Ludwig-Maximilians-Universität München
Institut für Informatik

Erstellung eines Testprogramms für den DHCP-Server

Fortgeschrittenenpraktikum am Lehrstuhl für Rechnernetze und Systemprogrammierung

Bearbeiter: Hussein May
Betreuer: Stephen Heilbronner
Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Abgabetermin: 20.Juli.1998

Inhaltsverzeichnis

Kapitel 1. Einführung

1.1 Motivation

Nicht ganz unproblematisch, aber immer wichtiger ist der Einsatz von TCP/IP, TCP/IP ist nicht nur Mittel der Wahl in großen Netzen, sondern auch in kleinen Netzen als Grundlage für Internet- und Intranet-Anwendungen essentiell. TCP/IP ist aber nicht nur durch seine Schnelligkeit und Flexibilität bekannt, sondern auch durch seine Komplexität: angefangen beim Schema der Adressierung mit Subnetzen über eine Fülle von zusätzlichen Protokollen wie DHCP gibt es so viele Konfigurationsmöglichkeiten, daß TCP/IP-Systemkonfiguration schnell unübersichtlich wird.
Es gibt jedoch ein wesentliches Problem, das es zu lösen gilt: die IP-Konfiguration.
Die Konfigurationsprotokolle für TCP/IP-Systeme (RARP, BOOTP) haben den Nachteil, daß sie sich nur bedingt bzw. überhaupt nicht für den Einsatz mit mobilen Computersystemen eignen. Als Abhilfe für diese Problemsituation wurde eine Erweiterung des BOOTP-Protokolls entwickelt. Das wichtigste Protokoll für die Lösung der IP-Konfiguration ist DHCP (Dynamic Host Configuration Protocol) .

1.2 Aufgabenstellung

Das Ziel dieses Fortgeschrittenenpraktikums bestand darin, einen DHCP-Client in Java zu implementieren. Der DHCP-Client sollte die folgenden Anforderungen erfüllen:

1. Darstellung des Client-Zustands
   → d.h. den aktuellen Zustand (z.B. INIT, SELECTING etc.) des Client anzuzeigen

2. Einzustellende Parameter:
   - Die Ports, die der DHCP-Server und der -Client zum Nachrichtenaustausch verwenden, können eingestellt werden (Standardports sind: PortNr. 67: auf diesem Port werden die Nachrichten vom DHCP-Client zum -Server gesendet, PortNr. 68: auf diesem Port werden die Nachrichten vom Server zum Client gesendet).

   - Die Hardwareadresse des Clients und die Serveradresse einzustellen.

   - Die “Lease Time“ vorzuschlagen.

   - Die Parameter “class identifier“ und “client identifier“ beim DHCP-Anfragen einzustellen. Der Client kann dieser Parameter verwenden, um sich bei dem Server zu identifizieren.

3. Ausgabe ein Teil der Konfigurationsparameter
   → d.h. Teil der vom DHCP-Server gelieferten Konfigurationsparameter auszugeben

4. Ein- Ausgehende Nachrichten
   → d.h. Die Nachrichten, die zwischen dem DHCP-Client und DHCP-Server ausgetauscht werden, anzuzeigen (z.B. DHCPREQUEST, DHCPOFFER, DHCPACK etc.)

1.3 Gliederung und Aufbau der Ausarbeitung

In Kapitel 2 wird das Konfigurationsprotokoll DHCP aufgrund seiner Definition vorgestellt, um eine Grundlage für das Verständnis des Vorgehens zu erhalten.

In Kapitel 3 kann man nun genauer die Szenarien untersuchen, unter denen DHCP eingesetzt werden kann.

In Kapitel 4 wird ein Modell für die Realisierung des DHCP-Client entwickelt, dann wird die Benutzung des DHCP-Clients und die Tests, die mit dem DHCP-Client ausgeführt wurden, beschrieben.

Kapitel 2. Das DHCP-Protokoll

2.1 Einleitung

Das Internet spielt bereits heute eine wichtige Rolle in der Industrie und in der Wissenschaft. Firmen setzen Computer für ihre Zwecke ein und vernetzen sie. Grundlage für diese Vernetzung ist eine Protokollvereinbarung namens TCP/IP (*Transmission Control Protocol / Internet Protocol*), welche bei der Mehrzahl der heutigen Datenkommunikationsnetze als Grundlage für die Datenübertragung dient.
Um eine möglichst effiziente Nutzung solcher Computer zu ermöglichen, strebt man es häufig an, daß Ressourcen wie Drucker oder Datenspeichergeräte über das Netz aus genutzt werden können. Ebenso sollen Computer, die von Benutzern bedient werden sollen, bestmöglich ausgelastet werden [Rie96].
Auf einem DHCP-Server werden IP-Konfigurationen definiert, die automatisch den Clients zugeordnet werden. Wenn der Client dann gestartet wird, sucht er im Netzwerk nach einem DHCP-Server und erhält von diesem die IP-Konfiguration mit seiner IP-Adresse und anderen wichtigen Parametern.
TCP/IP läßt sich nicht so einfach konfigurieren wie einige andere Protokolle. Dafür bietet TCP/IP auch eine ganze Menge mehr an.

- Es ist ein Protokollstandard, der nicht abhängig von bestimmten Betriebssystemen oder Computerhersteller ist.

- TCP/IP ist unabhängig von der verwendeten Hardware.

- Es gibt ein standardisiertes Adressierungsverfahren.

- Die Vernetzung von Netzen und das Routing zwischen diesen Netzen werden unterstützt.

Diese Charakteristiken tragen zum Nutzen und zur Flexibilität, aber auch zur Komplexität von TCP/IP bei. TCP/IP benötigt für die Konfiguration Information über die verwendete Hardware, die Adressierung und das Routing. Da es keine Abhängigkeit von einer bestimmten zugrundeliegenden Netzwerkhardware gibt, kann bei TCP/IP ein Teil der benötigten Informationen nicht durch die Hardware der Netzkomponenten festgelegt werden.
Die Informationen müssen komplett von einer Person oder System eingegeben werden, die für die Konfiguration verantwortlich ist. Das setzt voraus, daß jedes System von Leuten betrieben wird, die über genug Wissen verfügen, um die Rechner zu konfigurieren. Leider ist diese Voraussetzung nicht immer gegeben.
Ursprünglich war TCP/IP als ein robustes Netzwerk zur Verbindung von professionell betriebenen Großrechnern und Minicomputern gedacht. Die Computer in einem TCP/IP-Netzwerk werden als *Peers* angesehen - als Gleiche unter Gleichen , die für manche Anwendung als Server dienen, während sie gleichzeitig andere Anwendungen als Clients nutzen. TCP/IP macht dabei keinen Unterschied zwischen Großrechnern und PC's. Unter TCP/IP sind das alles *Hosts* - und alle benötigten dieselben grundlegenden Einstellungen [Hun96].
Natürlich gibt es eine Reihe von Protokollen, Softwaretools, die dazu beitragen, die Konfiguration zu vereinfachen. In diesem Kapitel wird 1 Tool dargestellt.

2.2 DHCP

Das *Dynamic Host Configuration Protocol* (DHCP) ist eine Erweiterung des BOOTP-Protokolls. Bei der Entwicklung wurde auf die Kompatibilität zu den älteren Versionen von BOOTP geachtet. Das Zusammenspiel von BOOTP-Clients und DHCP-Server sowie umgekehrt zwischen DHCP-Clients und BOOTP-Server wird in RFC 1534 definiert.
DHCP ist aber mehr als nur update von BOOTP - es erweitert BOOTP in den folgenden zwei Punkten:

- Die durch DHCP zur Verfügung gestellten Konfigurationsparameter enthalten alle Merkmale, die im RFC genannt werden. DHCPliefert dem Client alle Konfigurations-parameter der TCP/IP.

- DHCP erlaubt die automatische Vergabe von IP-Adressen.

DHCP nutzt den ursprünglich für herstellereigene Erweiterungen vorgesehen Teil des BOOTP-Paketes für die Angabe des DHCP-Pakettyps; die kompletten Konfigurationsinformationen sind ebenfalls darin untergebracht. Die Werte in diesem Teil des Paketes werden unter DHCP als Optionen und nicht mehr als herstellereigene Erweiterungen bezeichnet. Diese Bezeichnung ist auch exakter, denn DHCP legt genau fest, wie diese Optionen genutzt werden und läßt den Herstellern keinen Spielraum für eigene Definitionen. Um alle Informationen aus dem RFC unterbringen zu können, wurde das Feld für die Optionen von den ursprünglichen 64 Bytes auf nunmehr 312 Bytes erweitert ([Hun96], [Ale93]).

Sowohl DHCP als auch BOOTP erlauben die Zuweisung von Adressen auf Manuelle und Automatische Vergabe der IP-Adressen:

Manuelle Vergabe:

Der Netzwerkverwalter behält die Kontrolle über die Vergabe von Adressen, indem er sie selbst den Clients zuordnet ([Hun96], [Hei96]).

Automatische Vergabe:

Für die meisten Netzwerkadministratoren ist die automatische Vergabe von IP-Adressen eine interessante Möglichkeit.
Der DHCP-Server weist permanent den Clients eine Adresse aus einem Pool von verfügbaren Adressen zu. Der Administrator wird in die Details der Zuweisung von Adressen an die Clients nicht mit einbezogen ([Hun96], [Hei96]).

Eine wesentliche Neuerung in DHCP ist das Konzept des “IP-Adreß-Leasings“, d.h. die dynamische Vergabe von IP-Adressen an Client-Systeme für eine bestimmte Zeit:

Dynamische Vergabe:

Der Server weist dem DHCP-Client nur für eine „Lease“ eine bestimmte Adresse zu. Der Client kann die Adresse zu jeder Zeit an den Server zurückgeben, um die Adresse länger als für die erlaubte Zeit behalten zu können, muß der Client beim Server eine Verlängerung anfordern. Der Server zieht die Adresse nach Ablauf der erlaubten Benutzungszeit automatisch wieder ein, wenn keine Verlängerung beantragt wurde ([Hun96], [Hei96]).

2.3 Beschreibung von DHCP

Angelehnt an das Abstraktionsmodell für Managementanwendungen, die man dabei in Modelle der Organisation, Information, Funktion und an der zugrundeliegenden Kommunikation unterteilt, wird hier versucht, eine Beschreibung des DHCP-Protokolls zu geben.

2.3.1 Organisationsmodell

DHCP ist Client-Server-artig organisiert. Auf der einen Seite steht der DHCP-Server, der die Konfiguration-Informationen für ein oder mehrere Subnetze bereitstellt. Auf der anderen Seite steht der DHCP-Client. Dieser ist ein Hostsystem in einem Subnetz, das sich vom DHCP-Server die Informationen für seine Konfiguration sowie eine IP-Adresse geben läßt. Sollten DHCP-Server und DHCP-Client sich auf unterschiedlichen Subnetzen befinden, sorgt ein sogenannter *Relay Agent* für die Weiterleitung der Broadcast-Meldungen zwischen Server und Client [Hei96], [Com95], [Dro97].
Er übernimmt also gewissermaßen eine Router-Funktionalität auf DHCP-Ebene. Der Einsatz von Relay Agents hat den Vorteil, daß nicht für jedes Subnetz ein eigener DHCP-Server zur Verfügung gestellt werden muß. Andererseits besteht die Gefahr, daß beim Ausfall eines *Relay Agents* oder eines DHCP-Servers die Clients nicht in der Lage sind, am Netzwerkbetrieb teilzuhaben. Es ist also auf jeden Fall sinnvoll, sich über den Einsatz von Relay Agents genaue Gedanken zu machen und gegebenenfalls Rechner als Backup-DHCP-Server oder Backup-*Relay Agents* zu konfigurieren. Ein *Relay Agent* wird entweder in einen Internet-Router oder einen für diesen Zweck konfigurierten Rechner implementiert. DHCP ist weder für die Registrierung von neu konfigurierten Rechnern im Domain-Name-Server (DNS) noch für die Konfigurierung von (IP-) Routern konzeptioniert [Rie96].

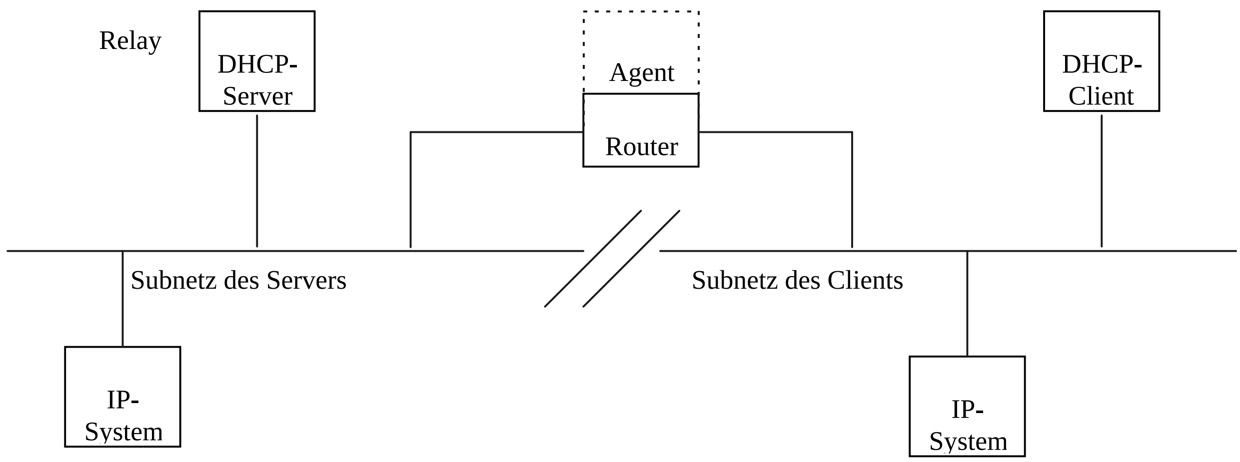


Abbildung 2.1: Kommunikation DHCP-Server - DHCP-Client [Rie96]

2.3.2 Informationsmodell

Die Informationen für die Konfiguration sowie die IP-Adresse werden dem DHCP-Client Mithilfe der sogenannten DHCP-Options mitgeteilt. Diese werden an Dienstdateneinheit /SDU’s angehängt, die zwischen DHCP-Server und DHCP-Client ausgetauscht werden. Die Länge des Option-Teils einer DHCP-SDU ist dabei variabel mit einer Höchstlänge von 312 Oktetten.
Welche Konfigurationsinformationen bzw. IP-Adresse zur Verfügung stehen, entnimmt ein DHCP-Server einer Datei mit Konfigurationsinformationen, auf die er lesend zugreift. Es wird weiterhin im Rahmen des Protokolls davon ausgegangen, daß sowohl beim Server als auch der Client eine konsistente Datenhaltung existiert, d.h. sowohl der Server als auch der Client merken sich die aktuelle Konfiguration des Clients. Dies geschieht beim Server innerhalb einer Datenstruktur zur Laufzeit wie auch innerhalb einer Datenbasis, die als Schutz vor Ausfällen des Serverprogramms in einem File niedergelegt wird. Das genaue Format dieser Datenbasis und der Datenstruktur wird aber vom Protokoll nicht vorgeschrieben; das Format hängt von der Implementierung innerhalb der DHCP-Server-/Clientsoftware ab [Rie96].

