Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lehrstuhl für Rechnernetze und Telematik

WS 2007/2008

Seminararbeit

DHCP
Dynamic Host Configuration Protocol

Felix Ruzzoli

11. Dezember 2007

Betreut durch Prof. Dr. Christian Schindelhauer

Abstract

Ein informationsverarbeitendes System erreicht sein volles Potential erst durch eine Vernetzung mit anderen solchen Systemen. Zu diesem Zweck bedient man sich seit einiger Zeit schon diverser Netzwerkprotokolle, allen voran das heutzutage weit verbreitete TCP/IP[1]. Doch vor dem Datenaustausch steht noch die Konfiguration der Netzwerkschnittstellen. Um dieses Problem dynamisch zu lösen wurden verschiedene Vorschläge gemacht, von denen ich einen, das *Dynamic Host Configuration Protocol*, hier beschreiben möchte.

Inhaltsverzeichnis

[1] Transmission Control Protocol/Internet Protocol

1 Einleitung

Unsere Geschichte beginnt im Jahr 1984 an der Stanford University, wo zu dieser Zeit der Vorschlag zu einem *Reverse Address Resolution Protocol* formuliert wurde[8]. Es wurde festgestellt, dass zwar ein *Address Resolution Protocol* existierte um mittels Protokolladresse eines Computers dessen Hardwareadresse (MAC[2]-Adresse) zu ermitteln, allerdings für den umgekehrten Fall, beispielsweise das Booten eines Thin Clients keine Möglichkeit vorhanden war um mittels der Hardwareadresse eine gültige Protokolladresse zu erhalten. Das war die Geburtsstunde des *Reverse Address Resolution Protocol* und somit der automatisierten Netzwerkkonfiguration.

2 Von RARP über BOOTP zu DHCP

2.1 RARP (1984)

Wie der Beschreibung des *Reverse Address Resolution Protocol* oben vielleicht zu entnehmen ist, war das RARP als Umkehrung zum ARP angedacht und war daher relativ simpel aufgebaut. Das Protokoll setzt einen zentralisierten Server voraus, der eine Datenbank der MAC-Adressen mit zugehörigen Protokolladressen vorhält[8]. Die Clients können dann in einem vom ARP übernommenen Paketformat ihre Anfragen an den Server senden. Die Tatsache, dass das Paket Format von ARP übernommen wird, erlaubt es RARP als eine Art Schicht über dem normalen ARP zu betreiben, das heisst, von den 4 erlaubten Op-Codes im Paket werden nur 3 und 4 von RARP bearbeitet, während 1 und 2 als normale ARP anfragen an dieses Protokoll weitergereicht werden.
Wie erwähnt ist das Design relativ simpel gehalten und die Kommunikation beschränkt sich auf 2 Op-Codes:

3 - request reverse
Übertragen wird:

- Hardwareadresse des Paket Senders
- Die Hardwareadresse des Zielrechners
  (Falls die eigene Protokolladresse ermittelt werden soll, enthält dieses Feld ebenfalls die Hardwareadresse des Paket Senders)

4 - reply reverse
Übertragen wird:

[2] Media Access Control, eine quasieindeutige Kennung zur Identifikation von Netzwerkgeräten, besteht bei Ethernet-Netzen aus 48 Bits also sechs Bytes und wird in der Regel hexadezimal notiert.

- Die Hardwareadresse des Sender des Antwort Pakets
- Die Protokolladresse des Sender des Antwort Pakets
- Die Hardwareadresse des Zielrechners
  (sollte die gleiche sein wie in der Anfrage)
- Die Protokolladresse des Zielrechners (die gewünschte Antwort)

Wir finden hier beim *Reverse Address Resolution Protocol* schon viele Details, die wir später auch bei DHCP wiederfinden werden, unter anderem die Client/Server Architektur, (wobei hier jeder Teilnehmer, der Antwortpakete auf Anfragepakete generiert als Server bezeichnet wird), die simple Kommunikation über wenige Datenpakete und das Ausnutzen eines vorhandenen broadcast Mediums, bei RARP z.B. Ethernet.

