INTERNET DRAFT

M. Henry, Intel Corp., Editor
D. Koeppen, Bootix Tech. GmbH
E. Dittert, Intel Corp.
V. Viswanathan, Intel Corp.

Obsoletes:
<draft-viswanathan-remote-boot-protocol-01.txt>

Jun 24, 1999
Expires December, 1999

Intel Preboot Execution Environment
<draft-henry-remote-boot-protocol-00.txt>

Status of this Memo

Abstract

This memo describes the Intel PXE (Preboot Execution Environment) remote boot definition (which is part of Intel's Wired for Management initiative), and provides the rationale for proposing an extended remote boot capability beyond what is currently specified in DHCP.

Table of Contents

1. Rationale for an Extended Remote Boot Capability

Internet Protocol provides for a remote boot capability through use of DHCP [1]and TFTP [2]. DHCP provides the booting client with a "bootfile" name and the IP address of a TFTP server, from which the client downloads the bootfile. Depending on the capabilities of the DHCP service, the network administrator may program the DHCP service to determine the bootfile name to provide to the booting client based on information presented by the client during the DHCP cycle.

While this mechanism is simple and powerful, we believe it could usefully be extended. Specifically:

1. Extension of the canonical list of client information (presented to the DHCP service) to include client instruction set architecture, network interface type, etc., would be useful in many cases, as would a well defined means to extend this list. Certainly this information can be defined and included in the client on a case by case basis, and the DHCP service can be specifically configured to recognize and use the information. And certainly the format for transmitting this information from client to DHCP service is defined (option 60 and perhaps option 43). However the content is

ad hoc by definition, and in the case of option 60, the format of the content is ad hoc if the content is divided into fields.

2. It would be useful to have a standard mechanism to expand the number of bootfile sources presented to the client beyond the single TFTP server defined in the DHCP response. Going beyond this, it is desirable in some cases to provide the client with a list of available proprietary and commercial bootservers.

3. If the booting client is redirected through the "siaddr" field to a remote tftp service (bootserver), there is no standard method of providing a redundant service. (Certainly, if the tftp service to be used is on the DHCP server then presumably any DHCP redundancy mechanism would apply to the tftp service.) Related to #2 above, there is no method of providing redundancy across a pool of heterogeneous bootservers. As the number of clients affected can be quite large if the bootserver selected is unavailable, it is critical to define a redundancy mechanism.

4. When simultaneously booting many (e.g. 100s or 1000s) of identical clients it would be desirable to use a multicast TFTP. The current specifications are silent on the subject of how the client obtains the bootfile. However, the lowest common denominator assumed to be available by commercial IP boot ROMs is TFTP.

5. It would be useful for the booting client to have the means to verify the integrity and source of the bootfile.

It is the goal of PXE to address these limitations and define a remote boot capability robust enough to allow a heterogeneous set of clients to be selectively configured for remote boot via DHCP, and subsequently to be supported by a heterogeneous set of boot servers. Behaviorally, the goal is that a blank, unconfigured, compliant system can be removed from it's shipping carton, plugged into the network, and upon power on (with no other configuration) find an appropriate bootserver.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

3. PXE Remote Boot

The PXE specification defines a complete remote boot mechanism, including the DHCP configuration of the PXE client necessary for the PXE client to use three new protocols to complete the remote boot. These new protocols are described in this memo.

The three new protocols are: Remote Boot Configuration Protocol (RBCP), remote boot multicast TFTP (mTFTP), and Boot Integrity Services (BIS). RBCP is used by the PXE client to discover an appropriate boot service. The boot service provides the client with a bootfile name (the PXE client does not necessarily receive its bootfile name from the DHCP service), and it provides the client with any configuration necessary for the client to download the bootfile via either TFTP or mTFTP. The PXE client (optionally) uses BIS to authenticate the bootfile.

In addition, PXE defines a proxy DHCP specifically for PXE clients. The proxy is used as an optional means to provide the initial DHCP configuration of PXE clients in the absence of a DHCP service capable of interpreting and/or responding to PXE client specific configuration requests. By definition, the proxy will only respond to PXE clients. The proxy may reside on the same server as the DHCP service or may reside on a separate server. Details of proxy usage are beyond the scope of this document.

