

IPv6 – Das neue Internetprotokoll



Florian Endler
University of Ulm, Germany

5. Juni 2002

1. Warum IPv6.....	3
2. Das IPv6 Design	4
2.1 Aufbau des IPv6 Header	4
2.2 Vergleich IPv4 und IPv6 Header	5
2.2.1 Vereinfachung von IPv6.....	5
Festes Header Format.....	5
Entfernen der Headerprüfsumme	5
Entfernen der Fragmentierung aus dem IP-Header	6
Entfernen des Type-of-Service Feld (TOS)	6
2.2.2 Überarbeitet Parameter.....	6
Ersetzen des Feldes Gesamtlängen durch das Feld Nutzdatenlänge	6
Ändern des Protokollfeldtyp in Nächster-Header-Typ.....	7
Ändern des Lebendzeitfeld (TTL) in Hop-Limit Feld	7
2.2.3 Neue Felder.....	7
Das Flusslabel Feld	7
Das Klassen Feld	7
2.3 Erweiterungsheader die neuen Optionen.....	8
Der Routingheader	9
Der Fragmentheader.....	12
Der Zieloptionsheader	14
Der Hop-by-Hop Optionsheader	15
Der Authentifizierungsheader	16
Der Verschlüsselte Sicherheitsnutzdatenheader	16
Erweiterungsheader Reihenfolge	17
3. Adressen	18
Unicast-Adressen.....	19
Anycast-Adressen.....	20
Multicast-Adressen	21
Appendix A Literaturverzeichnis	22
Appendix B Abkürzungen.....	22

1. Warum IPv6

Ein kurzer Überblick über die Entstehung des Internets und seiner Protokolle.

Als 1975 die erste Spezifikation von IPv4 veröffentlicht wurde, ahnte wahrscheinlich niemand der Beteiligten, zu was für einem Siegeszug das Internet ein Vierteljahrhundert später antreten würde. Als Mitte der achtziger Jahre die ersten Internet Service Provider (ISP) entstanden, die auch privaten Personen einen Zugang zum , bis dahin fast ausschließlich vom Militär und Bildungseinrichtungen genutzten, Internet ermöglichten, blieben erste kommerzielle Seiten nicht lange aus.

Anfang der neunziger Jahre begannen die IETF und IAB sich Gedanken über die zukünftige Größe des Internets zu machen, und wie der drohenden Knappheit von Adressen am besten zu begegnen sei. Es wurden verschiedene Vorschläge gemacht, von denen viele wenig bis gar nicht brauchbar waren, weil in der Regel zu statisch.

1992 dann, stellte Steve Deering eine Weiterentwicklung von IP in IP, dessen Idee es war, dass Internet in zwei Schichten zu trennen, nämlich in einen weltweiten Backbone und viele begrenzte Bereiche, SIP (Simple IP) vor. SIP sah zum einen eine Erhöhung des Adressraums auf 64-Bit vor, sowie den Einsatz von Kapselung statt Optionen. Mitte 1993 wurde das von Paul Francis vorgeschlagene PIP (Pauls Internet Protocol), das einige revolutionäre Ideen bezüglich Routingstrategien enthielt, mit SIP zu Simple IP Plus (SIPP), dem Vorgänger des heutigen IPv6, verschmolzen.

Allerdings hatte sich die Situation bezüglich der Verfügbarkeit von IPv4 Adressen mittlerweile durch die Entwicklung neuer Technologien wie z.B. NAT (Network Address Translation) und durch das Splitten der großen Class-A Netze in viele kleine Netze erheblich entschärft, weshalb wohl heute, trotz damaliger seriöser Hochrechnungen, noch IP-Adressen verfügbar sind. Jedoch hatte NAT insbesondere die IP-Puristen auf den Plan gerufen, die den Grundgedanken, dass jeder Rechner der am Internet teilnimmt über eine eigene eindeutige Adresse verfügen sollte, der Punkt-zu-Punkt Verbindung. Einer der Kritiker die sich besonders hervortaten, ist der ehemalige Leiter des IAB Christian Huitema, der mit seinem Buch IPv6 im November 1997 für erhebliches Aufsehen in der Fachwelt sorgte.

Für die Zukunft gilt es eigentlich als fast sicher, dass sich IPv6 als neues Internet Protokoll durchsetzen wird, da mittlerweile alle wichtigen Firmen (z.B. Cisco, Nokia und seit kurzem auch Microsoft) und Communities (Linux) an einer breiten Implementierung des Protokolls arbeiten, so dass davon ausgegangen werden kann, dass eines Tages tatsächlich jeder Kühlschrank, jedes Auto, jedes Mobiltelefon, etc. einen Anschluss an das Internet, über eine eigene IP-Adresse bekommen wird. Allerdings gibt es, insbesondere im Sicherheitsbereich, noch einige Entwicklungsarbeit zu leisten, um die selbe Flexibilität wie unter IPv4 zu erreichen.

Eine Anmerkung noch zum Schluss: die Versionsnummer des neuen Internet Protokoll ist IPv6 und nicht IPv5, weil die Version IPv5 bereits durch das experimentelle Stream Protocol Version 2 (ST-2) belegt war.

2. Das IPv6 Design

Ein Überblick über das IPv6 Protokoll und ein Vergleich zwischen IPv4 und IPv6.

Bei der Entwicklung von IPv6 stand, außer eines stark vergrößerten Adressraums, vor allem die Vereinfachung des IP-Headers im Vordergrund. Der praktische Umgang mit IPv4 hatte gezeigt, dass die meisten Optionen die direkt im IPv4 Header implementiert sind, in der Praxis so gut wie nie zum Einsatz kommen, da Sie in der Regel zu Performanceverlusten beim Routing führen, oder aber, die Implementierung zu aufwendig ist. Des Weiteren kam man überein, dass gewisse Funktionen, wie z.B. die Check-Sum nicht zwangsläufig implementiert werden müssen, da diese Funktion wesentlich besser von den einzelnen Transportprotokollen wahrgenommen werden kann.

Am Ende entstand dann ein Internet Protokoll Version 6, das tatsächlich auf sämtliche Optionsfelder im Header selber verzichtet, und das auch sonst von geringerer Ähnlichkeiten mit dem Internet Protokoll Version 4 ist. Die meisten Funktionen die von IPv4 unterstützt wurden, sind trotzdem auch weiterhin unter IPv6 nutzbar, wie sich später herausstellen wird.

2.1 Aufbau des IPv6 Header

Der Aufbau des IPv6 Header besteht aus einem 64-Bit Header gefolgt von zwei 128-Bit IPv6-Adressen für Quell- und Zieladresse, und hat demnach eine Gesamtlänge von 40 Bytes*.

Die ersten 64-Bit setzen sich wie folgt zusammen:

- Versionsfeld (4 Bit)
- Klasse (8 Bit)
- Flusslabel (20 Bit)
- Nutzdatenlänge (16 Bit)
- Nächster Header (8 Bit)
- Hop-Grenze (8 Bit)

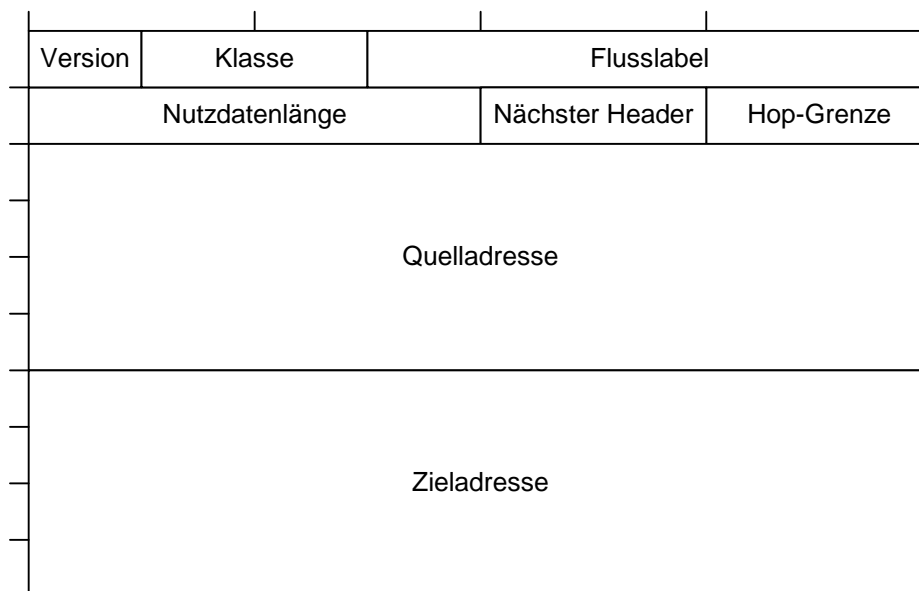


Abbildung 2-1:
IPv6 Header

* Zur Erinnerung 1 Byte (Oktett) sind 8 Bit. Daraus folgt eine Länge des IPv6-Headers von 40 Bytes (320-Bits)

2.2 Vergleich IPv4 und IPv6 Header

Vergleicht man den IPv4 und IPv6-Header, stellt man tatsächlich fest, das als einziges Feld mit gleicher Länge und Funktion, das Versionsfeld übrig geblieben ist. Über das Versionsfeld wird die Version der benutzten IP-Version definiert. Der Wert beträgt entweder vier für IPv4 (binär 0100), oder aber sechs (binär 0110) für IPv6. Ursprünglich war geplant so IPv4 und IPv6 Pakete zu unterscheiden, wovon allerdings später wieder Abstand genommen wurde. Wenn möglich sollen stattdessen die verschiedenen IP-Pakete auf Mediensicht (Ethernet, X25, Token-Ring, etc.) entflochten werden. Sechs Felder von IPv4 kommen im IPv6-Header überhaupt nicht mehr vor. Hierzu gehören die „Header Länge“ (IHL), der „Servicetyp“, die „Kennung“, die „Flags“, die „Header-Prüfsumme“ und das „Fragment-Offset“. Weitere drei Felder, „Gesamtlänge“, „Protokoll-Typ“ und „Lebenszeit“, wurden im Zuge der IPv6 Entwicklung verändert bzw. geringfügig anders definiert. Neu eingeführte Felder sind das „Klassenfeld“ und „Flusskontrollfeld“.

