Universität Karlsruhe (TH)
Institut für Telematik

TeleMatics Technical Reports

Mobiles Internet
Seminar WS04/05

Peter Baumung, Bernhard Hurler, Tobias Küfner, Christian Vogt, Martina Zitterbart
{baumung,hurler,kuefner,chvogt,zit}@tm.uka.de

April, 1st 2005

TM-2005-2

ISSN 1613-849X
http://doc.tm.uka.de/tr/

Institute of Telematics, University of Karlsruhe
Zirkel 2, D-76128 Karlsruhe, Germany

Vorwort

Das Seminar "Mobiles Internet" wurde im Wintersemester 2004/2005 in Form eines Blockseminars am 24. und 25. Februar 2005 am Institut für Telematik abgehalten. Die Themen des Seminars stammten aus den Bereichen

- "Routing in mobilen Ad-hoc-Netzen",
- "Optimierungen und Anwendungen mobile für Ad-hoc-Netze",
- "Optimierungen und Erweiterungen für Mobile IPv6",
- "Das Host-Identity-Protokoll (HIP)",
- "Alternative Ansätze für Mobilitätsmanagement im Internet" und
- "Paketkopfkomprimierung in drahtlosen Zugangsnetzen".

In diesem Seminarband werden nun die Ausarbeitungen der Studenten in Form eines internen Berichts zusammengefasst.

Routing in mobilen Ad-hoc-NetzenMobile IP

Ein mobiles Ad-hoc-Netz (MANET) besteht aus mobilen Knoten, die drahtlos und ohne zentrale Administration miteinander kommunizieren können. Um innerhalb dieser Netze über mehrere Zwischenknoten hinweg kommunizieren zu können, werden Routing-Protokolle benötigt, welche eine (durch mobile Knoten) sich ständig ändernde Netz-Topologie bewältigen können. Drei typische Vertreter solcher Protokolle (DSDV, DSR, AODV) werden im vorliegenden Seminarband vorgestellt.

Optimierungen und Anwendungen mobile für Ad-hoc-Netze

Gruppenkommunikation in mobilen Ad-hoc-Netzen ist ein aktuelles Forschungsthema. Da zur Erbringung solcher Dienste Endsystem-basierte Ansätze diverse Vorteile bieten, wird in einer Seminararbeit ein aktuelles Protokoll auf Anwendungsebene mit einem herkömmlichen und anerkannten Protokoll auf Netzwerkebene verglichen.

Optimierungen und Erweiterungen für Mobile IPv6

Mobiltelefone, PDAs und Notebooks erlauben dem Anwender seinen aktuellen Zugangspunkt zum Internet zu ändern, während er seine Anwendung fortsetzt. Befinden sich der alte und der neue Zugangspunkt in verschiedenen Subnetzen, muss der Knoten dabei seine IP-Adresse ändern. Die Hauptaufgabe von Mobile IPv6, der integrierten Mobilitätsunterstützung in IPv6, ist es, diese mobilitätsbedingten Adressänderungen vor den höheren Schichten zu verbergen. Im vorliegenden Seminarband werden zwei wichtige Erweiterungen bzw. Optimierungen von Mobile IPv6 genauer betrachtet. Die Unterstützung von mobilen Netzen – im Gegensatz zu einzelnen mobilen Knoten – ist insbesondere für Personal Area Networks (PANs) und Vehicular Area Networks (VANs) von Interesse. Fast Handover ist ein Protokoll, dass es mobilen Knoten erlaubt, einen Handover vorzubereiten, während sie noch über den alten Zugangspunkt kommunizieren.

Das Host-Identity-Protokoll (HIP)

IP-Adressen haben traditionsgemäß mehrere Funktionen. Anhand des Präfixes kann man einen Knoten lokalisieren. In voller Länge identifizieren sie ein Netzwerk-Interface. Und für die meisten Anwendungen und Transport-Protokolle identifizieren sie gleich den gesamten Knoten.

Ist der Knoten mobil, muss er ab und zu seine IP-Adresse ändern. Damit ändert sich für viele Anwendungen und Transport-Protokolle die Identität des Knotens. Als Folge müssen solche Anwendungen und Transport-Protokolle neu gestartet werden.

Das Host Identitity Protocol (HIP) schafft hier Abhilfe. Durch einen architekturellen Neuansatz fügt es einen so genannten "Host Identity Layer" in das bestehende ISO/OSI-Schichtenmodell ein. Identitäten sind kryptographisch gegen Fälschungen gesichert.

Alternative Ansätze für Mobilitätsmanagement im Internet

Mobile IPv6 könnte man als Standard für die Unterstützung von Mobilität im Internet der nächsten Generation bezeichnen und das Host Identity Protocol (HIP) vielleicht als seinen potentiellen Nachfolger in der übernächsten Generation. Daneben gibt es jedoch eine Reihe weiterer Protokolle auf unterschiedlichen OSI-Schichten, die Mobilität unterstützen können. Als standardisierte Lösung aus der Mobilfunkwelt wird GPRS-basiertes Mobililtätsmanagement beschrieben. Aufbauend auf der Fähigkeit des Schlüsselaustauschprotokolls IKEv2, Multi-homing zu unterstützen, wird das MOBIKE-Protokoll als alternative Grundlage für Mobilitätsmanagement betrachtet. Eine dritte Seminararbeit befasst sich mit der Möglichkeit, Mobilität in der Transportschicht zu unterstützen, basierend auf dem Stream Control Transmission Protocol (SCTP).

Paketkopfkomprimierung in drahtlosen Zugangsnetzen

Insbesondere in drahtlosen Zugangsnetzen, wo Bandbreite relativ teuer ist, spielt der Protokolloverhead eine entscheidende Rolle. Um Anwendungen wie Voice over IP, die überhaupt rentabel zu machen, muss ein effizientes Verfahren zur Paketkopfkomprimierung eingesetzt werden. Robust Header Compression (ROHC) ist ein solches Verfahren, dass sich im Gegensatz zu seinen Vorgängern speziell auch für fehleranfällige, langsame Funkverbindungen eignet. In drei Seminarbeiten werden die Funktionsweise, die Algorithmen und der notwendige Kontexttransfer für ROHC analysiert.

Inhaltsverzeichnis

Vorwort i

Rafael Sánchez-Moreno:
Destination-Sequenced Distance Vector (DSDV) 3

Wei Xing:
Ad-hoc On-Demand Distance Vector Routing 21

Johannes Güller:
Dynamic Source Routing 33

Djamal Guerniche:
Multicasting auf Anwendungsebene 47

Huiming Yu:
Unterstützung mobiler Netze mit Mobile IPv6 59

Liying Ren:
Predictive Configuration of a New Address in Mobile Ipv6 75

Iskander Louati:
HIP : eine Neue Internet-Architektur 89

Thorsten Grein:
Das Host-Identity-Protokoll: Das Grundprotokoll 99

Rim Helaoui:
Mobilität und multihoming 115

Jochen Furthmüller:
GPRS-basiertes Mobilitätsmanagement 127

Ruojing WEN:
MOBIKE-basiertes Mobilitätsmanagement 141

Philip Hoyer:
SCTP-basiertes Mobilitätsmanagement 153

Martin Walser:
Paketkopfkomprimierung in drahtlosen Zugangsnetzen 165

Maxim Feinleb:
Algorithmen der robusten Paketkopfkomprimierung 181

Andreas Schuster:
Kontexttransfer für robuste Paketkopfkomprimierung 199

Destination-Sequenced Distance Vector (DSDV)

Rafael Sánchez-Moreno

Kurzfassung

Die mobile Kommunikation findet in der heutigen Welt mehr Aufmerksamkeit. Geräte wie Laptops, Handys oder PDAs sind mittlerweile weit verbreitete Produkte. Die drahtlosen Netze finden somit langsam Einzug in den Alltag. Deswegen ist der Forschungsbereich der mobilen Ad-hoc-Netze (MANET) von hoher Wichtigkeit, der ein innovatives Design für den Betrieb von solchen Geräten in Netzen darstellt.

Es existieren Ad-Hoc Routingprotokolle, die in proaktive, reaktive und hybride unterschieden werden. Diese Ausarbeitung versucht die wichtigen Aspekte von proaktiven Routingprotokolle in Ad-hoc-Netzen zu beschreiben. Im ersten Abschnitt werden die Eigenschaften von Routingprotokollen dargestellt. Danach wird tiefer auf die proaktiven Protokolle eingegangen. Als Beispiel für proaktive Protokolle wird das Destination-Sequenced Distance-Vector Routing Protokoll (DSDV) näher erklärt. Dieses Protokoll stellt eine Verbesserung des für drahtgebundenen Netze entwickelten RIP Protokolls dar. Der Bellman-Ford Algorithmus wird für die drahtlose Übertragung angepasst, um zum Beispiel die Count-To-Infinity Problematik zu lösen. Eine weitere Eigenschaft, die vorgestellt wird, ist die Unterstützung von MAC-Adressen für das Routing. Im letzten Abschnitt wird ein kurzer Vergleich zwischen proaktiven, reaktiven und hybriden Protokollen gemacht.

1 Einleitung

Ad-hoc-Netze wurden für verschiedene Bereiche entwickelt wie zum Beispiel für die Kommunikation der Feuerwehr, der Polizei und der Armee im Einsatz. Vor allem Rettungsteams in Krisengebieten oder Soldaten im Kriegseinsatz haben große Vorteile bei der Nutzung von solchen Netzen gegenüber den gegenwärtigen Netzen. MANETs besitzen folgende besondere Eigenschaften:

- Infrastruktur: Diese Netze besitzen keine vordefinierte Infrastruktur. Jedes Gerät ist Endgerät und Router gleichzeitig.
- Mobilität: Die Knoten können sich in ständiger Bewegung befinden und trotzdem jederzeit miteinander kommunizieren. Es existieren keine Kabelverbindungen zwischen den Geräten.
- Verbindungsqualität: Die mobilen Knoten können sich jederzeit mit dem Netz verbinden oder sich vom Netz trennen. Dies passiert zum Beispiel, wenn sie nicht genügend Energie für die Datenübertragung besitzen oder wenn Störungen auftreten. Die Vermittlung von Daten erfolgt paketbasiert.
- Spontanität: Die Topologie des Netzes ändert sich ständig, dynamisch und spontan. Neue Knoten werden hinzugefügt bzw. alte Knoten vom Netz gelöscht.
- Autokonfiguration/Selbstorganisation: Den Knoten soll es möglich sein, sich in anderen Netzen anzumelden, ohne dass eine administrative Intervention stattfinden muss.

Alle möglichen Geräte können in Ad-hoc-Netzen als Knoten fungieren, egal ob es sich dabei um Handys, Laptops oder PDAs handelt. Da diese Knoten meistens auch keine große Funkreichweite haben, bedeutet das, dass sie nur einen kleinen Teil der anderen Knoten direkt erreichen können. Die anderen bleiben jedoch unbekannt. Um dennoch alle Knoten in so einem Netz erreichen zu können, werden Verfahren für die Routensuche (so genannte Routing-Protokolle) gebraucht. Dabei wird immer angenommen, dass die Knoten freiwillig mit diesen Protokollen Daten mit anderen Knoten austauschen. Die Herausforderung ist dann, eine drahtlose Mobilität und Konnektivität von mobilen Geräten zu unterstützen. In [Toh02] wird das Wort Ad-Hoc folgendermaßen definiert: „can take different forms“ oder „can be mobile, standalone or networked“.

Diese Geräte sollen andere Geräte erkennen, wobei sie gleichtzeitig auch den Typ und Eigenschaften identifizieren können. Rechenkapazität, Kommunikationsfähigkeiten, Speicherkapazität und Batteriedauer können stark vom Gerät zu Gerät variieren. Die Batteriedauer stellt eines der größten Probleme von drahtlosen Netzen dar. Deshalb müssen Routingprotokolle an diesen Eigenschaften angepasst werden.

Eine Herausforderung für Routingprotokolle in Ad-hoc-Netzen ist es, die Routensuche in einer sich ständig wechselnder Topologie zu ermitteln. Es existieren drei Eigenschaften, die diese Routingprotokolle erfüllen müssen [Schi00]. Sie müssen adaptiv, flexibel und effizient sein. Das Protokoll muss für verschiedene Netzwerktopologien einsetzbar sein und soll seine Metriken flexibel für verschiedene Applikationen optimieren. Das bedeutet eine geringe Belastung des Knotens mit einer geringen Latenzzeit. Dies sollte ohne administrative Intervention machbar sein. Schließlich sollen diese Protokolle eine bessere Performance erreichen als nicht adaptive Protokolle. Ein zusätzliches Problem stellt die Luft als Übertragungsmedium dar. Das Problem von versteckten bzw. ausgelieferten Geräte muss überwunden werden.

Die Kommunikation zwischen Knoten kann verschiede Formen aufweisen [Toh02]:

1. Zwei drahtlose Geräte in einem Ad-Hoc Netz. Die Kommunikation findet über eine bestimmte Zeitspanne statt, bis die aktuelle Sitzung beendet wird oder bis der Knoten sich bewegt und nicht mehr in Reichweite des anderen Knotens ist. Diese Situation spiegelt die Form der Peer-to-Peer Kommunikation wieder.
2. Zwei oder mehrere Geräte kommunizieren und wandern in Gruppen. Die Kommunikation findet über eine längere Zeitspanne statt. Diese Situation spiegelt die Form der Remote-to-Remote Kommunikation wieder.
3. Keine kohärente Kommunikation. Die Sitzungen sind kurz, abrupt und indeterministisch.

Das beschriebene Szenario, in dem mobile Geräte flexibel miteinander kommunizieren, ist nicht mehr weit von der Realität entfernt. Deswegen wird in den folgenden Abschnitten versucht das DSDV Protokoll als Lösung für die Wegewahlvermittlung vorzustellen.

2 Routingprotokolle

Protokolle im drahtlosen Bereich haben andere Anforderungen als Protokolle im Festnetzbereich. Das wichtigste Ziel im drahtlosen Bereich ist die Mobilitätsunterstützung. Die Verbindungen zwischen Knoten werden indeterministisch auf- und abgebaut. Existierende Protokolle sind nicht in der Lage diese Wechsel zu kontrollieren.

Protokolle im drahtlosen Bereich sollen schnell konvergieren und wenig Bandbreite für den Kontrollverkehr in Anspruch nehmen. Genau diese Anforderungen stellen ein nicht triviales

Problem dar, da traditionelle Routing Protokolle sehr langsam konvergieren. Andere Probleme im drahtlosen Bereich insbesondere von Ad-hoc-Netze sind:

- Hohe Leistungsaufnahme
- Geringe Bandbreite
- Hohe Fehlerrate

Genau deswegen sind neue Routingprotokolle notwendig um diese Problematik zu lösen. Die aktuellen Routingprotokolle werden in

- proaktive,
- reaktive und
- hybride

Protokolle eingeteilt. Im nächsten Kapitel werden die Eigenschaften dieser Protokollklassen dargestellt und als Beispiel für ein proaktives Protokoll wird das Destination-Sequenced Distance Vector Protokoll (DSDV) erläutert.

2.1 Proaktive Routingprotokolle

Ein Knoten, der ein proaktives Routingprotokoll als Basis benutzt, bestimmt fortlaufend Routen zu allen anderen Knoten im Netz. Dieser Knoten besitzt konsistente und aktuelle Routing Information von jedem Knoten zu jedem anderen Knoten im Netz. Diese Information wird in Routingtabellen gespeichert. Wenn Topologieänderungen im Netz erkannt werden, werden sie über das ganze Netz verbreitet.

Wenn ein Datenpaket gesendet wird, ist die ausgewählte Route schon aus der Routingtabelle bekannt und wird auch sofort benutzt. Wenn zwei Knoten eines Netzes miteinander kommunizieren wollen, müssen sie nur aus ihren gespeicherten Routingtabellen die notwendigen Informationen zur optimalen Weiterleitung der Daten ablesen und können ohne große Verzögerung mit der Datenübertragung beginnen. Diese Protokolle werden aufgeteilt in: Distance Vector Routing und Link State Routing.

Link State Routing

Dieses Verfahren basiert auf den Dijkstra's Shortest Path First Algorithmus [t01b]. Hier versendet jeder Knoten periodisch den Status seiner Links, zusammen mit der von Nachbarn empfangenen Information über den Status seiner Links. Somit kennt jeder Knoten die Gesamtnetztopologie mit den entsprechenden Linkkosten. Jeder Knoten versendet diese Kosten von ausgehenden Links zu allen Knoten mittels des Fluten-Algorithmus. Als Ergebnis aktualisiert jeder Knoten die Übersicht über das Netz. Das Problem bei solchen Verfahren ist, dass die Linkkosten veraltet sein können. Diese falsche Information würde sich über das gesamte Netz verbreiten.

Distance Vector Routing

Dieses Verfahren basiert auf dem Bellman-Ford-Algorithmus. In [t01a] wird der Algorithmus so definiert: „Der Bellman-Ford-Algorithmus ist ein modifizierter Dijkstra-Algorithmus, der genau wie der Algorithmus von Dijkstra auch zur Bestimmung der kürzesten Pfade in einem Graphen dient. Der Hauptunterschied zwischen den beiden Algorithmen besteht darin, das der Algorithmus von Bellman und Ford auch mit negativen Kantenbewertungen zurecht kommt."

Jeder Knoten ermittelt für jeden anderen Knoten im Netz den Nachbarn (auch Next Hop genannt), über den die kürzeste Route zu diesem Knoten besteht. Jeder Knoten kennt nur einen Teil der Topologie. Im Distributed Bellman Ford Algorithmus (DBF) sucht der Knoten i eine Route zum Zielknoten x über j:

$$d_{i,j}^{x}$$

Jeder Knoten besitzt einen Satz von Distanzen zu allen Nachbarn. Der Knoten j wird als Next Hop ausgewählt wenn:

$$min_j \left\{ d_{i,j}^{x} \right\}$$

Dieser Algorithmus hat mehrere Vorteile gegenüber anderen Verfahren. Er ist effizienter als Link State Algorithmen und einfach zu implementieren. Der Speicherbedarf im Knoten ist klein. Aber die Entwicklung von Loops (Schleifen) wirkt sich sehr problematisch aus.

Im Fall einer Entscheidung in der Routensuche kann es vorkommen, dass die vorhandene Information veraltet ist und die Entscheidung falsch getroffen werden kann. Man redet hier vom Count-to-infinity Problem. Es wurden viele Lösungsvorschläge diskutiert, leider sind diese Vorschläge nicht für hoch dynamische Netze gedacht.

Das beste Beispiel für ein Distance Vector Protokoll ist das RIP. In diesem Protokoll steht die Idee der Einfachkeit ganz oben. Die Lösung für das Count-to-Infinity Problem wird durch Poisoned-Reverse erreicht, indem alle gelernten oder empfangenen Routen als „nicht erreichbar“ gekennzeichnet und zurückgesendet werden. Das Problem von RIP ist, dass es nicht für dynamische Topologien gedacht ist. Poisoned-Reverse funktioniert hier nicht richtig wegen der schnellen Topologieänderungen. Als Lösung für Ad-hoc-Netze wurde das DSDV Protokoll entwickelt. Das Ziel dieser Routing Methode für Ad-hoc-Netze ist die Einfachkeit von RIP ohne das Looping Problem zu nutzen. Um dieses Problem zu lösen wird jeder Routingtabelleneintrag mit einer Sequenznummer gekennzeichnet, so dass alte Routen und neue Routen unterscheidbar sind und somit eine Vermeidung von Routing Loops gewährleistet werden kann.

2.2 Reaktive Routingprotokolle

Die so genannten On-Demand Routingprotokolle suchen, im Gegensatz zu den proaktiven Protokollen, die Route zwischen zwei Knoten nur nach Bedarf. Ein Knoten kennt nur die Routen, die er auch benötigt. Es erfolgen keine periodischen Aktualisierungen. Wenn eine Route gesucht wird, findet der „Route Discovery“ Prozess im Netzwerk statt. Sobald Daten für eine Station zum Versand anliegen, wird ein so genannter „Route Finder“ (Routensucher) im Netz gestartet. Wenn eine Route gefunden wurde und die Kommunikation etabliert wurde, wird sie vom Routingprotokoll aufrechterhalten, bis das Ziel über diese Route nicht mehr erreichbar ist oder die Route nicht mehr benötigt wird.

Einige Vorteile dieser Technik sind, dass der Verkehr (Netzlast) drastisch gesenkt werden kann und somit neue Ressourcen für die Datenübertragung zur Verfügung stehen. Nur benötigte Routen werden bestimmt und aufrecht erhalten. Mit dieser Technik kann auch Strom gespart werden, da die Geräte, die als Knoten in Ad-hoc-Netze miteinander kommunizieren keine aktuellen Routen zu anderen Geräten ermitteln müssen und kein periodisches Versenden von Nachrichten benötigt wird. Der größte Nachteil ist die Zeitverzögerung zu Beginn einer Kommunikation und der Kontrolloverhead abhängig von Anzahl der Verbindungen und Mobilität.

2.3 Hybride Routingprotokolle

Die hybriden Protokolle vereinigen Eigenschaften der proaktiven und reaktiven Protokolle. Die einzelnen Knoten tauschen Routing-Informationen über eine kleine abgegrenzte Zone (die so genannte Intra-Zone) aus, zeigen also in diesem Bereich proaktives Verhalten. Routen zu Knoten, die nicht in dieser Zone liegen, werden wie in einem reaktiven Protokoll erst auf Anfrage hin aufgebaut(Inter-Zone). Die Routensucheverzögerung wird durch den proaktiven Teil reduziert und dank des reaktiven Anteils findet eine kleinere Belastung der Ressourcen statt. Das Intra-Zone Routing ist proaktiv und wird durch eine Routingtabelle verwaltet. Das Inter-Zone Routing ist reaktiv und dient der Routensuche. Ein Route Request (RREQ) wird zu allen Knoten am Rand der Zone geleitet und wenn eine Route gefunden wird, wird ein Route Reply (RREP) zurückgesendet.

2.4 Beispiele

2.4.1 Ad-hoc On-demand Distance Vector Protocol

Das Ad-hoc On-demand Distance Vector (AODV) Protokoll ist ein **reaktives** Routingprotokoll. AODV baut auf das proaktive DSDV Protokoll auf. Bei diesem Protokoll wird aber die Anzahl der ausgetauschte Nachrichten minimiert um die Routingtabelle zu erstellen. Wir nehmen die Abbildung 1 als Beispielsnetz. Der Knoten MK_3 will mit dem Knoten MK_1 kommunizieren. Der Knoten MK_3 hat eine Routingtabelle, die aber nicht laufend aktualisiert wird. Es existiert kein Eintrag über Knoten MK_1. MK_3 weiß nicht, wo der Knoten MK_1 sich gerade befindet.

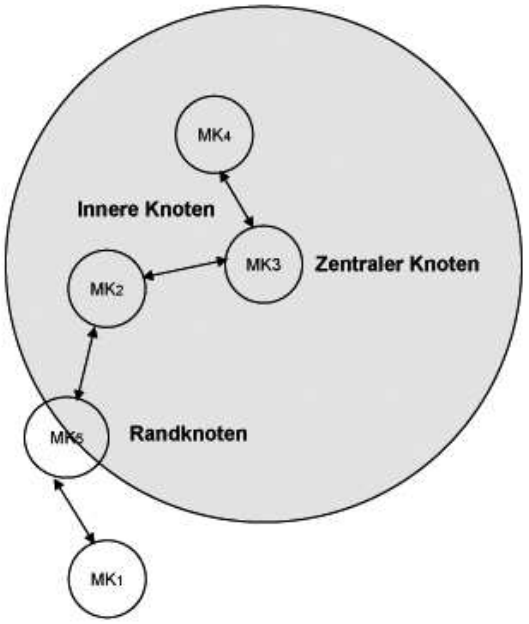


Abbildung 1: Ein Ad-hoc-Netz mit Radius = 2.

Um eine Route zum Zielknoten MK_1 zu finden, flutet AODV das Netz mit so genannten *Route Request* (RREQ) Nachrichten. Die Knoten MK_2 und MK_4 bekommen von Knoten MK_3 die Route Request Nachricht. Diese schicken diese Nachrichten wieder weiter an ihre Nachbarn (MK_2 nach MK_5) und speichern dabei die Zielroute. Dies wird weiter fortgesetzt, bis entweder der Zielknoten MK_1 erreicht ist oder ein Knoten auf dem Weg (zum Beispiel

der Knoten MK_5) eine Route zum Zielknoten in seiner Routingtabelle enthält. Eine *Route Reply* (RREP) Nachricht des Zielknotens MK_1 wird dann in Richtung Quellknoten MK_3 zurückgeschickt. Zur Adressierung der Knoten werden IP-Adressen eingesetzt.

Die Routingtabelle wird also nur bei einer Anforderung einer Route gefüllt. Wenn einer der Knoten sich bewegt oder Fehler bei der Übertragung auftreten wird der Mechanismus wieder angeworfen, welcher die Routen entdeckt.

2.4.2 Zone Routing Protocol

Das Zone Routing Protocol (ZRP) [HaPe02] ist ein **hybrides** Protkoll. ZRP bezeichnet das Framework, das aus **I**ntr**A**zone **R**outing **P**rotokoll (IARP), **I**nt**E**rzone **R**outing **P**rotokoll (IERP) und **B**ordercast **R**esolution **P**rotocol (BRP) gebildet wird. ZRP versucht die Vorteile aus proaktiven- und reaktiven Ansätzen zu verbinden.

Wir betrachten wieder das Netz mit fünf Knoten (Abbildung 1). Der Knoten MK_3 versucht wieder mit MK_1 zu kommunizieren. Das IntrAzone Routing Protocol (IARP) [HaPS02c] verwendet den Ansatz, dass die Routen von einem Knoten zu seinen Nachbarn in der Routingzone ständig aktuell gehalten werden.

Die Routingzone (definiert als Sammlung von Knoten, die mit einer festgelegten Anzahl von Hops zu erreichen sind) wird um den Knoten MK_3 realisiert, indem dieser seine periodisch ausgesendeten Updateinformationen seiner Verbindungen mit einer kleinen *Time-To-Live* Signal (TTL) versieht (mit TTL gleich 2). Der TTL-Wert gibt dabei gerade den Radius für die Routingzone an. Alle Knoten im Netz besitzen eine eigene Routingzone, das bedeutet auch, dass sich die Routingzonen von benachbarten Knoten überlappen können.

Die aufgezeichnete Routingzone gehört zum Knoten MK_3, den wir als zentralen Knoten (central node) der Routingzone bezeichnen. Die Knoten MK_4 und MK_2 sind innere Knoten (member nodes) der Routingzone. Der Knoten MK_1 ist drei Hops entfernt von MK_3 und deswegen außerhalb der Routingzone. Eine wichtige Teilmenge der Routingzone bilden die Knoten, bei denen der minimale Abstand zum zentralen Knoten gleich dem Radius der Zone ist. Diese Knoten werden Randknoten (peripheral nodes) genannt. In unserem Beispiel ist der Knoten MK_5 Randknoten von MK_3. Man muss immer beachten, dass die Zone keinen physikalischen Abstand beschreibt sondern die „Knotenkonnektivität“ (Hops).

Wenn eine Routenanfrage mit Hilfe der lokalen Informationen nicht gefunden werden kann, muss eine globale Suche gestartet werden. Um eine neue Route zu finden werden das IntErzone Routing Protocol (IERP) [HaPS02b] und das Bordercast Resolution Protocol (BRP) [HaPS02a] benutzt. Das IERP ist die reaktive Komponente des Zone Routing Protokolls. IERP ist zuständig für die Routensuche und die Wartung von entdeckten Routen zwischen den Knoten die sich außerhalb der Routingzone befinden. Eine Routenanfrage (Route Request) wird gesendet, wenn keine Route zum Ziel in der Routingzone verfügbar ist.

Jeder Knoten besitzt die Information über die Netzwerktopologie in seiner lokalen Umgebung, die so genannte *R-Hop-Nachbarschaft*. Diese von IARP gelieferte Information wird vom IERP ausgenutzt und somit werden lokale Anfragen eingespart. Wenn eine Anfrage benötigt wird, kann das Routingzone basierte Bordercast Protokoll genutzt werden. Dieses Protokoll verschickt die Anfrage zu den Randknoten einer Zone (MK_5), so erspart man sich das blinde Versenden von Nachrichten zu allen Nachbarn (Broadcast), zum Beispiel zu MK_4 und MK_2. Wenn die Route zu MK_1 gefunden wurde kann IERP diese Information benutzen um neue, verbesserte Routen zu berechnen. Redundante Routen werden auch erfasst. Der Vorteil besteht hier darin, dass wenn eine Unterbrechung der Verbindung zwischen MK_3 und MK_1 stattfindet, es sich nur um ein lokales Problem handelt. Mit Hilfe der redundanten Routen ist es unter Umständen möglich, dass diese Knoten diese Unterbrechung umgehen können falls

sie bereits über eine neue Verbindung verfügen. Mit ZRP kann man also sehr stabile und selbstreparierende Routen nutzen.

Das Bordercast Resolution Protocol (BRP) benutzt eine „Karte“ über eine verlängerte Routingzone, geliefert durch den proaktiven Anteil (IARP), um Bordercast-Bäume zu bilden. Über diese Bäume werden die Anfragen weiter geschickt. Bordercasting arbeitet viel effizienter als Broadcasting. Das Fluten mit Broadcast-Paketen ist hier nicht erforderlich, denn Knoten MK_3 weiß, ob sich der Knoten MK_1 in seiner Routingzone befindet. Wenn das nicht der Fall ist, reicht es aus, die Anfrage an den Randknoten (Bordernodes) MK_5 weiterzuleiten.

Das Bordercast Routing Protokoll legt fest, wie die Bordernodes ermittelt werden. Diese werden während der Aktivität des IntrAzone Routing Protocol (IARP) gefunden. Knoten MK_3 veröffentlicht seine Routingtabelle und versieht seine Pakete mit einem TTL-Wert gleich 2. Da dieser TTL-Wert gerade dem Radius der Routingzone entspricht, erkennen andere Knoten an dem TTL-Wert sobald er auf Null heruntergezählt und das Paket verworfen wurde, dass sie Bordernodes der Routingzone des Absenders des Paketes sind. Anschließend teilen sie der betreffenden Knoten mit, dass sie selbst Bordernodes ihrer Routingzone sind.

3 Destination Sequenced Distance Vector Protocol

Das Destination Sequenced Distance Vector (DSDV) Protokoll ist ein Vorschlag für das Routing in der drahtlosen Welt. Dieses Protokoll ist gut geeignet für Ad-hoc-Netze: keine Infrastruktur ist notwendig, eine dynamische Anpassung der Routingeinträge für die Informationsübertragung findet statt und wenn eine Basisstation verfügbar ist, dann ist dieses Protokoll auch kompatibel mit Routingprotokollen für drahtgebundene Netze. Eine zusätzliche Eigenschaft ist die Unterstützung der Schicht zwei des ISO/OSI Basisrefenzmodell für das Routing.

3.1 Eigenschaften

Das DSDV Protokoll stellt eine Verbesserung des Bellman Ford Algorithmus und des RIP Protokolls dar. Das Problem des RIP Protokolls ist, wie schon erwähnt, das Count-to-Infinity-Problem. Die Idee hier ist Routing Loops zu vermeiden. Um dieses Ziel zu erreichen werden Sequenznummer hinzugefügt und somit werden alte und neue Routen unterschieden. Jedes Paket enthält eine Sequenznummer und da Pakete unterschiedliche Routen nehmen, helfen diese Sequenznummer, die Pakete wieder in die richtige Reihenfolge zu bringen.

Die Dämpfung ist auch eine wichtige Eigenschaft dieses Protokolls. Die Daten werden nicht sofort weitergeleitet wenn eine vorübergehende Topologieänderung stattfindet und sie noch nicht stabil ist. Die Wartezeit ist die Differenz zwischen der ersten Pfadfindung und der besten Pfadfindung zu einem bestimmten Ziel. Diese werden verzögert gesendet, wobei noch nicht stabilisierte Routen nicht so schnell geflutet werden. Updates werden periodisch als Full Dump (alle Informationen) oder inkrementell (Änderungen nach Full Dump) gesendet. Diese Updates werden mittels Broadcast oder Multicast weitergeleitet. Die Metrik dieses Protokolls ist die Anzahl der Hops.

Die Einträge in der Routingtabelle eines Knotens ändern sich ständig wegen der Netztopologie. Deswegen werden Route Advertisements oft genug versendet, so dass alle Knoten fast alle anderen Knoten erreichen können.

3.2 Routingeinträge

Die Einträge, die weiter unten erklärt sind, werden in einer Routingtabelle gespeichert. In dieser Tabelle (siehe Tabelle 1) werden folgende Informationen gespeichert:

- die erreichbaren Zielknoten,
- die Anzahl der Hops zum Ziel,
- die Sequenznummer (vom Zielknoten erzeugt),
- die Installationszeit,
- die Flags und
- Stable-Data.

Ziel	Next Hop	Metrik	Sequenznummer	Install Time	Flags	Stable_data
MK_1	MK_2	2	$S406\_MK_1$	$T001\_MK_4$		$Ptr1\_MK_1$

Tabelle 1: Gespeicherte Routingtabelle

Als Ziel wird die Adresse des zu erreichbaren Knotens gespeichert. Die Schreibweise ist:

$$MK_i,$$

wobei i der Index des Knotens entspricht. Im Feld *Next Hop* steht der nächste direkt erreichbare Nachbar von der Quelle. Als *Metrik* wird der Abstand zwischen den Knoten benutzt. Die Schreibweise für die *Sequenznummern* ist:

$$SNNN\_MK_i,$$

wobei MK_i der Knoten ist, der die Sequenznummer erzeugt und SNNN ein Sequenznummerwert. Die *Installationszeit* ist die Zeit, zu der dieser Pfad das erste Mal gespeichert wurde. Sie legt fest, wann alte Routen gelöscht werden sollen. Das Löschen von alten Routen soll nicht zu oft stattfinden, da Übertragungsfehler sofort an andere Knoten weitergeleitet werden sollen. Dieses Feld zeigt auch die Verfügbarkeit von den Knoten für MK_i. Das Feld *Stable_data* gibt die Zeitspanne an, ab wann ein Eintrag als stabil angesehen werden kann. Hier stehen Zeiger, wenn es keine weiteren Routen im Netz gibt, die ersetzt werden sollten, das heißt es gibt keine Routen im Netz, die die aktuellen ersetzen könnten. Die Schreibweise ist:

$$Ptr1\_MK_i.$$

Im Feld *Flags* werden Änderungen bekannt gegeben. Die Tabelle 2 zeigt die Struktur der Routingtabelle, die weitergeleitet wird.

Ziel	Metrik	Sequenznummer
MK_1	2	$S406\_MK_1$

Tabelle 2: Gesendete Routingtabelle

Mit der gesendeten Routingtabelle wird auch die Hardwareadresse des Quellknotens gesendet. Die Sequenznummern sind für die Entscheidung der Weiterleitung von neueren Routen wichtig. Wenn zwei Pfade gleiche Sequenznummer besitzen, dann entscheidet die kleinste Metrik zwischen beiden Pfaden. Wenn ein Knoten neue Information empfängt, dann verschickt

er seine Routingtabelle und die empfangene Information (die Metrik wird um eins erhöht) weiter mittels Broadcast.

Die Zeit zwischen zwei Broadcasts kann problematisch werden. Wenn ein Knoten neue Information empfängt, dann wird er diese neue Information in der nächsten Aktualisierung mittels Broadcast weiterleiten. Das bedeutet, dass das DSDV Protokoll schnell konvergieren muss, sonst würde die Bandbreite des Mediums negativ beeinflusst.

3.3 Anpassung an Topologiewechsel

Wenn ein Knoten sich bewegt und den Ort wechselt, kann ein Link-Bruch stattfinden. Ein Link-Bruch soll auf Schicht-2 erkannt werden. Wenn das nicht der Fall ist, sollte ein Knoten einen Link-Bruch erkennen, wenn er keine neue Information vom betroffenen Knoten mehr bekommt. Die benutzte Metrik für einen Link-Bruch ist „∞“.

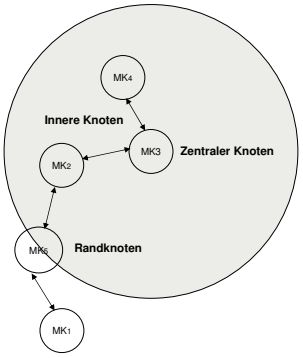


Abbildung 2: Ausschnitt eines Ad-hoc-Netzes.

Nehmen wir als Beispiel drei Knoten: A, B und C wie in Abbildung 2. A ist mit B und C verbunden. Der Link zwischen A und B bricht plötzlich ab (B hat sich bewegt und ist nicht mehr in der Reichweite von A). Alle Routen, die B als Next Hop in seiner Tabelle haben, bekommen ∞ als Metrik und eine neue Sequenznummer. Da diese Information für die Knoten wichtig ist, wird diese per Broadcast an allen Nachbarn weitergeleitet.

Diese neue Sequenznummer wird von irgendeinem Knoten erzeugt, im Gegenteil zur Erzeugung von Routingtabellen, wo nur der Quellknoten diese erzeugen kann. Das ist auch logisch, da der Quellknoten keine Verbindung mehr zu den anderen Knoten besitzt. Sequenznummern, die einen Link-Bruch aufweisen, sind ungerade Werte. Sequenznummern, die vom Quellknoten erzeugt werden, sind gerade Werte. Somit wird gesichert, dass gerade Sequenznummern bevorzugt werden, da es passieren kann, dass ein Knoten sich wieder in Reichweite befindet. Diese neue Information soll möglichst schnell an alle anderen weitergeleitet werden.

Wie schon erwähnt wurde, gibt es zwei Typen von Aktualisierungen: „Full Dump“ und „Incremental Dump“. Diese Entscheidung wurde getroffen um die Größe an weitergeleiteten Informationen klein zu halten. In einem *Full Dump* wird die gesamte Information gesendet. Diese benötigt mehrere NPDU's (Network Protocol Data Unit). In einem *Incremental Dump*

wird nur die Information gesendet, die seit dem letzten *Full Dump* geändert wurde. Diese Information passt normalerweise in ein NPDU.

3.4 Routenauswahl

Die Entscheidung, welche Route ausgewählt werden soll, besteht aus zwei Kriterien:

- die Sequenznummer und
- die Metrik (im Fall gleicher Sequenznummern).

Wenn ein Knoten neue Routinginformation empfängt, wird diese mit der vorhandenen verglichen. Es wird angenommen, dass der Knoten bereits eine Routingtabelle aufgebaut hat.

1. Wenn die neue Route eine neuere Sequenznummer als die vorhandene besitzt, wird diese neue Route benutzt und sofort per Broadcast weitergeleitet; die Metrik wird um eins erhöht.
2. Wenn die Route eine alte Sequenznummer aufweist, wird diese verworfen.
3. Wenn die neue Route die gleiche Sequenznummer wie der vorhandenen Route hat, entscheidet die bessere Metrik über die Routenauswahl.

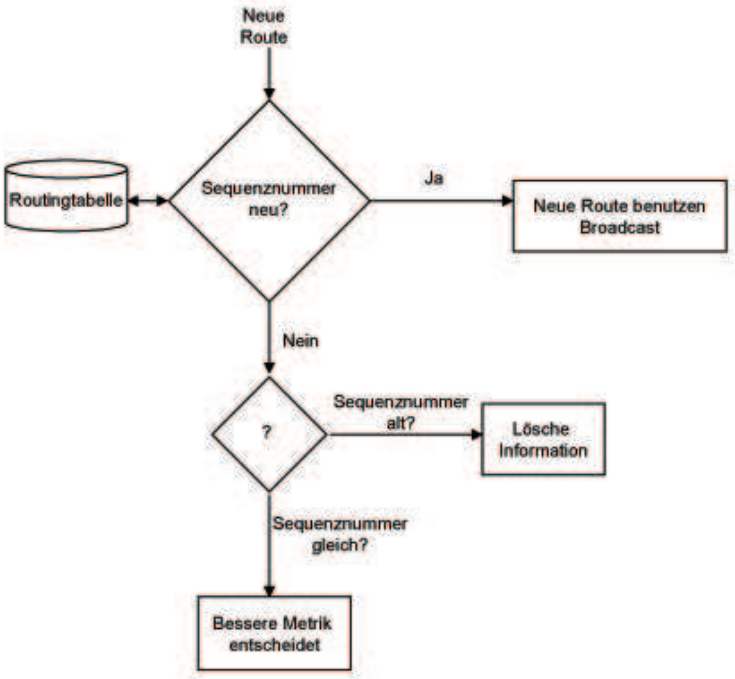


Abbildung 3: Entscheidungskriterien für die Wahl einer besseren Routinginformation.

Es kann auch passieren, dass ein Knoten zwei Routen zum selben Ziel bekommt. Wir nehmen als Beispiel die Abbildung 4. MK_1 empfängt zwei verschiedenen Routen (einmal über MK_2 und einmal über MK_3) zum Zielknoten MK_6 aber mit derselben Sequenznummer.

Für unser Beispiel machen wir folgenden Annahmen:

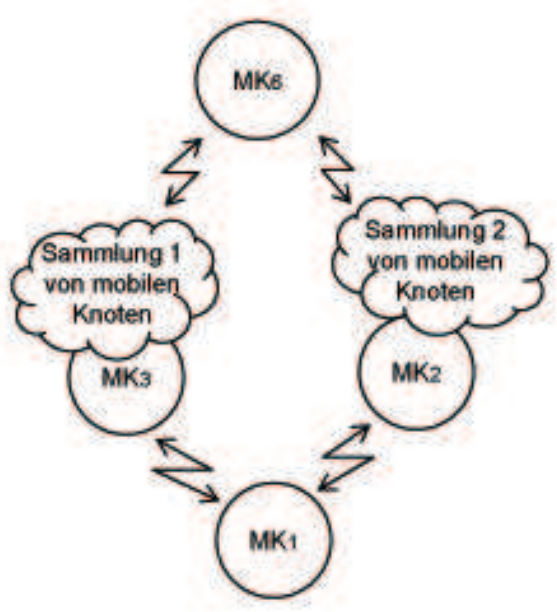


Abbildung 4: Bespiel eines Netzes mit Fluktuationen.

- Es gibt genügend Knoten im Netz um dieses Problem zu erzeugen.
- Es gibt zwei getrennte Sammlungen von Knoten mit nur dem gleichem Zielknoten (aus Sicht von MK_1).
- Die Aktualisierungszeit beträgt circa 15 Sekunden.
- Die Metrik für MK_6 über MK_2 beträgt 10 hops.
- Die Metrik für MK_6 über MK_3 beträgt 12 hops.
- Zusätzlich empfängt Knoten MK_1 vom Knoten MK_3 Informationen 10 Sekunden früher als die von Knoten MK_2.

So wie wir unser Netz aufgebaut haben, empfängt Knoten MK_1 die Routinginformationen zuerst von Knoten MK_3 und dann von Knoten MK_2. Es kann folgendes passieren:

1. Die neue Route hat eine bessere Metrik mit gleicher Sequenznummer.
2. Die neue Route hat eine schlechtere Metrik mit gleicher Sequenznummer.

Um eine Entscheidung zu treffen, wird die so genannte „Settling Time“ berechnet. Dieser Wert wird in einer Tabelle eingetragen (siehe Tabelle 3).

Zieladresse	Letzter Settling Time	Durchschnittliche Settling Time

Tabelle 3: Settling Time

Die *Settling Time* wird berechnet, indem man einen gewichteten Mittelwert von allen Aktualisierungen zu jedem Zielknoten speichert. Diese berechnete Zeit dient den Entscheidungen über die bessere Route. Neuere Aktualisierungen werden stärker gewichtet als ältere und ein

weiterer Parameter muss festgestellt werden: wie lang eine Route stabil sein soll, um sie als echt stabil bezeichnen zu können.

Im Fall, dass ein Knoten dieselbe Sequenznummer empfängt, aber mit einer schlechteren Metrik, muss er sie nicht sofort weiterleiten. Der Knoten wartet für die Zeit, die im Feld *Settling Time* eingetragen ist, bevor er die neue Routinginformation weiterleitet. Ein anderer Vorteil von dieser Zeit ist, dass im Fall eines Linkbruches die andere Route sofort benutzt werden kann. Das kann natürlich nur dann garantiert werden, wenn der Linkbruch schnell genug entdeckt wird.

Wir haben in diesem Beispiel gesehen, dass eine Route eine neuere oder gleiche Sequenznummer, aber eine schlechtere Metrik als eine andere Route besitzen kann. Die zweite Route kann eine ältere oder gleiche Sequenznummer, aber eine bessere Metrik besitzen. Wenn der Knoten sich für die Route mit schlechterer Metrik entscheiden würde und diese weiterleiten würde, könnte sich die schlechte Route durch das Netz fortpflanzen. Es kann auch passieren, dass Fluktuationen im Netz stattfinden. Der Knoten bewegt sich nicht, aber eine neue Routinginformation wird erzeugt. Deswegen wird die Weiterleitung von Information verzögert, wenn ein Knoten bemerkt, dass neue, bessere Information empfangen wird. Ein Knoten speichert für jede Route die gewichtete durchschnittliche Zeit zwischen dem ersten und dem besten Empfang von Routinginformation. So kann der Knoten berechnen, wie lang er warten soll, bis er die neue Routinginformation weiterleitet.

3.5 Schicht-2-Unterstützung

Auf Schicht 2 (Sicherungsschicht) werden MAC-Adressen, auf Schicht 3 (Vermittlungsschicht) IP-Adressen benötigt. Alle Routingprotokolle unterstützen Schicht-3-Adressen. Das DSDV Protokoll unterstützt zusätzlich auch Schicht-3-Adressen. Nichts desto trotz benötigt die Umwandlung von IP in MAC Adressen viel Bandbreite [t0194]. Die Lösung zu dieser Problematik ist die Schicht-2-Information in der zu sendenden Information auf Schicht 3 mit zu senden. Jeder Zielknoten verschickt zusätzlich zu der Routinginformation, welches Schicht-3-Protokoll er unterstützt. Diese Information wird in den Routingtabellen nur bei einer Änderung aktualisiert. Da manche Knoten viele Protokolle unterstützen, bleibt die Länge des reservierten Feldes variabel.

4 DSDV im Betrieb

In Abbildung 5 wird ein Ad-hoc-Netz mit fünf Knoten dargestellt. Wir betrachten dieses Netz aus Sicht von dem mobilen Knoten 4 (MK_4).

Die Tabelle für MK_4 sieht folgendermaßen aus:

Ziel	Next Hop	Metrik	Sequenznummer	Install Time	Flags	Stable_data
MK_1	MK_2	2	$S406\_MK_1$	$T001\_MK_4$		$Ptr1\_MK_1$
MK_2	MK_2	1	$S128\_MK_2$	$T001\_MK_4$		$Ptr1\_MK_2$
MK_3	MK_3	1	$S392\_MK_3$	$T002\_MK_4$		$Ptr1\_MK_3$
MK_4	MK_4	0	$S710\_MK_4$	$T001\_MK_4$		$Ptr1\_MK_4$
MK_5	MK_3	2	$S676\_MK_5$	$T001\_MK_4$		$Ptr1\_MK_5$

Tabelle 4: Routingtabelle für den mobilen Knoten MK_4

In der Tabelle wird die Information vor der Bewegung von MK_1 eingetragen. Einige Eigenschaften dieser Tabelle sind:

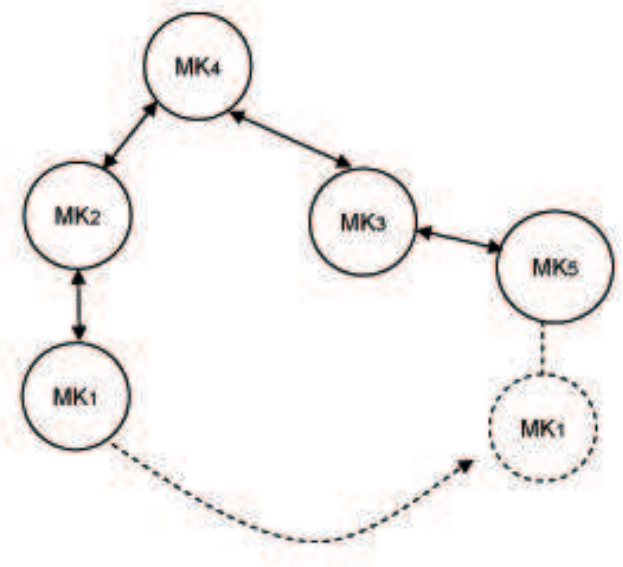


Abbildung 5: Beispiel eines Ad-hoc-Netzes mit 5 Knoten

- Der Knoten 4 hat alle Knoten fast zur gleichen Zeitpunkt gefunden, denn die Installationszeiten (Install Time) sind fast identisch.
- Es existieren keine Linkbrüche, da die Sequenznummern alle gerade sind.
- Es gibt keine Routen, die die aktuelle ersetzen könnten oder mit anderen um ein bestimmtes Ziel konkurrieren, da im Stable_data Feld nur Zeiger zu Nullstrukturen stehen.

Die Routingtabelle, die gesendet wird, sieht folgendermaßen aus:

Ziel	Metrik	Sequenznummer
MK_1	2	$S406\_MK_1$
MK_2	1	$S128\_MK_2$
MK_3	1	$S392\_MK_3$
MK_4	0	$S710\_MK_4$
MK_5	2	$S676\_MK_5$

Tabelle 5: Gesendete Routingtabelle

Nachdem die Information per Broadcast an allen Knoten weitergeleitet wurde, bewegt sich der Knoten MK_1 in die Reichweite von MK_5 und weg von MK_2 . Wenn MK_1 in die Nähe von MK_5 kommt, schickt er seine aktuelle Routinginformation zu MK_5 , der dasselbe mit MK_3 tut. MK_3 stellt fest, dass eine neue wichtige Routinginformation empfangen wurde und verschickt diese mit der aktuellen Information über MK_1 per Broadcast weiter.

Eine neue Routingtabelle (Tabelle 6) muss erstellt werden. In dieser neuen Tabelle können wir sehen, dass sich die Sequenznummer und die Metrik für MK_1 geändert haben. Die neue Information wird inkrementell verschickt bis der nächste Full Dump stattfindet. In der Tabelle 7 erscheint zuerst MK_4, da er die Aktualisierung durchführt. Dann erscheint MK_1, weil er eine Änderung aufweist. Die Regel besagt, dass Routen mit geänderter Metrik zuerst gesendet werden und der Rest der Metriken am Ende.

Wir haben in diesem Beispiel gesehen, wie ein Ortswechsel auf Sicht eines Knotens wahrgenommen wird. Ein Knoten hat ein Ortswechsel durchgeführt, was völlig normal in Ad-hoc-

Ziel	Next Hop	Metrik	Sequenznummer	Install Time	Flags	Stable_data
MK_1	MK_3	**3**	$S516\_MK_1$	$T810\_MK_4$	**M**	$Ptr1\_MK_1$
MK_2	MK_2	1	$S238\_MK_2$	$T001\_MK_4$		$Ptr1\_MK_2$
MK_3	MK_3	1	$S674\_MK_3$	$T002\_MK_4$		$Ptr1\_MK_3$
MK_4	MK_4	0	$S820\_MK_4$	$T001\_MK_4$		$Ptr1\_MK_4$
MK_5	MK_3	2	$S722\_MK_5$	$T001\_MK_4$		$Ptr1\_MK_5$

Tabelle 6: Aktualisierte Routingtabelle für den mobilen Knoten MK_4

Ziel	Metrik	Sequenznummer
MK_4	0	$S820\_MK_4$
MK_1	**3**	$S516\_MK_1$
MK_2	1	$S238\_MK_2$
MK_3	1	$S674\_MK_3$
MK_5	2	$S722\_MK_3$

Tabelle 7: Neu Gesendete Routingtabelle

Netzen ist. Die restlichen Knoten aktualisieren ihre Einträge und verschicken die aktualisierte Routinginformation weiter.

4.1 Schleifenfreie Pfade

Eine wichtige Eigenschaft des DSDV Protokoll ist die Schleifenfreiheit. Wir werden diese anhand unseres vorherigen Beispiels betrachten. Wir betrachten unser Netz mit 5 Knoten. Wir nehmen weiter an, dass alle Routingtabellen konvergiert haben. Es wird der Zielknoten X mit Graphen G(x) betrachtet. In unserem Beispiel ist MK_1 unser Zielknoten. Jeder Graphen G(MK_1) ist definiert durch die Knoten i und der nächste Nachbar von i für den Zielknoten x (p_i^x). Es werden disjunkte gerichtete Bäume aufgebaut, wo x die Wurzel ist. Das garantiert, dass zu jeder Zeit Schleifenfreiheit zugesichert ist.

Die Schleifenproblematik könnte aber auftreten, wenn der Knoten MK_1 seinen nächsten Nachbar wechselt (MK_2 auf MK_5). Es werden zwei Fälle betrachtet.

Im ersten Fall stellt Knoten MK_2 fest, dass sein Link zum nächsten Nachbar MK_1 abgebrochen wurde. In diesem Fall wird der Wert für p_2^1 zurückgesetzt und keine Schleifen treten auf.

Im zweiten Fall bekommt der Knoten MK_4 vom Knoten MK_3 eine neue Route zum Knoten MK_1 mit Sequenznummer S_3^1 ($S516\_MK_1$) und Metrik m= 3. Die neue Route ersetzt dann die alte. Wie in 3.4 schon erklärt, wird die neuere Sequenznummer vorgezogen oder die bessere Metrik, wenn die Sequenznummern gleich sind.

Im Fall einer neueren Sequenznummer $S_3^1 > S_2^1$ können keine Schleifen auftreten, da Knoten MK_1 nur dann die neue Sequenznummer verschickt, wenn diese Information von aktuellen nächsten Nachbar empfangen wurde. Da alle Knoten immer eine neuere Sequenznummer (als die empfangenen) benutzen, können keine Schleifen auftreten.

5 DSDV vs. andere Protokolle

Das drahtlose Übertragungsmedium hat einige Eigenschaften, die man bei der Entwicklung eines Protokolls beachten muss. Die Bandbreite ist eine knappe Ressource und Schleifen sind

nicht wünschenswert. Deswegen versucht das DSDV Protokoll diese Problematik zu lösen. Diese Schleifenfreiheit macht eine niedrige Speicheranforderung möglich und erreicht eine schnelle Konvergenz.

Da alle Routen proaktiv berechnet werden, kann eine Route aus der Routingtabelle sofort ausgewählt werden. Die gespeicherte Information in der Routingtabelle ist ähnlich zu heutigen Algorithmen, was ein Vorteil im Bereich Kompatibilität bedeutet. Die Implementierung von Schicht-2-Unterstützung ist auch ein wichtiger Vorteil dieses Protokolls. Die Sequenznummergenerierung durch den Zielknoten (mit der Ausnahme von Linkbrüchen) in jedem Routingeintrag dient einer besseren Verwaltung von Fluktuationen in Routing Updates.

Die sofortige Routenauswahl kann auch als ein Nachteil dieses Protokoll gesehen werden. Im hoch dynamischen Netze, kann eine komplette Übersicht der Topologie sehr lang dauern. Die zusätzliche Flusskontrolle für veraltete Routingeinträge wird auch als negativ gesehen. In einer Routingtabelle kann eine Route ständig abbrechen. Die Route wird repariert, obwohl keine Applikation sie benutzt. Das Ergebnis ist eine sinnlose Reparaturbelastung, die Bandbreite verringert sich und die Stauwahrscheinlichkeit im Medium steigt an. Dieser Effekt fällt gleich doppelt negativ ins Gewicht. Zum einem müssen die Routingeinträge gespeichert werden, das heißt, dass die Knoten des Ad-Hoc-Netzes Speicherplatz belegen, von dem nur ein geringer Anteil relevante Informationen enthält. Zum anderen werden viele Daten zwischen einzelnen Knoten ausgetauscht um Routen zu finden, die am Ende doch nicht gebraucht werden.

Um diese Problematik zu lösen wurden die reaktiven Protokolle entwickelt. Die Idee dahinter ist eine geringe Benutzung der verfügbaren Bandbreite für die Aufrechterhaltung von Routing Tabellen. Der Nachteil von solchen Protokollen ist die große Latenz, die Applikationen hinnehmen müssen. Eine große Verzögerung bei dem Routenaufbau findet statt, da diese zuerst angefragt werden müssen.

Ein Kompromiss in die richtige Richtung stellen die hybriden Protokolle dar. Sie verbinden reaktive und proaktive Ansätze und erreichen eine schnellere Latenz. Hybride Protokolle sind damit sehr effizient und skalierbar. Die Nutzung von Intra- bzw. Interzonen ermöglicht eine sehr stabile und selbstreparierende Netzwerkkapazität. Der Nachteil dieser Routingprotokolle ist, dass sie sich nicht so gut an die veränderten Netzwerktopologien angleichen lassen. Da die Routingzonen gleich groß sind, existieren nicht so viele Möglichkeiten zur Anpassung an die aktuelle Situation im Netz. Egal ob ein Knoten Datenverkehr produziert oder nicht, muss er ständig neue Routinginformationen von seinen Nachbarn anfordern. Meistens werden diese Routingzonen in der Hardware implementiert. Dies bedeutet, dass bevor das Netzwerk betriebsbereit wird, die Zonen festgelegt werden müssen.

6 Fazit

Welche Knoten sollen welche Routing Information besitzen/anfragen? Alle oder nur die gerade gebrauchten? Diese Kernfrage ist der Grund für die Weiterentwicklung von Protokollen für MANET's.

Das DSDV Protokoll gibt uns eine Antwort. Sie ist vielleicht nicht die beste Lösung für ein Ad-hoc-Netz, aber es ist sicherlich ein Schritt in die richtige Richtung. Mit der Benutzung von RIP als Basis wird die Speicherkapazität von mobilen Knoten geschont.

Trotzdem bleiben viele Fragen und Probleme auch weiterhin offen. Es ist nicht klar, wie andere Metriken behandelt werden. Die Knoten müssen selbst entscheiden, welche Änderungen signifikant genug sind und welche nicht. Wie das gemacht wird, ist fraglich. Ein Problem ist die Sicherheit im Netz. Eindeutige Adressen (sowohl MAC- als auch IP-Adressen) werden

gebraucht um das Netz zu sichern. Hierbei gilt die Funkübertragung als mögliche Angriffsquelle.

Obwohl Ad-Hoc-Netzwerke noch nicht vollständig implementiert worden sind, bleibt zu hoffen, dass bald eine globale Einführung dieser Technologien stattfinden wird. Das Ziel ein geringerer, schneller und genauer Zugriff auf Information, wenn diese gebraucht wird, ist mit der Zeit realistischer geworden. Denn sowohl im zivilen Bereich als auch im militärischen Bereich könnten diese drahtlosen Netze eine Revolution in der Kommunikation bedeuten. Wie [Toh02] es bezeichnen würde: „True information age style of civilization“.

Literatur

[HaPe02] Z.J. Haas und P. Pearlman, M.R.and Samar. Zone Routing Protocol (ZRP). Internet Draft, draft-ietf-manet-zrp-04.txt, Juli 2002.

[HaPS02a] Z.J. Haas, M.R. Pearlman und P. Samar. Bordercasting Routing Protocol (BRP). Internet Draft, draft-ietf-manet-brp-02.txt, Juli 2002.

[HaPS02b] Z.J. Haas, M.R. Pearlman und P. Samar. Interzone Routing Protocol (IERP). Internet Draft, draft-ietf-manet-ierp-02.txt, Juli 2002.

[HaPS02c] Z.J. Haas, M.R. Pearlman und P. Samar. Intrazone Routing Protocol (IARP). Internet Draft, draft-ietf-manet-iarp-02.txt, Juli 2002.

[Perk00] C. Perkins. *Ad-Hoc Networking*. Addison Wesley. 2000.

[Schi00] Jochen Schiller. *Mobilkommunikation*. Addison-Wesley. 2000.

[t01a] http://de.wikipedia.org/wiki/Bellman-Ford-Algorithmus.

[t01b] http://de.wikipedia.org/wiki/Dijkstras_Algorithmus.

[t01c] http://www.ietf.org/html.charters/manet-charter.html.

[t0194] Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, 1994.

[Toh02] C-K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall. 2002.

Abbildungsverzeichnis

1 Ein Ad-hoc-Netz mit Radius = 2. 7

2 Ausschnitt eines Ad-hoc-Netzes. 11

3 Entscheidungskriterien für die Wahl einer besseren Routinginformation. . . . 12

4 Bespiel eines Netzes mit Fluktuationen. 13

5 Beispiel eines Ad-hoc-Netzes mit 5 Knoten 15

Tabellenverzeichnis

1 Gespeicherte Routingtabelle 10

2 Gesendete Routingtabelle 10

3 Settling Time 13

4 Routingtabelle für den mobilen Knoten MK_4 14

5 Gesendete Routingtabelle 15

6 Aktualisierte Routingtabelle für den mobilen Knoten MK_4 16

7 Neu Gesendete Routingtabelle 16

Ad-hoc On-Demand Distance Vector Routing

Wei Xing

Kurzfassung

Aufgrund ihrer speziellen Charakteristiken, die durch drahtlose Kommunikation, sich häufig ändernde Netztopologie und den Bedarf nach effizienten dynamischen Routing-Protokollen gekennzeichnet sind, werden mobile Ad-hoc-Netzwerke(***MANETs***)[Secr05] mit „*Multihop*“[1]-Fähigkeit heutzutage auf den Gebieten der Forschung, des Militärs und sogar des täglichen Lebens weit eingesetzt. Ad-hoc-Netzwerke mit maximaler Bandbreite, minimalem Stromverbrauch und schleifenfreien Wegen sind erstrebenswert. Dabei spielen Routing-Protokolle eine wesentliche Rolle. In dieser Seminararbeit wird das Routing-Protokoll *AODV (Ad-hoc on demand Distance Vector)*, vorgestellt.

1 Einleitung

AODV wurde speziell für mobile Ad-hoc-Netze mit bis zu tausend mobilen Knoten konzipiert und dient als effiziente Lösung für eine schnelle Anpassung an die Netzwerktopologie. Wie bei anderen reaktiven Routing-Protokollen ist die Wegewahl bei *AODV* „bedarfsgesteuert“ (*on demand*). Allerdings muss nicht jedes Datenpaket von *AODV* die Routeninformationen von allen Nachbarknoten entlang der Route von der Quelle zum Ziel enthalten und während der Datentransmission mitnehmen. Stattdessen wird ein sogenannter „Rückwärtspfad“ (*reverse path*, Abschnitt 2.1.2) zum Zielknoten aufgebaut und als Routeneintrag temporär in der Routing-Tabelle gespeichert. Mit dieser speziellen Eigenschaft von *AODV* wird ein schnellerer Datentransport im Ad-hoc-Netzwerk erreicht, weil die Größe jedes Pakets dadurch verkleinert wird. Um schleifenfreie Wege zu erhalten, benutzt *AODV* die Zielsequenznummer (*destiantion sequence number*) wie bei üblichem Distanz-Vektor-Routing [HeUn88].

In den folgenden Abschnitten werden die allgemeinen sowie speziellen Eigenschaften von *AODV* im Vergleich mit anderen Protokollen des Ad-hoc-Netzwerks untersucht. Abschnitt 2 befasst sich mit dem Vorgehen der „Wegefindung“ (*route discovery*) und der Behandlung der Probleme anderer Routing-Protokolle in Ad-hoc-Netzwerken, die während der Routenermittlung eventuell auftauchen können. In Abschnitt 2.2.1 geht es dann um die Routenpflege bezüglich der Routing-Tabelle und des Pfades zwischen Quell- und Zielknoten. Dabei werden spezielle Mechanismen von *AODV* aufgezeigt. Mit diesen Mechanismen werden die durch Änderungen der Netzwerktopologie verursachten Proleme von Ad-hoc-Netzwerken mit einer großen Anzahl von Knoten gelöst. Zuletzt werden die charakteristischen Eigenschaften zweier „Konkurrenzprotokolle“ (*DSDV*[2] und *DSR* [3]) kurz beschrieben und damit eine Evaluation von *AODV* durchgeführt.

[1] Selbstorganisierende Funknetze, in denen jeder Knoten als ein potentieller Router betrachtet wird.
[2] *Destination-Sequenced Distance-Vector Routing*[PeBh]
[3] *Dynamic Source Routing*[JMCH04]

2 Routing-Verfahren von *AODV*

Beim proaktiven Routing, wie bei *DSDV*, tauschen ständig sämtliche Teilnehmer des Netzwerkes Routing-Informationen aus und verbrauchen damit unnötig Energie. Jeder Teilnehmer versucht, zu jeder Zeit einen optimalen Weg zu jedem anderen Teilnehmer im Netz bereitzuhalten, um schnelle Routenermittlung durchführen zu können. Als Nachteil ist anzusehen, dass die Routeninformationen stets als Einträge in der Routing-Tabelle gespeichert werden müssen, um Wege zu finden, obwohl nur eine sehr geringe Zahl der Einträge tatsächlich benutzt wird. Dies beschränkt die hauptsächlich für Nutzdaten zur Verfügung stehende Kapazität des Netzwerks. Als eine bessere, alternative Lösung ist die Routenermittlung von *AODV* zu sehen, wie sie im Abschnitt 2.1 beschrieben ist.

2.1 Routenermittlung in *AODV*

Die Routenermittlung verläuft ausschließlich bedarfsgesteuert und erfolgt nach einem Routen- anfrage/Routenbestätigungsprinzip. Routenanfragen werden mit Hilfe der RREQ-Nachrichten[4] publiziert. Informationen, die zur Bildung einer neuen Route beitragen, werden durch RREP[5]-Nachrichten verbreitet. Der generelle Ablauf des Routenermittlungsverfahrens ist folgendermaßen:

1. Wird eine Route von einem Quellknoten benötigt, sendet er per *Broadcast* eine Routenanfrage an seine Nachbarn.
2. Ein Rückwärtspfad wird automatisch temporär bei jedem Knoten gespeichert, immer wenn eine Routenanfrage bei ihm ankommt.
3. Jeder Knoten, einschließlich des Ziels selbst, kann eine Routenbestätigung an den anfragenden Knoten senden, wenn er eine gültige Route zum Ziel kennt.
4. Die Routeninformationen werden dezentral von jeden Knoten selbst verwaltet.
5. Alle Informationen, die durch eine Routenanfrage- oder Routenbestätigungsnachricht gewonnen wurden, werden mit weiteren Routing-Informationen in den entsprechenden Routing-Tabelle gespeichert.
6. Mit Hilfe der Sequenznummern werden veraltete Routen entdeckt und anschließend aus den Routing-Tabelle entfernt.

Die genaue Beschreibung eines solchen Ablaufes erfolgt in den nächsten Abschnitten.

2.1.1 Initiierung einer Routenanfrage

Wird eine Route von einem Quellknoten benötigt, schaut er zuerst in seiner Routing-Tabelle nach, ob ein gültiger Weg zum Zielknoten existiert. Sollte es einen solchen Weg nicht geben, initiiert der Quellknoten die Routenermittlung, indem er eine Routenanfrage (*RREQ*) mit der Zieladresse und der ihm letzten bekannten Zielsequenznummer per *Broadcast* abschickt. Eine Routenanfrage (*RREQ*) enthält die folgenden Felder: Quelladresse, Quellsequenznummer, Routenanfrage-ID, Zieladresse, Zielsequenznummer, *Hop-Count*. Abbildung 1 zeigt die Initiierung einer Routenanfrage. Es kann passieren, dass ein Knoten mehrmals wegen des

[4] *route request* [PeBR03], section 5.1
[5] *route reply*[PeBR03], section 5.2

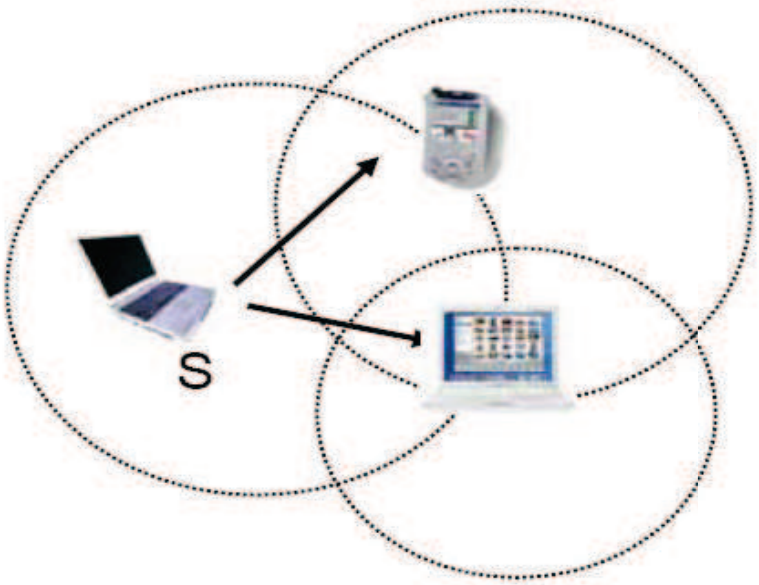


Abbildung 1: RREQ-Initiierung

Rundrufs der RREQ-Nachricht die gleiche Routenanfrage erhält. Wenn jeder Knoten immer die gleiche Routenanfrage weiterleiten würde, dann würde das zu einer Überlastung des Netzwerks führen. Um solche Überlastungen zu vermeiden und schleifenfreies Routing zu garantieren, enthält jeder Knoten zwei separate Zähler, nämlich die Sequenznummer und die Routenanfrage-ID.

Jeder einzelne Knoten pflegt intern in den dafür vorgesehenen Tabellen so genannte Sequenznummern. Es handelt sich dabei um eine monoton wachsende Integer-Zahl, mit der Netzwerkknoten auf die Aktualität ihrer Einträge in der Routing-Tabelle bzw. der empfangenen *AODV*-Nachrichten schließen können. Ein Knoten muss immer die ihm als neueste (höchste) bekannte Sequenznummer verwenden. Mit ihr wird das Fehlen eines zentralen Zeitgebers kompensiert. Dabei unterscheidet man zwischen Knoten- (seiner eigenen), Quell- und Zielsequenznummer. Die Quell- und Zielsequenznummern entnimmt der Knoten einfach aus den entsprechenden Feldern der *AODV*-Kontrollnachrichten (*RREQ, RREP, RERR* [6] usw.). Steht zum Beispiel in einer neuen RREQ-Nachricht zu einem bestimmten Ziel eine höhere Zielsequenznummer als die, die der Knoten in seinem Routing-Tabelle gespeichert hat, so aktualisiert er diesen Eintrag mit dem neuen, höheren Wert. Da immer nur die neuesten Sequenznummern verwendet werden, bleiben die Routen zum Ziel schleifenfrei. Das *AODV*-Protokoll funktioniert nur dann korrekt, wenn so verfahren wird. Ein Knoten muss seine eigene Sequenznummer erhöhen, bevor er

- als Quellknoten eine Routenanfrage erzeugt.
- als Zielknoten eine Routenbestätigung als Antwort auf eine Routenanfrage absetzt. Verliert ein Knoten durch einen Neustart seine Sequenznummer, so wartet er ab, ob er vielleicht von einem benachbarten Knoten eine Routenanforderung für sich erhält. Tritt dies ein, so ist die Sequenznummer aus der Anfrage höher als seine eigene. Deshalb kopiert der Knoten den Eintrag und zählt von nun an von diesem Wert ausgehend weiter. Anderenfalls initialisiert er seine Sequenznummer mit Null.

[6] *route error* [PeBR03], section 5.3

Die Sequenznummer eines Routeneintrags darf der Knoten ändern, wenn

- er selbst der Zielknoten ist und eine neue Route zu sich anbieten will,
- er eine *AODV*-Kontrollnachricht mit einer höheren Quellsequenznummer zu diesem Knoten empfangen hat oder
- der Pfad in Richtung des Ziels unterbrochen oder veraltet ist.

Ist eine Routenanfrage von einem Quellknoten gesendet, wird die Routenanfrage-ID um Eins erhöht. Das Paar „Routenanfrage-ID/Quelladresse“ dient als eindeutige Kennung einer Routenanfrage. Die RREQ-Nachricht enthält auch noch den Eintrag Distanz (*Hop Count*). Er gibt an, wie viele Teilstrecken das Paket bislang überwinden musste bzw. wie oft es schon weitergeleitet wurde.

Die RREQ-Nachricht wird solang im Netz weitergeleitet, bis eine gültige Route zum Ziel gefunden wird. Wie in Abschnitt 1 schon erwähnt wurde, muss jedes Patenpaket von *AODV* nicht wie bei *DSR* (ein Routing-Protokoll, das wie *AODV* auch zu den reaktiven Routing-Protokollen gehört) alle Routeninformationen von Nachbarknoten entlang der Route von der Quelle bis zum Ziel während der Datentransmission enthalten und mitnehmen. Stattdessen wird ein effizienter „Rückwärtspfad“-Mechanismus in *AODV* eingesetzt.

2.1.2 Rückwärtspfad (*reverse path*)[PeRo]

Bevor eine Routenanfrage während der Transmission behandelt wird, wird erst ein so genannter Rückwärtspfad bei jedem Knoten, bei dem die Anfrage ankommt, erzeugt, indem jeder Knoten die IP-Adresse seines Nachbarn, von dem er die erste Kopie der RREQ-Nachricht erhält, in die Routing-Tabelle aufnimmt. Ein solcher Eintrag bezüglich des Rückwärtspfades wird in der Routing-Tabelle eine bestimmte Zeit lang behalten. Dieser Vorgang wird in Abbildung 2 illustriert.

2.1.3 Behandlungen einer Routenanfrage

Wenn der Zielknoten bezüglich einer Routenanfrage überhaupt erreichbar ist, kommt diese Anfrage entweder direkt beim Zielknoten an oder landet auf den Knoten, der eine aktive Route bis zum Ziel kennt.

Ein Zwischenknoten empfängt von einer Quelle eine Routenanfrage für ein bestimmtes Ziel. Als erstes überprüft er, ob er das Paar „Routenanfrage-ID/Quelladresse“ schon in seiner Routing-Tabelle gespeichert hat. Dann erstellt er einen Rückwärtspfad (siehe Abschnitt 2.1.2).

Als nächstes überprüft er, ob er bereits eine ausreichend aktuelle und gültige Route zum Ziel kennt. Wenn nicht, erhöht er den „*Hop-Count*“ in der RREQ-Nachricht um Eins und leitet die Anfrage an seine Nachbarn weiter.

Wenn ein Zwischenknoten eine Route zum Ziel enthält, muss er zuerst durch Vergleich der in seiner Routing-Tabelle gespeicherten Zielsequenznummer und der in der angekommenen RREQ-Nachricht stehenden Zielsequenznummer feststellen, ob diese Route noch aktuell und gültig ist. Ist die Sequenznummer in der Nachricht größer als die Nummer in der Routing-Tabelle, dann braucht der Zwischenknoten nicht auf die Routenanfrage mit seinem Eintrag zu reagieren, sondern leitet sie weiter.

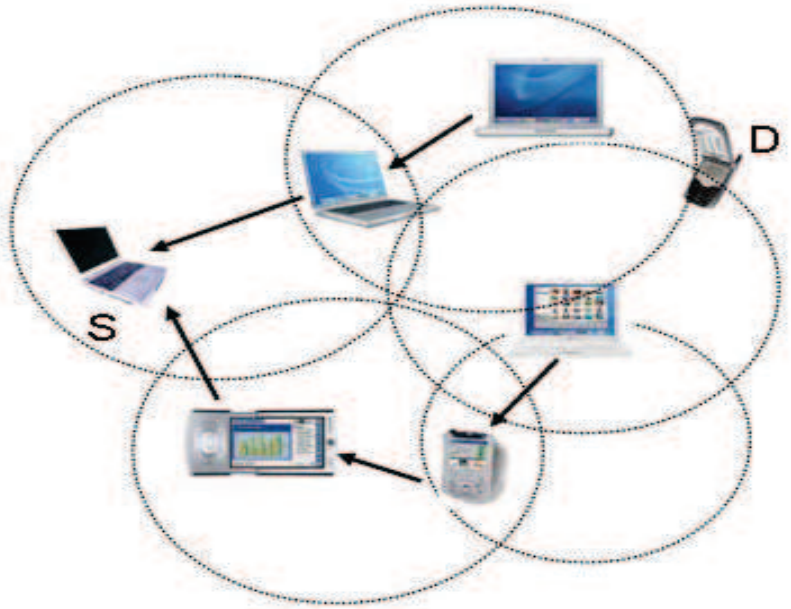


Abbildung 2: Rückwärtspfad

Nur wenn die Zielsequenznummer in der Routing-Tabelle größer oder gleich der Sequenznummer in RREQ ist, antwortet der Zwischenknoten per *unicast*[7] der Anfrage mit der RREP-Nachricht entlang des Rückwärtspfades. Eine RREP-Nachricht enthält folgende Felder: Quelladresse, Zieladresse, Zielsequenznummer, *Hop-Count*, Lebensspanne. Jeder Zwischenknoten (gegebenenfalls Zielknoten) entlang des Pfades, auf dem die RREP-Nachricht bis zum Ziel geleitet wird, aktualisiert die Lebensspanne für die Quell- und Zielknoten und speichert die letzte bekannte Zielsequenznummer ab. Während eine Routenbestätigung (*route reply*) zur Quelle zurückgesendet wird, wird der vorwärtsgerichtete Pfad zum Knoten, von dem aus die Routenbestätigung kommt, gebildet. Abbildung 3 präsentiert den Aufbau des Vorwärtspfades, während die RREP-Nachricht vom Zielknoten D zum Quellknoten S gesendet wird. Dabei kann gesehen werden, dass die Knoten, die sich nicht auf dem Weg der Routenbestätigung befinden, nach *ACTIVE-ROUTE-TIMEOUT(3000msec)*[PeRo] den Rückwärtspfad aus ihrer Routing-Tabelle löschen.

Ein Knoten leitet die erste angekommene Routenbestätigung zum Ziel weiter. Erhält er wieder die gleiche Routenbestätigung, aktualisiert er zuerst entsprechend der Kontrollnachrichten seine Routing-Tabelle. Eine weitere Routenbestätigung wird nur dann weitergeleitet, wenn sie eine größere Zielsequenznummer als die letzte oder aber die gleiche Sequenznummer aber eine kleinerer Distanz (*Hop Count*[8]) besitzt.

