Danh sách câu hỏi

[**Câu 1. Định nghĩa hệ phân tán, phân loại hệ phân tán?** 2](#_Toc498368660)

[**Câu 2. Mục tiêu của hệ phân tán?** 2](#_Toc498368661)

[**Câu 3. Các đặc điểm “concurrency of components”, “lack of a global clock”, “independent failures of components” là gì? Cho ví dụ về một hệ thống phân tán có cả ba đặc điểm này?** 3](#_Toc498368662)

[**Câu 4. Tại sao phải phần lớp cho quá trình truyền tin? Mô hình tham chiếu phân tầng OSI có mấy lớp. chức năng của mỗi lớp?** 3](#_Toc498368663)

[**Câu 5. So sánh TCP/IP và OSI?** 5](#_Toc498368664)

[**Câu 6. So sánh giao thức kết nối và giao thức không kết nối? Cho ví dụ việc sử dụng các dịch vụ cho hai giao thức này trong TCP/IP?** 5](#_Toc498368665)

[**Câu 7.Địa chỉ trong mô hình TCP/IP?** 5](#_Toc498368666)

[**Câu 8: Phân chia các tầng trong mô hình TCP/IP?** 5](#_Toc498368667)

[**Câu 9: Thuật ngữ “Forwarding” và “Delivery”?** 7](#_Toc498368668)

[**Câu 10: Xác định tên miền?địa chỉ IP,địa chỉ MAC?** 7](#_Toc498368669)

[**Câu 11: Middle ware là gì? Nằm ở đâu trong mô hình phân lớp OSI?** 7](#_Toc498368670)

[**Câu 12: Để xây dựng các giao thức tương ứng cho tầng middleware. Tóm tắt các bước trong 2 mô hình dịch vụ middleware?** 8](#_Toc498368671)

[**Câu 13: “Socket”, “message passing”, tiến trình (Process), luồng (Thread)?** 9](#_Toc498368672)

[**Câu 14: IPC (Inter Process Communication)?** 9](#_Toc498368673)

[**Câu 15: Các giao thức client/server?** 10](#_Toc498368674)

[**Câu 16: RPC dị bộ (Asynchronous RPC) vs. RPC đồng bộ (synchronuos RPC)?** 11](#_Toc498368675)

[**Câu 17: Little endian và big endian?** 11](#_Toc498368676)

[**18. Tại sao phải mật mã hóa? tính mật (confidentiality), tính toàn vẹn (intergrity), và tính khả dụng (availability) trong lý thuyết an ninh mạng, các loại mật mã?** 12](#_Toc498368677)

[**19. Các vấn đề về định thời, khái niệm “clock drift rate”, và “skew”?** 13](#_Toc498368678)

[**21. Định danh, tên và địa chỉ.** 15](#_Toc498368679)

[**22. Phân biệt URI, URN và URL. Cho ví dụ về việc phân giải tên miền.** 15](#_Toc498368680)

**Câu 1. Định nghĩa hệ phân tán, phân loại hệ phân tán?**

***Định nghĩa***. Có nhiều định nghĩa về hệ phân tán

***Định nghĩa 1***: Hệ phân tán là tập hợp các máy tính tự trị được kết nối với nhau bởi một mạng máy tính và được cài đặt phần mềm hệ phân tán.

***Định nghĩa 2***: Hệ phân tán là một hệ thống có chức năng và dữ liệu phân tán trên các trạm (máy tính) được kết nối với nhau bởi một mạng máy tính.

***Định nghĩa 3***: Hệ phân tán là một tập các máy tính độc lập giao tiếp với người dùng như một hệ thống thống nhất, toàn vẹn. Như vậy, có thể nói : Hệ phân tán = mạng máy tính + phần mềm hệ phân tán.

***Phân loại hệ phân tán:***

Trước đây, hệ phân tán được chia thành ba loại :  
+ hệ điều hành hệ phân tán,   
+cơ sở dữ liệu hệ phân tán   
+các hệ thống tính toán hệ phân tán.

Ngày nay, hệ phân tán được phân chia như sau:  
+Hệ phân tán mang tính hệ thống: hệ điều hành phân tán.  
+Hệ phân tán mang tính ứng dụng: các hệ thống truyền tin phân tán.

**Câu 2. Mục tiêu của hệ phân tán?**

Kết nối người sử dụng và tài nguyên Giải quyết bài toán chia sẻ tài nguyên trong hệ thống (resource sharing).

Chia sẻ nguồn tài nguyên (Resource Sharing):  
Là một đặc tính cơ bản của hệ thống phân tán, nó là cơ sở cho những đặc tính khác và nó ảnh hướng đến những kiến trúc phần mềm có sẵn trong các hệ phân tán. Trong hệ phân tán, tập tài nguyên chung là hữu hạn, số lượng các tiến trình cần chia sẻ tuy cũng là hữu hạn nhưng có thể rất lớn và phân tán trên nhiều trạm, vì thế cơ chế quản lý và chia sẻ tài nguyên trông hệ phân tán là rất phức tạp và khó khăn hơn nhiều so với hệ tập trung.

Các nguồn tài nguyên để chia sẻ có thể là phần cứng như ổ đĩa, máy in và các thiết bị ngoại vi khác, cũng có thể là phần mềm như hệ thống tập tin, cơ sở dữ liệu, các đối tượng dữ liệu … Các thiết bị phần cứng như máy in, ổ đĩa cứng dung lượng lớn và những thiết bị khác được chia sẻ để thuận tiện trong việc sử dụng và giảm chi phí.

Việc chia sẻ dữ liệu là một yêu cầu cần thiết trong nhiều ứng dụng máy tính, các nguồn tài nguyên của một máy tính đa người sử dụng thông thường được chia sẻ cho mọi người sử dụng. Những nguồn tài nguyên trong một hệ phân tán được gói gọn trong các máy tính và chỉ có thể truy nhập từ những máy tính khác bằng cách liên lạc.

Để việc chia sẻ hiệu quả, mỗi tài nguyên phải được quản lý bằng một chương trình, cung cấp một giao diện truyền thông cho phép tài nguyên được điều khiển truy cập và cập nhật một cách tin cậy và đồng nhất.

Ta có thể tiếp cận phương thức chia sẻ tài nguyên trên hai mô hình đặc trưng của hệ phân tán, đó là mô hình khách/chủ (Client/Server model) và mô hình hướng đối tượng (Object based model).

