Abstract

The File Transfer Protocol has a very long history, and despite the fact that today, other options exist to perform file transfers, FTP is still in common use. As such, it is important that in the situation where some client computers are IPv6-only while many servers are still IPv4-only and IPv6-to-IPv4 translators are used to bridge that gap, FTP is made to work through these translators as best it can.

FTP has an active and a passive mode, both as original commands that are IPv4-specific, and as extended, IP version agnostic commands. The only FTP mode that works without changes through an IPv6-to-IPv4 translator is extended passive. However, many existing FTP servers don’t support this mode, and some clients don’t ask for it. This document describes server, client and middlebox (if any) behavior that minimizes this problem.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.
Table of Contents

1. Introduction ......................................................... 3
2. Terminology ......................................................... 4
3. Client recommendations ............................................. 4
4. ALG functionality ................................................... 5
   4.1. Control channel translation ................................. 6
   4.2. EPSV to PASV translation ................................. 7
   4.3. EPRT to PORT translation ................................. 8
      4.3.1. Stateless EPRT translation ....................... 8
      4.3.2. Stateful EPRT translation ....................... 8
   4.4. Default port 20 translation ............................... 9
   4.5. Both PORT and PASV ....................................... 9
   4.6. Timeouts .................................................... 10
5. IANA considerations ............................................... 10
6. Security considerations ........................................... 10
7. References ........................................................ 10
   Appendix A. Server implementation recommendations .......... 11
   Appendix B. Acknowledgements .................................. 12
   Appendix C. Document and discussion information ............. 12
   Author’s Address .................................................. 12
1. Introduction

[RFC0959] specifies two modes of operation for FTP: active mode, in which the server connects back to the client and passive mode, where the server opens a port for the client to connect to. Without additional action, active mode with a client-supplied port doesn’t work through NATs or firewalls. And in both cases, an IPv4 address is specified, making both the original passive and active modes incompatible with IPv6. These issues were solved in [RFC2428], which introduces the EPSV (extended passive) mode, where the server only responds with a port number, and the EPRT (extended port) command, which allows the client to supply either an IPv4 or an IPv6 address (and a port) to the server.

A survey done in April of 2009 of 25 randomly picked and/or well-known FTP sites reachable over IPv4 showed that only 12 of them supported EPSV over IPv4. Additionally, only 2 of those 12 indicated that they supported EPSV in response to the FEAT command ([RFC2389]) that asks the server to list its supported features. One supported EPSV but not FEAT. In 5 cases, issuing the EPSV command to the server led to a significant delay, in 3 cases followed by a control channel reset. All 25 servers were able to successfully complete a transfer in traditional passive PASV mode as required by [RFC1123]. More tests showed that the use of an address family argument with the EPSV command is widely mis- or unimplemented in servers. The additional tests with more servers showed that approximately 65% of FTP servers support EPSV successfully and around 96% support PASV successfully. Clients weren’t extensively tested, but previous experience from the author suggests that most clients support PASV, with the notable exception of the command line client included with Windows, which only supports active mode. It uses the original PORT command when running over IPv4 and EPRT when running over IPv6.

Considering the above, this document describes the following recommendations:

**Servers:**

* Allow EPSV (even for IPv4-only servers)
* Use a predictable address in the response to the PASV command

**Clients:**

* Use EPSV over IPv6 rather than EPRT
* Fall back to PASV if EPSV fails (even over IPv6)
* Don’t use certain modes and options that trigger server bugs

Additionally, there are guidelines for operators choosing to implement application layer gateway functionality to provide connectivity between unupdated servers and/or clients. Clients that want to engage in more complex behavior, such as server-to-server transfers, may make an FTP ALG go into transparent mode by issuing an AUTH command.

The recommendations in this document apply to all forms of IPv6-to-IPv4 translation, including stateless translation such as [RFC2765] or [I-D.ietf-behave-v6v4-xlate] as well as stateful translation such as [I-D.ietf-behave-v6v4-xlate-stateful].

The FTP protocol allows for complex interactions, such as the situation where a client connects to two servers and directs the servers to exchange data between them. No attempt is made to address these other than through making ALGs transparent after an AUTH command.

2. Terminology

Within the context of this document, the words "client" and "server" refer to FTP client and server implementations, respectively. An FTP server is understood to be an implementation of the FTP protocol running on a server system with a stable address, waiting for clients to connect and issue commands and start data transfers. Clients interact with servers using the FTP protocol, and store (upload) or retrieve (download) files to/from one or more servers, either interactively under control of a user, or as an unattended background process. Most operating systems provide a web browser that implements a basic FTP client, as well as a command line client. Third-party FTP clients are also widely available.

