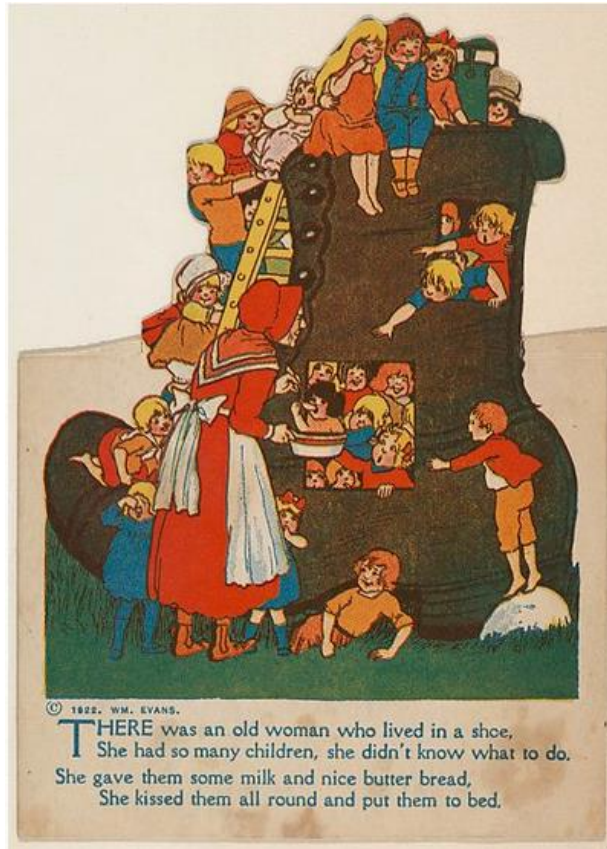


Transmission Control Protocol

MÃ HP: 123013 – LẬP TRÌNH MẠNG

Lớp vận chuyển và lớp mạng



12 đứa trẻ trong ngôi nhà của Tí gửi
thư cho 12 đứa trẻ trong nhà của Tèo:

hosts = houses (nhà)

processes = kids (đứa trẻ)

app messages = letters in envelopes (lá
thư)

Lớp vận chuyển và lớp mạng

Network layer: giao tiếp logic giữa các hosts

Transport layer: giao tiếp logic giữa các processes

12 đứa trẻ trong ngôi nhà của Tí gửi thư cho 12 đứa trẻ trong nhà của Tèo:

hosts = houses (nhà)

processes = kids (đứa trẻ)

app messages = letters in envelopes (lá thư)

transport protocol = Tí và Tèo là anh em trong nhà

network-layer protocol = Dịch vụ vận chuyển

Giao thức truyền tải TCP

TCP: Transmission Control Protocol

- Tin cậy
- Điều khiển tắc nghẽn
- Kiểm soát lưu lượng
- Thiết lập kết nối

Lập trình Socket

Ví dụ:

1. Máy khách đọc một dòng ký tự (dữ liệu) từ bàn phím của nó và gửi dữ liệu đến máy chủ
2. Máy chủ nhận dữ liệu và chuyển đổi các ký tự thành chữ hoa, máy chủ gửi dữ liệu đã sửa đổi đến máy khách
3. Máy khách nhận dữ liệu đã sửa đổi và hiển thị dòng trên màn hình của nó

Lập trình Socket với TCP

Client phải kết nối với Server

Server phải đang chạy (chờ tiếp nhận)

Server phải tạo Socket để tiếp nhận

Quan điểm: TCP cung cấp khả năng truyền luồng byte theo thứ tự, đáng tin cậy (“đường ống”) giữa các process của Client và Server

Client kết nối với Server:

Tạo TCP socket, chỉ định địa chỉ IP, số cổng của tiến trình máy chủ

Khi Client tạo Socket: TCP Client sẽ tạo kết nối với TCP Server

Quá trình TCP bắt tay 3 bước



Quá trình TCP bắt tay 3 bước

Server state

Client state

```
clientSocket = socket(AF_INET, SOCK_STREAM)
```

LISTEN

```
clientSocket.connect((serverName, serverPort))
```

SYNSENT

ESTAB

choose init seq num, x
send TCP SYN msg

SYNbit=1, Seq=x

SYNbit=1, Seq=y
ACKbit=1; ACKnum=x+1

received SYNACK(x)
indicates server is live;
send ACK for SYNACK;
this segment may contain
client-to-server data

