ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

NHẬP MÔN MẠNG MÁY TÍNH

LỚP: IT005.O118

BÁO CÁO BÀI TẬP 5 – NHÓM 12

Giảng viên hướng dẫn: ThS. Trần Mạnh Hùng

NoName – **"**Không tên nhưng không bao giờ vô danh**"**

MỤC LỤC

I. DANH SÁCH THÀNH VIÊN

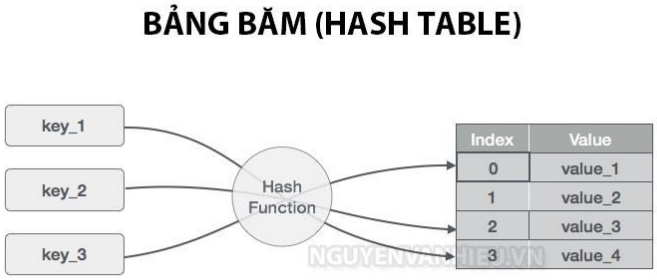
MSSV	Họ và tên	Phân công		Đánh giá
22521301	Mai Văn Tân (nhóm trưởng)	Trình bày báo cáo, câu 1, R4	Cùng kiểm tra lại tất cả các câu sau khi hoàn thành đáp án. Nhận xét, nêu thắc mắc tồn đọng.	100%
22520512	Nguyễn Bá Hưng	R4, P4		100%
22521539	Nguyễn Thị Trinh	Câu 4, Câu 5		100%
22521394	Trần Ỷ Thiên	Câu 1, Câu 3		100%
22520518	Nguyễn Thanh Hùng	R6, R7		100%
22520108	Nguyễn Gia Bảo	Câu 2, P1		100%

II. BÁO CÁO BÀI TẬP 5

Câu 1. Bảng Hash dùng để làm gì

Hashing được sử dụng rộng rãi trong việc **tìm kiếm** và **truy xuất dữ liệu** từ cơ sở dữ liệu có kích thước lớn. Bằng cách tạo các giá trị băm (hash values) cho dữ liệu, hệ thống có thể nhanh chóng xác định vị trí của dữ liệu cần truy cập, giúp giảm thời gian tìm kiếm.

Ngoài ra, Hash table còn được sử dụng để **tạo checksums** và **kiểm tra tính toàn vẹn** của dữ liệu. Checksums là các giá trị nhỏ được tạo ra từ dữ liệu, ví dụ như một tệp hoặc một khối dữ liệu cụ thể, nhằm kiểm tra xem dữ liệu đó có bị thay đổi không. Khi bạn tải xuống một tệp từ internet, chương trình kiểm tra tính toàn vẹn sẽ tạo ra một giá trị băm từ tệp tải về và so sánh nó với giá trị băm ban đầu của tệp gốc. Nếu hai giá trị băm này khác nhau, điều đó cho thấy tệp đã bị thay đổi hoặc hỏng.



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

Checksum là **một giá trị được tính toán từ một tập dữ liệu** để **xác minh tính toàn vẹn** của dữ liệu. Checksum được sử dụng trong nhiều ứng dụng, bao gồm **truyền dữ liệu**, **lưu trữ dữ liệu** và **kiểm tra lỗi**.

Checksum có thể được sử dụng để:

- **Xác minh tính toàn vẹn của dữ liệu**: Checksum có thể được sử dụng để xác minh rằng dữ liệu đã được truyền hoặc lưu trữ mà không bị thay đổi.
- **Kiểm tra lỗi**: Checksum có thể được sử dụng để phát hiện lỗi trong dữ liệu.
- **Tăng tốc độ truyền dữ liệu**: Checksum có thể được sử dụng để giảm thiểu số lượng gói dữ liệu cần thiết để truyền một tập dữ liệu.

Ví dụ: cho 2 dãy 16 bit: 1001 0001 1000 1100, 0001 0101 1001 0100.

