Open Firmware

Recommended Practice:

TFTP Booting Extension

Version 1.0

Published by the Open Firmware Working Group

Revision History

changes from version 0.6:
- as per resolution of **#379** , removed section 5.
- as per **#398** , add **`ping`** parameters
- as per **#424** , define `"bootp-response"` property; added reference [8].

changes from version 0.7:
- change version to 1.0
- remove change bars & unapproved draft indications

1. Introduction

The use of networks during booting becoming more common. Open Firmware does claim to support such network access. However, the base Open Firmware document ([1]) is very terse in its description of the **obp-tftp** support package; it only mentions the use of TFTP ([2]) and does not even explicitly reference BOOTP ([3]). Platform bindings have added requirements for BOOTP, but have not explicitly documented details of the expected support of the **obp-tftp** package or of the required methods of "network" devices beyond the **open**, **close** and **load** methods.

The specifications contained herein are based upon RFCs (Request For Comments) published by the Internet Engineering Task Force and existing practice in Open Firmware implementations.

1.1. Purpose

This document adds specific requirements for the Open Firmware **obp-tftp** support package, the Open Firmware user interface and drivers for "network" devices.

1.2. Scope

The specifications for the **obp-tftp** package and user interface contained within this document are recommended for all Open Firmware implementations. The requirements on "network" device drivers are recommended for all "network" devices, including plug-in cards.

2. References and Definitions

2.1. References

[1] *IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices*, published by IEEE.

[2] *RFC 1350, TFTP Revision 2*, published by IETF.

[3] *RFC 951, Bootstrap Protocol (BOOTP)*, published by IETF.

[4] *RFC 768, User Datagram Protocol*, published by IETF.

[5] *RFC 1542, Clarifications and Extensions for the Bootstrap Protocol*, published by IETF.

[6] *RFC 826, An Ethernet Address Resolution Protocol*, published by IETF.

[7] *RFC 792, Internet Control Message Protocol*, published by IETF.

[8] *RFC 791, Internet Protocol*, published by IETF.

2.2. Definitions

3. "network" booting

The base document ([1]) describes the use of TFTP ([2]) for booting over a network. However, it is often the case that TFTP can not be used directly because the booting system does not necessarily “know” the proper file to be **load**ed or event an appropriate server from which to request the file.

BOOTP ([3]) is an IETF standard that defines a mechanism by which a booting device that does not know “anything” can locate a server, determine its proper IP address and identify the appropriate file that can be **load**ed.

BOOTP allows for a range of system knowledge to be used in the process. At one extreme, the booting device knows nothing except for its network hardware address (e.g., its 6-byte EtherNet address). At the other end of the spectrum, the booting device can know the server’s IP address, its own IP address, any gateway necessary, etc. The **open** method arguments described below allow the user (via command-line and/or **boot-device** arguments) to specify the various pieces of knowledge.

3.1. BOOTP overview

The BOOTP mechanism relies on the transmission of UDP ([4]) packets using well-known ports; port 67 is used as the “server’ port and port 68 is used as the “client” (i.e., booting system’s) port.

This document will refer to several fields within BOOTP’s packet format. These fields are:

op	1=BOOTREQUEST, 2= BOOTREPLY
ciaddr	client’s IP address (the booting system)
yiaddr	’your’ (i.e., client) IP address (filled in by server)
siaddr	server’s IP address
giaddr	’gateway’ IP address
chaddr	client’s hardware address
file	boot file name

Note: this document does not go into complete details of the BOOTP or TFTP protocols. For complete details, refer to [3], [5] and [2].

3.2. Example of "network" booting

A common case of booting via "network" device is one in which the booting system knows nothing about the booting environment, including its own IP address. In this case, the client creates a BOOTREQUEST packet with zeros in the ciaddr, yiaddr, siaddr, giaddr and file fields (thus, indicating that it doesn’t know what they should be) and fills in the chaddr field with the client’s hardware address that is assumed to be known. The siaddr field is filled with 255.255.255.255 that indicates a “broadcast” IP address; i.e., all potential servers should look at the packet.

