Testo del progetto opzionale di Reti di Calcolatori, edizione 2010 (SFBTP v1.1)

Il progetto consiste nel progettare e implementare un applicazione Java (o, a scelta, C++ o Perl) per il file transfer veloce e affidabile di gruppi di file di piccole dimensioni, tra gruppi di host peer. I file trasferibili misurano da 1 a 50 byte al massimo, e i gruppi di file sono costituiti da non più di 10 elementi ciascuno. Il nome di ciascun file (sono ammessi solo caratteri ASCII a 7 bit) non può eccedere i 20 byte. Il gruppo di peer può essere costituito da 20 elementi al massimo. Chiameremo il protocollo di scambio tra peer SFBTP (Shared File Bunch Transfer Protocol). La definizione dei dettagli di funzionamento del protocollo SFBTP è a carico dello studente.

Ognuno dei peer funge da server sempre in ascolto dei messaggi provenienti dagli altri peer (l'elenco dei quali è memorizzato in una *peerList*), ed è configurato per accedere a una cartella contenente i propri file locali (*localFolder*).

Messaggi in ingresso da gestire

JOIN IP, PORT Notifica che il peer accessibile sull'indirizzo IP alla porta PORT richiede di entrare a far parte della *peerList*. Ogni altro peer viene informato della presenza del nuovo peer ed entra a far parte della *peerList* di ciascun elemento del gruppo. Non è necessario che i parametri IP e PORT coincidano con quelli origine del comando.

LEAVE IP, PORT Notifica che il peer accessibile sull'indirizzo IP, alla porta PORT è uscito dalla *peerList* del gruppo. Ogni altro peer deve essere informato dell'abbandono del peer segnalato e deve aggiornare la propria *peerList*. Non è necessario che i parametri IP e PORT coincidano con quelli origine del comando.

LISTPEERS Ottiene, in formato ASCII (una coppia IP <spazio> PORTA per linea), la *peerList* dal peer che riceve il messaggio.
LISTFILES Ottiene l'output del comando "**`ls -la`**" (o "**`dir`**") lanciato sul *localFolder* del peer che riceve il messaggio.

SHARE <FILES> Il peer che riceve il messaggio riceve il gruppo di files contenuti nel messaggio e li memorizza nel proprio *localFolder*. Ogni altro peer facente parte della rete deve ricevere e memorizzare gli stessi file nel proprio *localFolder.* In caso di nomi di file ripetuti, il file pre-esistente viene sovrascritto.

E' consentita l'aggiunta di qualsiasi altra tipologia di messaggio si ritenga necessaria.

Vincoli progettuali

1. Affidabilità: il peer che sottomette i comandi JOIN, LEAVE e SHARE deve avere garanzia che i comandi siano andati a buon fine, oppure avere notifica di fallimento;

2. Prestazioni: è necessario progettare le modalità di scambio di informazioni tra i peer in maniera tale da ridurre al minimo il traffico di dati scambiato e il tempo intercorso fra la sottomissione di un comando JOIN, LEAVE e SHARE e la notifica di avvenuta esecuzione.

3. Consistenza delle informazioni: la rete di peer deve tenere sincronizzata la propria *peerList* con gli altri membri del gruppo e rimuovere automaticamente quei peer che dovessero cessare la loro attività a causa di malfunzionamenti che non consentano di notificare l'abbandono con un comando LEAVE .

Devono essere inoltre messi a disposizione i comandi (eseguibili da shell)

joinGroup IP PORTA IP2 PORTA2 Invia un comando JOIN IP2 PORTA2 al peer accessibile su IP PORTA.
leaveGroup IP PORTA IP2 PORTA2 Invia un comando LEAVE IP2 PORTA2 al peer accessibile su IP PORTA.

shareFiles IP PORTA *files* Invia un comando SHARE <files> al peer accessibile su IP PORTA. Sulla linea di comando deve essere possibile specificare un elenco di files oppure delle wildcard con '*' (e.g. shareFiles 127.0.0.1 8081 sr* sottomette al peer specificato tutti i file della cartella corrente il cui nome inizia per 'sr'). Il comando ignora i file che eccedano la dimensione massima richiesta e file successivi al decimo.

listPeers IP PORTA Invia il comando LISTPEERS al peer accessibile su IP, PORTA e ne emette l'output a video.
listFiles IP PORTA Invia il comando LISTFILES al peer accessibile su IP e PORTA specificati e ne emette l'output.
launch PORTA localFolder avvia un peer SFBTP in ascolto su PORTA su tutte le interfacce dell'host corrente e con *localFolder*.

Il programma sarà testato in condizioni di alta probabilità di perdita di dati su un gruppo di 19 peer: i 19 peer saranno distribuiti su 2 o più domini di collisione distinti.

Non è consentito fare uso di librerie esterne o codice sorgente di terze parti al di là delle funzionalità messe a disposizione dal JDK 6, o dalle funzioni di libreria del linguaggio scelto. Si può usare qualsiasi protocollo di trasporto. La dimensione massima di un gruppo è di due persone.

Bonus: al gruppo che programma il sistema SFBTP più veloce sono attribuiti 2 punti aggiuntivi sul voto finale. Al gruppo secondo e terzo classificato, 1 punto. Non è prevista l'attribuzione automatica della lode a chi dovesse in tal modo conseguire un voto superiore a 30.
Le prestazioni del sistema saranno verificate mettendo a regime una rete SFBTP di 19 peer e determinando il tempo *TotalTime* nel seguente modo:

$$TotalTime = (JoinTime + LeaveTime) + ShareTime$$

Dove *JoinTime+LeaveTime* è determinato aggiungendo e togliendo per 10'000 volte lo stesso peer dalla rete SFBTP, facendo uso dei comandi shell *joinGroup* e *leaveGroup* e misurando i tempi di esecuzione finali; *ShareTime* è ottenuto condividendo per 10'000 volte consecutive lo stesso gruppo di 10 file, facendo uso ripetuto del comando *shareFiles.*
Tutti i gruppi che consegnano una versione funzionante del progetto (inclusiva di sorgenti) possono sostituire l'orale con la discussione del proprio codice, al quale sarà attribuito un voto appropriato. La valutazione negativa *individuale* del progetto opzionale comporta l'annullamento del progetto e il sostenimento della prova orale alla successiva prova d'esame (Esempio, gli studenti Alice e Bob consegnano lo stesso progetto insieme: Alice sostiene una buona discussione, Bob no → Bob deve fare l'orale, Alice supera l'esame).