3.1. Use of DHCP

3.1.1 DHCPDISCOVER

The PXE client's DHCPDISCOVER packet MUST include:

- Client Machine Identifier - UUID (DHCP Option#61).
- PXE Client Class Identifier (DHCP option #60 - "PXEClient:Arch:xxxxx:UNDI:yyyzzz")
- Parameter Request List (DHCP Option #9). List MUST include at least subnet(1), router(3), vendor(43), class(60)

3.1.2 DHCPOFFER

The DHCPOFFER includes encapsulated PXEW client vendor options in Option $43 that provide:

- A ASCII list of available bootservers. The client MAY display this list to the user and let the user select the bootserver of their choice.
- A header prompt for the ASCII bootserver list that the client MUST display to the user if the bootserver list is displayed.
- A timeout value in seconds for user to request the bootserver list to be displayed. The client MUST wait for this timeout period before the default bootserver is discovered.
- A list of bootserver types and their IP addresses. (IP address are only useful if unicast discovery is enabled.)
- An option that specifies whether bootservers are to be discovered by a broadcast, multicast or a unicast discovery method.
- A multicast discovery address that the client MAY use to locate the bootserver if the multicast discovery option is enabled through the

previous option.

At this time, the PXE client may receive DHCPOFFER messages from several DHCP servers. However, the PXE client MUST give preference to offers whose replies contain option tag 60 ("PXEClient").

3.2. PXE Use of DHCP Options

3.2.1 PXE Class Identifier - Option 60

This option identifies PXE clients. This option also identifies the client system's architecture and the version of the Universal Network Driver Interface (UNDI) provided by the client.
The code for this option is 60 and its length is 32 octets long.
This option is of the form

"PXEClient:Arch:xxxxx:UNDI:yyyzzz", where
xxxxx = Client System Architecture (5 octets long, value 0 to 65535)
yyy = UNDI Major Version Number (3 octets long, value 0 to 255)
zzz = UNDI Minor Version Number (3 octets long, value 0 to 255)

Code	Len	32-octet String
60	32	PXEClient:Arch:xxxxx:UNDI:yyyzzz

3.2.2 PXE Client Identifier (UUID) - Option 61

This option uniquely identifies a client machine. A 16-byte universally unique identifier (UUID) is used for this purpose [6]. A server can uniquely identify a client using this UUID and provide customized service.
The code for this option is 61 and its length is 17, including the type identifier that appears before the UUID. PXE requires a type-identifier value of zero.

Code	Len	Type	16-octet Identifier				
61	17	254	o1	o2	. . . .		o16

3.2.3 PXE Vendor specific information - Option 43

The DHCP vendor specific information option is used by clients and servers to exchange vendor specific information. The information is an opaque object of n octets. The Encapsulated vendor-specific options field MUST be encoded as a sequence of code/length/value fields of identical syntax to the DHCP options field.

Code	Len	Vendor-specific information		
43	n	i1	i2	...

Within this vendor specific information, we have several code/length/value fields to specify the PXE related configuration information.

3.2.3.1 PXE_DISCOVERY_CONTROL

This option specifies the different methods by which a client can discover bootservers.

The code for this option is 6 and its length is 1.

Code	Len	PXE_DISCOVERY_CONTROL
6	1	b1

The individual bits in this octet (bit 0 is the least significant bit) are defined as follows:

Bit 0 - If set, broadcast discovery of servers is NOT allowed.
Bit 1 - If set, multicast discovery of servers is NOT allowed.
Bit 2 - If set, only use and/or accept replies from servers in the list defined by PXE_BOOT_SERVERS tag

If this tag is not supplied, all bits are assumed to be 0.

3.2.3.2 DISCOVERY_MCAST_ADDR

This option specifies the multicast IP address that is used by the client to discover bootservers. The client sends DHCPREQUEST packets to this multicast address.
The code for this option is 7 and its length is 4.

Code	Len	DISCOVERY_MCAST_ADDR			
7	4	m1	m2	m3	m4

3.2.3.3 PXE_BOOT_SERVERS

This option specifies a list of bootserver types and their IP addresses.

The code for this option is 8 and its length is variable. It is a list containing multiple entries of the following 3 elements.