2.3.3 Funktionsmodell

Die Funktionen von DHCP werden durch SDU’s erbracht, die unter DHCP Namen tragen, welche das durch ihre Funktion kennzeichnen. Bei DHCP gibt es zwei Arten, welche sich durch ihre Verwendung entweder durch den DHCP-Server oder durch den DHCP-Client unterscheiden.[Dro97], [Hei96].

Nachrichten-Typen des DHCP-Clients
Der DHCP-Client verwendet folgende Nachrichten-Typen:

DHCPDISCOVER: Nachricht, die ein Client als Broadcast sendet, um das Netz auf verfügbare DHCP-Server zu testen.

DHCPREQUEST: Broadcast eines DHCP-Clients an alle DHCP-Server oder Unicast eines DHCP-Clients an einen bestimmten DHCP-Server. Mit dieser Nachricht werden die in der DHCPOFFER-Nachricht eines Servers angebotenen Parameter akzeptiert und gleichzeitig alle übrigen Angebote abgewiesen.

DHCPDECLINE: Nachricht eines DHCP-Client an einen Server, daß Konfigurationsparameter (z.B. die Netzwerk-Adresse) nicht akzeptiert werden.

DHCPRELEASE: Nachricht eines DHCP-Clients an einen Server, daß eine vergebene Netzwerkadresse nicht mehr benötigt wird.

Nachrichten-Typen des DHCP-Servers
Der DHCP-Server verwendet folgende Nachrichten-Typen:

DHCPOFFER: Nachricht eines DHCP-Servers auf DHCP-Discover, die als Broadcast oder Unicast gesendet wird. Dem Client werden Konfigurationsparameter angeboten.
DHCPACK: Nachricht eines DHCP-Servers an einen Client, die die Konfigurationsparameter zusammen mit der Netzwerkadresse beinhaltet.

DHCPNAK: Nachricht eines DHCP-Servers an einen Client, die eine Anfrage für bestimmte Konfigurationsparameter ablehnt.

Der Aufbau einer DHCP-Frame-Format ist dabei folgender (Nach [Hei96], [Ale93]):

Der Aufbau einer DHCP-Nachricht ist mit dem Aufbau einer BOOTP-Nachricht fast identisch. Lediglich das Flag- und das Options-Feld werden unter DHCP etwas anders verwendet. Das Flag-Feld wird von BOOTP nicht benutzt, und das Optionsfeld heißt unter BOOTP Vendor Field und enthält hersteller (vendor-)spezifische Einträge.

0	8	16	24 31
OP	HTYPE	HLEN	HOPS
TRANSACTION ID			
SECONDS		FLAGS	
CLIENT IP ADDRESS			
YOUR IP ADDRESS			
SERVER IP ADDRESS			
GATEWAY IP ADDRESS			
CLIENT HARDWARE ADDRESSE (16 OCTETS)			
SERVER HOST NAME (64 OCTETS)			
BOOT FILE NAME (128 OCTETS)			
OPTIONS (MAX. 312 OCTETS			

Abbildung 2.2 Aufbau einer DHCP-Nachricht [Hei96]

Die einzelnen Felder bedeuten dabei (Nach [Hei96], [Ale93]):

OP (Operation)
Das Operation-Feld definiert, ob es sich bei dem Paket um einen DHCPREQUEST (1) oder um eine DHCP-Antwort (2) handelt.

HTYPE (Hardware Type)
Definiert gemäß RFC 1340 (Assigned Numbers) das verwendete Übertragungsmedium, die Geschwindigkeit und die Datenstrukturen. Durch dieses Feld kann das DHCP flexibel auf unterschiedlichen Netzen eingesetzt werden:

Netztyp	Beschreibung
1	Ethernet (10 Mbit/s)
2	Experiment - Ethernet (3 Mbit/s)
3	Amateur Radio X.25
4	Proteon Token Ring
5	Chaos-Netz
6	ARCnet

Abbildung 2.3 Übertragungsmedien [Hei96]

HLEN (Hardware Length)
Definiert die Länge der Hardware-Adresse (in Bytes). Bei Ethernet beträgt der Wert immer 06.

HOPS
Der DHCP-Client muß immer das Hops-Feld auf den Wert = 0 setzen. Ein DHCP-Relay-Agent kann diesen Wert optional benutzen, wenn beim Initialisierungsprozeß auf ein solches Gerät zugegriffen wird.

Transaction ID
Das Transaction-ID-Feld enthält einen Integerwert, um den DHCP-Client das Zuordnen von Antworten und Requesten zu ermöglichen.

Seconds
Das Second-Feld gibt die Zeit in Sekunden an, die vergangen ist, seitdem der Client den Boot-Vorgang gestartet hat. Der maximale Wert für das Second-Feld beträgt 60 Sekunden.

Flags
Das höchstwertige Bit dieses Feldes zeigt an, ob ein Client während des Initialisierungsprozesses in der Lage ist, Unicast-Nachrichten zu empfangen. In einem solchen Fall verfügt der Client aus einer vorangegangenen Session noch über eine gültige IP-Adresse. Die restlichen Bits dieses Feldes sind auf 0 zu setzen und sind für späteren Gebrauch reserviert.

Client-IP-Adresse
Enthält die IP-Adresse des Client. Diese Adresse wird vom Client in einen DHCPREQUEST eingetragen, wenn vorher zugewiesene Konfigurationsparameter verifiziert werden sollen.

Your-IP-Adresse
Definiert die IP-Adresse, die einem Client von einem DHCP-Server zugewiesen wird.

Server-IP-Adresse
Definiert die IP-Adresse des Servers, der bei der nächsten Bootstrap-Prozedur benutzt werden soll. Dieser Wert wird von einem Server in einer DHCPOFFER-, DHCPACK- oder DHCPNAK-Nachricht mitgeteilt.

Gateway-Adresse
Legt die Adresse eines Relay Agents (Gateway-IP-Address) , die benutzt wird, wenn über einen Relay Agent gebootet wird. Diese Adresse wird nur von den jeweiligen Agents benutzt.

Client-Hardware-Adresse
Enthält die Hardware-Adresse des Clients .

Server Host Name
Optionaler Server-Name, der von BOOTP-Clients oder Clients ohne eigene permanenten Datenspeicher benutzt wird. Ein Client, der den Server Host Name eines Servers kennt, von dem er Informationen haben möchte, beträgt diese Information in das Server-Host-Name-Feld ein und stellt somit sicher, daß nur der spezifizierte Server auf den DHCP-Request antwortet. Enthält dieses Feld den Wert Null, so kann jeder DHCP-Server im Netz antworten.

Boot File Name
Der im Boot-File-Name-Feld definierte alphanumerische String ermöglicht einem DHCP-Requester, einer bestimmten Datei anzugeben, die er vom Boot Server laden möchte. Der BOOTP-Server ist dadurch in der Lage, anhand einer Datenbank die richtige Datei zu identifizieren und beispielsweise per TFTP dem Client zu übermitteln.

Optionen

Das Optionsfeld ist in seiner Verwendung sehr vielfältig und wird nachfolgend noch etwas ausführlicher behandelt. Die DHCP-Optionen haben das gleiche Format wie Vendor-Extensions des Bootstrap-Protokolls, die in RFC 1533 aufgelistet sind. Alle BOOTP Vendor Extensions sind gleichzeitig DHCP-Optionen. Wenn die Optionen im Zusammenhang mit dem Bootstrap-Protokoll verwendet werden, weist das System den ersten vier Oktetten der Wert des sogenannten Magic Cookie (RFC 951) zu, der den Modus festlegt, wie die nachfolgenden Daten interpretiert werden müssen. Der Wert des Magic Cookie ist 99.130.83.99 (dezimal). Die Optionen können feste oder variable Länge haben. Der Code besteht aus einem Oktett und gibt an, um welche Optionen es sich handelt. Die Optionen mit den Nummern 0 bis 49 können sowohl für BOOTP als auch für DHCP verwendet werden. Die Optionen mit den Nummern 50 bis 61 finden ausschließlich in DHCP ihre Verwendung. Das Längenfeld gibt an, wieviele Oktetten noch folgen. Im Wert-Feld folgen schließlich die zu übertragenden Optionen. Folgende DHCP-Optionen wurden in RFC 1533 bisher festgelegt (Die DHCP-Optionen, die in der Implementierung verwendet wurden, sind etwas dunkler markiert):

Code	**Länge**	**Bezeichnung**
0	1	Pad Option
1	4	Subnet Mask
2	4	Time Offset
3	n	Router Option
4	n	Time Server Option
5	n	Name Server Option
6	n	Domain Name Server Option
7	n	Log Server Option
8	n	Cookie Server Option
9	n	LPR Server Option
10	n	Impress Server Option
11	n	Resource Location Server Option
12	n	Host Name Option
13	2	Boot File Size Option
14	n	Merit Dump File
15	n	Domain Name
16	n	Swap Server
17	n	Root Path
18	n	Extensions Path
19	1	IP Forwarding Enable/Disable Option
20	1	Non Local Source Rou ting Enable/Disable Option
21	n	Policy Filter Option
22	2	Maximum Datagram Reassembly Size
23	1	Default IP Time-to-live
24	4	Path MTU Aging Timeout Optionen
25	n	Path MTU Plateau Table Option
26	2	Interface MTU Option
27	1	All Subnets are Local Option
28	4	Broadcast Address Option
29	1	Perform Mask Discovery Option
30	1	Mask Supplier Option
31	1	Perform Router Discovery Option

32	4	Router Solicatation Address Option
33	n	Static Route Option
34	1	Trailer Encapsulation Option
35	4	ARP Cache Timeout Option
36	1	Ethernet Encapsulation Option
37	1	TCP Default TTL Option
38	4	TCP Keepalive Interval option
39	1	TCP Keepalive Garbage Option
40	n	Network Information Service Domain Option
41	n	Network Information Servers Option
42	n	Network Time Protocol Servers Option
43	n	Vendor Specific Information
44	n	NetBIOS over TCP/IP Name Server Option
45	n	NetBIOS over TCP/IP Datagram Distribution Server Option
46	1	NetBIOS over TCP/IP Node Type Option
47	n	NetBIOS over TCP/IP Scope option
48	n	X Window System Font Server Option
49	n	X Window System Display Manager Option
50	4	Requested IP Address
51	4	IP Adress Lease Time
52	1	Option Overload
53	1	DHCP Message Type
54	4	Server Identifier
55	n	Parameter Request List
56	n	Message
57	2	Maximum DHCP Message Size
58	4	Renewal (T1) Time Value
59	4	Rebinding (T2) Time Value
60	n	Class-identifier
61	n	Client-identifier
255	1	End Option

Abbildung 2.4 Parameter der Option-Feld [Hei96]

DHCP ist ein Protokoll, mit dessen Hilfe ein Administrator viel Zeit und Mühe einsparen kann. Die sorgfältige Analyse der vorhandenen Netzstruktur und die Planung der Server-Konfiguration erfordern zwar einen gewissen Zeitaufwand, der jedoch durch die eingesparte Zeit im praktischen Betrieb sehr schnell wettgemacht wird. Die zentrale Verwaltbarkeit der Konfigurationsparameter und deren automatische Zuweisung stellen einen Vorteil dar, der nicht zu unterschätzen ist [Hei96].

Das Verhalten des DHCP-Clients läßt sich aus dem in Abbildung 2.5 darstellen.
Die Beschreibung dieses Ablaufdiagramms wird in dem folgenden Abschnitt „Funktion eines DHCP-Client“ erläutert.

