INTERNET-DRAFT
Obsoletes: 2090
Category: Experimental

G. Dion
iMagicTV Inc

May 2002

TFTP Multicast Option
draft-dion-tftp-multicast-option-01.txt

Introduction

Status of this Memo

Abstract

The Trivial File Transfer Protocol [1] is a simple, lock-step, file transfer protocol which allows a client to get or put a file onto a remote host.

This document describes a new TFTP option. This new option will allow the multiple clients to receive the same file concurrently through the use of Multicast packets. The TFTP Option Extension mechanism is described in [2].

Often when similar computers are booting remotely they will each download the same image file. By adding multicast into the TFTP option set, two or more computers can download a file concurrently, thus increasing network efficiency.

This document assumes that the reader is familiar with the terminology and notation of both [1] and [2].

Terminology and Definitions

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 [4].

Multicast Option Specification

The TFTP Read Request packet is modified to include the multicast option as follows:

```
+--------+----~~----+---+--~~--+---+-----------+---+---+
|  opc=1 | filename | 0 | mode | 0 | multicast | 0 | 0 |
+--------+----~~----+---+--~~--+---+-----------+---+---+
```

opc
The opcode field contains a 1, for Read Requests, as defined in [1].

filename
The name of the file to be read, as defined in [1]. This is a NULL-terminated field.

mode
The mode of the file transfer: "netascii", "octet", or "mail", as defined in [1]. This is a NULL-terminated field.

multicast
Request for multicast transmission of the file option, "multicast" (case insensitive). This is a NULL-terminated field. The value for this option request is a string of zero length.

If the server is willing to accept the multicast option, it sends an Option Acknowledgment (OACK) to the client including the multicast option, as defined in [2]. The OACK to the client will specify the multicast address and flag to indicate whether that client should send block acknowledgments (ACK).

```
+-------+-----------+---+-------~~-------+---+
|  opc  | multicast | 0 | addr, port, mc | 0 |
+-------+-----------+---+-------~~-------+---+
```

opc
The opcode field contains the number 6, for Option Acknowledgment, as defined in [2].

multicast
Acknowledges the multicast option. This is a NULL-terminated field.

addr

The addr field contains the multicast IP address. This field is terminated with a comma.

port

The port field contains the destination port of the multicast packets. The use of Registered Port number 1758 (tftp-mcast) is recommended. This field is terminated with a comma.

mc

This field will be either 0 or 1, to tell the client whether it is the master client, that is, it is responsible for sending ACKs to the server. This is NULL-terminated field.

Data Transfer

After the OACK is received by the client it will send an ACK for packet zero, as in [2]. With the multicast option being accepted this ACK will indicate to the server that the client wants the first packet. In other words the ACKs may now be seen as a request for the n+1th block of data. This enables each a client to request any block within the file that it may be missing.

To manage the data transfer the server will maintain a list of clients. Typically the oldest client on the list, from here on referred to as the Master Client, will be responsible for sending ACKs. When the master client is finished, the server will send another OACK to the next oldest client, telling it to start sending ACKs. Upon receipt of this OACK the new master client will send an ACK for the block immediately before the first block required to complete its download.

Any subsequent clients can start receiving blocks of a file during a transfer and then request any missing blocks when that client becomes the master client. When the current master client is finished, the server will notify the next client with an OACK making it the new master client. The new master client can start requesting missed packets. Each client must terminate the transfer by sending an acknowledgment of the last packet or by sending an error message to server. This termination can occur even if the client is not the master client.

Any subsequent OACKs to a client may have an empty multicast address and port fields, since this information will already be held by that client. In the event a client fails to respond in a timely manner to a OACK enabling it as the master client, the server shall select the next oldest client to be the master client. The server shall reattempt to send a OACK to the non- responding client when the new master client is finished. The server may cease communication with a client after a reasonable number of attempts.

Each transfer will be given a multicast address for use to distribute the data packets. Since there can be multiple servers on a given network or a limited number of addresses available to a given server, it is possible that their might be more than one transfer using the same multicast address. If it is found necessary to ensure that a client only accepts the correct packets, each transfer may use a unique port on the server. The port number will identify the data packets for the transfer. Thus, the server must send the unicast OACK packet to the client using the same port as will be used for sending the multicast data packets. The client "MAY" also authenticate the validity of the packet based on the source address of the multicast packet, but is not required to do so. This differs slightly from the original design in RFC2090 [5], which required the client to use both the source address and port number or the multicast packet for authentication. Doing so can create unnecessary network design issues when doing load-balancing.

