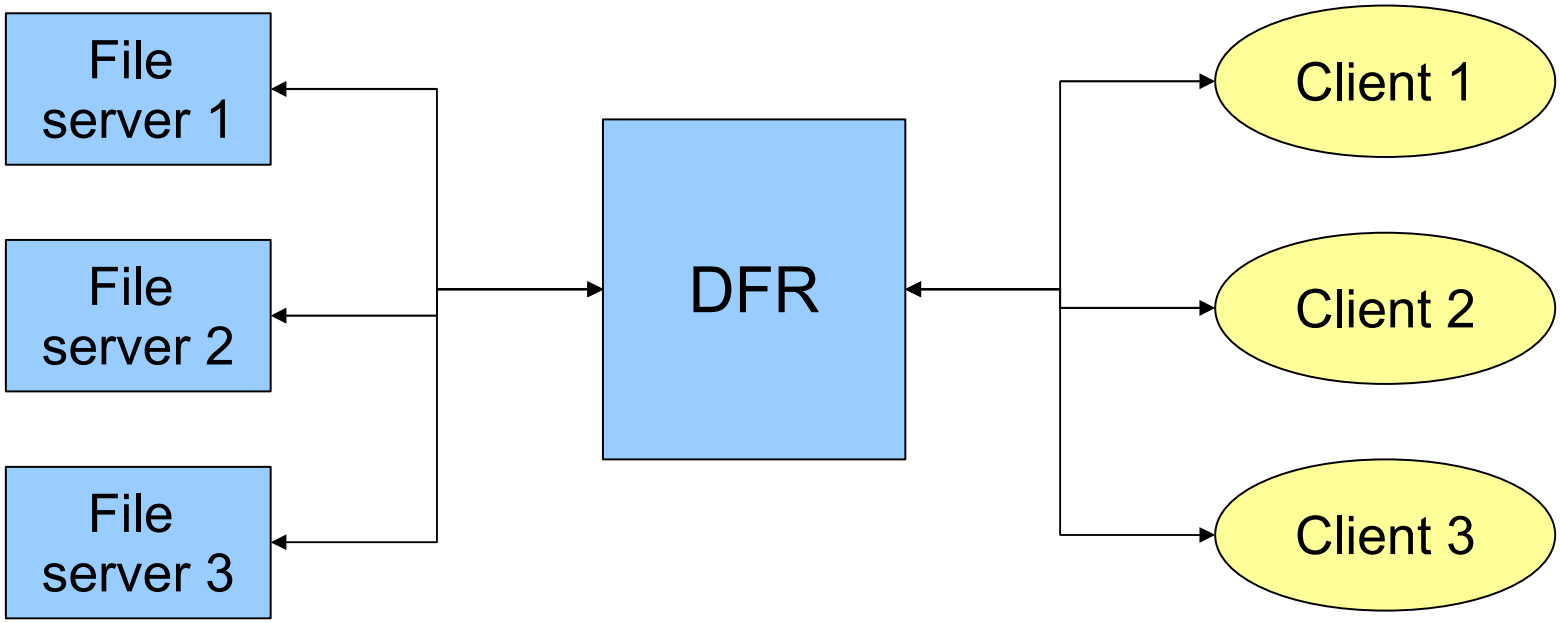
Scuola Superiore Sant'Anna

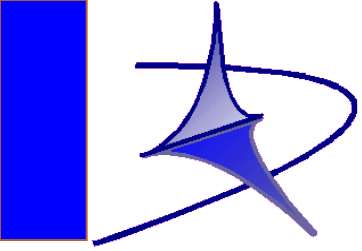
Progetto parte Unix

AA 2008-2009: Distributed File Repository

Distributed File Repository

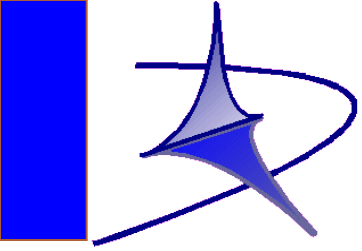
- Descrizione del sistema da realizzare
  - Progettare e implementare un server che mantiene una lista di file e delle loro “locazioni” in server remoti
  - Schema generale





