Specifiche dei Progetti Numero 1 e 2
per l'Esame di Reti di Calcolatori **a.a. 2010/11**
Prof. Fabio Panzieri e Vittorio Ghini

1 - Proxy miniHTTP/miniHTML con pre-fetching
2 - Proxy miniHTTP/miniHTML con parallelizzazione
(Versione Temporanea, 17 maggio 2011)

(NB: ogni gruppo deve comunicare al docente la scelta del progetto, spedendo una mail con subject "Scelta progetto Reti"
da quel momento ha 5 mesi per terminarlo, in ogni caso la consegna deve essere effettuata entro fine febbraio 2012)

1. Formato miniHTML delle Pagine.

Una pagina miniHTML è un documento di testo che può contenere alcune sequenze speciali. Un documento di tipo miniHTML non può contenere il carattere ascii di valore 0 (il terminatore delle stringhe C). In un documento miniHTML i caratteri **< >** sono riservati ed usati come iniziatori e terminatori di una sequenza riservata. Non possono comparire nel documento, se non come delimitatori di una sequenza speciale. Ciascuna sequenza riservata in un documento miniHTML inizia con un carattere < e termina con un carattere >.

Un client (browser) miniHTTP deve recuperare e visualizzare documenti di tipo miniHTML.

- La parte di testo di un documento miniHTML e' visualizzato così come è.
- Il carattere \n è visualizzato come una "andata a capo".
- Le sequenze speciali sono interpretate e visualizzate in modi diversi.

Ciascuna sequenza riservata può contenere uno tra due tipi di informazioni:

1) la sequenza **<REF=indirizzorisorsa>** contiene le informazioni "**indirizzorisorsa**" per recuperare (scaricare, download) un documento di solo testo che il browser miniHTTP dovrà visualizzare sostituendolo alla sequenza riservata.

2) la sequenza **<IDX=numerounivoco;REF=indirizzorisorsa>** contiene le informazioni "**indirizzorisorsa**" per recuperare un documento miniHTML. Il browser (il client) miniHTTP/miniHTML visualizza tale sequenza visualizzando solo il numero "numerounivoco". Tutti i numeriunivoci all'interno di una stessa pagina devono essere diversi tra loro. Il browser permette all'utente di scegliere un link da seguire semplicemente digitando il numero "numerounivoco". Quando l'utente del browser miniHTTP seleziona il numero "numerounivoco" allora il browser utilizza l'informazione "**indirizzorisorsa**" per richiedere al server il documento specificato da "**indirizzorisorsa**". La richiesta viene effettuata mediante messaggi di tipo miniHTTP.

Esempi di informazioni "numerounivoco": 1 2 3 4 5.
Formato di informazione "indirizzorisorsa": mhttp://130.136.4.23:60000/a.txt

Le informazioni per scaricare un documento, che seguono la parola chiave "REF=", sono formate da tre parti principali:
1) La prima parte è uno specificatore di protocollo che è sempre "**mhttp://**".
2) La seconda parte è una parte di testo che contiene l'indirizzo IP in forma "numeri e punti" e la porta TCP del server a cui chiedere le informazioni. Indirizzo IP e porta sono separati da un carattere. Tale seconda parte segue i due // della prima parte e termina con un altro / .
3) La terza parte è il percorso relativo per rintracciare la risorsa nel file-system del server, tale percorso parte dalla directory base in cui il server ha collocato i suoi file. Tale terza parte non può mai essere più lunga di 2048 bytes.

1.1. Esempio di documenti in formato miniHTML

Documento "dir/file1.txt"
un protocollo di rete

Documento "dir/file2.txt"
importanti informazioni

Documento "dir/file3.txt"
Vuoi ulteriori informazioni? **<REF=mhttp://130.136.4.23:60000/dir/file5.txt>**

Documento "dir/file4.txt"
Questo è un documento in formato miniHTML\n
Che fornisce **<REF=mhttp://130.136.4.23:60000/dir/file2.txt>** sul TCP.\n
Come noto, il TCP è **<REF=mhttp://130.136.4.23:60000/dir/file1.txt>**.\n
\n
Per ulteriori informazioni,\n
seleziona **<IDX=13;REF=mhttp://130.136.4.23:60000/dir/file3.txt>**

Documento "dir/file5.txt"
E chi se ne frega ! \n
Vaff..!\n

Ecco come viene visualizzato il documento /dir/file4.txt

Questo è un documento in formato miniHTML
Che fornisce importanti informazioni sul TCP.
Come noto, il TCP è un protocollo di rete.