**Câu 3. Các đặc điểm “concurrency of components”, “lack of a global clock”, “independent failures of components” là gì? Cho ví dụ về một hệ thống phân tán có cả ba đặc điểm này?**

***“concurrency of components”***:   
+ Các thành phần trong DS thực hiện đồng thời quy trình.  
+ Các thành phần truy cập và cập nhật các tài nguyên dùng chung.  
(ví dụ: biến, cơ sở dữ liệu, trình điều khiển thiết bị).  
+ Tính toàn vẹn của hệ thống có thể bị vi phạm nếu cập nhật đồng thời không được phối hợp  
+ Bảo quản tính toàn vẹn đòi hỏi sự đồng thờikiểm soát nơi truy cập đồng thời vào cùng một tài nguyên được đồng bộ.

***“independent failures of components”:***Tất cả các hệ thống máy tính có thể bị lỗi và nhà thiết kế hệ thống sẽchịu trách nhiệm hoạch định hậu quả của việc hệ thống có thể bị lỗi.Mỗi thành phần của một hệ thống phân tán có thể bị lỗi độc lập, để những người khác vẫn chạy. Bị lỗi có thể là do tai nạn hoặc phản ứng chậm.

***“lack of global clock”:***Trong một hệ thống phân phối có rất nhiều đồng hồ như có hệ thống. Các đồng hồ được phối hợp để giữ cho chúng phù hợp nhưng không có đồng hồ nào có thời gian chính xác. Ngay cả khi đồng hồ là một số đồng bộ, các đồng hồ cá nhân trên mỗi thành phần có thể chạy với tốc độ hoặc chi tiết khác nhau dẫn đến chúng không đồng bộ sau một chu kỳ đồng hồ cục bộ.

Thời gian chỉ được biết đến trong một độ chính xác nhất định. Ở những khoảng thời gian thường xuyên, đồng hồ có thể đồng bộ với đồng hồ đáng tin cậy hơn. Tuy nhiên, các đồng hồ không phải là chính xác như nhau vì thời gian trễ do truyền tải và thực hiện.

Xem xét một nhóm người đi dự một cuộc họp. Mỗi người đều có đồng hồ. Mỗi đồng hồ đều có thời gian tương tự, nhưng khác nhau. Ngay cả với những sai sót trong thời gian, nhóm có thể đáp ứng và tiến hành kinh doanh. Đây là cách hoạt động của thời gian phân phối.

Điều này trái với một đồng hồ trên một hệ thống duy nhất. Ở đây chỉ có một đồng hồ và nó cung cấp một thời gian thống nhất cho tất cả các thành phần phụ trên hệ thống riêng lẻ này.

**Câu 4. Tại sao phải phần lớp cho quá trình truyền tin? Mô hình tham chiếu phân tầng OSI có mấy lớp. chức năng của mỗi lớp?**

***Lớp ứng dụng (ApplicationLayer):*** là giao diện giữa các chương trình ứng dụng của người dùng và mạng. Lớp Application xử lý truy nhập mạng chung, kiểm soát luồng và phục hồi lỗi. Lớp này không cung cấp các dịch vụ cho lớp nào mà nó cung cấp dịch vụ cho các ứng dụng như: truyền file, gởi nhận E-mail, Telnet, HTTP, FTP, SMTP…

***Lớp trình bày (PresentationLayer):*** lớp này chịu trách nhiệm thương lượng và xác lập dạng thức dữ liệu được trao đổi. Nó đảm bảo thông tin mà lớp ứng dụng của một hệ thống đầu cuối gởi đi, lớp ứng dụng của hệ thống khác có thể đọc được. Lớp trình bày thông dịch giữa nhiều dạng dữ liệu khác nhau thông qua một dạng chung, đồng thời nó cũng nén và giải nén dữ liệu. Thứ tự byte, bit bên gởi và bên nhận qui ước qui tắc gởi nhận một chuỗi byte, bit từ trái qua phải hay từ phải qua trái. Nếu hai bên không thống nhất thì sẽ có sự chuyển đổi thứ tự các byte bit vào trước hoặc sau khi truyền. Lớp presentationcũng quản lý các cấp độ nén dữ liệu nhằm giảm số bit cần truyền. Ví dụ: JPEG,ASCCI, EBCDIC....

***Lớp phiên (SessionLayer):*** lớp này có chức năng thiết lập, quản lý, và kết thúc các phiên thông tin giữa hai thiết bị truyền nhận. Lớp phiên cung cấp các dịch vụ cho lớp trình bày. Lớp Session cung cấp sự đồng bộ hóa giữa các tác vụ người dùng bằng cách đặt những điểm kiểm tra vào luồng dữ liệu. Bằng cách này, nếu mạng không hoạt động thì chỉ có dữ liệu truyền sau điểm kiểm tra cuối cùng mới phải truyền lại. Lớp này cũng thi hành kiểm soát hội thoại giữa các quá trình giao tiếp, điều chỉnh bên nào truyền, khi nào, trong bao lâu. Ví dụ như: RPC,NFS,... Lớp này kết nối theo ba cách: Haft-duplex, Simplex, Full-duplex.

***Lớp vận chuyển (TransportLayer):*** lớp vận chuyển phân đoạn dữ liệu từ hệ thống máy truyền và tái thiết lập dữ liệu vào một luồng dữ liệu tại hệ thống máy nhận đảm bảo rằng việc bàn giao các thông điệp giữa các thiết bị đáng tin cậy.

***Lớp mạng (NetworkLayer):*** lớp mạng chịu trách nhiệm lập địa chỉ các thông điệp, diễn dịch địa chỉ và tên logic thành địa chỉ vật lý đồng thời nó cũng chịu trách nhiệm gởi packet từ mạng nguồn đến mạng đích. Lớp này quyết định đường đi từ máy tính nguồn đến máy tính đích. Nó quyết định dữ liệu sẽ truyền trên đường nào dựa vào tình trạng, ưu tiên dịch vụ và các yếu tố khác

***Lớp datalink***, các bít đến từ lớp vật lý được chuyển thành các frame dữ liệu bằng cách dùng một số nghi thức tại lớp này. Lớp data linkđược chia thành hai lớp con:

+ Lớp con LLC (logical link control): Lớp con LLC là phần trên so với các giao thức truy cập đường truyền khác, nó cung cấp sự mềm dẻo về giao tiếp. Bởi vì lớp con LLC hoạt động độc lập với các giao thức truy cập đường truyền, cho nên các giao thức lớp trên hơn (ví dụ như IP ở lớp mạng) có thể hoạt động mà không phụ thuộc vào loại phương tiện LAN. Lớp con LLC có thể lệ thuộc vào các lớp thấp hơn trong việc cung cấp truy cập đường truyền.