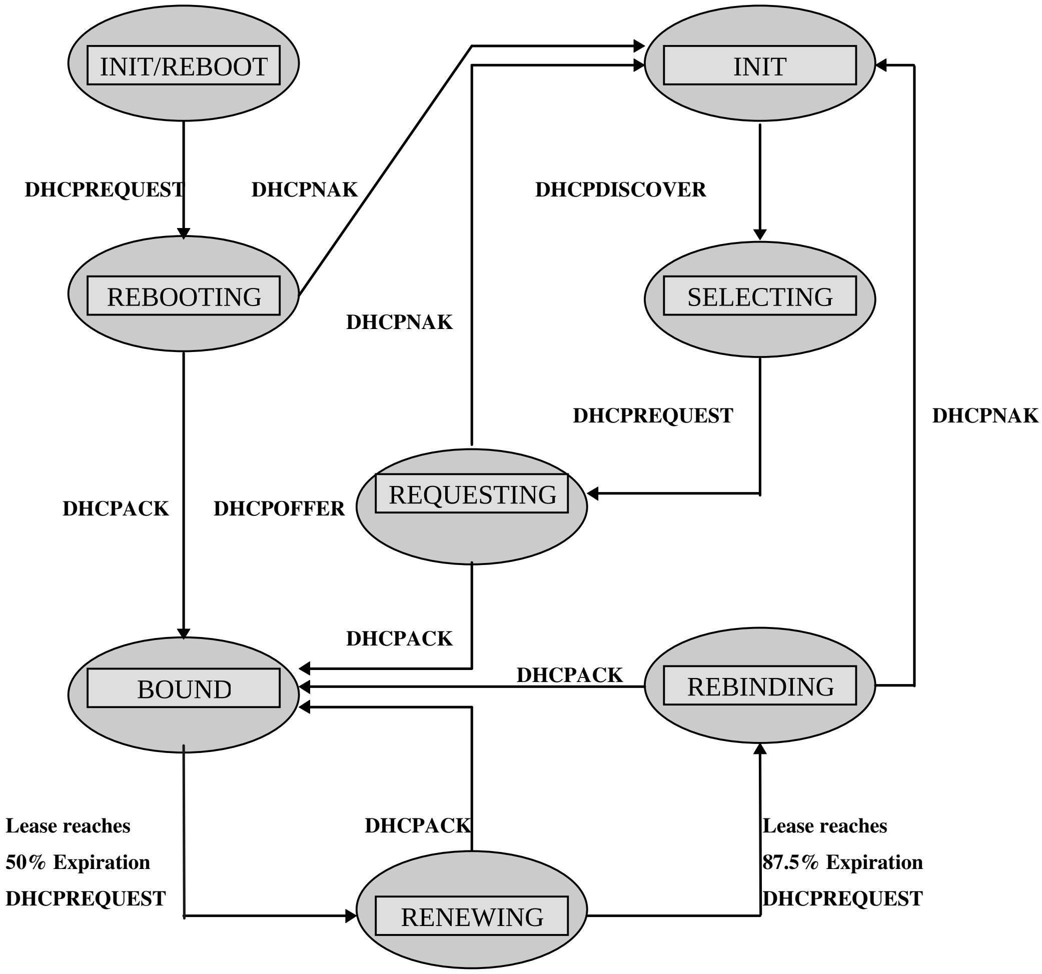


Abbildung 2.5: Ablaufdiagramm zwischen DHCP-Client und Server [Hei96]

Funktion eines DHCP-Client, Beschreibung der Abbildung 2.5 (Nach [Hei96], [Ale93]):
Mit den obengenannten Nachrichten wird der gesamte Informationsaustausch zwischen DHCP-Clients und DHCP-Server gesteuert. Nachfolgend wird der Initialisierungsprozeß aus der Sicht des Clients für eine dynamische Zuweisung beschrieben. Der Client befindet sich beim Hochfahren zunächst im Zustand INIT. Er sendet dann die Nachricht DHCPDISCOVER als Broadcast auf sein lokales Subnetz, um von einem DHCP-Server die für ihn benötigten Konfigurationsparamerter zu erhalten. Der Client darf zu diesem Zeitpunkt einen Vorschlag für die notwendigen Parameter machen. Dies kann der Fall sein, wenn der Client schon einmal mit den entsprechenden Werten versorgt war und er diese beispielsweise in einer Datei hinterlegt hatte. Ein Wert, der auf jeden Fall mit übergeben werden muß, ist die Hardware-Adresse des

Clients, da nicht erwartet werden kann, daß dieser über eine IP-Adresse für Nachrichten eines DHCP-Servers erreichbar ist. Die DHCPDISCOVER-Nachricht wird über eventuell vorhandene Relay-Agents auch an weitere Subnetze weitergeleitet, da nicht unbedingt jedes Subnetz über einen DHCP-Server verfügt. Der Client geht anschließend in den Zustand SELECTING über. In diesem Zustand wartet er auf Konfigurationsangebote in Form von DHCPOFFER-Nachrichten von Seiten der DHCP-Server, die auf seinen Request hin antwortet.
Jeder DHCP-Server kann auf ein DHCPDISCOVER mit einer DHCPOFFER-Message antworten. Ein Server wird versuchen, einem DHCP-Client direkt, also mit einem Unicast, auf eine Anfrage zu antworten. Dies ist jedoch nur möglich, wenn der Client schon über eine IP-Adresse, beispielsweise aus einer vorangegangenen Sitzung, verfügt, deren noch nicht abgelaufen ist. Ansonsten sendet ein DHCP-Server seine Nachricht an eine Broadcast-Adresse (normalerweise 255.255.255.255). In dieser Nachricht ist dann auch die Hardware-Adresse des Clients mitenthalten, die dieser mit DHCPDISCOVER übergeben hatte.
Sobald sich der Client für das Konfigurationsangebot eines Servers entschieden hat. geht er vom Zustand SELECTING in den Zustand REQUESTING über. Dabei versendet der Client die Nachricht DHCPREQUEST. Diese Mitteilung enthält einen Server Identifier, der die Bezeichnung desjenigen DHCP-Servers enthält, dessen Konfigurationsparameter sich der Client ausgewählt hat.
Eventuell weitere eintreffende DHCPOFFER-Nachrichten werden im Zustand REQUESTING vom Client ignoriert und stillschweigend verworfen. Der Server, der vom Client ausgewählt wurde, antwortet mit der Nachricht DHCPACK, die alle Konfigurationsparameter für den Client enthält. Mit dem Empfang von DHCPACK wechselt der Client in den Zustand BOUND. In diesem Zustand werden alle eintreffenden Nachrichten DHCPOFFER, DHCPACK und DHCPNAK vom Client ignoriert verworfen. Der Client ist von diesem Zeitpunkt an in der Lage, TCP/IP-Pakete von anderen Rechnern zu empfangen und auch selbst Pakete zu versenden. Auf der Client Seite werden zwei Timer T1 und T2 verwaltet, die für den weiteren Ablauf maßgeblich sind. Die Werte für T1 und T2 werden vom Server vorgegeben und haben folgende Werte: T1 = 0,5 * Lease-Dauer, T2 = 0,875 * Lease-Dauer.
Wenn die Zeit T1 abgelaufen ist, geht der Client in den Zustand RENEWING über. Er sendet eine DHCPREQUEST-Nachricht direkt an den Server, von dem er seine aktuelle Konfiguration erhalten hat. Mit dem Senden dieser Nachricht verfolgt der Client die Absicht, seine ihm zugeteilte Lease-Dauer zu verlängern. Empfängt der Client innerhalb einer Zeitdauer, die kleiner als T2 ist, ein DHCPACK mit einer neuen Lease-Dauer vom Server, so geht er wieder in den Zustand BOUND über.
Sollte innerhalb der Zeit T2 keine Nachricht vom Server erfolgen, so geht der Client in den Zustand REBINDING und schickt einen DHCPREQUEST an alle DHCP-Server (Broadcast), um seine Lease-Dauer zu verlängern.
Auch hier gilt wieder: Erhält der Client eine DHCPACK, so ist die Zeitspanne des Lease erneuert und der Client begibt sich in den Zustand BOUND. Wenn der Client im Zustand RENEWING oder REBINDING eine DHCPNACK-Nachricht von einem Server erhält oder die Lease-Dauer abläuft, so begibt er sich in den Zustand INIT. Er muß dann alle Netzwerkaktivitäten einstellen und die Initialisierungsprozedur erneut beginnen.

Kapitel 3. Szenarien für den Einsatz von DHCP

3.1 Einleitung

Nachdem das DHCP-Protokoll im zweiten Kapitel vorgestellt wurdet, kann man nun genauer die Szenarien darstellen, unter denen DHCP eingesetzt wird .
Auf der Hand liegt der Einsatz von DHCP mit mobilen Systemen mit oder ohne Mobile IP.

3.2 Konfiguration mobiler Systeme auf dem TCP / IP-Netz (Nach [Rie96])

Tragbare Computersysteme liefern bereits heute oft die effizienteste Möglichkeit, eine Arbeitsumgebung an einen anderen Ort zu transferieren. Typische Anwendungsfälle sind z.B. die folgenden:

- Mitarbeiter eines Unternehmens wechseln bei der Bearbeitung bestimmter Projekte übergangsweise in andere Abteilungen bzw. an andere Standorte

- Besucher und Gäste eines Unternehmens möchten z.B. während Arbeitstreffen der Konferenzen Teile der lokalen Infrastruktur (Drucker, E-Mail, Internet-Zugang etc.) mitbenutzen.

Das Dynamic Host Configuration Protocol ist in besonderer Weise für die Konfiguration mobiler Systeme geeignet. Unter mobilen Systemen versteht man z.B. Laptops mit Netzzugang, Computer, die drahtlos mit dem Internet verbunden sind. etc. .
Um ein mobiles System in einem Subnetz erfolgreich benutzen zu können, müssen dem mobilen System einige Parameter zugeteilt werden. Dies sind zum Beispiel die IP-Adresse, die die Identifikation des Systems unter TCP/IP ermöglicht, Domain Name Server, welche es mit DNS-Namen versorgen, etc. .
Voraussetzung für den Einsatz von mobilen Systemen unter DHCP ist die Installation eines DHCP-Client-Programms. Dieses nimmt dann Kontakt mit einem DHCP-Server auf und läßt sich die Daten für die Netzkonfiguration geben. Zu diesen Daten gehört die IP-Adresse des Host-Systems, der nächste Router etc. .
Es wird vorausgesetzt, daß das sogenannte dynamische Adreßvergabeverfahren eingesetzt wird, da die anderen Vergabeverfahren permanente IP-Adressen vergeben bzw. den Einsatz eines Netzadminstrators voraussetzen. Sollte ein Host sich dem Ende der Leasezeit nähern und seinen Lease nicht verlängern können, muß er sich um einen neuen Lease bemühen. Wenn ein Host vor dem Ende der Leasezeit sich wieder aus dem Netz verabschieden will, sendet der DHCP-Client ein DHCPRELEASE an den Server, um die IP-Adresse für einen neuen Benutzer freizugeben. Im Falle eines Absturzes des Host-Systems bleibt die zugeteilte IP-Adresse auf jeden Fall die gesamte Leasedauer dem Host erhalten. Um dieser Situation zu begegnen, sollte der Host sich die Konfiguration „merken“. wie dies passiert, hängt von der Implementierung der Client-Software ab.

3.2.1 DHCP und Mobile-IP

Im Rahmen des Einsatzes mobiler Systeme mit DHCP muß man auch den Einsatz von Mobile-IP berücksichtigen. Dabei wird jedem mobilen Knoten in einem Netz (z.B. ein Laptop mit Netzanschluß) eine sogenannte Home Address zugeordnet, unter der der Knoten im Netz erreichbar ist. Wenn sich der mobile Knoten aber in einem anderen Netz als seinem Heimatnetz befinden sollte, müssen die für ihn bestimmten Datagramme so geroutet werden, daß sie auch beim Host (unter seiner Home Address) ankommen. Zu diesem Zweck gibt es unter Mobile-IP die sogenannte *Care-Of-Address,* an die während der Abwesenheit des Knotens die Datagramme geschickt werden, die eigentlich für die Home Address gedacht waren. Der sogenannte Home Agent kümmert sich um diese Versendung von Datagrammen [Rie96].

3.3 Konfiguration auf dem TCP/IP-Netz fest installierter Systeme

Auch bei fest installierten Systemen wie UNIX Workstations oder Windows NT-Rechnern wird DHCP zunehmend eingesetzt, um IP-Systeme über das Netz zu konfigurieren. Bei Windows 3.11/95/NT ist DHCP-Client sogar im Lieferumfang enthalten. In der Praxis wird DHCP vorwiegend in diesem Bereich eingesetzt. Die meisten Serversoftwarevarianten unterstützen standardmäßig nur die manuelle oder automatische Vergabe von IP-Adressen, bei denen Adressen permanent vergeben werden. Das heißt, der Einsatz von DHCP zur Konfiguration mobiler Systeme ist noch nicht so verbreitet wie der Einsatz bei fest installierten Systemen.

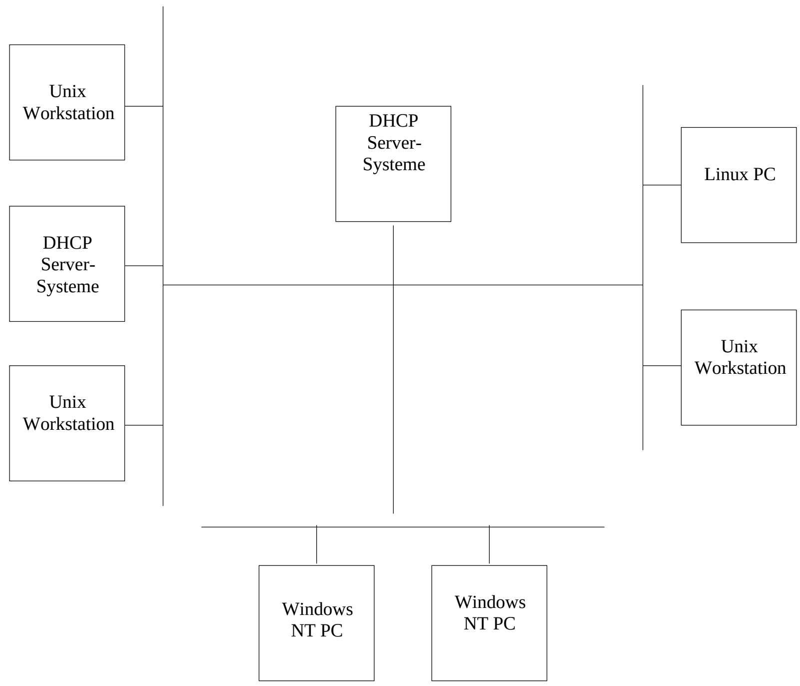


Abbildung 3.1: **DHCP** im Einsatz mit fest installierten Systemen

Damit sind die fest installierten Systeme für einen DHCP-“Provider“ leichter zu überwachen bzw. zu managen, da DHCP-Server und Clients stationär sind und keine dynamische Adressen- bzw. Konfigurationsverwaltung nötig ist. Da die Verwendung von DHCP mit stationären Systemen typisch ist, werden an dieser Stelle die Situationen untersucht, die sich aus seinem Einsatz ergeben [Rie96].

3.4 Szenarien unter DHCP

Im folgenden sollen Situationen dargestellt werden, die sich im Einsatz von DHCP zur Konfiguration von Computersystemen ergeben.

Folgende Szenarien können nach [Rie96] unterschieden werden:

3.4.1 Client meldet sich neu an

Wenn ein Hostsystem in eine Netzumgebung integriert werden will, sei es ein mobiles System oder ein stationäres, welches aufgrund einer Policy-Entscheidung der Netzbetreiber nur Zeitweise die Netzressourcen benutzen können soll, spielen sich folgende Abläufe ab:

Der Client benötigt eine IP-Adresse und Konfigurationsdaten.

