

Báo cáo thực tập công ty an ninh mạng viettel: Networking basic

Bùi Hoàng Dũng

February 2024

Mục lục

| | |
|---|-----------|
| 1 Networking basic | 2 |
| 1.1 Mô hình OSI | 2 |
| 1.1.1 Physical Layer | 3 |
| 1.1.2 DataLink Layer | 3 |
| 1.1.3 Network Layer | 4 |
| 1.1.4 Transport Layer | 4 |
| 1.1.5 Session Layer | 4 |
| 1.1.6 Presentation Layer | 5 |
| 1.1.7 Application Layer | 5 |
| 1.2 Mô hình TCP/IP | 5 |
| 1.3 Domain Name System | 6 |
| 1.4 File Transfer Protocol | 9 |
| 1.5 Simple Mail Transfer Protocol | 10 |
| 1.6 HTTP | 11 |
| 1.7 Telnet và SSH | 14 |
| 1.8 Client-server model | 15 |
| 1.9 Connectionless và connection-oriented | 15 |
| 1.10 TCP và UDP | 16 |
| 2 Basic Switching | 20 |
| 2.1 Ethernet frame | 20 |
| 2.2 Hoạt động của Switch trong mạng | 21 |
| 2.3 CSMA | 22 |
| 2.4 Half-duplex và Full-duplex | 25 |
| 2.5 Broadcast domain và collision domain | 26 |
| 2.6 Spanning tree protocol | 28 |
| 2.7 VLAN | 35 |
| 2.8 Inter VLAN Routing | 38 |
| 2.9 VTP | 39 |
| 3 LAB | 40 |
| 3.1 Configure IP | 40 |
| 3.2 VLAN | 42 |
| 3.3 STP | 44 |

1 Networking basic

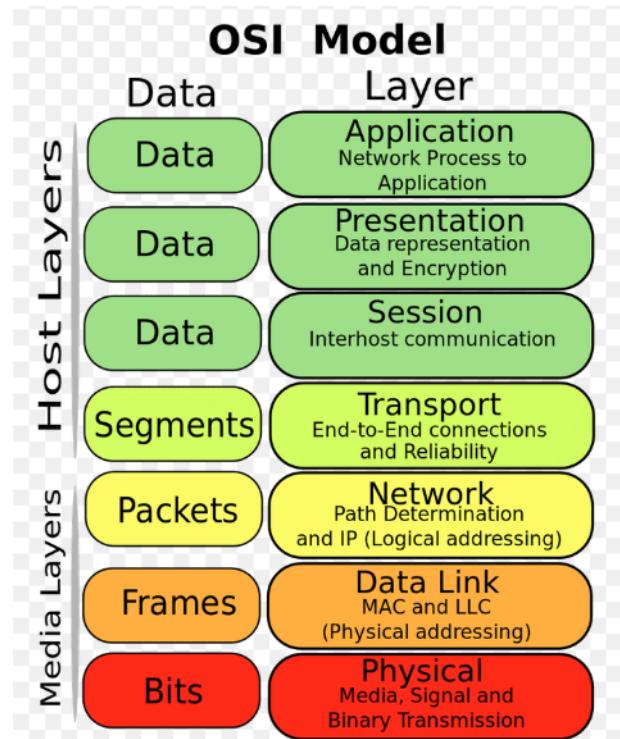
1.1 Mô hình OSI

- Mô hình OSI: được viết tắt là Open System Interconnection được phát triển bởi tổ chức ISO (International Organization for Standardization) vào năm 1984. Mô hình OSI bao gồm 7 lớp, trong đó mỗi lớp đều có những chức năng nhiệm vụ khác nhau trong việc chuyển dữ liệu trên mạng internet. Mô hình OSI là một mô hình đã được trừu tượng hóa, được đưa ra nhằm giải thích cho việc chuyển dữ liệu giữa các máy tính.

- Mô hình OSI có 7 lớp thành phần bao gồm những lớp như sau:

- Application Layer
- Presentation Layer
- Session Layer
- Transport Layer
- Network Layer
- Datalink Layer
- Physical Layer

- Mỗi lớp trong mô hình OSI đều có chức năng và nhiệm vụ riêng trong việc truyền dữ liệu trong mạng máy tính.



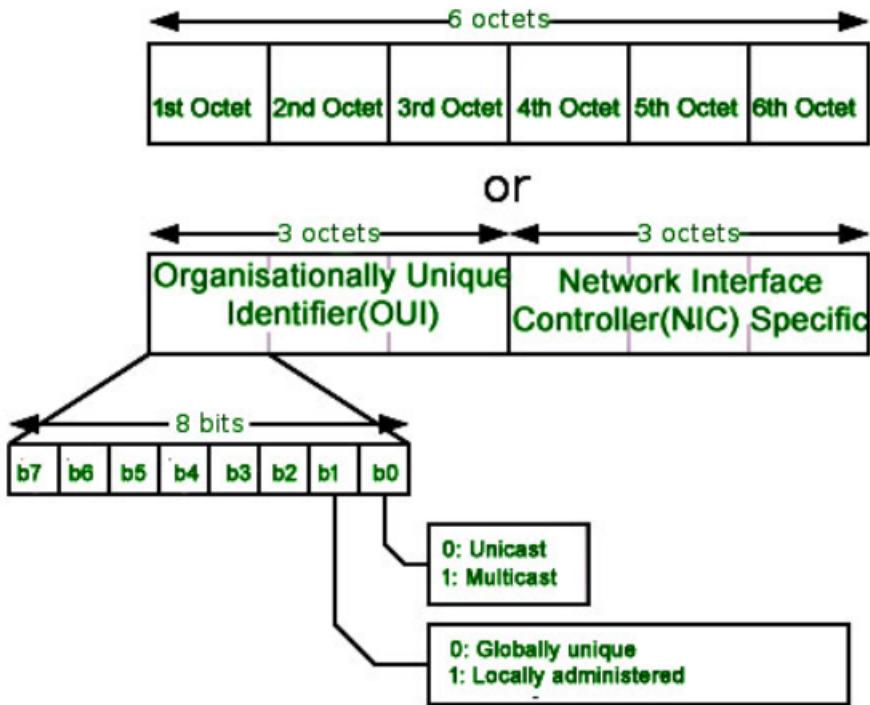
Hình 1: OSI model

1.1.1 Physical Layer

- Là lớp dưới cùng trong mô hình OSI đại diện cho những liên kết vật lý và các mức điện áp trong một hệ thống truyền thông. Những thiết bị như là connector, receiver, các loại cáp,..... đều thuộc vào lớp vật lý trong mô hình OSI. Lớp vật lý gửi các bit dữ liệu từ một thiết bị như máy tính tới các thiết bị khác. Lớp vật lý định nghĩa cách thức mã hóa dữ liệu như thế nào (bit 0 và bit 1 được mã hóa như thế nào trong tín hiệu). Lớp vật lý chịu trách nhiệm cho việc giao tiếp của các dữ liệu không cấu trúc được truyền qua một kênh truyền vật lý. Ngoài ra, lớp vật lý còn thực hiện các chức năng như sau: duy trì tỉ lệ dữ liệu (người gửi có thể gửi được bao nhiêu bit mỗi lần gửi), thực hiện việc đồng bộ các bits, Khi nhận được dữ liệu thì lớp này sẽ chuyển tín hiệu đó thành các bit 0 và bit 1 sau đó gửi các bit này tới lớp Datalink để chuyển các bit đó vào thành một frame. Physical layer còn có thể điều chỉnh được cách mà data được chuyển giữa 2 thiết bị được kết nối với nhau: simplex, half-duplex, full-duplex. Những thiết bị mạng như: Hub, repeater, modem và các loại cáp được hiểu là các thiết bị lớp vật lý.

1.1.2 DataLink Layer

- Là lớp chịu trách nhiệm cho việc chuyển tiếp tin nhắn từ node này đến node khác. Nhiệm vụ của Data Link Layer là đảm bảo rằng dữ liệu được chuyển đi từ node này sang node khác không bị lỗi thông qua kênh truyền vật lý. Khi một gói tin được chuyển trong mạng, nhiệm vụ của Data Link Layer là chuyển tiếp gói tin đó đến đích bằng cách sử dụng địa chỉ MAC (Media Access Control). Data Link Layer được chia ra làm 2 lớp con: Logical Link Control và Media Access Control. Logical Link Control đảm bảo việc đồng bộ, ghép kênh và kiểm tra cũng như sửa lỗi. Còn MAC được coi như là địa chỉ đặc trưng cho một thiết bị riêng biệt, địa chỉ MAC được gán riêng cho mỗi network interface card trong từng thiết bị. - Mỗi 1 địa chỉ MAC sẽ bao gồm 48 bit trong đó: 3 octets đầu (24bit) là định danh cho tổ chức hay đơn vị sản xuất ra card mạng đó, 2 bit cuối của octets đầu tiên trong địa chỉ MAC sẽ thể hiện đây là địa chỉ unicast hay là multicast, là địa chỉ riêng trong toàn thế giới hay chỉ trong một vùng nhỏ. Kích thước của mỗi gói tin nhận hay gửi được sẽ được chia nhỏ hay không phụ thuộc vào frame size của từng NIC. Data Link Layer cũng thực hiện việc đóng gói địa chỉ MAC của người gửi và người nhận trong phần header của mỗi frame.



Hình 2: Mac address

- Các chức năng của lớp Data Link Layer: framing, physical addressing, Error control, Flow control, Access control. Switch và bridge là những thiết bị mạng thuộc lớp Data Link.

1.1.3 Network Layer

- Network layer hoạt động để chuyển tiếp dữ liệu từ một host đến một host được đặt tại một mạng khác với host gửi. Ngoài ra Network layer còn đảm nhiệm chức năng routing (tìm đường đi) cho các gói tin trong mạng để lựa chọn quãng đường tối ưu nhất. Địa chỉ IP của host gửi và host nhận sẽ được đóng gói vào phần header của gói tin bởi network layer. Địa chỉ IP là một logical Addressing được đưa ra để xác định các thiết bị trong mạng, dựa vào địa chỉ IP ta có thể xác định được nguồn cũng như đích của các gói tin.

1.1.4 Transport Layer

- Transport layer chịu trách nhiệm truyền và trao đổi dữ liệu giữa các host qua kết nối end-to-end, dữ liệu được truyền trong lớp giao vận được gọi là các segments. Ngoài ra, lớp giao vận còn đảm nhận nhiệm vụ sửa lỗi điều khiển luồng, cũng như chia nhỏ gói tin thành các segment, sau đó lắp ráp lại thành gói tin hoàn chỉnh ở trạm đích. Lớp giao vận cũng thêm vào header của gói tin địa chỉ cổng đích và cổng nguồn để các ứng dụng trong từng host khác nhau có thể giao tiếp được với nhau. Những giao thức hay được sử dụng ở lớp giao vận đó là: TCP, UDP,....

1.1.5 Session Layer

- Lớp phiên đảm nhận chức năng thiết lập kết nối giữa các host, thực hiện duy trì các kết nối. Lớp phiên có thể cho phép 2 tiến trình ở các host có thể được khởi tạo, sử dụng và hủy kết nối. Session Layer cho

phép một tiến trình có thể thêm checkpoints để có thể đồng bộ dữ liệu được gửi cung như phát hiện lỗi, mất mát dữ liệu.

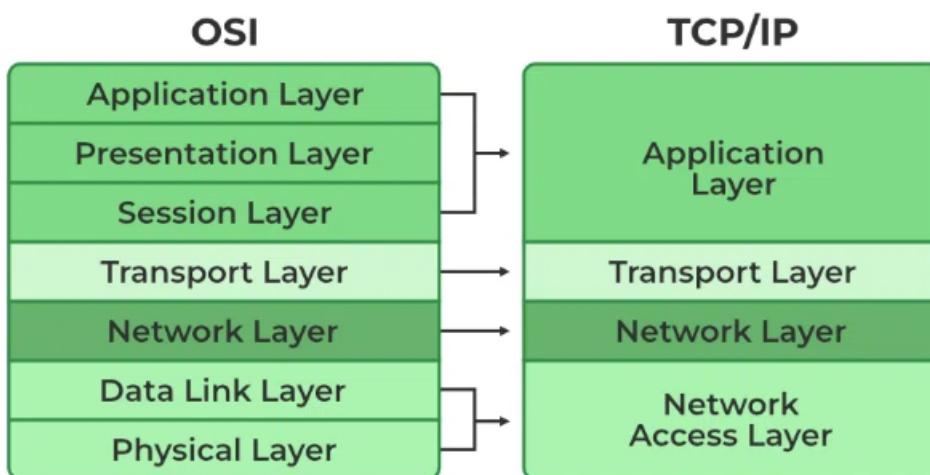
1.1.6 Presentation Layer

- Presentation Layer thực hiện công việc dịch những bit dữ liệu thành các định dạng khác nhau để truyền lên lớp Application cung cấp cho người dùng những dữ liệu có ý nghĩa. Presentation Layer có thể thực hiện việc mã hóa dữ liệu, nén dữ liệu để sao cho việc truyền dữ liệu được an toàn bảo mật, và nhanh nhất có thể.

1.1.7 Application Layer

- Lớp ứng dụng là lớp trên cùng trong mô hình OSI, Lớp ứng dụng thực hiện triển khai những Network Application thực hiện nhiệm vụ hiển thị các thông tin dữ liệu được truyền qua mạng cũng như nhận dữ liệu từ người dùng.

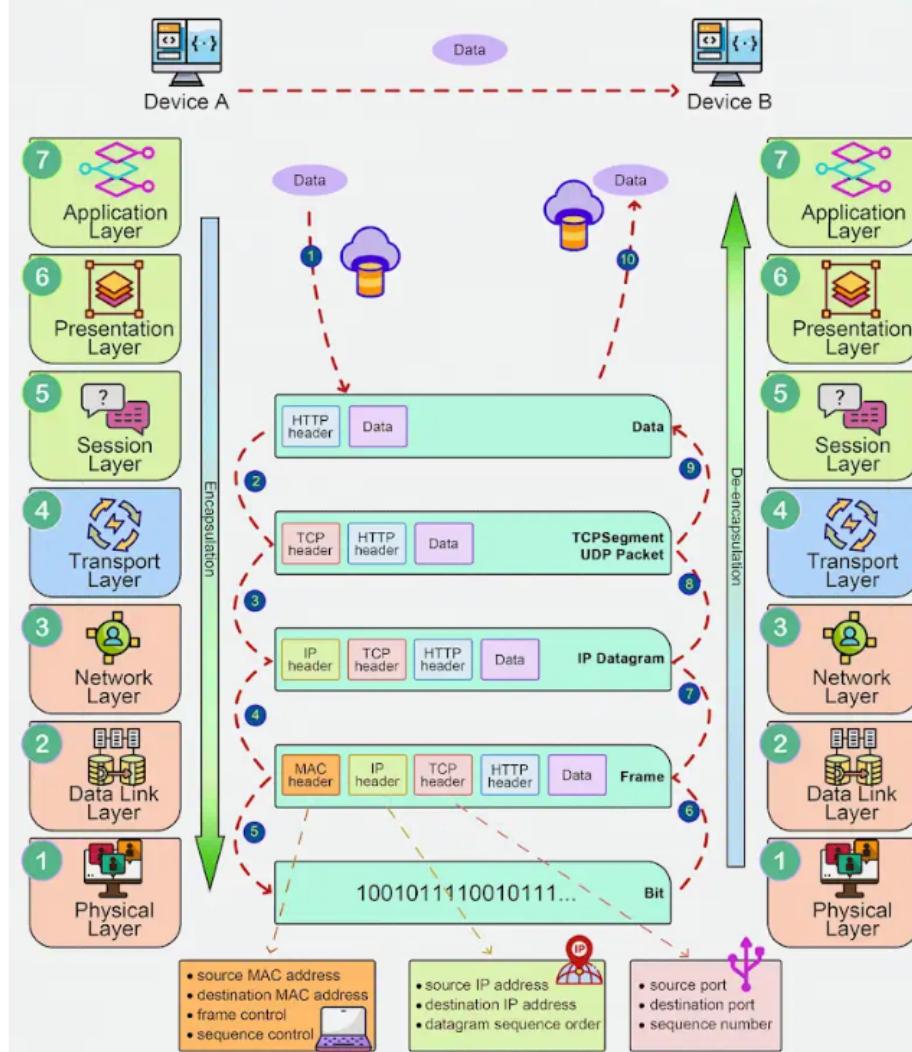
1.2 Mô hình TCP/IP



Hình 3: TCP/IP model

- Mô hình TCP/IP được coi là một phiên bản rút gọn của mô hình OSI, và ra đời vào năm 1960 bởi tổ chức DoD (Department of Defense). Mô hình TCP/IP bao gồm 4 lớp: Application Layer, Transport Layer, Network Access Layer. Dựa vào hình trên ta có thể thấy: lớp Application, lớp Presentation, lớp Session trong mô hình OSI được gộp thành lớp Application trong mô hình TCP/IP. Về mục đích cũng như vai trò của từng lớp trong mô hình TCP/IP tương đương với các thành phần tương ứng trong mô hình OSI. Mục đích chính của mô hình TCP/IP là giúp mọi người có cái nhìn ngắn gọn hơn về việc dữ liệu được chuyển tiếp như thế nào trong mạng máy tính. Mô hình TCP/IP cung cấp khả năng truyền dữ liệu một cách tin cậy bằng cách sử dụng giao thức TCP có thể phát hiện lỗi từ các gói tin. Mô hình TCP/IP cho phép các gói tin được gửi đến đích đã định sẵn bằng cách thêm thông tin dẫn đường vào các gói tin. IP (Internet Protocol) đảm bảo gói tin đến được đúng đích ban đầu.

- Encapsulation (đóng gói): là cách thêm vào gói tin trong mạng những phần header của từng lớp một với một đơn vị dữ liệu cụ thể (payload) hay còn thể hiểu là đóng gói dữ liệu từ tầng trên vào trong các khung dữ liệu của tầng dưới. Ví dụ trong gói tin IP nội dung của một trang web được đóng gói với phần header ngoài cùng là http sau đó là phần header của TCP và header của IP cuối cùng là header của khung dữ liệu. Khung dữ liệu này được chuyển đến nút đích dưới dạng một chuỗi bit, nơi nó được giải mã thành các đơn vị dữ liệu giao thức tương ứng và được hiểu tại mỗi tầng của nút nhận. Việc đóng gói các gói tin giúp các thiết bị mạng có thể hiểu được gói tin được truyền đi đâu cũng như truyền như thế nào dựa vào phần header được đóng gói của từng gói tin.



