalls zur Vereinfachung der Diskussion wird im Folgenden von symmetrischen Links ausgegangen.

Das **Routing-Problem** kann dann wie folgt definiert werden:

Finde im Graphen die Pfade mit den jeweils geringsten Kosten zwischen allen Knoten-Paaren.

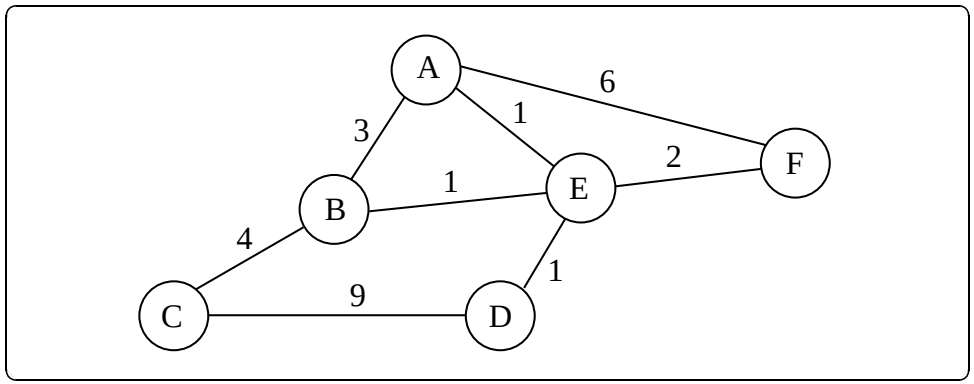

Abb. ROU–1 Ein Netzwerk repräsentiert als Graph

Statisches und dynamisches Routing

- Bei **statischem Routing** werden die Pfade manuell bestimmt und in Tabellen (z.B. Dateien oder nicht flüchtigem Speicher) eingetragen.

  - Dabei kann dann nicht reagiert werden auf Netzwerk-Änderungen wie:
    - ◇ Fehler von Knoten oder Links
    - ◇ Neue Knoten oder Links
    - ◇ dynamisch veränderte Link-Kosten durch hohe/niedrige Auslastung
  - Statisches Routing wird meist in Rechnern eingesetzt, die über ein einziges Netzwerk-Interface verfügen und darüber ihren Default-Router erreichen.
- Bei **dynamischem Routing** werden verteilte Algorithmen angewendet, mit deren Hilfe die Router Informationen zum Tabellen-Aufbau austauschen.
  - Dadurch kann automatisch auf Veränderungen im Netzwerk reagiert werden.
    ⇒ Skalierbarkeit

Distance Vector Routing (RIP)

Die Idee hinter dem Distance-Vector Algorithmus drückt sich im Namen des Verfahrens aus: Jeder Knoten konstruiert ein eindimensionales Feld (einen Vektor) der "Distanzen" (Link-Kosten) zu allen anderen Knoten. Diese Vektoren werden an alle direkten Nachbarn verteilt. Die Grundannahme des Verfahrens ist, daß jeder Knoten die Kosten der Links zu seinen direkten Nachbarn kennt. Nicht vorhandene (bzw. defekte) Links werden durch unendliche Kosten (∞) bezeichnet. Der andere bekannte Name dieses Verfahrens ist Bellman-Ford, nach den beiden Entwicklern.