Other terminology is derived from the documents listed in the reference section.

3. Client recommendations

All FTP clients should support EPSV when communicating over IPv6 and always attempt to use EPSV mode unless explicitly configured to use EPRT.

It is highly recommended that FTP clients react by retrying with PASV when the EPSV command fails, either because of an error response by the server (40x, 42x, 50x and 52x responses), because the data...
connection couldn’t be created or because the control channel session was terminated. In the latter two cases, a client may cache the name or address of the FTP server and issue PASV rather than EPSV in future sessions. In that case, the cache entry should be cleared if older than 7 days and the server indicates EPSV support in its FEAT response where it previously did not indicate EPSV support in its FEAT response. There is always a risk that an error was the result of a condition unrelated to IPv6-to-IPv4 translation. However, retrying with a PASV request has little potential for harm, so unless the error is clearly unrelated, retrying with PASV is the appropriate reaction.

When after attempting to initiate EPSV and/or EPRT modes unsuccessfully and a client retries with PASV, the server will respond to the PASV command with an IPv4 address that the client must use to connect to for the data connection. Even if the client has IPv4 reachability, it should ignore the server-supplied address and set up a data connection towards the IPv6 address of the server that is used for the control channel session. However, the port number used for the data connection is taken from the 227 response to the PASV command.

The main rationale for ignoring the IPv4 address in the 227 response, even if the client has IPv4 connectivity, is the fact that most servers will only allow a data connection from the same client address as seen in the control channel connection, see <http://cr.yp.to/ftp/security.html>. Using IPv6 for the control channel and IPv4 for the data channel means that the source address will almost certainly be different in both cases, making it unlikely that the data connection can be established successfully.

Clients should refrain from using any arguments with the EPSV command. "EPSV 2" to request IPv6 will fail across an IPv6-to-IPv4 translator. Also, this command is often not handled properly by IPv6 servers. "EPSV ALL" indicates that the client will use EPSV for all transfers, but an ALG may translate EPSV commands to PASV commands, conflicting with the earlier "EPSV ALL".

4. ALG functionality

The use of FTP application layer gateways for compatibility with IPv6-to-IPv4 translators is rejected by many within the IETF community. As such, it is recommended to update FTP clients and servers as required for IPv6-to-IPv4 translation support where possible, to allow proper operation of the FTP protocol without the need for ALGs.
On the other hand, network operators often have little influence over the FTP clients their customers run, let alone the FTP servers used throughout the Internet. For those operators, deploying an ALG may be the only way to provide a satisfactory customer experience. So, even though not the preferred solution, this document describes the functionality of such an ALG in order to promote consistent behavior between ALGs in an effort to minimize their harmful effects. However, the situation with regard to FTP servers and -clients, especially in IPv6-heavy deployments, may change fast, so within relatively little time it may become feasible to stop running an ALG. Operators are encouraged to keep revisiting the issue.

Note that the translation of EPSV through all translators and EPRT through a stateless translator is relatively simple and translation of EPRT through a stateful translator relatively difficult. As such, an ALG used with a stateful translator may choose to support only EPSV. However, an ALG used with a stateless translator should also support EPRT.

### 4.1. Control channel translation

The IPv6-to-IPv4 FTP ALG intercepts all TCP sessions towards IPv4 port 21 destinations. The FTP ALG implements the Telnet protocol ([RFC0854]) used for control channel interactions to the degree necessary to interpret commands and responses and re-issue those commands and responses, modifying them as outlined below. Option negotiation attempts by either the client or the server, except for those allowed by [RFC1123], should be rejected by the FTP ALG without relaying those attempts. This avoids the situation where the client and the server negotiate options unknown to the FTP ALG.

There are two ways to implement the control channel ALG:

1. The ALG terminates the IPv6 TCP session, sets up a new IPv4 TCP session towards the IPv4 FTP server, and relays commands and responses back and forth between the two sessions.

2. Packets that are part of the control channel are translated individually.

In the second case, an implementation must have the ability to track and update TCP sequence numbers when translating packets and break up packets into smaller packets after translation, as the control channel translation may modify the length of the payload portion of the packets in question. Also, FTP commands/responses or Telnet negotiations may straddle packet boundaries, so in order to be able to perform the ALG function, it may be necessary to reconstitute Telnet negotiations and FTP commands and responses from multiple
If the client issues the AUTH command the client is attempting to negotiate [RFC2228] security mechanisms which are likely to be incompatible with the FTP ALG function. In this situation, the FTP ALG must switch to transparently forwarding all data on the control channel in both directions until the end of the control channel session. This requirement applies regardless of the response from the server. In other words, it is the fact that the client attempts the AUTH negotiation that requires the ALG to become transparent, not whether or not the attempt is successful.

