Tài liệu Wiki về TCP, Client-Sever

TCP

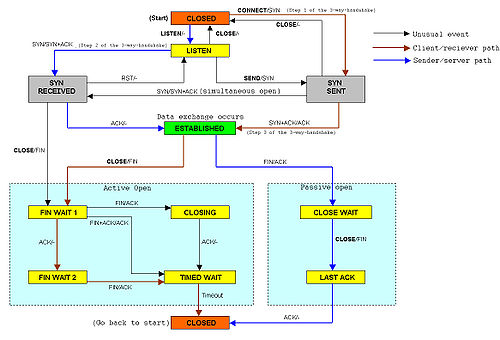
**TCP** (*Transmission Control Protocol* - "Giao thức điều khiển truyền vận") là một trong các giao thức cốt lõi của [bộ giao thức TCP/IP](https://vi.wikipedia.org/wiki/TCP/IP). Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các [gói tin](https://vi.wikipedia.org/w/index.php?title=G%C3%B3i_tin&action=edit&redlink=1). Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.

TCP hỗ trợ nhiều giao thức ứng dụng phổ biến nhất trên [Internet](https://vi.wikipedia.org/wiki/Internet) và các ứng dụng kết quả, trong đó có [WWW](https://vi.wikipedia.org/wiki/World_Wide_Web), [thư điện tử](https://vi.wikipedia.org/wiki/Th%C6%B0_%C4%91i%E1%BB%87n_t%E1%BB%AD) và [Secure Shell](https://vi.wikipedia.org/wiki/SSH).

Trong bộ giao thức TCP/IP, TCP là tầng trung gian giữa [giao thức IP](https://vi.wikipedia.org/wiki/IP) bên dưới và một [ứng dụng](https://vi.wikipedia.org/w/index.php?title=%E1%BB%A8ng_d%E1%BB%A5ng_ph%E1%BA%A7n_m%E1%BB%81m&action=edit&redlink=1) bên trên. Các ứng dụng thường cần các kết nối đáng tin cậy kiểu [đường ống](https://vi.wikipedia.org/w/index.php?title=%C4%90%C6%B0%E1%BB%9Dng_%E1%BB%91ng&action=edit&redlink=1) để liên lạc với nhau, trong khi đó, giao thức IP không cung cấp những dòng kiểu đó, mà chỉ cung cấp dịch vụ chuyển gói tin [không đáng tin cậy](https://vi.wikipedia.org/w/index.php?title=Kh%C3%B4ng_%C4%91%C3%A1ng_tin_c%E1%BA%ADy&action=edit&redlink=1). TCP làm nhiệm vụ của [tầng giao vận](https://vi.wikipedia.org/wiki/T%E1%BA%A7ng_giao_v%E1%BA%ADn) trong [mô hình OSI](https://vi.wikipedia.org/wiki/M%C3%B4_h%C3%ACnh_OSI) đơn giản của các [mạng máy tính](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_m%C3%A1y_t%C3%ADnh).

Các ứng dụng gửi các dòng gồm các [byte](https://vi.wikipedia.org/wiki/Byte) 8-bit tới TCP để chuyển qua mạng. TCP phân chia dòng byte này thành các [đoạn](https://vi.wikipedia.org/w/index.php?title=%C4%90%C6%A1n_v%E1%BB%8B_d%E1%BB%AF_li%E1%BB%87u_giao_th%E1%BB%A9c&action=edit&redlink=1) (*segment*) có kích thước thích hợp (thường được quyết định dựa theo kích thước của [đơn vị truyền dẫn tối đa](https://vi.wikipedia.org/w/index.php?title=MTU_(m%E1%BA%A1ng_m%C3%A1y_t%C3%ADnh)&action=edit&redlink=1) (MTU) của [tầng liên kết dữ liệu](https://vi.wikipedia.org/wiki/T%E1%BA%A7ng_li%C3%AAn_k%E1%BA%BFt_d%E1%BB%AF_li%E1%BB%87u) của mạng mà máy tính đang nằm trong đó). Sau đó, TCP chuyển các gói tin thu được tới giao thức IP để gửi nó qua một [liên mạng](https://vi.wikipedia.org/wiki/Li%C3%AAn_m%E1%BA%A1ng) tới mô đun TCP tại máy tính đích. TCP kiểm tra để đảm bảo không có gói tin nào bị thất lạc bằng cách gán cho mỗi gói tin một "số thứ tự" (*sequence number*). Số thứ tự này còn được sử dụng để đảm bảo dữ liệu được trao cho ứng dụng đích theo đúng thứ tự. Mô đun TCP tại đầu kia gửi lại "tin báo nhận" (*acknowledgement*) cho các gói tin đã nhận được thành công; một "đồng hồ" (*timer*) tại nơi gửi sẽ báo *time-out* nếu không nhận được tin báo nhận trong khoảng thời gian bằng một [round-trip time](https://vi.wikipedia.org/w/index.php?title=Round-trip_time&action=edit&redlink=1) (RTT), và dữ liệu (được coi là bị thất lạc) sẽ được gửi lại. TCP sử dụng [checksum](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_t%E1%BB%95ng_ki%E1%BB%83m) (*giá trị kiểm tra*) để xem có byte nào bị hỏng trong quá trình truyền hay không; giá trị này được tính toán cho mỗi khối dữ liệu tại nơi gửi trước khi nó được gửi, và được kiểm tra tại nơi nhận.