Version	IHL	Servicetyp (TOS)	Gesamtlänge	
Kennung		Flags	Fragment-Offset	
Lebenszeit (TTL)	Protokoll	Header-Prüfsumme		
Quelladresse				
Zieladresse				
Optionen (immer ein vielfaches von 32-Bit)			Füllbits	

Abbildung 2-2:
IPv4 Header

2.2.1 Vereinfachung von IPv6

Wie Eingangs schon beschrieben, stand außer eines größeren Adressraums bei der Entwicklung von IPv6 vor allem, die Vereinfachung des neuen Internet Protokolls im Mittelpunkt. So setzten sich folgende vier Hauptvereinfachungen durch: Festes Header Format, entfernen der Headerprüfsumme, und entfernen der Fragmentierungsoption sowie TOS-Feldes (Type Of Services) aus dem IPv6-Header.

Festes Header Format

Als eine weitere Folge der Vereinfachung des Headers wurde auf sämtliche optionalen Elemente im Header selber verzichtet. Stattdessen wurden sogenannte Erweiterungsheader eingeführt (siehe auch Kapitel 2.3), die hinter jedem Haupthead eingefügt werden können und in denen dann Optionen definiert und mitgeschickt werden können. Der Vorteil dieser Vereinfachung ist, dass der IPv6-Header eine feste Headerlänge bekommt, was der Verarbeitung zu gute kommt und somit der Performance. Des weiteren konnte durch die Einführung einer festen Headerlänge auf das IHL-Feld (Internet-Headerlänge) verzichtet werden.

Entfernen der Headerprüfsumme

Das entfernen der Headerprüfsumme hat den großen Vorteil, das ebenfalls der Aufwand bei der Headerverarbeitung massiv vermindert wird, weil die Notwendigkeit entfällt, nach jeder Übertragung diese zu überprüfen und gegebenenfalls zu aktualisieren. Die Gefahr, das Pakete, trotz Fehler weitergeleitet werden, ist darüber hinaus sehr gering, da die meisten Kapselungsprozeduren über eine eigene Paketprüfsumme verfügen (Adaptions-Layer für ATM-Verbindungen und für die Rahmenprozeduren des Point-To-Point Protokolls (PPP) für serielle Verbindungen).

Entfernen der Fragmentierung aus dem IP-Header

Bezüglich der Fragmentierung von Paketen, wurden von IPv4 die Headerfelder „Paketkennung“ (Kennung), „Kontroll-Flag“ und der „Fragment-Offset“ benötigt. Bei IPv6 wurde davon Abstand genommen, da der Host selber mit Hilfe der Prozedur Path MTU (Media Transmission Unit) Discovery die maximale Paketgröße die für die Übertragung zulässig ist, ermitteln soll. Schickt ein Host Pakete, die größer sind, als die erlaubten, weil der Host die Prozedur z.B. nicht unterstützt, werden diese vom Netzwerk abgelehnt, und nicht wie bisher automatisch fragmentiert. Stattdessen wurde ein optionaler Fragmentheader eingeführt (siehe auch Kapitel 2.3 „Erweiterungsheader die neuen Optionen“).

Mit der Veröffentlichung von RFC 2460 im Jahr 1998 wurde die minimale Paketgröße, die von allen Hosts unterstützt werden muss, auf 1280 Bytes festgelegt. Empfohlen wird allerdings eine Größe von mindestens 1500 Bytes. Sollte aus Platzgründen (z.B. Boot-ROM) nur eine einfache Implementierung von IPv6 möglich sein, könnte, um auf die Implementierung von Path MTU Discovery verzichten zu können, die maximal Größe der Oktetts grundsätzlich auf 1280 Bytes beschränkt werden.

Entfernen des Type-of-Service Feld (TOS)

Als letztes wurde noch das TOS-Feld (Type-of-Service) entfernt. Unter IPv4 konnten hier Präferenzen bezüglich der Route angegeben werden. Da dieses Feld in der Praxis so gut wie nie angewendet wird, wurde auf eine Implementierung unter IPv6 verzichtet.

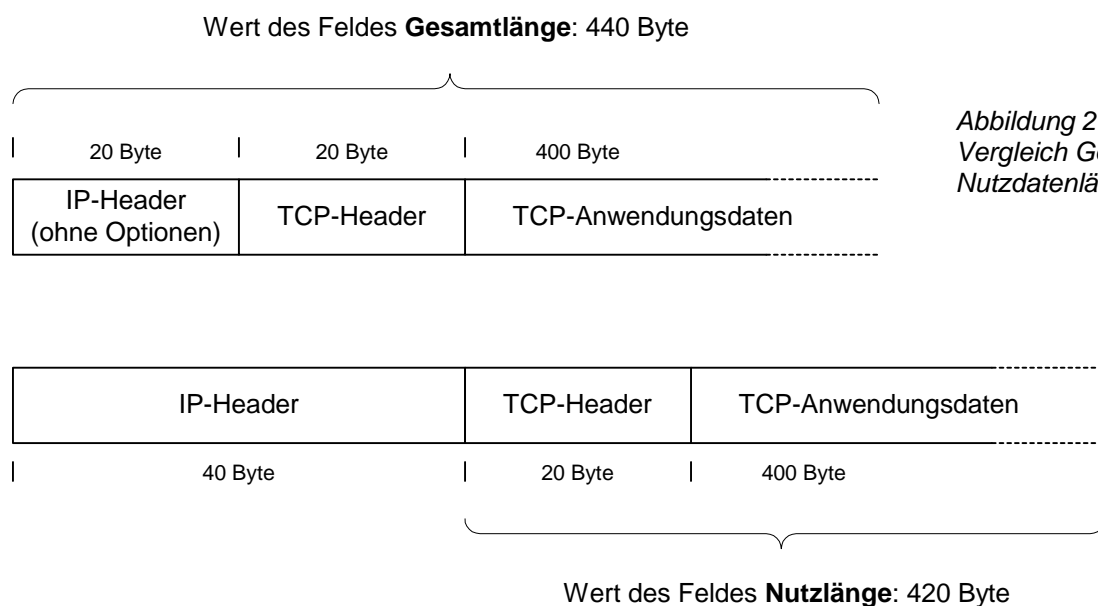
2.2.2 Überarbeitete Parameter

Mit der Implementierung von IPv6 mussten einige Parameter an das neue Protokoll angepasst werden. Hierzu zählen wie bereits erwähnt die Felder „Gesamtlänge“, „Protokoll-Typ“ und „Lebenszeit“.

Ersetzen des Feldes Gesamtlängen durch das Feld Nutzdatenlänge

Bei IPv6 wurde das Feld „Gesamtlänge“ durch das Feld „Nutzdatenlänge“ ersetzt. Der Unterschied besteht darin, dass per Definition von IPv6, die Nutzdatenlänge, die Länge der Daten entspricht, die nach dem IPv6-Header kommen. Der IP-Header wird also nicht mehr wie bisher in die Gesamtlänge mit eingerechnet.

Beispiel:



Das Feld „Nutzlänge“ wird bei IPv6 ebenfalls mit 16-Bit kodiert, weshalb wie auch bei IPv4 die maximale Paketgröße des Nutzdatenpakets auf 64-Kbytes (65538 Bytes) beschränkt ist. Allerdings gibt es unter IPv6 die Möglichkeit einen Erweiterungsheader zu definieren, um auch größere Pakete zu schicken (siehe hierzu auch Kapitel 2.3 Abschnitt „Hop-by-Hop Optionsheader“).

Ändern des Protokollfeldtyp in Nächster-Header-Typ

Der Protokollfeldtyp von IPv4 wurde in Nächster-Header-Typ (Next-Header-Type) umbenannt, um die neue Organisation der IP-Pakete besser wiederzugeben. Im Vergleich zu IPv4 Paketen, hinter denen immer sofort der Transportprotokollheader stand, gibt es wie schon mehrfach erwähnt, unter IPv6 die Möglichkeit einen oder mehrere Erweiterungsheader zur Verarbeitung von zusätzlichen Informationen einzufügen. Kommt ein solcher Erweiterungsheader zum Einsatz, wird der Wert des Next-Header-Type auf den ersten Erweiterungsheader gesetzt. Wird kein Erweiterungsheader benutzt, wird der Wert des Next-Header-Type Felds auch weiterhin, sowie beim Protokollfeldtyp unter IPv4, auf den entsprechenden Transportprotokollheader gesetzt, bzw. auf den Wert null (59) falls kein Nächster-Header existiert.