There have been FTP ALGs for the purpose of making active FTP work through IPv4 NATs for a long time. Another type of ALG would be one that imposes restrictions required by security policies. Multiple ALGs can be implemented as a single entity. Should such a multi-purpose ALG forbid the use of the AUTH command for policy reasons, the side effect of making the ALG stop performing the translations described here, as well as other possible interventions related to IPv6-to-IPv4 translation, must be retained even if the ALG responds to the AUTH command with an error and doesn’t propagate the command to the server. (Implementers are further advised that unlike hosts behind an IPv4 NAT, IPv6 hosts using an IPv6-to-IPv4 translator will normally have the ability to execute FTP over IPv6 without interference from the ALG.)

4.2. EPSV to PASV translation

Although many IPv4 FTP servers support the EPSV command, some servers react adversely to this command, and there is no reliable way to detect in advance that this will happen. As such, an FTP ALG may translate all occurrences of the EPSV command issued by the client to the PASV command, and reformat a 227 response as a corresponding 229 response.

For instance, if the client issues EPSV (or EPSV 2 to indicate IPv6 as the network protocol), this is translated to the PASV command. If the server with address 192.0.2.31 then responds with:

```
227 Entering Passive Mode (192,0,2,31,237,19)
```

The FTP ALG reformats this as:

```
229 Entering Extended Passive Mode (|||60691|)
```

If the server’s 227 response contains an IPv4 address that doesn’t match the destination of the control channel, the FTP ALG should send the following response to the client:
425 Can’t open data connection.

It is important that the response is in the 4xx range to indicate a temporary condition.

If the client issues an EPSV command with a numeric argument other than 2, the ALG must not pass the command on to the server, but rather respond with a 522 error.

If the client issues EPSV ALL, the FTP ALG must not pass this command to the server, but respond with:

    202 Command not implemented.

This avoids the situation where an FTP server may react adversely to receiving a PASV command after the client indicated that it will only use EPSV during this session.

4.3. EPRT to PORT translation

Should the IPv6 client issue an EPRT command, the FTP ALG may translate this EPRT command to a PORT command. The translation is different depending on whether the translator is a stateless one-to-one translator or a stateful one-to-many translator.

4.3.1. Stateless EPRT translation

If the address specified in the EPRT command is the client’s IPv6 address, then the FTP ALG reformats the EPRT command into a PORT command with the IPv4 address that maps to the client’s IPv6 address. The port number must be preserved for compatibility with stateless translators.

If the address specified in the EPRT command is not the client’s IPv4 address, the ALG’s response is undefined. It may pass along the command unchanged, respond with an error, or attempt to perform an appropriate translation.

4.3.2. Stateful EPRT translation

If the address in the EPRT command is the IPv6 address of the control channel client’s address, the stateful translator selects an unused port number in combination with the IPv4 address used for the control channel towards the FTP server, and sets up a mapping from that transport address to the one specified by the client in the EPRT command. The PORT command with the IPv4 address and port used on the IPv4 side of the mapping is only issued towards the server once the mapping is created. Initially, the mapping is such that either any
transport address or the FTP server’s IPv4 address with any port number is accepted as a source, but once the three-way handshake is complete, the mapping is narrowed to only match the negotiated TCP session.

If the address in the EPRT command is not the client’s IPv6 address, the ALG’s response is undefined.

4.4. Default port 20 translation

If the client doesn’t issue an EPSV/PASV or EPRT/PORT command, it is invoking the default active FTP behavior where the server sets up a TCP session towards the client. In this situation, the source port number is the default FTP data port (port 20) and the destination port is the port the client uses as the source port in the control channel session.

In the case of a stateless translator, this does not pose any problems. In the case of a stateful translator, the translator should accept incoming connection requests from the server on the IPv4 side if the transport addresses match that of an existing FTP control channel session, with the exception that the control channel session uses port 21 and the new session port 20. In this case, a mapping is set up towards the same transport address on the IPv6 side that is used for the matching FTP control channel session.

