Projektpartner: Westfälische Wilhelms-Universität Münster - Institut für Geoinformatik
Datum: 03.02.2003

# Lerneinheit 7: „Netze und Protokolle“

## Inhalt

**Lerneinheit 7: „Netze und Protokolle“ ..... 1**
IPv6 – Internet Protocol Version 6 ..... 2
mobileIP ..... 2
Care-of-address ..... 3
Funktionsprinzip ..... 3
Agent Discovery ..... 4
Registrierung der care-of-address ..... 4
Authentifizierung ..... 5
Tunneling ..... 6
Reverse Tunneling ..... 6
MobileIP und IPv6 ..... 7
Routing in Ad Hoc Netzwerken ..... 7
Tabellenorientierte Routing-Protokolle ..... 7
Destination-Sequenced Distance-Vector Routing (DSDV) ..... 8
Clusterhead Gateway Switch Routing (CGSR) ..... 8
On-Demand Routing ..... 9
Ad Hoc On-Demand Distance Vector Routing (AODV) ..... 9
Dynamic Source Routing (DSR) ..... 10
Associativity-Based Routing (ABR) ..... 11
Literatur ..... 13

bmb+f Bundesministerium für Bildung und Forschung

Das Projekt Geoinformation – Neue Medien für die Einführung eines neuen Querschnittsfaches ist ein Projekt im Rahmen des BMBF-Programms Neue Medien in der Bildung.

zip Ein Projekt im Zukunftsinvestitionsprogramm der Bundesregierung

Das Projekt wird im Rahmen des Zukunftsinvestitionsprogramms der Bundesregierung (zip) gefördert.

## IPv6 – Internet Protocol Version 6

Das Internet in seiner derzeitigen Ausprägung lässt einige wichtige Aspekte für den Einsatz von mobilen Geräten vermissen. Das genutzte Protokoll IPv4 (Internet Protokoll Version 4) unterstützt beispielsweise nicht:

- Adressweiterleitung an mobile Geräte, die sich erneut mit dem Netzwerk verbinden.
- Mächtige Authentifizierungsmechanismen, die benötigt werden, um die weiterleitende Infrastruktur über die neue Position eines mobilen Gerätes zu informieren.
- Mobilen Geräten zu entscheiden, ob das Netzwerk, mit dem sie sich gerade verbinden, das gleiche ist wie das alte Netzwerk.
- Mobilen Geräten ihren Kommunikationspartnern mitzuteilen, dass sie eine neue Position eingenommen haben.

All diese Mängel werden vom neuen Protokoll Version 6 aufgegriffen. Die treibende Kraft in der Entwicklung dieses Protokolls waren im Übrigen nicht die zusätzlichen Anforderungen von mobilen Geräten, sondern die Tatsache, dass die Adressen im Internet langsam knapp werden. IPv4 nutzt 32 bit Adressen. Durch die teilweise ineffiziente Zuteilung dieser Adressen sind in manchen Ländern freie Adressen mittlerweile Mangelware. IPv6 nutzt 132 bit Adressrepräsentationen, womit sich mehrere Milliarden individueller Adressmöglichkeiten erzeugen lassen. Diese Zahl mag zunächst sehr hoch erscheinen, wenn man sich die Anzahl von Computern und Mobiltelefonen auf der Welt vor Augen hält. Berücksichtigt man jedoch die Tatsache, dass zukünftig jeder in irgendeiner Form mit dem Internet verbundene Chip eine eigene Adresse benötigt (also auch jeder in die Kleidung integrierte Schaltkreis oder der Chip im Buch, der es auf Zuruf aus dem Regal katapultiert…) so ist der gewählte Adressraum mit Sicherheit nicht zu groß.

## mobileIP

Ein Kernproblem, das mobile Geräte, die über das Internet Protokoll mit anderen Geräten in Verbindung stehen, bei Änderung ihrer Position erfahren ist, dass sie die Kommunikation mit den anderen Rechnern zunächst unterbrechen, eine neue Adresse zugeordnet bekommen und dann die Verbindung wieder aufnehmen müssen. Dieses Problem wurde von der Internet Engineering Task Force (IETF, http://www.ietf.org ) Anfang der neunziger Jahre adressiert. Das Ergebnis wurde 1996 in Form des *proposed standards* MobileIP vorgelegt. MobileIP, auf der Netzwerkschicht angesiedelt, setzt auf dem Internet Protokoll auf und erweitert es um die notwendigen Features hinsichtlich Mobilität. Rechner können nun auch in Fremdnetzwerken integriert werden, ohne dass Kommunikationsverbindungen neu aufgebaut werden müssten. Die Rechner behalten ihre originäre IP-Adresse (bekommen aber eine zusätzliche hinzu).

Das Problem der Integration in Fremdnetzwerke liegt im Routingmechanismus des IP begründet. Jeder Knoten liest den Header eines zu transportierenden Pakets aus und leitet ihn an die entsprechende Adresse weiter. Da Fremdknoten nur den vorderen Teil auslesen, leiten sie jedes Paket an das Heimatnetz des Zielknotens weiter. Befindet sich dieser nun in einem fremden Netz, ist er nicht erreichbar.

An die Entwicklung von MobileIP wurde eine ganze Reihe an Anforderungen gestellt:

- Jeder mobile Knoten muss in der Lage sein, die Kommunikation zu anderen Knoten nach Änderung des Zugangspunktes ausschließlich mittels seiner Heimatadresse aufrecht zu halten.

- Mobile Knoten müssen mit solchen kommunizieren können, die MobileIP nicht implementiert haben.
- Für mobile Knoten sollen die gleichen Sicherheitsmerkmale gelten wie für stationäre.

