Baltzer Journals

Using DHCP with Computers that Move

CHARLES E. PERKINS[1], AND KEVIN LUO[2]

[1] *Mobile Systems*
IBM, T.J. Watson Research Center
Hawthorne, NY 10562
E-mail: perk@watson.ibm.com

[2] *Computer Science Department*
Columbia University
New York, NY 10027
E-mail: luo@cs.columbia.edu

The Dynamic Host Configuration Protocol *(DHCP)* was designed to allow the frequent allocation of resources and configuration information useful to Internet hosts at boot time, including Internet addresses in particular. It turns out that getting a new Internet address is crucial to the problem of enabling the movement of Internet hosts from one network to another, and thus DHCP is quite relevant to the problem of providing seamless, transparent mobility to Internet hosts. We decided to investigate the ways that DHCP could be of assistance in this regard. Since the DHCP protocol was not itself designed for the purpose of providing host mobility, a number of problems arise. Our experience with deploying DHCP, and our proposed mechanisms for the use of DHCP with mobile computers, are the subjects of this paper.

Keywords: mobile networking, mobile-IP, mobility, DHCP, portability

1 Introduction

The Internet, by virtue of its ability to provide worldwide connectivity between network-connected computers, has continued an exponential growth pattern for many years. It is now becoming the obvious candidate for providing a basic international information superhighway. Tools developed on the Internet are already starting to revolutionize the entire way that people think about academic and technical collaboration. Yet, historically it has been considered difficult to manage an Internet connection. One of the biggest areas of difficulty is just in getting started, which involves a good bit of bureaucratic registration, and negotiation with agencies that might provide basic access to some network already a part of the Internet. An important result of the registration is the acquisition a range of Internet addresses sufficient for the operational needs of the organization. Organizations typically request far more Internet addresses than they will need at first, in order to provide for the expected needs for future expansion.

The procedures required for communicating with the Internet continue long after an organization has completed its initial setup, especially when multiple networks are to be operated within the same organizational entity. One of the standard procedures, that has traditionally faced Internet users, is that of acquiring their own individual Internet addresses, selected from among one of the ranges of addresses initially allocated to their organization. That chore has to be done each time the computer is moved from one network to another within the organization, and introduces requirements for bookkeeping, and identifying the correct Internet addresses associated with each individual network. Thus, the users have to find out and apply certain technical information that makes little difference to them otherwise, or else initiate maintenance and installation requests from a service group within their organization. All of these inconveniences add up to major costs within the organization, and have been a factor inhibiting the acceptance of the Internet by casual users.

Recognizing this problem, the Internet Engineering Task Force created a working group to standardize a protocol, known as the Dynamic Host Configuration Protocol (DHCP) [5] which enables casual users to plug in their new computers and become full Internet citizens without necessarily knowing anything about the configuration or numbering of their organization's LANs. DHCP allows workstation software to automatically request and receive an Internet address from a service machine within the organization, and ensures that the Internet address issued to the client workstation is appropriate for correct operation at that client's particular network. With DHCP, new Internet users may benefit from global connectivity while still enjoying the convenience, offered by other networking protocols, of "plug-and-play" operation. The careful use of DHCP should drastically reduce the administrative burden on the network support groups at most installations using TCP/IP protocols.

DHCP offers possible benefits to the users of a newer class of Internet computer, the mobile computers. With battery-powered, lightweight laptops that can use wireless network data links, it is quite feasible to move from place to place and yet remain in contact with all the rich resources which make Internet connectivity so attractive. Yet, this very movement violates the design assumptions of the Internet [15], because the Internet address encodes location information as well as the specific identity of the device attaching the computer to the Internet. With DHCP, it is possible to imagine obtaining an Internet address appropriate for the LAN most convenient to a mobile computer at any particular time. Then, when movement occurs, the dynamic nature of the Internet address allocation managed by DHCP would allow the release of the previous address and the acquisition of a new address appropriate for the new location of the mobile computer.

In this paper, we explore the technical issues, advantages, and disadvantages of using DHCP for the purpose of supporting mobile computing. We describe first the overall operation of DHCP. We describe two different strategies for using DHCP to support the needs of mobile networking. Then we detail our implementation experience and observations about what we had to do on our various platforms to get DHCP working. We describe some of the procedures that will be required for the successful administration of DHCP within an organization, and mention some areas of discrepancy between our implementation and the standard. We also briefly describe some of the impact of using DHCP on the usual Domain Name Service operations, and the security deficiencies of DHCP. Lastly, we describe possible areas of future work, and summarize what we have learned from our investigation and development of DHCP for supporting mobile computing.

2 Basic Operation

DHCP is designed around a traditional client/server model of operation. The client system makes known its request to obtain an Internet address along with any other necessary configuration parameters, and the server supplies those parameters according to a multistep handshaking sequence. True to its name, DHCP provides an extensible mechanism for supplying a wide range of configuration information to its clients, and emerged in its final form as a mostly compatible upgrade to the *BOOTP* [4] protocol. Bootp was itself developed to send operating system images and other boot-time information to diskless workstations, but does not have any of the dynamic allocation features which characterize DHCP. A DHCP server leases an Internet address to a client for some stated amount of time (which can be infinite), and the client may renew the release as desired (and according to policy set by the local administration). Other values typically distributed to DHCP clients include the subnet mask, default router, and the address of some local Domain Name Server (*nameserver*, or *DNS*) hosts [11], along with the name of the domain they serve. All of these are configuration details that otherwise would have to be inserted somehow into the bootstrap sequence of the client system; typically this has been done by editing startup scripts and configuration files. Most experienced Internet users have made mistakes many times while performing these procedures; new users find it very frustrating to find the information, and figure out where it goes. Problems inevitably arise in a large enough user population.

To get configuration data from DHCP, the client must first discover a suitable server. It does this by broadcasting to a local recipient, at a well known DHCP

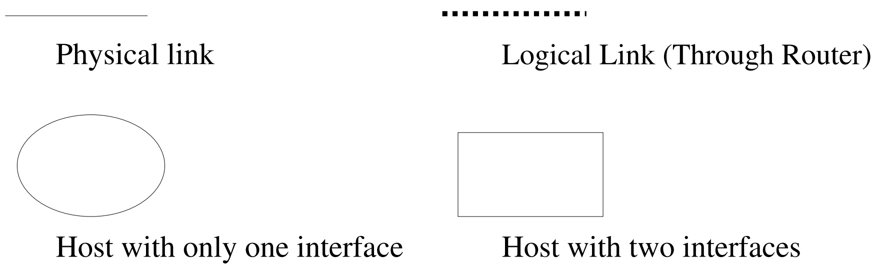

Figure 1: Notational conventions used in this paper

server UDP [14] port number[67]. The local recipient either satisfies the request if it is a DHCP server, or transmits it to a DHCP server if it is a relay. A DHCP server answers the discovery probe by offering a selection of configuration parameters to the DHCP client. If the DHCP server is not local to the same subnet as the client, the server transmits its request back through the relay agent which originally forwarded the client's discovery probe. Lastly, if the client accepts the configuration (in particular, the Internet address) offered by the server, the client requests that the server commit the data and resources to the client. For a typical configuration using a single DHCP server, see figure 2.

