1 Routing

Routing optimiert den Weg zum Ziel einzelner Pakete. Dabei gewährt es Dienstgüten (z.B. Konstante Latenz, Bitrate, Durchsatz, sowie Priorisierung von Paketen) und minimiert Kosten.

1.1 Begriffe

Router - Gerät, welches mindestens zwei unterschiedliche Netzwerke verbindet. Ist in der Lage, den besten Weg für eine Nachricht durch ein Netzwerksystem bis zum Empfänger zu bestimmen.

Routing - bezeichnet das Transportieren von Daten innerhalb eines Netzes

Passives/Source Routing - Route ist durch das zu versendende Datenpaket vorgegeben (z.B. im Header enthalten)

Aktives Routing - Router kann sich den Versandweg selbst aussuchen

Globales Routing - Ein Knoten hat vollständige Informationen zu Topologie und Kosten. Informationen über das komplette Netz sind vorhanden. Optimierung läuft an einer Stelle oder repliziert an mehreren.

Dezentrales Routing - Keine vollständigen Informationen in einem Knoten. Jeder Knoten kennt zunächst (initial) nur die Kosten verbundener Links. Berechnung erfolgt iterativ und dezentral.

Distanz-Vektor-Algorithmen - Kein Knoten hat Wissen über das komplette Netz und kennt daher nie die komplette Route von einer Quelle zu seiner Senke.

Link-State-Algorithmen - Knoten haben Wissen über die gesamte Netztopologie. Daher haben im stabilen Zustand alle Knoten die gleiche Sicht auf das Netz.

Statisches/nichtadaptives Routing - Ist beim Starten/Hochfahren des Systems fixiert.

Dynamisches/adaptives Routing - Routen ändern sich im laufenden Betrieb und passen sich so an Änderungen von Topologie, Netzlast, Kosten etc. an. Setzt zwingend ein Protokoll voraus.

1.2 Anforderungen

Die Anforderungen an Routing-Algorithmen sind:

Einfachheit - Algorithmus muss in kurzer Zeit mit wenig Ressourcen berechnet werden können.

Robustheit - Bewältigung von Änderungen der Topologie.

Stabilität - Erreichung eines stabilen Zustandes.

Fairness - Alle Knoten werden gleich bedient.

Optimalität - Gemäß der Kostenfunktion günstigster Weg wird gefunden.

Anmerkung: Ohne weiteres können sich diese Anforderungen widersprechen! So kann etwa die Kommunikation der Knoten $x \to x'$ die Kommunikation $a \to a', b \to b' undc \to c'$ stark negativ beeinflussen und damit den Gesamtdurchsatz (Optimalität) verringern.

1.3 Spannbäume

Optimalitätsprinzip: *Wenn ein Router J auf dem optimalen Pfad von Router I zu Router K liegt, dann gehört der optimale Weg von J nach K zum gleichen Pfad*[**?**]. So lässt sich eine allgemeine Aussage über optimale Pfade treffen, ohne dass konkrete Kenntnisse der Topologie des Netzwerkes vonnöten sind. Alle optimalen Routen von den Knoten zu einem bestimmten Ziel (Senke), bilden einen **Spannbaum**[1].

- Senke muss nicht zwangsläufig eindeutig sein (auf den gleichen Pfadstrecken kann es auch andere Bäume geben).
- Da Senke ein Baum ist, enthält sie keine Schleifen.
- Pakete werden auf einer endlichen, begrenzten Anzahl von Teilstrecken übertragen.

[1] Im Tanenbaum[**?**], welcher für die Entstehung des Skripts offensichtlich Modell gestanden hat, wird dieser als **Quelle-Senke-Baum** (sink tree) bezeichnet

1.4 Algorithmen

Eingabe: Graph G des Netzes, wobei jeder Knoten im Graphen einen Router und jede Kante eine Übertragungsleitung darstellt.
Problemstellung: Finde für einen Startknoten s und einen Endknoten e eines gewichteten Graphen $G = (V, E)$ und der Kostenfunktion $k(E)$, einen Weg zwischen s und e mit minimalen Kosten bezüglich k.

1.4.1 Dijkstra-Algorithmus und Link-State-Routing

Dijkstra arbeitet nur für nichtnegative Kantengewichte korrekt. Die Zeitkomplexität[2] beträgt $\mathcal{O}(n^2)$, bei der Verwendung eines Fibbonacci-Heaps kann diese auf $\mathcal{O}(m + n \log n)$ verbessert werden.

