File Transfer Facility

You must support client applications that will release software electronically. One part of that support is a mechanism that allows those applications to transfer software release files to remote sites. Therefore, you must provide software that allows a client application to transfer a specified file to a specified destination. (The client indicates the destination by providing a site name.)

Assume that your site already has low-level file transfer protocol classes, each instance of which interacts with one remote site. (These classes exist, so you need not design and implement them.) Two different protocol classes exist. One, a direct send protocol object, has the following (synchronous) public operation:

> `send(f: File)`, transfers the file and returns a status of `fileTransferred` or `transferFailed`.

The other is a partial send protocol. It can transfer files of up to 100KB in size. Longer files must be transferred in pieces. To facilitate this, however, the partial send protocol has the notion of a connection; all fragments sent between the opening and closing of a connection are assumed to be parts of the same file. A partial send protocol object has three public (synchronous) methods:

> `openConnection(filename)`, opens a connection to the remote site so that a file of the stated name can be transferred, returns a status of `connectionOpen` or `connectionDown`;
>
> `sendFilePart(file)`, sends a file (or portion of a file) of up to 100KB in size, returning a status of `transferSuccess` or `transferFailure`; and
>
> `closeConnection( )`, informing the remote site that all parts of this file have now been sent (and allowing that site to reconstruct the complete file), returning a status of `connectionClosed` or `connectionDown`.

The protocol you use to send a file must match the protocol the client site is using to receive the file. A client site may use different transfer protocols at different times, however, and so to select the correct type of protocol, you must query the remote site to determine what type of protocol it is currently using. You must then create a protocol object of the appropriate type. Assume that the constructors for the Direct Send Protocol and Partial Send Protocol classes take the remote site's address (such as an IP address) as an argument.

When transferring a file, each protocol object will place the transferred file in a special directory at the remote site. You can assume that software at that site will handle the receipt of files in that directory.

Because new protocols may by introduced in the future, you should make your design as tolerant of new protocols as possible.

Your file transfer package should handle transfers at any of three priorities: background, normal, and emergency. Files sent with emergency priority should be sent before files with normal priority, files with normal priority before files with background priority. Once a file is handed to a protocol object, however, its transfer cannot be interrupted.

You should allow transfers with retry counts. In the case of transmission errors, a transfer with a non-zero retry count will be retried that number of times. The default retry count should be zero.

Your package should also support transfers at (or about) time t. If a client asks to transfer a file at midnight, for example, that request should be postponed until that time.

A client must be able to query the status of a transfer. Possible status values include pending, in progress, completed successfully, and completed unsuccessfully. In addition, a client must be able to cancel a pending transfer. (An attempt to cancel a transfer that is no longer pending, however, should be ignored.)

You can support broadcasts to multiple sites, but you need not do so. (This feature is on someone's wish list.) If you permit this, you must determine how you will handle the status of the transfer (because the file may have been transferred to some sites but not to others) and the notification (such as whether a single notification or multiple notifications are sent).