Ändern des Lebenszeitfeld (TTL) in Hop-Limit Feld

Das Lebenszeitfeld (Time-to-Life – TTL) wurde durch das Hop-Limit Feld ersetzt. Bei IPv4 wurde die maximale Lebenszeit als eine Anzahl von Sekunden angegeben, dargestellt durch eine 8-Bit große Ganzzahl (daraus folgt eine maximale Lebensdauer 256 Sekunden). Jeder Router auf dem Weg verringert diesen Wert theoretisch mindestens um Eins, oder aber um die Zeit, die das Paket in der Warteschlange des Routers gewartet hat. Da die Wartezeit in der Praxis nur sehr schwer zu ermitteln ist, wird der Wert in der Regel aber unabhängig von der Wartezeit von den Routern nur um den Wert Eins verringert, auch wenn die tatsächliche Zeitspanne, die das Paket in bis zur Weiterleitung warten musste um ein vielfaches über diesem Wert liegt. Um dieser Tatsache gerecht zu werden, wurde in der Spezifikation von IPv6 von einer Lebenszeitangabe eines Paketes in Sekunden abgesehen, und statt dessen eine maximale Anzahl von Hops festgesetzt, die von jedem Router auf dem Weg um Eins verringert wird. Die Größe des Feldes, und somit der maximalen Hops wurde bei 8-Bit belassen (entspricht 256 Hops). Um das aufleben alter Pakete wirksam zu unterbinden, sollten in Zukunft die entsprechenden Transportprotokolle eigene Schutzvorkehrungen treffen, wie z.B. Zeitstempel oder große Nummerierungsfelder (siehe auch RFC 1323).

2.2.3 Neue Felder

Es gibt zwei neue Felder bei IPv6, die bei IPv4 nicht vorkamen: Das „Flusslabel“-Feld und das „Klassen“-Feld. Beide wurden hauptsächlich dafür entwickelt, um den sogenannten „Echtzeitverkehr“ zu erleichtern. Insbesondere das „Klassenfeld“ könnte unter Umständen ein neues TOS-Feld für IPv6 Pakete werden.

Das Flusslabel Feld

Das Flusslabel Feld wird benutzt um Pakete zu kennzeichnen, die alle von einer definierten Quelle zu einem definierten Ziel mit einer definierten Menge von Optionen gesendet werden. Weitere Stichworte bezüglich des Flusslabel Feld sind „Quality of Service“ und „Ressourcenreservierung“

Das Klassen Feld

Ursprünglich war das Klassen-Feld als Kontrollbits-Feld konzipiert worden und nur 4-Bit groß, und sollte als Prioritätsfeld dienen. Es stellte sich jedoch mit der Zeit heraus, dass die 4-Bit einen zu klein bemessenen Spielraum offen ließen, weshalb der Feldwert auf 8-Bit vergrößert wurde.

In der letzten Spezifikation des IPv6 Protokolls wurde noch intensiv an den beiden Feldern geforscht, weshalb an dieser Stelle nicht weiter auf die Eigenschaften der Felder eingegangen wird.

2.3 Erweiterungsheader die neuen Optionen

Wie eingangs schon erwähnt, wurden sämtliche Optionsfelder von IPv4 unter IPv6 nicht implementiert, und zwar deshalb, weil in der Regel Pakete mit Optionen, um die Performance der großen Masse von Paketen ohne Optionen zu verbessern, bei der Abarbeitung in den Routern oft in einer sekundär Schlange platziert werden, was den Einsatz von Optionen äußerst unattraktiv macht. Allerdings gibt es gute Gründe, eine besondere Behandlung für Pakete zu verlangen, um z.B. eine bestimmte Route zu definieren oder um die Nutzlast der zu transportierenden Daten zu vergrößern. Unter IPv6 wurde das Problem wie folgt gelöst. Hinter jedem IPv6-Header lassen sich beliebig viele Erweiterungsheader hängen, die über einen entsprechenden Headertyp gekennzeichnet werden. Jeder dieser Erweiterungsheader verfügt wiederum über ein entsprechendes „Nächster-Header“ Feld so dass beliebig viele Erweiterungsheader eingefügt werden können. Handelt es sich um den letzten Erweiterungsheader vor den Nutzdaten, so enthält dieser in seinem „Nächster-Header“ Feld einen Verweis auf den folgenden Nutzdatenheader.

IPv6-Header Nächster Header = Routing	TCP-Header + Daten
--	--------------------

Abbildung 2-4:
Beispiel für eine
Headerkette

IPv6-Header Nächster Header = Routing	Routingheader Nächster Header = TCP	TCP-Header + Daten
--	--	--------------------

IPv6-Header Nächster Header = Routing	Routingheader Nächster Header = Fragment	Fragmentheader Nächster Header = TCP	Fragmentheader TCP Daten
--	---	---	-----------------------------

In der letzten Spezifikation von IPv6 sind folgende sechs Erweiterungsheader definiert:

- Hop-by-Hop Optionsheader
- Routingheader Type 0
- Fragmentheader
- Authentifizierungsheader
- Verschlüsselte Sicherheitsnutzdatenheader
- Zieloptionsheader

Um zu verhindern, dass Erweiterungsheadertypen mit Protokolltypen kollidieren, wurde festgelegt, dass die Nummern für die Erweiterungsheader aus den selben 256 Nummern umfassenden Pool gebildet werden müssen, wie die Protokolltypen z.B. ICMP, TCP und UDP (siehe auch Abb. 2-5 „Auszug Protokoll und Headertypen“).

Dezimal-Wert	Binär-Wert	Schlüsselwort	Headertyp
0	0000 0000	HBH	Hop-by-Hob Optionen
5	0000 0101	ST	Stream
6	0000 0110	TCP	Übertragungsprotokoll
17	0000 1001	UDP	Benutzer-Datagramm
43	0001 1011	RH	Routing Header
44	0001 1100	FH	Fragmentierungs Header
45	0001 1101	IDRP	Interdomänen Routingprotokoll
50	0011 0010	ESP	Verschlüsselte Sicherheitsnutzdaten
51	0011 0011	AH	Authentifizierungsheader
58	0011 1010	ICMP	Internet Kontrollnachricht
59	0011 1011	Null	Kein Nächster Header
60	0011 1100	DEST	Zieloptionsheader
80	0101 0000	ISO-IP	ISO Internet Protokoll (CLNP)
88	0101 0100	IGRP	Interior Gateway Protocoll (Cisco)
89	0101 0101	OSPF	Open Shortest Path First
256	1111 1111	--	Reserviert

Abbildung 2-5:
Auszug Protokoll –
und Headertypen

Der Routingheader

Der Routingheader setzt sich aus vier mal 8-Bit Parameterfeldern sowie einem 32-Bit Parameterfeld, gefolgt von einer Adressliste zusammen, und ersetzt die Quellrouting-Option von IPv4.

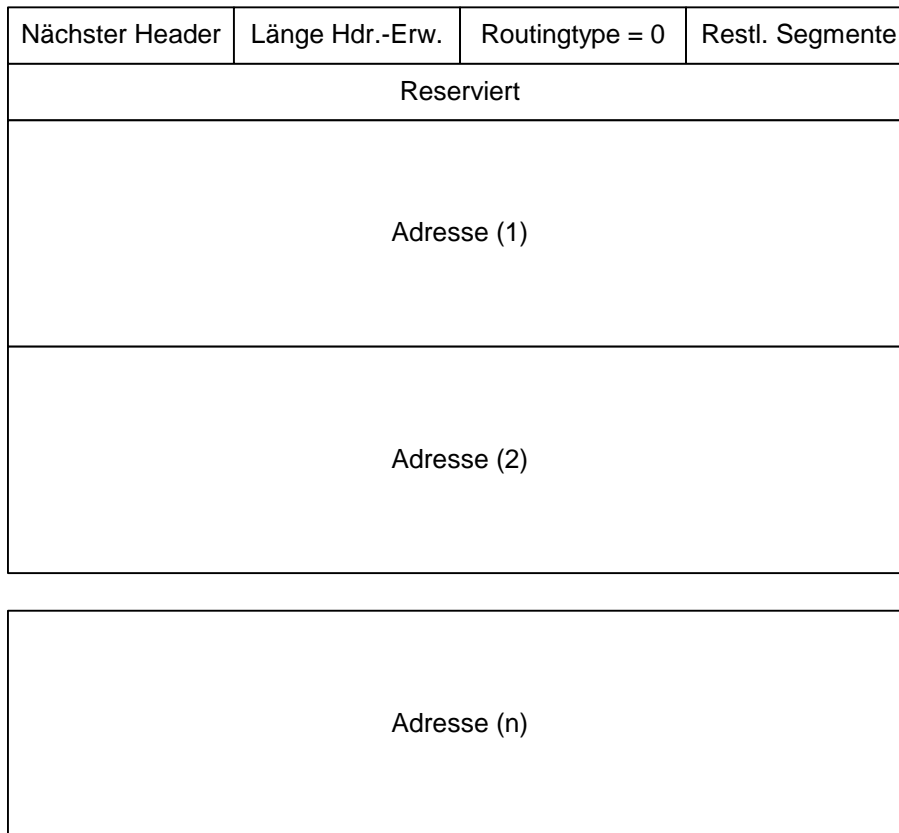


Abbildung 2-6:
Routingheader

Die ersten vier Parameterfelder enthalten jeweils eine 8-Bit Ganzzahl. Das Feld „Nächster-Header“ enthält den Typ des unmittelbar nach dem Routingheader stehenden Headers in der Headerkette. Das Feld „Länge Hdr.-Erw.“ (Header-Erweiterung) beschreibt die Länge des Routingheaders als eine Anzahl von 64-Bit Wörtern. Das „Routing-Type“ Feld wird default mäßig auf 0 gesetzt. Das Feld „Restliche Segmente“ enthält die Anzahl der restlichen Adressen in der Liste, über die das Routing bis zum Ziel laufen soll. Das 32-Bit Feld „Reserviert“ wurde ursprünglich eingeführt, um anzuzeigen, ob das Quellrouting strikt oder locker sein sollte. Da sich die Implementierung aber als äußerst schwierig erwies, wurde die Idee in der späteren Spezifikation wieder verworfen.