2.2 BOOTP (1985)

Hier hatte man schon größere Pläne für das frischgebackene Protokoll. Das **BOOT***strap* **P***rotocol* sollte Thin Clients[3] eine automatisierte Konfiguration der Netzwerkschnittstellen und ein nachfolgendes Herunterladen eines boot Images von einem passenden Server gestatten. Im RFC[4]951 [4] wird die Aufgabe grob in zwei Bereiche unterteilt von denen es nur den ersten, 'die Adressermittlung und Bootdatei Auswahl' behandelt und beim zweiten Teil auf das dafür verwendete TFTP[5] verweist, welcher dann letztlich nur noch den Transfer und in Gang setzen des Bootvorgangs zu erledigen hat.
Viele Dinge sind gegenüber RARP gleich geblieben, z.B. der Austausch weniger Pakete, nur zwei verschiedene Op-Codes (*request* und *reply*) und natürlich der Client/Server Ansatz. Es gibt jedoch auch einige Änderungen gegenüber RARP. So nutzt BOOTP nun IP/UDP Pakete um seine Informationen auszutauschen, erlaubt das zusätzliche Anfordern einer Bootdatei und unterstützt vor allem das Weitervermitteln von Anfragen über Gateways.

2.3 Andere relevante Technologien

Das RFC1531 (DHCP, proposed Standard, 1993) hält sich nicht zurück wenn es darum geht die Gründe für das neue Protokoll trotz der Menge schon geleisteter Arbeit in dieser Richtung zu verteidigen. Jedoch haben alle nachfolgend beschriebenen Protokolle und Standards nur ein Teil der Autokonfigurationsproblematik gelöst, eine Technik die alle

[3] Hiermit sind generell alle Computersysteme gemeint, welche nicht über einen eigenen lokalen Massenspeicher verfügen um von dort das Betriebssystem zu laden.
[4] Request for Comments
[5] Trivial File Transfer Protocol

vorausetzungen für die vollständige Konfiguration eines Internet Hosts erfüllt gab es bis zu DHCP nicht.

- RARP (und später auch DRARP) waren explizit für die Lösung des Problems der Netzwerkadressen-Erkennung und automatischen IP-Adressen-Vergabe gedacht.
- TFTP stellt Mittel zur Verfügung um Bootdateien von einem Bootserver zu einem Client zu transportieren.
- ICMP[6] erlaubt es einem Rechner andere Hosts via einer *ICMP redirect* Nachricht über neu hinzugekommene Router zu informieren. Es ist sogar möglich mittels einer *ICMP mask request*-Anfrage Informationen über die verwendete Subnetzmaske zu erhalten. Außerdem können Router durch *ICMP router discovery* gefunden werden.
- BOOTP ist ein Transportmechanismus für eine Sammlung von Konfigurationsinformationen. Es wurde später noch eine Erweiterung um dynamische IP-Adressenvergabe vorgeschlagen.
- NIP[7] wurde vom Athena Projekt am MIT verwendet und ist ein verteilter Mechanismus für dynamische IP-Adressvergabe.
- RLP[8] kümmert sich um das Auffinden von Services auf höheren Schichten.
- Sun Microsystems Thin Clients benutzen ein Bootprozedur, welche auf RARP, TFTP und einen RPC[9] Mechanismus der *bootparams* genannt wird, beruht.

3 DHCP (1993-2006)

Das *Dynamic Host Configuration Protocol* wurde eingeführt um die Vorteile schon existierender Autokonfigurationsprotokolle miteinander zu vereinen und einen Standard für diese Prozedur vorzustellen. Bis zu diesem Zeitpunkt nutzte jeder eine eigene Kombination aus proprietären Protokollen, freien Protokollen und anderen „Workarounds“ um eine vollständige Konfiguration eines Netzwerk Hosts zu realisieren.
Als man die ersten Entwürfe für das DHCP vorlegte, wollte man auf jeden Fall die Abwärtskompatibilität zu vorhandenen BOOTP Clients gewährleisten[5]. Das ist auch der Grund, warum DHCP das Paketformat von BOOTP weiterverwendet und seine Nachrichten nur über das *Options* Feld kennzeichnet. Außerdem wurde mit DHCP das

[6] Internet Comtrol Message Protocol
[7] Network Information Protocol
[8] Resource Location Protocol
[9] Remote Procedure Call

Lease[10]-Konzept für IP-Adressen eingeführt. Dies ermöglicht es, diesselben IP-Adressen kurz nacheinander an unterschiedliche Clients zu vergeben, ein vor allem für Laptops und andere mobile Netzwerkgeräte in Umgebungen mit begrenzten IP-Adressen Vorrat nicht von der Hand zu weisender Vorteil.
Wenn nun DHCP die vollständige Konfiguration eines an ein Netzwerk angeschlossenen Computers übernehmen soll, muss erst einmal definiert werden, welche Konfigurationsparameter ein solcher Rechner überhaupt benötigt. Die Antwort darauf liefern die *Host Requirements RFCs*[2, 1].