Wenn die Routenbestätigung schließlich bei der Quelle angekommen ist, muss sie nicht mehr weitergeleitet werden. Nachdem der Quellknoten in seiner Routing-Tabelle einen Eintrag für das Ziel erstellt hat, verwirft er die Routenbestätigung. Die Routenermittlungsphase ist damit abgeschlossen und die neue Route kann zum Senden von Nutzdaten verwendet werden.

[7] Punkt-zu-Punkt-Verbindung

[8] Gibt an wie viele Teilstrecken (*Hop Count*) ein Paket überwinden muss, bis es den Zielknoten erreicht. Erhält den Wert ∞ (unendlich), wenn für das Ziel ein Routenfehler (*RERR*) empfangen wurde.

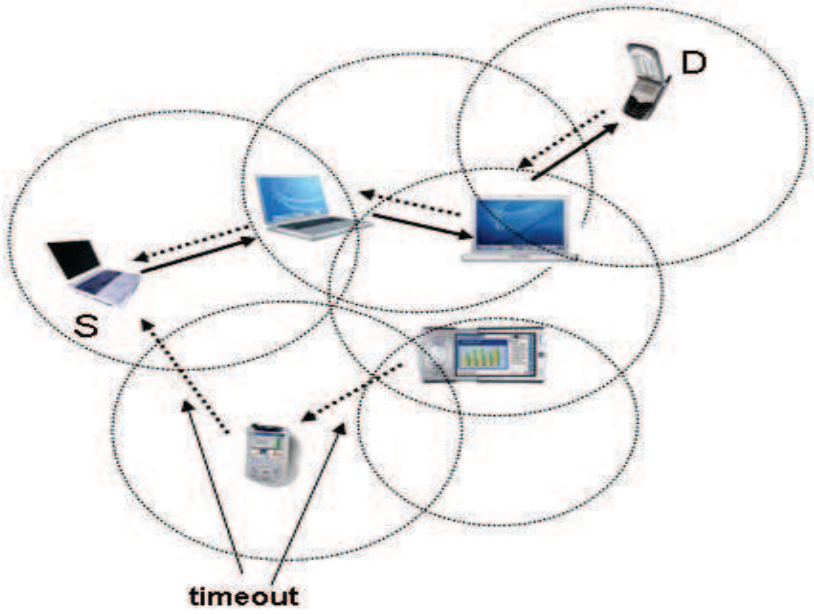


Abbildung 3: Vorwärtspfad

2.2 Management- und Pflegemechanismen in *AODV*

In diesem Abschnitt werden die in mobilen Ad-hoc-Netzwerken eventuell auftauchenden verschiedenen Probleme (z.B. das Aktivitäts- und Gültigkeitsproblem, das durch sich ändernde Netztopologie verursachte Problem usw.) mit speziellen Eigenschaften von *AODV* behandelt.

2.2.1 Management der Routing-Tabelle

AODV bietet verschiedene Mechanismen, um die Routeneinträgen in der Routing-Tabelle so zu verwalten, dass schnell festgestellt werden kann, ob z.B ein Rückwärtspfad gelöscht werden kann und ob ein Nachbarknoten oder ein Routeneintrag noch aktiv ist.

Die Routeneinträge in der Routing-Tabelle bestehen aus den unten angegebenen Feldern, deren Funktion erläutert wird:

- **Zielknoten (*Destination*)** Der Wert für den Zielknoten enthält die Adresse des Ziels, für das dieser Routeneintrag angelegt wurde.
- **Nächster Sprung (*Next Hop*)**: Für jedes Ziel wird im Feld *Next Hop* die Adresse des Nachbarn eingetragen, an den Datenpakete für das in Destination angegebene Ziel weitergeleitet werden sollen.
- **Anzahl der Stationen (*Metric*)**: Im Feld *Metric* wird eine Bewertung der Route gespeichert. Meist handelt es sich dabei um die Anzahl der Stationen (*Hops*) bis zum Ziel. Der *Metric*-Wert wird benötigt, um zwei Routen miteinander zu vergleichen, falls ihre Zielsequenznummer identisch sind.
- **Zielsequenznummer (*Sequence number for the destination*)**: Die Zielsequenznummer wird benötigt, um alte Routeninformationen von neueren zu unterscheiden. Neuere Routen haben größere Sequenznummern.
- **Aktive Nachbarn (*Active neighbors for this route*)**: Das Feld *Aktive Nachbarn* speichert eine Liste mit den Adressen aller Nachbarn, die diese Route nutzen. Diese Nachbarn sind zu informieren, falls die Verbindung unterbrochen wird.

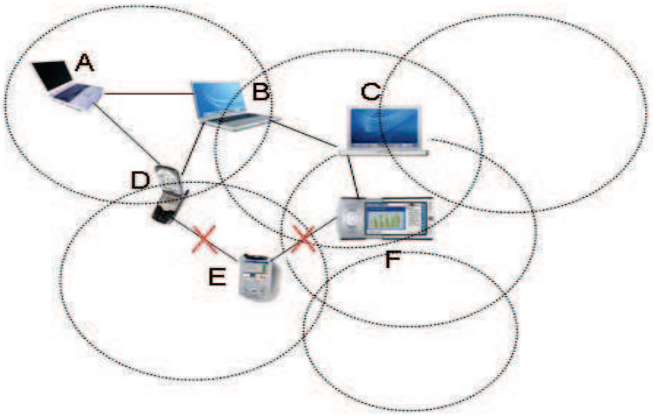


Abbildung 4: Verbindungsabbruch

- **Lebensspanne (*Expiration time for the route table entry*)**: Lebensspanne gibt den Zeitpunkt an, ab dem diese Route als nicht mehr aktuell angesehen wird. Wird dieser Weg genutzt, wird die Lebensspanne ständig neu gesetzt.

Es gibt zwei Arten des Schleifenproblems in Ad-hoc-Netzwerken, die durch Routenanfrage bewirkte Schleife und die durch ungültige Routeninformationen verursachte Schleife beim Distanz-Vektor-Routing. Das erste Prolem wurde schon in Abschnitt 2.1.1 erwähnt und kann mit dem Paar „Routenanfrage/Quelladresse“ von *AODV* gelöst werden. Zum zweiten Schleifenproblem wird bei *AODV* ein Mechanismus von *DSDV* übernommen, nämlich die Verwendung der Zielsequenznummern. Werden einem Knoten mögliche Routen übertragen, vergleicht dieser zunächst die Werte der Zielsequenznummer jeder Route und wählt stets den Eintrag mit der höheren Nummer, d.h. den neueren Eintrag. Sind die Sequenznummern beider Routen gleich, wird der Eintrag mit der besseren *Metric* gewählt. Dadurch sind die schleifenfreie Pfade garantiert.

2.2.2 „Wegewartung“

Durch die Bewegungen der mobilen Knoten im Ad-hoc-Netzwerk werden die angefragten Routen zum Zielknoten so beeinflusst, dass ein Datenpaket sein Ziel nicht mehr erreichen kann. Mit *AODV* wird das Problem in folgenden Fällen behandelt.

- Bei Bewegung von Knoten, die weder selbst mit einem anderen Knoten kommuniziert haben, noch Teil einer aktiven Verbindung waren, ist diese Veränderung nicht relevant und löst keinerlei Reaktionen aus.

- Falls ein Quellknoten sich bewegt, während die Routenanfrage von ihm zum Ziel weitergeleitet wird, kann der Quellknoten eine neue Routenermittlung starten, um eine neue Route zum Zielknoten zu erhalten.

- Geht es um die Bewegung von einem der Zwischen- oder Zielknoten, wird eine spezielle RREP-Nachricht zum betroffenen Quellknoten gesendet. Eine solche Nachricht wird auch als „*Hello Message*“ bezeichnet. Eine *Hello Message* dient auch zum Entdecken der Verbindungsabbrüche. Dies wird im Abschnitt 2.3 genauer erklärt.

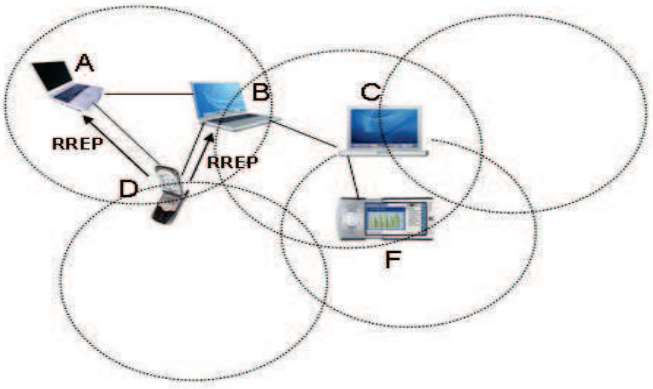


Abbildung 5: Behandlung nach dem Verbindungsabbruch

In den Abbildungen 4 und 5 ist das Szenario für die Behandlung der Verbindungsabbrüche aufgezeigt. Die Verbindung zwischen D und E bricht ab. D markiert den Pfad zu E in seiner Routing-Tabelle als ungültig (Distanz=∞) und überprüft welche Knoten als aktive Nachbarn im Routeneintrag zu E stehen. D erzeugt eine RREP-Nachricht[9] mit einer „frischen Sequenznummer“ (z.B. der bisher bekannten Sequenznummer + 1) und listet alle Ziele auf, die jetzt unerreichbar sind und sendet sie an die aktiven Nachbarn, die im Routeneintrag zu E stehen. Anschließend löscht er die Routen zu E. A und B empfangen RERR-Nachrichten von D und leiten sie an ihre Nachbarn weiter, bis alle aktiven Quellknoten diese empfangen. Haben alle Quellknoten keine weiteren Einträge zu E, löschen sie die Route zu E.

Nachdem ein Verbindungsabbruch ermittelt wurde, startet der Quellknoten eine neue Routenermittlung, indem er eine RREQ-Nachricht mit Zielsequenznummer+1 per *Broadcast* sendet, wenn er weiterhin eine Route zum Ziel braucht. Kein Nachbar wird auf diese Routenanfrage mit einer höhere Zielsequenznummer reagieren, wenn diese noch die alte Route für gültig halten. Somit kann keine Routing-Schleife existieren (siehe auch 2.2.1).

2.3 Lokales Verbindungsmanagement

Im letzten Abschnitt wird vorgestellt, wie ein Knoten auf die Änderungen der Netzwerktopologie reagiert. Zuletzt wird diskutiert, wie ein Knoten entlang einer bestimmten Route sicherstellen kann, dass die Verbindung zu seinen Nachbarn in Richtung des Ziels so lange erhalten und nutzbar bleibt wird, wie sie benötigt wird.

Die Überwachung kann auf zwei Arten erfolgen: vorbeugend (proaktiv) und bedarfsgesteuert (reaktiv). Im ersten Fall aktualisiert oder erzeugt der Knoten jedes Mal, wenn er eine Nachricht von einem bestimmten Knoten erhält, den entsprechenden Eintrag zum Quellknoten in seiner Routing-Tabelle. Dieser Eintrag hätte eine kurze Lebensspanne und hinge von der Zeitspanne ab, die er einem inaktiven Knoten zugesteht, bis er annimmt, dass die Verbindung abgebrochen ist. Solange der Nachbar erreichbar ist, bliebe der entsprechende Eintrag in der Routing-Tabelle gültig. Andererseits würde der Knoten, wenn die Lebensspanne des Eintrags abgelaufen ist, vermuten, dass die Verbindung nicht mehr besteht und sofort eine Routenfehlermeldung erzeugen.

Bei einem inaktiven Knoten, der z.B. längere Zeit nichts gesendet hat, lässt sich auch nicht auf seine Erreichbarkeit schließen. Aus diesem Grunde gibt es beim *AODV*-Protokoll die

[9] route Error-Nachricht [PeBR03]

so genannte „Hallo-Nachricht“(*hello message*), die in gewissen Zeitabständen per Broadcast verschickt wird. Wenn ein Knoten schon längere Zeit keine Datenpakete verschickt hat, zeigt er seine Präsenz, indem er eine spezielle Routenbestätigung mit sich selbst als Zielknoten, seiner aktuellen Zielfolgennummer, einem Distanzwert (*Hop*) von Null und einer Gültigkeitsdauer von Eins an seine Nachbarn schickt. Die Hallo-Nachricht wurde in das *AODV*-Protokoll aufgenommen, damit es unabhängig von den darunterliegenden Netzwerkschichten wurde.[10] Lokales Verbindungsmanagement kann deshalb auch nur mittels Hallo-Nachrichten betrieben und die darunterliegenden Netzwerkschichten können so gänzlich ignoriert werden.

Die Alternative zu diesem Verfahren würde bedarfsgesteuert (reaktiv) arbeiten. Das heißt, Verbindungsabbrüche könnten nur durch Übertragungsfehler festgestellt werden; in diesem Fall auf der MAC-Teilschicht der Sicherungsschicht (OSI-Schicht 2a (MAC)[Tear99]).

3 Evaluation

AODV ist als eine Verbesserung von *DSDV* und *DSR* bezüglich ihrer Nachteile. In diesem Abschnitt werden zuerst die charakteristischen Eigenschaften der „Konkurrenzprotokolle“ *DSDV* und *DSR* kurz beschrieben. Danach wird eine Evaluation von *AODV* durch einen Vergleich mit *DSDV* und *DSR* durchgeführt. Hierbei sind z.B Latenz, Stromverbrauch, Speicherplatz, Bandbreite und Netzwerksgröße die wesentlichen Vergleichsfaktoren.

3.1 *Destination Sequenced Distance Vector(DSDV)*

DSDV ist ein proaktives Routing-Protokoll, bei dem ein periodischer Broadcast zur Aktualisierung der Routeninformationen benötigt wird. Jeder Eintrag wird zusammen mit einer ursprünglich vom Ziel generierten Sequenznummer gespeichert, die es jeder Station ermöglicht, neuere Informationen zur Wegewahl von alten zu unterscheiden. Damit wird die Schleifenfreiheit im Ad-hoc-Netzwerk garantiert. Jeder Knoten des Ad-Hoc-Netzwerkes speichert sich in einer Routing-Tabelle einen Eintrag für jedes erreichbare Ziel. Diese Routing-Informationen teilt jeder Teilnehmer seinen Nachbarn in regelmäßigen Abständen mit. Besonders wichtige Veränderungen der Routeneinträge werden sofort per *Broadcast* allen Nachbarn mitgeteilt. Da die häufigen netzweiten Rundrufe die Größe des Netzwerks beschränken, ist *DSDV* effizient für Ad-hoc-Netzwerke mit einer kleinen Anzahl von mobilen Knoten.

3.2 *Dynamic Source Routing Protocol(DSR)*

DSR ist ein reaktives Routing-Protokoll, das auf die ständige Pflege von Wegen zwischen allen Knoten im Netzwerk verzichtet. Stattdessen werden Wege nur gesucht, falls Daten zu übertragen sind (*on demand*). Jeder Knoten im Netz kennt seinen besten Weg. Dieser Weg wird dann so lange wie mögliche genutzt, bis ein Problem damit auftritt. In einem solchen Fall muss ein neuer Weg aufgebaut werden. Beim *DSR* gibt der Quellknoten den gesamten Weg für ein Datenpaket vor. Im Paketkopf jedes Pakets wird eine Liste von Knoten angegeben, über die der Weg zum Ziel führt. In den meisten Fällen kann eine Route aus dem Cache des jeweiligen Knoten verwendet werden, die durch frühere Wegewahl bekannt ist. Es können bei *DSR* mehrere Routen zu einem Ziel abgelegt werden.

[10] Eine alternative Lösung mit wenigere Latenz in der Sicherungsschicht zur Entdeckung Verbindungsabbruch mit der Hilfe von *„Link-layer acknowledgments“(LLACKS)* [PeRo]

Bei *DSR* prüft ein Zwischenknoten nach Erhalt einer RREQ erst in seinem Routing-Cache nach, ob schon eine aktive Route zum Ziel gespeichert ist. Falls vorhanden, wird ein RREP-Paket inklusiv der korrekten Distanz (*Hop Counts*) für den Zielknoten zur Quelle zurückgesendet. Falls kein Eintrag vorhanden ist, fügt der Zwischenknoten seine eigene Adresse in den Broadcast-Aufruf ein, leitet die RREQ-Nachricht weiter und puffert die RREQ-Nachricht im Cache.

3.3 Vergleich *DSDV / AODV*

Beim *DSDV*-Protokoll erfolgt eine Routenermittlung schneller und einfacher als beim *AODV*-Protokoll, da die eventuell gesuchte Routen vor dem Bedarf schon in der Routing-Tabelle zur Verfügung stehen. Mit dem RREQ/RREP-Prinzip hat *AODV* gegenüber *DSDV* eine größere Latenz. Das periodische Aktualisieren des ganzen Netzes mit Routeninformationen bei *DSDV* führt zu hohem Netzwerkverkehr auch bei unveränderter Netztopologie. Zudem werden nie genutzte Routen ebenfalls aktualisiert. Dadurch geht Strom vom Endgerät und für Nutzdaten bereitstehende Bandbreite verloren.

Im Bezug darauf stellt *AODV* eine Verbesserung von *DSDV* dar, da bei diesem Protokoll die Anzahl der Nachrichten minimiert wird, um die Routing-Tabelle zu erstellen. Auch hier hat jeder Knoten eine Routing-Tabelle, nur wird diese nicht laufend aktualisiert. Wenn ein Knoten eine Route benötigt, schickt er an seine Nachbarn ein Paket, welches als RREQ bezeichnet wird. Diese schicken es wieder weiter an ihre Nachbarn und so weiter bis entweder der Zielknoten erreicht ist oder ein Knoten auf dem Weg eine Route zum Zielknoten in seiner Routing-Tabelle enthält. Anschließend wird der Quellknoten benachrichtigt, dass eine Route gefunden wurde und wie diese aussieht. Die Routing-Tabelle wird also nur bei einer Anforderung einer Route gefüllt. Wenn einer der Knoten sich bewegt oder Fehler bei der Übertragung auftreten wird der Mechanismus wieder angeworfen, welcher die Routen entdeckt.

3.4 Vergleich *AODV / DSR*

DSR und *AODV* sind reaktive Routing-Protokolle. Routen werden nur bei Bedarf ermittelt. Die Routenermittlung von beiden Protokollen basiert auf dem Routenanfrage-/Routenbestätigungsprinzip. Routing-Informationen werden auch in Zwischenknoten entlang der Route in der Form von Routing-Tabelleneinträgen (*AODV*) oder im Routing-Cache (*DSR*) gespeichert. Trotz der Gemeinsamkeiten gibt es einige wichtige Unterschiede der beiden Protokolle, die bedeutende Leistungsunterschiede verursachen können.

Erstens kann z.B. ein Quell- oder ein Zwischenknoten bei *DSR* mehrere alternative Routen speichern, da *DSR* auf alle Routenanfrage antwortet. Diese alternativen Routen werden sehr nützlich, wenn die optimale (kürzeste) Route ausgefallen ist. Um alternative Routen abrufen zu können, müssen auch viele verschiedene Routeninformationen für ein gleiches Ziel zwischengespeichert werden. Dadurch kann einerseits eine angefragte Route sehr schnell ermittelt werden, andererseits kann der Netzwerkverkehr durch mehrere Routenbestätigungen belastet werden. Bei *AODV* reagiert ein Zielknoten dagegen nur einmal auf die erste angekommene Routenanfrage und ignoriert die übrigen Anfragen. In der Routing-Tabelle wird höchstens ein Eintrag pro Ziel gepflegt.

Zweitens gibt es bei der gegenwärtigen Spezifikation von *DSR* noch keine Mechanismen, um veraltete Routen im Cache zu erkennen, oder um eine aktuellere Route aus mehreren Möglichkeiten zu erkennen. Beim Verwenden veralteter Routen besteht die Gefahr, dass die veralteten Routen die Caches anderen Knoten „verschmutzen“ ([MBJJ99]) und unkorrekte Routenbestätigungen bezüglich der Routenanfragen geliefert werden. Im Gegensatz dazu wird

bei *AODV* eine aktuellere Route, die auf Zielsequenznummern basiert, immer ausgewählt. Wenn ein Routeneintrag innerhalb seiner Lebensspanne nicht benutzt wird, wird der Eintrag in der Routing-Tabelle als ungültig markiert. Allerdings kann ein gültiger Routeneintrag dadurch auch ungültig werden. Eine passende Lebensspanne lässt sich schwer bestimmen, da die Sendegeschwindigkeit von verschiedenen Zielknoten sowie die Knotenmobilität sich sehr unterscheiden und dynamisch ändern kann.

Zuletzt werden bei AODV alle betroffenen, aktiven Nachbarn durch eine RERR-Nachricht informiert, wenn ein Verbindungsabbruch auf dem Weg von der Quelle zum Ziel entdeckt wird. In *DSR* verfolgt eine RERR-Nachricht jedoch denselben Weg des Datenpaketes zurück, auf dem das Datenpaket wegen eines Verbindungsabbruchs nicht weitergeleitet werden konnte. Das deutet an, dass die Nachbarn, die sich nicht auf dem Weg des Datenpaketes befinden, nicht durch die RERR-Nachricht über den Verbindungsabbruch informiert werden, obwohl sie auch vom Verbindungsabbruch betroffen sind.

Im Vergleich mit *AODV* eignet sich *DSR* besser für ein weniger „stressiges“ Umfeld mit geringerer Knotenzahl, geringerer Netzauslastung und geringerer Mobilität der Knoten. *AODV* meistert hingegen das „stressigere“ Umfeld besser mit höherer Knotenzahl, höherer Netzauslastung und höherer Mobilität der Knoten.

4 Zusammenfassung

In den vorhergehenden Abschnitten wurde das elegante und leistungsfähige *AODV*-Protokoll detailliert vorgestellt. Seine Verfahren und Lösungskonzepte wurden ausführlich beschrieben. Zuletzt wurde *AODV* mit *DSDV* und *DSR* verglichen. Es gibt zahlreiche Ansätze zu Verbesserungen von *AODV*, wie z.B. Multicast und Elimination der Hallo-Nachricht u.ä., auf deren nähere Erläuterung in dieser grundlegenden Beschreibung von AODV verzichtet wurde.

Literatur

[HeUn88] C. Hedrick und Rutgers University. *RFC 1058 - Routing Information Protocol*. IETF. 1988.

[JMCH04] David B. Johnson, David A. Maltz, Andrew Campbell und Yih-Chun Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*. IETF-Network Working Group. 2004.

[MBJJ99] David A. Maltz, Josh Broch, Jorjeta Jetcheva und David B. Johnson. *The Effects of On-demand Behavior in Routing Protocols for Multihop Wireless Ad Hoc Networks*. IEEE JSAC, vol. 17, no. 8. 1999.

[PeBh] Charles E. Perkins und Pravin Bhagwat. *Highly Dynamic Destination-Sequenced Distance-Vector Routing*. New York.

[PeBR03] C. Perkins und E. Belding-Royer. *RFC 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing*. IETF-Network Working Group. 2003.

[PeRo] Charles E. Perkins und Elizabeth M. Royer. Ad hoc On Demand Distance Vector Routing.

[Secr05] IETF Secretariat. *Mobile Ad-hoc Networks (manet)*. IETF. 2005.

[Tear99] Diane Teare. Introduction to Internet. 1999.

Abbildungsverzeichnis

1 RREQ-Initiierung 23

2 Rückwärtspfad 25

3 Vorwärtspfad 26

4 Verbindungsabbruch 27

5 Behandlung nach dem Verbindungsabbruch 28

Dynamic Source Routing

Johannes Güller

Kurzfassung

Ad-hoc-Netzwerke sind Ansammlungen mobiler Knoten, die drahtlos und ohne stationäre Infrastruktur oder zentrale Administration miteinander kommunizieren. Aufgrund der begrenzten Sendereichweite der Knoten, ist es notwendig, dass diese über mehrere Zwischenknoten mit weiter entfernten Knoten kommunizieren können. Durch Bewegung der Knoten ergeben sich ständig Veränderungen in der Netztopologie, weshalb ein Routingprotokoll für Netzwerke solcher Art in der Lage sein sollte, schnell auf diese Änderungen reagieren zu können. Nur so ist ein ausreichender Datenfluss und möglichst wenig Paketverluste gewährleistet. Das Dynamic Source Routing Protokoll (DSR) ist ein Routingprotokoll für Ad-hoc-Netzwerke und verspricht Einfachheit und einen geringen *Overhead* durch Routingpakete. DSR stellt Mechanismen bereit, die es den einzelnen Knoten des Netzwerks erlauben, Routen zu beliebigen Zielknoten zu finden und aufrecht zu erhalten. DSR gehört zu den reaktiven Routingprotokollen. Folglich werden die Mechanismen zum Auffinden und zur Überwachung von Routen nur bei Bedarf ausgelöst, was DSR mit steigender Knotenanzahl skalieren lässt. Simulationen und reale Testläufe haben dies bestätigt.

1 Einleitung

Das Dynamic Source Routing Protokoll (DSR) ist ein einfaches und effizientes Routingprotokoll, das auf den Einsatz in drahtlosen Ad-Hoc-Netzwerken zugeschnitten ist. Durch seine selbstorganisierende und selbstkonfigurierende Struktur ist keine feste Netzwerkinfrastruktur oder -administration notwendig. Es unterscheidet sich hauptsächlich dadurch von anderen Protokollen, dass es keine periodischen Aktualisierungen seiner Routinginformationen benötigt. Routinginformationen werden nur nach Bedarf ausgetauscht. Wenn keine Daten zu übertragen sind oder keine Bewegungen der Knoten stattfinden, findet auch keinerlei Austausch von Routinginformationen statt. Jeder Knoten des Ad-hoc-Netzes ist mobil und könnte jederzeit ohne Vorankündigung seinen Standort wechseln und dadurch die Netzwerktopologie unter Umständen stark verändern. Zusätzlich erzwingt die Mobilität der Knoten eine beschränkte Sendereichweite, wegen des begrenzten Energievorrats oder der geringen Gerätegröße. Somit ist es unumgänglich, dass viele Knoten nicht direkt miteinander kommunizieren können. Dieses Problem wird in Abbildung 1 deutlich. Ein Knoten **C** befindet sich außerhalb der Reichweite von Knoten **A**. Eine Datenübertragung zwischen **A** und **C** ist somit nur über den Knoten **B** möglich, der in beider Reichweite liegt. Knoten **A** benötigt zunächst die Information, dass das Paket über **B** an **C** weitergeleitet werden kann. Sollte die Route zwischen **A** und **C** irgendwann unterbrochen werden, muss **A** über diesen Zustand informiert werden, da andernfalls **A** weiterhin versucht Pakete über die defekte Route an C zu senden. Dadurch sind die beiden essentiellen Aufgaben von DSR grob umrissen. DSR sollte automatisch auf die sich möglicherweise ständig ändernde Netztopologie schnell reagieren und die Routinginformationen der Knoten dynamisch anpassen können. Änderungen in der Netztopologie entstehen durch die Bewegung einzelner oder mehrerer Knoten (s. Abbildung 2). Auswirkungen dieser

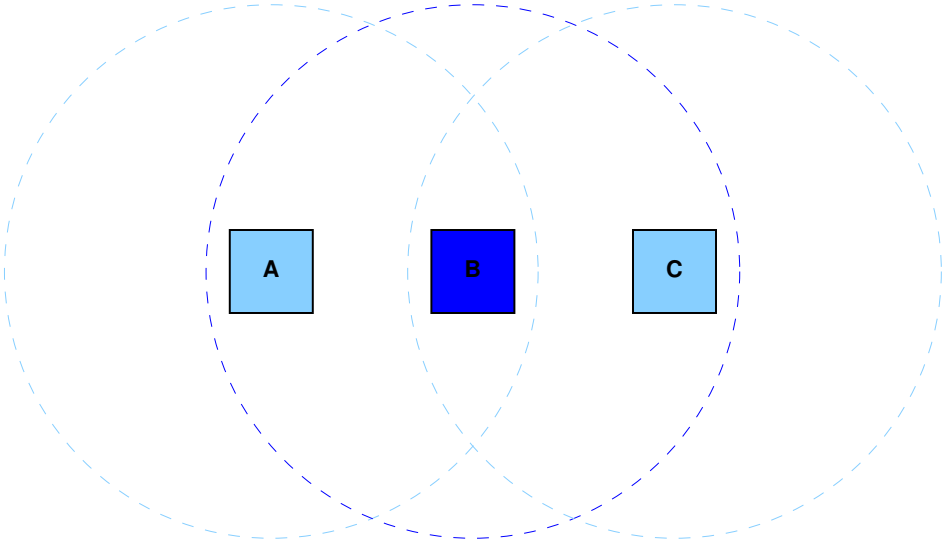


Abbildung 1: Knoten **C** liegt außerhalb der Reichweite von **A** und kann nur über **B** erreicht werden. [JoMa96]

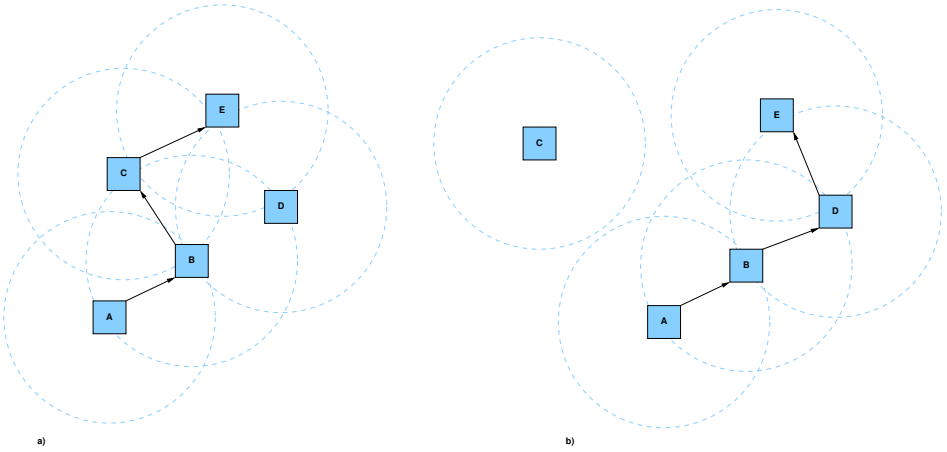


Abbildung 2: *a)* Die Route über Knoten **C** wird unterbrochen. *b)* Eine neue Route (hier über Knoten **D**) muss gefunden werden.

Bewegungen können unter Umständen Veränderungen der Empfangsverhältnisse sein oder auch, dass Knoten das Ad-hoc-Netz betreten oder verlassen. Es ist auch möglich, dass das Netz temporär oder sogar dauerhaft in Teilnetze aufgeteilt wird. Mit all diesen Situationen muss DSR umgehen können, damit möglichst geringe Verzögerungen und kein Datenverlust entsteht. DSR verwendet *Source Routing*, d.h. die genaue Route zum Zielknoten muss dem Sender vor dem Absenden des Pakets bekannt sein. Die Route besteht aus einer Sequenz von Hops, die *Route Record* genannt wird. Diese *Route Record* wird in die Header der Datenpakete geschrieben und dient jedem Knoten, den das Paket durchläuft, dazu, den als nächsten folgenden Knoten zu identifizieren. Die Vorteile des *Source Routing* sind zum einen die Schleifenfreiheit und die Tatsache, dass die Zwischenknoten keine aktuellen Informationen über das Netz vorhalten müssen, sondern ihren „Nachfolger“ anhand der *Route Record* herausfinden können. Abbildung 3 zeigt beispielsweise, wie Knoten **A** versucht eine Route zu Knoten **E** zu finden und diese Hop für Hop aufgebaut wird.

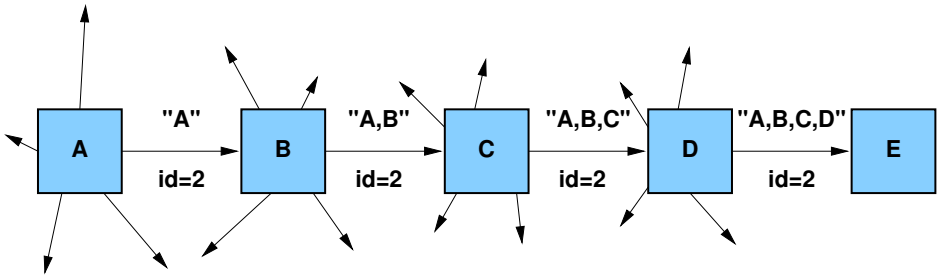


Abbildung 3: Die *Route Record* wird Hop für Hop aufgebaut. [JoMB01]

2 Aufbau des DSR-Protokolls

Um ein Paket an einen anderen Knoten **D** zu senden, muss der Sender **S** zunächst eine geeignete Route zum Zielknoten kennen. Ermittelt werden solche Routen durch das *Route Discovery*, das im folgenden Abschnitt genauer erläutert wird. Hat **S** eine oder auch mehrere Routen mittels *Route Discovery* gefunden, trägt er diese in seinen *Route Cache* ein. In diesem *Route Cache* werden alle zu einem Ziel bekannten Routen gespeichert. Befindet sich dort schon eine Route zu **D**, dann ist kein weiteres *Route Discovery* nötig. **S** fügt nur noch diese Route in den Paketheader ein. Das Paket durchläuft nun die Knoten des Netzwerks in der Reihenfolge, in der sie in der *Route Record* aufgeführt sind. Dabei sendet jeder Knoten auf dem Weg das Paket jeweils an den nächsten in der *Route Record* folgenden Knoten, bis das Paket den Zielknoten **D** erreicht hat. Der Einsatz des *Route Cache* sorgt dafür, dass die Zahl der *Route Discoveries* niedrig gehalten wird. Bricht ein Link zwischen zwei Knoten zusammen, z.B. weil ein Knoten abgeschaltet wurde oder außer Reichweite geraten ist, müssen die Knoten, die diesen Link momentan zur Datenübertragung verwenden, darüber informiert werden. Der dafür zuständige Mechanismus nennt sich *Route Maintenance* und wird ebenso wie das *Route Discovery* nur bei Bedarf eingesetzt. Es gibt folglich bei DSR keine periodisch versendeten Pakete wie bei anderen Protokollen (z.B. periodische *Routing Advertisements*, *Link State Sensing*, *Neighbor Detection Packets*, etc.). Weiß der Sender **S** nun über einen fehlerhaften Link Bescheid, kann er auf eine möglicherweise im *Route Cache* vorhandene alternative Route zurückgreifen oder muss ein erneutes *Route Discovery* durchführen.

2.1 Route Discovery

Ein *Route Discovery* besteht im Wesentlichen aus dem Versenden eines speziellen Pakets (Route Request - RREQ) als *Broadcast*. Der Sender des Route Request wird *Initiator* und der Zielknoten *Target* genannt. Enthält der *Route Cache* des Knoten **S** keine Route zu **D**, generiert er einen Route Request, der die Adressen des *Initiators* und des *Targets*, eine leere *Route Record* und eine vom Initiator vergebene *Request ID* enthält. Praktisch alle Knoten in der direkten Umgebung von **S** sollten diesen Route Request empfangen. Der Route Request wandert als *Broadcast* durch das Netz und jeder Knoten fügt seine Adresse an die *Route Record* an. Dabei müssen einige Regeln eingehalten werden, um ein übermäßiges Fluten des Netzes und unbrauchbare Routen zu verhindern. Jeder Knoten, der diesen Route Request empfängt, prüft, ob er genau diesen kürzlich schon einmal empfangen hat. Dazu unterhält jeder Host eine Liste der empfangenen Route Requests, die aus Paaren der Form *<initiator address, request id>* besteht. Hat er den Route Request tatsächlich schon einmal empfangen, so verwirft er den ihn. Findet er seine eigene Adresse bereits in der *Route Record*, so verwirft er ebenfalls den *Route Record* (s. Abbildung 4). Dies verhindert sehr effektiv die Bildung von Schleifen, da keine Hostadresse zweimal in der *Route Record* auftreten kann. Ist

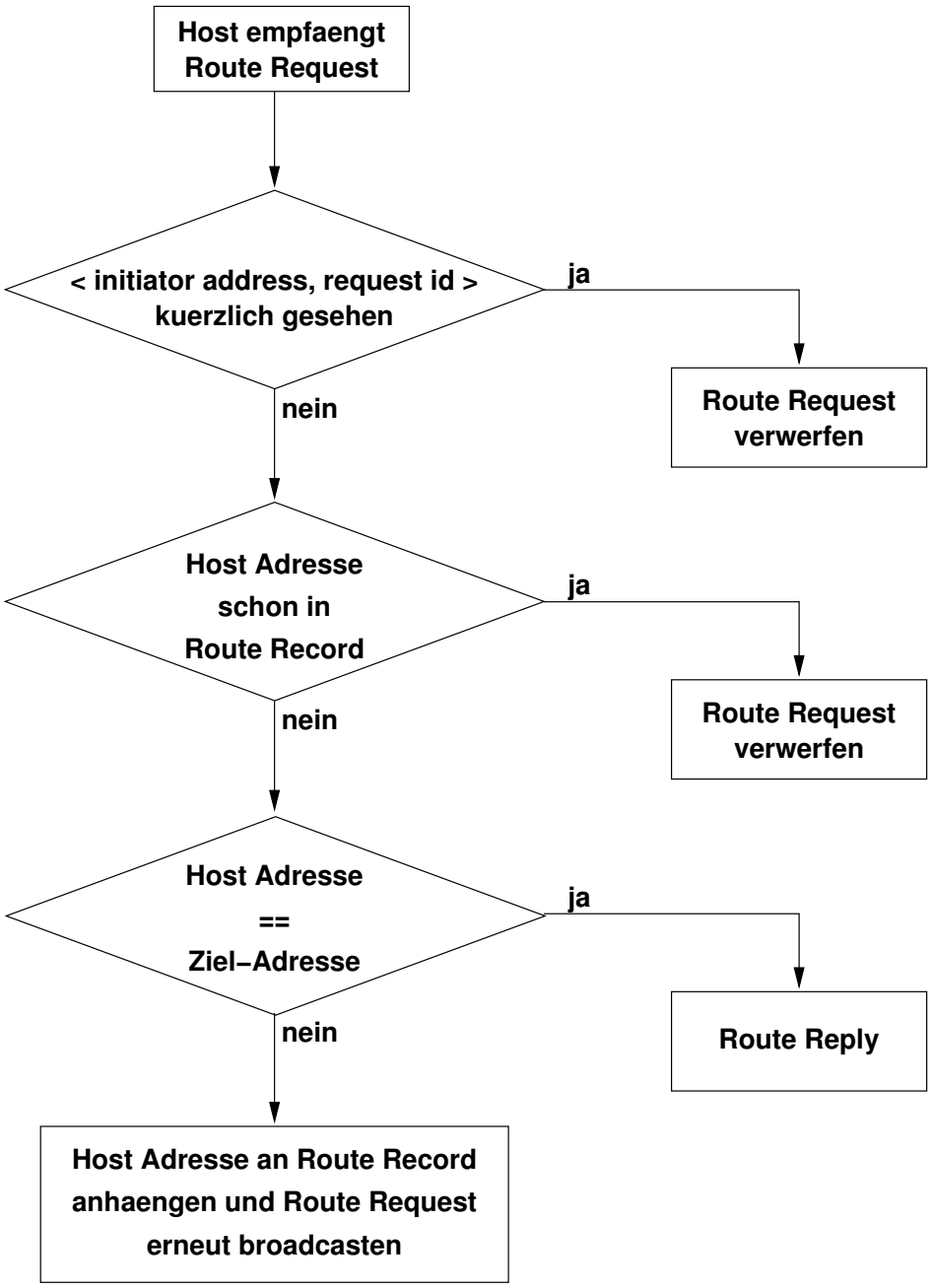


Abbildung 4: Ablauf der Bearbeitung eines ROUTE REQUEST in einem Knoten.

der Empfänger selbst das Ziel des ROUTE REQUEST, verpackt er Knoten die *Route Record* in einen ROUTE REPLY (RREP) und sendet diese an den Initiator zurück (s. Abbildung 5). Wenn

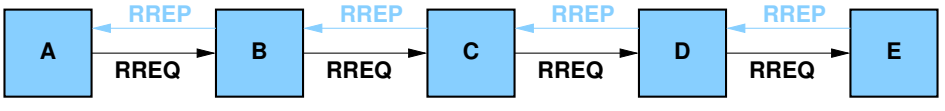


Abbildung 5: Rücksendung eines fertigen ROUTE REPLY an den Initiator.

der Initiator den ROUTE REPLY erhält, speichert er die Route in seinem *Route Cache* und verwendet sie zum Senden der weiteren Datenpakete bis ein ROUTE ERROR eintritt. Um den ROUTE REPLY an den Initiator zurückzuschicken, muss jedoch der Zielknoten ebenfalls eine

Route zum Initiator kennen. Die bevorzugte und einfachste Methode, wenn der *Route Cache* von **D** keine Route zu **S** enthält, ist, die Liste aus dem *Route Reply* zu invertieren. Dies setzt jedoch voraus, dass auf dieser Route ausschließlich bidirektionale Links auftreten, was z.B. auf der MAC-Schicht von IEEE 802.11 zwingend der Fall sein muss. Garantiert die MAC-Schicht keine bidirektionalen Links, dann muss **D** ein *Route Discovery* mit **S** als Ziel durchführen, wobei **D** jedoch den ROUTE REPLY als *Piggyback* im Paket an **S** sendet, um weitere rekursive *Route Discoveries* zu verhindern. Jedes Paket, für das ein *Route Discovery* durchgeführt wird, wird im *Send Buffer* des *Initiators* gespeichert und mit einem *Timer* versehen. Wenn kein ROUTE REPLY eintrifft, wird von Zeit zu Zeit ein erneutes *Route Discovery* angestoßen. Läuft der *Timer* aus oder droht der *Send Buffer* überzulaufen, wird das Paket gelöscht. Eine Übertragungswiederholung wird den höheren Schichten des Protokollstacks überlassen. Der *Initiator* muss die Rate, mit der neue *Route Discoveries* durchgeführt werden, begrenzen, um ein Überfluten des Netzwerkes mit *Route-Discovery*-Paketen zu verhindern. Dabei muss auch bedacht werden, dass das Netz vorübergehend in unabhängige Teilnetze zerfallen sein könnte und der Zielknoten somit möglicherweise nicht mehr erreichbar ist und folglich auch keine ROUTE REPLIES zurücksenden kann. Um die Rate der*Route Discoveries* zu begrenzen, wird ein exponentieller *Back-Off* eingesetzt. Dieser Mechanismus ähnelt dem, der zur Begrenzung von *ARP-Requests* im Internet eingesetzt wird.

2.2 DSR Route Maintenance

Route Maintenance dient zur Überwachung von Routen und zur Benachrichtigung von Knoten über nicht mehr funktionierende Hops. Bei den meisten drahtlosen Netzen ist ein *Hop-by-Hop-Acknowledgement* in der Sicherungsschicht implementiert, was das *Route Maintenance* wesentlich erleichtert. Meldet die Sicherungsschicht einen Fehler (z.B. durch Erreichen der maximalen Anzahl an Sendewiederholungen eines Pakets), sendet der Host einen ROUTE ERROR (RERR) (s. Abbildung 6). Der ROUTE ERROR enthält die Adressen der beiden Hosts an

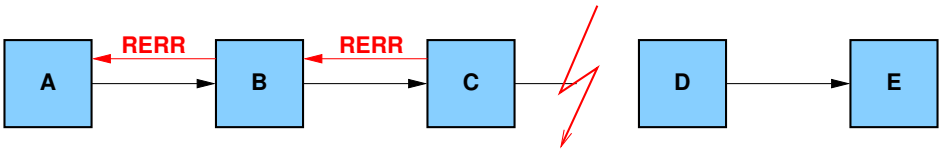


Abbildung 6: Fällt ein Link aus, sendet der Knoten, der den Ausfall bemerkt, einen ROUTE ERROR an den *Initiator*.

den Seiten des fehlerhaften Hops. Der ROUTE ERROR wird an alle Quellknoten gesendet, die momentan Pakete über die Route mit dem fehlerhaften Link senden. Empfängt ein Quellknoten oder ein Zwischenknoten einen ROUTE ERROR, löscht er alle Routen, die den fehlerhaften Link verwenden aus seinem *Route Cache*. Um einen ROUTE ERROR zurückzusenden, stehen dieselben Mechanismen wie beim ROUTE REPLY zur Verfügung (eine Route im *Route Cache* suchen, die *Route Record* des letzten Pakets invertieren, oder per *Piggybacking* z.B. mit einem ROUTE REQUEST an den *Initiator* zurücksenden). Konventionelle Routingprotokolle fassen *Route Discovery* und *Route Maintenance* zusammen, indem sie ständig periodische Updates der Routinginformationen oder HELLO-Nachrichten *(beacons)* senden. Empfängt ein Host eine zeitlang keine *beacons* von einem Nachbarn, geht er davon aus, dass der Link gestört ist. Bei DSR erfolgt diese Überwachung implizit beim Weiterleiten eines Pakets. Jeder Knoten auf der Route muss sicherstellen, dass der nach ihm folgende Knoten das Paket korrekt empfangen hat. Das Paket muss im Zweifelsfall so oft gesendet werden, bis eine Quittung eintrifft. Diese Quittung kann oft ohne weitere Kosten eingeholt werden, entweder als Teil des zugrundeliegenden MAC-Protokolls oder als passive Quittung *(passive Acknowledgement)*. Bsp.: Knoten

B braucht eine Bestätigung, dass Knoten **C** das Paket erhalten hat. Hört **B** nun, dass **C** das Paket an **D** weitersendet, so impliziert dies den korrekten Empfang durch **C**. Sollten diese Arten der Bestätigung nicht möglich sein, kann ein Knoten eine DSR-spezifische Quittung vom nächsten Knoten anfordern. Diese Quittung wird in der Regel direkt gesendet, kann bei unidirektionalen Verbindungen aber auch über mehrere andere Knoten laufen.

2.3 Optimierungen

2.3.1 Verwendung von Informationen mitgehörter Pakete

Die aktiven Routen bilden im *Route Cache* jedes Hosts einen Baum. Lernt der Host eine Route von **A** nach **D**, so erhält er damit auch gleichzeitig die Routen von **A** nach **B** und von **A** nach **C**. Eine Anpassung der Routen bei Veränderungen im Netz ist einfach möglich. Kommt z.B. später noch die Route **A** nach **E** hinzu, genügt das Hinzufügen des Links **D-E** in den Baum (s. Abbildung 7). Entdeckt **A**, dass zu **D** auch direkt gesendet werden kann, wird der

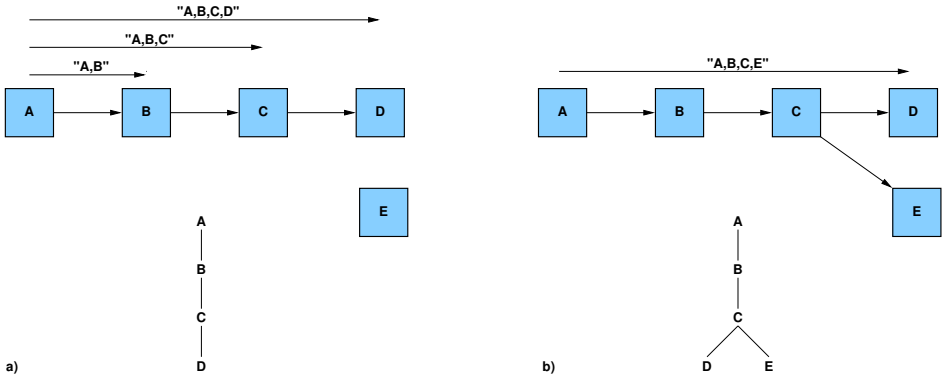


Abbildung 7: Die gelernte Route bildet einen Baum im *Route Cache* von **A**. Damit kennt **A** auch alle Zwischenrouten. Die Knoten **B** und **C** können durch Mithören der Pakete ebenfalls diese Route in ihren *Route Cache* aufnehmen.

Baum angepasst und alle anderen Routen, die diesen Link enthalten werden verkürzt (*Route Shortening*, s. Abbildung 8). Zudem kann jeder Host die Informationen aus der *Route Record*

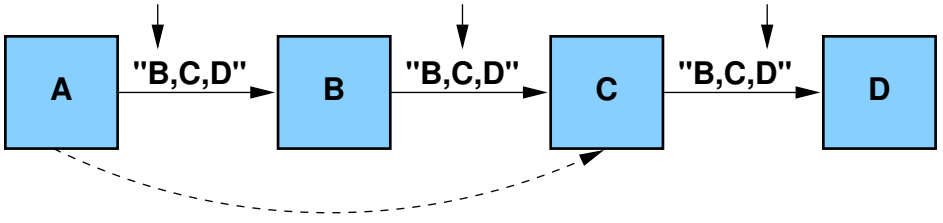


Abbildung 8: *Route Shortening*: **A** erkennt, dass **C** auch direkt erreichbar ist und streicht **B** aus den betreffenden Routen.

jedes Pakets, das er weiterleitet, in seinen *Route Cache* einfügen. Wenn z.B. **B** ein Paket von **A** nach **D** weiterleitet, kann er die enthaltenen Routen (**B** nach **D**) selbst verwenden. Der gleiche Vorgang kann auch bei weitergeleiteten ROUTE REPLIES angewendet werden. Hat der Host die Möglichkeit, sein Interface in den *Promiscuous Mode* zu schalten, kann er sogar die Pakete sämtlicher Knoten in Reichweite abhören und die in den Paketen enthaltenen Routinginformationen verwenden. All dies führt zu einer Verringerung der Anzahl an *Route Discoveries* und verringert damit den Paket-*Overhead*.

2.3.2 Verwenden des Route Cache für Route Replies

Angenommen Knoten **B** empfängt einen ROUTE REQUEST von **A** zu **D** und der ROUTE REQUEST erfüllt alle Bedingungen, um nicht von **B** verworfen werden. Er erreicht also **B** zum ersten Mal und **B** ist nicht der Zielknoten. Kennt **B** bereits eine Route von sich zu **D**, so kann er diese an den *Route Record* anhängen und das ganze als ROUTE REPLY zu Knoten **A** zurücksenden. Dieses Vorgehen kann jedoch zu Problemen führen, wenn mehrere Knoten in Reichweite von **A** eine solche Route im *Route Cache* haben und gleichzeitig einen ROUTE REPLY zu **A** senden *(Route Reply Storm)*. Dadurch können sowohl Paketverluste als auch Stausituationen auftreten. Deshalb muss ein Host, der aus seinem *Route Cache* antwortet folgende Schritte ausführen: Er verzögert das Senden um eine Zeitspanne $d = H \times (h - 1 + r)$, wobei h die Länge der *Route Record* in Hops, r eine Zufallszahl zwischen 0 und 1 und H eine kleine konstante Verzögerung ist. Während dieser Zeitspanne d hört der Host alle Pakete an den *Initiator* des *Route Discovery* ab. Empfängt er einen ROUTE REPLY an den *Initiator* und ist die *Route Record* dieses ROUTE REPLY kürzer als h, so bricht der Host ab und verwirft den ROUTE REPLY, da er davon ausgehen muss, dass der *Initiator* den ROUTE REPLY mit der kürzeren Route empfangen hat. Durch das Verwenden des *Route Cache* für ROUTE REPLIES entsteht zudem das Problem, dass die Schleifenfreiheit der Routen nicht mehr garantiert werden kann. Sowohl der *Route Record* des ROUTE REQUEST als auch der Eintrag im *Route Cache* sind zwar schleifenfrei, jedoch kann durch Zusammenfügen der beiden eine Schleife entstehen. Bemerkt nun ein Host, der nicht der Zielhost ist, dass sein ROUTE REPLY eine Schleife enthalten würde, so muss er den ROUTE REPLY verwerfen.

2.3.3 Route Request Hop Limits

Der *Initiator* eines *Route Discovery* kann ein *Hop Limit* für den ROUTE REQUEST setzen und damit dessen Ausbreitung beschränken. Hält ein Knoten in unmittelbarer Nähe eine Route zum Zielhost in seinem *Route Cache*, so kann durch den Einsatz des *Hop Limits* eine unnötige Ausbreitung des ROUTE REQUEST verhindert werden. Dazu setzt der *Initiator* das *Hop Limit* zunächst auf 1. Erhält er keinen ROUTE REPLY, wird ein neuer ROUTE REQUEST mit einem höheren *Hop Limit* gesendet. Dadurch kann sehr einfach ermittelt werden, ob sich der Zielhost in unmittelbarer Nähe befindet oder einer der direkten Nachbarknoten eine Route zum Zielhost kennt. Dies kann auch als eine Erweiterung des eigenen *Route Cache* um die *Route Caches* der Nachbarn gesehen werden.

3 Performance von DSR

In [Malt01] und [DaPR00] wurden unabhängig voneinander Messungen der Leistungsfähigkeit des DSR-Protokolls durch verschieden Testszenarien auf einem ns-2-Simulator [FaKV05] durchgeführt. Als Grundlage diente die Simulation von 50 mobilen Knoten in einem rechteckigen Gebiet (1500m x 300m). Die Dauer eines Testlaufs betrug 15 Minuten (900s). Das Bewegungsmodell der Knoten lässt sich folgendermaßen beschreiben: Ein Knoten verweilt für eine bestimmte Zeitspanne *(Pause Time)* an seinem aktuellen Ort. Nach Ablauf der *Pause Time* wählt er einen neuen Punkt zufällig aus, bewegt sich dorthin und wählt nach der *Pause Time* wieder einen neuen Punkt aus. Dieses Verhalten wird über die ganze Laufzeit der Simulation fortgeführt. In den unterschiedlichen Testläufen werden die *Pause Times* zwischen 0s und 900s variiert, d.h. von ständiger Mobilität (*Pause Time* = 0s) und keinerlei Bewegung (*Pause Time* = 900s). Zusätzlich wurden die Anzahl der Paketquellen variiert, die jeweils einen Datenstrom mit konstanter Bitrate *(CBR)* an zufällig gewählte Zielknoten

senden. Weitere variierte Parameter waren die Bewegungsgeschwindigkeit der Knoten, die Paketgröße und die Zahl der gesendeten Pakete pro Sekunde. Um vergleichbare Ergebnisse für die verschiedenen Protokolle zu erhalten, wurden verschiedene Bewegungs- und Kommunikationsmuster aufgezeichnet und für alle Messungen verwendet, um gleiche Voraussetzungen für alle Protokolle zu schaffen. Weitere Details zur Simulation und zu den Implementierungen der einzelnen Protokolle finden sich in [Malt01] und [DaPR00].

3.1 Protokolle im Vergleich

Zum Vergleich mit DSR in [Malt01] wurden DSDV [PeBh94], AODV [CPDa00] und TORA [PaCo97] herangezogen. In [DaPR00] wurden lediglich AODV und DSR untersucht. Diese Protokolle wurden ausgewählt, um einen möglichst großen Bereich der verschiedenen Routingverfahren einzuschließen. So verwenden TORA und DSDV periodisch versendete Routingpakete *(Periodic Routing Advertisements)*, wohingegen DSR und AODV nur im Bedarfsfall solche versenden *(on-demand)*. Während DSDV diese Routingpakete benötigt um fehlerhafte Links zu entdecken, bedienen sich DSR und AODV der Funktionen der MAC-Schicht. Grundsätzlich unterscheiden sich die Protokolle auch in ihrer Art der Paketweiterleitung: DSR verwendet *Source Routing*, dagegen verwenden DSDV, AODV und TORA *Hop-by-Hop-Routing*. Im weiteren soll die Diskussion auf hauptsächlich DSR und AODV beschränkt bleiben, da diese in beiden Simulationen untersucht wurden.

3.1.1 Destination Sequenced Distance Vector (DSDV)

Das Destination Sequenced Distance Vector Protokoll (DSDV) ist wie der Name schon sagt ein Distanz-Vektor-Routingprotokoll. Distanz-Vektor-Protokolle bedingen den periodischen Austausch von Routinginformationen zwischen den Knoten eines Netzwerkes durch *Broadcasts*. Die Besonderheit von DSDV gegenüber anderen Distanz-Vektor-Protokollen ist, dass es Schleifenfreiheit garantiert. Jeder Knoten unterhält eine Routingtabelle, die für jedes erreichbare Ziel einen Eintrag mit dem nächsten zu wählenden Knoten und der Anzahl der Hops bis zum Zielknoten enthält. Jede in der Routingtabelle eingetragene Route erhält eine Sequenznummer, die die Aktualität der Route darstellt. Treffen Routinginformationen mit der gleichen Sequenznummer ein, werden diejenigen mit der kleineren Metrik verwendet. Bei Änderungen in der Topologie werden *Incremental-Updates*, die nur die Änderung mitteilen, oder gleich alle Routinginformationen per *Full-Update* versendet. Wird ein Link unterbrochen, senden die Knoten *Updates* der betroffenen Routen mit Sequenznummern und unendlicher Metrik. Alle Knoten, die diese Informationen empfangen, tragen sie in ihre Routingtabellen ein und überschreiben die defekte Route erst wieder, wenn sie eine aktuellere Route empfangen.

3.1.2 Ad Hoc On-Demand Distance Vector (AODV)

AODV stellt eine Mischung aus DSR und DSDV dar. Es benutzt grundlegende Mechanismen zum Auffinden und Aufrechterhalten von Routen wie DSR, d.h. *Route Discovery* und *Route Maintenance*. Von DSDV stammen das *Hop-by-Hop-Routing*, Sequenznummern und periodische *Updates* der Routinginformationen. Die *Route Discoveries* funktionieren wie bei DSR, nur dass dem Route Request noch die letzte bekannte Sequenznummer des Ziels angehängt wird. Der Route Request wird durch das Netz geflutet und jeder Knoten, der den Route Request empfängt erstellt eine Route zum Ursprungsknoten zurück *(Reverse Route)*. Empfängt ein Knoten, der bereits eine Route zum Zielknoten kennt, den Route Request, so

erstellt er ein ROUTE REPLY, das die Anzahl der Hops bis zum Ziel und die aktuellste Sequenznummer, die der Knoten momentan kennt, enthält. Jeder Knoten, der wiederum diesen ROUTE REPLY erhält, baut eine Route zum Zielknoten auf *(Forward Route)*. Es wird jedoch nicht die komplette Route gespeichert, sondern nur der jeweils nächste Knoten auf dem Weg zum Zielknoten. Entlang des somit aufgebauten Weges werden dann die Datenpakete von der Quelle bis zum Zielknoten weitergeleitet. Um nicht mehr funktionierende Links aufzuspüren, sendet jeder Knoten HELLO-Nachrichten mit einer bestimmten Rate. Bleiben drei aufeinander folgende HELLO-Nachrichten aus, gehen die umlegenden Knoten davon aus, dass der Link nicht mehr funktioniert. Daraufhin wird jeder Knoten, der kürzlich Pakete an ein Ziel über diesen Link gesendet hat, mit einem UNSOLICITED ROUTE REPLY über den fehlerhaften Link benachrichtigt. Die betreffende Route erhält eine unendliche Metrik und der Knoten muss vor dem Senden weiterer Pakete ein neues *Route Discovery* durchführen.

3.2 Ergebnisse

Die Untersuchungen in [Malt01] und [DaPR00] zeigen, dass eines der Hauptziele von DSR, nämlich ein möglichst geringer *Overhead* durch Routing, erreicht wurde. Misst man den *Routingoverhead* auf Paketebene, so ist dieser bei DSR von den verglichenen Protokollen durchweg am geringsten *Overhead* (s. Abbildung 9). Zudem zeigen die Ergebnisse in [DaPR00], dass

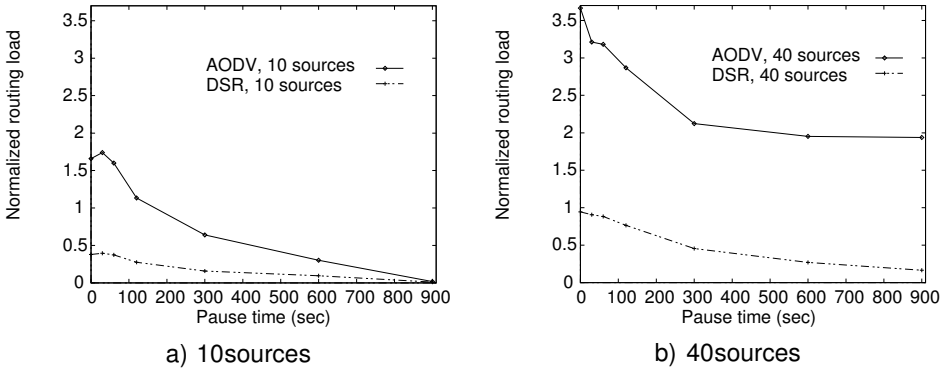


Abbildung 9: Diese Abbildungen zeigen den *Routingoverhead* in Paketen und in Bytes. [Malt01]

DSR mit der Anzahl der Quellen linear skaliert. Diese Ergebnisse lassen sich jedoch nicht auf Messungen auf Byteebene übertragen, da die Routingpakete bei den verschiedenen Protokollen starke Unterschiede in der Größe aufweisen. So hat DSR aufgrund der in den Paketen mitgeführten *Route Record*s einen *Byteoverhead*, der bei kleinen *Pause Times* geringer, aber bei höheren *Pause Times* wesentlich größer als der *Overhead* von AODV ist (s. Abbildung 10). Ein weiterer wichtiger Gesichtspunkt bei der Beurteilung eines Protokolls ist der Anteil der erfolgreich übertragenen Pakete gemessen an der Gesamtzahl der gesendeten Pakete *(Packet Delivery Fraction)*. [DaPR00] zeigt, dass DSR und AODV bei wenigen Quellen (10-20 Quellen) in etwa gleich auf liegen, unabhängig von der *Pause Time*. Bei vielen Quellen (30-40 Quellen) liegt DSR bei hoher Mobilität (geringe *Pause Time*) unter AODV, d.h. es werden weniger Pakete erfolgreich übertragen (s. Abbildung 11). Werden die *Pause Times* jedoch vergrößert, übertrifft DSR das andere Protokoll. Ein ähnliches Bild zeichnet sich für die Ende-zu-Ende-Verzögerung. Bei wenigen Quellen unterscheiden sich die Werte von DSR und AODV nur geringfügig. Wird die Zahl der Quellen erhöht, hat DSR für kleine *Pause Times* eine etwas größere, für große *Pause Times* jedoch eine teilweise wesentlich geringere Verzögerung. In [Malt01] werden etwas andere Erkenntnisse beschrieben. Hier liefern AODV und DSR auch bei vielen Quellen (30 Quellen) etwa gleich viele Pakete erfolgreich bei den

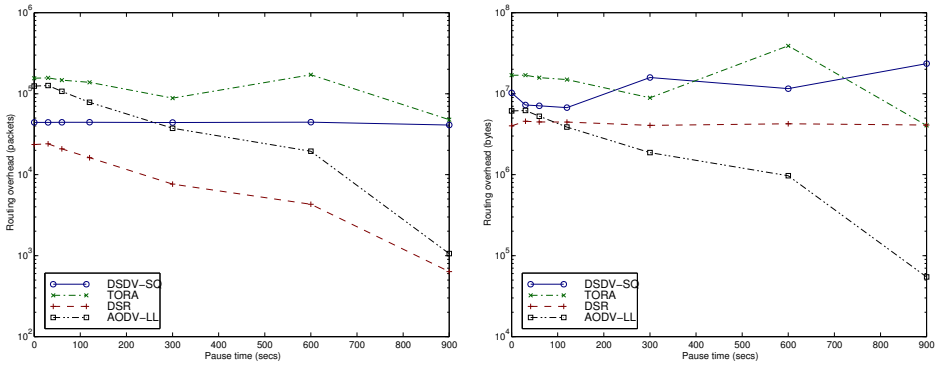


Abbildung 10: Diese Abbildungen zeigen die Zahl der Routingpakete pro übertragenem Datenpaket als Funktion der *Pause Time* für verschiedene Quellenanzahlen. [DaPR00]

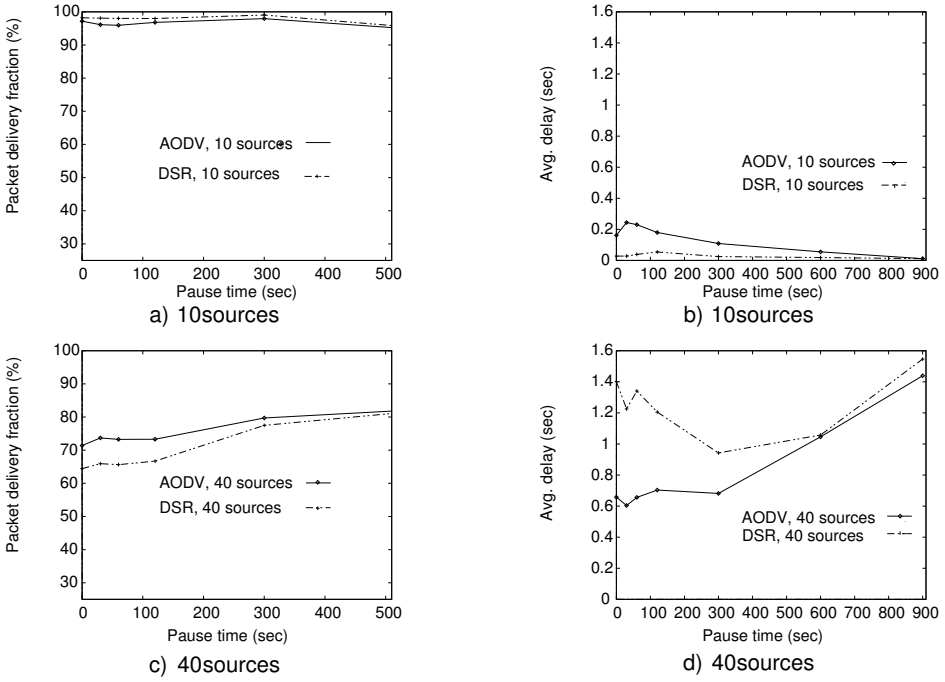


Abbildung 11: *a)* und *c)* zeigen den Anteil der erfolgreich übertragenen Pakete an der Gesamtzahl der gesendeten Pakete als Funktion der *Pause Time* und für verschiedene Quellenanzahlen. *b)* und *d* zeigen die durchschnittliche Ende-zu-Ende-Verzögerung als Funktion der *Pause Time* und für verschiedene Quellenanzahlen. [DaPR00]

Zielknoten ab, während DSDV bei hoher Mobilität stark einbricht. Die Differenzen der beiden Simulationen dürften auf Unterschiede in der Implementierung, wie z.B. unterschiedliche Paketgrößen zurückzuführen sein. Übereinstimmend belegen beide den wesentlich geringeren *Overhead* durch Routingpakete von DSR gegenüber den restlichen Protokollen. Die Ergebnisse von DSDV übertreffen die von DSR in keiner Disziplin [Malt01], Weder beim *Routingoverhead* noch bei der *Packet Delivery Fraction*, diese ist jedoch bei kleinen *Pause Times* wesentlich geringer als bei den anderen Protokollen. Der *Routingoverhead* von DSDV ist für alle *Pause Times* ungefähr konstant, was an den periodisch versendeten Routingpaketen liegt.

4 Zusammenfassung

Ziel dieser Ausarbeitung war die Beschreibung des Dynamic Source Routingprotokolls (DSR) und ein kurzer Überblick über die Leistungsfähigkeit. Im Wesentlichen wurde DSR mit AODV verglichen, da es sich bei beiden um reaktive Routingprotokolle handelt *(on-demand)*. Auf DSDV und TORA wurde nicht genauer eingegangen, ein ausführlicherer Vergleich dieser Protokolle findet sich in [Malt01]. DSR hat eines der wesentlichen Designziele erreicht: Der Overhead durch das Routing ist vergleichsweise niedrig und das Protokoll skaliert gut mit steigender Quellenzahl. In Situationen mit hohem „Stress“ (hohe Mobilität, viele Datenquellen) zeigt DSR Schwächen gegenüber AODV [DaPR00]. Bei niedrigerer Intensität zeigt dagegen DSR seine Stärken. Das *Cachen* von mehreren Routen hilft DSR bei der einfachen Überwindung von gestörten Links. Die aggressive Nutzung des *Route Cache* und das Fehlen von Mechanismen zum Erkennen von veralteten Routen führt laut [DaPR00] zu *Performanceproblemen*. Die Verwendung eines solchen Mechanismus könnte die *Performance* von DSR wesentlich steigern. Dafür erreicht DSR durch diese aggressive Nutzung des *Route Caches* den niedrigen *Routingoverhead* und bei niedrigen Netzlasten auch einen *Performancevorteil* gegenüber AODV.

Literatur

[CPDa00] E. M. Royer C. Perkins und S. R. Das. Ad Hoc On Demand Distance Vector (AODV) Routing, 2000.

[DaPR00] S. R. Das, C. E. Perkins und E. E. Royer. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. In *INFOCOM (1)*, 2000, S. 3–12.

[FaKV05] K. Fall und Eds. K. Varadhan. *The ns Manual (formerly ns Notes and Documentation)*. available at http://www.isi.edu/nsnam/ns/, 2005.

[JoMa96] D. B. Johnson und D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski und Korth (Hrsg.), *Mobile Computing*, Band 353. Kluwer Academic Publishers, 1996.

[JoMB01] D. Johnson, D. Maltz und J. Broch. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, Kapitel 5, S. 139–172. Addison-Wesley. 2001.

[Malt01] D. Maltz. Demand Routing in Multi-hop Wireless Mobile Ad Hoc Networks, 2001.

[PaCo97] V. D. Park und M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *INFOCOM (3)*, 1997, S. 1405–1413.

[PeBh94] C. Perkins und P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994, S. 234–244.

Abbildungsverzeichnis

1 Knoten **C** liegt außerhalb der Reichweite von **A** und kann nur über **B** erreicht werden. [JoMa96] 34

2 *a)* Die Route über Knoten **C** wird unterbrochen. *b)* Eine neue Route (hier über Knoten **D**) muss gefunden werden. 34

3 Die *Route Record* wird Hop für Hop aufgebaut. [JoMB01] 35

4 Ablauf der Bearbeitung eines ROUTE REQUEST in einem Knoten. 36

5 Rücksendung eines fertigen ROUTE REPLY an den Initiator. 36

6 Fällt ein Link aus, sendet der Knoten, der den Ausfall bemerkt, einen ROUTE ERROR an den *Initiator*. 37

7 Die gelernte Route bildet einen Baum im *Route Cache* von **A**. Damit kennt **A** auch alle Zwischenrouten. Die Knoten **B** und **C** können durch Mithören der Pakete ebenfalls diese Route in ihren *Route Cache* aufnehmen. 38

8 *Route Shortening*: **A** erkennt, dass **C** auch direkt erreichbar ist und streicht **B** aus den betreffenden Routen. 38

9 Diese Abbildungen zeigen den *Routingoverhead* in Paketen und in Bytes. [Malt01] 41

10 Diese Abbildungen zeigen die Zahl der Routingpakete pro übertragenem Datenpaket als Funktion der *Pause Time* für verschiedene Quellenanzahlen. [DaPR00] 42

11 *a)* und *c)* zeigen den Anteil der erfolgreich übertragenen Pakete an der Gesamtzahl der gesendeten Pakete als Funktion der *Pause Time* und für verschiedene Quellenanzahlen. *b)* und *d* zeigen die durchschnittliche Ende-zu-Ende-Verzögerung als Funktion der *Pause Time* und für verschiedene Quellenanzahlen. [DaPR00] 42

Multicasting auf Anwendungsebene

Djamal Guerniche

Kurzfassung

Diese Seminararbeit befasst sich mit dem Multicasting in Mobilen Ad-Hoc Netzen (MANETs). Zunächst wird im Abschnitt 1. gezeigt, was Mobile Ad-Hoc Netze und Multicasting sind. Daraufhin wird im Abschnitt 2. eines der besten Multicast Routingprotokolle auf Netzwerkebene für MANETs, nämlich ODMRP, ausführlich beschrieben. Im Abschnitt 3 wird ALMA, eines der besten Multicast-Protokolle für MANETS auf Anwendungsebene, im Detail beschrieben. Letztendlich wird in Abschnitt 4 ein Vergleich zwischen Multicasting auf Anwendungsebene und Multicasting auf Netzwerkebene, das von der Autoren von ALMA anhand der beiden Protokolle ALMA und ODMRP realisiert wurde, vorgestellt.

1 Einleitung

1.1 Mobile Ad-Hoc Netze

Mobile Ad-hoc Netze sind eine Ansammlung von mobilen Knoten, die spontan(Ad-Hoc) ein Netz bilden und miteinander, ohne jegliche Infrastruktur oder zentrale Verwaltung, über Funk kommunizieren können. MANETs sind hoch dynamische Netze, die kontinuierlich ihre Topologie und ihre aus verschiedenen Knoten Zusammensetzung verändern können (Knoten können sich plötzlich an das Netz anschliessen oder es verlassen), was zu einer häufigen neuen Routeberechnung führt. Die Multihop-Kommunikationen sind anhand spezifischer Routingprotokolle möglich, wo jeder Knoten als Router fungieren kann.

1.2 Multicasting in Mobile Ad-Hoc Netze

Multicast ist eine Kommunikationsform, in der ein Sender eine einzige Kopie der Daten an mehrere Empfänger gleichzeitig sendet. Diese Form der Kommunikation eignet sich sehr für mehrere Anwendungen (z.B die rechnergestützten Konferenzsysteme), die eine Gruppenkommunikation mit sich bringen. Alle existierenden Multicast-Routing-Protokolle basieren auf den folgenden theoretischen Verfahren.

1. Flooding: Das einfachste Verfahren zur Übertragung von Multicast-Nachrichten über mehrere Router hinweg stellt das sog. Flooding oder auch Fluten dar. Die Nachricht wird in diesem Fall analog zum Broadcast von allen Routern im Netz weitergeleitet. Dabei entstehen die folgenden Nachteile bei der Weiterleitung von Multicast-Daten über Router. Vor allem ist die Bildung von Schleifen zu vermerken, bei der ein Router die Multicast-Nachricht an seine Nachbar-Router weiterleitet, die ggf. über weitere Router diese Nachricht schließlich wieder an ihn versenden. Dieses Problem wird beim Improved Flooding vermieden. Dazu kommmen eine hohe Netzlast und die Möglichkeit der Entstehung von Duplikaten der Dateneinheiten.

2. Improved Flooding: Das Improved Flooding oder auch „verbesserte Fluten“ bezeichnet Ansätze, die die vom Flooding bekannten Probleme der Schleifenbildung vermeiden. Dabei werden in der Regel in jedem Router separate Listen der empfangenen Multicast-Daten gespeichert. Anhand dieser Listen kann ein bereits empfangenes Multicast-Paket identifiziert und bei erneutem Empfang gezielt verworfen werden. Es ist zu bemerken, dass diese Erweiterung des Flooding lediglich die Schleifenbildung kontrolliert, jedoch die anderen Nachteile völlig unangetastet lässt. So werden die Multicast-Daten weiterhin analog zum Broadcast im gesamten Netz verteilt und Bandbreite sowie ggf. Kosten im WAN-Bereich verschwendet. Außerdem ist die Pflege einer Liste der empfangenen Multicast-Pakete nur ein theoretisches Konzept. In der Praxis würde diese Liste sehr schnell einen sehr großen Umfang erlangen und zu inakzeptablen Wartezeiten bei deren Abarbeitung und damit der Paketweiterleitung führen.

3. Steiner-Bäume: Um das Problem der Bandbreitenverschwendung zu lösen, wird bei der Übertragung von Multicast-Daten in Netzen ein Multicast-Baum gebildet. Dieser Baum etabliert ein Overlay-Netz zur Vermeidung von Schleifen und zur verbindung aller Router, deren Subnetze Teilnehmer einer Multicast-Gruppe beinhalten, ohne dabei Schleifen zu erzeugen. Es handelt sich somit gewissermaßen um die Verbindung mit den geringstmöglichen Kosten bzw. die geringste Anzahl von nötigen Kanten (unter Berücksichtigung von deren Kosten). Die Bildung eines solchen Baums ist auch als Steiner-Problem bekannt, weshalb man den Baum auch als Steiner-Baum bezeichnet.

4. Bäume mit Rendezvous-Stellen: Um ein gezieltes Weiterleiten der Daten vor mehreren Sendern an mehrere Empfänger zu erreichen, wird pro Gruppe ein Spannbaum etabliert und ein oder mehrere Rendezvous-Stellen selektiert.Im Gegensatz dazu berücksichtigen die auf Steiner-Bäumen basierenden Verfahren nur einen Sender. Die Gruppenmitglieder melden sich bei der Rendezvous-Stelle durch das Versenden entsprechender Datenpakete an. Die Router, die sich zwischen Gruppenmitglied und der Rendezvous-Stelle befinden, leiten die Datenpakete in Richtung Rendezvous-Stelle weiter. Die Router benötigen hierzu, im Gegensatz zum quellenbasierten Routing, lediglich eine Information pro Gruppe. Die Sender von Multicast-Daten müssen nicht Mitglieder der Gruppe sein. Der wesentliche Nachteil dieses Multicast-Routing-Verfahrens ist die Verkehrskonzentration an den Rendezvous-Stellen.

5. Empfängerbasiertes Routing: Die bisher beschriebenen Verfahren benutzen alle einen gemeinsamen Multicast-Baum. Das Empfängerbasierte Routing geht hier einen anderen Weg, indem es ausgehend von der Quelle die jeweils günstigste Route zum Empfänger als separate Route vermerkt. Der jeweilige Router, der der Multicast-Gruppe beitreten möchte, versendet daher eine Join-Nachricht (Beitritt) an den Multicast-Router, der das Subnetz der Quelle (des Senders) verwaltet. Dieser ermittelt dann rückwärts den kürzesten Pfad zum Empfänger. Bei der Ermittlung dieses Pfades wird ein sehr einfacher Algorithmus verwendet, der wie folgt beschrieben werden kann: Ein Router, der ein Multicast-Paket empfängt, leitet dieses Paket genau dann an alle seine Ausgangsleitungen, außer der, auf der er es empfangen hat, weiter, wenn er das Paket auf der Schnittstelle empfangen hat, die aus seiner Sicht die kürzeste Verbindung zur Quelle darstellt. Auf diese Art und Weise wird ein vollständiger Multicast-Baum erzeugt, der keine Schleifen enthält. Dieses Verfahren wird auch als Reverse Path Forwarding (RPF) bezeichnet. Ein wesentlicher Nachteil dieses Verfahren ist die Tatsache, dass auch Netzteile, in denen sich keine Gruppenmitglieder befinden, mit Multicast-Daten belastet werden. Es gibt viele Erweiterungen dieses Verfahrens, wie z.B. das RPM (Reverse Path Multicasting) und einige weitere, die die Gruppenmitgliedschaft berücksichtigen und damit den vorher erwähnten Nachteil beseitigen.