So for instance, the client is 2001:db8:31::6 and the server is 192.0.2.4. The translator has prefix 2001:db8:ffff::/96 as its translator prefix and 10.0.0.1 as its IPv4 address. On the IPv6 side, the transport addresses for an FTP control channel session could then be 2001:db8:31::6,49152 to 2001:db8:ffff::c000:204,21 on the IPv6 side and 10.0.0.1,60000 to 192.0.2.4,21 on the IPv4 side. If then the FTP server initiates a session from 192.0.2.4,20 to 10.0.0.1,60000, the translator sets up a mapping from those addresses to source 2001:db8:ffff::c000:204,20 destination 2001:db8:31::6,49152.

If there is no (unambiguous) match for an existing data channel session when an incoming session request on port 20 arrives, the connection is refused with a TCP RST.

4.5. Both PORT and PASV

[RFC0959] allows a client to issue both PORT and PASV to use non-default ports on both sides of the connection. However, this is incompatible with the notion that with PASV the data connection is made from the client to the server, while PORT reaffirms the default behavior where the server connects to the client. As such, the
behavior of an ALG is undefined when a client issues both PASV and PORT.

4.6. Timeout

Wherever possible, control channels should not time out while there is an active data channel. A timeout of at least 30 seconds is recommended for mappings created by the FTP ALG that are waiting for initial packets.

Whenever a command from the client isn’t propagated to the server, the FTP ALG instead issues a NOOP command in order to keep the keepalive state between the client and the server synchronized. The response to the NOOP command is not relayed back to the client.

5. IANA considerations

None.

6. Security considerations

In the majority of cases, FTP is used without further security mechanisms. This allows an attacker with passive interception capabilities to obtain the login credentials, and an attacker that can modify packets to change the data transferred. However, FTP can be used with TLS in order to solve these issues. IPv6-to-IPv4 translation and the FTP ALG don’t impact the security issues in the former case nor the use of TLS in the latter case. However, if FTP is used with TLS or another authentication mechanism, the ALG function is not performed so only passive transfers from a server that implements EPSV or a client that supports PASV will succeed.

7. References


Appendix A. Server implementation recommendations

As EPSV works through IPv6-to-IPv4 translation transparently without additional effort on the part of the client, the server or an application layer gateway, it is highly recommended that all servers implement EPSV.

[RFC2428] suggests that the EPSV mode is useful both for clients with IPv6 connectivity and for clients operating behind a NAT device. As such, it is common for IPv6-capable clients to use EPSV even when communicating over IPv4. If a server doesn’t implement EPSV and responds with a 501 or 502 error, the client simply retries with PASV. This works well with both servers that have working EPSV and servers that don’t implement EPSV. However, there is a class of servers that does implement EPSV, but is unable to use this mode because the data connection can’t be established successfully. This is very likely the result of a middlebox monitoring the control channel interactions, and creating firewall or translation state according to the information 227 response after a PASV command. If the server supports EPSV but the middlebox doesn’t, the result is that the data connection cannot be established and the data transfer fails.
To avoid this, it is highly recommended that server implementers include a configuration setting that makes it possible to disable EPSV and EPRT support and respond with a 502 (command not implemented) error instead. Server operators can thus disable EPSV support in servers located behind PASV-only middleboxes so clients that issue EPSV can fall back to PASV gracefully rather than a timeout.

The test performed by Dan Wing showed that existing implementations present the address used for the server side of the control channel connection in the 227 response to a PASV command. Clients conforming to this specification depend on this behavior and it allows ALGs to translate a 227 PASV response to a 229 EPSV response without loss of information; as such it is highly recommended that servers continue to implement this limitation.

Many servers that support the FEAT command do not list EPSV and EPRT as a supported feature in the response to the FEAT command. It is recommended that EPSV and EPRT capability is included in the FEAT response, unless EPSV and/or EPRT are administratively disabled as outlined above.

Appendix B. Acknowledgements

Kentaro Ebisawa, Remi Denis-Courmont, Mayuresh Bakshi, Sarat Kamisetty, Reinaldo Penno, Alun Jones, Dave Thaler, Mohammed Boucadair, Mikael Abrahamsson and Dapeng Liu contributed ideas and comments. Dan Wing ran experiments with a large number of FTP servers that were very illuminating; many of the choices underlying this document are based on his results. This document adopts several important design decisions from [I-D.liu-behave-ftp64].

Iljitsch van Beijnum is partly funded by Trilogy, a research project supported by the European Commission under its Seventh Framework Program.

Appendix C. Document and discussion information

Please direct questions and comments to the BEHAVE mailinglist. The latest version of this document will always be available at http://www.muada.com/drafts/.
Author’s Address

Iljitsch van Beijnum
IMDEA Networks
Avda. del Mar Mediterraneo, 22
Leganés, Madrid 28918
Spain

Email: iljitsch@muada.com