Der Rest des Routingheaders besteht aus einer Liste von 128-Bit Adressen, nummeriert von 1 bis n. Die Abarbeitung der Liste geschieht wie in Abb 2-6 beschrieben. Grundsätzlich wird der Routingheader jedoch nur dann bearbeitet, wenn der Router eine seiner Adressen im Feld „Zieladresse“ im IPv6-Header findet, was einen enormen Zeitvorteil gegenüber des in IPv4 implementierten Quellroutings ausmacht, da dort jeder Router gezwungen war alle gesetzten Optionen zu überprüfen.

Theoretisch wäre, dank der Codierung der Headergesamtlänge („Länge Hdr.-Erw.“) in 64-BitWörter, (d.h. eine Adresse belegt immer 2-Bit) eine gesamt Anzahl von $2^8 / 2 = 128$ Adressen möglich.

Anmerkung: Der oben gezeigt Routingheader ist der sogenannte Routingtype 0 oder auch „Service Routing“ genannt. Hierbei handelt es sich momentan um den einzigen definierten Routingheadertyp.

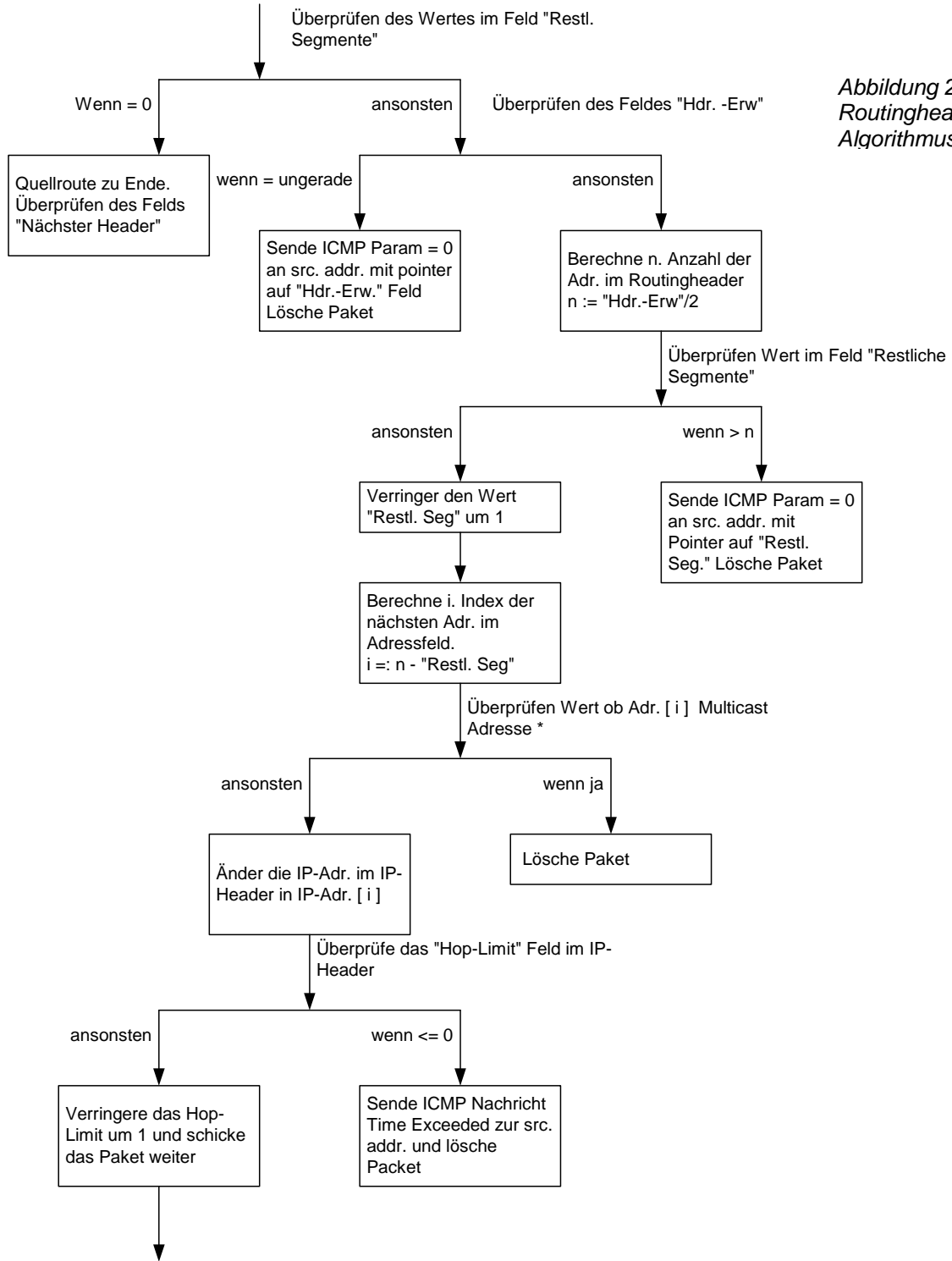
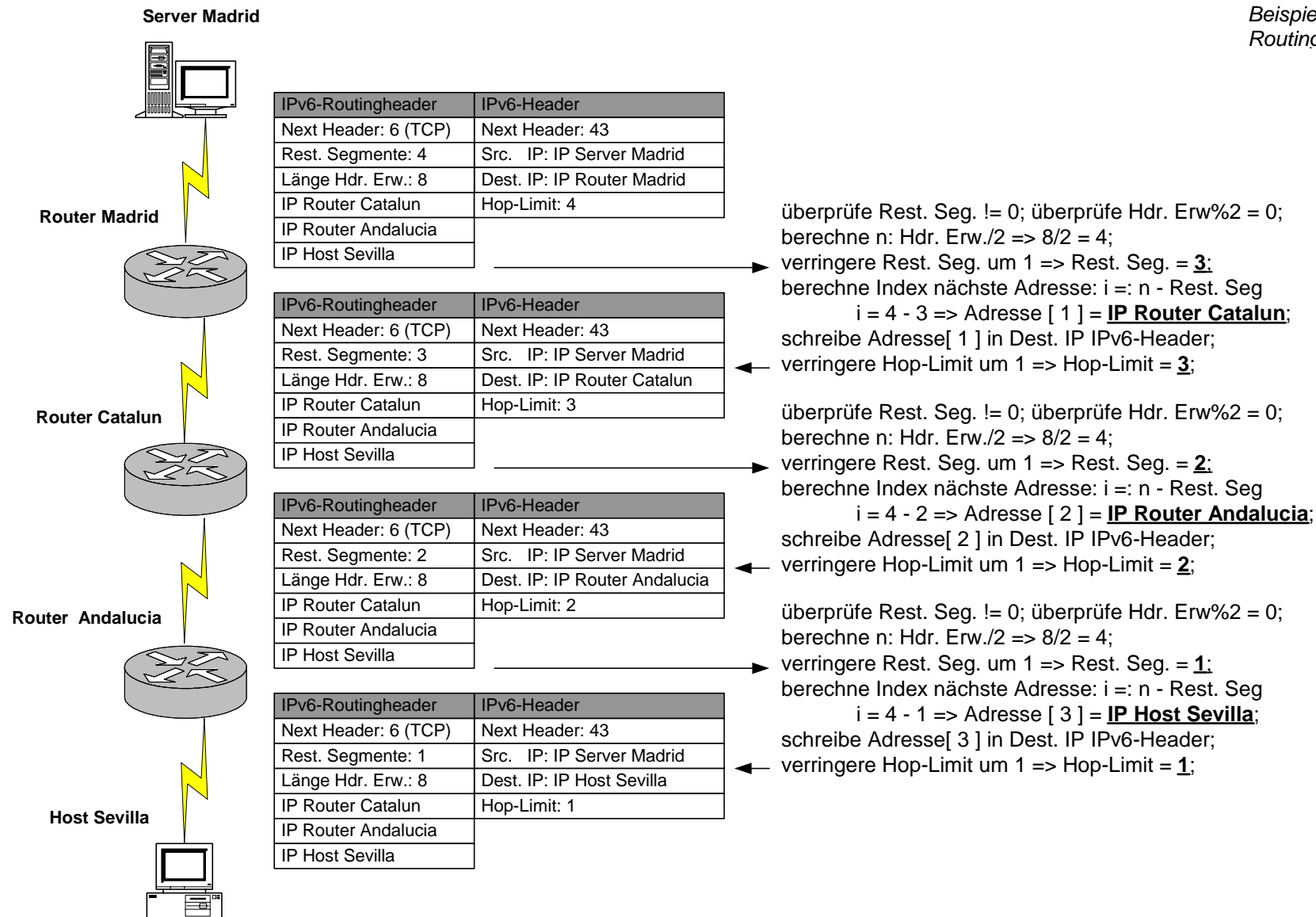