The DHCP protocol allows for multiple DHCP servers with overlapping domains of service. Thus, it is expected that a DHCP client may receive multiple offers of resources from various DHCP servers which the local administration maintains. From the several offers, the client picks the one which best meets its needs (or, the first one) and implicitly rejects the others. For instance, there may be several DHCP servers with an Internet address available for the client, but only one server may have all the machine-specific configuration details that the client needs for complete operation. Lastly, the DHCP server reassures the client that the commit operation has occurred by acknowledging the client's acceptance of the original offer. For instance, in figure 3, there are multiple DHCP relays serving the same DHCP client population on a network, each sending requests to their own non-disjoint collection of DHCP servers. The DHCP servers known to each DHCP relay must be determined by the local network administration.

A client may also indicate its preference for the re-issuance of a previously allocated Internet address. In that case it is likely that only one DHCP server can satisfy the client's preferences. To simplify the operation of the protocol, the above outlined four-step handshake is then abbreviated to only two steps. The client just indicates its preference to the server, and if the server approves the request it indicates the fact by sending a positive acknowledgment. There is no longer any need to select from among a set of competing service offers. The DHCP specification allows the server to indicate to the client how long to wait before trying to renew the lease, typically half the total duration of the lease. If the

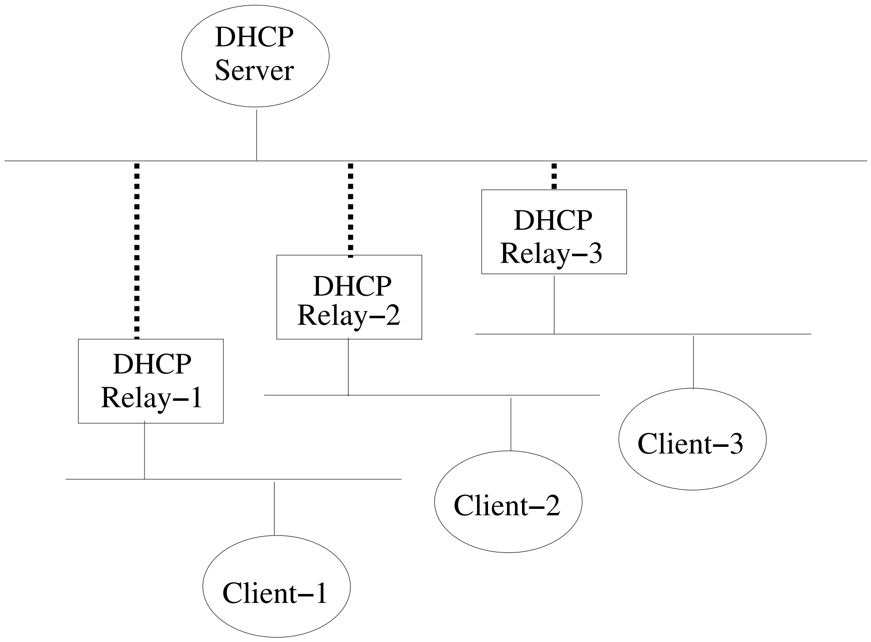

Figure 2: DHCP client/server model

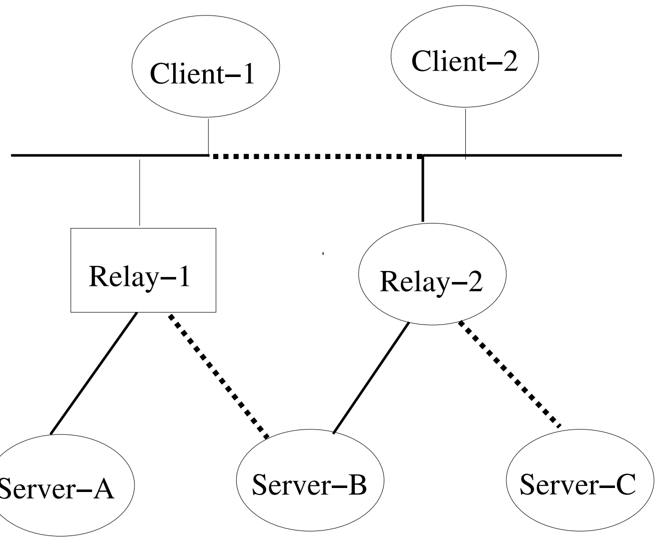

Figure 3: Relaying messages to several DHCP servers

client tries to renew a lease but fails with the original DHCP server, it can try to renew the lease with all available DHCP servers; but, once the lease expires, the client is expressly prohibited from continuing the use of that particular Internet address. A client can release its Internet address at any time.

DHCP relays allow a single server to provide its services to clients on many interconnected networks, instead of requiring separate DHCP servers on each network. This important function is what makes DHCP useful in many typical environments. DHCP clients cannot conveniently be configured to know anything about DHCP servers; the whole motivation for DHCP is to avoid manually configuring clients. So, the client must send its requests for service to the well known DHCP server port at a broadcast address. The relay essentially collects client requests for information on a local subnet, and modifies the packet before relaying it to whichever DHCP servers the relay is configured for. The modified packet contains one of the relay's Internet addresses, in a field called "giaddr" for "gateway internet address", for the DHCP server to use as the destination address for return packets; the server also uses this field to determine which IP addresses will work for the client. We will describe later in the paper the implementation difficulties related that we encountered.

3 Using DHCP with Computers that Move

Internet addresses encode routing information that has to change as computers move from place to place. Therefore, some Internet address associated with a computer, if not the actual address of the computer itself, must change if the computer is to be moved from one network to another. The details of this association and the way the changes are managed define various methods for achieving portability and mobility for Internet hosts. Since DHCP is designed to manage the dynamic association between particular Internet hosts and their Internet addresses, it is suited for some application in enabling computers to move (i.e., change their point of attachment to the Internet).

