Documentazione Della Repository GitHub

Corso di Programmazione e Sicurezza delle Reti

Repository di Simone Mattioli

Università degli Studi di Verona
Docente: Prof. Davide Quaglia

Studente: Dott. Mattioli Simone
GitHub Repository

Anno Accademico 2024/2025

Indice

1 Introduzione

Questo documento fornisce una documentazione completa del repository GitHub **"Corso-Programmazione-Sicurezza-delle-Reti"** sviluppato da Simone Mattioli per il corso di Programmazione e Sicurezza delle Reti presso l'Università degli Studi di Verona, tenuto dal Prof. Davide Quaglia.

Il repository rappresenta una collezione organizzata di materiali didattici, codice sorgente, appunti e risorse pratiche sviluppate durante il corso nell'anno accademico 2024/2025.

1.1 URL del Repository

https://github.com/simo-hue/Corso-Programmazione-Sicurezza-delle-Reti

2 Obiettivi del Repository

Il progetto è stato creato con i seguenti obiettivi principali:

- **Esempi Pratici:** Fornire implementazioni concrete di programmazione di rete utilizzando diversi linguaggi (C, Java, HTML/JavaScript)
- **Documentazione di Laboratorio:** Documentare procedure pratiche per l'utilizzo di strumenti come Wireshark, MQTT, WebSocket e REST
- **Materiale di Studio:** Conservare materiali didattici, appunti personali e risorse per la preparazione all'esame
- **Riferimento Completo:** Creare un punto di riferimento organizzato per tutti gli argomenti del corso

3 Requisiti di Sistema

Per utilizzare completamente i materiali presenti nel repository, sono necessari i seguenti software:

Software	**Versione Minima**	**Note**
GCC / Clang	9.0	Su Windows è consigliato utilizzare Cygwin o WSL
Make	–	Alcuni esempi includono Makefile per la compilazione automatica
Wireshark	4.x	Strumento fondamentale per l'analisi del traffico di rete
Mosquitto	2.x	Necessario solamente per le esercitazioni MQTT
Python 3	3.10	Per gli script di supporto opzionali

Tabella 1: Requisiti software minimi

Nota Importante

Per istruzioni dettagliate sull'installazione dell'ambiente C su Linux, macOS e Windows, consultare il file `Impostazione-PC-applicazioni-socket.pdf` presente nel repository.

4 Struttura del Repository

Il repository è organizzato in cartelle tematiche per facilitare la navigazione e l'utilizzo dei materiali:

Cartella	Descrizione
`DOMANDE ESAME/`	Raccolta di domande degli esami orali delle sessioni precedenti
`FILE FORNITI DAL PROF/`	File originali forniti dal docente tramite la piattaforma Moodle del corso (senza soluzioni)
`MIEI ESERCIZI/`	Soluzioni personali sviluppate dall'autore per gli esercizi di laboratorio
`PER ESAME/`	Soluzioni commentate e materiale ottimizzato per la preparazione all'esame
`APPUNTI GOODNOTES/`	Appunti manoscritti in formato PDF esportati da GoodNotes (in arrivo)
`LUCIDI/`	Slide ufficiali del corso in formato PDF
`.vscode/`	Configurazioni e task per Visual Studio Code

Tabella 2: Struttura delle cartelle del repository

5 Guida Rapida all'Utilizzo

5.1 Compilazione degli Esempi

Per compilare ed eseguire i principali esempi di programmazione di rete:

5.1.1 Esempi UDP

```
# Compilazione del server UDP
gcc network.c serverUDP.c -o serverUDP -lpthread

# Compilazione del client UDP
gcc network.c clientUDP.c -o clientUDP -lpthread
```

Listing 1: Compilazione server/client UDP

5.1.2 Esempi TCP

```
# Compilazione del server TCP
gcc network.c serverTCP.c -o serverTCP -lpthread

# Compilazione del client TCP
gcc network.c clientTCP.c -o clientTCP -lpthread
```

Listing 2: Compilazione server/client TCP

5.2 Esecuzione e Test

5.2.1 Test Socket UDP/TCP

1. Aprire due terminali separati
2. Nel primo terminale, compilare ed eseguire il server:

```
./serverUDP  # oppure ./serverTCP

```

3. Nel secondo terminale, compilare ed eseguire il client:

```
./clientUDP  # oppure ./clientTCP

```

4. Utilizzare Wireshark per filtrare per porta o protocollo e osservare i pacchetti scambiati

5.2.2 Server HTTP

```
# Compilazione
gcc serverHTTP.c -o serverHTTP -lpthread

# Esecuzione sulla porta 8000
./serverHTTP 8000

# Apertura nel browser
xdg-open "http://127.0.0.1:8000/"
```

Listing 3: Avvio server HTTP

5.2.3 WebSocket

Per testare le comunicazioni WebSocket:

1. Avviare il server (file `websocket_server.c` o equivalente)
2. Aprire il file `client.html` in un browser moderno
3. Utilizzare l'interfaccia per inviare messaggi broadcast a tutti i client connessi