1.2.1 Multicasting auf Netzwerkebene in MANETs

Netzwerkebene-Multicast-Routing-Protokolle erfordern, dass alle Knoten im Netz sich an dem Replizieren und Weiterleiten von Multicast-Daten beteiligen. In mobilen Ad-Hoc Netzen existieren in der Praxis mehrere Netzwerkebene-Multicast-Routing-Protokolle. Die am meisten davon bekannten sind ARMIS [WuTa99], MAODV [E.Pe99], CAMP [JLAea99] und ODMRP(siehe Abschnitt 2). Das Adhoc-Multicast-Routing-Protocol (ARMIS) benutzt eine inkrementelle Id-Nummer und erzeugt einen Multicast Weiterleitungsbaum, um die Multicast-Daten weiterzuleiten. Multicast AODV (MAODV) ist eine Erweiterung des Ad Hoc On Demand Distance Vector (AODV). MAODV erzeugt bei Bedarf (on-demand) Multicast-Routen und benutzt diese, um Multicast-Daten weiterzuleiten. Core-Assisted Mesh Protocol (CAMP) baut anhand eines Empfänger-initialisierten Ansatzes ein sogenanntes Multicast-Mesh zwischen den Knoten auf und basiert auf einer Rendevous Stelle, um das Mesh zwischen den Knoten aufrechtzuerhalten. On-Demand Multicast Routing Protocol (ODRMP) baut ebenfalls ein Mesh zwischen den Knoten, um die Multicast Daten weiterzuleiten. Da ODMRP in dieser Ausarbeitung als Vertreter der Netzwerkebenen-Multicasting-Protokolle für den Vergleich des Multicasting auf Anwendungsebene und Netzwerkebene benutzt wird, wird es im Abschnitt 2 genauer betrachtet.

1.2.2 Multicasting auf Anwendungsebene in MANETs

Multicasting-Protokolle auf Anwendungsebene verwenden das sog. Overlay-Netzwerk-Prinzip, um Multicast-Daten weiterzuleiten. Ein Overlay-Netzwerk ist ein logisches Netzwerk über dem physikalischen Netzwerk, das nur aus Gruppenmitgliedsknoten besteht (siehe Bild3). Diese Knoten sind für das Replizieren und für die Weiterleitung von Multicast Daten zuständig. Die wesentlichen Vorteile der Multicasting-Protokolle auf Anwendungsebene sind die Folgenden:

- Die Einfachheit zum Einsatz.
- Die Unabhängigkeit von den darunter liegenden Protokollen (d.h. sie können mit jedem beliebigen Protokoll-Stack operieren).
- Die Fähigkeit, die Zuverlässigkeit und die Sicherheit, die durch die darunter liegenden Protokolle (z.B. TCP) garantiert werden können, zu nutzen.
- Die Knoten im Netz müssen nicht alle „multicast-fähig“ sein.

Die am meisten bekannten Anwendungsebene-Multicasting-Protokolle für MANETS sind AMRoute [Xiea02], PAST-DM [GuMo03] und ALMA (siehe Abschnitt 3). Adhoc Multicast Routing Protocol (AMRoute) benutzt Transportschicht-Verbindungen, um die Multicast Gruppenmitglieder in einem logischen Mesh zu verbinden. Danach bildet es in dem Mesh einen Verteilbaum für die Multicast Datenweiterleitung. Progressively Adapted Sub-Tree in Dynamic Mesh (PAST-DM) bildet ebenfalls ein logisches Mesh, um die Multicast- Gruppen-Mitglieder zu verbinden. Application Layer Multicast Algorithm (ALMA) verzichtet auf ein logisches Mesh und benutzt stattdessen einen logischen Multicast-Baum, um die Gruppenmitglieder zu verbinden. Da ALMA als Vertreter der Anwendungsebenen-Mulicasting-Protokolle für den Vergleich zwischen Multicasting auf Anwendungsebene und Netzwerkebene benutzt wird, wird es im Abschnitt 3 genauer beschrieben.

2 ON-DEMAND MULTICAST ROUTING PROTOKOL (ODMRP)

ODMRP ist ein reaktives Multicast-Routing-Protokol, das die on-demand (bei Bedarf) Techniken anwendet und eine Verbindung erst dann sucht, wenn sie auch benötigt wird, um Overhead zu vermeiden und um die Skalierbarkeit zu verbessern. Es bildet(basierend auf dem Forwarding-Group-Konzept) ein Weiterleitungsmesh für jede Multicast-Gruppe. Forwarding group(Weiterleitungsgruppe) ist eine Menge von Knoten, die für die Multicast-Daten-Weiterleitung zuständig sind. Die Multicast-Daten werden auf dem kürzesten Pfad in dem Mesh zwischen jedem Gruppenmitgliederpaar weitergeleitet. ODMRP benutzt ein Mesh anstelle eines Baumes, um die Nachteile eines Baumes (Z.B. Verkehrskonzentration, häufige Baum-Rekonfiguration und der nicht kürzestete Pfad in einem Verteilbaum(d.h. in einem Baum werden die Daten nicht auf dem kürzesteten Pfad vom Sender zu den Empfängern weitergeleitet)) zu vermeiden. In ODMRP werden keine expliziten Kontroll-Nachrichten benötigt, um einer Gruppe beizutretten oder sie zu verlassen.

2.1 Mullticast-Gruppen und -Routen

Die Gruppen-Mitgliedschaft und die Multicast-Routen werden in ODMRP von dem Sender bei Bedarf, analog zu den on-demand Unicast-Routing- Protokollen, etabliert und aktualisiert. Wenn ein Multicast-Sender Daten zu senden hat, flutet er periodisch das gesamte Netz mit einem Datenpaket namens Join-Query(siehe Bild 1). Wenn ein Knoten ein nicht dupliziertes Join-Query-Paket empfängt, speichert er die ID des Nachbarknotens, von dem er es bekommen hat, und leitet es an alle seine Nachbarn weiter. Wenn der Empfänger des Join-Request-Paket ein Mitglied der adressierten Multicast-Gruppe ist, dann erzeugt oder aktualisiert er den Quelle-Eintrag in seiner Member-Table(Mitglieder-Tabelle) und sendet, so lange gültige Einträge in der Member-Table sind, periodisch ein Join-Table an seine Nachbarn. In die Join-Table sind der Sender des Join-Query und der nächste Knoten, der das Paket übersandte, enthalten. Empfängt ein Knoten eine Join-Table, überprüft er, ob eine der eingetragenen Nächster-Knoten IDs mit seiner eignen ID übereinstimmt. Ist das der Fall, setzt er das FG-Flag und bildet, aus den übereinstimmenden Einträgen, seine eigene Join-Table, die er wiederum an seine Nachbarn sendet. Damit wird er ein Mitglied der Forwarding-Group. Auf diese Weise wird die Join-Table von jedem Forwarding-Group- Mitglied propagiert bis sie den Multicast-Sender, über den kürzesten Pfad, erreicht. Dieser Vorgang baut ein Mesh zwischen den Forwarding-Group-Mitgliedern und erzeugt oder aktualisiert die Routen von Multicast-Sendern zu Multicast-Empfängern. Im Bild 2 wird das Forwarding-Group-Konzept veranschaulicht. Alle Knoten in der Blase sind Multicast-Mitglieder oder Forwarding-Group-Mitglieder, wobei ein Multicast-Empfänger auch ein Forwarding-Group-Mitglied sein kann, wenn er auf dem Pfad zwischen einem Multicast-Sender und einem anderen Multicast-Empfänger ist. Das Mesh bietet verglichen mit einem Baum, mehr Redundanz hinsichtlich des Verbindungsgrades zwischen den Multicast-Gruppenmitgliedern, was dazu führt, dass die Häufigkeit der Rekonfiguration, die durch die Knotenmobilität und die damit einhergehende Änderung der Netztopologie notwendig ist, im Vergleich zu einem Baum relativ gering ist.

2.2 Weiterleitung von Daten und das Verlassen der Gruppe

Empfängt ein Knoten, nach dem Gruppen- und Routen-Bildungsprozess, ein nicht dupliziertes Multicast-Datenpaket und ist sein FG-FLAG für die im Paket adressierten Multicast-Gruppe gesetzt, dann leitet er es weiter. Damit wird das Verkehrs-Overhead minimiert und das Senden der Pakete über veralteten Routen vermieden.

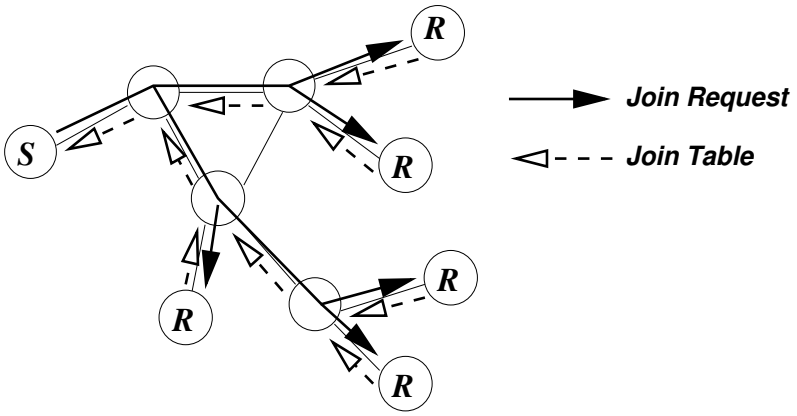


Abbildung 1: On-Demand Verfahren für die Gruppenmitgliedschaft und -aufrechterhaltung.

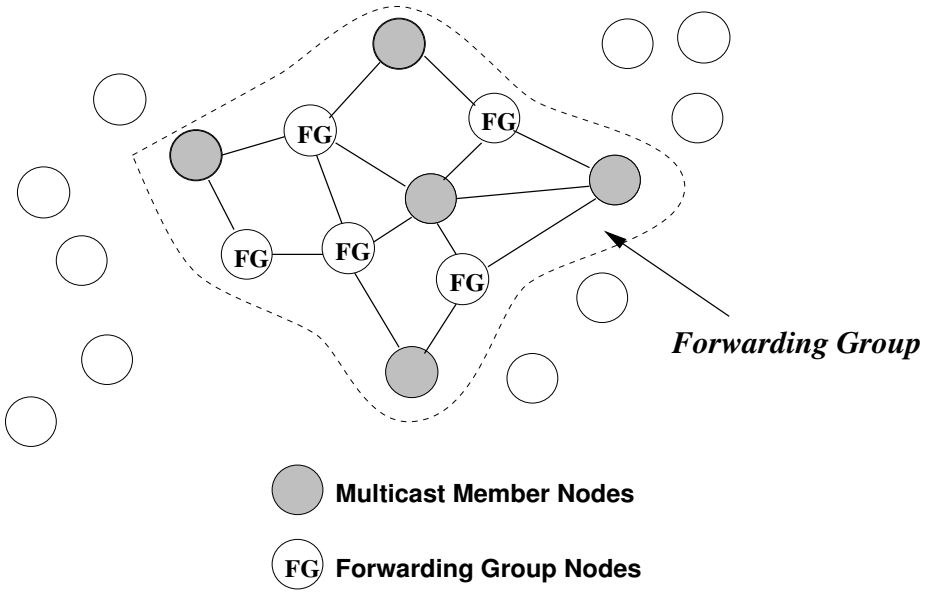


Abbildung 2: Forwarding Group Konzept.

Wenn ein Multicast Sender die Gruppe verlassen will, dann hört er einfach auf, Join-Request-Paketen zu senden. Will ein Empfänger keine Daten mehr von einer bestimmten Multicast-Gruppe empfangen, entfernt er einfach den entsprechenden Eintrag aus seiner Member-Table. Die Forwarding-Group-Knoten verlassen automatisch die Forwarding-Group, wenn sie keine Join-Tables empfangen bevor ihre timeout abgelaufen ist. In [Leea02] finden sich weitergehende Informationen und Simulationsergebnisse zur Leistungsfähigkeit.

2.3 Datenstrukturen

Jeder Knoten, der ODMRP implementiert, muss folgende Datenstrukturen unterstützen.

1. Mitglieder-Tabelle(Member-Table): Jeder Multicast-Empfänger speichert in der Mitglieder-Tabelle für jede Gruppe, an der er sich beteiligt, die Sender-ID und die Zeit

der letzten empfangenen JOIN-REQUEST. Empfängt ein Knoten innerhalb einer Auffrischungsperiode keine JOIN-REQUEST, löscht er den Eintrag aus seiner Mitglieder-Tabelle.

2. Routing-Tabelle(Routing-Table): Eine Routing-Tabelle wird on-demand erzeugt und wird von jedem Knoten aufrechterhalten. Empfängt ein Knoten ein nicht dupliziertes Join-Request, speichert oder aktualisiert er das Ziel(i.e. die Quelle des Join-Request) und den nächsten Hop zum Ziel(i.e. der Knoten, von dem das Join-Request empfangen wurde). Die Routing-Tabelle liefert die Next- Hop-Information, wenn die Join-Table gesendet werden soll.

3. Weiterleitungsgruppe-Tabelle(Forwarding-Group-Table): Jedes Mitglied der Weiterleitungsgruppe speichert die Mulicast-Gruppen-IDs und die Zeit der letzten Auffrischung in der Forwarding-Group-Table.

4. Message Cache: Das Message Cach wird von jedem Knoten benutzt, um duplizierte Daten zu erkennen. Empfängt ein Knoten ein neues Join- Request oder ein neues Datenpaket, speichert er die Quellen-ID und die Sequenznummer des Paketes

3 APPLICATION LAYER MULTICAST ALGORITHM (ALMA)

ALMA ist ein adaptives Empfänger-gesteuertes Protokoll, das einen logischen Multicast-Baum zwischen den Gruppenmitgliedern erzeugt. Um den Overhead gering zu halten, hat ALMA auf die Benutzung eines Meshes verzichtet. Dazu kommt, dass der Hauptvorteil eines Meshes, nämlich die Zuverlässigkeit, die durch ein zuverlässiges Transportschichtprotokoll wie z.B. TCP garantiert werden kann, da ALMA mit jedem beliebigen Protokoll-Stac operieren kann. Jede Kante dieses logischen Baumes stellt eine logische Verbindung dar, die einem Pfad in der Netzwerkebene entspricht. Im Bild 3 ist eine einzige logische Verbindung zwischen den Knoten C und D, die 3 physikalische Verbindungen auf Netzwerkebene entspricht(C-Y, Y-Z, Z-D).

3.1 Empfänger-gesteuerter-Ansatz

Wenn ein Knoten sich an die Gruppe anschließen möchte, sucht sich selbst einen Elternknoten. Nachdem er sich der Gruppe angeschlossen hat, kann ein Gruppemitglied entscheiden, ob er Kindknoten aufnehmen will oder nicht. Empfängt ein Knoten von dem Sender ein Datenpaket, macht er mehrere Kopien davon und leitet zu jeden seiner Kindknoten eine Kopie weiter. Jedes Mitglied ist dafür zuständig, seine Verbindung mit dem Elternteil aufrechtzuerhalten. Fällt die Leislungsfähigkeit einer Verbindung zwischen einem Mitglied und seinem Elternknoten unter einen vordefinierten Schwellenwert, sucht sich das Mitglied einen neuen Elternknoten und führt damit eine lokale Baumrekonfiguration durch.

3.2 Gruppenbeitritt und -austritt

Möchte der Gruppe ein neues Mitglied beitreten, sendet er Join-Messages zu den möglichen existierenden Mitgliedern. Empfängt ein existierendes Mitglied solch ein Join-Message, antwortet er dann, wenn er neue Kindknoten aufnehmen kann und will. Wenn das neue Mitglied mehrere Antworten empfängt, kann er eine davon auswählen. Das Auswahlkriterium kann sehr variieren(siehe Abschnitt 3.3). ALMA fordert einen expliziten Gruppenaustrittsmechanismus.

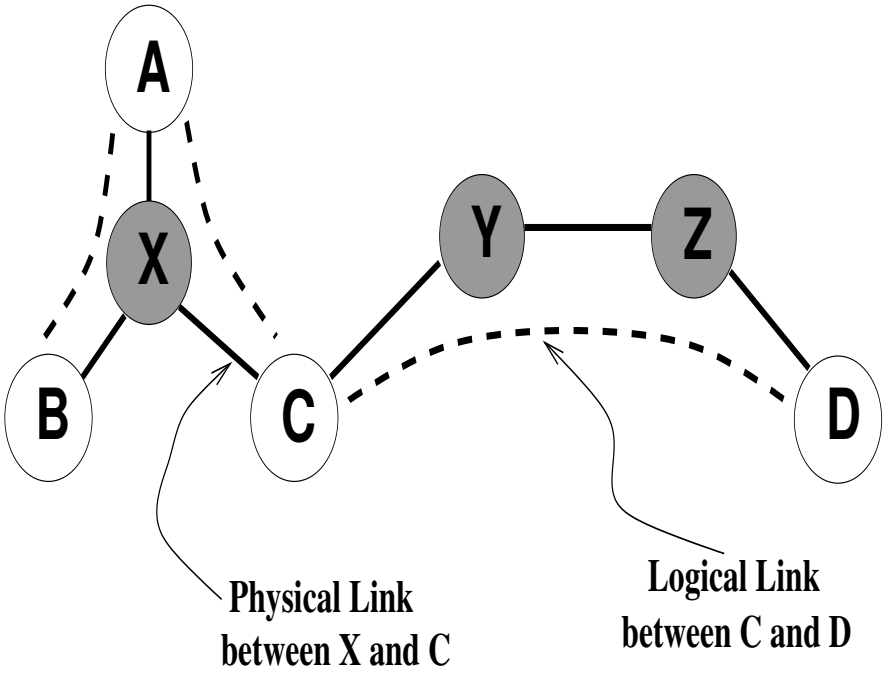


Abbildung 3: Logische Verbindungen vs physikalische Verbindungen.

Möchte ein Mitglied die Multicast-Gruppe verlassen, muss er eine explizite Leave-Message zu seinem Elternknoten und seinen Kindknoten senden. Empfängt ein Elternknoten ein solchen Leave-Message löscht er den Knoten aus seiner Kindknotenliste. Wenn ein Kindknoten eine Leave-Message empfängt, versucht er sich wieder an die Gruppe anzuschließen. Das Verlassen einer Gruppe ohne Ankündigung, wird als einen Knotenausfall betrachtet. Um diese Ereignisse und die Netzwerkpartition zu behandeln, senden die Mitglieder periodisch Hello-Messages zu ihren Elternknoten. Empfängt ein Mitglied innerhalb einer vordefinierten Timeout-Periode keine Antwort von seinem Elternknoten, geht es davon aus, dass sein Elternknoten ausgefallen ist und versucht sich wieder, an die Gruppe anzuschließen.

3.3 Elternknotenauswahl und Baumrekonfiguration

Die Kinder überwachen die Qualität der Verbindungen zu ihren Elternteilen. Stellt ein Kindknoten fest, dass sich die Verbindungsqualität zu seinem Elternknoten verschlechtert hat(was durch das Beitreten neuer Gruppenmitglieder, das Verlassen der Gruppe von existierten Mitgliedern und die Knotenmobilität geschehen kann), versucht er den Elternknoten zu wechseln. Das Kriterium, das zur Messung der Qualität einer logischen Verbindung zwischen zwei Knoten benutzt wird, ist die auf Anwendungsebene gemessene Round-Trip-Time(RTT). Um die RTT zu messen, sendet jedes Mitglied periodisch, wie schon im Abschnitt 3.2 erwähnt wird, Hello-Messages an seinen Elternknoten. Nach Empfang der Antwort von dem Elternknoten, schätzt das Mitglied die durchschnittliche RTT. Übersteigt die durchschnittliche RTT einen vordefinierten Schwellenwert(ein Protokollparameter), versucht das Mitglied einen neuen Elternknoten zu finden, der Daten mit einer geringen RTT liefern kann. Statt die RTT zwischen zwei Knoten als Kriterium für die Baumkonfigurationsqualität zu benutzen, können auch z.B. die Ende-zu-Ende-Verzögerung(als die Summe der durchschnittlichen RTTs von allen logischen Verbindungen zwischen dem Sender und dem Kindknoten) oder die Anzahl der Hops(was allerdings die Anwendungsebenenatur des Protokolls verletzen wird) benutzt wer-

den. Eine detaillierte Beschreibung dieses Protokolls und Simulationsergebnisse finden sich in [GKF].

4 vergleich von Multicast auf Anwendungs- und Netzwerkebene

In diesem Abschnitt wird der Vergleich zwischen ALMA und ODMRP, der von der Autoren Min Ge, Srikanth, and Michalis Faloutsos anhand des Simulators GloMoSim(Globale Mobile Information System [LaLa]) realisiert wurde, vorgestellt. Da ODMRP keinen zuverlässigen Datentransfer wie die am meisten bekannten Netzwerkebene-Routing-Protokolle, garantieren kann, benutzt ALMA das unzuverlässige Transportprotokoll UDP für die logischen Verbindungen, um fair zu bleiben.

4.1 Simulationsszenario

Das simulierte Netzwerk besteht aus 50 mobilen Knoten, die sich nach dem RANDOM-WAY-POINT-MOBILITY Modell auf einem Feld von 1000x1000 Metern bewegen. Jeder Knoten hat eine Reichweite von 250 Metern und eine maximale Bandbreite von 2 MBit/s. Bei dem RANDOM-WAY-POINT-MOBILITY Modell wurden die minimale und die maximale Geschwindigkeit auf den gleichen Wert gesetzt(i.e., die Geschwindigkeit ist für alle Knoten konstant jedoch variabel von einem Simulationsdurchlauf zum anderen) und die Pausezeit beträgt 30 Sekunden. Jeder Knoten schließt sich am Anfang der Simulation der Gruppeund an und bleibt in der Gruppe bis zum Ende der Simulation. Jede Simulation dauert 1000 Sekunden. Die Gruppengröße variiert zwischen den einzelnen Szenarios zwischen 5 bis 40 und die Bewegungsgeschwindigkeit zwischen 0m/s bis 12m/s. Der generierte Verkehr hat eine konstante Bitrate(CBR).

4.2 Leistungsmetriken

Die Leistungsmetriken, die zur Evaluation von ALMA und zum Vergleich mit ODMRP benutzt wurden, sind die Folgender:

1. Packet Delivery Ratio: Das Verhältnis zwischen der Anzahl der tatsächlich zugestellten Datenpakete für einen Multicast-Empfänger und der erwarteten Anzahl von Datenpaketen.

2. Goodput: Die Anzahl der Nutz-Bytes(ausgenommen duplizierte Bytes), die von dem Anwendungsprozess auf einem Empfänger pro Zeiteinheit empfangen wird.

3. Stress: Der Stress einer physikalischen Verbindung ist die Anzahl der Kopien eines selben Multicast-Datenpaketes, die diese physikalische Verbindung überqueren müssen.

4.3 Simulationsergebnisse

Die Simulationsergebnisse wurden nach der Gruppendichte sortiert. Die Gruppendichte ist das Verhältnis zwischen der Anzahl der Multicast- Gruppemitglieder und der Anzahl aller Knoten im Netz. Die drei folgenden Gruppegrößen wurden betrachtet.

1. Mittlere Gruppengröße: In Abbildung 4 wird die Veränderung der Paketzustellrate(Packet Delivery Ratio) in Abhängigkeit der Bewegungsgeschwindigkeit graphisch dargestellt. Aus Abbildung 4 lässt sich ablesen, dass ALMA, für mittlere Gruppengrößen mit einer Gruppedichte von 20% (Gruppengröße von 10 Mitgliedern), im vergleich zu ODMRP ein ausgezeichnetes Goodputs erzielt. ALMA übertrifft ODMPR um fast 15%. Das sei auf zwei Gründen zurückzuführen: Einerseits die Fähigkeit von ALMA Knoten, die hoch belastet sind, zu vermeiden. Dies wird daduch bewältigt, dass der Baum rekonfiguriert wird, wenn die beobachtete RTT zunimmt. Andererseits versucht ODMRP die Hop-Anzahl vom Sender zu jedem Empfänger zu minimieren, was zur Überlastung einiger Knoten führen kann. Ein ähnliches Verhalten ist aus Abbildung 6, wo die Veränderung des Goudputs in Abhängigkeit der Bewegungsgeschwindigkeit graphisch dargestellt ist, abzulesen. ALMA übertrifft ODMPR um fast 20% .

2. Große Gruppengröße: Mit einer Gruppengröße von 20 Mitgliedern(Gruppendichte von 40%) und mit geringen Bewegungsgeschwindigkeiten(von 0m/s bis zu ungefähr 6m/s), siehe Abbildung 5, übertrifft die Leistung des ALMA-Protokolls(hinsichtlich der Paketzustellrate) diejenige von ODMRP-Protokolls. Aber mit zunehmender Geschwindigkeit(ab 6m/s) fällt die Leistung des ALMA Protokolls rapider ab und geht unter diejenige des ODMRP Protokolls, die nur langsamer abnimmt. Diese Leistungsabnahme von ALMA sei u.a. auf den zunehmenden Stress(i.e. die Anzahl der Multicast-Kopien, die die gleiche physikalische Verbindung überqueren) und die dadurch erhöhte Überlastung zurückzuführen. Dazu kommt auch die erhöhte Rekonfigurationsfrequenz, die wiederum die Anzahl der Kontrollpakete erhöht. Goodput von ALMA und von ODMRP sind immer gleich über das Intervall der betrachteten Geschwindigkeiten. Dies wurde allerdings aus Platzgründen in der Literatur nicht graphisch dargestellt.

3. Extrem große Gruppengröße: Mit zunehmender Gruppengröße(siehe Abbildung 7) nimmt die Leistung von ALMA ab, während diejenige von ODMRP zunimmt. ALMA übertrifft ODMRP für Gruppengrößen zwischen 5 Mitgliedern bis zu ungefähr 23 Mitgliedern. Ab Gruppengrößen von über 23 Mitgliedern ist das Gegenteil der Fall.

4.3.1 Schluss

Aus den vorgestellten Simulationsergebnissen geht Folgende hervor:

- ALMA eignet sich besser als ODMRP für Szenarien mit kleinen bis mittleren Gruppengrößen und kleinen Geschwindigkeiten.
- ALMA skaliert schlecht mit zunehmender Gruppengröße und ist nicht robust gegen Geschwindigkeit.
- ODMRP eignet sich besser als ALMA für Szenarien mit großen Gruppengrößen und großen Geschwindigkeiten.
- ODMRP skaliert gut mit zunehmender Gruppengröße und ist robust gegen Geschwindigkeit.

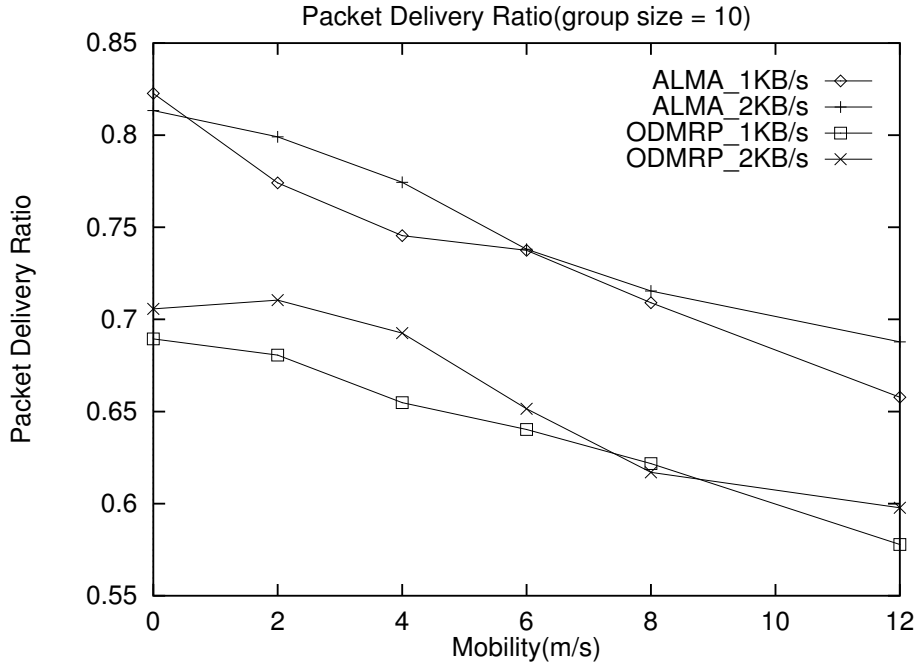


Abbildung 4: ALMA Autoren: Paket Delivery Ratio gegen Geschwindigkeit(Gruppegröße=10).

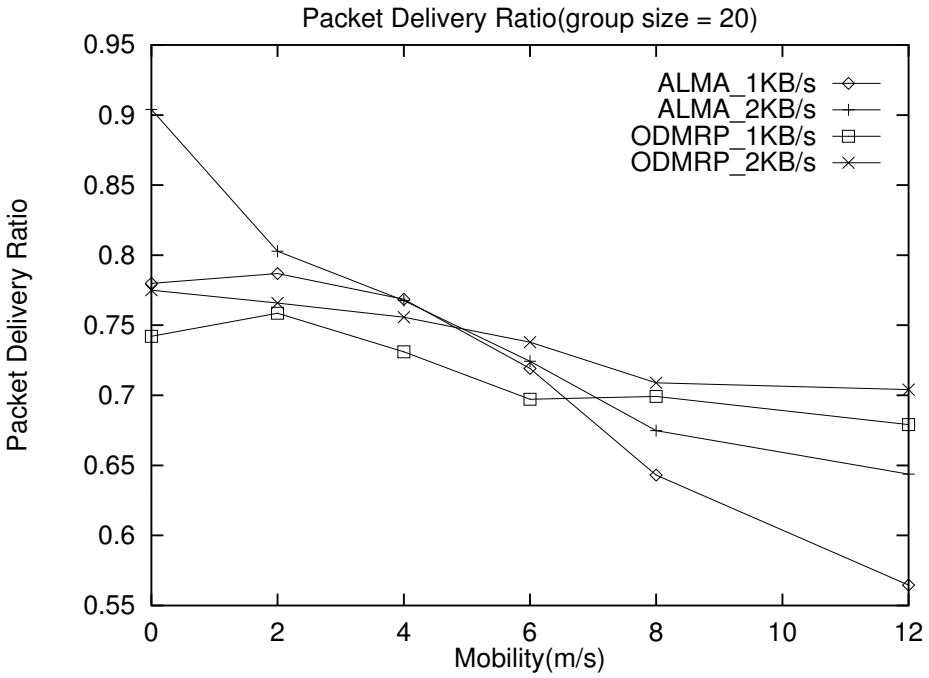


Abbildung 5: ALMA Autoren: Paket Delivery Ratio gegen Geschwindigkeit(Gruppegröße=20).

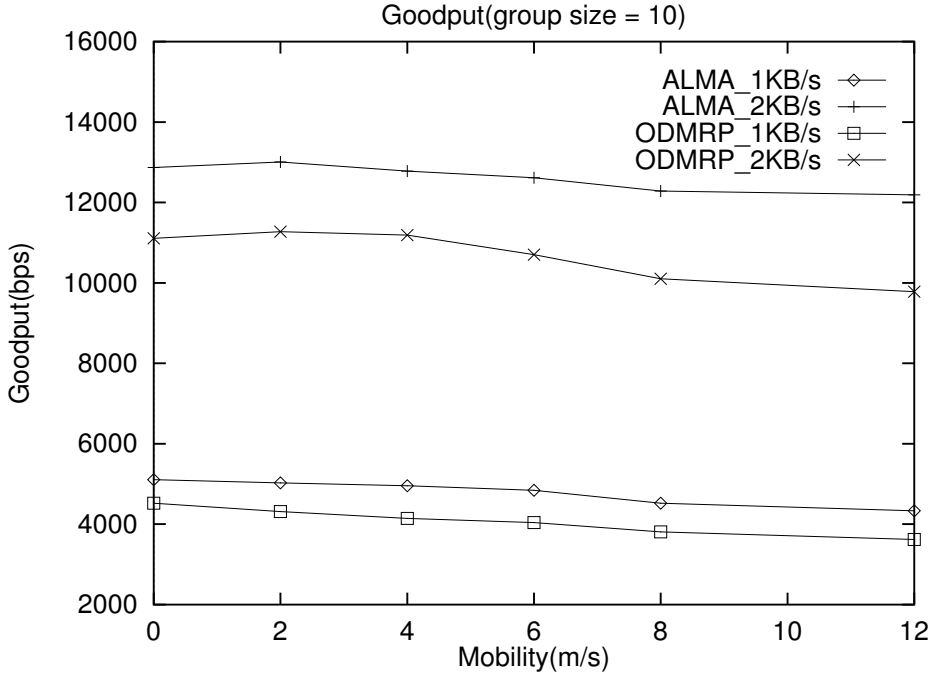


Abbildung 6: ALMA Autoren: Goodput gegen Geschwindigkeit(Gruppegröße=10).

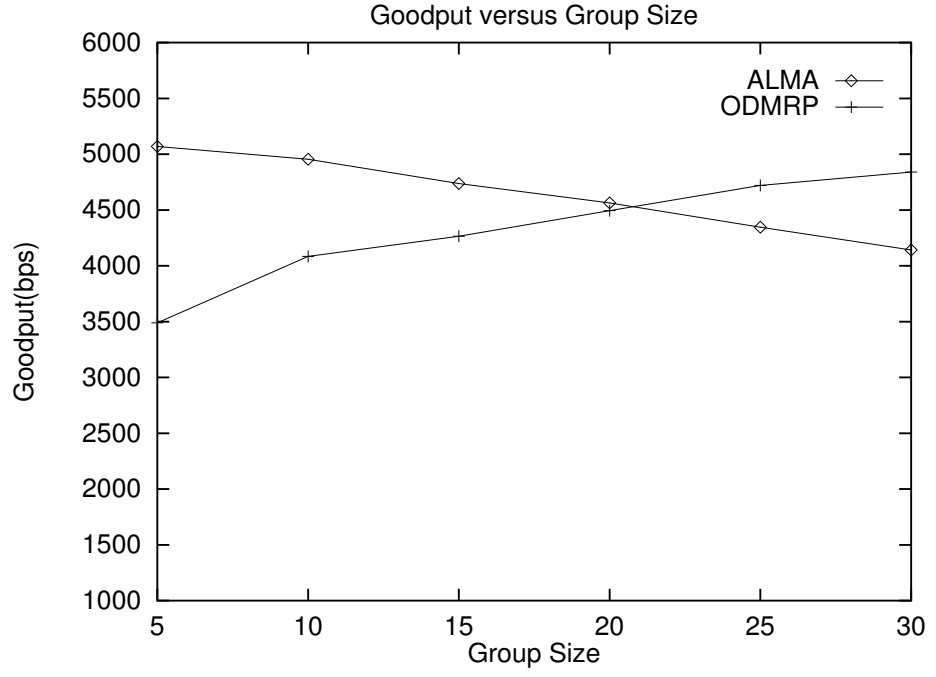


Abbildung 7: ALMA Autoren: Goodput gegen die Gruppegröße(Geschwindigkeit=6m/s).

Literatur

[E.Pe99] E.Royer und C.E. Perkins. „Multicast Operations of the Ad-hoc On-Demand Distance Vector Routing Protocol“. Proceedings of ACM/IEEE MOBICOM'99, August 1999.

[GKF] Min Ge, Srikanth V. Krishnamurthy, und Michalis Faloutsos. „Overlay Multicasting for Ad Hoc Networks“. Department of Computer Science and Engineering, University of California, Riverside, CA, 92521.

[GuMo03] C. Gui und P. Mohapatra. „Efficient Overlay Multicast for Mobile Ad Hoc Networks“. Proceedings of IEEE WCNC 2003, März 2003.

[JLAea99] J.J.Garcia-Luna-Aceves und et al. „The Core-Assisted Mesh Protocol“. *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks* vol.17(No.8), August 1999.

[LaLa] UCLA Parallel Computing Laboratory und Wireless Adaptive Mobility Laboratory. *GloMoSim: A Scalable Simulation Environment for Wireless and Wired Network Systems.*

[Leea02] S.J. Lee und et al. „On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks“. *ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communications in Wireless Mobile Networks* vol.7(No.6), Dezember 2002.

[WuTa99] C.W Wu und Y.C Tay. „ARMIS: A Multicast Protocol for Ad hoc Wireless Networks“. Proceedings of IEEE MILCOM'99, November 1999.

[Xiea02] J. Xie und et al. „AMRoute: Ad Hoc Multicast Routing Protocol“. *ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communications in Wireless Mobile Networks* vol.7(No.6), Dezember 2002.

Abbildungsverzeichnis

1 On-Demand Verfahren für die Gruppenmitgliedschaft und -aufrechterhaltung. 51

2 Forwarding Group Konzept. 51

3 Logische Verbindungen vs physikalische Verbindungen. 53

4 ALMA Autoren: Paket Delivery Ratio gegen Geschwindigkeit(Gruppegröße=10). 56

5 ALMA Autoren: Paket Delivery Ratio gegen Geschwindigkeit(Gruppegröße=20). 56

6 ALMA Autoren: Goodput gegen Geschwindigkeit(Gruppegröße=10). 57

7 ALMA Autoren: Goodput gegen die Gruppegröße(Geschwindigkeit=6m/s). . 57

Unterstützung mobiler Netze mit Mobile IPv6

Huiming Yu

Kurzfassung

Mobilität wird immer wichtiger in unserem Leben. Das Handy wird schon ein untrennbarer Teil von unserem Leben und Arbeiten geworden. Mit Laptop, PDA, Handy oder anderen Geräten drahtlose ins Internet einzusteigen ist die heutige und zukünftige Tendenz. Mobile Netze sind die nächste Anforderung von Mobilität nach mobilen Knoten. In diesem Artikel werden die Definitionen und Funktionsweise von mobilen Netzen besprochen.

1 Einleitung

Was ist ein mobiles Netz? Haben Sie ein Handy? Ja, natürlich. Und wenn Sie auch ein PDA haben, so reicht das. Über Bluetooth Technik verbindet sich PDA mit dem Handy, und dieses verbindet sich mit dem zellulären Netz. So kann man mit PDA das Internet besuchen. Um z. B Email zu senden oder zu empfangen usw. Das ist ein einfaches Beispiel eines mobilen Netzes und das Handy funktioniert hier als ein mobiler Router. Das Handy ist ein Knoten eines mobilen Netzes.

Wenn Sie mit dem Zug in eine andere Stadt fahren, verbinden Sie Ihren Laptop mit dem von dem Zug angebotenen Internetzugang und können im Internet surfen. Das ist auch ein Beispiel eines mobilen Netzes.

Mobile Netze haben noch zwei andere Anwendungsszenarien. Das Erste ist ein mobiles Ad-hoc-Netz mit einem mobilen Router, mit dem das mobiles Ad-hoc-Netz Internetzugang hat. Das Zweite ist ein Fahrzeug, welches die mehrere Netzwerkknoten enthält und Internetverbindung ermöglicht.

Ein mobiles Netz ist ein Netz, das auch in Bewegung seine Zugangspunkt zum Internet und seine Erreichbarkeit in der Topologie dynamisch wechseln kann. In einem mobilen Netz gibt es ein oder mehrere mobilen Subnetze, welche über ein oder mehrere mobile Router mit dem Internet verbunden sind. Die interne Struktur eines mobilen Netzes ist relativ stabil. Die Bewegung dieses Netzes hat keine Effekte für die darin befindlichen Knoten.[ErLa04]

- Mobiler-Knoten

  Ein Rechner oder ein Router, welcher über eine Heimatadresse ständig erreichbar ist, auch beim Wechseln des Internet Zugangspunkt.

- Mobile Router

  Ein Router, der dynamisch den Zugangspunkt zum Netz wechseln und die Nachrichten zwischen zwei oder mehrere Schnittstellen übertragen kann.

- Home-Adresse

  Im Heimatnetz bekommt ein mobiler Knoten eine Heimatadresse, über die er ständig erreichbar ist.

- Home-Agent

  Ein Router, der sich im Heimatnetz befindet. Er speichert die Informationen über mobile Knoten und mobile Router, wenn sie nicht im Heimatnetz sind und er kann die Nachrichten in mobilen Knoten oder mobile Netze weiterleiten.

- Heimatnetz

  Das Netzsegment, in dem der mobile Knoten oder das mobile Netz sich ursprünglich befindet.

- Fremdnetz

  Das Netzsegment, das kein Heimatnetz von einem mobilen Knoten oder mobilen Netz ist.

- Korrespondierender Knoten

  Ein Rechner oder Router, der mit dem mobilen Knoten oder mobilen Netz kommuniziert. Er kann ein mobiler Knoten oder ein fixer Knoten sein.

- c/o-Adresse(care of address)

  Eine IP Adresse, die mobile Knoten oder mobile Router im Fremdnetz konfigurieren. Der Heimatagent leitet die Nachrichten zu dieser Adresse weiter.

- Präfix des mobilen Netzes

  Eine Bit-Kette, die einige primäre Bits von einer IP-Adresse besitzt, welche das mobile Netz im ganzen Internet identifizieren kann. Es konfiguriert auf dem Heimatagent.

2 Mobile Knoten und Mobile Netze

Über mobile IPv6 geben mobile Knoten die Möglichkeit, einen Knoten(z.B. ein Laptop oder ein Handy) immer zu erreichen auch während der Bewegung zwischen unterschiedlichen Netzen. Mobile Netze ermöglichen es, dass ein ganzes Netz dynamisch ihre Internet Zugangspunkt wechseln kann, aber die Knoten des mobilen Netzes bleiben ständig erreichbar.

2.1 Mobile Knoten

Ein mobiler Knoten kann ein Rechner oder ein Router sein, aber der Router muss auch als ein normaler Knoten funktionieren. Mobile IPv6 kann so mobile Knoten gut unterstützen. Ein mobiler Knoten muss ein von mobilem IPv6 unterstützter Knoten sein, aber der korrespondierende Knoten kann auch ein fixer Knoten sein. Der mobile Knoten besitzt eine Heimatadresse und noch eine c/o-Adresse wenn er sich außerhalb des Heimatnetzes befindet. Er wird durch diese Heimatadresse identifiziert.

Wenn der mobile Knoten sich aus dem Heimatnetz bewegt und eine c/o-Adresse vom anschließenden Fremdnetz bekommt, registriert er sich beim Heimatagent und schickt dem Heimatagent seine aktuelle c/o-Adresse. Der Vorgang heißt Binding-Update. Mit einer so genannten Binding-Acknowledgement Vorgang bekommt der mobile Knoten die Bestätigung.

Beim Binding-Update Vorgang schickt der mobile Knoten seinem Heimatagent ein leeres IP Paket, das die Option des Ziel enthaltet. Alle Pakete, die ein Binding Update enthalten, müssen auch eine Option der Heimatadresse enthalten.

Die Felder haben folgende Bedeutungen:

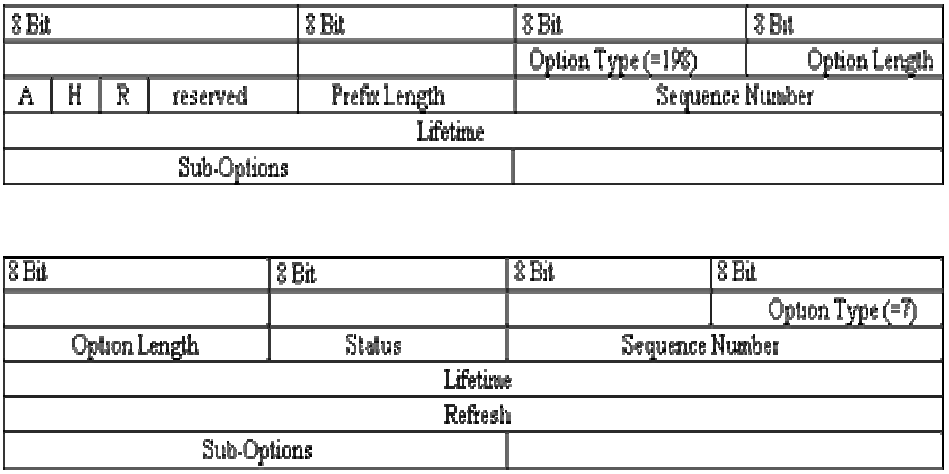


Abbildung 1: Binding Update und Binding Acknowledgement

- Option Length

  Länge der Option in octets ohne Längen und Type Feld

- Acknowledge (A)

  Es wird benutzt, um den Home Agent zum Senden eines Binding-Acknowledgement aufzufordern.

- Home Registration (H)

  Es wird benutzt, um den Empfänger aufzufordern als Home Agent für die mobile Station zu agieren.

- Router (R)

  Wenn R auf eins gesetzt wird, zeigt es, dass die mobile Station ein Router ist; dieses Bit darf nur zusammen mit dem H Bit gesetzt werden.

- Prefix Length

  Es darf nur in Verbindung mit dem H Flag verwendet werden; Hier wird die Präfixlänge der Heimatadresse übergeben. Diese wird vom Router dazu verwendet, alle anderen Heimatadressen des mobilen Benutzers zu errechnen.

- Sequence Number

  Es dient zur Sicherung gegen Fälschungen.

- Lifetime

  Es gibt die Gültigkeit der Bindung in Sekunden an.

- Sub-Options

  Zusätzliche Optionen, welche bei Bedarf übergeben werden können.

Die Felder des Binding-Acknowledgement entsprechen den beim Binding-Update.

Der Refresh Wert zeigt an, dass wie viele Sekunden der mobile Knoten ein Paket der Aktualisierung für Binding-Update schicken sollt.

Der Status Feld hat folgende Bedeutungen:

0 Bindung wurde erfolgreich aufgebaut.

128 unbekannter Fehler.

130 Bindung wurde vom Administrator untersagt.

131 keine ausreichenden Ressourcen vorhanden.

132 Heimatagent Funktion wird nicht unterstützt.

133 Rechner liegt nicht in diesem Subnetz.

135 Antwort auf dynamische Anforderung.

136 falsche Längen der Identifizierung.

137 kein Heimatagent für den mobile Knoten.

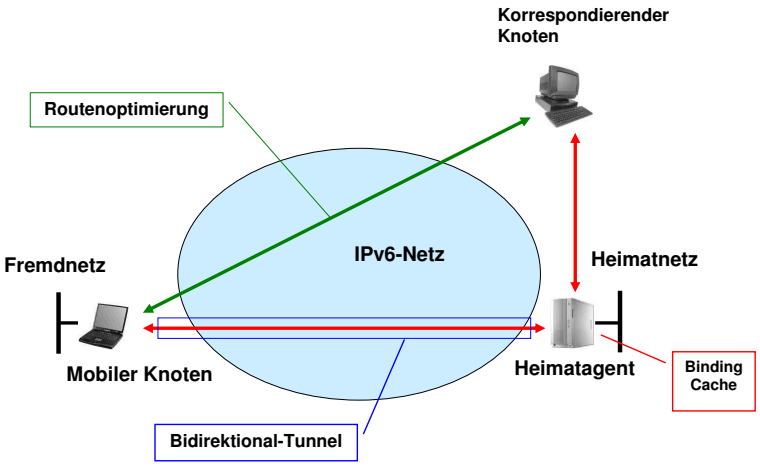


Abbildung 2: Mobile Knoten

Ein korrespondierender Knoten schickt ein Paket an den mobilen Knoten und das Paket erreicht zuerst den Heimatagent des mobilen Knoten. Der Heimatagent sucht die Heimatadresse des mobilen Knoten im Binding Cache. Wenn es gefunden ist, wird das Paket durch den Heimatagent zur die c/o-Adresse des mobilen Knoten, die sich mit seinem Heimatadresse verbindet, weiterleitet.

Es ist möglich, dass ein Antwortpaket von diesem mobilen Knoten direkt zum korrespondierenden Knoten geschickt wird. Das Antwortpaket enthält seine c/o-Adresse. So können

der mobile Knoten mit dem Anderen direkt kommunizieren. Das heißt die Optimierung der Route. Der Tunnel zwischen dem mobile Knoten und dem Heimatagent heißt bidirektionale Tunnel.

2.2 Mobile Netze

Die Realisierung eines mobilen Netzes ist viel komplizierter als mobile Knoten. Ein mobiles Netz kann ein oder mehrere Subnetze haben. Die Knoten, die sich in dem mobilen Netz befinden, heißen Knoten des mobilen Netzes. Diese Knoten des mobilen Netzes können lokale fixe Knoten, lokale mobile Knoten und fremde mobile Knoten sein.

Nur über einen oder mehrere mobile Router können die Knoten des mobilen Netzes an das Internet angeschlossen werden. Ein mobiler Router hat eine oder mehrere Ausgangsschnittstellen, die an dem Heimatnetz oder Fremdnetz anschließen und eine oder mehrere Eingangsschnittstellen, die an das mobile Netz anschließt.

Der mobile Router kann auch als ein mobiler Knoten funktionieren. Wenn R auf Null gesetzt wird, arbeitet er als ein mobiler Knoten, und setzt man R auf Eins, so arbeitet er als mobile Router.

Wenn der mobile Router als ein mobiler Knoten sich bei seinem Heimatagent meldet, leitet der Heimat Agent nur die Nachrichten weiter, die von korrespondierenden Knoten zum mobilen Router geschickt wird. Die Nachrichten, die nach die Knoten des mobilen Netzes geschickt werden, leitet der Heimat Agent nicht weiter. Wenn er als ein mobiler Router arbeitet, leitet der Heimat Agent alle Nachrichten weiter.

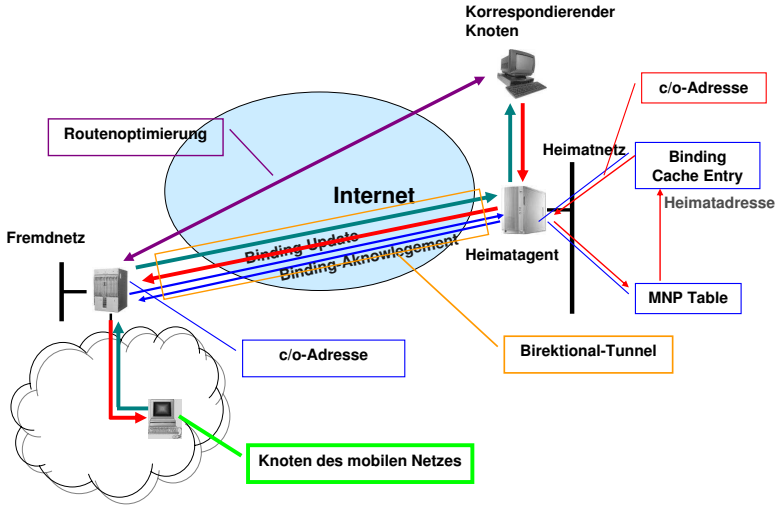


Abbildung 3: Mobile Netz

Um für alle Knoten des mobilen Netzes, die sich in einem mobilen Netz befinden ständige Erreichbarkeit zu haben, muss die Bewegung des mobilen Netzes für die Knoten des mobilen Netzes transparent sein. Es soll erreicht werden, dass die Knoten des mobilen Netzes keine andere besondere Unterstützungsfunktion mehr brauchen. Alle Knoten des mobilen Netzes haben eine eigene permanente IP Adresse.

Mit mobilem IP hat man schon die Lösung für den mobilen Knoten gefunden und die Lösung ist auch hilfreich für die Realisierung des mobilen Netzes. Aber nur mit mobilem IPv6 erreicht die Unterstützung des mobilen Netzes eigentlich nicht. Ein korrespondierender Knoten schickt ein Paket an den mobilen Knoten und das Paket erreicht zuerst den Heimatagent des mobilen Knoten. Aber der Heimatagent kann nicht die Heimatadresse des mobilen Knoten im "Binding Cacheffinden. So das Paket kann nicht vom Heimatagent zu den Knoten des mobilen Netzes weiterleiten.

Mobile IPv6 ist der Grundstein für das mobile Netz. Die Arbeitsgruppe NEMO(Network Mobility) versucht die Unterstützung des mobilen Netze in zwei Phasen: grundlegende Unterstützung und erweiterte Unterstützung.

3 Anforderungen des mobilen Netzes

3.1 Kompatibilität

Die Unterstützung des mobilen Netzes basiert auf der Lösung von mobilen Knoten, die durch mobile IPv6 realisiert wird. Mobile Knoten und mobile Netz erfüllen unterschiedliche Anforderungen. Sodass das Protokoll, welches das mobile Netz unterstützt, voraussichtlich noch für längere Zeit neben dem mobilen IPv6 existieren muss.

Heutiges Internet benutzt noch IPv4. IPv4 wird nicht in kurze Zeit durch IPv6 ersetzt werden. Es ist möglich, dass das MRHA Tunnel über solches von IPv4 unterstützte Internet aufbauen wird. Deshalb ist die Fähigkeit angefordert, dass die Nachrichten durch solche im oben beschriebenen Internet durchlaufen können.

Unterschiedliche physikalische Medien müssen auch von mobilem Netz unterstützt werden können, z.B. Funk, Bluetooth usw.

3.2 Sicherheit

Sicherheit ist immer ein wichtiges Thema. Für die Unterstützung des mobilen Netzes müssen Mechanismen für Sicherheit vorlegen.

In der grundlegenden Unterstützung des mobilen Netzes müssen der korrespondierende Knoten und der Knoten des mobilen Netzes über dem Heimatagent kommunizieren. Der Heimatagent bekommt die c/o-Adresse durch das Binding Update. Aber der aktuelle Standort des mobilen Netzes muss für alle anderen Knoten unbekannt sein.

Ein Mechanismus der Authentifizierung wird angefordert. Diese überprüft, ob der Sender authentifiziert wird.

3.3 Transparenz

Die Mobilität muss für die Knoten des mobilen Netzes transparent sein. Während das mobile Netz seinen Zugangspunkt zum Internet wechselt, müssen alle Knoten des mobilen Netzes ständige Erreichbarkeit mit dem Korrespondierenden Knoten beibehalten. Die durch die Übergabe entstehenden Störungen der Applikationen müssen wie möglich minimiert werden. Beispiele sind der Verlust oder die Verspätung von Paketen.

3.4 Verschachtelte Topologie

Diese Knoten des mobilen Netzes können lokale fixe Knoten, lokale mobile Knoten und fremde mobile Knoten sein. Der Knoten des mobilen Netzes kann auch ein Router sein. Sodass können die mobile Knoten oder mobile Netz mit dem Router anschließen. Die Topologie in dem mobilen Netz kann sehr kompliziert sein und die Zahl der verschachtelten Ebenen kann beliebig groß. [Erns04]

4 Funktionsweise der grundliegender Unterstützung

Die Unterstützung mobiler Netze hat zwei Phasen: grundlegende Unterstützung und erweiterte Unterstützung. Die grundlegende Unterstützung ist direkt aus der Unterstützung mobiler Knoten (mobile IPv6) entwickelt. Aber die erweiterte Unterstützung ist bisher nur in einer Planungsphase. Mit grundlegender Unterstützung kann ein Knoten des mobilen Netzes mit einem korrespondierenden Knoten durch bidirektionale Tunnel kommunizieren. Route-Optimierung wird bei erweiterter Unterstützung entwickelt. In diesem Kapitel wird nun die Funktionsweise der grundlegenden Unterstützung diskutiert.

Die Knoten des mobilen Netzes können mobile Knoten und fixe Knoten sein. Alle Knoten des mobilen Netzes können nur durch mobile Router, welche die Mobilität des mobilen Netzes verwaltet, ans Internet angeschlossen werden. Der mobile Router kann als mobile Host oder mobile Router funktionieren. Ein Feld (R) wird dafür in das Binding Update eingesetzt. Wenn das Feld (R) auf Eins gesetzt wird, bedeutet es, dass der mobile Router als ein mobiler Router funktioniert; und auf Null gesetzt, als einem mobilen Host. Wir diskutieren im Folgenden solche Szenarien, bei denen der mobile Router als ein mobiler Router funktioniert. [DeWT04]

4.1 Binding Update

Der mobile Router besitzt eine Heimatadresse, mit der der mobile Router für seinen Heimatagenten erreichbar ist. Es ist möglich, dass der mobile Router ein oder mehrere Präfix des mobilen Netzes konfigurieren kann; Und es ist auch möglich, dass das Präfix dynamisch (z. B durch DHCPv6) konfiguriert wird. Wenn das mobile Netz sein Heimatnetz verlässt, verbindet es sich mit einem Fremdnetz, das an einen andere Access Router angeschlossen ist. Wenn der mobile Router eine c/o Adresse Von dem Fremdnetz zugewiesen wird, schickt er sofort eine Nachricht zu seinem Heimatagent. Die Nachricht enthält das (R) Feld, Präfix Information, Mobilitätseinstellungen und seine neue c/o-Adresse.

Das Binding Update, das von einem mobilen Router geschickt wird, ist ähnlich wie das von einem mobilen Knoten. Aber es enthält mehr Informationen. Ein neue Feld (S) wird definiert. (siehe Abb. 4)

Prefix Status(S): Das Feld S wird von dem mobilen Router gesetzt, wenn der mobile Router die Liste der Präfix anfordert. Wenn S gesetzt wird, müssen A und R auch gesetzt werden.

Die Option des Präfixes des mobilen Netzes oder die Option, die eine Anforderung des Präfixes anzeigt, können im Binding Update enthalten sein.

Die Abbildung zeigt das Format der Option des Präfixes des mobilen Netzes. (siehe Abb. 4)

- Persistent (P)

  Das Feld wird gesetzt, wenn der mobile Router das Präfix in einem langen Zeitraum, der länger als die Gültigkeit des Binding Update ist, beinhalten will.

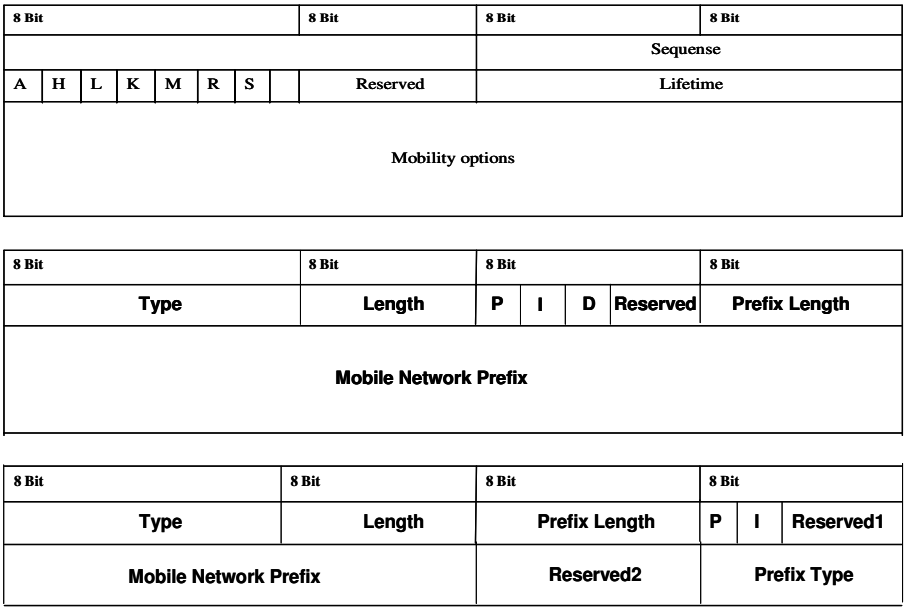


Abbildung 4: Binding Update, Option des Präfix und Option des Präfixanforderung

- Implicit (I)

  Wenn das Feld gesetzt wird, bedeutet es, dass das Präfix den mobile Router zuordnet und leiten lassen will.

- Delegated(D)

  Es zeigt an, dass das vorher konfigurierte Präfix eigentlich schon installiert ist und von dem mobilen Router leitbar ist.

Die Abbildung zeigt das Format der Option, die eine Anforderung des Präfixes anzeigt. (siehe Abb. 4)

Prefix Type:

0 Nicht definiert.

1 Privat

2 Ausschließlich lokal 3 Global

Wenn ein dynamisches Route Protokoll nicht ausgeführt wird, informiert der mobile Router in zwei Moden (implizite und explizite Mode) den Heimatagent, wie das Präfix des mobilen Routers erkennt werden soll. Das Präfix ist nicht im Binding Update enthalten, wenn der mobile Router in der impliziten Mode funktioniert oder er eine dynamische Route Protokoll benutzt. Ein oder beide Moden müssen vom mobilen Router adoptiert werden.

- Implizite Mode

In diesem Mode wird keine Präfix Information in das Binding Update eingesetzt. Ein beliebiger Mechanismus kann von dem Home Agent adoptiert werden, z.B manuelle Methode.

- Explizite Mode

  In diesem Mode werden ein oder mehrere Präfix Einstellungen in das Binding Update eingesetzt, und dort steht die Information des Präfix des mobilen Netzes.

4.2 Bindung Bestätigung(Binding Acknowledgement)

Der Heimatagent bekommt das Binding Update vom mobilen Router und dann wird eine Prüfung des Binding Updates durchgeführt. Es wird geprüft, ob das Heimat Anmeldung Symbol (H) gesetzt ist und ob die in dem Binding Update gesetzte Heimat Adresse zu dem Präfix passt.

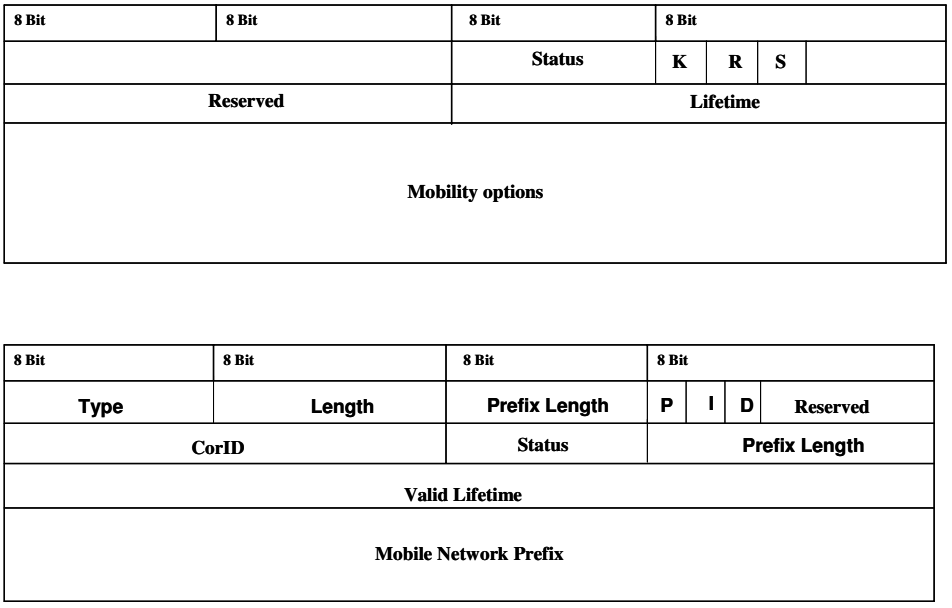


Abbildung 5: Binding Acknowledgement und Prefix Confirmation Option

In der Binding Acknowledgement wird ein Status Code eingesetzt. Dieser hat unterschiedlichen Wert und Bedeutung. Es folgt eine Abbildung, die das Format des Binding Acknowledgement anzeigt. Wenn der Status Code auf 0 gesetzt wird, bedeutet es, dass der Binding-Update Vorgang erfolgreich war. 140 bedeutet, dass der mobile Router nicht als mobile Router funktionieren kann. 141 bedeutet ungültiges Präfix. 142 meint, dass der mobile Router keine Erlaubnis für die Verwendung der Heimat Adresse hat. Wenn der Aufbau der Weiterleitung verfehlt wird, wird der Status Code auf 143 gesetzt. Ein Wert weniger als 128, bedeutet den Erfolg des Binding-Update Vorgangs. (siehe Abb. 5)

Der Heimatagent beinhaltet die Binding Cache Entity eines jeden mobile Router, der bei dem Heimatagent angemeldet ist. Das R Feld wird auch in der Binding Cache Entität gespeichert.

Eine Mobile Network Prefix Confirmation Option wird in diesem Binding Acknowledgement eingesetzt. Die folgende Abbildung zeigt ihr Format Wenn der mobile Router die richtige Binding-Acknowledgement bekommt, wird ein bidirektionaler Tunnel zwischen dem mobile Router und dem Heimat Agent hergestellt. Die zwei Endpunkte des Tunnels sind c/o Adressen des mobilen Routers und die Adresse des Heimatagenten. Unterschiedlicher Wert des Status hat unterschiedliche Bedeutung. (siehe Abb. 5)

0 OK.

1 Präfix ist zurzeit nicht angemeldet.

2 Die Option des mobilen Netzes wird nicht vom mobilen Router geschickt.

3 Ungültiges Präfix.

4 Präfix kann nicht vom Heimatagent konfiguriert werden.

5 Präfix gehört nicht zu diesem mobilen Router.

6 Route wird nicht hergestellt.

7 Die Konfiguration der Länge des Präfixes wird nicht erlaubt.

8 Persistente Präfix wird nicht unterstützt.

9 Flüchtiges Präfix wird nicht unterstützt.

10 Das Präfix Impliziter Modus wird nicht unterstützt.

Prefix Type:

0 Nicht definiert.

2 Ausschließlich lokal

3 Global

Die Tabelle des Präfixes untersucht die Attacken, die aus fremden mobilen Routern kommen. Darin werden die Heimatadresse eines mobilen Routers und das Präfix eines mobilen Netzes einem mobilen Router dargestellt, der abhängig von dieser Heimatadresse ist.

4.3 Weiterleiten

Ein korrespondierender Knoten will mit einem Knoten des mobilen Netzes kommunizieren. Das Paket zum Knoten des mobilen Netzes erreicht zuerst den Heimatagent. In diesem Paket wird die Heimatadresse des Knoten des mobilen Netzes eingesetzt. Dadurch bekommt der Heimatagent das Präfix des mobilen Netzes, in dem sich der Knoten des mobilen Knoten befindet. Die Tabelle des Präfixes enthält die Bindung zwischen die Heimatadresse des mobilen Routers und die Präfixe des mobilen Netzes. Wenn das Präfix in der Tabelle des Präfixes finden kann, wird die Heimatadresse des mobilen Routers ausgegeben. Dann der Heimatagent sucht c/o-Adresse in der Binding Cache. Wenn treffen, wird das Paket von dem Heimatagent mit die Adresse des Herkunft und die Zieladresse in IPv6 Kopf einkapselt und zur die c/o Adresse des mobile Router weitergeleitet. Der mobile Router bekommt das Packet und schickt es zur Schnittstelle, die mit dem mobile Netz verbunden ist. Schließlich erreicht das Paket den mobilen Netz Knoten. [PeJu03]

4.4 Returning Home

Wenn der mobile Router nach Hause will, muss er mit seinem Heimatagent abmelden. Der mobile Router schickt eine Nachricht zu seinen Heimatagent, und der Heimatagent löscht die Entität des mobilen Routers in der Tabelle des Präfix und Binding Cache.

5 Heimatnetze mobiler Router im Detail

Das Heimat Netz kann mit vier Moden organisiert werden. Das sind erweiterte Heimat Netz, gesamte Heimat Netz, virtuelle Heimat Netz und mobile Heimat Netz. [ThWD04]

5.1 Erweiterte Heimatnetze

Erweiterte Heimatnetze behalten ein physikalisches Heimatnetz und ein paar mobile Netze. Das Heimat Netz und die mobile Netze überdecken sich nicht gegenseitig. Dadurch ist es möglich, dass mobile Knoten und mobile Router koexistieren. Die Länge des Präfixes von dem erweiterten Heimatnetz ist größer als die Länge des Präfixes von dem Heimatnetz oder der mobilen Netze. Z.B die Länge des Präfix von dem erweiterte Heimat Netz ist 48 Bits, und von dem Heimat Netz oder der mobile Netze sind 64 Bits.

Wenn ein mobiler Router, der mit einem Fremdnetz anschließt, daheim rückführen will, braucht er nur den bidirektionale Tunnel absetzen und direkt wieder mit seinem Heimatnetz anschließen.

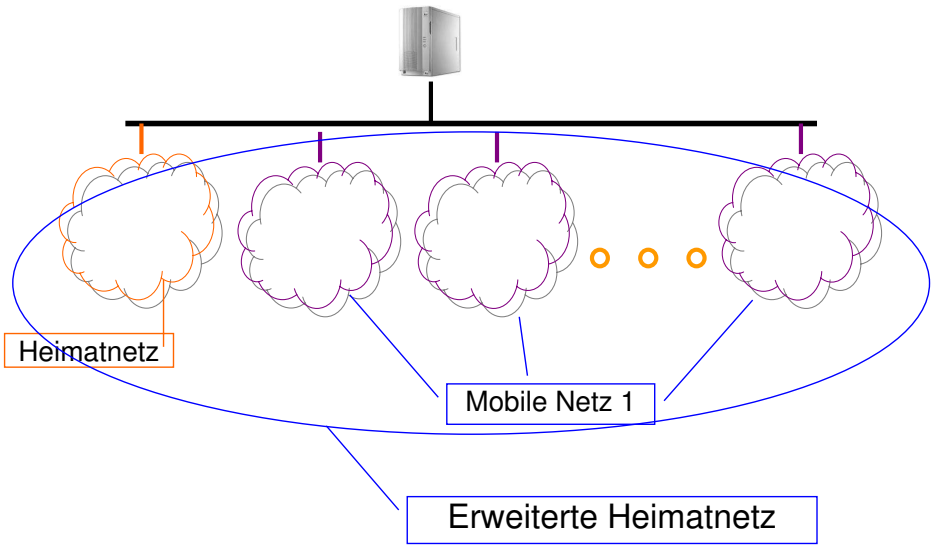


Abbildung 6: Erweiterte Heimatnetz

5.2 Gesamte Heimatnetz

Alle mobilen Netze bilden das gesamte Heimatnetz. Genauso wie bei erweiterten Heimatnetz ist das Präfix des erweiterten Heimatnetzes länger als die Präfixe der mobilen Netze. Z.B 56 von dem gesamten Heimatnetz und 64 von den mobilen Netzen.

Wenn ein mobiler Router daheim rückführen will, gibt es zwei Möglichkeiten. Die erste Möglichkeit ist über die Ausgangsschnittstelle. In diesem Modus wird eine Brücke zwischen dem Heimatnetz und der Eingangsschnittstelle des mobilen Routers hergestellt. Die andere Möglichkeit ist über die Eingangsschnittstelle. In dem Modus kann der mobile Router durch dem Fremd Link mit dem Heimat Link anschließen.

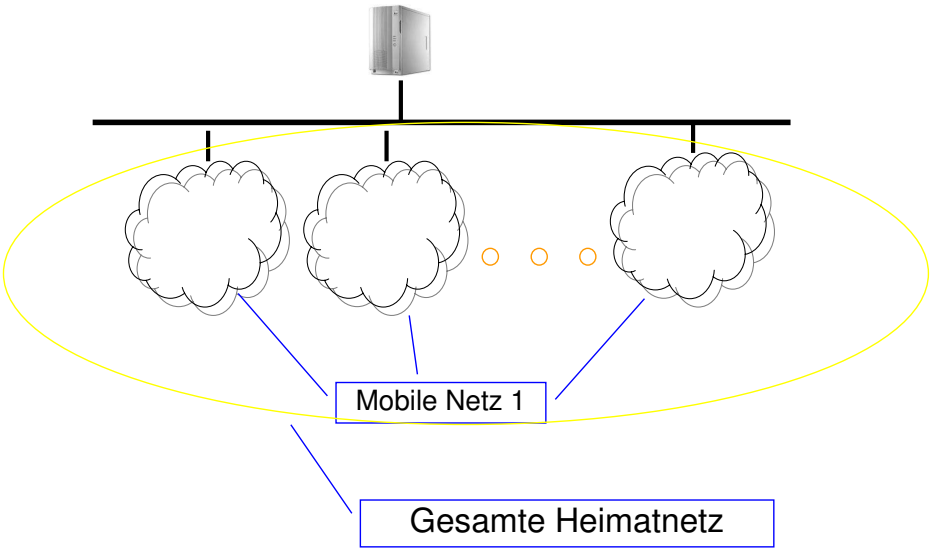


Abbildung 7: Gesamte Heimatnetz

5.3 Virtuelle Heimatnetz

Das Heimatnetz ist kein physikalisches Netz. Die zwei oben besprochenen Moden können mit virtueller Heimat Netz angepasst werden.

Virtuelle Heimatnetze haben deutliche Vorteile gegen physikalische Heimatnetze. Der mobile Router braucht nicht die Prozedur des Heim Rückführ ausführen. Es ist noch stabiler als ein physikalische Netz.

5.4 Mobile Heimatnetz

Die Struktur eines mobilen Heimatnetzes kann wie ein Baum organisiert werden. Die erste Ebene ist das globale Heimat Netz. Darunter sind das erweiterte Heimat Netz und das mobile Heimat Netz. Das erweiterte Heimat Netz hat auch eine eigene Struktur. Dies ist eine ineinander geschachtelte Struktur.

Wenn ein Knoten eines mobilen Netzes ein Paket nach ein Knoten anderen mobilen Netzes schickt, erreicht das Paket zuerst das Heimatagent. Das Paket wird nicht nach dem Internet weitergeleitet, sonst direkt zur das mobile Netz, in dem das Zielknoten sich befindet. Für

speziale Anwendungsszennarien, z.B Die Kommunikationen zwischen Schiffs ist die Funktionsweise des mobilen Heimatnetzes sehr praktisch. (siehe Abb. 9)

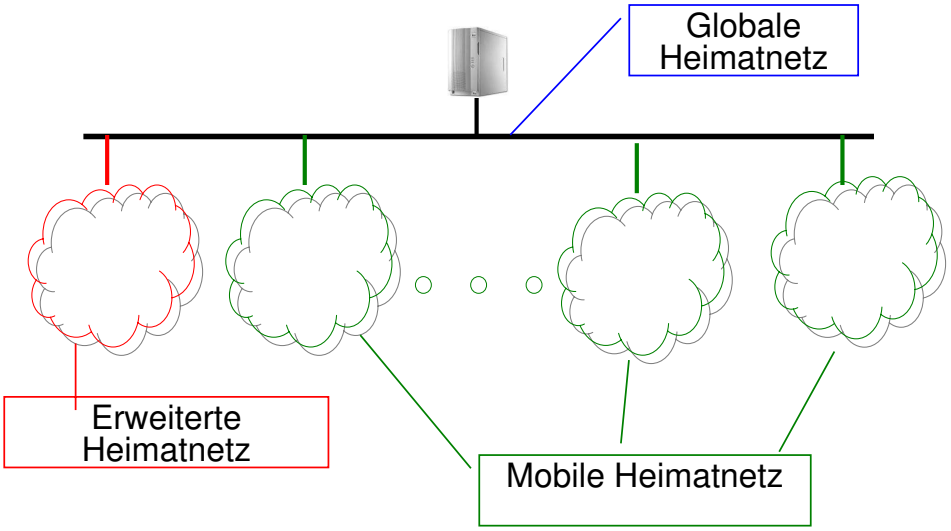


Abbildung 8: Mobile Heimatnetz(1)

6 Fazit

Es würde eine neue Definition vorgestellt: mobile Netz. Mobile Netze sind die nächste Anforderung von Mobilität nach mobilen Knoten. Die Unterstützung des mobilen Netzes basiert auf der Lösung des mobilen Knotens. Mobile IPv6 ist Grundstein der Realisierung des mobilen Netzes. Die Unterstützung des mobilen Netzes wird in zwei Phasen realisiert: grundlegende Phase und erweiterte Phase. Das Protokoll, welch die Realisierung des mobilen Netzes unterstützt, muss mit IPv6 kompatibel sein.

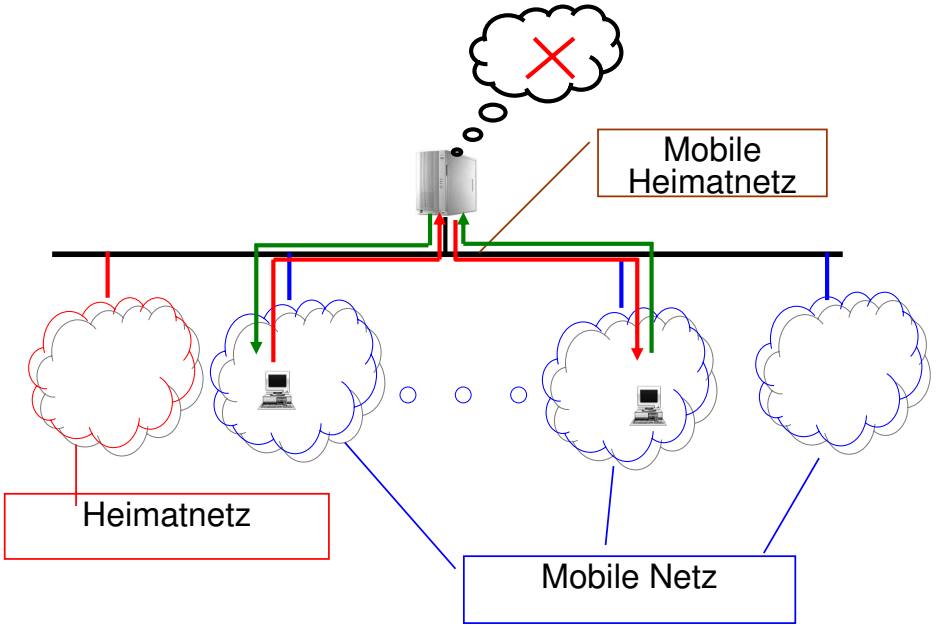


Abbildung 9: Mobile Heimatnetz(2)

Literatur

[DeWT04] Vijay Devarapalli, Ryuji Wakikawa und Alexandru Petrescu andPascal Thubert. Network Mobility (NEMO) Basic Support Protocol. draft-ietf-nemo-basic-support-03(Informational), Juni 2004.