Da er keine Serveradresse kennt, sendet er ein DHCPDISCOVER als Broadcast auf das Subnetz [Dro97]. Broadcast bedeutet, daß die Meldung an die Adresse 255.255.255.255 gesendet wird, was bewirkt, daß sie an alle Systeme, die in diesem Subnetz sind, weiter geleitet wird. Den *Relay Agents* kommt dabei die Aufgabe zu, als eine Art Gateway das DHCP-DISCOVER auch auf andere Subnetze weiterzuleiten [Dro97].

Dieses DHCPDISCOVER *kann bereits Wunschparameter des DHCP-Client enthalten, z.B. die gewünschte IP-Adresse oder die gewünschte Lease-Zeit.*

Die DHCP-Server, die das DHCPDISCOVER mitbekommen, überprüfen, ob sie freie IP-Adressen in ihrem Adreßpool haben und stellen eine DHCPOFFER zusammen, in der sie dem Client auch schon bestimmte Konfigurationsparameter anbieten können.
Falls kein DHCP-Server antwortet, weil z.B. keiner aktiv ist oder gerade kein DHCP-Server freie Adressen hat, wiederholt der Client nach einem definierten Wartealgorithmus sein DHCPDISCOVER [Dro97].

Empfängt der DHCP-Client innerhalb einer definierten Zeit mehrere DHCPOFFER’s, so sammelt er die Angebote und wählt eines davon aus. Mit diesen Daten stellt er eine DHCP-REQUEST zusammen und schickt sie als Unicast an den Server, von dem dieses Angebots kam.

Nun gibt es mehrere Möglichkeiten des weiteren Geschehens:

- Server empfängt kein DHCPREQUEST , z.B. wegen Netzproblem.
  Wenn der Server kein DHCPREQUEST empfängt, startet er auch keine Aktion.

- Server empfängt und entnimmt der Meldung, daß sie nicht für ihn bestimmt ist.
  In diesem Fall verwirft der Server das DHCPREQUEST.

- Server empfängt DHCPREQUEST und entnimmt der Meldung, daß sie für ihn bestimmt.
  Jetzt gibt es wieder mehrere Möglichkeiten:

  - Server hat weiterhin freie IP-Adresse:

Er erstellt ein DHCPACK, merkt sich die Konfiguration in einer internen Tabelle und sendet das DHCPACK an den Client. Falls dieser feststellt, daß die IP-Adresse schon benutzt wird, beendet er die Konfiguration mit einem DHCPDECLINE und beginnt von neuem mit einem DHCPDISCOVER.

- Server hat keine freie IP-Adresse:
  Er schickt eine DHCPNAK an den Client. Dies bedeutet, daß der Client sich momentan nicht von diesem Server konfigurieren lassen kann.

3.4.2 Client will Lease verlängern (Nach [Rie96])

Ein Client ist von einem DHCP-Server konfiguriert worden und versucht nun, seine Leasezeit verlängern zu lassen.

Der Client sendet als erstes ein DHCPREQUEST an den Server, von dem er konfiguriert wurde.
Dieses enthält in der Requested IP Adresse Option die gewünschte Adresse.
Die Server, die die Konfiguration des Clients kennen, entscheiden nun:

- Der Request / die Nachfrage kann erfüllt werden:
  Der Server sendet ein DHCPACK an den Client

- Dem Request kann nicht entsprochen werden, weil z.B. der Client sich in einem neuen Subnetz befindet:
  Der Server sendet eine DHCPNAK an den Client.

Mögliche Aktionen des Clients:

- Der Client wartet vergeblich auf DHCPACK:
  Das führt nach einer definierten Zeit zu einem Timeout;
  Der Client versucht daraufhin, nach dem definierten Wiederholalgorithmus, welcher nach der sogenannten Randomized Exponential verfährt, sein DHCPREQUEST zu wiederholen [Dro97].
  Er kann aber seine aktuelle Adresse bis zum Ende der Leasezeit weiterbenutzen.

- Der Client hat DHCPNAK empfangen:
  Der Client muß nun den Konfigurationsprozeß von neuem starten (DHCPDISCOVER etc.)

- Der Client hat DHCPACK empfangen:
  Er testet die zugeteilte IP-Adresse, was folgende Möglichkeiten eröffnet:

  - Die Adresse wird nicht von einem anderen System benutzt:
    Der Client ist fertig konfiguriert.
  - Die Adresse wird von einem anderen System benutzt:
    Der Client muß ein DHCPDECLINE senden und die Konfiguration von neuem starten.

3.4.3 Client will Lease beenden (Nach [Rie96])

Falls der Client aus irgendwelchen Gründen seine IP-Adresse nicht mehr benötigt, so kann er seine Adresse freigeben. Dies geschieht, indem der Client ein DHCPRELEASE an den Server sendet, von dem er konfiguriert wurde. Der oder die Server, die seine Konfiguration kennen, löschen daraufhin seine Konfigurationsinformation und kennzeichnen intern seine IP-Adresse als verfügbar.

3.4.4 Kritische Situationen (Nach [Rie96])

Beim Einsatz von DHCP kann man sich verschiedene Kritische Situationen vorstellen, die evtl. behandelt werden müssen:

- Mehrere Server auf dem Netz vergeben die gleiche IP-Adresse an unterschiedliche Clients: Diese Situation ist in der Realität nicht denkbar, da es unter dem DHCP-Protokoll nicht möglich ist, mehrere DHCP-Server mit sich überschneidenden Adreßbereichen auszustatten, ohne von vornherein Konsistenzprobleme in Kauf zu nehmen. Dies Möglichkeit wird von den Netzbetreibern derzeit beim Gebrauch der derzeitigen Implementierungen von DHCP-Servern meistens vermieden.
- DHCP -Server vergibt fehlerhafte Konfigurationsinformationen:
  Dieser Fall könnte z.B. dann auftreten, wenn sich die Netztopologie verändert (Netzkomponenten wie Drucker etc. bekommen eine andere IP-Adresse, oder werden ganz aus dem Netz entfernt), aber die Netzbetreiber es versäumen, die den DHCP-Servern zur Verfügung stehenden Konfigurationsdaten wie Routeradressen etc. zu aktualisieren. Sollte dieser Situation eintreten, so liegt es in der Verantwortung der Betreiber der DHCP-Serversoftware, die Konsistenz der Daten zu gewährleisten.

- DHCP-Client sendet DHCPREQUEST ohne vorheriges DHCPDISCOVER:
  Diese Situation ist nicht realisierbar, da bei einem DHCPREQUEST die Serverhardwareadresse und eine Transaction ID benötigt wird, um es zu verschicken. Diese kann der Client aber nur kennen, wenn er mindestens einmal ein DHCPDISCOVER gesendet hat.

- DHCP lehnt Lease mit DHCPDECLINE / DHCPRELEASE ab,
  Diese Situation könnte z.B. dann eintreten, wenn ein Benutzer sich einen unendlichen Lease erschleichen möchte, weil er weiß, daß der DHCP-Server keine benutzten IP-Adressen vergibt (dies könnte in Abhängigkeit von der Implementierung der Serversoftware vorkommen). Sollte ein derartiges Verhalten von Netzbenutzern auftreten, so liegt es in der Verantwortung des Netzbetreibers, durch geeignete Maßnahmen wie das Kontrollieren von Logdateien der Server dieses Verhalten unmöglich zu machen.

- DHCP-Client erhält ein DHCPNAK, verhält sich aber als fertig konfiguriert:
  Diese Situation ist nicht real, da DHCP-Server DHCPNAK’s nur versenden, falls eine Adresse, die sie vergeben wollen, schon benutzt wird.

Kapitel 4.

Entwicklung eines Testprogramms (DHCP-Client) für das DHCP-Server

4.1 Einleitung

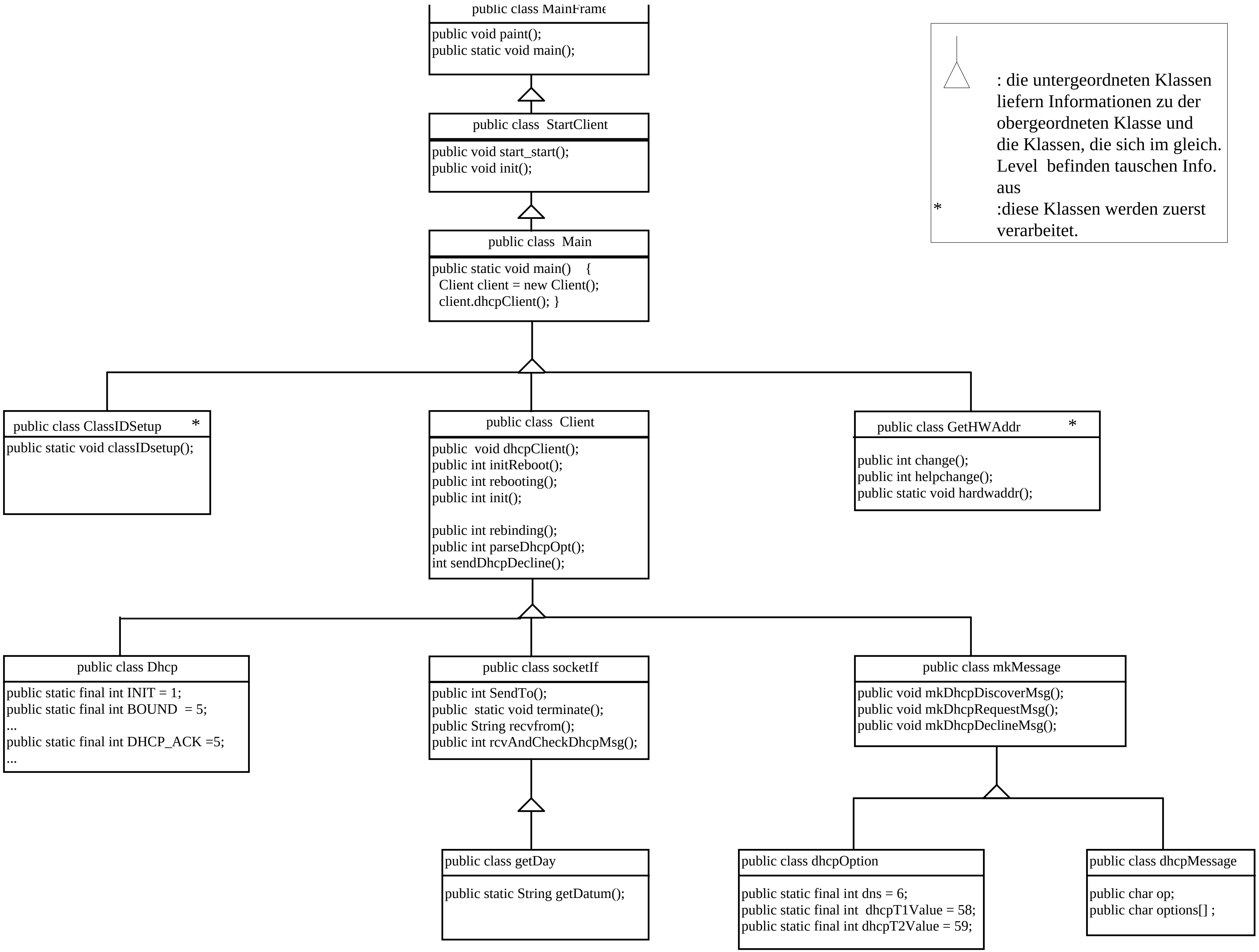
Nachdem im Kapitel 1 die Anforderungen festgelegt wurden, wie der DHCP-Client aussehen kann, muß nun eine Festlegung getroffen werden, wie die Klassenstruktur aufgebaut werden kann, um diese Anforderungen zu realisieren. Die Realisierungs-programmiersprache ist JAVA.

4.2 Warum Java?

Java ist eine **objektorientierte Programmiersprache**, die im Frühjahr 1995 von Sun Microsystems eingeführt wurde. Java bietet die Möglichkeit, plattformunabhängige Anwendungen zu entwickeln. Eine umfangreiche Programmierschnittstelle macht sie insbesondere für Multimediale und netzwerkorientierte Anwendungen geeignet.
Die Sprache Java sich syntaktisch an Sprachen wie C und C++ orientiert - was einen einfachen Einstieg für C/C++ Programmierer sicherstellt -.
Java hat das Potential, sowohl die Gestalt des Internet - speziell des World Wide Web - als auch die Art der Softwareentwicklung für das Intranet und Internet gravierend verändert.
In Bezug auf die Gestalt des Internet hat sich durch Java der Trend etabliert, multimediale ausführbare Daten in World Wide Web-Seiten zu integrieren. Seitdem dürfen sich Seiten zurecht mit Attributen wie „lebendig“, „interaktiv“ oder „dynamisch“ schmücken.
Auch in Bezug auf die **Softwareentwicklung** für das Intranet und das Internet setzt Java Maßstäbe. Durch Techniken wie die virtuelle Java Maschine und Byte Code Interpretation sind Java-Programme prinzipiell plattformunabhängig, selbst in traditionell problematischen Bereichen wie grafischen Benutzeroberflächen. Der Entwickler kann also mit seinen Anwendungen ohne Mehraufwand eine vielzahl von Hard- und Software-Plattformen bedienen. Hierzu zählen bereits Solaris, Windows 95, Windows NT und Apple Macintosh.
Die Entwicklung von Techniken zur Anbindung von Java-Programmen an Datenbanken (JDBC) und Objektnetzwerke in heterogenen Umgebung (CORBA), die Integrierbarkeit nativer Funktionen für maximale Performanz und die Verfügbarkeit professioneller Entwicklungsumgebungen bestätigen den Charakter von Java und rechtfertigen die rapide wachsende Verbreitung und Beliebtheit dieser Sprache [BuKr96].