- bootserver type (2 octets long)
- number of IP addresses (1 octet long) supporting the above type.
- IP addresses of those servers ([4 * number of IP addresses] octets long).

```
 Code   Len    BS type1  count    'n1' IP addresses
+-----+-----+-----+-----+-----+-----..-----+-----..----+---
|  8  |  v  |  t1 |  t2 |  n1 | IP addr 1  | IP addr 2 | . .
+-----+-----+-----+-----+-----+-----..-----+-----..----+---

   BS type2  count    'n2' IP addresses
+-----+-----+-----+-----..-----+-----..----+---
|  t1 |  t2 |  n2 | IP addr 1  | IP addr 2 | . .
+-----+-----+-----+-----..-----+-----..----+---
```

3.2.3.4 PXE_BOOT_MENU

This option specifies a string description for each bootserver type specified in option #8 above. The client MUST display this list to the user on request to provide the user with descriptions of the bootserver types.

The code for this option is 9 and its length is variable. It is a list containing multiple entries of the following 3 elements.

- bootserver type (2 octets long)
- length of the description string (1 octet long)
- string describing the bootserver type ('n' octets long).

```
Code   Len    BS type1 str-len      description
 +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----+
 |  9  |  v  |  t1 |  t2 |  n1 | n1 octets of bootserver desc.    |
 +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----+

   BS type2   str-len  description
 +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
 |  t1 |  t2 |  n2 | n2 octets of bootserver description . . |
 +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

3.2.3.5 PXE_MENU_PROMPT

This option specifies the prompt message that the client can display to the user before proceeding with the remote boot. This option controls how the information obtained through the tag PXE_BOOT_MENU is presented to the user and the duration for which it is displayed on the screen.

The code for this option is 10 and its length is variable.

```
 Code   Len  timeout Prompt String
+-----+-----+-----+-----+-----+-----+
| 10  |  v  |  t  |  prompt string .|
+-----+-----+-----+-----+-----+-----+
```

Upon user request before boot, the prompt string and the menu obtained through tag #8 "PXE_BOOT_MENU" are displayed. The prompt is followed by the number of seconds remaining before the first item in the "PXE_BOOT_MENU" is automatically selected. If this option #10 is not returned, the menu MUST be displayed without prompt and timeout. Similarly, if the timeout is 255, the menu and the prompt MUST be displayed indefinitely till there is a user interaction. If the timeout is zero, the client MUST immediately select the first item in the menu.

3.2.3.6 PXE_BOOT_ITEM

This option specifies the bootserver type that the client is discovering and also the "layer" number of the boot image that is requested. "Layer 0" always indicates the first bootfile for a particular bootserver type.

The code for this option is 71 and its length is 4.

Code	Len	BS Type		Layer #	
71	4	t1	t2	n1	n2

If the MSBit of "Layer #" is 0, then the containing message refers to the bootfile itself. If the MSBit of "Layer #" is 1, then the containing message refers to credentials for the bootfile. If the MSBit of "Layer #" is 1 and the containing message is a DHCPREQUEST, then the message MUST also include a PXE_CREDENTIAL_TYPES option.

3.2.3.7 PXE_CREDENTIAL_TYPES

This option specifies the types of credentials acceptable to the client for authenticating a bootfile.

The code for this option is 12 and its length is variable.

Code	Len	Credential Types	
12	v	t1(4)	. . .

Each credential type is a 4-byte code that specifies a public key that the client trusts for the authentication of bootfiles. (This also indicates, implicitly, the signature verification algorithms implemented by the client.) The exact formulation of these codes and the format of the credential files is specified in detail in [7].

3.3. Remote Boot Configuration Protocol (RBCP)

To use RBCP, the client must have an IP address and have been configured to use RBCP by the DHCP service as defined in the previous section. In this case the client will know the type of one or more bootservers. The client discovers one of these bootservers by sending a DHCPREQUEST packet to the bootserver using either unicast, multicast, or broadcast per the discovery instructions in Option #43, Tag #6 (PXE_DISCOVERY_CONTROL). This packet MUST also include options 61 and 60.

3.3.1 Bootserver Discovery

The client displays the menu prompt that it received from the previous step and lets the user pick a bootserver type. Once the user makes a selection, the client tries to locate a bootserver of that type. This is done by sending a DHCPREQUEST packet with the same information as in along with the type of the bootserver it is trying to discover. The discovery method by which this is done is determined by the option received through the previous step. That option could specify a multicast discovery, in which case, the client sends a multicast DHCP REQUEST packet to port 4011. If there is no response to the multicast discovery, then the client tries to broadcast the same request to the DHCP server port (port 67). If there is no response to this request as well, then the client tries to unicast the request to port 4011 using the list of IP addresses (one after another) obtained through step 4. The DHCPREQUEST packet that the client sends out contains the following information.

- Client UUID (DHCP option #61)
- PXE Client Class Identifier Tag (DHCP option #60)
- PXE_BOOT_ITEM tag embedded in vendor specific information option (DHCP option #43) specifying layer 0, not credentials

3.3.2 Bootserver Response

Bootservers receiving the DHCPREQUEST packet from the client MUST respond if they are a bootserver of the type requested in the PXE_BOOT_ITEM tag and they have bootfiles for the client system architecture defined in the Option 60. If the above conditions are satisfied, the bootserver responds with a DHCPACK packet containing the following information.

- PXE Client Class Identifier (DHCP option #60 - "PXEClient")
- Bootfile name specified in the fixed DHCP header bootfile portion
- Several mTFTP related options embedded in vendor specific information option #43. These are described more in detail in the internet draft "Multicast TFTP in the Intel PXE Remote Boot Environment"

- PXE_BOOT_ITEM tag embedded in vendor specific information option (DHCP option #43)

The specific encapsulated vendor specific options provided by the boot service are:

3.3.2.1 MTFTP_MULTICAST_IP_ADDRESS

This option specifies the multicast IP address that is used by the client to listen for and receive the multicast packets transmitted by the server.

The code for this option is 1 and its length is 4.