MobileIP definiert zunächst einmal drei funktionale Entitäten:

- Mobiler Knoten (**mobile node**): Ein mobiler Knoten ist ein Knoten, der seinen Zugangspunkt ins Netz beliebig verändern kann, dabei die laufende Kommunikation mit anderen Knoten aufrecht erhält und zur Identifikation ausschließlich seine Heimatadresse nutzt.
- Heimatagent (**home agent**): Der Heimatagent ist ein Knoten (typischer Weise ein Router) mit Schnittstelle zum Heimatnetz des mobilen Knotens, der dafür sorgt, dass der mobile Knoten erreichbar bleibt, auch wenn er nicht beim Heimatnetz angemeldet ist.
- Fremdagent (**foreign agent**): Fremdagenten sind Knoten (typischer Weise Router) im Fremdnetzwerk, die dem mobilen Knoten beim Empfang von Datenpaketen assistiert, die an die *care-of-address* geliefert werden.

## Care-of-address

Ein mobiler Knoten soll unter seiner Heimatadresse erreichbar sein. Da sich in der Adresse eines Knoten auch sein Heimatnetz ausdrückt, muss eine zweite Adresse genutzt werden, wenn sich ein Knoten in einem Fremdnetz aufhält: Die care-of-address. Diese ist fremdnetzwerkspezifisch (um den Transport von Paketen in das richtige Zielnetzwerk zu gewährleisten) und ändert sich somit jedes Mal, wenn der mobile Knoten einem anderen Fremdnetzwerk beitritt. Sie stellt diejenige Adresse dar, an die die Pakete, vormals adressiert an die Heimatadresse, weitergeleitet werden. Ein mobiler Knoten hat somit im Fremdnetzwerk zwei IP-Adressen. Eine statische und eine wenn man so will topologische, die seine Position im Netz beschreibt.

Care-of-Adressen lassen sich übrigens nochmals unterscheiden in *foreign agent care-of-address* und *collocated care-of-adress*. Sie unterscheiden sich dahingehend, dass erstere zum Fremdagenten gehört und in ihrem Präfix nicht dem des Netzes entsprechen muss, in dem sich der mobile Knoten gerade aufhält. Somit können mehrere mobile Knoten eine care-of-address nutzen. Der Fremdagent kümmert sich dann um die korrekte Zustellung. Letztere wird genutzt, wenn kein Fremdagent zur Verfügung steht. Sie muss in ihrem Präfix dem Fremdnetz entsprechen und kann nur von einem einzigen mobilen Knoten genutzt werden.

## Funktionsprinzip

Heimat- und Fremdagenten zeigen ihre Anwesenheit in ihren Netzen durch periodischen Versand von Multicast oder Broadcast Messages an. Diese Nachrichten werden als *Agent Advertisements* bezeichnet. Mobile Knoten hören diese agent advertisements ab und entscheiden darauf hin, ob sie sich in einem Fremd- oder Heimatnetz befinden. Dieser Vorgang, der auch vom mobilen Knoten initiiert werden kann, nennt man *Agent Discovery*. Lokalisiert sich ein Knoten in seinem Heimatnetz, macht er keinen Gebrauch von der MobileIP Funktionalität, sondern verhält sich wie jeder statische Knoten innerhalb dieses Netzes auch. Ermittelt er jedoch ein Fremdnetz, so bekommt er eine care-of-address zugewiesen.

Die care-of-address ist dabei in den agent advertisements enthalten. Für den Fall, dass kein Fremdagent seine Präsenz signalisieren sollte, wird von anderen Methoden der Adresszuweisung Gebrauch gemacht, beispielsweise über DHCP. Natürlich kann auch manuell eine collocated care-of-address zugewiesen werden. Der mobile Knoten registriert seine care-of-address während des *Registration*-Vorgangs bei seinem Heimatagenten.

Um ein Paket vom Heimatnetz zu empfangen (alle Pakete von Drittknoten sind weiterhin an das Heimatnetz adressiert, sie wissen nichts von der Mobilität des Knotens) muss es vom Heimatagenten an die care-of-address weitergeleitet werden. Dazu ist das Paket so zu modifizieren, dass die care-of-address als Paketzieladresse im Header eingetragen wird. Diese Modifikation kann als *packet transformation*, oder spezieller als *redirection* begriffen werden. Wenn das Paket an der care-of-address ankommt wird es vom Fremdagenten nach *reverse transformation* dem mobilen Knoten zugestellt.

Der Heimatagent versieht weiterzuleitende Pakete mit einem neuen Header, der die originäre Zieladresse verbirgt und stattdessen die care-of-address aufweist. Diese Kapselung wird als *tunneling* bezeichnet.

Pakete, die der mobile Knoten an andere Rechner verschickt, werden nicht getunnelt. Der mobile Knoten schickt sie an den als Router dienenden Fremdagenten, der sie direkt weiterleitet.

## Agent Discovery

Agent Discovery nennt man die Lokalisierungsprozedur, während der ein mobiler Knoten feststellt, ob er sich in einem Fremd- oder Heimatnetz befindet und ob er sich seit seinem letzten Login bewegt hat (*move detection*). Wir haben bereits erfahren, dass der mobile Knoten während dieser Prozedur agent advertisements empfängt. Es gibt noch eine weitere Art von Botschaft: *Agent Solicitations*. Diese werden vom suchenden Knoten ausgesandt und sind eine Aufforderung an Fremdagenten, sofort eine agent advertisement Nachricht zu verschicken.