[ErLa04] T. Ernst und H-Y. Lach. Network Mobility Support Terminology. draft-ietf-nemo-terminology-02(Informational), Oktober 2004.

[Erns04] T. Ernst. Network Mobility Support Goals and Requirements. draft-ietf-nemo-requirements-03, Oktober 2004.

[PeJu03] Pekka und Juhani. Mobile Network Prefix Delegation extension for Mobile IPv6. draft-paakkonen-nemo-prefix-delegation-00, März 2003.

[ThWD04] P. Thubert, R. Wakikawa und V. Devarapalli. NEMO Home Network models. draft-ietf-nemo-home-network-models-01, Oktober 2004.

Abbildungsverzeichnis

1 Binding Update und Binding Acknowledgement 61

2 Mobile Knoten 62

3 Mobile Netz 63

4 Binding Update, Option des Präfix und Option des Präfixanforderung 66

5 Binding Acknowledgement und Prefix Confirmation Option 67

6 Erweiterte Heimatnetz 69

7 Gesamte Heimatnetz 70

8 Mobile Heimatnetz(1) 71

9 Mobile Heimatnetz(2) 72

Predictive Configuration of a New Address in Mobile Ipv6

Liying Ren

Kurzfassung

Along with the development of the mobile technique, application of Mobile IPv6 has become far and wide. Within the process of movement there is a key technique, which is called handover. It guarantees the connectivity form mobile node to the Internet. But we must notice the unsatisfactory disadvantage in the handover that is the handover latency. The primary reasons of this latency are movement detection, new care-of address configuration and binding update. In order to reduce the handover latency we will make a farther amelioration to handover, which is named fast handover. This seminar composition describes primarily how a new care-of address could be configured for a mobile node in a predictive subnet, when it still resides on the existing link, i.e. fast handover. By applying the fast handover the mobile node can move through different subnets without any stagnancy when it sends or receives packets. In part one there are some technical terms, they will appear in the figures. In the second part we will introduce the two mechanisms of IP address configuration in IPv6. It is the foundation of IP address configuration in IPv6, whatever the client is a mobile node or a stationary host. In part three the normal mobile node handover process will be briefly described. In the fourth part the whole address configuration procedure Mobile IPv6 of fast handover will be expatiated. In the last part there are 2 amendments to fast handover to be introduced.

1 Terminology

- The mobile node's present router before its handover is called previous access router(PAR).
- The mobile node's anticipated router after its handover is the new access router(NAR).
- The mobile node's care-of address valid on precious access router is called previous care-of address(PCoA).
- The mobile node's care-of address valid on new access router is new care-of address(NCoA).
- A message from the mobile node to the previous access router requesting information for a potential handover is router solicitation for proxy advertisement(RtSolPr).
- A message from the previous access router to the mobile node that helps in movement detection. The message also acts as a trigger for network-initiated handover is proxy router advertisement(PrRtAdv).

- A message from the mobile node instructing its previous access router(PAR) to redirect its traffic (towards new access router) is called fast binding update(FBU).
- A message from the previous access router(PAR) in response to fast binding update(FBU) is fast binding acknowledgment (FBack).
- A message from the mobile node to the new access router(NAR) to announce attachment and to confirm use of New Care of Address(NCoA) when the mobile node has not received fast binding acknowledgment(FBack)is fast neighbor acknowledgment(FNA).
- A message from the previous access router (PAR) to the new access router(NAR) to initiate handover is handover initiate(HI).
- A message from the NAR to the PAR as a response to handover initiate(HI) is handover acknowledge(HAck).
- A message, inquire whether neighbor used that address or not, is neighbor advertisement(NA).

2 Address autoconfiguration for a normal client in IPv6

IPv6 provides two kinds of different mechanism to support the address configuration: stateless and stateful address autoconfiguration [ThNa98]. With the stateful autoconfiguration IPv6 realizes the plug and play of a client. It means a client can connect with the network without any manual assistance. The networks devices can automatic perceive the nodes, which want to connect with the link, and carry out the IP address assignment. IPv6 completes the address detection, the network information detection, automatic address modification, support of mobile nodes, neighbour discovery and so on by the use of the Dynamic Host Configuration Protocol (DHCP)[BVLP+03]. A DHCP server owns an IP address pool. The client leases the IP address from a DHCP server and obtains the relevant information. Different from stateful autoconfiguration, stateless autoconfiguration doesn't require a specific server to support the address configuration. The stateless autoconfiguration requests the local link supporting multicast and the interface of network should have the ability to send and receive the packs in multicast form. The client must confirm the own link-local address and make sure the uniqueness of this address with help of its previous and new access router. After that it combines this address with a prefix from the new access router as its new IP address. The two kinds of mechanism have many difference, but they can also be used united. In the following section the particular processes of the both mechanism will be introduced.

2.1 stateful address autoconfiguration

The stateful autoconfiguration in IPv6 actually is the autoconfiguration service, which already existing in the times of IPv4. It is based on the Dynamic Host Configuration Protocol. The service of stateful address autoconfiguration can be completed in 4 phases.

- Discovery phase. As the first time a DHCP client connects into a new link, it can't find any information about the IP address. Then sends it a multicast DHCP discover package in order to search for the DHCP server. There are also some other information about DHCP discovery in the package. Normally, the wait time of DHCP discovery is one second. That is, if in one second the DHCP discovery has not get any response, the second DHCP discovery will be send at once. The client sends the DHCP discover

totally four times, if it still hasn't received any response. If it can't get a response from a DHCP server, the client will display the mistake information and announce the failure of DHCP discovery. This process of DHCP discovery will be repeated every five minutes.

- Provide phase. After receive the DHCP discovery, all of the DHCP server on the link will response it with a DHCP offer. DHCP servers choose an IP address, that didn't give out, and send it with other information within a DHCP offer to the client. In the DHCP offer will include an agreement of address rental term.
- Choice phase. In this phase the DHCP client will choose an IP address. If there are several DHCP servers on the link and all of them send DHCP offer to the client. The client will only accept the IP address in the first received DHCP offer as its IP address. Then it sends a DHCP request in a multicast in order to notice the other servers, which IP address is used by it.
- Conform phase. After receive the DHCP request the DHCP server will send a DHCP acknowledgement to the client, that means the client can use the IP address provided by it. Along with the IP address becomes into use, the stateful autoconfiguration process is achieved.

2.2 Stateless address autoconfiguration

Relative to the stateful address autoconfiguration the stateless address autoconfiguration is simpler and more convenient. This kind of mechanism allows the client to confirm the IP address itself. The client combines the link-local prefix and its interface identifier as its new link-local address, if it is unique. For verifying this, the client sends a multicast named Neighbor Discovery. If it doesn't be responded to, that is to say, the link-lock address is unique. Otherwise, the client will use a random produced interface identifier as its link-local address and perform the duplication address detection on it. The duplication address detection is also a new function afforded by IPv6. The access router in IPv6 should have the ability to test the uniqueness of a IP address on the link. The test procedure will archive with help of a neighbor solicitation message from the access server. When in one second there is no response to this message, it is to say, this IP address is not used by any client on the link at present. And then the client sends a Router Solicitation to all of routers on the link for the configuration information. The routers should send a Router Advertisement and other information about configuration.

3 New care-of address Autoconfiguration for a mobile node in Mobile IPv6

Now we have known how to configure an new IP address for a stationary client in IPv6. But it is not enough for a mobile node, which can move through several different subnets. In order to keep the connectivity between a mobile node and the Internet we must introduce a new concept care-of address. In this section, the configuration process of a new care-of address will be presented. This process is called handover.

In mobile IPv6 every mobile node has a Home Address, which it uses a source address in communication while it resides on its home link. Even if the mobile node moves out of the home link, the home IP address must be kept constant. When the mobile node moves into another network or it finds some other routers on the link, it must use one of the address autoconfiguration mechanisms of IPv6 to get the new care-of address from the router on

other link, or confirm it itself. Only after obtains the new care-of address, the mobile node can connect with the new network. After that it informs its home agent and several correspondent nodes of the new care-of address with a binding update. From now on the home agent could send packages with home address to the new care-of address through a tunnel. In this way the mobile node realizes the movement between different subnets. But there are also a couple of problems in the handover. In next part we will investigate these problems.

4 fast handover

In the handover process the movement detection, new care-of address configuration and binding update will take a lot of time. In order to insure the service quality of mobile IPv6, we must cut down even remove this part of time for ensuring the continuity and fluency of the connectivity. And the ideal model is, we should find out a mechanism, with which the mobile node can get the care-of address before its movement. The fast handover [Cent04] carries out this for us. Fast handover can be also distinguished into two kinds: predictive fast handover and reactive fast handover. The primary difference of them is on which link a valid fast binding acknowledgment is received by the mobile node. When the mobile node sends the fast binding update on its current link and receives the fast binding acknowledgement successfully, then moves away form this subnet. In this case it will be called predictive fast handover. Otherwise it is reactive fast handover. The detailed process of them will be introduced below.

4.1 predictive fast handover

In mobile IPv6 all of the mobile nodes should have the function of candidate access router discovery. It will be achieved with a specific mechanism in the mobile nodes. When this mechanism detects there is another access router on its link or on another link, the mobile node will send a router solicitation for proxy advertisement to its access point for getting the information, which help to configure the new care-of address. The mobile node can send router solicitation for proxy advertisement at any time. For example, the mobile node perceives that the signal from another Router is better as its router. After received the router solicitation for proxy advertisement, the previous access router sends a proxy router advertisement message to the mobile node as a response to router solicitation for proxy. In the proxy router advertisement the information which are used for configuration of a prospective new care-of address will be provided. By this time the mobile node should still connect with its previous network. In such a way the latency of router discovery and address autoconfiguration in normal handover is eliminated.

After getting the new care-of address the mobile node sends fast binding update to its previous access router to let it bind the previous care-of address and new care-of address. Actually, the previous access router should build a tunnel between itself and the new access router, in order that all of the packages send to the previous care-of address can be tunneled to the new care-of address. Before sending fast binding acknowledgment to the mobile node, the previous access router must determine whether the new care-of address is unique on the new link. It sends the purposed new care-of address with in a handover initiate to the new access router. If it is acceptable, the new access router should answer with a handover acknowledgement to say that the purposed new care-of address can be used. If the purposed new care-of address is be used by another node on the new link, the new access router should assign a new care-of address to the mobile node within the handover acknowledgement. The mobile node must use the new care-of address assigned by the new access router. Normally, this series of actions should also be done on the previous link. In this way the latency of binding update could also be eliminated.

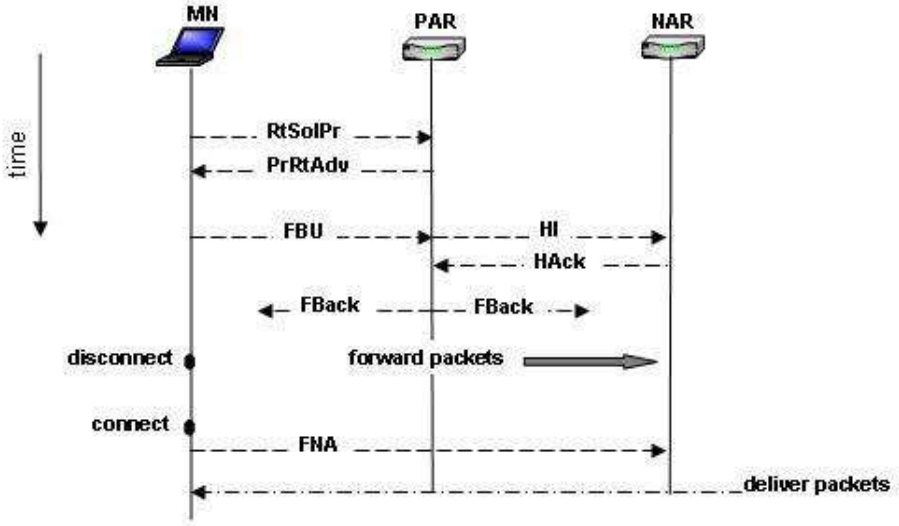


Abbildung 1: 'predictive' fast Handover

Commonly, the new care-of address can be used, as soon as the mobile node connects with the new link, when the mobile node has gotten the fast binding acknowledgement on the previous link. Once it could not been carried out, the purposed new care-of address can also be used, so far as the mobile node sends a fast neighbor acknowledgement on the new link. In the event there is no reduplicate IP address on the link, the new access router will make no feedback. In this wise, the new care-of address configuration latency could be reduced.

As soon as the mobile node connects with the new link, it should send a fast neighbor advertisement to all of neighbors on the link for noticing its new care-of address. If the neighbors want to make contact with the mobile node, they should send the packages to the new care-of address.

This scenario in which a mobile node sends fast binding update and receives fast binding acknowledgement on previous access router's link is illustrated in Figure 1 .

4.2 Reactive fast handover

The criterion to distinguish the predictive and reactive fast handover is on which link a valid fast binding acknowledgment is received by the mobile node. Normally, mobile node should send the fast binding update on the previous link, as it still connects with this network. And then it receive the fast binding acknowledgement from the previous access router before its movement. But some times it can not receive any fast binding acknowledgement before it leaves. There are two reasons to conduce that. The first is the mobile node hasn't sent the fast binding update on the previous access router's link before its movement. The other is after sending the fast binding update and before receiving any fast binding acknowledgement, the mobile node has moved away from the existing subnet. In this case, the fast binding update is useless; The mobile node can not receive the fast binding acknowledgement certainly. If so, it must send a fast binding update as soon as it attaches to new access router. Since the fast

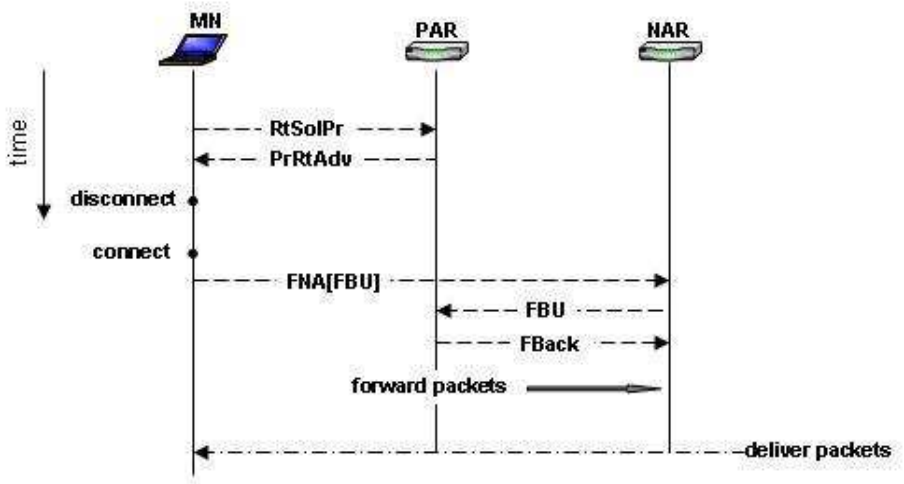


Abbildung 2: 'reactive' fast Handover

binding update has not been archived on the previous link, we must draw attention to the proposed new care-of address. Because it is possible, the proposed new care-of address comes into conflict with an address, which is already used by some other node on the new link. In this case, sending the fast binding update to the new access router in time seems to be very important. In order to confirm the uniqueness of the new care-of address and connect with the new access router the mobile node should send the fast binding update (within a fast neighbor acknowledgment) as early as possible. When the in fast binding update included purposed new care-of address is not unique in the new subnet, the new access router will assign an alternative IP address to the mobile node in a neighbor advertisement acknowledge.

This scenario in which the mobile node sends fast binding update from new access router's link is illustrated in Figure 2.

5 2 amendments in new care-of address configuration for fast handover

In predictive fast handover the new care-of address confirmation can be achieved on the previous access router's link. That is to say, when the mobile node moves into the predictive new link, the new care-of address can be used at once. Compared with it the reactive fast handover has a great shortage. Because of the invalid fast binding update the purposed new care-of address must be confirm on the new link. In this case, the time that mobile node exchanges the link information with new access router will prolong, and during this time the data packets can't be delivery to its destination. There is no doubt that the continuity of the mobile correspondence will been affected. It seems that the verification of the duplication-free address is also very important in mobile IPv6 address configuration. Otherwise, the handover latency will increase quickly. In order to accelerate the duplicate address confirmation we

will introduce 2 approaches. They are different complementarities to fast handover. By using them the fast handover the address configuration is always successful, and the new care-of addresses are always confirmed before the movement of the mobile node.

5.1 duplication-free address pool [AIT04]

Now we will introduce a scheme which improves the new care-of address configuration and confirmation for fast handover in mobile IPv6. The amelioration is implemented in the new access router, but it is very available for the existing stateless and stateful new care of address configurations. It can make address configuration and confirmation fleetly and insure uniqueness of the address.

The scheme has following specific characteristics:

- duplication-free address pool

  Every access router holds and manages a 'duplication-free address pool', if it provides an interface for mobile nodes. The access router has the ability to generate randomly globally addresses and perform duplicated address detection on the address. The access router can reserve the address into its duplication-free address pool, only when the address has passed the duplication address detection. In the duplication-free address pool can reserve 10 IP addresses generally, but the number can also be artificially setup.

- passive-proxy for duplicat

  After stores 10 addresses in the pool, the access router will work as a 'passive' proxy. Since that the access router must not use Proxy Neighbor Discovery and multicast the neighbor advertisement onto its link. When the access router finds that an address which is set in the target address field of a neighbor solicitation message is the same as an address which is stored in its pool, the access router must not reply the neighbor solicitation. Such a neighbor solicitation sometimes may come because of a duplication address detection process from another node. The access router should just silently delete this address specified in the Target Address field from its 'duplication-free address pool'. After the deletion of address, access router should formulate a new address, carry out the duplication address detection on it and store it in the pool. The total number of addresses should be kept at the capacity of the pool(here is 10). By obeys these rules, access router can guarantee the addresses which reserved in the pool are unique.

- new access router operation

  When a mobile node discovers some others access router , it will exchange with its previous access router the router solicitation for proxy advertisement and proxy router advertisement messages at first. And then, it sends the fast binding update from previous access router's link. As receives the fast binding update, the previous access router sends a handover initiate to the new access router in order to ask for a new care-of address (In standard fast handover, which there is not a duplication-free address pool, the new care-of address should be formulated by the mobile node. In order to confirm the uniqueness of new care-of address on the new link, it should be sent within the handover initiate from the previous access router to the new access router). In order to response the handover initiate message, the new access router should at first select a duplication-free address from its pool and delete it from the pool. And then it sends the address within a handover acknowledgement to the previous access router as a reply to the handover initiate. When in the handover initiate the code has been set with 'S', the 'code' field

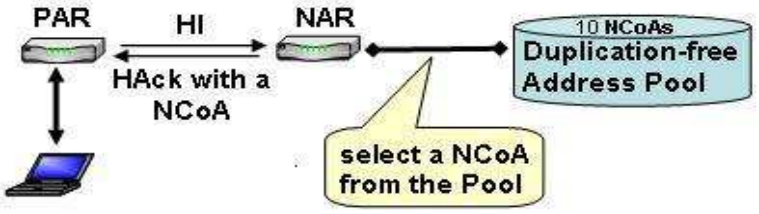


Abbildung 3: duplication-free address pool

in handover acknowledgement could be set to '3', that is to say 'handover has been accepted and the new care-of address is assigned'. Otherwise, the 'Code' field is set to '2', which means 'handover is accepted, new care-of address is in use'. The handover acknowledgment contains always the option about the selected duplication-free new care-of address. In succession the new access router starts 'normal' proxying for the assigned address and creates a proxy neighbor cache entry to defend the address. After doing all of the above things, it checks if there are other things to do. If not, new access router should try to generate a new address or get one from the DHCP Server, and run duplication address detection on it. At last it reserves the new address into its pool in order to keep the total number of address.

This scenario is illustrated in Figure 3 .

5.2 stateful new care-of address configuration [KoKi03]

We have introduced several of schemes to configure the new care-of address configuration. Some of them are stateless address autoconfiguration, in case the new care-of address should be created by the mobile node. And the assignment of duplication address detection will always be archived by the new access router. This time we will introduce one more stateful new care-of address configuration's scheme. As we have presented in part one, the stateful address autoconfiguration can be accomplished by using a DHCP server. In this scheme we will try to append the function of DHCP server to the normal access router , for example, the address modification, address assignment and so on. In this way some information exchanged between mobile node and previous access router order previous access router and new access router are no more necessary. It can farther reduce the handover latency.

In stateful new care-of address configuration in order to perform fast handover each access router should also create and manage the new care-of address pools to ensure that each new care-of address contained in the pool is unique and confirmed by using the duplication address detection process, so that the new care-of address could be immediately available to response the new care-of address request from the mobile node. Since the address pool is created, the fast handover process is free from the duplication address detection.

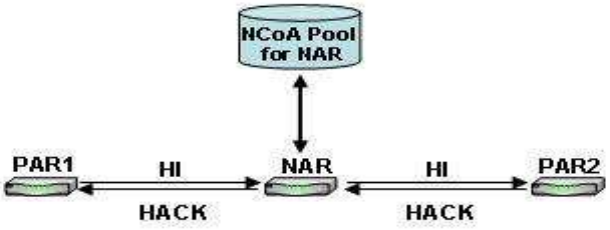


Abbildung 4: network model for reactive new care-of address configuration

Basis of different locus of the new care-of address pool, the new care-of address Pool based configuration schemes can be distinguished into the following two classes: reactive stateful new care-of address configuration and proactive stateful new care-of address configuration. Below we will introduce them respectively.

5.2.1 Reactive stateful new care-of address configuration (NAR-based)

In this reactive stateful new care-of address configuration scheme, the new care-of address pool is located at new access router, and the new access router maintains a list of new care-of address all the time. Each of the new care-of address is confirmed by running duplication address detection on it. When one of the neighboring previous access routers sends a handover initiate message to ask for handover and requests the new care-of address, the new access router will immediately respond with a new care-of address from the pool to the previous access router in a handover Acknowledgement message. This process is shown in figure 4 . It notes that the new access router will respond with the new care-of address each of its neighboring previous access router, which request for it.

Let's see the complete process of this mechanism: When a mobile node initiates the handover with a router solicitation for proxy advertisement, the previous access router will send a handover initiate message to new access router for requesting a new care-of address. The new access router selects a confirmed new care-of address from the pool, and the delivers it to previous access router in a handover Acknowledgment message. The previous access router sends the new care-of address to the mobile node via the proxy router advertisement message. In this scheme the correspondence between previous access router and new access router is earlier as in other schemes, just after receiving the router solicitation for proxy advertisement the previous access router should send the handover initiate to new access router to request the new care-of address. And the mobile node must not formulate the new care-of address itself with the router solicitation for proxy advertisement and proxy router advertisement message any more. It must just directly request it from the access router without any excrescent process. In Figure 5 the process for fast handover using the reactive stateful new care-of address configuration is be shown.

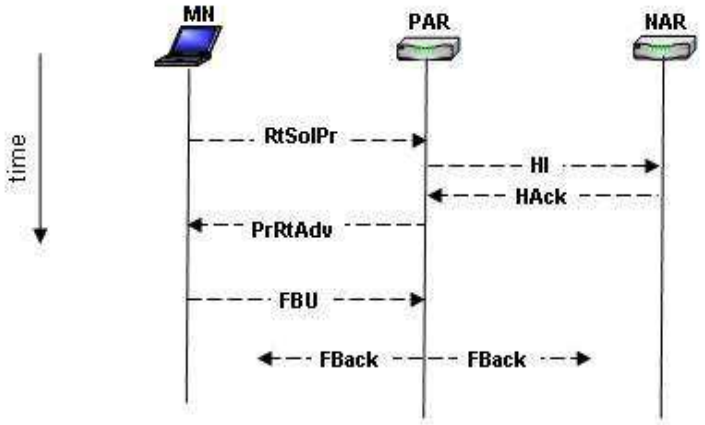


Abbildung 5: reactive new care-of address configuration in mobile IPv6

5.2.2 Proactive stateful new care-of address configuration (PAR-based)

The proactive stateful new care-of address configuration scheme is different from all other schemes, because in this scheme each previous access router should create the 'new care-of address Pools' for every it's respective neighboring Access Routers (i.e. the quantity of pool is equal to the New Access Routers). For example, the addresses in the new care-of address Pool for new access router 1 can only be assigned to the mobile node, which want to build connection with the new access router 1. And the mobile node, which will move into the network of access router 2, will obtain a new care-of address from the new care-of address Pool for new access router 2. The figure 6 shows that the previous access router manages an new care-of address pool for each of the neighboring new access routers.

This scheme is even more convenient than the others, which keep the address pool in new access router. Without exchange so many messages between previous access router and new access router or mobile node and new access router the mobile node can obtain a new care-of address easily. When receiving a router solicitation for proxy advertisement from mobile node, the previous access router selects a confirmed new care-of address from the new care-of address pool, which specially stores addresses for the proposed New access router . Then it responds the mobile node with the new care-of address via proxy router advertisement. In the proactive configuration scheme the previous access router exchange the handover initiate and handover acknowledgement messages only for establishing a tunnel but not for the new care-of address configuration. In this way, the time which is used for handover initiate and handover acknowledgement could be more reduced. Figure 7 illustrates the fast handover by Proactive scheme.

As others new care-of address configuration mechanisms, in the proactive new care-of address configuration the access routers must also generate new care-of address and manage the pools to ensure that the each of new care-of address is unique in the new access router. For obtaining the new care-of address one of the following two processes should be implemented:
1) The new care-of address is configured by the previous access router. When the previous access router want to create a new care-of address for a new access router, the network prefix

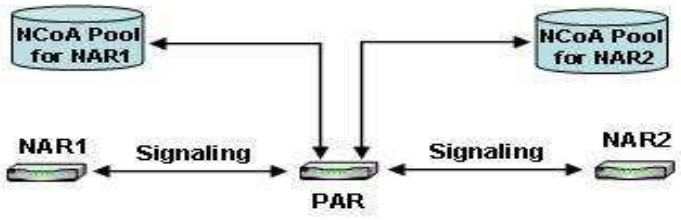


Abbildung 6: network model for proactive new care-of address configuration

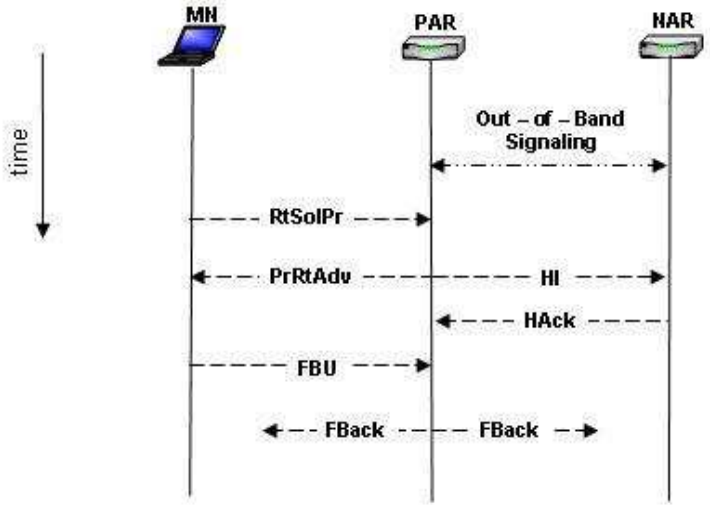


Abbildung 7: proactive new care-of address configuration in mobile IPv6

of this new access router must be used as the network prefix portion of the new care-of address. The remaining interface ID portion may be filled with a random number.
2) Otherwise, the new access router should have the ability to generate the new care-of address and store them as a list. It must forwardly deliver the list to PAR via an out-of-band signaling(figure 8), which is a key element for realizing the proactive stateful new care-of address configuration. It manages the new care-of address pool between previous access router and new access router. The main task of the signaling is to ensure that there exists always a list of the new care-of address for the new care-of address pool and the list could be updated by the corresponding new access router. These tasks will only be done, when the previous access router requests new access router to do that.

Literatur

[AIT04] SAMSUNG AIT. A Supplementary Scheme for New Care-of Address Configuration and Confirmation in FMIPv6, Januar 2004.

[BVLP+03] J. Bound, B. Volz, T. Lemon, C. Perkins und M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6), Juli 2003.

[Cent04] Nokia Research Center. Fast Handovers for Mobile IPv6, Oktober 2004.

[KoKi03] Seok Joo Koh und Dae Young Kim. Address Pool based Stateful NCoA Configuration for FMIPv6, August 2003.

[ThNa98] S. Thomson und T. Narten. IPv6 Stateless Address Autoconfiguration, Dezember 1998.

Abbildungsverzeichnis

1 'predictive' fast Handover 79

2 'reactive' fast Handover 80

3 duplication-free address pool 82

4 network model for reactive new care-of address configuration 83

5 reactive new care-of address configuration in mobile IPv6 84

6 network model for proactive new care-of address configuration 85

7 proactive new care-of address configuration in mobile IPv6 85

HIP : eine Neue Internet-Architektur

Iskander Louati

Kurzfassung

Im aktuellen Internet werden Endsysteme durch deren IP-Adresse identifiziert. Diese Adressen hängen wiederum vom topologischen Standort dieses Endsystems ab. In anderen Worten, die IP-Adresse wird semantisch überladen insofern, dass sie gleichzeitig eine topologische Adresse darstellt und ein Endsystem identifiziert. Das Host Identity Protokoll ermöglicht es, die Identität eines Endsystems und dessen Aufenthaltsort auseinander zu halten. HIP stellt einen neuen Namespace kryptografischer Natur vor. Die IP-Adressen werden weiterhin für Routingszwecke eingesetzt. In dieser Arbeit wird die neue Internet-Architektur mit HIP vorgestellt, und es werden Vor- und Nachteile der Einführung solch eines Protokolls analysiert.Dazu werden die zum neuen Protokoll gebundenen Fachbegriffe erläutert, insbesondere solche die in enger Verbindung zum krytografischen Aspekt des neuen Protokolls stehen.

1 Hintergründe für den Entwurf einer neuen Internet-Architektur

1.1 Existierende Namespaces und deren Mängel

Um die Erreichbarkeit von Endsystemen leichter zu machen hat man Namespaces verwendet. Die Knoten, die bei Datenübertragung durchlaufen werden bekommen jeweils einen „Namen“ . Somit können die zum Datenfluss nötigen Verbindungen zwischen den einzelnen Knoten hergestellt werden. Durch strukturierte Namensgebung entstehen die Namensräume (Eng. Namespaces).
Im aktuellen Internet finden zwei Namespace-Klassen Anwendung: die IP-Adressen und die Domain Names. Bei den erwähnten Namespaces sind drei Probleme ersichtlich:

- Dynamische Readressierung: kann nicht direkt gehandhabt werden, da Transport-Schichten und IP-Adressen sehr stark gekoppelt sind.
- Anonymität: DNS bietet nicht die gewünschte konsistente und vertrauenswürdige Anonymität.
- Sicherheitsmangel : aufgrund fehlender Kryopthografie-Mechanismen, die Authentifizierungszwecken dienen.

1.2 Ziele der neuen Internet-Architektur mit Host-Identity

1.2.1 Ziele

Das neue Namespace-Konzept für Endsysteme sieht vor, dass der Namespace bei Ende-zu-Ende Verbindungen unabhängig von Veränderungen bei der Vermittlungsschicht (3) verwendet werden kann. Dies könnte Readressierungs-Prozesse bei Schicht 3 aufgrund von Mobilität,

Rehoming oder Adreßraumänderung beschleunigen. Der neue Namespace basiert auf Public Key Kryptographie und bietet somit Authentifizierungsdienste an. Um den Aufwand bei der Einführung des vorgeschlagenen Namespace gering zu halten, werden folgende Ziele angestrebt:

- Der Namespace sollte die Vermittlungsschicht von den oberen schichten entkoppeln. In anderen Worten, die Namensgebung sollte alle Vorkommnisse der IP-Adressen innerhalb der Anwendungen ersetzen. Dies könnte allerdings Änderungen bei den aktuellen APIs erfordern. Langfristig könnte dies neue APIs erfordern.
- Die Einführung des neuen Namespace sollte keinen hohen infrastrukturellen Aufwand darstellen.
- Der neue Namespace sollte die aktuellen Protokolle und APIs berücksichtigen.
- Die Names sollten eine festgelegte Länge besitzen, damit deren Einführung in den Packetköpfen von Datagrammen und in den existierenden Schnittstellen reibungslos geschieht.
- Die Namensgebung soll global möglichst eindeutig erfolgen. Um die Wahrscheinlichkeit von Kollisionen zu vermindern und eine Kompatibilität mit IPv4 und IPv6 zu gewährleisten, sollte man für die Größe des „Name“ mit 128 Bit rechnen.
- Der Namespace sollte Authentifizierungsdienste gewährleisten.
- Die Names sollten einerseits für längere Zeit einsetzbar sein, denn andernfalls würde man eine zentrale Namespace-Infrastruktur benötigen um die Names uptodate zu bewahren. Andererseits sollten sie jeder Zeit austauschbar sein.

Ein Namespace der diese Anforderungen erfüllt ist der Host-Identity Namespace. Das benutzen von Host-Identities bedarf eines eigenen Protokolls: das Host-Identity Protokoll.

1.2.2 Architektur

Um die Host-Identity Architektur besser zu verstehen und zu charakterisieren, vergleichen wir sie mit der aktuellen TCP/IP Architektur.

IP-Adressen sind einerseits ein Routings-Vektor, der dafür sorgt, dass Pakete bei ihrer Auslieferung einen bestimmten Weg nehmen und sind somit als Locator zu bezeichnen. Andererseits ist die IP-Adresse gleichzeitig der topologische Standort eines ans Netz angeschlossenen Systems (Schnittstelle) und wird somit als end-point Identifier bezeichnet.

Demzufolge sollte die IP-Adresse als die eindeutige Identifikation eines Endsystems nicht geändert werden. Jedoch ist solch eine Änderung notwendig falls sich der topologische Standort eines Systems ändern sollte. Offensichtlich ist IP nicht dazu fähig, Stabilität und dynamisches Readressieren zu garantieren.

Bei der HIP-Architektur werden Locator und Identifier voneinander getrennt, so dass die IP-Adresse weiterhin die Rolle eines Locators spielt und der Host-Identifier nun die Rolle des End-Eystems Identifier übernimmt.
Typischerweise ist der Aufenthaltsort eines Endsystems für die Anwendungen nicht von Interesse, sondern dessen Identität. Das Entkoppeln der Anwendung von der Routing-information wird durch die Host Identity Schicht realisiert. Zwischen HI-Schicht und Vermittlungs-Schicht wird die Host identity auf die zugehörige IP-Adresse abgebildet. Das neue Schichtenmodell wird in Abbildung 1 skizziert.

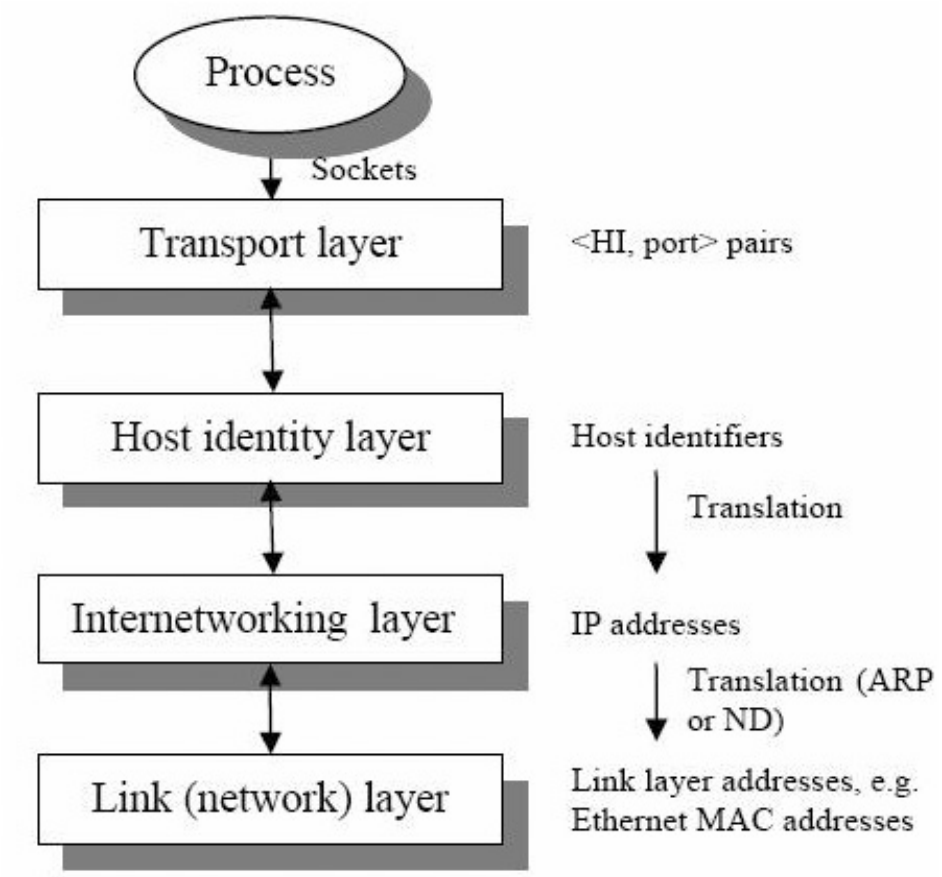


Abbildung 1: Das neue Schichtenmodell

2 Begriffe

2.1 Authentifizierung

2.1.1 Authentifizierung im globalen Netzwerksicherheits-Kontext

1. Authentizität: Hierunter versteht man den sicheren Nachweis der Identität der beteiligten Kommunikationspartner bzw. Herkunft der übertragenen Daten. Die Authentizität ist eine wichtige Voraussetzung für das Erreichen anderer Sicherheitsziele, insbesondere der Integrität, Konfidentialität und Verbindlichkeit.

2. Integrität: Es soll ausgeschlossen sein, dass Daten während einer Übertragung verfälscht werden können, weder durch zufällige übertragungsfehler noch durch mutwillige Angriffe. Die beiden Sicherheitsziele Authentizität und Integrität sind eng miteinander verknäpft, denn die Integrität der übertragenen Daten ist nur dann von Nutzen, wenn auch der Absender authentisch ist.

3. Konfidentialität: oder Vertraulichkeit. Unter Vertraulichkeit versteht man das Verbergen sensitiver Informationen vor Außenstehenden. Bei den verborgenen Informationen

kann es sich sowohl um die übertragenen Daten als auch um die Identität der Kommunikationspartner handeln (Anonymität). Konfidentialität wird dadurch erreicht, dass man die zu verbergenden Informationen verschlüsselt.

4. Verfügbarkeit: Hierunter versteht man die ständige Erreichbarkeit und korrekte Funktion der in einem Netz angebotenen Dienste sowie der an das Netz angeschlossenen Endsysteme.

2.1.2 Authentifizierung mit Public Key-Kryptographie:

Die Public Key-Kryptographie ist ein asymmetrisches Kryptographie-Verfahren basierend auf:

- Public Key: öffentlicher Schlüssel. Dieser Schlüssel wird von seinem „Eigentümer“ mitgeteilt, damit die Sender die an ihn gerichteten Nachrichten verschlüsseln können und damit alle Empfänger dessen Signatur verifizieren können.
- Private Key: privater Schlüssel .Dieser Schlüssel wird geheim gehalten und wird sowohl zum Signieren von den eigenen Nachrichten als auch zum Entschlüsseln der erhaltenen Nachrichten eingesetzt. Siehe dazu Abbildung 2

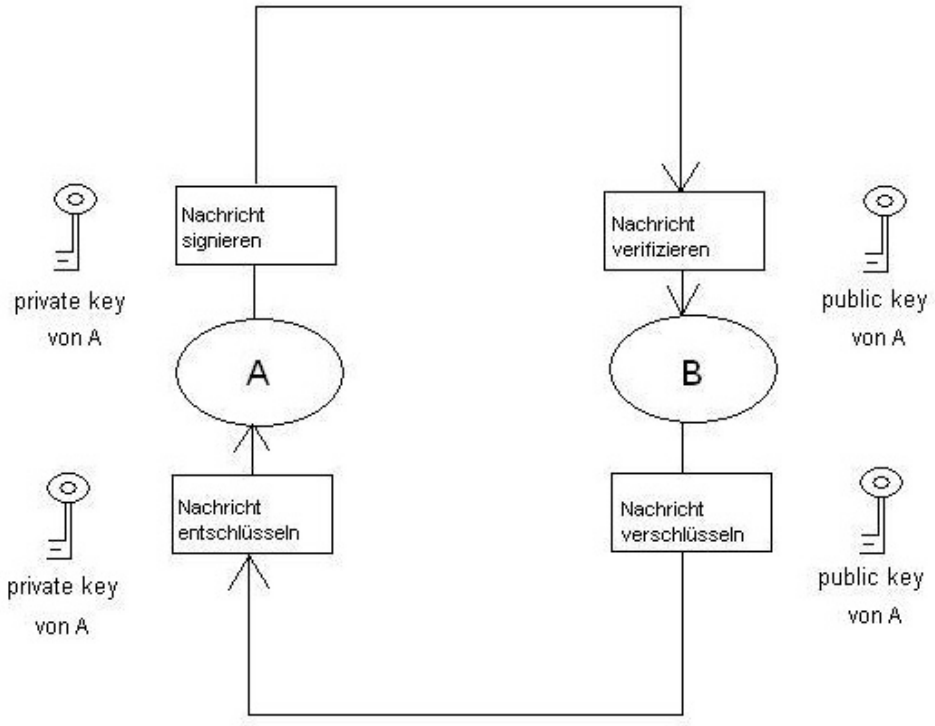


Abbildung 2: Public Key

- Die Algorithmen: eine Reihe von Verschlüsselungsalgorithmen unterstützen die Authentifizierungsmechanismen.Diese Verfahren lassen sich in zwei Kategorien einteilen:
  - Aymmetrische Verfahren: Jeder Host hast seinen eigenen Public bzw. Private Key. Am häufigsten finden RSA (Rivest-Shamir-Adleman)und DSA (Digital Signature

Algorithm) Anwendung.Bei HIP wird die DSA um die eigenen Pakete zu signieren. Eine detaillierte Beschreibung des DSA Verfahrens findet man im RFC2536.

– Sysmmetrische Verfahren: Jeder Host besitzt einen eigenen Public Key.Es wird jedoch aus den jeweiligen Public Keys für die gesicherte Kommunikation zwischen zwei Hosts ein gemeinsamer geiheimer Key errechnet. Diffie Hellman gilt in diesem Sinne als Grundlage für das Host Identity Protokoll

Zum Signieren einer Nachricht nutzt DSA den Secure Hash Algorithm SHA. Ein Hash Algorithmus basiert auf eine Einweg-Funktion, deren Umkehrfunktion schwer zu berechnen ist. Diese Funktion ermittelt aus der zu sendenden Nachricht und dem privaten Schlüssel des Senders einen so genannten Hashwert. Dieser Wert wird der Nachricht angehängt. Somit können beim Empfänger die Integrität der Nachricht und Identität des Senders überprüft werden.Da der Hashwert in der Regel deutlich kleiner ist als der zugehörige Public Key, erfolgt die Authentifizierung beim Empfänger schneller.

2.2 Host Identity

Um die Identität eines Endsystems im Internet möglichst eindeutig festzulegen wurden innerhalb HIP folgende Begriffe eingeführt:

2.2.1 Host Identifier:

Der öffentliche Schlüssel eines asymmetrischen Schlüsselpaars stellt den Host Identifier oder kurz HI dar.

2.2.2 Host Identity Tag:

Oder kurz HIT. Ein HIT ist ein Hash auf den Host Identifier der Länge 128 Bit. Es ist im Grunde genommen einfach eine weitere Darstellung des HI. Ein HIT weist drei Eigenschaften auf:

- Es hat die gleiche Länge wie eine IPv6-Adresse und darf somit innerhalb von APIs und Protokollen Adressfelder stehen.
- Es ist ein Zertifikat für sich. Bei gegebenem HIT ist es rechnerisch sehr aufwendig den dazu passenden HI herauszufinden.
- Ein HIT ist authentisch. Die Wahrscheinlichkeit einer Hit-Kollision zwischen zwei Hosts ist sehr gering.

Bei der zweiten und dritten Eigenschaft handelt es sich um Eigenschaften einer kryptografischen Einweg-Funktion.
Zwei Vorteile sind aus der Anwendung von HITs als Darstellung der Identität ersichtlich. Einerseits ist es für die Protokollimplementierung von großem Vorteil, dass ein HIT eine einheitliche Größe besitzt. Andererseits ist es ein konsistentes Format für die Protokolle unabhängig davon, welche kryptografische Algorithmen zugrunde liegen. Ein HIT entsteht aus der Konkatenation von „01“ mit den 126 niederwertigen Bits aus dem SHA-1 Hash über den HI.

2.2.3 Local Scope Identifier:

Local Scope Identifier, kurz LSI, ist eine 32 bit- lokale Darstellung einer Host Identity. Grund für die Einführung von LSI ist, das Ermöglichen der Einführung von HIP in existierenden IPv4-basierten Protokollen und APIs. LSI kann also in allen Systemprozessen Anwendung finden, wo üblicherweise IP-Adressen verwendet werden. Beispiele hierfür sind IPv4 API Aufrufe und FTP PORT Kommandos.
LSIs dürfen nur vom betroffenen Teilnetz zugewiesen werden. Dadurch wird auf API-Ebene leicht zwischen IPv4 und LSI unterschieden. In der Regel haben HIT und LSI die gleichen 24 niederwertigen Bits. Ansonsten erfüllt LSI nicht die Eigenschaften eines HIT: es ist weder ein Zertifikat für sich noch ist es garantiert authentisch. Folglich sind sie mit Vorsicht zu verwenden.

3 Argumente für und gegen HIP

Sicherlich bringt HIP einige Vorteile mit sich, speziell aufgrund einer neuen Internet-Architektur die auf kryptografische Sicherheit basiert ist; der Einführung und Weiterentwicklung von HIP stehen allerdings einige Hindernisse im Wege.

3.1 Vorteile von HIP:

3.1.1 Entkoppeln von Host- und Routing-Information:

Durch Einführung der Host-Identity - Schicht wird die enge Kopplung zwischen Vermittlungs- und Transport-Schicht aufgelöst. Folglich kann jede der beiden Schichten unabhängig von der anderen weiterentwickelt werden. Die anwendungsorientierten Schichten abstrahieren von dem topologischen Aufenthaltsort des Zielhosts, indem in den APIs und Datagrammen die IP-Adressen durch HIs ersetzt werden. Lediglich das Auflösen der Host-Identity in IP-Adresse erfolgt in der HI-Schicht und stellt somit den einzig wirklichen Aufwand dar.

3.1.2 Unterstützung von Mobilität und Multi-Homing:

HIP entkoppelt Vermittlung Transport- und Vermittlungs-Schicht, und binden die Transportverbindungen mit den Host Identities durch HITs bzw. LSIs. Foglich bietet HIP Mobilität und Multi-homing gegen geringen infrastrukturellen Aufwand. Die Mobilität eines Systems zeichnet sich durch dynamische Änderung der IP-Adresse. Dies wird von HIP garantiert, unabhängig davon, was eine dynamische IP-Adressänderung nach sich zieht, sei es durch DHCP, PPP, erneute IPv6 Adress-Präfix Zuweisung oder durch das Remapping einer NAT-Einrichtung.
Im Falle des Multi-Homing handelt es sich um ein End-System, das durch mehrere IP-Adressen erreicht werden kann. HIP verbindet IP-Adressen, wenn mehrere einer und derselben Host Identity gehören. Falls eine Adresse nicht mehr gültig ist, oder neue Adressen zur Verfügung stehen, können bestehende Transport-Verbindungen leicht von einer zu einer anderen Adresse verlinkt werden. HIP unterstützt außerdem Cluster-basierte Dienste falls es möglich ist, die Prozesse einer Host-Identity über mehrere End-Punkte zu verteilen. Auf der Client-Seite müssen keine Änderungen vorgenommen werden

3.1.3 Kryptografische Sicherheit:

Einige der im Unterabschnitt 2.1.1 definierten Sicherheitsmerkmale werden von HIP gewährleistet:

- Authentizität: ein HIT ist ein Hash auf den öffentlichen Schlüssel des Absenders. Somit ist der HIT unmittelbar an die Identität des Absenders gebunden und ermöglicht ihre Verifizierbarkeit beim Empfänger.
- Integrität: zur Bewahrung der Integrität der zu sendenden Nachricht, erfolgt bei HIP die Verschlüsselung mit dem DH-Algorithmus.
- Vertraulichkeit: ist eine direkte Folgerung aus der Authentizität und Integrität der Nachricht.

3.2 Probleme bei der Einführung von HIP:

3.2.1 Kollisionen

Zwischen HITs ist eine Kollision aufgrund der 128 Bit Länge sehr unwahrscheinlich. Beispielsweise beträgt die Kollisionswahrscheinlichkeit in einem Netz mit 9.10^{16} Hosts nur 0.001%.
Die Kollisionsproblematik stellt sich aber zwischen LSIs die mit 32 Bit Länge einen deutlichen kleineren Namensraum anbieten als HITs.
Während eines HIP-Handshakes kann ein Host feststellen, dass der aus dem HIT erzeugten LSI mit einem anderen lokalbenutzten LSI kollidiert. Beide HITs haben also gleiche 24 niederwertige Bits. In diesem Fall muss der Host die Kollision lokal so behandeln, dass die Aufrufe der betroffenen Anwendung nicht an Mehrdeutigkeit von LSIs scheitern. Eine mögliche Lösung wäre, beim betroffenen Host eine NAT-Funktion anzulegen, die den LSI-Wert vom Partner-Host lokal konvertiert um ihn später nach Beendigung des Anwendungsaufrufs dann zurückzukonvertieren, bevor die Daten wieder auf die Leitung gelegt werden. Der Aufwand solch eines Verfahrens ist sicherlich nicht zu unterschätzen.

3.2.2 Auflösungen:

Eine intuitive Vorgehensweise wäre DNS so zu erweitern, dass eine Auflösung von DNS-Namen auf IP-Adressen bzw. Host-Identity möglich ist. Allerdings, wenn eine Auflösung von HI auf IP bzw. DNS nötig ist, würde man eine zusätzliche Auflösungs-Infrastruktur benötigen. Das Einsetzen verteilter Hash-Tabellen ist in diesem Zusammenhang denkbar, würde dennoch einen ziemlich hohen Entwicklungsaufwand erfordern.

3.2.3 Entscheidung über die Auflösung:

Das Auflösen von IP-Adressen außerhalb einer Anwendung, könnte dazu führen, dass die Absichten der Anwendung missverstanden werden. Drei Szenarien sind in diesem Zusammenhang möglich:

- Die Anwendung weiß über die Verwendung von HIP bescheid, d.h. die Anwendung ist HIP-bewusst, und deshalb wird entweder eine HIP-basierte API verwendet oder es wird auf eine Auflösungseinrichtung zugegriffen um die HI zu ermitteln.

- Im zweiten Szenario wird angenommen, dass die Anwendung nicht HIP-bewusst ist. Die Auflösungseinrichtung versorgt die Anwendung mit einem HIT anstatt einer IP-Adresse. Falls die Anwendung wirklich auf die IP-Adresse angewiesen ist, könnte das Erteilen des HITs bzw. LSIs die Anwendung zum Abstürzen bringen.
- Hier spielt es keine Rolle, ob die Anwendung HIP-bewusst ist oder nicht. Anhand der von der Anwendung gelieferten IP-Adresse wird eine Ende-zu-Ende Verbindung zum Host hergestellt, dessen Host-Identity mit der angegebenen IP-Adresse übereinstimmt. Falls der Zielhost über eine dynamische IP verfügt, und dieser im Laufe des Kommunikationsprozesses eine neue zugeteilt bekommt, lässt sich nicht feststellen, ob die Anwendung so abläuft wie gewünscht, denn es ist durchaus möglich, dass für die Anwendung lediglich die IP-Adresse relevant ist und nicht die Host-Identity.

Hierdurch wird klar, dass die Implementierung der Anwendungen so erweitert werden muss, dass der HI-Schicht mitgeteilt werden soll, ob eine Auflösung von IP auf HI erfolgen soll. Dies könnte den Aufwand für die Einführung von HIP erheblich steigern.

Abbildungsverzeichnis

1 Das neue Schichtenmodell 91
2 Public Key 92

Das Host-Identity-Protokoll: Das Grundprotokoll

Thorsten Grein

Kurzfassung

Seminar im Wintersemester 2004/2005 „Mobiles Internet“ (2 SWS) Das Seminar befasst sich mit der Unterstützung von Mobilität im Internet.

1 Host Identity Protokoll - Einführung

Das Host Identity Protokoll HIP ist ein IP-Protokoll. Die Nutzdaten könnten zwar in jedem Datagramm gesendet werden, jedoch sind die HIP Datagramme ziemlich umfangreich (mindestens 40 Bytes). Das Encapsulating Security Payload (ESP) 3.1 hat bereits alle Funktionalitäten um eine sichere Verbindung aufrecht zu erhalten, ist aber deutlich kleiner. Die HIP Nutzlast wird komprimiert und als ESP Nutzlast übertragen nachdem eine Verbindung aufgebaut wurde. Aus diesem Grund benötigt man ein HIP-Paket eigentlich nur für zwei verschiedene Vorgänge:

- Aufbau einer Datenverbindung
- Änderung des Status einer bestehenden HIP-Verbindung

Im Folgenden benutze ich immer die Begriffe Sender und Empfänger für die Kommunikationspartner beim Verbindungsaufbau. Der Sender bezeichnet die Station, die eine Verbindung zum Datenaustausch aufbauen will und der Empfänger die Station, die die Nachricht, eine Verbindung aufzubauen, erhält. Im Englischen werden die Begriffe Initiator (der Aufrufende) und Responder (der Antwortende) verwendet. Daraus ergeben sich auch die Bezeichnungen für die Pakete I1, I2 vom Sender (Initiator) und R1, R2 vom Empfänger (Responder).

1.1 Das Host Identity Protokoll (HIP)

Der Grund für den Datenaustausch mittels HIP ist die Erstellung einer sicheren Verbindung zwischen Sender und Empfänger. Als erstes verschickt der Sender ein I1-Paket. Daraufhin wird der Empfänger veranlasst mit R1 zu antworten und startet die eigentliche Verbindung. In diesem Paket R1 ist ein Puzzle enthalten, dass vom Sender der Verbindung gelöst werden muss, bevor er den Verbindungsaufbau fortsetzen kann. Ohne eine korrekte Lösung wird die Senderantwort I2 verworfen. Für die Lösung des Puzzels verbraucht der Sender einige Rechenzeit, der Empfänger kann durch einen einfachen Vergleich die Richtigkeit prüfen. Dies schützt den Empfänger davor, sich zu lange mit einem Verbindungsaufbau zu beschäftigen und dadurch nicht mehr erreichbar zu sein. Die letzten drei Pakete, R1, I2 und R2, werden mittels beglaubigten Schlüsselaustausches durch das D-H Protokoll übertragen. Der grundsätzliche Vorgang wird in Abbildung 1 veranschaulicht.

Dies soll einen Überblick über den grundsätzlichen Ablauf eines Verbindungsaufbaues geben. Die Details werden auf den nächsten Seiten näher erläutert.

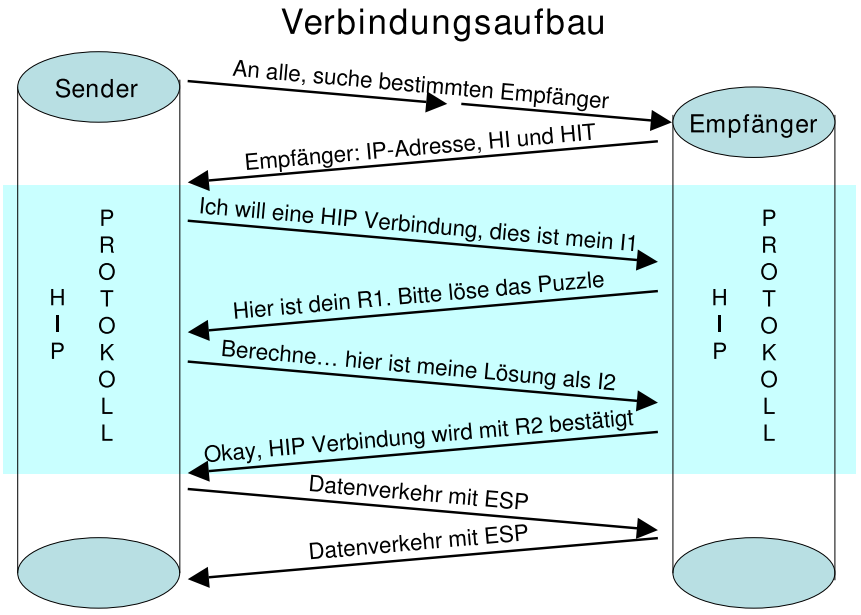


Abbildung 1: Der einfache Verbindungsaufbau

1.2 D-H - Das Verfahren von Martin Hellman, Whitfield Diffie und Ralph Merkle

Das Diffie-Hellman Verfahren (D-H) dient nicht zur Ver- und Entschlüsselung, sondern beschreibt vielmehr die Möglichkeit eines Austausches von Schlüsseln über „unsichere“ Kanäle. Seine Sicherheit basiert auf der Problematik, dass es keine eindeutigen, sicheren mathematischen Verfahren gibt, den diskreten Logarithmus zu berechnen, es aber umgekehrt sehr einfach ist, eine Zahl zu potenzieren. Bei diesem Verfahren wird kein bestimmter Schlüssel ausgetauscht, vielmehr ermöglicht es den Kommunikationsteilnehmern, durch den eigenen und den empfangenen Schlüssel einen gemeinsamen Schlüssel zu generieren. Ein Beispiel für den Ablauf soll die Abbildung 2 deutlich machen.
Das Diffie-Hellman Verfahren ist auf dem heutigen Stand der Technik sehr sicher und wird deshalb weltweit eingesetzt.
RFC 2631 : D-H - Diffie-Hellman
Wikipedia.org

1.3 Anwendung des D-H - Protokolls

Bei der Übertragung von R1, I2 und R2 wird standardmäßig das Diffie-Hellman-Protokoll verwendet. Der Empfänger sendet seinen öffentlichen D-H Schlüssel mit seinem öffentlichen Authentifizierungsschlüssel, also seiner Kennung, als Host (HI, Host Identitiy) mit der R1-Nachricht an den Sender. Die Signatur in R1 ermöglicht es dem Sender zu prüfen, ob R1 wirklich von dem Empfänger erzeugt wurde. Jedoch wurde das Paket vorher berechnet, es besteht die Gefahr von Wiederholungsangriffen, weil es nicht im Ganzen geschützt ist.

Wenn der Sender ein R1-Paket erhält, berechnet er den D-H Schlüssel für diese Verbindung. Mit diesem Schlüssel generiert er einen kodierten Schlüssel von seiner Host Kennung

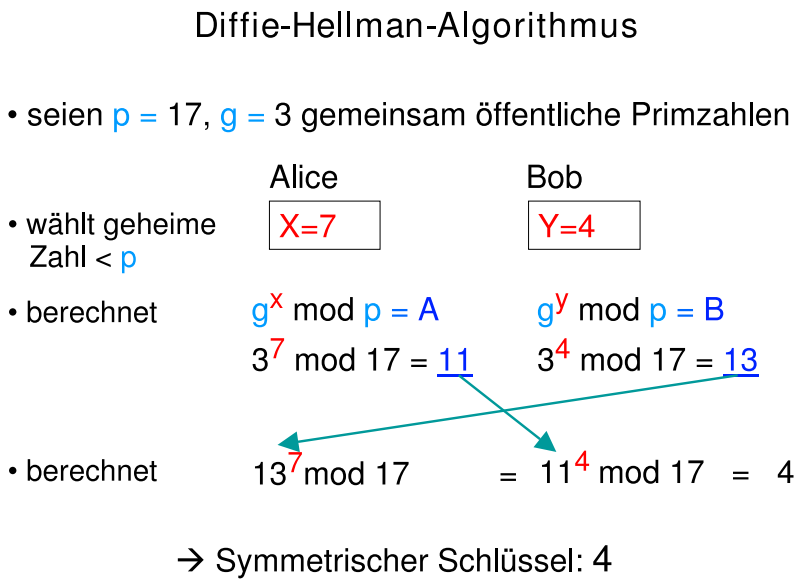


Abbildung 2: Der D-H Algorithmus

mittels HIP Security Association (SA) 3.2. Der Sender übermittelt mit dem I2-Paket seinen D-H Schlüssel und den kodierten öffentlichen Schlüssel. Mit seiner Signatur überdeckt er das ganze I2-Paket.

Der Empfänger gewinnt aus der I2-Nachricht den öffentlichen D-H Schlüssel des Senders und berechnet den D-H Schlüssel für diese Verbindung. Daraus generiert er einen passenden HIP SA, mit dem er den öffentlichen Authentifizierungsschlüssel dekodieren kann. Der Empfänger kann die Signatur benutzen, um den Authentifizierungsschlüssel zu prüfen.

Die abschließende Nachricht R2 wird gebraucht, um den Sender vor Wiederholungsangriffen zu schützen.

1.4 Der Puzzle Mechanismus

Der Hauptzweck den der HIP-Puzzle-Mechanismus erfüllt, ist, eine Denial of Service- Attacke abzuwehren. Dies bedeutet, den Empfänger davor zu schützen, dass er durch eine Vielzahl von Verbindungsversuchen nicht mehr innerhalb einer festgelegten Zeit die Anfragen bearbeiten kann und dadurch anscheinend nicht erreichbar ist. Der Empfänger ist überlastet und kann nicht mehr gesteuert werden. Um diesen Effekt zu vermeiden, wurde im HIP Protokoll der Puzzle Mechanismus implementiert. Er erlaubt dem Empfänger, seinen bisherigen Zustand zu behalten, bis er ein I2-Paket erhält. Die Lösung des Puzzles ist vom Empfänger mit verhältnismäßig wenig Rechenaufwand auf Gültigkeit prüfbar. Der Empfänger kann sicher sein, dass die Anfrage „aufrichtig“ ist, weil der Sender einige CPU-Zeit mit der Lösung des Puzzles verbraucht hat.

Der Puzzle Mechanismus wurde so entworfen, dass genug Raum für verschiedene Implementierungsvarianten bleibt. Er erlaubt es den Empfänger so zu implementieren, dass der Zustand erst mit einer gültig empfangenen I2-Nachricht gewechselt wird. In solch einem Fall wird eine gültig formatierte I2-Nachricht erst abgelehnt, wenn der Empfänger die Gültigkeit mittels Hashfunktion überprüft hat. Andererseits lässt das Design auch eine Implementie-

rung zu, die es erlaubt, den Status zu behalten und I2 diesem Status anzupassen. Dies spart Rechenleistung, die man sonst zur Berechnung der Hashfunktion verbrauchen würde. Der Nachteil der zweiten Methode liegt darin, dass der Zustand geändert werden muss. Schließlich erlaubt es die Implementierung, jede mögliche Kombination zu verwenden mit der sich entweder Rechenleistung oder Speicheraufwand sparen lässt.

Eine Möglichkeit für den Empfänger, zustandslos zu bleiben und die meisten falschen I2s nicht zu beachten ist, grundsätzlich die Auswahl der Puzzles durch eine Funktion über die Identität des Senders einzuschränken. Die Idee ist, dass der Empfänger eine (möglicherweise variable) Anzahl von vorausberechneten R1-Paketen hat. Aus diesen Paketen wird abhängig von der empfangenen I1-Nachricht ein Paket ausgewählt. Wenn der Empfänger jetzt ein Puzzle mit dem I2-Paket erhält, überprüft er nur das vorhandene Puzzle aus der gesendeten R1-Nachricht mit dem empfangenen auf Gültigkeit. Für einen Angreifer ist es kompliziert, zuerst ein I1 / R1 auszutauschen und dann eine große Zahl von falschen I2-Nachrichten zu erzeugen, die unterschiedliche IP-Adressen benutzen oder unterschiedliche HITs. Diese Methode schützt nicht vor Angriffen mit fester IP bzw. fester HIT.

Der Empfänger kann die Schwierigkeitsstufe für den Sender beliebig einstellen, je nachdem wie vertrauensvoll dieser bisher war. Der Empfänger sollte Heuristiken verwenden, um einen DoS- Angriff festzustellen und die Schwierigkeitsstufe entsprechend anzupassen.

Der Empfänger beginnt mit dem Austausch des Puzzle sobald er eine I1-Nachricht bekommt. Er liefert eine Zufallszahl I und verlangt vom Sender, eine Zahl J zu suchen. Damit der Sender eine korrekte Zahl J auswählt, muss er J mit I und seinem HIT verknüpfen. Danach wird ein SHA-1 Hash durchgeführt. Die niedrigsten K-Bits des Ergebnisses müssen Null sein. Die Abbildung 3 in Pseudo-Code soll das berechnen des gültigen Js verdeutlichen.

HIP-Puzzle Mechanismus

```
Pseudo - Code:

FOR ( J=1; J < 2^(k+2); J++ )
{
     SHA-1 ( I | HIT-R | HIT-I | J );
     IF ( ersten K-Stellen des Hashs == Null ) THEN Abbruch;
}
```

Abbildung 3: Berechnung der Hash-Funktion

Um eine korrekte Zahl J zu erzeugen, muss der Sender einige Js erstellen bis er die Zahl findet, die nach ausführen der Hashfunktion Null liefert. Wenn beim Sender

$$2^{(k+2)}$$

Versuche erfolglos waren, gibt er auf und startet den Verbindungsaufbau noch mal von vorne. Der Empfänger muss jetzt eine neue Aufgabe senden, und die vom Sender empfangene

Lösung auf Gültigkeit prüfen.

Um im Voraus berechnete Angriffe abzuwehren, muss der Empfänger die Is so wählen, dass der Sender sie nicht vorher ermitteln kann. Des Weiteren muss sichergestellt sein, dass der Empfänger immer feststellen kann, ob das Ergebnis von ihm gewünscht war oder vom Sender zufällig gewählt wurde.

Der Empfänger hat die Möglichkeit, in einem versteckten Datenfeld in ECHO-REQUEST eine Signatur hinzuzufügen, die vom Sender unverfälscht mit der I2-Nachricht wieder zurückgesendet wird. Der Empfänger kann die Signatur benutzen, um z.B. das gesendete I, geheime oder andere damit in Verbindung stehende Daten verdeckt zu übermitteln. Durch die Verwendung geheimer Daten kann der Empfänger sicher sein, dass die ursprüngliche Nachricht von ihm gesendet wurde. Der Empfänger sollte jedoch die geheimen Daten von Zeit zu Zeit ändern.

Es ist notwendig, dass der Empfänger nach wenigen Minuten neue Puzzles und neue R1-Nachrichten erzeugt. Die alten Puzzles sollten aber noch mindestens 60 Sekunden gespeichert bleiben, bevor sie verworfen werden. Dadurch wird es langsamen Sendern ermöglicht, das Puzzle innerhalb der erforderlichen Zeit zu lösen, gleichzeitig wird die Möglichkeit für einen Angreifer, ein gültiges Puzzle zu benutzen, reduziert.

Mit R1 werden die Werte I und K als Big Endian gesendet 3.5. Genauso werden in I2 die Werte I und J gesendet. Der Hashwert von SHA-1 wird durch Verknüpfung der empfangenen Werte als Big Endian ermittelt.

Aus nachfolgenden Daten wird in der aufgeführten Reihenfolge aneinander gesetzt und der Hashwert erzeugt:

- 64-bit Zufallszahl I, als Big Endian, kommt in R1 und I2 vor.
- 128-bit HIT des Senders als Big Endian, kommt in den Nutzdaten bei R1 und I2 vor.
- 128-bit HIT des Empfängers als Big Endian, kommt in den Nutzdaten bei R1 und I2 vor.
- 64-bit Zufallswert J, als Big Endian, kommt in I2 vor.

Damit das Puzzle korrekt gelöst ist, müssen mindestens die K ersten Bits vom Ergebnis der SHA-1 Hashfunktion Null sein. Durch Variierung von K, ist es möglich den Schwierigkeitsgrad der Hashfunktion beliebig anzupassen.

Vorausberechnet vom Empfänger:

- Setzt die Schwierigkeitsstufe für K fest.
- Erstellt ein signiertes R1-Paket und speichert es zwischen.

Empfänger:

- Wählt ein geeignetes R1-Paket aus dem Zwischenspeicher.
- Erstellt eine Zufallszahl I.
- Sendet I als Puzzle und K in der R1 Nachricht.

- Speichert I und K für einen gewissen Zeitraum.

Sender:

- Versucht so lange das Puzzle zu lösen, bis ein passendes J gefunden ist.
- Ltrunc (SHA-1 (I | HIT-I | HIT-R | J) , K) == 0.
- Sendet I und J in mit dem I2-Paket zurück.

Empfänger:

- Prüft das empfangene I, ob es dem gespeicherten entspricht.
- Berechnet V := Ltrunc (SHA-1 (I | HIT-I | HIT-R | J) , K).
- Verwirft die Nachricht, wenn V != 0.
- Akzeptiert die Nachricht, wenn V == 0.

1.4.1 Anmerkung: Zeitstempel

Die Entwickler des Protokolls haben sich überlegt, einen Zeitstempel in R1 zu übertragen, um den Empfänger vor Wiederholungsangriffen zu schützen. Es wurde aber vereinbart keinen Zeitstempel einzufügen. Im Draft wird diese Aussage zwar nicht begründet, aber es ist davon auszugehen, dass das vom Empfänger gespeicherte I ausreicht um sich vor Wiederholungsangriffen zu schützen.

1.4.2 Anmerkung: Hash-Funktion

Die Länge der gehashten Daten ist 48 Byte. Alle Daten der Hashfunktion müssen als Big Endian vorliegen. Die Anordnung der HITs von Sender und Empfänger ist unterschiedlich in den R1- und I2-Paketen. Es ist darauf zu achten, dass die Werte in der richtigen Reihenfolge in die Hashfunktion geladen werden.

1.5 Schutz vor Wiederholungsangriffen

Im HIP Protokoll ist ein Verfahren enthalten, dass vor böswilligen Angriffen schützt. Empfänger sind vor falschen I1-Paketen geschützt, indem sie Ihren Status beibehalten und ein vorher berechnetes R1-Paket senden. Der Sender schützt sich vor falschen R1-Paketen, indem er einen „R1-Generationszähler“ konstant erhöht, der sich in den R1-Paketen befindet. Empfänger sind vor falschen I2-Paketen durch den Puzzle Mechanismus geschützt. Außerdem kann der Empfänger die verdeckten Daten im R2-Paket mit senden, die in I2 enthalten sein müssen. Hosts sind vor Wiederholungsangriffen von R2- und UPDATE-Nachrichten durch das HMAC-Verfahren 3.3 geschützt.

Der R1-Generationszähler ist ein sich stetig erhöhender Zähler, der mit einer beliebigen 64-bit Zahl initiiert wird. Der Bereich sollte systemweit sein, aber dennoch eindeutig für jeden Host, falls mehr als ein Host in diesem Bereich liegt. Der Wert des Zählers sollte während eines Systemneustarts und anderen Systemaufrufen erhalten bleiben. Der Zähler markiert auch die gegenwärtige Generation des Puzzles. Das Protokoll muss so implementiert

sein, dass Puzzles von der aktuellen Generation akzeptiert werden müssen und Puzzles von vorherigen Generationen akzeptiert werden können. Der Zähler muss stetig erhöht werden, spätestens wenn eine alte R1-Nachricht ungültig wird. Der Zähler sollte niemals verringert werden, es sei denn, der Host erwartet vorher generierte R1-Nachrichten. Zudem darf der R1-Generationszähler niemals überzählen, d.h. nicht automatisch wieder bei Null (oder Minus Maximum) anfangen. Wenn der Generationszähler sein Maximum erreicht, muss er vom entsprechenden Host zurücksetzen und neu initiieren.

Es besteht die Möglichkeit, dass ein Host mehrere R1-Nachrichten empfängt, z.B. weil er mehrere I1s gesendet hat oder er eine alte R1-Nachricht empfängt. Wenn ein Host mehrere I1s sendet, sollte der Sender eine gewisse Zeit warten ob noch weitere R1-Nachrichten ankommen und dann die Nachricht mit der größten Generationszahl wählen, um eine Antwort zu generieren. Empfängt ein Sender eine R1-Nachricht oder hat bereits eine I2-Nachricht gesendet (und wartet gerade auf die Antwort R2) und erhält dann nochmals eine R1-Nachricht mit einer größeren R1-Generationszahl, kann er wählen, ob er auf die neue R1-Nachricht antwortet oder die bereits vorher empfangene beibehält.

Nachdem eine Verbindung zu einem anderen Host aufgebaut wurde, sollte der R1-Generations-zähler erneuert und an den aktuellen Host angepasst werden. Es wird empfohlen, den R1-Generationszähler zu erneuern, weil es bei einem Host vorkommen kann, dass der Zähler verringert wurde; z.B. wegen eines Hardwarefehlers.

1.6 TCP und UDP Pseudo - Header Berechnung

Um eine TCP- und UDP- Verbindung über HIT oder LSI herzustellen, müssen die Checksummen mit einer virtuellen IPv6 Adresse berechnet werden. Zusätzlich müssen die HITs anstelle der IPv6-Adressen im IPv6-Pseudo-Header eingesetzt werden.

1.7 Aktualisierung einer HIP Verbindung

Von Zeit zu Zeit müssen bestehende Verbindungen zwischen zwei Hosts erneuert werden, um die Sicherheit der Verbindungen zu gewährleisten oder um die IP-Adresse zu ändern. In diesem Entwurf wird nur auf die UPDATE-Nachricht eingegangen.

HIP stellt eine universelle UPDATE-Nachricht zu Verfügung, das mit einer Vielzahl von Parametern übermittelt wird, um den Status zweier Stationen zu erneuern. Die UPDATE-Nachricht enthält die folgenden Eigenschaften:

- UPDATE-Nachrichten haben eine Sequenznummer, die stetig erhöht wird und ausdrücklich nur von der Gegenstelle bestätigt werden kann.
- Wenn eine Bestätigungs- oder UPDATE-Nachricht verloren geht, kann sie über eine erneute Übertragung zurück gewonnen werden.
- Mehrere UPDATE-Nachrichten können offen sein.
- UPDATE-Nachrichten sind geschützt durch HMAC und HIP-Signaturen, nur eine UPDATE-Signatur alleine würde das System nicht vor DoS Angriffen schützen.

1.8 Fehlerbehandlung

Das Verhalten bei einer Störung hängt in erster Linie davon ab, ob eine aktive Verbindung besteht oder nicht. Besteht eine sichere Verbindung, sollte der Empfänger im Allgemeinen mit einer NOTIFIY-Nachricht reagieren. Wenn keine aktive sichere Verbindung zwischen Sender und Empfänger besteht oder der Empfänger nicht die Identität des Absenders kennt, kann der Empfänger mit einer ICMP Nachricht 3.4 an den Sender reagieren.

2 HIP Übersicht

Einen typischen Datenaustausch zwischen Sender und Empfänger zeigt die Abbildung 4. Es werden insgesamt 4 Pakete (I1, R1, I2, R2) ausgetauscht.

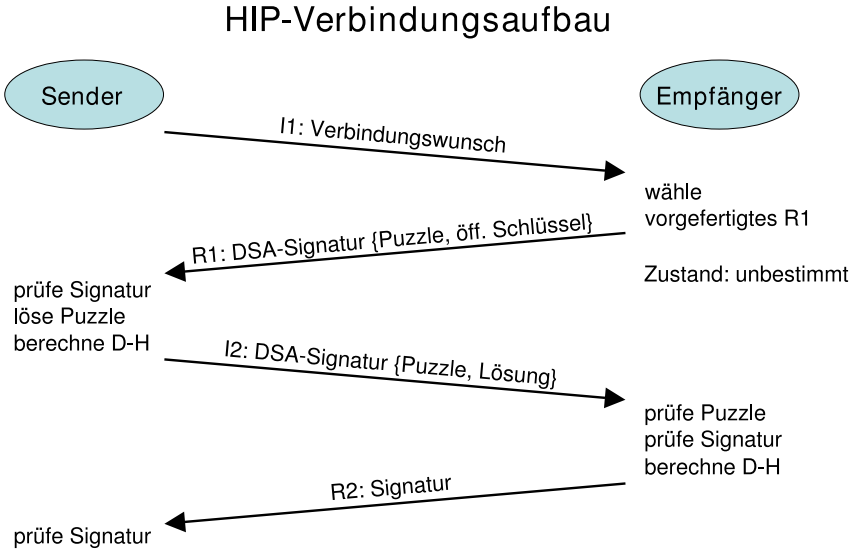


Abbildung 4: 4-Wege-Handshake

Als Sender wird immer die Station bezeichnet, die eine Verbindung zum Datenaustausch aufbauen will. Der Empfänger ist die Station, die zuerst eine Nachricht zum Verbindungsaufbau erhält. Dieser Status gilt immer nur für einen Verbindungsaufbau. Sobald eine Verbindung besteht, ist diese Bezeichnung hinfällig.

2.1 HIP Szenarien

Das HIP Protokoll und der Zustandsautomat wurden entworfen, damit sich eine Verbindung erholen kann, obwohl eine Seite ihren Zustand verloren hat. Die folgenden Verfahren beschreiben die grundsätzlichen Vorgehensweisen.

2.1.1 Es existiert kein Zustand zwischen den Systemen

Das System, das die Daten senden will, ist der Sender (Initiator). Der Ablauf ist der gängige 4-Wege Handshake, der von der SA herausgebracht wurde

2.1.2 Das System mit den zu sendenden Daten hat keinen Status, aber der Empfänger hat noch einen verbliebenen Status

Das System, das die Daten senden will, ist der Sender. Der Sender reagiert genauso, als ob er keine Status hätte, er schickt sein I1-Paket und bekommt ein R1-Paket zurück. Wenn der Empfänger ein gültiges I2-Paket erhält verwirft er den alten SA und benutzt den Neuen.