Thực hiện phép cộng 2 chuỗi bit trên:

1001 0001 1000 1100

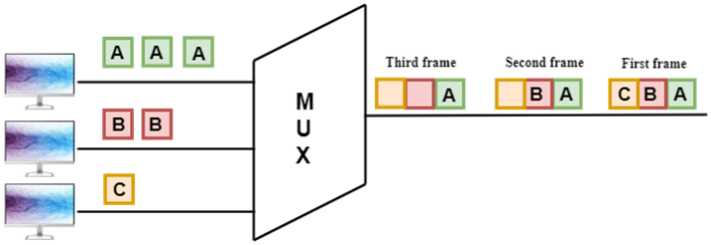
0001 0101 1001 0100

1010 0111 0010 0000 : sum

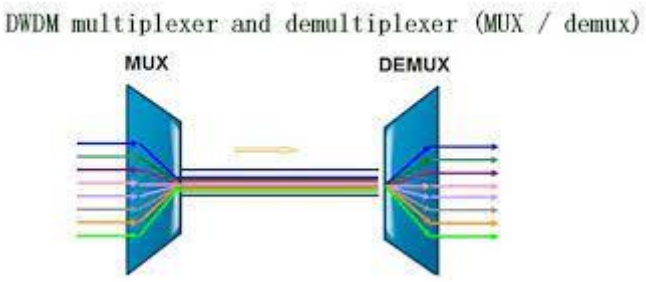
0101 1000 1101 1111 : checksum

Câu 3. Phân biệt Multiplexing vs Demultiplexing trong internet.

Multiplexing là một kỹ thuật được sử dụng để **kết hợp** và **gửi nhiều luồng dữ liệu** qua một phương tiện duy nhất. Quá trình kết hợp các luồng dữ liệu được gọi là Multiplexing và phần cứng được sử dụng để Multiplexing.



Demultiplexer hay mạch giải ghép kênh hay Demux là phần tử, thường là IC, bố trí ở cuối mạng **nhận dòng tín hiệu đã được ghép kênh trước đó** (ví dụ bởi một Multiplexer) ở ngõ vào đơn và chọn chuyển dữ liệu tới ngõ ra chọn lựa theo mã địa chỉ.



So sánh giữa Multiplexing và Demultiplexing:

Giống nhau:

- Mục tiêu chung: Cả hai Multiplexing và Demultiplexing đều liên quan đến việc **truyền** và **nhận dữ liệu qua mạng**.
- Liên quan đến đa truy cập: Cả hai quá trình đều liên quan đến việc **xử lý nhiều luồng dữ liệu** từ nhiều nguồn khác nhau.

Khác nhau:

	Multiplexing	Demultiplexing
Chức năng	**Tổng hợp nhiều luồng dữ liệu** thành một luồng duy nhất để truyền qua mạng.	**Tách luồng dữ liệu đầu vào** thành các luồng riêng biệt và định hướng chúng đến các ứng dụng hoặc dịch vụ cụ thể.
Vị trí trong quá trình truyền dữ liệu	Xảy ra tại **nguồn dữ liệu** hoặc tại một **điểm trung gian** trước khi dữ liệu được truyền đi.	Xảy ra tại **điểm đích** sau khi dữ liệu đã được truyền đi và đến tới máy chủ hoặc thiết bị đích.
Cấp độ tầng mạng	Xảy ra ở **nhiều tầng** khác nhau của mô hình OSI, nhưng **thường xảy ra ở tầng Transport** (tầng 4)	Xảy ra ở **tầng Transport** (tầng 4) hoặc **tầng Application** (tầng 7) trong mô hình OSI, tùy thuộc vào việc định hướng dữ liệu đến ứng dụng cụ thể.

➔ Tóm lại, Multiplexing và Demultiplexing đều liên quan đến quá trình truyền và xử lý dữ liệu trong mạng, nhưng chúng có chức năng và vị trí khác nhau trong quá trình này. Multiplexing tổng hợp dữ liệu thành một luồng duy nhất, trong khi Demultiplexing tách luồng dữ liệu đầu vào thành các luồng riêng biệt và định hướng chúng đến các ứng dụng hoặc dịch vụ cụ thể.

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.

