The **F**ile **T**ransfer **P**rotocol (**FTP**) is one of the first protocols of the **Internet**. Data is sent in plain text. Clients have interactive or non-interactive modes. **FTP** has two data transfer modes:

- Active:
  The server "pushes" data to the client on a random high-numbered port. This method is not compatible with firewalls or **NAT** networks.
- Passive:
  The client requests a passive connection and the server opens a random high-numbered port for data transfer.

10.4. FTP Server

vsftpd (**V**ery **S**ecure **FTP D**aemon) was suggested by **SANS**, **IBM** and others for security reasons. Security concerns do not allow uploads and downloads of same content.

vsftpd features include:

- Virtual IPs.
- Virtual users.
- Stand-alone daemon or **inetd**-ready.
- Per-User configuration.
- Per-Source configuration.
- Optional **SSL** integration.

10.5. Anonymous FTP

To create an anonymous **ftp** download site, follow the steps below:

1. **Verify that an ftp user exists, or create one. Creating an ftp user is sometimes done by distribution:**

```
# getent passwd | grep ftp
```

2. **Verify or create a default directory. Set permissions:**

```
# mkdir -p /var/ftp/pub
# chown -R root:root /var/ftp/
# chmod 755 /var/ftp
# chmod 755 /var/ftp/pub
```

3. **Verify anonymous access is enabled in `vsftpd.conf`:**

```
anonymous_enable=YES
```

4. **Start or restart the vsftpd service:**

```
# service vsftpd restart
```

This is used for testing and debugging.

10.6. Authenticated FTP Sessions

To allow system users to create authenticated **FTP** sessions, you should make the following changes to **`vsftpd.conf`**:

```
local_enable=YES
write_enable=YES
local_umask=022
```

vsftpd uses **PAM** by default for authentication. You may not need to make changes to the **PAM** configuration file, but if you do, edit the `/etc/pam.d/vsftpd` file.

System security is enforced by the files `/etc/vsftpd/ftpusers` (`/etc/ftpusers` on **OpenSUSE**) and `/etc/vsftpd/users_list`:

- Account names listed in **`ftpusers`** are **NOT** allowed to login via **FTP**.
- Depending on the value of the **`userlist_deny`** setting in **`vsftpd.conf`**, the users in `/etc/vsftpd/users_list` act either as a whitelist or a blacklist.

10.7. rsync

The **rsync** protocol was written as a replacement for **rcp** and uses an advanced algorithm to intelligently transfer files. Only the files or parts of files which have changed are copied. **rsync** uses **Delta** encoding and requires the source and destination to be specified (either or both may be remote).

By itself, the **rsync** protocol does not have in-transit security. However, **rsync** can be tunneled over the **SSH** protocol.

An example of using **rsync** to copy the **CentOS** mirror is provided below:

```
$ rsync -av rsync://rsync.gtlib.gatech.edu/centos/6.3/os/x86_64/.\
               /srv/mirror/centos/6.3/os/x86_64/.
```

10.8. rsync Daemon

Even though most use **rsync** over **SSH**, the **rsync** protocol by itself is very useful. Starting the **rsync** daemon is as easy as running the following command:

```
# rsync --daemon
```

When running as a daemon, **rsync** will read the `/etc/rsyncd.conf` configuration file. Each client that connects to the **rsync** daemon will force **rsync** to re-read the configuration. There is no need to restart the daemon when making configuration changes.

There are two ways of starting the **rsync** daemon:

- Stand-alone mode, being started from a system **init** script.
  This is the default on **Ubuntu** systems.
  This is optional on **OpenSUSE** systems.
- Via the **xinetd** or **inetd** daemons.
  xinetd is the default on **CentOS** systems.
  xinetd is optional on **OpenSUSE** systems.

10.9. rsync Daemon Configuration

rsync daemon configurations include global settings (port, address and socket options) and module settings (**`path`**, **`comment`** and **`max connections`**).