4.3 Entwurf des Klassenmodells

Im folgenden wird ein Klassenmodell angegeben, wie die Informationen zu verarbeiten sind. Zu diesen Informationen gehören auch die Nachrichten, welche zwischen dem DHCP-Client und -Server ausgetauscht werden. In den folgenden Abschnitten wird dieses Klassenmodell ganz genau betrachtet.
Dieses Klassenmodell wird im folgenden graphisch vorgestellt:


public class MainFrame
public void paint();
public static void main();
public class StartClient
public void start_start();
public void init();
public class Main
public static void main() {
Client client = new Client();
client.dhcpClient(); }
public class ClassIDSetup *
public static void classIDsetup();
public class Client
public void dhcpClient();
public int initReboot();
public int rebooting();
public int init();
public int rebinding();
public int parseDhcpOpt();
int sendDhcpDecline();
public class GetHWAddr *
public int change();
public int helpchange();
public static void hardwaddr();
public class Dhcp
public static final int INIT = 1;
public static final int BOUND = 5;
...
public static final int DHCP_ACK =5;
...
public class socketIf
public int SendTo();
public static void terminate();
public String recvfrom();
public int rcvAndCheckDhcpMsg();
public class mkMessage
public void mkDhcpDiscoverMsg();
public void mkDhcpRequestMsg();
public void mkDhcpDeclineMsg();
public class getDay
public static String getDatum();
public class dhcpOption
public static final int dns = 6;
public static final int dhcpT1Value = 58;
public static final int dhcpT2Value = 59;
public class dhcpMessage
public char op;
public char options[] ;
: die untergeordneten Klassen liefern Informationen zu der obergeordneten Klasse und die Klassen, die sich im gleich. Level befinden tauschen Info. aus
* :diese Klassen werden zuerst verarbeitet.

4.3.1 Beschreibung des Klassenmodells

Die Beschreibung des Klassenmodells wird von unten nach oben (Bottom-Up) betrachtet.

4.3.1.1 Aufbau des DHCP-Messages

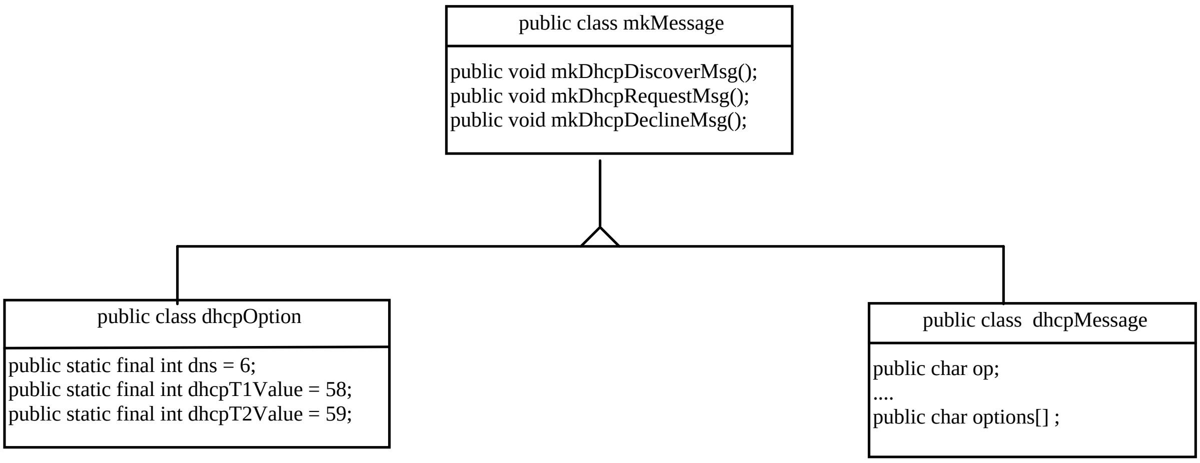
Das folgende Teil des Klassenmodells baut insgesamt 4 Nachrichten-Typen , die vom DHCP-Client an den -Server gesendet werden, auf.
Die Nachrichten-Typen sind:

DHCPDISCOVER**:** Nachricht, die ein Client als Broadcast verschickt, um das Netz auf verfügbare DHCP-Server zu testen.

DHCPREQUEST: Broadcast eines DHCP-Clients an alle DHCP-Server. Mit dieser Nachricht werden die in der DHCPOFFER -Nachricht eines Servers angebotenen Parameter akzeptiert und gleichzeitig alle übrigen Angebote abgewiesen.

DHCPDECLINE**:** Nachricht eines DHCP-Client an einen Server, daß Konfigurationsparameter wie die Netzwerk-Adresse ungültig sind.

DHCPRELEASE: Nachricht eines DHCP-Clients an einen Server, daß eine vergebene Netzwerkadresse nicht mehr benötigt wird und wieder zur Verfügung stellt.



Das DHCP-Frame-Format ist in 2.5.3 beschrieben, wie man sieht, das DHCP-Frame kann als „struct“ realisiert werden. Für den Nachrichtenaustausch ist das Optionsfeld in diesem Frame sehr entscheidend: Wenn die Optionen im Zusammenhang mit dem Bootstrap-Protokoll verwendet werden, weist das System den ersten vier Oktetten der Wert des sogenannten „Magic Cookie“ (RFC 951) zu, der den Modus festlegt, wie die nachfolgenden Daten interpretiert werden müssen. Der Wert des „Magic Cookie“ ist 99.130.83.99 (dezimal).
Die Optionen können feste oder variable Länge haben. Der Code besteht aus einem Oktett und gibt an, um welche Option es sich handelt.

Wie ist das DHCP-Frame-Format realisiert ?

Man definiert drei Klassen:

public class dhcpMessage: Diese Klasse dient als „struct“. Sie enthält alle Felder des DHCP-Protokolls, die an den Server gesendet werden.

public class dhcpOption: Diese Klasse enthält die Optionen, die im Feld „options“ verwendet werden. Die gewünschte Optionen werden dann an den Server gesendet, der wieder entsprechend antwortet.

Der Inhalt des DHCP-Frame sowie des Optionsfeldes muß nun vollständig eingetragen werden, und daraus die verschiedenen Nachrichten-Typen erzeugen. Dazu dient die folgende Klasse:

public class mkMessage: In Bezug auf „dhcpMessage“ und „dhcpOption“ ist diese Klasse für die Erzeugung einer der obengenannten Nachrichtentypen zuständig.

4.3.1.2 Ausgabe das Datum und die Uhrzeit

public class getDate: Diese Klasse repräsentiert Daten und Zeiten. Sie läßt Sie systemunabhängig mit ihnen arbeiten. Diese Daten und Zeiten werden in der Klasse „socketIf“ benötigt, um die Sende- und die Empfang-Zeit der Nachrichten auszugeben.

public class getDay
public static String getDatum();

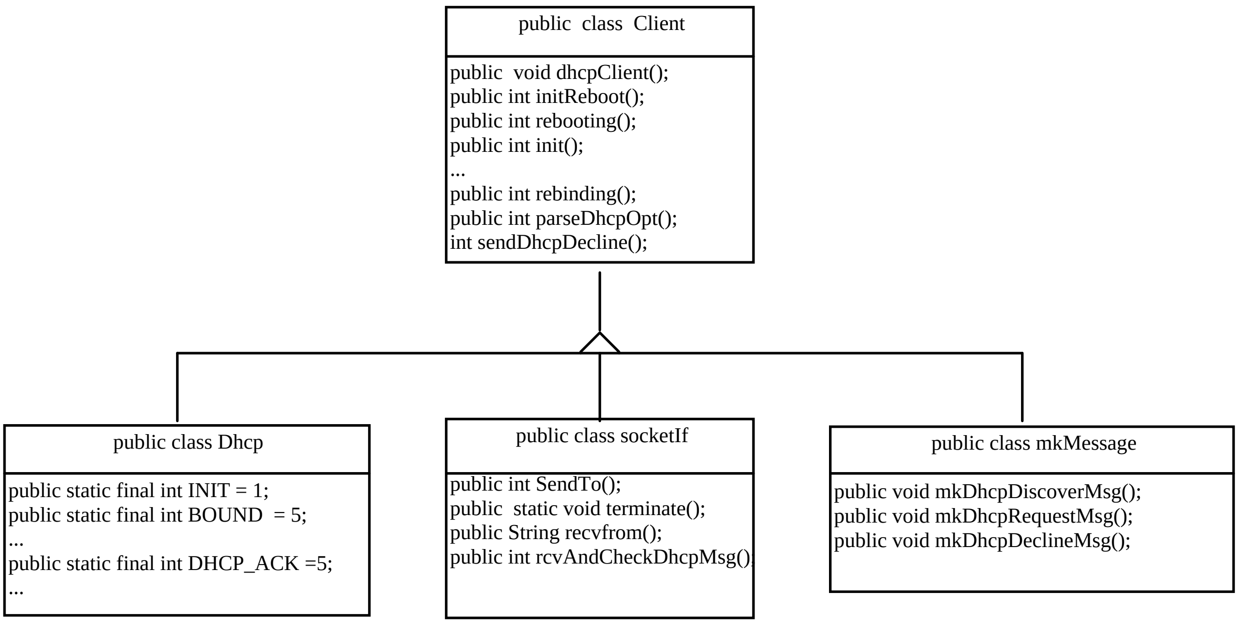
Wie wird die Datum-Ausgabe in Java 1.1 ermittelt?

In dem Package „java.util“ wurde in Java 1.0 die Klasse „Date“ für die Ausgabe des Datums und Uhrzeit manipuliert. In Java 1.0 können Sie ein Date erzeugen, indem Sie in Millisekunden seit der „Epoche“ (Mitternacht GMT, 1. Januar 1970) angeben oder indem Sie Jahr, Monat, Tag und optional die Stunde, Minute und Sekunde angeben. Jahre können als Zahl in Jahren seit 1900 angegeben werden. Wenn Sie den Date-Konstruktor ohne Argumente aufrufen, wird Date auf das aktuelle Datum und die aktuelle Zeit gesetzt. Die Instanz-Methode der Klasse ermöglicht es Ihnen, die verschiedenen Datums- und Zeit-Arrays auszulesen bzw. zu setzen und die Daten aus und in ihre Stringform zu konvertieren.
Aber in der neuen Version von Java 1.1 sind Änderungen aufgetreten, wie man das aktuelle Datum und die aktuelle Zeit auslesen. In Java 1.1 wird nicht mehr die Klasse „Date“ manipuliert, um das aktuelle Datum und die aktuelle Zeit auszulesen, sondern wird die neu definierte Klasse „Calender“ manipuliert.
Sie ist eine abstrakte Klasse. Mithilfe der Methode „getInstance()“ kann ein Object der Klasse „Calender“ instanziert werden .
Die Methode „getDatum“ in unserer Klasse getDay() ist mit Java 1.1 implementiert und sieht wie folgt aus:

```
public class getDay{
     public static String getDatum(){
          Calender calender = Calender.getInstance();
          Date date = calender.getTime();
          TimeZone tz = TimeZone.getTimeZone( “ECT“);
          DateFormat formatter = DateFormat.getDateTimeInstance(
                                          DateFormat.LONG,DateFormat.LONG,
                                          Locale.GERMAN);
          String result = formatter.format(date);
          return result;
     }
}
```

4.3.1.3 Steuerung des DHCP-Clients

Es wurden bis jetzt die Klassen, die für den Aufbau der verschiedenen DHCP-Nachrichten manipuliert, vorgestellt. In diesem Abschnitt werden wir die Klassen, die für das Senden und Empfangen der Nachrichten zwischen dem DHCP-Client und DHCP-Server manipuliert werden, betrachten. wir werden auch sehen, daß die Methode „dhcpClient“ den gesamten Ablauf des DHCP-Clients steuert.



class public mkMessage: Im Abschnitt 4.3.1.1 wurde sie genau erläutert.

class public Dhcp: Diese Klasse definiert die Werten von vielen Konstanten, die eine große Rolle beim Nachrichtenaustausch spielt, fest.
Um die DHCP-Client-Zustände zu steuern, werden die Zustände in dieser Klasse durch feste Integer-Werte definiert.
Es werden auch die Werte der Nachrichten-Typen (z.B., ob DHCPDISCOVER, DHCPACK) durch feste Integer-Werte definiert.
Die Klasse sieht wie folgt aus:

```
public class Dhcp{
      /* Value of Magic Cookie Field */
      public static final int    MAGIC_COOKIE = 0x63825363;

      /* DHCP Message Op Code */
      public static final int BOOTREQUEST   = 1;
      public static final int BOOTREPLY     = 2;

      /* DHCP Message Type */
      public static final int DHCP_DISCOVER = 1;
      public static final int DHCP_OFFER     = 2;
      public static final int DHCP_REQUEST   = 3;
      public static final int DHCP_DECLINE   = 4;
      public static final int DHCP_ACK       = 5;
      public static final int DHCP_NAK       = 6;
      public static final int DHCP_RELEASE   = 7;

      /* DHCP Client State */
      public static final int MAX_STATES     = 8;
      public static final int INIT_REBOOT    = 0;
      public static final int INIT           = 1;
      public static final int REBOOTING      = 2;
      public static final int SELECTING      = 3;
      public static final int REQUESTING     = 4;
      public static final int BOUND          = 5;
      public static final int RENEWING       = 6;
      public static final int REBINDING      = 7;
      public static final int EXCEPTION      = -1;

      /* max. Number of DHCP Options */
      public static final int MAXNOPT        = 312;

      /* Hardware Type support Ethernet only (s. Absch. 3.5.2)*/
      public static final int HTYPE_ETHER    = 1;


}
```

public class socketIf: Diese Klasse ist für das Senden, Empfangen und überprüfen der Pakete zuständig. Die Methoden dieser Klasse werden kurz beschrieben.

I. Die Methode „sendTo“:
Sie sendet ein Discover-Message auf dem angegebenen Port zu der gewünschten Adresse oder als Broadcast (Broadcast Messages werden erst ab Java 1.1 unterstützt).

```
public int SendTo(int destPort, String toAddress, dhcpMessage
                                                         msg){
}
```

II. Die Methode „SendRequest“:
Sie sendet ein Request-Message auf dem angegebenen Port zu dem DHCP-Server.

```
public int SendRequest(int destPort, String toAddress,
                                             dhcpMessage msg){
}
```

III. Die Methode „recvfrom“:
Diese Methode ist dazu zuständig, die Pakete, die vom DHCP-Server zum DHCP-Client als Antwort auf seine Anfragen nach Konfigurationsparameter, abzufangen. Hier wird die Gelegenheit genutzt, um die Erweiterung der Methode „receive“ in dem Package „java.net“ bei der neuen Version Java 1.1 gegenüber Java 1.0 zu beloben. Denn die Methode „receive“ hat bei Java 1.0 nach ihrer Aufruf das Programm blockiert, solange keine Pakete eingetroffen sind.