> „Be liberal in what you accept, and conservative in what you send“[2]

Wir finden das absolute Minimum an Informationen, die ein Host zum Routen von Paketen nach außen benötigt in dem Dokument unter Internet Layer → Specific Issues → Routing Outbound Datagrams → Initialization:

> The following information MUST be configurable:
>
> (1) IP address(es).
>
> (2) Address mask(s).
>
> (3) A list of default gateways, with a preference level.

Das heißt wir benötigen also auf jeden Fall diese drei Informationen um erfolgreich mit der Außenwelt kommunizieren zu können. Im Folgenden werde ich beleuchten wie DHCP an diese Informationen gelangt.

3.1 IP-Adressenermittlung über das IP Protokoll

Die Antwort darauf, wie es möglich ist UDP Nachrichten, welche ja bekanntlich in IP-Pakete gekapselt sind, zu verwenden um sich selbst eine IP-Adresse zuteilen zu lassen, ist relativ einfach: Es wird eine vom IP-Standard definierte *limited broadcast* Adresse als Zieladresse verwendet, diese spezielle IP-Adresse, welche ausschliesslich aus Einsen besteht (255.255.255.255) kann von IP-Software empfangen und gesendet werden, selbst wenn die eigene lokale IP-Adresse noch nicht bekannt ist. *limited broadcast* wird sowohl für Anfragen der Clients als auch die erste Antwort vom Server verwendet, da zu diesem Zeitpunkt die IP-Adresse des Clients noch nicht in den ARP Caches vorliegt. Broadcasting stellt daher die einzige Möglichkeit zur Paketvermittlung zu diesem Zeitpunkt der Initialisierung dar.

[10] *leasing* bezeichnet dabei im Prinzip ein „Ausleihen“ der IP-Adresse.

3.2 Maßnahmen gegen die Unzuverlässigkeit von UDP

Wie man weiß, sind UDP-Pakete eine weitaus unkontrolliertere Form der Datenübertragung als TCP-Pakete. Es gibt keine Flusskontrolle, keine Verlässlichkeit, keine Garantie für Zeitnähe, etc. Datenpakete können außerdem fehlerhaft beim Zielsystem ankommen. Um alle diese Unwägbarkeiten zu umschiffen setzt DHCP die Verwendung von UDP-Kontrollsummen voraus. Außerdem wird das *do not fragment* Bit bei den UDP-Paketen gesetzt um so Clients mit wenigen Ressourcen das eventuelle Zusammenfügen der Pakete zu ersparen. DHCP geht auf recht simple aber effektive Weise mit mehreren erhaltenen Antworten um: Es akzeptiert und verarbeitet schlicht immer die Erste die eintrifft[3].
Der Umgang mit Paketverlust wird mittels der konventionellen Technik des *timeout and retransmission* gelöst. Hierbei wird vom Client gleichzeitig mit dem Versenden einer Anfrage ein Timer gestartet und falls dieser ausläuft ohne dass eine Antwort erhalten wurde, muss eine erneute Anfrage gesendet werden. Um zum Beispiel nach einem simultanen Ausfall mehrerer Rechner und deren Neustart nicht unnötige Verzögerungen und Netzlast zu erzeugen, wenn alle Clients ihre Anfragen zum gleichen Zeitpunkt stellen würden, spezifiziert DHCP, dass zufällige Verzögerungen gepaart mit einer Verdopplung der Wartezeiten verwendet werden sollen, um so eine zeitliche Verteilung der Anfragen sicherzustellen und die Server zu entlasten.