+ Lớp con MAC (media access control): Lớp con MAC cung cấp tính thứ tự truy cập vào môi trường LAN. Khi nhiều trạm cùng truy cập chia sẻ môi trường truyền, để định danh mỗi trạm, lớp cho MAC định nghĩa một trường địa chỉ phần cứng, gọi là địa chỉ MAC address. Địa chỉ MAC là một con số đơn nhất đối với mỗi giao tiếp LAN (card mạng).

***Lớp vật lý (PhysicalLayer):*** định nghĩa các qui cách về điện, cơ, thủ tục và các đặc tả chức năng để kích hoạt, duy trì và dừng một liên kết vật lý giữa các hệ thống đầu cuối. Một số các đặc điểm trong lớp vật lý này bao gồm:

+Mức điện thế.  
+Khoảng thời gian thay đổi điện thế.  
+Tốc độ dữ liệu vật lý.  
+Khoảng đường truyền tối đa.  
+Các đầu nối vật lý.

**Câu 5. So sánh TCP/IP và OSI?**

***Các điểm giống nhau:***- Cả hai đều có kiến trúc phân lớp.- Đều có lớp Application, mặc dù các dịch vụ ở mỗi lớp khác nhau.- Đều có các lớp Transport và Network.- Sử dụng kĩ thuật chuyển packet (packet-switched).- Các nhà quản trị mạng chuyên nghiệp cần phải biết rõ hai mô hình trên.

***Các điểm khác nhau***:  
- Mô hình TCP/IP kết hợp lớp Presentation và lớp Session vào trong lớp Application.  
- Mô hình TCP/IP kết hợp lớp DataLink và lớp Physical vào trong một lớp.  
- Mô hình TCP/IP đơn giản hơn bởi vì có ít lớp hơn.  
- Nghi thức TCP/IP được chuẩn hóa và được sử dụng phổ biến trên toàn thế giới.

**Câu 6. So sánh giao thức kết nối và giao thức không kết nối? Cho ví dụ việc sử dụng các dịch vụ cho hai giao thức này trong TCP/IP?**

***Đặc điểm của giao thức không kết nối:***- Không kiểm soát đường truyền- Dữ liệu không bảo đảm đến được nơi nhận- Dữ liệu thường dưới dạng datagrams

Ví dụ: giao thức UDP của TCP/IP

***Đặc điểm của giao thức hướng kết nối:***

- Ngược lại với giao thức không kết nối , kiểm soát được đường truyền  
- Dữ liệu truyền đi tuần tự, nếu nhận thành công thì nơi nhận phải gởi tín hiệu ACK (ACKnowledge)

Ví dụ: các giao thức TCP, SPX

**Câu 7.Địa chỉ trong mô hình TCP/IP?**

Khi truyền dữ liệu , quá trình tiến hành từ tầng trên xuống tầng dưới, qua mỗi tầng dữ liệu được them vào thông tin điều khiển gọi là Header. Khi nhận dữ liệu thì quá trình xảy ra ngược lại. dữ liệu được truyền từ tấng dưới lên và qua mỗi tầng thì phần header tương ứng sẽ được lấy đi và khi đến tầng trên cùng thì dữ liệu không còn phần header nữa.

**Câu 8: Phân chia các tầng trong mô hình TCP/IP?**

***Gồm 4 tầng***:Tầng ứng dụng(Application),Tầng giao vận(Transport),Tầng mạng(Network),Tầng liên kết(Data Link)

***Tầng ứng dụng:***-Nơi các chương trình mạng thường dùng làm việc nhất nhằm liên lạc giữa các nút trong một mạng-Giao tiếp xảy ra trong tầng này là tùy theo các ứng dụng cụ thể và dữ liệu được truyền từ chương trình, trong định dạng được sử dụng nội bộ bởi ứng dụng này, và được đóng góitheo một giao thức tầng giao vận.-Hai giao thức tầng thấp thông dụng nhất là TCP và UDP. Mỗi ứng dụng sử dụng dịch vụ của một trong hai giao thức trên đều cần có cổng. Hầu hết các ứng dụng thông dụng có các cổng đặc biệt được cấp sẵn cho các chương trình phục vụ (server)(HTTP - Giao thức truyền siêu văn bản dùng cổng 80; FTP - Giao thức truyền tệp dùng cổng 21, v.v..) trong khi các trình khách (client) sử dụng các cổng tạm thời (ephemeral port).

***Tầng giao vận:***Kết hợp các khả năng truyền thông điệp trực tiếp (end-to-end) không phụ thuộc vào mạng bên dưới, kèm theo kiểm soát lỗi (error control), phân mảnh (fragmentation) và điều khiển lưu lượng. Việc truyền thông điệp trực tiếp hay kết nối các ứng dụng tại tầng giao vận có thể được phân loại như sau:

- Định hướng kết nối (connection-oriented), ví dụ TCP

- Phi kết nối (connectionless), ví dụ UDP

Tầng giao vận có thể được xem như một cơ chế vận chuyển thông thường, nghĩa là trách nhiệm của một phương tiện vận tải là đảm bảo rằng hàng hóa/hành khách của nó đến đích an toàn và đầy đủ.

Tầng giao vận cung cấp dịch vụ kết nối các ứng dụng với nhau thông qua việc sử dụng các cổng TCP và UDP. Do IP chỉ cung cấp dịch vụ phát chuyển nỗ lực tối đa (best effort delivery), tầng giao vận là tầng đâu tiên giải quyết vấn đề độ tin cậy.

***Tầng mạng:***Theo định nghĩa ban đầu, tầng mạng giải quyết các vấn đề dẫn các gói tin qua một mạng đơn. Một số ví dụ về các giao thức như vậy là X.25, và giao thức Host/IMP của mạng ARPANET.

Trong bộ giao thức liên mạng, giao thức IP thực hiện nhiệm vụ cơ bản dẫn đường dữ liệu từ nguồn tới đích. IP có thể chuyển dữ liệu theo yêu cầu của nhiều giao thức tầng trênkhác nhau; mỗi giao thức trong đó được định danh bởi một số hiệu giao thức duy nhất: giao thức ICMP (Internet Control Message Protocol) là giao thức 1 và giao thức IGMP(Internet Group Management Protocol) là giao thức 2.