ACKbit=1, ACKnum=y+1

received ACK(y)
indicates client is live

```
serverSocket = socket(AF_INET, SOCK_STREAM)  
serverSocket.bind(('', serverPort))  
serverSocket.listen(1)  
connectionSocket, addr = serverSocket.accept()
```

LISTEN

SYN RCVD

ESTAB

Quá trình TCP bắt tay 3 bước

Trước khi dữ liệu ứng dụng bắt đầu truyền giữa Client và Server, kết nối TCP phải được thiết lập, cả hai bên cần thống nhất về trình tự gói và số lượng biến kết nối TCP.

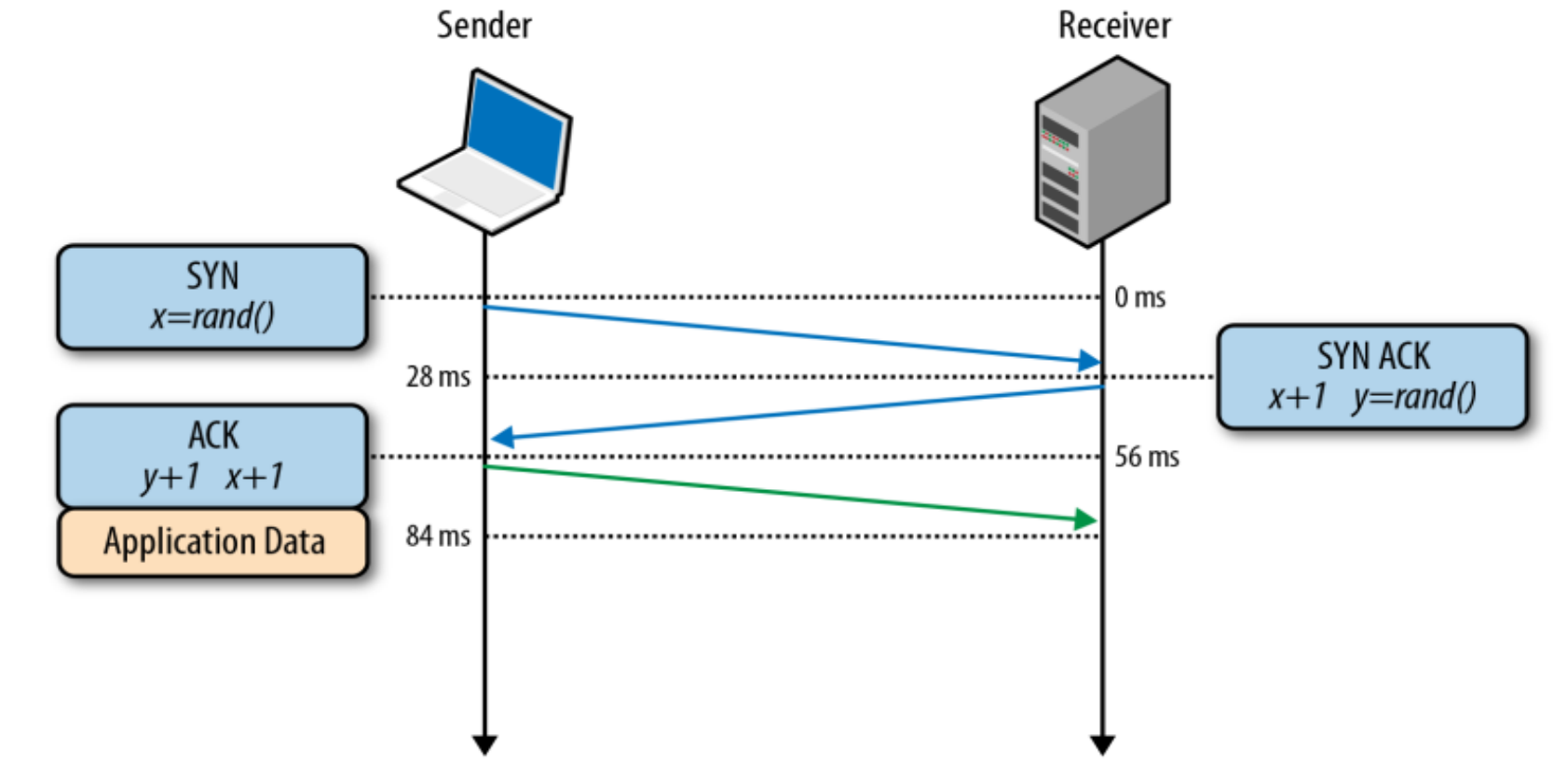
Đây là cách nó bắt đầu:

SYN – Client chọn một số thứ tự ngẫu nhiên và gửi một gói SYN với các cờ và tùy chọn TCP khác.

SYN ACK - Server tăng x lên 1, tạo số thứ tự ngẫu nhiên y của riêng nó, gắn thêm bộ cờ & tùy chọn của riêng nó và gửi phản hồi.

ACK – Client tăng cả x & y lên 1 & gửi phản hồi ACK cuối cùng để hoàn tất quá trình bắt tay.

Quá trình TCP bắt tay 3 bước



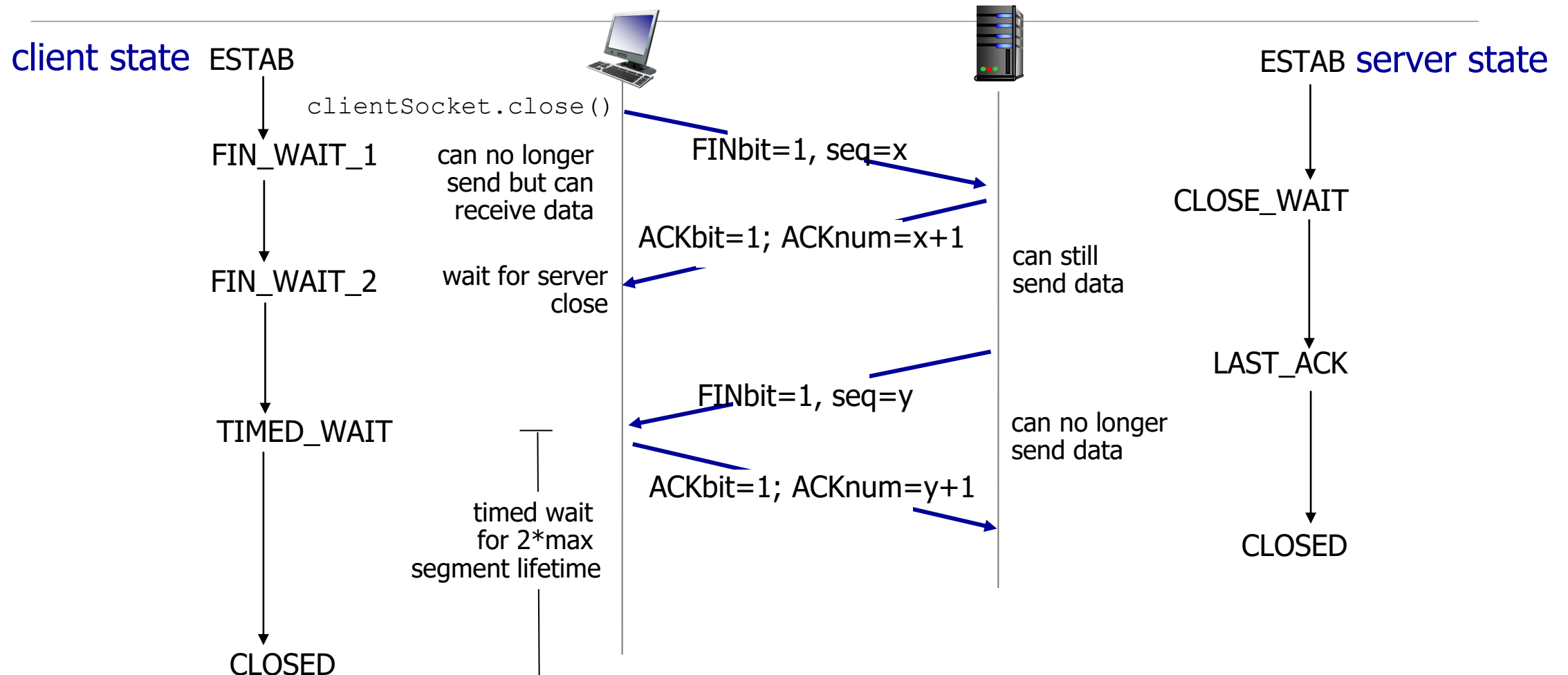
Quá trình TCP bắt tay 3 bước

Client ngay lập tức bắt đầu gửi gói dữ liệu sau khi gửi gói ACK, nhưng máy chủ phải đợi sự xuất hiện của ACK.