Agent advertisement und agent solicitation sind im MobileIP Standard genau definiert. Ein mobiler Knoten kann durch Abgleichen des Netzwerkpräfixes im header eines agent advertisement mit seinem Heimatnetzwerkpräfix feststellen, ob er sich in einem Fremdnetzwerk oder in seinem Heimatnetzwerk befindet. Stellt er dabei fest, dass er gerade in sein Heimatnetzwerk zurückgekommen ist, löscht er die Registrierung bei seinem Heimatagenten (*deregistration*).

Für den Fall, dass er sich in einem Fremdnetz aufhält muss er feststellen, ob er zwischen diesem und dem letzten Login das Netzwerk gewechselt hat. Das kann ebenfalls über den Abgleich der Netzwerkpräfixe erfolgen. Agent advertisements enthalten neben der Netzwerkkennung auch Zeitstempel und sogenannte *Lifetime* Informationen. Diese geben an, zu welchem Zeitpunkt das nächste agent advertisement gesendet werden wird (in Wirklichkeit werden sie häufiger gesendet, um möglichen *package loss* (Paketverluste) Problemen zu begegnen). Empfängt ein Knoten nach Ablauf der lifetime kein neues agent advertisement, muss er davon ausgehen, das Netzwerk gewechselt zu haben und eine neue care-of-address beantragen.

## Registrierung der care-of-address

Sobald ein mobiler Knoten in einem Fremdnetzwerk eine care-of-address zugewiesen bekommen hat, muss er diese in seinem Heimatnetzwerk registrieren. Abbildung 1 illustriert diesen Prozess.

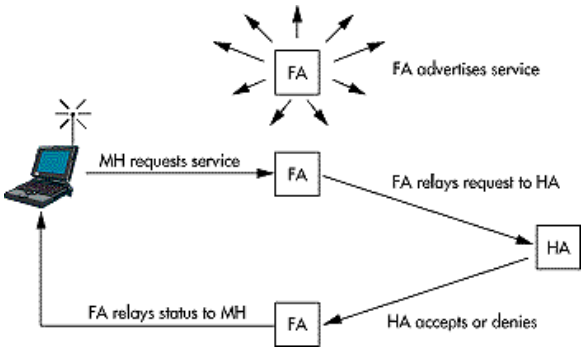


**Abbildung 1: MobileIP Registration (Perkins 1997)**

Die Registrierungsprozedur beginnt damit, dass er, eventuell mit Hilfe des Fremdagenten, einen Registrierungsrequest mit der care-of-address an den Heimatagenten schickt. Dieser überprüft die Anfrage, akzeptiert sie (oder lehnt sie ab), fügt die Adresse typischer Weise seiner Routingtabelle hinzu und sendet eine *registration reply* an die care-of-address zurück. Registrations haben wie agent advertisements auch eine festgelegte Gültigkeitsdauer. Läuft diese ab, muss sich ein Knoten erneut registrieren.

Die registration requests enthalten neben anderen Parametern und flags die Heimatnetzwerkadresse, die care-of-address und die registration lifetime. Dieses Triplet bezeichnet man auch als *binding* für den mobilen Knoten. Ein registration request ist folglich ein *binding update*.

Ein binding update ist ein *remote redirect*, d.h. von einer entfernten Station wird eine Aktualisierung der Heimatagenten-Routingtabelle beantragt. Der Heimatagent muss folglich sicher sein können, dass die Anfrage originär wirklich vom mobilen Knoten stammt und unterwegs nicht modifiziert wurde. Dies macht eine sichere Authentifizierung notwendig.

## Authentifizierung

So verwundert es nicht, dass der Authentifizierungsmechanismus eine wichtige Rolle bei der Entwicklung und bei der späteren Akzeptanz von MobileIP spielte. Heimatagent und mobiler Knoten machen zur Authentifizierung Gebrauch vom MD5-Algorithmus (Message Digest 5, RFC 1321) und teilen sich 128-bit Schlüssel. Dieser Algorithmus erlaubt es, für beliebig lange Datensätze Datenblöcke mit fester Länge zu berechnen, den so genannten *Message Digest*. Ein mobiler Knoten errechnet seinen message digest über eine Sequenz von Bytes, die den geheimen Schlüssel, den registration request mit allen Erweiterungen und erneut den geheimen Schlüssel enthalten und trägt den message digest in ein spezielles Authentifizierungsfeld innerhalb des registration requests ein. Der Heimatagent errechnet selbst den gleichen message digest und vergleicht beide. Bei Übereinstimmung kann er davon ausgehen, dass der request originär vom mobilen Knoten stammt und unterwegs nicht verändert wurde. Bei registration replies findet die inverse Prozedur statt.

Um so genannte *replay attacks* zu verhindern (das sind Angriffe, bei denen versucht wird, durch erneutes Senden von abgefangenen validen requests die Sicherheitsmechanismen zu unterlaufen), enthält jeder request ein spezielles Registrierungsfeld, welches sich bei jedem request ändert. Somit sind zwei MD5 hash in der Praxis niemals gleich. Der wiederholte Empfang von ein und demselben hash kann als Angriff gedeutet werden. Die Semantik dieses Feldes ist recht komplex, im Wesentlichen gibt es aber zwei Wege, die Eineindeutigkeit des Identifikationsfeldes des requests zu sichern. Der erste nutzt Zeitstempel, was dazu führt, das jeder spätere request einen neueren Zeitstempel aufweist. Die zweite Möglichkeit beruht auf der Berechnung von Zufallszahlen. Die MobileIP Spezifikation sieht dafür 32bit Zufallszahlen vor, was als ausreichend groß erscheint, damit sich in der Praxis keine zwei requests gleichen.