Dieser Algorithmus wird beim *Link-State-Routing* verwendet. Zur Durchführung werden komplette (Topologie- und) Kosteninformationen benötigt. Dazu propagieren die Knoten die Kosten zu ihren Nachbarn mittels Broad-/und Multicasts im gesamten Netzwerk.

Beschreibung Dijkstra-Algorithmus nach [**?**]:

$d(v_1) \leftarrow 0$
for j := 2 to n **do** $d(v_j) \leftarrow \infty$
end for
$OFFEN \leftarrow V$
while $OFFEN \neq \emptyset$ **do**
  wähle ein $v \in OFFEN$ mit $d(v) = min\{d(w) : w \in OFFEN\})$
  $OFFEN \leftarrow OFFEN \backslash \{v\}$
  for all $w \in OFFEN$ mit $(v, w \in E)$ **do** $d(w) \leftarrow min\{(d(w), d(v) + k(v, w))\}$
  end for
end while

1.4.2 Oszillation

Oszillation ist ein Problem, welches bei zu häufiger Ausführung und der Berücksichtigung der Netzauslastung entsteht. Dabei schaltet der Verkehr ständig zwischen verschiedenen alternativen Pfaden hin und her. Grund hierfür ist die Reaktion auf die Auslastung der einzelnen Verbindungen: Wird der Verkehr über Alternative A geleitet und Alternative B hat eine geringere Auslastung, wird der Pfad entsprechend geändert. Dadurch ist die Auslastung von Alternative A nun geringer und es findet wieder ein Wechsel statt.
Diesem Phänomen lässt sich entgegenwirken, indem man a) die Netzauslastung nicht zur Berechnung der Routen mit einbezieht, wobei dies die Verwendung von Alternativrouten bei Lastspitzen verhindert, b) schnelle Änderungen der Routen verbietet (Hold-down), c) die durchschnittliche Auslastung (oder anderweitig aggregierte Werte) heranzieht[**?**] oder d) verhindern, dass Router den Algorithmus gleichzeitig durchführen, z.B. durch zufällige Wartezeiten.

1.5 Bellmann-Ford-Algorithmus und Distance-Vector-Routing

Beim Distance-Vector-Routing verwaltet jeder Knoten eine Routingtabelle mit Informationen zu bekannten Zielen. Zu jedem Ziel wird der zu wählende Ausgang[3] mit den damit verbundenen Kosten gespeichert. Es wird jeweils angenommen, dass es sich dabei um die beste (= günstigste) bekannte Weiterleitung handelt. Die Informationen werden an die direkten Nachbarn weitergeleitet. Nachbarn können so neue Ziele kennenlernen oder Routen zu bekannten Zielen verbessern. Errechnen lassen sich die Kosten zum Ziel als Summe der Kosten zum Erreichen des Nachbarn und dessen Kosten zum Erreichen des Ziels.
Distance-Vector-Algorithmen sind **verteilt**, **iterativ** und **asynchron**. Bei **Änderung von Kosten** läuft der Algorithmus neu an, bis er sich auf allen Knoten stabilisiert hat. Haben sich die Kosten verbessert, findet eine schnelle Adaption (**Good news travels fast**) statt, bei Verschlechterungen ist die Reaktionszeit relativ hoch ((**Bad news travels slow**)).

Beschreibung Bellmann-Ford-Algorithmus nach [**?**]:

$d(v_1) \leftarrow 0$
for j := 2 to n **do** $d(v_j) \leftarrow \infty$
end for
for $i \leftarrow 1$ to $|V| - 1$ **do**
  for all $(u, v) \in E$ **do**
    $d(v) \leftarrow min(d(v), d(u) + k(u, v))$
  end for
end for

[2] Abhängig von Kantenzahl m und Knotenzahl n
[3] Interface, Next Hop, whatever

for all $(u, v) \in E$ **do**
  if $d(v) > d(u) + k(u, v)$ **then return** FALSE
  end if
end for