***Tầng liên kết:***Tầng liên kết - phương pháp được sử dụng để chuyển các gói tin từ tầng mạng tới các máy chủ (host) khác nhau - không hẳn là một phần của bộ giao thức TCP/IP, vì giao thức IP có thể chạy trên nhiều tầng liên kết khác nhau. Các quá trình truyền các gói tin trên một liên kết cho trước và nhận các gói tin từ một liên kết cho trước có thể được điều khiển cả trong phần mềm điều vận thiết bị (device driver) dành cho cạc mạng, cũng như trong phần sụn (firmware) hay các chipset chuyên dụng. Những thứ đó sẽ thực hiện các chức năng liên kết dữ liệu chẳng hạn như bổ sung một tín đầu (packet header) để chuẩn bị cho việc truyền gói tin đó, rồi thực sự truyền frame dữ liệu qua một môi trường vật lý.

Tầng liên kết còn có thể là tầng nơi các gói tin được chặn (intercepted) để gửi qua một mạng riêng ảo (virtual private network). Khi xong việc, dữ liệu tầng liên kết được coi là dữ liệu của ứng dụng và tiếp tục đi xuống theo chồng giao thức TCP/IP để được thực sự truyền đi. Tại đầu nhận, dữ liệu đi lên theo chồng TCP/IP hai lần (một lần cho mạng riêng ảo và lần thứ hai cho việc định tuyến).

Tầng liên kết còn có thể được xem là bao gồm cả tầng vật lý - tầng là kết hợp của các thành phần mạng vật lý thực sự (hub, các bộ lặp (repeater), cáp mạng, cáp quang, cáp đồng trục (coaxial cable), cạc mạng, cạc HBA (Host Bus Adapter) và các thiết bị nối mạng có liên quan: RJ-45, BNC, etc), và các đặc tả mức thấp về các tín hiệu (mức hiệu điện thế, tần số, v.v..).

**Câu 9: Thuật ngữ “Forwarding” và “Delivery”?**

***Forwarding*** là quá trình chuyển tiếp một port cụ thể từ mạng này đến mạng khác. Kỹ thuật chuyển tiếp này cho phép người dùng bên ngoài có thể truy cập vào mạng bên trong bằng cách sử dụng port đó từ bên ngoài thông qua bộ định tuyến (đã mở NAT -Network Address Translation).

Về cơ bản, Port Forwarding cho phép máy tính bên ngoài kết nối đến máy tính bên trong mạng nội bộ. Thông thường, router/server mở sẵn một số port, chẳng hạn port 21 cho dịch vụ chia sẻ tập tin qua FTP, port 80 cho các máy chủ dịch vụ web (web server)... Để quản lý Port Forwarding, các HĐH như Mac và BSD (Berkeley Software Distribution) sẽ được cài đặt sẵn cách dịch vụ tương ứng trong nhân (kernel), tường lửa ip (ipfirewall - ipfw), còn Linux cũng sẽ thêm iptable để hỗ trợ Port Forwarding.

***Delivery***

Dùng trong thuật ngữ CDN(content delivery network), CDN hiểu một cách đơn giản là một mạng máy tính trải dài toàn cầu có nhiệm vụ lưu trữ hình ảnh, video hoặc các file nào đó của bạn và phân phối đến người dùng gần nhất về mặc địa lý

**Câu 10: Xác định tên miền?địa chỉ IP,địa chỉ MAC?**

***Địa chỉ IP*** (hay địa chỉ giao thức Internet) dùng để nhận dạng máy tính trong mạng và những thiết bị trong một mạng. Khi các máy tính truyền thông với nhau trên Internet hay một mạng nội bộ, chúng gửi thông tin cho địa chỉ IP của nhau. Các địa chỉ IP có thể là địa chỉ công cộng hoặc địa chỉ riêng. Địa chỉ IP công cộng là địa chỉ khả dụng trên Internet, trong khi địa chỉ riêng thì không

***Địa chỉ MAC*** là mã duy nhất được gán bởi nhà sản xuất cho từng phần cứng mạng (như cạc không dây hoặc cạc Ethernet). MAC viết tắt của Media Access Control, và mỗi mã là duy nhất cho một thiết bị.

Địa chỉ MAC là một bộ sáu cặp hai ký tự, cách nhau bằng dấu hai chấm. Ví dụ 00:1

B:44:11:3A:B7 là một địa chỉ MAC.

**Câu 11: Middle ware là gì? Nằm ở đâu trong mô hình phân lớp OSI?**

***Middleware*** là phần mềm máy tính với nhiệm vụ kết nối các thành phần phần mềm hoặc các ứng dụng với nhau. Phần mềm loại này bao gồm một tập các dịch vụ cho phép sự tương tác giữa các tiến trình chạy trên một hoặc nhiều máy khác nhau. Công nghệ middleware đã được phát triển để cung cấp khả năng hoạt động tương hỗ, phục vụ cho các kiến trúc phân tán thường được để hỗ trợ và đơn giản hóa các ứng dụng phân tán phức tạp.

Middleware nằm ở giữa các ứng dụng phần mềm chạy trên các hệ điều hành khác nhau. Nó tương tự với tầng giữa của một kiến trúc hệ thống đơn 3 tầng, chỉ khác ở chỗ nó trải rộng qua các hệ thống và ứng dụng khác nhau. Ví dụ là các phần mềm EAI, phần mềm truyền thông, Transaction Processing System, và các phần mềm thông điệp-và-hàng đợi.

đôi khi còn được gọi là đường ống (plumping) vì nó kết nối hai ứng dụng với nhau và truyền dữ liệu giữa chúng. Middleware cho phép một cơ sở dữ liệu truy nhập dữ liệu tại một cơ sở dữ liệu khác

***MiddleWare nằm giữa tầng mạng và tầng ứng dụng(tầng phiên)***

tên miền được tạo thành từ các nhãn không rỗng phân cách nhau bằng dấu chấm (.); những nhãn này giới hạn ở các chữ cái ASCII từ a đến z (không phân biệt hoa thường), chữ số từ 0 đến 9, và dấu gạch ngang (-), kèm theo những giới hạn về chiều dài tên và vị trí dấu gạch ngang. Đó là dấu gạch ngang không được xuất hiện ở đầu hoặc cuối của nhãn, và chiều dài của nhãn nên trong khoảng từ 1 đến 63 và tổng chiều dài của một tên miền không được vượt quá 255.

**Câu 12: Để xây dựng các giao thức tương ứng cho tầng middleware. Tóm tắt các bước trong 2 mô hình dịch vụ middleware?**

***Tóm tắt các bước trong mô hình gọi thủ tục từ xa RPC*** (Remote Procedure Call):

\* client stub và server stub là các hàm chuyển đổi tham số giúp giảm nhẹ công việc cho client và server.

\* Việc truyền tham số từ client đến có 2 cách truyền: tham biến và tham trị.