Hoạt động của giao thức[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=1" \o "Sửa đổi phần \“Hoạt động của giao thức\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=1)]

[](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:TCP_state_diagram.jpg)

Sơ đồ trạng thái của TCP - phiên bản đơn giản hóa

Không như giao thức [UDP](https://vi.wikipedia.org/wiki/UDP) - giao thức có thể lập tức gửi gói tin mà không cần thiết lập kết nối, TCP đòi hỏi thiết lập kết nối trước khi bắt đầu gửi dữ liệu và kết thúc kết nối khi việc gửi dữ liệu hoàn tất. Cụ thể, các kết nối TCP có ba pha:

1. Thiết lập kết nối
2. Truyền dữ liệu
3. Kết thúc kết nối

Trước khi miêu tả các pha này, ta cần lưu ý các trạng thái khác nhau của một socket:

1. LISTEN
2. SYN-SENT
3. SYN-RECEIVED
4. ESTABLISHED
5. FIN-WAIT
6. CLOSE-WAIT
7. CLOSING
8. LAST-ACK
9. TIME-WAIT
10. CLOSER

đang đợi yêu cầu kết nối từ một TCP và cổng bất kỳ ở xa (trạng thái này thường do các TCP server đặt)

**SYN-SENT**

đang đợi TCP ở xa gửi một gói tin TCP với các cờ SYN và ACK được bật (trạng thái này thường do các TCP client đặt)

**SYN-RECEIVED**

đang đợi TCP ở xa gửi lại một tin báo nhận sau khi đã gửi cho TCP ở xa đó một tin báo nhận kết nối (*connection acknowledgment*) (thường do TCP server đặt)

**ESTABLISHED**

cổng đã sẵn sàng nhận/gửi dữ liệu với TCP ở xa (đặt bởi TCP client và server)

**TIME-WAIT**

đang đợi qua đủ thời gian để chắc chắn là TCP ở xa đã nhận được tin báo nhận về yêu cầu kết thúc kết nối của nó. Theo [RFC 793](http://tools.ietf.org/html/793), một kết nối có thể ở tại trạng thái TIME-WAIT trong vòng tối đa 4 phút.

**Thiết lập kết nối**[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=2" \o "Sửa đổi phần \“Thiết lập kết nối\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=2)]

Để thiết lập một kết nối, TCP sử dụng một quy trình [bắt tay](https://vi.wikipedia.org/w/index.php?title=B%E1%BA%AFt_tay_(tin_h%E1%BB%8Dc)&action=edit&redlink=1) 3 bước (*3-way handshake*) Trước khi client thử kết nối với một server, server phải đăng ký một cổng và mở cổng đó cho các kết nối: đây được gọi là mở bị động. Một khi mở bị động đã được thiết lập thì một client có thể bắt đầu mở chủ động. Để thiết lập một kết nối, quy trình bắt tay 3 bước xảy ra như sau:

1. Client yêu cầu mở cổng dịch vụ bằng cách gửi gói tin SYN (gói tin TCP) tới server, trong gói tin này, tham số **sequence number** được gán cho một giá trị ngẫu nhiên **X**.
2. Server hồi đáp bằng cách gửi lại phía client bản tin SYN-ACK, trong gói tin này, tham số **acknowledgment number** được gán giá trị bằng X + 1, tham số **sequence number**được gán ngẫu nhiên một giá trị **Y**
3. Để hoàn tất quá trình **bắt tay ba bước**, client tiếp tục gửi tới server bản tin ACK, trong bản tin này, tham số **sequence number** được gán cho giá trị bằng X + 1 còn tham số **acknowledgment number** được gán giá trị bằng Y + 1

Tại thời điểm này, cả client và server đều được xác nhận rằng, một kết nối đã được thiết lập.

**Truyền dữ liệu**[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=3" \o "Sửa đổi phần \“Truyền dữ liệu\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=3)]

Một số đặc điểm cơ bản của TCP để phân biệt với [UDP](https://vi.wikipedia.org/wiki/UDP):

* Truyền dữ liệu không lỗi (do có cơ chế sửa lỗi/truyền lại)
* Truyền các gói dữ liệu theo đúng thứ tự
* Truyền lại các gói dữ liệu mất trên đường truyền
* Loại bỏ các gói dữ liệu trùng lặp
* Cơ chế hạn chế tắc nghẽn đường truyền

Ở hai bước đầu tiên trong ba bước bắt tay, hai máy tính trao đổi một số thứ tự gói ban đầu (*Initial Sequence Number* -ISN). Số này có thể chọn một cách ngẫu nhiên. Số thứ tự này được dùng để đánh dấu các khối dữ liệu gửi từ mỗi máy tính. Sau mỗi byte được truyền đi, số này lại được tăng lên. Nhờ vậy ta có thể sắp xếp lại chúng khi tới máy tính kia bất kể các gói tới nơi theo thứ tự thế nào.

Trên lý thuyết, mỗi byte gửi đi đều có một số thứ tự và khi nhận được thì máy tính nhận gửi lại tin báo nhận (ACK). Trong thực tế thì chỉ có byte dữ liệu đầu tiên được gán số thứ tự trong trường số thứ tự của gói tin và bên nhận sẽ gửi tin báo nhận bằng cách gửi số thứ tự của byte đang chờ.

Ví dụ: Máy tính A gửi 4 byte với số thứ tự ban đầu là 100 (theo lý thuyết thì 4 byte sẽ có thứ tự là 100, 101, 102, 103) thì bên nhận sẽ gửi tin báo nhận có nội dung là 104 vì đó là thứ tự của byte tiếp theo nó cần. Bằng cách gửi tin báo nhận là 104, bên nhận đã ngầm thông báo rằng nó đã nhận được các byte 100, 101, 102 và 103. Trong trường hợp 2 byte cuối bị lỗi thì bên nhận sẽ gửi tin báo nhận với nội dung là 102 vì 2 byte 100 và 101 đã được nhận thành công.