Per ulteriori informazioni,
seleziona <13>

Se l'utente che usa il browser miniHTTP digita su tastiera il numero 13 e preme il tasto <ENTER> allora viene scaricato e visualizzato il file dir/file3.txt, visualizzando ciò che segue.

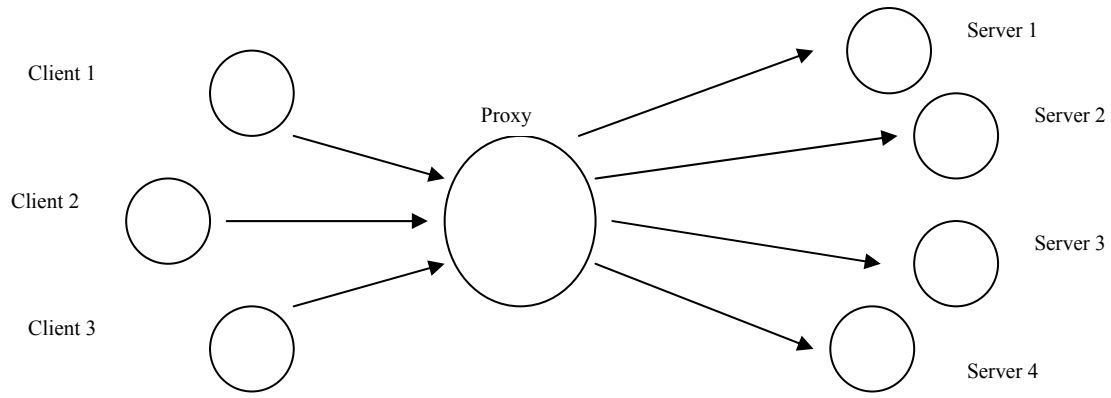
Vuoi ulteriori informazioni? E chi se ne frega !
Vaff..!

2. Download delle pagine - Protocollo miniHTTP.

Le entità di tipo client (browser), proxy e server miniHTTP trasferiscono pagine in formato miniHTML utilizzando un protocollo di comunicazione denominato miniHTTP, che utilizza connessioni TCP.

Per scaricare ciascuna pagina, un client
- instaura una connessione TCP con il proxy (di cui conosce l'indirizzo IP e porta),
- tramite questa connessione invia il messaggio miniHTTP,
- ancora mediante questa connessione riceve una risposta in formato miniHTTP,
- infine chiude la connessione.

Il proxy, dopo avere ricevuto una richiesta da un client,
- legge la richiesta individuando indirizzo e porta del server,
- instaura una connessione TCP con il server,
- tramite questa connessione invia il messaggio miniHTTP al server,
- ancora mediante questa connessione riceve una risposta in formato miniHTTP dal server,
- infine chiude la connessione con il server.
- Poi spedisce la risposta al client, eventualmente salvando localmente la risorsa scaricata
- Infine chiude la connessione con il client.



Attenzione, **Attenzione**, **Attenzione**, **Attenzione**, **Attenzione**!!!!! le connessioni soffrono dei problemi della rete, quindi la connessione con il server/proxy potrebbe terminare inaspettatamente, oppure potrebbe smettere di trasmettere dati senza dare altri segnali di malfunzionamento. Per ulteriori dettagli vedi il punto 5.

I messaggi miniHTTP si distinguono in **Richieste** e **Risposte**.

Esistono due tipi di Richieste, una detta GET l'altra detta INF.

- La richiesta GET può contenere una opzione detta RANGE. La richiesta GET chiede di scaricare un documento, l'opzione RANGE chiede di scaricare solo una porzione del documento. L'opzione RANGE specifica l'indice del primo e dell'ultimo byte dell'intervallo da scaricare. Tali indici iniziano da 1 e finiscono con la lunghezza del file. Nell'opzione RANGE l'indice di inizio deve essere minore o uguale all'indice di fine. E' possibile chiedere i byte che vanno da un certo indice fino alla fine del file anche senza conoscere la lunghezza del file.