\* Một máy A muốn thực hiện một thủ tục nào đó trên máy B khác thì nó sẽ thực hiện một lời gọi thủ tục từ xa đến máy B. Thủ tục đó sẽ được thực hiện ở máy B dựa trên các tham số, thông điệp được truyền đến từ máy A và kết quả sẽ được truyền về lại cho máy A tương ứng. Lời gọi này được thực hiện thông qua các bước sau:

1. Client thực hiện gọi đoạn hàm chuyển đổi tham số của client khi truyền giữa client và server.

2. Hàm chuyển đổi tham số sẽ tạo nên một tin nhắn và gọi cho hệ điều hành cục bộ.

3. Client gửi tin nhắn đến cho bộ điều khiển từ xa.

4. Bộ điều khiển từ xa sẽ chuyển tin nhắn đến cho hàm chuyển đổi tham số của server.

5. Hàm chuyển đổi tham số của server sẽ lấy tham số và gọi cho server.

6. Server sẽ làm việc và trả về kết quả cho hàm chuyển đổi tham số của server.

7. Hàm chuyển đổi tham số của server sẽ đóng gói kết quả lại thành một tin nhắn và yêu cầu gửi trả về lại cho client.

8. Server sẽ gửi tin nhắn đó đến cho client.

9. Client chuyển tin nhắn này đến cho hàm chuyển đổi tham số của client.

10. Hàm chuyển đổi tham số của client thực hiện lấy, kiểm tra tin nhắn và gửi lại kết quả cho client.

b) Mô hình dịch vụ hướng thông điệp (Message - oriented Middleware)

1. Truyền thông liên tục (persistent communication): khi một client muốn gửi một bản tin đi trên mạng, nó sẽ gửi bản tin đó đến server cục bộ thông qua việc truyền đi trong mạng cục bộ. Server này tạm thời lưu trữ bản tin đó vào bộ nhớ đệm của mình, xác định địa chỉ client đích, rồi gửi đến cho server cục bộ của client đích tương ứng (có thể đi qua nhiều server trung gian khác). Khi đã đến được server cuối cùng, bản tin lúc này sẽ được lưu lại trước khi chuyển về cho client tương ứng.

2. Truyền thông nhất thời (Transient communication): bản tin gửi đi chỉ được lưu lại trong phiên trao đổi đó. Khi phiên trao đổi đã hoàn thành hoặc khi kết nối bị hủy bỏ thì các bản tin đó cũng bị hủy bỏ trên các server. Do vậy nếu như một trong các server trung gian vì lý do nào đó không thể chuyển tiếp bản tin thì bản tin đó sẽ bị hủy bỏ.

3. Truyền thông đồng bộ (Synchronous communication): khi một client gửi đi một bản tin, nó sẽ ở trạng thái khóa (blocked) cho đến khi nhận được thông báo bản tin đó đã đến đích thành công.

4. Truyền thông bất đồng bộ (Asynchronous communication): khi client gửi đi một bản tin, nó vẫn sẽ tiếp tục thực hiện công việc của mình. Điều này có nghĩa là bản tin đó được lưu lại trên bộ nhớ đệm của client gửi hoặc của server cục bộ.

**Câu 13: “Socket”, “message passing”, tiến trình (Process), luồng (Thread)?**

***Socket:*** một cổng logic mà một chương trình sử dụng để kết nối với một chương trình khác chạy trên một máy tính khác trên Internet. Chương trình mạng có thể sử dụng nhiều Socket cùng một lúc, nhờ đó nhiều chương trình có thể sử dụng Internet cùng một lúc.

Có 2 loại socket:  
+Stream socket: Dựa trên giao thức TCP( Tranmission Control Protocol) việc truyền dữ iệu chỉ thực hiện giữa 2 quá trình đã thiết lập kết nối. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.  
+Datagram Socket: Dựa trên giao thức UDP( User Datagram Protocol) việc truyền dữ liệukhông yêu cầu có sự thiết lập kết nối giữa 2 quá trình. Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thế không đúng trình tự và lặp lại.

***Message passing:*** Các tiến trình trao đổi, liên lạc trực tiếp thông qua cách gửi và nhận các thông điệp. Có 2 dạng synchromous - blocking gửi nhận và asynchronous – non-blocking gửi.  
synchromous – blocking  
asynchronous – non-blocking

***Process***: là trạng thái tức thời của một chương trình đang chạy trên máy tính . Nó bao gồm bộ nhớ cần thiết để chạy chương trình (không gian địa chỉ của quá trình) và khả năng kiểm soát hiện trạng của bộ xử lý trong quá trình thực thi chương trình (tiến trình điều khiển của quá trình).

***Thread***: là một hay một phần của chương trình đang thực hiện, mức thấp hơn của process, mỗi process có thể gồm nhiều thread, các thread hoạt động giống như process nhưng các thread cùng chia sẻ không gian địa chỉ chung, các process thì hoàn toàn độc lập.

**Câu 14: IPC (Inter Process Communication)?**

- Trong một ứng dụng đơn khối (monolithic), các components tương tác với nhau thông qua việc truy vấn method và function. Ngược lại, các ứng dụng Microservices là một hệ thống phân tán chạy trên nhiều máy. Mỗi service instance là một quá trình đặc trưng. Do đó, sự tương tác giữa các services cần đến kỹ thuật IPC, viết tắt của inter-process communication: hành động trao đổi dữ liệu giữa các tiến trình riêng biệt, sử dụng giao thức kết nối.

- Khi lựa chọn kĩ thuật IPC cho một dịch vụ, ta cần hiểu các services tương tác với nhau như thế nào. Có vô số cách tương tác giữa client-service. Ta có thể chia chúng thành 2 nhóm theo 2 cách:

Cách 1: tương tác/ kết nối 1-1 hoặc 1-nhiều – gọi tắt là 1-n  
Cách 2: kết nối đồng bộ (synchromous) hoặc bất đồng bộ (asynchronous)

- Một API của dịch vụ là một hợp đồng giữa dịch vụ đó và các client của nó. Bất kể sự lựa chọn của bạn về cơ chế IPC, điều quan trọng là phải xác định chính xác API của dịch vụ bằng cách sử dụng một số loại ngôn ngữ định nghĩa giao diện (IDL - Interface Definition Language). Thậm chí có những lý lẽ tốt cho việc sử dụng một cách tiếp cận API đầu tiên (API First-Approach) để định nghĩa các dịch vụ. Bạn bắt đầu phát triển một dịch vụ bằng cách viết định nghĩa giao diện và xem xét nó với các nhà phát triển client. Đó là chỉ sau khi lặp qua định nghĩa API mà bạn thực thi dịch vụ đó. Việc thực hiện trước thiết kế này làm tăng cơ hội của bạn trong việc xây dựng một dịch vụ đáp ứng nhu cầu của các client.