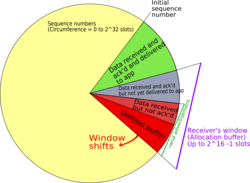
Giả sử ta có 10.000 byte được gửi đi trong 10 gói tin 1.000 byte và có 1 gói tin bị mất trên đường truyền. Nếu gói bị mất là gói đầu tiên thì bên gửi sẽ phải gửi lại toàn bộ 10 gói vì không có cách nào để bên nhận thông báo nó đã nhận được 9 gói kia. Vấn đề này được giải quyết trong giao thức [SCTP](https://vi.wikipedia.org/wiki/SCTP) (*Stream Control Transmission Protocol* - "Giao thức điều khiển truyền vận dòng") với việc bổ sung báo nhận chọn lọc.

Số thứ tự và tin báo nhận giải quyết được các vấn đề về lặp gói tin, truyền lại những gói bị hỏng/mất và các gói tin đến sai thứ tự. Để phục vụ mục đích kiểm tra, các gói tin có trường [giá trị tổng kiểm](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_t%E1%BB%95ng_ki%E1%BB%83m) (*checksum* - Xem thêm phần [#Cấu trúc gói](https://vi.wikipedia.org/wiki/TCP#C.E1.BA.A5u_tr.C3.BAc_g.C3.B3i)).

Với trình độ hiện tại, kỹ thuật [kiểm tra tổng](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_t%E1%BB%95ng_ki%E1%BB%83m) trong TCP không đủ mạnh. Các [tầng liên kết dữ liệu](https://vi.wikipedia.org/wiki/T%E1%BA%A7ng_li%C3%AAn_k%E1%BA%BFt_d%E1%BB%AF_li%E1%BB%87u) với xác suất lỗi bit cao có thể cần được bổ sung các khả năng phát hiện lỗi tốt hơn. Nếu như TCP được thiết kế vào thời điểm hiện tại, nhiều khả năng nó sẽ bao gồm trường [kiểm tra độ dư tuần hoàn](https://vi.wikipedia.org/wiki/CRC) (*cyclic redundancy check* - CRC) với độ dài 32 bit. Điểm yếu này một phần được bù đắp bằng CRC hay những kỹ thuật khác tại [tầng thứ 2](https://vi.wikipedia.org/wiki/T%E1%BA%A7ng_li%C3%AAn_k%E1%BA%BFt_d%E1%BB%AF_li%E1%BB%87u) (trong [mô hình 7 lớp OSI](https://vi.wikipedia.org/wiki/M%C3%B4_h%C3%ACnh_OSI)) ở bên dưới cả TCP và IP như trong các giao thức điểm-điểm ([PPP](https://vi.wikipedia.org/wiki/PPP_(giao_th%E1%BB%A9c))) hoặc [Ethernet](https://vi.wikipedia.org/wiki/Ethernet). Tuy nhiên điều này cũng không có nghĩa là trường kiểm tra tổng của TCP là không cần thiết: thống kê cho thấy các sai sót do cả phần cứng và phần mềm gây ra giữa các điểm áp dụng kỹ thuật kiểm tra CRC là khá phổ biến và kỹ thuật kiểm tra tổng có khả năng phát hiện phần lớn các lỗi (đơn giản) này.

Điểm cuối cùng là khả năng hạn chế tắc nghẽn.

[Tin báo nhận](https://vi.wikipedia.org/w/index.php?title=Tin_b%C3%A1o_nh%E1%BA%ADn&action=edit&redlink=1) (hoặc không có tin báo nhận) là tín hiệu về tình trạng đường truyền giữa 2 máy tính. Từ đó, hai bên có thể thay đổi tốc độ truyền nhận dữ liệu phù hợp với điều kiện. Vấn đề này thường được đề cập là [điều khiển lưu lượng](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%81u_khi%E1%BB%83n_l%C6%B0u_l%C6%B0%E1%BB%A3ng), [kiểm soát tắc nghẽn](https://vi.wikipedia.org/w/index.php?title=Ki%E1%BB%83m_so%C3%A1t_t%E1%BA%AFc_ngh%E1%BA%BDn&action=edit&redlink=1). TCP sử dụng một số cơ chế nhằm đạt được hiệu suất cao và ngăn ngừa khả năng nghẽn mạng. Các cơ chế này bao gồm: [cửa sổ trượt](https://vi.wikipedia.org/w/index.php?title=C%C6%A1_ch%E1%BA%BF_c%E1%BB%ADa_s%E1%BB%95_tr%C6%B0%E1%BB%A3t&action=edit&redlink=1) (*sliding window*), [thuật toán slow-start](https://vi.wikipedia.org/w/index.php?title=Thu%E1%BA%ADt_to%C3%A1n_slow-start&action=edit&redlink=1), [thuật toán tránh nghẽn mạng](https://vi.wikipedia.org/w/index.php?title=Thu%E1%BA%ADt_to%C3%A1n_tr%C3%A1nh_ngh%E1%BA%BDn_m%E1%BA%A1ng&action=edit&redlink=1) (*congestion avoidance*), thuật toán truyền lại và phục hồi nhanh,... Hiện nay, vấn đề cải tiến TCP trong môi truyền truyền dẫn tốc độ cao đang là một hướng nghiên cứu được quan tâm.

[](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Tcp.png)

**Kích thước cửa sổ TCP**[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=4" \o "Sửa đổi phần \“Kích thước cửa sổ TCP\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=4)]

Chuỗi số thứ tự gói và cửa sổ trong TCP hoạt động giống như một cái đồng hồ. Kích thước của cửa sổ (đo bằng byte) được thiết lập bởi khả năng tiếp nhận của máy tính nhận. Cửa sổ này được dịch đi mỗi khi máy tính nhận nhận được dữ liệu và gửi tin báo nhận. Khi chuỗi thứ tự tăng đến tối đa thì lại quay lại về 0.

Kích thước của cửa sổ là chiều dài (byte) của khối dữ liệu có thể lưu trong bộ đệm của bên nhận. Bên gửi chỉ có thể gửi tối đa lượng thông tin chứa trong cửa sổ này trước khi nhận được tin báo nhận.