3.3 Das Format von DHCP Nachrichten

Das Nachrichtenformat von DHCP ist einfach und baut auf dem von BOOTP auf. Anfragen der Clients, wie auch Antworten der Server nutzen das gleich Format. Um eine einfache Implementierung zu ermöglichen sind alle Daten in Feldern fester Größe abgelegt[6].

Abbildung 1: In Klammern die Größenangaben in Bit

op (8)	htype (8)	hlen (8)	hops (8)
xid (32)			
secs (16)		flags (16)	
ciaddr (32)			
yiaddr (32)			
siaddr (32)			
giaddr (32)			
chaddr (128)			
sname (512)			
file (1024)			
options (var.)			

- **op** - Ist der *Operation Code* und zeigt ob die Nachricht ein *request*(1) oder ein *reply*(2) ist.
- **htype, hlen** - Spezifizieren den Hardwaretyp des Netzwerks und die Länge der Hardwareadressen (genau wie bei ARP)
- **hops** - Clients setzen diesen Wert immer auf 0. Diesen Feld erlaubt Servern eine Weiterleitung und somit das Bootstrapping über mehrere Router hinweg.
- **xid** - Transaction ID, ein Integerwert, der der Zuordnung von Antworten zu Anfragen dient. Ein Client setzt diesen Wert zufällig.
- **secs** - Die Anzahl an Sekunden, die seit dem Beginn der Adressakquisition des Clients vergangen sind.
- **flags** - Siehe weiter unten.

[3]Das 16 Bit breite **flags** Feld dient momentan ausschliesslich der Kodierung einer 1 Bit großen Information, die restlichen Bits sind reserviert. Ein Client kann damit bestimmen

ob die Antworten eines DHCP Servers als Broadcast (Broadcast Flag gesetzt) oder als Unicast (Broadcast Flag nicht gesetzt) gesendet werden sollen. Der Grund dafür ist, das einige Hosts keine IP Unicast Pakete empfangen können bevor ihre TCP/IP Software komplett konfiguriert ist.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|B|             MBZ             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