Hình 4: Caption

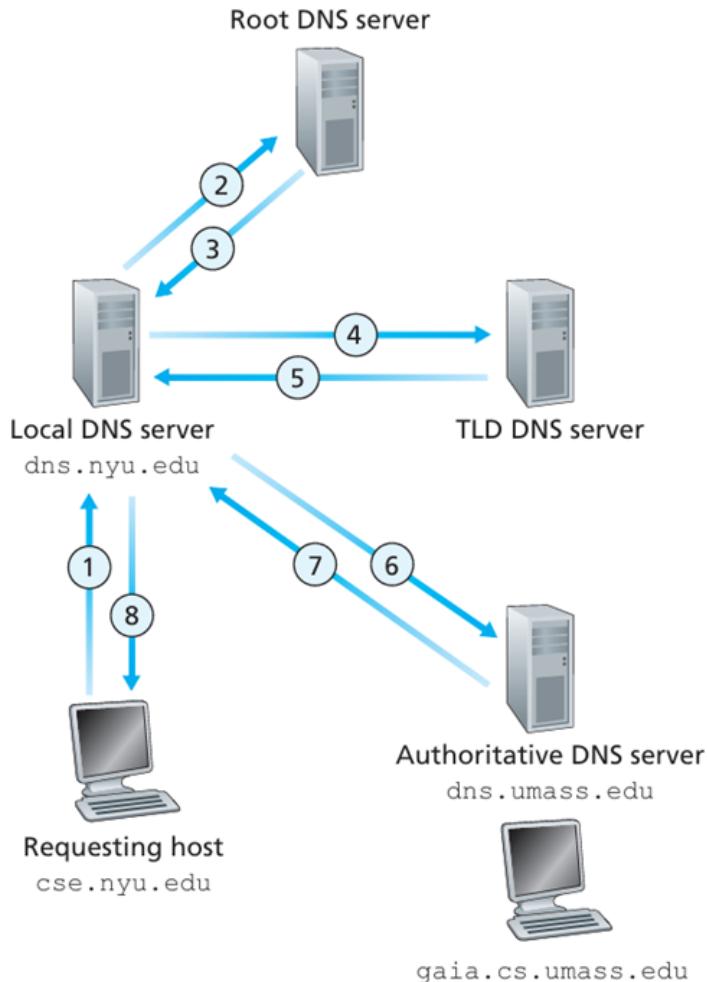
1.3 Domain Name System

- DNS (Domain Name System): là một giao thức chuyển đổi tên miền thành địa chỉ IP, người dùng khi muốn sử dụng dịch vụ của một server thay vì phải nhớ địa chỉ IP của server thì chỉ cần phải nhớ tên miền của nó, tên miền thường là những cái tên dễ nhớ. DNS là một cơ sở dữ liệu cấp bậc rời rạc trong DNS server và là một application layer protocol cho phép host có thể truy vấn cơ sở dữ liệu đó. DNS có một thiết kế rời rạc và cơ sở dữ liệu phân cấp.

- Có 3 loại máy chủ DNS bao gồm:

- Root DNS server.
- TLD DNS server (Top-level domain)
- Authoritative DNS server

- Ngoài ra còn có một loại DNS server khác gọi là Local DNS server thường được cung cấp và quản lý bởi ISP, Local DNS server này hoạt động như một proxy chuyển tiếp DNS query đến với Root DNS server.



Hình 5: Caption

Hình trên diễn tả quá trình mà một host (cse.nyu.edu) thực hiện DNS query để tìm địa chỉ IP của host name (gaia.cs.umass.edu). Quá trình DNS request/reply được thực hiện một cách đệ quy thông qua Local DNS server của host. Đầu tiên host gửi request đến cho Local DNS server, sau khi nhận được request của host thì Local DNS server chuyển tiếp request đó đến với Root DNS server, Root DNS server thực hiện việc tìm kiếm trong bản ghi của mình những resource có .edu sau đó trả lại cho Local DNS server một list các địa chỉ IP của TLD server của tên miền .edu. Sau khi nhận được một list các địa chỉ

IP trên, Local DNS server thực hiện gửi query messages cho một trong những TLD server nhận được. Sau khi nhận được messages từ Local DNS server thì TLD server sẽ thực hiện tìm kiếm tiếp tên miền .umass.edu và gửi lại về cho Local DNS server địa chỉ IP của authoritative server (dns.umass.edu). Sau khi có được địa chỉ IP của authoritative server thì Local DNS server sẽ gửi query message đến thẳng authoritative server và tìm ra được địa chỉ IP của tên miền mà host cần.

- DNS caching là một hình thức lưu lại một cặp hostname và địa chỉ IP tương ứng vào trong bộ nhớ đệm của một DNS server. Khi mà có một query request đến với yêu cầu tìm địa chỉ IP với hostname đã được lưu trong bộ đệm thì DNS server sẽ thực hiện cung cấp luôn địa chỉ IP đó (dù có không phải là authoritative server). Tuy nhiên cặp địa chỉ này chỉ được lưu trong một khoảng thời gian nhất định thường là 2 ngày.

- **DNS record và messages:** DNS record thường được gọi là những resource record mỗi resource record sẽ có cung cấp một ánh xạ từ hostname sang địa chỉ IP. Mỗi DNS reply message sẽ có thể mang một hoặc nhiều resource record. Một resource record bao gồm 4 trường: **Name**, **Value**, **Type**, **TTL**. Ứng với mỗi giá trị của Type thì sẽ có một name và value khác nhau. Còn TTL ở đây là Time to live xác định khoảng thời gian mà resource đó bị loại khỏi bộ nhớ của server.

Type = A thì name sẽ là hostname, values sẽ là địa chỉ IP

Type = NS thì name sẽ là tên miền, còn value là hostname của authoritative server

Type = CNAME thì name sẽ là tên miền còn value là hostname là một alias của name

Type = MX thì name sẽ là tên miền còn value là hostname của mail server là một alias của name

Nếu DNS server là một authoritative server của host name thì DNS server sẽ chứa bản ghi loại A. Nếu một server không phải là authoritative server thì server sẽ chứa bản ghi loại NS cho tên miền và bao gồm cả host name.

DNS message bao gồm 2 loại là DNS query và reply message, và có chung một format: Trường đầu tiên ở phần header 12 bytes là Identification. Trường này dùng để matching query đã gửi với reply message đã nhận được. Trường Flags để diễn tả những bit có được set xem nó có là bản tin query hay reply,.....

| Identification | Flags | |
|---|--------------------------|--|
| Number of questions | Number of answer RRs | 12 bytes |
| Number of authority RRs | Number of additional RRs | |
| Questions (variable number of questions) | | Name, type fields for a query |
| Answers (variable number of resource records) | | RRs in response to query |
| Authority (variable number of resource records) | | Records for authoritative servers |
| Additional information (variable number of resource records) | | Additional "helpful" info that may be used |

Hình 6: Caption

1.4 File Transfer Protocol

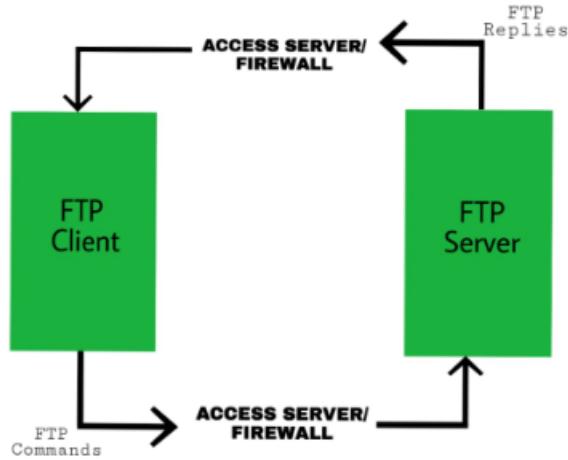
File transfer protocol (FTP) là một giao thức giúp chuyển file từ một máy tính này sang máy tính khác bằng cách cung cấp khả năng truy cập thư mục hoặc tệp tin từ một máy tính khác cho phép những phần mềm, dữ liệu, text file được chuyển thông qua internet. End-user trong kết nối FTP được gọi là localhost còn server mà cung cấp dữ liệu thì được gọi là remote host.

FTP được hoạt động dựa trên mô hình client-server. FTP client là một chương trình chạy trên máy tính của người dùng cho phép người dùng có thể kết nối và có được file từ remote host. Chương trình này bao gồm một tệp lệnh để thiết lập kết nối giữa 2 host, trao đổi file hoặc có thể đóng kết nối.

Kết nối FTP được tạo giữa 2 hệ thống và chúng giao tiếp với nhau sử dụng một mạng truyền thông. Do đó để thiết lập kết nối với server thì người dùng phải được cấp quyền từ server. Khi một kết nối FTP được thiết lập thì sẽ có 2 loại kênh truyền thông được thiết lập đó là command channel và data channel. Command channel được sử dụng để chuyển command và phản hồi từ client đến server và từ server đến với client, còn data channel sẽ được dùng để chuyển dữ liệu. Chi tiết về hoạt động của FTP được mô tả như sau:

- FTP client giao tiếp với FTP server ở port 21 sử dụng giao thức TCP
- Client sẽ được xác thực bởi server và có được quyền sử dụng kết nối với server
- Client thực hiện tương tác với thư mục trên server bằng cách gửi những command thông qua kết nối.
- Khi nhận được command để truyền dữ liệu, server mở một liên kết TCP data đến client.
- Sau khi đã truyền được file server sẽ thực hiện đóng kết nối này.

- Server mở một liên kết TCP data thứ hai đến client, và cứ như thế đến khi nào mà nhận được command đóng kết nối từ client thì thôi.



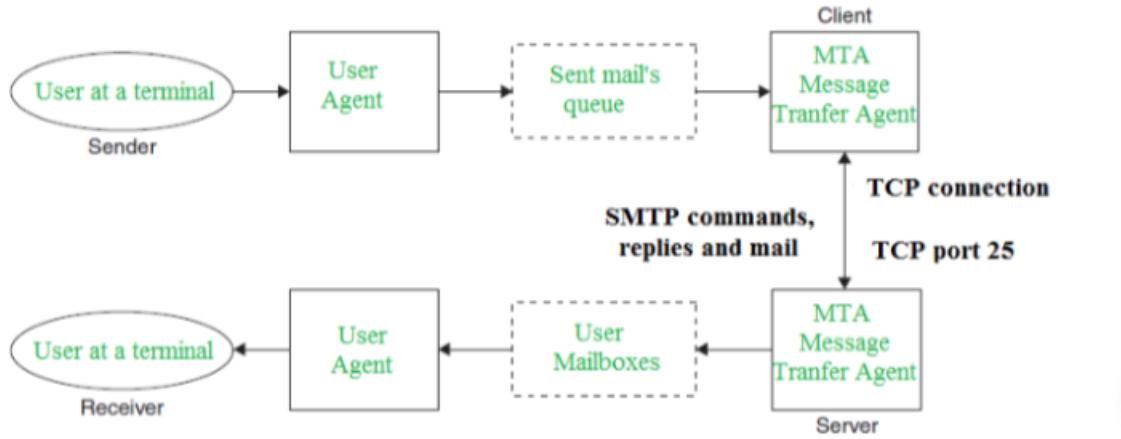
Hình 7: Caption

1.5 Simple Mail Transfer Protocol

SMTP (Simple Mail Transfer Protocol) là một loại giao thức thuộc lớp application. Client người muốn gửi một mail sẽ mở một phiên kết nối TCP đến SMTP server và sau đó gửi mail thông qua kết nối này. SMTP server sẽ luôn luôn được đặt ở trạng thái listen. Ngay sau khi, nghe được kết nối từ những client, SMTP server sẽ khởi tạo một kết nối qua port 25, sau khi thiết lập xong một phiên kết nối TCP, client sẽ thực hiện việc gửi mail đến server.

Mô hình SMTP gồm 2 loại:

- End-to-end: được sử dụng để giao tiếp giữa các tổ chức khác nhau.
- Store-and-forward: được sử dụng để giao tiếp trong một tổ chức.



Hình 8: Caption

Các thành phần của SMTP:

- - Mail user agent (MUA): là một chương trình máy tính giúp cho việc gửi và nhận mail, chịu trách nhiệm cho việc tạo tin nhắn để truyền tới Mail transfer agent (MTA).
- Mail Submission Agent (MSA): là một chương trình máy tính nhận mail từ một MUA và tương tác với MTA để chuyển mail.
- Mail Transfer Agent (MTA): là một phần mềm thực thi công việc gửi mail từ một hệ thống này sang hệ thống khác.
- Mail Delivery Agent (MDA): là một hệ thống giúp việc chuyển mail đến đúng với local system

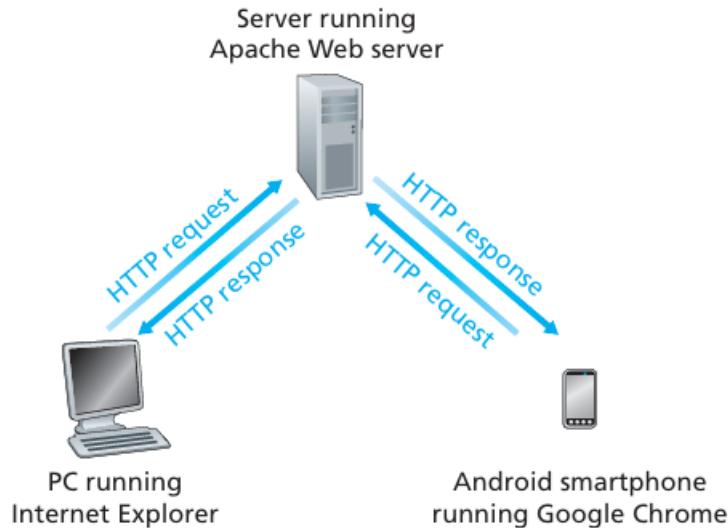
Hoạt động của SMTP:

1. Sender's user agent chuyển bị tin nhắn và gửi nó đến cho MTA. MTA nhận được và gửi mail thông qua mạng đến cho MTA ở bên nhận. Để gửi được mail yêu cầu cả bên gửi và bên nhận đều phải có MTA.
2. Mail được gửi bởi một loạt những tin nhắn request và response giữa client và server. Tin nhắn được gửi sẽ bao gồm phần header và phần body. Null line sẽ được dùng để hủy đi phần header và sau null line này sẽ là phần nội dung của tin nhắn.
3. User agent ở bên phía server thực hiện kiểm tra mailboxes liên tục trong từng khoảng thời gian một. Nếu có thông tin nào được nhận thì nó sẽ thông báo cho bên server về tin nhắn đó. Mailboxes sẽ cung cấp cho người dùng một list các mail với những short description đi kèm với mỗi mail.

1.6 HTTP

HTTP (Hyper Text Transfer Protocol) là một loại giao thức ở lớp ứng dụng. HTTP được triển khai ở trong 2 chương trình: chương trình ở trên client và chương trình ở trên server. Chương trình ở trên client và chương trình ở trên server được chạy trên các end system khác nhau và có thể tương tác với nhau bằng cách trao đổi HTTP messages.

Một Web page sẽ bao gồm nhiều objects. Objects ở đây có thể là một HTML file, một file JPEG, hoặc một video,...và có địa chỉ là một URL. Mỗi URL thường có 2 phần: hostname và tên của đường dẫn đến objects (object's path name). Web browsers (Edge, Chrome, FireFox) được triển khai ở phía client, còn Web server sẽ được triển khai ở phía server là nơi để lưu trữ các web object mỗi object này đều có một URL làm địa chỉ riêng. HTTP sẽ định nghĩa cách mà web client request đến web page của một web server và web server chuyển web page đó đến client như thế nào.



Hình 9: Caption

Khi một user request một web page, browser sẽ gửi HTTP request message cho object ở trong web page đó. Server nhận được request và sẽ phản hồi lại bằng HTTP response message bao gồm object đó. HTTP sử dụng TCP như là một giao thức để truyền thông phía bên dưới. HTTP client sẽ khởi tạo một phiên kết nối TCP đến server. Khi mà kết nối được thiết lập, browser và server sẽ thực hiện quá trình kết nối với nhau thông qua socket interface. Client sẽ gửi HTTP request message đến socket interface của nó và nhận được HTTP response message từ chính socket này.

VD: Khi thực hiện chuyển một web page từ server sang client:

- - HTTP client sẽ thực hiện khởi tạo một kết nối TCP tới server ở port 80, đi cùng với kết nối TCP này sẽ lại một socket interface ở bên phía client và một socket interface ở bên phía server.
- HTTP client sẽ gửi HTTP request message thông qua socket của nó, request message sẽ bao gồm path name của object
- HTTP server nhận được request message từ socket của nó và thực hiện tìm object được request từ path name và đóng gói object đó lại vào một HTTP response message gửi đến client thông qua socket.
- HTTP server thực hiện đóng kết nối TCP thông qua một bản tin TCP terminated

```

HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 .....
Content-Length: 6821
Content-Type: text/html

data data data data data ...

```

Hình 11: Http-response

- HTTP client nhận được response message, TCP connection terminates, và object được đóng gói(có thể là HTML file, video, file ảnh). Client thực hiện việc giải mã file từ response message.

VD kể trên là về HTTP non-persistent connections, với HTTP persistent connections thì với những object được request phía server sẽ không cần phải đóng lại kết nối TCP cho mỗi object mà sử dụng luôn kết nối TCP đó cho mỗi object được request tiếp theo, server chỉ thực hiện đóng kết nối TCP khi nó không còn sử dụng kết nối TCP đó nữa. HTTP message format:

```

GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr

```

Hình 10: HTTP-request

Hình trên là một HTTP request message, dòng đầu tiên là request line, những dòng phía sau được gọi là header line. Request line sẽ gồm 3 thành phần: method file (GET, POST, HEAD, PUT, DELETE), object's path name, phiên bản của HTTP . Ở đây sử dụng GET method khi mà browser yêu cầu một object với path name được xác định. Host ở đầu là chỉ hostname của host chứa object đó. Connection: close ở đây chỉ khi mà thực hiện xong quá trình truyền thì bên server sẽ đóng luôn kết nối TCP. User agent là loại browser sử dụng, Accept-language thể hiện user muốn nhận được phiên bản có loại ngôn ngữ nào hơn.

Với các method khác như là POST được sử dụng khi người dùng muốn cung cấp thông tin hay điền một nội dung vào một form nào đó (vd google search), HEAD giống với GET nhưng mà sẽ bỏ đi phần object request thường được dùng để debug, PUT thường được sử dụng để tương tác với Web page nó cho phép người dùng upload một object đến một đường dẫn đặc biệt ở bên Web server, còn DELETE thường được sử dụng để xóa một object bên web server.