**Dãn kích thước cửa sổ**[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=5" \o "Sửa đổi phần \“Dãn kích thước cửa sổ\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=5)]

Để tận dụng khả năng truyền dẫn của mạng thì cửa sổ dùng trong TCP cần được tăng lên. Trường điều khiển kích thước cửa sổ của gói TCP có độ dài là 2 byte và do đó kích thước tối đa của cửa sổ là 65.535 byte.

Do trường điều khiển không thể thay đổi nên người ta sử dụng một hệ số dãn nào đó. Hệ số này được định nghĩa trong tài liệu [RFC](https://vi.wikipedia.org/wiki/RFC)1323 có thể sử dụng để tăng kích thước tối đa của cửa sổ từ 65.535 byte lên tới 1 gigabyte. Tăng kích thước cửa sổ lớn hơn nữa cũng cần thiết trong [TCP Tuning](https://vi.wikipedia.org/w/index.php?title=TCP_Tuning&action=edit&redlink=1).

Việc tăng kích thước cửa sổ chỉ được dùng trong giao thức bắt tay 3 pha. Giá trị của trường co giãn cửa sổ thể hiện số bit cần được dịch trái đối với trường kích thước cửa sổ. Hệ số dãn có thể thay đổi từ 0 (không dãn) tới 14 (dãn tối đa).

**Kết thúc kết nối**[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=6" \o "Sửa đổi phần \“Kết thúc kết nối\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=6)]

Để kết thúc kết nối hai bên sử dụng quá trình bắt tay 4 bước và chiều của kết nối kết thúc độc lập với nhau. Khi một bên muốn kết thúc, nó gửi đi một gói tin FIN và bên kia gửi lại tin báo nhận ACK. Vì vậy, một quá trình kết thúc tiêu biểu sẽ có 2 cặp gói tin trao đổi.

Một kết nối có thể tồn tại ở dạng "nửa mở": một bên đã kết thúc gửi dữ liệu nên chỉ nhận thông tin, bên kia vẫn tiếp tục gửi.

Các cổng TCP[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=7" \o "Sửa đổi phần \“Các cổng TCP\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=7)]

TCP sử dụng khái niệm [số hiệu cổng](https://vi.wikipedia.org/w/index.php?title=C%E1%BB%95ng_(tin_h%E1%BB%8Dc)&action=edit&redlink=1) (*port number*) để định danh các ứng dụng gửi và nhận dữ liệu. Mỗi đầu của một kết nối TCP có một số hiệu cổng (là số không dấu 16-bit) được gán cho ứng dụng đang nhận hoặc gửi dữ liệu. Các cổng được phân thành ba loại cơ bản: nổi tiếng, được đăng ký và động/cá nhân. Các cổng nổi tiếng đã được gán bởi tổ chức [Internet Assigned Numbers Authority](https://vi.wikipedia.org/w/index.php?title=Internet_Assigned_Numbers_Authority&action=edit&redlink=1) (IANA) và thường được sử dụng bởi các tiến trình mức hệ thống hoặc các tiến trình của [root](https://vi.wikipedia.org/w/index.php?title=Root&action=edit&redlink=1). Ví dụ: [FTP](https://vi.wikipedia.org/wiki/FTP) (21), [TELNET](https://vi.wikipedia.org/wiki/Telnet) (23), [SMTP](https://vi.wikipedia.org/wiki/SMTP) (25) và [HTTP](https://vi.wikipedia.org/wiki/Hypertext_Transfer_Protocol) (80). Các cổng được đăng ký thường được sử dụng bởi các ứng dụng người dùng đầu cuối (*end user application*) với vai trò các cổng phát tạm thời (khi dùng xong thì hủy đăng ký) khi kết nối với server, nhưng chúng cũng có thể định danh các dịch vụ có tên đã được đăng ký bởi một bên thứ ba. Các cổng động/cá nhân cũng có thể được sử dụng bởi các ứng dụng người dùng đầu cuối, nhưng không thông dụng bằng. Các cổng động/cá nhân không có ý nghĩa gì nếu không đặt trong một kết nối TCP. Có 65535 cổng được chính thức thừa nhận.

Sự phát triển của TCP[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=8" \o "Sửa đổi phần \“Sự phát triển của TCP\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=8)]

TCP là một giao thức phức tạp và vẫn còn tiếp tục được phát triển. Tuy nhiên, mặc dù có nhiều cải tiến đã được áp dụng và đề xuất nhưng các hoạt động cơ bản của giao thức vẫn giữ nguyên như mô tả ban đầu trong tài liệu [RFC 793](http://tools.ietf.org/html/793) ban hành năm 1981. Tài liệu [RFC 1122](http://tools.ietf.org/html/1122) - Các yêu cầu của máy mạng Internet - đưa ra một số yêu cầu khi thực hiện TCP. [RFC 2581](http://tools.ietf.org/html/2581) - Điều khiển tránh nghẽn mạng, một trong những tài liệu quan trọng trong bộ RFC trong những năm gần đây - mô tả thuật toán dùng để giảm khả năng tắc nghẽn mạng. Năm 2001, [RFC 3168](http://tools.ietf.org/html/3168) mô tả một cơ chế báo hiệu chống nghẽn mạng có tên là [Thông báo nghẽn mạng](https://vi.wikipedia.org/w/index.php?title=Th%C3%B4ng_b%C3%A1o_ngh%E1%BA%BDn_m%E1%BA%A1ng&action=edit&redlink=1) (*Explicit Congestion Notification*). Vào thời điểm đầu [thế kỷ 21](https://vi.wikipedia.org/wiki/Th%E1%BA%BF_k%E1%BB%B7_21), khoảng 95% gói tin trên Internet là TCP. Các ứng dụng tiêu biểu sử dụng TCP là [HTTP](https://vi.wikipedia.org/wiki/Hypertext_Transfer_Protocol)/[HTTPS](https://vi.wikipedia.org/wiki/HTTPS) ([World Wide Web](https://vi.wikipedia.org/wiki/World_Wide_Web)), [SMTP](https://vi.wikipedia.org/wiki/SMTP)/[POP3](https://vi.wikipedia.org/wiki/POP3)/[IMAP](https://vi.wikipedia.org/wiki/IMAP) ([e-mail](https://vi.wikipedia.org/wiki/Th%C6%B0_%C4%91i%E1%BB%87n_t%E1%BB%AD)) và [FTP](https://vi.wikipedia.org/wiki/FTP) (truyền file). Sự phổ biến của TCP chứng tỏ rằng nó đã được thiết kế rất tốt.