Vì vậy SYN -> SYN_ACK mất toàn bộ thời gian khứ hồi trong mạng, ACK cũng thêm thời gian đó. Sự chậm trễ của việc thiết lập kết nối thông qua bắt tay 3 bước là rất đáng kể và gây ra độ trễ trong giao tiếp giữa Client và Server.

Độ trễ này là do thời gian lan truyền giữa Client và Server, không phải do băng thông của bất kỳ phía nào. Do đó, việc quản lý một kết nối TCP liên tục/tái sử dụng kết nối có ý nghĩa quan trọng hơn là chỉ mở một kết nối mới mỗi lần.

Quá trình đóng kết nối TCP



Quá trình đóng kết nối TCP

Phía Client/Server sẽ thực hiện đóng kết nối

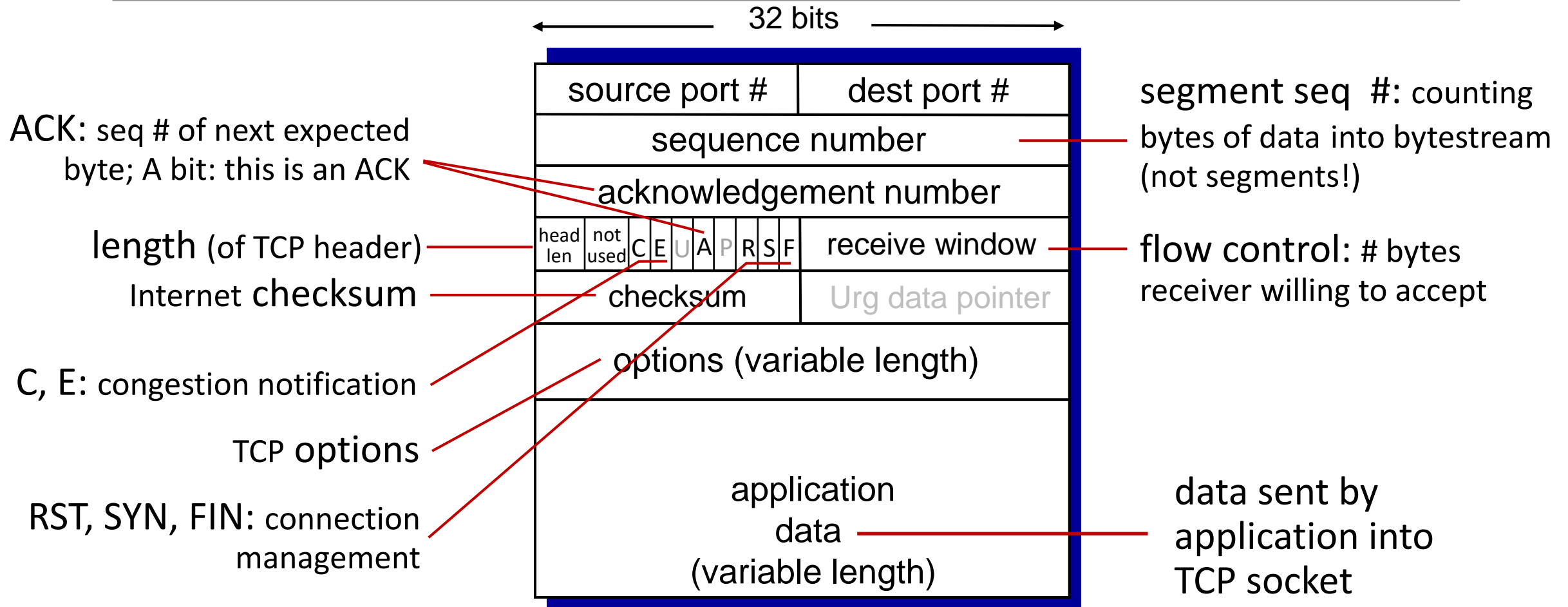
- Gửi phân đoạn (segment) TCP với FIN bit = 1