Hình trên mô tả một HTTP response message cũng gồm 3 thành phần: initial status line, header lines, entity body. Các thành phần tương tự đã được giải thích phía bên trên. Ở đây, mình chỉ đề cập

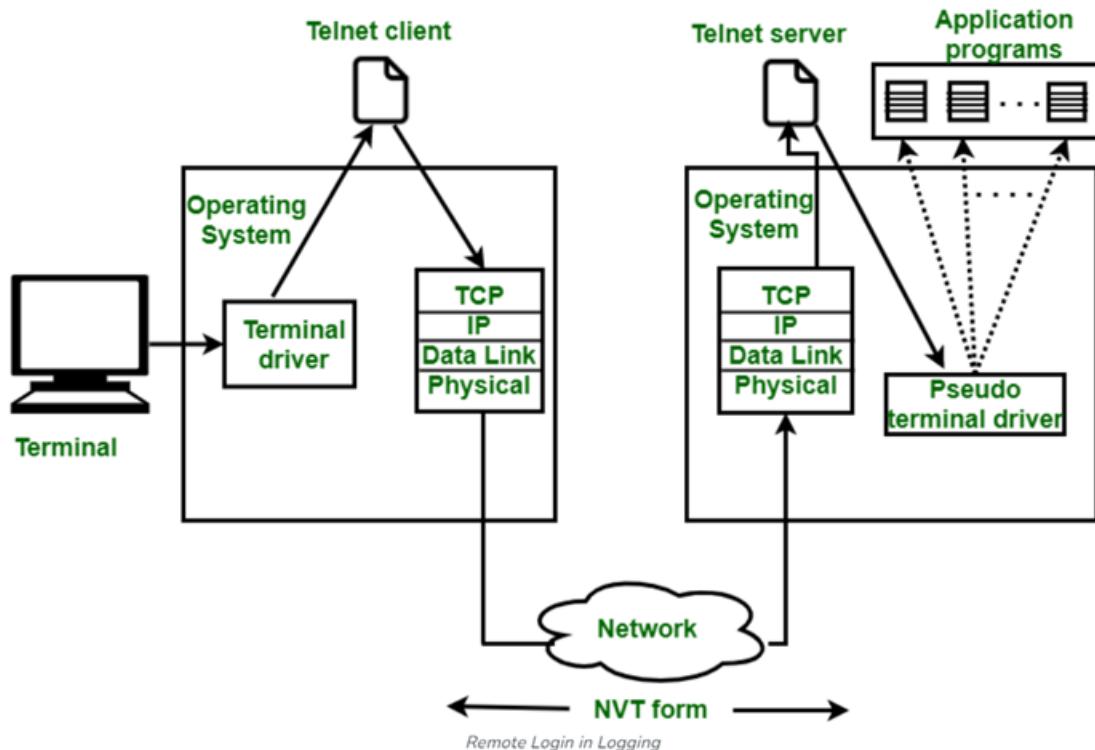
đến status code:

- 200 OK: request thành công và dữ liệu đang được truyền về
- 301 Moved Permanently: request object đã được di chuyển đi đâu đó, URL mới sẽ được thông báo ở trường Location trong phần header
- 400 Bad Request: Request không được hiểu ở bên server (request lỗi)
- 404 Not Found: Request object không có ở bên server
- 505 HTTP Version Not Supported: HTTP version không được hỗ trợ bởi server

1.7 Telnet và SSH

Telnet (Teletype Network) là một loại giao thức cho phép một máy tính có thể kết nối và truy cập đến với máy tính khác, telnet được triển khai theo mô hình client-server cung cấp một kênh giao tiếp hai chiều cho phép tương tác giữa 2 máy tính. Telnet thực hiện lệnh người dùng theo giao thức TCP/IP để tạo các session từ xa, cung cấp CLI để kết nối với remote computer.

Telnet gửi dữ liệu theo plaintext nên không an toàn về mặt bảo mật.

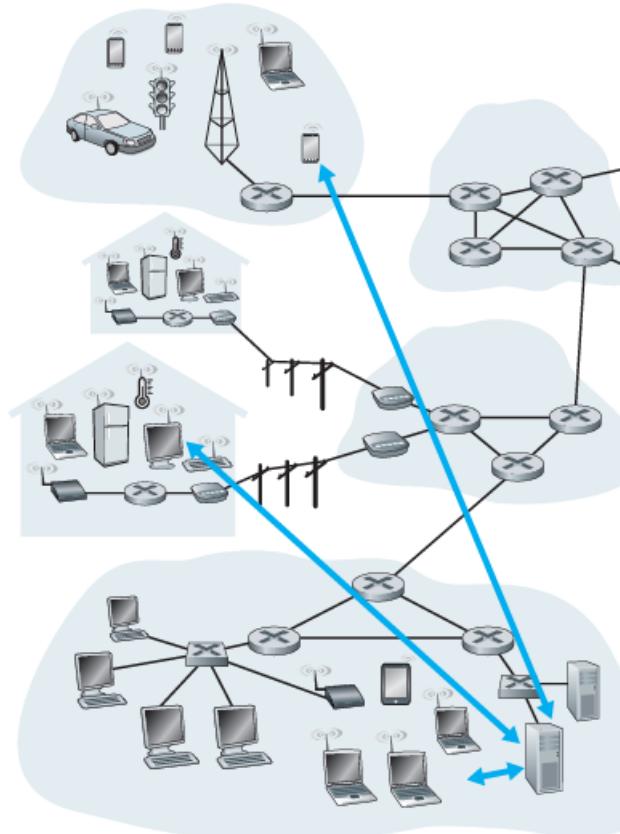


Hình 12: telnet

SSH (Secure Shell) là một giao thức có chức năng giống như telnet được phát triển vào năm 1995 có 2 phiên bản là version 1 và version 2, cung cấp tính năng bảo mật dữ liệu như là việc mã hóa dữ liệu và tính năng xác thực bằng cách chỉ có SSH server và SSH client mới có khả năng giải mã được những dữ liệu đã được gửi. SSH sử dụng giao thức TCP qua port 22

1.8 Client-server model

Là một mô hình trao đổi dữ liệu nơi mà server (luôn luôn được bật) sẽ nhận được những service request từ những client. Lấy một ví dụ cơ bản là một web application Web server sẽ luôn luôn được bật để nhận được request từ các client, nó phản hồi các request từ client bằng cách gửi những requested object đến client. Trong mô hình client server thì các clients không được tương tác trực tiếp với nhau. Thông thường ở các ứng dụng sử dụng mô hình client-server thì thường sử dụng nhiều hơn một server để có thể handle được tất cả các request đến từ các client nếu số lượng request lớn -> data center để có thể chứa được nhiều host và thường để tạo nhiều những server ảo qua công nghệ ảo hóa.



a. **Client-server architecture**

Figure 2.2 ◆ (a) Client-server architecture; (b)

Hình 13: Client-server

1.9 Connectionless và connection-oriented

Connection-oriented là một service giúp đảm bảo gói tin được chuyển đến đúng đích và cung cấp cả tính năng điều khiển luồng. Do đó trước khi truyền dữ liệu thì hai thiết bị gửi và nhận cần phải thiết lập một kết nối để đảm bảo được rằng dữ liệu sẽ được chuyển đến đúng nơi yêu cầu **Connectionless**: trái với connection-oriented thì connectionless không cần phải thiết lập kết nối trước khi truyền, mà truyền thẳng luôn mà không cần phải quan tâm rằng dữ liệu có đến đúng đích hay không. Việc này sẽ khiến cho việc truyền dữ liệu được nhanh hơn. Tuy nhiên không đảm bảo được tính tin cậy khi truyền

tin. Ngoài ra connectionless không có tính năng điều khiển luồng cũng như chống tắc nghẽn.

1.10 TCP và UDP

TCP (Transmission Control Protocol) là một connection-oriented protocol. Trước khi gửi dữ liệu đến đích, 2 bên gửi và nhận sẽ giao tiếp để thiết lập 1 kết nối. Khi kết nối này đã được thiết lập thì việc trao đổi dữ liệu sẽ diễn ra. TCP cung cấp kết nối tin cậy, destination host phải biết được ACK number mà nó được nhận của mỗi TCP segment, nếu mà segment đó không được acknowledged thì gói tin sẽ phải gửi lại. TCP thực hiện đánh số thứ tự các segment, sequence number trong TCP header giúp cho destination host ghép các segment vào đúng vị trí dù cho các segment đến destination host không đúng thứ tự hay không. TCP cung cấp tính năng điều khiển luồng, destination host có thể nói cho source host biết rằng nên tăng hay giảm lưu lượng dữ liệu được gửi.

| TCP segment header | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|-------|--|---|---|---|---|---|---|---|------------------------------------|-------------|---------|---------|-------|---------|-------|-------------|---|---|---|---|---|---|---|---|
| Offsets | Octet | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | 3 | | | |
| Octet | Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | Source port | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Sequence number | | | | | | | | Acknowledgment number (if ACK set) | | | | | | | | | | | | | | | |
| 8 | 64 | Data offset Reserved 0 0 0 | | | | | | | | N S | C W E R G E | U R C K | A C S H | P R T | R S Y N | F I N | Window Size | | | | | | | | |
| 12 | 96 | Checksum | | | | | | | | Urgent pointer (if URG set) | | | | | | | | | | | | | | | |
| 16 | 128 | Options (if data offset > 5. Padded at the end with "0" bytes if necessary.) | | | | | | | | ... | | | | | | | | | | | | | | | |
| 20 | 160 | ... | | | | | | | | ... | | | | | | | | | | | | | | | |

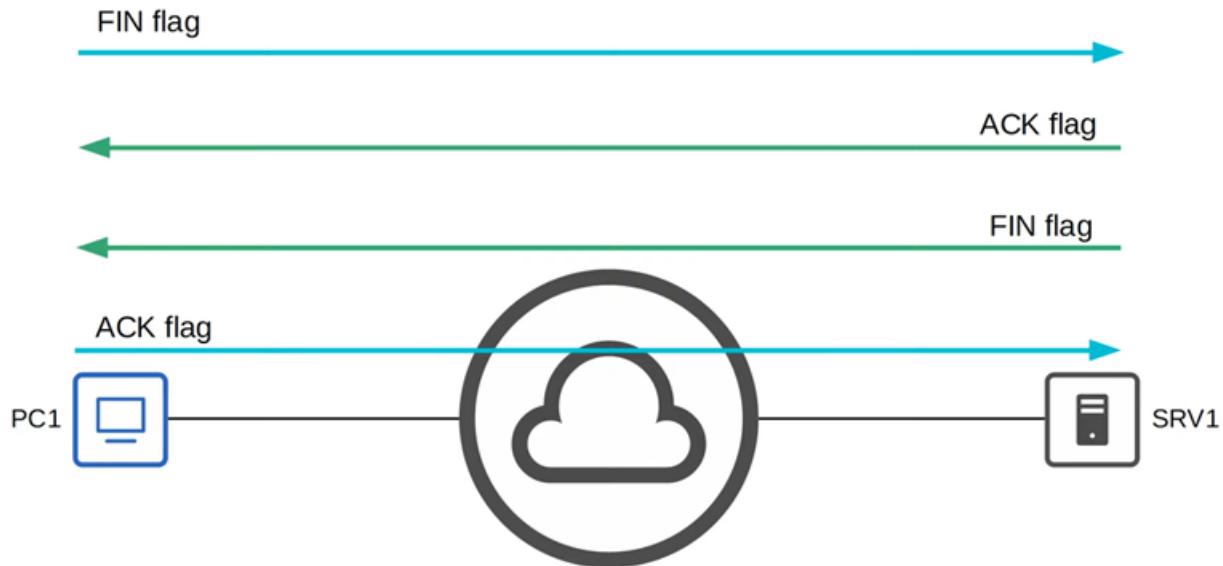
Hình 14: TCP-header

Hình trên là cấu trúc của một TCP header: bao gồm port nguồn, port đích mỗi port đều có độ dài là 16 bit độ dài -> có 2 mứ 16 số cổng trong TCP. Tiếp theo là sequence number và acknowledgment number. Tiếp theo là các bit cờ ở đây mình tập trung đến 3 bit cờ đó là ACK, SYN và FIN. Tiếp theo là window size trường này để hỗ trợ khả năng điều khiển luồng (flow control) cho TCP khi nó có thể điều chỉnh lưu lượng gửi.



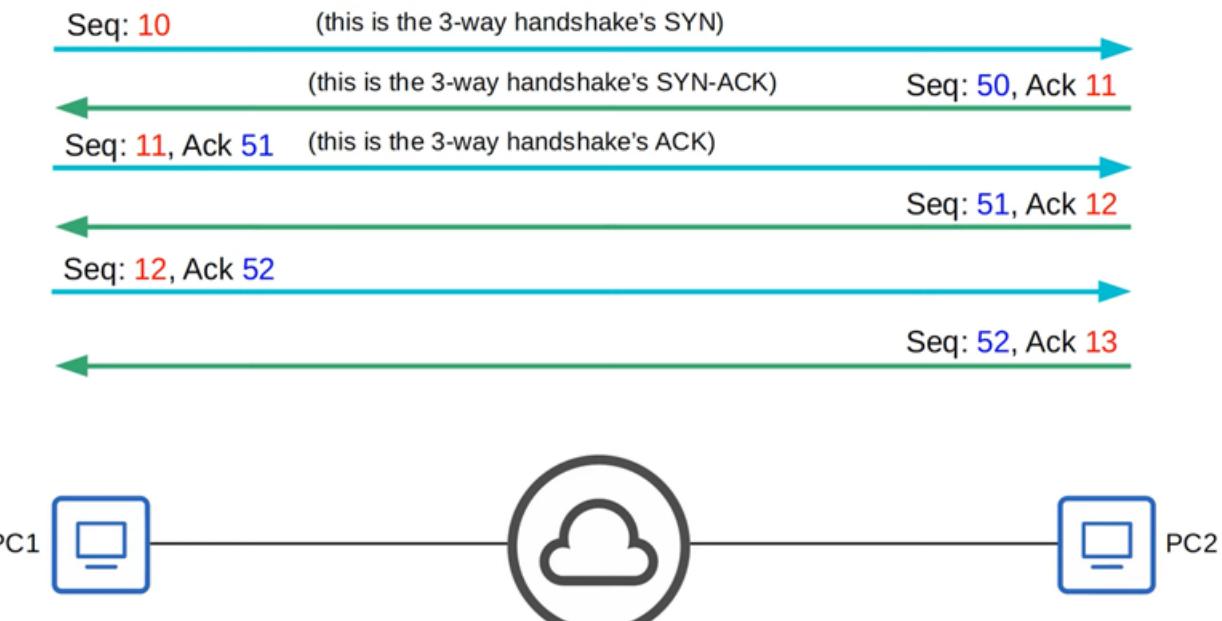
Hình 15: Threeways handshake

TCP three-way handshake quá trình này miêu tả 3 gói tin được gửi giữa các host. Đầu tiên để thiếp lập một phiên kết nối TCP, PC1 cần phải gửi một gói tin SYN (bit SYN trong TCP header được set bằng 1). Sau đó server sẽ reply lại bằng một gói tin SYN/ACK với bit SYN và bit ACK được set bằng 1. Khi nhận được gói tin này, PC1 sẽ reply lại bằng một gói tin ACK (bit ACK được set bằng 1). Khi đó quá trình three-way handshake được hoàn thành và phiên kết nối TCP được thiết lập và PC1 có thể bắt đầu gửi request đến cho server.



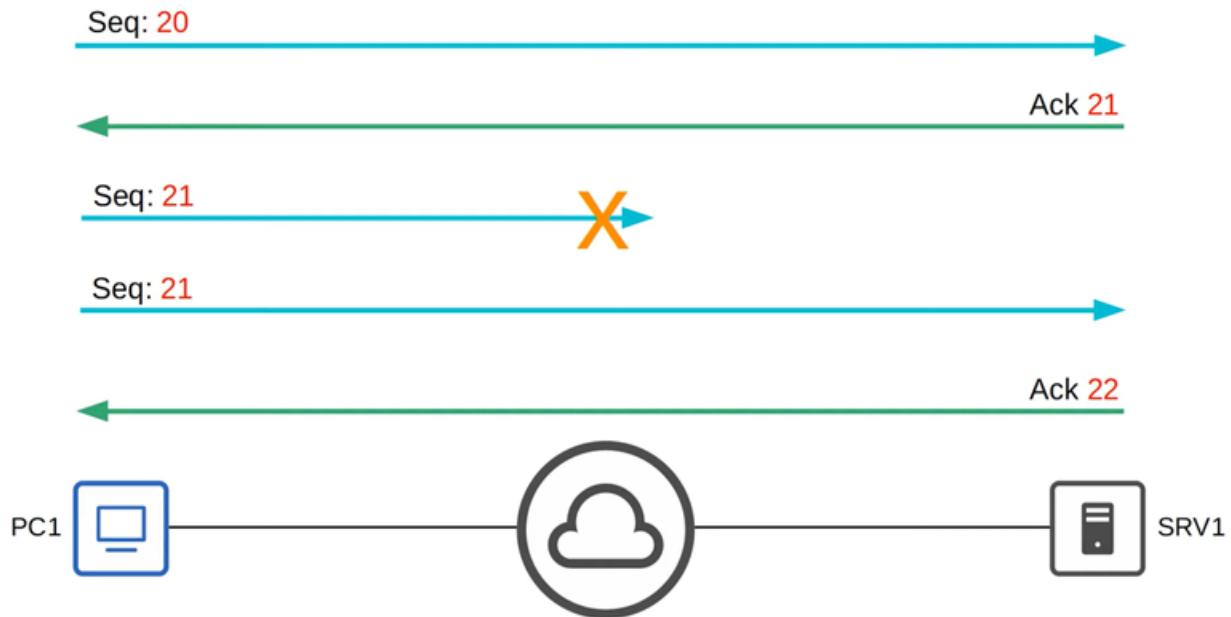
Hình 16: Fourway handshake

TCP four-way handshake (terminated connection): Đầu tiên để hủy kết nối thì PC1 sẽ gửi gói tin FIN (bit cờ FIN được set bằng 1) đến cho server. Sau khi nhận được gói tin này server reply lại bằng 2 gói tin đầu tiên server sẽ gửi một gói tin ACK và sau đó gửi luôn 1 gói tin FIN đến cho PC1. PC1 nhận được gói tin thì gửi lại gói tin ACK cho server và tiến hành đóng kết nối.



Hình 17: TCP-sequencing

Hình trên diễn tả cơ chế sequencing/acknowledgment của TCP, đầu tiên để bắt đầu phiên kết nối TCP như thông thường sẽ là quá trình TCP three-way handshake, PC1 sẽ chọn một số ngẫu nhiên để đặt cho trường sequence number (seq =10) trong TCP header và gửi cho server. Sau đó server cũng sẽ tự tạo cho mình một sequence number ngẫu nhiên (seq =50) và để chứng nó rằng nó đã được nhận bản tin ACK từ PC thì server sẽ thực hiện gửi ACK number bằng sequence number của PC cộng với 1, cũng như yêu cầu next segment cần phải nhận từ phía PC (ACK = 11) cho PC . Sau khi PC nhận được gói tin thì nó sẽ cập nhật sequence number của mình thành 11 và set ACK number bằng 51 để yêu cầu server gửi cho mình segment có sequence number bằng 51. Tương tự server sau khi nhận được gói tin sẽ gửi cho PC có sequence number bằng 51 và ACK = 12. Forward acknowledgement (ACK) thường được sử dụng để chỉ ra sequence number của segment tiếp theo mà host mong muốn được nhận. Tiếp theo là vấn đề retransmission khi mà gói tin có sequence number bằng 21 bị lost vì một lý do nào đó thì PC sẽ tiến hành gửi lại và TCP vẫn thực hiện cơ chế sequencing và acknowledgement như bình thường. Lưu ý rằng trong thực tế sequence number có thể được cộng nhiều hơn một phụ thuộc vào kích cỡ của segment được gửi.