2.1.3 Das System mit den zu sendenden Daten hat ein SA, aber der Empfänger hat kein SA

Das System sendet Daten mit dem auslaufenden SA. Der Empfänger erkennt diese Situation sobald er das ESP Paket mit einer unbekannten SPI empfängt. Der Empfänger muss dieses Paket laut IPSec Richtlinien verwerfen, aber er kann dem Sender eine ICMP Nachricht schicken.

2.1.4 Ein System stellt fest, dass das es die ESP Sequenznummer zurücksetzen oder einen neuen Schlüssel anfordern muss

Das System muss ein HIP-UPDATE-Paket senden. Das korrespondierende System reagiert mit einem HIP-UPDATE-response-Paket. Durch dieses Paket werden die SA auf beiden Seiten erneuert.

2.2 HIP Verbindung zurückweisen

Ein Host kann einen HIP Verbindungswunsch zurückweisen. Dies kann vorkommen, wenn er so eingestellt ist, dass er selbst nur Verbindungen als Sender aufbauen will. Solch ein Vorgehen macht sicherlich Sinn, denn nur die HI des Senders einer HIP Verbindung ist geschützt. Das Problem daran ist, dass keiner eine Verbindung annimmt und es damit zu keinen Verbindungen kommen kann.

Wenn ein Host keine HIP Verbindung eingehen darf, sollte mit einer ICMP-Nachricht „Ziel nicht erreichbar, administrativ verboten“ reagieren. Dabei wird auf komplexere HIP-Pakete verzichtet, um DoS Angriffen vorzubeugen.

2.3 Neustart und Ablauf der SA-Kennung erzwingen eine neue HIP Verbindung

Einen Verlust des eigenen Status zu simulieren ist eine mögliche Angriffsvariante. Der folgende Ablauf wurde entwickelt, um den Status wiederzuerlangen, ohne einen DoS-Angriff zu ermöglichen.

Durch einen Neustart, oder wenn die SA-Kennung abgelaufen ist, geht der Status verloren. Wenn diese Station nun ein zu sendendes Datenpaket erhält, muss sie eine neue HIP-Verbindung aufbauen. Der Empfänger reagiert mit einem R1-Paket, gibt aber seinen Status nicht auf, bis er ein I2-Paket erhalten hat. Das I2-Paket muss eine gültige Lösung des Puzzels enthalten sowie falls angegeben, auch die versteckten Daten, die der Empfänger in die R1-Nachricht eingefügt hat und natürlich die gültige Signatur. Durch dieses Vorgehen kann es vorkommen, dass die Rollen von Sender und Empfänger gegenüber der vorherigen Verbindung vertauscht wurden.

Wenn ein System ein ESP Paket erhält, den SPI aber nicht kennt, ist es möglich, dass das System den Status verloren hat und nicht der Empfänger. Das System sendet eine ICMP-Nachricht mit den Parametern des Problems. Ob das System überhaupt auf unbekannte SPI reagiert, ist wieder von der Implementierung und der Vertrauenswürdigkeit des Senders abhängig.

2.4 HIP Zustände

2.4.1 Unbestimmt

Die Station hatte bisher keinen Zustand oder es ging ein Fehlerfall voraus. In diesem Zustand werden alle ankommenden Pakete, außer einem Verbindungsaufbauwunsch, verworfen. Wenn die Station ein Datagramm zum senden erhält, veranlasst die Station selbst einen Verbindungsaufbauwunsch (I1) und wechselt in den Zustand „I1-gesendet“. Falls die Station ein gültiges I2-Paket empfängt veranlasst sie ein R2-Paket und wechselt in den Zustand „R2-gesendet“

2.4.2 I1-gesendet

Die Station wartet auf eine gültige R1-Nachricht um mit dem Verbindungsaufbau fort zu fahren. Erhält die Station ein ungültiges R1, oder erreicht einen maximalen Zählerstand, dann führt dies zu einem Fehler. Falls die Station ein gültiges I2-Paket empfängt veranlasst sie ein R2-Paket und wechselt in den Zustand „R2-gesendet“. Bei allen anderen empfangen Paketen verbleibt die Station im aktuellen Zustand.

2.4.3 I2-gesendet

Die Station wartet auf eine gültige R2-Nachricht um die Verbindung zu erstellen. Erhält die Station ein ungültiges R2, oder erreicht einen maximalen Zählerstand, führt dies zu einem Fehler. Falls die Station ein gültiges I2-Paket empfängt veranlasst sie ein R2-Paket und wechselt in den Zustand „R2-gesendet“. Bei allen anderen empfangen Paketen verbleibt die Station im aktuellen Zustand.

2.4.4 R2-gesendet

Die Station veranlasst eine Verbindung und wechselt nach einer vorgegebenen Zeit in den Zustand „Verbindung erstellt“. Falls die Station ein Update-Paket oder ein ESP for SA empfangen hat wird ein neuer SA verwendet. Bei allen anderen empfangen Paketen verbleibt die Station im aktuellen Zustand.

2.4.5 Verbindung erstellt

Die Station hat erfolgreich eine Verbindung mit dem HIP-Protokoll aufgebaut. Falls die Station ein gültiges I2-Paket empfängt veranlasst sie ein R2-Paket und wechselt in den Zustand „R2-gesendet“. Die SA wird ebenfalls erneuert. Bekommt die Station ein Update-Paket oder benötigt einen neuen Schlüssel, wechselt Sie in den Zustand „Neuer Schlüssel“. Bei allen anderen empfangen Paketen verbleibt die Station im aktuellen Zustand.

2.4.6 Neuer Schlüssel

Die Station benötigt einen neuen Schlüssel. Empfängt die Station nun ein gültiges Update-Paket wechselt Sie in den Zustand „Verbindung erstellt“ Falls die Station ein gültiges I2-Paket empfängt veranlasst sie ein R2-Paket und wechselt in den Zustand „R2-gesendet“. Wird ein maximaler Zählerstand erreicht löst dies einen Fehler aus. Bei allen anderen empfangen Paketen verbleibt die Station im aktuellen Zustand.

2.4.7 Fehler

Die Station hatte ein ungültiges Paket empfangen oder der Zähler hatte sein Maximum erreicht. Nach einer vorgegebenen Zeit wechselt die Station in den Zustand „Unbestimmt“.

2.5 HIP Zustandsdiagramme

Das HIP Protokoll hat sehr wenige Zustände. Es existieren nur Sender und Empfänger. Sobald ein SA ausgetauscht wurde, geht die Unterscheidung verloren. Falls ein HIP-Status erneuert werden muss, ist es nur von Bedeutung, welcher Zustand gerade vorliegt, und wer ein zu sendendes Paket hat. Das folgende Zustandsdiagramm versucht, die Abläufe etwas klarer aufzuzeigen.

2.5.1 Einfachter Zustandsautomat

Das Zustandsdiagramm ist rein informativ aufzufassen. Spezielle Implementierungen sind nicht vorgegeben. Dieses Diagramm konzentriert sich auf HIP I1, R1, I2, R2 und das Update Paket. Die wichtigsten Zustandsübergänge sind in diesem Automaten (Abbildung 5) zusammengefasst.

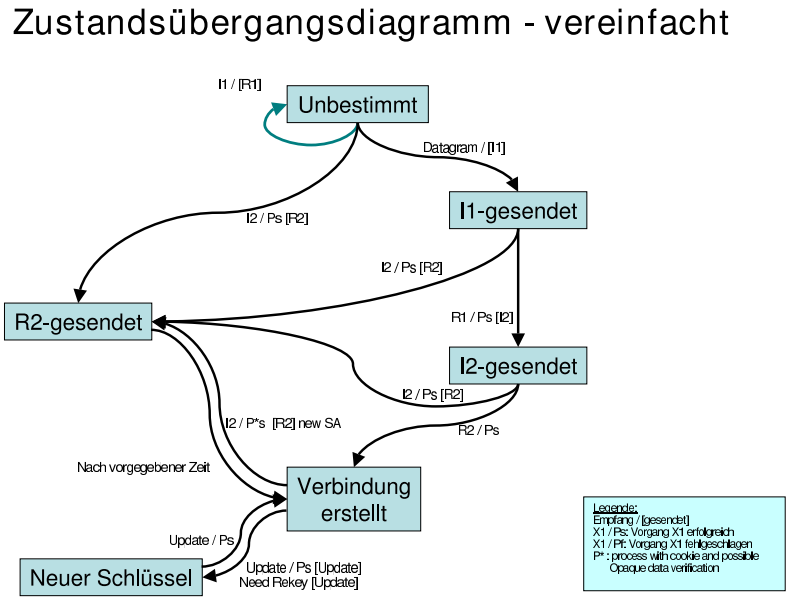


Abbildung 5: einfacher Zustandsautomat

2.5.2 Vollständiger Zustandsautomat

Alle Zustände und alle möglichen Übergänge werden in Abbildung 6 aufgeführt. In der folgenden Tabelle hat mein eine bessere Übersicht, wie eine Station in den verschiedenen Zuständen auf ein empfangenes Paket reagiert. (siehe Abbildung 7)

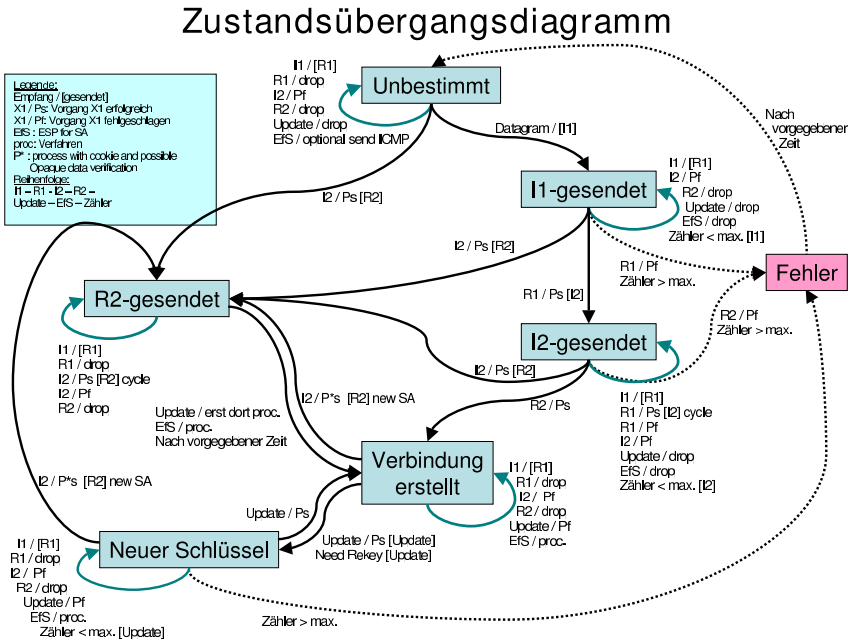


Abbildung 6: Vollständiger Zustandsautomat

Zustandsübergangstabelle

Empfang / Zustand	I1	R1	I2	R2	UPDATE	ESP for SA
Unbestimmt	R1 .	⬇ .	Ps: R2 [R2-sent] Pf: .	⬇ .	⬇ .	unknown SA optional ICMP .
I1-sent	R1 .	Ps: I2 [I2-sent] Pf: [Error]	Ps: R2 [R2-sent] Pf: .	⬇ .	⬇ .	⬇ .
I2-sent	R1 .	Ps: I2↺ . Pf: .	Ps: R2 [R2-sent] Pf: .	Ps:[Established] Pf: [Error]	⬇ .	⬇ .
R2-sent	R1 .	⬇ .	Ps: R2↺ . Pf: .	⬇ .	[Established] proc	proc [Established]
Established	R1 .	⬇ .	P*s: R2⬇ SA, new SA [R2-sent] Pf: .	⬇ .	Ps: UPDATE [Rekey] Pf: .	proc [Established]
Rekeying	R1 .	⬇ .	P*s: R2⬇ SA, new SA [R2-sent] Pf: .	⬇ .	Ps: [Established] Pf: .	proc .

. Bleibe im aktuellen Zustand
[Zustand] gehe zu Zustand
⬇ drop
↺ cycling
Ps Processing successful
Pf Processing failed
proc Processing
P*s Process with cookie and possible opaque verification

Abbildung 7: Tabelle mit Zuständen und empfangenen Paketen

3 Anhang: Weiterführende Erklärungen

3.1 ESP - Verpackte Sicherheitsdaten

Encapsulatied Security Payload (ESP) soll die Authentifizierung, Integrität und Vertraulichkeit von IP-Paketen sicherstellen. Im Unterschied zu AH wird der Kopf des IP-Paketes nicht mit berücksichtigt.

ESP beinhaltet zwei Betriebsmodi: Den Tunnel-Modus und den Transport-Modus. Der Tunnel-Modus wird eingesetzt, wenn zwei Netzwerke über eine unsichere Strecke miteinander verbunden werden sollen. Es wird dann das komplette IP-Datenpaket vor der Übertragung verschlüsselt. Davor wird ein neuer IP-Header gesetzt. Im Transport-Modus werden lediglich die Daten verschlüsselt und der alte IP-Header unverändert belassen. Die Daten sind so zwar geschützt, ein Angreifer kann jedoch zumindest eine bestehende VPN-Verbindung zwischen zwei Stationen feststellen.
Wikipedia.org

3.2 SA - Sicherheitsvereinbarung

Security Associations (SA) sind Sicherheitsvereinbarungen, die zwei mittels IPSec miteinander kommunizierende Instanzen vor der Kommunikation untereinander austauschen. SAs werden für den Authentification Header (AH) und den Encapsulated Security Payload (ESP) jeweils individuell getroffen. Sie gelten für die unidirektionale Kommunikation, also nur für eine Übertragungsrichtung. Da eine Kommunikation bidirektional erfolgt, sind mindestens zwei SAs für die Übertragung erforderlich. Security Associations sind die grundlegende individuelle Basis jeder IPSec-Verbindung. Sie definieren exakt, wie der Host oder das Security Gateway eine Verbindung zur Zielkomponente aufbauen und erhalten muss. Eine SA ist stets einzigartig und wird durch drei wesentliche Komponenten beschrieben: Den Security Parameter Index (SPI), die IP-Zieladresse und den Security Protocol Identifier.
Link im Internet

3.3 HMAC - Verschlüsseltes Hashing

Im RFC 2104 wird ein Verfahren zur Authentisierung beschrieben, welches ohne ein RSA-Verfahren auskommt. Dazu wird die Erzeugung eines Hashwerts von einem Schlüssel abhängig gemacht.

Aufgabe von HMAC ist es, ein erhaltenes Dokument darauf hin zu überprüfen, ob es bei der Übertragung vorsätzlich verfälscht wurde. Ein aktiver Angreifer könnte das Dokument verfälschen und einen neuen Hashwert erzeugen. DSA und ähnliche Verfahren verschlüsseln diesen Hashwert mit einem privaten Schlüssel. HMAC fügt in das Dokument einen geheimen Schlüssel ein und hashed das so entstandene neue Dokument. Dies kann von einem Angreifer nicht nachvollzogen werden. Er könnte höchstens den Hashwert fälschen, aber damit wäre die Authentisierung völlig gescheitert, also kein gefälschtes Dokument als echt anerkannt. Ist der geheime Schlüssel nur zwei Parteien bekannt, so können diese anhand dieses Schlüssels die Unverfälschtheit einer Nachricht nachweisen. Dies gilt auch, wenn sich eine Instanz vergewissern will, dass ein gespeichertes Dokument nicht absichtlich verfälscht wurde.

RFC 2104 nennt folgende Entwurfsziele:

- Vorhandene Hash-Funktionen unverändert einsetzen können, die in Hardware oder Software zur Verfügung stehen und deren Code frei verfügbar ist.
- Die Schnelligkeit der Hash-Funktionen ohne Leistungseinbußen ausnutzen.
- Auf einfache Weise die Schlüssel behandeln können.
- Eine gut verstandene kryptographische Analyse der Stärken des Algorithmus kennen, die auf vernünftigen Annahmen der zugrunde liegenden Hash-Funktionen beruht.
- Die zugrunde liegende Hashfunktion einfach ersetzen zu können, wenn schnellere oder sicherere Hashfunktionen gefunden oder benötigt werden.

Der Gedanke dahinter ist, zu einem Text einen Schlüssel hinzuzufügen, so dass der Fingerabdruck nicht nur vom Text, sondern auch vom Schlüssel abhängt.
RFC 2104 : HMAC - Keyed-Hashing for Message Authentication
Link im Internet

3.4 ICMP - Internet Nachrichten Protokoll

Internet Control Message Protocol (ICMP) ist ein Protokoll der zweiten Schicht (internet layer) des TCP/IP-Modells. Das ICMP hat die Aufgabe, Status- Kontroll- und Fehlermeldungen für IP zu transportieren. ICMP benutzt IP genauso wie TCP und UDP, als ob es ein Protokoll der Transportschicht wäre. In Wirklichkeit ist ICMP ein Bestandteil von IP. Jedes IP-Modul muss ICMP implementiert haben. Da das Internet-Protokoll nicht zuverlässig ist, wurde ICMP entwickelt, um die beteiligten Rechner über mögliche Kommunikationsprobleme zu informieren. Die Zuverlässigkeit von IP wird durch ICMP nicht beeinflusst. Für eine verlässliche Datenübertragung sind die Protokolle der höheren Schicht (z.B. TCP) selbst verantwortlich.
RFC 792 : ICMP - Internet Control Message Protocol

3.5 Exkurs - Vom großen und vom kleinen Ende

Im vorliegenden Internet-Draft ist die Rede von „network byte order“, was nichts anderes bedeutet, als dass die Reihenfolge der Bytes immer festgeschrieben ist. Dieses Format, bei dem das Byte, das am weitesten rechts steht, den kleinsten Wert hat und das Byte, das am weitesten links steht, den größten Wert oder als ein Vorzeichen zu interpretieren ist, wird als „big endian“ Format bezeichnet. Bei dieser Schreibweise liegen die Zahlen wie im Dezimalsystem vor. Der Ursprung der Bezeichnung „big endian“ oder „little endian“ stammt aus dem Roman „Gullivers Reisen“ von Jonathan Swift. Er beruht auf dem Umstand, dass die Bewohner sich nicht einig waren, auf welcher Seite man ein Ei aufschlägt. Es gibt zwei Möglichkeiten, am dicken oder am spitzen Ende.

Abbildungsverzeichnis

1 Der einfache Verbindungsaufbau 100
2 Der D-H Algorithmus 101
3 Berechnung der Hash-Funktion 102
4 4-Wege-Handshake 106
5 einfacher Zustandsautomat 109
6 Vollständiger Zustandsautomat 110
7 Tabelle mit Zuständen und empfangenen Paketen 110

Mobilität und multihoming

Rim Helaoui

Kurzfassung

Mobilität ist seit je her ein Grundbedürfnis des Menschen. Mit Mobilität verband man bisher vor allem die physische Freiheit, sich von Ort zu Ort bewegen zu können. Durch die Techniken der modernen Datenübertragung jedoch, ist in den letzten Jahren eine weitere Ausprägung hinzugekommen, die Mobilität in der Kommunikation. Unter Kommunikation fallen alle Formen, des sich Mitteilens. Also die Mensch-Zu-Mensch-Kommunikation und Informationen oder Daten von Standort A nach Standort B zu bringen.

Verbindet man die Mobilität, also Beweglichkeit, mit der Kommunikation, so ergibt sich ein Bild eines Menschen, der in allen Lebenslagen von jedem Ort zu jeder Zeit mit jeder Person kommunizieren kann. Dieses Bild stellt den Zustand dar, den viele Menschen unserer Gesellschaft erreicht haben: die vollständige Erreichbarkeit.

Dieses Dokument,gibt einen Überblick auf die dafür im Rahmen des Host Identity Protokolls vorgeshlagene Lösung.

1 Motivation

Stellen Sie Sich vor, dass Sie eine lange Reise unternehmen, die ein Jahr dauert. Sie sind während dieser Zeit Weitweg zu Ihrem Haus und sollen eine Lösung dafür finden, wie Sie Ihre Post weiter bekommen können, auch, wenn sich Ihre Adresse sehr oft ändert. Wie könnte eine solche Lösung aussehen? Eine Möglichkeit wäre, dass Sie immer wieder eine Postkarte an alle Ihre Freunde und Bekannten senden, sobald sich Ihre Adresse ändert.Die wäre aber unpraktisch.

Erstens, weil Sie bei jedem Umzug wieder neue Postkarten senden müssten.

Zweitens, könnten Sie jemanden vergessen, und so eventuell eine wichtige Nachricht versäumen.

Drittens, gäbe es keine Garantie, dass Ihre Konkurrenten nicht ähnliche Postkarten senden würden, um Ihre Post zu sich umzuleiten.

Alternativ könnten Sie, aber, einen Antrag bei Ihrer Poststelle abgeben, damit sie Ihnen Ihre Post weiterleiten kann, und zwar an Ihrer aktuelle vorübergehende Adresse. Der Vorteil dabei wäre, dass nur die Poststelle von Ihrer Adressen änderung Bescheid wissen würde. In diesem Fall, bräuchten Sie aber einige Sicherheitsmaßnahmen als Nachweis dafür, dass Sie doch die behauptete Identität wirklich haben.

Klar könnten einige Briefe verloren gehen, wenn Sie genau dann an Ihrer veralteten Adresse ankommen, wenn Sie gerade beim Umzug sind.Das könnten Sie vermeiden, wenn Ihre letzte Poststelle jede bekommene Post zu Ihrer Heimat Poststelle zurückschickt, sobald Sie umgezogen sind. Oder auch, indem Sie Ihre Bekannten und Freunde vorwarnen, dass sie Ihnen bei einer fehlenden Antwort, die Post nochmals zu senden.

Wenn wir, anstelle von den Wörtern ´Post´ ´Internet Protocol data packet´ , und von ´weiterleiten´ ´tunneling´ ,und von ´Poststelle´ ´Mobiles IP router´ dann erhalten wir eine konkrete Einführung ins Multihoming und Mobilitätsproblem.

1.1 Multihoming

Ein Host oder Router ist multihomed, wenn er über mehrere IPv6 Adressen 1 verfügt und unter der er zu erreichen ist. Zwei Fälle stellen sich vor:

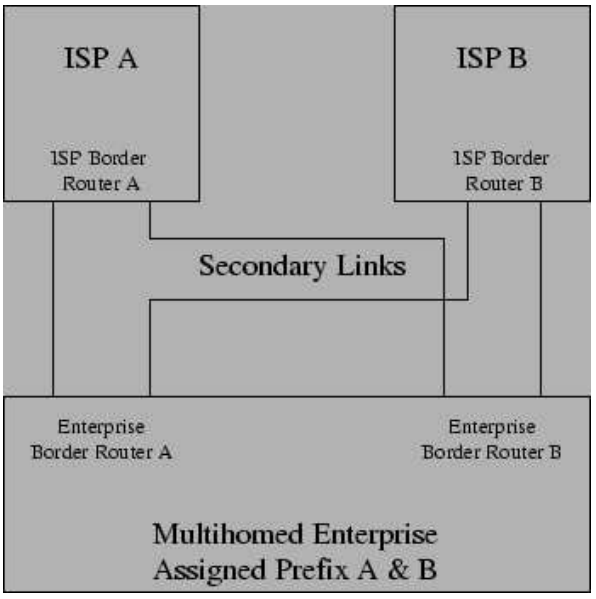


Abbildung 1: Multihoming

- Multi-prefixed : Der host ist verbunden zu einem Link mit mehreren Präfixen. D.h er arbeitet über eine Schnittstelle mit mehreren IP-Adressen . Da es sich um eine Schnittstelle handelt, bedeutet das, dass das System damit aus einem Subnetz auch unter mehr als einer Adresse erreichbar ist.
- Multi-interfaced: Der Host verfügt über mehrere Schnittstellen ,unter denen er zu wählen hat.

Ein Multihomed-Network wäre daher, ein Netz, der mittels mehr als ein Router, oder mit einem multihomed-Router zum Internet verbunden ist.

2 Probleme dabei

Wie schon gemerkt, erweist sich diese Vollständige Erreichbarkeit in der Praxis als nicht unproblematisch. Der Wegewahl eines Packets Basiert auf IP-Zieladresse, Netzwerk-Präfix (zum Beispiel 129.13.42) legt physikalisches Subnetz fest. Wird das Subnetz gewechselt, so muss auch die IP-Adresse passend gewechselt werden (normales IP) oder ein spezieller Routing Eintrag vorgenommen werden. Je nach Lokation wird entsprechende IP-Adresse gewählt. Wie sollen Rechner nun gefunden werden?

DNS-Aktualisierung dauert lange, und zwar wegen seiner hierarchischen Struktur 13, TCP-Verbindungen brechen ab, Sicherheitsprobleme!

Die neue IP-version IPV6, der Nachfolger des aktuellen Internet Protokolls, IPV4 bringt dafür einige Neuerungen und Verbesserungen mit sich, die die Verwendung von mobilen Systemen erleichtert.

2.1 überblick auf IPV6 Erweiterungen(IPV4 vs. IPV6)

Natürlich stellt sich die Frage, ob es überhaupt notwendig ist, das gesamte Internet auf eine neue Protokollversion umzustellen, zumal es einige Jahre dauern wird bis sich IPv6 durchgesetzt hat. Portable Geräte, ob Notebooks, Handys oder digitale Assistenten, mit Internetanschluss und eigener IP-Adresse werden immer beliebter. Dadurch erlangen Schlagworte, z. B. Adressraum, Autokonfiguration, Mobilität und Sicherheit völlig neue Dimensionen.

Die IPV6 ist eine Weiterentwicklung von IPV4. Der Hauptgrund für die Einführung des Internet Protcols Version 6 (IPv6) sind die 4 Milliarden IP-Adressen (Version 4) im Internet, die bald aufgebraucht sind. Obwohl der Adressraum nur zu 60 Prozent belegt ist, wurde durch eine zu großzügige Adressvergabe der Adressraum schnell aufgebraucht. Da weltweit immer mehr Menschen, Maschinen und Geräte an das Internet mit einer eindeutigen Adresse angeschlossen werden wollen, ist der Rest an IP-Adressen sehr bald aufgebraucht. Die nächste Generation von IP, das IP Version 6, erhöht den Adressumfang auf 2 Hoch 128. Damit wäre es möglich jeden Quadradmillimeter der Erde mit rund 600 Billionen Adressen zu belegen. Dafür ist der Aufbau der IP-Adressen nicht mehr wie früher in der Punktnotation der Form A.B.C.D, sondern werden jetzt in hexadezimalen 16 Bit Blöcken, getrennt durch Doppelpunkte, geschrieben. Obwohl die IPV6 einen viel größeren Adressraum und mehr Flexibilität bietet, nimmt die Größe des Paketvorspanns gegenüber IPv4 nur um den Faktor zwei zu. Erreicht wird das vor allem durch den Wegfall vieler reservierter Header-Felder.

Eigentlich verfügt sie über zwei Arten von Headers : Den Basis- und den Erweiterungs-Header. Router schauen sich nur den Basis-Header an, der keine rechenaufwendigen Felder wie die in IPv4 übliche Prüfsumme enthält. Ansonsten ist der Basis-Header 2 mit dem IPv4-Header vergleichbar.

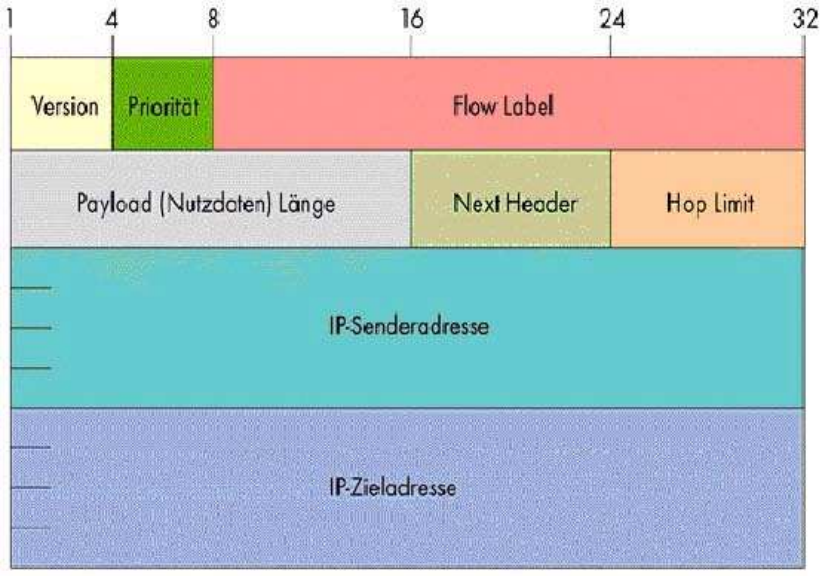


Abbildung 2: IPV6 Header

Bei der IPv6 hat man sich nicht nur um die Adresserweiterung gekümmert, sondern auch gleich eine Generalüberholung des Protokolls vorgenommen. Zählte zur Hauptaufgabe der heutigen IPv4-Router das Prüfen von Checksummen und Fragmentieren von Daten, so ist die Arbeit für IPv6-Router sinnvoll minimiert worden. IPv6 führt keine Prüfsumme mehr im Header mit. Stattdessen wird dem übergeordneten Transport-Protokoll TCP die Aufgabe

überlassen kaputte Pakete zu erkennen und neu anzufordern. Dieser Vorgang wird komplett beim Empfänger bearbeitet. Zu große Datenpakete werden von IPv6-Routern nicht mehr selber fragmentiert. Ist ein Paket zu groß wird dem Absender eine Fehlermeldung geschickt.

Eine weitere neue Funktionalität stellt sich dabei dar: Die Autokonfiguration. Die macht die Nutzung des Internet noch bequemer für den Anwender. Sie spiegelt die Tendenz, umfangreiche Internet- bzw. Intranet-Systemlösungen ohne spezialisierte Kenntnisse, nach dem so genannten 'Plug-and-Play'-Prinzip, anzubieten, wieder. Das spielt eine extrem wichtige Rolle für die Mobilität und multihoming. Es wird ein wichtiger Schritt in die Richtung gemacht, den Internetanschluß als eine SSteckdoseßu sehen, die überall (z. B. für ein Notebook) zur Verfügung steht.

Ein Anwender kann auch über seine Internet-Kennung in mehreren ortsentfernten LANs einen Netzzugriff besitzen. Vereinfacht und kurzfristig kann ein solcher Netzzugriff auch in einem fremden Netz eingerichtet werden, was einer Umleitung bzw. einer Vervielfältigung des Internet-Anschlusses entspricht.

Um über Internet kommunizieren zu können, sind eine IP-Adresse und die Router-Adresse beziehungsweise ein Subnet-Präfix erforderlich. IPv6 ist so ausgelegt, daß der Host oder Router diese Informationen automatisch findet. Dabei geht IPv6 zunächst von PCs aus, die ohne Server oder Router arbeiten. Für die Generierung der IP-Adresse zieht das neue Internet-Protokoll im ersten Schritt die 48 Bit lange Nummer der Ethernet-Hardware - MAC Adresse oder Ethernet-Token Adresse - heran und setzt diese an die letzten 48 Stellen. Anschließend sendet das System eine Message aus, um zu erkunden, ob ein Router oder Server vorhanden ist. Bleibt die Anfrage unbeantwortet, ergänzt IPv6 die Nummer um ein lokales Präfix.

Die letzte große Neuerung von IPv6 betrifft Sicherheitsaspekte. Während momentan Sicherheitsfeatures auf der Applikationsebene umgesetzt werden müssen, sind sie im neuen Internet-Protokoll auf der Kernel-Ebene implementiert. IPv6 hat Authentifizierung und Verschlüsselung bereits eingebaut und stellt sie als „Dienstleistung" den Anwendungen zur Verfügung. Dazu werden beide Leistungsmerkmale als eigene Header-Felder aufgeführt.

Durch zwei Header-Typen erlangt das neue Protokoll eine hohe Sicherheit. In der Internet-Schicht befinden sich zwei Sicherheitsmechanismen: Authentifizierung und Sicherheitseinkapselung (Encapsulating Security Payload, ESP). Der Authentifizierungs-Header stellt eine Art fälschungssichere Unterschrift dar, um zu bezeugen, daß ein IP-Paket wirklich von dem angegebenen Absender stammt und unterwegs nicht verfälscht wurde.

Die Technologie, die eine solche sichere Kommunikation im Internet garantiert ist IP Securtity genannt (übliche abkürzung:IPsec 3).

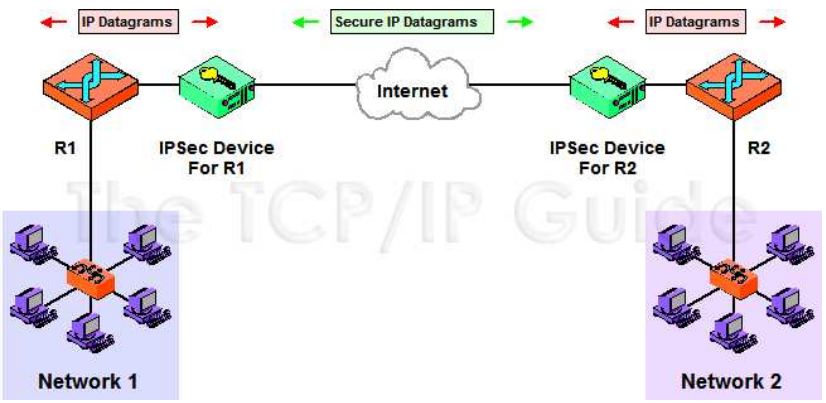


Abbildung 3: IPsec Architektur I

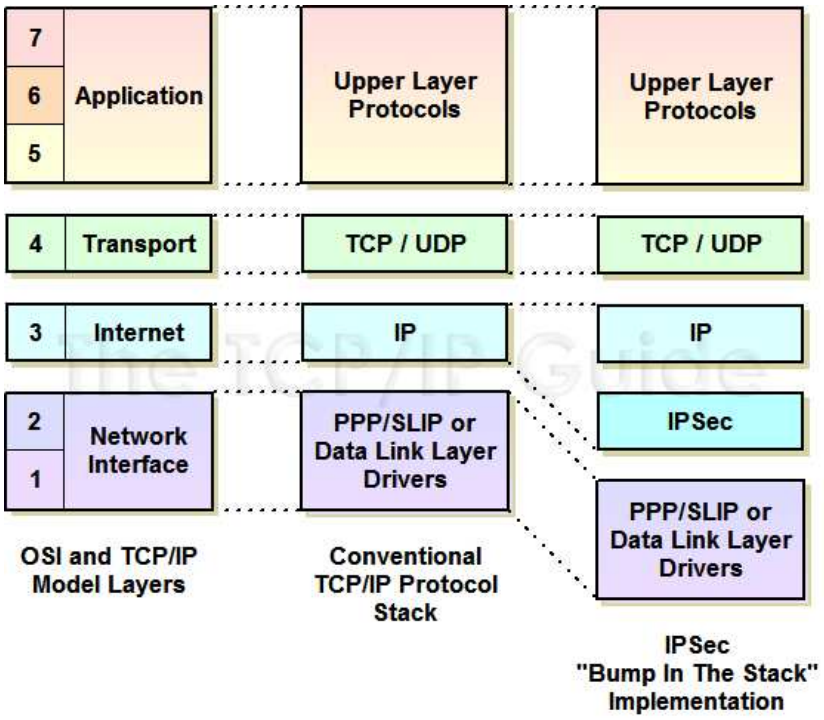


Abbildung 4: IPsec Architektur II

In der IP-Sec Spezifikation wird der Begriff der Security Association(SA) geprägt, diese beschreibt die Parameter einer IP-Sec gesicherten Verbindung zweier Teilnehmer. Eine Verbindung wird hierbei durch das Tripel SPI (Security Parameter Index), der Ziel-IPAdresse und dem verwendeten Protocol (ESP und AH verfügen über separate SAs) definiert. Beim SPI handelt es sich um eine zufällig generierte 32-bit Zahl, welche für die Verbindung generiert wurde, diese wird in dem Protokoll-Header übermittelt. Die durch die SA assoziierten Informationen einer Verbindung enthalten Angaben über den Verwendeten Algorithmus (Verschlüsselung, Authentifizierung), von den Algorithmen benötigte Informationen(Zertifikat, Schlüssel) sowie angaben zur Gültigkeitsdauer der Verbindung.

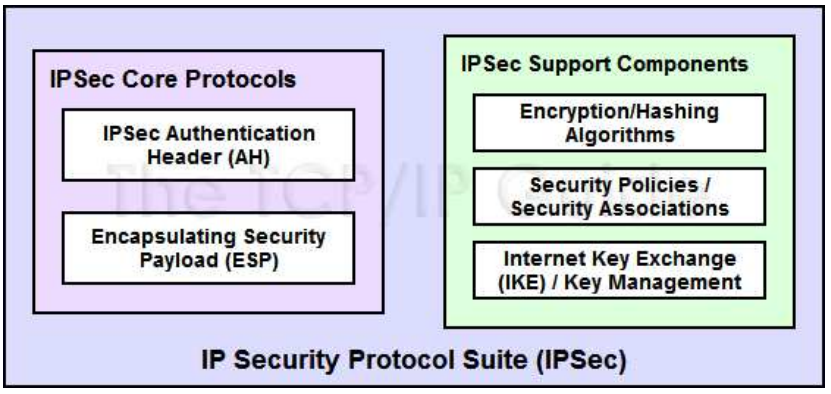


Abbildung 5: IPsec Protokolle

Eine Separation zwischen den topologischen Ort und der Identität mit dem vorgeschlagenen HIP würde also den Routing und die Packet Identifikation trennen. Der Host erkennt dann den Sender mit dem korrekten Schlüssen bevor er mit der Entschlüsselung beginnt. Die im Packet

stehenden IP Adressen sind deswegen Irrelevant. Diese Strategie stellt dein Host Identity Protokoll als eine alternative Lösung des Mobilität- und multihoming Problem.

3 Alternativlösung: Der Host Identity Protocol

Das HIP Mobilität- und Multihoming Protokoll bestimmt ein READRESS Packet (REA), das enthält die aktuelle IP Adresse des HIP mobile Hosts (HMH) enthält. Wenn dieser seinen Ort und seine Adresse ändert, erstellt er ein REA Packet, fügt das Private Key des benutzten HI und schickt dem Peer das Packet.

3.1 Grundlegender HIP Parameter: Das REA

Die logische Struktur des REA hat drei Ebenen: Host, IPsec (SPI) und Adresse. Die stehen wie folgt in Beziehung 6:

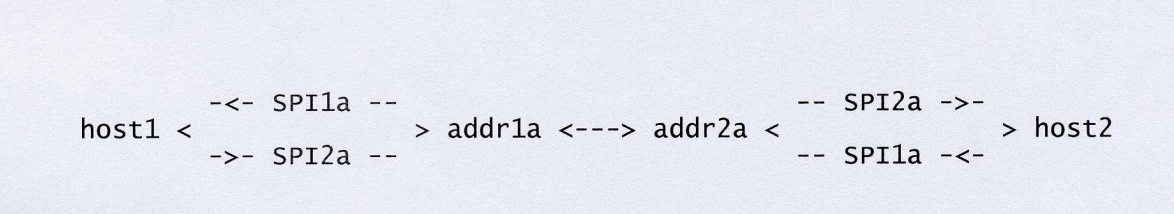


Abbildung 6: Beziehung zwischen hosts, SPIs, and addresses

Dabei sind die SA´s unidirektional. Host1 und Host2 wählen die SPI für ihre inbound SA. Adress1 und Adress2 sind die Adressen, die im Base HIP Exchange aufgetreten. Es sei denn, dass ein Host über mehr als ein inbound / (outbound) vefügt, und jede SPI könnte mehrere zugehörige Adressen vereinigen, wie im folgenden Schema 7.

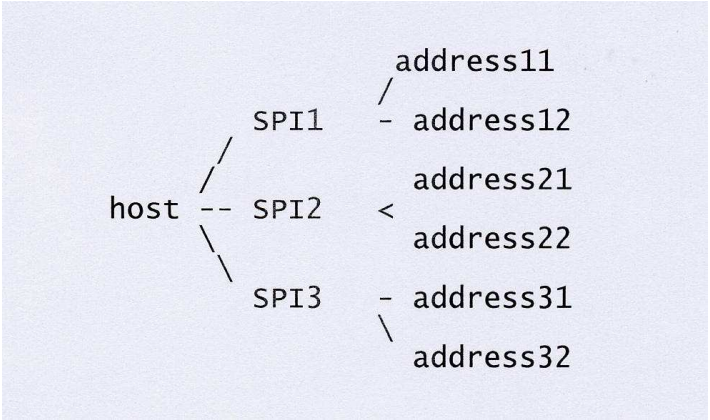


Abbildung 7:

Diese Vereinigung informiert darüber, durch welche Adressen der Host eigentlich zu erreichen ist. Ein Host kann ja mehrere SA´s (or SPI´s) mit einem Peer festlegen. Das Hauptziel is es, diese Adressen zu gruppieren und zwar diejenigen, bei denen ein ähnliches Verhalten anfällig ist, wie z.B. wenn der Host seine SPI2 Adresse ändert, ist es hochwahrscheinlich, dass Adresse21 und Adresse22 beides veraltet werden. Das REA informiert zwar über IP Adressen, gibt aber keine Garantie dafür, dass der Host unter denen erreichbar ist. Deswegen müsste ein

HIP Host diese Erreichbarkeit erst nachprüfen. Dies geschieht im Rahmen eines bestimmten Protokolls, das wir im Folgenden näher betrachten werden.

3.2 Protokoll überblick

3.2.1 Mobilitätsprotokoll

Wie schon erwähnt, muss ein mobiler Host über seine Adressen änderung informieren, wenn der Kommunikationskontext zu bewahren ist. Die nächsten Szenarien geben einen überblick auf das dafür zuständige Protokoll:

- Im ersten Beispiel ist der mobile Host kurz vom Peer Host getrennt, während er seine IP Adresse wechselt. Beim Erhalten einer neuen IP Adresse, schickt der mobile Host ein REA Packet an den Peer Host mit der Adressen Lebensdauer und ob sie eine bevorzugte Adresse ist.

  Im Falle eines Rekeying,das ist wenn der mobile Host ein neues inbound SA erstellt hat, dann steht der neue SPI auch im REA. Anderenfalls wird der existierende SPI im REA erkannt und es wird, als nächstes, auf das 'Acknowledgement (Ack)' gewartet.

  Abhängig davon, ob der mobile Host in einem Rekeying Zustand ist, und auch ob der Peer Host ein Rekeying oder eine Adressen Verifikation benötigt, sind ein Paar Rückmeldungen möglich.

  Im nächsten Schema 8 führt der Peer Host eine Adresse Verifikation aus. Ist sie nicht erwünscht, braucht er keine Rückmeldung für sein Update mehr, welcher dann nur als mobile Host Update-Ack dient.

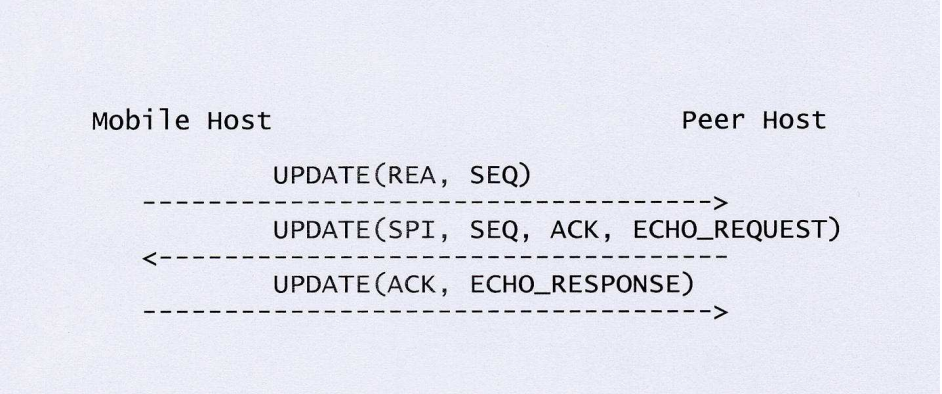


Abbildung 8: Readdress ohne Rekeying, aber mit Adressen Verifikation

- Besteht der Fall, dass beide Hosts ein Rekeying initiieren, trifft das dem folgenden Szenario 9
- Bei der übrigen Möglichkeit 10 ist das Rekeying vom Peer Host initiiert. Der Peer Host schickt seine Updates an die neue mobiler Host Adresse.

3.2.2 Multihomingprotokoll

Der früher eingeführte Begriff 'Multihoming' gibt dem REA Parameter eine andere Funktionalität. Dies entspricht zum Beispiel einem Host, der über mehr als eine Schnittstelle verfügt

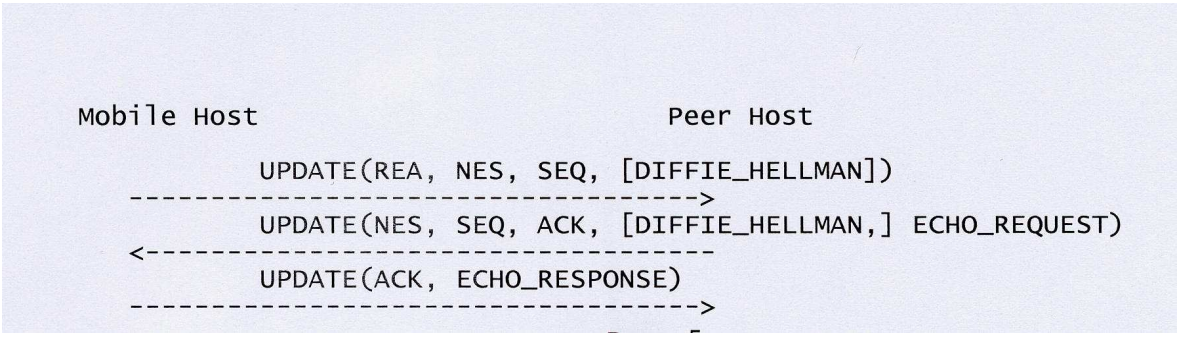


Abbildung 9: Readressierung mit mobile-initiierten Rekeying

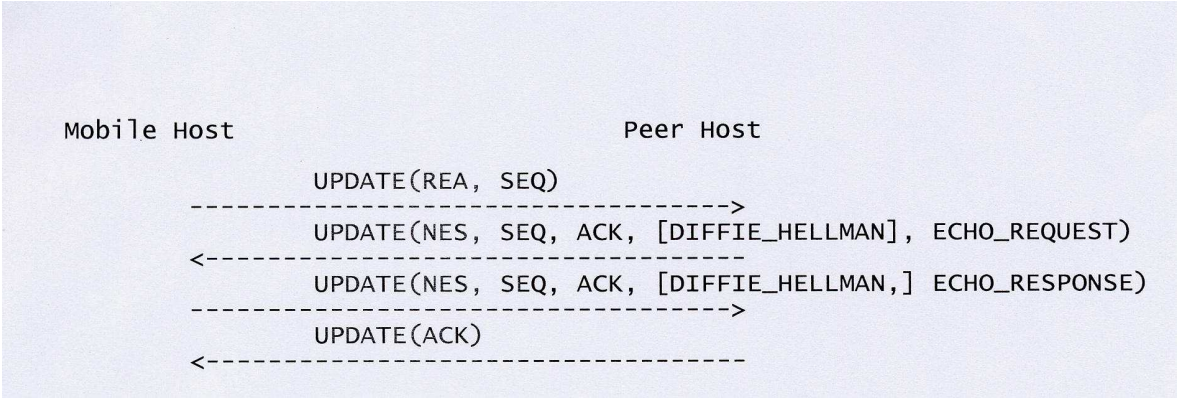


Abbildung 10: Readdress mit peer-initiiertem rekeying

und daher mehr als eine Adresse hat. Die werden im REA Parameter gemeldet, und es wird auf die bevorzugte Adresse hingewiesen. Eine neue verfügbare Schnittstelle wird über ein REA mit ein NES Parameter angedeutet. Im REA steht also der alte Index, im NES stehen aber beide SPI s, das Alte und das Neue, um anzuzeigen, dass der neue SPI den Alten nicht ersetzt. Wie bei den Mobillitaätsprotokoll Szenarien, kann der Peer Host ,im Rekeying Fall, eine Adressen Verifikation 11 ausführen.

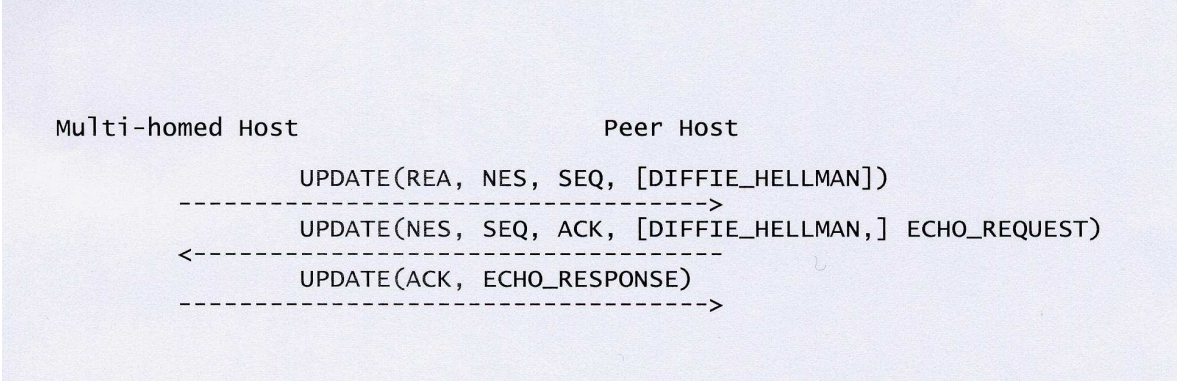


Abbildung 11: multihoming Szenario

In ähnlicher Weise geht es auch mit einem der vielfachen Internet Provider (zum Beispiel werden beide IPV4 und IPV6 Adressen angeboten) Zu beachten ist, dass ein Host gleichzeitig mobil und multihomed sein kann, und ein Multihomed Host kann aus Sicherheitsgründen, ei-

nige Adressen bekannt machen andere aber gleichzeitig verbergen.Einmal der Host über seine Adressen änderungen benachrichtigt hat, sollte er, als HIP Host, seine HI–IP Abbildung aktualisieren, sonst führt die ungültige IP Adresse zum Durchfall des HIP Base Exchange.Diese Vorraussetzung wird im nächsten Abschnitt erläutert.

3.3 Rendez-vous-Server

Ein DNS Server beschäftigt sich mit der HI–>IP Abbildung 12:

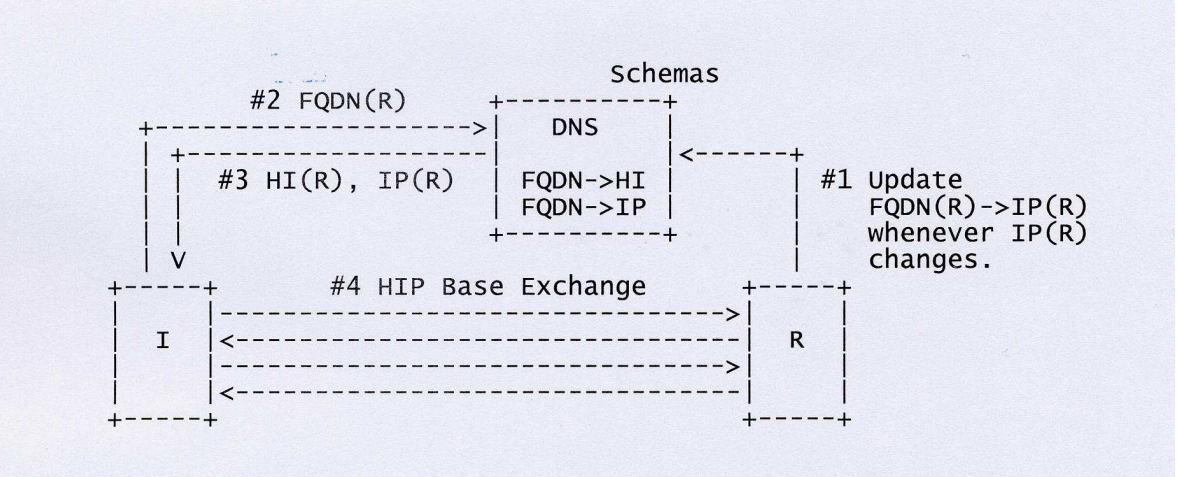


Abbildung 12: Base Exchange mit DNS Updates

Die Aktualisierung findet bei jeder änderung der R´s IP Adresse dynamisch statt.Wegen seiner hierarchischen Strukturr 13 des DNS´s wird der Aufwand dafür aber erheblich.

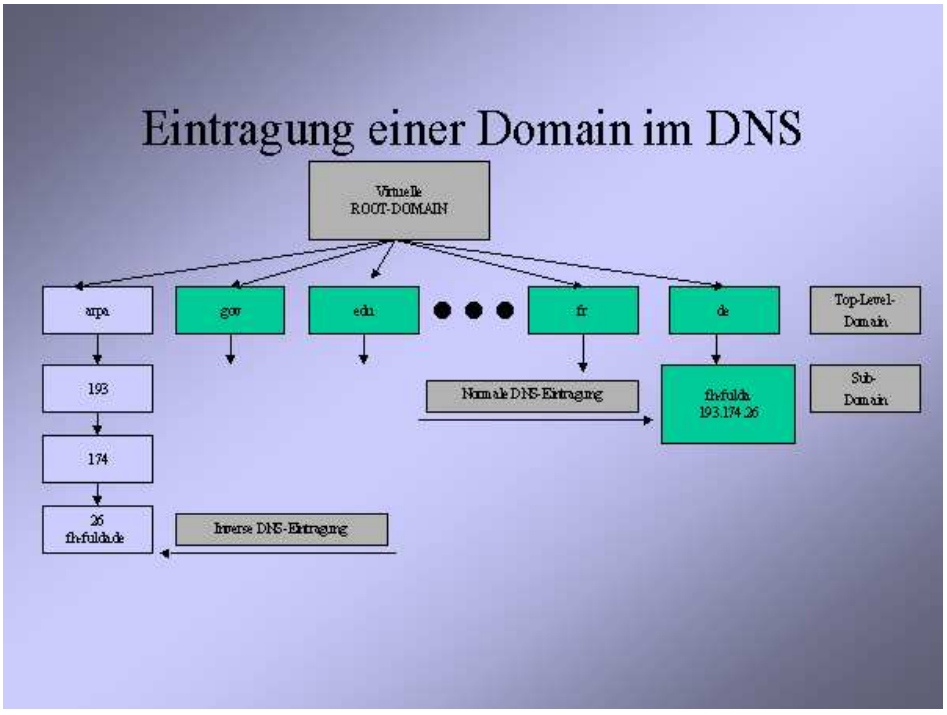


Abbildung 13: Struktur eines DNS

Diese Lösung wird daher bei einer oft geänderte IP Adresse besonders ineffizient. Mit der HIP Architektur kommt ein neuer Begriff zur Verwendung : Der Rendez-vous-Server. Dieser reduziert den erwähnten Aufwand, indem er der DNS Updates reduziert. Dabei speichert ein HIP Host im FQDN(fully qualified domain name)–>IP Eintrag nicht seine eigene IPAdresse sondern, die von seinem Rendez-vous-Servers.Angenommen ist, dass der Rendez-vous-Server seine IP Adresse nur selten ändert,damit wird ein geringerer Aufwand geschaffen. Dieser Rendez-vous server bildet der HI auf die IP Adresse ab, indem sie bei jeder Adressen änderung ihres HIP Hosts Bescheid gibt. Der HIP rendez-vous-Server ersetzt die IP von seinem Klient mit seiner Adresse, wie es im Folgenden 14 dargestellt ist.

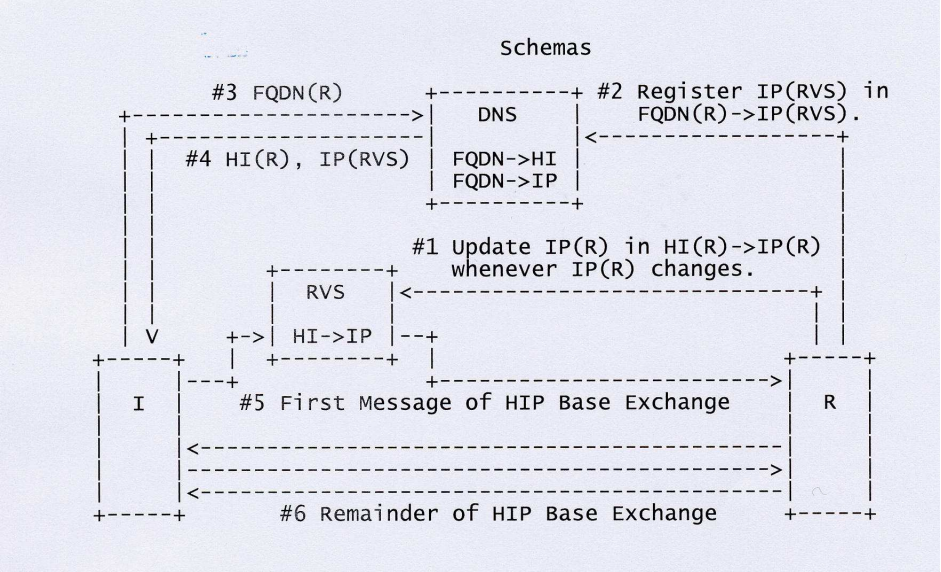


Abbildung 14: HIP Lookup und Base Exchange mit Rendezvous Serve

Frank Pählke.Mobilitätsunterstützung in Packetvermittelten Kommunikationsnetzen

www.watersprings.org/pub/id/draft-eggert-hip-rvs-00.txt

www.watersprings.org/draft-nikander-hip-mm-02.txt

www.elektronik-kompendium.de/sites/net/0906191.htm

www.elektronik-kompendium.de/sites/net/0812201.htm

www.fh-fulda.de/klingebiel/nbs-kolloquium/ipng1

www.tik.ee.ethz.ch/kurse/mobikom

Abbildungsverzeichnis

1 Multihoming 116
2 IPV6 Header 117
3 IPsec Architektur I 118
4 IPsec Architektur II 119
5 IPsec Protokolle 119
6 Beziehung zwischen hosts, SPIs, and addresses 120
7 120
8 Readdress ohne Rekeying, aber mit Adressen Verifikation 121
9 Readressierung mit mobile-initiierten Rekeying 122
10 Readdress mit peer-initiiertem rekeying 122
11 multihoming Szenario 122
12 Base Exchange mit DNS Updates 123
13 Struktur eines DNS 123
14 HIP Lookup und Base Exchange mit Rendezvous Serve 124

GPRS-basiertes Mobilitätsmanagement

Jochen Furthmüller

Kurzfassung

Inhalt dieser Seminararbeit ist das Mobilitätsmanagement der "General Packet Radio Service" - Technologie (GPRS). Dabei soll zunächst eine knappe Einführung in GPRS als Erweiterung von GSM erfolgen. Insbesondere werden neu zum GSM-Netzwerk hinzugekommene Netzwerkelemente und deren Funktion betrachtet. Schwerpunkt der Ausarbeitung ist die Darstellung von Abläufen und Protokollen, mit denen in GPRS die permanente Erreichbarkeit eines mobilen Endgerätes gewährleistet wird. Insbesondere werden Hard Handover und Soft Handover, die SRNS Relocation und das GPRS Tunneling Protocol (GTP) dargestellt.

1 GPRS: eine Einführung

1.1 Einführung

Der General Packet Radio Service (GPRS) ist eine Mobilfunktechnologie, die den General Standard of Mobilcommunication (GSM) um Dienste in der paketvermittelnden Domäne erweitert [Rudo03]. Während bei Kommunikation auf GSM-Basis für jeden Kommunikationsvorgang eine Leitung vermittelt wird (Circuit Switching, CS), gelangt bei GPRS jedes Datenpaket unabhängig von allen anderen von der Quelle zur Senke. Man spricht hier von Paketvermittlung (Packet Switching, PS).

Die Verwendung des Begriffes GPRS erfolgt sowohl im Sinne einer Technologie, die zwischen GSM und UMTS einzuordnen ist (Mobilfunk der 2,5-ten Generation), als auch im Sinne einer UMTS-Funktionalität. Diese Arbeit will das Augenmerk auf das Mobilitätsmanagement in der PS-Domäne richten. Da dies in beiden Fällen im Wesentlichen übereinstimmend geschieht, ist diese unterschiedliche Betrachtungsweise nicht von großer Bedeutung. Sie wird im Rahmen dieser Arbeit nur am Rande erwähnt.

Im Vergleich zu GSM bringt GPRS eine potentiell höhere Datenrate mit sich. Diese höhere Leistungsfähigkeit wird vor allem durch die paketorientierte Datenübertragung, neue Kodierungsverfahren und Kanalbündelung erreicht. Als weiterer Vorteil lässt sich die bessere Ressourcenausnutzung für den Netzbetreiber anführen. Ressourcen werden nicht mehr unabhängig vom tatsächlich zu transportierenden Datenvolumen beansprucht, und somit deutlich effizienter ausgenutzt. Demzufolge besteht die Möglichkeit mit einer volumenbasierten Abrechnung einen Teil dieses Vorteils an die Endkunden weiterzugeben.

Ein gravierendes Merkmal des GPRS-Ansatzes ist des weiteren, dass es auf der bestehenden und in Europa etablierten GSM-Technologie aufsetzt und somit bestehende Infrastruktur übernommen und erweitert werden konnte.

1.2 Neue Netzwerkelemente und Konzepte in GPRS

Um das Dienstspektrum um Dienste in der PS-Domäne zu erweitern, muss selbstverständlich die Netzinfrastruktur bestehender GSM-Netze ergänzt werden. Deshalb finden sich in GPRS-fähigen Mobilfunknetzen zusätzlich zu den GSM-Netzelementen sogenannte GPRS Support Nodes.

Hierbei wird weiter unterschieden zwischen dem Gateway GPRS Support Node (GGSN) und dem Serving GPRS Support Node (SGSN). Die Aufgabe und Einbindung dieser Knoten soll im Folgenden knapp erläutert werden.

1.2.1 Gateway GPRS Support Node (GGSN)

Wie in [Rudo03] beschrieben, kann der GGSN als Hauptvermittlungs- und Verbindungsknoten im GPRS-Netzwerk bezeichnet werden. Er dient als Schnittstelle zu anderen Datennetzen, zum Beispiel dem Internet.

Damit ein mobiles Endgerät erreichbar ist, müssen die Datenpakete entsprechend durch das Netzwerk geleitet werden. Dieses Routing ist eine Aufgabe des GGSN. Die dafür benötigten Benutzerdaten sind im Home Location Register (HLR) enthalten. Auf dieses Register kann der GGSN direkt mittels einer eigens dafür definierten Schnittstelle zugreifen. Diese Schnittstelle ist jedoch nicht zwingend vorhanden. Eine weitere Möglichkeit ist, über einen SGSN auf das HLR zuzugreifen. Jeder SGSN ist mit einem HLR verbunden, und kann somit über die benötigten Informationen verfügen. Um den Verwaltungsaufwand gering zu halten wird oft auf die optionale Schnittstelle zwischen GGSN und HLR verzichtet und der Zugriff über die SGSN durchgeführt.

Um eine abgesicherte Übertragung von Informationen zwischen verschiedenen GGSN zu ermöglichen, unterstützen die GGSN das Tunneln von Protocol Data Units.

Ferner sammelt der GGSN Daten für die Rechnungserstellung und unterstützt die Anfertigung einer Routing-Tabelle sowie das sogenannte Address Mapping.

1.2.2 Serving GPRS Support Node (SGSN)

Der SGSN sorgt für die Bereitstellung elementarer Funktionalitäten wie das Mobilitätsmanagement und die Sicherheit der Datenübertragung [Rudo03]. Der SGSN verfügt über einen Packet Data Protocol Context. Dabei handelt es sich um ein Diensteprofil, das benötigt wird, um Daten zwischen dem GGSN und der Mobilstation austauschen zu können. Darin ist zum Beispiel verzeichnet, in welchem Base Station Subsystem (eine Funknetzkomponente des GPRS-Netzwerkes) sich ein Benutzer aufhält.

Im Gegensatz zu der Vorgehensweise in GSM, bei der die Verschlüsselung von den Sendern bzw. Empfängern in den Basisstationen erledigt wurde, werden bei GPRS die Datenströme zwischen dem SGSN und der Mobilstation verschlüsselt. So kann die Übertragungssicherheit erhöht werden.

Um die Möglichkeit einer volumenbasierten Rechnungsstellung zu nutzen, kann der SGSN Buch über das Transfervolumen einer bestimmten Mobilstation führen. Dies schließt jedoch nicht aus, dass auch Gebühren für die Dauer einer Verbindung erhoben werden.

Schließlich leitet der SGSN die für eine Mobilstation bestimmten Pakete an die entsprechende Funknetzkomponente weiter.

1.2.3 Lokalisierungszonen

Um eine Lokalisierung des Benutzers zu ermöglichen, ist das Mobilfunktnetz in geographische Zonen unterteilt. Für die beiden Dienstbereiche, die Paketvermittlung und die Leitungsvermittlung, sind hierfür verschiedene Zonen definiert. Während man in der leitungsvermittelnden Domäne von einer Location Area (LA) spricht, wird in der paketvermittelnden Domäne der Begriff Routing Area (RA) verwendet.

Zwar sind Routing Areas und Location Areas grundsätzlich unabhängig voneinander definiert, dennoch können gewisse Gesetzmäßigkeiten festgestellt werden:

- Eine Location Area enthält eine oder mehrere Routing Areas ,
- eine Routing Area gehört immer zu genau einer Location Area.

Jede Routing Area wird von ausschließlich einem SGSN betreut. Verschiedene in einer Location Area enthaltene Routing Areas können jedoch zu verschiedenen SGSN gehören. Dies ist in [Lesc02a] beschrieben.

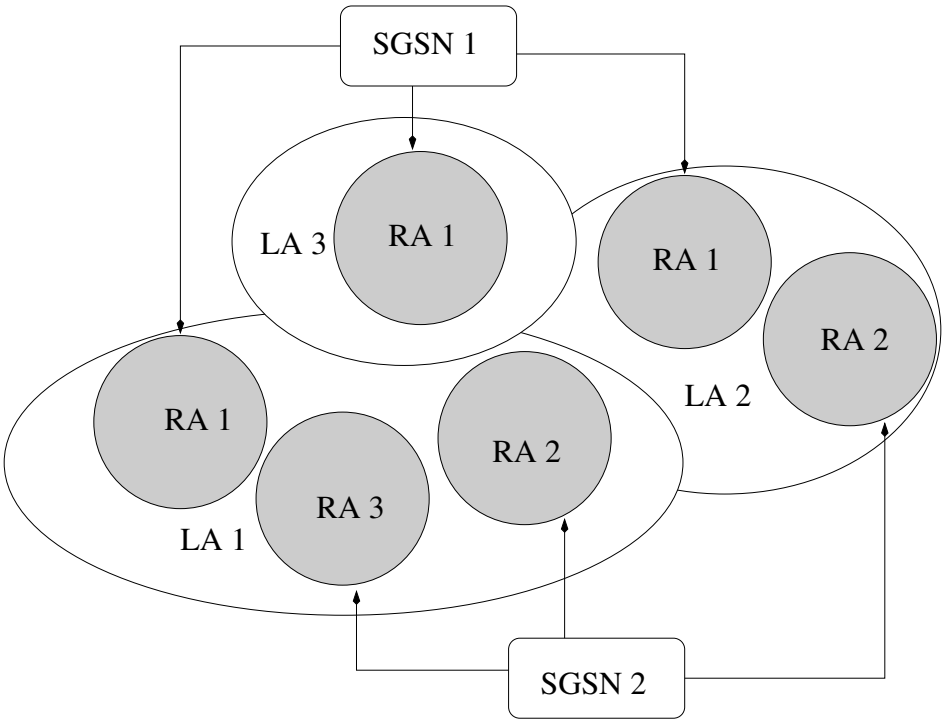


Abbildung 1: Routing- / Location Area

Sobald ein Endgerät eingeschaltet wird, meldet es sich beim Netz an, das von nun an den Aufenthaltsort des Gerätes kennt. Jede Zelle des Funknetzes signalisiert dem Mobilgerät ihre Zugehörigkeit zu einer bestimmten Lokalisierungszone durch das Senden von spezifischen Informationen über den sogenannten Peilkanal BCCH (Broadcast Control Channel). Über diesen Kanal werden allgemeine Netzwerkinformationen versendet.

Die Größe der Lokalisierungszonen wird vom Netzbetreiber bestimmt und muss der Abwägung zwischen zahlreichen Zellwechseln (bei kleinen Zellen) und starker Beanspruchung der Ressourcen einer einzelnen Zelle (bei großen Zellen) Rechnung tragen.

1.2.4 Das GPRS Tunneling Protocol (GTP)

Zwischen GGSN, der Zugangstelle zum IP-Netz, und dem SGSN müssen Benutzerdaten ausgetauscht werden. Für diesen Datentransport wird das GPRS Tunneling Protocol verwendet. Eine Beschreibung dieses Protokolls ist in [Lesc02b] zu finden.

Der Begriff Tunneln bedeutet, dass ein zu übertragendes IP-Paket auf der Strecke zwischen GGSN und SGSN in ein spezielles GTP-Paket eingepackt wird.

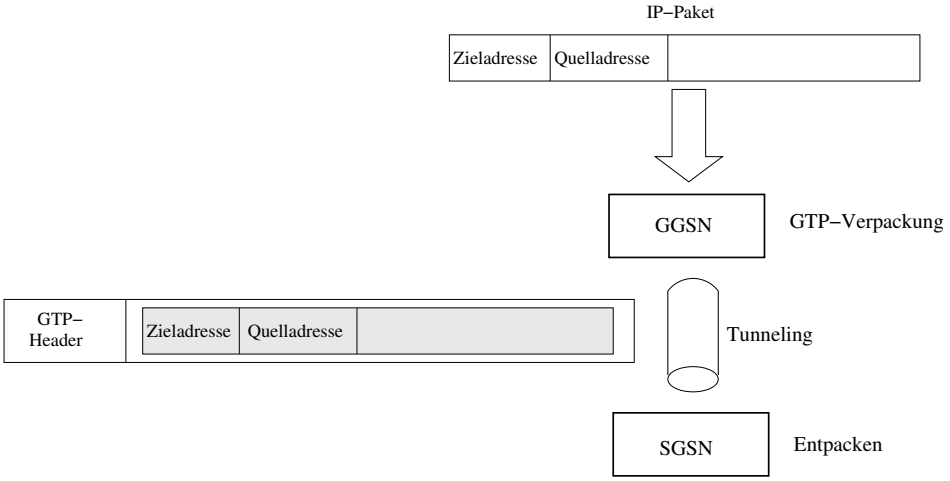


Abbildung 2: Datenübertragung zwischen SGSN und GGSN

In dieser Hinsicht entspricht der GGSN dem Home Agent in MobileIP, während man den SGSN mit dem Foreign Agent assoziieren kann. So wird Mobilität zwischen verschiedenen Subnetzen eines gemeinsamen Netzes ermöglicht.

Die an den mobilen Knoten adressierten Pakete gelangen zunächst zum GGSN. Der verpackt diese in ein GTP-Paket und tunnelt sie zum aktuellen SGSN der Mobilstation. So ist es möglich, das Routing von Paketen in zwei unabhängige Prozesse zu untergliedern:

1. das Routing zu den IP-Adressen der Endgeräte
2. das der Mobilität des Endgerätes geschuldete Routing vom Referenznetz (GGSN) zum aktuellen Netz (SGSN)

Wie Abbildung 3 verdeutlicht, werden die in GTP-Pakete verpackten IP- oder X.25-Pakete mittels der Transportprotokolle UDP oder TCP verschickt. Die untersten beiden Schichten werden in der Norm nicht näher spezifiziert, hier kann zum Beispiel Frame Relay eingesetzt werden.

Ein weiterer Tunnel (der SNDCP-Tunnel) wird zwischen dem SGSN und dem Mobilgerät verwendet. Dieser Tunnel soll hier aber lediglich erwähnt und seine Betrachtung nicht vertieft werden.

1.3 Anwendungen von GPRS

GPRS findet Anwendung überall dort, wo mittels Mobilfunk Daten übertragen werden sollen, und die Datentransferrate von GSM nicht ausreicht. So wird GPRS zum Beispiel als Datenübertragungsdienst für WAP-Seiten oder im Multimedia Messaging Service (MMS) für Mobilfunkgeräte genutzt. Außerdem können GPRS-fähige Mobiltelefone oder aber GPRS-Netzwerkkarten als Modem eine Internetverbindung für einen Laptop ermöglichen [Rudo03].

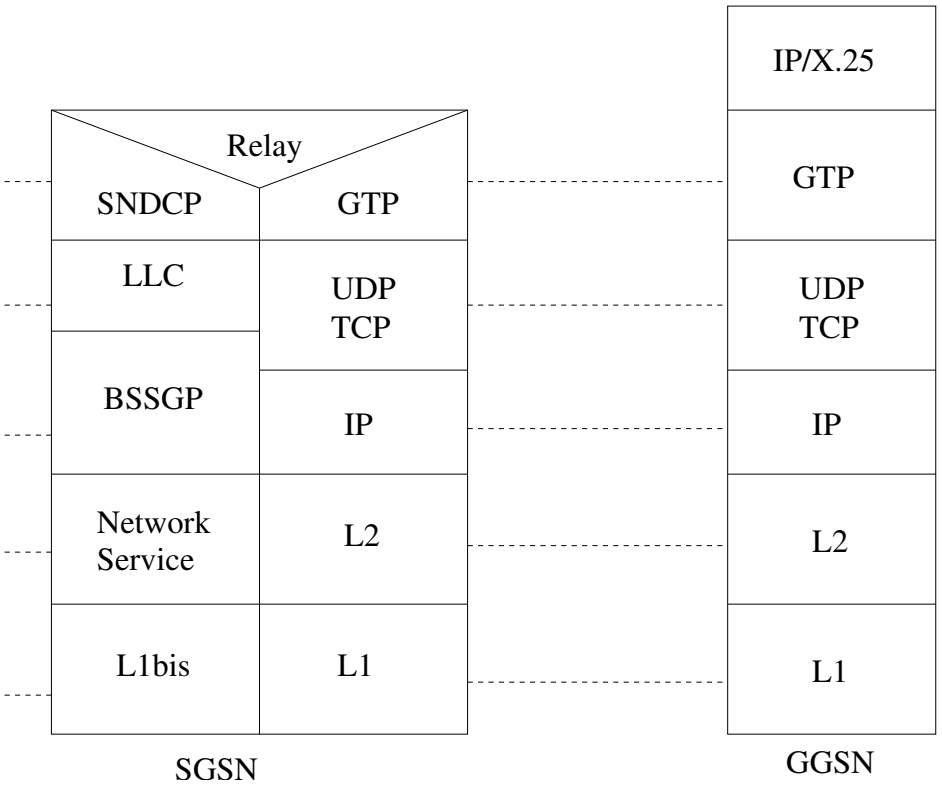


Abbildung 3: Einordnung des GTP in die Protokollschichten von GPRS

2 Prozesse des Mobilitätsmanagements in GPRS

Mobilitätsmanagement bedeutet in erster Hinsicht eine Buchführung über den Aufenthaltsort eines Mobilgerätes. Das Mobilfunknetz muss Kenntnis darüber haben, wo sich ein Endgerät befindet, um die ihm zugedachten Datenpakete geschickt weiterleiten zu können.

Dieses Wissen über die Lokalisierung eines Mobilknotens muss immer dann aktualisiert werden, wenn der mobile Knoten eine Zelle wechselt. Darum soll es jetzt um Zellwechsel in GPRS gehen.

Grundsätzlich wird hierbei zwischen Zellwechsel im aktiven Modus und Zellwechsel im passiven Modus unterschieden. Deshalb werden zunächst die zugrundeliegenden Zustände im Mobilitätsmanagement erläutert.

2.1 Die Zustände im Mobilitätsmanagement

Um die Vorgänge zwischen einer Mobilstation und dem sie betreuenden SGSN zu veranschaulichen, werden endliche Zustandsautomaten verwendet [LHCC01]. Obwohl sich diese Automaten für die paketvermittelnde Domäne in UMTS und GPRS als eigenständige Technologie nur marginal unterscheiden, wurden doch neue Zustandsbezeichnungen eingeführt:

Während die Zustände in GPRS mit IDLE, STANDBY und READY bezeichnet werden, finden im UMTS-Kontext die Begriffe PMM-DETACHED, PMM-IDLE und PMM-CONNECTED Anwendung.