B:  BROADCAST flag

MBZ:  MUST BE ZERO (reserved for future use)
```

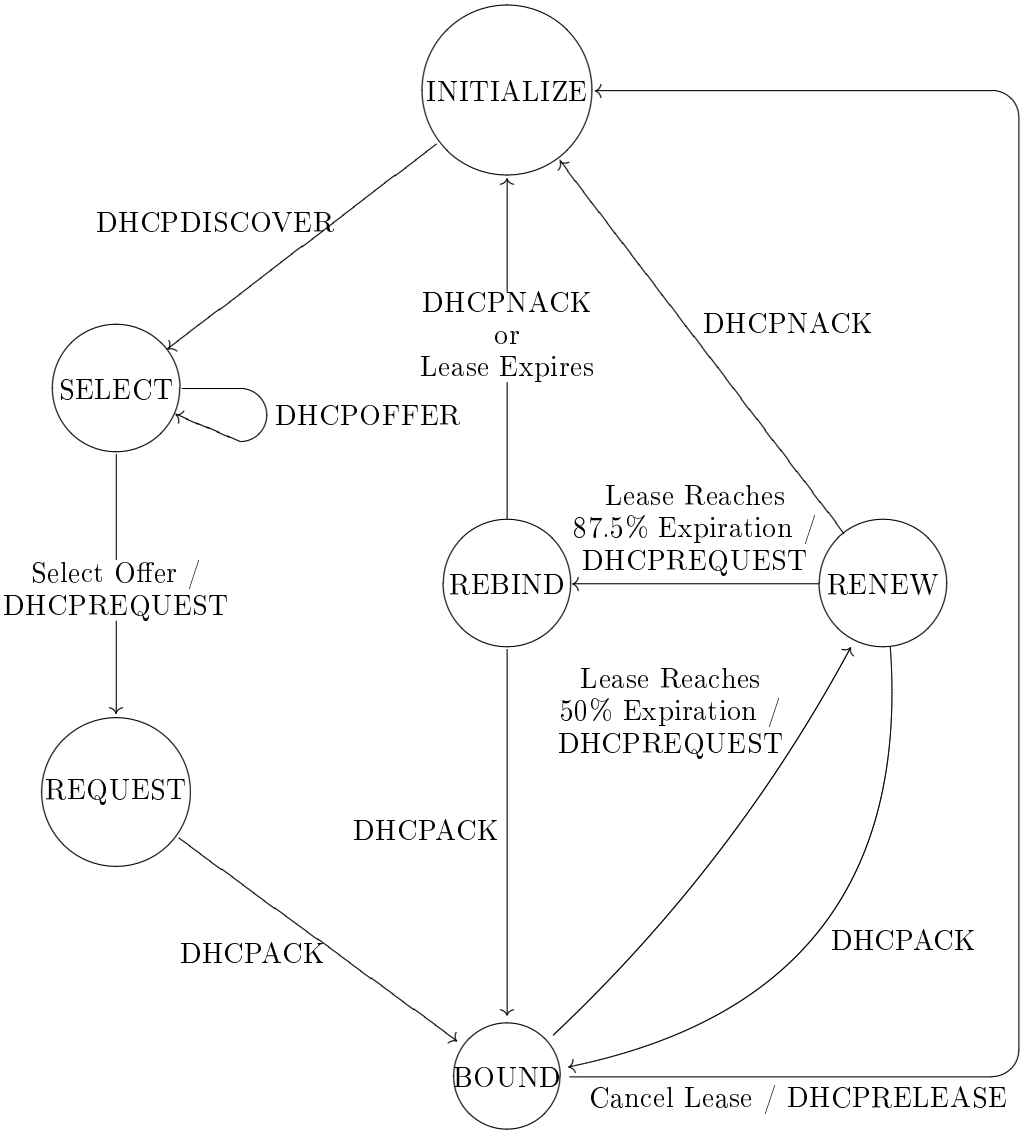
Nun kommen die für DHCP wichtigsten Felder, welche zur Erhöhung der Flexibilität vom Client wahlweise ausgefüllt oder auf Null gesetzt werden können. Auf diese Weise kann ein Client gezielt sein Datenbestände um fehlende Konfigurationsparameter ergänzen. Beispielsweise kann ein Client durch setzen des *Server IP Address* oder des *Server Host Name* Feldes einen bekannten DHCP Server für seine Anfragen auswählen. Entsprechend würde in diesem Fall natürlich auch nur ein Server mit passender IP-Adresse bzw. Hostnamen antworten. So kann auch ein Client, welcher seine IP-Adresse schon kennt, über einen DHCP weitere Informationen bezüglich seines Netzes erfragen.

- **ciaddr** - Client IP Address (falls IP-Adresse schon bekannt)
- **yiaddr** - Your IP Address (Die vom Server vergebene IP-Adresse in der Antwort)
- **siaddr** - Server IP Address (Die IP-Adresse des nächsten zu verwendenden DHCP Servers)
- **giaddr** - Router IP Address (findet Verwendung beim Bootstrapping über Router hinweg)
- **chaddr** - Client Hardware Address (normalerweise die MAC-Adresse des Clients)
- **sname** - ein optionaler Server Hostname als Null terminierter String
- **file** - Name einer Bootdatei als Null terminierter String

3.4 Zustandsdiagramm für die Ermittlung einer IP-Adresse

Ein Client, welcher DHCP zum ermitteln seiner IP-Adresse verwendet, befindet sich jederzeit in einem von 6 verschiedenen Zuständen.

Abbildung 2: Zustandsdiagramm eines DHCP Clients



Nach dem Bootvorgang befindet sich ein Client typischerweise im INITIALIZE Zustand. Um eine Anfrage nach einer IP-Adresse stellen zu können muss der Client nun erst einmal

die DHCP Server im lokalen Netzwerk ermitteln. Dies wird mittels Broadcasting einer DHCPDISCOVER Nachricht erledigt. Als Zielport dient der BOOTP Port (67). Daraufhin wechselt der Client in den SELECT Zustand und wartet auf DHCPOFFER. Es können nun null oder mehr DHCPOFFER Nachrichten beim Client ankommen. Dieser muss sich nun für eine davon entscheiden, in der Regel schlicht die Erste und mit einem DHCPREQUEST antworten. Der Zustand wechselt dabei zu REQUEST und es wird auf eine Bestätigung, in Form eines DHCPACK, vom Server gewartet. Kommt nun das erwartete DHCPACK beim Client an, so wechselt dieser in den BOUND Zustand und beginnt die Nutzung der zugeteilten IP-Adresse.
Dieses Modell erlaubt den Clients außerdem ihre IP-Adressen vorzeitig wieder freizugeben, falls sie nicht mehr benötigt werden. Dies geschieht über eine DHCPRELEASE Nachricht vom Client an den Server. Daraufhin muss die Verwendung der betroffenen IP-Adresse sofort eingestellt werden und der Client gegebenfalls mit dem INITIALIZE Zustand neu beginnen.