5.2.4 API REST

```
# Compilazione e avvio del server REST
gcc serverHTTP-REST.c -o rest_server -lpthread
./rest_server 8080

# Test con curl
curl "http://127.0.0.1:8080/api/somma?x=5&y=7"
```

Listing 4: Test API REST

5.2.5 MQTT

```
# Avvio del broker Mosquitto
mosquitto -v

# Esecuzione del publisher
./publisher

# Esecuzione del subscriber
./subscriber
```

Listing 5: Test protocollo MQTT

6 Analisi del Traffico con Wireshark

Il repository include file di cattura (`.pcapng`) per l'analisi del traffico di rete. Per utilizzarli efficacemente:

1. Aprire il file `.pcapng` corrispondente all'esercizio
2. Utilizzare la funzione "Follow TCP/UDP Stream" per ricostruire il dialogo completo
3. Analizzare i numeri di sequenza, i flag, l'handshake e le eventuali ritrasmissioni
4. Confrontare il comportamento osservato con quello teorico atteso

7 Documentazione Teorica

7.1 Slide del Corso

Nella cartella `LUCIDI/` sono disponibili le slide ufficiali del corso che coprono i seguenti argomenti:

- Programmazione socket
- Protocollo HTTP
- Tecnologie WebSocket
- Architetture REST
- Protocollo MQTT
- Architettura TCP/IP

7.2 Appunti Personali

Gli appunti personali dell'autore sono disponibili in `APPUNTI GOODNOTES/`, mentre il file `2_ripasso-reti.pdf` fornisce un riassunto rapido e conciso dei concetti fondamentali di rete in vista dell'esame.

8 Roadmap e Sviluppi Futuri

8.1 TODO List

Il repository è in continua evoluzione. Gli sviluppi futuri pianificati includono:

- **Automazione Build:** Aggiunta di `Makefile` e script CI con GitHub Actions
- **Test di Integrazione:** Automatizzazione dei test di integrazione per i servizi REST
- **Documentazione:** Trascrizione degli appunti GoodNotes in formato Markdown per una migliore accessibilità
- **Espansione Contenuti:** Aggiunta di ulteriori esempi pratici e casi d'uso avanzati

9 Contributi e Collaborazione

9.1 Linee Guida per i Contributi

Il repository accoglie contributi dalla comunità. Per mantenere la coerenza del progetto:

1. Utilizzare branch descrittivi (es. `feature/nuova-funzionalità`, `fix/correzione-bug`)
2. Seguire la convenzione di stile `clang-format` fornita nel file `.clang-format` (work-in-progress)
3. Aggiornare la documentazione quando si modificano o aggiungono funzionalità
4. Testare accuratamente le modifiche prima di sottomettere una pull-request

9.2 Processo di Contribuzione

1. Fork del repository
2. Creazione di un branch per la propria modifica
3. Implementazione delle modifiche
4. Test delle funzionalità
5. Sottomissione di una pull-request con descrizione dettagliata

10 Licenza e Utilizzo

Importante - Licenza

Il materiale è fornito esclusivamente per uso didattico. La licenza è attualmente in fase di definizione. È necessario contattare l'autore prima di ridistribuire pubblicamente il contenuto del repository.

11 Crediti e Riconoscimenti

11.1 Autore Principale

Simone Mattioli

- Studente di Informatica, Università degli Studi di Verona
- Sviluppatore del repository
- Anno di corso: 2024/2025

11.2 Docente del Corso

Prof. Davide Quaglia

- Docente del corso di Programmazione e Sicurezza delle Reti
- Università degli Studi di Verona
- Supervisione accademica del progetto

12 Link Utili e Risorse Aggiuntive

12.1 Repository e Codice

- Repository GitHub Principale
- Profilo GitHub dell'autore
- Linkedin dell'autore

12.2 Strumenti e Software

- Wireshark - Analizzatore di protocolli di rete
- Eclipse Mosquitto - Broker MQTT
- GCC - GNU Compiler Collection
- Visual Studio Code

12.3 Documentazione di Riferimento

- RFC 793 - Transmission Control Protocol
- RFC 768 - User Datagram Protocol
- RFC 2616 - Hypertext Transfer Protocol HTTP/1.1
- RFC 6455 - The WebSocket Protocol
- RFC 3986 - Uniform Resource Identifier (URI)

13 Conclusioni

Il repository "Corso-Programmazione-Sicurezza-delle-Reti" rappresenta una risorsa completa e ben organizzata per lo studio della programmazione di rete e dei protocolli di sicurezza. La struttura modulare, la documentazione dettagliata e la varietà di esempi pratici lo rendono uno strumento prezioso sia per gli studenti del corso che per chiunque voglia approfondire questi argomenti.

La continua evoluzione del progetto, testimoniata dalla roadmap di sviluppi futuri e dall'apertura ai contributi della comunità, garantisce che questa risorsa rimanga aggiornata e sempre più completa nel tempo.

L'approccio pratico, combinato con una solida base teorica fornita dalle slide del corso e dagli appunti personali, offre un percorso di apprendimento completo che spazia dalla teoria all'implementazione pratica, dall'analisi del traffico di rete alla comprensione dei protocolli moderni.

Ultima revisione: 22 luglio 2025