Nun soll kurz auf die einzelnen Zustände und die Übergänge zwischen diesen eingegangen werden:

Zustände:

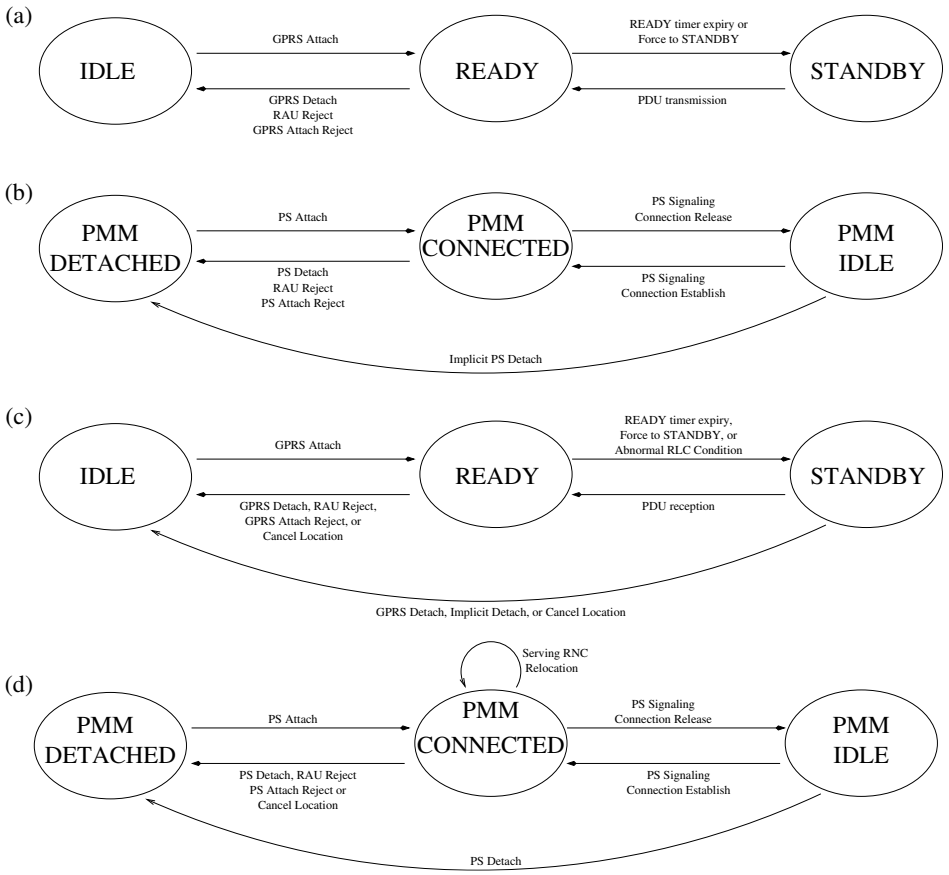


Abbildung 4: Zustände beim Mobilitätsmanagement

- IDLE oder PMM-DETACHED: Das Endgerät kann keine Daten empfangen, da das GPRS-Netz keine Kenntnis von dem Endgerät hat.
- STANDBY oder PMM-IDLE: Im Mobilgerät und im SGSN wurde ein Kontext aufgebaut.
- READY oder PMM-CONNECTED: In diesem Zustand findet der Austausch von Nutzdaten statt.

Zustandsübergänge:

- IDLE -> READY (PMM-DETACHED -> PMM-CONNECTED): Dieser Zustandsübergang wird durch die Mobilstation veranlaßt, wenn sie sich beim Netz anmeldet.
- READY -> IDLE (PMM-CONNECTED -> PMM-DETACHED): Veranlasst vom SGSN oder der Mobilstation geschieht dieser Übergang wenn das Netz oder das Endgerät eine Abmeldung (Detach) vornimmt. Der SGSN kann auch einen solchen Zustandswechsel initiieren wenn er vom HLR über die Ungültigkeit der bisherigen Lokalisierung informiert wird, oder der SGSN eine Anfrage für ein Routing-Area-Update oder einen Anmeldeversuch der Mobilstation zurückweist.
- READY -> STANDBY (PMM-CONNECTED -> PMM-IDLE): Sowohl SGSN als auch die Mobilstation können diesen Übergang veranlassen. Dies geschieht, wenn ein Timer abläuft, der vom SGSN gesetzt wurde, und keine LLC PDU übermittelt wurde.

- STANDBY -> READY (PMM-IDLE -> PMM-CONNECTED): Diesen Zustandswechsel stößt die Mobilstation an, indem sie eine Dateneinheit der Schicht 2 (Linklayer Control Protocol Data Unit, LLC PDU) an den SGSN schickt.

- STANDBY -> IDLE (PMM-IDLE -> PMM-DETACHED): Wenn der SGSN über eine gewisse Zeit hinweg keine Routing Area-Updates des mobilen Knoten empfängt, läuft ein Timer ab, woraufhin der SGSN das Mobilgerät als abgemeldet betrachtet und diesen Übergang durchführt.

  Eine andere Ursache für die Durchführung dieses Übergangs durch den SGSN kann sein, dass der SGSN vom HLR eine Nachricht erhält, die die bisherige Lokalisierung des Mobilgeräts für ungültig erklärt. Der Verbindungskontext wurde dann bereits zu einem neuen SGSN transferiert, der alte kann gelöscht werden.

  Als eine Neuerung in UMTS gegenüber GPRS kann dieser Übergang auch von der Mobilstation initiiert werden, wenn etwa die Batterie oder die SIM-Karte entfernt wird.

2.2 Mobilitätskontrolle im passiven Modus

Auch wenn kein Nutzdatenaustausch stattfindet muss dem Netz der Aufenthaltsort einer sich im Standby-Modus befindlichen Mobilstation bekannt sein. Standby heißt hierbei, dass das Endgerät eingeschaltet ist, jedoch über keine Verbindung zum Netz verfügt. Demzufolge muss ein Endgerät auch im passiven Modus in gewissem Umfang "aktiv" bleiben, also das Netz über eventuell anstehende Zellwechsel informieren.

2.2.1 Aktualisieren der Routing Area

Selbst wenn eine Mobilstation in der gleichen Routing Area verbleibt, ist es notwendig, in bestimmten zeitlichen Abständen eine Aktualisierung durchzuführen. Der Zeitraum zwischen zwei Aktualisierungen beträgt zwischen sechs Minuten und 25,5 Stunden und wird vom Netzbetreiber festgelegt. Um die Mobilstationen darüber zu informieren, mit welcher Periodendauer sie ihre Zugehörigkeit zu einer Routingarea aktualisieren müssen, wird diese Periodendauer über den Peilkanal jeder Zelle gesendet.

Die Abbildung 5 illustriert die Aktualisierung einer Routing Area, wie sie in [Lesc02a] beschrieben wird, unter der Voraussetzung, dass die alte Routing Area und die neue nicht von dem selben SGSN betreut werden.

1. Zunächst wird eine RRC-Verbindung (Radio Resource Control Connection) zwischen dem Mobilgerät und dem Radio Network Subsystem (RNS) hergestellt. Mittels dieser wird der Wunsch des Mobilgeräts, zu einer neuen Routing Area zu gehören, an seinen neuen SGSN geschickt. Dabei übergibt es seinen alte Routing Area Identifier (RAI) an den SGSN.

2. Der neue SGSN erhält vom alten SGSN auf Anfrage die eindeutige Bezeichnung des Mobilgerätes in Form der International Mobil Station Identity (IMSI) und einen Datenvektor mit dem das Endgerät authentisiert werden kann. So kann die Verbindung verschlüsselt werden. Über die jetzt gesicherte Verbindung werden sensible Daten wie zum Beispiel die neue TMSI (Temporary Mobil Station Identity) des Endgerätes ausgetauscht.

3. Im dritten Schritt bringt der neue SGSN das Home Location Register (HLR) auf einen aktuellen Stand. Daraufhin benachrichtigt das HLR den alten SGSN über den Wechsel

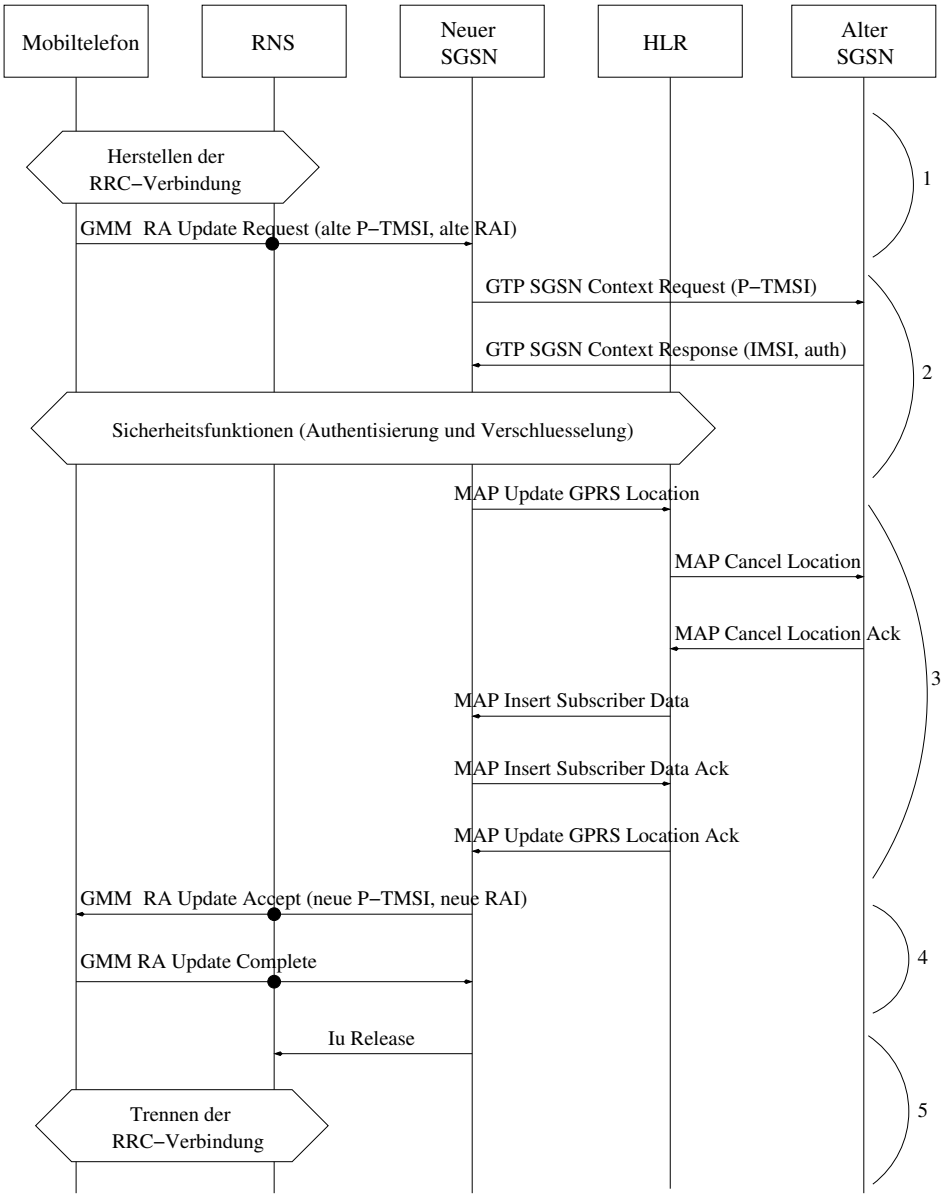


Abbildung 5: Aktualisierung der Routing Area

der Routing Area. Nun kann dieser alle nicht mehr benötigten Daten löschen. Die vom HLR an den neuen SGSN übermittelten Insert Subscriber Data enthalten Informationen über die Dienste, die einem Abonnenten zur Verfügung stehen.

4. Die Mobilstation wird nun über die erfolgte Routing Area Aktualisierung benachrichtigt. Unter anderem erhält es vom jetzigen SGSN eine neue TMSI.

5. Nach diesen Schritten wird die Verbindung zwischen Mobilgerät und Netz wieder getrennt.

2.2.2 Reselektion der Zelle

Auch im passiven Modus (Standby) gehört das Endgerät zu genau einer Zelle im Netz. Diese Zelle wird beim Einschalten des Gerätes ausgewählt und heißt dann Initialzelle. Wenn sich die Mobilstation innerhalb des Netzes fortbewegt, kann es sein, dass die momentan gewählte

Zelle nicht mehr akzeptabel oder eine andere einfach besser geeignet ist. Dann erfolgt eine Reselektion der Zelle. Zur Auswahl einer Zelle verwendet das Gerät drei Mechanismen: das S-Kriterium, das R-Kriterium und das Messungskriterium, die in [Lesc02a] weitergehend erläutert werden.

- Das S-Kriterium (Selection) dient zur Auswahl überhaupt in Frage kommender Kandidatenzellen
- Das R-Kriterium (Ranking) dient dazu, die in Frage kommenden Kandidatenzellen gemäß ihrer Eignung zu ordnen. Die "beste" Zelle hat das höchste R-Kriterium.
- Das Messungskriterium sorgt dafür, dass das Mobilfunkgerät nicht zu viel Zeit und Energie auf das Messen in benachbarter Zellen verwendet. Solange die Qualität der momentanen Zelle über einer bestimmten Schwelle liegt, werden umliegende Zellen gar nicht in Betracht gezogen.

2.3 Mobilitätskontrolle im aktiven Modus

Im aktiven Modus, auch als Connected Mode bezeichnet, ist ein Anwender mit dem Netz verbunden. Wenn er sich innerhalb des Netzes fortbewegt, soll die Konnektivität möglichst ununterbrochen fortbestehen. Dies ist Aufgabe des Mobilitätsmanagements in GPRS.

Ob die Mobilitätskontrolle des Mobilgerätes durch das Mobilgerät selbst, oder aber durch das Zugangsnetz erfolgt, hängt vom Zustand der RRC-Verbindung (Radio Resource Control) zwischen dem Mobilgerät und dem Serving Radio Network Controller (SRNC) ab. An dieser Stelle soll nicht weiter auf Details der RRC-Verbindung eingegangen werden. Es genügt zu erwähnen, dass es verschiedene Zustände gibt, von denen es abhängt, wer die Kontrolle über den Zellwechsel innehat.

Eine Übersicht darüber gibt die folgende Tabelle:

RRC-Zustände und Mobilitätskontrolle [Lesc02a]

RRC-Zustand	**Mobilitätskontrolle**	**Andwendung**
CELL_DCH	Netz	Handover
CELL_FACH	Netz oder Mobilgerät	Handover oder Cell-Update
CELL_PCH	Mobilgerät	Cell-Update
URA_PCH	Mobilgerät	URA-Update

Tabelle 1: RRC-Zustände und Mobilitätskontrolle

2.3.1 Mobilitätskontrolle durch das Endgerät

Während in den Zuständen CELL_PCH und URA_PCH trotz einer bestehenden Verbindung keine Anwenderdaten ausgetauscht werden, findet im CELL_FACH-Zustand sehr wohl ein Datenaustausch statt. Je nach anfallendem Volumen kann das Netz die Kontrolle jedoch an das Mobilgerät delegieren.

Dann muss das Mobilgerät das Netz, im Speziellen den Serving Radio Network Controller (SRNC), über seine neue Position benachrichtigen. Hierfür dient die Prozedur Cell Update.

Wenn eine Mobilstation lediglich ein niedriges Aktivitätsniveau innehat, oder es sich sehr schnell fortbewegt, kann der RRC-Verbindungszustand zu URA_PCH wechseln. URA

(UTRAN Registration Area) bezeichnet eine Gruppe von Zellen. So wird die Beanspruchung des Zugansnetzes durch die Cell Update Prozedur vermindert.

Der SRNC dient somit als Bezugspunkt der Mobilität im Connected Mode. Dadurch können die Aktualisierungsvorgänge im Zugangsnetz von denen im Kernnetz getrennt werden [Lesc02a].

2.3.2 Mobilitätskontrolle durch das Netz (Handover)

Wenn ein Endgerät zu einer anderen Funkeinrichtung wechselt und dies vom Netz kontrolliert wird, gibt es verschiedene Möglichkeiten, diesen Handover zu gestalten: bei GSM wird der sogenannte Hard Handover eingesetzt, in UMTS das Soft Handover beziehungsweise Softer Handover [Lesc02a].

- Hard Handover: Ein GSM-Mobilfunkgerät kann stets nur eine Funkverbindung haben. Darum entsteht beim Wechsel zu einer neuen Funkeinrichtung eine kurze Unterbrechung. Die alte Verbindung wurde bereits abgebaut, die neue noch nicht aufgebaut. In dieser Zeit können Datenpakete verloren gehen, was bei reiner Sprachübertragung hinnehmbar ist.
- Soft Handover: Für Soft Handover ist vonnöten, dass ein Mobilfunkgerät mit bis zu sechs Sendestationen zugleich verbunden sein kann. Diese gleichzeitig zwischen Netz und Mobiltelefon bestehenden Verbindungen bezeichnet man als Active Set. Die zu versendenden Daten werden simultan auf allen Verbindungen gesendet und empfangen. Vor dem Abbau einer Verbindung werden also Verbindungen mit einer oder mehreren Nachbarzellen aufgebaut. Es besteht also eine ununterbrochene Konnektivität zwischen dem Netz und dem Mobilfunkgerät.
- Softer Handover: Dabei handelt es sich um einen Soft Handover bei dem alle am Handover-Verfahren beteiligten Zellen zum gleichen NodeB gehören.

2.3.3 Relocation

Durch die schon oben erwähnte RRC-Verbindung ist ein mobiler Knoten mit einem SRNC verbunden. Es gibt Situationen in denen ein Mobilgerät einem neuen SRNC zugeordnet wird. Dieser Vorgang, der in [Lesc02a] erklärt wird, wird Relocation genannt und wird durchgeführt, um das Routen im UTRAN zu optimieren oder um einen Hard Handover zu unterstützen.

- Optimierung des Routens: Bewegt sich ein Mobilfunkgerät fort, so entfernt es sich notwendigerweise von seinem SRNC. Bleibt der SRNC während der ganzen RRC-Verbindung der gleiche, so entstehen hohe Kosten. Also wird mittels der Relocation-Prozedur ein neuer SRNC bestimmt.
- Unterstützung des Hard Handover: Manchmal ist kein Soft Handover möglich, weil entweder eine dafür nötige Schnittstelle nicht verfügbar ist, oder aber die fraglichen Zellen verschiedene Funkfrequenzen oder Zugangstechnologien verwenden. Dann muss ein Hard Handover durchgeführt werden. Falls Start und Zielzelle nicht zum gleichen RNC gehören geschieht dies einschließlich einer SRNC-Relocation.

Grundsätzlich kann man zwei verschiedene Szenarien für Relocation betrachten. Im einen Fall hängen beide, der alte sowie der neue SRNC, vom gleichen SGSN ab. Dann spricht man von

einer Intra-SGSN-Relocation. Falls dies nicht der Fall ist, stimmen sich der alte und neue SGSN mittels des GPRS Tunneling Protocols ab. Man spricht dann von einer Inter-SGSN-Relocation.

Eine solche Relocation mit Wechsel des SGSN kann in drei Phasen einteilt werden:

1. Vorbereitung: die Relocation-Anfrage des momentanen SRNC wird mittels der Forward Relocation Request an den Ziel SGSN weitergeleitet. Sobald der Ziel-RNC die erforderlichen Ressourcen zugesichert hat, erteilt der Ziel-SGSN eine Zusage.

2. Durchführung: Der alte SGSN wird über den neuen SGSN informiert. Das Mobilfunkgerät ist mit dem neuen SGSN verbunden. Der GGSN leitet nun alle Anwenderdaten, die von außen kommen, an den neuen SGSN weiter.

3. Freigabe nicht mehr benötigter Ressourcen: Der alte SRNC wird angewiesen, nicht mehr benötigte Ressoucen freizugeben.

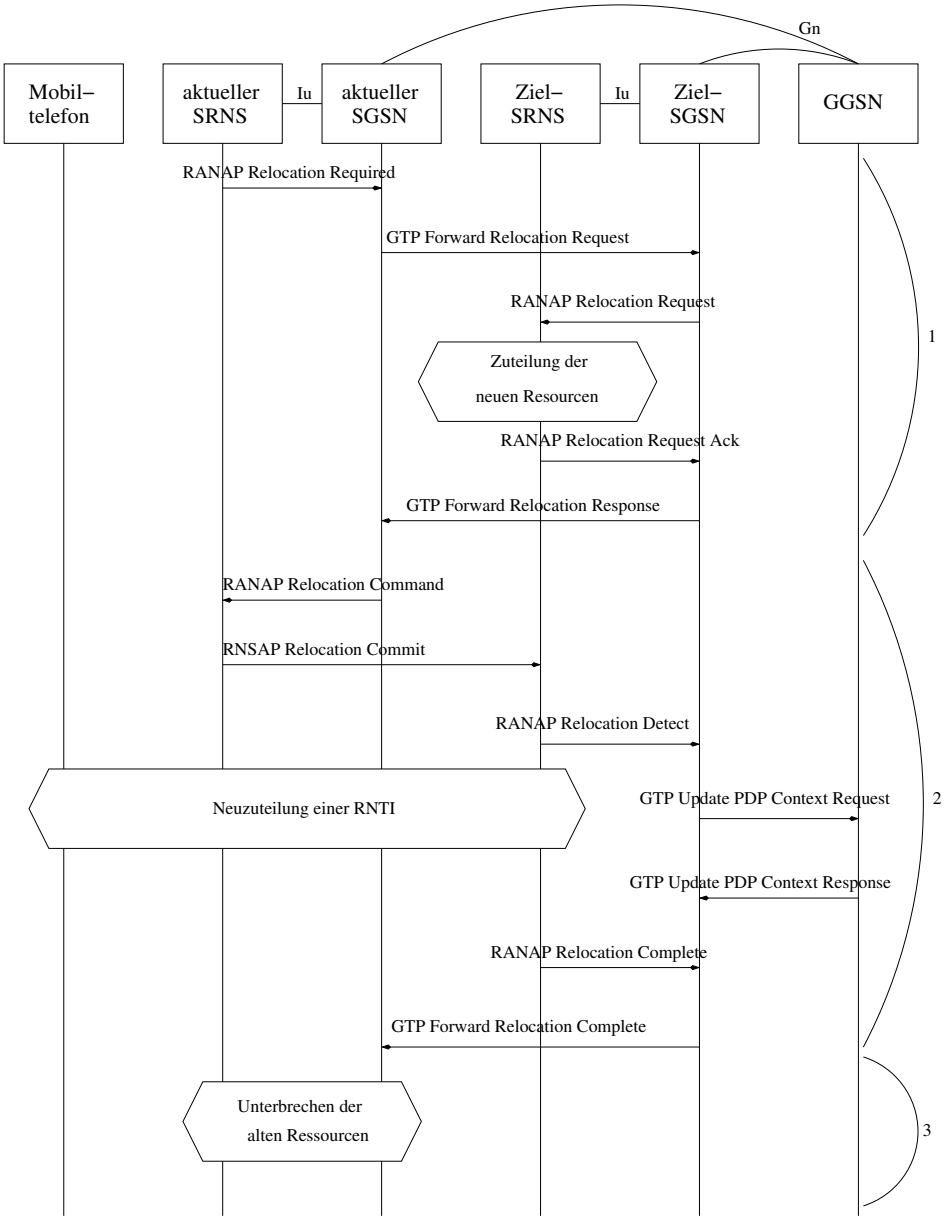


Abbildung 6: Relocation mit SGSN-Wechsel

Literatur

[Lesc02a] Pierre Lescuyer. *Mobilitätsmanagement*, Kapitel 8, S. 217–263. dpunkt-Verlag. 2002.

[Lesc02b] Pierre Lescuyer. *Mobilitätsmanagement*, Kapitel 2, S. 44–45. dpunkt-Verlag. 2002.

[LHCC01] Yi-Bing Lin, Yieh-Ran Haung, Yuan-Kai Chen und Imrich Chlamtac. Mobility Management: from GPRS to UMTS. *Wireless Comunications and Mobile Computing* Band 27, August 2001, S. 339–359.

[Rudo03] Rolf Rudolf. *GPRS Basics: die Grundkonzepte des General Packet Radio Service*. Hrsg. von T.O.P. BusinessInteractive. 2003.

Abbildungsverzeichnis

1 Routing- / Location Area 129

2 Datenübertragung zwischen SGSN und GGSN 130

3 Einordnung des GTP in die Protokollschichten von GPRS 131

4 Zustände beim Mobilitätsmanagement 132

5 Aktualisierung der Routing Area 134

6 Relocation mit SGSN-Wechsel 138

Tabellenverzeichnis

1 RRC-Zustände und Mobilitätskontrolle 135

MOBIKE-basiertes Mobilitätsmanagement

Ruojing WEN

Kurzfassung

Die Nachfrage nach mobiler Nutzung des Internets wächst heutzutage sehr schnell und Benutzer fordern immer mehr Flexibilität. Die allgegenwärtige Erreichbarkeit und unbeschränkte Kommunikation werden als einer der größten Trends in Anspruch genommen. Um diese Anforderungen zu erfüllen und gleichzeitig Sicherheit im Internet zu gewährleisten müssen entsprechende Sicherungsprotokolle im Internet erstellt werden. MOBIKE ist deshalb als eine Erweiterung des Protokolls IKEv2 zu entwickeln. Hier soll nun MOBIKE-basiertes Mobilitätsmanagement näher vorgestellt werden.

1 Einleitung

1.1 Übersicht

Diese Ausarbeitung beschreibt die MOBIKE-basierte Verwaltung der Mobilität. Abschnitt 2 erklärt das Schlüsselaustausch-Protokoll Version 2. Eine Erweiterung dieses Protokolls ist das MOBIKE, das in Abschnitt 3 beschrieben wird. Das MOBIKE-Protokoll unterstützt die Mobilität und Multihoming im Internet. In dieser Ausarbeitung wird insbesondere die Mobilitätsunterstützung betrachtet. Abschnitt 4 konzentriert sich auf die Unterstützung und die Verwaltung der Mobilität basierend auf MOBIKE. Zuletzt werden in Abschnitt 5 die Entwurfsentscheidungen des Protokolls diskutiert.

Im folgenden Unterabschnitt wird zum besseren Verständnis eine grobe Grundlage der Sicherheitsarchitektur für Internet-Protokolle gegeben.

1.2 Grundlage der Sicherheitsarchitektur für Internet-Protokolle

Die Sicherheitsarchitektur für Internet-Protokolle (IPsec) bietet die Sicherheit auf der IP-Schicht an, der von höheren Schichten verwendet werden kann. Die meisten Sicherheitsdienstleistungen werden durch zwei Sicherungsprotokolle, nämlich Authentication Header Protokoll (AH) und Encapsulating Security Payload Protokoll (ESP), sowie durch kryptografische Schlüsselverwaltung realisiert.

Das AH- und ESP-Protokoll werden verwendet, um verbindungslose Datenintegerität und Schutz gegen Wiederholungen zu bieten sowie die übertragenen Daten und Protokollinformation zu authentifizieren. Außerdem bietet das ESP-Protokoll die Datenvertraulichkeit. Wenn die Datenvertraulichkeit jedoch ohne Integrität oder/und Authentifizierung verwendet wird, wird sie wahrscheinlich durch aktive Angriffe unterlaufen. Deswegen steht sie immer als eine Option für Integerität und Authentifizierung zur Verfügung. AH und ESP können gemeinsam, alleine oder verschachtelt genutzt werden.

Die einfachste Form der Schlüsselverwaltung ist reine manuelle Konfiguration. Ein System kann durch eine Person mit Schlüsselmaterial und relevanten Verwaltungsdaten manuell konfiguriert werden, um die Kommunikation mit anderen Systemen zu sichern. Diese Technik ist geeignet für kleine und statische Umgebungen. Aber wenn die Anzahl der Rechner und/oder der Gateways steigt, skaliert die manuelle Technik nicht. Das Schlüsselaustausch-Protokoll (IKE) wurde für die Verwaltung und den Austausch der IPSec-Schlüssel entwickelt und ist ein dynamisches Schlüsselverwaltungsprotokoll. Zur Zeit steht IKE Version 1 als Standard zur Verfügung. Die IKE Version 2 (IKEv2) wurde eingerichtet, um die Version 1 zu ersetzen. Die beiden Versionen interoperieren nicht, aber sie haben genügend Kopf-Format gemeinsam, damit die beiden eindeutig über den selben UDP-Port laufen können[Kauf04]. MOBIKE soll basierend auf IKEv2 zu entwickelt werden. Deshalb konzentriert sich diese Ausarbeitung nur auf IKEv2.

2 Schlüsselaustausch-Protokoll Version 2 (IKEv2)

IKEv2 bietet Verfahren für die Authentifizierung zwischen zwei IPsec-Teilnehmern und Generierung der gemeinsam genutzten Schlüsseln. Die Authentifizierung kann zwischen zwei Rechnern, zwischen zwei Gateways, oder zwischen einem Rechner und einem Gateway ausgeführt werden. Abbildung 1 illustriert das grundlegende Konzept des Protokolls IKEv2. Um die Kommunikation zwischen zwei IPsec-Teilnehmern zu sichern müssen mehrere Parameter ausgetauscht werden. Alle diese Parameter, die in der Sicherheitsbeziehung des IKEv2s beschrieben werden, schreiben das Schlüsselaustauschverfahren, das Authentifizierungsverfahren, das Entschlüsselungsverfahren, den Algorithmus, den Schlüssel und dessen Gültigkeitsdauer vor.

Abbildung 1: Sclüsselaustausch-Protokoll version 2

Die Nachrichten von IKEv2, die Sicherheitsparameter enthalten, werden paarweise zwischen IPsec-Teilnehmern ausgetauscht. Jeder IKEv2-Austausch besteht aus einer Anfrage und einer Antwort. Abbildung 2 zeigt Ablauf des Protokolls IKEv2.

In der Anfangsphase gibt es zwei IKE-Austausche, die in [Kauf04] als IKE_SA_INIT und IKE_AUTH bezeichnet werden. Das erste Paar Nachrichten IKE_SA_INIT sind für das Aushandeln der kryptografischen Algorithmen, Nonce Austausch und einen Diffie-Hellman-Austausch verantwortlich. Das zweite Paar Nachrichten IKE_AUTH authentifizieren die vorherigen Nachrichten, tauscht Identitäten und Zertifikate aus und erzeugen die erste Sicherheitsbeziehung von IPsec.

Der erste Austausch, nämlich IKE_SA_INIT, muß in allen Fälle vor den anderen IKE-Austauschen komplett durchgeführt werden, danach dürfen die zwei Nachrichten IKE_AUTH ausgetauscht werden. Wenn diese zwei Austausche durchgeführt werden, d.h. ein gesicherter Kanal aufgebaut worden ist, dann können die übrigen IKE-Austausche in beliebiger Reihenfolge ausgeführt werden. Nachdem die Sicherheitsbeziehung des IKEv2s sich etabliert hat, können die relevanten Sicherheitsbeziehungen von IPsec auch aufgebaut werden.

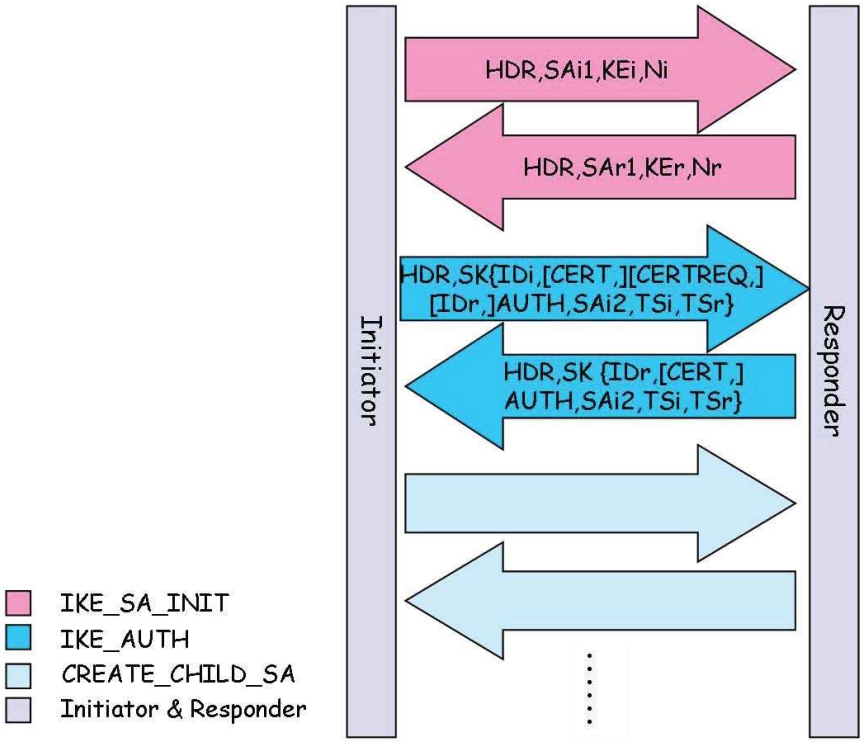


Abbildung 2: Ablauf des Protokolls IKEv2

3 MOBIKE Protokoll

Dank der drahtloser Übertragungstechnik kann man im Alltag mittels Laptop, PDA oder Handy in Verbindung zum Internet stehen. Allerdings unterstützen IKEv2- und IPsec-Protokolle solche mobile Nutzungen unzureichend oder nicht.

Es ist zu bemerken, daß MOBIKE WG [1] sich entschieden hat, daß das derzeitige MOBIKE nur den Tunnel-Modus betrachtet. Der Transport-Modus wird in [Dupo04b] geschrieben. In dieser Ausarbeitung wird nur der Transport-Modus behandelt.

Wie schon betrachtet, wird sowohl die Sicherheitsbeziehung des IKEv2s als auch die Sicherheitsbeziehungen von IPsec zwischen 2 IP-Instanzen hergestellt. Nachdem die Sicherheitsbeziehungen zwischen zwei IPsec-Teilnehmern entstanden sind, werden das einzelne Paar Adressen, nämlich die Quelleadresse und Zieladresse im äußeren IP-Kopf für den Tunnel Modus, eindeutig festgelegt. Es ist nicht möglich, die beiden Adressen später zu ändern. Aus diesem Grund wird ein neues Protokoll erforderlich, um die Aktualisierung der IP-Adresse für Sicherheitsbeziehung des IKEv2s und Sicherheitsbeziehungen von IPsec durchzuführen. Das MOBIKE ist so eine Erweiterung von IKEv2 Protokoll, die Mobilität unterstützt.

[1] http://www.ietf.org/html.charters/mobike-charter.html

3.1 Szenario für MOBIKE

In folgendem Unterabschnitt werden zwei typische Szenarien für MOBIKE vorgestellt.

3.1.1 Mobilität Szenario

Wie in Abbildung 3 dargestellt ist, erhält ein Laptop beispielsweise eine Verbindung zum Sicherheitsgateway per WLAN. Durch diesen Sicherheitsgateway wird der Laptop mit Internet verbunden. Wenn der Laptop bewegt wird, kann er zu einem anderen Netzwerk gelangen und eine neue IP-Adresse von diesem Netzwerk bekommen. Der Pfad, durch den der erste IKE-Austausch stattfindet, hat sich geändert. Der Benutzer will die VPN-Verbindung trotz der Änderung der IP-Adresse erhalten, ohne den IKEv2-Austausch zu wiederholen.

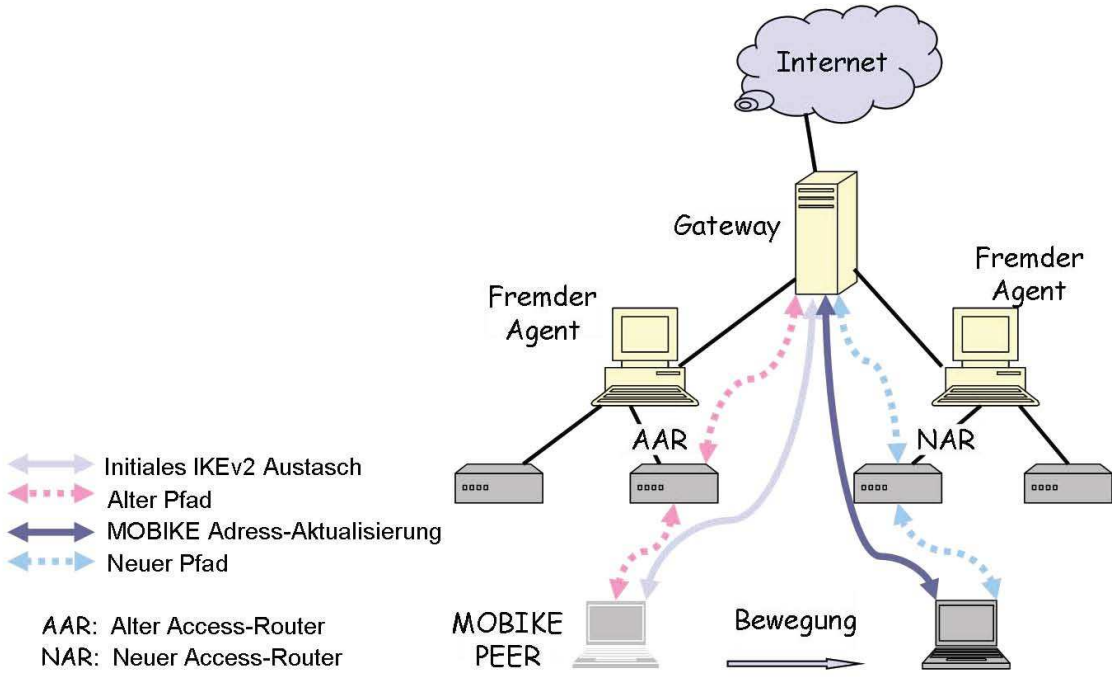


Abbildung 3: Mobilität Szenario

3.1.2 Multihoming Szenario

Multihoming Unterstüzung für einen Laptop bedeutet, daß mehrere Adresse anstatt einer einzelnen Adresse integriert werden können. In zweitem Szenario wird der Laptop in Betracht gezogen, der auf mehrere Arten mit dem Internet in Verbindung stehen kann. Der Zugang könnt Ethernet, WLAN und GPRS sein, die in verschiedenen Zeiten verwendet werden können. Der Laptop-Benutzer will die ganze Zeit die effizienteste Verbindung haben, aber jene Verbindung sich ändern könnte. Wenn der Laptop-Benutzer unterwegs ist, verwendet er beispielsweise GPRS; Wenn er zuhause oder im Büro ist, wechselt er zu WLAN. Der Benutzer will einfach die VPN-Verbindung erhalten und die Anwendungen entdecken diese Änderungen überhaupt nicht.

3.2 Schlüsselerneuerung in IKEv2

Die Sicherheitsbeziehung des Protokolls IKEv2 verwendet Schlüssel, die nur für begrenzte Lebenszeit benutzt werden sollen, um einen begrenzten Betrag der Datenmenge zu schützen. Wenn die Lebenszeit der Schlüssel abgelaufen ist oder die zu schützende Datenmenge den Maximalbetrag überschreitet, müssen neue Schlüssel ausgehandelt werden. Diese Vorgangsweise wird in IKEv2 als Schlüsselerneuerung bezeichnet. Bei der Schlüsselerneuerung wird die neue Sicherheitsbeziehung des IKEv2s mittels der vorhandenen alten Sicherheitsbeziehung des IKEv2s hergestellt.

Das Multihoming Szenario in 3.1.2 ist das sogenannt "make-before-break" Szenario. In diesem Fall können sowohl die Schlüsselerneuerung als auch das MOBIKE Protokoll zum Einsatz kommen, weil der Laptop in diesem Fall eine neue IP-Adresse bekommen kann, bevor die Erreichbarkeit der alten IP-Adresse verloren geht.

3.3 MOBIKE-Unterstützung für Mobilität

Im Mobilität Szenario in 3.1.1 ist es jedoch nicht möglich die Schlüsselerneuerung einzusetzen. Sie ist zu aufwendig für dieses Szenario, weil die IP-Adresse sich oft und schnell ändert. Da soll MOBIKE zum Einsatz kommen. Im diesem Fall handelt es sich möglicherweise um das sogenannt "break-before-make" Szenario. Allerdings ist es nicht das Ziel des MOBIKEs, dieses "break-before-make" Szenario zu unterstützen, weil MOBIKE nicht als Komplettlösung für Mobilität entwickelt wird. MOBIKE konzentriert sich nur auf die traditionelle VPN-Anwendung. Es wird in [KiTs04] angenommen, dass MOBIKE nur eingesetzt werden kann, wo das MOBIKE Peer eine neue IP-Adresse bekommt, während die alte IP-Adresse von ihm noch erreichbar ist. In folgendem Abschnitt wird MOBIKE basierend auf dem Mobilität Szenario vorgestellt.

4 MOBIKE-basiertes Mobilitätsmanagement

4.1 Terminologien

Im MOBIKE Protokoll werden viele relevante Terminologien eingeführt. Hier werden nur die nötigen Begriffe für diese Ausarbeitung eingewiesen. Ausführliche Informationen können in [KiTs04] gefunden werden.

- Peer: Peers werden als IKEv2-Knoten im Netzwerk bezeichnet. Durch ein Peer werden MOBIKE sowie IKEv2 implementiert.
- Peer-Adresse Menge: Wenn ein Peer A mit einem anderen Peer B im Netzwerk das MOBIKE-Protokoll einsetzen will, muß eine Peer-Adresse Menge für das Peer A definiert werden. Eine Peer-Adresse Menge besteht aus IP-Adressen, die für das Peer A gültig sind. Die Peer-Adresse Menge wird von A zu B gesendet und steht für B zur Verfügung.
- Bevorzugte Adresse: Mit dieser Adresse will ein Peer die MOBIKE-Nachrichten und durch IPsec gesicherte Daten erhalten. Ein Peer darf in einem bestimmten Zeitpunkt nur eine einzige aktive Bevorzugte Adresse haben. Hier wird die kurze Zeitspanne, in der die alte bevorzugte Adresse zu neuen bevorzugten Adresse wechselt, ignoriert.

4.2 Ablauf des Protokolls MOBIKE

Es wird angenommen, daß eine gesicherte Verbindung zwischen zwei Peers A und B im Netzwerk schon durch IKEv2 aufgebaut wurde und die beiden Peers diese Verbindung weiterhin für sich behalten wollen. Das MOBIKE Protokoll startet, sobald ein Peer seine IP-Adresse ändert.

Hier soll es betont werden, dass wenn beide betroffene Peers im Netz MOBIKE unterstützen, das MOBIKE-Protokoll normal laufen soll. Wenn eins dieses Protokoll nicht unterstützt, sollen die beiden Peers MOBIKE ignorieren und sich einfach mit Schlüsselerneuerung in IKEv2 authentifizieren.

- Benachrichtigung über die Änderung der IP-Adresse

  Zunächst muß die Änderung der IP-Adresse bekanntgemacht werden. Das Peer A, als Laptop im Szenario erwähnt, ändert seine IP-Adresse. Es gibt zwei Möglichkeiten, diese änderung zu benachtichten. Eine Möglichkeit ist, daß das Peer A eine authentifizierte Nachrichten direkt sendet, bevor es seine bevorzugte Adresse ändert. Die andere Möglichkeit ist, daß das Peer B durch die Beobachtung der Netzwerkumgebung indirekt über die Änderung von A benachrichtigt wird. Beispielsweise könnte die indirekte Benachrichtigung eine ICMP-Nachrichten, Information über einen Verbindungsausfall oder Änderung der Quelladresse für das erhaltene Datenpaket sein. Wenn solche Hinweise vorkommen, könnte das MOBIKE Peer B sich für einen Dead-Peer-Detection Austausch entscheiden. Wenn es dadurch merken könnte, daß das Peer A eine andere bevorzugte Adresse als früher hat, könnte es eine Authentifizierung mit diese Adresse durchführen.

- Aktualisieren der IP-Adresse der Sicherheitsbeziehung des IKEv2s

  Basierend auf Informationen über Wechsel der bevorzugten Adresse für Peer A kann die Adresse der Sicherheitsbeziehung des IKEv2s Aktualisiert werden.

  Bei direkter Benachrichtigung kann die neue bevorzugte Adresse für das Peer A in der zugesendeten Nachrichten enthalten sein. In dem fall kann sich das Peer B über die neue bevorzugte Adresse von A einfach aus der erhaltenen Nachrichten informieren.

  Bei indirekter Benachrichtigung könnte das Peer B einen Dead-Peer-Detection Austausch einsetzen. Der Dead-Peer-Detection kann gleichzeitig von Peer B ausgeführt werden, d.h. Peer B sendet gleichzeitig Datenpakete zu allen Adressen, die sich in Peer-Adresse Menge von A befinden. Peer A sendet eine einzige Antwort zurück, sobald es das erste Datenpaket von B erhält. Dadurch kann Peer B die neue bevorzugte Adresse von A entdecken.

  Wenn der Dead-Peer-Detection Austausch scheitert, d.h. das Peer B bekommt keine Antwort trotz mehrerer Versuche, wird Sicherheitsbeziehung des IKEv2s zwischen beiden Peers gelöscht. Folglich werden die relevanten Sicherheitsbeziehungen von IPsec auch gelöscht.

- Wechseln zu neuer IP-Adresse

  Es ist nicht nötig für Peer A, der Laptop im Szenario, die neue bevorzugte Adresse von B zu authentifizieren, wenn B eine unveränderte Peer-Adresse Menge hat, z.B. wenn B ist ein Sicherheitsgateway ist. In der Tat wurden alle Adresse von B schon während der Initialisierungsphase der Sicherheitsbeziehung des IKEv2s authentifiziert.

  In anderen Fällen sollte ein Konnectivitätstest durchgefürht werden, bevor die Adresse verwendet wird. Daher kann es sichergestellt werden, daß das Peer an der angezeigten Adresse erreichbar ist.

- Aktualisieren der Adresse für die Sicherheitsbeziehungen von IPsec

  Die Änderung der bevorzugten Adresse hat beim Tunnel Modus jedoch einen Einfluß auf Sicherheitsbeziehungen von IPsec. Der äußere Kopf, in dem die Quelle- und Zieladresse enthalten sind, muß modifiziert werden. Die Sicherheitsbeziehungen von IPsec sollen das neue Paar Adressen verwenden, zwischen den das MOBIKE signalisiert, damit die von IPsec gesicherten Daten richtig gemäß des Pfades von MOBIKE weitergeleitet werden können.

4.3 Verwaltung der IP-Adresse

Die Aufgabe des MOBIKEs ist Aktualisierung der IP-Adresse, die mit der Sicherheitsbeziehung des IKEv2s und der Sicherheitsbeziehungen von IPsec verbunden sind, ohne IKEv2 neu zu starten oder Schlüsselerneuerung in IKEv2 einzusetzen. Die Verwaltung der IP-Adresse spielt dafür eine wichtige Rolle und wird im folgenden Unterabschnitt in 2 Aspekten beschrieben werden.

4.3.1 Verwaltung der IP-Adresse bzgl. IPv6/ IPv4

In diesem Unterabschnitt wird es angenommen, dass NAT-T bei IPv4 vermieden werden soll. Aus Sicht von IPv6 bzw. IPv4 hat ein mobiles Peer eine Heimat-Adresse und eine Care-of-Adresse [Zitt04]. Unter der Heimat-Adresse kann ein mobiles Peer identifiziert werden. Falls sich ein Mobiles Peer in einem Fremdnetz befindet, ist die Heimat-Adresse der Eingangspunkt für den Tunnel-Modus. Die Care-of-Adresse stellt im Gegensatz die aktuelle Lokation eines mobilen Peers dar und ist der aktuelle gültige Endpunkt für den Tunnel-Modus.

Wie in [KiTs04] erklärt wird, unterstützt das bisherige MOBIKE-Protokoll nur den Tunnel-Modus. Die bevorzugte Adresse in MOBIKE ist die sogenannte Care-of-Adresse des Peers. In dieser Ausarbeitung sind die beide Begriffe äquivalent. MOBIKE soll, wie in Mobilität Szenario erklärt wurde, nur Kommunikation zwischen dem MOBIKE-Peer und dem Sicherheitsgateway verwalten. Aber es ist auch sinnvoll, dass die Kommunikation zwischen der bevorzugten Adresse und der Heimat-Adresse in der Betracht gezogen werden kann. Während MOBIKE und IKEv2 mit der bevorzugte Adresse funktionieren, können die relevanten Sicherheitsbeziehungen von IPsec die Heimat-Adresse des Peers als “Traffic Selector“ verwenden, um seine Mobilitätssignalisierung zu sichern. MOBIKE muss jedoch nicht für jedes Handoff solche Sicherheitsbeziehungen von IPsec aktualisieren, sondern soll hier Mechanismus besser entwickelt werden.

MOBIKE wird nicht als eine Komplettlösung für Mobilität entwickelt und unterstützt die gleichzeitige Bewegung nicht [KiTs04]. Dieses Problem kann durch bessere Verwaltung der Adresse gelöst werden. Gleichzeitige Bewegung bedeutet, dass beide mobilen Peers gleichzeitig ihre Adressen ändern, z.B in “break-before-make“ Szenario. Sie können da nicht mehr miteinander kommunizieren, weil sie ihre Kommunikationspartner nicht mehr im Netz finden können. In der Tat können die beiden mobilen Peers ihre Heimat-Adressen als eine authentifizierte Adresse in Peer-Adresse Menge einfügen, dann haben die beiden die Gelegenheit, den Kommunikationspartner wieder zu finden. Der Home-Agent wird dazu als ein Peer mit stabiler Infrastruktur genutzt.

Um die IP-Adresse zu verwalten, müssen noch die Peer-Adresse-Anzeigen für MOBIKE eingerichtet werden. In den Anzeigen sollen folgende Parametern enthalten sein,

- Die maximale Anzahl der gespeicherten IP-Adresse muss beschränkt sein.

- Die Art der IP-Adresse kann gezeigt werden, nämlich IPv4 bzw. IPv6.
- Die einzige bevorzugte Adresse muss markiert werden können.
- Wenn die Anzahl der Adresse(n) in einer Peer-Adresse Menge mehr als 1 ist, sollen die allen geordnet werden.
- 3 Operationen zu den Adressen sollen unterstützt werden, nämlich “UPDATE“, “ADD“ und “DEL“. Die bevorzugte Adresse eines Peers kann durch eine neu Adresse ersetzt werden. Neue Adressen können in die Peer-Adresse Menge eingefügt werden bzw. alte Adresse können gelöscht werden.
- Wenn eine der 3 Operationen durchführt, soll ein spezifischer String als “Flag“ eingesetzt werden, um zu zeigen, welche Operation durchgeführt wird.
- Sobald eine Peer-Adresse Menge sich ändert, soll diese Peer-Adresse Menge aktualisiert und neu gesendet werden.

4.3.2 Zusammenarbeiten mit NAT-T

Wenn ein mobiles Peer IPv4 benutzt, ist es sehr wahrscheinlich, dass es hinter dem NAT steht. Hier wird diese Situation spezifisch behandelt. Das in IKEv2 beschriebene NAT-T(Network Address Translation Traversal) wird dazu verwendet, um Rechnern eines privaten Netzes einen gemeinsamen Zugang zum Internet zu ermöglichen. Der Vorteil liegt darin, dass die Rechnern, die innerhalb eines privaten Netzes miteinander kommunizieren, nur ein Eingang zum Internet haben.

Der Charter von MOBIKE WG[2] erlaubt keine Änderung von NAT-T. Wenn MOBIKE NAT-T nicht ändert, sondern unterschützen könnte, wäre es sehr sinnvoll. MOBIKE will eine oder mehrere Adressen in Peer-Adresse Menge sicher austauschen und eine bevorzugte Adresse sicher festlegen. Aber mit NAT-T hat ein MOBIKE Peer überhaupt keine Kontrolle mehr auf seinen Adressen. Wenn kein anderes Protokoll dem MOBIKE Peer hilft, die von NAT-T zugewiesenen IP-Adresse und Port Information wieder zu finden, ist es dann nicht möglich Angriffe gegen das MOBIKE Peer zu vermeiden.

Es hat in [KiTs04] folgende Vorschläge gegeben,

- Ein anderes Protokoll unterschtützt das MOBIKE unter dieser Situation, z.B. MIDCOM (Middlebox Communications Protocol Requirements)[3].
- Ein neues Protokoll wird entwickelt, um das Netz hinter NAT zu untersuchen und sogar beteiligen zu können.
- NAT-T wird deaktiviert, wobei es angezeigt wird, daß Nutzung der Information von dem nicht authentifizierte Kopf niemals erlaubt.

Wenn NAT jedoch während der Sicherheitsbeziehung des IKEv2s Etablierung entdeckt wird, wird das MOBIKE automatisch deaktiviert. Das Peer soll NAT-T anwenden. Die Unterstützung für NAT-T ist sicher eine der wichtigsten Entwurfsentscheidungen mit einem Einfluss auf andere Protokollaspekte.

[2] http://www.ietf.org/html.charters/mobike-charter.html
[3] http://rfc3304.x42.com/

5 Diskussion über die Entwurfsentscheidung

5.1 Signalisierung der MOBIKE-Unterstützung

Wie schon erwähnt, müssen die zwei MOBIKE Peers müssen signalisieren können, ob sie das MOBIKE unterstützen.

Einerseits kann eine in IKEv2 beschriebene Vendor-ID-Payload während der Anfangsphase von IKE-Austausch angewendet werden. Ein spezifischer String zeigt das MOBIKE an und signalisiert die MOBIKE-Unterstützung. Andererseits kann eine in IKEv2 beschriebene Notification-Payload auch durch einen spezifischen String genutzt werden. Beide Payloads sind aus der technischen Sicht äquivalent und schon in IKv2 vorhanden, deshalb sind die beiden gute Alternativen.

Die MOBIKE WG hat vor kurzem für die Notifikation Nutzlast entschieden.

5.2 Dead-Peer-Detection in IKEv2

Der Dead-Peer-Detection Mechanismus wurde so entwickelt, daß ein Peer A zu dem anderen Peer B ein Leer-Information-Austausch Datenpaket sendet. Peer B will mit einer Bestätigung antworten, wenn es das Datenpaket erhalten kann. Wenn Peer A nach gewißer Zeit trotz mehrerer Versuche keine Bestätigung bekommt, dann ist Peer B für A nicht erreichbar.

Bei dem Protokollablauf wird schon auf den Dead-Peer-Detection Mechanismus hingewiesen. Durch Dead-Peer-Detection kann die neue aktive bevorzugte Adresse eines Peers informiert werden. Hier wird die neue aktive bevorzugte Adresse durch das erste erhaltene Datenpaket ausgehandelt, obwohl diese möglicherweise nicht die meist bevorzugte Adresse ist. Der Grund dafür ist, daß solche Aushandlung infolge der Implementierung das Problem von Vervielfachung der Datenpakete im Netzwerk vermeiden kann.

5.3 Konnektivitätstest

Der Rück-Routbarkeit-Test untersucht die Konnektivität zwischen zwei MOBIKE-Peers. In der Tat ist das Basis Format von Rück-Routbarkeit-Test ähnlich wie das Dead-Peer-Detection's Format und der Dead-Peer-Detection könnte für den MOBIKE Rück-Routbarkeit-Test geeignet sein. Aber es ist zu bemerken, daß die Sicherheitsbeziehung des IKEv2s durch Spezifikation in IKEv2 gelöscht werden kann, wenn der Dead-Peer-Detection scheitert. Daher sollen wir doch einige Abänderung hinzufügen.

Hier muss festgelegt werden, wann Rück-Routbarkeit-Test gestartet werden soll.

- Die erste Alternative ist, daß der Rück-Routbarkeit-Test regelmäßig durchgeführt wird.
- Jedes Mal, wenn die neue IP-Adresse genutzt wird, wird Rück-Routbarkeit-Test starten.
- Die in draft-dupont-mipv6-3bombing [Dupo05] eingeführte Methode wird angewendet, d.h. das Peer kann nur von einer anderen Adresse anstatt der bevorzugten Adresse die Aktualisierung senden, dann führt es den Rück-Routbarkeit-Test nicht aus.
- Wenn das Peer durch indirekte Benachrichtigung über Änderung der IP-Adresse informiert, dann tut es den Rück-Routbarkeit-Test nicht.
- MOBIKE Rück-Routbarkeit-Test wird ignoriert.

5.4 Vorschläge beim Aktualisieren der Adresse der Sicherheitsbeziehungen von IPsec

In [KiTs04] werden zwei Vorschläge dafür als Alternative gegeben.

Eine Alternative ist, daß die zusammenhängenden Sicherheitsbeziehungen von IPsec automatisch mit der Sicherheitsbeziehung des IKEv2s zu der neuen Adresse weitergeleitet werden, sobald die Adresse der Sicherheitsbeziehung des IKEv2s wechselt. Dafür muss nur der Pfad, durch den Sicherheitsbeziehung des IKEv2s die Sicherheitsbeziehungen von IPsec etablierte, behalten werden. Die neue IP-Adresse kann einfach geholt werden. Die Sicherheitsbeziehungen von IPsec können auch alternativ separat gewechselt werden. In dem Fall ist ein Format effizienter als Notification-Payload notwendig, weil ein Notification-Payload nur ein Sicherheitsparameter-Index pro Payload abspeichern kann. Wenn das separate Payload als eine Entwurfsentscheidung ausgewählt wird, könnte solches Payload sehr groß sein, weil es möglich viele Sicherheitsbeziehungen von IPsec geben könnte, es sei denn, daß der Wert von Sicherheitsparameter-Index unterstützt werden sollte.

5.5 Sicherheitsberücksichtigung

Weil alle MOBIKE Nachrichten durch IKEv2 authentifiziert werden, ist es nicht möglich, daß ein Angreifer das Datenpaket modifiziert. Die IP-Adresse in IP-Kopf wird jedoch nicht authentifiziert. MOBIKE muß darauf beachten, daß Änderung der IP-Adresse andere Änderung nicht direkt verursachen darf.

Wenn ein MOBIKE Peer über die Änderung der IP-Adresse von dem anderen Peer informieren will, könnte ein Angreifer eine verfälschte ICMP Nachrichten erzeugen, wobei die Gefahr und Schwierigkeit für MOBIKE Peer entstehen könnten.

Literatur

[Dupo04a] F. Dupont. Address Management for IKE version 2. draft-dupont-ikev2-addrmgmt-06, Oktober 2004. Work In Progress.

[Dupo04b] F. Dupont. IPsec transport mode in Mobike environments. draft-dupont-mobike-transport-01, Oktober 2004. Work In Progress.

[Dupo05] F. Dupont. A note about 3rd party bombing in Mobile IPv6. draft-dupont-mipv6-3bombing-01, Januar 2005. Work In Progress.

[Eron04] P. Eronen. Mobility Protocol Options for IKEv2 (MOPO-IKE). draft-eronen-mobike-mopo-01, Oktober 2004. Work In Progress.

[IETF04] IETF. IKEv2 Mobility and Multihoming (Mailing Lists). IETF MOBIKE WG charter, 2004.

[Kauf04] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. IETF draft-ietf-ipsec-ikev2-17, September 2004. Work In Progress.

[KeAt98a] S. Kent und R. Atkinson. IP Authentication Header. IETF RFC 2402, November 1998. Standards Track.

[KeAt98b] S. Kent und R. Atkinson. IP Encapsulating Security Payload (ESP). IETF RFC 2406, November 1998. Standards Track.

[KeSe98] S. Kent und K. Seo. Security Architecture for the Internet Protocol. IETF RFC 2401, November 1998. Standards Track.

[KiTs04] T. Kivinen und H. Tschofenig. Design of the MOBIKE Protocol. IETF draft-ietf-mobike-design-01, Dezember 2004. Work In Progress.

[TsEr04] Hannes Tschofenig und Pasi Eronen. Simple Mobility and Multihoming Extensions for IKEv2 (SMOBIKE). draft-eronen-mobike-simple-01, März 2004. expired.

[Zitt04] M. Zitterbart. Vorlesung Mobilkommunikation. www.tm.uka.de, Oktober 2004.

Abbildungsverzeichnis

1 Sclüsselaustausch-Protokoll version 2 142

2 Ablauf des Protokolls IKEv2 143

3 Mobilität Szenario 144

SCTP-basiertes Mobilitätsmanagement

Philip Hoyer

Kurzfassung

Durch die Zunahme der mobilen Kommunikation mit neuen Techniken entsteht auch eine Reihe von Problemen, die mit einem Mobilitätsmanagement gelöst werden. Das Mobilitätsmanagement regelt z.B. Handovers, so dass Kommunikationsverbindungen nicht abbrechen, wenn ein mobiler Knoten sich von einem Netz in ein Anderes bewegt. In dieser Ausarbeitung wird gezeigt, wie dies durch ein neues Transportprotokoll, das „Stream Control Transmission Protocol“ (SCTP) mit einer Erweiterung zum dynamischen Assoziieren von IP-Adressen als reine Ende-zu-Ende-Lösung gelöst wird. Dabei werden keine zusätzlichen Hardwarekompontenten benötigt. Außerdem werden die Neuerungen, die SCTP gegenüber bereits betagten Protokollen hat, wie TCP, kurz vorgestellt. Im weiteren Verlauf wird SCTP mit anderen mobilen Lösungen eingesetzt, um ein vollständiges Mobilitätsmanagement zu realisieren, und auch die Lokalisierung mobiler Knoten in Fremdnetzen zu ermöglichen.

1 Einleitung

Unter Mobilitätsmanagement versteht man die Bewältigung von Problemen, die durch die Mobilität der Kommunikationsendpunkte entstehen. Dabei sind in den Funknetzen der vierten Generation (4G networks) zwei Kriterien entscheidend: Handover Management und Location Management. Ersteres dient zur Aufrechterhaltung der bestehenden Verbindungen eines mobilen Knotens/MN beim Wechsel zwischen zwei (Funk-)Netzen. Letzteres dient zum Überwachen der Position des MN und seinen Positionswechseln, damit auch Verbindungen zum MN aufgebaut werden können, ohne dessen Aufenthaltsort zu kennen [KüHV04].

1.1 Handover Management

Ein MN bewegt sich frei in der Welt. Bei einem Netzwechsel, z.B. von WLAN zu UMTS, werden ohne Handover alle Kommunikationsverbindungen abgebrochen, da keine Weiterleitung auf das neue Netz erfolgt. Daher muss bei jedem Netzwechsel ein Handover, oder auch Handoff genannt, durchgeführt werden. Dabei werden die bestehenden Kommunikationsverbindungen des MN von einem Netz an ein neues Netz weitergeleitet. Ziel einer solchen Übergabe ist es, den Dienstausfall so gering wie möglich zu halten [KLRM04].

Eine Lösung ist z.B. Mobile IP (MIP). Bei MIP existieren eine Reihe von Varianten, um einen Handover durchzuführen, beispielsweise „Low Latency“ und „Fast handover“. Diese Lösungen basiert auf dem Tunneln der Verbindung zwischen alten und neuen Zugangsrouter. MIP setzt also die Unterstützung der bestehenden Netze voraus, insbesondere die der Netzwerkrouter. Im Gegensatz zu MIP ist ein SCTP-basierender Handover nicht auf solche Unterstützung angewiesen. Ein SCTP-basiertes Handover ist als reine Ende-zu-Ende-Lösung realisiert, die außer der mSCTP-Unterstützung der Anwendung keine weitere Modifikationen oder zusätzliche Netzwerkkomponenten erfordert [KoXi04].

1.2 Location Management

Ein weiterer Knoten, der mit dem MN Verbindung aufnehmen möchte, muss den genauen Aufenthaltsort des MN kennen, um diesen kontaktieren zu können. Daher beinhaltet Mobilitätsmanagement auch die Lokalisierung, also das Ausfindigmachen des MN, egal in welchem Netz dieser sich gerade aufhält.

MIP sieht hier die Benutzung von Agenten vor, wie den Home Agent (HA) und den Foreign Agent (FA, nur bei IPv4). Da SCTP auf dem Ende-zu-Ende-Konzept basiert (es sind keine dritten Komponenten beteiligt), können mit SCTP nur Handovers realisiert werden, keine Lokalisierungen. Daher ist SCTP bei vollständigem Mobilitätsmanagement noch mit einem weiteren Protokoll einzusetzen, welches das Location Management übernimmt, wie MIP und SIP. Solche Lösungen, die auf SCTP und einem weiteren Protokoll basieren, werden im weiteren Verlauf der Ausarbeitung vorgestellt.

2 Stream Control Transmission Protocol

Das „Stream Control Transmission Protocol“ (SCTP) ist ein Transportprotokoll. Entwickelt wurde SCTP von der IETF SIGTRANS Gruppe und im Oktober 2000 im „Request for Comments 2960“ [SXMS00] veröffentlicht.

Ursprünglich wurde SCTP entwickelt, um Zeichengabenachrichten (SS7) aus Telefonnetzen über IP-Netzwerke übertragen zu können. Für diese Zwecke fanden die Entwickler TCP nicht mehr ausreichend, da z.B. manche Anwendungen einen zuverlässigen Dienst benötigen, bei dem es aber auf die Reihenfolge der Auslieferung nicht ankommt. TCP bietet aber nur einen zuverlässigen und reihenfolgetreuen Dienst, UDP nur einen unzuverlässigen Dienst.

2.1 SCTP Grundlagen

SCTP ist das dritte standardisierte Transportprotokoll und arbeitet neben TCP und UDP auf Schicht 4 des OSI-Referenzmodells, über IP und unter den Anwendungsschichten. SCTP besitzt viele Gemeinsamkeiten mit TCP, so bietet auch SCTP eine zuverlässige, verbindungsorientierte Übertragung zwischen zwei Endpunkten (Ende-zu-Ende). Daneben gibt es eine Reihe von Neuerungen, die die Vorteile von TCP und UDP vereinen. Auch für die Nutzung in mobilen Netzen ist SCTP geeignet.

Übertragungen können immer nur zwischen zwei SCTP-Endpunkten stattfinden. Eine solche Verbindung wird Assoziation genannt, da SCTP ein erweitertertes Konzept einer Verbindung als TCP hat. Zum Aufbau einer Assoziation wird ein Vier-Wege-Handshake benutzt, um resistenter als TCP gegen „Denial of Service“-Attacken zu sein. Auch der Abbau einer Assoziation funktioniert in der Regel geordnet über einen Drei-Wege-Handshake. SCTP unterstützt keine halboffenen Verbindungen. Hat ein Endpunkt die Assoziation abgebaut, werden danach noch eintreffende Pakete verworfen.

Normalerweise arbeitet SCTP wie TCP reihenfolgetreu, d.h. beim Empfänger werden die Pakete in gleicher Reihenfolge an die obere Schicht gereicht, wie sie der Sender übermittelt hat. Die Reihenfolgetreue kann aber auch außer Kraft gesetzt werden, so dass eintreffende Pakete immer sofort an die Anwendungsschicht übergeben werden. Auf diese Weise kann SCTP auch einen reihenfolgelosen Dienst bieten, um z.B. Signalisierungen zu übertragen.

SCTP ist ein zuverlässiger Dienst. Pakete die verloren gehen, müssen erneut gesendet werden. Es gibt aber die „Partial Reliability“-Erweiterung, durch die die Zuverlässigkeit der Auslieferung von Paketen beeinflusst werden kann. In dieser Erweiterung kann über einen Parameter

der „reliability level“, der Grad der Zuverlässigkeit, gesteuert werden. Damit ist gemeint, unter welchen Bedingungen eine Wiederholung der Übertragung bei einem nicht oder fehlerhaften empfangenen Paket stattfindet. [SXMS00]

Damit kann SCTP also auch ein Datagram-ähnlicher Dienst realisieren, der ohne Bestätigungen und Wiederholungen von Paketen arbeitet. So können z.B. auch Echtzeitgespräche übertragen werden [SXMS00].

2.2 Paketaufbau

Ein SCTP-Paket besteht in der Regel aus einem allgemeinem Paketkopf und einem oder mehreren Chunks. Chunks sind Teilpakete in einem SCTP-Paket, die entweder Steuerungsinformationen oder Daten enthalten. Dazu gibt es neben dem Daten-Chunk DATA eine Reihe von Steuerungs-Chunks, wie INIT, INIT-ACK, COOKIE-ECHO und COOKIE-ACK zum Aufbau einer Verbindung und SHUTDOWN, SHUTDOWN-ACK und SHUTDOWN-COMPLETE zum Abbau einer Verbindung. Im SCTP-Standard sind 15 Chunk-Typen definiert, mit den derzeitigen Erweiterungen sind es 20 Chunks.

Der Paketkopf enthält die wichtigsten Daten, wie SCTP-Quell- und Ziel-Portnummer und eine Prüfsumme. Nach dem Kopf können sich ein oder mehrere Chunks anschließen. Bis auf ein paar Ausnahmen, so müssen z.B. INIT und SHUTDOWN-COMPLETE die einzigen Chunks in einem Paket sein, kann das Paket bis zur maximalen Größe (MTU) mit Chunks aufgefüllt werden. Die Daten, die ein Chunk neben den Nutzdaten, der Typenbezeichnung und seiner Länge enthält, variieren von Typ zu Typ. Einige der Chunktypen, wie INIT, sind nochmals in Parameter unterteilt.

Da ein Chunktyp mit 8 Bit definiert wird, kann es maximal 256 Chunk-Typen geben, so dass Erweiterungen für SCTP mit neuen Chunk-Typen definiert werden können. Eine solche Erweiterung ist z.B. die „Partial Reliability“-Erweiterung, die oben bereits erwähnt wurde oder die „Dynamic Adress Reconfiguration“-Erweiterung, die weiter unten in dieser Ausarbeitung ausführlich beschrieben wird.

SCTP-Endpunkte, die eine solche Erweiterung nicht kennen, können durch die zwei höchsten Bits des Chunk-Typs angewiesen werden, wie sie bei einem unbekannten Chunk-Typ verfahren sollen (z.B. unbekannte Chunks ignorieren). Dadurch ist es relativ einfach, SCTP-Erweiterungen in Umgebungen zu implementieren, die Erweiterungen nicht unterstützen.

2.3 Multistreaming

TCP arbeitet byteorientiert, d.h. es schickt nur eine Bytefolge von der Anwendungsschicht an den Empfänger. Möchte der Sender voneinander unabhängige Nachrichten senden, so muss die logische Trennung dieser Nachrichten aus der Bytefolge von der Anwendung übernommen werden. Werden ein oder mehrere TCP-Pakete nur fehlerhaft oder gar nicht empfangen, müssen diese Pakete erneut gesendet werden und die gesamte Übertragung aller nachfolgenden Nachrichten verzögert sich.

SCTP arbeitet im Gegensatz zu TCP nachrichtenorientiert. Es können mehrere, voneinander unabhängige Bytefolgen (Nachrichten) gesendet werden, die auf der Empfängerseite auch unabhängig behandelt werden. Eine solcher „Kanal“ für eine Nachricht wird bei SCTP mit dem etwas unglücklich gewählten Begriff „Stream“ bezeichnet. Die Reihenfolgentreue wird innerhalb eines Streams eingehalten (wenn gewünscht), Streamfolgen untereinander werden aber unabhängig davon an die Anwendungen geliefert. Das heißt bei fehlerhafter Übertragung eines Streams wird die Auslieferung dieser Nachricht durch die erneute Übertragung verzögert,

aber alle anderen Nachrichten, die auf anderen Streams versendet werden, sind davon nicht betroffen.

Die Anzahl der ein- und ausgehenden Streams wird beim Assoziationsaufbau zwischen den Endpunkten verhandelt. Der Daten-Chunk bietet dazu zum Übertragen zwei Felder. Mit dem „Stream Identifier“ wird der Stream eindeutig identifiziert. Mit der „Stream Sequence Number“ (SSN) wird die Reihenfolgetreue innerhalb des Streams sichergestellt. SCTP bietet auch über ein Flag die Möglichkeit, die Reihenfolgentreue für einzelne Streams außer Kraft zu setzten, wie bereits weiter oben erwähnt. Die SSN ist dann leer und wird ignoriert, empfangene Pakete werden immer sofort an die nächste Schicht durchgereicht.

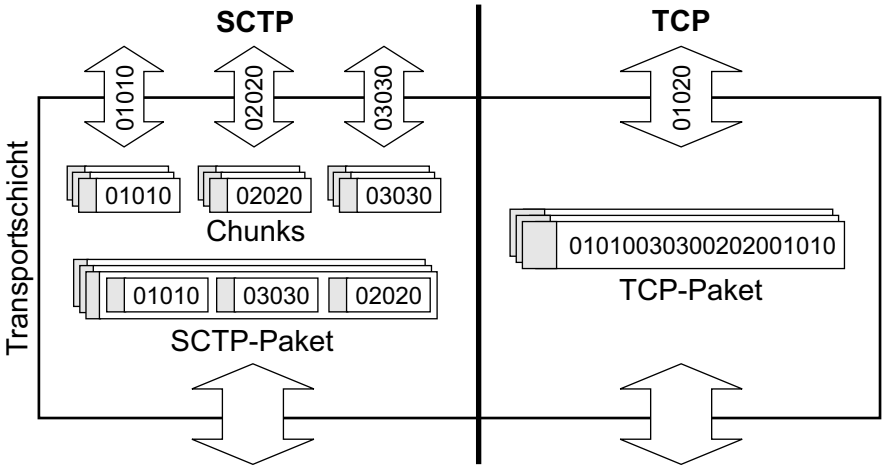


Abbildung 1: SCTP (Chunks und Multistreaming) vs. TCP

2.4 Multihoming

Die unter dem Gesichtspunkt des Mobilitätsmanagement wichtigste Eigenschaft von SCTP, ist „Multihoming“. Ein SCTP-Endpunkt ist dadurch nicht mehr nur unter einer IP-Adresse erreichbar, sondern unter beliebig Vielen, die Assoziazion besteht aber weiterhin nur aus zwei Endpunkten mit jeweils nur einer SCTP-Portnummer. Ein SCTP-Endpunkt besteht also aus einer Menge mit mindestens einer IP-Adressen und genau einer Portnummer. Multihoming ist eine Technik, um Ausfälle durch Leitungs- oder Übertragungsstörungen zu mindern.

Beim Assoziationsaufbau werden die verfügbaren IP-Adressen und die SCTP-Portnummer ausgetauscht, so dass ein Endpunkt die IP-Adressen der Gegenseite kennt und benutzen kann. Ob ein Endpunkt dabei alle, einige oder nur eine seiner verfügbaren IP-Adresse mitteilt, liegt an der entsprechenden Anwendung. Bei den IP-Adressen, die ein SCTP-Endpunkt nutzt, wird kein Unterschied zwischen IPv4 und IPv6 gemacht, die IP-Adressen können auch gemischt (IPv4 und IPv6) angebunden werden.

Jeder SCTP-Endpunkt besitzt genau eine primäre IP-Adresse, unabhängig wieviele IP-Adressen dieser angebunden hat. SCTP-Anwendungen sollen Pakete in der Regel über die primäre IP-Adresse senden und diese auch als Absenderadresse im IP-Paket angegeben. Ist die primäre IP-Adresse eines SCTP-Endpunktes gestört und somit nicht mehr erreichbar, werden die Pakete zu sekundäre IP-Adressen umgeleitet.

3 Mobile SCTP

In der SCTP Grundversion werden durch die Multihoming-Fähigkeit verfügbare IP-Adressen beim Aufbau der Assoziation der Gegenseite bekannt gemacht. Diese IP-Adressen können

dann für die Dauer der Assoziation nicht mehr geändert werden. Ein MN, der durch verschiedene Netze wechselt, erhält aber ständig neue IP-Adressen, während alte IP-Adressen ungültig werden. Wechselt der MN das Netz und erhält eine neue IP-Adresse, so wird die Assoziation abgebrochen, da die neue IP-Adresse erst wieder bei erneutem Assoziationsaufbau der Gegenseite mitgeteilt wird.

Deshalb wurde eine Erweiterung zu SCTP definiert, die die Multihoming-Fähigkeiten eines SCTP-Endpunkt ausbaut. Ein mobiler Knoten ist damit in der Lage, während einer aufgebauten Assoziation selbstständig seine eigenen IP-Adressen zu ändern, also IP-Adressen hinzufügen, zu löschen und die primäre IP-Adresse zu ändern und diese Änderungen dem CN mitzuteilen. Diese Erweiterung heißt „Dynamic Address Reconfiguration“ oder kurz ADDIP-Erweiterung und wurde im September 2003 als IETF Draft veröffentlicht. SCTP mit dieser Erweiterung ist definiert als Mobile SCTP oder kurz mSCTP [SRXT04].

3.1 mSCTP Chunktypen

Um eine dynamische Assoziation von IP-Adressen zu ermöglichen, werden bei der ADDIP-Erweiterung den bereits bestehenden SCTP-Chunk-Typen zwei Neue hinzugefügt.

- *„Address Configuration Change“-Chunk (ASCONF)* informiert die Gegenseite über eine Änderungen an den eigenen IP-Adressen. Eine solche Änderung kann das Hinzufügen einer neu erhaltenen IP-Adresse, das Löschen einer nicht mehr benötigten IP-Adresse oder das Wechseln der primären IP-Adresse sein. Dazu enthält der Chunk u.a. die von der Änderung betroffene IP-Adresse und den Parameter der Änderung. Die IP-Adresse kann sowohl eine IPv4-Adresse, als auch eine IPv6-Adresse sein.
- *„Address Configuration Acknowlege“-Chunk (ASCONF-ACK)* bestätigt dem Sender eines ASCONF-Chunks den Erhalt desselben und teilt ihm mit, ob die Änderung erfolgreich zur Kenntnis genommen wurde oder ein Fehler aufgetreten ist. Nach jedem ASCONF-Chunk muss zunächst der Sender erst die Bestätigung abwarten, bevor ein neuer ASCONF gesendet werden kann.

Die beiden neuen Chunk-Typen sind so definiert, das ein SCTP-Endpunkt, der ohne die ADDIP-Erweiterung arbeitet, diese für ihn unbekannten Chunk-Typen zwar in seiner Bearbeitung ignoriert, aber den Sender des unbekannten Chunks ein ERROR-Chunk zukommen lässt, indem er mitteilt dass der empfangene Chunk-Typ für ihn unbekannt ist.

3.2 mSCTP Parameter

Die beiden neuen Chunktypen ASCONF und ASCONF-ACK arbeiten, ähnlich wie der INIT und INIT-ACK-Chunk mit Parametern. Folgende Parameter sind zusätzlich definiert, die fast alle entweder in Chunks vom Typ ASCONF oder vom Typ ASCONF-ACK vorkommen dürfen [SRXT04].

- *IP-Adresse hinzufügen:* Teilt der Gegenseite mit, dass eine neue IP-Adresse zu den Eigenen hinzugefügt wird. Die neue IP-Adresse wird als sekundäre Adresse hinzugefügt. Dieser Parameter steht nur in Chunks vom Typ ASCONF.
- *IP-Adresse löschen:* Teilt der Gegenseite mit, dass eine eigene IP-Adresse gelöscht wird. Die zu löschende IP-Adresse muss als sekundäre Adresse vorliegen. Dieser Parameter steht ebenfalls nur in Chunks vom Typ ASCONF.