Ähnlich einfach ist auch eine Verlängerung der *Leasetime* realisiert. Bei Beginn der Nutzung einer IP-Adresse, also beim Wechseln in den BOUND Zustand, werden vom DHCP Client drei Timer für die Erneuerung, Neubindung und den Auslauf des *Lease* gestartet. Diese Timer laufen der Reihe nach ab und sobald der Erste abgelaufen ist, muss ein ein Client sich um eine Erneuerung der IP-Adresse kümmern. Er fragt dazu mittels DHCPREQUEST den DHCP Server von dem er den *Lease* erhalten hat nach einer Verlängerung der Nutzungszeit der gerade zugeteilten IP-Adresse. Er geht anschliessend in den RENEW Zustand üeber und wartet auf Antwort. Erreicht ihn keine Antwort vom Server so wird in den REBIND Zustand gewechselt und ein neuer DHCPREQUEST Broadcast gesendet um einen anderen DHCP Server zu finden. Gelingt auch dies nicht, bevor der letzte Timer für das Auslaufen der *Leasetime* abgelaufen ist, so muss der Client die Nutzung der IP-Adresse einstellen.

3.5 Die Nachrichtentypen

Die Nachrichtentypen bei DHCP werden aus Gründen der Abwärtskompatibilität zu BOOTP im *Options* Feld kodiert. DHCP unterscheidet dabei zwischen 8 verschiedenen Typen.

- **DHCPDISCOVER (1)**
  Client Broadcast um vorhandene Server zu finden
- **DHCPOFFER (2)**
  Server an Client, Antwort auf DHCPDISCOVER, Konfigurationsparameter werden angeboten

- **DHCPREQUEST (3)**
  Client an Server, Anfrage der angebotenen Parameter,
  Bestätigung der Korrektheit einer vorher zugewiesenen Adresse oder verlängern der *leasetime*
- **DHCPDECLINE (4)**
  Client teilt dem Server mit, dass die angebotene Adresse bereits verwendet wird
- **DHCPACK (5)**
  Server teilt dem Client die Konfigurationsparameter inklusive der zugeteilten IP-Adresse mit
- **DHCPNACK (6)**
  Server an Client, die verwendete Netzwerkadresse ist nicht länger gültig
- **DHCPRELEASE (7)**
  Client an Server, Abbruch des *Lease*, IP-Adresse wird freigegeben
- **DHCPINFORM (8)**
  Client fragt den Server nur nach lokalen Konfigurationsparametern, IP-Adresse wurde auf anderem Wege ermittelt

4 Die Zukunft oder Autokonfiguration bei IPv6

Bei IPv6 hat man sich schon ziemlich früh Gedanken um eine automatische Konfiguration der Netzwerkschnittstellen gemacht, wahrscheinlich weil man den Nutzen von DHCP bei IPv4 erkannt hatte. Die ersten Ideen, was die automatisierte Schnittstellen Konfiguration für IPv6 angeht, stammen von 1996 [9]. Hier sprich man noch von *Neighbor Discovery*[9], eine Erweiterung des ARP um benachbarte Hosts in einem Netzwerk ausfindig zu machen und die Erreichbarkeit prüfen zu können. Kurz danach wird dann die Autokonfiguration von IPv6 Hosts definiert[10]. Die Spezifikation durchläuft mehrere kleinere Änderungen, auch die Terminologie wird später noch einmal abgeändert. Im Gegensatz zur *Stateless Address Autoconfiguration*, welche einzelnen Hosts ohne zentralen Server eine autarke Konfiguration der Netzwerkadresse erlaubt, wir nun nicht mehr von *Stateful Configuration* gesprochen, sondern einfach der Terminus DHCPv6[11] verwendet[11]. Hierbei ist dann die vom DHCP für IPv4 her bekannte Client/Server Architektur des Protokolls gemeint, bei der die Adressvergabe zentral verwaltet werden kann.
Im folgenden soll kurz auf die Funktionsweise der beiden verschiedenen Methoden der automatisierten Konfiguration unter IPv6 eingegangen werden.

[11] Dynamic Host Configuration Protocol for IPv6

4.1 DHCPv6 (2003)