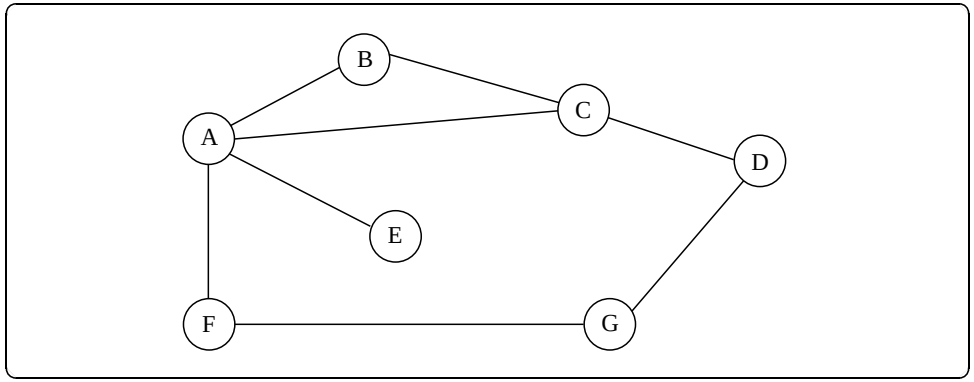

Abb. ROU–2 Beispiel-Netzwerk für Distance Vector Routing

Die Funktionsweise des Distance-Vector Routings läßt sich am besten an Hand eines Beispiel-Netzwerks wie in Abb. ROU–2 erklären. Hier wird zur Vereinfachung angenommen, daß alle Links die gleichen Kosten haben. Somit werden alle Kosten mit dem Wert "1" angenommen; die Distanz-Vektoren zählen somit nur die Links zwischen jeweils zwei Knoten.

	Distanz zu Knoten						
Information bei	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Initiale Distanz-Vektoren zu Abb. ROU–2

Die obige Tabelle zeigt die initialen Distanz-Vektoren, wenn jeder Knoten ausschließlich seine direkten Nachbarn kennt. Dabei ist zu beachten, daß keiner der Knoten über die in der Tabelle konzentrierte, globale Information verfügt! Jeder Knoten kennt einzig seinen eigenen Distanz-Vektor, also jeweils eine Zeile der Tabelle.

Ziel	Kosten	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Initiale Routing-Tabelle bei Knoten A zu Abb. ROU–2

Diese Tabelle zeigt die initiale Routing-Tabelle bei Knoten A. Die Knoten D und G sind somit (noch) nicht erreichbar.

Nach erfolgter Initialisierung senden die Knoten ihre Distanz-Vektoren jeweils zu allen direkten Nachbarn. Ein Knoten, der einen Vektor von einem Nachbarn erhält, überprüft dann für alle Knoten in seinem eigenen Distanz-Vektor, ob jene Knoten über den Nachbarn mit niedrigeren Kosten zu erreichen sind, als über die bis dahin besten, bekannten Routen. Wenn ja, dann werden die entsprechenden Einträge im Distanz-Vektor (und in der Routing-Tabelle) geändert. Dabei ist eine neue Route über den Nachbarn dann besser, wenn die Kosten zum Nachbarn plus dessen Kosten zum Zielknoten geringer sind als die bisherigen Kosten zum Zielknoten. Wenn sich das Netzwerk nicht verändert, dann konvergieren die Distanz-Vektoren (und Routing-Tabellen) nach einigen Austauch-Zyklen.

	Distanz zu Knoten						
Information bei	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Konvergierte Distanz-Vektoren zu Abb. ROU–2

Ziel	Kosten	NextHop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Konvergierte Routing-Tabelle bei Knoten A zu Abb. ROU–2

Übung: Erstellen Sie Routing-Tabellen für die Knoten B und G für den Zustand konvergierter Distanz-Vektoren! Frage: Gibt es jeweils genau eine mögliche Tabelle? Wovon hängt das ab?

Schließlich sollte noch bestimmt werden, *wann* die einzelnen Knoten jeweils ihre Distanz-Vektoren versenden. Einerseits sollte die Information überall relativ aktuell sein, andererseits sollte aber das Netzwerk nicht durch die Routing-Nachrichten selbst belastet werden. Somit gilt es hier eine Abwägung zu treffen. Die Knoten versenden ihre Distanz-Vektoren wenn eine der beiden Bedingungen erfüllt ist.

- Zyklisch nach Ablauf eines Zeitintervalls (z.B. in Sekunden- oder Minuten-Intervallen). Diese Information dient auch als Meldung, daß der jeweils sendende Knoten noch immer erreichbar ist.
- Sofort nachdem eine Änderung des eigenen Distanz-Vektors eingetreten ist. Somit werden Änderungen (wie z.B. Link-Ausfälle) direkt verbreitet.

