P+2: A peer-to-peer file-sharing protocol

In the search of a bare bones P2P protocol that is easy to understand and to program I have ended up designing my own one. Unfortunately that means it is not compatible or even based on any standard protocol. First of all, let us see what are the basic tasks our file-sharing P2P application needs to do.

Protocol Basics

Each peer can perform a set of operations:

- **Peer list maintenance:** A peer will connect to other known peers to obtain new peers to be added to its list.
- **Query:** When the user wants to obtain a file, a query operation will be triggered. Each known peer will be interrogated for files whose name contains a given string.
- **Download:** The user will chose one of the answers (if any) of a query operation and the file will be downloaded from the peer who provided that answer.

Regardless of the user actions, each peer will serve any request received. At any moment a peer may receive a requests to provide its list of peers, to search for a file whose name contains a certain substring or to deliver a given file requested by another peer.
Instead of using application level commands to tell other peer what the caller wants, this protocol uses different port numbers. Depending on the action desired the requesting peer will connect to a different port number (ie. p, p+1 or p+2).

Port numbers

Each peer is identified on the peer lists as a single string that consists of an IP address, expressed as dotted decimals plus a port number separated by a colon. This, peer 1.2.3.4:12345 represents a peer located at address 1.2.3.4 and port 12345. However, it's been mentioned above that peers are going to use different port numbers depending on what operation it's being requested. In our previous example, the port 12345 will server for the purpose of peer list management. Port 12346 will respond to query requests and the port 12347 will allow the download of a local file from another peer.

Action	**Port number**
peer list	p
query	p+1
download	p+2

So for the purpose of identifying a peer only the port number "p" is used. This system means that no fixed port number is used for this protocol. A more important reason for choosing this is that several peers may be hosted on the same computer easily. This is a very interesting feature while testing a software implementation.

Message format

Each time a peer connects to other peer the message exchange works the same way: A single text string is sent and next the other peer will respond with the result. For peer list maintenance (port p) and query operations (port p+1) the answer will be a sequence of text lines, containing the list of peers or the filenames. However, for download operations (port p+2) the response will be the binary contents of the file requested (no matter it was a text or a binary file).

For each operation the string the requesting peer sends is different:

- A peer list maintenance request will transmit a text line containing the IP address and the port number (p) of this peer separated by a colon.
- A file query request will transmit a text line containing the word (or characters) the peer is trying to find on the callee's filenames.
- A download request will transmit a text line containing the exact filename of the file caller peer wants to download.

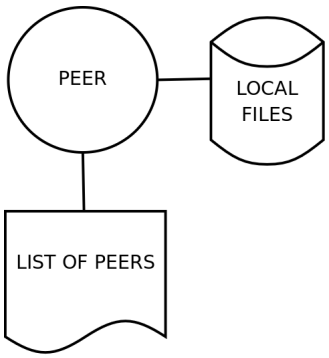
Every time, the connection will be initiated by the calling peer and it will be closed by the callee once requested data (if any) has been transmitted.

Protocol internals

In order not to overload peers, the action of connecting with other peers to find new peers is going to be rate limitted. Each connection attempt to another peer will not happen until 10 seconds have elapsed since the last connection.
Three different servers are needed listening on ports p, p+1 and p+2 to be able to answer any request coming from another peer. It is possible that a given request may have an empty answer (i.e. wrong filename being requested, empty answer for a query, empty peer list).
The initial connection once the peer is started is a critical issue, as the list of peers is empty on startup. It is safe to assume that every time a peer is started a initial peer identity is provided too, so this peer will connect to this initial peer to start populating its peer list with new peers.

Node details

Each peer has a set two basic pieces of information handled locally. Firstly, a peer has a private folder where shared files are located. At any moment the peer may check for filenames matching a certain query. Secondly, a peer has a list of known peers, this list grows every time the peer learns about new peers.



User interaction

Each peer is operated by an interactive user. The main goal of this software is for the user to obtain new files of interest. Each peer is also sharing all the files present on a peer owned folder on the filesystem. All upload and download operations will use this folder only.
The user interface will start waiting for a query from the user. When the user types a query, the peer will go connect sequentially to each other peer on the list to find the desired file. A file is considered to be of interest when the filename matches the query string (ie the query string is contained on the file name). Many matches are possible on

a given query request on a given peer.
The user interface will show a list of all the matches obtained for the user to chose the desired filename. Once the user selects a choice, the selected filename will be downloaded from the peer who has this file. Once download is over the user will have a chance to repeat the process.
The program does not have a command to quit (it has to be killed instead).