3.1 Using DHCP for Portability

DHCP can be used to enable portability for computing devices (e.g., laptop computers). The method is straightforward. At each new attachment point, the laptop sends out a request for a new Internet address to whatever DHCP server happens to be available. When a suitable Internet address has been negotiated, the client can begin any network transactions using that new address just as any other system on the same network. Since DHCP, by design and definition, allocates an Internet address which is specifically suited for use with the subnet on which the DHCP client resides, the natural operation of the Internet protocol en-

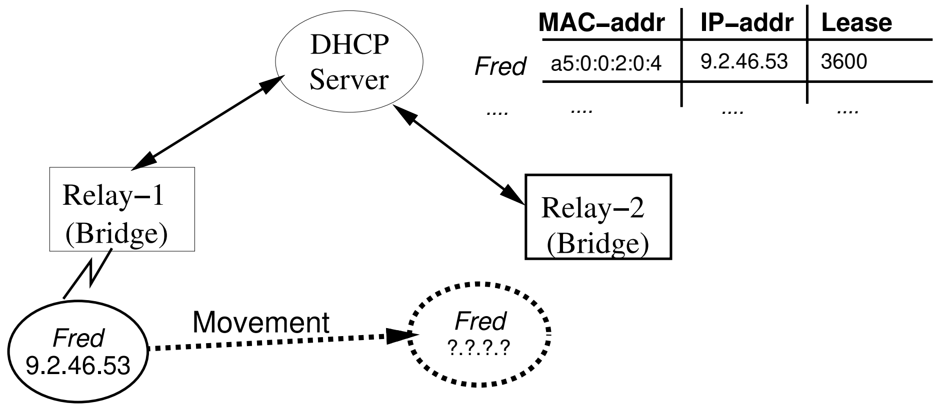

Figure 4: Moving and getting a new Internet address from a DHCP server; fragment of server's table of clients also shown

sures that packets will arrive at the mobile client's current location. See figure 4 for a schematic picture of operation. As shown, using wireless mobile computers, the access points are both bridges and DHCP relays.

Unfortunately, there are other relevant factors that prevent this from being a total solution to the problem of enabling transparent mobility for the users. In the first place, it is not so easy for other systems on the Internet to initiate transactions with the portable computing device. There is no clear way for those systems to determine the address which has been allocated to the portable device by the DHCP server. Worse, most transactions between Internet computers are mediated by resolving human-readable names into Internet addresses, and this name resolution is performed by the Domain Name Server. The DNS then needs to update its internally stored mapping for the name of a DHCP client, each time the client obtains a new Internet address. The precise mechanism by which this mapping should be updated is currently undecided, although there are some good candidates for standardization ([16], [7]). Any use of DHCP to provide a portable computer with its only Internet address will almost certainly place additional emphasis on the need for such a dynamic update mechanism for DNS. We will revisit this topic later when we discuss the security implications of mobile computers.

In addition, any time the DHCP client obtains a new Internet address, it will most likely have to reinitialize all of its network connections to operate with the new address instead of the old one. This means that certain operations which previously only occurred at boot time on the clients will have to be performed whenever a new Internet address is obtained, creating a new set of operational requirements for the functioning of the TCP/IP protocol stack with mobile DHCP

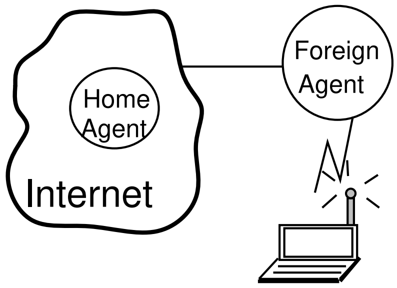

Figure 5: Mobile-IP

clients. It also means that all network connections will need to be re-established when the client moves and obtains a new Internet address.

3.2 Using DHCP for Mobility

As just described, a straightforward application of the DHCP protocol enables a partial solution to the needs of portable computing but many problems remain. Another strategy is to use DHCP in conjunction with the emerging mobile networking protocol [8] being discussed within the IETF mobile-IP working group. This protocol, hereafter called 'mobile-IP', defines a way for laptop stations to keep the same Internet address even while moving from network to network within the global Internet. Although it is not the purpose of this paper to fully describe mobile-IP, a short overview will be given in order to better explain the way that DHCP can be used to support transparent mobility instead of only portability.

Mobile-IP proposes that mobile computers be allocated permanent addresses on a Home Network, which may be either a physical network or only a virtual network (i.e., a network served by a router but not embodied in any particular extent of physical wiring). When the mobile computer moves to a new location, it obtains by some means a temporary forwarding address (a "care-of address"). The router on the Home Network (the "home agent") will deliver the mobile node's packets to the care-of address. The most straightforward implementation of a network entity supporting a care-of address is known as a "foreign agent", in contradistinction to the home agent on the Home Network. See Figure 5. The home agent and foreign agent collaborate to provide, to all of the mobile node's possible Internet correspondents, the illusion that the mobile node is always available on its home network. One requirement of this scheme is that the mobile host report each new care-of address to its home agent. Protocol details can be found in the current draft under consideration by the mobile-IP working group [8].

With DHCP available, the mobile host can obtain a new care-of address each time it needs one, even if there is no foreign agent available. If the mobile host is also a DHCP client, and reports its newly allocated address as the care-of address to its home agent, then, as before, the care-of address will certainly be appropriate for operation on the local subnet. In this case, the mobile host will effectively be operating as its own foreign agent, and will have to participate in whatever protocol arrangement the home agent normally uses to deliver packets, destined for the mobile host, to the care-of address. The details of that mechanism will not be discussed here, but it is a simple encapsulation operation of the original packet with a new IP header that includes the care-of address as the new destination address. Allocating temporary IP addresses for use by mobile clients has been suggested numerous times, within the mobile-IP working group of the IETF, in work by John Iaonnidis([9]) where he names such mobile clients "popups", and in earlier projects here at IBM. We expect that, in the medium term, DHCP will often be used to provide dynamically allocated care-of addresses; eventually real foreign agents may be so prevalent that "popups" will rarely occur.

The schematic picture of operation is still as shown in figure 4, except that now the home address of the mobile node does not change; the question marks refer to the new care-of address which is to be allocated. That new Internet address is reported to the node's home agent as its care-of address. The access point is as shown before, and is not a mobile-IP foreign agent.

This strategy has a much better chance of success, since the mobile host can keep using its permanently allocated Internet address. Thus, DNS no longer needs to manage the frequent updates to its address mappings. Correspondents wishing to contact the mobile host may do so by using (as always) its permanent (home) address. All the users Internet applications work almost as expected, and there is the hope for complete transparency.

Nevertheless, there are a few remaining problems. The main difficulty is that the movement is still not transparent; packets which are in transit towards a particular care-of address might be dropped if the mobile host (acting as its own foreign agent, now) has departed and obtained a new temporary care-of address at an attachment point on a new subnet. Clearly, dropping packets is an action incompatible with the goal of providing transparent mobility. In fact, there are special problems with the way that TCP interprets such network events as dropped packets, which have the effect of magnifying the disruption they can cause [2]. To understand the problem introduced by TCP, one must understand that the high reliability of today's wired networks has caused a shift in the way that dropped or delayed packets affect TCP's recovery algorithms. Starting in the mid '80s, TCP was modified to reflect the reality of that time, that all dropped packets were the result of network congestion. The clearest solution to problems caused by network congestion is to "back off" from retransmitting extra packets until the congestion had cleared [10, 3]. Packets are deemed to be dropped if their acknowledgment had not arrived before some time depending on the estimated round trip time between communicating Internet hosts.