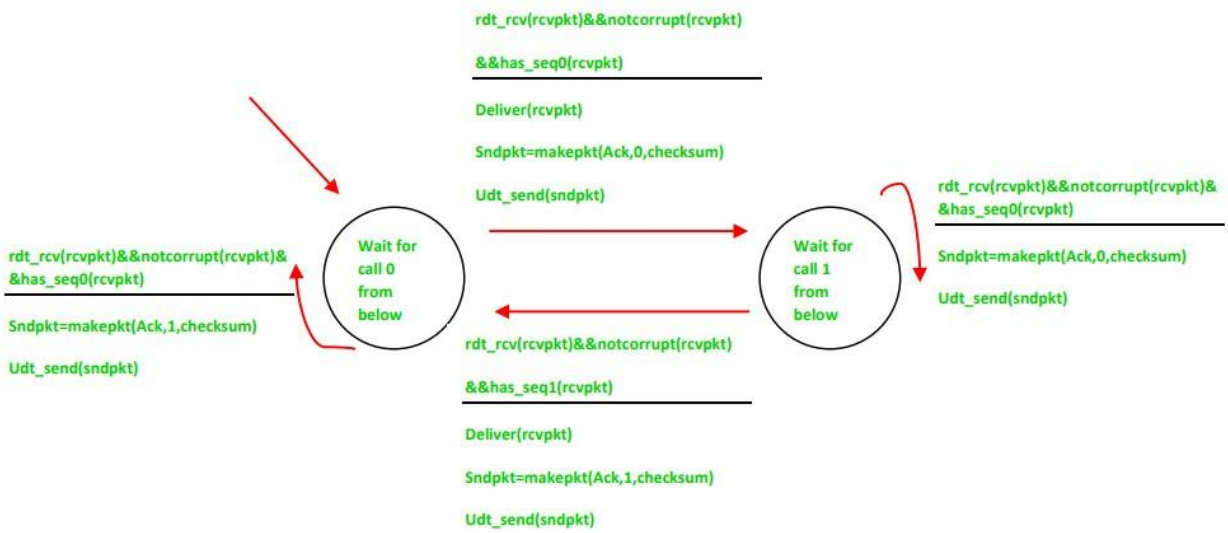
Các nguyên lý truyền tin tin cậy (reliable data transfer - RDT) được áp dụng trong việc truyền dữ liệu qua mạng để **đảm bảo tính tin cậy** và **đúng đắn** của thông tin. Việc sử dụng các nguyên lý RDT quan trọng vì nó giúp **giải quyết các vấn đề có thể xảy ra** trong quá trình truyền dữ liệu, như mất mát dữ liệu, lỗi bit, sự trễ trong việc truyền tải, và thứ tự không đúng của các gói tin.

So sánh các nguyên lý truyền tải:

Tên	Đặc điểm	Bên gửi	Bên nhận
rtd 1.0	Truyền dữ liệu tin cậy. (Giả sử không có lỗi hay mất mát gì)	Gửi gói tin	Nhận gói tin
rtd 2.0	Kênh truyền không làm mất gói. (Nhưng gói tin truyền có thể bị sai sót)	Gửi gói, đợi phản hồi (Stop and Wait Protocol)	Dùng checksum, ACK (Acknowledgement) và NAK (Negative ACK) để check lỗi gói tin gửi về sender
rtd 2.1	Kênh truyền có ACK, NAK bị lỗi (ACK, NAK bị sai checksum)	Đánh số thứ tự 0, 1 cho các gói từ sender, gửi lại gói nếu NAK hoặc ACK/NAK bị lỗi (nhận biết bằng checksum trong gói), nhận được ACK thì mới gởi gói đánh số tiếp theo	Có cơ chế loại packet bị trùng
rtd 2.2	Không dùng NAK, thay bằng gửi ACK gói gần nhất nhận thành công	Gửi lại gói nếu nhận ACK trùng lặp	Gửi ACK gói gần nhất thành công, đánh số thứ tự 0,1 cho ACK
rtd 3.0 => Hiệu suất thấp: Usender =	Xuất hiện mất gói	Chờ ACK trong khoảng thời gian "Hợp lí". Dùng Timeout (Bộ định thì). Gửi lại pckt khi hết thời gian. - Các trường hợp gởi lại gói: Mất gói, mất	

(L/R)/(RTT+L/R) do phải nhận ACK rồi mới gửi gói tiếp được		ACK, thời gian chờ ngắn/ delayed ACK	

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



Reliable Data Transfer (3.0): Receiver FSM

***Review Question 4*.**
Describe why an application developer might choose to run an application over UDP rather than TCP.