Und das wäre ein Hindernis gewesen, um unsere Implementierung zu realisieren, weil das Programm dadurch gelähmt wird.
Aber das Problem existiert nicht mehr bei Java 1.1, denn man kann eine Wartezeit zum Empfangen der Pakete setzen, dann kann die „recive-“Methode das Programm nicht mehr blockieren.

```
public String recvfrom (int fromport){
}
```

IV. Die Methode rcvAndCheckDhcpMsg:
In dieser Methode werden die abgefangene Pakete überprüft, ob sie DHCP-Pakete sind, wenn ja, dann werden sie weiter verarbeitet, sonst werden sie verworfen.

```
public int rcvAndCheckDhcpMsg(dhcpMessage msg, dhcpMessage
                                                     msgSend){
      recvfrom();
}
```

V. Die Methode „terminate“:
Die Methode wird manipuliert, um das Programm zu terminieren.

```
public static void terminte(){
}
```

public class Client: Wir sind im Moment in der Lage Nachrichten aufzubauen, zu senden und zu empfangen, wir müssen jetzt die Zustände des DHCP-Client steuern, wann gesendet und empfangen darf?
In dieser Klasse wird der Ablauf des DHCP-Clients gesteuert, wann in welchem Zustand gehen darf. Da werden die Zustände (init, init_reboot, selecting etc.) definiert. Da wird sogar der Option-Feld aus dem empfangenen Pakete zerlegt und die Werte der Optionen ausgelesen und an der richtigen Stelle weiter ermittelt. Die Hauptmethode des Code ist „dhcpClient“, und sie ist in dieser Methode definiert. Diese Methode wird zur besseren Verständlichkeit beschrieben:

```
public void dhcpClient(){

      /*diese Variable enthält den nächsten Zustand des Clients*/
      int next_state;

     /**
        Wähle den Start-Zustand des Clients INIT / INIT-REBOOT /
        START
      **/
      /* starte in INIT - Zustand */
      if (Client.init_reboot_button_state = 0){
      /* starte in INIT_REBOOT */
      }else if ( Client.init_reboot_button_state = 0){
      /* starte normale Vorgang */
      }else{
      }
     /**
        Solange keine Terminierung erfolgt, findet eine endlose
        Schleife statt.
      **/
           while(socketIf.Terminate_State == false){
                switch(next_state){
                     case Dhcp.INIT_REBOOT:
                          MainFrame.Tree_State = 0;
                          next_state = initReboot();
                          break;
                     case Dhcp.INIT:
```

```
                                MainFrame.Tree_State = 1;
                                next_state = init();
                                break;
                           case Dhcp.REBOOTING:
                                MainFrame.Tree_State = 2;
                                next_state = rebooting();
                                break;
                           case Dhcp.SELECTING:
                                MainFrame.Tree_State = 3;
                                next_state = selecting();
                                break;
                           case Dhcp.REQUESTING:
                                MainFrame.Tree_State = 4;
                                next_state = requesting();
                                break;
                           case Dhcp.BOUND:
                                MainFrame.Tree_State = 5;
                                next_state = bound();
                                break;
                           case Dhcp.RENEWING:
                                MainFrame.Tree_State = 6;
                                next_state = renewing();
                                break;
                           case Dhcp.REBINDING:
                                MainFrame.Tree_State = 7;
                                next_state = rebinding();
                                break;
                                Dhcp.EXCEPTION:
                                System.out.println(“Occured (1) in
                                dhcpClient()“);
                                System.out.println(“Please Try Agean“);
                                System.exit(1);
                                break;
                           default:
                                System.out.println(“Occured (2) in
                                dhcpClient()“);
                                System.exit(1);
                                break;
                      }
                 }
      }

      // Behandlung des Zustandes init-reboot
      public int initReboot(){
      }

      // Behandlung des Zustandes rebooting
      public int rebooting(){
      }

      // Behandlung des Zustandes init
      public int init(){
      }

      // Behandlung des Zustandes  selecting
      public int selecting(){
      }

      // Behandlung des Zustandes requesting
      public int requesting(){
      }

      // Behandlung des Zustandes bound
      public int bound(){
```

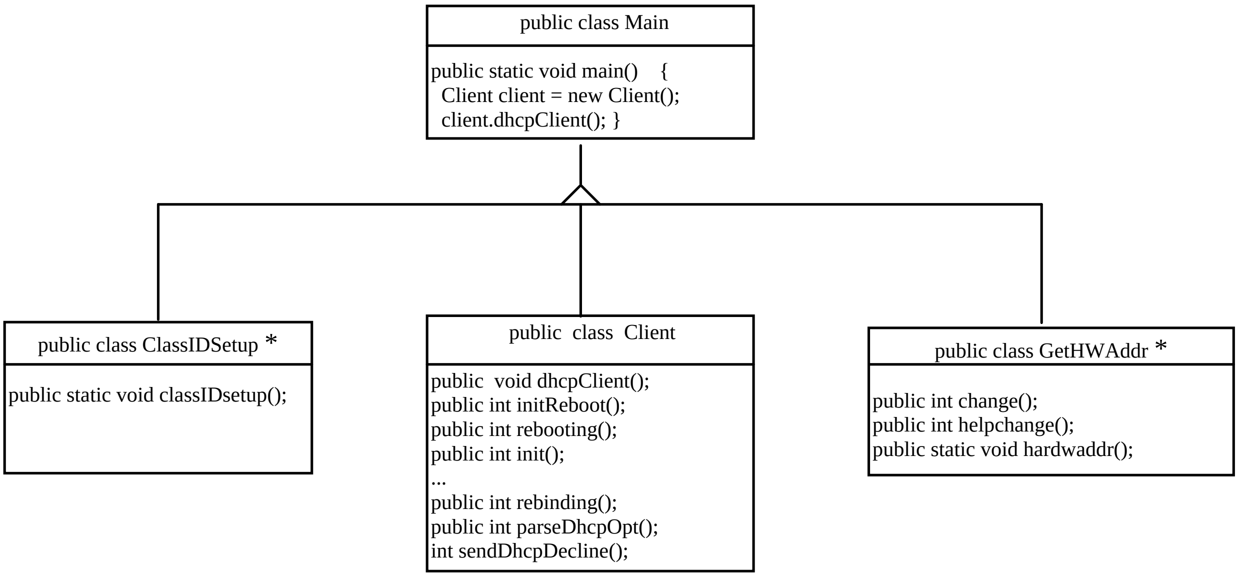
```
}

// Behandlung des Zustandes renewing
public int renewing(){
}

// Behandlung des Zustandes rebinding
public int rebinding(){
}
```

4.3.1.4 Kommunikationskomponenten des DHCP-Clients initialisieren

Das Programm kann noch nicht gestartet werden, vorher muß auch die Hardwareadresse und der class identifier verarbeitet werden. Dafür sorgen die folgende Klassen:



public class GetHWAddr: Diese Klasse verarbeitet die Hardwareadresse, die an den Server gesendet wird. Die Hardware muß die Länge 6 haben. Die Hardwareadresse wird als String gelesen, dieser String wird zerlegt und in Integer-Zahlen umgewandelt.

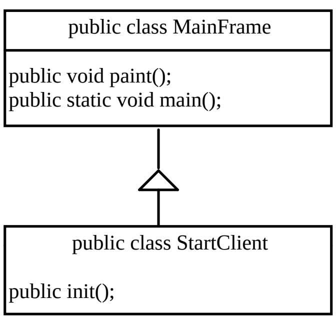
public class ClassIDSetup: Diese Klasse ist für die Aufstellung des *class identifier* zuständig. Man kann einen beliebigen *class identifier* angeben.

public class Client: Im Abschnitt 4.3.1.3 wurde sie genau erläutert.

public class Main: In dieser Klasse erfolgt die Initialisierung der Hardwareadresse, Server-IP-Adresse, Kommunikationsports, Interface Name, Class-Identifier und anschließend der Aufruf der Hauptmethode dhcpClient.

4.3.1.5 Frame mit den Komponenten der Oberfläche erzeugen

In diesem Abschnitt wird der folgende Abschnitt von unserer Klassenmodell erläutert:



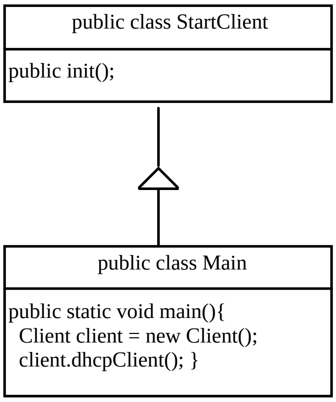
Die Anforderung auf eine Benutzeroberfläche wird durch die folgenden Klassen realisiert. Die Idee ist den Frame zu erzeugen und dann den Zustandsbaum auf diesem Frame zu bezeichnen und die anderen Komponenten einzufügen.
In unserer Implementierung wurde der GridBagLayout-Layout-Manager manipuliert, um Komponenten in unserem Frame anzuordnen. Dieser Layout-Manager ist sehr mächtig, aber auch recht verwirrend.

public class MainFrame extends Frame implements Runnable: Die Klasse Frame wurde als Unterklasse der Klasse Frame deklariert, d.h., MainFrame erbt von der Klasse Frame alle Methoden und Variablen. Die Klasse Runnable vererbt Eigenschaften an die Unterklasse MainFrame. Diese Klasse erzeugt den Frame und positioniert den Baum und fügt die anderen Komponenten in ihm hinzu.
Für die Bezeichnung und Plazierung des Zustandsbaumes wird die „paint()-“methode in der Klasse „java.awt.Componenet“ überschrieben.

public class StartClient: Die Klasse StartClient wurde als Unterklasse der Klasse Applet deklariert, d.h., StartClient erbt von der Klasse Applet alle Methoden und Variablen. Die Klasse Runnable vererbt Eigenschaften an die Unterklasse StartClient.
Diese Klasse Initialisiert die Komponenten vor dem Start und legt die verschiedenen Listener und ihrer Funktionalität fest.

4.3.1.6 DHCP-Client starten

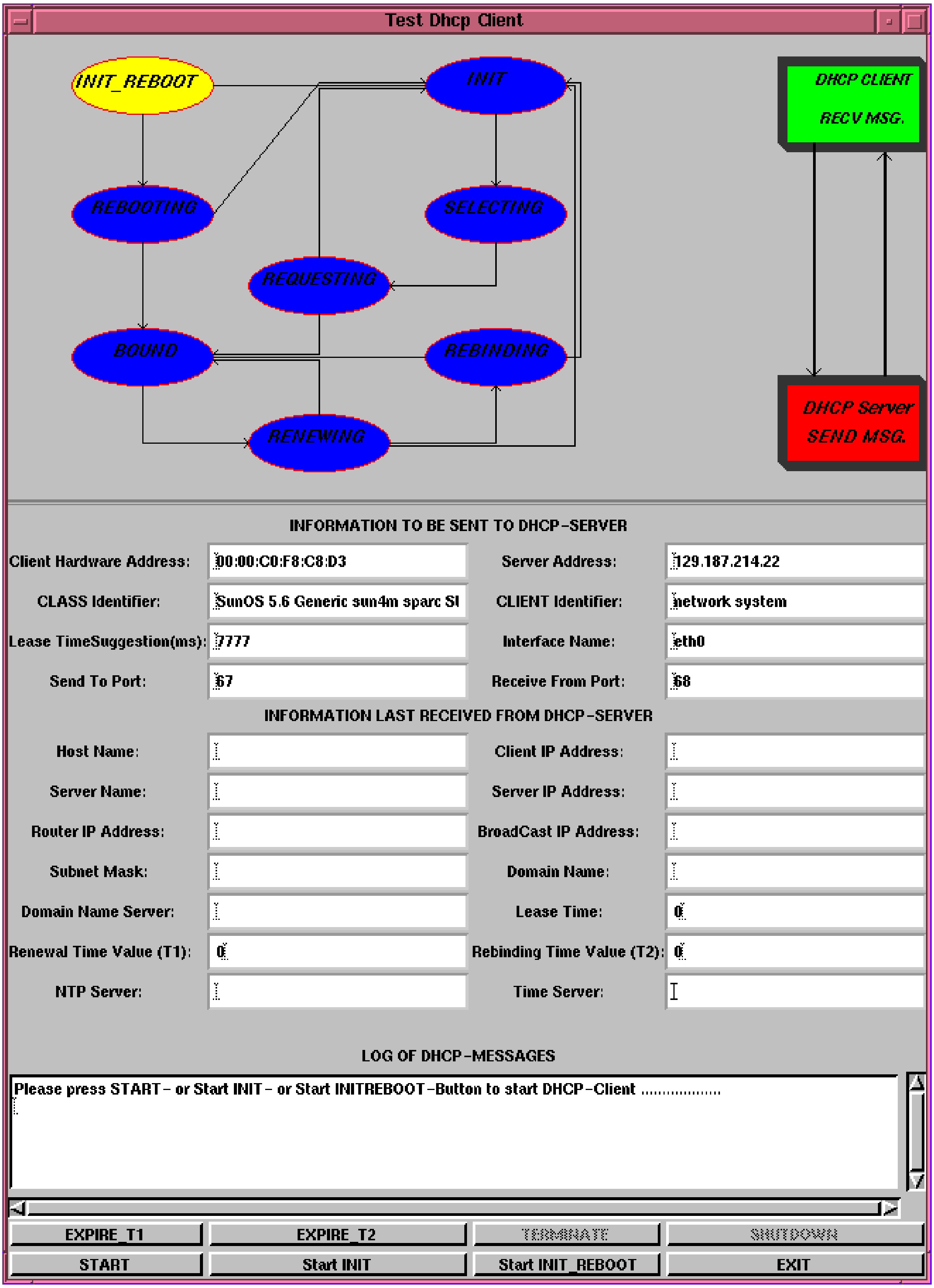
Wir betrachten zuletzt den folgenden Abschnitt unserer Klassenmodell



Beide Klassen wurden in den vorherigen Abschnitten erläutert. Da werden auf den Zusammenhang beider Klassen eingehen.
Der DHCP-Client kann jetzt gestartet werden, da die Klasse StartClient die Komponenten definiert, die die „main()-“Methode in der Klasse Main startet, wobei die „main()-“Methode in der Klasse Main unsere Hauptmethode „dhcpClient“ aufruft.