## Tunneling

Wenn sich ein mobiler Knoten in einem Fremdnetzwerk aufhält müssen an ihn adressierte Pakete umgeleitet werden. Abbildung 2 zeigt die tunneling Operationen, die bei MobileIP zum Einsatz kommen. Eingehende Pakete müssen gekapselt, mit einem neuen Header versehen und weitergeschickt werden. Der Standardkapselungsmechanismus, der von allen Agenten unterstützt werden muss, ist *IP-within-IP*. Dabei schreibt der Heimatagent, die *tunnel source*, den neuen IP header, oder *tunnel header*, vor den IP header des weiterzuleitenden Datagrams. Der neue tunnel header nutzt die care-of-address als neue IP Zieladresse, oder *tunnel destination*. Die tunnel source IP Adresse ist die des Heimatagenten und der tunnel header nutzt eine vier für die *higher protocol number*, die anzeigt, dass der nächste Protokoll header ebenfalls ein IP header ist. Bei IP-within-IP wird der gesamte originäre Header inklusive Daten (Payload) erhalten. Der Fremdagent muss folglich nur den tunnel header eliminieren und kann dann das Paket an den mobilen Knoten zustellen.

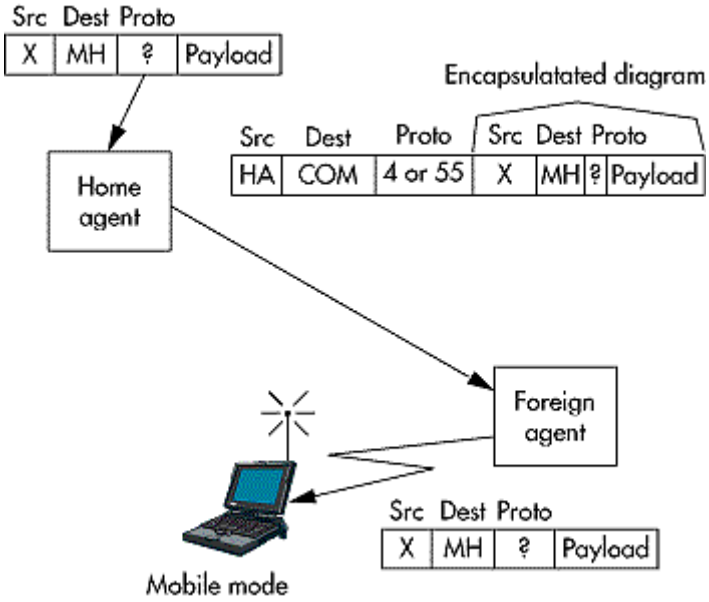


**Abbildung 2: MobileIP Tunneling (Perkins 1997)**

Ein weiteres Verfahren, dass an Stelle von IP-within-IP angewandt werden kann ist *minimal encapsulation*. Angezeigt durch die Protokollnummer 55 kombiniert dieses Verfahren Teile des tunnel headers und des inneren headers; der Originalheader wird also ersetzt. Dieses kompliziertere Verfahren vermeidet den bei Ip-within-IP auftretenden header overhead.

## Reverse Tunneling

MobileIP beruht auf der Annahme, dass das Routing von Paketen auf Basis der Zieladresse im Header beruht. Aus Sicherheitsgründen analysieren manche Router aber auch die Absenderadresse. Wenn diese nicht mit der Herkunft des Pakets konsistent zu sein scheint, wird das Paket vernichtet, so genanntes *Ingress Filtering*. Dieses Problem tritt auf, wenn ein mobiler Knoten mit seiner Heimatadresse als Absender Pakete zu einem Router im Fremdnetzwerk schickt. In diesem Fall stimmen Absenderadresse und topologische Herkunft der Pakete nicht überein, sie werden vernichtet.

Reverse tunneling ist ein Mechanismus, der dieses Problem umgeht. Der mobile Knoten macht den Wunsch nach reverse tunneling durch das Setzen eines flags im registration request deut-

lich. Er nutzt dann seine care-of-address, die Pakete werden vom Fremdagenten zum Heimatagenten „zurückgetunnelt", dort mit der Heimatabsenderadresse versehen und schließlich zum eigentlichen Ziel geroutet.

### MobileIP und IPv6

Ipv6 bringt eine Menge Neuerungen mit sich, insbesondere mit Hinblick auf mögliche Mobilität der Knoten, wie beispielsweise *Stateless Address Autoconfiguration* and *Neighbor Discovery*. Die Mobilitätsunterstützung von IPv6 folgt in weiten Teilen den Designvorschlägen, die in Mobile IPv4 gemacht wurden. Es behält die Ideen des Heimatnetzwerkes, des Heimatagenten und die Kapselung von Paketen zum Transport vom Heimatnetz zur aktuellen Knotenposition bei. Während care-of-address weiterhin benötigt werden, wird der Gebrauch von Fremdagenten überflüssig, da er durch die oben erwähnten Techniken stateless address autoconfiguration und neighbour discovery ersetzt werden kann. IPv6-within-IPv6 wird IP-within-IP ablösen.

Für weitere Informationen bzgl. MobileIP siehe das sehr gute Tutorial von Charles E. Perkins unter http://computer.org/internet/v2n1/perkins.htM, sowie Seite **Fehler! Textmarke nicht definiert.**.