Hình 18: TCP retransmission

Flow control: TCP điều khiển luồng qua cơ chế slicing window thông qua trường window size trong TCP header. Trường này được sử dụng bởi người nhận để chỉ cho người gửi biết lượng dữ liệu mà nó có thể chấp nhận được. Window size quy định số lượng byte dữ liệu mà người nhận có thể xử lý một cách hiệu quả. Khi dữ liệu đã truyền đạt đến giá trị cửa sổ, người gửi sẽ mong đợi nhận được giá trị cửa sổ mới từ người nhận cùng với xác nhận cho cửa sổ vừa nhận. Window size không phải một giá trị ngẫu nhiên nó được tính toán dựa trên thông số như băng thông và Round trip time,... giữa người nhận và người gửi.

UDP: User Datagram Protocol là một giao thức connection-oriented. Người gửi sẽ không cần phải thiết lập kết nối trước với người nhận trước khi gửi dữ liệu, data sẽ được truyền ngay lập tức luôn. UDP không cung cấp tính năng truyền tin một cách tin cậy. Khi UDP được sử dụng thì **acknowledgement** sẽ không được gửi cho những segment được gửi, nếu một segment bị mất thì UDP sẽ không có cơ chế gửi lại. UDP không cung cấp tính năng **sequencing**, đối với UDP header sẽ không có trường sequence number, nếu segment được gửi và đến đích không đúng thứ tự thì UDP cũng không có cơ chế khiến các segment được gửi đúng theo thứ tự. UDP cũng không có tính năng flow control như TCP (nhìn UDP header không có trường window size).

| UDP datagram header | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---------------------|-------|-------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Offsets | Octet | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | 0 | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Length | | | | | | | | | | | | | | | | Checksum | | | | | | | | | | | | | | | |

Hình 19: UDP-header

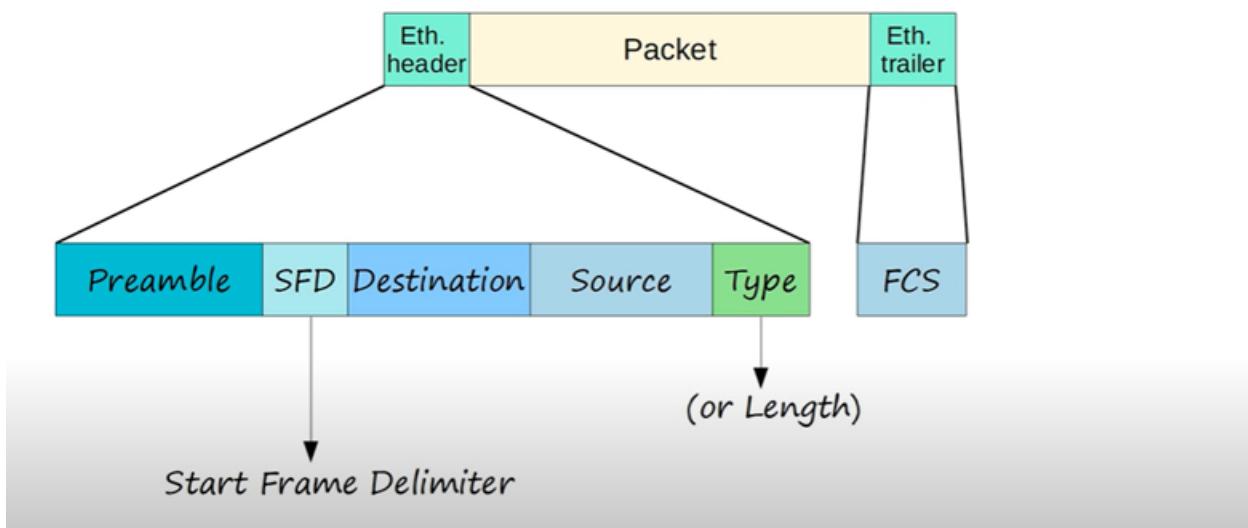
So sánh giữa TCP và UDP

| TCP | UDP |
|--------------------------------------|--------------------------------|
| Connection-oriented | Connectionless |
| Reliable | Unreliable |
| Sequencing | No sequencing |
| Flow control | No flow control |
| Use for downloads, file sharing, etc | Used for VoIP, live video, etc |

Hình 20: TCP và UDP

2 Basic Switching

2.1 Ethernet frame

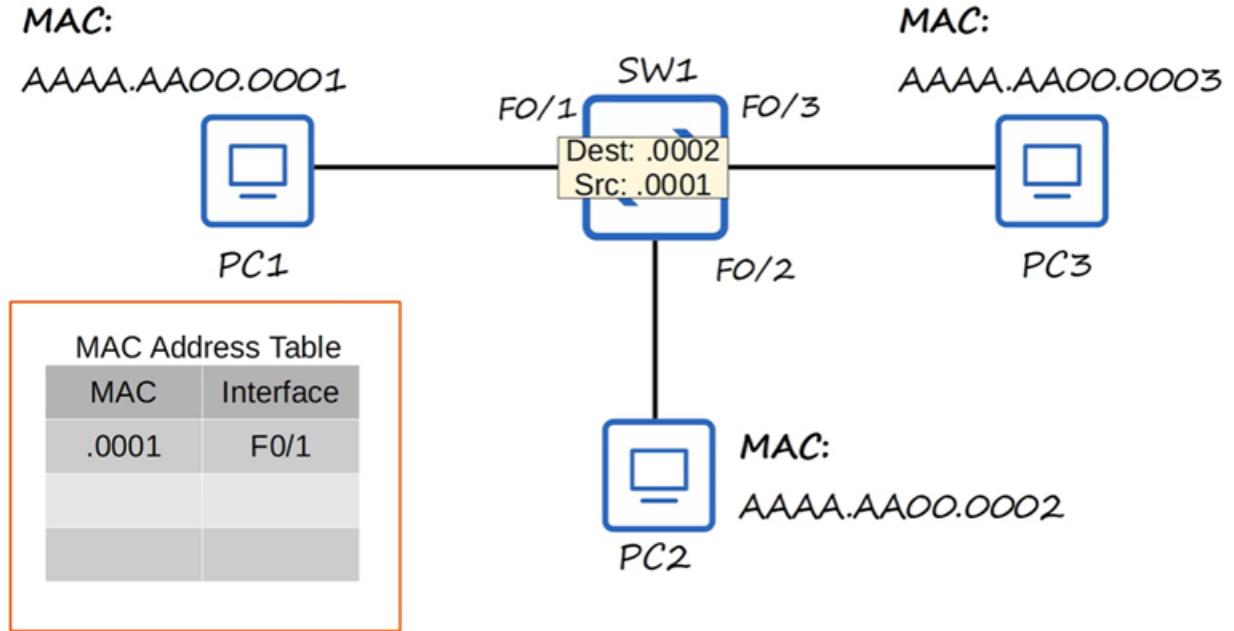


Hình 21: Ethernet-frame

Dầu tiên là trường Preamble có độ dài là 7 bytes là một chuỗi nhị phân 0, 1 xen kẽ nhau với mục đích là để đồng bộ xung clock với người nhận đảm bảo rằng người nhận sẵn nhận frame và dữ liệu bên trong. Trường tiếp theo là SFD (Start Frame Delimiter) có độ dài là 1 byte có cấu trúc là 10101011 dùng để đánh dấu giữa kết thúc của trường preamble và phần bắt đầu của một frame.

Trường tiếp theo là destination và source field chứa các địa chỉ MAC (48bit) có tác dụng để chỉ ra người gửi và người nhận frame.

Trường tiếp theo là type (length) có độ dài 16 bit được sử dụng để chỉ ra loại gói tin được đóng gói hoặc chiều dài của gói tin được đóng gói. Nếu mà giá trị của trường này bằng 1500 hoặc ít hơn thì nó



Hình 22: Switching

sẽ chỉ ra chiều dài của gói tin được đóng gói theo bytes. Nếu mà giá trị là 1536 hoặc lớn hơn thì nó sẽ chỉ ra loại gói tin được đóng gói (thường là Ipv4 hoặc Ipv6) và chiều dài của gói tin sẽ được xác định ở một chỗ khác (Ipv4 = 0x800, Ipv6 = 0x86DD)

Trường cuối cùng là FCS (Frame Check sequence) có độ dài là 4 bytes được sử dụng để kiểm tra lỗi dữ liệu bằng cách chạy thuật toán CRC (Cyclic Redundancy Check) qua dữ liệu được nhận.

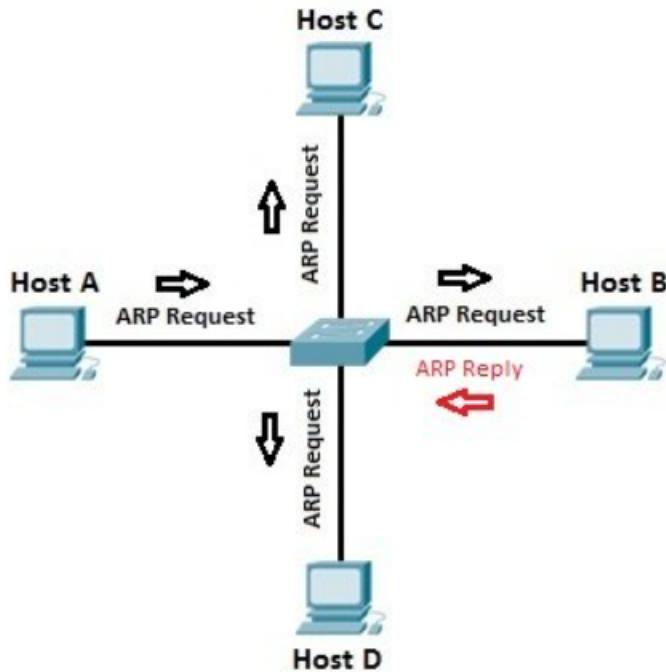
2.2 Hoạt động của Switch trong mạng

Hình trên diễn tả PC1 muốn gửi một gói tin cho PC2 với địa chỉ MAC nguồn là của PC1 địa chỉ MAC đích là của PC2. Khi Switch nhận được frame từ PC1 thì ngay lập tức switch sẽ tự học được địa chỉ MAC của PC1 và thêm vào MAC address table của nó. Sau đó vì không biết địa chỉ MAC của PC2 (Unknown Unicast Frame), Sw1 sẽ thực hiện flood frame ra tất cả các interface của nó trừ interface nối với PC1. Lúc này PC3 và PC2 nhận được frame và so sánh địa chỉ MAC của mình với địa chỉ MAC đích thì PC2 nhận thấy trùng và gửi lại bản tin reply với địa chỉ nguồn là địa chỉ MAC của PC2 và địa chỉ đích là địa chỉ MAC của PC1, PC3 thấy không khớp và drop frame. Sau khi nhận được frame gửi từ PC2 thì Switch sẽ học luôn địa chỉ MAC của PC2 và tiến hành gửi frame ra F0/1 cho PC1

Giao thức ARP:

- Giao thức ARP (Address resolution protocol) là giao thức truyền thông được sử dụng để chuyển địa chỉ từ lớp mạng L3 sang lớp mạng L2. ARP được sử dụng để từ một địa chỉ IP tìm ra một địa chỉ MAC
- Trong mạng Ethernet và WLAN các gói IP không được gửi trực tiếp mà phải bỏ vào một khung(frame) Ethernet rồi mới được gửi đi. Các địa chỉ này là địa chỉ MAC của card mạng, một card mạng sẽ nhận được các frame có địa chỉ đích là địa chỉ MAC của card mạng đó.

[H]



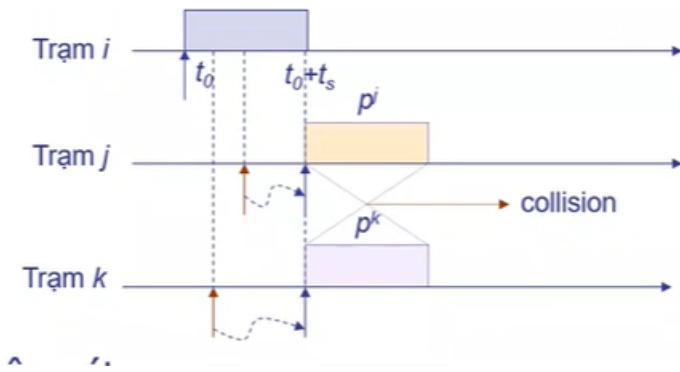
Hình 23: ARP

Lấy ví dụ ở hình bên trên, có 4 host được kết nối với nhau trong một mạng LAN qua một switch. Host A muốn gửi một bản tin tới Host B, giả sử Host A có địa chỉ IP là **192.168.1.1** còn Host B có địa chỉ IP là **192.168.1.2**. Để gửi bản tin đến Host B thì A cần phải biết được địa chỉ MAC của B. Đầu tiên A sẽ tìm trong bảng ARP lưu trữ để tìm kiếm địa chỉ IP của B cho bất kỳ ghi nhận nào hiện có của địa chỉ MAC của B. Nếu tìm thấy địa chỉ IP của B thì A sẽ gửi một frame đến B với địa chỉ đích là địa chỉ MAC của B chứa gói tin IP. Nếu không tìm thấy, thì A sẽ phải gửi một ARP request với địa chỉ MAC đích là **FFFF.FFFF.FFFF** tới tất cả các host có trong mạng LAN (broadcast).Những host nào có địa chỉ IP là 192.168.1.2 sẽ trả lời lại A bằng một bản tin ARP reply với địa chỉ IP và địa chỉ MAC của nó(ở đây là Host B như ta đã đề cập ở trên). Sau đó cả B và A sẽ ghi lại địa chỉ MAC gắn với IP của nhau để có thể sử dụng lại trong tương lai mà không cần broadcast nữa.

2.3 CSMA

CSMA (Carrier Sense Multiple Access) là một cơ chế để giảm xác suất va chạm trên mạng LAN thông qua cách kiểm tra trạng thái kênh truyền. Trước khi truy nhập kênh, mỗi trạm gửi sẽ có cơ chế kiểm tra trạng thái kênh truyền, nếu phát hiện có sóng mang tức kênh truyền đang bận thì phải đợi đến khi không có sóng mang tức là kênh truyền rỗng. Có 3 loại CSMA cơ bản là: 1-persistent CSMA, p-persistent CSMA, None-persistent CSMA.

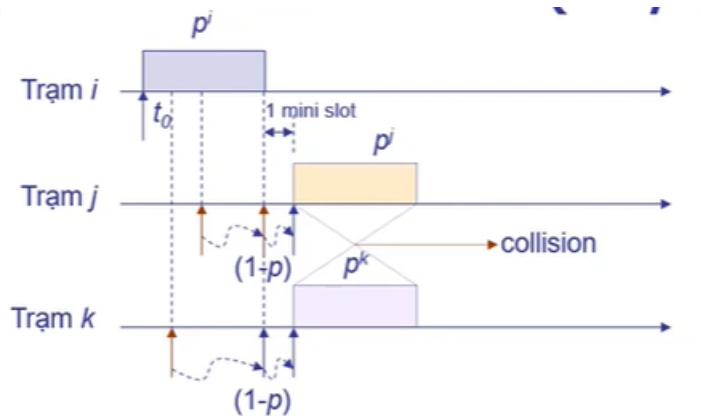
1-persistent CSMA



Hình 24: 1-persistent CSMA

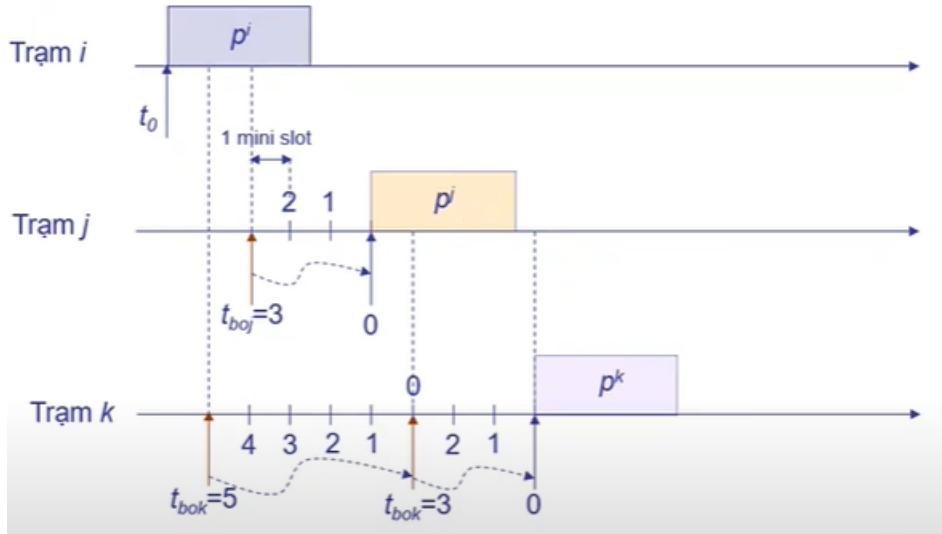
Tại thời điểm t_0 , trạm i kiểm tra trạng thái kênh truyền thì thấy kênh dỗi và tiến hành truyền dữ liệu, trong quá trình trạm i truyền dữ liệu thì trạm j và trạm k tiến hành kiểm tra kênh và thấy rằng kênh đang bận và chuyển sang trạng thái đợi. Đến thời gian t_0+t_s sau khi trạm i đã truyền xong thì trạm j và trạm k đều cùng thực hiện truy nhập kênh cùng 1 lúc và lúc này xảy ra va chạm.

p-persistent CSMA: đưa ra giá trị mini slot với $T_{ms} < T_s$ (thời gian lan truyền tối đa của tín hiệu trên kênh). Khi kênh truyền dỗi, trạm truy nhập kênh với xác suất p . Nếu kênh truyền bận thì trạm sẽ đợi một mini slot với xác suất $1-p$ sau đó kiểm tra lại trạng thái kênh. Ở hình thức này nếu p càng nhỏ thì xác suất va đập càng nhỏ tuy nhiên hiệu suất kênh sẽ bị giảm.