- La richiesta INF chiede delle informazioni sul file.
Ogni messaggio di richiesta è composto di una o più righe, ciascuna riga è delimitata da un carattere finale **\n** . Ciascun messaggio di richiesta termina sempre con una riga costituita dal solo carattere terminatore **\n** .

FORMATO MESSAGGI DI RICHIESTA GET SENZA RANGE:

GET mhttp://130.136.2.23:55554/dir/a.txt\n
\n

FORMATO MESSAGGI DI RICHIESTA GET CON RANGE SPECIFICATO: tale messaggio chiede di scaricare i byte del file a.txt dal 33-esimo byte al 46'esimo byte compresi gli estremi.

GET mhttp://130.136.2.23:55554/dir/a.txt\n
Range 33-46\n
\n

FORMATO MESSAGGI DI RICHIESTA GET CON RANGE con FINE INDEFINITA: tale messaggio chiede di scaricare i byte del file a.txt dal 33-esimo byte fino alla fine del file.

GET mhttp://130.136.2.23:55554/dir/a.txt\n
Range 33-\n
\n

FORMATO MESSAGGI DI RICHIESTA GET CON RANGE che chiede tutto il file.

GET mhttp://130.136.2.23:55554/dir/a.txt\n
Range 1-\n
\n

FORMATO MESSAGGI DI RICHIESTA INF

INF mhttp://130.136.2.23:55554/dir/a.txt\n
\n

Esistono diversi tipi di messaggi di risposta, qui di seguito descritti:

FORMATO MESSAGGIO DI RISPOSTA 404 FILE NOT FOUND, viene ricevuto quando la richiesta specifica un file che il server non ha.

404\n
\n

FORMATO MESSAGGIO DI RISPOSTA 405 INTERVAL NOT FOUND, viene ricevuto quando la richiesta specifica un file che il server ha ma è stato specificato un intervallo che cade parzialmente o totalmente fuori dal file.

405\n
\n

FORMATO MESSAGGIO DI RISPOSTA 403 WRONG REQUEST, viene ricevuto quando la richiesta inviata ha un formato non aderente alle specifiche miniHTTP.

403\n
\n

FORMATO MESSAGGIO DI RISPOSTA 402 UNKNOWN ERROR, viene ricevuto quando

la richiesta inviata ha provocato un errore che il server non riesce a specificare meglio..

402\n
\n

FORMATO MESSAGGIO DI RISPOSTA 200 OK, viene ricevuto dopo una richiesta di tipo GET che NON specificava un range. Dopo il \n che termina il messaggio c'e' un blocco di bytes che è il contenuto del file che è stato richiesto, la cui lunghezza è specificata dal campo Len.

200\n
Len 1493\n
Empire 600\n
\n
BLOCCO di 1493 BYTES

FORMATO MESSAGGIO DI RISPOSTA 201 OKRANGE, viene ricevuto dopo una richiesta di tipo GET che specificava un range. Dopo il \n che termina il messaggio c'e' un blocco di bytes che è il range richiesto del file, la cui lunghezza è specificata dal range.

201\n
Range 40-67\n
Empire 600\n
\n
BLOCCO di 67-40+1 BYTES

FORMATO MESSAGGIO DI RISPOSTA 202 INFO, viene ricevuto dopo una richiesta di tipo INF. Viene specificata la lunghezza del file, e la validità temporale della risposta.

202\n
Len 79\n
Empire 600\n
\n

3. Obiettivo Progetto 1: Proxy miniHTTP con Prefetching.

Si deve realizzare un programma che svolga il ruolo di proxy miniHTTP/miniHTML e che presenti alcune caratteristiche particolari, qui di seguito elencate:

Il proxy accetta connessioni TCP dai client e instaura connessioni TCP con dei server.
Su ogni connessione accettata il proxy riceve una richiesta GET in formato miniHTTP, poi risponde con una risposta in formato miniHTTP, infine chiude la connessione.
Su ogni connessione TCP instaurata con un server, il proxy manda una richiesta in formato miniHTTP, poi riceve una risposta in formato miniHTTP, infine chiude la connessione.
Il protocollo miniHTTP è stato specificato precedentemente.