1.5.1 Count-to-infinity-Problem

Hinweis: Zur Vereinfachung wird in diesem Abschnitt angenommen, dass als Metrik die Anzahl der Hops zum Ziel gewählt wird.
In bestimmten Fällen kann es bei Distance-Vector-Algorithmen dazu kommen, dass das Verfahren nicht konvergiert. Dies ist beispielsweise dann der Fall, wenn ein Router (oder jede Verbindung zu ihm) ausfällt. Die verbleibenden Router sind vom Ausfall nicht informiert, stattdessen erhalten sie weiterhin von ihren Nachbarn veraltete Informationen zur Entfernung.
Der vormals direkte Nachbar, welcher zuvor von einer Entfernung von 1 kannte, erhält nun von einem anderen Nachbarn die Information, dass dieser den Ausgefallenen mit Kosten von 2 erreichen kann. Dies führt dazu, dass die fehlerhafte Information übernommen wird und sich die Kosten — zumindest theoretisch — bis unendlich *hochschaukeln*.
Gegenmaßnahmen sind a) **Split Horizon** - hierbei werden Updates nie an den Nachbarn gesendet, von dem die beste Route gelernt wurde. Stattdessen gibt es Updates erst nach einem Timeout. Variante b) **Split Horizon with Poisoned Reverse** funktioniert, wie folgt: Ein Router A glaubt, dass er über einen Router B zum Ziel gelangt. Dann teilt A B mit, dass er selbst keine Route zum Ziel kennt. Dies kostet allerdings unnötig Bandbreite und funktioniert wohl in der Praxis nicht ganz so gut. Kein Plan.

1.6 Routing in MANETs

Ein Mobile Ad Hoc Network (MANET) ist ein drahtloses, selbstkonfigurierendes Netzwerk mobiler Knoten. Zumindest einige dieser mobilen Knoten müssen auch als Router arbeiten können. Alle Knoten bewegen sich unkoordiniert und nur schlecht vorhersagbar. MANETs können Übergänge in andere Netze haben, z.B. in das Internet.

1.6.1 Herausforderungen und Probleme:

- Netzwerke ohne feste Infrastruktur.
- Knoten sind in der Regel mobil, daher können keine statischen Routen verwendet werden.
- Zielfunktion des Routings enthält weitere Faktoren
  - Energieeffizienz
  - Durchsatz einzelner Strecken
  - Gesamtdurchsatz
  - QoS für bestimmte Dienste

Beim Routing in MANETs bestehen insbesondere Probleme bei der **Konvergenz** (Routen müssen gefunden werden, bevor Knoten wieder außer Reichweite sind) und der **Skalierbarkeit** (Overhead durch Routing).

1.6.2 Lösungsansätze

- Einfache Ansätze
  - Reaktiv (bei Bedarf), Suche nach dem Ziel
  - Proaktiv (auf Vorrat), Regelmäßige Verbreitung aller Pfade
  - Flooding
    - Nachricht wird in alle Richtungen verschickt
    - Bisheriger Weg wird gespeichert
    - Nachricht, die zuerst am Ziel ankommt enthält die günstigste Route
- Erweiterte Ansätze
  - Auswahl besonderer Knoten als Router
  - Geografisches Routing
  - Vorausberechnung der Knotenbewegung
  - Scalable Source Routing[**?**] (Nicht im Skript)

1.6.3 Backbone-Netze

Auswahl bestimmter Knoten als Router. Jeder Knoten hat genau einen Backbone-Nachbarn. Berechnung des Minimum Dominating Sets[4] ist schwieriges Problem (NP-vollständig). Jeder Knoten außerhalb des Backbones hat ein *default gateway* im Backbone. Routing im Backbone nach den üblichen Routing-Verfahren. Erhaltung des Backbones bei wegfallenden Verbindungen ist ebenso ein schwieriges Problem.

1.6.4 Proaktives und Reaktives Routing

	Vorteile	**Nachteile**
Proaktiv	Geringe Latenz durch Routing und Zustandsinformationen über alle Verbindungen liegen vor	Overhead und Reparatur von Routen hängt von der Aktualisierungshäufigkeit ab
Reaktiv	Kein Overhead für das Vorausberechnen der Routen	Latenz durch Routenberechnung

Geometric Routing

- Alle Knoten kennen ihre Position z.B. GPS bei mobilen Knoten
- Verschiedene Varianten
  - Wahl des Knotens, der am nächsten an der „Luftlinie“ liegt.
  - Wahl des Nachbarknotens, der möglichst nahe am Ziel liegt.
  - Wahl des Knotens, der in möglichst derselben Richtung wie das Ziel liegt.