- API của service thay đổi theo thời gian. Với ứng dụng đơn khối, cập nhật API nhìn chung khá dễ dàng. Đối với ứng dụng kiểu microservices, nó khó gấp 10 lần. Ta không thể ép các clients update cùng thời điểm với dịch vụ. Do đó, việc cho ra đời các phiên bản mới, đồng thời hỗ trợ các bản cũ là điều tất yếu. Việc đề ra chiến thuật hợp lý cho vấn đề này là rất quan trọng.

- Có hàng tá các công nghệ IPC khác nhau. Bạn có thể chọn các kỹ thuật tương tác đồng bộ, chẳng hạn giao thức HTTP dựa trên REST, hoặc Thrift. Một số lựa chọn khác thuộc kỹ thuật tương tác không đồng bộ: AMQP, STOMP. Bên cạnh đó, định dạng của thông báo (message) cũng vô cùng phong phú. Dịch vụ có thể sử dụng những định dạng gần gũi với ngôn ngữ con người như JSON, XML hoặc dạng nhị phân (không gần gũi tẹo nào, nhưng hiệu năng cao!). Các đại diện cho message dạng nhị phân: Avro, Protocol Buffers. Hãy để dành kỹ thuật IPC đồng bộ cho đoạn tiếp theo, và bây giờ, chúng ta hãy dành thời gian cho kỹ thuật IPC không đồng bộ (asynchronous IPC mechanisms).

+Kỹ thuật tương tác không đồng bộ dựa trên hệ thống thông điệp:  
+Kỹ thuật tương tác đồng bộ dựa trên request/response IPC:

**Câu 15: Các giao thức client/server?**

• Để cho một client có thể giao tiếp được với server thì giữa chúng phải có một chuẩn để nói chuyện, giao tiếp với nhau, chuẩn này được gọi là giao thức. Client muốn yêu cầu lấy thông tin từ server thì phải tuân theo giao thức mà server đó đưa ra. Hiện nay có rất nhiều giao thức chuẩn trên mạng ra đời nhằm đáp ứng nhu cầu này :

◦ ***HTTP***: giao thức truyền tải siêu văn bản, được dùng để liên hệ thông tin giữa máy chủ cung cấp dịch vụ và máy sử dụng dịch vụ, thuộc bộ giao thức TCP/IP.

◦ ***FTP***: giao thức truyền tập tin dùng để trao đổi tập tin qua mạng lưới truyền thông dùng bộ giao thức TCP/IP. Cần có một máy chủ FTP lằng nghe yêu cầu và các máy khác chạy phần mềm FTP để sử dụng dịch vụ.

◦ ***SMTP***: giao thức truyền tải thư điện tử đơn giản, mục đích dùng để liên lạc với server từ xa và gửi email từ mail client đến mail server và sau đó gửi đến server mail của email nhận.

◦ ***POP***: giao thức dùng để lấy thư điện tử từ server mail và tải xuống client thông qua kết nối TCP/IP.

◦ ***IMAP***: dạng thế hệ mới hơn của giao thức POP, khác biệt so với POP là nó chỉ lấy email headers về, nội dung email vẫn còn trên server.

**Câu 16: RPC dị bộ (Asynchronous RPC) vs. RPC đồng bộ (synchronuos RPC)?**

***RPC dị bộ (Asynchronous RPC):***

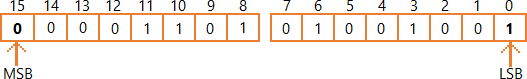
◦ Ý tưởng: client gửi tới server lời gọi thủ tục và chờ bản tin chấp nhận từ phía server. Phía server sẽ gửi lại bản tin chấp nhận về cho client thông báo đã nhận được yêu cầu và bắt đầu thực hiện yêu cầu RPC đó. Lúc này client sẽ tiếp tục thực hiện công việc của mình mà không phải ở trạng thái khóa để chờ kết quả từ server như mô hình RPC truyền thống.

***RPC đồng bộ (Synchronous RPC):***

◦ Ý tưởng: Client khi gửi tới server một yêu cầu thì nó sẽ ở trạng thái khóa, không thực hiện bất kỳ công việc nào khác cho đến khi nhận được trả lời từ phía server hoặc trong một khoảng thời gian tùy chọn do người dùng xác định. Cơ chế này đảm bảo rằng client sẽ nhận được phản hồi của server tương ứng với lời yêu cầu.

**Câu 17: Little endian và big endian?**

***Endian*** là cách tổ chức dữ liệu trên một nền tảng máy tính. Bộ nhớ máy tính có thể được xem như một mảng có kích thước lớn, chia làm nhiều ô, mỗi ô có kích thước 1 byte. Nếu dữ liệu của bạn có thể được gói gọn trong 1 byte, nghĩa là mỗi lần bạn chỉ xử lý 1 byte thì không có gì khác biệt. Vấn đề xảy ra khi dữ liệu của bạn vượt quá 1 byte, ví dụ như khi lưu trữ một số nguyên, khi đó 4 byte của biến này sẽ được lưu trữ như thế nào ?

***BIG ENDIAN****:*  
Đối với các nền tảng dùng big endian, Least Significant bit (LSB) luôn được lưu ở ô nhớ có địa chỉ lớn nhất còn Most Significant Bit (MSB) được lưu ở ô nhớ có địa chỉ nhỏ nhất trong vùng lưu trữ của biến. Trong đó, LSB là bit có trọng số nhỏ nhất, nằm ở ngoài cùng bên phải; MSB là bit có trọng số lớn nhất, nằm ở ngoài cùng bên trái của một biến.  
  