Lo scopo del proxy miniHTTP con pre-fetching consiste di due parti:
1) mantenere sul proxy una cache delle pagine miniHTML richieste dai client, eliminandole quando termina la validità della singola pagina.
2) intercettare le richieste mandate dai client, scandire le risposte dei server, individuando le risorse che i client dovranno successivamente richiedere (quelle indicate dalle sequenze speciali REF ed IDX+REF), e richiederle anticipatamente ai server prima ancora che i client

ne facciano richiesta. In tal modo il proxy può scaricare anticipatamente i files richiesti dal client, riducendo così i tempi di attesa dei client stessi.

1. si parla di pre-fetching di primo livello intendendo il download dei files specificati nelle sequenze REF della risorsa, files che il client deve sempre caricare.
2. si parla di pre-fetching di secondo livello intendendo il download anche dei files specificati nelle sequenze IDX+REF della risorsa, files che il client non deve sempre caricare.
3. si parla di pre-fetching di terzo livello intendendo il download dei files specificati nelle sequenze REF delle risorse specificate dalle sequenze IDX+REF.
4. E così via salendo di livello in livello.

Il proxy deve implementare obbligatoriamente il pre-fetching di primo e secondo livello. Il pre-fetching di terzo livello è opzionale e, se correttamente implementato, migliora la valutazione del progetto. Il pre-fetching di livello superiore al terzo non è consentito.

NB: All'avvio il proxy deve eliminare tutti i file precedentemente salvati in cache.

Gli eseguibili da utilizzare per questo progetto sono denominati Client1.exe e Server1.exe.

4. Obiettivo Progetto 2: Proxy miniHTTP con download parallelo.

Si deve realizzare un programma che svolga il ruolo di proxy miniHTTP/miniHTMLe che presenti alcune caratteristiche particolari, qui di seguito elencate:

Il proxy accetta connessioni TCP dai client e instaura connessioni TCP con dei server.
Su ogni connessione accettata il proxy riceve una richiesta GET in formato miniHTTP, poi risponde con una risposta in formato miniHTTP, infine chiude la connessione.
Su ogni connessione TCP instaurata con un server, il proxy manda una richiesta in formato miniHTTP, poi riceve una risposta in formato miniHTTP, infine chiude la connessione.
Il protocollo miniHTTP è stato specificato precedentemente.

Il proxy miniHTTP può utilizzare richieste INF e richieste GET, queste ultime SOLO con l'opzione Range. In questa situazione il proxy per velocizzare il download può fare contemporaneamente tante richieste GET con Range per scaricare in parallelo più porzioni di file, in modo da ottenere il file più velocemente e poterlo inviare al client più velocemente.
Deve essere rispettato il seguente vincolo: il proxy non può utilizzare mai più di 5 connessioni col server contemporaneamente. Prima di utilizzarne una sesta deve chiudere uno dei socket TCP che attualmente lo connettono col server.
Volendo, il proxy può cominciare ad inviare il file al client anche prima di averlo ricevuto completamente.

Il proxy può mantenere in cache i files precedentemente ricevuti, ma li deve eliminare allo scadere del loro periodo di validità. Se i files sono ottenuti mediante più richieste Range, ciascuna parte richiesta potrebbe avere una scadenza diversa. La scadenza del file intero sarà la scadenza di una qualunque delle sue parti, a vostra scelta. Inoltre, all'avvio il proxy deve

eliminare tutti i file eventualmente già salvati in cache.

Gli eseguibili da utilizzare per questo progetto sono denominati Client1.exe e Server1.exe.

5. Tipologia di problemi sulla connessione

Primo tipo di problema della connessione: errore esplicito
Si ha un problema sulla connessione quando una system call (read, write, send, recv, etc.) applicata al socket di quella connessione restituisce un errore "grave" che rende non più utilizzabile quel socket.

Secondo tipo di problema della connessione: starvation
Si ha un problema sulla connessione anche quando un socket si aspetta di ricevere dai dati ma questi non arrivano anche se non vengono segnalati errori sulle system calls. Si può considerare che se per INACTIVITY_TIMEOUT_SECONDS=10 secondi non arrivano dati allora la connessione non è più utilizzabile.
Si ricordi che il server simula i problemi di rete qui sopra descritti.
Si ricordi inoltre che il server simula anche dei ritardi di rete, variabili nel tempo.
Si ricordi inoltre che anche il client simula dei ritardi di rete.

