Relazione Homework 2: `System Programming`

Francesco Zompanti 2012601
Vladut Florian Stanciu 2109405
Luca Panetta 2136770
Corso di Sistemi Operativi 2

Obiettivo e contesto

Il progetto sviluppa un'architettura client-server in C per la trasmissione sicura di dati mediante cifratura simmetrica basata su operatore `XOR` a 64 bit. La comunicazione avviene su socket TCP, con gestione concorrente delle connessioni in ricezione lato server e parallelizzazione del processo di cifratura e decifratura tramite API POSIX `pthreads`.

Architettura del sistema

Il sistema è suddiviso nei seguenti moduli software:

- **Modulo Client**: acquisisce un file di input, lo segmenta in blocchi da 64 bit, applica la cifratura in parallelo e invia il payload cifrato al server remoto.
- **Modulo Server**: riceve il flusso cifrato, esegue la decifratura parallela e persiste il contenuto decifrato su file.
- **Modulo Utility**: raccoglie funzioni di supporto condivise da client e server (gestione segnali, log, parsing, validazione argomenti, sincronizzazione).

Protocolli e specifiche di trasmissione

Il client invia un buffer strutturato secondo il seguente schema:

- B: sequenza di blocchi cifrati da 64 bit (`uint64_t`).
- L: intero non segnato rappresentante la lunghezza effettiva in byte del file originale.
- K: chiave simmetrica a 64 bit utilizzata nella cifratura.

La cifratura utilizza l'operazione bitwise `XOR`:

$$C_i = P_i \oplus K$$

dove P_i è l'i-esimo blocco di testo in chiaro.

Pipeline del client

1. Parsing degli argomenti da riga di comando: file da cifrare, chiave, grado di parallelismo, IP/porta del server.
2. Allocazione e lettura del contenuto in memoria.

3. Suddivisione in blocchi di 64 bit e cifratura concorrente con `pthread_create`.
4. Composizione del messaggio e invio tramite `send()` su socket TCP.
5. Chiusura socket e deallocazione memoria.

Pipeline del server

1. Avvio in ascolto su una porta definita, con massimo l connessioni concorrenti gestite tramite `fork()` o `pthread`.
2. Ricezione del messaggio: lettura sequenziale dei blocchi cifrati, dimensione e chiave.
3. Decifratura concorrente con `pthread` (stessi algoritmi del client ma in senso inverso).
4. Scrittura del contenuto su file (nome generato dinamicamente in base a un prefisso e ID univoco).
5. Invio conferma (ACK) e chiusura della connessione.

Gestione della concorrenza e della sincronizzazione

- Il parallelismo viene implementato con `pthread_t`, assegnando a ciascun thread un intervallo di blocchi da cifrare/decifrare.
- Le strutture dati condivise sono protette mediante `pthread_mutex_t`.
- I segnali asincroni sono gestiti tramite `sigaction` e `sigset_t` per prevenire race condition e garantire una chiusura pulita del server.

Testing e validazione

Sono stati effettuati test su file di dimensioni diverse per verificare:

- Integrità della decifratura (verifica `diff` tra file originale e ricostruito).
- Stabilità del server sotto carico di connessioni concorrenti.
- Corretto smistamento del lavoro tra thread.

Conclusioni

Il progetto implementa un sistema affidabile e modulare per cifratura e trasmissione sicura di dati su rete locale, rispettando i principi di:

- Separazione delle responsabilità tra client, server e utility.
- Utilizzo corretto delle primitive di sincronizzazione.
- Parallelizzazione efficiente delle operazioni critiche (cifratura/decifratura).
- Robustezza del protocollo su TCP.