Cơ chế điều khiển tránh tắc nghẽn của TCP ban đầu là [TCP Reno](https://vi.wikipedia.org/w/index.php?title=TCP_Reno&action=edit&redlink=1) và gần đây đã có một số thuật toán khác được đề xuất:

* [High Speed TCP](https://vi.wikipedia.org/w/index.php?title=High_Speed_TCP&action=edit&redlink=1) của [Sally Floyd](https://vi.wikipedia.org/w/index.php?title=Sally_Floyd&action=edit&redlink=1) trong tài liệu [RFC 3649](http://tools.ietf.org/html/3649)
* [TCP Vegas](https://vi.wikipedia.org/w/index.php?title=TCP_Vegas&action=edit&redlink=1) của Brakmo và Peterson (làm việc tại [Đại học Arizona](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BA%A1i_h%E1%BB%8Dc_Arizona&action=edit&redlink=1))
* [TCP Westwood](https://vi.wikipedia.org/w/index.php?title=TCP_Westwood&action=edit&redlink=1) của [Đại học California tại Los Angeles](https://vi.wikipedia.org/wiki/%C4%90%E1%BA%A1i_h%E1%BB%8Dc_California_t%E1%BA%A1i_Los_Angeles)
* [BIC TCP](https://vi.wikipedia.org/w/index.php?title=BIC_TCP&action=edit&redlink=1) của Injong Rhee (làm việc tại [Đại học North Carolina](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BA%A1i_h%E1%BB%8Dc_North_Carolina&action=edit&redlink=1))
* [H-TCP](http://www.hamilton.ie/net/htcp/) của [Viện Hamilton](https://vi.wikipedia.org/w/index.php?title=Vi%E1%BB%87n_Hamilton&action=edit&redlink=1)
* [Fast TCP](https://vi.wikipedia.org/w/index.php?title=Fast_TCP&action=edit&redlink=1) (*Fast Active queue management Scalable Transmission Control Protocol*) của [Caltech](https://vi.wikipedia.org/wiki/Caltech).
* [TCP Hybla](http://seclists.org/lists/linux-kernel/2005/Feb/4954.html/) của [Đại học Bologna](https://vi.wikipedia.org/wiki/%C4%90%E1%BA%A1i_h%E1%BB%8Dc_Bologna)

Bên cạnh đó cũng có rất nhiều nghiên cứu so sánh sự công bằng và hiệu suất của TCP khi sử dụng các thuật toán tránh tắc nghẽn khác nhau.

TCP trên mạng không dây[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=9" \o "Sửa đổi phần \“TCP trên mạng không dây\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=9)]

TCP cũng được sử dụng cho mạng không dây. Ở đây trường hợp mất gói tin cũng được xem là nghẽn mạng và kích thước cửa sổ do đó cũng sẽ được giảm xuống. Tuy nhiên trong nhiều trường hợp đối với các mạng không dây thì việc mất các gói tin thường xảy ra một cách ngẫu nhiên do ảnh hưởng của fading, chuyển giao giữa các cell... và chúng ta không thể xem đây là nghẽn mạng. Do đó, việc giảm kích thước cửa sổ không đúng sẽ làm cho hiệu quả sử dụng đường truyền giảm một cách đáng kể. Nhiều nghiên cứu đã tập trung để giải quyết vấn đề này. Các giải pháp được đề ra có thể phân loại thành các nhóm: giải pháp đầu cuối (liên quan tới việc thay đổi tại client/server), giải pháp tại tầng liên kết dữ liệu (chẳng hạn giao thức [RLP](https://vi.wikipedia.org/w/index.php?title=Radio_Link_Protocol&action=edit&redlink=1) trong chuẩn [CDMA2000](https://vi.wikipedia.org/wiki/CDMA2000)) và giải pháp dựa trên proxy (thay đổi trong mạng mà không cần thay đổi các thiết bị đầu cuối).

Gỡ rối trong TCP[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=10" \o "Sửa đổi phần \“Gỡ rối trong TCP\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=10)]

Các phần mềm đọc gói (*packet sniffer*) TCP có thể sử dụng để gỡ rối/theo dõi bằng cách đọc tất cả các gói TCP được truyền trong mạng. Ví dụ: Wireshark(trên Windows và Linux), tcpdump(trên Linux)...

Các lựa chọn khác ngoài TCP[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=11" \o "Sửa đổi phần \“Các lựa chọn khác ngoài TCP\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=11)]

Đối với một số ứng dụng thì TCP không thích hợp. Vấn đề lớn nhất là phía nhận không thể tiếp nhận các gói tin đến sau một gói bị lỗi trước khi chính gói bị lỗi được truyền lại. Điều này khiến TCP không thích hợp cho các [ứng dụng thời gian thực](https://vi.wikipedia.org/w/index.php?title=%E1%BB%A8ng_d%E1%BB%A5ng_th%E1%BB%9Di_gian_th%E1%BB%B1c&action=edit&redlink=1) (*real-time*) chẳng hạn như đa phương tiện trực tuyến, [trò chơi trực tuyến](https://vi.wikipedia.org/wiki/Tr%C3%B2_ch%C6%A1i_tr%E1%BB%B1c_tuy%E1%BA%BFn) và [thoại trên nền IP](https://vi.wikipedia.org/w/index.php?title=Tho%E1%BA%A1i_tr%C3%AAn_n%E1%BB%81n_IP&action=edit&redlink=1) (VoIP) bởi vì các ứng dụng này cần các gói tin kịp thời hơn là nhận đủ các gói tin theo đúng thứ tự.

