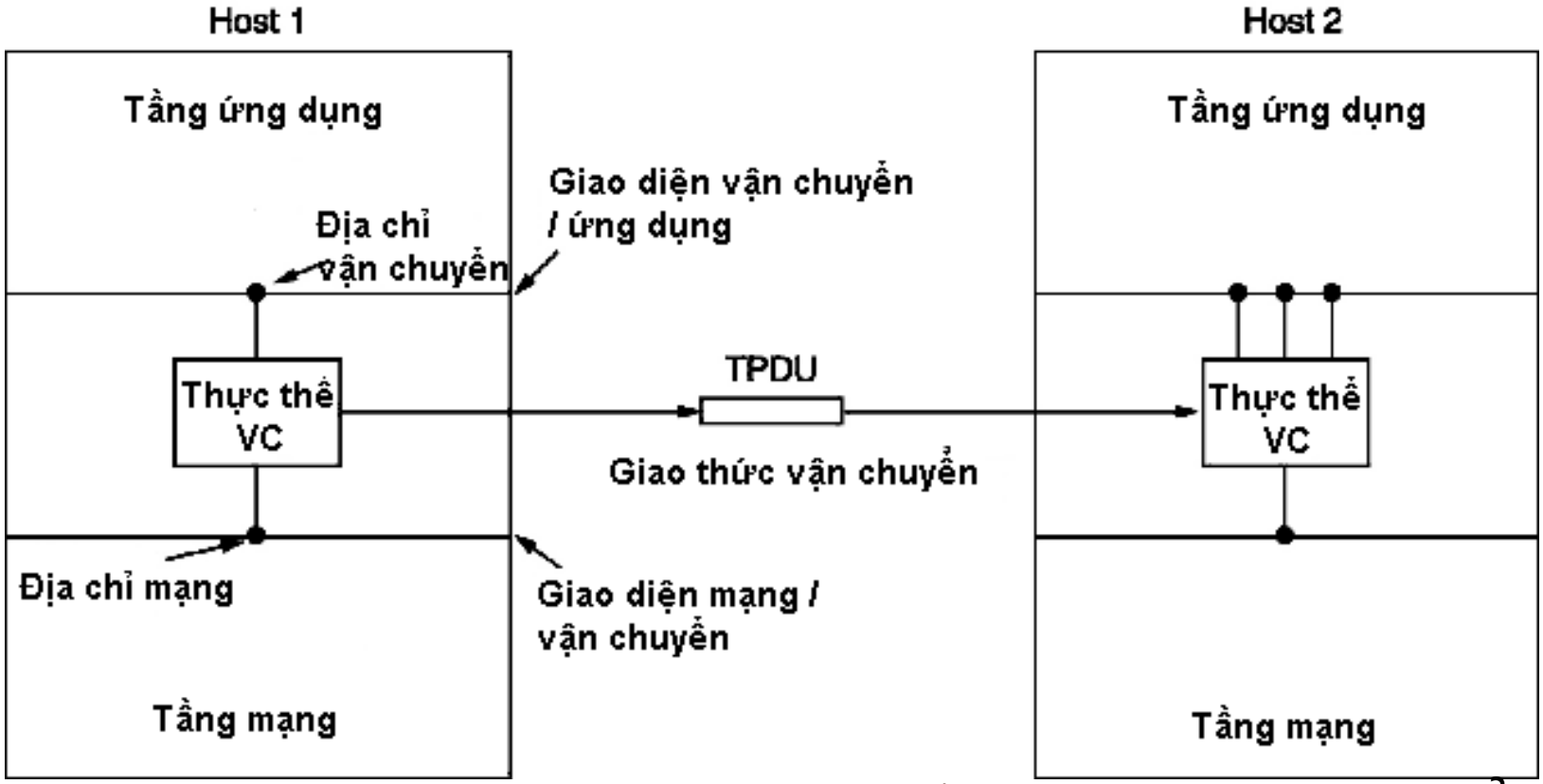
TẦNG VẬN CHUYỂN (Computer Network)

Trần Duy Quang

Nhiệm vụ của tầng vận chuyển

- Tầng vận chuyển đảm bảo truyền tải kiểu End point –to- End point
- (Tầng mạng đảm bảo truyền tải kiểu Host -to- Host)
- End point là các chương trình ứng dụng
- Cung cấp dịch vụ vận chuyển gói tin hiệu quả, tin cậy và tiết kiệm chi phi cho người dùng

Vị trí của tầng vận chuyển



Dịch vụ cung cấp bởi tầng vận chuyển

- Hai kiểu dịch vụ
  - Có nối kết:
    - Thiết lập nối kết
    - Truyền dữ liệu
    - Hủy nối kết
  - Không nối kết