1.7 Routing-Protokolle

Routing-Protokolle werden grob in **Interdomain-** (Interior Gateway Protocols, IGPs) und **Intradomain-**Routing-Protokolle (Exterior Gateway Protocols, EGPs) unterteilt. IGPs werden innerhalb eines Netzwerkes (*autonome Systeme*, z.B. innerhalb der Uni) genutzt. Um Nachrichten zwischen diesen Netzen, die jeweils unterschiedliche Routing-Protokolle (mit unterschiedlichen Metriken und Verfahren) zu Routen, werden EGPs benötigt.

1.7.1 RIP

Das **Routing Information Protocol** ist ein einfaches *Interior Gateway Protocol*. Es verwendet **Distance Vector Routing**, die einzige Metrik sind dabei Hops, die maximale Entfernung beträgt dabei 15.

Ein Router, der mit RIP arbeitet, verwaltet in seiner Routing-Tabelle die besten bekannten Distanzen zu allen bekannten Zielen. Initial sind alle direkten Nachbarrouter (mit einer Entfernung von 1) bekannt. Periodisch werden diese um Übermittlung der aktuellen Routing-Tabelle gebeten und die eigene Routing-Tabelle entsprechend auf den neuesten Stand gebracht. Dazu wird die Multicast-Adresse 224.0.0.9 (in v1 noch Broadcast) genutzt. Die Geräte lauschen auf UDP Port 520.

Weitere Regelungen (z.B. zur Vermeidung des Count-to-Infinity-Problems, Abschnitt 1.5.1):

- Alle 30s ein Response
- Im Laufe der Zeit würden sich alle Updates synchronisieren, daher Verwendung eines kleinen zufälliger Offset (maximal 5s)
- Timeout für Routen 180s (Distanz wird dann auf *unendlich* gesetzt).
- Garbage Collection nach 120s (solange werden Routen mit Distanz *unendlich* noch verbreitet).

RIPv1 verwendete noch kein CIDR(Classless Inter-Domain Routing), es wurden also keine Informationen über Subnetze versand, was zur Folge hatte, dass alle Netze dieselbe Größe haben mussten. Dies wurde erst mit RIPv2 hinzugefügt, ebenso die Möglichkeit der Authentifizierung, allerdings nur mit MD5-Hashes.

Eine RIPv1-Nachricht besteht aus einem **Command-Feld** (z.B. Request oder Response), der **Version**, 2 Bytes Nullen (in v2 stehen hier beispielsweise das Route Tag oder der Authentication Type) und 1 bis 25 **RIP Entries**. Jeder RIP Entry enthält einen **Adress family identifier** (?), die **Netzadresse**, die **Distanz**, sowie ganz viele, schwer erklärbare Füll-Nullen. Bei RIPv2 können gibt es außerdem RIP-Entries mit Subnetzmaske (enthalten zusätzlich die Subnetzmaske und den Next Hop; diese stehen in den in v1 mit Nullen aufgefüllten Feldern) und einen RIP-Entry mit Authentifizierung (enthält 16 Byte-Eintrag, also wohl eine MD5).

[4] Eine dominierende Menge eines Graphen ist eine Teilmenge D von V, sodass jedes $v \in V \setminus D$ einen Nachbarn in D hat.

1.7.2 OSPF

Open Shortest Path First ist ein *Interior Gateway Protocol*. Im Gegensatz zu RIP handelt es sich um ein **Link State Protocol**. Jeder Router berechnet aufgrund der ihm zur Verfügung stehenden Informationen selbstständig die kürzesten Wege zu den bekannten Zielen (Shortest Path Tree).

Die Link State Database (LSDB) wird periodisch zwischen allen Routern ausgetauscht (Multicast 224.0.0.5 und 224.0.0.6). Dabei werden unterschiedliche Nachrichtentypen zum Erkennen von Links und zum Austausch der LSDB benutzt. Netze werden zu Areas[5] zusammengefasst. Area 0 ist Backbone, sonstige Areas repräsentieren andere Teilbereiche. Ein durch OSPF verwaltetes autonomes System enthält typischerweise mehrere Areas. Das Backbone Area hat die ArealID 0.0.0.0 und ist mit allen anderen Areas verbunden. Informationen über erreichbare Netze werden als **Link State Advertisements** unter den Routern ausgetauscht.