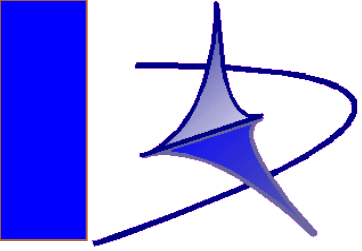
Specifica DFR

- Il Distributed File repository (DFR):
  - Mantiene internamente delle strutture dati che memorizzano dove si trova un file
  - Accetta richieste da parte dei FileServer (FS) e da parte dei Client (CL):
    - Da parte dei FS accetta richieste di *registrazione* di un file, con una serie di parametri
    - Da parte dei CL, accetta richieste di conoscere la locazione dei file (cioè, quale file server possiede il file in questione)
  - Per semplicità, supporremo che un file venga identificato in maniera univoca in tutto il sistema dal suo nome



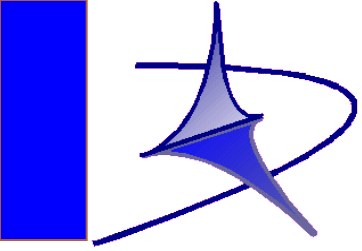
Specifica FS

- Un FS può:
  - Registrare un file presso il DFR, inviando
    - Inviando il nome del file
    - l'IP e la posta su cui il FS accetta richieste di trasferimento del file
    - Il numero massimo di clienti che possono richiedere il trasferimento contemporaneamente
  - De-registrare un file presso il DFR
    - Per segnalare che il file non è più disponibile



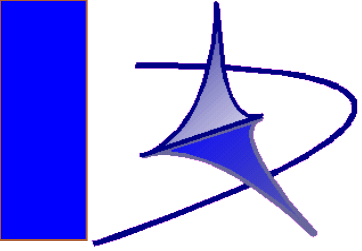
Specifica CL

- Un CL può:
  - Richiedere di cominciare un trasferimento per un certo file:
    - Viene restituito l'indirizzo e la porta del FS che è disponibile al trasferimento del file
    - Se il file non è attualmente disponibile, viene comunicato un messaggio di errore
  - Comunicare la fine del trasferimento
  - Richiedere la registrazione di una notifica
    - In caso un file non sia presente o non sia attualmente accessibile, il client può richiedere che il DFR invii una notifica non appena il file si renderà disponibile



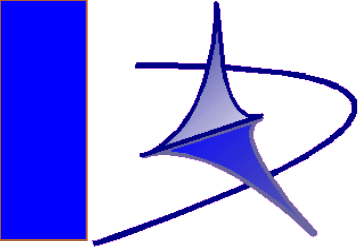
Specifiche aggiuntive

- Bisogna tenere conto anche dei seguenti requisiti
  - Uno stesso file può essere disponibile presso uno o più FS
  - Se un file server ha specificato che al massimo x CL possono trasferire un certo file contemporaneamente, e tale numero è già stato raggiunto, il prossimo CL che richieda il trasferimento riceve un messaggio di errore del tipo “too many clients”
  - Nel caso un CL riceva un messaggio di errore “too many clients” può richiedere una registrazione delle notifiche, ed essere avvisato non appena il numero dei client che accedono al file sia inferiore al numero massimo x
  - Un CL può richiedere più notifiche su più file



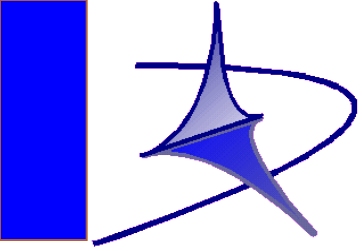
Struttura interna FS

- FileServer
  - Deve essere un processo multi-threaded,
    - un thread manager che inizialmente comunica con il DFR e poi si mette in attesa di richieste dal client, e un thread per ogni client differente
  - Invia i seguenti comandi al DFR
    - **register_file** (file_name, max_clients, port);
    - **unregister_file** (file_name);
  - Riceve dai CL i seguenti comandi:
    - **get_file** (filename);
      - Response is OK (the file is immediately sent), ERR_FILE_NOT_EXIST, or ERR_TOO_MANY_CLIENTS (no file is sent)



Struttura interna DFR

- DFR
  - Deve essere un processo multi-threaded
  - Un thread in attesa di richieste dai FS, e un thread in attesa di richieste dai CL, più un thread per ogni richiesta da un client
  - Implementa una struttura dati Repository, che contiene i dati sui file
  - Implementa il meccanismo di notifica tramite condition variables
    - Quando un CL richiede una notifica, il thread corrispondente si blocca su una variabile condition
    - Quando la notifica deve essere effettuata, si segnala il thread che a sua volta manda un messaggio al CL