## Routing in Ad Hoc Netzwerken

Ein Ad Hoc Netzwerk besteht aus einer Menge von mittels Funkwellen verbundener, mobiler Computer (s.a. Kapitel 3). Die Kommunikation in solch einem Netzwerk läuft über sog. Routing-Protokolle. Mit ihnen sollen möglichst effizient Routen zwischen zwei Knoten im Netzwerk gefunden und aufrechterhalten werden, so dass Daten in einem angemessenen Zeitrahmen übertragen werden können. Unter Effizienz versteht man in diesem Fall eine möglichst fehlerfreie Kommunikation bei möglichst geringem Energieverbrauch und bestmöglicher Ausnutzung der zur Verfügung stehenden Bandbreite. Deshalb werden sie ständig erweitert bzw. verbessert. Im Folgenden sollen einige Routing-Protokolle kurz vorgestellt werden.

Die vorhandenen Routing-Protokolle lassen sich in zwei Bereiche einteilen, zum einen in die tabellenorientierten und zum anderen in die on-Demand Protokolle. Während bei den tabellenorientierten Verfahren versucht wird, die gesamte Netztopologie in Tabellen zu speichern und nachzuführen, versuchen die on-Demand Verfahren erst dann einen Pfad zwischen zwei Knoten zu finden, wenn dieser gebraucht wird. Dies spart zum einen Speicherplatz und zum anderen werden die hohen Kommunikationskosten in tabellenorientierten Ad Hoc Netzwerken vermieden, die allein zur Nachführung der sich ständig ändernden Netztopologie benötigt wird.

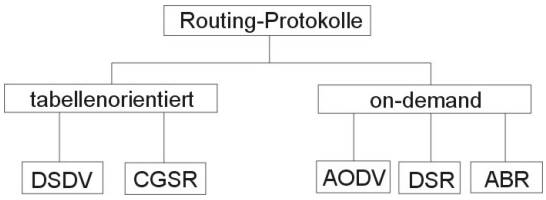


**Abbildung 3: Übersicht einiger Routing-Protokolle**

### Tabellenorientierte Routing-Protokolle

Die Grundlage tabellenorientierter Routing-Protokolle bilden konsistente und stets aktuelle Routing-Informationen für alle Knoten. Jeder Knoten speichert die Routen von sich selbst zu allen anderen Knoten in einer oder mehreren Tabellen. Hierdurch können jederzeit Daten versandt werden, ohne zuerst geeignete Routen ermitteln zu müssen. Änderungen der Netz-

topologie werden über Updates im gesamten Netzwerk verbreitet. Die vorhandenen Protokolle unterscheiden sich zum einen in der Anzahl der benötigten Tabellen und zum anderen in der Weise, wie Änderungen im Netzwerk behandelt werden.

## Destination-Sequenced Distance-Vector Routing (DSDV)

Im DSDV speichert jeder Knoten in einer Tabelle die im Netzwerk vorhandenen Zielknoten, den nächsten Knoten auf dem kürzesten Weg zum Zielknoten und die insgesamt benötigten *hops* (Zwischenstationen auf dem Pfad zum Zielknoten).

Um die Routing-Tabelle jederzeit auf dem neuesten Stand zu halten, werden in regelmäßigen Abständen broadcasts versandt. Um den Netzverkehr hierfür so gering wie möglich zu halten, werden zwei Typen von broadcasts unterschieden: zum einen der full dump, der die insgesamt zur Verfügung stehenden Routing-Informationen beinhaltet, und zum anderen das *incremental package*, das nur die Informationen beinhaltet, die sich nach dem letzten full dump geändert haben. Offensichtlich bedingt das Senden eines full dump eine viel größere Netzlast als beim Senden eines incremental package. Das Senden eines incremental package ist jedoch nur in Netzwerken mit geringer Mobilität der Knoten sinnvoll anwendbar.

Um Schleifen beim Routing zu vermeiden, verwendet DSDV Zählernummern, die anzeigen, wie aktuell eine gegebene Route ist. Bei einem Update werden Routen mit einer höheren Nummer immer den Routen mit einer geringeren vorgezogen. Bei gleicher Zählernummer wird die Route mit der geringeren Anzahl hops als aktuelle Route gespeichert.

## Clusterhead Gateway Switch Routing (CGSR)

Bei CGSR handelt es sich um ein multihop-Netzwerk, welches DSDV als unterliegendes Routingschema benutzt. Jedoch wird hier ein hierarchisches *cluster-head-to-gateway* Schema verwandt, um Daten zu übertragen. Das Netzwerk ist in Cluster aufgeteilt. Jedem Cluster ist wiederum ein Clusterhead zugeteilt, der mit anderen Clusterheads über sog. gateways kommuniziert, die im Empfangsbereich von zwei oder mehr Clusterheads liegen.

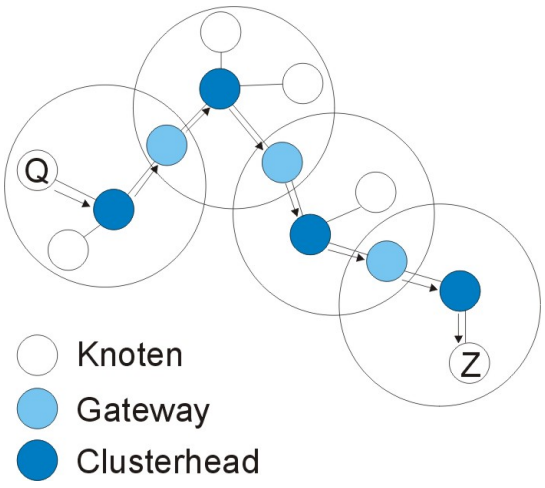


**Abbildung 4: CGSR Routing**

Die Clusterheads können über einen speziellen Algorithmus bestimmt werden. Problematisch würde sich jedoch ein ständiger Wechsel von Clusterheads auswirken, da die Netzwerkknoten hierbei mehr mit dem Bestimmen eines neuen Clusterhead beschäftigt wären als mit der Da-

tenübertragung selbst. Deshalb wird der sog. *Least Cluster Change* (LCC) Algorithmus angewandt. Mit LCC wechseln Clusterheads nur, wenn sich zwei Clusterheads berühren würden oder ein Knoten im Netzwerk aus dem durch Cluster abgedeckten Bereich geraten würde.