Hình 25: p-persistent CSMA

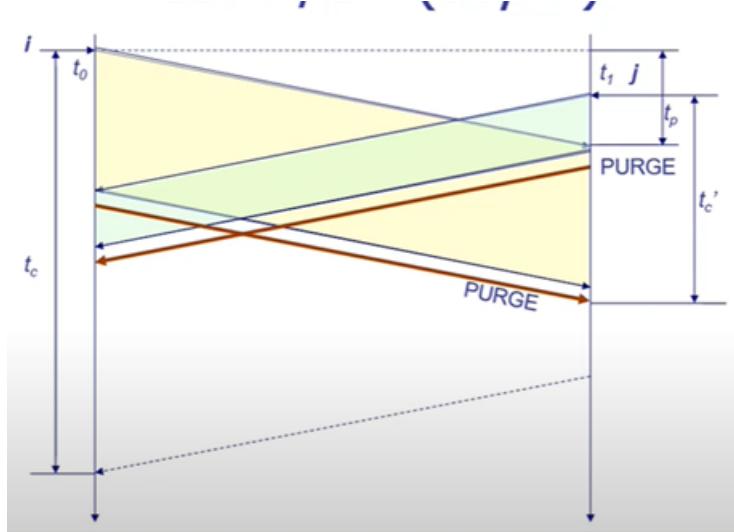
Non-persistent CSMA: Khi kênh truyền bận thì một trạm sõ trì hoãn thời gian truy nhập kênh một khoảng thời gian ngẫu nhiên bằng số nguyên lần minislot (back-off)



Hình 26: None-persistent CSMA

CSMA/CD: Giống như CSMA nhưng đưa thêm cơ chế phát hiện va đập (collision detection) bằng cách giám sát ở card mạng khi có dòng lớn hơn 24 mA hay không . Khi va đập xảy ra thì các trạm thực hiện các bước:

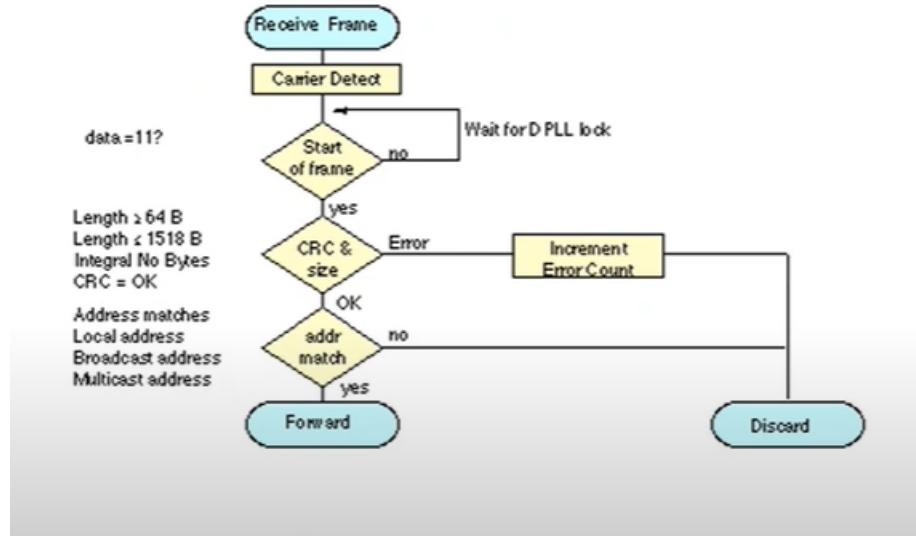
1. Trạm dừng truyền gói
2. Gửi bản tin PURGE để báo hiệu cho các trạm khác
3. Thực hiện thuật toán back-off để tránh va chạm ở lần phát lại



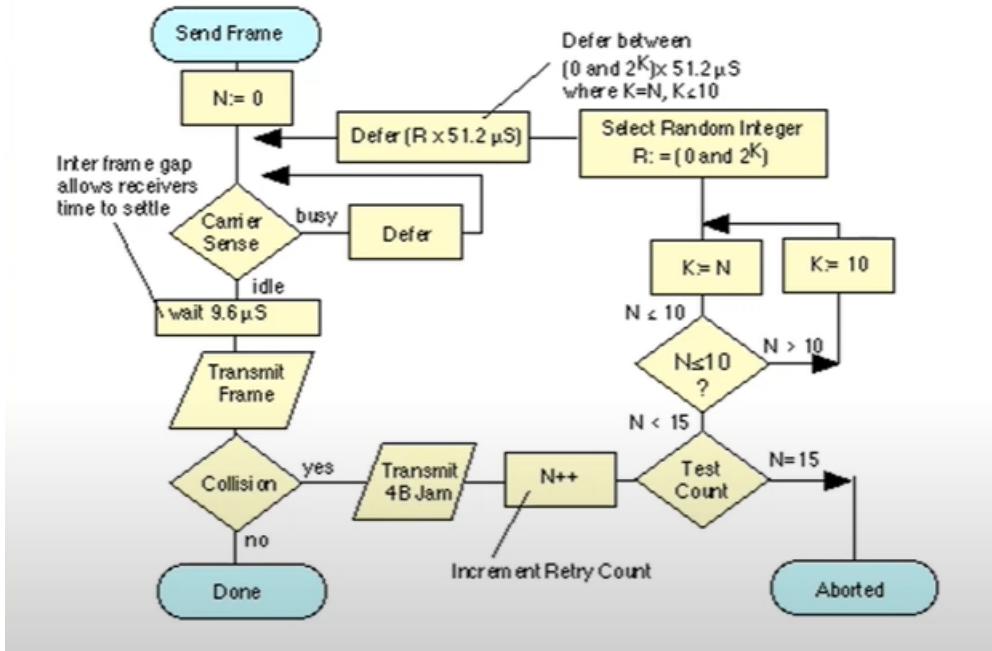
Hình 27: CSMA/CD

Thời gian back-off sẽ khác với Non-persistent CSMA ($0 < t < 2 m\mu k$) với k là số lần truy nhập không thành công. CSMA/CD sẽ xảy ra va chạm nếu như mà độ dài kênh truyền lớn. Trong CSMA/CD, quy định các frame phải được phát cách nhau ít nhất 9,6 micro giây.

CSMA/CD Transmitter: 51,2 micro giây là thời gian để phát một khung có kích thước bé nhất



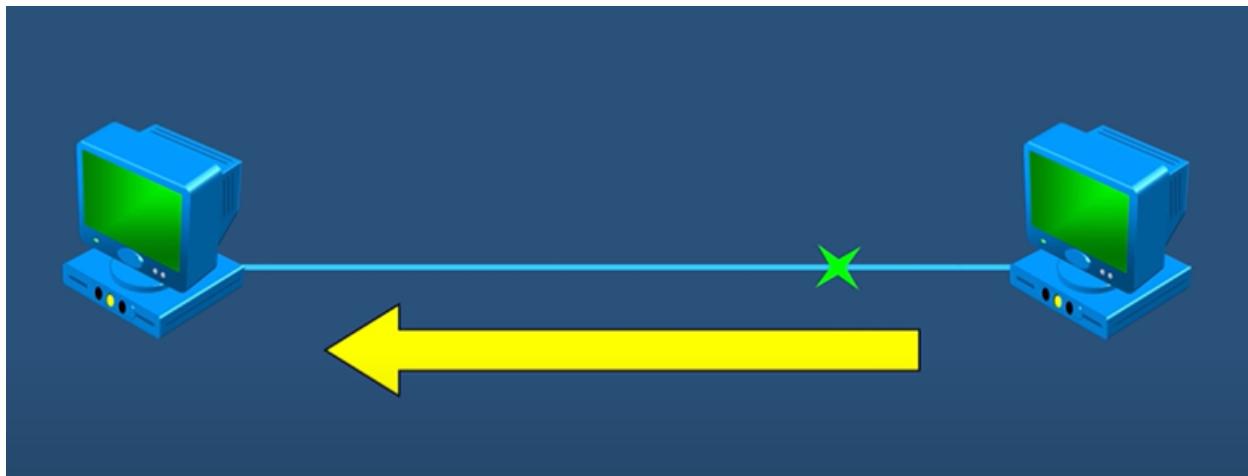
Hình 29: CSMA/CD receiver



Hình 28: CSMA/CD transmission

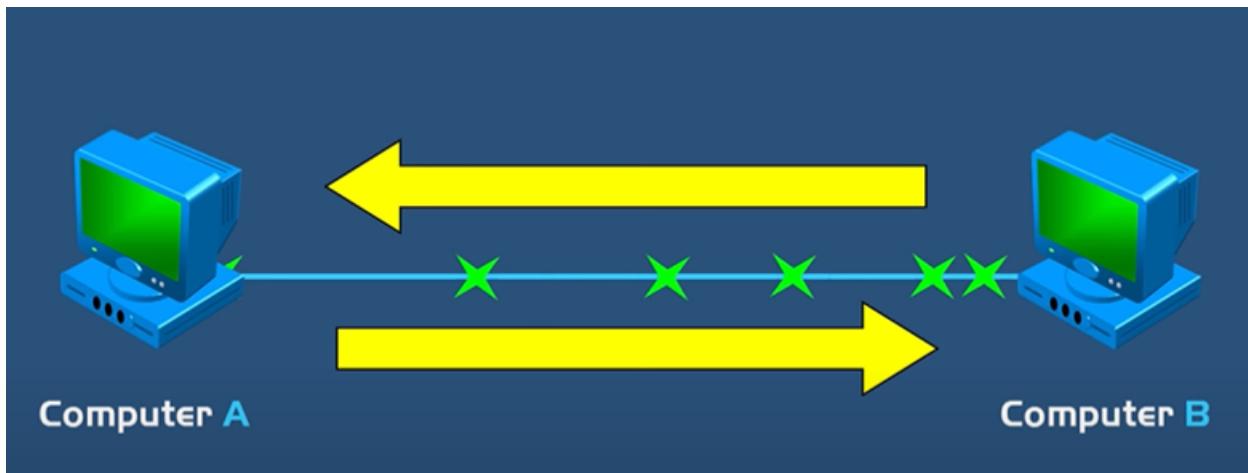
2.4 Half-duplex và Full-duplex

Half-duplex là một kiểu trao đổi dữ liệu trong mạng, khi mà một host chỉ được trao đổi dữ liệu theo một chiều cùng lúc mà không thể trao đổi dữ liệu theo 2 chiều cùng lúc. Ví dụ khi 2 máy muốn trao đổi dữ liệu cho nhau thì máy 2 phải đợi máy 1 gửi xong dữ liệu thì mới được gửi dữ liệu cho máy 1. Luồng dữ liệu sẽ chỉ đi theo một hướng mà thôi



Hình 30: Half-duplex

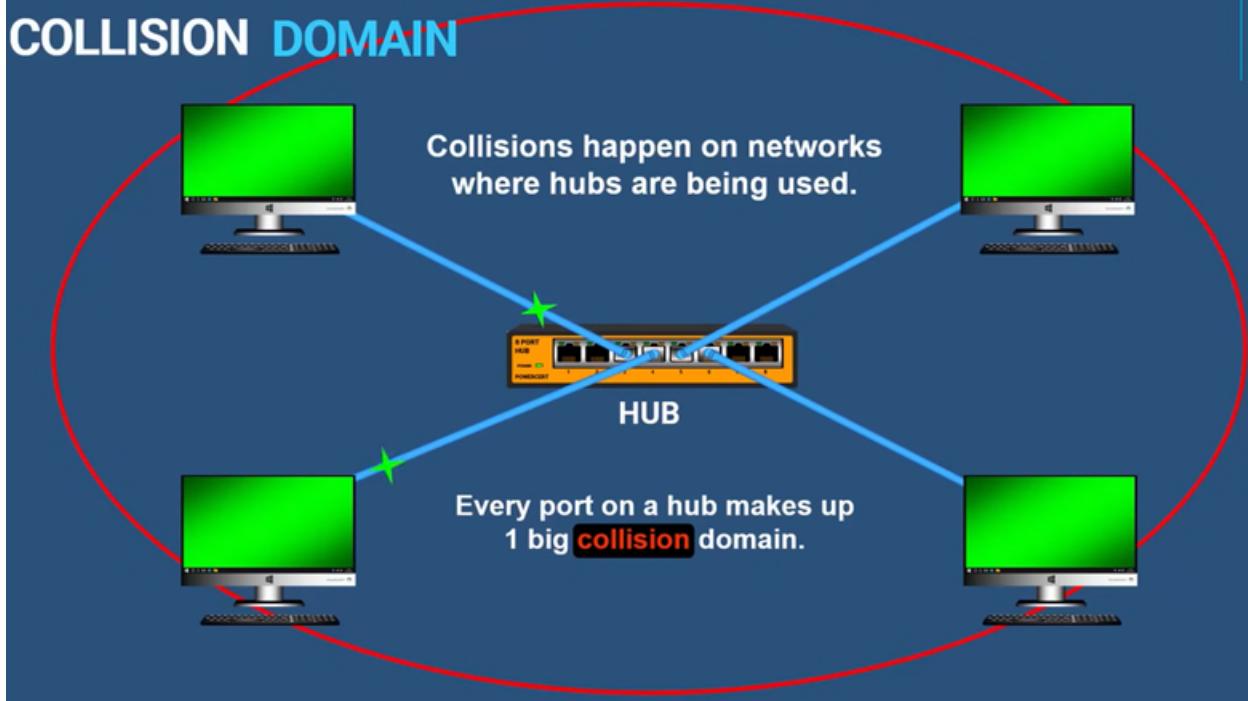
Full-duplex: cũng là một kiểu trao đổi dữ liệu trong mạng máy tính nhưng khác với half-duplex, full-duplex cho phép trao đổi dữ liệu theo 2 chiều cùng một lúc. Thiết bị có thể gửi và nhận dữ liệu cùng một lúc



Hình 31: Full-duplex

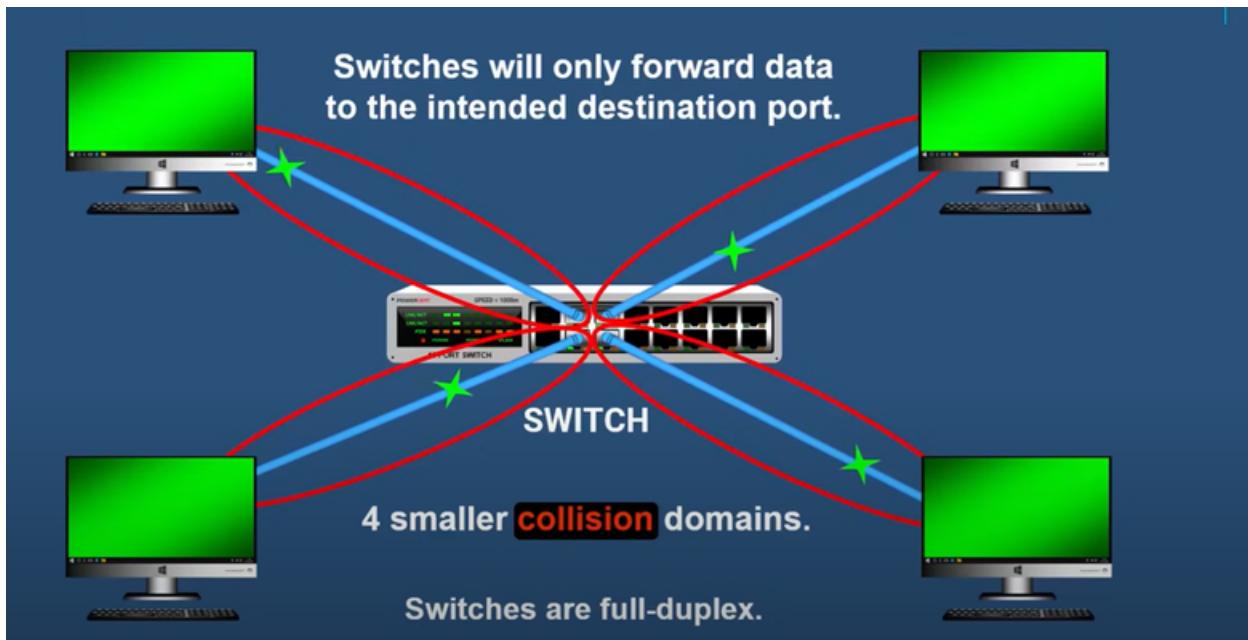
2.5 Broadcast domain và collision domain

Collision domain hay còn gọi là miền xung đột là một phần của mạng nơi mà dữ liệu được gửi từ các trạm có thể bị xung đột với nhau xảy ra khi các trạm trong mạng gửi dữ liệu vào trong cùng một khoảng thời gian làm xảy ra va chạm. Collision xảy ra trong mạng khi hubs được sử dụng. Tất cả các port được sử dụng trong hub đều tạo ra một collision domain và hub hoạt động theo cơ chế half-duplex (chỉ truyền được một hướng trong một thời gian) cũng như forward gói tin ra tất cả các cổng của nó



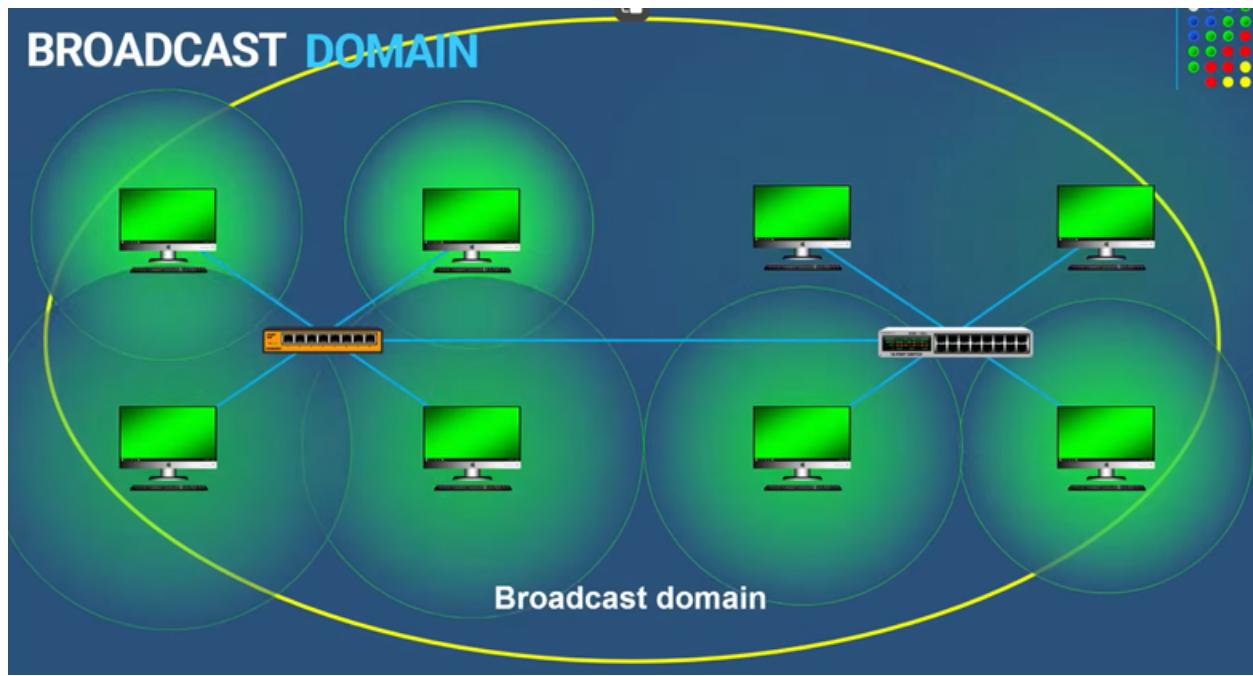
Hình 32: Collision domain

Vì thế mà switch được tạo ra để chia nhỏ collision domain do switch có khả năng gửi đến đúng đích mà không cần phải đẩy gói tin ra tất cả các port và switch hoạt động theo cơ chế full-duplex nên nó có thể giao tiếp theo 2 chiều cùng một lúc