Beispiel: Ausfall des Links zwischen F und G

Angenommen, die Verbindung zwischen F und G fällt aus und beide Knoten erkennen dies (in etwa) gleichzeitig. Dies führt dazu, daß ihre Distanz-Vektoren geändert werden, indem jeweils der Partner die Entfernung ∞ bekommt. Da sich die Distanz-Vektoren damit ändern, leiten F und G diese an ihre (verbleibenden) Nachbarn weiter. Da auch diese nun teilweise Pfade zu anderen Knoten verlieren (bzw. die Distanz ∞ wird), verbreitet sich die Information über den Link-Ausfall, bis alle Pfade entfernt sind, die bislang die Strecke F—G verwendet haben. Dies führt zu den folgenden Distanz-Vektoren, wobei davon ausgegangen wird, daß in der oben gezeigten Tabelle der Link F—G nur verwendet wurde, wenn er kürzere Wege ermöglicht hat. (also nicht bei gleich langen Alternativen)

	Distanz zu Knoten						
Information bei	A	B	C	D	E	F	G
A	0	1	1	2	1	1	∞
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	∞	1
E	1	2	2	3	0	2	∞
F	1	2	2	∞	2	0	∞
G	∞	3	2	1	∞	∞	0

Konvergierte Distanz-Vektoren nach Ausfall des Links F—G

Die obige Tabelle zeigt den Zustand, nach dem die Information über den Ausfall des Links verbreitet wurde. Nach Ablauf des nächsten Zeitintervalls wird dann die Information über alternative Routen im Netzwerk propagiert.

Übung: Vervollständigen Sie die folgende Tabelle, in dem Sie die Distanzen der neuen Pfade (die Umgehungen der Strecke F—G) eintragen!

	Distanz zu Knoten						
Information bei	A	B	C	D	E	F	G
A	0	1	1	2	1	1	
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3		1
E	1	2	2	3	0	2	
F	1	2	2		2	0	
G		3	2	1			0

Konvergierte Distanz-Vektoren mit Umgehungen der Strecke F—G (Übung)

Count-to-Inifinity Problem

Leider konvergieren die Distanz-Vektoren nicht immer so wie im Beispiel des Ausfalls der Strecke F—G. Bedingt durch die Tatsache daß die Knoten lediglich lokale Information besitzen, kann es zu einer sog. *Race Condition* kommen, also einem Verhalten das von der zeitlichen Reihenfolge abhängig ist, in der Nachrichten empfangen werden. Das Problem tritt auf, wenn eine bisher verwendete Strecke ausfällt, ein Zyklus im Graphen exisitert, und die Distanz-Vektoren in einer unglücklichen Reihenfolge empfangen werden.

Im Beispiel von Abb. ROU–2 kann das Count-to-Inifinity Problem z.B. dann auftreten, wenn A den Ausfall der Strecke A—E erkennt. A's Distanz-Vektor ändert sich somit; A sendet diesen an B. Nun läuft bei C das Zeitintervall ab und C sendet ebenfalls an B. Dies führt bei B zu einer Änderung, die B an A weitermeldet. A's Vektor ändert sich dabei, was an C gemeldet wird. Und nun meldet C wieder an B usw. usw. Dabei wird die Distanz zu E jedesmal um 1 hochgezählt, bis in's Unendliche...

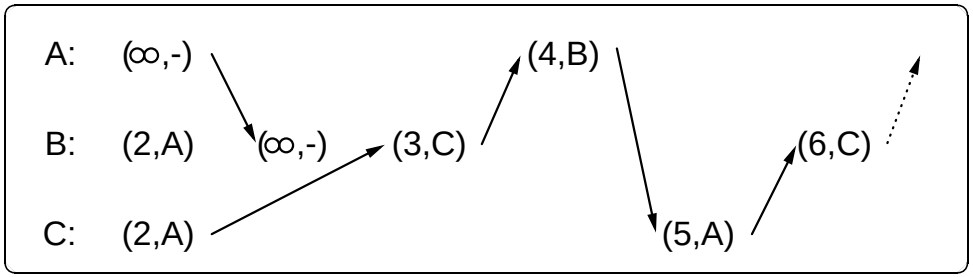

Abb. ROU–3 Count-to-Infinity Problem
gezeigt ist die Information über den Knoten E,
jeweils bei A, B und C
Paare: (Distanz, Next Hop)