- *Primäre IP-Adresse festlegen:* Teilt der Gegenseite mit, dass eine eigene IP-Adresse primär wird. Da nur eine IP-Adresse primär sein kann, wird die alte primäre Adresse zur sekundären Adresse. Der Parameter steht in Chunks vom Typ ASCONF, aber auch in INIT und INIT-ACK, um bereits beim Assoziationsaufbau der Gegenseite die gewünschten Präferenzen mitzuteilen.
- *Erfolgsmitteilung:* Teilt dem Sender eines Chunks vom Typ ASCONF mit, dass die dortigen Parameter erfolgreich zur Kenntnis genommen wurde. Dieser Parameter steht nur in Chunks vom Typ ASCONF-ACK.
- *Fehlermitteilung:* Teilt dem Sender einer ASCONF mit, dass bei der Ausführung der Parameter ein Fehler aufgetreten ist, z.B. wenn versucht wird, die letzte verbleibende IP-Adresse zu löschen. Dieser Parameter steht nur in Chunks vom Typ ASCONF-ACK.

Ausserdem ist noch ein sechster Parameter definiert, der aber nur dazu dient, Mitteilungen an die oberen Schichten durchzureichen. Dieser Parameter darf nur in INIT und INIT-ACK Chunks vorkommen.

4 Handover Management

Mit Hilfe der ADDIP-Erweiterung ist mSCTP in der Lage ein Handover durchzuführen, welches auf reiner Ende-zu-Ende-Kommunikation basiert. Dadurch erfordert der Handover keine weitere Unterstützung der bereits bestehenden Hardware (z.B. der Zugangsrouter) oder die Anschaffung neuer Hardwarekomponenten (z.B. Home Agent bei Mobile IP). Die einzige Voraussetzung ist die Unterstützung von mSCTP bei den Anwendungen der beiden Endpunkte. Damit kann der MN mit ausschließlichem Einsatz von mSCTP eine unterbrechungsfreie Kommunikation zu einem CN aufbauen, die Umkehrung ist aber mangels Location Management nicht möglich [KLRM04].

4.1 Beispiel eines mSCTP Handovers

Im Folgenden ist beschrieben, welche Schritte bei einem mSCTP Handover durchgeführt werden. Als Beispiel dient dazu ein MN, der sich zwischen zwei Netzen bewegt. Der MN startet im Netz A und bewegt sich zu Netz B. Dabei hält er eine Assoziation mit einem korrespondierenden Endpunkt (CN). In den meisten Fällen läuft der Zugang zum Netz über einen Zugangsrouter (AR, engl. Access Router), so auch in diesem Beispiel. Bis auf die Anforderung einer neuen IP-Adresse bei Eintritt in ein Netz, macht es keinen Unterschied, ob dieses Netz auf IPv4 oder IPv6 basiert. Im Beispiel arbeitet das Netz A mit IPv4, das Netz B mit IPv6 (siehe Abbildung 2).

Die folgenden vier Schritte werden durchlaufen, sobald sich der MN in Reichweite des Netzes B befindet. Solange die Assoziation vom MN zum CN andauert, werden diese Schritte immer abgearbeitet, wenn ein neues Netze in Reichweite ist. Der Handover-Prozess muss vom MN angestoßen werden, da nur er die Bewegungen kennt und neue Netze entdecken kann.

1. *Anfordern einer neuen IP-Adresse:* Die untersten Schichten des MN informieren den MN, sobald dieser sich in Reichweite des Netzes B befindet. Der MN fordert von dem Zugangsrouter des Netzes eine IP-Adresse an. In der Regel geschieht dies in IPv4-Netzen über DHCP und in IPv6-Netzen über DHCPv6 oder über die „Stateless Address Autoconfiguration“. Die neue IP-Adresse des MN muss SCTP bekannt gemacht werden, also bis zur Transportschicht gereicht werden. Im Beispiel erhält der MN, sobald er sich in Reichweite des Netzes B befindet die IPv6-Adresse 2ffe:ffff:f202::2.

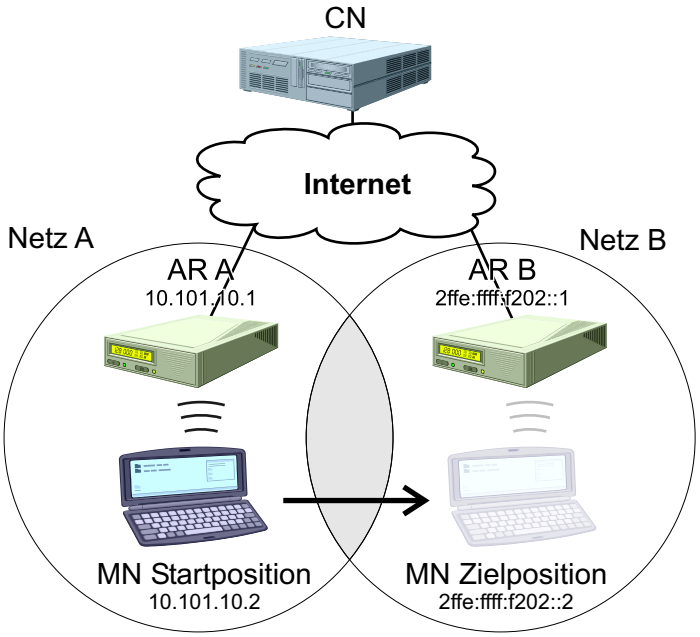


Abbildung 2: mSCTP Handover

2. *Hinzufügen der IP-Adresse zur Assoziation:* Der MN teilt dem CN über einem Chunk des Typs ASCONF und einem darin enthaltenen Parameter vom Typ „IP-Adresse hinzufügen“ mit, das er die IPv6-Adresse 2ffe:ffff:f202::2 vom Netz B erhalten hat und diese als eine sekundäre Adresse zu seiner Address-Liste hinzugefügt hat. Diese kann dann vom CN genutzt werden. Die IP-Adresse muss aber nicht sofort zu der Assoziation hinzugefügt werden, sondern kann auch von Bedingungen aus anderen Schichten abhängig gemacht werden. Der CN antwortet mit einem Chunk vom Typ ASCONF-ACK und bestätigt so den Erhalt des ASCONF-Chunks. Erst wenn der MN diese Bestätigung vom CN bekommen hat, kann der MN die neue IP-Adresse auch als Quelladresse nutzen.

3. *Ändern der primären IP-Adresse:* Da sich beide Netze überlappen, ist der MN „multihomed“ und in zwei unterschiedlichen Netzen erreichbar, wenn er sich in dieser Region aufhält. Diese Region ist in Abbildung 2 grau markiert. Der MN und der CN senden in der Regel noch über die IP-Adresse 10.101.10.2 des Netzes A, da diese IP-Adresse noch primär ist. Daher muss, wenn der MN sich weiter zu Netz B bewegt, die primäre IP-Adresse ändern. Dazu sendet der MC wieder ein ASCONF mit dem Parametertyp zum ändern der primären IP-Adresse und die neue primäre IP-Adresse 2ffe:ffff:f202::2. Den Erhalt und die Änderung muss wiederum vom CN mit ASCONF-ACK bestätigt werden.

4. *Löschen der alten IP-Adresse von der Assoziation:* Wenn der MN die Reichweite des Netzes A verlässt, wird die IP-Adresse 10.101.10.2 inaktiv und muss aus der Adressen-Liste des MN entfernt werden. Wann der MN eine IP-Adresse als inaktiv betrachtet ist nicht klar definiert. Wenn die Verbindung zum Netz A abbricht, werden Pakete, die dann verloren gehen von mSCTP in das Netz B umgeleitet (basic rerouting). Es könnten aber auch Informationen der physischen Schicht genutzt werden (z.B. die Signalstärke in einem Funknetz), so dass es erst gar nicht zu einer Unterbrechung der Assoziation kommt. Das Löschen einer IP-Adresse wird dem CN mit entsprechendem Parameter in einem ASCONF-Chunk mitgeteilt. Auch das Löschen einer IP-Adresse muss vom CN mit ASCONF-ACK bestätigt werden.

4.2 Ändern der primären IP-Adresse

Wann die Änderung der primären IP-Adresse durchgeführt wird, ist ein wichtiger Punkt beim mSCTP-Handover, der bei häufigen Netzwechseln entscheidenden Einfluss auf die Übertragungsrate und die Dienstgüte hat [KLRM04]. Die Änderung kann in Verbindung mit dem Löschen oder Hinzufügen einer IP-Adresse oder in einem eigenen ASCONF-Chunk erfolgen. Im Folgenden sind drei Möglichkeiten aufgeführt, wann eine Änderung der primären IP-Adresse stattfinden könnte.

- *Sobald eine neue IP-Adresse bekannt wird:* Nachdem die IP-Adresse zu der Address-Liste des MN hinzugefügt wurde, wird sie sofort zur primäre IP-Adresse. Diese Änderung ist bei einem sich schnell und linear bewegenden MN von Vorteil, da der Handover sehr schnell durchgeführt werden kann, ohne auf weitere Kriterien Rücksicht nehmen zu müssen. Bewegen sich der MN allerdings häufig zwischen zwei Netzen hin und her, ist diese Art des Wechsels eher nachteilig. Der Wechsel der IP-Adresse erfolgt unter Umständen vorschnell, wenn der MN nur kurz die Reichweite des Netzes streift und sich dann wieder entfernt.
- *Bei einem Signal von unteren Schichten:* Hier können Signale von Schichten unterhalb der Transportschicht ein Wechsel bewirken. Bestes Beispiel ist physische Schicht, die die Signalstärke bei Funknetzen (z.B. WLAN) an die Transportschicht meldet. Ab einer gewissen Signalstärke wird eine Änderung der primären IP-Adresse eingeleitet. Besonders bei Netzen gleicher Art (z.B. zwei unterschiedliche WLAN-Netze) ist diese Übergabe von Vorteil.
- *Bei einem Signal von oberen Schichten:* Eine Änderung der primären IP-Adresse kann auch von den Anwendungsschichten ausgelöst werden. So kann z.B. der MN aufgrund von vorher vom Benutzer festgelegten Prioritäten die IP-Adresse erst so spät bzw. so früh wie möglich ändern. Solche Prioritäten können aufgrund von Bandbreite, Verfügbarkeit, Kosten etc. entschieden werden. Das ist besonders bei vertikalen Handovers von Vorteil, da ein Netz meistens qualitativ höherwertig ist (z.B. WLAN und UMTS).

Untersuchungen mit mSCTP-Handovers in normalen WLANs haben gezeigt, dass ein aggresives Hinzufügen von IP-Adressen in Kombination mit einem eher konservativen Wechsel der IP-Adresse die besten Resultate bringt. Damit ist gemeint, dass eine IP-Adresse erst ab einem gewissen Schwellenwert der Signalstärke zu einem SCTP-Endpunkt hinzugefügt wird und dass der Schwellenwert für den Wechsel der primären IP-Adresse nur leicht höher liegt, als der Schwellenwert für das Hinzufügen [KoCL04].

Die Probleme, die MIPv6 mit Route Optimization hat, sind aufgrund des Fehlens eines HA auch bei mSCTP Handovers zu finden. Insbesondere sind das Sicherheitsprobleme, da zwischen MN und CN kein Vertrauensverhältnis besteht, sowie die Sichtbarkeit der momentanen IP-Adresse des MN im Fremdnetz und die damit verbundene Preisgabe des momentanen Aufenthaltsortes des MN [ZiWS04].

5 Location Management

mSCTP ist in erster Linie ein Protokoll, was auf dem klassischen Client-Server-Prinzip beruht. Der MN ist der Client, er baut die Assoziationen zum Kommunikationspartner auf, welcher der Server ist [KoXi04]. Soll die Kommunikation auch in umgekehrter Richtung funktionieren, so dass der Kommunikationspartner eine Assoziation zum mobilen Knoten aufbauen kann,

	mSCTP	**Mobile IP**	**SIP**
Protokoll-schicht	Transport	Netzwerk	Anwendung
Location Management	nein	ja	ja
Handover Management	ja	ja, mit Fast Handover MIP	ja, mit mSCTP
Netzwerk-unterstützung	nicht benötigt	benötigt	nicht benötigt
Spezielle Komponenten	nein	Home Agent, (Foreign Agent)	SIP Server

Tabelle 1: Protokolle für Mobilitätsmanagement [KLRM04]

so muss noch ein Protokoll eingesetzt werden, was die Lokalisierung des MN im Fremdnetz übernimmt, da der Partner die aktuelle(n) IP-Adresse(n) des MN nicht kennt.

Mögliche Protokolle, die ein Location Management in Verbindung mit mSCTP bieten, sind Mobile IP (MIP) oder SIP (Session Initiation Protocol). Insbesondere sind für die Lokalisierung noch zusätzliche Komponenten erforderlich, die mSCTP nicht bieten kann, da es eine reine Ende-zu-Ende-Lösung ist [KLRM04]. Tabelle 1 fasst die drei Protokolle mit ihren Möglichkeiten und Vorraussetzungen kurz zusammen.

5.1 mSCTP und Mobile IP

In diesem Szenario soll mSCTP zusammen mit Mobile IP (MIP) eingesetzt werden um ein vollständiges Mobilitätsmanagement zu realisieren. MIP arbeitet unter mSCTP auf der Netzwerkschicht. Obwohl MIP Location- und Handover Management bietet, soll MIP hier nur für das Location Management zuständig sein, während Handovers mit mSCTP wie unter Punkt 4 beschrieben durchgeführt werden.

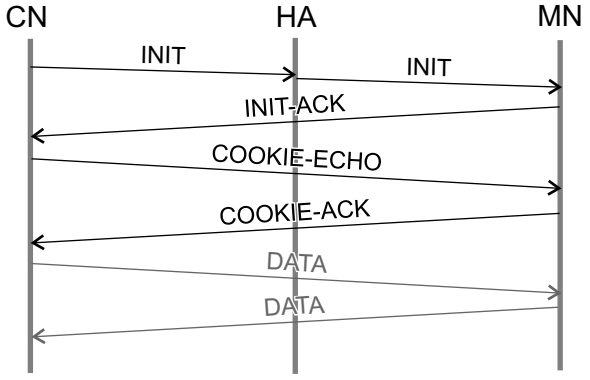


Abbildung 3: Location Management mit mSCTP und MIP

MIP nutzt zum Lokalisieren des MN einen Home Agent (HA). Der HA ist in der Regel ein Router, der sich im Heimatnetz des MN befindet. Bei MIPv4 teilt der mobile Knoten einem Netzwechsel mit der neuen IP-Adresse des Zugangsrouters im Fremdnetz (Foreign Agent) dem HA mit. Bei MIPv6 ist der Foreign Agent nicht mehr notwendig. Hier wird dann die IP-Adresse des Endgerätes übermittelt. Der Kommunikationspartner übermittelt seinen Verbindungswunsch an den HA, der dann ggf. über den FA oder direkt eine Verbindung

mit dem MN herstellt. Dazu muss der Partner MIP nicht unterstützen, nur für den mobilen Knoten ist dies erforderlich. [KLRM04]

Im jetztigen Stand von mSCTP kann bei MIPv4 die Care-of-Address nicht benutzt werden, da es sich um die Adresse des FA handelt, die für eine Assoziation zwischen MN und seinem Parter nicht zu gebrauchen ist. Hier muss dann die direkte IP-Adresse des MN übermittelt werden (Co-located CoA). Bei MIPv6 ist die CoA die IP-Adresse des Endgerätes [KoXi04].

Möchte nun ein CN eine Assoziation zum MN aufbauen, sendet er einen INIT-Chunk zum HA des MN. Der HA kennt zu jeder Zeit die aktuelle IP-Adresse des MN und tunnelt den INIT-Chunk zum MN zu der zur der Zeit gültigen IP-Adresse des MN. Die Antwort des MN (INIT-ACK) wird nicht, wie es normalerweise bei MIP der Fall ist, wieder über den HA zum CN geschickt, sondern direkt zum CN (kein Reverse Tunneling). Der mobile Knoten führt deswegen nach dem Erhalt des INITs vom CN ein „Binding Update“ aus. In dem INIT-ACK-Chunk vermerkt der MN seine aktuelle Adresse im Fremdnetz (CCoA bei MIPv4 bzw. CoA bei MIPv6). Die weitere Kommunikation wird direkt zwischen den Knoten ausgetauscht. Der CN kann nach dem Assoziationsaufbau die IP-Adresse des HA aus der Assoziation entfernen, da sie für die Dauer der Assoziation nicht mehr benötigt wird [KoXi04].

5.2 mSCTP und SIP

Das Session Initiation Protocol (SIP) wird besonders in der IP-Telefonie eingesetzt, um die Verbindung zwischen Teilnehmern herzustellen. SIP ist aber nicht nur auf Telefonverbindungen beschränkt, sondern wurde entwickelt um beliebige Verbindungen mit mehreren Teilnehmern zu ermöglichen. SIP arbeitet über mSCTP auf der Anwendungsschicht. Es ist inspiriert durch HTTP und benutzt auch eine ähnliche Header-Struktur. SIP benutzt für die Identifizierung und Lokalisierung der Teilnehmer „Unified resource identifieres“ (URI), die eine ähnliche Schreibweise wie E-Mail-Adressen (name@domain) haben [Inc.].

Die Lokalisierung wird durch SIP über SIP-REGISTER-Nachrichten ausgeführt. Dazu sendet der MN sobald er ein neues Netz betritt eine solche Nachricht mit Kontaktinformationen im Header, wie IP-Adresse oder SIP URL, an den Heimat-SIP Server des MN, der daraufhin einen entsprechenden Eintrag in der Lokalisierungs-Datenbank aktualisiert.

Möchte der CN eine Verbindung zum MN aufbauen, schickt der CN eine SIP-INVITE-Nachricht an den Heimat-SIP Server des MN. Dieser holt die aktuelle Position des MN aus der Datenbank und leitet die Nachricht über den SIP Proxy Server des Netzes, in dem sich der MN momentan aufhält an den MN weiter. Wenn die Assoziation mit Hilfe der SIP Signalisierung aufgebaut ist, erfolgt der Datentransport mit mSCTP [KLRM04].

5.3 Weitere Möglichkeiten

Neben MIP und SIP gibt es noch weitere Protokolle, die zur Lokalisierung eingesetzt werden können. Mit „Reliable Server Pooling“ (RSerPool) erstellt der MN einen Pool, in dem er sich als einzige Poolunit einträgt. Die IP-Adresse dieses Pools kann der CN dann abfragen [KLRM04]. Dynamic DNS (DDNS) benutzt eine feste Domain, deren DNS-Eintrag aber vom MN ständig an die aktuelle IP-Adresse des Fremdnetzes angepasst wird. [Inc.]

6 Schlussbetrachtungen

Da mSCTP Anwendungsunterstützung benötigt, ist ein Einsatz in bereits bestehenden Netzen schwierig und kostenintensiv, da die eingesetzten Anwendungen modifiziert werden müssten.

Außerdem ist die fehlende Lokalisierung nur durch den Einsatz eines weiteren Protokolls zu umgehen, wobei bei den vorgestellten Lösungen in Punkt 5 zusätzliche Komponenten erforderlich wären und der Vorteil einer Ende-zu-Ende-Lösung nicht mehr gegeben ist.

Allerdings benötigt mSCTP keine Modifikationen an der Netzstruktur oder zusätzliche Komponenten, wenn auf Location Management verzichtet werden kann. Daher bietet sich mSCTP für Projekte an, die mitsamt Anwendungen neu entwickelt werden müssen und dabei nur Verbindungen erforderlich sind, die von mobilen Knoten zu einem festen Server aufgebaut werden. Hier ist mSCTP dann eine kostengünstige Lösung [KüHV04].

An einer SCTP-Open-Source-Implementierung für Linux wird momentan gearbeitet. Beta-Versionen sind unter http://sourceforge.net/projects/lksctp/ erhältlich. Weitere Implementierungen sind für Solaris 9 von SUN und für FreeBSD/KAME erhältlich [Tuex].

Literatur

[Inc.] Wikimedia Foundation Inc. Wikipedia - Die freie Enzyklopädie. Internet: http://www.wikipedia.de.

[KLRM04] Seok J. Koh, Mee Jeong Lee, Maximilian Riegel und Mary Li Ma. Mobile SCTP for Transport Layer Mobility. IETF draft-sjkoh-sctp-mobility-04.txt, Juni 2004. Work In Progress.

[KoCL04] Seok Joo Koh, Moon Jeong Chang und Meejeong Lee. mSCTP for Soft Handover in Transport Layer. *IEEE Communications Letters* 8(3), März 2004, S. 189–191.

[KoXi04] Seok J. Koh und Qiaobing Xie. Mobile SCTP with Mobile IP for Transport Layer Mobility. IETF draft-sjkoh-mobile-sctp-mobileip-04.txt, Juni 2004. Work In Progress.

[KüHV04] Tobias Küfner, Dirk Hofmann und Christian Vogt. Mobility Management Approaches for 4G Networks, Dezember 2004.

[MYLR04] Li Ma, Fei Yu, Victor C. M. Leung und Tejinder Randhawa. A New Method to Support UMTS/WLAN Vertical Handover Using SCTP. *IEEE Wireless Communications*, August 2004, S. 44–51.

[RiTu04] Maximilian Riegel und Michael Tuexen. Mobile SCTP. draft-riegel-tuexen-mobile-sctp-04.txt, Oktober 2004. Work In Progress.

[SRXT04] R. Stewart, M. Ramalho, Q. Xie und M. Tuexen. SCTP Dynamic Address Reconfiguration. IETF draft-ietf-tsvwg-addip-sctp-09.txt, Juni 2004. Work In Progress.

[SXMS00] R. Stewart, Q. Xie, K. Morneault und C. Sharp. Stream Control Transmission Protocol. Request for Comments: 2960, Oktober 2000.

[Tuex] Michael Tuexen. Stream Control Transmission Protocol (SCTP). Internet: http://www.sctp.de/sctp.html.

[ZiWS04] Prof. Dr. Martina Zitterbart, Dipl.-Ing. Kilian Weniger und Dipl.-Inform. Oliver Stanze. Vorlesungsfolien Mobilkommunikation, 2004.

Abbildungsverzeichnis

1 SCTP (Chunks und Multistreaming) vs. TCP 156

2 mSCTP Handover 159

3 Location Management mit mSCTP und MIP 161

Tabellenverzeichnis

1 Protokolle für Mobilitätsmanagement [KLRM04] 161

Paketkopfkomprimierung in drahtlosen Zugangsnetzen

Martin Walser

Kurzfassung

Im folgenden wird ein Verfahren zur Paketkopfkompression beschrieben. Da der Informationsgehalt der Paketköpfe recht redundant ist, sind dabei Kompressionsraten von über 90% möglich. Des weiteren erweist sich das hier vorgestellte Verfahren, anders als seine Vorgänger, als äußerst robust, so daß es auch in drahtlosen Netzen verwendet werden kann.

1 Einleitung

1.1 Sinn des Paketkopfes

Der komplex aufgebaute Paketkopf ist für die Ende-zu-Ende Kommunikation extrem wichtig. Ein Paketkopf enthält neben protokollspezifischen Informationen die nötigen Angaben, um die Daten aus den einzelnen Paketen auf der Empfängerseite wieder richtig zusammensetzen zu können.

Abbildung 1: ein Datenpaket.

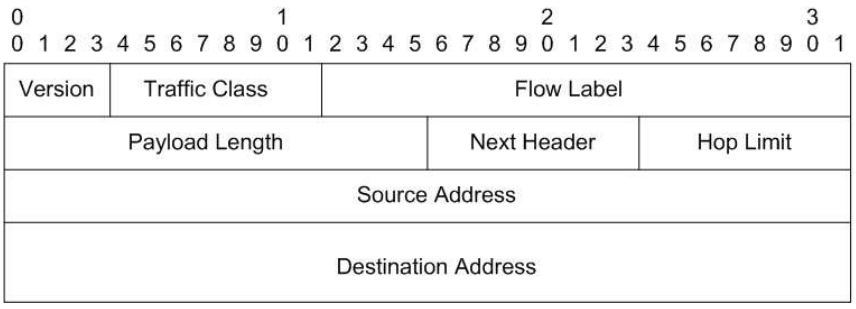
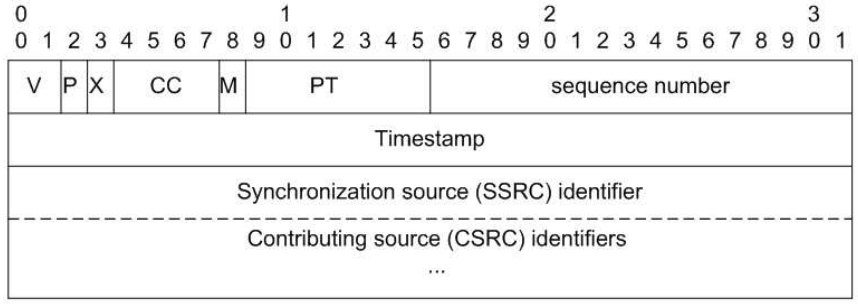


Abbildung 2: ein IPv6-Paketkopf.



V	Versionsnummer	CC	Anzahl der CSRC-Identifikatoren
P	Padding-Indikator	M	Markierung (indiv. Interpretation)
X	Indikator für Paketkopf-Erweiterung	PT	Nutzlast-Typ
SSRC	identifiziert Quelle des Zeitstempels	CSRC	Mixer-Information zu Datenquellen

Abbildung 3: ein RTP-Paketkopf.

1.2 Gründe für Paketkopfkomprimierung

Gerade in drahtlosen Netzen stellt die zur Verfügung stehende Bandbreite die teuerste Resource überhaupt dar, während Rechenleistung im Vergleich dazu ziemlich günstig ist. Anwendungen wie Voice over IP, Onlinespiele oder Messaging, bei denen der Anteil der Nutzdaten oft gleichgroß, wenn nicht sogar kleiner als der Anteil des jeweiligen Headers im Paket ist, wären deswegen geradezu prädestiniert für den Einsatz eines paketkopfkomprimierenden Verfahrens.

1.3 Ansätze für Paketkopfkomprimierung

1. Wie schon gezeigt, stellt der Paketkopf ein recht komplexes Gebilde dar. Vor jedem Hop kann der Paketkopf allerdings komprimiert und am anderen Ende wieder dekomprimiert werden. Kompressionsraten von über 90% stellen dabei ein nicht zu verachtendes Einsparungspotential dar.

2. Sowohl innerhalb eines Paketkopfes selbst als auch zwischen den verschiedenen Paketköpfen eines Paketstromes findet sich eine große Anzahl an Redundanzen - ein guter Ansatz für Kompressionen !

1.4 Vorteile der Paketkopfkomprimierung

1. **Weniger Paketverluste (bei ROHC):** ROHC paßt sich den jeweiligen Gegebenheiten des Netzes an und vermeidet so unnötige Paketverluste.

2. **Verbesserte Antwortzeiten (alle Verfahren):** Kleinere Paketgrößen führen letztendlich zu besseren Antwortzeiten.

2 Historie der Verfahren zur Paketkopfkomprimierung

2.1 CTCP - Das originale Kompressionsverfahren von Van Jacobsen

Van Jacobsen entwickelte das erste Header Compression Verfahren mit dem Ziel IP/TCP Datenströme über Schmalbandnetze zu beschleunigen. Um dies zu erreichen bediente er sich der Delta Kompression, bei der nur jeweils die wertmäßigen Differenzen in den veränderten

Feldern des Paketkopfes übermittelt werden und so die Anzahl der zu übertragenden Bits minimiert wird. Mit diesem Verfahren gelang es ihm den Paketkopf von 40 Bytes auf nur mehr durchschnittlich 4 Bytes zu reduzieren - sprich, er erreichte einen Kompressionsgrad von etwa 1:10. Leider hatte das Verfahren auch seine Nachteile:

1. Van Jacobsen sah keine Unterstützung von IP/UDP vor, da zu der Zeit als er CTCP entwickelt hat, der Datentransfer über UDP noch recht wenig Benutzung fand.
2. CTCP stützt sich auf die TCP-eigene Fehlerbehandlungsroutinen um Bitfehler während der Übertragung und Fehler aufgrund von Paketverlusten zu beheben.

Durch diese Nachteile ist das Verfahren leider völlig ungeeignet für drahtlose Netzwerke und Multimediaanwendungen, wie z.B. Audio- & Videostreaming.

2.2 IPHC & CRTP

IPHC (IP Header Compression) versucht die Nachteile von CTCP auszubügeln. Endlich fanden nicht mehr nur IP/TCP sondern auch UDP Paketköpfe Unterstützung. Um ferner auch RTP Pakete zu unterstützen wurde nochmals ein gesondertes Protokoll entwickelt - das CRTP (Compressed Real Time Protocol). Es unterstützt IPv4, UDP und natürlich RTP Header und kann diese von ursprünglich 40 Bytes auf bis zu 4 Bytes bzw. sogar nur mehr 2 Bytes (bei deaktivierter UDP Checksumme) komprimieren. Da UDP/RTP nicht wie IP/TCP Pakete von sich aus neu verschickt, wenn sie nicht beim Empfänger angekommen zu sein scheinen, mußte für IPHC und CRTP ein neues fehlerkorrigierendes Verfahren her: Sobald ein Fehler eintritt, teilt der Dekompressor auf der Empfängerseite dieses dem Kompressor auf der sendenden Seite mit. Daraufhin wird der Kontext, der die für die Dekompression wichtigen Informationen enthält, zwischen Kompressor und Dekompressor neu hergestellt, wofür einige Pakete unkomprimiert über das Netz geschickt werden müssen. Alle Pakete die ab dem fehlerhaften oder fehlenden Paket verschickt wurden werden verworfen. (Ausführliche Informationen zum Thema Kontext und Kontextwiederherstellung finden Sie in Abschnitt 3.2 dieses Seminars)

Dieses Verhalten sorgt leider auch dafür, daß IPHC und CRTP für einige Anwendungen die auf fehleranfälligen Verbindungen mit langen round-trip Times aufsetzen, wie z.B. Voice over IP über Funknetze, nicht geeignet sind, da sie bedingt durch die teure Fehlerbehandlung schlichtweg zu viel Overhead produzieren. Über einen 'TWICE' getauften Mechanismus wurde zwar versucht diesen Nachteil zu kompensieren, allerdings wird es hierfür zwingend notwendig die UDP Checksumme zu aktivieren, wodurch ein komprimierter Header wieder bis zu der doppelten Größe anwächst als es ohne TWICE der Fall wäre. Auch sehen nicht alle Anwendungen eine aktivierte UDP Checksumme vor. Als klassisches Beispiel dient hier wieder Voice over IP über drahtlose Netze bei dem es oft die bessere Wahl ist beschädigte Pakete einfach zu übertragen um keine Unterbrechungen im Sprachstrom zu riskieren.

2.3 ROHC

Es mußte also eine robustere Lösung gefunden werden, die weniger anfällig für Störungen während dem Transfer zwischen Kompressor und Dekompressor ist, ohne gleichzeitig eine Vergrößerung des komprimierten Headers in Kauf nehmen zu müssen. Nur ein solches Protokoll würde die Grundvorraussetzungen bieten um auch für IP Telefonie über drahtlose Netze attraktiv zu bleiben.

Eine weitere Hauptursache für die schlechte Leistung der bereits genannten Protokolle waren die langen Rount-Trip Times. Für beide Probleme mußte eine Lösung gefunden werden, die

nicht nur auf ein weniger fehleranfälliges bzw. schnelleres Netz vertraute, sondern schon in bestehenden Netzen funktioniert.

Für die Kompression sorgt bei ROHC ein window based least significant bits Algorithmus, mit dem es möglich ist die Headergröße auf bis zu ein Byte zu reduzieren. Mit dieser besseren Kompressionsrate ziehen auch verbesserte Antwortzeiten einher.

Für die Robustheit in drahtlosen Netzen sorgt ein Feedback Mechanismus über einen Rückkanal.

Um offen für zukünftige Entwicklungen zu sein ist ROHC von Anfang an als erweiterbares Framework konzipiert worden, das zusätzlich zu den schon vorhandenen Profilen um weitere Profile erweitert werden kann.

ROHC kennt dabei momentan 4 Basisprofile:

Profil 0: ROHC Uncompressed komprimiert Pakete, die von keinem der folgenden Profile komprimiert werden können

Profil 1: ROHC RTP komprimiert Pakete mit IP/UDP/RTP Paketköpfen

Profil 2: ROHC UDP komprimiert Pakete mit IP/UDP Paketköpfen

Profil 3: ROHC ESP komprimiert Pakete mit IP/ESP Paketköpfen

3 Funktionsweise von ROHC

3.1 Kodierungen und Kompressionsalgorithmen im ROHC Framework

Wie bereits in der Einleitung erwähnt, nutzt ROHC für die Kompression gewisse Redundanzen innerhalb eines Paketkopfes selbst bzw. zwischen den verschiedenen Paketköpfen eines Paketstromes aus. Um dabei verschiedene Paketströme auseinanderzuhalten besitzt ROHC einen Paketklassifizierer, der Pakete über Parameter wie den Paketkopf, Urspungs- & Zieladresse oder Urspungs- & Zielports auseinanderzuhalten weiss.

Redundanzen ergeben sich dadurch, daß sich einige Paketkopffelder im Paketstrom nie ändern, wie z.B. Ursprungs- & Zieladressen in den IP-Paketköpfen oder Ursprungs- & Zielports in UDP/TCP Paketköpfen. Wieder andere Felder ändern sich während der Übertragung gemäß einem bestimmten Muster, wie das Identifier Feld im IP Paketkopf, das um 1 oder eine andere beliebige feste Zahl, die vom Kompressor zu ermitteln ist, erhöht wird, aber auch konstant bleiben kann. Felder, die konstant bleiben, brauchen natürlich überhaupt nicht mehr übertragen werden, sobald der Kontext auf Kompressor- und Dekompressorseite steht. Felder die sich laufend ändern, ohne ein vorhersagbares Verhalten aufzuweisen, müssen allerdings komplett übertragen werden.

Für Felder mit einem konstanten, vorhersagbaren Veränderungsmuster existieren mehrere Kompressionsalgorithmen.

3.1.1 Delta-Kompression

Frühere Verfahren wie das Originale Van Jacobsen Verfahren oder das IPHC Kompressionsverfahren für TCP Paketköpfe stützten sich alleine auf die Delta-Kompression. Dabei wird zunächst das erste Paket unkomprimiert übertragen. Für die nachfolgenden Pakete werden jeweils die Veränderungen in den Paketkopffeldern zwischen zwei aufeinanderfolgenden Paketen

berechnet und nur diese übertragen und auf der Dekompressorseite mit dem bestehenden alten Wert als Referenz verrechnet. So ist es möglich den selben Wert mit weniger übertragenen Bits darzustellen.

Vorteil:

- Leicht zu implementierendes, einfaches Verfahren

Nachteile:

- Ineffizient bei Paketverlusten vor und nach der Kompression
- Kaum robust gegenüber Bitfehlern und kann sogar zu Paketverlusten auf der Dekompressorseite führen

3.1.2 Kompression mit Window-based Least Significant Bit (W-LSB)

Bei der LSB Kompression handelt es sich um einen komplexeren Algorithmus, der sich besonders gut zur Kompression von Paketkopffeldern, die nur kleinen Veränderungen unterliegen, eignet. Um eine höhere Robustheit für das ROHC Protokoll zu erreichen wurde das LSB Grundkonzept um Prüfsummen erweitert und es wurden einige Mechanismen angepasst. So entstand der W-LSB Algorithmus.

Die Grundkonzepte der W-LSB Kompression:

- Der Dekompressor benutzt genau einen der zuvor bereits dekomprimierten Werte des betreffenden Paketkopffeldes als Referenzwert v_ref. Es handelt sich dabei um den letzten Wert, den er empfangen hat und den er mit einer vom Kompressor beigefügten Prüfsumme erfolgreich verifizieren konnte.
- Der Kompressor hält ein Sliding Window of Values (VSW) bereit, das diejenigen Werte enthält, die vom Dekompressor potentiell als Referenz benutzt werden könnten. Weiterhin merkt er sich den Maximalwert (v_max) und den Minimalwert (v_min) des VSW.
- Wenn nun die Kompression eines Wertes v ansteht, bestimmt der Kompressor zunächst die Entfernung von den VSW Grenzen

  $r = max(|v - v_{max}|, |v - v_{min}|)$.

  Die Anzahl der zu übertragenden Bits wird mit

  $k = ceiling(log_2(2 * r + 1)$

  bestimmt. D.h. die k LSBs des Wertes stellen den komprimierten Wert dar, der übertragen wird.
- Falls der Kompressor sich dazu entschließt den Wert v als Referenzwert aufzunehmen, fügt er ihn in das VSW ein, und weist v_min und v_max ggf. neue Werte zu.
- Der Dekompressor bestimmt schließlich den dekomprimierten Wert indem er aus den v_ref umgebenden Werten denjenigen auswählt, dessen letzten k Bits dieselben sind, wie die des empfangenen komprimierten Wertes und der gleichzeitig am nächsten an diesem Wert v_ref liegt.

Um zu verhindern, daß das Sliding Window immer größer wird oder immer an der selben Stelle verharrt, müssen Werte zum VSW hinzugefügt oder aus diesem gelöscht werden. Hinzugefügt werden automatisch alle Werte, die der Kompressor mit einem CRC versieht und somit für den Dekompressor einen Referenzwertkandidaten darstellen. Aus dem VSW gelöscht werden diejenigen Werte, von denen der Kompressor sich sicher sein kann, daß sie für in Zukunft zu komprimierende Werte keine Referenz mehr darstellen können. Bekommt der Kompressor vom Dekompressor den Erhalt eines mit CRC ausgestatteten Paketes mittels einem ACK Paket bestätigt, wie dies im R-Mode Betriebsmodus des Kompressors (siehe 3.4.3) der Fall ist, kann er sich danach sicher sein, daß Werte aus Paketen, die älter sind als das bestätigte dem Dekompressor in Zukunft nicht mehr als Referenzwert dienen werden. Somit kann er diesen Wert aus seinem VSW entfernen. In den beiden anderen Betriebsmodi ist das VSW schlicht größenbeschränkt. Die gewählte Größe des Fensters hängt dabei von der Implementierung ab.

Die folgenden Beispiele sollen die Funktionsweise der Komprimierung in verschiedenen Szenarien veranschaulichen. Die angegebenen Werte stehen dabei stellvertretend für die Werte in einem beliebigen Paketkopffeld.

Beispiel 1: Keine vorhergehenden Paketverluste und keine vertauschten Pakete nach dem Eintreffen beim Kompressor

Inhalt der Paketkopffelder, der vom Kompressor bereits verschickten Pakete: 278, 279, 280, 281, 282, 283

Potentielle Referenzwerte: 279, 283

Damit sieht das VSW folgendermaßen aus. {279, 283}

Der Kompressor empfängt nun ein Paket in dem das entsprechende Paketkopffeld den Wert 284 hat.

Neuer Wert: 284

$v_{max} = 283$

$v_{min} = 279$

$r = max[|284 - 279|, |284 - 283|] = 5$

$\#LSBs : k = ceiling(log2(2 * 5 + 1)) = 4$

Wir nehmen an, daß 284 als neuer Referenzkandidat nicht in Frage kommt. Damit bleibt der VSW unverändert. Das Feld wird mit diesem Fall mit 4 bits kodiert und der kodierte Wert entspricht den letzten 4 Bits in der Binärdarstellung des Wertes $284_{10} = 10011100_2$, also 1100_2

Beispiel 2: Vorhergehender Paketverluste

Inhalt der Paketkopffelder, der vom Kompressor bereits verschickten Pakete: 278, 279, 280, 281, 282, 283

Potentielle Referenzwerte: 279, 283

Damit sieht das VSW folgendermaßen aus. {279, 283}

Der Kompressor empfängt nun ein Paket in dem das entsprechende Paketkopffeld den Wert 290 hat.

Neuer Wert: 290

$v_{max} = 283$

$v_{min} = 279$

$r = max[|290 - 279|, |290 - 283|] = 11$

$\#LSBs : k = ceiling(log_2(2 * 11 + 1)) = 5$

Damit wird das Feld mit 5 Bits kodiert. Der kodierte Wert entspricht den letzten 5 Bits in der Binärdarstellung des Wertes $290_{10} = 100100010_2$, also 00010_2

Wir nehmen an, daß der Wert als neue Referenz benutzt werden soll. VSW enthält nun die Werte 279, 283, 290

Beispiel 3: Vertauschte Pakete

Inhalt der Paketkopffelder, der vom Kompressor bereits verschickten Pakete: 277, 279, 280, 281, 282, 283

Potentielle Referenzwerte: 279, 283

Damit sieht das VSW folgendermaßen aus. {279, 283}

Der Kompressor empfängt nun ein Paket in dem das entsprechende Paketkopffeld den Wert 278 hat.

Neuer Wert: 278

$v_{max} = 283$

$v_{min} = 279$

$r = max[|278 - 279|, |278 - 283|] = 5$

$\#LSBs : k = ceiling(log_2(2 * 5 + 1)) = 4$

Damit wird das Feld mit 4 Bits kodiert. Der kodierte Wert entspricht den letzten 4 Bits in der Binärdarstellung des Wertes $278_{10} = 10010110_2$, also 0110_2

Wir nehmen an, daß der Wert als neue Referenz benutzt werden soll. VSW enthält nun die Werte 283, 290, 278

Das Verhalten des Dekompressors ist in allen Beispielen dasselbe. Als Referenz benutzt er den letzten Wert, den er mit einer Prüfsumme verifizieren konnte. Per Definition muß ist dieser auch im VSW enthalten.

Beispiel: Angenommen der letzte erfolgreich dekomprimierte Wert war 291 und der Dekompressor empfängt nun als komprimierten Wert 01111_2. Die zwei dem Referenzwert nächstliegenden Dezimalzahlen mit der entsprechenden Endung in ihrer Binärdarstellung wären 271 und 303. 303 liegt dem Referenzwert 291 näher und wird daher als Interpretation des zu dekomprimierenden Wertes gewählt.

Vorteile von W-LSB:

- Robust gegenüber Bitfehlern und Paketverlusten
- Verbessert die Antwortzeiten

Nachteile:

- -

3.1.3 Kompression mit skaliertem RTP Zeitstempel (Scaled RTP Timestamp encoding)

W-LSB eignet sich hervorragend zur Komprimierung von Feldern, die sich zwar ständig und undeterministisch, aber doch nur gering verändern. Da der Wert des Zeitstempels in einem RTP Paket größeren, aber auch konstanten Veränderungen unterliegt wurde für diese Pakete ein eigenes Kompessionsverfahren gewählt.

Der Wert des RTP Zeitstempels (TS) eines RTP Paketes erhöht sich von Paket zu Paket um ein festes Vielfaches n der Einheit TS_{STRIDE}. Für eine Audiodatei, die mit 8 kHz abgetastet wird und bei der jedes Einzelpaket 20ms abdeckt, nimmt TS_{STRIDE} den Wert 160 (= 8000Hz * 0,02s) an. Der Wert des Zeitstempels erhöht sich demnach mit jedem Paket um n * 160.

Bei diesem Kodierungsverfahren wird der Zeitstempel vor der Kompression um einen Faktor von TS_{STRIDE} herunterskaliert. Die Einsparung dabei beträgt $floor(log_2(TS_{STRIDE}))$ bits für jedes komprimierte Zeitstempelfeld. TS und TS_{SCALED} genügen dabei folgender Gleichung: $TS = TS_{SCALED} * TS_{STRIDE} + TS_{OFFSET}$

TS_{STRIDE} wird dem Dekompressor explizit, TS_{OFFSET} implizit mitgeteilt.

Die Kodierung läuft über 3 Stufen ab.

Stufe 1: Initialisierung

Der Kompressor schickt zum Dekompressor den Wert von TS_{STRIDE} und den absoluten Wert einiger Zeitstempel Felder. Aus diesen Werten kann der Dekompressor mittels TS modulo TS_{STRIDE} den Offset ermitteln.

Stufe 2: Kompression

Nach der Initialisierungsphase komprimiert der Kompressor nicht mehr den eigentlichen Wert des Zeitstempels sondern den mittels TS_{STRIDE} herunterskalierten Wert: $TS_{SCALED} = TS/TS_{STRIDE}$. Zur Kompression kann entweder der schon erwähnte W-LSB Algorithmus verwendet werden oder aber auch ein Systemzeituhr-basiertes Kompressionsverfahren eingesetzt werden.

Stufe 3: Dekompression

Aus dem empfangenen, komprimierten Wert von TS_{SCALED} stellt der Dekompressor nun wieder den ursprünglichen Wert her: $TS = TS_{SCALED} * TS_{STRIDE} + TS_{OFFSET}$

3.2 Der Kontext

Der Kontext umfasst Informationen über statische Felder, dynamische Felder und deren Verhaltensmuster bzgl. fortlaufender Veränderungen im Paketkopf. Diese Informationen benutzt der Kompressor um den Paketkopf so effizient wie möglich komprimieren zu können. Der Dekompressor benutzt sie schließlich wieder um den ursprünglichen Zustand wiederherzustellen.

Für die Kompression der verschiedenen Paketköpfe in einem Paketstrom bedarf es einiger Vorarbeit. Zunächst werden einige Pakete unkomprimiert verschickt um den Kontext auf beiden Seiten der Verbindung herzustellen.

3.3 Zustandsmodelle

Nicht immer funktioniert die Übertragung reibungslos. Es kann vorkommen, daß die Paketreihenfolge vor dem Eintreffen beim Kompressor durcheinandergewirbelt wird oder daß sogar einzelne Pakete während dem Transfer schlicht verloren gehen. Sowohl der Kompressor als auch der Dekompressor können deshalb verschiedene Zustände passend zur jeweiligen Situation annehmen.

3.3.1 Kompressorzustände

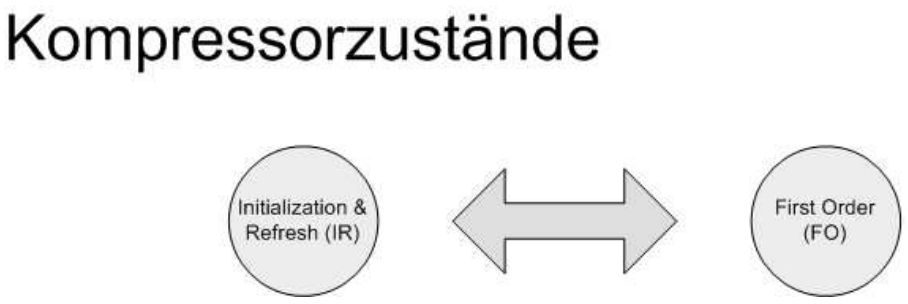


Abbildung 4: Vereinfachtes Zustandsübergangsmodell des Kompressors.

Der Kompressor kennt 3 Zustände:

- Initialization and Refresh (IR)
- First Order (FO)
- Second Order (SO)

Die Zustände beschreiben die unterschiedlichen Sicherheitseinstufen bezüglich der Korrektheit des Kontextes auf der Dekompressorseite. Second Order stellt den Zustand dar bei dem

zum einen der Kompressor das meiste Vertrauen in die noch vorhandene Korrektheit des Kontextes auf Dekompressorseite hat und mit dem zum anderen auch die höchsten Kompressionsraten zu erzielen sind. IR ist umgekehrt die niedrigste Einstufung mit den niedrigsten Kompressionsraten.

Zum Beginn einer neuen Übertragung startet der Kompressor im Zustand IR um den Kontext auf der Dekompressorseite herzustellen. Hierfür werden grundstätzlich alle Felder übertragen. Sobald er sich sicher sein kann, daß der Dekompressor aus dem Datenstrom den statischen Kontext gewonnen hat, wechselt er in den First Order Zustand. Ab jetzt müssen alle Felder, die sich nie ändern, also statisch sind, nicht mehr übertragen werden. Sind im FO Zustand auch irgendwann alle Parameter für die Felder gewonnen, die sich dynamisch ändern, wird schließlich in den Second Order Zustand gewechselt. Nun werden nur noch Felder übertragen, die sich völlig unregelmäßig ändern.

Tritt nun auf der Dekompressorseite bei der Dekompression eines Paketes ein (CRC-)Fehler auf, teilt der Dekompressor über einen Rückkanal (wir gehen an dieser Stelle noch davon aus, daß ein solcher auf jeden Fall existiert) dieses dem Kompressor mit, der daraufhin in einen niedrigeren Zustand wechselt. Ein Wechsel in einen niedrigeren Zustand erfolgt auch, wenn der Kompressor bemerkt, daß ein eintreffendes Paket mit dem bisher gewonnenen Veränderungsmuster nicht mehr konform geht und deswegen der Kontext erneuert werden muß. Nun wird versucht mit einer geringeren Kompression den Kontext wiederherzustellen. Sollte dies im Zustand FO nicht gelingen, muß in den niedrigsten Zustand IR gewechselt werden und der Kontext erst mittels gänzlich unkomprimierter Pakete wiederhergestellt werden, ehe wieder in einen höheren Zustand gewechselt werden kann.

Der Zustand wird also dynamisch der Qualität der Verbindung bzw. der Häufigkeit des Auftretens von Fehlern, über deren Eintreten der Kompressor vom Dekompressor informiert wird, angepaßt. Wenn kein Rückkanal zur Verfügung steht erniedrigt der Kompressor von Zeit zu Zeit von sich aus seinen Zustand, um sicherzustellen, daß der Zustand auf der Dekompressorseite noch korrekt ist. Welche Bedingungen nun genau für einen Zustandsübergang notwendig sind hängt vom aktuellen Operationsmodus ab. Die unterschiedlichen Operationsmodi werden im Abschnitt 3.4 behandelt.

3.3.2 Dekompressorzustände

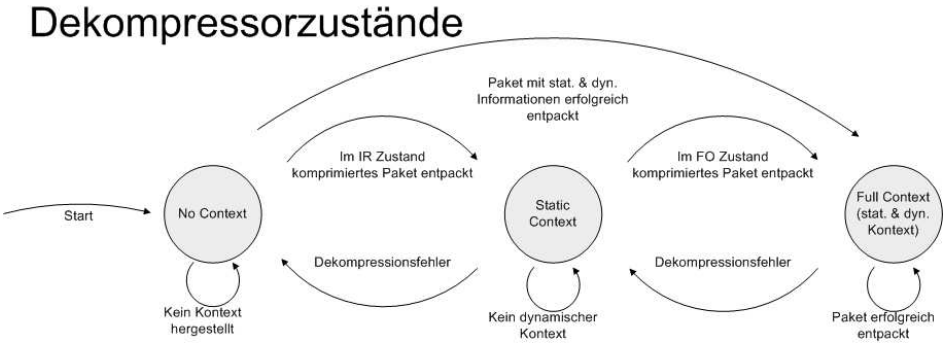


Abbildung 5: Zustandsübergangsmodell des Dekompressors.

Auch der Dekompressor kennt 3 Zustände:

- No Context
- Static Context

- Full Context (statischer und dynamischer Kontext hergestellt)

Zu Beginn einer neuen Übertragung befindet sich der Dekompressor im Status 'No Context', da ihm zu diesem Zeitpunkt noch keine Kontextinformationen zur Verfügung stehen. Sobald er erfolgreich vom Kompressor ein Paket, das Informationen über sich statisch als auch dynamisch ändernde Felder enthält, empfangen und dekomprimiert hat ist auch schon auf Dekompressorseite der Kontext hergestellt, und er kann direkt in den FFull ContextSZustand wechseln. Treten nun wiederholt Fehler auf, wird in einen niedrigeren Zustand gewechselt. Sobald er im Static Context Zustand ein Paket empfängt, das der Kompressor im First Order Zustand verschickt hat, kann wieder in den Full Context Modus gewechselt werden. Sollte die Dekompression von Paketen im Static Context Zustand jedoch mehrfach scheitern, wird weiter in den No Context Zustand zurückgegangen.

3.4 Operationsmodi

ROHC kann in 3 Operationsmodi arbeiten. Die Wahl des Modus ist dabei abhängig von Parametern wie

- Verfügbarkeit eines Rückkanals (Dekompressor -> Kompressor)
- Fehleranfälligkeit der Übertragung
- Veränderungsmuster der Paketkopffelder

3.4.1 Unidirektionaler Modus (U-mode)

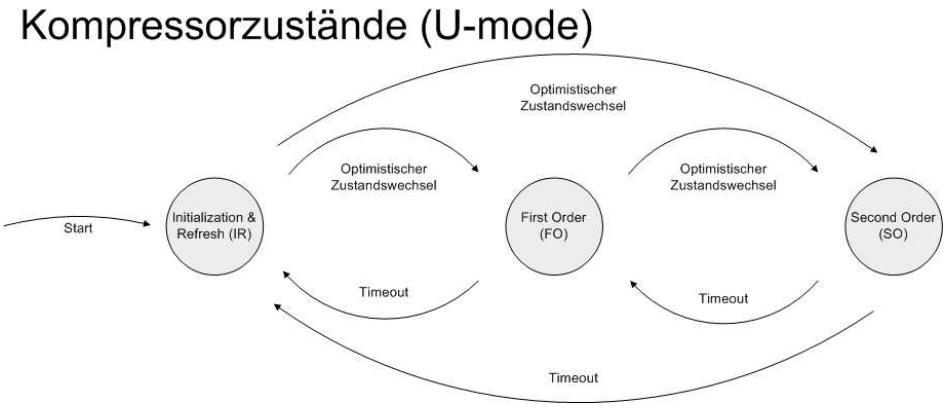


Abbildung 6: Zustandsübergangsmodell des Kompressors im U-mode.

Dieser Operationsmodus ist für Netze gedacht in denen kein Rückkanal zur Verfügung steht. Der Ablauf der Zustandsübergänge gestaltet sich in diesem Modus folgendermaßen:

Der Kompressor startet im Ïnitialization and RefreshSZustand und beginnt eine Anzahl an Paketen zum Dekompressor zu schicken, damit dieser den Kontext herstellen kann. Die Anzahl der zu verschickenden Pakete hängt dabei von netzeigenen Eigenschaften wie zum Beispiel der Round Trip Time ab. Nachdem die Pakete übertragen wurden und angenommen werden kann, daß der Dekompressor den Kontext hergestellt hat, erfolgt der Sprung in einen höheren Zustand.

In festen zeitlichen Abständen springt der Kompressor dann wieder in niedrigere Zustände um den Kontext zu synchonisieren und verschickt First Order und Initialization and Refresh Pakete.

Nachteil:

- Dieser Modus ist ineffizient sowohl was den Statuswechsel bei Kontextfehlern angeht, als auch was die Kontextsynchronisation als ganzes betrifft. Das hat zur Auswirkung, daß ein geringer Gewinn aus der Kompression geschöpft wird.

Initial startet der Kompressor immer im U-Mode. Sobald der Dekompressor allerdings über einen Rückkanal den Empfang eines Pakets mit einem ACK Paket quittiert, kann in einen der beiden folgenden Operationsmodi gewechselt werden.

3.4.2 Bidirektionaler Optimistischer Modus (O-mode)

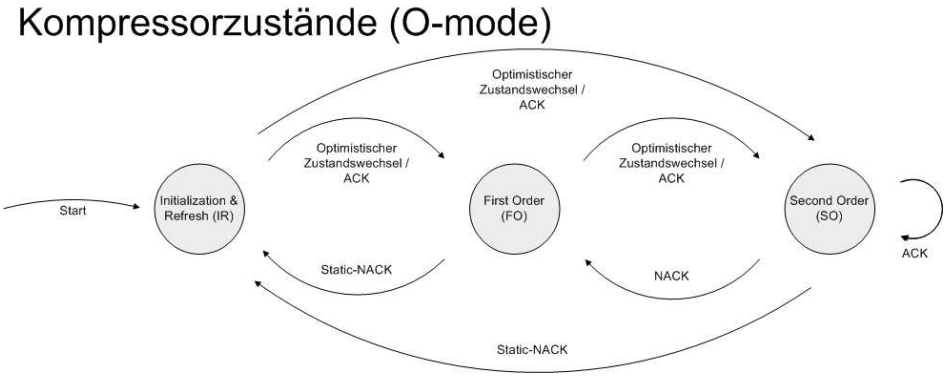


Abbildung 7: Zustandsübergangsmodell des Kompressors im O-mode.

Sobald in diesen Modus umgewechselt wird steht fest, daß ein Rückkanal vorhanden ist. Über diesen kann der Dekompressor zum einen mittels NACK Pakete dem Kompressor mitteilen kann, daß eine Fehlersituation eingetreten ist, und zum anderen kann er über ACK Pakete diesem mitteilen, daß der Kontext erfolgreich aktualisiert wurde.

Um auf Kompressorseite in einen höheren Zustand zu wechseln wird entweder wie im U-Mode ein optimistischer Ansatz verwendet oder auf ein ACK-Paket gewartet. Ein ACK Paket ist nur als Antwort auf ein IR Paket notwendig, wohingegen es für andere Kontext Updates optional ist.

Der Wechsel in einen niedrigeren Zustand erfolgt, wenn der Dekompressor NACK bzw. Static NACK Pakete verschickt. Bei Empfang eines Static NACKs erfolgt auf Kompressorseite ein sofortiger Wechsel in den IR Zustand.

Vorteile:

- Höherer Kompressionsgewinn als im U-Mode
- Der Rückkanal wird selten benutzt
- Weniger Paketverluste durch ungültig gewordene Kontexte

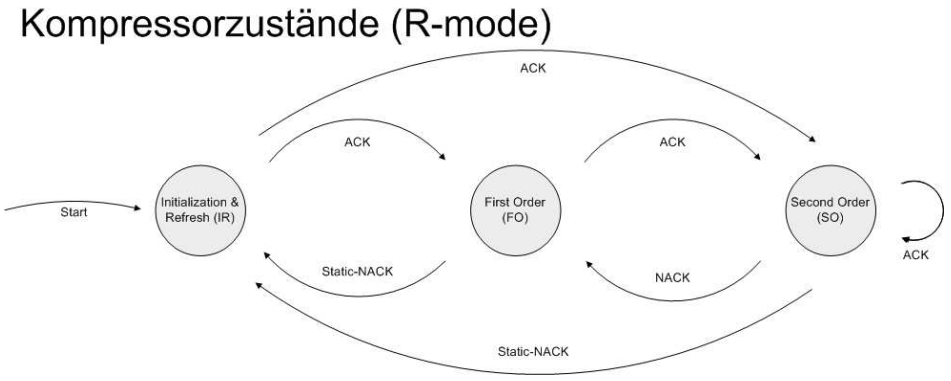


Abbildung 8: Zustandsübergangsmodell des Kompressors im R-mode.

3.4.3 Bidirektionaler Zuverlässiger Modus (R-mode)

In diesem Modus findet der Rückkanal exzessive Verwendung um Paketverluste durch ungültige Kontexte zu vermeiden. Dem Sicherheitsprinzip wird hier also dem optimistischen Ansatz der beiden vorhergehend erwähnten Modi den Vortritt gelassen.

Für jeden Kontextwechsel auf Kompressorseite ist zuvor ein ACK Paket auf Dekompressorseite nötig. Kontext Update Pakete werden dabei in periodischen Zeitabständen solange verschickt bis ein ACK Paket eintrifft. Diese Vorgehensweise schlägt sich daher direkt in einer geringeren Kompressionseffizienz nieder.

Das Ziel des R-Modes ist es eine bessere Robustheit gegenüber Paketverlusten und Bitfehlern zu erreichen, und tatsächlich ist die Wahrscheinlichkeit, daß Kontexte ungültig werden sehr gering. Tritt dieser Fall jedoch ein, kann das zur Folge haben, daß die Versendung einer großer Anzahl unbrauchbarer Pakete in die oberen Schichten erfolgt.

3.4.4 Das Zusammenspiel der verschiedenen Operationsmodi

Wie schon erwähnt, startet der Kompressor immer im U-Mode, und sobald auf Kompressorseite ein ACK-Paket eingetroffen ist, erfolgt der Wechsel in einen anderen Modus. Die Entscheidung welcher Modus nun gewählt wird trifft ein Operator, dessen Aufgabe es ist die eingangs in 3.4 erwähnten Parameter zu ermitteln und auszuwerten. Je nach Situation kann der Operator später weitere Modiwechsel anordnen.

4 ROHC Einsatzgebiete

1. **ROHC in 3GPP und 3GPP2 Netzen:** In 3GPP Netzen findet ROHC seit Version 4 verwendet und stellt seit Version 5 eine der Kernkomponenten dar. Auch in 3GPP2 Netzen ist ROHC eine feste Komponente und wird dort für Multimediadaten benutzt.

2. **ROHC unter IPv6:** IPv6 bringt es zusammen mit UDP/RTP auf einen 60 Byte großen Paketkopf. Um diesen beachtlichen und gerade bei Audio- und Videokommunikation störenden Overhead zu begrenzen bietet sich ROHC geradezu an.

3. **ROHC in Satellitennetzen:** Ein relativ neues Betätigungsfeld der Betreiber von Satellitennetzen ist die Versorgung mit Breitbandinternet in eher abgelegenen Gebieten, die noch nicht mit breitbandigen Festnetzen erschlossen sind. Doch Satellitennetze werden von hohen Bitfehlerraten und langen Round Trip Times geplagt. Durch den Einsatz

von ROHC werden die Effekte dieser beiden Faktoren minimiert, eine bessere Quality of Service für den Benutzer und eine bessere Ausnutzung der Netzkapazität für den Anbieter erreicht.

4. **WLAN Netze:** Auch WLAN Netze zeichnen sich durch eine hohe Bitfehlerrate aus. Bedingt durch die Frequenz, die sie nutzen, können sie leicht von Haushaltsgeräten, wie Mikrowellenherden, gestört werden. Durch die kleineren Paketgrößen bei Einsatz von ROHC werden die Auswirkungen von Funkstörungen jedoch geringer und Verzögerungen, bedingt durch notwendige Neuversendungen von Paketen, nehmen ab.

5 Abschließende Worte

In den vergangenen Abschnitten wurden die Funktionsweise und die Vorteile einer robusten Paketkopfkompression aufgezeigt. Nicht tiefergehend behandelt wurden die einzelnen in Abschnitt 2.3 gezeigten Profile. Auch wurde die Problematik des zwangsläufigen Kontextverlustes bedingt durch den Wechsel des Zugangspunktes in drahtlosen Netzen ausgeklammert. Deswegen seien dem Leser an dieser Stelle noch die Arbeiten von [Fein05] und [Schu05] ans Herz gelegt.

Literatur

[Borm01] C. Bormann. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. RFC 3095 (Standards Track), July 2001.

[Dege04] M. Degermark. Requirements for robust IP/UDP/RTP header compression. RFC 3096 (Standards Track), September 2004.

[DeHi98] S. Deering und R. Hinden. Internet Protocol, Version 6 (IPv6). RFC 2460 (Standards Track), December 1998.

[Fein05] M. Feinleb. Algorithmen der robusten Paketkopfkomprimierung. Seminararbeit, Februar 2005.

[SCFJ96] H. Schulzrinne, S. Casner, R. Frederick und V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Standards Track), January 1996.

[Schu05] A. Schuster. Kontexttransfer für robuste Paketkopfkomprimierung. Seminararbeit, Februar 2005.

Abbildungsverzeichnis

1 ein Datenpaket. 165

2 ein IPv6-Paketkopf. 165

3 ein RTP-Paketkopf. 166

4 Vereinfachtes Zustandsübergangsmodell des Kompressors. 173

5 Zustandsübergangsmodell des Dekompressors. 174

6 Zustandsübergangsmodell des Kompressors im U-mode. 175

7 Zustandsübergangsmodell des Kompressors im O-mode. 176

8 Zustandsübergangsmodell des Kompressors im R-mode. 177

Algorithmen der robusten Paketkopfkomprimierung

Maxim Feinleb

Kurzfassung

Diese Seminararbeit beschreibt einige mögliche Verfahren um den in den Paketköpfen einer Datenverbindung übertragenen Overhead auf ein Minimum zu reduzieren. Es wird anhand des RTP/UDP/IP-Protokollstapels beispielhaft gezeigt, wie die einzelnen Headerfelder bestimmten Verhaltensklassen zugeordnet werden können. Abschließend wird diskutiert, wie sich diverse Kompressionsstrategien auf einige konkrete Felder bzw. Verhaltensklassen anwenden lassen.

1 Einleitung

In den letzten zehn Jahren gab es vor allem zwei Technologien, die wie keine anderen in den deutschen Haushalten Einzug hielten. Zum Einen waren es Computer und zum Anderen Mobiltelefone. Laut einer Pressemitteilung des statistischen Bundesamtes vom 5. August 2004, erfolgte der Zugang zum Internet im Jahre 2003 fast ausschließlich (98%) über den PC. Dabei besitzen, laut der gleichen Mitteilung, mittlerweile drei Viertel aller Haushalte ebenfalls ein Mobiltelefon und es ist davon auszugehen, dass fast alle dieser Geräte internetfähig sind.

Versucht man das beobachtete Ungleichgewicht beim Zugriff auf das Medium Internet zu erklären, erkennt man schnell, dass die Schranken nicht in den Endgeräten zu suchen sind. Vielmehr ist es die Beschaffenheit der Luftschnittstelle (U_m) und die Kosten, die für ihre Benutzung entstehen. Momentan sind diese - pro Megabyte gerechnet - mehrere Tausend Mal höher, als die für die kabelgebundene Variante.

Generell kann man davon ausgehen, dass heutige Festnetzpreise für ein Megabyte übertragener Daten im Bereich zwischen wenigen Hundertsteln und einigen Zehnteln Cent liegen. Bei dieser Preislage und entsprechender Übertragungsgeschwindigkeit ist man gerne bereit den mitgetragenen Overhead unbeachtet zu lassen. Diese Einstellung ändert sich jedoch schnell wendet man sich dem Mobilkommunikationssektor zu. Genau an diesem Punkt setzt das im [Borm01] vorgestellte Verfahren der „Robust Header Compression“ ein.

1.1 Einführung in Robust Header Compression (ROHC)

Beim Kommunikationsaufbau, sowie während der gesamten Zeit, in der eine Verbindung besteht, werden Daten übertragen, die sich stark wiederholen. Vor allem sind diese redundanten Daten in Paketköpfen (Headern) zu finden. [Borm01]

ROHC nutzt diese Tatsache und stellt ein Framework zur Verfügung, mit dem die tatsächlich übertragenen Headerinformationen bis auf ein Minimum reduziert werden. Gleichzeitig wird die Robustheit der Verbindung gewährleistet und der Feedbackdatenfluss optimiert.

Als Einsatzgebiete für ROHC eignen sich vor allem langsame und/oder fehlerbehaftete Verbindungen. ROHC ließe sich beispielsweise bei „Voice over IP“ mit einer drahtlos angebundenen

Station sinnvoll anwenden. Es werden viele kleine Nachrichten bei hoher Störungswahrscheinlichkeit verschickt, manche Pakete sogar mit mehr Overhead als Nutzdaten. In diesem oder einem beliebigen ähnlichen Fall kann ROHC einen bedeutenden Leistungsgewinn erzielen.

Eine weitere Funktion von ROHC ist das automatische Erkennen und Konfigurieren mehrere paralleler Streams, die sich gemeinsam eine Verbindung teilen. Es ist gut vorstellbar, dass die oben erwähnte Station parallel zu einer „Voice over IP“ Übertragung auch HTML Seiten darstellen kann. Die Station in unserem Beispiel ist drahtlos und so ist die Wahrscheinlichkeit gross, dass beide Prozesse denselben Kanal zur Übertragung nutzen. Wäre ROHC nicht in der Lage die Übertragungsstreams einzeln zu erkennen bzw. einen optimalen Profil für jeden einzelnen dieser Streams zu bestimmen, wäre eine vernünftige Komprimierung kaum möglich.

2 ROHC-Profile im Überblick

ROHC ist als ein flexibles Framework konzipiert und somit nicht an bestimmte Protokolle gebunden. Vielmehr lässt es sich beinahe beliebig an eine ganze Reihe von Protokollen anpassen. Der Schlüssel hierfür sind die Profile. Diese legen unter anderem fest, mit welchem Verfahren die einzelnen Felder des Headers komprimiert werden bzw. ob bei manchen Feldern keine Kompression angewandt werden darf.

2.1 Bereits standardisierte Profile

Zum Zeitpunkt der Ausarbeitung dieser Seminararbeit, waren unter anderem folgende Profile bereits standardisiert (hier einige wichtige):

Profil 0x0000 - IP, keine Komprimierung: Dieses Profil wird immer dann benutzt, wenn kein anderes Profil auf die behandelten Daten angewandt werden kann bzw. keine Komprimierung erwünscht ist.

Profil 0x0001 - RTP/UDP/IP: Das RTP-Profil ist das allgemeinste Profil, es lassen sich von ihm alle hier aufgeführten Profile ableiten. Es behandelt die RTP-Sequenznummer als Bezugspunkt für alle dynamischen Felder und überträgt diese mit geeigneter Komprimierung (siehe auch Abschnitt 4.3 und Kapitel 6).

Profil 0x0002 - UDP/IP: UDP-Header führen keine Sequenznummern mit, weshalb eine virtuelle 16-bit Sequenznummer vom Komprimierer vergeben wird. Diese Nummer wird als Bezug für alle sich verändernden Felder benutzt. Dieses Profil ist ansonsten sehr ähnlich zu dem RTP-Profil.

Profil 0x0003 - ESP/IP: Die ESP-Header besitzen zwar Sequenznummern, können aber generell nicht komprimiert werden, sobald der Verschlüsselungsalgorithmus einsetzt. Bis dahin kann allerdings dieses Profil verwendet werden, das dem UDP-Profil sehr ähnlich ist.

Profil 0x0004 - IP: Alle von ROHC behandelten Pakete sind IP-Pakete und somit ist in den meisten Fällen zumindest eine Komprimierung auf der Vermittlungsschicht wünschenswert. Dieses Profil unterscheidet sich vom UDP-Profil insbesondere darin, dass am Ende des IP-Headers kein UDP-Header erwartet wird. Weitere Informationen sind unter [JoPe04] zu finden.

2.2 Profile in der Entwicklung

Folgende Profile waren zum Zeitpunkt der Seminarausarbeitung in der Standardisierungsphase (einige wichtige):

Profil 0x0006 - TCP/IP: Trotz einiger vorhergehender Versuche einen Komprimierungsalgorithmus für TCP zu finden, war man bis heute nicht in der Lage TCP/IP-Pakete mit hinreichend hoher Effizienz zu komprimieren und anschließend mit genügender Robustheit zu übertragen. Besondere Schwierigkeiten bereiten vor allem kurzlebige TCP-Verbindungen sowie die Übertragung von TCP-Optionen. Ein weiterer Ansatz die TCP Problematik unter Einsatz von ROHC in den Griff zu bekommen, mündete bisher in diesem Profil, der im Draft [PeJo04] definiert wird.

Profil 0x0008 - UDP-Lite: Das UDP-Lite-Protokoll ist aus dem UDP-Protokoll abgeleitet und besitzt daher nur eine Untermenge dessen Felder. Es wird hauptsächlich bei fehlertoleranter Übertragung eingesetzt (bspw. „Voice over IP“), bei welcher der Empfang eines Paketes wichtiger ist als seine Korrektheit. In dem Bestreben eine optimale Kompression für dieses Protokoll zu finden, entstand dieses Profil. Im Draft [Pell04] lassen sich dazu weitere Informationen finden.

3 Das Komprimierungssystem

Die Komprimierung in ROHC basiert auf der Interaktion zweier Zustandsautomaten, die für jeden übertragenen Stream jeweils eingerichtet werden. Einer dieser Automaten ist der Komprimierer und der andere - der Dekomprimierer. Beide Automaten besitzen drei Zustände die miteinander stark korrelieren.

3.1 Zustände des Komprimierers

Der Komprimierer in ROHC besitzt drei Zustände, wobei der Zustand „Initialisieren und Aktualisieren“ (im Englischen mit „IR“ abgekürzt) den Startzustand darstellt. Der Komprimierer verbleibt in jedem Zustand solange, bis er davon ausgehen kann, dass der Dekomprimierer genügend Informationen gesammelt hat um in den nächsten Zustand mit höherer Kompression zu wechseln. Bei Fehlern wird jeweils der vorhergehende Zustand mit geringerer Kompression gewählt.

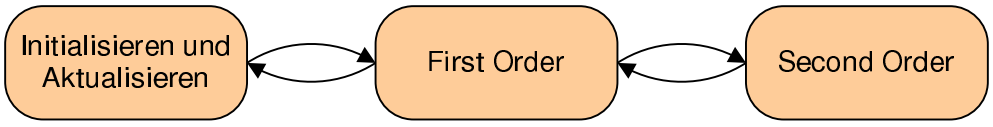


Abbildung 1: Die Zustände eines Komprimierers.

Initialisieren und Aktualisieren: Das Ziel dieses Zustandes ist entweder das Initialisieren der statischen Teile des Kontextes oder die Behebung von schwerwiegenden, aufeinanderfolgenden Fehlern. In diesem Zustand werden die Informationen unkomprimiert übertragen.

First Order: Dieser Zustand wird vom Komprimierer betreten um entweder geringe Unregelmäßigkeiten des statischen Kontextes effizient zu übermitteln oder um die Änderungsmuster dynamischer Felder bezogen auf die Sequenznummer zu aktualisieren.

Second Order: In diesem Zustand ist die Komprimierung optimal. Werden jedoch im Paketfluss gewisse Abweichungen vom festgelegten Muster festgestellt, muss der Zustand verlassen und das Muster aktualisiert werden.

3.2 Zustände des Dekomprimierers

Der Dekomprimierer fängt stets im Zustand „kein Kontext“ an und schaltet, sobald genügend Informationen vorliegen, sukzessiv in höhere Zustände weiter. Ist der Zustand „vollständiger Kontext“ einmal erreicht, behält der Dekomprimierer diesen typischerweise für die Dauer der gesamten Übertragung bei.

Abbildung 2: Die Zustände eines Dekomprimierers.

Kein Kontext: Der Dekomprimierer hat bisher kein einziges Paket erfolgreich dekomprimiert. Sobald ein Initialisierungspaket mit statischen und dynamischen Informationen vorliegt, kann der Dekomprimierer in höhere Zustände schalten.

Statischer Zustand: Dieser Zustand wird vom Dekomprimierer prinzipiell nur dann gewählt, wenn im Zustand „vollständiger Kontext“ wiederholt Fehler auftraten bzw. auftreten. Nach dem Empfang des ersten gültigen „First Order“-Paketes, schaltet der Dekomprimierer wieder in den höheren Zustand zurück.

Vollständiger Kontext: In diesem Zustand verbleibt der Dekomprimierer solange, bis wiederholt Fehler auftreten.

3.3 Mögliche ROHC-Arbeitsmodi

Das ROHC-Framework lässt drei mögliche Arbeitsmodi zu, deren Funktionsweise sich stark an der Einsatzumgebung orientiert. Jede Übertragung beginnt im unidirektionalen Modus und kann später, in Abhängigkeit von den zur Verfügung stehenden Ressourcen in einen beliebigen anderen Betriebsmodus wechseln. Diese Wechsel sind generell während der gesamten Übertragung möglich.

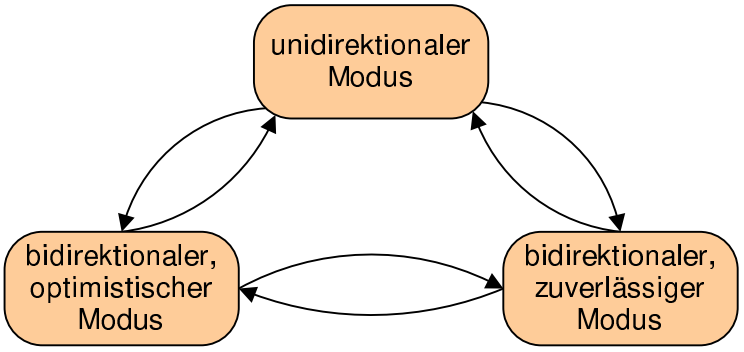


Abbildung 3: Die drei möglichen Arbeitsmodi von ROHC.

Unidirektionaler Modus: In diesem Modus werden die Pakete ausschließlich in die Hinrichtung gesendet. Dies ermöglicht zwar auch einen Betrieb über Verbindungen, die keinen Rückkanal zur Verfügung stellen, schränkt jedoch die Zuverlässigkeit und Effizienz von ROHC ein. Die Zustandsübergänge hängen in diesem Modus einzig von Timern und Unregelmäßigkeiten im Headermuster ab.

Bidirektionaler, optimistischer Modus: Dieser Modus ähnelt dem unidirektionalen. Hier wird jedoch ein Rückkanal verwendet um eventuell aufgetretene Fehler zu melden bzw. signifikante Kontextänderungen zu bestätigen. Auf periodische Aktualisierungen kann hier verzichtet werden.

Bidirektionaler, zuverlässiger Modus: Der zuverlässige Modus unterscheidet sich bedeutend von den beiden vorhergehenden. Er setzt einen regen Gebrauch der Rückleitung sowie eine strengere Logik auf beiden Seiten voraus. Doch genau dadurch ist es dem Modus auch bei hohen Fehlerraten möglich die Synchronisation von Komprimierer und Dekomprimierer zu gewährleistet.

4 Das ROHC-Protokoll

Für eine deterministische Kommunikation zwischen zwei beliebigen Instanzen ist ein Protokoll notwendig, welcher sowohl die Nachrichtenformate als auch die Zustandsübergangsregeln klar definiert. Ein solches Protokoll für die konkreten Instanzen Komprimierer und Dekomprimierer wurde für ROHC unter besonderer Beachtung der Schwerpunkte Effizienz und Robustheit entwickelt.

In diesem Kapitel betrachten wir den grundsätzlichen Aufbau einer Nachricht, sowie den Aufbau der einzelnen Nachrichten der Hin- und Rückrichtung. Die beiden Richtungen wollen wir wie folgt definieren: „Hinrichtung“ sei die Richtung vom Komprimierer zum Dekomprimierer; die „Rückrichtung“ entsprechend umgekehrt.

4.1 Allgemeiner Paketaufbau

Eine ROHC Nachricht wird durch ein Paket repräsentiert und ist in ROHC grundsätzlich wie in der Abbildung 4 aufgebaut. Es ist zu beachten, dass die kursiv dargestellten Komponenten optional sind.

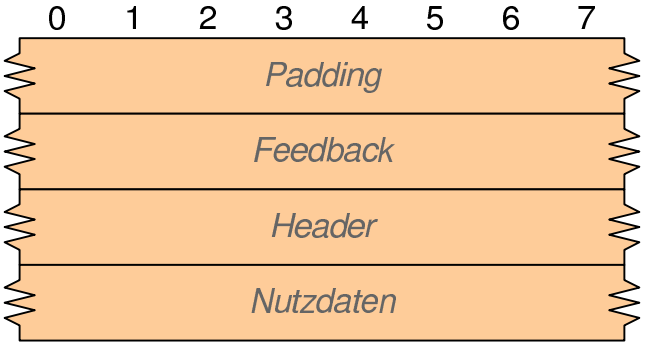


Abbildung 4: Der allgemeine Aufbau eines ROHC-Pakets.