Bei der IPv6 Variante von DHCP fällt als erstes auf, dass die Art der Kommunikation zwischen Server und Client nicht näher spezifiziert wird. Es wird davon ausgegangen, dass schon Möglichkeiten existieren um die DHCP Pakete auszutauschen[7]. Allerdings fordert das Protokoll aus Performanzgründen UDP Pakete. DHCPv6 ist als Ergänzung zur *Stateless Address Autoconfiguration* zu sehen, es stellt zusätzlich für Netzwerkclients nützliche Informationen zur Verfügung, so zum Beispiel Adressen von DNS[12] oder NTP[13] Servern. Grundsätzlich ist DHCPv6 nur in Details von DHCP bei IPv4 verschieden, es gibt zwar mehr Optionen und deutliche Unterschiede, was die Herangehensweise angeht, da man ja noch auf die *Stateless Address Autoconfiguration* zurückgreifen kann, jedoch ist ansonsten vieles gleich geblieben. Das Protokoll ist allerdings auf jeden Fall deutlich komplexer geworden, was man schon an der stark vergrößerten Anzahl an Op-Codes erkennen kann, von gerade einmal 2 verschieden Op-Codes bei RARP und BOOTP, über die 4 verschiedenen Nachrichtentypen bei DHCP, gibt es bei DHCPv6 jetzt 13 verschiedene Nachrichtentypen für die Kommunikation zwischen Server und Client.

4.2 Stateless Address Autoconfiguration (2007)

Hier zeigt sich erst ein grundlegender Unterschied zur alten Verfahrensweise bei IPv4. Ein Router in einem IPv6 Netz verbreitet regelmäßig eine *Prefix* genannte Kennung des Subnetzes mit Hilfe derer dann einzelne Hosts unter zuhilfenahme eines *interface identifiers* eine einzigartige Adresse in diesem Netz generieren[11]. Falls in dem betrachteten Netz keine Router existieren, kann nur eine für das lokale Netzwerk gültige IP-Adresse generiert werden, was jedoch für die Kommunikation mit Hosts die am selben Netz hängen ausreichen würde.
Dieser zustandslose Ansatz wird vor allem dann verwendet, wenn es in dem Netzwerk keine Rolle spielt welche Adressen genau zugeteilt werden, sondern einfach nur Einzigartigkeit und Gültigkeit (im Sinne von weiterleitbar) garantiert sein soll.
Falls man mehr Kontrolle über die automatisierte Adressvergabe wünscht empfiehlt sich der Einsatz von DHCPv6.

[12] Domain Name System
[13] Network Time Protocol

Literatur

[1] R. Braden. Requirements for Internet Hosts - Application and Support. RFC 1123 (Standard), October 1989. Updated by RFCs 1349, 2181.

[2] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), October 1989. Updated by RFCs 1349, 4379.

[3] Douglas E. Comer. *Internetworking with TCP/IP*, volume 1: Principles, Protocols and Architecture. Prentice-Hall, Englewood Cliffs, New Jersey, 4th edition, January 2000.

[4] W.J. Croft and J. Gilmore. Bootstrap Protocol. RFC 951 (Draft Standard), September 1985. Updated by RFCs 1395, 1497, 1532, 1542.

[5] R. Droms. Dynamic Host Configuration Protocol. RFC 1531 (Proposed Standard), October 1993. Obsoleted by RFC 1541.

[6] R. Droms. Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard), March 1997. Updated by RFCs 3396, 4361.

[7] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315 (Proposed Standard), July 2003. Updated by RFC 4361.

[8] R. Finlayson, T. Mann, J.C. Mogul, and M. Theimer. A Reverse Address Resolution Protocol. RFC 903 (Standard), June 1984.

[9] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 1970 (Proposed Standard), August 1996. Obsoleted by RFC 2461.

[10] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 1971 (Proposed Standard), August 1996. Obsoleted by RFC 2462.

[11] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), September 2007.

Abbildungsverzeichnis