At any point if a client, other than the master client, sends a ACK to the server, the server will respond with another OACK with the mc field holding a value of zero. If this client persists in sending erroneous ACKs, the server may send an error packet to the client, discontinuing the file transfer for that client.

The server may also send unicast packets to a lone client to reduce adverse effects on other machines. As it is possible that machines may be forced to process many extraneous multicast packets when attempting to receive a single multicast address.

Example

```
          clients                                      server  message
          ------------------------------------------------------------
 1  C1   |1|afile|0|octet|0|multicast|0|0| ->                    RRQ
 2                 C1 <- |6|multicast|224.100.100.100,1758,1|  OACK
 3  C1   |4|0| ->                                              ACK
 4                           M <- |3|1|1| 512 octets of data|  DATA
 5  C1   |4|1| ->                                              ACK
 6                           M <- |3|2|1| 512 octets of data|  DATA
 7  C2   |1|afile|0|octet|0|multicast|0|0| ->                    RRQ
 8                 C2 <- |6|multicast|224.100.100.100,1758,0|  OACK
 9  C2   |4|0| ->                                              ACK
10  C1   |4|2| ->                                              ACK
11                           M <- |3|3|1| 512 octets of data|  DATA
12  C3   |1|afile|0|octet|0|multicast|0|0| ->                    RRQ
13                 C3 <- |6|multicast|224.100.100.100,1758,0|  OACK
14  C1   |4|3| ->                                              ACK
15  C2   |4|0| ->                                              ACK
16               M (except C2) <- |3|4|1| 512 octets of data|  DATA
17  C1   |4|4| ->                                              ACK
18                           M <- |3|5|1| 512 octets of data|  DATA
```

```
   19  C1  |4|5| ->                                              ACK
   20                          M <- |3|6|1| 100 octets of data|  DATA
   21  C1  |4|6| ->                                              ACK
   22                                   C2 <- |6|multicast|,,1|  OACK
   23  C2  |4|0| ->                                              ACK
   24                          M <- |3|1|1| 512 octets of data|  DATA
   25  C2  |4|1| ->                                              ACK
   26                          M <- |3|2|1| 512 octets of data|  DATA
   27  C2  |4|3| ->                                              ACK
   28                          M <- |3|4|1| 512 octets of data|  DATA
   29  C2  |4|6| ->                                              ACK
   30                                   C3 <- |6|multicast|,,1|  OACK
   31  C3  |4|2| ->                                              ACK
   32                          M <- |3|3|1| 512 octets of data|  DATA
   33  C3  |4|6| ->                                              ACK


      Comments:
         1  request from client 1
         2  option acknowledgment
         3  acknowledgment for option acknowledgment,
            or request for first block of data
         4  first data packet sent to the multicast address
         7  request from client 2
         8  option acknowledgment to client 2,
            send no acknowledgments
         9  OACK acknowledgment from client 2
         15 OACK acknowledgment from client 3
         16 client 2 fails to receive a packet
         21 client 1 acknowledges receipt of the last block,
            telling the server it is done
         23 option acknowledgment to client 2,
            now the master client
         25 client 2 acknowledging with request for first block
         27 client 2 acknowledges with request for missed block
         29 client 2 signals it is finished
         31 client 3 is master client and asks for missing blocks
         33 client 3 signals it is finished
```

Conclusion

With the use of the multicast and blocksize[3] options TFTP will be capable of fast and efficient downloads of data. Using TFTP with the multicast option will maintain backward compatibility for both clients and servers.

Security Considerations

Security issues are not discussed in this memo.

References

[1] Sollins, K., "The TFTP Protocol (Revision 2)", STD 33, RFC 1350, MIT, July 1992.

[2] Malkin, G., and A. Harkin, "TFTP Option Extension", RFC 1782, Xylogics, Inc., Hewlett Packard Co., March 1995.

[3] Malkin, G., and A. Harkin, "TFTP Blocksize Option", RFC 1783, Xylogics, Inc., Hewlett Packard Co., March 1995.

[4] Bradner, S., "Keywords for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[5] RFC2090 TFTP Multicast Option. A. Emberson. February 1997. (Status: EXPERIMENTAL)

Author's Address

G. Dion
iMagicTV Inc.
1 Brunswick Square, 14th Floor
Saint-John, New-Brunswick
Canada E2L 3Y2

Phone: (506) 635-2301
EMail: gdion@imagictv.com