Denial of Service (DoS) Attack

Definition:
A DoS attack tries to make a **computer, network, or website** unavailable to users by flooding it with fake traffic or requests.

Key Point:
DoS attacks don't usually steal data but cause **system failure**, **crashes**, or **slowness**, wasting time and money.

Types of DoS Attacks:

1. **Buffer Overflow Attack:**
   - Sends more data than a program can handle.
   - Causes the system to crash or behave abnormally.
2. **SYN Attack:**
   - Abuses the **TCP handshake**.
   - Sends many connection requests but **doesn't respond to replies**, blocking real users.
3. **Teardrop Attack:**
   - Sends broken IP packets with wrong sequence.
   - Confuses the system, leading to a **crash**.
4. **Smurf Attack:**
   - Sends ping requests using **IP spoofing**.
   - Floods the victim with replies, making it **unable to respond to real users**.

Prevention Methods:

- Use **firewalls**, **access control lists**, and **deep packet inspection**.
- Configure routers and switches correctly.

RTP (Real-time Transport Protocol)

Definition:
RTP is a protocol used to deliver **real-time data** like **audio and video** over the internet.

Developed by:
Audio Video Transport Working Group (Published in 1996, RFC 1889)

Where RTP is used:

- Voice and video calls (VoIP)
- Teleconferencing
- Online streaming (TV, video apps)
- Web-based push-to-talk

How RTP Works:

- Works with **RTCP (RTP Control Protocol)**.
- RTP **sends the media data** (audio/video).
- RTCP **monitors quality**, **syncs streams**, and **sends feedback**.
- RTP uses **even port numbers**, RTCP uses the **next odd port**.
- Common RTP port: **UDP 7000**, RTCP: **UDP 7001**

Bonus Tip:
RTP doesn't guarantee delivery. It focuses on **timing and order** of media packets, not error correction.

a) What is a Firewall? What are its basic needs?

- A **firewall** is a security device/software placed **between trusted and untrusted networks**.
- It **monitors and filters** incoming and outgoing network traffic.
- Only **trusted data** is allowed, based on the **organization's security policy**.

Basic Need:
To **protect the internal network** from external threats like hackers, malware, etc.

---

b) Types of Firewalls

1. **Packet Filtering Firewall**
2. **Circuit-Level Gateway / Proxy**
3. **Application-Level Proxy**
4. **Stateful Multilayer Inspection Firewall**

---

c) What is Packet Filtering?

- A **packet filtering firewall** checks each packet **individually** at the **network layer (IP/TCP)**.
- It allows or blocks packets based on IP address, port number, and protocol.

Limitations:

1. **TCP Limitations** – Can only filter on port numbers, not on the content or connection state.
2. **UDP Limitations** – Cannot confirm if UDP traffic is part of a valid session. May block valid data or allow harmful packets.

---

d) What is the Purpose of a Proxy?

- A **proxy** acts as a **middleman server** between the user and the internet.
- All private network users go through it to access the web.
- It can **cache data**, **filter content**, and **restrict access** to certain websites.

Used for:

- Improving speed (caching)
- Blocking harmful or unwanted content (e.g., porn sites, hacking)
- Controlling employee internet usage

---

e) Functions of NAT (Network Address Translation)

- NAT **translates private IP addresses** into **public IP addresses** (and vice versa).
- It allows **multiple devices** on a local network to **share one public IP**.

Benefits:

- Improves **security** by hiding internal IPs
- **Extends IPv4 life** (reduces need for many public IPs)
- Supports **migration between IP versions** (IPv4 ↔ IPv6)

Quick Revision Table:

Concept	Keyword	Function
Firewall	Security filter	Blocks/Allows traffic
Types of Firewall	4 Types	Packet, Circuit, App Proxy, Stateful
Packet Filtering	IP/TCP layer	One packet at a time
Proxy	Middle Server	Filters content, caches data
NAT	IP Translator	Converts internal ↔ public IP

Multicast Addressing & Transport Gateway (SOCKS Server)

Multicast Addressing (Simple Explanation)

- **Multicast** means sending data to **a group of computers** instead of just one.
- These computers **may or may not be on the same physical network**.
- Multicast addresses fall in **Class D IPv4 range**:
  224.0.0.0 to 239.255.255.255
- It is mainly used for **streaming, online games, and group communication**.

Transport Gateway (SOCKS Server)

A **gateway** helps **transfer data between different networks or protocols**.

- **Transport Gateway** (like SOCKS) works at **Layer 4 (Transport layer)**.
- It supports:
  - **TCP connections**
  - **Authentication, encryption, and access control**
- **It is not transparent**, meaning the client must know and connect to it directly.

Working of SOCKS Transport Gateway (Simple Steps)

1. **User A** wants to connect to **User B** (destination: port 80).
2. A connects to the **SOCKS server** at port **1080**.
3. A sends request to SOCKS server to open connection to B.
4. **SOCKS server checks the request**, and if allowed:
   - Opens a **new TCP connection** to B on port 80.

    - Informs A that connection was successful.
5. **Data is relayed** between A and B via SOCKS server.
    - SOCKS server keeps **two separate TCP connections**:
        - A ↔ SOCKS (connection 1)
        - SOCKS ↔ B (connection 2)

Diagram Summary (Text Version):

```pgsql
A                       SOCKS Server                    B
|----- 1080 SYN ----->|                                 |
|<----- SYN ACK ------|                                 |
|------ ACK --------->|                                 |
|---- Connection Req (to B:80) ----->                   |
|                    |----- 80 SYN ------------------->|
|                    |<---- SYN ACK -------------------|
|                    |----- ACK ---------------------->|
|<-------- OK --------|                                 |
|------ data -------->|                                 |
|                    |---- relay to B:80 ------------->|
```

Key Points to Memorize:

Term	Meaning
Multicast	One-to-many data delivery
Class D Range	224.0.0.0 – 239.255.255.255
SOCKS Server	A proxy server working at TCP level
Port 1080	Default SOCKS port
Layer 4 Device	Works at Transport Layer (TCP/UDP)
Two TCP Connections	A–SOCKS and SOCKS–B, separate ACK/SEQ numbers

Serialization in Java

Serialization is converting an object into a byte stream (wire format) to **save** or **transfer** it.

- Useful in **distributed systems** (multiple JVMs) and for **saving object states**.
- Java provides two **marker interfaces**:
    - Serializable
    - Externalizable (both in java.io package)
- Used in **RMI**, **HttpSession**, etc., where objects are passed over the network.

RMI (Remote Method Invocation)

- RMI is used to create **distributed applications** in Java.
- It allows one object to call methods on another object **in a different JVM**.
- Uses **stub** and **skeleton** for communication.

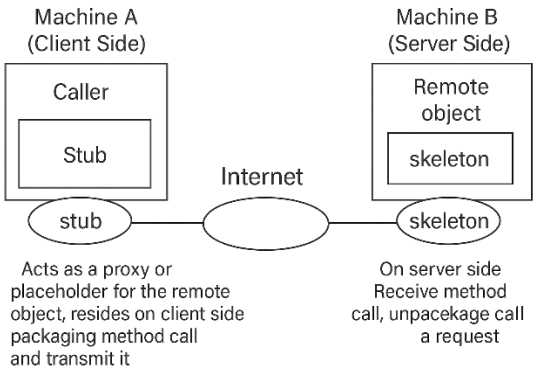
Stub (Client Side)

- Stub acts as a **gateway** on the **client side**.
- It represents the **remote object** and performs:
    1. Connects to remote JVM
    2. Sends method call and parameters (marshalling)
    3. Waits for response
    4. Reads result (unmarshalling)
    5. Returns result to caller

Skeleton (Server Side)

- Skeleton is the **gateway on the server side**.
- It handles **incoming requests** from the stub.
- Tasks performed by skeleton:
    1. Reads method parameters
    2. Calls the actual remote method
    3. Sends back the result (marshalling)

🩹 **Note:** From **Java 2**, skeleton is no longer needed — the **stub handles everything**.



Remote Interface in RMI

- All remote interfaces must **extend** `java.rmi.Remote`.
- `Remote` is a **marker interface** (no methods).
- Example remote interface:

```java
public interface BankAccount extends Remote {
    void deposit(float amount) throws RemoteException;
    void withdraw(float amount) throws OverdrawnException, RemoteException;
    float balance() throws RemoteException;
}
```

- All remote methods must declare throws RemoteException.

RMI Application – Steps

1. Define Remote Interface

- Must extend Remote.
- Declares methods to be called remotely.

```java
public interface AddServerInterface extends Remote {
    int sum(int a, int b) throws RemoteException;
}
```

2. Implement Remote Interface

- Class must:
  - Implement the interface
  - Extend `UnicastRemoteObject` or use `exportObject()`

```java
public class Adder extends UnicastRemoteObject implements AddServerInterface {
    Adder() throws RemoteException {
        super();
    }
    public int sum(int a, int b) {
        return a + b;
    }
}
```

3. Create and Host Server

- Use `Naming.rebind()` to bind remote object with name.

```java
public class AddServer {
    public static void main(String args[]) {
        try {
            AddServerInterface addService = new Adder();
            Naming.rebind("AddService", addService);
        } catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

4. Create Client Application

- Use `Naming.lookup()` to get remote object reference.

```java
public class Client {
    public static void main(String args[]) {
        try {
            AddServerInterface st = (AddServerInterface)
                Naming.lookup("rmi://" + args[0] + "/AddService");
            System.out.println(st.sum(25, 8));
        } catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

Key Points to Remember

- All remote objects must throw RemoteException.
- Remote object should be accessed via **interface**, not class.
- rebind() – Binds remote object on server.
- lookup() – Used by client to fetch remote object.

Steps to Run RMI Application

1. **Save all files** in a folder named `rmi`.
2. **Compile all Java files:**

```nginx
javac *.java
```

3. **Start RMI Registry:**

> In the same folder (rmi), open command prompt and run:

```sql
start rmiregistry
```

4. **Run Server:**

```nginx
java AddServer
```

5. **Run Client in a new command prompt:**

```nginx
java Client 127.0.0.1
```

Order to Remember:

Save → Compile → RMI Registry → Server → Client

a) RTP (Real-time Transport Protocol)

- **Purpose**: Transports time-sensitive media (audio/video) over IP.
- **Underlying Layer**: Usually runs over UDP (lightweight, low latency).
- **Key Features**:
    1. **Sequence Numbers**: Detect lost or out-of-order packets.
    2. **Timestamps**: Allow the receiver to play out media at correct intervals (synchronization).
    3. **Payload Type IDs**: Identify codec/format used.
- **RTCP (Control Protocol)**: Runs alongside RTP to provide:
    - **Receiver reports** (packet loss, jitter) → feedback on quality
    - **Sender reports** → timing and synchronization info
- **Remember**:
    - "Sequence → loss,"
    - "Timestamp → timing,"
    - RTCP = Reports & Timing

---

b) PageRank (Google's Ranking Algorithm)

- **Idea**: "Important pages get many links **from** other important pages."
- **Steps**:
  1. **Model the Web as a graph**: Pages = nodes, hyperlinks = directed edges.
  2. **Build link matrix A**:

2. **Build link matrix A**:
   - $A_{ij} = 1/(\text{out-links of j})$ if j → i, else 0.
3. **Google matrix G**:

$$G = \alpha A + (1-\alpha)\frac{1}{N}\mathbf{1}\mathbf{1}^T,$$

   where $0 < \alpha < 1$ (damping factor, usually 0.85), N =total pages.

4. **Compute dominant eigenvector r**:

$$Gr = r, \quad \sum_i r_i = 1.$$

   Use the **power method**: iteratively multiply by G until convergence.

Mnemonics: Links → Matrix → Eigenvector → Rank (L MER)

RTP, PageRank & RTSP [5 Marks]

(a) How RTP handles timing issues in real-time transport:

- **RTP** = Real-Time Transport Protocol (used for audio/video).
- Works with **UDP** (faster, no delay).
- Adds **timestamp**, **sequence number**, and **source ID**.
  - 🔹 **Timestamp** = sync audio/video.
  - 🔹 **Sequence number** = arrange packets in order.
  - 🔹 **No guarantee of delivery**, but good for speed.

✅ Helps in smooth playback by managing **delay**, **jitter**, and **order**.

(b) PageRank algorithm (Search engine ranking):

- Used by Google to **rank webpages**.
- More links from **important pages = higher rank**.
- Pages are nodes, links are edges (like a graph).
- Uses **probability** and **matrix math** to calculate rank.

  - Repeats until ranks become stable.
  - Formula includes a **damping factor** to avoid getting stuck.

(c) How RTSP delivers live streaming:

- **RTSP** = Real-Time Streaming Protocol.
- Controls streaming: **Play, Pause, Stop** (like a remote).
- Works with **RTP** to deliver actual media.
- Uses **TCP** for control, **UDP** for fast delivery.

- Used in **live streaming apps, IP cameras**, etc.

(c) RTSP – Real-Time Streaming Protocol

- RTSP is a **control protocol** used to manage the delivery of multimedia content (audio/video) over IP networks.
- It is **similar to HTTP** but specifically designed for **real-time streaming**.
- RTSP uses TCP for control messages and works with RTP for media data delivery.

RTSP Operation (Key Steps):

1. **Client establishes TCP connection** on port 554 (default RTSP port).
2. Client sends RTSP commands like **DESCRIBE**, **SETUP**, and **PLAY** to request media.
3. Server responds with required session setup details.
4. After setup, the client sends **PLAY** to start streaming.
5. To stop, the client sends a **TEARDOWN** command to close the session.

RTSP is only used for **control**, not for actual media transfer. RTP handles the data transmission.

(a) Packet Filtering Firewall

- Packet filtering firewalls operate at the **Network (Layer 3)** and **Transport (Layer 4)** layers of the OSI model.
- They filter traffic based on:
  - Source IP
  - Destination IP
  - Port number

- Protocol (TCP/UDP)

Working:

- Each incoming or outgoing packet is checked against a set of predefined rules.
- If it matches the rules, it is **allowed**; otherwise, it is **blocked**.

Limitations:

- Cannot inspect the full context of TCP connections (e.g., handshake completion).
- Cannot determine if UDP packets are part of an active session.
- Cannot filter packets with spoofed headers or invalid source information reliably.

Packet filtering is fast but offers limited protection compared to stateful firewalls or application layer gateways.

(e) SSL – Secure Socket Layer

- SSL is a **protocol for secure communication** over networks (e.g., TCP/IP).
- Used between client and server (e.g., HTTPS).
- Provides:
  - **Authentication**
  - **Data integrity**
  - **Encryption**
- SSL uses digital certificates and keys for **secure data transfer**.
- Developed by **Netscape**; major versions include **SSL 2.0, 3.0** (later evolved into **TLS**).
- **TLS = SSL Version 3.1**, defined in RFC 2246.

(f) IP Security (IPsec)

- IPSec is a **protocol suite** for securing IP communication.
- It **authenticates and encrypts** each IP packet.
- Works in two modes:
  - **Host-to-host**
  - **Network-to-network**
- IPSec provides:
  - **Authentication** (AH – Authentication Header)
  - **Confidentiality** (ESP – Encapsulating Security Payload)
  - **Key exchange and session protection**
- Used in VPNs and secure remote access.

(g) HTML DOM (Document Object Model)

- HTML DOM is the **programming interface for HTML**.
- Represents HTML as a **tree structure** (Document → Elements → Nodes).
- Allows JavaScript to:
  - Access and modify **HTML elements and content**
  - Change **HTML attributes**
  - Add or remove **elements dynamically**
  - Handle **events** on the page
- Helps in creating **interactive web pages**

(g) IP Security (IPSec) [Revised]

- **IPSec** secures IP communications by **encrypting** and **authenticating** packets.
- Works in:
  - **Host-to-host**
  - **Network-to-network**
  - **Host-to-network**
- Two main protocols:
  1. **AH (Authentication Header)** – Provides **authentication**, integrity, but **no encryption**.
  2. **ESP (Encapsulating Security Payload)** – Provides **confidentiality**, authentication, integrity.

(h) HTML DOM Programming Interface

- **HTML DOM** = Document Object Model used to interact with HTML via **JavaScript**.
- All HTML elements are objects with:
  - **Properties** (e.g., innerHTML)
  - **Methods** (e.g., getElementById)
- JavaScript can:
  - Add/remove elements
  - Change content or styles
  - React to user actions/events

(b) Before 3-Way Handshake (TCP Initialization)

- To establish a connection, TCP uses a **3-way handshake**:
  1. **Sender** sends SYN to receiver.
  2. **Receiver** replies with SYN + ACK.
  3. **Sender** replies with ACK.
- This ensures **synchronization** and sets **initial sequence numbers** for reliable data transfer.

(i) VPN (Virtual Private Network)

- VPN creates a **secure, encrypted connection** over the Internet.
- It allows private access to network resources remotely.
- Works using **tunneling**, where original data is **encapsulated and encrypted**.
- Common use: Secure connection from home to office network.

Components:

- **VPN Client**: User's device with VPN software
- **VPN Server**: Authenticates and connects user
- **Tunnel**: Encrypted path over public Internet