- Lösungsansätze für das Count-to-Inifnity Problem:
  - Begrenzung des numerischen Werts für ∞, z.B. auf 16
  - *Split-Horizon* Ansatz:
    Sende Route nicht an "next hop" zurück.
    Beispiel: B hat (E,2,A), sendet (E,2) nicht an A
  - *Split-Horizon with Poison Reverse*:
    Sende Route an "next hop" zurück, aber mit unendlicher Distanz.
    Beispiel: B hat (E,2,A), sendet (E,∞) an A

Keine dieser Lösungsansätze kann das Problem in jedem Fall beheben. Split-Horizon funktioniert nur, wenn der Zyklus aus genau zwei Knoten besteht, also z.B. schon nicht im Beispiel der Abb. ROU–3. Die numerische Begrenzung des Wertes für "unendlich" kann bestenfalls als Schadensbegrenzung angesehen werden.

Routing Information Protocol (RIP)

Das Routing Information Protocol (RIP) ist eine Implementierung des Distance Vector Algorithmus, wobei Link-Kosten immer den Wert 1 haben; "Unendlich" wird auf 16 begrenzt; weiterhin wird die Split-Horizon Erweiterung verwendet.

RIP hat weite Verbreitung gefunden als Bestandteil von BSD Unix. Die Annahme $\infty = 16$ spiegelt die Zielsetzung des Routing-Verfahrens auf begrenzte Routing-Domains (z.B. innerhalb eine Uni-Campus) ab. Abb. ROU–4 zeigt ein Beispiel-Netzwerk für RIP. Die Abbildung zeigt die Router A bis D, die die Netzwerke 1 bis 6 verbinden. Die ausgetauschten Distanz-Vektoren enthalten die Distanzen zu den Netzwerken, wobei als *next hop* jeweils einer der Router verwendet wird.

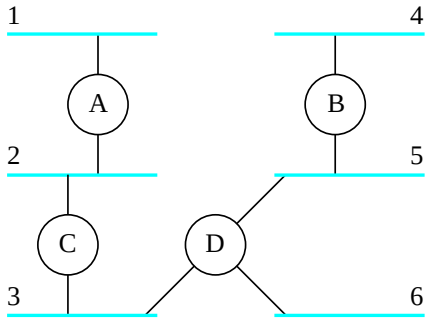

Abb. ROU–4 Beispiel-Netzwerk für RIP

Link State Routing (OSPF)

Trotz seiner weiten Verbreitung hat RIP zwei Nachteile. Zum einen werden die Link-Kosten immer auf den Wert 1 gesetzt (obwohl Distance Vector Routing auch mit beliebigen Werten funktioniert). Zum anderen bleibt das Count-to-Infinity Problem, bedingt durch das Fehlen von globaler Information bei den Routern.

Das als Link State Routing bekannte Verfahren behebt diese Probleme. Es wurde im Open Shortest Path First (OSPF) Protokoll implementiert und damit weit verbreitet.

Beim Link State Routing wird die Kosten-Information der jeweils an einem Router vorhandenen Links an alle Router im Netzwerk weitergeleitet. Dadurch erhalten alle Router Information über alle Links im Netzwerk. Auf dieser Grundlage kann jeder Router die jeweils kürzesten Pfade zu allen anderen Routern bestimmen.

Link State Routing besteht aus den folgenden fünf Schritten:

- Kennenlernen der direkten Nachbarn
  - Broadcast von *Hallo* Nachrichten, die von anderen Routern beantwortet werden müssen
- Messung der Delays zu den direkten Nachbarn
  - Es wird die Zeit gemessen, die vom Senden der *Hallo* Nachricht bis zum Empfang der Antwort vergeht. Die Hälfte dieser Zeit wird als Delay zum Nachbarn angenommen.
  - Messungen werden mehrmals durchgeführt und gemittelt.
- Erzeugen von *Link State Packets* mit folgendem Inhalt
  - ID des erzeugenden Routers
  - Liste der direkten Nachbarn mit zugehörigen Link-Kosten
  - Sequenz-Nummer
    - ◇ Nur Packete eines Routers mit größerer Sequenz-Nummer als in bisher vom selben Router empfangenen Packeten werden beachtet.
  - Time-to-Live (TTL)
    - ◇ Wenn ein Link-State-Packet TTL=0 erreicht hat, bevor es durch ein neues Packet vom selben Router abgelöst wurde, wird dieses Packet verworfen.
- Versenden der Link-State-Packete an alle Router
  - implementiert durch *Reliable Flooding* – siehe Abb. ROU–5
    - ◇ Bei Flooding leitet jeder Knoten ein ankommendes Packet auf allen Links weiter, mit Ausnahme desjenigen Links, auf dem das Packet empfangen wurde.
    - ◇ Bei Reliable Flooding werden alle Nachrichten mit Acknowledgement Packeten beantwortet und ggf. wiederholt.

(vgl. Kapitel *Sicherung der Übertragung*)

- ◇ Die Sequenznummern verhindern dabei die unendliche Wiederholung der Packete in Situationen wie in Abb. ROU–5(c), wenn Zyklen im Graphen entstehen.
- Das Versenden erfolgt zyklisch (in Stunden-Intervallen) sowie sofort bei Topologie-Änderungen (Link-Fehler).
  - ◇ Die Intervalle können wesentlich größer sein als bei Distance Vector Routing, da hier die Information über alternative Pfade ständig verfügbar ist. (Konsequenz der global vorliegenden Information)

- Lokale Bestimmung der kürzesten Pfade zu allen Knoten
  - Dijkstra's Algorithmus (Shortest Path)

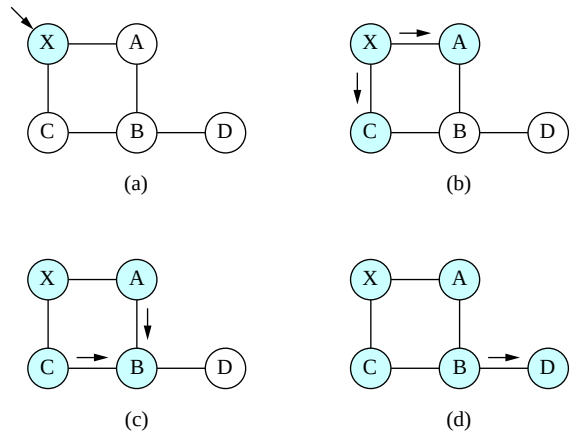

Abb. ROU–5 Flooding von Link-State Packeten

Dijkstra's Algorithmus (Shortest Path)

Gegeben sei die Knoten-Menge N, die Link-Kosten zwischen zwei Knoten i und j: $l(i,j)$, sowie der Start-Knoten s, der die kürzesten Wege zu allen anderen Knoten bestimmen will. Gesucht die die Pfad-Kosten $C(n)$ zu allen Knoten $n \in N - \{s\}$.

```
M = {s}
for each n in N – {s}
    C(n) = l(s,n)
while (N ≠ M)
    M = M ∪ {w} such that C(w) is minimum for all w in (N–M)
    for each n in (N–M)
        C(n) = min(C(n),C(w)+l(w,n))
```

Übung: Wenden Sie Dijkstra's Algorithmus auf das Beispiel-Netzwerk aus Abb. ROU–6 an und stellen Sie die Routing-Tabelle von Knoten A auf!

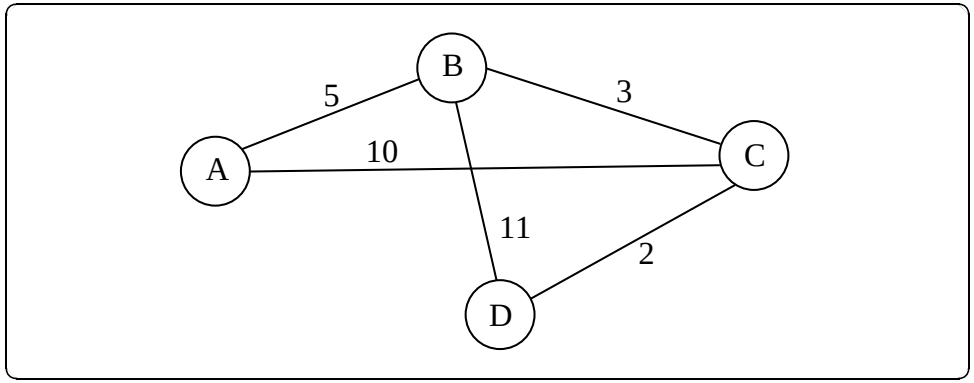

Abb. ROU–6 Beispiel-Netzwerk für Link State Routing

Open Shortest Path First (OSPF)