Ngoài ra sự phức tạp của TCP cũng gây ra vấn đề với các [hệ thống nhúng](https://vi.wikipedia.org/wiki/H%E1%BB%87_th%E1%BB%91ng_nh%C3%BAng) (*embedded system*). Ví dụ tiêu biểu là netbooting sử dụng giao thức [TFTP](https://vi.wikipedia.org/w/index.php?title=TFTP&action=edit&redlink=1). Cuối cùng, độ phức tạp của TCP cũng gây khó khăn cho một số vấn đề khác như truyền thông tin giữa 2 máy tính nằm sau hệ thống chuyển đổi địa chỉ (NAT).

Thông thường, khi TCP không thích hợp thì UDP được sử dụng. UDP cung cấp một số tính năng giống TCP như đa công và kiểm tra tổng nhưng nó không đảm bảo việc truyền lại gói tin lỗi hay thứ tự các gói tin. Vì thế, người phát triển ứng dụng có thể áp dụng các phương thức khác ở các tầng trên để giải quyết vấn đề tùy theo yêu cầu cụ thể.

Giao thức điều khiển truyền vận dòng (*Stream Control Transmission Protocol* - [SCTP](https://vi.wikipedia.org/wiki/SCTP)) cũng là một giao thức dựa trên nền IP không khác nhiều so với TCP. SCTP được phát triển sau và có cấu trúc phức tạp hơn TCP. SCTP được thiết kế để sử dụng trong điều kiện yêu cầu độ tin cậy và gần thời gian thực. Tuy nhiên SCTP chưa được sử dụng rộng rãi.

TCP cũng bộc lộ một số vấn đề khi dùng trong môi trường truyền dẫn tốc độ cao. Thuật toán tránh nghẽn mạng làm việc tốt trong môi trường không dự tính trước nhưng đối với môi trường xác định hơn chẳng hạn như ATM (*Asynchronous Transfer Mode*) thì TCP không tận dụng được khả năng của hệ thống bên dưới.

Cấu trúc gói tin[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=12" \o "Sửa đổi phần \“Cấu trúc gói tin\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=12)]

Một gói tin TCP bao gồm 2 phần

* header (có độ dài 20 bytes)
* dữ liệu

Phần header có 11 trường trong đó 10 trường bắt buộc. Trường thứ 11 là tùy chọn (trong bảng minh họa có màu nền đỏ) có tên là: options

**Header**[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=13" \o "Sửa đổi phần \“Header\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=13)]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **+** | **Bít 0 - 3** | **4 - 9** | **10 - 15** | **16 - 31** | | **0** | Source Port | | | Destination Port | | **32** | Sequence Number | | | | | **64** | Acknowledgement Number | | | | | **96** | Data Offset | Reserved | Flags | Window | | **128** | Checksum | | | Urgent Pointer | | **160** | Options (optional) | | | | | **160/192+** | Data | | | | |

**Source port**

Số hiệu của cổng tại máy tính gửi.

**Destination port**

Số hiệu của cổng tại máy tính nhận.

**Sequence number**

Trường này có 2 nhiệm vụ. Nếu cờ SYN bật thì nó là số thứ tự gói ban đầu và byte đầu tiên được gửi có số thứ tự này cộng thêm 1. Nếu không có cờ SYN thì đây là số thứ tự của byte đầu tiên.

**Acknowledgement number**

Nếu cờ ACK bật thì giá trị của trường chính là số thứ tự gói tin tiếp theo mà bên nhận cần.

**Data offset**

Trường có độ dài 4 bít quy định độ dài của phần header (tính theo đơn vị từ 32 bít). Phần header có độ dài tối thiểu là 5 từ (160 bit) và tối đa là 15 từ (480 bít).

**Reserved**

Dành cho tương lai và có giá trị là 0.

**Flags (hay Control bits)**

Bao gồm 6 cờ:

**URG**

Cờ cho trường Urgent pointer

**ACK**

Cờ cho trường Acknowledgement

**PSH**

Hàm Push

**RST**

Thiết lập lại đường truyền

**SYN**

Đồng bộ lại số thứ tự

**FIN**

Không gửi thêm số liệu

**Window**

Số byte có thể nhận bắt đầu từ giá trị của trường báo nhận (ACK)

**Checksum**

16 bít kiểm tra cho cả phần header và dữ liệu. Phương pháp sử dụng được mô tả trong [RFC 793](http://tools.ietf.org/html/793):

*16 bít của trường kiểm tra là bổ sung của tổng tất cả các từ 16 bít trong gói tin. Trong trường hợp số octet (khối 8 bít) của header và dữ liệu là lẻ thì octet cuối được bổ sung với các bít 0. Các bít này không được truyền. Khi tính tổng, giá trị của trường kiểm tra được thay thế bằng 0,*

Nói một cách khác, tất cả các từ 16 bít được cộng với nhau. Kết quả thu được sau khi đảo giá trị từng bít được điền vào trường kiểm tra. Về mặt thuật toán, quá trình này giống với [IPv4](https://vi.wikipedia.org/wiki/IPv4).

