Esercizio: Progettazione di un Protocollo per un Ristorante

Contesto
Un ristorante desidera implementare un sistema per la gestione degli ordini tramite un servizio di rete. L'obiettivo è consentire ai camerieri di inviare ordini in modo semplice ed efficace al sistema centrale del ristorante, evitando l'uso del protocollo HTTP. Questa scelta permette al ristorante di sviluppare un protocollo privato, che garantisce una comunicazione più diretta, controllata e ottimizzata per le esigenze interne.

Operazioni del Protocollo:

inserire anche cancellazione ordine?

1. Invio dell'"ordine: I camerieri possono inviare un "ordine che include una lista di piatti e bevande scelti dai clienti.
2. Richiesta dello Stato dell'"ordine: I camerieri possono interrogare il sistema per ottenere informazioni sullo stato di un "ordine specifico (ad esempio, se è in preparazione o già servito).
3. "cancellazione dell'"ordine: I camerieri hanno la possibilità di annullare un "ordine già inviato in caso di modifiche dell'ultimo minuto.

Obiettivo dell'Esercizio
Progettate un protocollo applicativo che soddisfi le operazioni sopra indicate. Si progetti il protocollo applicativo secondo lo schema RFC fornito dai docenti (vedere template con elenco aspetti da considerare)

Soluzione

Restaurant "Order Management Protocol (ROMP)

Stato del documento

Stato attuale del protocollo: **draft**. Questo protocollo è ancora in fase di definizione e potrebbe subire modifiche in base alle esigenze specifiche del ristorante e agli input degli sviluppatori e degli utenti.

Riepilogo

Il protocollo ROMP è progettato per facilitare la comunicazione tra i camerieri e il sistema centrale del ristorante. Permette l'invio di ordini, la richiesta dello stato degli ordini e la "cancellazione degli ordini.

Panoramica del Protocollo

Il protocollo è progettato per garantire una comunicazione efficiente tra i camerieri e il sistema centrale del ristorante. ROMP utilizza un approccio client-server in cui i camerieri fungono da client e il sistema centrale del ristorante funge da server. La comunicazione avviene tramite messaggi strutturati che seguono un formato definito.

Il server ROMP ascolta sulla porta **9090**. Questa porta è stata scelta per evitare conflitti con altre applicazioni comuni e per garantire un'adeguata sicurezza e gestione del traffico.

Ogni messaggio sarà costituito da una stringa formattata nel seguente modo:

<Tipologia_Messaggio>||[]|...|[]

I messaggi inviati dal client al server sono di tre tipologie principali:

1. **ORD**: Utilizzato per inviare un nuovo ordine al server. Questo messaggio include la lista dei piatti ordinati e la lista delle bevande. Una volta ricevuto, il server elabora l'ordine e invia una conferma (ACK) al cameriere, includendo l'id dell'ordine.
2. **STATUS**: Permette ai camerieri di verificare lo stato attuale di un ordine. Invia un ID d'ordine e il server restituisce lo stato attuale dell'ordine, come "In preparazione" o "Servito".
3. **CANCEL**: Consente di annullare un ordine già inviato. Il cameriere invia l'ID dell'ordine da cancellare e il server conferma la cancellazione con un messaggio di risposta.

Il server ROMP ascolta sulla porta **9090**. Questa porta è stata scelta per evitare conflitti con altre applicazioni comuni e per garantire un'adeguata sicurezza e gestione del traffico.

Esempio di Interazione

```
1. Cameriere invia un'ordine:
   • Messaggio: "ORD|Pizza Margherita, Lasagna|Acqua, Vino Rosso"
   • Risposta del Server: "ACK|12345""

2. Cameriere richiede stato dell'ordine:
   • Messaggio: "STATUS|12345"
   • Risposta del Server:"STATUS|12345|In preparazione"

3. Cameriere cancella l'ordine:
   • Messaggio:"CANCEL|12345"
   • Risposta del Server:"CANCEL_ACK|12345"
```

L'id ordine fornisce un indirizzamento a livello applicativo.

Il protocollo non è connesso, in quanto il client apre una connessione TCP per tutta la durata dell'interazione con il server (nota: una versione connessa sarebbe più efficiente, perché?)

Il protocollo offre un servizio di conferma per i messaggi ORD e CANCEL.

Il protocollo prevede la gestione degli errori tramite messaggi di errore in caso di problemi di comunicazione. Ogni messaggio contiene un codice di stato che consente di gestire gli errori. Il server invia messaggi di errore specifici in caso di problemi durante l'elaborazione delle richieste.