6. Importante: Ipotesi di Progetto.

Si assumono le seguenti ipotesi:

- La stringa che specifica il percorso di una risorsa non supera mai 2048 bytes.
- La dimensione di un file miniHTML non supera mai i 5000 bytes.
- In un file miniHTML non possono mai essere presenti più di 10 riferimenti REF.
- In un file miniHTML non possono mai essere presenti più di 10 riferimenti IDX+REF.
- Non possono esistere più di 5 connessioni contemporanee con un server. Prima di aprirne un'altra bisogna chiuderne una precedente.

7. Importante: Vincoli di Progetto.

Lo scopo del progetto è acquisire conoscenza delle tecniche di progettazione mediante I/O MULTIPLEXING e/o pthreads, e per questo motivo è vietato l'utilizzo di tecniche di progettazione basate su I/O asincrono e segnali. In particolare:

1) Non possono essere usate le signal(), sigvec(), sigaction(), e simili.
2) Non possono essere usate named pipe (dette anche FIFO), cioè quelle create con mknod(,,).
3) Non possono essere usati i socket unix named, cioè creati con socket(AF_UNIX, ,)
4) Possono essere usate le pipe, cioè quelle create con int pipe(int filedes[2]), ma bisogna tenere conto che le pipe non permettono di essere utilizzate mediante le invocazioni alle system call send e recv, e quindi non possono essere utilizzati i comportamenti definiti dai flags MSG_DONTWAIT, MSG_PEEK, etc.
5) Possono essere usati i socket unix anonimi, cioè quelli creati mediante la system call socketpair(AF_UNIX,SOCK_STREAM,0, ..), e puo' essere comodo perché su questi socket

possono essere invocate le primitive send e recv, e quindi possono essere utilizzati i comportamenti definiti dai flags MSG_DONTWAIT, MSG_PEEK, etc.
6) Possono essere usati pthread e processi ausiliari.

7) Ovviamente, **può essere utilizzato solo ed esclusivamente il linguaggio ANSI C**, e non possono essere utilizzate librerie altrui.

8. Fase di Test.

La valutazione verrà effettuata utilizzando una coppia di semplici applicazioni, un **Server** ed un **Client** il cui codice è messo a disposizione dal docente assieme alle specifiche ed ai files miniHTML usati per le prove. Verranno lanciati un server, il proxy realizzato dagli studenti ed uno o due client simultaneamente.
Gli eseguibili da utilizzare per il progetto numero 1 sono denominati Client1.exe e Server1.exe, mentre gli eseguibili da utilizzare per il progetto numero 2 sono denominati Client2.exe e Server2.exe.
Nel file all_files.tgz oltre ai sorgenti delle applicazioni sono contenuti anche alcuni file di esempio di tipo .mhtml utili a svolgere le prove, in quanto sono quelli utilizzati quando le applicazioni sono lanciate con i parametri di default. Si noti che, quando i client vengono lanciati con i parametri predefiniti, tra i file richiesti dal client ce n'è anche uno che non esiste; non è un errore, al contrario serve a verificare che il proxy realizzato sia in grado di gestire anche risposte di tipo 404 "file not found". All'atto della demo, il docente potrebbe chiedere di utilizzare file .mhtml diversi da quelli predefiniti.

Sequenza di "lancio" delle Applicazioni

La prima applicazione che deve essere lanciata e' il server, poi deve essere lanciato il proxy ed infine il o i client.

Parametri a riga di comando passati al Server

Nel momento in cui viene lanciato, il Server pretende in input 6 parametri, ad esempio:

```
./Server.exe  ./  127.0.0.1  55555  100  0.2  0.2
```

- Il primo parametro e' il percorso nel filesystem a partire dal quale il server cerca i files richiesti dal client.
- Il secondo è l'indirizzo IP locale su cui il server fa la bind.
- Il terzo è la porta TCP locale su cui il server fa la bind.
- Il quarto parametro e' il ritardo artificialmente introdotto dal server, espresso in msec.
- Il quinto parametro è la probabilità che accada un errore nella trasmissione di una risposta.
- Il sesto è la probabilità che accada uno stallo durante la trasmissione di una risposta.