Điểm khác nhau chỉ ở chỗ dữ liệu dùng để tính tổng kiểm tra. Dưới đây là một header của IP:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **+** | **Bít 0 - 3** | **4 - 7** | **8 - 9** | **10 - 15** | **16 - 31** | | **0** | Source address | | | | | | **32** | Destination address | | | | | | **64** | Zeros | | Protocol | | TCP length | | **96** | Source Port | | | | Destination Port | | **128** | Sequence Number | | | | | | **160** | Acknowledgement Number | | | | | | **192** | Data Offset | Reserved | | Flags | Window | | **225** | Checksum | | | | Urgent Pointer | | **257** | Options (optional) | | | | | | **257/289+** | Data | | | | | |

Các địa chỉ nguồn và đích là các địa chỉ [IPv4](https://vi.wikipedia.org/wiki/IPv4). Giá trị của trường protocol là 6 (giá trị dành cho TCP, xem thêm: [Danh sách số hiệu giao thức IPv4](https://vi.wikipedia.org/w/index.php?title=Danh_s%C3%A1ch_s%E1%BB%91_hi%E1%BB%87u_giao_th%E1%BB%A9c_IPv4&action=edit&redlink=1)). Giá trị của trường TCP length field là độ dài của toàn bộ phần header và dữ liệu của gói TCP.

**Urgent pointer**

Nếu cờ URG bật thì giá trị trường này chính là số từ 16 bít mà số thứ tự gói tin (*sequence number*) cần dịch trái.

**Options**

Đây là trường tùy chọn. Nếu có thì độ dài là bội số của 32 bít.

**Dữ liệu**[[sửa](https://vi.wikipedia.org/w/index.php?title=TCP&veaction=edit&section=14" \o "Sửa đổi phần \“Dữ liệu\”) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=TCP&action=edit&section=14)]

Trường cuối cùng không thuộc về header. Giá trị của trường này là thông tin dành cho các tầng trên (trong mô hình 7 lớp OSI). Thông tin về giao thức của tầng trên không được chỉ rõ trong phần header mà phụ thuộc vào cổng được chọn.

Client-sever

**Mô hình client-server** là một mô hình nổi tiếng trong [mạng máy tính](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_m%C3%A1y_t%C3%ADnh), được áp dụng rất rộng rãi và là mô hình của mọi trang web hiện có. Ý tưởng của mô hình này là máy con (đóng vài trò là máy khách) gửi một yêu cầu (request) để máy chủ (đóng vai trò người cung ứng dịch vụ), máy chủ sẽ xử lý và trả kết quả về cho máy khách.

Một mô hình ngược lại là mô hình [master-slaver](https://vi.wikipedia.org/w/index.php?title=Master-slaver&action=edit&redlink=1), trong đó máy chủ (đóng vai trò ông chủ) sẽ gửi dữ liệu đến máy con (đóng vai trò nô lệ) bất kể máy con có cần hay không.

**Mô hình Web client/server**:

Thuật ngữ server được dùng cho những chương trình thi hành như một dịch vụ trên toàn mạng. Các chương trình server này chấp nhận tất cả các yêu cầu hợp lệ đến từ mọi nơi trên mạng, sau đó nó thi hành dịch vụ và trả kết quả về máy yêu cầu. Một chương trình được coi là client khi nó gửi các yêu cầu tới máy có chương trình server và chờ đợi câu trả lời từ server. Chương trình server và client nói chuyện với nhau bằng các thông điệp (messages) thông qua một cổng truyền thông liên tác IPC (Interprocess Communication). Để một chương trình server và một chương trình client có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn để nói chuyện, chuẩn này được gọi là giao thức. Nếu một chương trình client nào đó muốn yêu cầu lấy thông tin từ server thì nó phải tuân theo giao thức mà server đó đưa ra. Bản thân chúng ta khi cần xây dựng một mô hình client/server cụ thể thì ta cũng có thể tự tạo ra một giao thức riêng nhưng thường chúng ta chỉ làm được điều này ở tầng ứng dụng của mạng. Với sự phát triển mạng như hiện này thì có rất nhiều giao thức chuẩn trên mạng ra đời nhằm đáp ứng nhu cầu phát triển này. Các giao thức chuẩn (ở tầng mạng và vận chuyển) được sử dụng rộng rãi nhất hiện nay như: giao thức TCP/IP, giao thức SNA của IBM, OSI, ISDN, X.25 hoặc giao thức LAN-to-LAN NetBIOS. Một máy tính chứa chương trình server được coi là một máy chủ hay máy phục vụ (server) và máy chứa chương trình client được coi là máy tớ (client). Mô hình mạng trên đó có các máy chủ và máy tớ giao tiếp với nhau theo 1 hoặc nhiều dịch vụ được gọi là mô hình client/server. Thực tế thì mô hình client/server là sự mở rộng tự nhiên và tiện lợi cho việc truyền thông liên tiến trình trên các máy tính cá nhân. Mô hình này cho phép xây dựng các chương trình client/server một cách dễ dàng và sử dụng chúng để liên tác với nhau để đạt hiệu quả hơn.