Abbildung 2-7:
Routingheader
Algorithmus

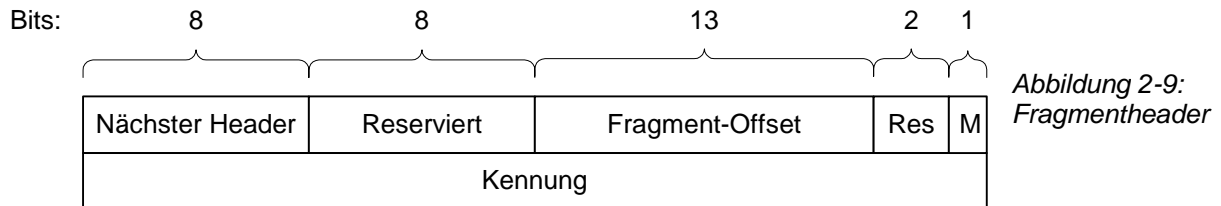
* Siehe Kapitel Multicast

Abbildung 2-8:
Beispiel: funktionsweise
Routingheader



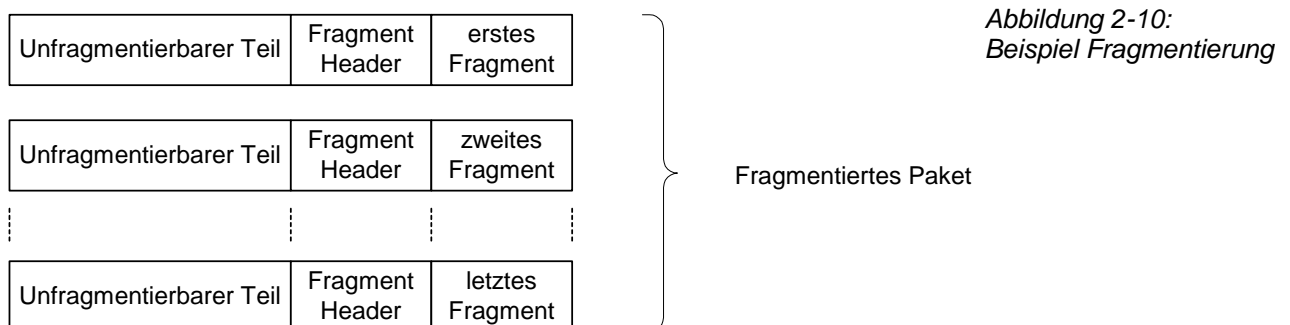
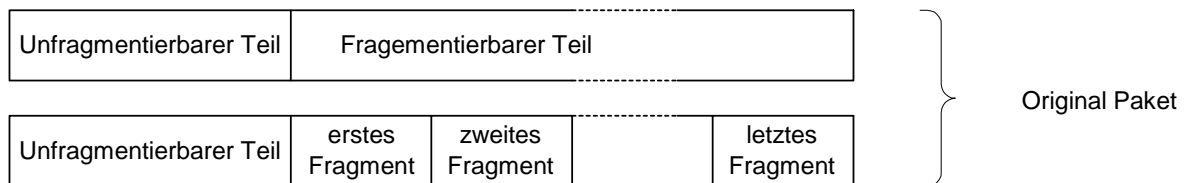
Der Fragmentheader

Im Gegensatz zu IPv4 teilen IPv6 Router übergroße Pakete nicht in Fragmente auf. Wird ein Paket größer als die zulässige MTU des nächsten Hops geschickt, wird dieses abgelehnt, und eine ICMP Nachricht an den Versender geschickt. Mit dem Fragmentheader ist es allerdings möglich, schon vor dem Versenden Pakete die zu groß sind, in mehrere kleine aufzuteilen. Hierzu wird das Paket in, der Path MTU entsprechend große, Fragmente aufgeteilt. Jedes dieser Fragmente wird für sich alleine geroutet und erst am Ziel durch die in den Fragmentheadern gespeicherten Fragmentinformationen wieder zusammen gesetzt.



- Nächster Header (8 Bit): Gibt den nächsten Header in der Headerkette an.
- Reserviert (8 Bit): Wird für die Übertragung des Paketes auf null gesetzt und von dem Empfänger ignoriert.
- Fragment-Offset (13 Bit): Beschreibt den Offset, in 8-Bytes Einheiten, der Daten die diesem Header folgen, relativ zum Start des fragmentierten Teil des Paketes. Weißt dem Paket seine Stelle im Originalpaket zu.
- Res. (2 Bit): Wie schon das reservierte 8-Bit Feld wird dieses Feld ebenfalls auf null gesetzt, und von dem Empfänger ignoriert.
- M (1 Bit): Die sogenannte More-Flag. Beschreibt, ob es weitere Fragmente gibt. Wenn ja ist der Wert 1 ansonsten 0.
- Kennung (32 Bit): Kennzeichnet zu welchem Originalpaket das Fragment gehört.

Zum Fragmentieren wird für jedes Paket vom Versender ein Identifikationswert gebildet, der für alle anderen Fragmente aus der selben Übertragung gleich sein muss, und sich von eventuellen anderen Fragmentpaketen mit gleicher Quell- und Zieladresse eindeutig unterscheiden lassen muss. Das ursprüngliche Paket besteht aus einem sogenannten nicht-fragmentierbaren, und einem fragmentierbaren Teil. Der nicht-fragmentierbare Teil enthält den IPv6 Header, sowie alle Erweiterungheader, die für die Übertragung des Paketes notwendig sind (z.B. Routingheader, Hop-by-Hop Header). Der fragmentierbare Teil besteht aus dem Rest des Pakets, und den Erweiterungsheadern, die nur von der Zieladresse ausgewertet werden müssen, sowie die Upper-Layer Headern wie z.B. TCP oder UDP. Fragmente sollten immer ein vielfaches von 64-Bit groß sein.



Um die Fragmente am Ziel wieder zusammensetzen zu können, werden die Quelladresse und das Identifikationsfeld überprüft. Nur wenn diese Werte übereinstimmen und alle Pakete vorhanden sind, kann das Paket wieder in den original Zustand versetzt werden.

Beispiel:

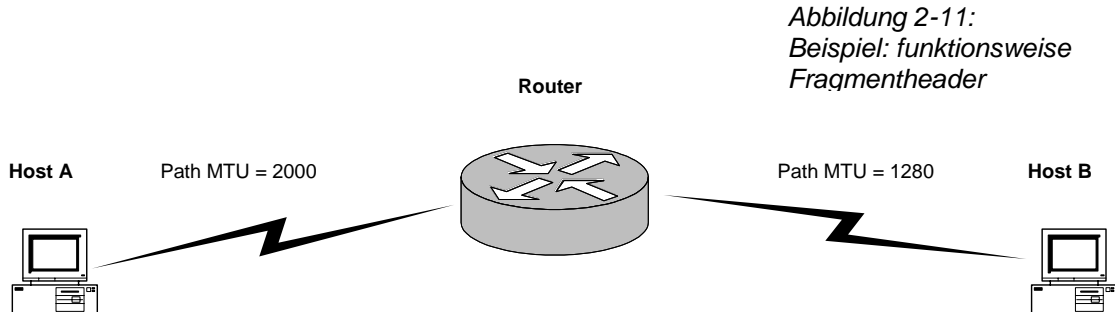


Abbildung 2-11:
Beispiel: funktionsweise
Fragmentheader

Host A möchte ein 1500 Oktett (Bytes) Nutzlast an Host B schicken (Länge des Pakets wäre folglich 1500 Bytes plus 40 Bytes für den IPv6-Header; also 1540 Bytes). Da Host A keine Path MTU Discovery unterstützt, schickt er das Paket an Host B unfragmentiert. Der Router stellt fest, dass das Paket zu groß ist und schickt eine ICMP Meldung „packet too big“ mit Zeiger auf die Path MTU 1280 zurück, und verwirft das Paket. Hätten wir jetzt ein UDP-Paket geschickt, wäre der Vorgang vermutlich zu Ende, da auf IP-Ebene kein automatisches „Retransmit“ stattfindet, und UDP auch keine solche Funktion vorsieht. Man könnte ein UDP-Paket auch mit einem Postpaket vergleichen, das irgendwo verloren geht. Würde das UDP-Paket allerdings noch mal angefordert, so würde dank der ICMP Meldung Host A diesmal eine automatische Fragmentierung durchführen.

Gehen wir davon aus, wir hätten zuerst ein TCP-Paket der Größe 1540 Bytes geschickt. Auch dieses Paket wäre an der MTU zwischen Router und Host B gescheitert, und wäre verworfen worden. Der Router hätte auch diesmal eine ICMP Fehlermeldung „packet too big“ mit einem entsprechenden Zeiger auf die Path MTU an Host A geschickt. Die Retransmit-Funktion von TCP würde jetzt allerdings dafür sorgen, dass das Paket erneut an Host B gesendet würde. Dank der ICMP-Meldung kennt der IPv6-Stack jetzt aber die zulässige Fragmentgröße. Und würde eine entsprechende Fragmentierung durchführen.

Bei der Fragmentierung muss berücksichtigt werden, dass jeweils noch der IPv6-Header (40 Bytes) und der Fragmentheader (8 Bytes) zu jedem Paket dazukommen. Daraus folgt dann folgende Fragmentierung:

Nutzlast: 1500 Bytes

Reservierte Bytes für Header p. Paket: 68 (40 Bytes IPv6-Hdr., 8 Bytes Frag.-Hdr. 20 Bytes TCP-Hdr.)

1. Paket: 1280 Bytes. Entspricht 1212 Bytes Nutzlast und 68 Bytes IPv6-, Fragment- TCP-Header.
2. Paket: 356 Bytes. Entspricht 288 Bytes Nutzlast. und 68 Bytes IPv6-, Fragment- TCP-Header.

Anmerkung: Wahrscheinlich wäre, zumindest wenn wir das TCP-Paket von einer Linux-Maschine aus geschickt hätten, diese Konstellation niemals zustande gekommen, da TCP selber schon dafür gesorgt hätte, dass von vornherein aus Performancegründen nur entsprechend große Pakete geschickt worden wären.

