**ĐÁNH GIÁ CHẤT LƯỢNG KỸ NĂNG**

**NETWORKING – FINAL TEST**

**THỜI GIAN: 180 phút**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Yêu cầu:**

* *Tạo file google doc với tên*
  + *docs: [Mã lớp]\_[Họ Tên] chứa toàn bộ nội dung bài thi*
  + *Ví dụ: D25TXCN07B\_TranMinhCuong*
* *Sau khi hoàn thành project, upload link vào trong sheet nộp bài*

## Trắc nghiệm: 20 điểm

Lựa chọn đáp án và đưa ra giải thích ngắn gọn

### 1. HTTP hoạt động ở tầng nào của mô hình OSI?

A. Transport B. Application C. Session D. Presentation

HTTP là giao thức tầng Ứng dụng trong mô hình OSI.

### 2. Giao thức TCP là loại giao thức:

A. Không kết nối – Không tin cậy  
 B. Có kết nối – Tin cậy  
 C. Không kết nối – Tin cậy  
 D. Có kết nối – Không tin cậy

TCP thiết lập kết nối và đảm bảo dữ liệu truyền đúng, đủ, theo thứ tự.

### 3. UDP phù hợp nhất cho ứng dụng nào?

A. Email B. Truyền file C. Video call D. Web

UDP phù hợp ứng dụng cần tốc độ cao, chấp nhận mất gói nhỏ.

### 4. PDU của tầng Transport (TCP) được gọi là:

A. Frame B. Packet C. Segment D. Bit

PDU của TCP/UDP ở tầng Transport gọi là Segment.

### 5. Cổng mặc định của HTTPS là:

A. 21 B. 110 C. 80 D. 443

HTTPS mặc định dùng cổng 443.

### 6. Chức năng chính của tầng Transport là:

A. Routing  
 B. Multiplexing, Segmentation  
 C. Mã hóa dữ liệu  
 D. MAC Address

Transport chia nhỏ dữ liệu và phân biệt ứng dụng qua port.

### 7. Giao thức SMTP dùng để:

A. Gửi email B. Nhận email C. Truyền file D. Tra cứu tên miền

SMTP dùng để gửi email từ client lên server.

### 8. HTTP Request thuộc tầng nào?

A. Presentation B. Transport C. Application D. Network

HTTP Request thuộc tầng Ứng dụng.

### 9. Khi truyền dữ liệu bằng TCP, gói tin phải qua bước:

A. Handshake  
 B. Encryption  
 C. DNS lookup  
 D. ARP

TCP cần bắt tay 3 bước trước khi truyền dữ liệu.

### 10. GET và POST thuộc loại gì?

A. DNS Query B. HTTP Methods  
 C. TCP Flags D. IP Options

GET và POST là các phương thức HTTP.

### 11. Tính chất “stateless” của HTTP nghĩa là:

A. Server lưu lại mọi request  
 B. Mỗi request độc lập, không lưu trạng thái  
 C. Dữ liệu luôn được mã hóa  
 D. Phải dùng session

HTTP không tự ghi nhớ trạng thái các request trước.

### 12. Phần nào dưới đây thuộc HTTP Request?

A. SYN Flag B. Request Line  
 C. TTL D. MAC Header

HTTP Request gồm Request Line, Header, Body.

### 13. Port dùng để xác định:

A. Máy đích B. Ứng dụng đích  
 C. Địa chỉ lớp mạng D. Địa chỉ vật lý

Port xác định ứng dụng/dịch vụ trên máy đích.

### 14. UDP thiếu đặc điểm nào?

A. Kiểm tra lỗi  
 B. Gửi nhanh  
 C. Không cần handshake  
 D. Dùng ít tài nguyên

UDP không đảm bảo kiểm soát lỗi và truyền lại.

### 15. TCP đảm bảo thứ tự gói bằng:

A. TTL B. Sequence Number  
 C. MAC Address D. ARP

TCP dùng số thứ tự để sắp xếp gói tin.

### 16. HTTPS khác HTTP ở điểm:

A. Dùng port 80  
 B. Dùng TLS/SSL mã hóa  
 C. Không chạy trên trình duyệt  
 D. Không dùng TCP

HTTPS là HTTP có thêm lớp mã hóa bảo mật.

### 17. Đơn vị PDU của ứng dụng:

A. Data B. Segment C. Packet D. Frame

PDU của tầng Application gọi chung là Data.

### 18. Phần Body của HTTP Request chứa:

A. Dữ liệu gửi kèm  
 B. IP Header  
 C. TCP Flags  
 D. MAC Address

Body chứa dữ liệu client gửi lên server (form, JSON…).

### 19. POST thường dùng để:

A. Lấy dữ liệu  
 B. Cập nhật trang web  
 C. Gửi dữ liệu từ client lên server  
 D. Xóa tài nguyên

POST dùng để gửi dữ liệu (tạo/cập nhật).

### 20. Transport Layer cung cấp:

A. Xác định đường đi gói tin  
 B. Port và kiểm soát luồng  
 C. Mã hóa và nén  
 D. Mapping domain

Transport quản lý port, kiểm soát luồng và lỗi.

### 21. Nếu ứng dụng cần tốc độ và latency thấp → nên dùng:

A. TCP B. UDP C. SMTP D. ICMP

UDP cho tốc độ nhanh, độ trễ thấp.

### 22. HTTP Response bao gồm:

A. Status Line – Header – Body  
 B. SYN – ACK – FIN  
 C. MAC – IP – TCP  
 D. GET – POST

Cấu trúc chuẩn của HTTP Response.

### 23. Khi mất gói TCP sẽ:

A. Bỏ qua và tiếp tục  
 B. Gửi lại gói  
 C. Xóa kết nối  
 D. Đổi port

TCP tự động truyền lại gói bị mất.

### 24. UDP không đảm bảo điều nào?

A. Thứ tự  
 B. Tốc độ  
 C. Gửi được broadcast  
 D. Không handshake

UDP không đảm bảo thứ tự gói tin.

### 25. Layer nào tạo Process-to-Process Communication?

A. Network B. Transport C. Physical D. Data Link

Transport tạo giao tiếp tiến trình–tiến trình.

### 26. Cookie dùng để:

A. Lưu trạng thái phiên  
 B. Chọn đường đi  
 C. Sinh port động  
 D. Mã hóa tín hiệu vật lý

Cookie giúp duy trì trạng thái cho HTTP stateless.

### 27. HTTP Header cho biết:

A. Trạng thái kết nối TCP  
 B. Thông tin người dùng – trình duyệt – loại dữ liệu  
 C. Địa chỉ MAC  
 D. Cấu trúc DNS

Header mô tả metadata của request/response.

### 28. Một HTTP GET không có:

A. Header  
 B. Body (thường)  
 C. Method  
 D. URL

GET thường không có Body.

### 29. Quá trình segmentation diễn ra ở:

A. Transport B. Network C. Physical D. Session

Segmentation diễn ra tại tầng Transport.

### 30. HTTP lỗi 404 nghĩa là:

A. Server lỗi  
 B. Không tìm thấy tài nguyên  
 C. Không có kết nối  
 D. Yêu cầu bị từ chối

404 nghĩa là server không tìm thấy URL yêu cầu.

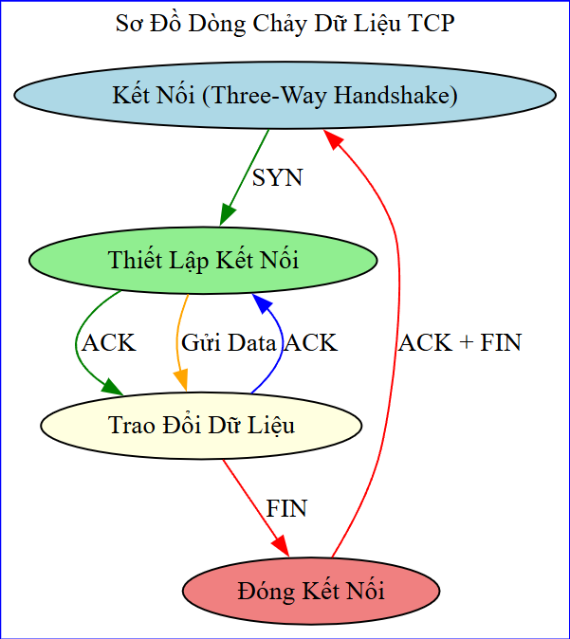
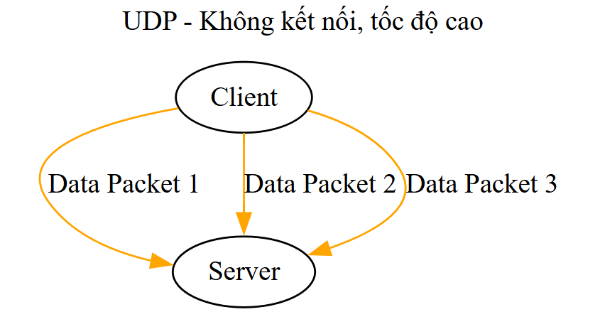
## Thực hành: 40 điểm

### Câu 1 (10 điểm)

**So sánh TCP và UDP theo bảng dưới đây và vẽ sơ đồ minh họa:**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **TCP** | **UDP** |
| Kiểu kết nối | **Connection-oriented**: Phải thiết lập kết nối (bắt tay 3 bước) trước khi truyền dữ liệu. | **Connectionless**: Không cần thiết lập kết nối, dữ liệu được gửi đi ngay lập tức. |
| Độ tin cậy | **Cao**: Đảm bảo dữ liệu đến đích đúng thứ tự và không bị mất (có xác nhận ACK). | **Thấp**: Không đảm bảo dữ liệu đến đích hay đúng thứ tự. Có thể bị mất gói tin. |
| Kiểm soát lỗi | Có cơ chế kiểm soát luồng, kiểm soát tắc nghẽn và truyền lại gói tin lỗi/mất. | Chỉ có kiểm soát lỗi cơ bản bằng Checksum (nếu lỗi thì hủy gói tin, không gửi lại). |
| Tốc độ | **Chậm hơn**: Do phải xử lý các thủ tục thiết lập, xác nhận và kiểm soát. | **Rất nhanh**: Do tối giản hóa quy trình, không có độ trễ chờ xác nhận. |
| Ứng dụng phù hợp | Web (HTTP/HTTPS), Email (SMTP/IMAP), Truyền file (FTP), SSH. | Livestream, Video Call, Game online, DNS, VoIP. |

**Yêu cầu kèm theo:**

* Vẽ sơ đồ **dòng chảy dữ liệu TCP vs UDP**  
   
* Vẽ **kích thước header cơ bản** của mỗi giao thức

| Source Port | Destination Port |

| Sequence Number |

| Acknowledgment Number |

| Flags | Window | Checksum |

| Data... |

**TCP header to → nhiều thông tin → chậm hơn**

| Source Port | Destination Port |

| Length | Checksum |

| Data... |

**UDP header nhỏ → gửi nhanh**

### Câu 2 (10 điểm)

**Trình bày mô hình TCP/IP (tập trung tầng Application & Transport).**

Bao gồm:

* Tên các tầng
* Chức năng từng tầng
* Ví dụ giao thức trong mỗi tầng
* *Vẽ mapping TCP/IP sang OSI nhưng chỉ highlight Application & Transport*

**1. Các tầng trong mô hình TCP/IP**

*Mô hình TCP/IP tiêu chuẩn gồm 4 tầng chính:*

1. ***Application Layer*** *(Tầng Ứng dụng)*
2. ***Transport Layer*** *(Tầng Giao vận)*
3. ***Internet Layer*** *(Tầng Mạng)*
4. ***Network Access Layer*** *(Tầng Truy cập mạng)*

***2. Chức năng và Giao thức tiêu biểu***

*Dưới đây là chi tiết về chức năng của từng tầng, đặc biệt nhấn mạnh vào Application và Transport:*

***A. Tầng Application (Ứng dụng)***

* ***Chức năng:*** *Là giao diện trực tiếp giữa người dùng và mạng. Nó chịu trách nhiệm hỗ trợ các ứng dụng trao đổi dữ liệu qua mạng, định dạng dữ liệu và quản lý các phiên làm việc.*
* ***Giao thức ví dụ:***
  + ***HTTP/HTTPS:*** *Duyệt web.*
  + ***FTP:*** *Truyền file.*
  + ***SMTP/POP3/IMAP:*** *Gửi và nhận Email.*
  + ***DNS:*** *Phân giải tên miền thành địa chỉ IP.*

***B. Tầng Transport (Giao vận)***

* ***Chức năng:*** *Đảm bảo việc truyền dữ liệu giữa hai tiến trình trên các thiết bị đầu cuối. Tầng này chịu trách nhiệm phân mảnh dữ liệu lớn thành các phân đoạn nhỏ (segments) và kiểm soát luồng dữ liệu.*
* ***Giao thức ví dụ:***
  + ***TCP:*** *Truyền dữ liệu tin cậy (có bắt tay, có kiểm tra lỗi).*
  + ***UDP:*** *Truyền dữ liệu nhanh (không kết nối, dùng cho livestream/game).*

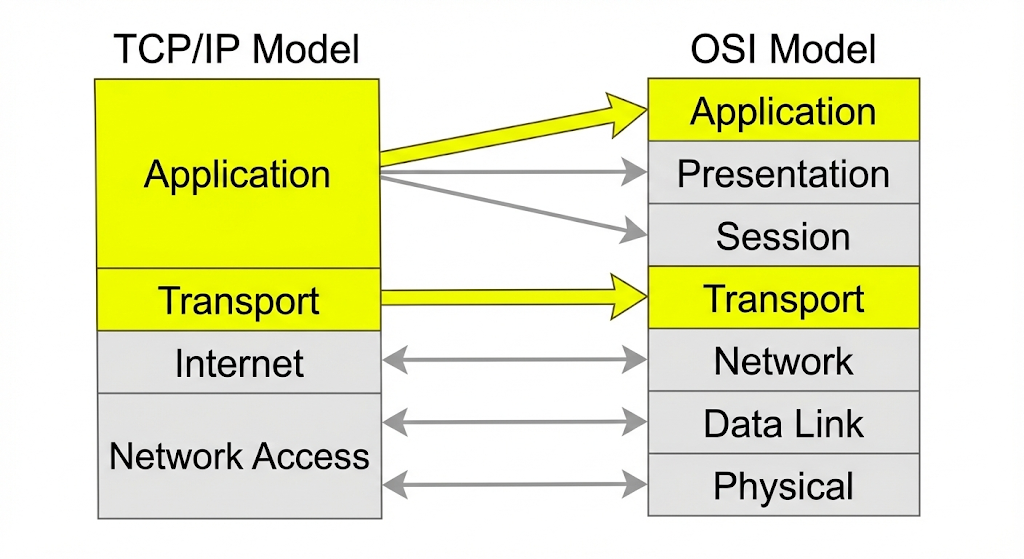
***C. Tầng Internet (Mạng)***

* ***Chức năng:*** *Định tuyến các gói tin (packets) từ nguồn đến đích qua các mạng khác nhau bằng cách sử dụng địa chỉ IP.*
* ***Giao thức ví dụ:*** *IPv4, IPv6, ICMP.*

***D. Tầng Network Access (Truy cập mạng)***

* ***Chức năng:*** *Quản lý việc truyền dữ liệu trên các phương tiện vật lý (cáp quang, wifi, cáp đồng).*
* ***Giao thức ví dụ:*** *Ethernet, Wi-Fi (802.11).*

**3.Mapping TCP/IP sang OSI**

****

### Câu 3 (10 điểm)

**Giải thích cơ chế 3-way handshake của TCP.**

Yêu cầu:

1. Vẽ sơ đồ trao đổi gói tin có:  
   * SYN
   * SYN/ACK
   * ACK
   * Sequence Number & Ack Number
2. Tại sao cần handshake trước khi truyền dữ liệu?
3. Điều gì xảy ra nếu handshake bị lỗi?

1.Sơ đồ trao đổi gói tin   
**Client Server**

**| |**

**|--------------- --- SYN, Seq = x ------------------------> |**

**| |**

**| 🡨------- SYN + ACK, Seq = y, Ack = x + 1 --------- |**

**| |**

**| ----------- ACK, Seq = x + 1, Ack = y + 1 -----------> |**

**| |**

**| ==== KẾT NỐI TCP ĐƯỢC THIẾT LẬP ===== |**

2. **Tại sao cần handshake trước khi truyền dữ liệu?**

Việc "bắt tay" không đơn thuần là chào hỏi mà phục vụ các mục đích sống còn cho độ tin cậy:

1. **Đồng bộ hóa số thứ tự (Sequence Numbers):** Đảm bảo cả hai bên biết bắt đầu đếm từ đâu để có thể sắp xếp lại các gói tin bị đến sai thứ tự trong quá trình truyền.
2. **Kiểm tra khả năng truyền dẫn:** Xác nhận rằng cả đường đi (Client $\rightarrow$ Server) và đường về (Server $\rightarrow$ Client) đều đang hoạt động tốt.
3. **Thỏa thuận thông số:** Thiết lập các thông số về bộ đệm (Window Size) để kiểm soát luồng, tránh việc một bên gửi quá nhanh làm bên kia bị quá tải.

3. **Điều gì xảy ra nếu handshake bị lỗi?**

Nếu một trong các bước trên không hoàn tất, kết nối sẽ không được thiết lập:

* **Mất gói tin SYN hoặc SYN/ACK:** TCP sẽ kích hoạt cơ chế **Retransmission**. Client (hoặc Server) sẽ gửi lại gói tin sau một khoảng thời gian chờ (Timeout). Nếu sau nhiều lần thử vẫn thất bại, kết nối sẽ bị hủy.
* **Server từ chối kết nối:** Nếu cổng (port) trên Server đang đóng, Server sẽ gửi lại một gói tin **RST (Reset)** thay vì SYN/ACK để thông báo cho Client biết.
* **Tấn công SYN Flood:** Đây là một lỗi bảo mật cố ý, nơi kẻ tấn công gửi hàng loạt gói SYN nhưng không bao giờ gửi lại gói ACK cuối cùng, khiến Server cạn kiệt tài nguyên vì phải chờ đợi ở trạng thái SYN-RECEIVED.

### Câu 4 (10 điểm)

Phương thức HTTP hoạt động như thế nào trên mô hình TCP

Trình bày và vẽ mô hình:

### 1. Trình duyệt tạo HTTP Request

* Request Line: Chứa phương thức (GET, POST...), URL và phiên bản HTTP.
* Header: Chứa thông tin bổ sung như User-Agent, Content-Type, Cookie.
* Body: Chứa dữ liệu gửi kèm (thường có trong POST, PUT).

### 2. Tầng Transport chia dữ liệu thành TCP Segment

* Chia dữ liệu HTTP thành các TCP Segment.
* Gắn Source Port, Destination Port.
* Gán Sequence Number để đảm bảo thứ tự.
* Đảm bảo độ tin cậy khi truyền.

### 3. Server nhận và ghép lại segment → xử lý → trả HTTP Response

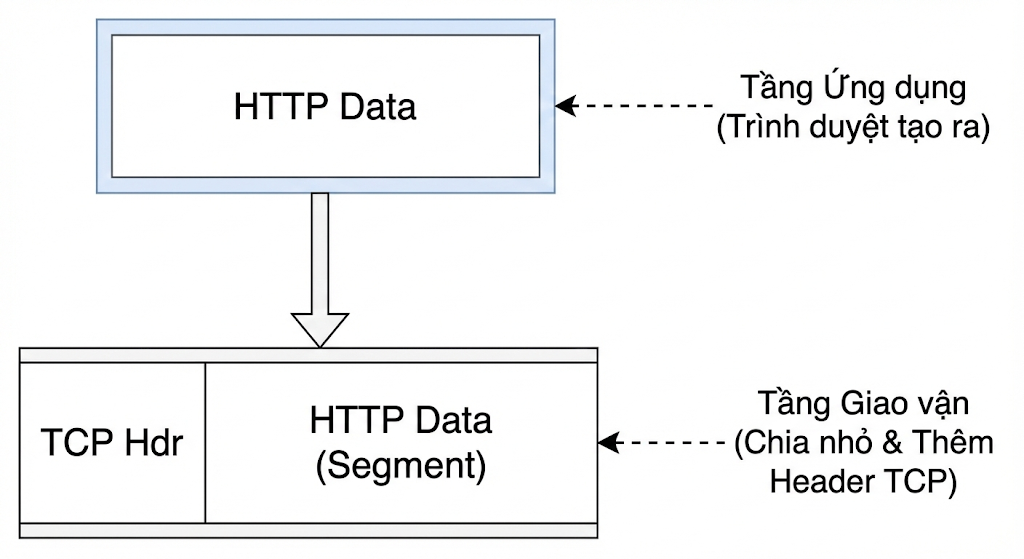
* TCP kiểm tra lỗi và ghép lại dữ liệu.
* Chuyển dữ liệu hoàn chỉnh lên tầng Application.
* Server xử lý request và tạo HTTP Response:
* Status Line: (200 OK, 404 Not Found...).
* Header: Thông tin bổ sung.
* Body: Nội dung trả về.

### Vẽ sơ đồ minh họa dòng chảy

A diagram of a software process

AI-generated content may be incorrect.

**Minh họa cấu trúc dữ liệu qua các tầng:**

****

### 5. Liệt kê 4 HTTP Method phổ biến: GET, POST, PUT, DELETE

**GET:** Lấy dữ liệu từ server

**POST:** Gửi dữ liệu từ client lên server

**PUT:** Cập nhật hoặc thay thế tài nguyên

**DELETE:** Xóa tài nguyên

## Tình huống: 40 điểm

### Câu 5 (20 điểm)

Một ứng dụng **streaming video** yêu cầu:

* Độ trễ thấp
* Tốc độ cao
* Chấp nhận mất một vài gói

Trả lời:

1. Chọn TCP hay UDP? Giải thích.
2. Nếu dùng UDP → nêu **4 kỹ thuật giảm mất dữ liệu** (FEC, jitter buffer, resend selective, adaptive bitrate…)
3. Liệt kê **4 ứng dụng thực tế** cũng nên dùng UDP
4. Vẽ mô hình truyền dữ liệu qua UDP

Bài Làm:

**1: Giao thức UDP (User Datagram Protocol).**

**Giải thích:** Dựa trên yêu cầu của bài toán là *độ trễ thấp, tốc độ cao* và *chấp nhận mất gói tin*, UDP là lựa chọn tối ưu vì:

* **Loại bỏ độ trễ (Latency):** UDP không thực hiện quy trình bắt tay (handshake) hay thiết lập kết nối phức tạp như TCP, giúp dữ liệu được gửi đi ngay lập tức.
* **Không truyền lại (No Retransmission):** Trong Streaming, việc dừng hình ảnh để chờ một gói tin bị mất được truyền lại (cơ chế của TCP) sẽ gây ra hiện tượng "giật/lag" (buffering) rất khó chịu. Với UDP, nếu mất một gói tin, video chỉ bị nhiễu nhẹ trong tích tắc rồi tiếp tục phát sang khung hình tiếp theo ngay, đảm bảo tính thời gian thực (Real-time).
* **Header nhỏ gọn:** Header của UDP chỉ 8 bytes (so với 20 bytes của TCP), giúp tiết kiệm băng thông và tăng tốc độ xử lý.

2. **Các kỹ thuật giảm ảnh hưởng của mất dữ liệu (Khi dùng UDP)**

Vì UDP không có cơ chế sửa lỗi nội tại, tầng Ứng dụng (Application Layer) phải tự xử lý bằng các kỹ thuật sau:

1. **FEC (Forward Error Correction - Sửa lỗi hướng tới):**
   * Bên gửi sẽ gửi kèm thêm các dữ liệu dự phòng (redundant data). Nếu một vài gói tin bị mất, bên nhận có thể dùng dữ liệu dự phòng này để tái tạo lại gói tin bị mất mà không cần yêu cầu gửi lại.
2. **Jitter Buffer (Bộ đệm chống rung):**
   * Do UDP không đảm bảo thứ tự, các gói tin có thể đến không đều đặn. Bên nhận sẽ có một bộ nhớ đệm nhỏ để giữ các gói tin lại trong một khoảng thời gian ngắn, sắp xếp chúng đúng thứ tự rồi mới phát lên màn hình, giúp video mượt mà hơn.
3. **Resend Selective (Gửi lại có chọn lọc / ARQ):**
   * Thay vì gửi lại mọi gói tin bị lỗi như TCP, ứng dụng chỉ yêu cầu gửi lại những gói tin cực kỳ quan trọng (ví dụ: Key-frame/I-frame dùng để dựng hình) nếu bị mất. Các gói tin phụ (P-frame/B-frame) nếu mất có thể bỏ qua.
4. **Adaptive Bitrate (ABR - Tốc độ thích ứng):**
   * Ứng dụng liên tục giám sát băng thông mạng. Nếu mạng yếu (gói tin bị rớt nhiều), nó sẽ tự động hạ chất lượng video (ví dụ từ 1080p xuống 720p hoặc 480p) để giảm tải, giúp video không bị đứng hình.

3. **Liệt kê 4 ứng dụng thực tế nên dùng UDP**

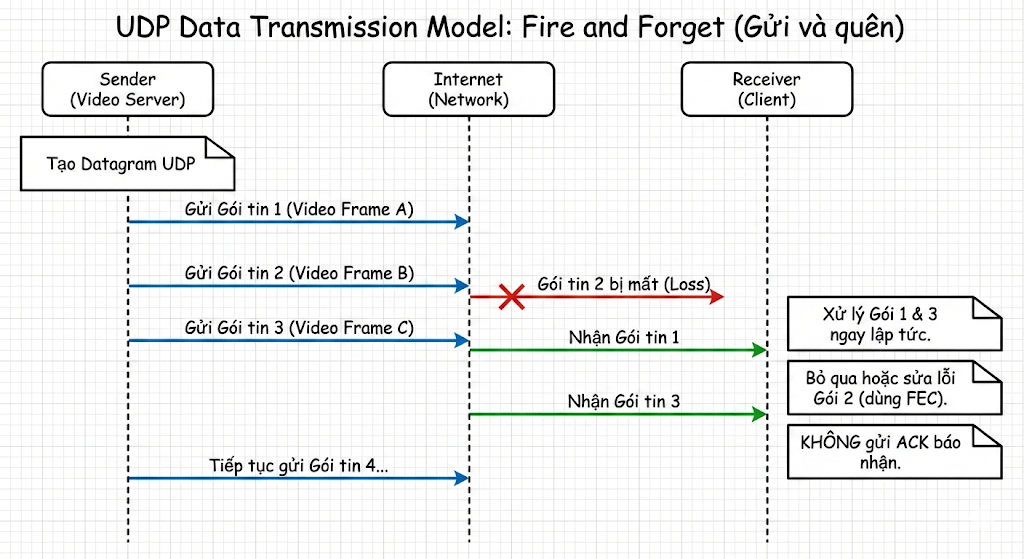
1. **VoIP (Voice over IP):** Các ứng dụng gọi thoại như Zalo Call, Skype, Zoom (ưu tiên nghe được tiếng ngay lập tức dù đôi khi âm thanh bị rè).
2. **Online Gaming (Game thời gian thực):** Các game bắn súng (FPS) hoặc MOBA (Liên Minh Huyền Thoại) cần phản hồi thao tác ngay lập tức.
3. **DNS (Domain Name System):** Dịch vụ phân giải tên miền cần tốc độ phản hồi nhanh, gói tin truy vấn nhỏ.
4. **DHCP (Dynamic Host Configuration Protocol):** Giao thức cấp phát IP động, cần cơ chế broadcast đơn giản và nhanh gọn để thiết bị kết nối mạng.

4. **Mô hình truyền dữ liệu qua UDP**

Khác với TCP là dòng chảy liên tục (stream), UDP bắn dữ liệu theo từng gói độc lập (datagram) và không cần xác nhận nhận hàng.

**Mô tả sơ đồ:**

* **Server (Sender):** Liên tục đẩy các gói Datagram ra đường truyền.
* **Client (Receiver):** Nhận được gói nào thì xử lý gói đó.
* **Packet Loss:** Trong sơ đồ có gói tin bị mất (Gói 2), nhưng Server vẫn tiếp tục gửi Gói 3, 4 mà không dừng lại.



### Câu 6 (20 điểm)

Một web server nhận HTTP Request từ trình duyệt.

**Hãy mô tả:**

1. Application Layer tạo HTTP Request
2. Transport Layer đóng gói → Segment
3. Network Layer (brief, chỉ nhắc tên PDU) → Packet

**Yêu cầu:**

* Vẽ sơ đồ Encapsulation 3 tầng
* Ghi rõ PDU từng tầng
* Mô tả quá trình server **De-encapsulation**

Bài Làm:

**I. Mô tả quá trình Encapsulation (từ Client → Server)**

**1. Application Layer – Tạo HTTP Request**

* Trình duyệt (Browser) hoạt động ở Application Layer
* Tạo HTTP Request (ví dụ: GET /index.html HTTP/1.1)
* HTTP Request gồm:
  + Request Line
  + Header
  + Body (nếu có)

**PDU tầng Application: Data (HTTP Request)**

**2. Transport Layer – Đóng gói thành Segment**

* HTTP Request được chuyển xuống Transport Layer (TCP)
* TCP:
  + Gán Source Port (port ngẫu nhiên của client)
  + Gán Destination Port (80 hoặc 443)
  + Chia dữ liệu thành các Segment
  + Thêm Sequence Number, ACK, Checksum

**PDU tầng Transport: TCP Segment**

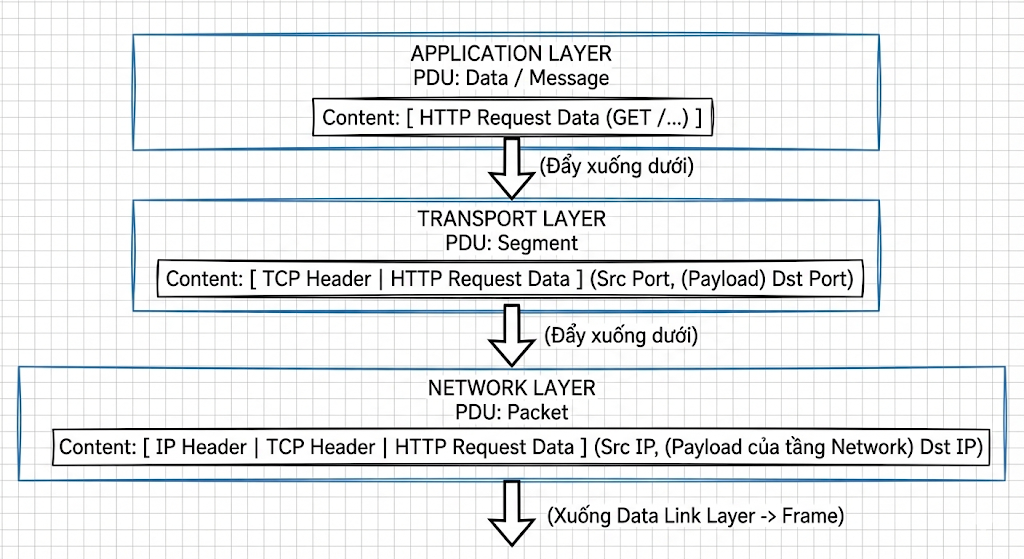
**3. Network Layer – Đóng gói thành Packet (brief)**

* TCP Segment được chuyển xuống Network Layer
* IP thêm:
  + Source IP (IP client)
  + Destination IP (IP server)

**PDU tầng Network: IP Packet**

**II. Sơ đồ Encapsulation 3 tầng (bắt buộc vẽ)**

* Vẽ sơ đồ Encapsulation 3 tần



**Mô tả quá trình Server De-encapsulation (Mở gói)**

Khi tín hiệu điện đi vào Card mạng (NIC) và được chuyển thành dữ liệu số, quá trình xử lý qua 3 tầng chính diễn ra như sau:

**1. Tại Network Layer (Tầng Mạng)**

* **Đầu vào:** Nhận **Packet** (Gói tin) từ tầng liên kết dữ liệu đẩy lên.
* **Hành động (Xử lý):**
  + Hệ điều hành (OS) đọc **IP Header** bao bọc bên ngoài.
  + **Kiểm tra:** So sánh Destination IP (IP đích) trong header xem có trùng khớp với địa chỉ IP của Server không. Nếu đúng, nó chấp nhận gói tin.
* **Hành động (Mở gói):** Gỡ bỏ (Strip) toàn bộ **IP Header**.
* **Đầu ra:** Phần dữ liệu còn lại (Payload) chính là **Segment**, được đẩy lên tầng Transport.

**2. Tại Transport Layer (Tầng Giao vận)**

* **Đầu vào:** Nhận **Segment** từ tầng Network.
* **Hành động (Xử lý):**
  + Hệ điều hành đọc **TCP Header**.
  + **Kiểm tra:** Kiểm tra Checksum để đảm bảo dữ liệu không bị lỗi. Quan trọng nhất, nó đọc Destination Port (Ví dụ: Port 80 hoặc 443) để xác định ứng dụng nào đang chờ dữ liệu này.
* **Hành động (Mở gói):** Gỡ bỏ (Strip) **TCP Header**.
* **Đầu ra:** Phần dữ liệu cốt lõi (Payload) chính là **Data** (HTTP Request), được đẩy lên tầng Application (cụ thể là tiến trình Web Server đang lắng nghe tại Port đó).

**3. Tại Application Layer (Tầng Ứng dụng)**

* **Đầu vào:** Nhận **Data / Message** (Dữ liệu thô) từ tầng Transport.
* **Hành động (Xử lý):**
  + Phần mềm Web Server (như Apache, Nginx, IIS...) nhận chuỗi dữ liệu này.
  + Nó phân tích (Parse) chuỗi ký tự để hiểu đây là một **HTTP Request**.
  + Nó đọc các thông tin: *Method (GET/POST), URL (/index.html), Headers (User-Agent, Cookies)*.
* **Kết quả:** Server hiểu Client muốn gì và bắt đầu quy trình xử lý để tạo ra phản hồi (HTTP Response).