Hình 33: Collision domain 2

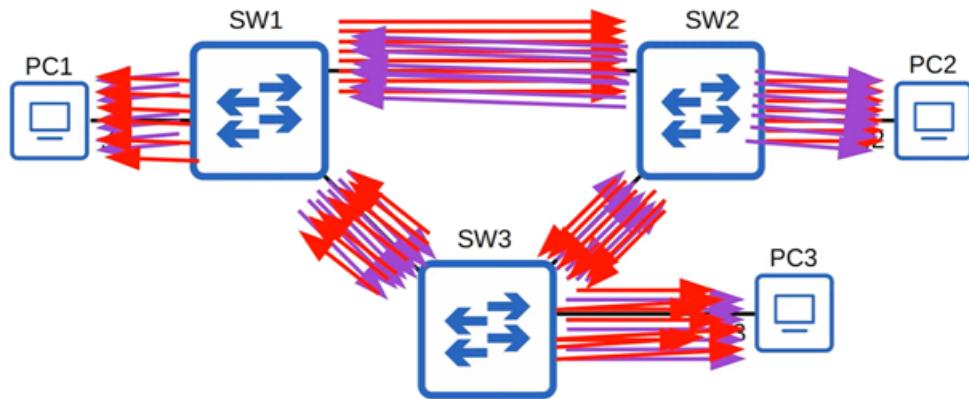
Boardcast domain là một phần của mạng nơi mà thiết bị mạng có thể nhận được bản. Và để chia nhỏ một boardcast domain như trên ta chỉ việc thêm router ở giữa hub và switch để tạo ra 2 boardcast



Hình 34: Broadcast domain

2.6 Spanning tree protocol

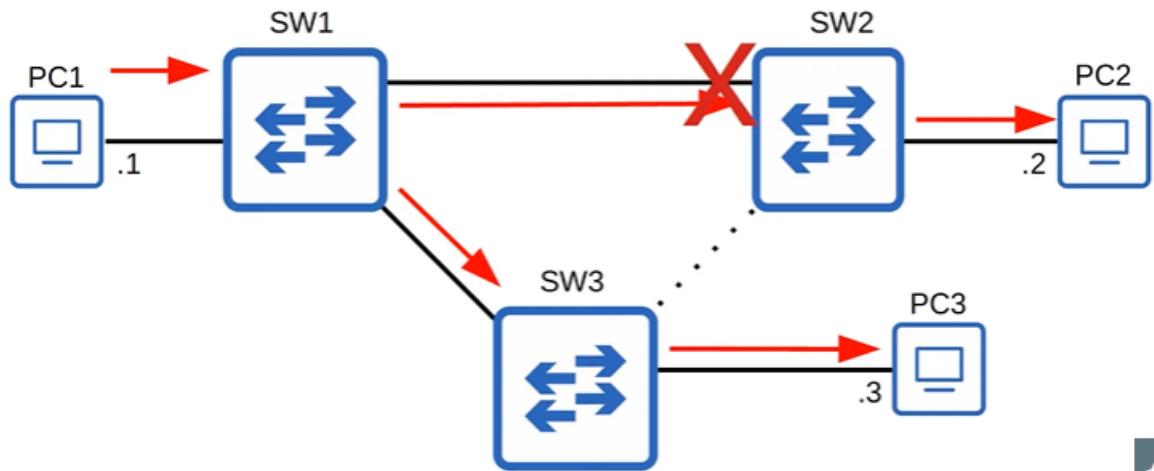
Broadcast storms: bởi vì ethernet frame không có trường TTL. Những broadcast frame này sẽ lặp lại khắp trong mạng. Nếu lượng broadcast frame này đủ lớn nó sẽ khiến cho mạng bị tắt nghẽn và gọi là broadcast storm. Ngoài ra việc khi mà một frame đến một port trên switch thì switch sẽ sử dụng source MAC address để học và thêm vào bảng MAC address. Khi mà frame với cùng một source MAC address xuất hiện lặp lại ở các port khác nhau của switch khiến cho switch sẽ phải liên tục update lại interface trong bảng MAC address (MAC address Flapping)



Hình 35: Broadcast-storm

STP ngăn chặn layer 2 loop bằng cách đặt một port ở trong trạng thái block hay là disable interface đó đi. Những interface này hoạt động như là một backups có thể chuyển sang trạng thái forwarding

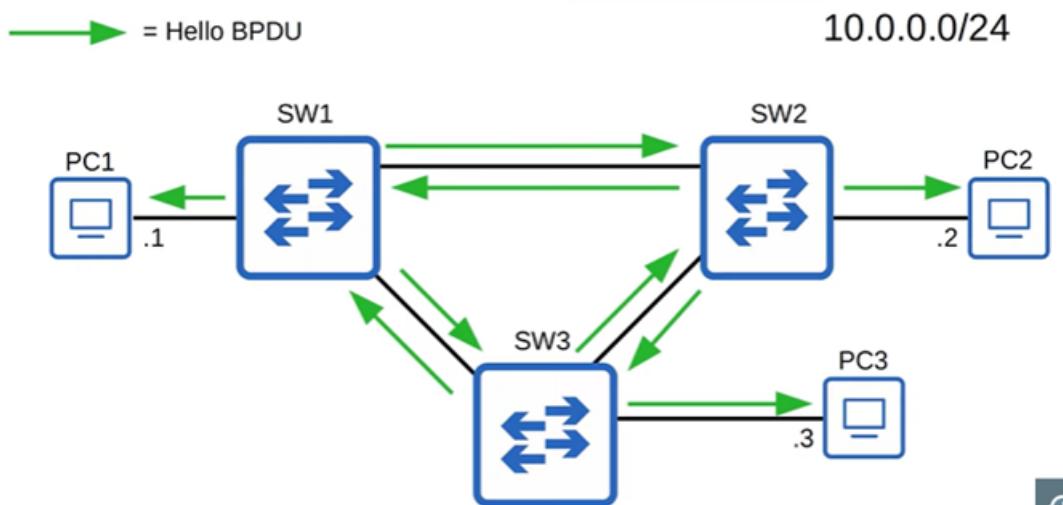
nếu một interface đang được đặt trong trạng thái forwarding bị lỗi. Interface trong trạng thái block chỉ nhận hoặc gửi những STP messages (gọi là BPDU = Bridge Protocol Data Unit)



Hình 36: STP work

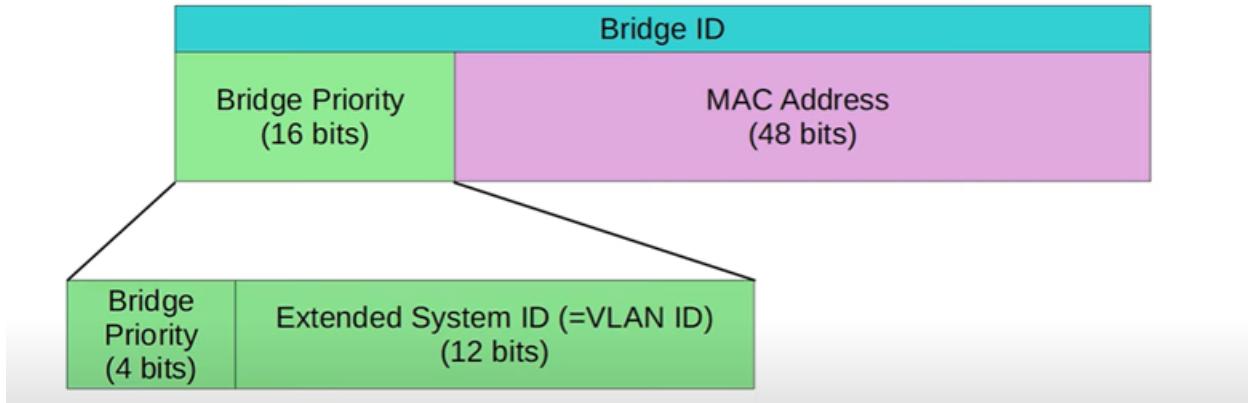
Giả sử trong trường hợp này thì interface nối sw3 với sw2 sẽ ở trong trạng thái block và sẽ ngăn chặn được loop khi phát bản tin broadcast, còn nếu trong trường hợp đường link nối sw1 và sw2 bị lỗi thì interface nối sw3 với sw2 sẽ lại được bật lên để có thể forward bản tin từ PC1 sang PC2. Bằng cách lựa chọn port nào đang forwarding port nào đang blocking thì STP sẽ tạo ra duy nhất một đường đến hoặc đi đến mỗi điểm trong mạng để ngăn chặn Layer 2 loops. Quá trình để STP xác định được port nào được forwarding, port nào được blocking sẽ được diễn tả như sau:

- Switch sẽ gửi hoặc nhận những bản tin Hello BPDU ở tất cả các interface của nó, với khoảng thời gian mặc định là 2s (switch sẽ gửi một Hello BPDU ra khỏi mỗi interface mỗi 2s).
- Nếu một switch được nhận một Hello BPDU ở interface của nó thì nó sẽ biết được interface này được nối với switch khác bởi vì Router hay PC không sử dụng STP nên là không thể gửi được Hello BPDU



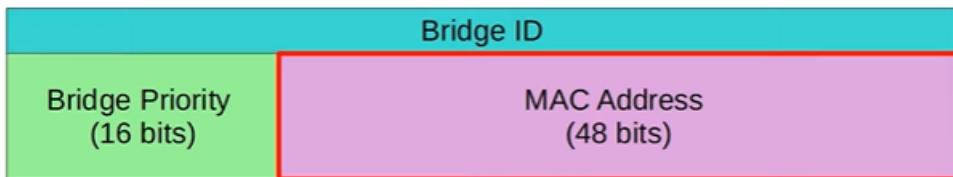
Hình 37: BPDU

Switch sẽ dùng một trường trong STP BPDU là trường Bridge ID để chọn một root bridge cho



Hình 39: Bridge ID 2

mạng. Switch có Bridge ID thấp nhất sẽ được chọn làm root bridge. Tất cả các port trong root bridge sẽ được đặt trong trạng thái forwarding và các switches khác trong topo sẽ phải có đường đi đến root bridge



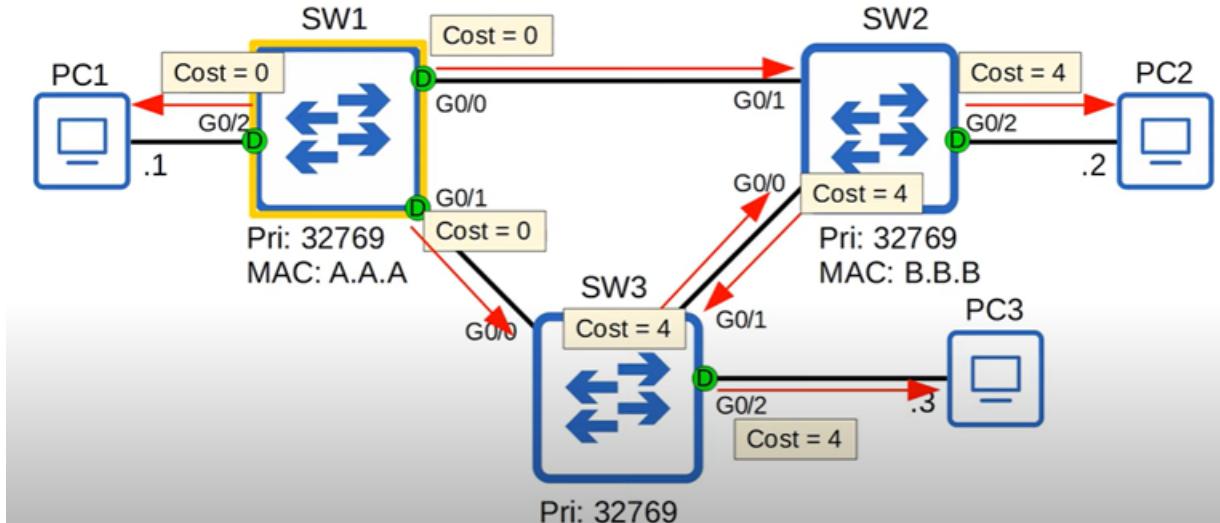
Hình 38: Bridge ID

Giá trị bridge priority mặc định của mỗi switch sẽ là 32768 nên do đó nhờ vào địa chỉ MAC của mỗi switch sẽ xác định được root bridge (địa chỉ MAC thấp nhất sẽ trở thành root bridge). Nhưng trong thực tế thì bridge priority đã được cập nhật thành 2 phần đó là: bridge priority và VLAN ID. Các loại switch của cisco thì được sử dụng phiên bản STP có tên là PVST (Per-VLAN Spanning Tree). PVST chạy STP trên mỗi VLAN do đó trên mỗi VLAN interface có thể ở trong trạng thái block hoặc forwarding (VD: một interface có thể được block ở VLAN1 nhưng lại được forwarding ở VLAN2). Bằng cách thêm trường VLAN ID vào Bridge Priority thì switch có thể có bridge id khác nhau trên mỗi VLAN. Để thay đổi Bridge ID thì ta chỉ có thể thay đổi được Bridge Priority mà thôi vì VLAN ID đã được set thì không thể thay đổi được.

Khi một switch được bật thì nó coi như nó là root bridge, nó sẽ gửi bản tin BPDU (tất cả switch ban đầu đều có thể gửi được bản tin BPDU). Nó sẽ chỉ từ bỏ vị trí này chỉ khi nhận được bản tin BPDU có bridge ID bé hơn. Khi mà topo mạng đã được hoàn chỉnh thì các switch sẽ đồng ý rằng chỉ có root bridge mới được gửi BPDU và khi các switch khác sẽ chỉ forward BPDU đi thôi mà không tạo ra BPDU của chính nó.

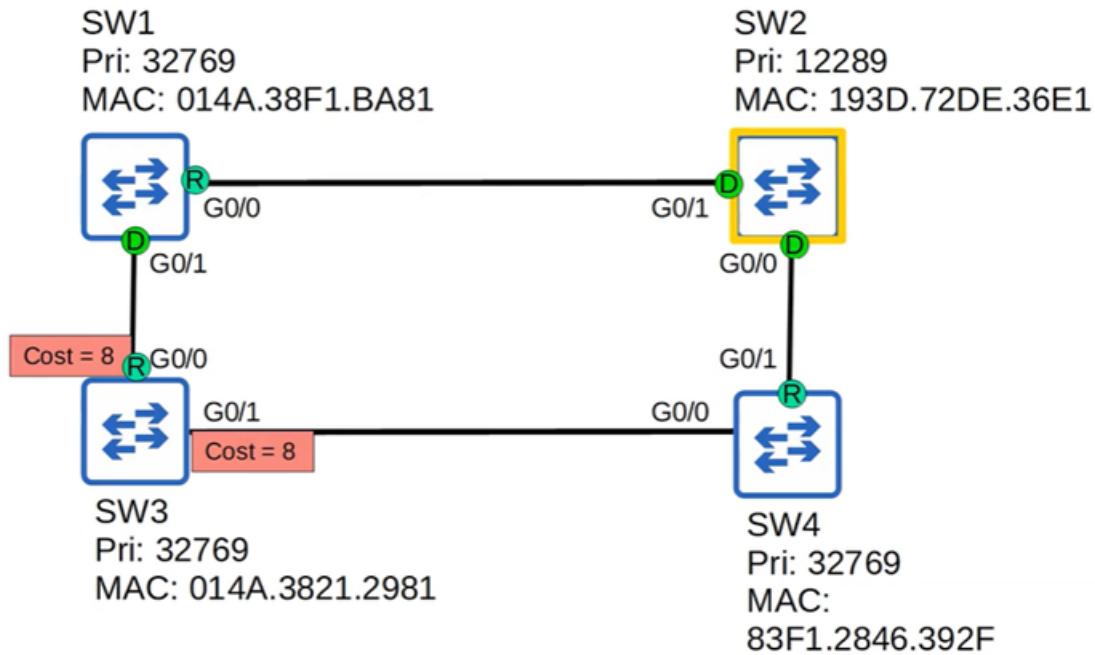
- Switch giá trị bridge ID thấp nhất sẽ được chọn làm root bridge. Tất cả port trên root bridge sẽ được ở trạng thái forwarding.
- Switch khác sẽ chọn một trong những interface của nó làm **root port**. Interface với giá trị root cost bé nhất sẽ được coi là root port. Root port cũng sẽ luôn luôn được đặt ở trạng thái forwarding. Mỗi

interface có đều có một STP cost và root cost sẽ được tính theo đường đi của bản tin BPDU. Ví dụ là 10Mbps:100, 100Mbps:19, 1Gbps:4, 10Gbps:2.



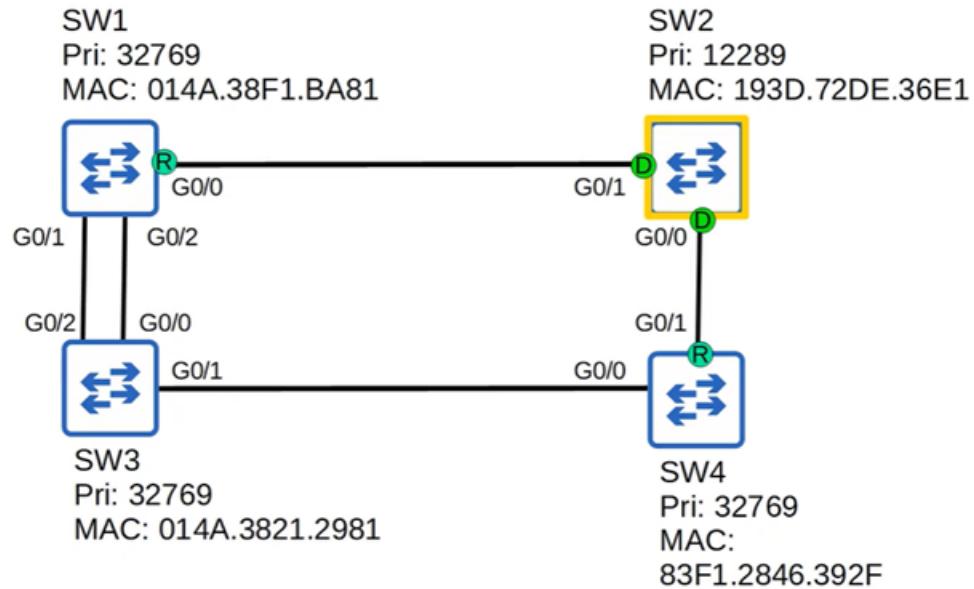
Hình 40: STP work 2

Ở trong trường hợp trên tại sw2 khi mà nhận được bản tin cost =0 tại G0/1 thì cost tại G0/1 sẽ bằng 4 (1Gbps=4), còn đối với G0/0 nhận được bản tin quảng bá từ sw3 có cost là 4 nên tổng cost tại G0/0 là 8. Do đó mà sw2 sẽ chọn G0/1 làm root port. Quá trình chọn root port của sw3 tương tự như sw2 khi G0/0 của sw3 được chọn làm root port. Lưu ý rằng port mà được kết nối tới root port của một switch khác luôn phải ở trạng thái forwarding (designated port) bởi vì root port là đường đi đến root bridge nên các switch không được block nó. Với trường hợp mà root cost giống nhau thì switch chọn root port dựa trên **lowest neighbor bridge ID**.