```
  Code   Len  MTFTP_MULTICAST_IP_ADDRESS
   +-----+-----+-----+-----+-----+-----+
   |  1  |  4  |  m1 |  m2 |  m3 |  m4 |
   +-----+-----+-----+-----+-----+-----+
```

3.3.2.2 MTFTP_CLIENT_UDP_PORT

This option specifies the UDP port that the client uses for the mTFTP transfers.

The code for this option is 2 and its length is 2.

```
    Code   Len  MTFTP_CLIENT_UDP_PORT
   +-----+-----+-----+-----+
   |  2  |  2  |  c1 |  c2 |
   +-----+-----+-----+-----+
```

3.3.2.3 MTFTP_SERVER_UDP_PORT

This option specifies the UDP port that the server uses for the mTFTP transfers. The client MUST send its mTFTP requests to this port on the server.

The code for this option is 3 and its length is 2.

```
    Code   Len  MTFTP_SERVER_UDP_PORT
   +-----+-----+-----+-----+
   |  3  |  2  |  s1 |  s2 |
   +-----+-----+-----+-----+
```

3.3.2.4 MTFTP_TRANSMISSION_START_DELAY

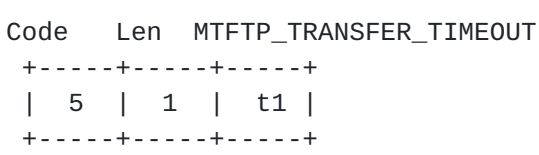
This options specifies, in seconds, the amount of time a client should wait before initiating a mTFTP transfer.

The code for this option is 4 and its length is 1.