4.4 Die Benutzung des DHCP-Clients

Die Benutzeroberfläche sieht wie folgt aus:



Wie funktioniert unser DHCP-Client?

1. Man muß die Hardwareadresse von dem Rechner, der nach einem DHCP-Server anfragen senden möchte, in dem Feld „Client Hardware Address“ eingeben.

2.. In dem Feld „Server Address“ muß man den Rechner-Name oder seine IP-Adresse , wo der DHCP-Server gestartet wurde (z.B. hpheger0), eingeben.
Man kann auch eine Broadcast-Adresse eingeben (z.B. 129.187.214.255 oder 255.255.255.255), falls man keinen bestimmten Server nach Konfiguration anfragen will.

3.. Man kann ein Class Identifier in dem Feld „CLASS Identifier“ eingeben.

4.. Man kann ein Client Identifier in dem Feld „CLIENT Identifier“ eingeben.

5. Man kann auch eine Lease Time vorschlagen, indem man in dem Feld „Lease Time Suggestion“ in ms als Zahl eingibt.

6. Man kann die Interface Name festlegen, indem man in dem Feld „Interface Name“ eingibt (z.B. lan0 oder eth0).

7. Die Ports, auf die Nachrichtenaustausch stattfinden muß, können gewählt werden.

8. Man kann die Renewing Time auf kleinen Wert, falls es nötig ist, festsetzen, indem man das Button „EXPIRE_T1“ aktiviert.
Dasselbe gilt auch für die Rebinding Time, da muß aber das Button „EXPIRE_T2“ aktiviert werden.

9. Das Programm wird gestartet, wenn man ein der folgenden Button aktiviert:

- START: Der DHCP-Client trifft die allein die Entscheidung in welchem Zustand gestartet werden soll.

- Start INIT: Der DHCP-Client schlägt keine IP-Adresse vor, falls eine von einer alten Sitzung vergeben war.

- Start INIT_REBOOT: Der DHCP-Client schlägt keine IP-Adresse vor, falls eine von einer alten Sitzung vergeben war.

10. Man kann eine RELEASE-Message an den DHCP-Server senden, falls die IP-Adresse nicht benötigt wird, indem man das Button „SHUTDOWN“ im Zustand BOUND aktiviert.

11. Der DHCP-Client wird angehalten -d.h., die laufende Sitzung zwischen dem DHCP-Client und -Server wird abgebrochen, um neue Anfrage mit neuen Einstellung zu starten- wenn man das Button „TERMINATE“ aktiviert wird.

12. Die Sitzung des DHCP-Client kann mit „EXIT“ beendet werden.

4.5 Die Durchführung von Tests

4.5.1 Beschreibung der Endgeräte:

1. dhcpnm8:

IP-Adresse	129.187.214.18
Betriebsystem	Linux/Debian 2.0.30
DHCP-Server	Internet Software Consortium DHCPD $Name:DHCP-970609$

2. pchegering7: auf diesem Rechner sind zwei Versionen von DHCP-Server installiert.

IP-Adresse	129.187.214.47
Betriebsystem	Linux /Debian 2.0.30
DHCP-Server (1)	Internet Software Consortium DHCPD $Name DHCP-970609$
DHCP-Server (2)	Internet Software Consortium DHCPD $Name V1-0-1$

3. hpheger0:

IP-Adresse	129.187.214.20
Betriebsystem	HP-UX 10.20 (Unix)
DHCP-Server	HP-UX BOOTPD $Name bootpd 2.4 1.17.112.7$

4. hpheger7:

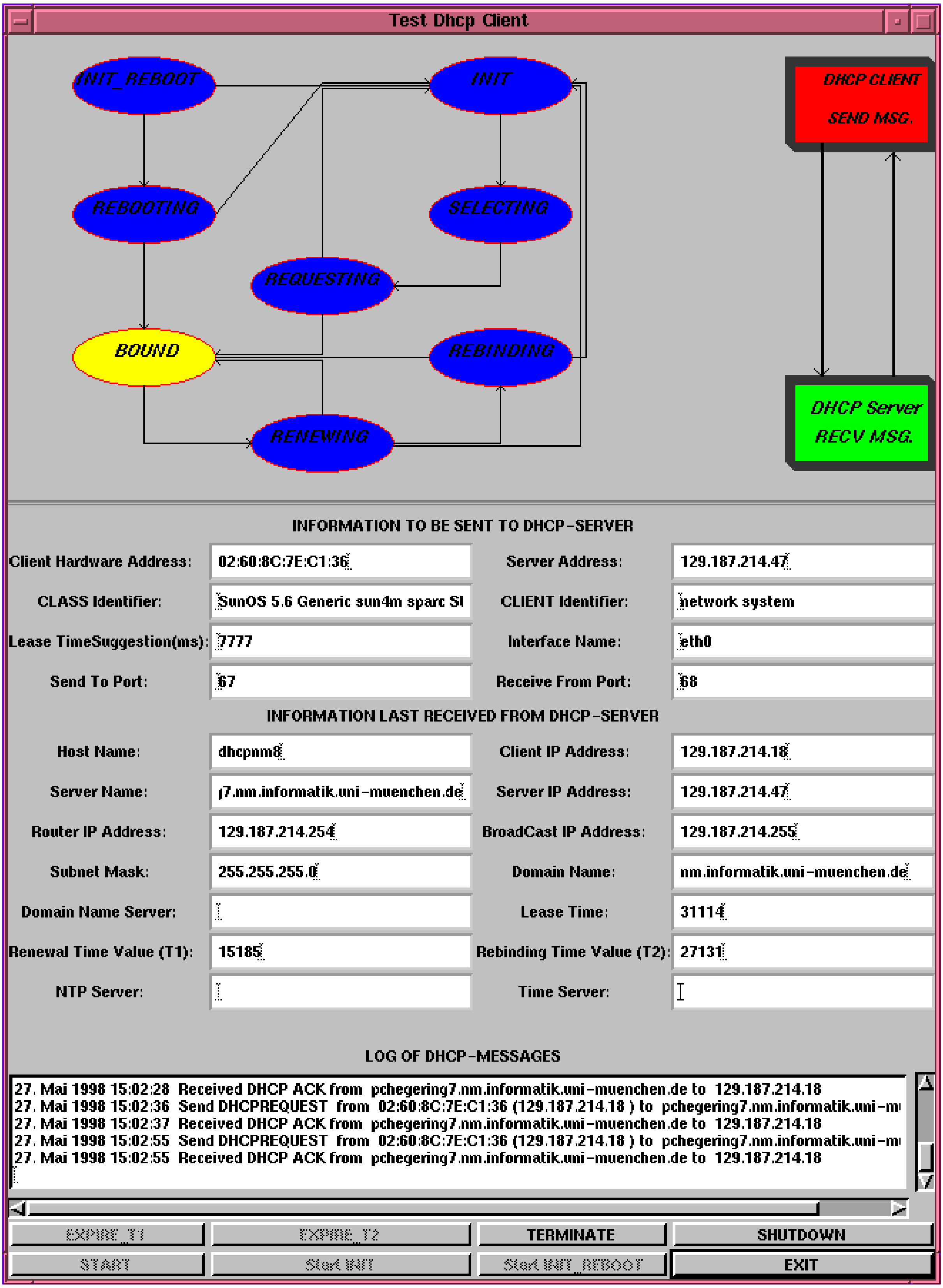
IP-Adresse	129.187.214.27
Betriebsystem	HP-UX 9.20 (Unix)
DHCP-Server	Internet Software Consortium DHCPD $Name V1-0-1$

4.5.2 Durchführung der Tests:

Im folgenden werden Tests auf den DHCP-Client unter verschiedenen Umgebungen durchgeführt, und die Resultate werden zu jedem Test angegeben.

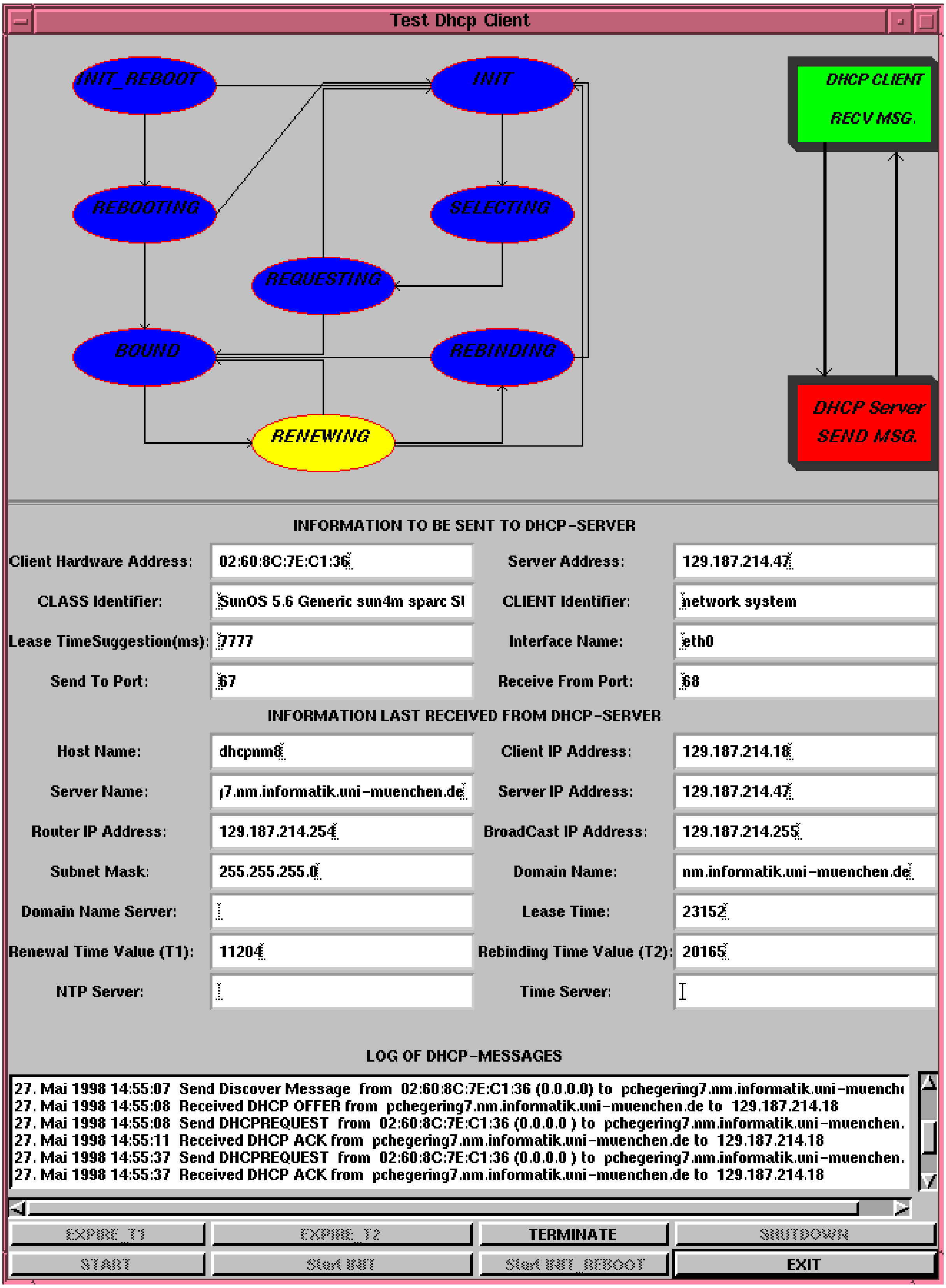
1. pchegering7 / Server (1) ⟷ dhcpnm8 / Client

Das Ergebnis des Tests:



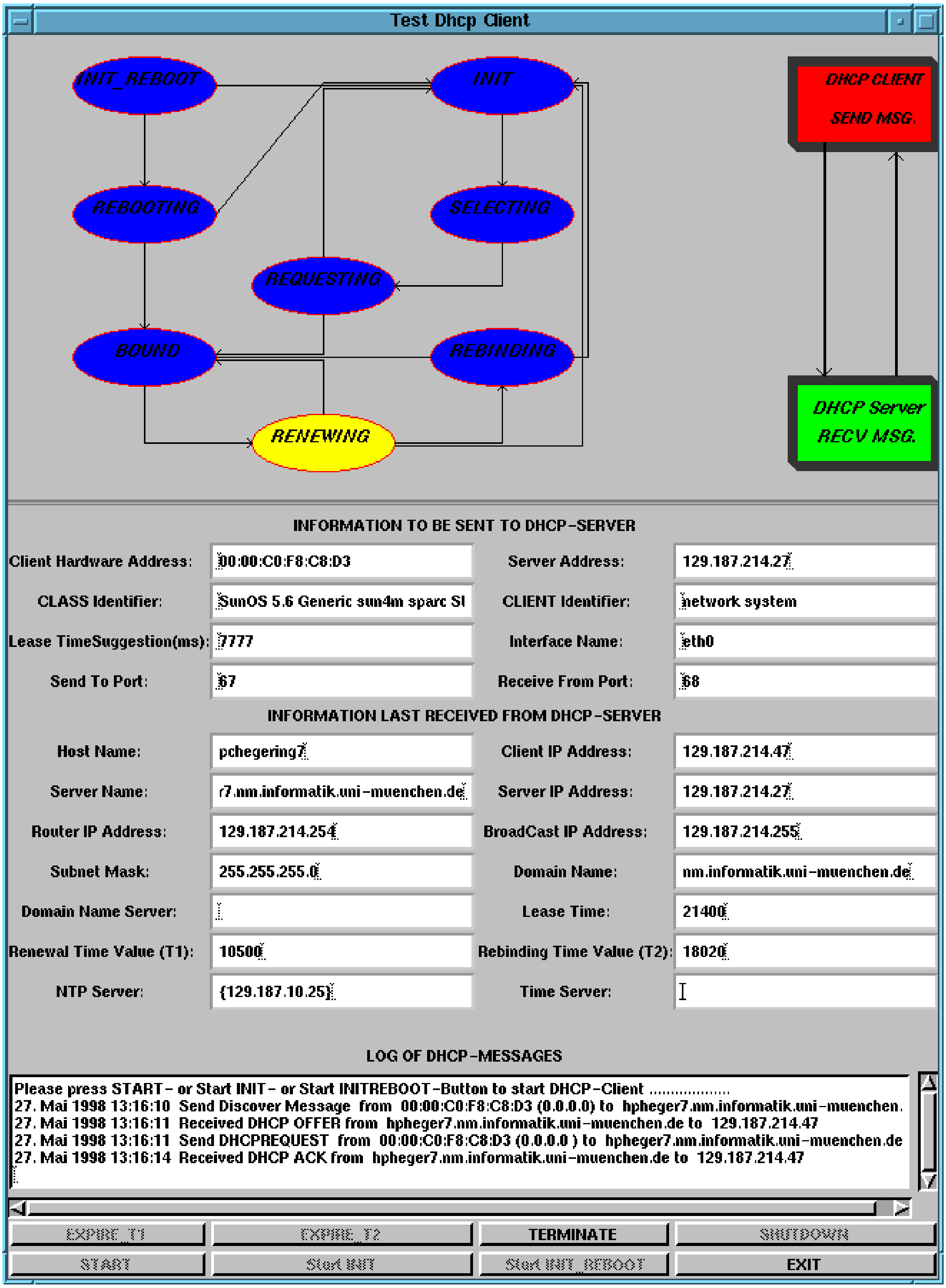
2. pchegering7 / Server (2) ⟷ dhcpnm8 / Client