Hãy mô tả tại sao một nhà phát triển ứng dụng có thể chọn chạy ứng dụng qua UDP thay vì TCP.

Nhà phát triển ứng dụng có thể không muốn ứng dụng của mình sử dụng tính năng kiểm soát tắc nghẽn của TCP, điều này có thể làm giảm tốc độ gửi của ứng dụng vào những thời điểm bị tắc nghẽn. Thông thường, các nhà thiết kế ứng dụng điện thoại IP và hội nghị truyền hình IP chọn chạy ứng dụng của họ qua UDP vì họ muốn tránh sự kiểm soát tắc nghẽn của TCP. Ngoài ra, một số ứng dụng không cần truyền dữ liệu đáng tin cậy do TCP cung cấp.

***Review Question 6*.**
Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?

Có khả năng cho một ứng dụng thực hiện truyền dữ liệu đáng tin cậy ngay cả khi ứng dụng chạy qua UDP không? Nếu có, thì làm thế nào?

Có, nhà phát triển ứng dụng có thể thêm tính năng truyền dữ liệu đáng tin cậy vào giao thức tầng ứng dụng. Tuy nhiên, điều này đòi hỏi một lượng công việc đáng kể và debug để thực hiện.

***Review Question 7*.**
Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts.

Giả sử một tiến trình trên Máy C có một socket UDP với số cổng là 6789. Giả sử cả Máy A và Máy B đều gửi một đoạn UDP đến Máy C với cổng đích là 6789. Liệu cả hai đoạn này có được định tới cùng một socket trên Máy C không? Nếu có, thì làm thế nào tiến trình trên Máy C biết rằng hai đoạn này xuất phát từ hai máy khác nhau.

Có. Cả hai segments sẽ được chuyển hướng tới cùng 1 socket. Với mỗi segment, tại giao diện socket, hệ điều hành sẽ cung cấp process với địa chỉ IP để xác định xem nguồn gốc của từng segments.

***Problem 1*.**

Suppose Client A initiates a Telnet session with Server S. At about the same time, Client B also initiates a Telnet session with Server S. Provide possible source and destination port numbers for:

a. The segments sent from A to S.

b. The segments sent from B to S.

c. The segments sent from S to A.

d. The segments sent from S to B.

e. If A and B are different hosts, is it possible that the source port number in the segments from A to S is the same as that from B to S?

f. How about if they are the same host?

Giả sử Client A khởi tạo một phiên Telnet với Server S. Đồng thời, Client B cũng khởi tạo một phiên Telnet với Server S. Cung cấp số cổng nguồn và số cổng đích có thể sử dụng cho:

a. Các đoạn gửi từ A tới S.
- Cổng nguồn: Một số cổng nguồn ngẫu nhiên có sẵn trên Client A, đó là một số ngẫu nhiên nằm trong khoảng từ 1024 đến 65535.
- Cổng đích: Cổng Telnet (thường là cổng 23) trên Server S.

b. Các đoạn gửi từ B tới S.
- Cổng nguồn: Một số cổng nguồn ngẫu nhiên có sẵn trên Client B, đó là một số ngẫu nhiên nằm trong khoảng từ 1024 đến 65535.
- Cổng đích: Cổng Telnet (thường là cổng 23) trên Server S.

c. Các đoạn gửi từ S tới A.
- Cổng nguồn: Cổng Telnet (thường là cổng 23) trên Server S.
- Cổng đích: Cổng được sử dụng bởi Client A (cùng số cổng nhận được từ A đến S).

d. Các đoạn gửi từ S tới B.
- Cổng nguồn: Cổng Telnet (thường là cổng 23) trên Server S.
- Cổng đích: Cổng được sử dụng bởi Client B (cùng số cổng nhận được từ B đến S).

e. Nếu A và B là hai máy khác nhau, có khả năng số cổng nguồn trong các đoạn từ A tới S giống nhau so với B tới S không?

