Esame di Programmazione - Simulatore di Rete "NetManage"

Obiettivo: Si richiede la progettazione e implementazione di un sistema per la simulazione di un'architettura di rete aziendale. Il sistema deve gestire nodi (device), connessioni attive e traffico dati.

1. Classi da Implementare

A. Gerarchia dei Nodi (Device)

- **Nodo (Classe Base):** Caratterizzato da:
  - indirizzoIP (stringa, es: "192.168.1.10").
  - macAddress (stringa univoca).
  - stato (boolean: *Online/Offline*).
- **Client (Derivata da Nodo):** Rappresenta un pc/utente. Caratterizzato da:
  - livelloSicurezza (int): Indica i permessi (1 = Guest, 5 = Admin).
- **Server (Derivata da Nodo):** Rappresenta una macchina che eroga servizi. Caratterizzato da:
  - capacitàMassima (int): Numero massimo di connessioni simultanee gestibili.
  - connessioniAttuali (int): Contatore delle connessioni attive.
  - tipoServer (string): Può essere "WEB" o "DATABASE".

B. Classe Pacchetto (Dati)

- Rappresenta l'unità di informazione scambiata.
- Attributi: sorgente (IP), destinazione (IP), dimensione (in MB).

C. Classe Connessione

- Rappresenta un canale attivo tra un Client e un Server.
- Deve memorizzare puntatori/riferimenti ai due nodi coinvolti e la durata della sessione (in millisecondi).

2. Classe Gestore: NetworkController

Questa è la classe centrale (come la "Biblioteca" o l'"Agenzia").

- Gestisce un **array/vector di Nodi** (tutti i device della rete).
- Gestisce un **array/vector di Connessioni** attive.

Funzionalità Richieste:

1. **addNode(Nodo n)**: Aggiunge un device alla rete.
2. **startConnection(Client c, Server s)**: Tenta di stabilire una connessione.

- **Vincolo 1 (Stato):** Entrambi i nodi devono essere *Online*.
- **Vincolo 2 (Capacità):** Il Server non deve aver raggiunto la capacitàMassima.
- **Vincolo 3 (Sicurezza - *Difficile*):**
  - Se il server è di tipo "DATABASE", il Client deve avere un livelloSicurezza >= 4 (Admin).
  - Se il server è "WEB", tutti possono accedere.
- *Se i vincoli sono soddisfatti, incrementa le connessioni del server e registra la connessione.*

3. **stopConnection(Client c)**: Termina la connessione del client, liberando lo "slot" sul server (decrementa connessioniAttuali).
4. **loadBalancer(String tipoServer)**:
   - Il metodo deve cercare tra tutti i server di quel tipo (es. tutti i "WEB") e restituire quello con il **carico minore** (ovvero il minor numero di connessioni attive al momento).
5. **sendPacket(Client c, Pacchetto p)**:
   - Il client può inviare un pacchetto solo se ha una connessione attiva.

3. Main di Test

Nel main, simulare il seguente scenario:

1. Creare il NetworkController.
2. Inserire:
   - 2 Server "WEB" (Capacità: 2 ciascuno).
   - 1 Server "DATABASE" (Capacità: 5).
   - 3 Client (2 Guest, 1 Admin).
3. Tentare di connettere un Guest al Database (**Deve fallire** per vincolo sicurezza).
4. Connettere l'Admin al Database (**Deve riuscire**).
5. Simulare un picco di traffico: Connettere molteplici client al primo Server WEB fino a saturarlo.
6. Utilizzare il metodo loadBalancer per trovare un server libero per un nuovo client.
7. Stampare la lista delle connessioni attive con i dettagli (IP Client -> IP Server).