Da die Route eines Paketes sich immer wieder ändern kann, setzt die MTU-Prozedur eines Hosts eine ermittelte MTU nach Ablauf einer Zeitspanne automatisch wieder hoch. Bleibt ein Paket erneut hängen, wird die MTU entsprechend wieder angepasst.

Der Zieloptionsheader

Es gibt zwei Möglichkeiten, IPv6 Paketen zusätzliche Informationen in Form von Optionen mitzugeben. Die eine ist, einen Erweiterungsheader zu definieren, in dem die Optionen dann gespeichert, und für jeden Link auf der Route sichtbar durch das Netzwerk transportiert werden. Das Problem hierbei ist, dass, wie bereits erwähnt, der Pool um solche Erweiterungsheader zu definieren sehr klein ist; nämlich gerade mal 256 Einträge umfassend. Die meisten dieser Nummern sind zudem schon vergeben (siehe auch Abb. 2.5 „Ausschnitt Protokoll- und Headertypen“). Die Spezifikation des Zieloptionsheaders ist relativ einfach. Insgesamt besteht der Header aus drei Feldern, wobei die ersten beiden Felder, das „Nächste Header“ und „Länge Hdr.-Erw.“, jeweils eine 8-Bit große Ganzzahl enthalten, und das dritte Feld die eigentlichen Optionen beinhaltet. Das Optionsfeld muss immer ein vielfaches des Wertes von 8 Bytes sein.

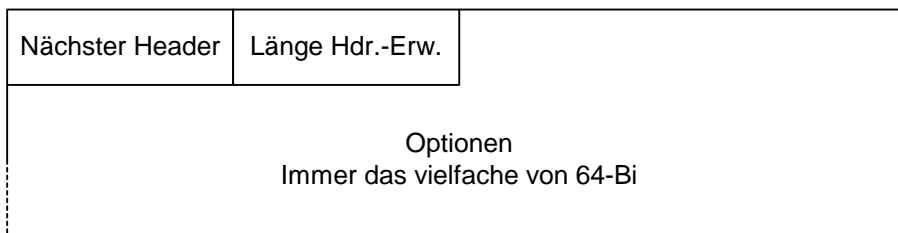


Abbildung 2-12:
Zieloptionsheader

Im „Nächster Header“ Feld steht, wie bei allen anderen Headern auch, der Wert des nachfolgenden Headers in der Headerkette. Das Längenfeld „Hdr.-Erw.“ enthält, wie schon das Längenfeld des Fragmentheaders 64-Bit Wörter. Dies bedeutet, dass, ist das Optionsfeld z.B. 8-Bytes groß, beträgt der Wert des Längenfelds 0. Bei einer Größe von 32-Bytes würde der Wert des „Hrd.-Erw.“ Feldes drei sein. Dies ermöglicht also, theoretisch 64 x 256 Bit (16384 Bit oder 2 KBytes) Optionen mitzuschicken.

Die Optionen die im Optionsfeld gespeichert sind müssen folgende Form aufweisen.

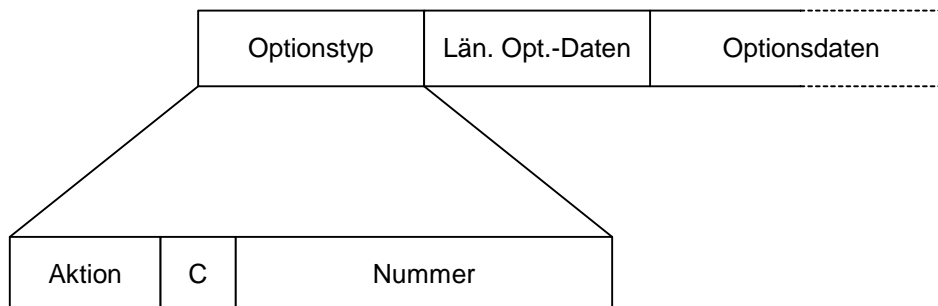


Abbildung 2-13:
Optionen

Der Optionstyp (8-Bit) beschreibt um welches Aktion es sich handelt (5-Bit). Optionstypen müssen bei der IANA (Internet Assigned Numbers Authority) beantragt und eingetragen werden. Des Weiteren kann eine Aktion (2-Bit) siehe auch Abb. 2-12) definiert werden, die durchgeführt werden soll, wenn der ausführende Knoten die Option nicht unterstützt. Das Dritte Bit, Feld „C“, ist das sogenannte Change-route Bit, das anzeigt, ob die Option unterwegs geändert wurde. Der Wert 0 bedeutet keine Änderungen, der Wert 1 bedeutet die Option wurde geändert.

Bits	Aktion
00	Diese Option überspringen
01	Das Paket wegwerfen, kein ICMP Bericht
10	Das Paket wegwerfen, ICMP Bericht senden auch wenn Multicast-Adresse *
11	Das Packet wegwerfen, ICMP Bericht nur dann senden wenn keine Multicast-Adresse

Abbildung 2-14:
Übersicht Aktionen

* Siehe auch Kapitel 3. Abschnitt „Multicast-Adressen“

Der Hop-by-Hop Optionsheader

Der Hop-by-Hop Optionsheader besitzt die gleiche Form und Funktion wie der Zieloptionsheader, mit dem Unterschied, dass Optionen, die in dem Hop-by-Hop Optionsheader definiert wurden, von jedem Knoten auf der Paketroute ausgewertet werden müssen. Dies kann z.B. dann sinnvoll sein, wenn ein Paket versendet werden soll, dessen Länge nicht in 16-Bit kodiert werden kann. In diesem Fall wird das IPv6 Längenfeld auf null gesetzt, und jeder Router auf der Strecke müsste den entsprechenden Hop-by-Hop Optionsheader dann decodieren, um so die Länge festzustellen. Eine solche, sogenannte Jumbo-Nutzdaten Option, sollte nur dann benutzt werden, wenn die Größe des zu sendenden Pakets kleiner 65535-Bytes (64 KBytes) ist. Des weiteren dürfen keine Fragmentheader in Verbindung mit einer Jumbo-Nutzdaten Option benutzt werden.

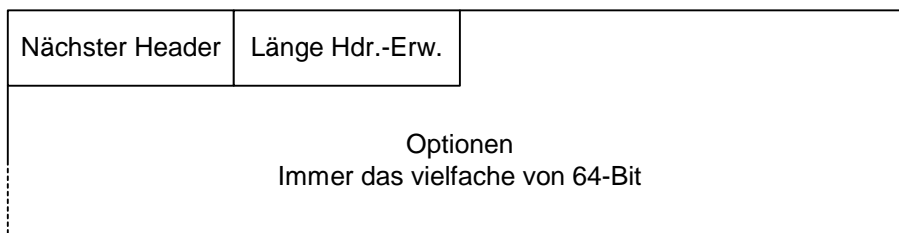


Abbildung 2-15:
Hop-by-Hop
Optionsheader

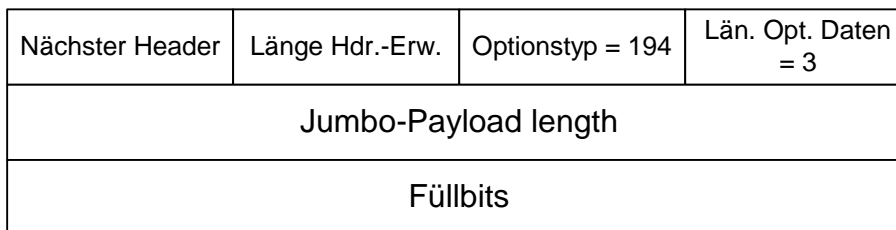


Abbildung 2-16:
Jumbo-Nutzdatenheader

Anmerkung: Jumbonutzdatenpakete machen eigentlich nur dann Sinn, wenn die Laufzeiten der gesendeten Pakete extrem lang sind, und die verwendeten Leitungen eine entsprechend große Kapazität gewährleisten.

Der Authentifizierungsheader

Der Authentifizierungsheader ist ein generischer Erweiterungsheader, der typischerweise zwischen den IPv6 Header und den End-To-End Nutzdaten (siehe auch Abb. 2-18). Der Einsatz eines Authentifizierungsheader verändert die Struktur und das Verhalten der Mitgeschickten Nutzdaten nicht. Er dient einzig dazu, der Zieladresse die Echtheit des Versenders zu garantieren.

Anmerkung:

Da es sich beim Authentifizierungsheader, wie auch bei dem Verschlüsselten Sicherheitsnutzdatenheader, um zwei sehr komplexe Themen handelt, werden diese nur der Vollständigkeit wegen angesprochen.