Lưu trữ số nguyên 7411 vào bộ nhớ máy tính trên nền tảng big endian. Ta cũng xét ví dụ sau:  
Số 7411 được viết dưới dạng nhị phân như sau:   
**1 1100 1111 0011**Minh họa việc lưu số 7411 vào bộ nhớ máy tính như sau:



***LITTLE ENDIAN****:*

Ngược lại với các nền tảng sử dụng big endian, ở các nền tảng sử dụng little endian, LSB luôn được lưu ở ô nhớ có địa chỉ nhỏ nhất còn MSB được lưu ở ô nhớ có địa chỉ lớn nhất trong vùng lưu trữ của biến.  
Xét lại ví dụ trên, việc lưu trữ số nguyên 7411 vào bộ nhớ máy tính được minh họa như sau:

Ưu điểm của Big Endian và Little Endian:

* + Big endian và little endian đều do con người sáng tạo nên và đều không gây ảnh hưởng đến tốc độ xử lý của CPU. Do đó sẽ không có câu trả lời hợp lý cho câu hỏi: cái nào lợi thế hơn cái nào?
  + Với big endian, do dữ liệu được ghi từ trái qua phải, bạn sẽ dễ dàng kiểm tra một số là số dương hay số âm do bit dấu được lưu trữ ở bit đầu tiên của biến. Ta không cần biết số đó chiếm bao nhiêu byte, cũng như không cần nhảy qua byte nào để xác định byte lưu trữ dấu của số đó. Các số cũng được lưu trữ đúng như trình tự mà nó được hiển thị, do đó sẽ hiệu quả trong phép toán đổi từ nhị phân sang thập phân.
  + Với little endian, một biến có thể được đọc với độ dài bất kì mà không cần phải thay đổi memory address. Do đó sẽ thuận tiện hơn trong các phép toán ép kiểu, chuyển kiểu, dễ dàng viết các phép toán phức tạp …

Chuyển đổi giữa Big Endian và Little Endian:

* + Điều này rất quan trọng trong lập trình đa nền tảng, vì các chương trình được viết ở nền tảng sử dụng big endian thì không thể sử dụng được các nền tảng khác sử dụng little endian. Muốn sử dụng được trước hết ta phải đồng bộ hai nền tảng bằng cách xem xét và đảo lại vị trí các byte cho phù hợp với hệ thống mới.

**18. Tại sao phải mật mã hóa? tính mật (confidentiality), tính toàn vẹn (intergrity), và tính khả dụng (availability) trong lý thuyết an ninh mạng, các loại mật mã?**

***Tại sao phải mật mã hóa?***  
Mã hóa sẽ mang lại tính an toàn cao hơn cho thông tin, đặc biệt là trong thời đại Internet ngày nay, khi mà thông tin phải đi qua nhiều trạm trung chuyển trước khi đến được đích. Nếu không có mã hóa, khả năng thông tin mật của chúng ta sẽ bị ai đó xem trộm trong quá trình truyền tải rồi lợi dụng để làm việc xấu là hoàn toàn hiện hữu.

***Tính mật***  
Chỉ người dùng có thẩm quyền mới được truy cập thông tin.  
- Bí mật về nội dung thông tin  
- Bí mật về sự tồn tại thông tin  
Cơ chế: Quản lý truy xuất, mật mã hóa

***Tính toàn vẹn***  
Thông tin không bị mất đi hoặc thay đổi ngoài ý muốn. Thông tin chỉ có thể được sửa đổi bởi người dùng có thẩm quyền.  
- Toàn vẹn về nội dung  
- Toàn vẹn về nguồn gốc  
Cơ chế: Hàm băm, chữ ký số, quản lý truy xuất, các giao thức xác thực.

***Tín khả dụng***  
Đảm bảo hệ thống phải hoạt động liên tục. Luôn sẵn sàn phục vụ các truy cập hợp lệ bất cứ lúc nào.  
Là khía cạnh quan trọng để đảm bảo sự tin cậy của user vào hệ thống. Là đặc trưng cơ bản nhất của hệ thống thông tin.

***Các loại mật mã***  
Hiện nay có 3 loại mật mã phổ biến  
+Mã hóa đối xứng (symmetric key encryption)  
Cách mã hóa này sử dụng chung một “key” cho cả việc mã hóa và giải mã.  
VD: Cổ điển: Ceasar, Monoalphabetic, Playfair, Hill Cipher, …  
Hiện đại (Dạng block): DES, AES, TEA  
  
+Mã hóa bất đối xứng (public key encryption)  
Phương pháp này sử dụng đến hai “key” sử dụng cho Mã hóa và Giải mã, mỗi công việc sử dụng một “key”. Người ta gọi “key” dùng để mã hóa là “public key”, dùng để giải mã là “private key”. “Public key” có thể chi sẽ cho tất cả mọi người biết để họ mã hóa. Tuy nhiên chỉ người nào có được “private key” mới có thể giải mã được.  
VD: RSA

+Mã hóa một chiều (hash)  
Phương pháp này sử dụng như đúng tên gọi của nó, chỉ có thể mã hóa nhưng không thể giải mã. Ứng dụng trong việc xác thực, đảm bảo tính toàn vẹn của dữ liệu.  
VD: SHA, MD5

**19. Các vấn đề về định thời, khái niệm “clock drift rate”, và “skew”?**

***+ “clock drift rate”*** là một hệ số thể hiện độ sai lệch của đồng hồ. Sau một khoảng thời gian, đồng hồ sẽ bị sai lệch so với một đồng hồ được đặt làm chuẩn khi cả hai bắt đầu chạy cùng lúc. Ví dụ đồng hồ chạy bằng xung thạch anh sẽ có clock drift rate là 10-6 giây/giây. Tức là cứ mỗi 1,000,000 giây (11.6 ngày) đồng hồ sẽ chạy lệch 1 giây so với đồng hồ chuẩn.

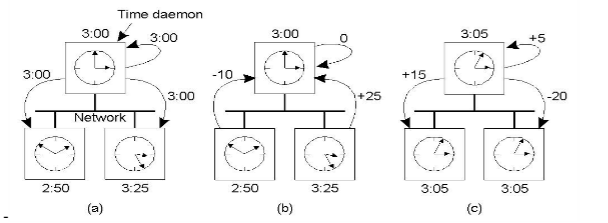
***+ “skew” hay “clock skew”*** là sự sai lệch đồng hồ tức thì. Hiện tượng này xảy ra khi so sánh thời gian của hai đồng hồ. Thực tế trong một hệ thống phân tán, mỗi đồng hồ đều chạy dựa vào xung thạch anh, tuy nhiên giữa chúng lại không cùng tần số giao động do đó về dài các đồng hồ sẽ chạy lệch nhau.

**20. Giải thuật Cristian và Giải thuật Berkeley?**  
***+ Cristian:***  
- Tổng quan:  
Trong hệ thống phân tán có một máy Server chứa đồng hồ. Tất cả các Process sẽ đồng bộ đồng hồ với Server này. Cứ mỗi một khoảng thời gian nhất định, các Process sẽ gửi một thông điệp đến máy chủ để hỏi thời gian hiện tại. Máy chủ sẽ gửi lại thời gian cho Process.  
  