OSPF wurde unter Leitung der IETF (Internet Engineering Task Force) entwickelt. Das Attribut *Open* weist dabei darauf hin, daß es sich hier um einen nicht-proprietären Standard handelt (nicht an einen Hersteller gebunden). *Shortest Path First* beschreibt die Vorgehsweise in Dijkstra's Algorithmus.

OSPF implementiert Link State Routing mit den folgenden Erweiterungen:

- Authentisierung von Routing Nachrichten
  - Kontrolle z.B. gegenüber fehlerhaft arbeitenden Routern
  - Zusätzliche Hierarchie-Ebene, Routing kann zu Gruppen von Netzen erfolgen
    ⇒ Skalierbarkeit
  - Load Balancing
    - Mehrere Routen zum gleichen Ziel werden parallel verwendet. Zu versendende Nachrichten werden entsprechend der Kapazitäten auf die Routen verteilt.

Link Metrics

Bei der bisherigen Diskussion wurde davon ausgegangen, daß die einzelnen Link-Kosten vorgegeben waren, oder (bei Link State Routing) durch Messungen bestimmt wurden. Im Folgenden soll die Tauglichkeit der verschiedenen Metriken (Kosten-Bestimmungen) für Netzwerk-Links diskutiert werden.

RIP

Hier werden alle Link-Kosten gleich eins gesetzt. Dieser Ansatz eignet sich nur bedingt, z.B. in lokalen (Campus-) Netzwerken, wenn alle Leitungen in etwa gleich sind, oder nur wenig ausgelastet werden. Diese (triviale) Metrik ignoriert die Faktoren Latenz, Bandbreite, sowie die Auslastung der Links.

Erste ARPANET-Metrik

Im ARPANET (dem Vorgänger des heutigen Internet) wurde zuerst eine Metrik verwendet, bei der die Links über die Länge der jeweiligen Warteschlange in den Routern bewertet werden. Dem liegt die Annahme zu Grunde, daß lange Warteschlangen auch lange Zustellungszeiträume bedeuten. Peterson/Davie vergleichen diese Metrik mit dem Versuch, im Supermarkt an den Kassen schneller voran zu kommen, in dem man jeweils zur Kasse mit der kürzesten Schlange wechselt, ohne auf andere Faktoren (wie z.B. die Anzahl der Artikel in der Einkaufswagen der Leute) zu achten.

Da diese Metrik relativ künstlich war und Latenz und Bandbreite der Links ignorierte, wurde sie durch die folgende Metrik abgelöst.

Zweite ARPANET-Metrik

Hier werden die Links über das Delay der versendeten Packete bewertet.

Delay = (DepartTime − ArrivalTime) + TransmissionTime + Latency

Alle Packete werden beim Empfang mit einem Timestamp (ArrivalTime) versehen. Beim Weiterleiten wird die DepartTime aufgezeichnet. Nach Erhalt des Acknowledgements (auf der Link-Ebene) wird dann das Delay bestimmt. Bei Packet-Verlusten (und wiederholter Aussendung) wird die DepartTime neu gesetzt; somit werden Links mit vielen Verlusten automatisch schlechter bewertet. TransmissionTime und Latency repräsentieren Bandbreite und Latenz des Links.

Diese Metrik funktionert recht gut, solange die Links nicht stark belastet werden. Bei starker Belastung entsteht jedoch der Effekt des “Oszillierens” zwischen mehreren Alternativen zum gleichen Ziel: Sobald ein Link stark ausgelastet wird, wird er schlecht bewertet und der Verkehr wird auf andere Links umgeleitet. Als Konsequenz wird der zuerst belastete Link nicht mehr benutzt und dafür die Alternativen überlastet. Im nächsten Schritt wechseln wiederum die Metriken und der erste Link wird wieder beansprucht, während die Alternativen ungenutzt bleiben.