Nächster Header	Nutzdatenlänge	Reserviert
Sicherheitsparameter-Index (SPI)		
Folgenummer-Feld		
Authentifizierungsheader (variable)		

Abbildung 2-17:
Authentifizierungsheader

IPv6-Header	Authen. Header	TCP-Header	TCP-Anwendungsdaten
-------------	----------------	------------	---------------------

Abbildung 2-18:
Beispiel: Einsatz Auth.
Header

IPv6-Header	Routingheader	Authen. Header	TCP-Header	TCP-Anwendungsdaten
-------------	---------------	----------------	------------	---------------------

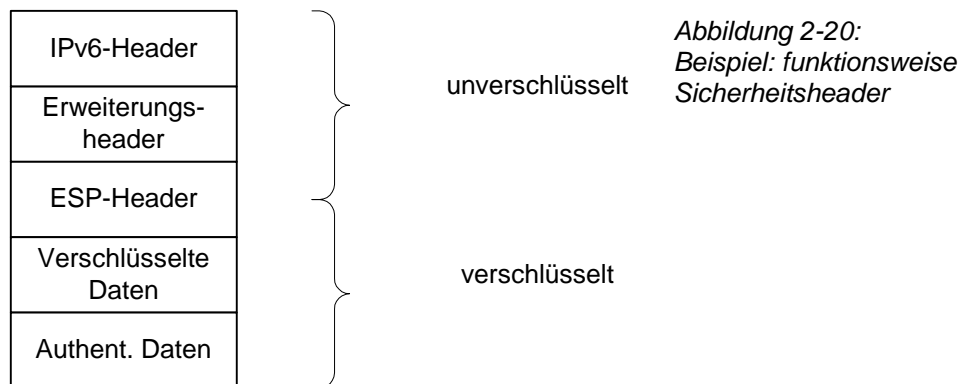
IPv6-Header	Auth. Header	End-To-End Optionen	TCP-Header	TCP-Anwendungsdaten
-------------	--------------	---------------------	------------	---------------------

Der Verschlüsselte Sicherheitsnutzdatenheader

32-Bit Sicherheitsparameter-Index (SPI)
32-Bit Folgenummer
Verschlüsselte Daten Variabler Länge
Authentifizierungsdaten

Abbildung 2-19:
Verschlüsselter
Sicherheitsdatenheader

Im Gegensatz zum Authentifizierungsheader verschlüsselt der Sicherheitsnutzdatenheader die zu transportierenden Daten. Hierbei werden sämtliche Daten ab dem Sicherheitsnutzdatenheader (siehe auch Abbildung 2-20), der als letzter Header in einer IPv6 Headerkette definiert sein sollte, durch einen beliebigen Verschlüsselungsalgorithmus codiert.



Erweiterungsheader Reihenfolge

Jeder Erweiterungsheader darf nur einmal pro Paket vorkommen, um Inkonsistenzen bei der Verarbeitung zu verhindern, abgesehen davon, dass es in der Regel auch überhaupt keinen Sinn machen würde einen Erweiterungsheader doppelt zu definieren.

Kommen mehrere Erweiterungsheader zum Einsatz, sollten diese in einer bestimmten Reihenfolge (siehe Abbildung 2-20) hinter dem IPv6-Header gehängt werden. Diese Reihenfolge ist aber nicht zwingend. Es gilt, dass die Erweiterungsheader in der Reihenfolge abgearbeitet werden wie sie in der Headerkette stehen.

IPv6-Header	320-Bit (40 Bytes)
Hop-by-Hop-Optionsheader	variabel - min. 64-Bit (8 Bytes) max. 16384-Bit (2 KBytes)
Zieloptionsheader (1)	variabel - min. 64-Bit (8 Bytes) max. 16384-Bit (2 KBytes)
Routing-Header	variabel - min. 320-Bit (40 Bytes) max. 16384-Bit (2 KBytes)
Fragment-Header	8-Bit (1 Byte)
Zieloptionsheader (2)	variabel - min. 64-Bit (8 Bytes) max. 16384-Bit (2 KBytes)
Authentication-Header	variabel
Verschlüsselte Sicherheitsnutzdatenheader	variabel

*Abbildung 2-21:
Reihenfolge
Erweiterungsheader*

3. Adressen

Eine kurze Einführung über das Adress-Design und die Selbstkonfiguration von Hosts.

Das neue Erscheinungsbild der IPv6-Adressen unterscheidet sich im Vergleich zur IPv4-Adresse komplett. Der Adressbereich, die 128-Bit, wurden in jeweils acht 16-Bit lange Adressabschnitte gegliedert, die durch einen Doppelpunkt voneinander getrennt werden. Da die IPv6-Adressen sowieso schon wesentlich länger als die IPv4-Adressen sind, entschied man sich für eine hexadezimale Darstellung der einzelnen Abschnitte. Eine gültige IPv6-Adresse wäre z.B.

FE80:BA72:1080:0C00:0000:0000:0A00:0001

Um die Darstellung etwas zu vereinfachen, wurde festgelegt, dass wie auch schon bei IPv4-Adressen, führende Nullen weggelassen werden können. So könnte das oben angegebene Beispiel auch so geschrieben werden:

FE80:BA72:1080:C00:0:0:A00:1

Eine weitere Vereinfachung ist, dass Blöcke die nur aus Nullen bestehen auch ganz weggelassen werden können. Hierzu werden die Blöcke einfach komplett aus der Adresse entfernt, und durch zwei Doppelpunkte als Platzhalter ersetzt.

FE80:BA72:1080:C00::A00:1

Beim wiederherstellen der Adresse werden einfach entsprechend viele Nullen anstelle des „:“ Platzhalters eingefügt, damit die Adresse wieder 128-Bit groß wird. Dies impliziert allerdings, dass der Platzhalter nur einmal pro Adresse eingesetzt werden kann, da sich ansonsten keine Eindeutige Aussage bezüglich der ursprünglichen Position Einzelner Abschnitte, zwischen zwei Platzhaltern, machen lassen würde.

Beispiel:

Nehmen wir folgende Adresse an:

1080:0:0:0:C:0:0:1

Würden wir jetzt beide „Nullbereiche“ durch den Platzhalter ersetzen, wüssten wir beim wiederherstellen der Adresse nicht, an welcher Stelle das C aus Abschnitt 5 ursprünglich stand.

1080::C::1

Mögliche Adresskombinationen wären:

1080:0:C:0:0:0:0:1

1080:0:0:C:0:0:0:1

...

1080:0:0:0:0:C:0:1

Die Adressen wurden bei IPv6, wie Eingangs schon besprochen auf 128-Bit erweitert, was 2^{128} Adressen entspricht. Das sind $3,4028236692093846346337460743177e+38$ Adressen; also eine ganze Menge. Geht man davon aus, dass auf unserer Erde ca. 6,5 Milliarden Menschen leben, so würde das für jeden Menschen in etwa $2^{128} / 6,5 * 10^9 = 52351133372452071302057631912,58$ oder grob $52 * 10^{27}$ Adressen ergeben.

Soweit die Theorie. Natürlich wurden unter IPv6, wie auch schon unter IPv4, verschiedene Adressmerkmale definiert, die über ein bestimmtes Präfix am Anfang der Adresse, oder über bestimmte Abschnitte der Adresse definiert werden, und die die oben aufgestellte Rechnung etwas relativieren. Die drei bekanntesten Arten sind Unicast-, Anycast und Multicast-Adressen.

Um auch IPv4 Adressen kompatibel zu sein, lässt sich eine IPv4 Adresse unter IPv6 folgendermaßen schreiben:

0:0:0:0:A00:1 oder auch ::A00:1

Um die Umrechnung von Dezimal in Hexadezimal zu vermeiden, können die letzten 32-Bit auch in der gewohnten, durch Punkte getrennten, Dezimalform dargestellt werden. So wäre z.B.

0:0:0:0:10.0.0.1 oder auch ::10.0.0.1

durchaus eine zulässige IPv6-Adresse.

Des weiteren gibt es ebenfalls, wie unter IPv4, die Möglichkeit Präfixe zu definieren. Dabei ist die Schreibweise identisch zu IPv4.

FE80:BA72:1080::/40 definiert die ersten 40-Bit als Präfix.

Unicast-Adressen

Eine Unicast-Adresse wird genau an ein Interface eines Routers oder Endsystems geschickt, das dezidiert durch eine entsprechende Unicast-Adresse spezifiziert ist, weshalb Unicast-Kommunikation auch 1:1 Kommunikation genannt wird (siehe auch Abbildung 3-1). Es gibt mehrere Typen von Unicast-Adresse, die sich durch das Präfix unterscheiden (siehe RFC 2373).

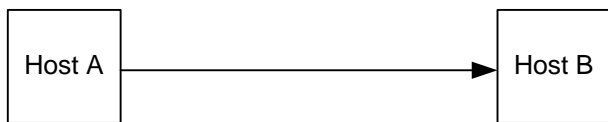
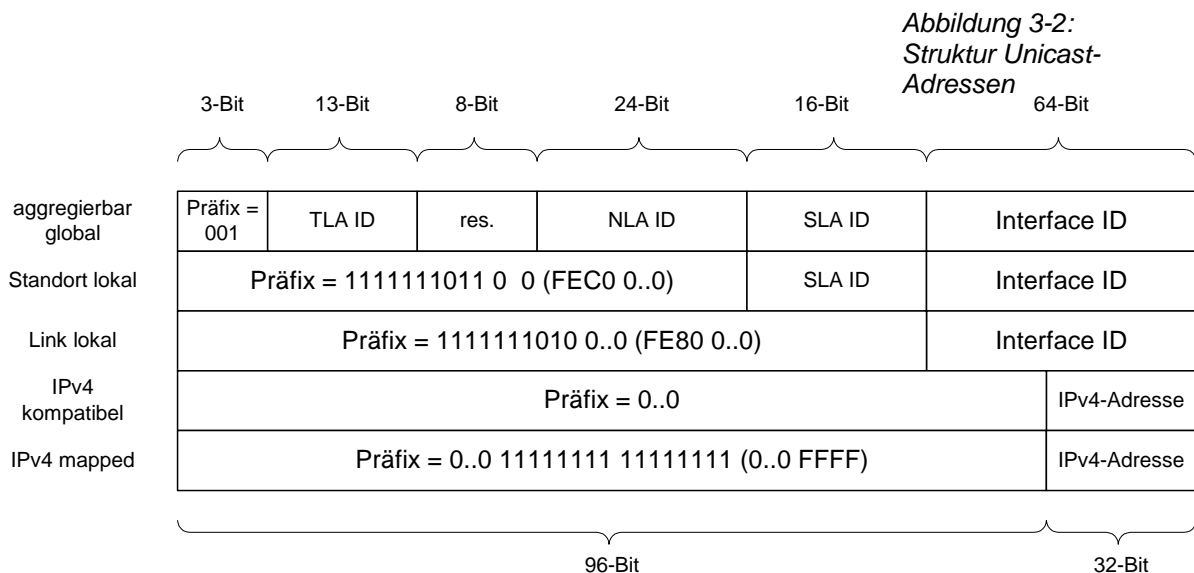