Das Ergebnis des Tests:



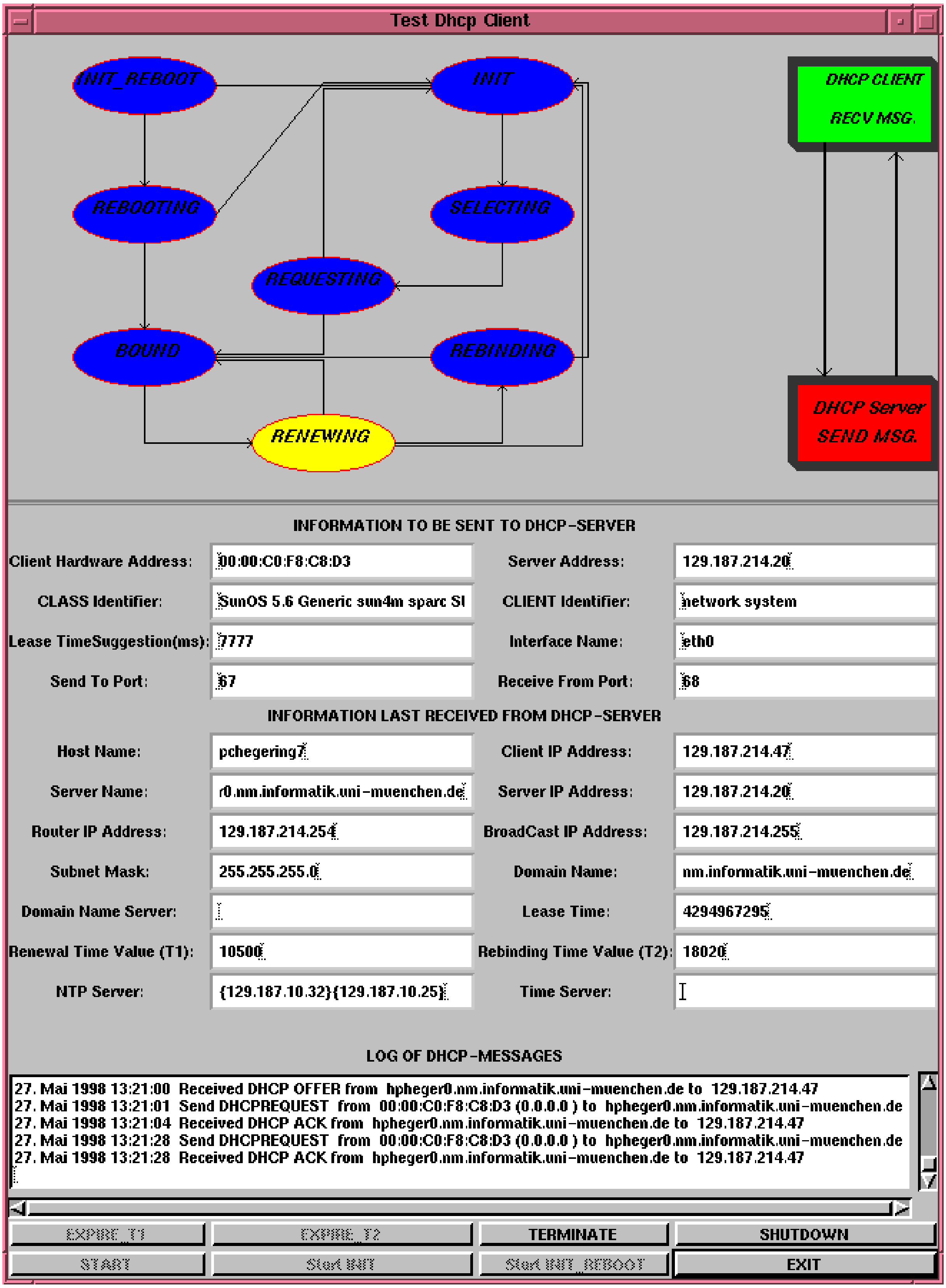
3. hpheger7 / Server (2) ⟷ pchegering7 / Client

Das Ergebnis des Tests:



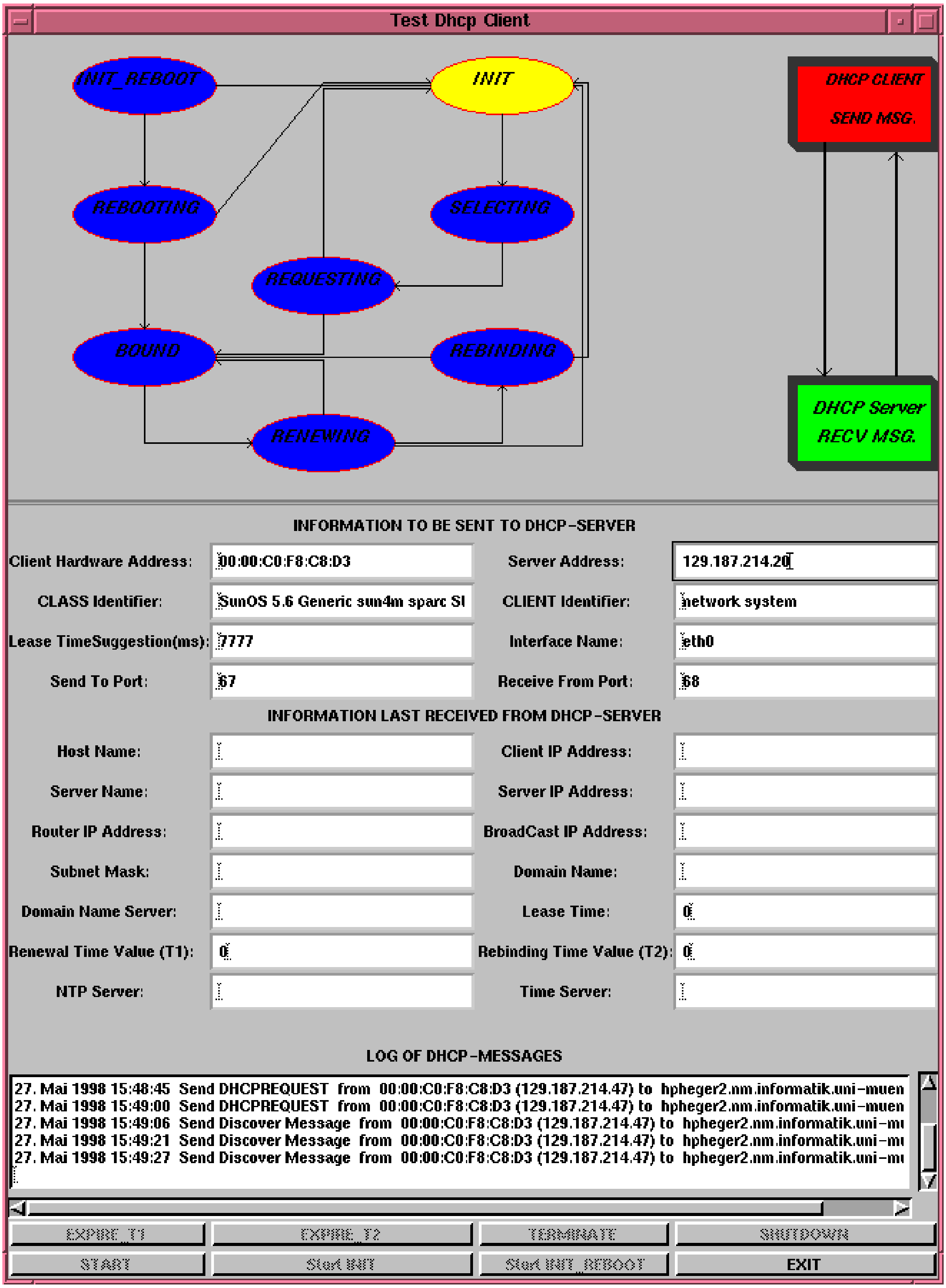
4. hpheger0 / Server ⟷ pchegering7 / Client

Das Ergebnis des Tests:



5. Der Server auf hpheger0 gestoppt pchegering7/Client

Das Ergebnis des Tests:



4.6 Installation der DHCP-Client-Software

Der Software-Code in dem DhcpClient-Package muß in einem Verzeichnis mit dem Namen “DhcpClient“ gespeichert werden, und man erstellt ein Verzeichnis mit dem Namen “classes“, um die Klassen nach dem Compilieren getrennt zu speichern. In dem Makefile.Main werden die Pfade von dem Java-Interpreter, -Compiler und von dem CLASSPATH festgelegt. Für die verschiedenen Systeme (SUN, LINUX, HP-UX10) wurde es jeweils ein Makefile mit dem passenden Pfad von dem Java-Interpreter -Compiler festgelegt.

Graphische Darstellung der Verzeichnisstruktur sieht wie folgt aus:

```
Sourcen ..................  DhcpClient enthält *.java
    :
    :
    :.....................  classes enthält *.class
    :
    :
    :.....................  Makefile.Main enthält die Pfade von dem
                            Compiler, Interpreter und Classes
```

Welche JDK-Version wird benötigt ?

Der JDK1.1 ist erforderlich, um die Klassen zu compilieren und zu interpretieren.
Der JDK1.1 spielt insbesondere für diese Implementierung eine große Rolle, da er z.B. die Broadcast-Nachrichten unterstützt, was bei der älteren Versionen nicht möglich war.

Ist die Software plattformabhängig ?

Nein, die Software kann auf jeder Plattform gestartet werden.

Schritte der Installation der DhcpClient:

1. JDK1.1 muß installiert werden.

2. Überprüfen der Korrektheit der Pfaden in dem Makefile.

3. Compilieren der Klassen, indem der Befehl “make“ in dem Verzeichnis „src“ aufgerufen wird.

4. Wenn die Klassen bereits compiliert sind, dann kann man jetzt den DhcpClietn starten.
   Mit dem Befehl “make dhcpclient“ wird der DhcpClient gestartet.

5. *.class können mit dem Befehl “make clean“ gelöscht werden, falls das nötig ist.

4.7 Zusammenfassung und Ausblick

Für die Installation von TCP/IP auf einem PC werden eine Menge von Konfigurationswerten benötigt. Diese Werte können über einen Konfiguration-Server bereitgestellt werden.
Das Dynamic Host Configuration Protocol (DHCP) erweitert BOOTP auf den kompletten Satz von Konfigurationsparameter, wie sie im RFC definiert sind. Es bietet außerdem eine dynamische Vergabe von IP-Adressen und ermöglicht dadurch die optimale Nutzung eines eingeschränkten Adreßbereichs.

Die Tests haben die prinzipielle Funktionsfähigkeiten der getesteten DHCP-Client-Implementierung und ihre Verträglichkeit mit den verschiedenen BOOTPD- und DHCP-Servern gezeigt. Die Robustheit der Software im realen Gebrauch könnte verbessert werden.

Ferner hat sich die Notwendigkeit von der neuen Compiler- und Interpreter-Version der Programmiersprache Java 1.1 gezeigt, um die Implementierung ohne irgendeine Unterstüzung der anderen Programmiersprachen zu zwingen. Denn ohne diese neuen Version von Java war die Realisierung der Broadcast-Nachrichten bei Java 1.0 ohne Unterstützung der Programmiersprache C nicht möglich.Da mußte die Klasse „DatagramPacket“ in dem Java-Package „java.net“ mit nativen Methoden ergänzt werden.
Die neue Java-Version bietet auch eine bessere Version der Methode „receive“ der Klasse „DatagramSocket“ in dem Packege „java.init“ zur Verfügung. Die Methode „receive“ wird nach einer bestimmt definierten Zeit abgebrochen, und das Programm findet wieder seinen normalen Ablauf, was in Java 1.0 ohne Abhilfe der Programmiersprache C nicht möglich war, da in Java 1.0 die „receive-“Methode das gesamte Programm blockiert hat, falls keine Pakete eingetroffen sind.

Literaturverzeichnis

[Ale93] S. Alexander, “DHCP Options and BOOTP Vendor Extensions“, RFC 1541, Cathegory: Standards Track, October 1993.

[BuKr96] Peter Buhrmann, Thomas Kretzberg, Java, Addison-Wesley Verlag 1996.

[Com95] Douglas E. Comer. Interworking with TCP/IP, Volume 1: Principles, Protocols and Architecture. Prentice Hall, Third Edition, 1995.

[Cro85]Bill Croft, “ARPA-Internet Community“, RFC 951 Cathegory: Stanford University, Sun Microsoft, Septemper 1985.

[Dro97] Ralph Droms, “Dynamic Host Configuration Protocol“, RFC 2131, Cathegory: Standards Track, March 1997.

[Rie96] Gernot Riegert. Diplomarbeit an der TU-München “Erstellung einer Managementschnittstelle für DHCP-Server“; 1996.

[Hei96] Mathias Hein, TCP/IP Internet-Protokolle im professionellen Einsatz; An International

[Hun96 Craig Hunt. TCP/IP Netzanbindung von PCs; O’Reilly Verlag, 1996. Thomson Publishing Company, 1996.