The client then broadcasts (i.e., using a network-specific broadcast hardware address) the BOOTREQUEST packet. Potential servers examine the packet to determine if they should respond to the request. This is typically done by consulting a table, indexed by the chaddr field.

A server that determines that it should respond does so by creating a BOOTREPLY packet filling in the yiaddr field with the proper IP address of the client and the file field with the filename of the “boot-image” appropriate for the system and sending the packet to the client.

The client examines the BOOTREPLY packet and uses the yiaddr field as its IP address for subsequent transmissions and the siaddr field as the IP address of its server. It also uses the file field of the reply packet as the filename to be used for the TFTP request that it will generate.

At this point, the client may know all the information necessary to attempt to load the boot-image file. The one piece of information that is not explicitly handled by BOOTP is obtaining the hardware address of the server. If the client can obtain the server’s hardware address (HA) by access to the hardware header of the BOOTREPLY packet, then it can use that as the server's HA.

However, it may be necessary to use ARP ([6]) to map the IP address of the server to its HA.

Once it has obtained all of the relevant information about itself and its server, the client loads the boot-image. It starts this process by sending a TFTP read request (RRQ) packet, using the filename obtained from the BOOTREPLY packet. This RRQ packet is sent to the server at its proper HA and IP address obtained during the BOOTP process.

Assuming that the RRQ is successful, the server will reply with a TFTP DATA packet. The loading process then proceeds with a sequence of ACK replies sent by the client and DATA packets sent by the server. The process ends when the length field of a DATA packet is 0, indicating the entire file has been sent.

Since UDP is not a reliable protocol, and since a server may not be found, the actual BOOTP (and, subsequent TFTP loading) process contains time-out and retry provisions that were ignored above so that the basic data flow could be simply described.

3.3. BOOTP options

The above example is a typical case for booting a client. However, BOOTP allows various alternatives by allowing the client to fill in more details in its initial BOOTREQUEST packet. For example, the client may know what server it wishes to use (i.e., its IP number), or it may know what file it wants to load from any server that can supply it.

The description of the arguments for the **open** method of the **obp-tftp** package describes how these various alternatives can be specified.

4. Additional requirements for the OBP-TFTP support package.

This section describes methods that must be provided the **obp-tftp** support package. The **open** method is required by the base document ([1]), but details are missing from that description and new requirements for argument parsing are added by this document. The responsibilities of **open** versus **load** are not specified in the base document; they are specified here.

4.1. arguments to the obp-tftp open method

The **open** method of the **obp-tftp** support package *shall* support the following arguments:

[**bootp**,]siaddr,filename,ciaddr,giaddr,bootp-retries,tftp-retries

All of the arguments are optional. The external format of the siaddr, ciaddr and giaddr fields are Internet-standard "dotted-decimal" notation.

bootp is an optional constant (i.e., exactly the letters "bootp") that specifies the use of BOOTP as the "discovery" protocol to be used. If specified, the **obp-tftp** package shall use the BOOTP (as opposed, say, to RARP/ARP). If this argument is not specified, **obp-tftp** is not required to use BOOTP.

siaddr is the IP address of the intended server. If this field is specified, its decoded value is used as the siaddr field of the BOOTREQUEST packet (and subsequent TFTP packets). If this field is not specified, the value of 0.0.0.0 is used for the siaddr field of the BOOTREQUEST packet, and the server's IP address is determined by the siaddr of the BOOTREPLY packet.

filename is the filename of the file that is to be loaded by TFTP from the server. If specified, the `file` field of the `BOOTREQUEST` field is set from this field; if not specified, the `file` field is set to zero. Since this filename may be a "generic" name, the `file` field of the `BOOTREPLY` packet is used as the `Filename` field of the `RRQ` packet during the subsequent TFTP loading.