Die Kommunikation zwischen zwei Knoten läuft über einen Pfad aus Clusterheads. Daten werden zuerst zum Clusterhead des Quellknotens geleitet, der diese dann über den kürzesten zur Verfügung stehenden Pfad aus Clusterheads zum Clusterhead des Zielknotens leitet. Dieser übermittelt die Daten dann an den Zielknoten selbst.

Bei CGSR muss jeder Knoten zwei Tabellen speichern. Zum einen die *cluster member table*, in der zu jedem Knoten der zugehörige Clusterhead gespeichert wird. Diese Tabelle wird wie bei DSDV über periodisch ausgesandte broadcasts auf dem neuesten Stand gehalten. Zum anderen muss jeder Knoten eine Routing-Tabelle speichern, in welcher der nächste Knoten auf dem Weg zu einem Zielknoten gespeichert ist. Routing von Daten wird dann anhand dieser zwei Tabellen möglich. Über die cluster member table kann ein Knoten den nächsten Clusterhead auf dem Weg zu einem Zielknoten ermitteln und über die Routing-Tabelle kann dann der gateway bestimmt werden, über den mit diesem Clusterhead kommuniziert werden kann.

## On-Demand Routing

Im Gegensatz zu den tabellenorientierten Routing-Protokollen werden bei den on-demand-Verfahren Routen nur bei Bedarf gesucht und etabliert. Ein Knoten, der senden möchte, initiiert einen *route discovery*-Prozess. Nachdem eine oder mehrere Routen gefunden wurden, wird versucht, diese durch einen *route maintenance*-Prozess aufrechtzuerhalten. Der Prozess endet, wenn entweder die Verbindung nicht mehr aufrechterhalten werden kann (u.a. bedingt durch die Mobilität der im Pfad benutzten Knoten) oder die Verbindung nicht mehr erwünscht ist. Hierdurch wird die für das Routing benötigte Kommunikation zwischen den Knoten auf ein Minimum reduziert.

## Ad Hoc On-Demand Distance Vector Routing (AODV)

AODV baut ebenso wie CGSR auf dem DSDV auf. Ähnlich wie in DSDV benutzt auch AODV Zählernummern, um die Aktualität einer Route anzuzeigen und Schleifen zu vermeiden. Bei AODV speichert jeder Knoten aktive Routen in Routing-Tabellen.

Um eine Route zu finden, wird ein *Route Request* (RREQ) vom Quellknoten ausgesandt, der im Netzwerk verteilt wird, bis entweder der Zielknoten selbst oder ein Knoten mit einer aktuellen Route zum Zielknoten gefunden wurde. Ein RREQ kann über die IP des Knotens, der den RREQ ausgesandt hat, und einen Zähler, den jeder Knoten für jeden neuen, von ihm ausgesandten RREQ inkrementiert, eindeutig identifiziert werden. Auf dem Weg des RREQ werden in jedem Knoten temporäre Einträge in der Routing-Tabelle erzeugt, die auf den Vorgängerknoten verweisen, von dem der RREQ empfangen wurde. Später eintreffende Duplikate eines RREQ werden verworfen. Auf diese Weise wird gewährleistet, dass nur der kürzeste Weg gefunden wird.

Sobald der Zielknoten oder ein Knoten mit einer frischen Route zum Zielknoten gefunden wurde, antwortet dieser Knoten mit einem *Route Reply* (RREP). Dieser wird zu dem Knoten gesandt, von dem der RREQ empfangen wurde, der diesen dann wieder weiterleitet etc. Auf dem Weg zum Quellknoten werden die Einträge in den Routing-Tabellen gelöscht und neue Einträge erzeugt, die diesmal jedoch auf den nächsten Knoten auf dem Weg zum Zielknoten zeigen. Auf diese Weise wird eine aktive Route vom Quell- zum Zielknoten erzeugt. Für jede aktive Route eines Knotens läuft ein timer. Wenn dieser abgelaufen ist, die Route also nicht mehr benutzt wird, wird der Eintrag aus der Routing-Tabelle gelöscht.

Durch Bewegung von Knoten einer aktiven Route kann diese unterbrochen werden. Bewegt sich der Quellknoten, muss dieser einen neuen RREQ abschicken. Wird die Route an einer anderen Stelle unterbrochen, so sendet der Knoten, der die Unterbrechung bemerkt hat, eine

*link faillure notification* zurück zum Quellknoten. Dieser kann dann entscheiden, ob die Route immer noch benötigt wird, was einen neuen RREQ zur Folge hätte.

Doch wie erkennt ein Knoten, dass eine Route unterbrochen wurde? Hierzu können *hello*-Nachrichten benutzt werden. Diese werden periodisch von jedem Knoten an seine Nachbarknoten gesandt. Wenn diese Nachrichten von einem zuvor als Nachbarn erkannten Knoten nicht mehr empfangen werden, kann von einer Änderung der Netztopologie ausgegangen werden. Routen, die über diesen Knoten laufen würden, gelten dann als unterbrochen. Unterbrochene Routen können auch auf andere Weise von einem Knoten erkannt werden. Zum einen kann ein Knoten sich den Empfang eines gesendeten Datenpaketes bestätigen lassen (aktive Bestätigung). Zum anderen kann ein Knoten die gesendeten Daten eines Nachbarknoten „mithören" und so erkennen, ob ein Datenpaket weitergeleitet wird (passive Bestätigung).

