Università di Ferrara

Architettura di Reti

Lab 4

Carlo Giannelli
carlo.giannelli@unife.it
http://www.unife.it/scienze/informatica/insegnamenti/architettura-reti/
http://docente.unife.it/carlo.giannelli

Protobuf

- Installazione librerie
  - sudo apt install protobuf-c-compiler
  - sudo apt-get install libprotobuf-dev
  - sudo apt-get install libprotobuf-c-dev

- Step principali
  1. definire i tipi di messaggio "protocol buffer" in un file .proto
  2. compilare il file .proto col compilatore di protobuf per generare codice language-specific
     - "protoc-c --c_out=. amessage.proto" per generare "amessage.pb-c.c" e "amessage.pb-c.h"
  3. utilizzare il codice generato per serializzare e deserializzare
  4. compilare e linkare tutto assieme
     - gcc -c amessage.pb-c.c
     - gcc -o example.out example.c amessage.pb-c.c -L/usr/lib -lprotobuf-c

Radice Quadrata Remota

Si realizzi un'applicazione distribuita in C/Unix che, sfruttando la socket API, implementi un servizio di calcolo remoto della radice quadrata basato su autenticazione. Il Client deve presentare la seguente interfaccia:

rsqrt hostname porta username password valore

dove "hostname" è il nome dell'host dove risiede il Server, "porta" è il numero di porta a cui esso è associato, "username" è l'identificativo dell'utente, "password" è la password dell'utente e "valore" è il numero di cui si vuole calcolare la radice quadrata.

Il Client deve inviare al Server username, password e numero. Il Server verifica l'identità del cliente e (in caso di verifica positiva) risponde con due valori, uno per la parte intera e uno per la parte decimale (due cifre) della radice quadrata del numero inviato dal cliente.

Radice Quadrata Remota – cont.

Username, password e valore da Client a Server (e similmente parte intera e parte decimale da Server a Client) posso essere trasmesse, ad esempio:

1) **separatamente**, ad esempio username e password in pacchetti datagram distinti oppure tramite write successive
2) in un'unica **struct**
3) in un unico messaggio **protobuf**
4) …

Si ricorda: https://linux.die.net/man/3/sqrt

Dimensione del messaggio protobuf

In fase di deserializzazione dei messaggi protobuf è necessario specificare la dimensione del messaggio stesso.

Con **datagram socket** la dimensione del messaggio protobuf è dato dalla **dimensione del payload del pacchetto datagram** (per ipotesi, un solo messaggio protobuf in ciascun pacchetto datagram)

Con **stream socket**

1. ipotesi semplificativa: la read legge l'intero messaggio protobuf (tutto e solo) e quindi la sua dimensione è data dal **valore restituito dalla read**
2. sia invia prima **un unico byte con la dimensione del messaggio protobuf** e poi il messaggio protobuf vero e proptio

Versioni da implementare (1)

Si richiede di implementare l'interazione Client/Server con **stream socket** nelle seguenti modalità:

1. **struct**: read e write di struct
2. **protobuf**: read e write di messaggi protobuf
3. **dim+protobuf**: read e write di un byte per dimensione, poi messaggio protobuf vero e proprio

Notare che:

- struct: OK message boundary (ma avendo cura di leggere proprio sizeof byte), KO eterogeneità
- protobuf: KO message boundary, OK eterogeneità
- dim+protobuf: OK message boundary, OK eterogeneità (se dim come singolo byte sempre in complemento a 2)

Versioni da implementare (2)

Suggerimenti:

1. Tralasciare la parte di verifica delle credenziali.
2. Nella versione con struct, valore come float e username/password come array di char di dimensione al più 15 char compreso il terminatore di stringa.
3. Nella versione con protobuf concentrarsi innanzitutto sull'invio della richiesta, con e senza dimensione. Solo in un secondo momento estendere la soluzione con l'invio della risposta.