The `/etc/rsyncd.conf` file defines global options, as well as **rsync modules** which will be served by the **rsync** daemon. A simple example of this is the following:

```
port = 873
address = 192.168.122.11

[mymodule]
     comment = Default new module, made by me
     path = /srv/rsync/mymodule
```

10.10. rsync Daemon Security

The **rsync** daemon introduces some security considerations:

- The **`use chroot`** option will change the daemon's root to the module's directory, reducing the impact of security bugs.
- The **`list`** option will modify the visibility of your module in the output of a remote listing.
- The **`filter`**, **`exclude`**, **`include`**, **`exclude from`**, and **`include from`** options restrict or enable access to files and/or patterns.
- The **`hosts allow`**, and **`hosts deny`** options control which **IP** addresses can connect to the server.
- The **`read only`** option (the default) disallows any write access. If **`read only`** is false, the filesystem permissions determine the access the **rsync** daemon has.
- The **`write only`** option will disallow read access, turning the module into a 'dead drop'.

10.11. Advanced rsync Usage

Advanced **rsync** usage provides a time machine backup, syncs only files which exist in the destination and deletes them, while backing them up.

Below is a time machine-like backup solution:

```
$ DATE=$(date "+%FT%T")
$ rsync -aP --link-dest=/srv/backup/current \
         /things/to/backup /srv/backup/backup-$DATE
$ rm -f /srv/backup/current
$ ln -s /srv/backup/backup-$DATE /srv/backup/current
```

To only copy files which already exist in the destination folder, do:

```
$ rsync -r --existing /some/src/. /some/dest/.
```

To delete items which do not exist on the source, but keep a backup, do:

```
$ rsync -r --backup --backup-dir=backup-$(DATE +%Y.%m.%d) --delete /some/src/./some/dest/.
```

10.12. scp and sftp

scp (**S**ecure **C**o**p**y) is non-interactive, while **sftp** (**S**imple **F**ile **T**ransfer **P**rotocol) is interactive.

An example of copying a file using **scp** is the following:

```
$ scp root@server:/etc/hosts /tmp/server.hosts
```

An example of copying a file using **sftp** is:

```
$ sftp root@server:/etc/
sftp> get hosts
Fetching /etc/hosts to hosts
/etc/hosts
sftp> quit
```

The commands **scp** and **sftp** will use your system or the user **ssh** client configuration files (`~/.ssh/config`).

rsync over **SSH** has the same command options as the stand-alone **rsync**. **rsync** uses your **ssh** configuration.

The **rsync** command will default to using the **SSH** protocol, unless you specify the protocol `rsync://`.

To copy an entire directory using **rsync** over **ssh,** use:

```
$ rsync -av root@server:/etc/. /srv/backup/server-backup/etc/.
```

10.14. WebDAV

WebDAV is an extension to **HTTP** for read/write access to resources and is available on most systems.

Windows, **Linux**, and **Mac OSX** have native support for **WebDAV**. Also, a command-line tool **cadaver** can be used to manipulate a **WebDAV** share. The **Apache** module **`mod_dav`** is one method to enable **WebDAV**.

You can enable **WebDAV** with a stanza like this:

```
Alias /mydav/ /srv/mydav/
<Directory /srv/mydav/>
     Options +Indexes
     Order Allow,Deny
     Allow from all
     Dav On

     AuthType Basic
     AuthName DAV
     AuthUserFile mydav.passwd

     <LimitExcept GET OPTIONS>
            Require user admin
     AuthUserFile mydav.passwd

     <LimitExcept GET OPTIONS>
            Require user admin
     </LimitExcept>
</Directory>
```

Mod_dav requires a defined lockfile, which should be writable by the **Apache** user:

```
DavLockDB davlockdb
```

WebDAV can be enabled in either a **<Directory>** or **<Location>** stanza. However, it is more secure to create an alias and use the directory options to increase security.

You should not enable **WebDAV** until you have secured your web server. Uploads, **PUT**, **POST**, and similar methods should not be allowed, except for by an authenticated user. Using plain-text authentication is also frowned upon. You can use **Digest**, or **SSL** to encrypt your **WebDAV** session.

BitTorrent is a protocol which allows for very efficient transfer of large files or directories (e.g. **ISO** files) using a distributed peer-to-peer architecture. A file served over **BitTorrent** is divided into small *chunks* and is distributed by anyone who is connected to the same tracker. This allows for much lesser resource use by the server hosting the original content, as the file needs to be downloaded only once. Also, **BitTorrent** becomes more efficient the more concurrent clients are participating.

Some of the different **BitTorrent** clients are:

- **Ktorrent: GUI client for KDE.**
- **rtorrent: CLI client for Linux.**
- **Transmission: Client for Windows, Linux, and Mac OSX.**
- **mktorrent**: **CLI** tool for creating `.torrent` files.