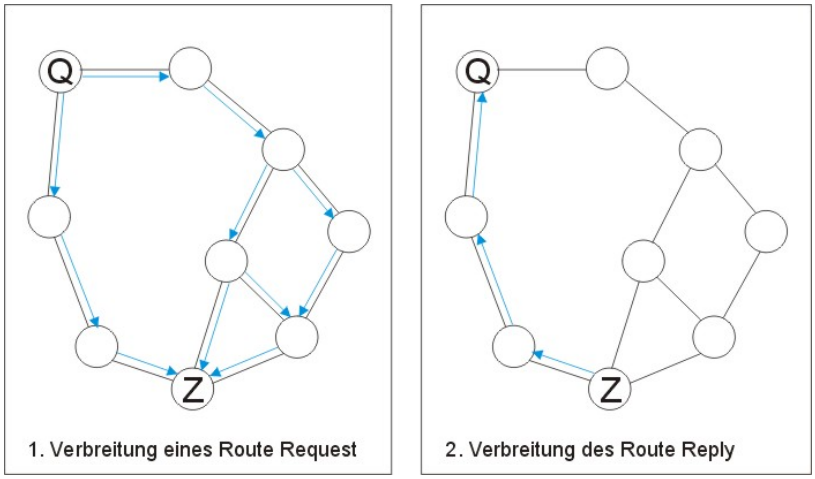


**Abbildung 5: AODV Routing**

## Dynamic Source Routing (DSR)

Beim Dynamic Source Routing verwaltet jeder Knoten einen eigenen Route-Cache. In dem Route-Cache eines Knotens werden alle aktuellen Routen gespeichert, die von diesem Knoten zu einem anderen Knoten im Netzwerk führen. Die Einträge dieses Cache werden ständig aktualisiert.

Benötigt ein Knoten eine Route zu einem anderen Knoten, sucht er zuerst in seinem Route-Cache nach einer bekannten Route zu dem jeweiligen Zielknoten. Wird eine Route gefunden, so kann diese zur Übermittlung benutzt werden. Ansonsten wird ein Route Request (RREQ) ausgesandt, der die Adressen sowohl des Quell- als auch des Zielknotens zusammen mit einer eindeutigen ID enthält. Jeder Knoten, der einen solchen RREQ empfängt, konsultiert wiederum zuerst seinen Route-Cache. Falls keine Route gefunden wurde, fügt der Knoten seine Adresse dem Header des RREQ hinzu und sendet den modifizierten RREQ an alle Nachbarn. Auf diese Weise enthält ein RREQ in seinem Header die bereits angesteuerten Knoten in der Reihenfolge, wie sie vom Quellknoten aus besucht wurden. Um die Netzlast möglichst gering zu halten, werden nur solche RREQs bearbeitet, die noch nicht empfangen wurden und deren Header den empfangenden Knoten noch nicht enthält – hierdurch werden auch Schleifen vermieden.

Erreicht ein RREQ den gesuchten Zielknoten oder einen Knoten, der eine Route zum Zielknoten in seinem Cache enthält, so wird ein Route Reply (RREP) erzeugt. Handelt es sich bei dem den RREQ empfangenden Knoten um den Zielknoten, fügt dieser die Route aus dem Header des RREQ dem RREP hinzu und schickt diesen an den Quellknoten zurück. Im anderen Fall hängt der Knoten die im Route-Cache gefundene Route an die Route im Header des RREQ an

und sendet diese anschließend mittels eines RREP an den Quellknoten. Für das Senden des RREP kann bei Bedarf eine andere Route benutzt werden als die Route, die der RREQ genommen hat (asymmetrische links).

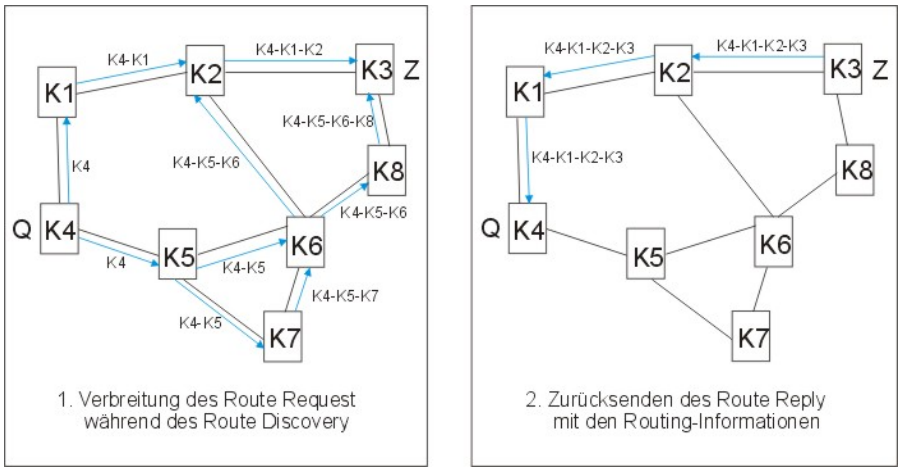


**Abbildung 6: Dynamic Source Routing**

Route maintenance wird hierbei über *Route error packets* und Bestätigungen erreicht. Route error packets werden erzeugt, wenn ein Knoten feststellt, dass eine zuvor existente Verbindung zwischen zwei Knoten nicht mehr besteht. Alle Routen im Route-Cache, die diese Verbindung benutzen, werden daraufhin gelöscht. Neben den error packets werden wie beim AODV aktive und passive Empfangsbestätigungen benutzt, um Unterbrechungen von Routen feststellen zu können.