```
    Code   Len  MTFTP_TRANSMISSION_START_DELAY
   +-----+-----+-----+
   |  4  |  1  |  t1 |
   +-----+-----+-----+
```

3.3.2.5 MTFTP_TRANSFER_TIMEOUT

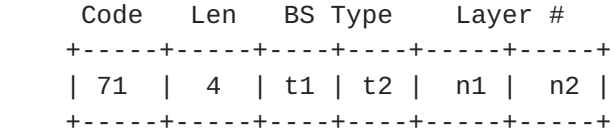
This options specifies, in seconds, the amount of time a client should wait before re-sending a MTFP request.
The code for this option is 5 and its length is 1.



3.3.2.6 PXE_BOOT_ITEM

This option specifies the bootserver type that the client is discovering and also the "layer" number of the boot image that is requested. "Layer 0" always indicates the first bootfile for a particular bootserver type.

The code for this option is 71 and its length is 4.



If the MSBit of "Layer #" is 0, then the containing message refers to the bootfile itself. If the MSBit of "Layer #" is 1, then the containing message refers to credentials for the bootfile. If the MSBit of "Layer #" is 1 and the containing message is a DHCPREQUEST, then the message MUST also include a PXE_CREDENTIAL_TYPES option.

3.4. Remote Boot Multicast TFTP (mTFTP)

This protocol is exactly the same as the standard TFTP protocol [2] but for a simple difference. All the TFTP responses from the server MUST be directed to the multicast IP address (MTFTP_MULTICAST_IP_ADDRESS) as the destination IP address. This enables multiple clients to listen to and receive the same packet that is transmitted by the server.

Three things happen for a successful mTFTP transfer:

1. The bootserver acquires a block of multicast IP addresses that it allocates to booting clients. (How the bootserver is allocated blocks of multicast addresses is beyond the scope of this document. Presumably the bootserver will use MADCAP [8]for this purpose.)
2. The bootserver configures the client to use the multicast IP address to download the client bootfile. (This configuration was covered in 3.3.2 above.)
3. The client initiates a mTFTP transfer or joins one already in progress. To do this, the client uses the protocol defined in the

next section.

3.4.1 Multicast Trivial File Transfer Protocol Details

A client goes through the following 3 stages during a mTFTP transfer:

- mTFTP Open
- mTFTP Receive
- mTFTP Close

3.4.1.1 mTFTP Open

Any client wishing to download a file through mTFTP first determines whether there is a multicast TFTP transfer already in progress for the file that it wants to download. (The mTFTP server MUST use a unique multicast address for each of the files that it can support in a mTFTP transfer). The client, after binding to the port MTFTP_CLIENT_UDP_PORT, waits for MTFTP_TRANSMISSION_START_DELAY seconds to see whether there are any packets addressed to the MTFTP_MULTICAST_IP_ADDRESS. Depending on whether there is a response or not, the client follows different steps as explained in the next 2 sections.

3.4.1.1.1. Multicast transfer already in progress

If the client receives a response packet within the MTFTP_TRANSMISSION_START_DELAY period, then a multicast transfer is already in progress. The client starts receiving the packets and stores them in an internal list (or uses some other mechanism to keep track of the received packets). As this client is not the one to initiate the original transfer, it MUST not send a TFTP ACK packet to any of the received packets. When the file transfer finishes (this happens when the TFTP server sends the last block of data of the file), the client would only have received the ending portion of the file. All the beginning blocks of the file were missed by this client, when it joined a multicast transfer which was in progress already. We will discuss how to receive the rest of the file in the section under "mTFTP Receive".

3.4.1.1.2. No multicast transfer in progress

If the client does not receive any packets during the initial MTFTP_TRANSMISSION_START_DELAY period, then it assumes that there is no multicast transfer in progress at that time. At the end of this period, the client requests the server to start a new transfer. This is done by the client sending a regular TFTP open packet (opcode set to 1). However, this packet is sent to the server's MTFTP_SERVER_UDP_PORT so that the server knows that it is a multicast TFTP request versus a standard TFTP request.

3.4.1.1.3. Server response to a MTFTP_OPEN request

When the server receives a MTFTP_OPEN request on its MTFTP_SERVER_UDP_PORT, it checks to make sure that a transfer is not already in progress. If there is a transfer already in progress for the requested file, then the server ignores this request. The client soon starts to receive some block of the file transfer that is in progress on its MTFTP_CLIENT_UDP_PORT. However, if there is no transfer in progress, then the server unicasts a response back to the client for the first data packet and also follows that by multicasting the same packet so that all other interested clients can also receive that.

3.4.1.2 mTFTP Receive

As mentioned in the previous section, the first packet of a multicast transfer MUST be sent both as unicast and multicast UDP packets. Subsequent packets are only multicast. The recipient of the first unicast packet becomes the master client which acknowledges each received packet. The master client (i.e. the acknowledging client) MUST acknowledge all packets even if that client has received the entire file. A server MUST transmit the complete file. Therefore, clients that start listening to a transfer part way through can wait and then get the rest of the file on the next mTFTP transfer to make up for what was missed during the first transmission.