Das wesentliche Problem der zweiten ARPANET-Metrik liegt in den Wertebereichen der einzelnen Faktoren; einerseits sind die durch Auslastung bedingten Schwankungen zu groß, andererseits sind die Parameter nicht aufeinander abgestimmt.

Revised ARPANET Routing Metric

Unter diesem Namen wurde die folgende Weiterentwicklung der zweiten ARPANET Metrik bekannt. Dabei werden die einzelnen Faktoren zur Bestimmung des Delays durch eine Vielzahl von Parametern aufeinander abgestimmt. Diese Parameter müssen jedoch sorgfältig (und manuell) bestimmt, ausgetestet, und ggf. angepaßt werden. Folgende Eigenschaften bestimmen die Revised ARPANET Routing Metric:

- Ein ausgelasteter Link hat niemals mehr als die dreifachen Kosten ohne Auslastung.
- Der langsamste Link hat höchstens sieben mal die Kosten des schnellsten Links.
- Bandbreite ist wichtiger als Latenz.
  - (Router sollen vor allem hohen Durchsatz erzielen.)
- Die Parameter hängen nur bei hoher Last von der Auslastung ab.

Abb. ROU–7 zeigt, wie vier verschiedene Links (mit hoher und niedriger Bandbreite sowie hoher und niedriger Latenz) abhängig von ihrer Auslastung bewertet werden.

Zusätzlich zur Kompression der Wertebereiche werden Änderungen der Bewertung eines Links abgefedert, in dem der aktuell bestimmte Wert und der letzte Wert miteinander gemittelt werden.

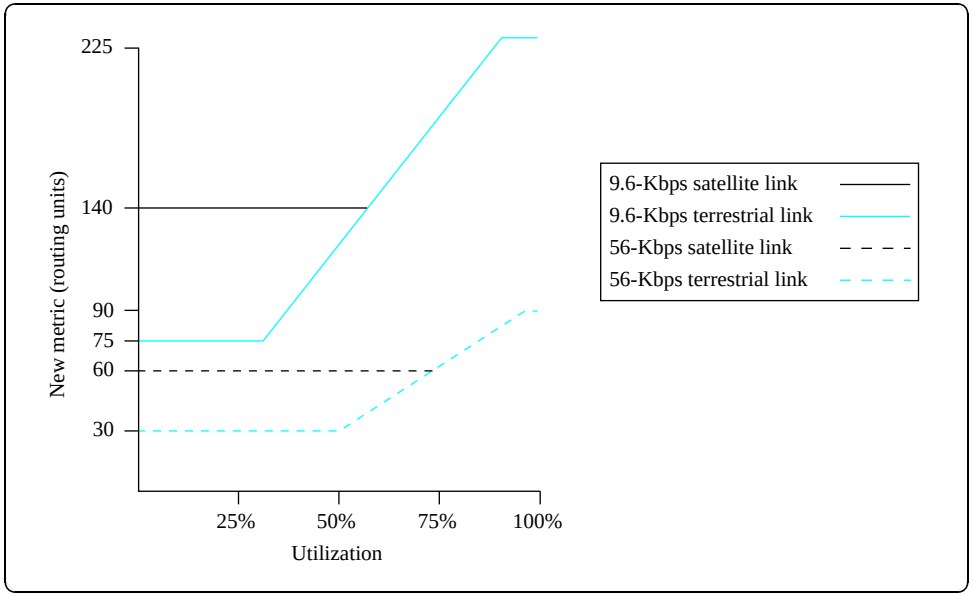

Abb. ROU–7 Revised ARPANET Routing Metric

Dadurch werden Änderungen langsam weitergeleitet, wodurch Oszillieren verhindert und kurzfristige Lastspitzen ausgeblendet werden.