Nếu A và B là các máy khác nhau, có thể xảy ra trường hợp số cổng nguồn trong các đoạn tin từ A đến S giống số cổng từ B đến S. Điều này xảy ra vì số cổng nguồn được gán độc lập bởi các máy khách tương ứng A và B và không được phối hợp. Nhưng điều đó rất

không thường xảy ra bởi vì xác suất chọn cùng một số cổng ngẫu nhiên từ một khoảng 64512 giá trị có thể có là 1/64512, tương đương khoảng 0.0000155.

f. Còn nếu chúng là cùng một máy?

Nếu A và B là cùng một máy, không thể xảy ra trường hợp số cổng nguồn trong các đoạn tin từ A đến S giống số cổng trong các đoạn tin từ B đến S, vì TCP không cho phép hai kết nối có cùng bộ tứ (địa chỉ IP nguồn, số cổng nguồn, địa chỉ IP đích, số cổng đích). Nếu A và B cố gắng sử dụng cùng một số cổng nguồn, một trong hai sẽ không thể thiết lập kết nối với S.

***Problem 4*.**

a. Suppose you have the following 2 bytes: 01011100 and 01100101. What is the 1s complement of the sum of these 2 bytes?

b. Suppose you have the following 2 bytes: 11011010 and 01100101. What is the 1s complement of the sum of these 2 bytes?

c. For the bytes in part (a), give an example where one bit is flipped in each of the 2 bytes and yet the 1s complement doesn't change.

a. Giả sử bạn có 2 byte sau: 01011100 và 01100101. Bù 1 của tổng của 2 byte này là gì?

Cộng hai byte sẽ được 11000001. Lấy số bù một sẽ được 00111110.

b. Giả sử bạn có 2 byte sau: 11011010 và 01100101. Bù 1 của tổng của 2 byte này là gì?

Cộng hai byte sẽ được 01000000; phần bù một là 10111111.

c. Đối với byte trong phần (a), hãy cung cấp một ví dụ trong đó một bit trong mỗi trong 2 byte này được đảo và vẫn giữ nguyên giá trị bù 1 của tổng.

Byte đầu tiên = 01010100; byte thứ hai = 01101101.

III. NHẬN XÉT

Qua bài báo cáo bài tập 5 này, chúng em đã học được một số kiến thức cơ bản về mạng máy tính như:

- Bảng băm (Hash table) được sử dụng để tìm kiếm và truy xuất dữ liệu hiệu quả. Bảng băm cũng được dùng để tạo checksums và kiểm tra tính toàn vẹn của dữ liệu.
- Checksum là giá trị được tính từ dữ liệu để xác nhận tính toàn vẹn của dữ liệu. Checksum có thể detect lỗi truyền tin hoặc lưu trữ.
- Multiplexing và Demultiplexing liên quan đến việc truyền và xử lý nhiều luồng dữ liệu trong mạng. Multiplexing tổng hợp dữ liệu thành một luồng duy nhất, còn Demultiplexing thì tách luồng ban đầu ra thành nhiều luồng và định tuyến chúng.
- Các nguyên lý truyền tin tin cậy như RDT đảm bảo tính chính xác và đáng tin cậy của dữ liệu truyền qua mạng. Bạn đã phân tích sự khác nhau giữa các phiên bản RDT 1.0, 2.0, 2.1, 2.2 và 3.0.
- Hiểu được mô hình trạng thái hữu hạn (FSM) của giao thức RDT.
- Một số kiến thức cơ bản về cổng, giao thức UDP và TCP trong mạng máy tính.

IV. THẮC MẮC

Nhóm chúng em có 1 thắc mắc như sau:

1. Bài báo cáo có nêu ví dụ tính checksum 16 bit đơn giản. Nhưng trong thực tế, các gói tin có kích thước lớn hơn nhiều. Vậy làm thế nào để tính toán checksum cho các gói tin lớn hơn một cách hiệu quả?

V. NGUỒN THAM KHẢO

1. Slide bài giảng môn học
2. Computer Networking: A Top-Down Approach, 6th Edition By Kurros and Ross
3. https://vi.wikipedia.org/wiki/Demultiplexer
4. researchgate.net
5. https://www.geeksforgeeks.org/reliable-data-transfer-rdt-3-0/

- Hết -