This reaction is precisely wrong for the case of a mobile DHCP client which has received a newly allocated care-of address. Since the problem is caused by an actual disconnection instead of network congestion, the most effective reaction would be for the mobile client's correspondents to retransmit dropped packets as soon as possible. One solution to the problem would be for current Internet hosts to be modified, so that their reaction to dropped packets could be tailored to the new realities of wireless mobile communications. Even disregarding the infeasibility of such a massive change to the existing Internet infrastructure, such a solution is made difficult by the theoretical intractability of characterizing the cause of dropped packets. After all, a mobile computer could still experience network congestion, so neither strategy is correct under every circumstance.

An existing foreign agent, however, could well be equipped with code to forward packets to the new location of a mobile client after it moves [12]. This would be arranged by the mobile host with the cooperation of its new foreign agent, and provides a good method for avoiding dropped packets as long as it's physically possible to transmit the packets in transit either directly to the mobile host or indirectly through its new foreign agent.

If a mobile node is programmed to accept packets sent to multiple care-of addresses, and if there is good enough physical coverage, then it can avoid dropped packets. The key is to keep the previously allocated Internet address until the newly allocated Internet address has been registered, and keep accepting packets sent to the previous care-of address as well as the new one. Soon, all packets which were destined to the previous care-of address may be presumed to have been transmitted, forwarded, and delivered; then, the previous care-of address can be released to the DHCP allocation pool again. This solves the black-hole problem almost completely, except for Internet agents which may have cached the mobile node's previous care-of address and not been notified of the movement [12].

Even without such sophistication in the mobile node implementation, DHCP can allow mobile clients to move about with a good deal of performance transparency (if not perfect), and applications will continue to work even in the face of packets dropped in transit, because of the nature of the operation of TCP. UDP applications will also continue to work in most cases, so that the goals of mobile computing is achieved in large measure by applying DHCP to the needs of the mobile client to obtain a temporary care-of address.

How smoothly the mobile client can move from place to place by using DHCP in this way will depend upon whether the DHCP server will allow one of its clients to temporarily have two addresses. In order for a mobile client to get service as soon as possible at a new Internet attachment point, it will probably want to get a new temporary care-of address before releasing the previous one. If the client is forced to release the previous address first, then more time will elapse during which more packets will be unable to be routed to the mobile client. This may cause an unacceptable interruption in the mobile client operation.

DHCP may also be used by mobile clients to obtain temporary addresses from a home network, in analogy to the way that static DHCP clients obtain temporary addresses appropriate to whatever network they happen to reside on. Mobile clients need to obtain addresses on a home network, because those are the networks that is to be populated by mobile computers. In that case, however, the mobile DHCP client can no longer be pre-configured with the address of its home agent. In fact, many participants of the IETF mobile-IP working group have recognized the desirability of allowing the mobile computer to send updates to its home agent without necessarily knowing the address of its home agent. One suggested technique is for the mobile host to use the "directed broadcast" address of its home network as the destination address, when notifying its home agent of its new care-of address. The directed broadcast will naturally be delivered to the home network, and the home agent will naturally be able to receive that packet for further processing. For DHCP mobile clients, another possibility would be to define a new DHCP option, but this alternative carries with it the undesirable consequence of requiring additional static entries to be made to the DHCP server configuration files, introducing yet another source of errors to plague network administrators.

One possible way to simplify the task of network administration with mobile-IP and DHCP is to co-locate the services of home agent and DHCP server. That would have positive benefits, especially if some modification of the mobile-IP basic protocol can be devised to take advantage of the situation. For instance, if the DHCP server receives a request from a mobile DHCP client, and the client's MAC address indicates that it is on the home list of the co-located home agent, then the normal registration process could be drastically shortened.

We suggest co-locating a DHCP relay with every foreign agent. Then, any mobile client entering the service area offered by the foreign agent could expect to get a new home address, if necessary, via that foreign agent, for use as its permanent address as long as the lease indicates. That newly allocated home address would then be useful even in the service areas of other foreign agents, until the time at which the mobile DHCP client was finished and desired to release that home address. This depends not only on having appropriately functional foreign agents, but again on being able to reliably apply updates to the controlling DNS for the domain name by which the mobile DHCP client is known. Standardization of the further modifications needed to support these advanced features seems to be proceeding at a reasonable pace[13].

4 Implementation Details

We implemented and deployed DHCP on several platforms under AIX, OS/2, and OS/400, which are three of IBM's current operating system offerings. Each implementation had special characteristics of interest to people who are involved with developing DHCP services. Our job was considerably simplified by the availability of an existing implementation of DHCP for us to start with on AIX and OS/2. In this section we will outline some of the significant problems we had to solve to make our DHCP services really work.

AIX is IBM's version of Unix, and it runs on a variety of hardware platforms. We were interested in using DHCP on various models of PS/2 and RS/6000 computers. We started with a basic working AIX implementation of client and server DHCP daemons for the RS/6000, which had also been shown to work on another version of OS/2. One of our primary goals was to make sure that the same source code was able to be used on all of our various systems. Unfortunately for us, our compile and link environment needed to be tailored to our version of the OS/2 networking kernel; we had to modify the code to work around the lack of various library support routines, as well as other differences caused by changing compilers and include file structures.

After we had working versions of the client and server codes interoperating on OS/2 and AIX for the RS/6000, we tried to run it on AIX for PS/2. This was simple. The only complicating factor was that some of the include paths had to be changed; practically all of the structures were the same and no significant difficulties were encountered. We also created a DHCP relay, adapted from the DHCP server code. The most general configurations of DHCP server and relay functions present significant design challenges, and cannot be coded in any truly portable way. Our implementations have significant limitations, and we haven't yet determined the best way around them. We will discuss these problems in a little more detail later.

When we were satisfied with the interoperation of our AIX and OS/2 clients, servers, and relays, we tackled the goal of porting our server code to the AS/400 computer system. Since the AS/400 is not usually configured as a router, we did not think that we would invest the necessary effort to make it perform the DHCP relay function. Since our goal was to enable mobile clients to interact with DHCP servers on a variety of IBM products, we did not plan to implement DHCP client functions on the AS/400. Current models of AS/400 products do not seem appropriate for use as mobile computers, although that may change before long. The design and implementation of the TCP/IP on the AS/400 were not derived from any Berkeley Source Distribution; nevertheless, the semantics of the socket calls are remarkably consistent with the standard (and AIX) socket subroutine calls.

```
AIX:                                OS2:
    initialize                          initialize
    demonize                            ... /* no demonize */
    main loop {                       main loop {
        block until alarm(),              block until time expires
           or packet arrives                 or packet arrives

        if (time interrupt)               if (time interrupt)
           cleanup;                           cleanup;
        if (packet arrived)               if (packet arrived)
           process;                           process;
           generate a reply.                  generate a reply.
     }                                  }

AS400:
     initialize
     ... /* no demonize */
     main loop {
        block until packet arrives

        check whether it is time
         yet to cleanup repository

        process packet;
        generate a reply
     }
```

Figure 6: DHCP server structures

The same DHCP client and server code operated on different OS/2 installations without difficulty, mostly due to the fact that all of our platforms use compatible processor and bus architectures.