3.4.1.3 mTFTP Close

A mTFTP transfer is finished when the acknowledging client has received all packets and disconnects. Clients who joined the transfer part way through the transfer can initiate a new transfer if one has not already started. If there are multiple clients who joined part way through, then there is an algorithm to minimize the number of clients simultaneously trying to initiate a new transfer. Before a new transfer is started, there is a calculated delay. An algorithm based on the number of packets received modifies the default delay of MTFTP_TRANSMISSION_START_DELAY seconds. Clients who received fewer packets (because of the faster transfer rate of the server) wait for a shorter time than those who received more. This algorithm ensures that

- Slower clients define the transmission speed as they are more likely to become the acknowledging clients.
- Clients with a large number of received packets may disconnect from the transfer after they received all missing packets as they are less likely to become acknowledging clients.

3.5. Boot Integrity Services (BIS)

3.5.1 NBP Authentication Request

At this point the client MAY contact the bootserver again to obtain authentication information for the bootfile. The protocol steps to do this are very similar to the steps for obtaining the bootfile. In particular, the client sends (unicast) a request to port 4011 on the bootserver from which the bootfile was obtained. The DHCPREQUEST packet that the client sends out contains the following information.

- Client UUID (DHCP option #61)
- PXE Client Class Identifier Tag (DHCP option #60)
- PXE_BOOT_ITEM tag embedded in vendor specific information option (DHCP option #43) specifying layer 0, credentials
- PXE_CREDENTIAL_TYPES tag embedded in vendor specific information option (DHCP option #43) specifying the type(s) of credentials accepted by the client.

3.5.2 NBP Authentication Response

Bootservers receiving the packet described in MUST respond if they have a credentials file of the type specified in the PXE_CREDENTIALS_TYPE tag for a boot image file corresponding to the bootserver type specified in the PXE_BOOT_ITEM tag and the client system architecture defined in the class identifier tag. If the above conditions are satisfied, the bootserver responds with a DHCPACK packet containing the following information.

- PXE Client Class Identifier (DHCP option #60 - "PXEClient")
- Several mTFTP related options embedded in vendor specific information option #43. These are described more in detail in the internet draft "Multicast TFTP in the Intel PXE Remote Boot Environment" [work in progress]
- PXE_BOOT_ITEM tag embedded in vendor specific information option (DHCP option #43)
- Credentials file name specified in the fixed DHCP header bootfile portion

3.5.3 NBP Credentials Delivery to Booting Client

The client, on receiving the DHCPACK packet from the bootserver, contacts the (m)TFTP service on the bootserver which sent the reply and downloads the credentials file. If the Multicast TFTP option is chosen by the client, then the client follows the guidelines provided in the internet draft "Multicast TFTP in the Intel PXE Remote Boot Environment"

4. Security

This memo defines methods for bootfile authentication (covered in section 3.5). Otherwise the protocols and usage of the protocols in this memo are not secure. However, as PXE is based on DHCP, methods being devised to protect DHCP [9] should generally apply to PXE.

5. References:

[1] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.
[2] Sollins, K., "The TFTP Protocol (Revision 2)", RFC 783, June 1981.
[3] Preboot Execution Environment (PXE) Specification Version 2.0 ftp://download.intel.com/ial/wfm/pxespec.pdf
[4] Intel's Wired for Management Baseline v2.0 Specification, ftp://download.intel.com/ial/wfm/base20.pdf
[5] Alexander, S., and R. Droms, "DHCP Options and BOOTP Vendor Extension", RFC 2132, March 1997.
[6] CAE Specification DCE 1.1: Remote Procedure Call Document Number: C706 Universal Unique Identifier Appendix Copyright (c) 1997 The Open Group http://www.opengroup.org/onlinepubs/9629399/toc.htm
[7] Boot Integrity Services Application Programming Interface Version 1.0 ftp://download.intel.com/ial/wfm/bisspec.pdf
[8] S. Hanna, et al, "Multicast Address Dynamic Client Allocation Protocol (MADCAP)" (Work in Progress)
[9] R. Droms, W. Arbaugh, "Authentication for DHCP Messages" (Work in Progress)

6. Authors' Addresses

Michael Henry
Intel Corporation, MS JF3-206
5200 NE Elam Young Pkwy
Hillsboro, OR 97124
Phone: (503) 264-9689
Email: mike.henry@intel.com

Eric Dittert
Intel Corporation, MS JF3-206
5200 NE Elam Young Pkwy
Hillsboro, OR 97124
Phone: (503) 264-8561
Email: eric.dittert@intel.com

Dirk Koeppen
Bootix Technology GmbH (formerly InCom)
Geranienstr. 19
D-41466 Neuss
Germany
Phone: (011) 49 69 89 3000
Email: dirk@incom.de

Vish Viswanathan
Intel Corporation, MS JF3-303
5200 NE Elam Young Pkwy
Hillsboro, OR 97124
Phone: (503) 264-9693
Email: vish.viswanathan@intel.com