Hình 41: STP work 3

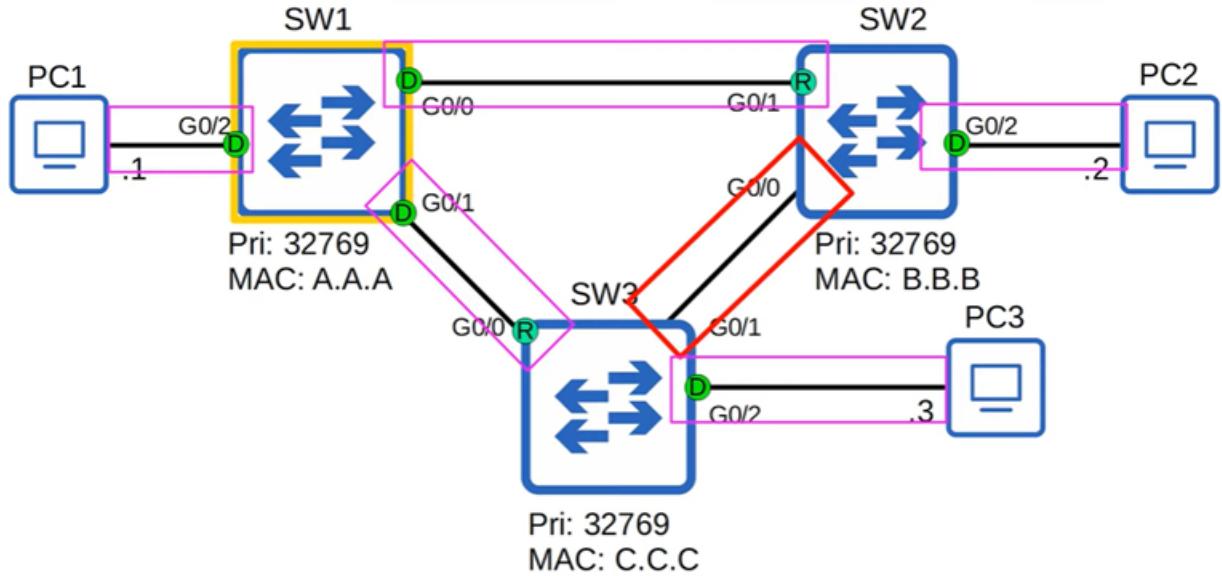
Ví dụ dưới đây vì G0/0 được kết nối với switch có địa chỉ MAC address thấp hơn nên G0/0 trên SW3 sẽ được chọn làm root port, và G0/1 của SW1 sẽ là designated port. Trong trường hợp mà neighbor bridge ID giống nhau thì sẽ chọn dựa trên lowest neighbor port ID. Khi đó STP port ID = port priority (default 128)+ port number, còn port number được xem bởi câu lệnh show spanning-tree trên switch.



Hình 42: STP work 4

Ví dụ ở trên thì G0/2 của SW3 sẽ được chọn là root port và G0/1 sẽ phải là designated port.

- Khi đã chọn được xong root port ở trên các switch, STP sẽ tiến hành chọn designated port trên mỗi collision domain. Switch với lowest root cost sẽ làm cho port của nó là designated port, nếu root cost bằng nhau thì sẽ xét switch nào có lowest bridge ID sẽ được cho port của nó làm designated port. Còn switch còn lại sẽ làm cho port còn lại blocking.



Hình 43: STP work 5

Ở trên hình trên thì mỗi collision domain đều có một designated port duy nhất chỉ có đường link giữa SW2 và SW3 chưa có. Vì vậy ở đây STP sẽ tiến hành chọn designated port trên collision domain này. Vì root cost giống nhau nên sẽ xét bridge ID, bridge ID của SW2 bé hơn nên G0/0 của SW2 sẽ được chọn làm designated port còn G0/1 của SW3 sẽ bị block

Spanning tree port state:

| | |
|-------------------|---------------------|
| STP Port State | Stable/Transitional |
| Blocking | Stable |
| Listening | Transitional |
| Learning | Transitional |
| Forwarding | Stable |

Hình 44: Port State

Dầu tiên là blocking state : non-designated port sẽ được coi là blocking state, interface ở blocking state bị disable để ngăn chặn vòng lặp, không được gửi hoặc nhận những traffic mạng thông thường, chỉ nhận được STP BPDU, và cũng không được forward STP BPDU

Tiếp theo là listening state : interface ở trong trạng thái này có thể là Designated port hoặc root port, có độ dài vào khoảng 15s, được quyết định bởi forward delay timer. Ở trạng thái này port chỉ được nhận và gửi STP BPDU chứ không được gửi hoặc nhận những traffic mạng thông thường

Tiếp theo là learning state : sau khi qua trạng thái listening thì Designated port hoặc root port sẽ được vào trạng thái learning, trạng thái này dài vào khoảng 15s và cũng được gửi và nhận STP BPDU và không được gửi hoặc nhận những traffic mạng thông thường. Tuy nhiên ở trạng thái này interface có thể học được địa chỉ MAC từ traffic đến với interface.

Cuối cùng là forwarding state thì sẽ có tất cả khả năng bao gồm gửi nhận traffic thông thường, học địa chỉ MAC. Spanning tree timer:

| STP Timer | Purpose | Duration |
|----------------------|--|----------------------|
| Hello | How often the root bridge sends hello BPDUs | 2sec |
| Forward delay | How long the switch will stay in the Listening and Learning states (each state is 15 seconds = total 30 seconds) | 15sec |
| Max Age | How long an interface will wait <u>after ceasing to receive Hello BDPUs</u> to change the STP topology. | 20sec (10* hello) |

Hình 45: STP timer

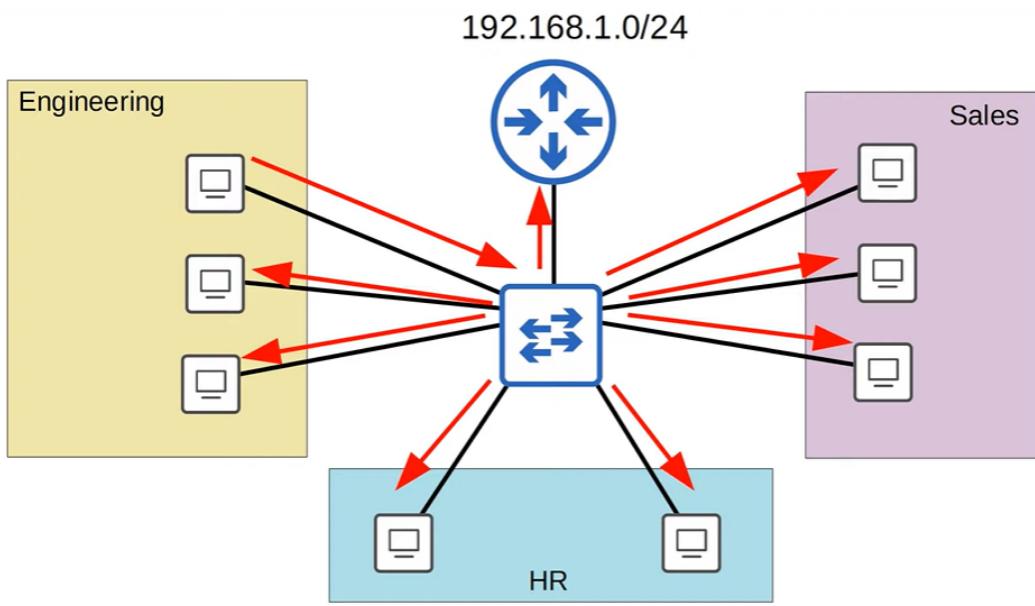
Dối với Max age timer nếu như BPDU được nhận trước khi max age timer bị xuống còn 0 thì thời gian sẽ được reset về 20 và không có sự thay đổi về topo mạng nào được diễn ra. Tuy nhiên khi mà không nhận được BPDU và max age timer bị xuống còn 0 thì switch sẽ tính toán loại lựa chọn STP của nó bao gồm root bridge, local root, designated, non-designated. Sau lựa chọn đó nếu một non-designated port được chọn để trở thành designated hoặc root port thì port đó sẽ chuyển từ trạng thái block sang listening rồi learning và cuối cùng là forwarding. Timer và những STP state sẽ đảm bảo được rằng loops sẽ không xảy ra nếu ta chuyển interface đó về trạng thái forwarding quá sớm.

STP optional feature:

- Portfast: Những interface được nối với PC ở switch sẽ không phải đợi để sang trạng thái forwarding và được chuyển thẳng thành trạng thái forwarding do những interface này không có khả năng tạo loop.
- BPDU guard: nếu interface được config chế độ này khi interface nhận được bản tin BPDU thì switch sẽ thực hiện shutdown luôn interface để tránh việc tạo ra vòng lặp.

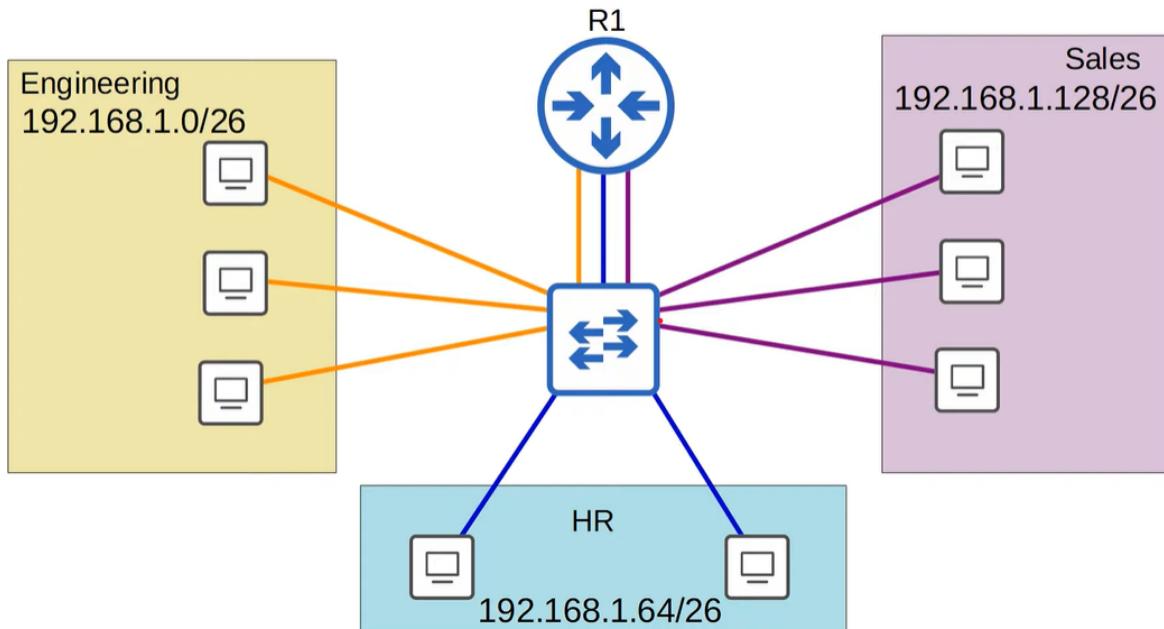
2.7 VLAN

LAN là một single broadcast domain, bao gồm tất cả các thiết bị trong broadcast domain đó. Xét một mô hình mạng trong 1 công ty dưới đây:



Hình 46: Topology

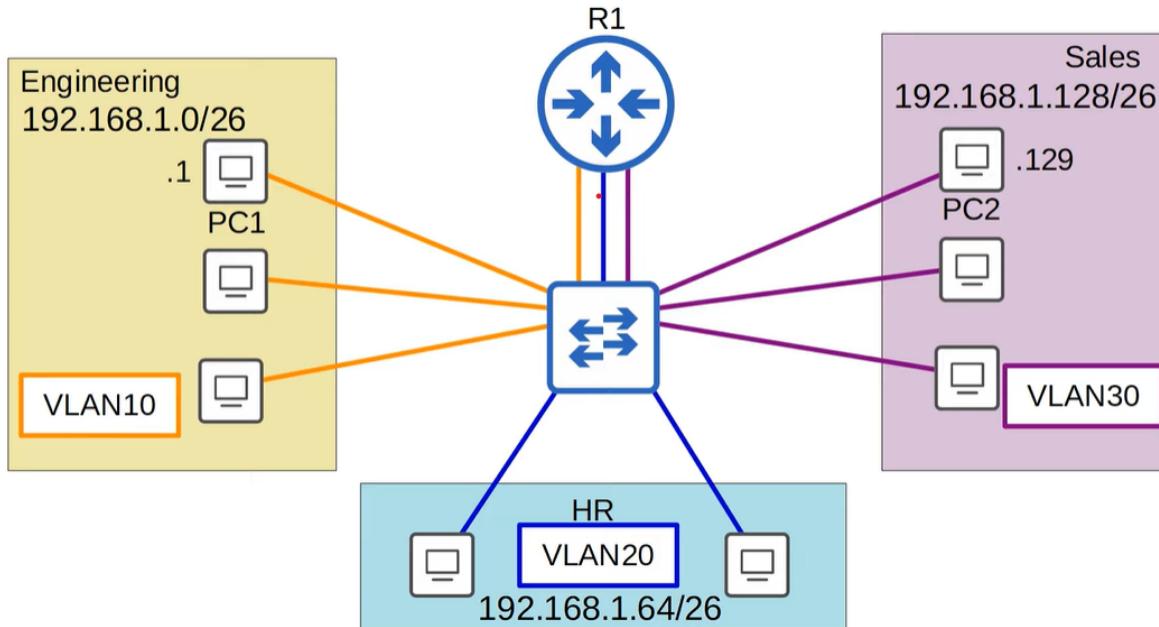
Giả sử khi một PC trong khu vực engineering muốn gửi broadcast frame đến tất cả các PC khác thuộc khu vực engineering thì khi switch nhận được broadcast frame nó sẽ flood broadcast frame đến tất cả các interface của nó. Do đó mà các PC khác không thuộc khu vực engineering sẽ nhận được broadcast frame. Điều này làm tăng traffic cho mô hình mạng ảnh hưởng đến performance của mạng cũng như liên quan đến các vấn đề bảo mật. Bởi lẽ, ta sẽ phải giới hạn xem ai sẽ là người có quyền được truy cập vào mỗi khu vực trong một công ty. Vì vậy một giải pháp đặt ra là ta sẽ chia cho mỗi khu vực 1 subnet riêng:



Hình 47: Topology 2

Giả sử PC1 trong engineering có địa chỉ IP là 192.168.1.1 muốn gửi 1 bản tin đến PC2 trong Sales có địa chỉ IP là 192.168.1.129. Do biết được PC2 nằm trong subnet khác với mình nên PC1 sẽ phải gửi bản tin qua switch rồi default gateway ở R1. Khi R1 nhận được bản tin nó sẽ gửi lại cho switch rồi switch mới gửi bản tin đến PC2 ở khu vực Sales. Ở đây switch chỉ có nhiệm vụ forward gói tin. Tuy nhiên ở cách trên, vẫn xảy ra một vấn đề đó là khi PC1 gửi broadcast frame hay unknown unicast frame. Vì switch chỉ quan tâm đến L2 nên khi nhận được broadcast frame với địa chỉ MAC đích là: FFFF.FFFF.FFFF thì nó vẫn sẽ flood broadcast frame ra tất cả các interface của switch dẫn đến việc tất cả các PC thuộc khu vực khác đều nhận được frame. Kết quả là hạn chế như ở mô hình cũ vẫn tồn tại. Do đó VLANs ra đời để khắc phục những điểm hạn chế trên.

VLANs hay còn là Virtual Local Area Network là một giải pháp chia một mạng LAN thành các VLANs nhỏ ở L2. Mỗi khu vực sẽ được coi là 1 VLANs. Để làm được việc này ta sẽ phải cấu hình cho interface của switch nằm trong một VLAN cụ thể nào đó và end-hosts khi được kết nối với interface đó sẽ là một phần của VLAN đó. Vì switch coi mỗi VLAN là một mạng LAN riêng biệt do đó nên nó sẽ không forward traffic giữa các VLAN bao gồm cả broadcast và unknown unicast traffic. Vì vậy nên khi thực hiện gửi broadcast frame thì switch sẽ chỉ forward đến các thiết bị trong cùng 1 VLAN thôi mà không phải forward ra tất cả các interface của nó.



Hình 48: VLAN

Trunk port Trong một mạng network nhỏ với vài VLANs, có thể dùng từng interface một cho từng VLANs khi kết nối switches to switches và switches to routers. Tuy nhiên, khi số lượng VLANs tăng lên, cách này không còn khả thi nữa vì nó sẽ gây lãng phí interface và thông thường một router sẽ không đủ interfaces cho mỗi VLAN. Trong tình huống này ta có thể dùng trunk port để carry traffic từ nhiều VLANs chỉ qua 1 interface mà thôi. Để nhận biết được traffic của VLANs nào, thì switches sẽ đánh dấu các frame được gửi qua trunk link. Điều này sẽ cho phép switch biết được frame thuộc VLANs nào.

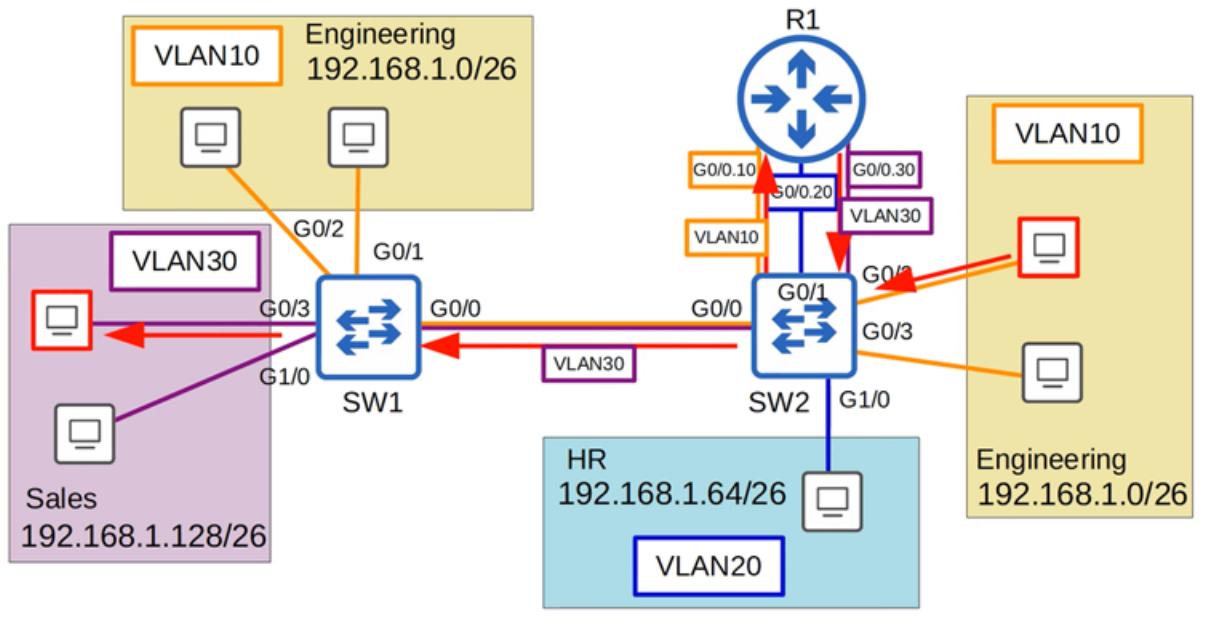
VLANs Tagging : Có độ dài 12 bits -> có khoảng 2 mũ 12 VLANs (0-4095). Dùng để xác định frame này thuộc VLANs nào.