4.1 Server Implementation

Porting our implementation of the server presented problems which are familiar to most programmers who have been involved with such efforts. Typical pitfalls surround code which makes non-portable assumptions about the size of address pointers, size of an integer, and so on. Even though some of the pitfalls are not unfamiliar, debugging the resulting problems can still remain challenging and quite time consuming, as is often the case with multicomponent client/server systems.

Consider one view of the structure of our server code on the various systems, as shown in figure 6. Notice that in our implementations, the server doesn't create another process to handle the client's request, as other network programs do. The reason is that the time it takes to process the client's request and generate a reply is relatively short. Moreover, if there are multiple server processes which could possibly allocate the same resources to DHCP clients, then some sort of mutual exclusion would have be enforced to prevent such conflicts.

Time-based services are one area of significant difference between our operating system environments. A DHCP server needs to monitor on the time clock for cleaning up its IP allocation repository, meanwhile waiting on its well-known port for incoming client requests. On AIX, *alarm()* is used to generate an interrupt when it is time to perform cleanup actions. On OS/2, we *DosCreateThread()* a separate process to keep track of elapsed time and post to a semaphore when the time expires. When the semaphore is posted, the main process is awakened. Neither of the above two methods work in AS400. There, the main process always waits at the well-known port for incoming packets. When a packet arrives, it checks to see if enough time has elapsed to require a check for cleaning up the repository. Afterwards, the main process can continue to process the client request.

Our AIX implementations use *fork()* to offer the additional nicety of "demonizing" the DHCP server – i.e., allowing it to operate without keyboard interruption and console display. In OS/2 this is accomplished by the simple expedient of running it from CONFIG.SYS, or (after initialization) in a window which is iconified and then forgotten.

One interesting question is how well servers on the various platforms handle time-consuming client requests when many requests are queued. This problem is most critical on the AS/400, since it performs garbage collection only after being awakened by the arrival of a client request. If client requests fail to be processed before being timed out and reissued, we will look for ways to speed up AS/400 response time, or possibly increase the client time-out interval before a second request is generated. We may make some provision for using a back off scheme for client requests that remain unanswered.

In the general case of a DHCP server with more than one network interface, the Internet addresses which might be offered to a client sending a DHCP_DISCOVER packet are determined by the network interface the packet came from. This information may be unavailable to user-level programs which use standard, portable coding techniques, because of the way sockets are implemented with Berkeley Unix and other compatible systems. The problem arises because the server daemon must listen for packets addressed to Internet address 255.255.255.255 (all ones), at the well known DHCP server port (67). That's the only possible address to which a DHCP client could send DHCP_DISCOVER messages, since the client does not know the subnet mask appropriate for the local subnet or any other local information. The all-ones address becomes the only client identification available to the user-level program, and it is clearly not enough to identify the originating network of the client's packet. Our current server implementation does not have this problem, because it is restricted to work with a single network interface adapter.

4.2 Client Implementation

We implemented DHCP clients for AIX and OS/2 systems on several platforms. In all cases the client program was a finite state machine. The client's states are defined as follows:

- CS_NONE (No state change)
- CS_INIT
- CS_INITREBOOT
- CS_REBOOTING
- CS_SELECTING
- CS_REQUESTING
- CS_BOUND
- CS_RENEWING
- CS_REBINDING

These states correspond exactly to the state machine described in the DHCP specification (see Figure 7).

A table specifies for each client state and event, which actions must be taken and the next state to which the client must transfer. For each possible client state, the state table contains, for each possible input to the state machine, a transition to a new state and a list of actions. There are local actions, actions relevant to the transmission of packets to the server (for instance, setting values in fields within the packets), and timer settings.

For example, in the CS_SELECTING state, the state table is constructed as follows (as illustrated in figure 8):

The comment on the left hand side describes client events. For example, "OFFER" specifies that the client receives a DHCP_OFFER packet from the server. LACT_RECORDLEASE means the client will record the lease time. RACT_REQUEST | RACT_BCASTFLAG | RACT_SERVERID | RACT_REQIPADDR means the client will send a DHCP_REQUEST packet with broadcast flag on, specified server id and requested Internet address. TIMER_RESTART | TIMER_Ta means the client will start a new timer Ta, and CS_REQUESTING specifies the client's next state.