ciaddr is the `IP` address of the client (i.e., the system being booted). If this field is specified, its decoded value is used as the `ciaddr` field of the `BOOTREQUEST` packet (and, subsequent TFTP transactions); i.e., the `yiaddr` field of `BOOTREPLY` packets is ignored. If this field is not specified, the `ciaddr` field is set to `0.0.0.0` and the client's `IP` address is determined from the `yiaddr` field of the `BOOTREPLY` packet. (Note: this behavior is specified by [5].)

giaddr is the `IP` address of the `BOOTP` 'gateway'. (Note that this is not necessarily the same as the network gateway; see [3] and [5] for details.) If this field is specified, its decoded value is used as the value of the `giaddr` field of the BOOTREQUEST packet; if not specified, the `giaddr` field of the BOOTREQUEST packet is set to `0.0.0.0`.

bootp-retries is the maximum number of retries that are attempted before the `BOOTP` process is determined to have failed. I.e., if, after `bootp-retries` attempts at transmission of a `BOOTREQUEST` packet are made with no responding `BOOTREPLY` packets are received, the **obp-tftp** package shall report the failure of its **load** method call (which is what performs the `BOOTP` sequence) by executing a **throw** (with a non-zero value).

tftp-retries is the maximum number of retries that are attempted before the TFTP process is stopped. The retries count shall include time-outs and bad `DATA` packets. If the count is exceeded, the TFTP **load** method shall **abort**.

4.2. Standard methods for the obp-tftp package

The following methods shall be supported by the obp-tftp package.

open (-- okay?) S

Standard method to prepare the **obp-tftp** package for use by **load**.

The **open** method shall parse the input arguments (as described in section 4.1. arguments to the `obp-tftp open` method), checking for correctness. If any errors are detected in the arguments, **open** shall report failure by returning a value of **false**. If no errors are detected, a value of **true** shall be returned. **open** shall not cause any network traffic; i.e., it simply parses the arguments. Only a subsequent **load** will cause network activity.

close (--) S

Standard method to close the package.

load (addr -- len) S

Standard method to load a file from the network server.

If the parameters specified by the **open** call to this instance were insufficient to specify the server `IP` address and filename, a `BOOTP` sequence shall be initiated to acquire this information, using the `bootp-retries` as the maximum number of `BOOTP REQUESTs` to attempt before reporting failure.

If a `BOOTP` sequence is performed and succeeds, the `BOOTP` [3] `BOOTREPLY` packet shall be reported by creating a `"bootp-response"` property under the `"/chosen"` node. The contents of this property are encoded as with **`encode-bytes`**, and contain only the BOOTP [3] packet returned by the server, with the encapsulating UDP [4], IP [8] and link-layer header and trailers omitted.

> If the `BOOTP` is successful, or if the **`open`** call's arguments fully specified a filename, etc., a `TFTP RRQ` packet shall be sent to the designated server to start reading the file. If the `RRQ` does not succeed (e.g., a `ERROR` packet is received or `tftp-retries` is exhausted), **`load`** shall report failure.
>
> Upon successful execution of the `RRQ`, the file shall be read, using the `tftp-retries` value specified on the **`open`** call. If the file read fails (e.g., because of exhausting the `tftp-retries` value), **`load`** shall report failure. Upon successful reading of the file, the actual number of bytes read shall be reported as the result of **`load`**.
>
> Failure of **`load`** shall be reported by executing a **`throw`** with a non-zero value.

5. `ping` User Interface command

The following User Interface command should be implemented:

```
ping device-path:[device-args,]server-ip,
     [client-ip],[gateway-ip][,timeout]
```

where <u>`device-path`</u> is the path-name of the network device, <u>`device-args`</u> are additional arguments to the device, <u>`server-ip`</u> is the IP address of the intended target of the `ping`, <u>`client-ip`</u> is the IP address of this system, <u>`gateway-ip`</u> is the IP address of the gateway to be used, respectively, in network dotted decimal notation. The <u>`timeout`</u> value is the time (in decimal milliseconds) to wait for a response before reporting failure; the default is 10.

`ping` should print a response reporting success or failure of the ping attempt.