Abbildung 3-1:
Unicast-Kommunikation

Laut Spezifikation von Unicats-Adressen wird momentan zwischen fünf verschiedenen Unicast-Adresstypen unterschieden (siehe auch Abbildung 3-2)



Aggregierbar global

Die aggregierbare, globale Unicast Adresse unterteilt eine Adresse in einen öffentlichen und einen lokations spezifischen Teil. Der globale Teil besteht aus einem Präfix, Top-Level-Aggregator (TLA) und Next-Level-Aggregator (NLA) und beschreibt eine Lokation (Site) im globalen Internet. Diese Informationen werden für das Routing von Paketen benutzt, weshalb TLA-Identifikatoren ausschließlich an ISP vergeben werden, die einen öffentlichen Transit-Dienst anbieten.

NLA-Identifikatoren werden an untergeordnete Service-Provider oder an Firmen die einen direkten Anschluss an einen TLA-Service Provider besitzen vergeben. Der lokations spezifische Teil besteht aus einem sogenannten Site-Level-Identifikator (SLA ID) und einer Interface-ID. Die SLA-ID beschreibt die Subnetz-Struktur innerhalb einer Lokation. Die Interface-ID beschreibt das Interface, auf das die IP-Adresse gebunden wird. Typischerweise wird hierzu die 48-Bit umfassende MAC-Adresse des entsprechenden Netzwerkadapters verwendet.

Standort lokal

Standort lokale Adressen sollten dann benutzt werden, wenn eine Lokation nicht an das globale Internet angeschlossen ist. Wird die Lokation später dann an das gloable Internet angeschlossen, muss nur noch der Standort-Lokale-Teil durch einen Provider-basiertes Präfix ersetzt werden. Subnetz und Interface-Kennung können unverändert übernommen werden

Link lokal

Link lokale Adressen enthalten neben dem Präfix nur eine Interface-ID. Solche Adressen werden während der automatischen Konfiguration oder in Netzen ohne Router generiert. Router dürfen Link-lokale Adressen nicht weiterleiten.

IPv4 kompatible und IPv4 mapped Adressen

Diese Adressen sollten nur beim Übergang von IPv4 zu IPv6 benutzt werden, solange in einem Netzwerk IPv4 und IPv6 Verkehr parallel geroutet werden.

Anycast-Adressen

Eine Anycast-Adresse kennzeichnet eine Menge von Interfaces, die typischerweise zu verschiedenen Zwischensystemen gehören. Ein Paket das an eine Anycast-Adresse gesendet wird, wird genau an ein Interface dieser Menge von Interfaces, entsprechend der Routing-Metrik, geliefert. In der Regel handelt es sich hierbei um das nächstgelegene Interface (siehe auch Abbildung 3-3). Anycast Adressen werden aus dem Unicast-Adressbereich allokiert, und sind deshalb syntaktisch gesehen nicht unbedingt von diesen zu Unterscheiden. Konten, deren Interfaces zu einer Anycast-Adresse gehören, müssen so konfiguriert sein, das sie eine Anycast-Adresse auch als solche identifizieren können. Anycast Adressen sollten nur als Zwischenstation, um z.B. eine Route für ein Paket über das Netz eines bestimmten Providers festzulegen, im optionalen Routingheader definiert werden, nicht aber als Quelladresse und/oder Zieladresse.

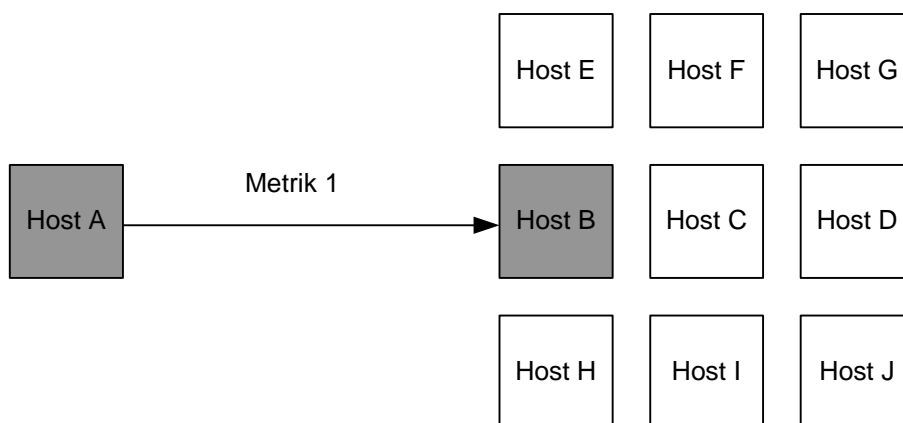
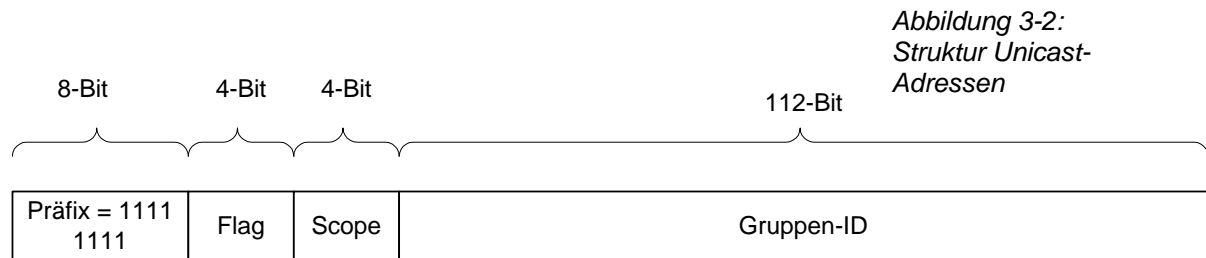


Abbildung 3-3:
Anycast-Kommunikation

Multicast-Adressen

IP-Multicast Adressen sind für die Gruppenkommunikation im Internet notwendig. Multicast erlaubt es, einem Sender ein Paket das er an eine Multicast-Adresse schickt, an alle Gruppenmitglieder zu schicken. Multicast-Adressen setzen sich aus einem 8-Bit großem Präfix, einem jeweils 4-Bit großem Flag- und Scope-Feld sowie einer 112-Bit großen Gruppen ID zusammen (siehe auch Abbildung 3-4).



Das Flag-Feld zeigt an, ob es sich um eine permanente Gruppe, also um eine durch die IANA registrierte Adresse handelt, oder um eine transiente, eine nicht ständig eingerichtete Multicast-Adresse. Das Scope-Feld kodiert die den Gültigkeitsbereich der Multicast-Gruppe. Dieser kann auf den Knoten, den Link, den Standort oder die Organisation beschränkt sein oder eine globale Gültigkeit besitzen. Zwei in unterschiedlichen Organisationen eingerichtete, transiente Gruppen mit dem Gültigkeitsbereich Organisation stehen in keinerlei Beziehung zueinander.

Appendix A Literaturverzeichnis

RFC 2460. (Dezember 1998) Autoren: **S. Deering (Cisco), R. Hinden (Nokia).**

IPv6 – The Next Generation. (November 1997) Autor: **Christian Huitema.**

Implementing IPV6 (März 2000) Autoren: **Mark Miller, P. E. Miller.**

Appendix B Abkürzungen

Abkürzung	Bedeutung:	Beschreibung:
ESP	Encrypted-Security-Payload	Beschreibt die Menge der verschlüsselten Sicherheitsnuzdaten.
ICMP	Internet-Control-Message-Protocol	Protokoll für Internet-Kontrollnachrichten
IHL	Internet-Header-Length	Feld unter IPv4 das die Länge des IPv4-Headers beschreibt.
IAB	Internet-Architecture-Board	Gremium das für die Architektur des Internets (mit-) verantwortlich ist.
IANA	Internet-Assigned-Number-Authority	Abteilung der IAB. Vergibt und kontrolliert fast alle numerischen Bezeichnungen, wie z.B. IP-Ports, für das Internet.
IETF	Internet Engineering Task Force	Organisation zur Festlegung von Internetstandarts
IP	Internet Protokoll	Transportschicht für Upper-Layer wie z.B. TCP, UDP, etc.
MTU	Media Transmission Unit	Beschreibt die Anzahl der maximalen Bytes pro Paket.
NAT	Network Address Translation	Unter Linux auch als IP-Masquerading bekannt. Beschreibt den Umstand das viele lokale Adressen eine oder mehrere globale Adressen zur Kommunikation mit dem Internet nutzen.
TCP	Transport-Controll-Protocol	Protokoll zum Verschicken von Daten
TOS	Type-Of-Services	Feld unter IPv4 das zur Angabe von Rountingpräferenzen genutzt wird
TTL	Time-To-Life	Feld unter IPv4 das die Lebensdauer eines IPv4 Pakets angibt.
UDP	User-Datagram-Protocol	Protokoll zum Verschicken von Daten