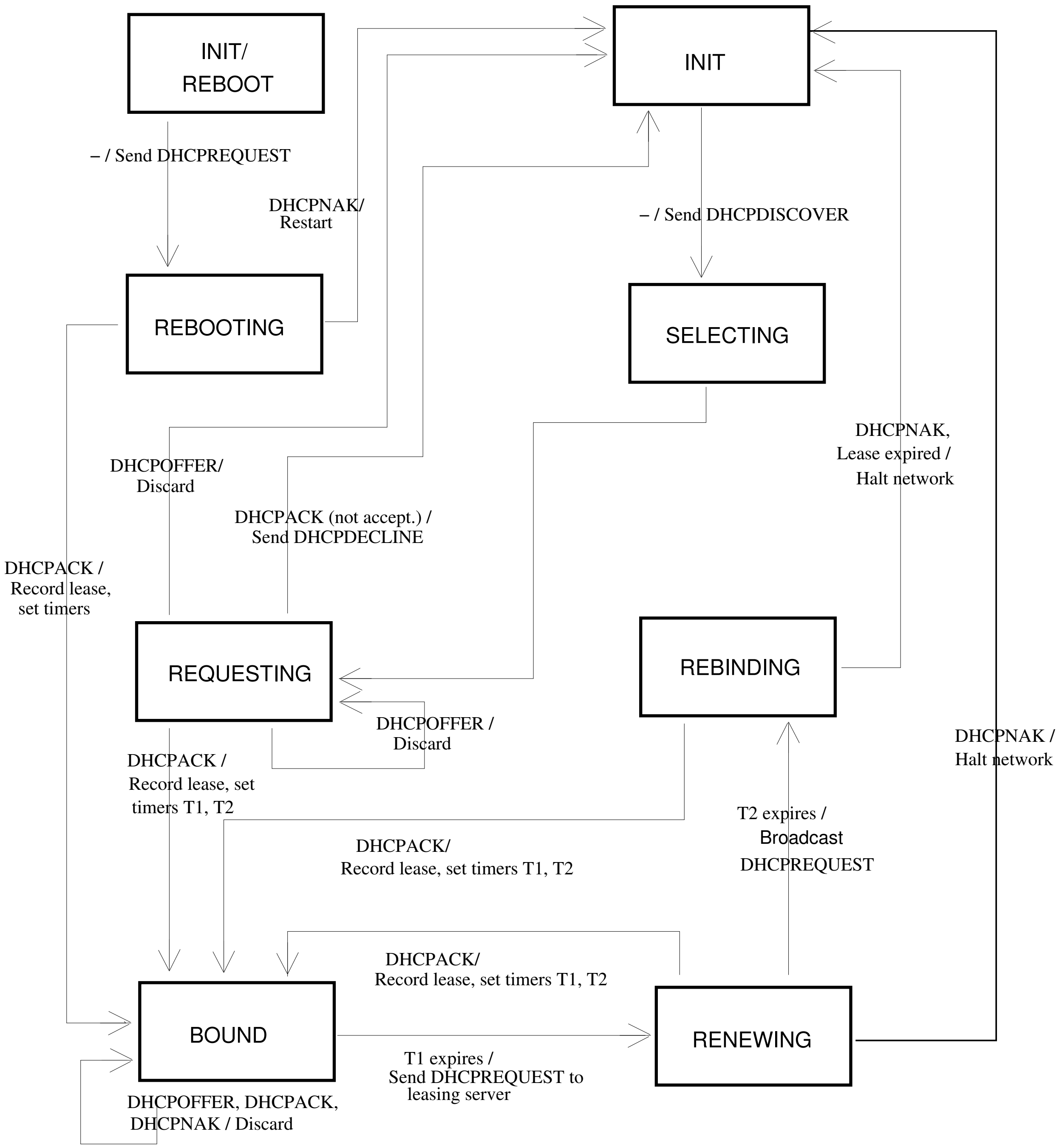

Figure 7: DHCP client state machine

```
/* START    */  { 0, 0, 0, 0 },
/* OFFER    */  { LACT_RECORDLEASE,
                  RACT_REQUEST | RACT_BCASTFLAG |
                          RACT_SERVERID | RACT_REQIPADDR,
                  TIMER_RESTART|TIMER_Ta, CS_REQUESTING },
/* BADOFFER */  { 0, RACT_DECLINE, 0, 0 },
/* ACK      */  { 0, 0, 0, 0 },
/* BADACK   */  { 0, 0, 0, 0 },
/* NAK      */  { 0, 0, 0, 0 },
/* T1       */  { 0, 0, 0, 0 },
/* T2       */  { 0, 0, 0, 0 },
/* T3       */  { 0, 0, 0, 0 },
/* Ta       */  { LACT_DELAY, RACT_DISCOVER | RACT_BCASTFLAG,
                  TIMER_START | TIMER_Ta, 0 },
/* TaLIMIT  */  { 0, 0, 0, 0 },
/* Tb       */  { 0, 0, 0, 0 },
/* TbLIMIT  */  { 0, 0, 0, 0 },
/* GRACE    */  { 0, 0, 0, 0 },
/* RELEASE  */  { 0, 0, 0, 0 }
```

Figure 8: Transitions from CS_SELECTING

The structure of the client is similar on all platforms, except that the OS/2 implementation uses the "ifconfig" command to put some of the configuration options into effect.

In our implementation, the client assigns each packet an random number, "xid", which identifies the packet. If the client receives a delayed or duplicated packet with a different "xid" than expected, the client simply discards that packet.

4.3 DHCP Relay

The DHCP relay is almost the same as a bootp relay, and DHCP packets are specifically designed to be as compatible as possible [1, 6]. The DHCP relay is a little different than a bootp relay because it understands DHCP packets and unicasts them if necessary. Packets from DHCP clients are marked as BOOTREQUESTs, and packets from the DHCP servers to their clients are marked as BOOTREPLYs, just as with bootp relays.

We have implemented the DHCP relay on AIX and OS/2 platforms. The structure is similar to the server; the relay daemon waits at a port and processes the request when a packet comes in. Since the relay is implemented from the server code, it has the same difficulty in determining the originating network of client

DHCP_DISCOVER requests. The result, in the case of the DHCP relay, is that it will be unable to faithfully identify which interface to record in the "giaddr" field of the packet to be sent to the DHCP server, and ultimately unable to identify from which interface to issue the corresponding DHCP_REQUEST message. Without good information in the "giaddr" field, the DHCP server cannot offer appropriate Internet addresses to its clients. Again, we handle this problem in the short term by limiting DHCP relays to operation on a single network. This is a much more severe limitation, however, because the expected location for a DHCP relay will be on a router connecting two different networks. In the particular case of a DHCP relay co-located with a foreign agent, we can still correct operation by assuming that the mobile DHCP clients send requests from the wireless interface.

The differences between implementations of the DHCP relay on the dissimilar operating systems are just as described above for the DHCP server, and will not be repeated here.

5 Testing

We perform our testing mainly by running all three modules together to see how they comply with DHCP Standard. Configuration options for the server are located in a config file which is processed by the daemon when it is launched. Administering these configuration files is likely to be a recurring headache for any enterprise attempting the use of DHCP, but a headache that is gladly suffered considering the alternative of administering the multiplicity of individual Internet hosts that would otherwise be required.

For DHCP client, configuration options are to be avoided. Anything necessary to be sent to the server could be put in a standardized config file, as long as typical users can remain ignorant of the config file and still get work done. However, for testing purposes it is important that the DHCP program be able to configure its operation by use of such configuration files.

The DHCP specification defines a wealth of features. At the time of this writing there are 68 predefined options (configuration items) that clients can ask for, each of which should be tested. The server implementation must be tested for correct operation in many different cases, including:

- client Internet address preferences
- renewals
- implicit rejection of DHCP_OFFER
- timing out non-renewed resources

- requests for already allocated resources

In addition, although it is not required by the protocol, for efficient operation most DHCP servers should avoid allocating resources (and in particular Internet addresses) which have only recently been deallocated. Those resources are the ones most likely to be specifically requested again by clients.

The client implementation is likely to be guided by the state machine detailed above in section 4.2, as found in the DHCP specification [5]. The client can occupy 8 distinct states, and in our implementation there are 15 separate events possible. Thus, even though not all events are legal in each client state, there are 120 state transitions that should be tested for correct operation or error recovery. The precise actions taken on state transitions can also depend on certain global variables which must be taken in to account. Besides checking for agreement between actual operation and the requirements of the specification, other tests should be performed. The DHCP server should be tested for crash recovery. After a crash, the server should be able to recover most of its relevant state from non-volatile storage. It would be nice if the DHCP server could reboot without affecting the validity of leases on resources still in use by its clients. Indeed, some of the leases may have expired during crash recovery (especially if there was a hardware problem). In that situation, the DHCP server will have to perform various kinds of cleanup actions before offering its services again.