Native VLAN Native VLAN được đặt mặc định là VLAN 1 trên tất cả các trunk ports, tuy nhiên có

có được được cấu hình là một VLAN khác. Khi switch nhận được frame không có tag trên trunk port thì nó sẽ cho là frame đó thuộc về native VLAN.

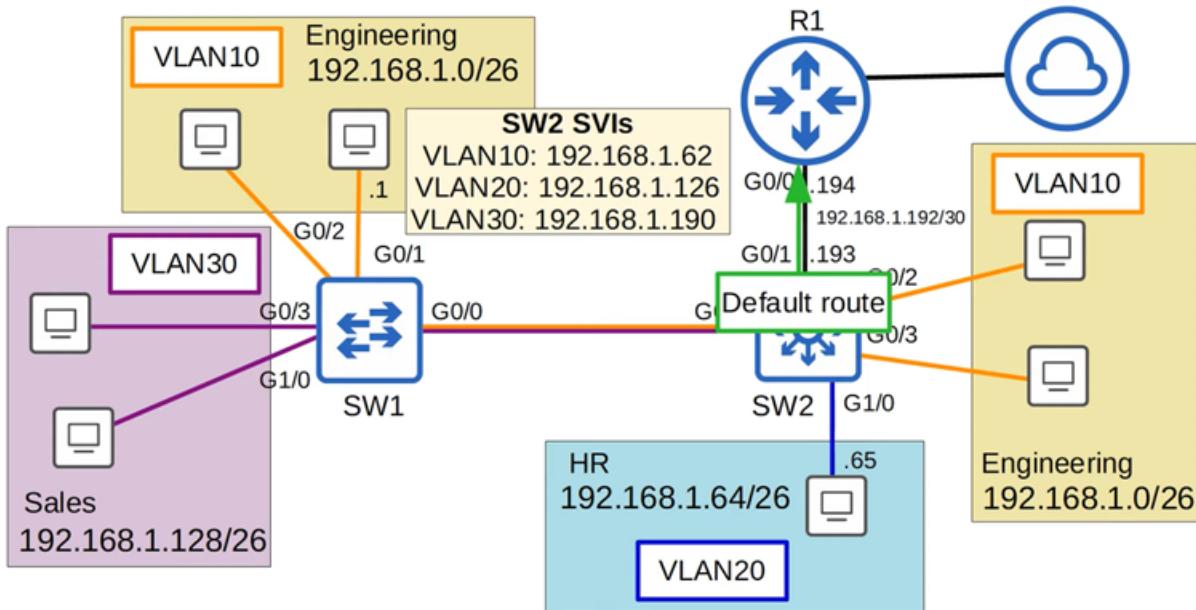
2.8 Inter VLAN Routing

- Sử dụng Router on a stick:

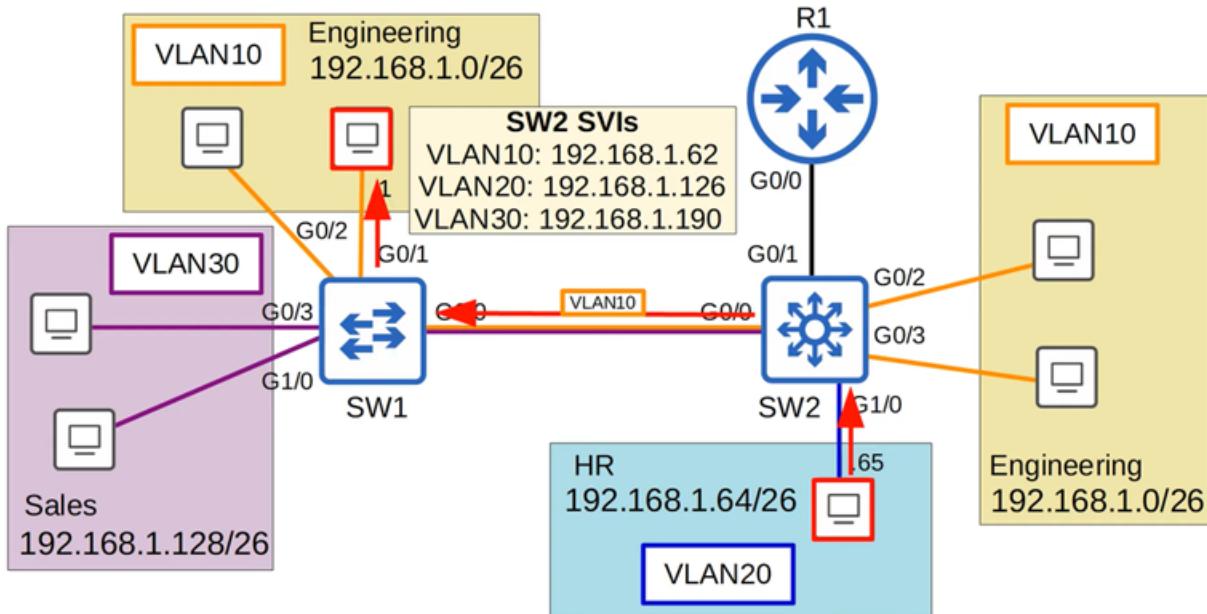


Hình 49: Vlan-routing

Ở đây, khi VLAN10 muốn gửi gói tin đến VLAN30, PC ở VLAN10 gửi đến G0/2 của SW2 và SW2 gửi frame đến R1 thông qua G0/1 và tag gói tin đó với tag name của VLAN10, R1 nhận được gói tin ở G0/0 và xác định được là gói tin đang đi vào subinterface G0/0.10 vì gói tin có tag VLAN10. Dịch đến thuộc vào subnet 192.168.1.128/26 được nối với R1 thông qua subinterface G0/0.30 sau đó R1 gửi gói tin ra khỏi G0/0 và tag gói tin đó với tag name của VLAN30 SW2 gửi gói tin thông qua trunk port đến SW1. SW1 sẽ thực hiện chuyển gói tin đến với đích trong VLAN30 - Trên Multi-layer switch: Thực hiện inter-VLAN routing thông qua SVI (Switch Virtual Interfaces) là một interface ảo mà ta có thể gán địa chỉ IP vào trong multi-layer switch, khi muốn gửi traffic đến VLAN khác thì switch sẽ tự route đường đi cho traffic mà không cần phải gửi lên cho router. Lúc này địa chỉ IP được gán cho các interface ảo sẽ được coi như là default gateway của các PC trong các VLAN



Hình 51: Vlan-routing 3



Hình 50: Vlan-routing 2

Khi mà muốn gửi traffic ra ngoài internet thì phải cấu hình default route cho SW2, tất cả các traffic mà có đích ngoài mạng LAN thì sẽ được chuyển đến cho R1

2.9 VTP

VTP cho phép cấu hình VLANs trên mỗi VTP server switch và những switch khác sẽ đồng bộ hóa cơ sở dữ liệu VLAN đến server. VTP được thiết kế cho một mạng lớn với nhiều VLAN để ta có thể không cần cấu hình mỗi VLAN trên từng switch. Nếu một switch với no VTP domain (domain NULL) nhận VTP

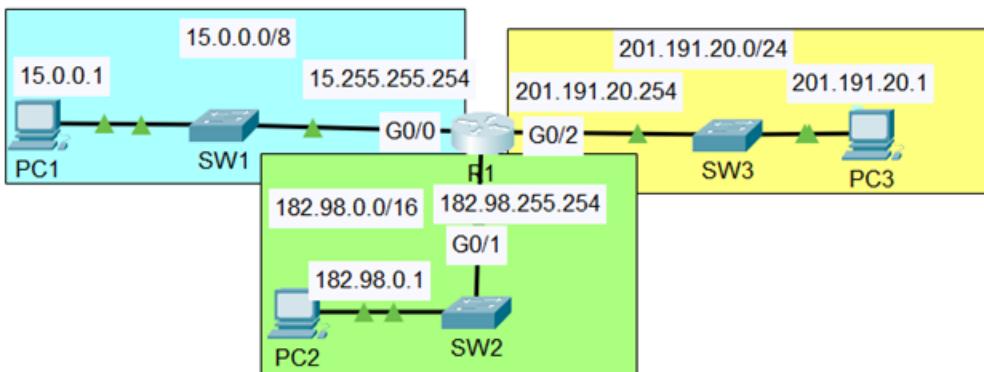
advertisement với một VTP domain name thì switch này sẽ join vào VTP domain đó. Nếu switch nhận được một VTP advertisement có revision number cao hơn thì nó sẽ tự động cập nhật VLAN database. VTP có 3 phiên bản 1,2 và 3, có 3 chế độ VTP: server, client, transparent.

Ở chế độ **server**: có thể xóa thêm, sửa VLANs, giữ một cơ sở dữ liệu VLAN, tăng revision number với mỗi lần một VLAN được xoá, thêm, sửa, quảng bá phiên bản mới nhất của VLAN database qua trunk interface và VTP client sẽ đồng bộ hóa VLAN database trên đó. Ở chế độ **client**: Không thể thêm sửa xóa VLAN, không lưu VLAN database (trừ phiên bản 3), đồng bộ hóa VLAN database đến server với revision number cao nhất trong vùng VTP, quảng bá phiên bản mới nhất của VLAN database qua trunk interface. Ở chế độ **transparent**: không tham gia vào VTP domain, duy trì VLAN database của riêng switch và có thể thêm sửa xóa VLAN và không quảng bá tới các switches khác.

3 LAB

3.1 Configure IP

Dùng một topo như sau:



Hình 52: Topology

Vào CLI của R1: en

Set hostname cho R1: hostname R1

```
Router>en
Router>enable
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#hos
Router(config)#hostname R1
R1(config)#do show ip interface brief
Interface          IP-Address      OK? Method Status          Protocol
GigabitEthernet0/0  unassigned     YES unset administratively down down
GigabitEthernet0/1  unassigned     YES unset administratively down down
GigabitEthernet0/2  unassigned     YES unset administratively down down
Vlan1              unassigned     YES unset administratively down down
```

Hình 53: Config router

Thấy R1 chưa được config IP cho các interface của nó Thực hiện config IP cho g0/0 các interface còn lại làm tương tự

Vào interface: interface gigabitEthernet0/0

Add ip: ip add 15.255.255.254 255.0.0.0

Lưu cấu hình: no shutdown

Sau khi đã thực hiện kiểm tra lại ip interface trên R1:

```
R1#show ip interface brief
Interface          IP-Address      OK? Method Status          Protocol
GigabitEthernet0/0  15.255.255.254 YES manual up           up
GigabitEthernet0/1  182.98.255.254 YES manual up           up
GigabitEthernet0/2  201.191.20.254 YES manual up           up
Vlan1              unassigned     YES unset administratively down down
```

Hình 54: ip-interface

Write để lưu lại tất cả cấu hình trên R1, Sau khi config IP trên Router thực hiện config IP trên các PC1,PC2,PC3 và dùng lệnh ping

```
C:\>ping 201.191.20.1

Pinging 201.191.20.1 with 32 bytes of data:

Request timed out.
Reply from 201.191.20.1: bytes=32 time<1ms TTL=127
Reply from 201.191.20.1: bytes=32 time=1ms TTL=127
Reply from 201.191.20.1: bytes=32 time<1ms TTL=127

Ping statistics for 201.191.20.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>ping 201.191.20.1

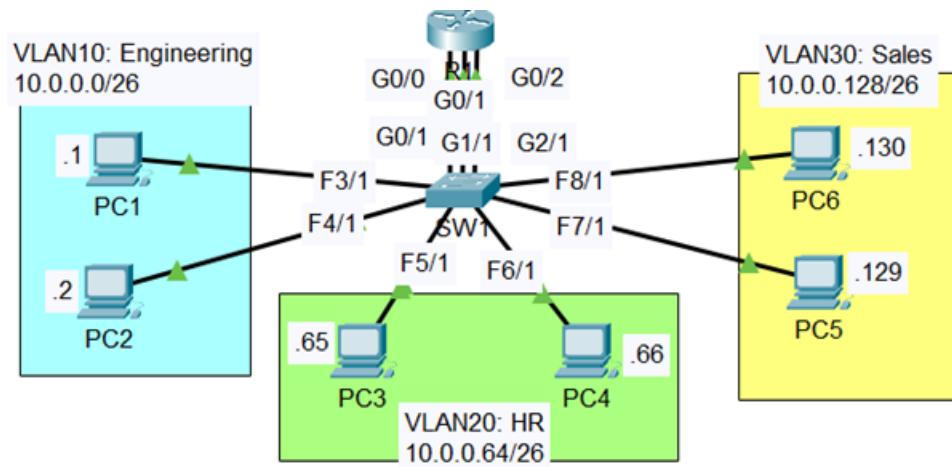
Pinging 201.191.20.1 with 32 bytes of data:

Reply from 201.191.20.1: bytes=32 time<1ms TTL=127
Reply from 201.191.20.1: bytes=32 time<1ms TTL=127
Reply from 201.191.20.1: bytes=32 time=6ms TTL=127
Reply from 201.191.20.1: bytes=32 time<1ms TTL=127

Ping statistics for 201.191.20.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 6ms, Average = 1ms
```

Hình 55: Ping

3.2 VLAN



Hình 56: Topo

Dặt IP cho từng host và đặt default gateway với IP là IP cuối trong subnet

Config interface cho router như trên:

```
R1(config)#do show ip int br
Interface          IP-Address      OK? Method Status
Protocol
GigabitEthernet0/0  10.0.0.62       YES manual up
GigabitEthernet0/1  10.0.0.126      YES manual up
GigabitEthernet0/2  10.0.0.190      YES manual up
Vlan1              unassigned      YES unset   administratively down down
R1(config)#

```

Hình 57: router

Thực hiện config VLAN cho switch:

```

SW1(config)#int range g0/1,f3/1,f4/1
SW1(config-if-range)#swi
SW1(config-if-range)#switchport m
SW1(config-if-range)#switchport mode ac
SW1(config-if-range)#switchport mode access
SW1(config-if-range)#sw
SW1(config-if-range)#switchport ac
SW1(config-if-range)#switchport access vlan 10
% Access VLAN does not exist. Creating vlan 10
SW1(config-if-range)#int range g1/1,f5/1,f6/1
SW1(config-if-range)#swit
SW1(config-if-range)#switchport mode
SW1(config-if-range)#switchport mode ac
SW1(config-if-range)#switchport mode access
SW1(config-if-range)#swi
SW1(config-if-range)#switchport acc
SW1(config-if-range)#switchport access vlan 20
% Access VLAN does not exist. Creating vlan 20
SW1(config-if-range)#int range g2/1,f8/1,f7/1
SW1(config-if-range)#sw mode ac
SW1(config-if-range)#sw ac val
SW1(config-if-range)#sw ac vl
SW1(config-if-range)#sw ac vlan 30
% Access VLAN does not exist. Creating vlan 30

```

Hình 58: VLAN

| VLAN Name | Status | Ports |
|-------------------------|--------|----------------------|
| 1 default | active | Fa9/1 |
| 10 ENGINEERING | active | Gig0/1, Fa3/1, Fa4/1 |
| 20 HR | active | Gig1/1, Fa5/1, Fa6/1 |
| 30 SALES | active | Gig2/1, Fa7/1, Fa8/1 |
| 1002 fddi-default | active | |
| 1003 token-ring-default | active | |
| 1004 fddinet-default | active | |
| 1005 trnet-default | active | |

Hình 59: VLAN

Check ping giữa 2 host khác vlan:

```

C:\>ping 10.0.0.65

Pinging 10.0.0.65 with 32 bytes of data:

Reply from 10.0.0.65: bytes=32 time=8ms TTL=127

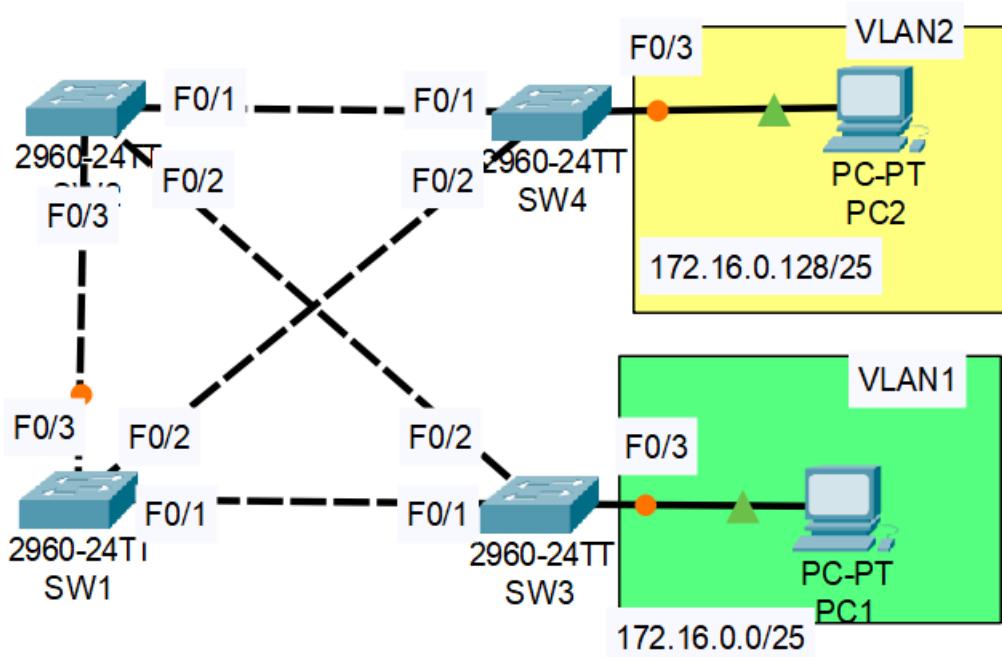
Ping statistics for 10.0.0.65:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 8ms, Average = 8ms

```

Hình 60: Ping

3.3 STP

STP lab:



Hình 61: STP

Thực hiện config SW1 là primary root cho VLAN1 và secondary root cho VLAN2. Còn SW2 là primary root cho VLAN2 và secondary root cho VLAN1.

```

SW2(config)#sp
SW2(config)#spanning-tree vlan 1 root secon
SW2(config)#spanning-tree vlan 1 root secondary
SW2(config)#sp
SW2(config)#spanning-tree v
SW2(config)#spanning-tree vlan 2 root pri
SW2(config)#spanning-tree vlan 2 root primary
SW2(config)#do show spanning-tree
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    24577
              Address     0060.2F90.D14A
              Cost         19
              Port        3 (FastEthernet0/3)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    28673  (priority 28672 sys-id-ext 1)
              Address     0001.4301.4B81
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time   20

  Interface      Role Sts Cost      Prio.Nbr Type
  -----  -----
Fa0/3          Root FWD 19       128.3    P2p
Fa0/1          Desg FWD 19      128.1    P2p
Fa0/2          Desg FWD 19      128.2    P2p

VLAN0002
  Spanning tree enabled protocol ieee
  Root ID    Priority    24578
              Address     0001.4301.4B81
              This bridge is the root
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    24578  (priority 24576 sys-id-ext 2)
              Address     0001.4301.4B81

```

Hình 62: stp config