Padding: Ein oder mehrere Padding-Oktette, welchen entweder der Header oder das Feedback folgen muss.

Feedback: Feedback Elemente beginnen immer mit dem Feedback-Pakettyp Identifikator und beinhalten Informationen für den Komprimierer (siehe auch Abschnitt 4.2.1).

Header: Der Header ist entweder profilspezifisch oder ein IR- bzw. IR-DYN-Header (siehe auch Abschnitt 4.3).

Nutzdaten: Repräsentiert die eigentlich zu übertragenden Nutzdaten.

4.2 Paketfluss: Dekomprimierer → Komprimierer

Wie bereits unter Kapitel 3 betrachtet, verfügt ROHC über mehrere Arbeitsmodi, von welchen zwei bidirektional sind. Das bedeutet, dass es in diesen Modi neben einem Hin- auch ein Rückkanal existiert. Dabei können beide Kanäle sowohl Nutz- als auch Feedbackdaten übertragen. Überträgt ein Kanal Nutz- und Feedbackdaten gleichzeitig, so wird dies als „Piggybacking“ der Feedbackdaten bezeichnet, da die Feedbackinformationen so in gleichen Nachrichten wie die Nutzdaten verschickt werden.

In folgenden Abschnitten betrachten wir zunächst den allgemeinen Aufbau der Feedbackeinheiten und später einige konkrete Beispiele.

4.2.1 Feedback-Pakete

Wie eben gesehen, müssen sich die Feedbackinformationen sowohl per Piggybacking als auch einzeln effizient transportieren lassen. Diese Voraussetzung stellt damit auch gewisse Bedingungen an den Aufbau einer Feedbackeinheit. Die Nachrichten dürfen keine redundanten Informationen tragen und doch müssen sie möglichst einfach zu parsen sein.

Der generelle Aufbau der Feedbackpakete ist in der Abbildung 5 dargestellt. Die kursiv angezeigten Komponenten sind optional.

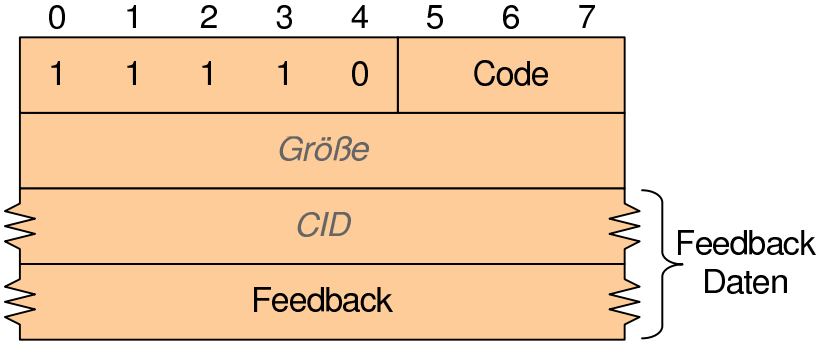


Abbildung 5: Der vereinfachte Aufbau eines Feedback-Paketes.

Code: Gibt die Größe der „Feedback Daten“-Einheit in Byte an. Falls „Code“ = 0 ist, wird das Feld „Größe“, für die Größenangabe verwendet.

Größe: Dieses Feld ist optional und gibt im Falle, dass „Code“ = 0 ist, die Größe der „Feedback Daten“-Einheit in Byte an.

Feedback Daten: Diese Einheit beinhaltet die eigentlich wichtige Feedbackinformation (siehe auch Abschnitt 4.2.2).

4.2.2 Die „Feedback Daten“-Einheit

Der für den Komprimierer entscheidende Teil der Feedbacknachricht wird in dieser Einheit übertragen, die ihrerseits aus folgenden Feldern besteht:

CID: Werden mehrere Streams im Übertragungskanal verwendet, so identifiziert das Feld einen Stream aus der Menge. Anderenfalls ist das Feld nicht vorhanden.

Feedback: Kann zwei mögliche Formen, wie in den Abbildungen 6 und 7 dargestellt, annehmen.

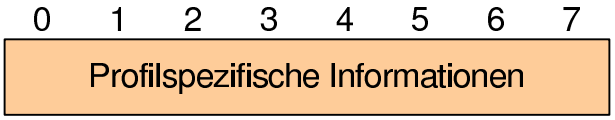


Abbildung 6: FEEDBACK-1

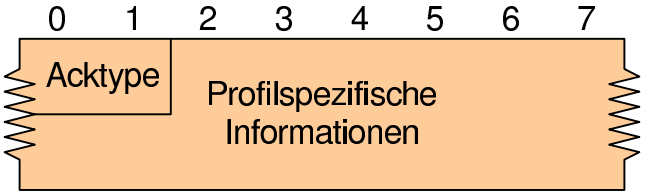


Abbildung 7: FEEDBACK-2 (mindestens 2 Byte)

Profilspezifische Informationen: Beinhaltet Informationen, die von Profil zu Profil unterschiedlich sein können.

Acktype: Dieses Feld kann einen der folgenden Werte annehmen:

ACK: Signalisiert die erfolgreiche Dekomprimierung eines Pakets. Dies bedeutet gleichzeitig, dass der Kontext mit hoher Wahrscheinlichkeit auf dem neuesten Stand ist.

NACK: Zeigt an, dass der dynamische Kontext nicht mehr synchron ist. Ein NACK wird generiert, nachdem mehrere aufeinander folgende Pakete nicht erfolgreich dekomprimiert werden konnten.

STATIC-NACK: Wird immer dann versendet, wenn der statische Kontext des Dekomprimierers entweder nicht gültig oder noch nicht initialisiert ist.

4.2.3 Beispiele

Hier betrachten wir einige Feedback-Beispielpakete. Sie sollen für ein RTP-Paket mit der Sequenznummer 17, der im bidirektionalen, zuverlässigen Modus übertragen wurde und dem Stream mit der Kontext-ID („CID“) 8 angehört ein ACK signalisieren.

In der Abbildung 8 handelt es sich um ein Paket des Typs „FEEDBACK-2“. Der Pakettyp zeigt ein Feedbackpaket mit „Code“ = 3 an. Das heißt, dass diesem Oktett drei weitere folgen, die an den Komprimierer übergeben werden. „ADD_CID“ deutet darauf hin, dass eine Kontext-ID angegeben wird, in unserem Fall „CID“ = 8. „ACKTYPE“ = 0 meldet ein ACK, „MODE“ = 3 entspricht dem bidirektionalen, zuverlässigen Modus. Die restlichen Daten beinhalten die Sequenznummer des entsprechenden Paketes.

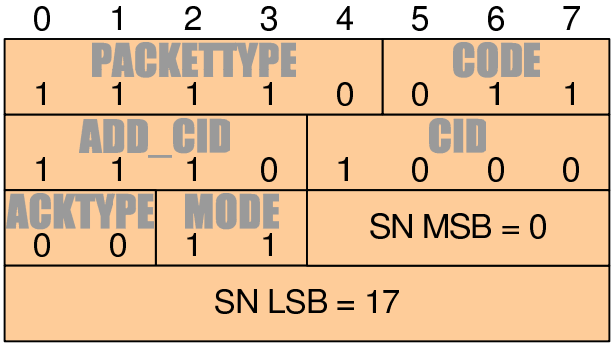


Abbildung 8: Beispiel.

Nun nehmen wir genau das gleiche Beispiel, wählen aber den Typ „FEEDBACK-1“. Das Ergebnis ist in der Abbildung 9 zu sehen. Der Komprimierer bekommt in diesem Fall („Code“ = 2) nur die unteren zwei Oktette.

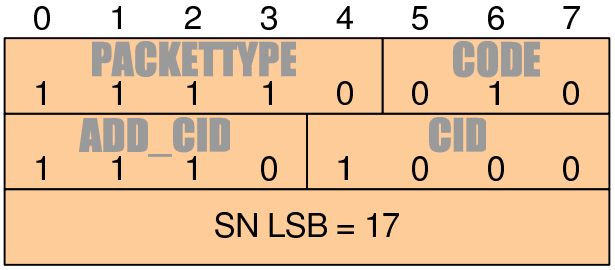


Abbildung 9: Beispiel.

Im letzten Beispiel - dargestellt auf der Abbildung 10 - betrachten wir den Fall, dass die Kontext-ID („CID“) des Streams die Null ist. Es sind demnach keine weiteren Streams vorhanden, die über die gleiche Verbindung übertragen werden. Im Sinne der Optimierung kann also das ganze „ADD_CID“ Oktett weggelassen werden. Es werden somit nur zwei Byte für eine Quittung benötigt.

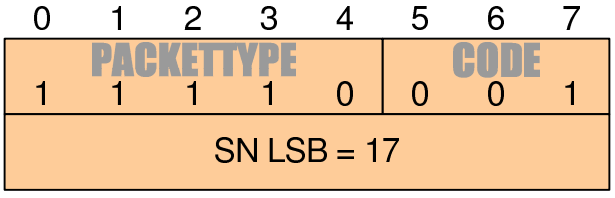


Abbildung 10: Beispiel.

4.3 Paketfluss: Komprimierer → Dekomprimierer

ROHC komprimiert und überträgt die Nutzdaten in einer Interaktion von zwei Zustandsautomaten (siehe auch Kapitel 3). Die Interaktion vollzieht sich durch einen Nachrichtenaustausch zwischen Komprimierer und Dekomprimierer, wobei der Hinfluss vorgeschrieben und der Rückfluss optional ist. Nachdem wir in Abschnitt 4.2 den Rückpfad betrachtet haben, wenden wir uns nun dem Hinpfad zu.

ROHC verfügt über eine ganze Reihe von Paketen, die auf ihre Aufgabe optimiert sind und die wir im Folgenden betrachten.

Pakete vom Typ 0: Der Zweck dieser Pakete besteht lediglich in der Übermittlung der Nutzdaten sowie der zugehörigen RTP-Sequenznummer. Die erreichte Kompression ist mit diesen Paketen optimal, da lediglich ein Oktett zusätzlich zu den Nutzdaten übertragen werden muss.

Pakete vom Typ 1: Diese Pakete werden verwendet, wenn die Anzahl der Bits für die Darstellung der Sequenznummer nicht mehr ausreicht oder die Parameter der Funktionen für die Berechnung des RTP-„Timestamp“ bzw. der IP-„ID“ aus der Sequenznummer sich geändert haben.

Pakete vom Typ 2: Dieser Pakettyp kann verwendet werden um Funktionen für die Berechnung beliebiger dynamischer Felder aus der Sequenznummer zu ändern.

IR-Pakete: IR-Pakete werden überwiegend während der Initialisierungsphase verwendet, ihre Benutzung ist jedoch nicht auf diese beschränkt. Sie initialisieren bzw. aktualisieren den gesamten statischen Teil des Kontextes, optional können sie auch den dynamischen Teil beinhalten. Für weitere Informationen siehe auch Abschnitt 4.3.1.

IR-DYN-Pakete: Mit diesen Paketen wird der dynamische Teil des Kontextes übertragen (bspw. die Parameter einer nicht-konstanten von der Sequenznummer abhängigen Funktion). Siehe Abschnitt 4.3.2 für weitere Informationen.

4.3.1 IR-Pakete

IR-Pakete werden insbesondere während der Initialisierungsphase verwendet um einen Stream mit einem Profil zu assoziieren und dessen Kontext zu initialisieren. Die Initialisierung kann sowohl auf den statischen als auch den dynamischen Teil des Kontextes angewandt werden. Allgemein lassen sich die IR-Pakete auch zur Aktualisierung des Kontextes oder einer Assoziierung des Streams mit einem anderen Profil verwenden.

Ein IR-Paket ist grundsätzlich wie in der Abbildung 11 aufgebaut (optionale Felder sind kursiv dargestellt).

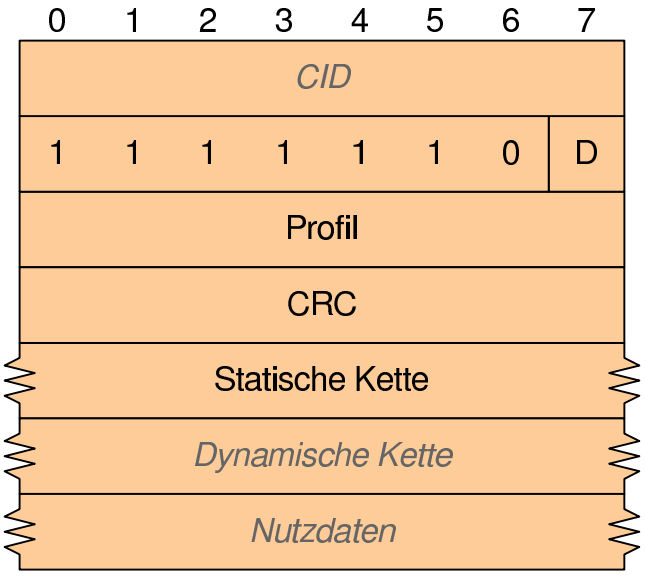


Abbildung 11: Der vereinfachte Aufbau eines IR-Paketes

CID: Werden mehrere Streams im Übertragungskanal verwendet, so identifiziert das Feld einen Stream aus der Menge. Anderenfalls ist das Feld nicht vorhanden.

D: Ist „D“ = 1, so enthält das IR-Paket auch eine dynamische Kette.

Profil: Der vorher durch die „CID“ angegebene Stream wird mit dem Profil assoziiert, dessen Bezeichner durch seine acht niederwertigsten Bits in diesem Feld angegeben ist.

CRC: Acht bittiger CRC (siehe auch Kapitel 5).

Statische Kette: Sie beinhaltet eine Kette von statischen Headerinformationen.

Dynamische Kette: Die „Dynamische Kette“ ist präsent, wenn das Feld „D“ = 1 gesetzt ist. In diesem Fall beinhaltet das Feld dynamische Headerinformationen.

Nutzdaten: Dieses Feld enthält die eigentlichen Nutzdaten, falls diese bereits vorhanden sind.

4.3.2 IR-DYN-Pakete

Die IR-DYN-Pakete können sowohl in der Initialisierungsphase als auch zur Laufzeit eingesetzt werden, um den dynamischen Teil des Kontextes auf den neuesten Stand zu bringen. Dies kann sowohl eine Routine zur Fehlerbehebung sein, als auch eine umfassende Kontextaktualisierung. Diese Pakete können ebenfalls, ähnlich den IR-Paketen (siehe Abschnitt 4.3.1) Streams mit neuen Profilen assoziieren.

IR-DYN-Pakete sind prinzipiell den IR-Paketen sehr ähnlich, beinhalten aber keine „Statische Kette“. Der vereinfachte Aufbau ist in der Abbildung 12 dargestellt (optionale Felder werden kursiv angezeigt).

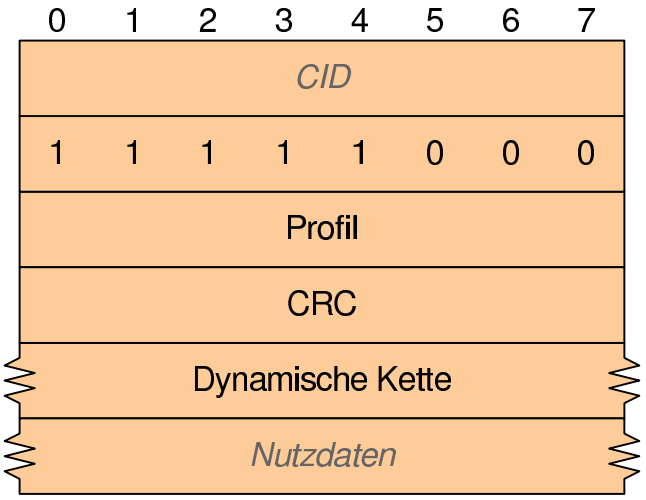


Abbildung 12: Der vereinfachte Aufbau eines IR-DYN-Paketes

CID: Bei mehr als einem Stream im Übertragungskanal wird dieses Feld verwendet, um so einen Stream aus der Menge zu identifizieren. Bei nur einem Stream ist das Feld „CID“ nicht vorhanden.

Profil: Der durch die „CID“ angegebene Stream wird mit dem Profil assoziiert, dessen Bezeichner durch seine acht niederwertigsten Bits in diesem Feld angegeben ist.

CRC: Acht bittiger CRC (siehe auch Kapitel 5).

Dynamische Kette: Die „Dynamische Kette“ beinhaltet dynamische Headerinformationen, welche es zu initialisieren bzw. zu aktualisieren gilt.

Nutzdaten: Dieses Feld - falls vorhanden - transportiert die eigentlichen Nutzdaten.

5 Cyclic Redundancy Check

Das CRC ist ein weit verbreitetes und häufig eingesetztes Verfahren um Beschädigungen von Daten mit hoher Wahrscheinlichkeit zu erkennen. Auch das ROHC Framework setzt diese Methode ein, um mögliche Fehler in übertragenen, komprimierten Headern festzustellen. Die in ROHC verwendete Prüfsumme bezieht sich aber nicht auf die eigentlichen Nutzdaten, sondern ausschließlich auf die Headerfelder.

CRC-STATIC: Es ist allgemein bekannt, dass das CRC Verfahren zwar effizient in Hardware implementierbar ist, jedoch die Berechnung dergleichen in Software sich relativ aufwändig gestaltet. Zur Effizienzsteigerung nutzt ROHC die Tatsache aus, dass einige Headerfelder sich während der gesamten Übertragung nicht ändern. ROHC berechnet die Prüfsumme über diese als CRC-STATIC klassifizierten Felder nur ein Mal und legt das Ergebnis anschließend im Zwischenspeicher ab.

CRC-DYNAMIC: Die effektive Berechnung bezieht in ROHC nur die Felder ein, bei welchen davon auszugehen ist, dass sie sich bei jeder erneuten Erfassung ändern. Diese Felder werden in diesem Sinne als CRC-DYNAMIC klassifiziert.

Durch den Einsatz der oben beschriebenen Technik lässt sich die Komplexität der CRC Prüfsummenberechnung im Durchschnitt beim Zusammenspiel der Protokolle RTP, UDP und IP um circa 80% reduzieren. Für weitere Informationen bzgl. der Felderklassifizierung siehe auch Abschnitt 6.1.

6 Das RTP-Profil

Das RTP-Profil wurde entwickelt um die Übertragung des RTP-Protokolls sowohl über langsame als auch fehlerbehaftete Verbindungen zu ermöglichen. Dies ist vor allem deshalb wichtig, da das RTP-Protokoll ein Ende-zu-Ende-Dienst mit einigen für den Echtzeitbetrieb benötigten Funktionen zur Verfügung stellt. Fast selbstverständlich ist hierbei die besondere Wichtigkeit, die die Minimierung vom übertragenen Overhead für ein Echtzeit-Szenario darstellt. Heutzutage, wäre ohne der Aussicht einer redundanzfreien Datenübertragung eine vernünftige, echtzeitfähige Kommunikation mit einer drahtlos angebundenen Station kaum möglich.

In diesem Kapitel betrachten wir die einzelnen Schritte, die für den Entwurf und die Durchführung einer effizienten und robusten Komprimierung erforderlich sind.

Das in der Profilbezeichnung auftretende Protokoll RTP bezeichnet an der Stelle den gesamten RTP/UDP/IP-Protokollstapel. Für weitere Informationen zu den einzelnen Protokollen siehe auch die nun folgenden Abschnitte sowie [DeHi98], [Post80] und [SCFJ96].

6.1 Klassifizierung der Feld-Änderungsmuster

Wie bereits aus „Paketkopfkomprimierung in drahtlosen Zugangsnetzen“ bekannt, wird die Komprimierung erst durch die Tatsache ermöglicht, dass sich die meisten Headerfelder nicht zufällig von einem zum nächsten Paket unterscheiden. Vielmehr lässt sich in den meisten Fällen ein statisches oder ein gut vorhersehbares Verhalten feststellen. Um eine effiziente Komprimierung gewährleisten zu können, muss zuerst das Verhalten einzelner Headerfelder untersucht und anschließend klassifiziert werden. Auf dieser Auswertung beruhend, betrachten wir letztlich die möglichen Vorgehensweisen um die gewünschte Komprimierung zu erreichen.

Für eine sinnvolle Kompression benötigen wir eine exakte Unterteilung der Headerfelder in Klassen, die das Verhalten einzelner Felder möglichst präzise repräsentieren. Später können wir für die einzelnen Klassen die optimale Komprimierungsmethode bestimmen. Zu diesem Zwecke definieren wir hier einige wichtige Klassen.

Es folgt eine vereinfachte Liste:

Statische Felder: Im folgenden sind Klassen für Felder aufgelistet, die sich während der gesamten Übertragungszeit nicht ändern. Diese Felder werden üblicherweise als „Statische Kette“ eines IR-Paketes während der Initialisierungsphase übertragen (siehe auch Abschnitt 4.3.1).

- **INFERRED:** Diese Klasse beinhaltet alle Felder, deren Werte sich von Werten anderer Felder ableiten lassen und somit vom Komprimierer außer Acht gelassen werden können.
- **STATIC:** Die Felder dieser Klasse, behalten ihre Werte durchgehend, während der gesamten Übertragung. Somit brauchen sie lediglich ein Mal übertragen zu werden.
- **STATIC-KNOWN:** Hierunter fallen alle Felder, deren Inhalte wohlbekannt sind und aus diesem Grund nicht übertragen werden brauchen.
- **CHANGING-STATIC (CS):** Hier werden alle Felder zusammengefasst, die sich zwar generell ändern können, unter bestimmten Annahmen sich allerdings wie Felder der Klasse „STATIC“ verhalten.

Dynamische Felder: Hier sind Klassen für die Felder aufgelistet, die sich im Laufe der Übertragung verändern können. Diese Felder werden normalerweise als „Dynamische Kette“ eines IR- oder IR-DYN-Paketes übertragen (siehe auch Abschnitt 4.3).

SEMISTATIC: Unter dieser Klasse sind Felder zu finden, welche sich die meiste Zeit von den Feldern der Klasse „STATIC“ nicht unterscheiden, gelegentlich jedoch ihren Wert für eine bestimmte Zeit ändern können.

RARELY-CHANGING (RC): In dieser Klasse befinden sich Felder, die ab und zu Ihre Werte ändern und die neuen Werte auch behalten.

ALTERNATING: Dazu gehören Felder, die zwischen einer kleinen Zahl von möglichen Werten alternieren.

IRREGULAR: Für diese Felder können keine sinnvollen Vorhersagen getroffen werden.

6.1.1 Klassifizierung des IPv6-Headers

Das IP-Protokoll liegt jeder von ROHC erfassten Datenübertragung auf der Vermittlungsschicht zugrunde. In der Abbildung 13 ist der generelle Aufbau eines IPv6-Headers dargestellt. Für weitere Informationen bezüglich des IPv6-Protokolls siehe auch [DeHi98].

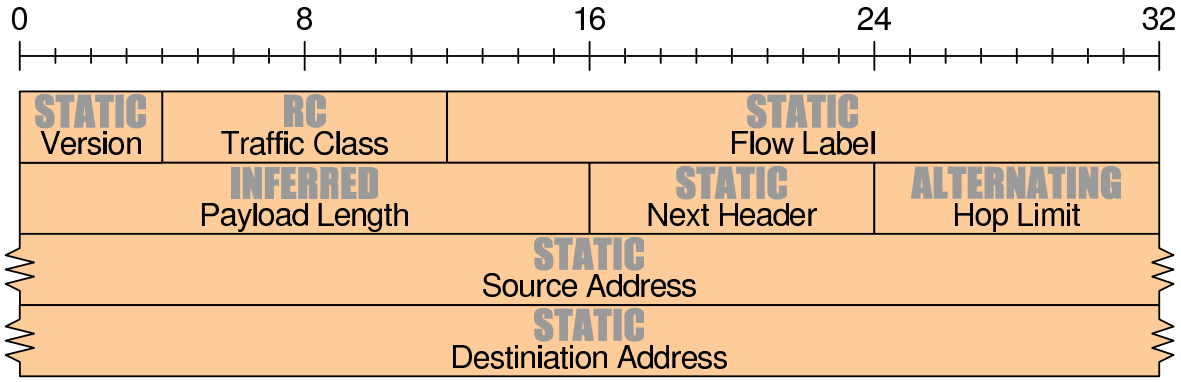


Abbildung 13: Der vereinfachte Aufbau eines IPv6-Headers.

Eine kurze Beschreibung der dargestellten Felder:

Version: Zeigt die benutzte Protokoll-Version an, bei IPv6 ist „Version“ = 6.

Traffic Class: Gibt die Klasse der über diese Verbindung transportierten Daten an.

Flow Label: Bezeichnet die zu einem IPv6 Stream gehörende Pakete.

Payload Length: Gibt die Größe der Nutzdaten des IP-Paketes an.

Next Header: Bestimmt den Typ des Headers, der dem IPv6-Header direkt folgt.

Hop Limit: Wird bei jedem Hop um eins verringert. Ist der Wert = 0, so wird das Paket verworfen.

Source Address: Die IP-Adresse des Absenders.

Destination Address: Die IP-Adresse des Zielsystems.

Es ist schnell zu erkennen, dass die überwiegende Mehrheit der Felder als „STATIC“ bzw. „INFERRED“ markiert ist. Diese statischen Felder werden deshalb typischerweise primär während der Initialisierungsphase übertragen und verbleiben danach im statischen Teil des Kontextes. Weitere Informationen zu den einzelnen Kompressionsstrategien finden sich im Abschnitt 6.2 und unter [Borm01].

6.1.2 Klassifizierung des UDP-Headers

Das UDP-Protokoll ist ein einfaches verbindungsloses Protokoll der Transportschicht, das für die Übermittlung der RTP-Dateneinheiten zuständig ist. Sein Aufbau sowie die Klassifizierung seiner Felder, lassen sich der Abbildung 14 entnehmen. Für weitere Informationen bezüglich des UDP-Protokolls siehe auch [Post80].

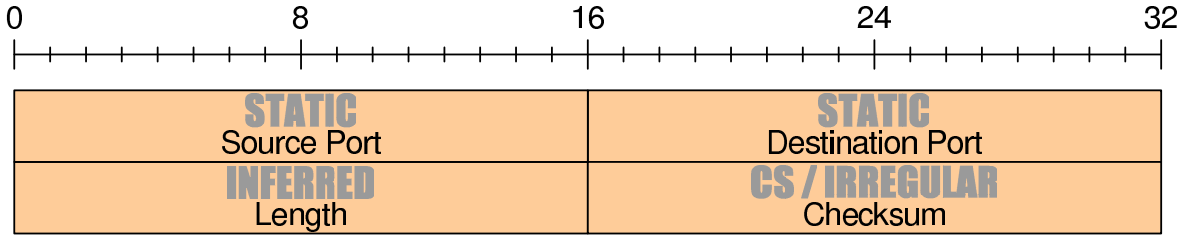


Abbildung 14: Der Aufbau eines UDP-Headers.

Eine kurze Beschreibung der dargestellten Felder:

Source Port: Bestimmt den Port des Absenders.

Destination Port: Identifiziert den Empfangsport auf dem Zielsystem.

Length: Gibt die Länge des gesamten UDP-Paketes an.

Checksum: Prüfsumme über den IP-Pseudoheader, den tatsächlichen UDP-Header und die eigentlichen Nutzdaten.

Auch UDP-Pakete bestehen zum größeren Teil aus Feldern, welche als „STATIC“ bzw. „INFERRED“ klassifiziert werden können. Lediglich das Feld „Checksum“ lässt sich nicht immer komprimieren. Sollte die Option der UDP-Prüfsumme aktiviert sein, so wird das Feld als „IRREGULAR“ markiert und mit jedem Paket verschickt. Anderenfalls wird es als „CHANGING-STATIC“ gekennzeichnet und nur während der Initialisierungsphase übertragen. Danach verbleibt es mit den anderen als „STATIC“ markierten Feldern im statischen Bereich des Kontextes. Genaue Informationen über die Kompressionsstrategien finden sich im Abschnitt 6.2 und unter [Borm01].

6.1.3 Klassifizierung des RTP-Headers

Das RTP-Protokoll bietet einen Ende-zu-Ende Dienst mit darüberhinausgehenden Funktionen an, die für einen Echtzeitbetrieb unabdingbar sind. Die zur Verfügung gestellten Dienste beinhalten beispielsweise die Identifikation der Nutzdaten, die Zeitstempelfunktion und die Überwachung der ankommenden Pakete. Meistens wird RTP zusammen mit UDP verwendet, um von dessen Multiplextechniken und dessen Prüfsummenfunktion zu profitieren. Eine vereinfachte Darstellung des RTP-Headers findet sich auf der Abbildung 15.

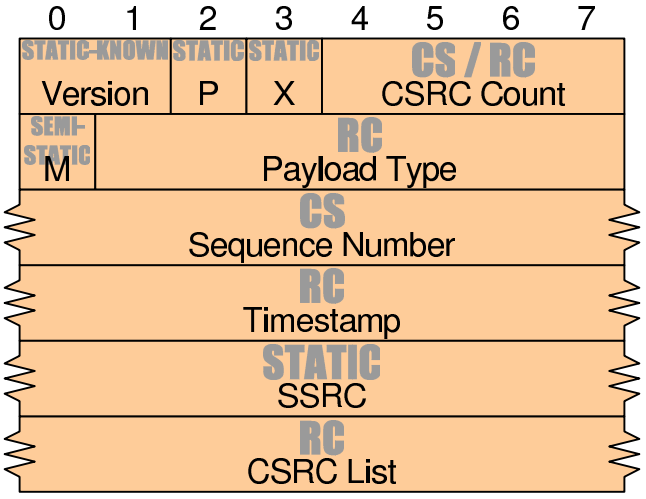


Abbildung 15: Der vereinfachte Aufbau eines RTP-Headers.

Eine kurze Beschreibung der dargestellten Felder:

Version: Bestimmt die Version des Protokolls, wobei alle gängigen RTP-Versionen eine Zwei in diesem Feld tragen.

Padding (P): Gibt an, ob Padding-Oktette verwendet werden.

Extension (X): Falls eine Paketkopferweiterung verwendet wird, zeigt es dieses Feld an.

CSRC Count: Bestimmt die Anzahl der Quellen, welche diesem Paket Daten beisteuern.

Marker (M): Die Interpretation dieses Feldes, hängt von dem verwendeten RTP-Profil ab.

Payload Type: Legt den Inhalt der Nutzdaten und deren Interpretation seitens der Anwendung fest.

Sequence Number: Identifiziert jedes einzelne Paket, indem die Sequenznummer für jedes neue Paket um eins erhöht wird.

Timestamp: Vermerkt die Zeit, in der das erste Oktett der Nachricht erstellt wurde.

SSRC: Enthält eine eindeutige Kennung des Senders.

CSRC List: Identifiziert die einzelnen Datenquellen.

Bei RTP gestaltet sich die Komprimierung um einiges aufwändiger als bei den beiden vorhergehenden Protokollen. Es lassen sich aber auch hier die meisten Felder nach einer genauen Untersuchung ihrer möglichen Wertebereiche effizient komprimieren. Genaue Informationen darüber finden sich im Abschnitt 6.2 und unter [Borm01].

6.2 Kompressionsstrategien

In diesem Abschnitt beschäftigen wir uns mit der Zuteilung einzelner Felder zu bestimmten Komprimierungsstrategien. Wir haben im vorhergehenden Abschnitt 6.1 gesehen, dass verschiedene Felder, verschiedener Header zu gleichen Klassen zusammengefasst werden können. Das bedeutet auch gleichzeitig, dass die verschiedenen Klassen von Feldern mit unterschiedlicher Häufigkeit übertragen werden müssen. Manche Klassen brauchen erst gar nicht übermittelt zu werden. Logischerweise ergeben sowohl die Einschränkung des möglichen Wertebereiches als auch die Reduzierung der Übertragungshäufigkeit eines Feldes eine höhere Komprimierung.

Im Folgenden betrachten wir anhand des eingeführten RTP-Profils einige Strategien mit diversen Beispielen.

6.2.1 Strategie: Nicht Senden

Alle Felder mit ausschließlich wohlbekannten Werten, sowie alle Felder deren Werte von Werten anderer Felder abhängen, brauchen nicht gesendet zu werden. Diese Strategie könnte bspw. auf die Klassen „INFERRED“ und „STATIC-KNOWN“ angewandt werden. Einige Beispielfelder:

- IPv6 - „Payload Length“
- RTP - „Version“

6.2.2 Strategie: Nur Initialisieren

Felder, die während der gesamten Übertragungszeit ihre Werte nicht ändern, brauchen nur einmal übermittelt und dann zwischengespeichert zu werden. Diese Strategie lässt sich gut auf die Klasse „STATIC“ anwenden. Hier einige konkrete Beispiele:

- IPv6 - „Flow Label“
- UDP - „Source Port“
- RTP - „Padding“

6.2.3 Strategie: Initialisieren und änderbar lassen

Diese Strategie lässt sich am Besten mit der „RARELY-CHANGING“ Klasse einsetzten. Felder dieser Klasse werden in der Initialisierungsphase übertragen und verbleiben im statischen Bereich des Kontextes. Dennoch muss es auch später im laufenden Betrieb möglich sein, den Inhalt dieser Felder zu aktualisieren. Einige Beispiele für solche Felder:

- IPv6 - „Next Header“
- IPv6 - „Traffic Class“
- RTP - „CSRC Counter“

6.2.4 Strategie: Robustheit garantieren

Einige Felder der Klasse „CHANGING-STATIC“ verhalten sich wie Zähler mit einer festen Schrittweite. Bei diesen Feldern gilt es sicherzustellen, dass mögliche Paketverluste während der Übermittlung tolerierbar bleiben. Ein Beispiel dafür:

- RTP - „Sequence Number“

6.2.5 Strategie: Senden, sobald Daten vorhanden

Felder der Klasse „IRREGULAR“ lassen sich generell nicht komprimieren. Sie werden deshalb an den Dekomprimierer geschickt, sobald sie dem Komprimierer vorliegen. Ein Beispiel:

- UDP - „Checksum“ (wenn die Prüfsummenfunktion aktiviert ist)

7 Fazit

Wir haben gesehen, dass die Redundanz stets einen hohen Anteil in den von uns verschickten Daten ausmacht. Bei der Übertragung von Audiosignalen können wir davon ausgehen, dass aufgrund kleiner Pakete, die dafür umso häufiger verschickt werden müssen, das Verhältnis von Overhead- zu Nutzdatenanteil gewaltig ist. Wir haben ebenfalls einige Mechanismen kennen gelernt, die uns dabei behilflich sein können die enorme Overheadflut zumindest auf dem letzten „Hop“ einzudämmen. Wir können allgemein davon ausgehen, dass die dafür benötigte Rechenleistung bereits in allen aktuellen Mobilstationen vorhanden ist. Es gibt sogar erste Hinweise darauf, dass einige Netzbetreiber bereits mit dem ROHC Framework experimentieren und dennoch ist es bisher nicht abzusehen, wann die beschriebene Technik die Marktreife erreichen wird.

Literatur

[Borm01] C. Bormann. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. RFC 3095 (Standards Track), July 2001.

[DeHi98] S. Deering und R. Hinden. Internet Protocol, Version 6 (IPv6). RFC 2460 (Standards Track), December 1998.

[JoPe04] Lars-Erik Jonsson und Ghyslain Pelletier. RObust Header Compression (ROHC): A Compression Profile for IP. RFC 3843 (Standards Track), June 2004.

[PeJo04] Ghyslain Pelletier und Lars-Erik Jonsson. RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP). INTERNET-DRAFT (draft-ietf-rohc-tcp-08.txt), October 2004.

[Pell04] Ghyslain Pelletier. RObust Header Compression (ROHC): Profiles for UDP-Lite. INTERNET-DRAFT (draft-ietf-rohc-udp-lite-04.txt), June 2004.

[Post80] J. Postel. User Datagram Protocol. RFC 768 (Standards Track), August 1980.

[SCFJ96] H. Schulzrinne, S. Casner, R. Frederick und V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Standards Track), January 1996.

Abbildungsverzeichnis

1 Die Zustände eines Komprimierers. 183
2 Die Zustände eines Dekomprimierers. 184
3 Die drei möglichen Arbeitsmodi von ROHC. 184
4 Der allgemeine Aufbau eines ROHC-Pakets. 185
5 Der vereinfachte Aufbau eines Feedback-Paketes. 186
6 FEEDBACK-1 187
7 FEEDBACK-2 (mindestens 2 Byte) 187
8 Beispiel. 188
9 Beispiel. 188
10 Beispiel. 188
11 Der vereinfachte Aufbau eines IR-Paketes 190
12 Der vereinfachte Aufbau eines IR-DYN-Paketes 191
13 Der vereinfachte Aufbau eines IPv6-Headers. 193
14 Der Aufbau eines UDP-Headers. 194
15 Der vereinfachte Aufbau eines RTP-Headers. 195

Kontexttransfer für robuste Paketkopfkomprimierung

Andreas Schuster

Kurzfassung

Mit der Weiterentwicklung moderner drahtloser Telekommunikationsnetze in 3GPP und 3GPP2 einhergehend, gewinnen Verfahren zur Paketkopfkomprimierung zunehmend an Bedeutung. Diese erhöhen die nutzbare Datenrate auf den kostbaren und knappen Funkfrequenzen. Die für Mobilkommunikationsnetze charakteristischen häufigen Wechsel des Zugangspunktes sorgen allerdings zwangsweise für Effizienzeinbußen bei allen Paketkopfkomprimierungsverfahren. Da beim Handover der Kontext, also der Zustand des (De-)Kompressors verlorengeht, müssen zur Etablierung eines frischen Kontexts am neuen Zugangspunkt erneut unkomprimierte Pakete übertragen werden. Um diesen Kontextverlust zu verhindern bietet sich als Lösung das Kontexttransferprotokoll CTP an, dessen Funktionsweise im folgenden näher beschrieben werden soll. Insbesondere wird auch das Zusammenspiel von CTP mit den Paketkopfkomprimierungsmechanismen von ROHC näher beleuchtet.

1 Einleitung

Die rasante Entwicklung in der mobilen Telekommunikation lässt sich an nackten Zahlen festmachen [Sta04]: So erreichte im Januar 2003 der Bestand an Mobiltelefonen den mittleren Wert von 114 Geräten pro 100 Haushalte! Die zunehmende Nutzung mobiler Kommunikationstechnologie stellt nicht nur einen großen Markt für die Netzbetreiber dar, sondern auch zunehmend große Herausforderungen, mit den teuer erkauften Funklizenzen und deren Bandbreite gut hauszuhalten, um dem großen Bedarf nachkommen zu können.

Nicht erst mit dem breiten Aufkommen allgemein finanzierbarer Funktechnologie, stellt verfügbare Bandbreite einen Kostenfaktor dar. Auch schon in den noch gar nicht so lange vergangenen Zeiten schmalbandiger Internet Einwahl-Verbindungen, stellte die möglichst effiziente Nutzung der verfügbaren Übertragungsrate ein Problem dar. Eine bereits aus diesem Anwendungsumfeld bekannte Lösung stellt die Paketkopfkomprimierung dar.

Die für den drahtgebundenen Datenverkehr entworfenen Verfahren zur Paketkopfkomprimierung eignen sich jedoch aufgrund der höheren Fehlerraten und anderen Fehlerarten nur schlecht für drahtlose Verbindungen. Glücklicherweise steht mit Robust Header Compression (ROHC) mittlerweile ein Verfahren zur Verfügung das auf diese veränderten Eigenschaften Rücksicht nimmt. Wie im Verlauf dieser Arbeit gezeigt werden wird, genügen diese Maßnahmen allerdings nicht um in allen Situationen zufriedenstellende Ergebnisse zu erzielen. Insbesondere die durch Zugangspunktwechsel verursachten Kontextverluste schmälern den durch ROHC erzielbaren Bandbreitengewinn beträchtlich.

1.1 Paketkopfkomprimierung mit Robust Header Compression (ROHC)

Im Folgenden wird eine kurze Einführung in die Funktionsweise von Robust Header Compression (ROHC) [BBDF+01] gegeben. Dabei liegt besonderes Augenmerk auf den für die

folgenden Betrachtungen wichtigen Mechanismen. Für weiterführende Erklärungen sei auf die Arbeiten von [Wals05] und [Fein05] verwiesen.

Paketkopfkomprimierungsmechanismen basieren auf der Erkenntnis, dass sich die meisten Paketkopffelder eines Paketstroms während einer Sitzung nicht ändern, wie etwa Ziel- und Quelladresse, oder dies nach einem vorhersehbaren Muster tun, wie z.B. Sequenznummern. Anstatt diese Paketkopffelder jetzt mit jedem Paket redundant mitzusenden werden nur noch die Änderungsmuster übertragen.

Dieses Vorgehen erfordert allerdings das die beteiligten Knoten sich die redundanten Informationen merken, also einen Kontext für jeden komprimierten Paketstrom vorhalten.

ROHC bietet Paketkopfkomprimierung für eine Reihe von möglichen Protokollstapeln an, in der ROHC-Terminologie spricht man von Profilen, z.B. für IP/UDP/RTP- oder IP/ESP-Ströme. Je nach Profil werden unterschiedliche Paketkopffelder im Kontext auf den Knoten abgespeichert und nicht mehr bei jedem Paket mitgesendet.

Abbildung 1: ROHC Kompressor Zustände

Der ROHC-Kompressor arbeitet in einem von 3 möglichen Zuständen, wie in Abbildung 1 dargestellt. Er startet immer im Zustand „Initialization and Refresh“, in dem die Pakete unkomprimiert an den Dekompressor gesendet werden. Entscheidet der Kompressor, entweder anhand von Feedback-Paketen oder durch statistische Erwägungen, das der Dekompressor genügend Informationen gesammelt hat um einen statischen Kontext aufzubauen, so wechselt der Kompressor in den nächsthöheren Zustand „First Order“. In diesem Zustand werden die statischen Paketkopffelder, wie Ziel- und Quelladresse, nicht mehr mitübertragen und nur noch gelegentlich Informationen über die sich dynamisch ändernden Felder übertragen. Sobald der Kompressor davon ausgehen kann, dass der Dekompressor genügend Informationen über den vollen Kontext besitzt, wechselt er in den höchsten Zustand „Second Order“, in dem schließlich die höchste Kompressionsrate erzielt wird.

Abbildung 2: ROHC Dekompressor Zustände

Der Zustandsautomat des ROHC-Dekompressors ist in Abbildung 2 aufgezeichnet. Während einer Verbindung sollten diese Zustände analog zu denen des Kompressors wechseln. Trifft ein Paket ein, welches vom Kompressor in einem höheren Zustand als dem Zustand des Dekompressors komprimiert wurde kann der Dekompressor das Paket aufgrund mangelnder Kontextinformationen nicht wiederherstellen und muss es verwerfen.

Die konkreten Ereignisse, die zu Zustandsübergängen zwischen den einzelnen Zuständen der beiden Automaten führen, hängen vom momentan aktiven ROHC-Modus ab. Die folgenden drei Modi, zwischen denen während einer Verbindung gewechselt werden kann, sind definiert: Der unidirektionale Modus (U-Mode) für Verbindungen ohne Feedback-Kanal, der bidirektionale optimistische Modus (O-Mode) und der bidirektionale zuverlässige Modus (R-Mode).

1.2 Das Kontexttransferprotokoll CTP

Die Notwendigkeit für ein Protokoll, das einen geordneten Kontexttransfer regelt, wie in [Kemp02] gefordert, ergibt sich aus den Eigenschaften drahtloser Zugangsnetze: Die Möglichkeit der (potentiell) mobilen Stationen, ihren Netzzugangspunkt häufig und abrupt zu wechseln stellt eine nicht zu vernachlässigende Herausforderung an die Netzinfrastruktur dar. Die mit dem Zugangspunktwechsel einhergehende Änderung des Weiterleitungspfades wird durch ein Mobilitätsprotokoll wie Mobile IPv6 [JoPA04] gewährleistet. Damit allein lassen sich die Auswirkungen des Zugangspunktwechels allerdings nicht vollständig kompensieren. Zwar werden dadurch alle Pakete über den jeweils aktuellen Zugangspunkt an das Endgerät weitergeleitet, sämtliche auf einem alten Pfad gewonnenen Zustandsinformationen, also der sogenannte Kontext, gehen jedoch verloren und müssen nach einem Zugangspunktwechsel erneut aufgebaut werden.

Um diesen Kontextverlust beim Zugangspunktwechsel zu verhindern, wurde mit dem Kontexttransferprotokoll CTP [LNPK04] die Möglichkeit zur kontrollierten Übergabe des Kontexts zwischen Netzinfrastrukturkomponenten geschaffen. Einen tieferen Einblick in die Arbeitsweise von CTP wird in Abschnitt 3 gegeben, zunächst aber sollen die Auswirkungen von Zugangspunktwechseln näher betrachtet werden.

2 Analyse der Handover-Problematik

Die Annahme, das Problem der Zugangspunktwechsel oder auch Handover als Randproblem mit geringen Auswirkungen abzutun, mag aufgrund der Tatsache, dass sich der Verlust des Kontexts immer nur kurz und zu Beginn einer neuen Zugangspunktverbindung auswirkt, naheliegend sein. Im Folgenden wird aber gezeigt, dass sie keinesfalls zutreffend ist.

Zunächst sollen dazu an einem allgemeinem Handover-Modell einige Punkte qualitativ diskutiert werden, um anschließend die Ergebnisse eines mathematischen Modells über die Auswirkungen eines Kontextverlusts auf die Performance von ROHC vorzustellen.

2.1 Allgemeines Modell

In Abbildung 3 ist ein Handover Vorgang dargestellt (im Folgenden findet die weitgehend etablierte englische Terminologie Verwendung):

Ein mobiles Endgerät MN (Mobile Node) wechselt seinen Netzzugangspunkt AR (Access Router), wobei der alte Zugangspunkt mit pAR (previous AR) und der neue mit nAR (new AR) bezeichnet ist. Während des Handovers besteht eine Verbindung zwischen dem MN und einem entfernten Kommunikationspartner, dem CN (Correspondent Node).

Während die Verbindung zwischen MN und CN über den alten Zugangspunkt pAR läuft, sammelt pAR Informationen über den Datenstrom. Ob dies implizit durch Verarbeitung der über ihn laufenden Pakete geschieht oder durch explizite Signalisierung ist für die folgende Betrachtung unerheblich. Wichtig ist soweit nur die Tatsache das auf pAR ein Kontext etabliert wurde der für die Erbringung bestimmter Dienste (z.B. QoS oder Paketkopfkomprimierung) genutzt wird.

Durch die Mobilität des MN kommt es nun irgendwann zu einem Wechsel des Zugangspunktes. Nachdem die Routenänderungen vom Mobilitätsprotokoll vorgenommen wurden, treffen alle Pakete für den MN über den neuen Zugangspunkt nAR ein. Der auf pAR etablierte Kontext ist jedoch nAR nicht bekannt.

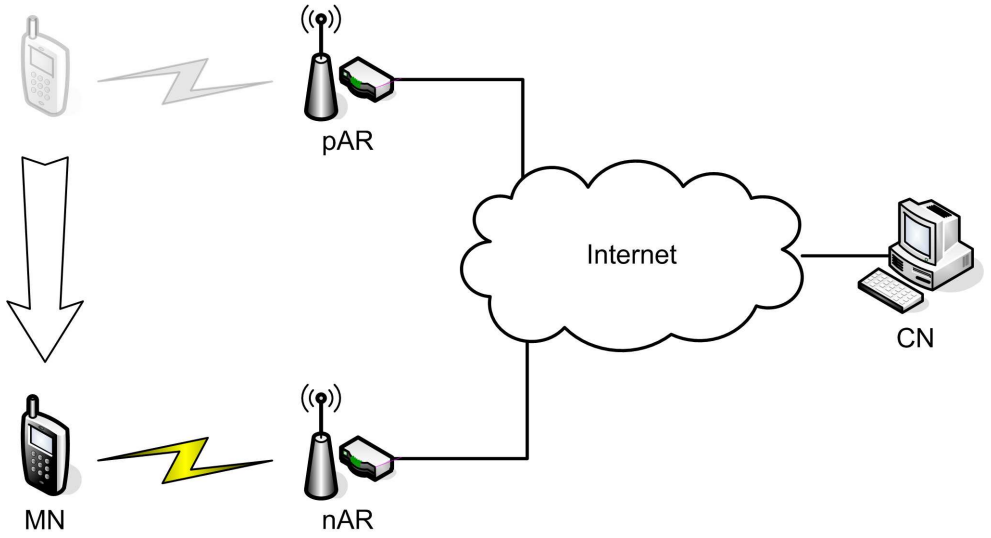


Abbildung 3: Ein mobiles Endgerät wechselt seinen Netzzugangspunkt

Dies hat zur Folge das je nach Art des Dienstes entweder der auf den Kontext angewiesene Dienst bis zur erneuten impliziten Etablierung des soft-state Kontext nicht angeboten werden kann, oder die Latenz durch eine erneute explizite Signalisierung zusätzlich zur sowieso schon vorhandenen Handover-Latenz noch einmal erhöht wird. Während dies für flexible Datenströme wie WWW-Verkehr hinnehmbar ist, gibt es doch zahlreiche Anwendungen in denen keiner dieser Fälle tolerierbar ist.

Als konkretes Beispiel für die Auswirkungen des Kontextverlustes soll nun ein Blick auf die Anwendung von Paketkopfomprimierung durch ROHC auf der Funkstrecke zwischen AR und MN geworfen werden.

2.2 Verhalten von ROHC bei Kontextverlust

Unter der plausiblen Annahme, dass in Abbildung 3 die Luftschnittstelle wohl die Verbindung mit der größten Link-Latenz und vor allem geringsten Bandbreite darstellt, erscheint es angebracht, die zu übertragende Datenmenge durch ein Verfahren zur Paketkopfkomprimierung zu minimieren. Als Protokoll zur Paketkopfkomprimierung kommt aufgrund der schlechten Eignung der älteren Verfahren für drahtlose Verbindungen eigentlich nur ROHC in Frage.

Bei der Betrachtung eines Handovers unter dieser Prämisse lassen sich folgende Beobachtungen tätigen: Ausgangspunkt soll der bereits eingeschwungene Zustand zwischen pAR und MN sein.

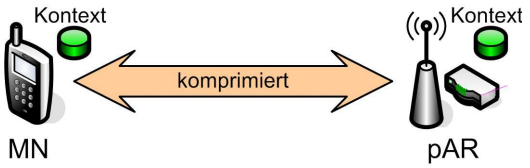


Abbildung 4: Datenstrom zwischen MN und pAR vor dem Handover

Wie in Abbildung 4 zu erkennen, haben sowohl MN als auch pAR im eingeschwungenen Zustand vollen Kontext und können daher mit maximaler Effizienz („Second Order“ Zustand) komprimieren.

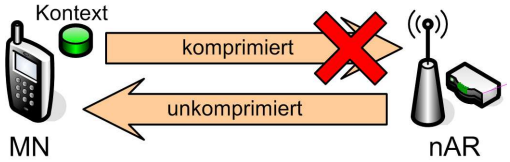


Abbildung 5: Datenstrom zwischen MN und nAR nach dem Handover

Nach einem Handover ändert sich die Situation, wie in Abbildung 5 dargestellt. Da der neue Zugangspunkt nAR keinerlei Kontextinformationen besitzt kann nAR die Daten von CN zunächst nur unkomprimiert an MN weiterleiten, was die benötigte Bandbreite an der Luftschnittstelle erhöht. Darüberhinaus kann er mit den vom MN, auf der Grundlage des alten mit pAR ausgehandelten Kontextes komprimierten Daten nichts anfangen und muss diese verwerfen. Durch die für die Etablierung eines neuen Kontextes nötigen unkomprimierten Sendewiederholungen durch MN, erhöht sich erneut die benötigte Bandbreite auf dem drahtlosen Übertragungsweg. Aus dem gleichen Grund erhöht sich außerdem die Latenz zwischen MN und CN während der Initialisierungsphase.

Ein Handover mit einhergehendem Kontextverlust hat also offensichtlich Auswirkungen auf die Latenz und den Bandbreitenbedarf der mit ROHC komprimierten Verbindungen. Wie stark dies die Leistung in Praxisszenarien wirklich beeinträchtigt, soll im folgenden Abschnitt geklärt werden.

2.3 Mathematische Modellierung der Auswirkungen

Am Beispiel einer Voice over IP Sitzung lässt sich zeigen wie viel Ersparnis die Verwendung von ROHC bringen kann: Die Paketköpfe von IPv6, UDP und RTP benötigen zusammen insgesamt 84 Byte, die sich aus den insgesamt 60 Byte für die drei Paketköpfe und 24 Byte für die Angabe der Home-Adresse durch Mobile IP im jeweils von der Übertragungsrichtung abhängigen Extension Header zusammensetzen. Die Nutzdaten belegen in diesem Beispiel gerade einmal 30 Byte. Durch ROHC kann der komplette Overhead in allen Paketköpfen auf 1 Byte (im „Second Order"-Zustand) reduziert werden, was einer Ersparnis von 83/114 = 73% entspricht! Diese Zahl lässt sich wie im Folgenden gezeigt werden wird aber leider nicht naiv auf eine Netzinfrastruktur übertragen, in der es z.B. aufgrund von Handovern zu Kontextverlusten kommen kann.

In [WeKo03] wird ein mathematisches Modell entworfen, mit dessen Hilfe sich die durch mögliche Kontextverluste ergebenden Implikationen auf die Gesamtperformance von ROHC-Verbindungen abschätzen lassen. Im Folgenden werden die grundlegenden Überlegungen dieses Modells zusammengefasst und die daraus resultierenden Ergebnisse vorgestellt.

Die Unterscheidung von First Order und Second Order Kompressorzuständen finden zugunsten eines einfacheren Modells und aufgrund der geringen Differenz der übertragenen Datenmengen im Vergleich zu unkomprimierten Paketköpfen, keinen Niederschlag in der Betrachtung. Stattdessen werden die zwei vereinfachten Pseudozustände FH (Full Header) und CH (Compressed Header) eingeführt.

Modelliert wird die Standardsituation eines Handovers wie sie bereits in Abbildung 3 eingeführt wurde. Das zu erwartende Verkehrsmuster nach einem Handover ist in Abbildung 6 dargestellt. Für die Verbindung zwischen AR und CN wird vereinfachend nur ein Hop dargestellt. Selbstverständlich gilt dieses Verkehrsmuster ebenso für den Uplink zum AR und nicht nur für den dargestellten Downlink zum MN.

Nachdem MN und AR einem Kontextverlust festgestellt haben, müssen erneut unkomprimierte Daten gesendet werden um auf beiden Systemen einen neuen ROHC Kontext zu etablieren.

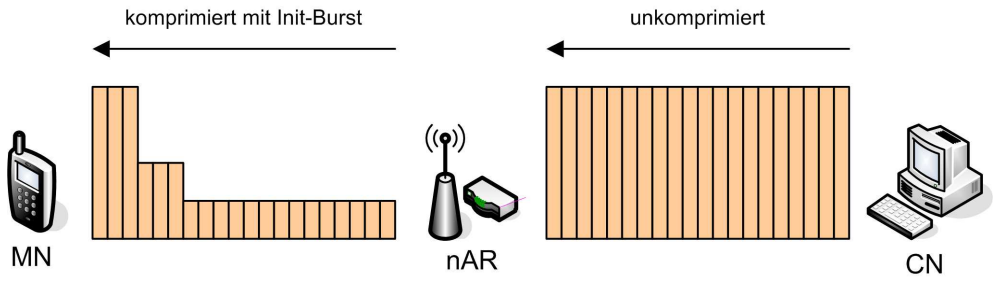


Abbildung 6: Traffic Burst bei Initialisierung eines neuen Kontexts

Das führt zu einem Traffic Burst, also einem überdurchschnittlich hohen Bandbreitenbedarf nach einem Handover mit einhergehendem Kontextverlust.

Die konkreten Kosten in Byte eines solchen Traffic Bursts lassen sich beispielsweise für den bestätigten Modus (bidirektional zuverlässig, R-Mode) recht einfach beziffern. Ausgehend von einer RTT (Round Trip Time) von X ms zwischen MN und AR und der Annahme das alle y ms ein FH Paket gesendet wird, ergibt sich, dass mindestens $\lceil \frac{X}{y} \rceil$ FH Pakete gesendet werden, bevor eine Bestätigung empfangen werden kann. In der Regel benötigt der Dekompressor mindestens z Pakete, bevor er diese Bestätigung abschicken kann, so dass sich die Gesamtkosten bei einer Paketgröße für FH Pakete von f Bytes folgendermaßen zusammensetzen:

$$cost = (\lceil \frac{X}{y} \rceil + z) * f$$

Wesentlich diffiziler gestaltet sich die Abschätzung der in [WeKo03] so genannten „sekundären Kontext-Initialisierungskosten“. Diese Kosten beziffern die zusätzlich zu allozierende Bandbreite pro Kommunikationskanal um dem erhöhten Bandbreitenbedarf während eines Initialisierungs-Bursts Rechnung zu tragen, und liegen oberhalb der von einem reinen Strom von CH Paketen benötigten Bandbreite. Abhängig von der verwendeten Kanalzuweisungsstrategie muss die zusätzliche Burst-Bandbreite bei der Kanalzuteilung passend berücksichtigt werden, um zu verhindern das die burstartigen Kontext-Initialisierungen den Nutzdatenstrom von konstanter Datenrate beeinträchtigen:

- **Getrennte Kanäle mit konstanter Bitrate:** Stehen nur Kanäle mit konstanter Bitrate zur Verfügung so muss die reservierte Rate um α über der tatsächlich benötigten Bitrate liegen. Dabei ist α abhängig von der maximal tolerierbaren Verzögerung, die durch den Initialisierungs-Burst für die Nutzdaten entsteht. Damit ergibt sich eine effektive Nutzung des Kanals von $\frac{1}{1+\alpha}$.
- **Getrennte Kanäle mit gemeinsamem Signalisierungskanal:** In diesem Szenario wird ein Signalisierungskanal explizit für den Aufbau des Kompressions-Kontextes von allen Teilnehmern gemeinsam genutzt. Der gleichmäßigere Strom von CH Paketen läuft über jeweils getrennte Kanäle. Mit $0 \leq \alpha_P \leq 1$ lässt sich die verfügbare Gesamtbandbreite aufteilen in einen Anteil von $(1 - \alpha_P)$ für den Signalisierungsverkehr und eine Gesamtanteil von α_P für die getrennten Nutzlast-Kanäle.
- **Gemeinsam genutzter Kanal:** Steht, wie z.B. in einer klassischen WLAN Umgebung (802.11a/b/g), nur ein einziger gemeinsam nutzbarer Kanal zur Verfügung, so degradiert die bereitstellbare Dienstqualität ab einer gewissen kritischen Anzahl an Teilnehmern, und damit an potentiellen Kontextverlusten, zunehmend.

Anhand einiger konkreter Zahlenbeispiele in den oben stehenden Einsatzszenarien, zeigen die Autoren von [WeKo03], dass sich, aufgrund der nötigen Vorkehrungen für Traffic Bursts

nach einem Kontextverlust, die effektiv wirksame Größe des komprimierten Paketkopfes von nominal 1 Byte auf real 9 bis 31 Byte erhöht. Dadurch wird gezeigt, dass Kontextverluste den anfangs dieses Abschnitts postulierten Gewinn durch ROHC, am Beispiel von Voice over IP von immerhin 73%, beträchtlich schmälern. Aus der Verbesserung um Faktor vier wird je nach Situation eine Verbesserung um nur noch Faktor zwei bis drei.

Nun sprechen diese Ergebnisse natürlich keinesfalls gegen die Verwendung von ROHC in Kontextverlust-anfälligen Umgebungen. Es soll vielmehr aufzeigen, welche weiteren zusätzlichen Gewinne durch ein Verfahren zur koordinierten Kontextübergabe realisierbar sind.

3 Das Kontexttransferprotokoll CTP

Das Kontexttransferprotokoll CTP (Context Transfer Protocol) [LNPK04] stellt eine mögliche Lösung dar, um den Kontextverlust durch einen Zugangspunktwechsel zu verhindern. CTP bietet dazu eine Repräsentation von Kontextinformationen, Nachrichten zur Initialisierung und Authorisierung von Kontexttransfers, zur Information des MN über den aktuellen Transferstatus und letztlich zum eigentlichen Transfer von Kontextinformationen vor, während und nach einem Handover an.

Im nächsten Abschnitt wird zunächst auf die grundlegende Arbeitsweise von CTP eingegangen, ohne dabei auf die Art des auf den Kontext angewiesenen Dienstes einzugehen, um im darauf folgenden Abschnitt am Beispiel von ROHC die Zusammenarbeit zwischen CTP und einem konkreten Dienst zu erläutern.

3.1 Allgemeine Funktionsweise

Der mit Hilfe von CTP durchzuführende Kontexttransfer kann durch verschiedene Ereignisse veranlasst werden. Fordert der MN den Kontext-Transfer an, so spricht man von einem Mobilgerät-kontrollierten („mobile-controlled“) Transfer. Entscheidet dahingegen einer der AR aufgrund eines internen Triggers, dass ein Transfer von Nöten ist, so handelt es sich um einen Netzwerk-kontrollierten („network-controlled“) Transfer. Darüberhinaus lassen sich Kontexttransfers, unabhängig von deren Initiator, auch noch in prädiktive und reaktive Transfers untergliedern. Erstere finden vor dem Wechsel des Zugangspunktes statt und verkürzen dadurch die Übergabezeit, während letztere erst nach dem Handover gestartet werden, dafür aber nicht auf ein frühzeitiges Erkennen einer Handoversituation durch die Netzinfrastruktur angewiesen sind.

Ein bestimmter Kontext wird in CTP jederzeit anhand der an pAR gültigen IP Adresse des MN und des jeweiligen Feature Profile Type (FPT) eindeutig identifiziert. FPTs sind 16-Bit Ganzzahlen, die von der IANA vergeben werden, und die Art des Kontexts sowie die zu übertragenden Parameter eindeutig kennzeichnen.

3.1.1 Protokollablaufszenarien

Aus der Sicht des pAR sind zwei Protokollablaufszenarien möglich.

Im ersten möglichen Szenario, wie in Abbildung 7 dargestellt, erhält pAR eine Context Transfer Activate Request (CTAR) Nachricht vom MN. Alternativ kann pAR auch durch einen internen Link-Layer Trigger über den bevorstehenden Handover informiert werden, falls die vorhandene Netzinfrastruktur und die verwendeten Mobilgeräte nur den Einsatz einer Netzwerk-kontrollierten Transferinitiierung und nicht der einer Mobilgerät-kontrollierten

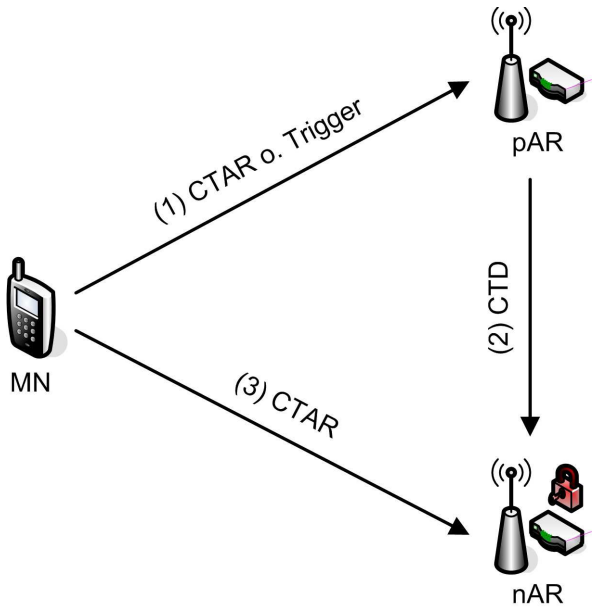


Abbildung 7: CTP Szenario 1

erlauben. In jedem Fall sendet pAR daraufhin prädiktiv mittels einer Context Transfer Data (CTD) Nachricht die Kontextinformationen an nAR. Unabhängig vom Szenario sendet MN immer eine CTAR Nachricht an nAR, da der MN nicht sicher stellen kann, ob der Kontext erfolgreich übertragen wurde. Falls der gewünschte Kontext tatsächlich bei nAR unbekannt ist wird wie in Szenario 2 verfahren.

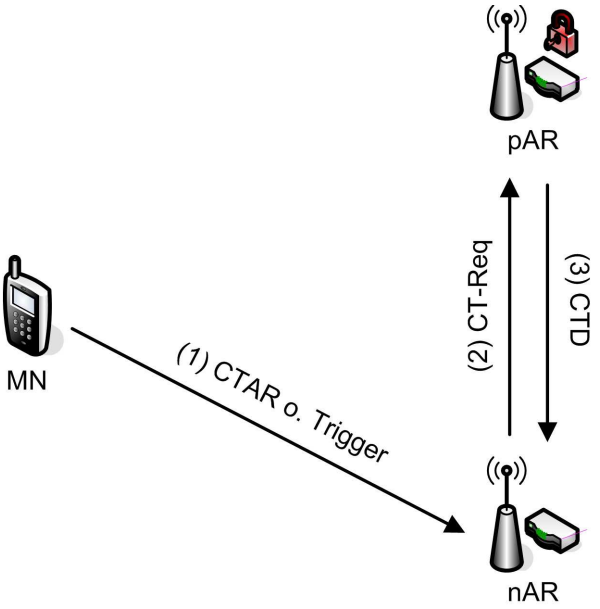


Abbildung 8: CTP Szenario 2

Im zweiten mögliche CTP Szenario (siehe Abbildung 8) erhält pAR eine Context Transfer Request (CT-Req) Nachricht von nAR, die dieser aufgrund eines von MN gesendeten CTAR oder eines internen Triggers absendet, falls der betreffende Kontext nicht schon zuvor wie im ersten Szenario beschrieben von pAR transferiert wurde. Daraufhin sendet pAR in einer CTD Nachricht den aktuellen Kontext an nAR.

3.1.2 Nachrichten

Für die Kommunikation zwischen MN und AR wird auf das ICMP-Protokoll aufgesetzt. Inter-AR Kommunikation findet über SCTP [SXMS+00] gesichert durch IPSec ESP statt.

- **Context Transfer Activate Request (CTAR):** Wie bereits im vorigen Unterabschnitt gesehen wird die CTAR Nachricht immer nach einem Zugangspunktwechsel vom MN an nAR gesendet. Darüberhinaus kann eine CTAR Nachricht auch an pAR gesendet werden um diesen bereits vor einem Handover aufzufordern, den Kontext an nAR zu übertragen.

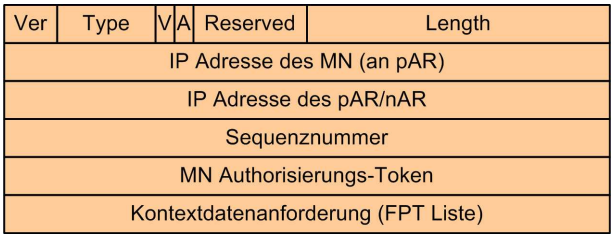


Abbildung 9: CTAR Nachrichtenformat

Unter anderem enthält die CTAR Nachricht die an pAR gültige IP Adresse des MN, die IP Adresse des jeweils anderen AR (also von pAR wenn die Nachricht an nAR geht und umgekehrt), einen Authorisierungs-Token (siehe Unterabschnitt 3.1.3) und eine Liste der zu übertragenden Kontextinformationen, gekennzeichnet durch ihre jeweiligen FPTs.

- **Context Transfer Data (CTD):**

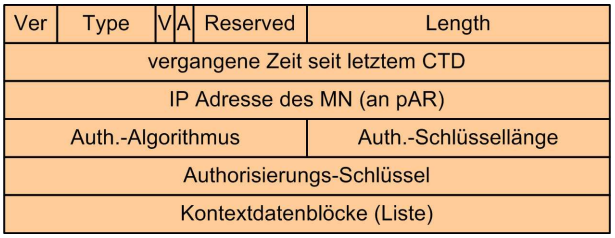


Abbildung 10: CTD Nachrichtenformat

Mit der CTD Nachricht sendet pAR an nAR unter anderem die an pAR gültige IP Adresse des MN, den gemeinsamen Schlüssel von AR und MN (siehe Unterabschnitt 3.1.3) und nicht zuletzt die angeforderten Kontextinformationen.

- **Context Transfer Request (CT-Req):**

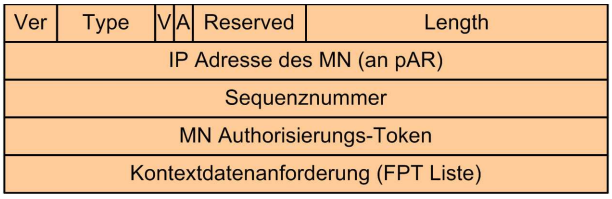


Abbildung 11: CT-Req Nachrichtenformat

Mittels CT-Req kann nAR den pAR veranlassen, ihm den aktuellen Kontext zu übersenden. Die CT-Req Nachricht enthält unter anderem die an pAR gültige IP Adresse des

MN, den Authorisierungs-Token (siehe Unterabschnitt 3.1.3) aus der CTAR Nachricht des MN, sowie eine Liste der zu übertragenden Kontextinformationen, gekennzeichnet durch ihre jeweiligen FPTs

Für alle Anfragen besteht die Möglichkeit für den Sender, im CTP Header der Anfragenachricht per A-Flag eine Bestätigungsnachricht anzufordern. Das V-Flag ist für IPv6 Adressen zu setzen, sonst werden IPv4 Adressen in den Datenfeldern verwendet.

3.1.3 Authorisierung

Die Authorisierung einer Kontexttransferanforderung ist aus offensichtlichen Gründen ein sicherheitskritisches Feature von CTP. Zur Absicherung dieses Vorgangs baut CTP auf ein gemeinsames Geheimnis zwischen MN und AR, aus dem von beiden für jede Anforderung ein fälschungssicheres Authorisierungs-Token gewonnen werden kann. Das Token wird durch eine kryptografisch sichere Hashfunktion, wie SHA1 [EaJo01], mit dem gemeinsamen Schlüssel von AR und MN aus der an pAR gültigen IP Adresse des MN und weiteren Details der Kontexttransferanforderung erzeugt.

Der gemeinsame Schlüssel von MN und AR wird beim Kontexttransfer mit übergeben, was bedeutet, dass der MN zwangsläufig jedem AR dem er ein authorisiertes CTAR schickt gleichermaßen vertrauen muss. Die Überprüfung des Tokens findet grundsätzlich immer an demjenigen AR statt, der augenblicklicher Besitzer des Kontexts ist. In den Abbildungen 7 und 8 ist der betreffende AR durch ein Schlosssymbol gekennzeichnet.

3.2 Verwendung von CTP zur Kontextübergabe bei ROHC

Wie in Abschnitt 2.3 erläutert wurde, verspricht man sich vom Einsatz eines kontrollierten Kontexttransfers einen geringeren Bandbreitenbedarf und damit unter dem Strich eine effizientere effektiv wirksame Kompressionsrate von ROHC-Verbindungen. Aufgrund der dadurch obsolet gewordenen erneuten Etablierung eines Kompressions- und Dekompressionskontextes und damit des Ausbleibens des Initialisierungs-Bursts ist diese Erwartung durchaus berechtigt. Im Folgenden soll daher die in [TiKo03] vorgeschlagene Vorgehensweise zum Einsatz von CTP in Zusammenarbeit mit ROHC vorgestellt werden.

Nachdem der pAR durch einen der bereits in Abschnitt 3.1.1 beschriebenen Mechanismen veranlasst wurde die Kontextdaten zu transferieren, sendet er die angeforderten ROHC Kontextinformationen als Teil der CTD Nachricht an nAR. Die ROHC Kontextinformationen bestehen im wesentlichen aus der „Static Chain“ und der „Dynamic Chain“, in denen die unveränderlichen bzw. die sich nach einem festen Muster ändernden Felder des Paketkopfs gespeichert sind.

Analog zu den Profildefinitionen in ROHC werden von der IANA eindeutige Compression Profile Type (CPT) Identifikatoren vergeben, z.B. für das Profil IPv6/UDP/RTP. Die CPTs kennzeichnen in der CTD Nachricht das verwendete Profil der übertragenen Kontextinformationen. Darüberhinaus werden zwei neue ROHC Profile „IR-CT-U“ und „IR-CT-D“ definiert um die bei einem Kontexttransfer übertragenen Daten von den durch einen MN gesendeten Kontextupdates zu unterscheiden.

Ändert sich durch den Zugangspunktwechsel die IP Adresse des MN, so muss diese Änderung dem vom pAR auf den nAR migrierten Kontext mitgeteilt werden, da dieser nach wie vor von der alten IP Adresse ausgeht. Um lediglich die Quelladresse eines Kontexts zu ändern ist in [BBDF$^+$01] kein Mechanismus vorgesehen. Daher empfiehlt es sich eine neue ROHC Nachricht „New-IP-Address-Uplink Extension“ zu definieren, die genau diese Aufgabe erfüllt.

Um die Fehlertoleranz gegenüber Nachrichtenverlusten zu erhöhen, muss pAR den Kontext nach dem Transfer für mindestens weitere HC_CONTEXT_SAVE_TIME ms (Empfehlung: 2000) im Speicher halten, und nach spätestens HC_CONTEXT_PURGE_TIME ms (Empfehlung: 5000) sämtliche mit dem entsprechenden MN verknüpften Kontextinformationen löschen.

Auf die Frage, wann genau der Kontexttransfer vollzogen werden soll, wird in [TiKo03] nicht näher eingegangen, sondern sie wird wie in der Spezifikation von CTP der Netzinfrastruktur überlassen. Der MN muss nach seinem optional bestätigten CTAR an nAR davon ausgehen, das der aktuelle Kompressionskontext an nAR übertragen wurde und stellt spätestens dann die komprimierte Kommunikation über pAR ein.

4 Abschließender Ausblick

Im Rahmen dieser Arbeit wurde gezeigt, das der mit einem Handover einhergehende Kontextverlust für erhebliche Leistungseinbußen sorgen kann, wenn nicht geeignete Maßnahmen dagegen ergriffen werden. Eine dieser möglichen Maßnahmen stellt die geordnete Übergabe des Kontexts zwischen den gewechselten Netzzugangspunkten dar. Als Protokoll dafür bietet sich CTP an.

Eine herausgehobene Stellung in der zurückligenden Betrachtung stellte die Anwendung von ROHC dar. Anhand dieser Anwendung wurden die besonders drastischen Auswirkungen nach einem Handover dargestellt und eine mögliche Lösung in Zusammenarbeit mit CTP diskutiert.

Doch ROHC ist bei weitem nicht der einzige Dienst der von Kontexttransfers bei einem Zugangspunktwechsel profitieren könnte. Weitere Kandidaten, in der Terminologie von RFC 3374 [Kemp02] auch als „Context Transfer Candidate Services“ bezeichnet, sind beispielsweise QoS-Mechanismen und AAA-Funktionalität. Diese könnten ebenfalls von der „Mitnehmbarkeit“ der einmal mit dem ersten Zugangspunkt ausgehandelten Paramater profitieren. Vorteile ergeben sich dabei insbesondere wenn die Latenzzeiten während des Handovers minimal gehalten werden sollen und deshalb der Verkehr an der Luftschnittstelle nicht durch zusätzlichen Signalisierungsverkehr erhöht werden soll, sondern ein Großteil der nötigen Informationen direkt zwischen den Zugangspunkten ausgetauscht werden kann.

Literatur

[BBDF+01] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura und H. Zheng. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. IETF RFC 3095, Juli 2001. Standards Track.

[EaJo01] D. Eastlake und P. Jones. US Secure Hash Algorithm 1 (SHA1). IETF RFC 3174, September 2001. Informational.

[Fein05] Maxim Feinleb. Algorithmen der robusten Paketkopfkomprimierung. Seminararbeit, 2005.

[JoPA04] D. Johnson, C. Perkins und J. Arkko. Mobility Support in IPv6. IETF RFC 3775, Juni 2004. Standards Track.

[Kemp02] J. Kempf. Problem Description: Reasons For Performing Context Transfers Between Nodes in an IP Access Network. IETF RFC 3374, September 2002. Informational.

[LNPK04] J. Loughney, M. Nakhjiri, C. Perkins und R. Koodli. Context Transfer Protocol. IETF draft-ietf-seamoby-ctp-11.txt, August 2004. Work In Progress.

[Sta04] Statistisches Bundesamt Deutschland. Ausstattung privater Haushalte mit Informations- und Kommunikationstechnik in Deutschland 2001-2003. http://www.destatis.de/basis/d/evs/budtab2.php, 2004.

[SXMS+00] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang und V. Paxson. Stream Control Transmission Protocol. IETF RFC 2960, Oktober 2000. Standards Track.

[TiKo03] Manish Tiwari und Rajeev Koodli. Context Relocation for Seamless Header Compression in IP Networks. IETF draft-koodli-seamoby-hc-relocate-02.txt, Juni 2003. Work In Progress.

[Wals05] Martin Walser. Paketkopfkomprimierung in drahtlosen Zugangsnetzen. Seminararbeit, 2005.

[WeKo03] Cédric Westphal und Rajeev Koodli. IP Header Compression: A Study of Context Establishment. In *Proceedings of IEEE WCNC*, 2003.

Abbildungsverzeichnis

1 ROHC Kompressor Zustände 200

2 ROHC Dekompressor Zustände 200

3 Ein mobiles Endgerät wechselt seinen Netzzugangspunkt 202

4 Datenstrom zwischen MN und pAR vor dem Handover 202

5 Datenstrom zwischen MN und nAR nach dem Handover 203

6 Traffic Burst bei Initialisierung eines neuen Kontexts 204

7 CTP Szenario 1 206
8 CTP Szenario 2 206
9 CTAR Nachrichtenformat 207
10 CTD Nachrichtenformat 207
11 CT-Req Nachrichtenformat 207