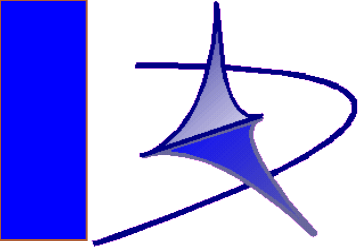


Struttura DFR - II

- DFR
  - Accetta dal FS i seguenti messaggi:
    - **register_file** (file_name, max_clients, port);
      - Response is OK
    - **unregister_file** (file_name);
      - Response is OK or ERR_FILE_NOT_REGISTERED

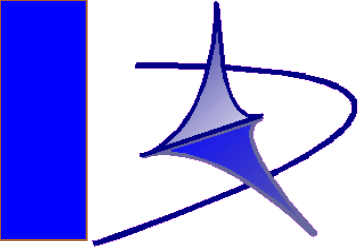
Struttura DFR - III

- Continua
  - Accetta dai CL i seguenti messaggi:
    - **reg_notification** (file_name);
      - Response is OK
    - **start_file_transfer** (file_name);
      - Response is OK, ERR_TOO_MANY_CLIENTS, or ERR_FILE_NOT_EXISTS
    - **end_file_transfer** (file_name);
      - Response is OK, ERR_NO_TRANSFER



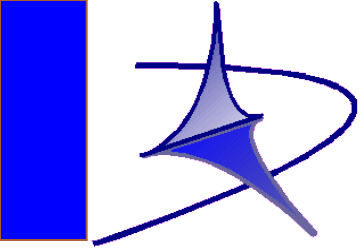
Struttura DFR - IV

- Continua
  - Il DFR notifica il client mandandogli indietro il messaggio:
    - **notify** (file_name);
      - Si aspetta di ricevere OK, se lo riceve, automaticamente suppone sia stato cominciato un comando di **start_file_transfer** (file_name);
      - Se riceve ERR_NO_NOTIFICATION, non fa niente



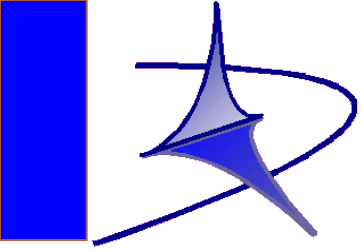
Test

- Il progetto deve contenere delle configurazioni di test che testino la funzionalità dei seguenti scenari
  - Accesso concorrente
    - 2 FS specificano ognuno 3 file (di cui uno in comune tra i due FS), per ogni file un massimo di 2 CL
    - 10 client cercano periodicamente e concorrentemente di accedere al file in questione
  - Notifiche multiple
    - 10 CL richiedono una notifica un file che ancora non esiste nel repository
    - Successivamente, un FS si registra fornendo il file in questione per un massimo di 3 CL, e tutti i CL vengono notificati (3 alla volta)



Implementazione

- Linguaggio e OS
  - Si consiglia lo sviluppo sul sistema operativo Linux o FreeBSD
  - E' possibile scegliere fra C o di C++ (il secondo è consigliato)
  - Si consiglia l'uso di Doxygen per documentare il codice (http://www.doxygen.org)
  - E' richiesto l'uso del comando make e dei makefiles
- Organizzazione del codice
  - Conviene cominciare con il progettare e implementare i meccanismi di comunicazione di livello più basso (invio messaggi e eventuale attesa della risposta), e testarli separatamente con piccoli programmi di test. Progettare un'interfaccia semplice da utilizzare
  - Poi, progettare le strutture dati (in particolare il repository dei file)
  - Infine progettare il codice del DFR e dei FS e CL in maniera incrementale
  - Cercare (per quanto possibile) di testare separatamente i vari pezzi di codice con degli appositi programmini da mettere nella directory test



Come strutturare il codice

- Includes
  - Tutti gli include in una directory include/
  - Almeno un .h comune tra DFR e FS, e un .h comune tra DFR e CL, contenenti la specifica dei messaggi e dei codici di errori;
- Implementation
  - 3 directory src/ separate: src_CL, src_DFR e src_FS, contenenti rispettivamente i file .c/.cpp con il codice per i CL, per il DFR e per il FS
  - Una directory test/ contenente i file (script, repositories, file sorgenti, etc.) per i due test
  - Usare un makefile diverso per ogni directory sorgente e un makefile globale nella directory principale
- Documentazione e commenti
  - Un file README nella directory principale
  - Commenti dettagliati nel file .h, davanti i prototipi delle funzioni (stile doxygen)
  - Pochi commenti nei file .c/.cpp