Im Vergleich zum Ad Hoc On-Demand Distance Vector Routing verursacht das Dynamic Source Routing unter Umständen einen signifikanten Routing-Overhead, da bei DSR die für das Routing benötigten Datenpakete die Adressen aller Knoten einer Route mitführen müssen. Ein Vorteil zum AODV ergibt sich aus der Möglichkeit, dass DSR asymmetrische Links benutzen kann. Ein weiterer Vorteil von DSR ist, dass keine periodisch wiederkehrenden broadcasts benötigt werden. Das spart zum einen Bandbreite und zum anderen Akkuleistung. Außerdem können bei DSR mehrere Routen zu einem Zielknoten gespeichert werden. Hierdurch lassen sich einige Route-Discovery-Prozesse einsparen.

DSR wird in Netzwerken mittlerer Größe und durchschnittlicher Mobilität angewandt.

## Associativity-Based Routing (ABR)

Das Associativity-Based Routing definiert eine im Vergleich zu den bisher behandelten Protokollen völlig neue Metrik: die *degree of association stability*. Routen werden in ABR anhand der Stabilität der Verbindung zwischen Knoten ermittelt. Hierzu sendet jeder Knoten periodisch einen *beacon* aus, um seinen Nachbarn seine Existenz anzuzeigen. Jeder Knoten, der einen solchen beacon empfängt, erhöht einen internen Zähler für diesen Knoten, den *associativity ticker*. Hohe Zählerwerte zeigen eine hohe Stabilität der Verbindung zwischen zwei Knoten an, was wiederum auf eine geringe Mobilität im Netzwerk hindeutet. Niedrige Zähler zeigen dagegen labile Verbindungen an, die Mobilität der Knoten ist hoch. Wird von einem Knoten kein beacon mehr empfangen, gilt die Verbindung zu diesem Knoten als unterbrochen.

Daraufhin wird der interne Zähler für diesen Knoten wieder auf 0 gesetzt. Auf diese Weise können in ABR langlebige Routen ermittelt werden.

Um eine Route zu ermitteln, sendet ein Knoten einen broadcast aus, der den Zielknoten und alle associativity ticks für die Nachbarn des sendenden Knotens enthält. Jeder Knoten, der ein solches Signal erhält und nicht der Zielknoten ist, löscht alle associativity ticks für die anderen Nachbarknoten des Knotens, von dem das Signal empfangen wurde. Daraufhin wird wieder ein Signal ausgesandt, das nun jedoch neben den associativity ticks zu den Nachbarn auch die bislang ermittelte Route zum Quellknoten enthält. Gelangt das Paket schließlich zum Zielknoten, kann dieser die stabilste Route anhand der associativity ticks erkennen. Routen mit gleicher Stabilität werden anhand der im Pfad vorhandenen Zwischenstationen unterschieden. Der Zielknoten sendet dann einen REPLY über die ermittelte Route an den Quellknoten. Nur die Knoten auf dem Pfad des REPLY können im Folgenden für die Datenübertragung zwischen Quell- und Zielknoten benutzt werden. Hierdurch werden doppelte Pakete vermieden.

Was geschieht nun, wenn sich die Knoten einer Route im Netzwerk bewegen? Bewegt sich der Quellknoten, so werden alle Knoten der alten Route darüber informiert, dass diese Route nicht mehr gültig ist. Danach wird vom Quellknoten aus eine neue Route gesucht. Wird jedoch innerhalb der Route eine Unterbrechung angezeigt, so versucht der Knoten, der nun das Ende der vom Quellknoten ausgehenden Route markiert, eine neue Route zum Ziel zu ermitteln. Diese neue Route darf höchstens die gleiche Anzahl hops wie die zuvor benutzte Route benötigen. Gelingt dies nicht, so versucht der nächste Knoten solch eine Route zu finden. Der Prozess endet, wenn die Hälfte der ursprünglichen Route ohne Erhalt einer neuen Route erreicht wird. In diesem Fall muss eine gänzlich neue Route gesucht werden.

Wird eine Route nicht mehr benötigt, so wird ein Route Delete (RD) vom Quellknoten ausgesandt, der sich im Netzwerk verteilt. Durch ihn werden alle Einträge für die nicht mehr benötigte Route aus den Routing-Tabellen gelöscht.

Durch Asscociativity-Based Routing können langlebigere Routen ermittelt werden als in den anderen vorgestellten Protokollen. Hierdurch wird der Datendurchsatz merklich erhöht. Nachteilig wirkt sich jedoch das Senden der beacons aus. Um die Mobilität des Netzwerkes reflektieren zu können, müssen unter Umständen die Übertragungsintervalle der beacons verkleinert werden, was sich letztlich negativ auf die Akkukapazität auswirken würde.

## Literatur

(1) Frodigh, M., Johansson, P. and Larsson, P. (2000) Wireless ad hoc networking—The art of networking without a network. Ericsson Review no. 04, document-type: pdf http://www.ericsson.com/

(2) Perkins, C. E. (1997). MobileIP registration.

(3) Perkins, C. E. (1997). MobileIP Tunneling.

(4) Royer, E. M. und Toh, C. K. (1999) A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Netwerks. IEEE Personal Communications, document-type : pdf, accessed :14.5.2003