Trả lời đã nhận FIN với ACK

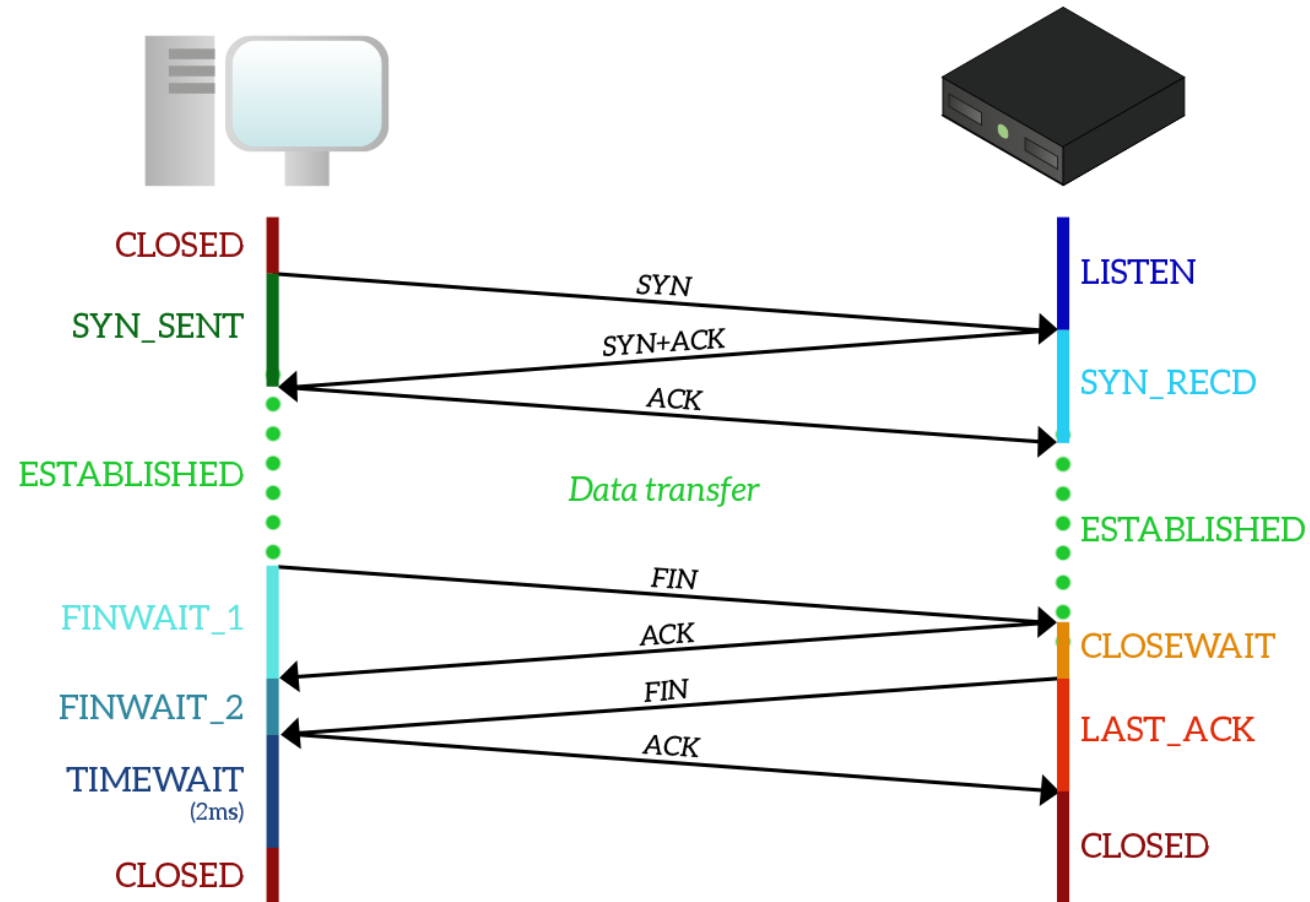
- Khi nhận FIN, ACK có thể được kết hợp với FIN riêng

