Domanda 1 (fino a 5 punti)

Si descrivano l'architettura generale ed i principali protocolli (SMTP, POP3, IMAP) del servizio di posta elettronica.

Domanda 2 (fino a 5 punti)

Si descrivano gli attacchi su rete di tipo SYN flooding e UDP flooding.

Esercizio 1 (fino a 5 punti)

Si realizzi un Web Service che permette di gestire alcuni dati sulle vendite di una catena di aziende vinicole. In particolare, il servizio espone:

1. un metodo per registrare le vendite di un'azienda, che riceve l'id dell'azienda, il nome del vino, la quantità venduta e l'importo incassato per la vendita;
2. un metodo che, dato l'id di un'azienda, restituisce un oggetto contenente il nome del vino che ha realizzato i maggiori incassi complessivi, la quantità venduta e l'importo totale incassato;

come specificato nel file WSDL allegato. Si implementi in Java una classe che implementa il servizio.

```
<wsdl:types>
<schema targetNamespace="http://aziende">

   <wsdl:types>
      <schema targetNamespace="http://DefaultNamespace">
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>

        <complexType name="IncassoProdotto">
            <sequence>
              <element name="nomeVino" type="xsd:string"/>
              <element name="quantita" type="xsd:integer"/>
              <element name="importo" type="xsd:double"/>
           </sequence>
        </complexType>
      <schema>
   <wsdl:types>

   <wsdl:message name="VenditaRequest">
      <wsdl:part name="idAzienda" type=”xsd:integer"/>
      <wsdl:part name="nomeVino" type="xsd:string"/>
      <wsdl:part name="quantita" type="xsd:integer"/>
      <wsdl:part name="importo" type="xsd:double"/>
   </message>


   <wsdl:message name="VenditaResponse">
   </message>

   <wsdl:message name="MaggioreIncassoRequest">
      <wsdl:part name="idAzienda" type=" xsd:integer"/>
   </message>


   <wsdl:message name="MaggioreIncassoResponse">
      <wsdl:part name="incasso" type="impl:IncassoProdotto"/>
   </message>

   <wsdl:portType name="AziendaService">
      <wsdl:operation name=”Vendita” parameterOrder=”idAzienda nomeVino quantita importo">
         <wsdl:input message=”impl:VenditaRequest” name=”VenditaRequest” />
         <wsdl:output message=”impl:VenditaResponse” name=”VenditaResponse” />
      </operation>

      <wsdl:operation name="MaggioreIncasso" parameterOrder="idAzienda">
         <wsdl:input message="impl:MaggioreIncassoRequest" name=”MaggioreIncassoRequest” />
         <wsdl:output message="impl:MaggioreIncassoResponse" name=”MaggioreIncassoResponse” />
      </operation>
  </wsdl:portType>
 <wsdl:binding ...>    ...
```

Esercizio 2

Si deve realizzare in Java un'applicazione di rete per gestire delle aste online. Ogni asta è identificata da un ID univoco ed è caratterizzata dal nome del prodotto da vendere (es. smartphone modello XYZ). In particolare, il sistema è composto da:

- *n* nodi *Client,* che inviano richieste al server;
- 1 nodo *Server,* che gestisce le richieste dei client.

Il sistema funziona nel seguente modo:

- Quando il Server ha un prodotto da mettere all'asta, apre un server socket su una porta **P**, scelta a caso tra 30000 e 40000. Il server socket rimane aperto per 60 minuti, trascorsi i quali si chiude. Il Server, una volta aperta l'asta, invia un messaggio contente l'ID dell'asta, la porta **P** e il nome del prodotto (stringa di lunghezza max 200 caratteri) sul gruppo **multicast** caratterizzato dalla porta **UDP 5000** e dall'indirizzo **230.0.0.1**.
- I client ricevono la notizia dell'apertura asta sul gruppo multicast. Ogni client che intende partecipare all'asta, quindi, apre una connessione socket **TCP sulla porta P** e invia al server un oggetto **Offerta** contente il codice fiscale del cliente, l'ID dell'asta e l'importo di denaro offerto. Il Server risponde con un booleano (true/false) che indica l'accettazione o il rifiuto dell'offerta.
- Allo scadere di ogni asta, il Server invia un messaggio sulla porta **UDP 4000** del cliente che ha offerto il prezzo più alto per quell'asta.

Il Server deve essere in grado di gestire più aste contemporaneamente.

Si realizzino le classi che implementino le funzionalità sopra descritte. Inoltre, si realizzino due main: 1) il primo main crea e avvia il Server con 2 o più aste attive; 2) il secondo main crea e avvia un Client invia al Server una richiesta di partecipazione ad un'asta e si mette in attesa dell'esito.