Dịch vụ cung cấp bởi tầng vận chuyển

- Các hàm dịch vụ cơ sở để gọi các dịch vụ vận chuyển và các hàm này là đơn giản, duy nhất và độc lập với các hàm cơ sở ở tầng mạng
- Các hàm dịch vụ cơ sở - Có nối kết: LISTEN, CONNECT, SEND, RECEIVE, DISCONNECT
- Các hàm dịch vụ cơ sở - Không nối kết: SEND, RECEIVE

Các yếu tố cấu thành giao thức vận chuyển

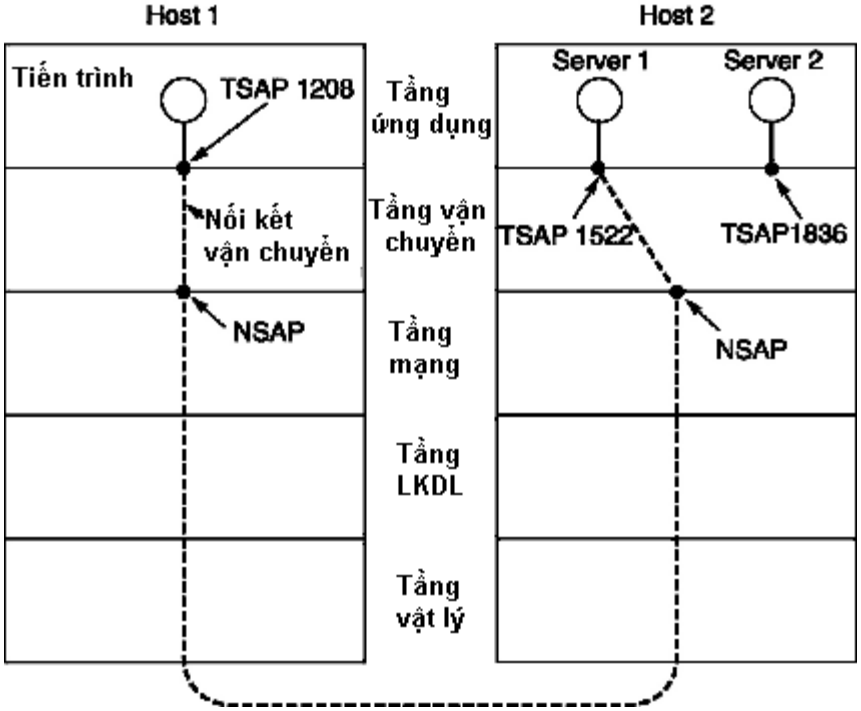
- Điều khiển lỗi, đánh số thứ tự gói tin và điều khiển luồng dữ liệu.
- Môi trường giao tiếp qua một tập các mạng trung gian.

Các yếu tố cấu thành giao thức vận chuyển

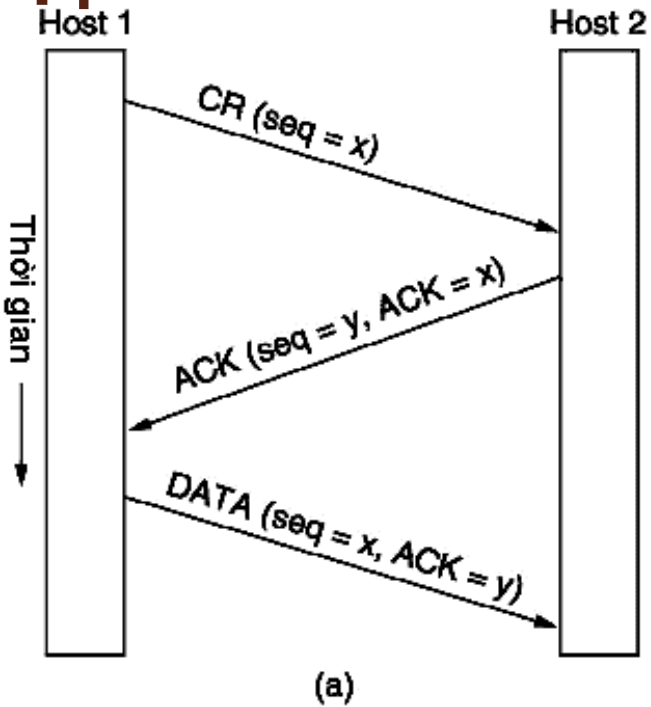
- Những vấn đề cần quan tâm:
  - Định địa chỉ các tiến trình trên các host
  - Xử lý những trường hợp mất gói tin, gói tin đi chậm dẫn đến hết hạn và gởi thêm một gói tin bị trùng lắp
  - Đồng bộ hóa hai tiến trình đang trao đổi dữ liệu khi mà chúng đang ở rất xa nhau