Il protocollo non implementa:

- frammentazione, se serve frammenterà il protocollo di trasporto
- servizi di trasmissione e controllo del flusso sono demandati al protocollo di trasporto
- incapsulamento in quanto ogni messaggio consiste di solo payload

7. Controllo del Flusso
   Non è previsto controllo del flusso a livello applicativo; TCP gestirà questo aspetto a livello di trasporto.
8. Multiplexing e Demultiplexing
   Non è previsto multiplexing a livello applicativo
9. Servizi di Trasmissione
   Non vengono forniti servizi aggiuntivi come sicurezza o qualità del servizio.

Comandi del server

I server ROMP risponde ai comandi del client con i seguenti messaggi:

1. **Acknowledge (ACK)**: Conferma la ricezione di un'ordine.
   • **Formato**: "ACK|<ID_ordine>"
   • **Esempio**: "ACK|12345"
2. **Errore (ERR)**: Indica un errore nell'elaborazione della richiesta.
   • **Formato**: "ERR|<Codice_Errore>|<Messaggio_Errore>"
   • **Esempio**: "ERR|404|ordine non valido"
3. **Stato dell'"ordine ("STATUS)**: Fornisce lo stato attuale di un "ordine.
   • **Formato**: "STATUS|<ID_ordine>|
   • **Esempio**: "STATUS|12345"|"In preparazione"
4. **cancellazione dell'"ordine ("CANCEL_ACK)**: Conferma la cancellazione di un ordine.
   • **Formato**: "CANCEL_ACK|12345"
   • **Esempio**: "CANCEL_ACK|12345"

Messaggi di Errore Specifici: Il protocollo ROMP prevede i seguenti messaggi di errore, che possono essere inviati dal server al client in caso di problemi:

Formato generale: "ERR|<Codice_Errore>|<Messaggio_Errore>"

1. **Errore di Invio Ordine**: Indica che l'ordine inviato è malformato o non valido.
   • **Esempio**: "ERR|400|Bad request - Formato ordine non valido"
2. **Errore Ordine Non Trovato**: Indica che l'ordine specificato non è stato trovato nel sistema.
   • **Esempio**: "ERR|404|Ordine non trovato"
3. **Errore cancellazione**: Indica che si è tentato di cancellare un ordine già servito o inesistente.
   • **Esempio**: "ERR|410|cancellazione non valida"
4. **Errore Stato Non Disponibile**: Indica che non è possibile recuperare lo stato di un ordine.
   • **Esempio**: "ERR|414|Stato dell'ordine non disponibile"
5. **Errore Numero Ordini elevato**: Indica che non è possibile creare un nuovo ordine.
   • **Esempio**: "ERR|424|Impossibile gestire nuovi ordini"

Comandi del Client

I comandi inviati dal client al server seguono i seguenti formati:

1. **Invio dell'"ordine (ORD)**:
   • **Formato**: "ORD|<Lista_Di_Piatti>|<Lista_Di_Bevande>"
   • **Esempio**: "ORD|Pizza Margherita, Lasagna|Acqua, Vino Rosso
2. **Richiesta dello Stato dell'"ordine (STATUS)**:
   • **Formato**: "STATUS|<ID_ordine>"
   • **Esempio**: "STATUS|12345"
3. **Cancellazione dell'"ordine (CANCEL)**:
   • **Formato**: "CANCEL|<ID_ordine>"
   • **Esempio**: "CANCEL|12345"
4. **Chiusura connessione (QUIT)**:
   • **Formato**: "QUIT"
   • **Esempio**: "QUIT"

Sequenza Temporale

Esempio 1
title Interazione Normale: Invio di un Ordine
participant Cameriere
participant Server
Cameriere->>Server: "ORD|12345|Pizza Margherita, Lasagna|Acqua, Vino Rosso"
Server->>Cameriere: "ACK|12345"
Cameriere->>Server: "STATUS|12345"
Server->>Cameriere: "STATUS|12345|In preparazione"
Cameriere->>Server: "CANCEL|12345"
Server->>Cameriere: "CANCEL_ACK|12345"

Esempio 2
title Interazione con Errore: Ordine Non Valido
participant Cameriere
participant Server
Cameriere->>Server: "STATUS|99999"
Server->>Cameriere: "ERR|404|Ordine non trovato"

Esempio 3
title Interazione con Errore: Errore di Ordinazione
participant Cameriere
participant Server

Cameriere->>Server: "ORD|12345|Pasta|Acqua"
Server->>Cameriere: "ERR|400|ID ordine già esistente"