Mobility is mainly supported, in the scenarios we have described, by the more frequent allocation and deallocation of Internet addresses. Thus, careful testing of requests for new addresses, releasing old addresses, and the arrival of those requests in various orders should be required. Some wireless media are less reliable than the wired networks we have become accustomed to, so conditions resulting from dropped packets must be constructed and tested for reasonable operation. DHCP uses the UDP transport protocol, so there is no guarantee of delivery; the client and server implementations must take explicit actions to handle packet loss. This is done by introducing explicit timeouts at various stages in the protocol. Usually, the foreign agents which serve DHCP mobile clients should also be DHCP relays. Thus they should be tested for correct operation with interleaved requests for DHCP service and registration of mobile computers with their home agents (possibly using the addresses and subnet mask information which they have relayed back to the mobile DHCP clients from the DHCP servers). If the DHCP server has been implemented to inform its mobile clients about the Internet address of their home agent, that feature must also be tested.

If there are multiple DHCP servers, clients can expect a higher level of service reliability. In other words, if a DHCP server goes down, the resources it controls could possibly be managed by cooperating peer servers. The ideal would be to allow a client to renew its leases with another server, and in particular to at least keep using its leased IP address even if the DHCP server which allocated the address goes down. Unfortunately, this is not possible unless all servers are synchronized or share some common knowledge base. This style of synchronized operation is already hinted at by careful reading of the specification, and it introduces several new testing requirements. In particular, the correct operation of the DHCP servers when they control overlapping ranges of Internet addresses must be ascertained. This involves the maintenance of what is essentially a distributed database, and the use of techniques developed for that branch of database management.

For servers that are not multi-threaded, special care must be given to testing that the client allows sufficient delay before reissuing initial DHCP_DISCOVER messages. The idea is to make sure that client requests are usually serviced with the loads which are expected to be handled by the servers. One strategy is to add some feedback to the delay factor. For example, if a client doesn't receive an answer after 1 second, the next time it should wait 2 seconds before issuing another request.

Testing the DHCP relay is mostly just a matter of making sure that it doesn't interfere with the client/server relationship of the other programs involved. A good relay must be configurable to handle several DHCP servers. However, the relay must not try to perform any load balancing between the servers. If it did, then client requests for particular Internet addresses and renewals would be less likely to arrive at the correct server. If the DHCP relay is purchased externally, it should be tested to find out how it handles messages to clients in response to DHCP_DISCOVER requests.

6 Administering DHCP

Unfortunately, administering DHCP to handle populations of hundreds or thousands of Internet hosts is likely to be a challenging and frustrating process. Maintaining the server configuration files and enforcement of consistency on the administered networks will be difficult. A big improvement would be to specify groups or classes of machines, according to some criteria, and specify each client's group or class id. Then most changes which affect a broad class of clients (for instance, all those of a particular model) could be made by modifying the information in only one place. Of course configuration data for individual clients within a class should still be supported. Perhaps some class inheritance mechanism could be defined, or more simply a mechanism to insert other predefined class definitions.

Similarly, it will be important for DHCP servers to maintain global configuration variables. Instead of modifying the server's configured response to each possible client's request for nameservers, it must be possible for the DHCP server configuration file to specify the nameservers for all clients simultaneously, or at least all clients of a given class or group. Indeed, in many instances networks at several sites will be administered by a central authority. In that case, a single DHCP server may have to maintain separate lists of nameservers; a different list may be appropriate for separate groups of DHCP clients, which operate at the separate sites. Similar considerations apply for the configured server response to most other DHCP options.

Administering the pools of allocatable Internet addresses may be the stickiest problem facing administrators. The simplest solution, but the most wasteful, will be to have separate pools of addresses for each DHCP server. Even in that case, there should be multiple servers with some addresses available for each subnet within the collection of networks that are being administered. Clients on a subnet with with only one DHCP server available are vulnerable to server crashes; this is an especially important consideration until DHCP technology has been around for a long time, and DHCP servers are well known commodity items.

To avoid fragmentation of the pools of available Internet addresses, will require even more careful administration and testing of currently unavailable DHCP server features for handling the distributed resource databases mentioned previously. There should also be careful attention paid to ensuring that Internet addresses within the range to be allocated by the DHCP servers are never used by statically configured Internet hosts. Finally, having available the facilities to perform the test procedures mentioned in section 5 would go a long way towards simplifying any required troubleshooting, which inevitably is required.

Since DHCP servers and bootp servers are functionally similar, it is possible to operate DHCP servers as bootp servers also. However, this has to be specifically enabled by choice of the system administrator. By default, DHCP servers will not satisfy requests from bootp clients.

Whenever the DNS system becomes configured to accept dynamic updates from DHCP entities, so that known domain names are to be associated with newly allocated IP addresses, very tight testing requirement should be instituted. Since the DNS is at the heart of the enterprise networking services, any disruption will likely become a severe denial of service, and customers who have no other relationship with the DHCP services will experience failed network connections and other inconveniences. Our experience indicates that any activities which adversely affect nameservers will quickly cause dozens or hundreds of telephone calls to be placed to the local network administration, placing an already overburdened organization under additional and extremely unwelcome strain. Thus, we strongly recommend thorough testing of the dynamic update features to DNS under all possible scenarios of addition, modification, deletion, dropped acknowledgement packets, and any other relevant protocol details, once the update mechanism is finally standardized.

7 Adherence to the DHCP standard

In this section, we will discuss differences between our implementation and the DHCP specification. Afterwards, we will mention one or two ways in which our experience indicates that the DHCP standard could be improved.

The biggest area of difference is that our implementation does not support the distributed maintenance of overlapping or shared Internet address ranges between DHCP servers. Thus, at least for our initial effort, mobile DHCP clients will be vulnerable to single failures of DHCP servers. This is an area which we hope to repair. Also, our mobile clients do not check whether any other Internet host is already using an address when it is received in the server's DHCP_OFFER response. This check could also be done by the DHCP server before allocating the address. When we can finally begin supporting the use of overlapping address pools with different DHCP servers, we hope to provide some administrative tools to check occasionally for the consistency of the resource databases, and provide automatic feedback to the network administrators.

Our initial implementation of the DHCP relay has only one choice about which network to receive DHCP packets from; therefore, we do not have any of the difficulties outlined above with determining from which network the DHCP_DISCOVER packets have originated. We have code which eliminates this limitation, and are evaluating which of several strategies we will pursue to overcome the problem described in section 4.3, of determining the right value for the "giaddr" field by the relay.