Định địa chỉ

- Địa chỉ tiến trình là TSAP (Transport Service Access Point).
  - Mạng Internet dùng số hiệu cổng (port),
  - Mạng ATM dùng AAL-SAP.
- Tầng mạng được gọi là NSAP (Network SAP)



Thiết lập nối kết



CR (Connection Request)

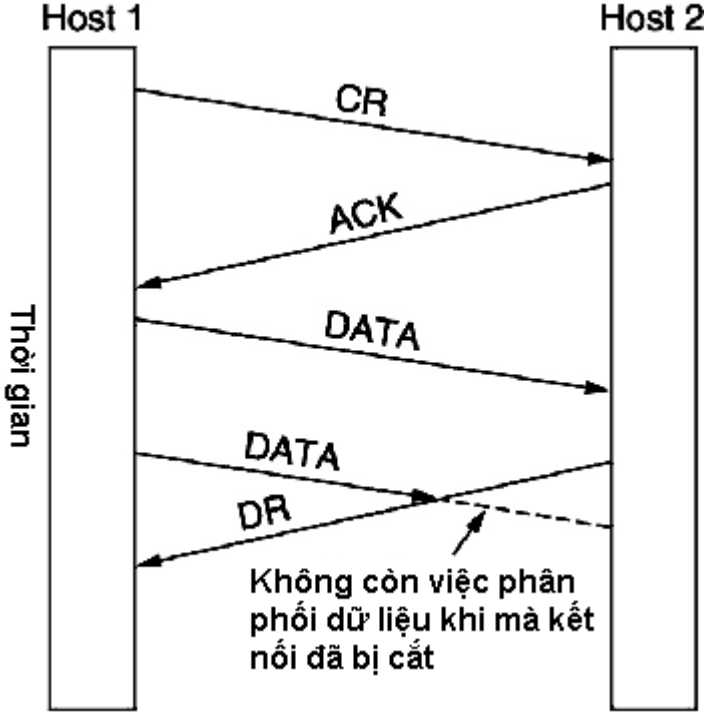
Three-way hand-shake
Hoạt động bình thường

Giải phóng nối kết

- Hai kiểu giải phóng nối kết:
  - Kiểu dị bộ hoạt động như sau: khi một bên ngắt nối kết, kết nối sẽ bị hủy bỏ (giống như trong hệ thống điện thoại).
  - Kiểu đồng bộ làm việc theo phương thức ngược lại: khi cả hai đồng ý hủy bỏ nối kết, nối kết mới thực sự được hủy.

Giải phóng nối kết dị bộ

- DR (Disconnection Request)

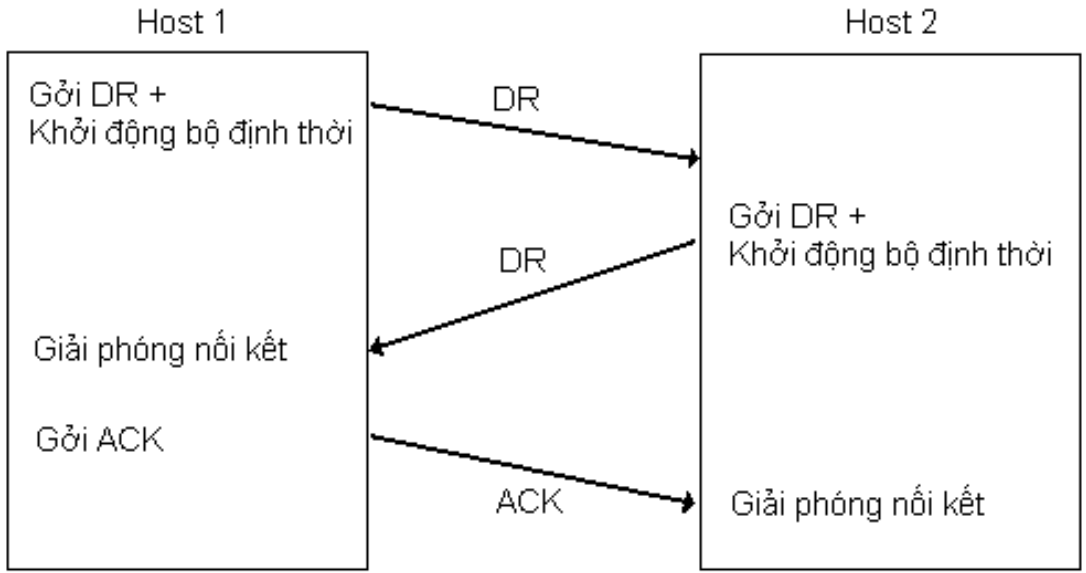


Giải phóng nối kết đồng bộ

- