**BÀI TẬP 5**



**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**







**NHẬP MÔN MẠNG MÁY TÍNH**

**LỚP: IT005.O118**

**Nhóm: CYBER SQUAD**

**Học ít hiểu nhiều**



**MỤC LỤC**

**[BẢNG ĐÁNH GIÁ THÀNH VIÊN 2](#_Toc8619)**

**[I. TRẢ LỜI CÂU HỎI. 3](#_Toc30986)**

[Câu 1: Bảng Hash dùng để làm gì? 3](#_Toc30822)

[Câu 2: Checksum dùng để làm gì, cho ví dụ tính check sum 16 bit? 4](#_Toc22938)

[Câu 3: Phân biệt Multiplexing và Demultiplexing trong internet. 5](#_Toc2327)

[Câu 4: Vì sao phải dùng các nguyên lý truyền tin tin cậy, so sánh sự khác nhau của các nguyên lý RDT 1.0, RDT 2.0, RDT 2.1, 2.2 và RDT 3.0. 7](#_Toc28290)

[Câu 5: Vẽ lại mô hình FSM của rdt 3.0 bên nhận. 8](#_Toc23304)

**[II. BÀI TẬP TRONG SÁCH. 9](#_Toc5589)**

[R4: 9](#_Toc8259)

[R6: 9](#_Toc23722)

[R7: 10](#_Toc12244)

[P1: 11](#_Toc2792)

[P2: 11](#_Toc29416)

**[III. BÀI HỌC RÚT RA. 13](#_Toc10848)**

**[NGUỒN THAM KHẢO: 14](#_Toc25599)**

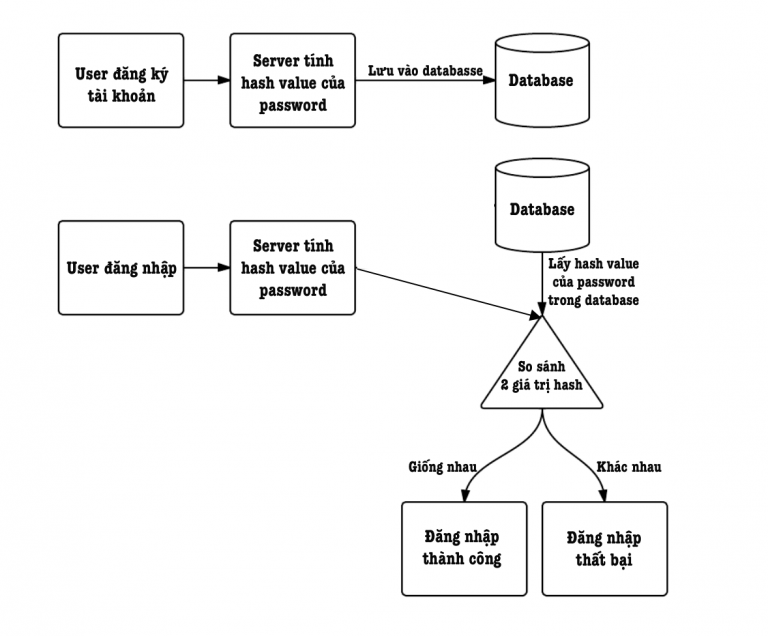
**BẢNG ĐÁNH GIÁ THÀNH VIÊN**

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Tên thành viên** | **Phân chia công việc** | **Đánh giá** |
| 22521060 | Lê Minh Nhựt(C) | Câu 5, P2 | 100% |
| 22520195 | Trần Đình Khánh Đăng | Câu 3, P1 | 100% |
| 22521189 | Thái Ngọc Quân | Câu 2, R7 | 100% |
| 22521078 | Bùi Nhật Phi | Câu 1, R4 | 100% |
| 22520127 | Võ Ngọc Bảo | Câu 4, R6 | 100% |

1. **TRẢ LỜI CÂU HỎI.**

**Câu 1: Bảng Hash dùng để làm gì?**

***Bảng hash* được sử dụng để lưu trữ và truy xuất dữ liệu một cách hiệu quả**. *Bảng hash* sử dụng hàm băm để chuyển đổi dữ liệu thành một giá trị số nguyên duy nhất, sau đó sử dụng giá trị này để xác định vị trí của dữ liệu trong bảng.



*Bảng hash* có thể được sử dụng cho nhiều mục đích khác nhau trong mạng máy tính, bao gồm:

* **Lưu trữ thông tin người dùng:** *Bảng hash* có thể được sử dụng để lưu trữ thông tin người dùng, chẳng hạn như tên người dùng, mật khẩu và thông tin liên hệ. Điều này giúp người dùng dễ dàng đăng nhập và truy cập các dịch vụ mạng.
* **Lưu trữ dữ liệu ứng dụng:** *Bảng hash* có thể được sử dụng để lưu trữ dữ liệu ứng dụng, chẳng hạn như dữ liệu trò chơi, dữ liệu mạng xã hội và dữ liệu thương mại điện tử. Điều này giúp các ứng dụng có thể truy cập dữ liệu một cách nhanh chóng và hiệu quả.
* **Caching dữ liệu:** *Bảng hash* có thể được sử dụng để lưu trữ dữ liệu được sử dụng thường xuyên trong bộ nhớ cache. Điều này giúp cải thiện hiệu suất của các ứng dụng và dịch vụ mạng.

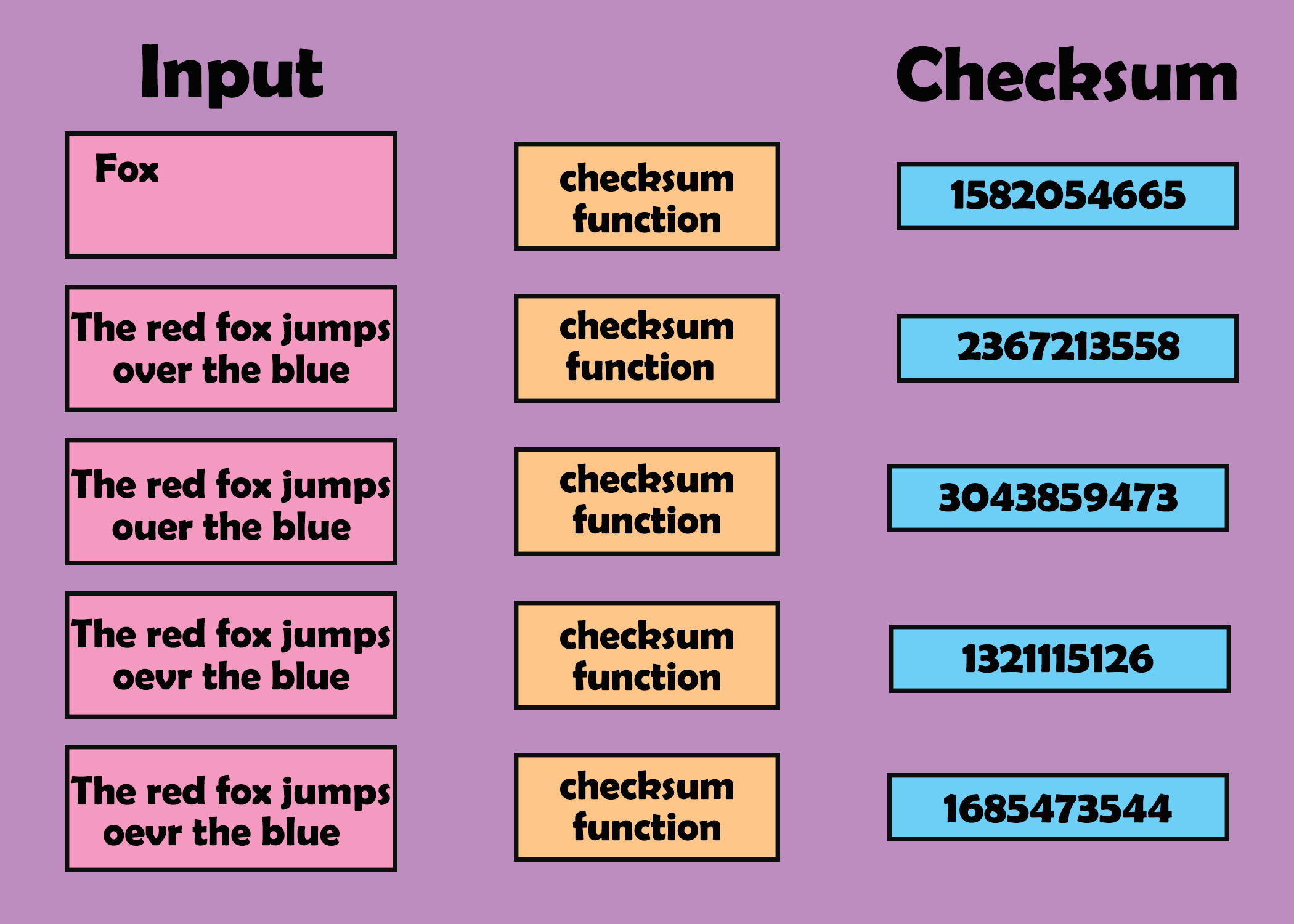
**Một số ví dụ cụ thể** về cách sử dụng bảng hash trong mạng máy tính:

* **Trong hệ thống phân tán:** *Bảng hash phân tán* (DHT) là một loại *bảng hash* được sử dụng trong các hệ thống phân tán. DHT phân tán dữ liệu giữa các nút mạng, giúp cải thiện khả năng mở rộng và hiệu suất.
* **Trong mạng chia sẻ tệp:** *Bảng hash* có thể được sử dụng để lưu trữ thông tin về các tệp chia sẻ. Điều này giúp người dùng dễ dàng tìm kiếm và tải xuống các tệp.
* **Trong mạng xã hội:** *Bảng hash* có thể được sử dụng để lưu trữ thông tin về người dùng và mối quan hệ của họ. Điều này giúp các ứng dụng mạng xã hội có thể đề xuất bạn bè và nội dung phù hợp với người dùng.

**Câu 2: Checksum dùng để làm gì, cho ví dụ tính check sum 16 bit?**

***Checksum*** làmột giá trị được tính toán từ một khối dữ liệu, **được sử dụng để phát hiện lỗi trong quá trình truyền** (các bit bị lỗi, từ 0 thành 1 hoặc ngược lại).

***Checksum* hoạt động bằng cách**: tính toán giá trị *checksum* trong khối dữ liệu bằng thuật toán nào đó, khi dữ liệu được nhận giá trị *checksum* được tính toán lại và so sánh với giá trị *checksum* gốc. Nếu hai *checksum* không khớp, thì dữ liệu đã bị lỗi. Nếu khớp, chưa phát hiện lỗi.



**Ví dụ tính *checksum* 16bit**:

1 0 1 0 0 1 0 0 1 1 0 1 0 1 1 0

1 0 1 1 0 0 1 0 1 1 1 0 1 0 0 1

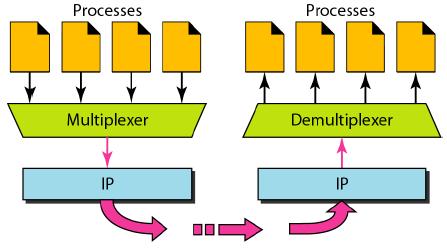
**wraparound**  **1** 0 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1

**1**

**sum** 0 1 0 1 0 1 1 1 1 1 0 0 0 0 0 0

**checksum** 1 0 1 0 1 0 0 0 0 0 1 1 1 1 1 1

**Câu 3: Phân biệt Multiplexing và Demultiplexing trong internet.**



**BẢNG PHÂN BIỆT MULTIPLEXING VÀ DEMULTIPLEXING**

|  |  |  |
| --- | --- | --- |
|  | **Multiplexing** | **Demultiplexing** |
| **Khái niệm** | Quá trình kết hợp các luồng dữ liệu khác nhau thành một luồng dữ liệu duy nhất | Quá trình tách một luồng dữ liệu duy nhất thành các luồng dữ liệu khác nhau |
| **Mục đích** | Để truyền nhiều luồng dữ liệu qua cùng một kênh truyền dẫn | Để phân phối các luồng dữ liệu đến các thiết bị hoặc ứng dụng đích |
| **Thiết bị** | Được thực hiện bởi thiết bị đầu cuối hoặc thiết bị mạng | Được thực hiện bởi thiết bị mạng |
| **Giao thức** | Sử dụng các giao thức multiplexing, chẳng hạn như Ethernet, PPP, và ATM | Sử dụng các giao thức demultiplexing, chẳng hạn như IP, TCP, và UDP |

**Câu 4: Vì sao phải dùng các nguyên lý truyền tin tin cậy, so sánh sự khác nhau của các nguyên lý RDT 1.0, RDT 2.0, RDT 2.1, 2.2 và RDT 3.0.**

*Các nguyên lí truyền tin tin cậy được sử dụng để* *đảm bảo rằng dữ liệu được truyền từ máy này sang máy khác một cách chính xác và đầy đủ*. Điều này là cần thiết cho nhiều ứng dụng, chẳng hạn như:

* ***Truyền file.***
* ***Truyền dữ liệu nhạy cảm***: chẳng hạn như dữ liệu tài chính hoặc dữ liệu y tế.
* ***Truyền dữ liệu thời gian thực***: ví dụ như video hoặc âm thanh.

**RDT 1.0** là nguyên lí truyền tin tin cậy đơn giản nhất. Nó sử dụng phương thức **Stop and Wait** để đảm bảo rằng mỗi gói dữ liệu được nhận thành công trước khi gói dữ liệu tiếp theo được gửi. Phương thức này có hiệu suất thấp, vì máy phát phải chờ phản hồi từ máy thu trước khi gửi gói dữ liệu tiếp theo.

**RDT 2.0** cải thiện hiệu suất của RDT 1,0 bằng cách sử dụng phương thức **Sliding Window**. Phương thức này cho phép máy phát gửi nhiều gói dữ liệu cùng một lúc, miễn là máy thu đã nhận được phản hồi cho các gói dữ liệu trước đó.

**RDT 2.1** cải thiện khả năng chịu lỗi của RDT 2.0 bằng cách thêm cơ chế phát lại. Nếu máy thu không nhận được dữ liệu, máy phá sẽ gửi lại gói dữ liệu đó.

**RDT 2.2** cải thiện hiệu năng chịu lỗi của RDT 2.1 bằng cách thêm cơ chế xác nhận. Máy thu gửi xác nhận cho mỗi gói dữ liệu nhận được. Máy phát phát chỉ gửi lại gói dữ liệu nếu không nhận được xác nhận.

**RDT 3.0** là nguyên lí truyền tin tin cậy hiệu quả nhất. Nó sử dụng phương pháp **Go Back N** để đảm bảo rằng tất cả các gói dữ liệu được nhận thành công . Phương pháp này sử dụng một của sổ trượt để xác định các gói dữ liệu chưa được nhận. Nếu máy thu không nhận được bất kì gói dữ liệu nào trong của sổ trượt, máy phát sẽ gửi lại tất cả các gói dữ liệu trong cửa sổ.

**Bảng so sánh các nguyên lí RDT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nguyên lí** | **Phương thức** | **Cơ chế phát lại** | **Cơ chế xác nhận** |
| **RDT 1.0** | Stop and Wait | Không có | Không có |
| **RDT 2.0** | Sliding Window | Không có | Không có |
| **RDT 2.1** | Sliding Window | Có | Không có |
| **RDT 2.2** | Sliding Window | Có | Có |
| **RDT 3.0** | Go Back N | Có | Có |

**Câu 5: Vẽ lại mô hình FSM của rdt 3.0 bên nhận.**

Wait for call 0 from below

Wait for call 1 from below

rdt\_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&

has\_seq1(rcvpkt)

Deliver(rcvpkt)

Sndpkt=makepkt(Ack,1,checksum)

Udt\_send(sndpkt)

rdt\_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&has\_seq0(rcvpkt)

Deliver(rcvpkt)

Sndpkt=makepkt(Ack,0,checksum)

Udt\_send(sndpkt)

rdt\_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&has\_seq0(rcvpkt)

Sndpkt=makepkt(Ack,1,checksum)

Udt\_send(sndpkt)

rdt\_rcv(rcvpkt)&&notcorrupt(rcvpkt)&&has\_seq0(rcvpkt)

Sndpkt=makepkt(Ack,0,checksum)

Udt\_send(sndpkt)

*Mô hình FSM của RDT 3.0 bên nhận*

1. **BÀI TẬP TRONG SÁCH.**

**R4:**

**Mô tả lý do tại sao nhà phát triển ứng dụng có thể chọn chạy ứng dụng qua UDP thay vì TCP.**

Một số lý do tại sao nhà phát triển ứng dụng có thể chọn chạy ứng dụng trên UDP thay vì TCP:

* **Tốc độ:** UDP nhanh hơn TCP vì nó không có các quá trình phức tạp như thiết lập kết nối và kiểm soát luồng. Điều này làm cho UDP phù hợp với các ứng dụng cần tốc độ cao, chẳng hạn như phát trực tiếp video, trò chơi trực tuyến và truyền tải dữ liệu thời gian thực.
* **Độ trễ thấp:** UDP có độ trễ thấp hơn TCP vì nó không phải chờ xác nhận dữ liệu. Điều này làm cho UDP phù hợp với các ứng dụng cần độ trễ thấp, chẳng hạn như VoIP và trò chơi trực tuyến.
* **Quy mô:** UDP có thể dễ dàng mở rộng hơn TCP vì nó không cần duy trì trạng thái kết nối. Điều này làm cho UDP phù hợp với các ứng dụng cần hỗ trợ một lượng lớn người dùng đồng thời, chẳng hạn như phát sóng trực tiếp video và trò chơi trực tuyến.
* **Độ bền bỉ:** UDP ít bền bỉ hơn TCP vì nó không đảm bảo việc phân phối dữ liệu theo thứ tự hoặc việc dữ liệu sẽ được phân phối thành công. Tuy nhiên, điều này có thể chấp nhận được đối với một số ứng dụng, chẳng hạn như phát trực tiếp video, vì người dùng có thể chấp nhận được việc mất một vài khung hình.

**R6:**

**Liệu một ứng dụng có thể tham gia việc truyền dữ liệu đáng tin cậy ngay cả khi ứng dụng chạy trên UDP không? Nếu có, thì bằng cách nào?**

**Có**, có thể cho một ứng dụng tham gia truyền dữ liệu đáng tin cậy ngay cả khi ứng dụng chạy trên UDP. Một số cách để đạt được điều này bao gồm:

* **Sử dụng giao thức đáng tin cậy trên lớp ứng dụng**. Ví dụ, một ứng dụng có thể sử dụng giao thức truyền tệp đáng tin cậy như FTP hoặc HTTP tren lớp ứng dụng. Giao thức lớp ứng dụng sẽ chịu trách nhiệm đảm bảo rằng dữ liệu được truyền đáng tin cậy, ngay cả khi giao thức lớp vận chuyển bên dưới (UDP) không đáng tin cậy.
* **Sử dụng cơ chế kiểm tra lỗi và sửa lỗi**. Một ứng dụng có thể sử dụng các cơ chế kiểm tra lỗi và sửa lỗi để phát hiện và sữa chữa lỗi dữ liệu. Ví dụ, một ứng dụng có thể sử dụng mã kiểm tra lỗi để phát hiện lỗi dữ liệu và sau đó yêu cầu truyền lại dữ liệu bị lỗi.
* **Sử dụng các kỹ thuật dự phòng.** Một ứng dụng có thể sử dụng các kỹ thuật dự phòng để giảm thiểu tác động của mất dữ liệu. Ví dụ, một ứng dụng có thể lưu trữ dữ liệu dự phòng trên nhiều máy chủ. Nếu một máy chủ bị mất dữ liệu, dữ liệu có thể được khôi phục từ một máy chủ khác.

Việc sử dụng các kỹ thuật này có thể giúp cải thiện độ tin cậy của truyền dữ liệu UDP. Tuy nhiên, điều quan trọng cần nhớ là UDP không phải là giao thức đáng tin cậy. Do đó, không có gì đảm bảo rằng dữ liệu sẽ được truyền một cách chính xác. Nếu độ tin cậy là yêu cầu quan trọng, thì nên sử dụng giao thức đáng tin cậy như TCP.

**R7:**

**Host C có socket UDP với số cổng 6789. Host A và Host B đều gửi 1 phân đoạn UDP đến Host C với số cổng đích 6789. Cả 2 phân đoạn này có được chuyển đến cùng 1 socket tại Host C không? Nếu vậy, làm thế nào mà tiến trình tại Host C sẽ biết rằng 2 phân đoạn này đến từ 2 host khác nhau?**

**Có**, cả hai phân đoạn sẽ được chuyển đến cùng một socket tại Host C. Đối với mỗi phân đoạn được nhận, tại giao diện socket, hệ điều hành sẽ cung cấp các tiến trình để xác định IP nguồn của từng phân đoạn.

**P1:**

**Giả sử Client A tạo 1 phiên Telnet với Client S. Cùng thời điểm, Client B cũng tạo 1 phiên Telnet với Client S. Hãy cho biết số port khả thi của nguồn và đích đến:**

1. **Segment gửi từ A tới S**

Port của server S là 23, port của client A là số bất kì nằm trong khoảng 1024 tới 65535

1. **Segment gửi từ B tới S**

Port của server S là 23, port của client B là số bất kì nằm trong khoảng 1024 tới 65535

1. **Segment gửi từ S tới A**

Port của client A là 23, port của server S là số bất kì nằm trong khoảng 1024 tới 65535

1. **Segment gửi từ S tới B**

Port của client B là 23, port của server S là số bất kì nằm trong khoảng 1024 tới 65535

1. **Nếu A B dùng các host khác nhau, số port của segment gửi từ A tới S có giống của segment từ B tới S không?**

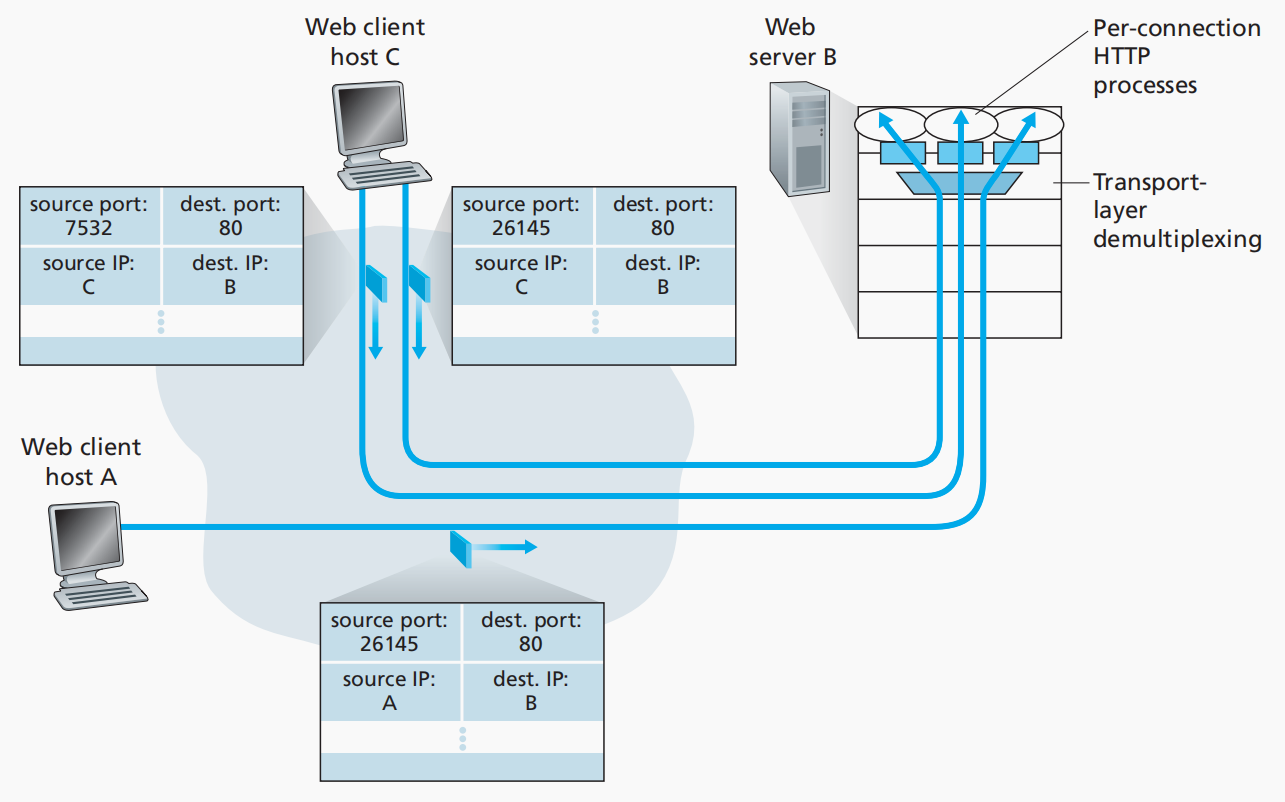
Có

1. **Nếu chung host thì sao?**

Không

**P2:**

**Hãy xem xét Hình 3.5. Các giá trị cổng nguồn (source port) và cổng đích (dest. port) trong các phân đoạn chảy từ máy chủ quay trở lại quy trình của khách hàng là gì? Các địa chỉ IP trong các datagram lớp mạng mang các phân đoạn lớp vận chuyển là gì?**

****

*Hình 3.5: Hai máy khách, sử dụng cùng một số cổng đích (80) để giao tiếp với cùng một ứng dụng máy chủ Web.*

Giả sử địa chỉ IP của các máy chủ A, B và C lần lượt là a, b, c. (a, b, c là khác nhau.)

**Đến máy chủ A:** source port = 80, địa chỉ IP nguồn = b, dest. port = 26145, địa chỉ IP đích = a.

**Đến máy chủ C, quy trình bên trái**: source port = 80, địa chỉ IP nguồn = b, dest. port = 7532, địa chỉ IP đích = c.

**Đến máy chủ C, quy trình bên phải**: source port = 80, địa chỉ IP nguồn = b, dest. port = 26145, địa chỉ IP đích = c.

1. **BÀI HỌC RÚT RA.**

* Biết được cách sử dụng bảng Hash.
* Hiếu được ý nghĩa của checksum và biết cách tính toán checksum.
* Giúp hiểu rõ 2 khái nhiệm Multiplexing và Demultiplexing, từ đó phân biệt được các điểm giống và khác nhau của chúng.
* Nắm được ý nghĩa của việc dùng các nguyên lý truyền tin tin cậy và biết được một số nguyên lý truyền tin khác nhau.
* Biết các mô hình FSM của rdt.

**\* Nhóm không có câu hỏi thắc mắc.**

**NGUỒN THAM KHẢO:**

* Slide bài giảng mạng máy tính.
* [Hash là gì ? 1 vài dạng hash cơ bản (pavietnam.vn)](https://kb.pavietnam.vn/hash-la-gi-1-vai-dang-hash-co-ban.html" \l ":~:text=M%E1%BB%A5c %C4%91%C3%ADch c%E1%BB%A7a hash function 1 1. Ki%E1%BB%83m,nhau. ... 2 2. X%C3%A1c minh m%E1%BA%ADt kh%E1%BA%A9u)
* [Checksum là gì? Dùng Checksum để làm gì? Ví dụ và giải thích - Wiki Máy Tính (wikimaytinh.com)](https://wikimaytinh.com/checksum-la-gi-dung-checksum-de-lam-gi.html" \l ":~:text=Checksum (t%E1%BB%95ng ki%E1%BB%83m tra) l%C3%A0 k%E1%BA%BFt qu%E1%BA%A3 c%E1%BB%A7a,b%E1%BA%A1n l%C3%A0 ch%C3%ADnh h%C3%A3ng v%C3%A0 kh%C3%B4ng c%C3%B3 l%E1%BB%97i.)
* [Multiplexing and Demultiplexing in Transport Layer - GeeksforGeeks](https://www.geeksforgeeks.org/multiplexing-and-demultiplexing-in-transport-layer/)
* [nguyên lý truyền dữ liệu đáng tin cậy - Tài liệu text (123docz.net)](https://text.123docz.net/document/2490640-nguyen-ly-truyen-du-lieu-dang-tin-cay.htm)
* [Reliable Data Transfer (RDT) 3.0 - GeeksforGeeks](https://www.geeksforgeeks.org/reliable-data-transfer-rdt-3-0/)