Thực hiện trao đổi FIN đồng thời

Cấu trúc gói tin TCP



Giao tiếp TCP giữa Client và Server

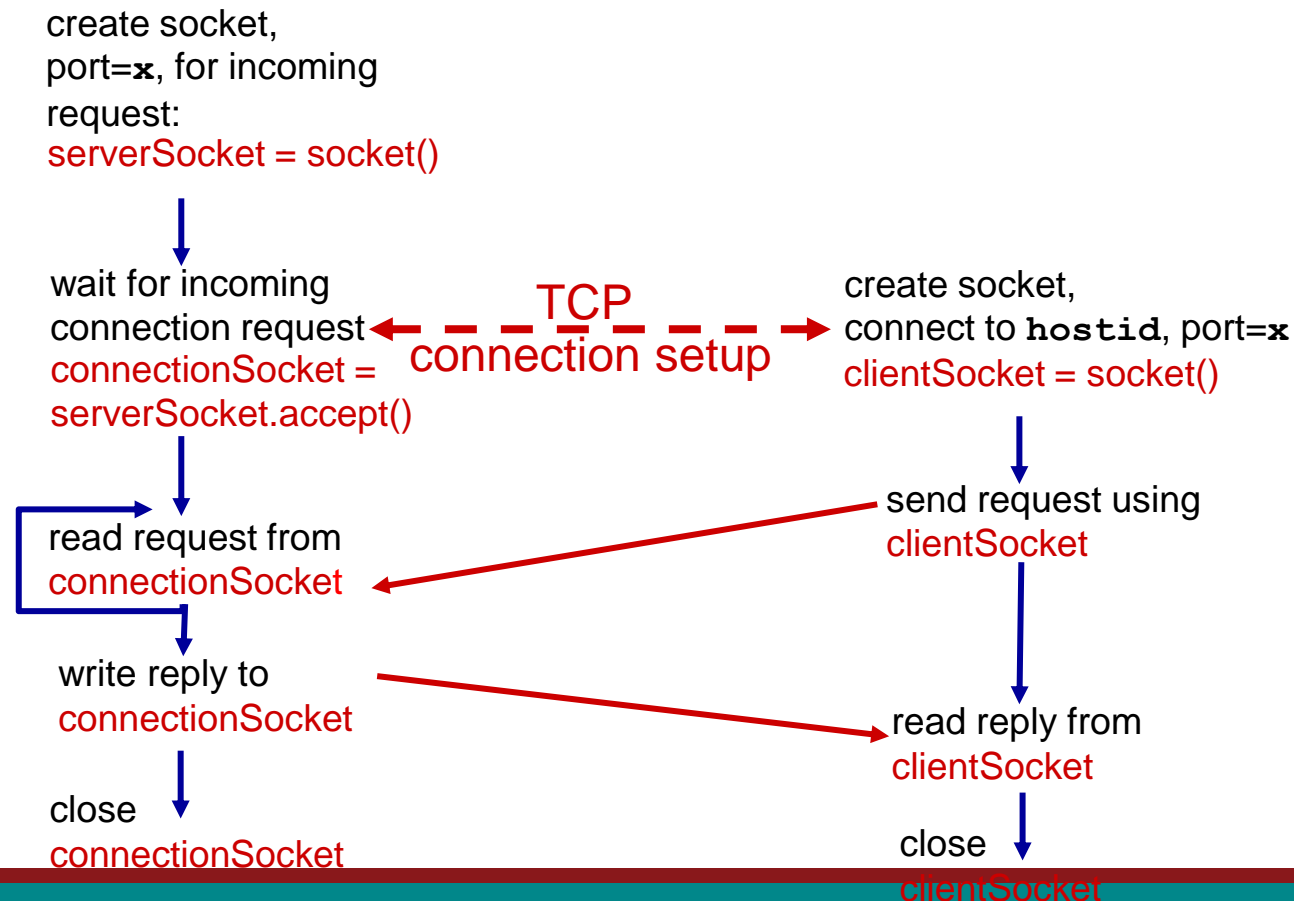


Giao tiếp TCP giữa Client và Server



server (running on `hostid`)

client



Ưu và nhược điểm TCP

THẢO LUẬN