Das **Hello-Protokoll** ist in Open Shortest Path First für den Netzwerkbetrieb ein integraler Bestandteil des gesamten Routingprozesses. Es ist verantwortlich für:

- Senden von Keepalives in bestimmten Intervallen (damit wird bestätigt, ob die Route noch besteht)
- Zur Entdeckung eines Nachbarn
- Aushandlung der Parameter
- Wahl eines Designated Routers (DR) und des Backup-DRs

Alle ankommenden Hello-Pakete werden auf die Area ID und anderen Parametern geprüft. Wenn diese mit den lokalen Einstellungen übereinstimmen, wird dieser Router als Nachbar eingetragen.≪[**?**]

1.7.3 BGP

Das Border Gateway Protocol (BGP) verwaltet eine Liste mit Prefixen (Netzwerken), um Routing zwischen Autonomen Systemen (AS) zu ermöglichen. BGP war die Voraussetzung für ein dezentrales Internet und den Wegfall eines zentralen Backbones.
BGP muss nicht direkt vom Router gesprochen werden, ein spezieller Host (Speaker) im AS kann die Aufgabe übernehmen. Falls mehrere „Speaker“ vorhanden sind, ist Synchronsierung notwendig.

BGP benutzt TCP als Transport Protokoll auf Port 179 und ist verbindungsorientiert und zustandsbehaftet.
BGP Pakete:
Bei der initialen Verbindung werden Routing-Tabellen ausgetauscht (sofern Export Policy dies erlaubt).

- Routing Information Bases (RIB) enthält Routen.
- Adj-RIB-Out Nachrichten werden versendet.

Später werden Routen über Updates aktualisiert. Ein UPDATE enthält Informationen über eine Route. Ansonsten sendet der BGP Speaker regelmäßig (default alle 60s) ein Keep-Alive-Paket.

Eine BGP UPDATE Nachricht dient zur Übermittlung einer nutzbaren Route oder Übermittlung von nicht länger nutzbaren Routen. Die Nachricht zeigt an, dass es in diesem AS eine Route zu einem bestimmten Ziel gibt. Das AS kann dieses Ziel entweder selbst enthalten oder als Transit anbieten.

1.7.4 OLSR

Das Optimized Link State Routing (OLSR) ist ein Routing-Protokoll für mobile Ad-Hoc-Netze, das eine an die Anforderungen eines mobilen drahtlosen LANs angepasste Version des Link State Routing darstellt. Es ist dezentral und vollständig verteilt. Da es ein proaktives Protokoll ist:

- Routen stehen bei Bedarf sofort zur Verfügung.
- Routen werden permanent aktualisiert.
- Topologieerkennung über Hello-Messages.
- Verbreitung der Topologie-Informationen über TC-Messages Topology Control).

Jeder Knoten lernt seine 1-Hop-Nachbarn über Hello-Messages kennen. Außerdem verbreitet jeder Knoten Informationen über seine Existenz und die seiner 1-Hop-Nachbarn über Hello-Messages. So kennt folglich jeder Knoten seine 1-Hop-Nachbarn und 2-Hop-Nachbarn.
Es werden spezielle Knoten (Multipoint-Relays - MPR) ausgewählt. Jeder Knoten wählt MPRs aus seinen

[5] Laut Wikipedia scheint das Areakonzept verdammt wichtig zu sein, wird in den Folien aber kaum erläutert.

1-Hop-Nachbarn aus, so dass er über diese jeden 2-Hop-Nachbarn erreichen kann. Die Wahl wird in den Hello-Messages geteilt. Lediglich diese MPRs leiten TC-Messages per Broadcast weiter. Auf diese Weise wird der Aufwand für Flooding reduziert und je weniger MPRs es gibt, desto geringer der Aufwand.

1.7.5 AODV

Der Ad-hoc On-demand Distance Vector-Routingalgorithmus (AODV) ist ein Algorithmus zum Weiterleiten von Daten durch ein mobiles Ad-hoc-Netz. Das Protokoll gehört zu den topologiebasierten, reaktiven Routing-verfahren, d.h. Routen zu bestimmten Zielen werden erst bei Bedarf ermittelt.