Mô hình client/server như sau: Client/Server là mô hình tổng quát nhất, trên thực tế thì một server có thể được nối tới nhiều server khác nhằm làm việc hiệu quả và nhanh hơn. Khi nhận được 1 yêu cầu từ client, server này có thể gửi tiếp yêu cầu vừa nhận được cho server khác ví dụ như database server vì bản thân nó không thể xử lý yêu cầu này được. Máy server có thể thi hành các nhiệm vụ đơn giản hoặc phức tạp. Ví dụ như một máy chủ trả lời thời gian hiện tại trong ngày, khi một máy client yêu cầu lấy thông tin về thời gian nó sẽ phải gửi một yêu cầu theo một tiêu chuẩn do server định ra, nếu yêu cầu được chấp nhận thì máy server sẽ trả về thông tin mà client yêu cầu. Có rất nhiều các dịch vụ server trên mạng nhưng nó đều hoạt động theo nguyên lý là nhận các yêu cầu từ client sau đó xử lý và trả kết quả cho client yêu cầu. Thông thường chương trình server và client được thi hành trên hai máy khác nhau. Cho dù lúc nào server cũng ở trạng thái sẵn sàng chờ nhận yêu cầu từ client nhưng trên thực tế một tiến trình liên tác qua lại (interaction) giữa client và server lại bắt đầu ở phía client, khi mà client gửi tín hiệu yêu cầu tới server. Các chương trình server thường đều thi hành ở mức ứng dụng (tầng ứng dụng của mạng). Sự thuận lợi của phương pháp này là nó có thể làm việc trên bất cứ một mạng máy tính nào có hỗ trợ giao thức truyền thông chuẩn cụ thể ở đây là giao thức TCP/IP. Với các giao thức chuẩn này cũng giúp cho các nhà sản xuất có thể tích hợp nhiều sản phẩm khác nhau của họ lên mạng mà không gặp phải khó khăn gì. Với các chuẩn này thì các chương trình server cho một dịch vụ nào đấy có thể thi hành trên một hệ thống chia sẻ thời gian (timesharing system) với nhiều chương trình và dịch vụ khác hoặc nó có thể chạy trên chính một máy tính các nhân bình thường. Có thể có nhiều chương server cùng làm một dịch vụ, chúng có thể nằm trên nhiều máy tính hoặc một máy tính. Với mô hình trên chúng ta nhận thấy rằng mô hình client/server chỉ mang đặc điểm của phần mềm không liên quan gì đến phần cứng mặc dù trên thực tế yêu cầu cho một máy server là cao hơn nhiều so với máy client. Lý do là bởi vì máy server phải quản lý rất nhiều các yêu cầu từ các clients khác nhau trên mạng. Ưu và nhược điểm chính Có thể nói rằng với mô hình client/server thì mọi thứ dường như đều nằm trên bàn của người sử dụng, nó có thể truy cập dữ liệu từ xa (bao gồm các công việc như gửi và nhận file, tìm kiếm thông tin,...) với nhiều dịch vụ đa dạng mà mô hình cũ không thể làm được. Mô hình client/server cung cấp một nền tảng lý tưởng cho phép tích hợp các kỹ thuật hiện đại như mô hình thiết kế hướng đối tượng, hệ chuyên gia, hệ thông tin địa lý (GIS)... Một trong những vấn đề nảy sinh trong mô hình này đó là tính an toàn và bảo mật thông tin trên mạng. Do phải trao đổi dữ liệu giữa 2 máy ở 2 khu vực khác nhau cho nên dễ dàng xảy ra hiện tượng thông tin truyền trên mạng bị lộ.

1. Client Trong mô hình client/server, người ta còn định nghĩa cụ thể cho một máy client là một máy trạm mà chỉ được sử dụng bởi 1 người dùng với để muốn thể hiện tính độc lập cho nó. Máy client có thể sử dụng các hệ điều hành bình thường như Win9x, DOS, OS/2... Bản thân mỗi một client cũng đã được tích hợp nhiều chức năng trên hệ điều hành mà nó chạy, nhưng khi được nối vào một mạng LAN, WAN theo mô hình client/server thì nó còn có thể sử dụng thêm các chức năng do hệ điều hành mạng (NOS) cung cấp với nhiều dịch vụ khác nhau (cụ thể là các dịch vụ do các server trên mạng này cung cấp), ví dụ như nó có thể yêu cầu lấy dữ liệu từ một server hay gửi dữ liệu lên server đó... Thực tế trong các ứng dụng của mô hình client/server, các chức năng hoạt động chính là sự kết hợp giữa client và server với sự chia sẻ tài nguyên, dữ liệu trên cả hai máy Vai trò của client Trong mô hình client/server, client được coi như là người sử dụng các dịch vụ trên mạng do một hoặc nhiều máy chủ cung cấp và server được coi như là người cung cấp dịch vụ để trả lời các yêu cầu của các clients. Điều quan trọng là phải hiểu được vai trò hoạt động của nó trong một mô hình cụ thể, một máy client trong mô hình này lại có thể là server trong một mô hình khác. Ví dụ cụ thể như một máy trạm làm việc như một client bình thường trong mạng LAN nhưng đồng thời nó có thể đóng vai trò như một máy in chủ (printer server) cung cấp dịch vụ in ấn từ xa cho nhiều người khác (clients) sử dụng. Client được hiểu như là bề nổi của các dịch vụ trên mạng, nếu có thông tin vào hoặc ra thì chúng sẽ được hiển thị trên máy client.

2. Server Server còn được định nghĩa như là một máy tính nhiều người sử dụng (multiuser computer). Vì một server phải quản lý nhiều yêu cầu từ các client trên mạng cho nên nó hoạt động sẽ tốt hơn nếu hệ điều hành của nó là đa nhiệm với các tính năng hoạt động độc lập song song với nhau như hệ điều hành UNIX, WINDOWS... Server cung cấp và điều khiển các tiến trình truy cập vào tài nguyên của hệ thống. Các ứng dụng chạy trên server phải được tách rời nhau để một lỗi của ứng dụng này khô ng làm hỏng ứng dụng khác. Tính đa nhiệm đảm bảo một tiến trình không sử dụng toàn bộ tài nguyên hệ thống. Vai trò của server. Như chúng ta đã bàn ở trên, server như là một nhà cung cấp dịch vụ cho các clients yêu cầu tới khi cần, các dịch vụ như cơ sở dữ liệu, in ấn, truyền file, hệ thống... Các ứng dụng server cung cấp các dịch vụ mang tính chức năng để hỗ trợ cho các hoạt động trên các máy clients có hiệu quả hơn. Sự hỗ trợ của các dịch vụ này có thể là toàn bộ hoặc chỉ một phần thông qua IPC. Để đảm bảo tính an toàn trên mạng cho nên server này còn có vai trò như là một nhà quản lý toàn bộ quyền truy cập dữ liệu của các máy clients, nói cách khác đó là vai trò quản trị mạng. Có rất nhiều cách thức hiện nay nhằm quản trị có hiệu quả, một trong những cách đang được sử dụng đó là dùng tên Login và mật khẩu