When the client is trying to renew the lease, it always broadcasts. Why not unicast to the server or relay agent? This could possibly be a selectable client option instead of standard. Also, we feel that more attention should be paid to the specification of the relay agent. If a client and server resides on different subnets, and the relay goes down after the client is setup, the client won't be able to renew its lease next time even though both the server and client knows perfectly about each other.

8 Security of DHCP

There was no attempt in the design of DHCP to protect against malicious Internet hosts, and consequently the protocol is vulnerable to a variety of attacks. The basic protocol, then should not be in situations where those attacks would compromise the necessary integrity of the installation. In many organizations there is tight control over how machines are to be connected into the internal networks. However, this control is often only meaningful because of the physical nature of today's wired networks. In the case of wireless network links, it may be no longer so difficult for intruders to create surreptitious links to the local area networks within an installation. If, for instance, there is a radio-frequency point of attachment to the Internet available at one of an enterprise's work locations, an intruder might be able to establish a link to the enterprise network even without entering any of the enterprise's buildings. Perhaps the best means to combat such problems is to use data encryption at a level below the network (Internet) protocol level. Mobile and wireless computing will indeed put new emphasis on the need for securing enterprise data links.

Since the DHCP server doesn't do any authentication of client DHCPDISCOVER requests, any intruder can effectively impersonate the identity of any client which divulges its identification information. Likewise, an intruder can impersonate a DHCP server, and send erroneous information to any local DHCP client. For instance, suppose an intruder observes some DHCP transaction on a local network. Then at any later time the intruder can send a request to the same DHCP server to renew the request, even after the real client has turned off power and presumably ended all connections with its correspondents. A malicious local agent can effectively disrupt all traffic between real DHCP servers and clients by impersonating a DHCP server and handing out bad Internet addresses to prospective clients, or perhaps even bad operating system images! The extent of damage to prospective DHCP clients which can be caused by malicious hosts in the area, is only limited by the extent to which those clients rely on their DHCP-obtained configuration information.

In order to bring all local Internet address allocation to a halt, a bad host only has to request one address after another until the pool of available addresses is exhausted. Since the intruder can change apparent MAC addresses, or other identification, as often as it desires, the only protection against this denial of service attack is for the administered DHCP server to keep a strict accounting of which are the legal MAC addresses and only provide service to those addresses. This is a highly nontrivial administrative requirement, however, and to meet it would greatly diminish the convenience of using DHCP in reducing administrative workload. Any attempt by the administration to impose passwords or other authentication mechanisms will also have the effect of greatly reducing the convenience of the system.

The effect of these problems may be amplified if there is not a careful design of the way that DHCP allocated addresses can be associated with previously assigned names in the enterprise's Domain Name Service. If it is possible for an intruder to obtain a mapping to arbitrary domain names, then the effect of the DHCP insecurity will extend much further than the local network in which the intruder operates.

9 Conclusions and future work

We have ported and deployed DHCP on a variety of platforms in our labs, and have developed several ways to apply it as an administrative tool in support of portability and mobility for Internet hosts. In the process of this development, we have identified a number of basic problems regarding the interaction of dynamic Internet address allocation with the needs of host mobility. We discussed two basic modes of operation with DHCP for Internet hosts capable of frequent movement – a portability mode which allows for intermittent connections, and a mobility mode which comes closer to allowing seamless connectivity, by dovetailing with the operation of the IETF mobile-IP protocol.

In the future, we would like to test our mobile computers in more configurations with DHCP servers and clients separated by DHCP relays and mobile-IP foreign agents. We expect to measure the performance of DHCP servers and make sure it is sufficient for meeting the demands of a moderate population (several hundred) of DHCP mobile clients. We expect that certain performance improvements will be needed before this is universally practicable on all of our platforms. We look forward with some trepidation to testing out our system with Domain Name Servers that can accept dynamic updates to their stored associations between host names and Internet addresses. When standards are developed for the interchange of management of overlapping Internet address pools by multiple DHCP servers, we will upgrade our systems to use that important feature to improve our system reliability in case of server crashes.

We particularly would like to suggest that a great deal of effort needs to be invested into finding ways to limit the security exposure represented by the existing DHCP design and implementations. Until this is done, DHCP will most likely find no application in environments that may harbor untrustworthy individuals. Since a wireless infrastructure supporting mobile computers can affect so drastically the accessibility of enterprise resources at the physical and link layers, this is a critical issue for enabling the further acceptability and penetration of DHCP for the administration of Internet resources for the mobile workforce.

Acknowledgments

We gratefully acknowledge the assistance of John Chu and Sanjay Khanna, who provided us with the basic DHCP implementation we adapted for use with all for all our platforms. Tangirala Jagganadh has been indispensable during the later development of this work. Lastly, thanks to the reviewers of this paper, who provided useful comments for improvement.

References

[1] S. Alexander and R. Droms. DHCP Options and BOOTP Vendor Extensions. RFC 1533, October 1993.

[2] Ramon Caceres and Liviu Iftode. The Effects of Mobility on Reliable Transport Protocols. In *Proceedings of the 14th International Conference on Distributed Computing Systems*, June 1994.

[3] Douglas E. Comer. *Internetworking with TCP/IP*, volume 1. Prentice Hall, 1991.

[4] B. Croft and J. Gilmore. Bootstrap Protocol (BOOTP). RFC 951, September 1985.

[5] R. Droms. Dynamic Host Configuration Protocol. RFC 1541, October 1993.

[6] R. Droms. Interoperation Between DHCP and BOOTP. RFC 1534, October 1993.

[7] Donald E. Eastlake and Charles W. Kaufman. Domain Name System Protocol Security Extensions. draft-ietf-dnssec-secext-04.txt – work in progress, June 1995.

[8] IETF Mobile-IP Working Group. IPv4 Mobility Support. ietf-draft-mobileip-protocol-11.txt – work in progress, June 1995.

[9] John Ioannidis and Gerald Q. Maguire Jr. The Design and Implementation of a Mobile Internetworking Architecture. In *Proceedings of the Winter USENIX Conference*, pages 491–502, January 1993.

[10] Phil Karn and Craig Partridge. Improving Round Trip Time Estimates in Reliable Transport Protocols. In *Proceedings of ACM SIGCOMM '87*, 1987.

[11] P. Mockapetris. Domain Names - Concepts and Facilities. RFC 1034, November 1987.

[12] Charles Perkins, Andrew Myles, and David Johnson. IMHP: A Mobile Host Protocol for the Internet. In *Proceedings of INET'94/JENC5*, page 642, June 1994.

[13] Charles E. Perkins. DHCP Home Address option . draft-perkins-homeaddr-dhcpopt-00.txt – work in progress, March 1995.

[14] J. Postel. User Datagram Protocol. RFC 768, August 1980.

[15] J. Postel. Internet Protocol. RFC 791, September 1981.

[16] S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS). draft-ietf-dnsind-dynDNS-03.txt – work in progress, August 1995.