**REMOTE BACKUP**

Il progetto può essere diviso in 3 parti fondamentali: Client,Server, Socket connection. Per cui è pensabile che ciascun componente del gruppo si occupi di una parte. Occorre rendere ciascuna parte indipendente dalle altre con una propria logica. Ciascuna parte inoltre deve comunicare e conformarsi con delle regole con le altre attraverso delle interfacce (sostanzialmente per lo scambio di dati ).

Come prima cosa (secondo me) occorre pensare e creare un protocollo applicativo usato per lo scambio di messaggi in rete.

**PROTOCOLLO APPLICATIVO**

Ho pensato ad un protocollo http- like con richieste tipo GET, POST,UPDATE,DELETE.  
Inoltre il protocollo è binario e non testuale (come http). Il problema sorge spontaneo, come gestire le terminazioni del messaggio??  
Le operazioni che devono essere fatte sono:  
- Aggiunta file: Il client invia un messaggio al server con scritto ADD <nomefile>. Il server crea il file e risponde con ok ( o errore , queste sono le due alternative). Il client allora invia il contenuto del file al server e alla fine invia un messaggio di terminazione. Il server invierà un messaggio di ok ( o errore)

-altre…..

E’ chiaro che il client monothread del client nella comunicazione si blocca e non monitora più per tutta la durata della comunicazione. Secondo me occorre creare due thread, uno che gestisce il monitoraggio e uno che gestisce la comunicazione. Schema simile al modello del produttore e consumatore. Il monitor inserisce in una coda le modifiche , il consumatore le inoltra al socket. Tale coda deve essere thread-safe.

**CLIENT**

Deve monitorare continuamente una cartella specifica (può essere specificata ad esempio come parametro della linea di comando) e verificare che tutto il suo contenuto sia sincronizzato con il server.   
E’ richiesto che al “boot” del programma l’utente si autentichi (presumibilmente con username e password). Si può fare una cosa simile ad applicazioni web ovvero si inserisce lato server uno (o più ) username e hash della password (quindi si salta la fase di creazione dell’utente, non è richiesta e si suppone che l’utente sia già registrato). Una volta autenticato l’utente deve specificare una cartella (path relativo o assoluto) dopodichè inizia il bordello.  
Occorre creare un FolderWatcher che monitori ogni tot tempo la cartella in modo ricorsivo (vedi l’ultimo link della parte client). Problemi che possono nascere ed eventualmente che andrebbero gestiti:

* Se la cartella contiene molte sottocartelle e file?? Potrebbe non bastare un singolo thread?? Una possibile soluzione è quella di conteggiare il numero di cartelle e file, stimare una soglia, se il numero è maggiore della soglia creare altri thread. Ciascun thread si occupa di una sottocartella (comunque non semplice, secondo me non necessario, il multithreading è da fare lato server e non nel client)