- Quy trình giải thuật:  
+ Process gửi một yêu cầu đến Server  
+ Server gửi lại cho process thời gian T (thời gian hiện tại của Server)  
+ Process đặt lại đồng hồ của mình là T + Tround/2 với độ chính xác +/- (Tround/2-min)

- Giải thích  
Tround là thời gian từ lúc Process gửi thông điệp đến khi nhận thông điệp từ server. Vậy thời gian T được đặt là T + Tround/2 (Tround/2 là thời gian ước tính trung bình), tuy nhiên thực tế không phải lúc nào P gửi thì S sẽ nhận được ngay mà tồn tại một khoảng thời gian min là thời gian trao đổi thông điệp (Là khoảng thời gian từ lúc P gửi đến khi S nhận cũng như S gửi đến khi P nhận). Lúc này thời gian t sẽ nằm trong khoảng [t+min, t+Tround-min]. Do đó độ chính xác sẽ dao động 2 bên điểm Tround/2, độ rộng của khoảng bằng Tround-2min  Tround/2-min => Độ chính xác: +/- (Tround/2-min).

Ví dụ:  
Cho T0=5:08:15.100 là mốc P gửi lên S  
T1=5:08:15.900 là mốc P nhận T từ S  
Ts=5:09:25.300 là thời gian S gửi về  
Tmin= 200msec

Tround = T1 – T0 = 800  
T = Tround/2 = Ts + 400 msec = 5:09:25.700  
Accuracy: +/- 200

***+ Berkeley***  
- Tổng quan:  
Trong hệ thống phân tán có một máy Master và nhiều Slave. Master sẽ cho biết thời gian hiện tại của mình đến các Slave. Các Slave sẽ gửi lại khoảng thời gian bù (offset) của mình cho Master biết. Master tính trung bình thời gian và gửi lại khoảng thời gian bù cho các Slave và chính mình để tất cả cùng điều chỉnh đồng hồ.  
  
-Quy trình giải thuật:  
+ Master gửi đến các Slave thời gian của mình  
+ Các Slave so sánh và gửi lại thời gian bù của mình cho Master  
+ Master nhận thông tin và tính toán đồng hồ của các Slave  
+ Master loại trừ các mốc thời gian sai lệch quá lớn  
+ Master tính trung bình cộng đồng hộ của các Slave và của chính mình  
+ Master so sánh để tính thời gian bù cho từng Slave và chính mình  
+ Master gửi thời gian bù đến các Slave và cả chính mình  
+ Slave nhận offset và chỉnh đồng hồ, Master tự chỉnh đồng hồ theo  
  
Ví dụ:  


**21. Định danh, tên và địa chỉ.**

***Định danh*** (Identifier) là cách đặt danh cho các tài nguyên trong hệ thống máy tính, dùng để nhận dạng các tài nguyên khác nhau, hiệu quả cho việc tìm kiếm và lưu trữ bằng phần mềm.

***Tên*** (Name) là một chuỗi ký tự gợi nghĩa mà con người có thể đọc và hiểu được, tên có thể được phân giải thành định danh hoặc địa chỉ. Tên rất phù hợp cho việc định danh vì ràng buộc giữa tên và tài nguyên có thể được thay đổi ngoài ra tên còn mang một ý nghĩa giúp người đọc dễ hình dung.

***Địa chỉ*** (Address) là một giá trị dùng để xác định vị trí của một đối tượng.

**22. Phân biệt URI, URN và URL. Cho ví dụ về việc phân giải tên miền.**

**URI** - Uniform Resource Identifier- tạm dịch là định danh tài nguyên. Là một chuỗi kí tự được sử dụng để định danh tên, hoặc tài nguyên trên Internet. Việc định danh này cho phép tương tác với các tài nguyên trên mạng sử dụng một giao thức xác định. URI được cấu tạo từ 5 phần: scheme (Sự xếp đặt), authority (nhà cung cấp), path (đường dẫn), query (truy vấn) và fragment (phân mảnh), trong đó chỉ có scheme và path là bắt buộc phải có trong mỗi URI.

URL và URN là hai dạng của URI. Hay chính xác hơn, URL và URN đều là URI.  
**URL** – tạm dịch là định danh tài nguyên theo địa chỉ. Sử dụng giao thức http, ftp,… Nó là một dạng của URI, nó sẽ thể hiện chính xác cách ta có thể lấy tài nguyên về. URL sẽ bao gồm giao thức và domain name cũng như path tới tài nguyên nó thể hiện.  
VD: http://uis.ptithcm.edu.vn/Default.aspx?page=gioithieu

**URN** – tạm dịch là định danh tài nguyên theo tên. Sử dụng giao thức urn, dùng để định vị nguồn tài nguyên qua một tên hợp lệ vĩnh viễn và qua đó có thể xác định chúng một cách rõ ràng. Ví dụ như định danh một cuốn sách (trả về thông tin) hoặc một trang web (trả về url)  
VD: urn:ISBN:021-61918-0

**Ví dụ phân giải tên miền**:

Phân giải địa chỉ website [http://mywebsite.com](http://mywebsite.com/)  
Trường hợp website chưa được lưu trong cache

+ Browser yêu cầu OS tìm ip của website trong bộ nhớ cache (File host trong windows)  
+ Nếu không tìm thấy, OS sẽ gửi yêu cầu cho Resolver, Resolver tìm trong bộ nhớ cache. (Resolver được đặt ở ISP (Nhà cung cấp dịch vụ mạng: vnpt, fpt, viettel), Resolver sẽ biết được địa chỉ của Root Server).  
+ Không tìm thấy trong cache, Resolver gửi yêu cầu đến Root Server (Root Server sẽ chứa địa chỉ của tất cả TLD – Top-Level-Domain, có tất cả 13 Root Server được phân tán trên toàn cầu). Root Server trả về cho Resolver địa chỉ của .COM TLD Server.  
+ Resolver gửi yêu cầu đến .COM TLD Server, .COM TLD Server sẽ tìm và trả về những Name Servers có thẩm quyền quản lý mywebsite.com (ns1.mywebsite.com, ns2.mywebsite.com,…). Thực ra .COM TLD Server lưu sẵn địa chỉ ip của các Name Servers. Vì thực tế ns1,ns2 là subdomain, không thể tìm được subdomain khi chưa biết ip của domain. Name Servers cung cấp các truy vấn phân giải tên miền cho website, mail. Có nhiều Name Servers cùng quản lý 1 tên miền, dự trù cho trường hợp Name Server bất kỳ bị sập.  
+ Resolver gửi yêu cầu đến từng Name Server và nhận được địa chỉ ip của website từ một Name Server bất kỳ.   
+ Sau khi có được địa chỉ ip, Resolver trả về cho OS, OS nhanh chóng lưu vào bộ nhớ cache đồng thời gửi cho Browser địa chỉ ip. Browser kết nối đến website.  
+ Tất cả chỉ xảy ra trong vòng một vài mili giây.