Se al momento del lancio dell'eseguibile server non viene specificato nessun parametro, il Server usa i parametri di default, che sono quelli specificati nell'esempio.

Parametri a riga di comando passati al Client

Nel momento in cui viene lanciato, il Client pretende in input 5 o più parametri, ad esempio

./Client.exe 1 2 127.0.0.1 55554 mhttp://127.0.0.1:55555/1.mhtml
mhttp://127.0.0.1:55555/2bis.mhtmlI

I parametri dal quinto in poi sono gli indirizzi delle pagine miniHTML che devono essere reperite e visualizzate dal Client. Se una pagina visualizzata contiene dei link (le sequenze speciali IDX+REF) può essere permesso o no al client di seguire alcuni di questi link e quindi può essere permesso o no all'utente di specificare quali link seguire.

- Il primo parametro e' il seme per l'inizializzazione del generatore di numeri casuali.
- Il secondo parametro è il tipo di input ammesso per scegliere quali link seguire per navigare a partire dalle risorse visualizzate. Per default viene scelta la modalità automatica e casuale.
    - 0 significa che non si può seguire i link contenuti nelle pagine miniHTML e quindi non viene utilizzato nessun input dall'utente
    - 1 significa che è l'utente a decidere quali link seguire digitando su tastiera gli identificatori numerici IDX dei link da seguire.
    - 2 significa che la scelta è fatta in modo automatico e casuale dal Client stesso senza intervento dell'utente. E questa la modalità che verrà usata durante la demo del progetto.
- Il terzo parametro è l'indirizzo IP del proxy
- Il quarto parametro è la porta TCP ldel proxy.

Come detto, i parametri dal quinto in poi sono gli indirizzi delle risorse da reperire, ciascuno espresso nella forma mhttp://INDIRIZZOIP:PORTA/PERCORSOFILE.

Se al momento del lancio dell'eseguibile server non viene specificato nessun parametro, il Server usa i parametri di default, che sono quelli specificati nell'esempio.

Compilazione di Client e Server.

Esistono due diversi server da utilizzare, uno chiamato Server1.exe per il progetto 1, l'altro chiamato Server2.exe per il progetto 2. Analogamente, esistono due diversi client da utilizzare, uno chiamato Client1.exe per il progetto 1, l'altro chiamato Client2.exe per il progetto 2

La compilazione per ottenere gli eseguibili Client1.exe, Client2.exe , Server1.exe e Server2.exe viene effettuata utilizzando l'utility make ed il file Makefile contenuto nel file all_files.tgz assieme ai sorgenti. Questo all_files.tgz va scompattato in una directory utilizzando il comando "tar xvzf all_files.tgz", ed in quella directory deve essere lanciato il comando make, che provvederà a compilare e linkare i diversi moduli producendo i file Client1.exe, Client2.exe , Server1.exe e Server2.exe. Nel file all_files.tgz sono contenuti anche alcuni file di esempio di tipo .mhtml utili a svolgere le prove, in quanto sono quelli utilizzati quando le applicazioni sono lanciate con i parametri di default.

9. Valutazione.

La valutazione del progetto realizzato seguirà alcuni criteri e in particolare terrà conto di:

- Difficoltà del progetto: il progetto numero 1 (pre-fetching) è un po' più difficile del progetto numero due (download parallelo).
- impostazione del progetto,
- insieme delle funzionalità realizzate (ad es: pre-fetching anche di terzo livello oppure

solo di primo e secondo livello).

- performance in termini di:
  - tempo di download da parte del Client
  - affidabilità,
  - uso di CPU,
  - occupazione di memoria,
  - consumo di bandwidth,
- portabilità del codice realizzato (gestione di endianess e allineamento struct)
- chiarezza del codice realizzato,
- numero di componenti del gruppo (TASSATIVAMENTE NON SUPERIORE a 4 COMPONENTI).

E' comunque **indispensabile** che il progetto realizzato:

- **rispetti l'interfaccia definita,**
- **realizzi il corretto download dei files richiesti, se questi esistono sui server**.
- **Se un file richiesto non esiste sul server il server restituisce un messaggio 404, e il proxy deve correttamente segnalare al client la non esistenza mediante un messaggio 404**.