Routing für mobile Rechner

Das Routing von IP-Packeten basiert auf den Netzwerk-Nummern. Somit wird ein einzelner Rechner erreicht, in dem das Netzwerk erreicht wird, in dem er angeschlossen ist. Ein mobiler Rechner (z.B. ein Laptop mit drahtlosem Ethernet) kann z.B. über DHCP dynamisch neue IP-Adressen bekommen.

Allerdings können über die Veränderung der IP-Adresse hinweg bereits bestehende Verbindungen nicht weiterlaufen. Der als *Mobile IP* bekannte Standard erlaubt dies jedoch unter der zusätzlichen wichtigen Randbedingung, daß weder die Kommunikationspartner noch bestehende Router verändert werden müssen. (Die Mobilität eines Rechners ist somit transparent für seine Kommunikationspartner.)

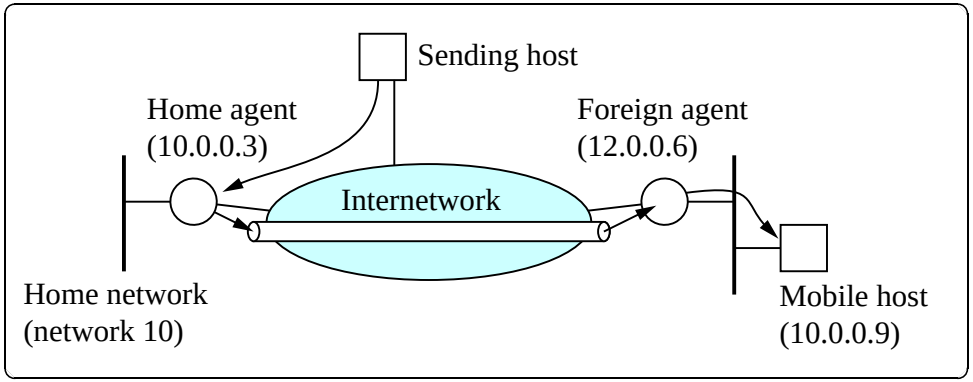

Abb. ROU–8 Mobiler Rechner und Mobility Agenten

Abb. ROU–8 zeigt den Ansatz von Mobile IP. Jeder mobile Rechner hat dabei ein *Home Network*, in dem er eine permanente IP-Adresse hat. Wenn der mobile Rechner sich in einem anderen Netzwerk aufhält, dann arbeitet ein *Home Agent* anstelle des mobilen Rechners, empfängt Packete und leitet sie entsprechend weiter. Der mobile Rechner bekommt in seinem temporären Netzwerk eine neue IP-Adresse zugeteilt. Im temporären Netzwerk arbeitet ein *Foreign Agent*, der mit dem Home Agent kommuniziert und Packete an den mobilen Rechner zustellt. Der Foreign Agent kann dabei auch als Bestandteil des mobilen Rechners ablaufen. Dies vereinfacht die Kommunikation.

Um dies zu realisieren, müssen die folgenden Probleme gelöst werden:

- Der Home Agent muß ankommende Packete bekommen. Er “übernimmt” dazu temporär die IP-Adresse und reagiert entsprechend auf ARP-Packete.
- Zwischen Home Agent und Foreign Agent wird ein IP-Tunnel aufgebaut, in dem die Packete zum mobilen Rechner geleitet werden.

- Der Foreign Agent muß Packete beim mobilen Rechner zustellen. Wenn beide Komponenten getrennt sind, muß direkt die MAC-Adresse verwendet werden. Ist der Foreign Agent Bestandteil des mobilen Rechners, dann können Packete direkt durch Kommunikation zwischen Prozessen zugestellt werden.

Triangle Routing Problem

Wenn sich ein mobiler Rechner relativ nahe bei seinem Kommunikationspartner aufhält, dann entsteht das Triangle Routing Problem: Die beiden Rechner kommunizieren miteinander über den Home Agent, der ggf. sehr weit entfernt ist. Die Konsequenz ist eine sehr langsame Kommunikation, im Vergleich zur direkten, lokalen Kommunikation.

Um dieses Problem zu beheben, muß jedoch der Kommunikationspartner verändert werden (Mobile IP implementieren). In diesem Fall kann der Kommunikationspartner direkt einen IP-Tunnel zum Foreign Agent aufbauen.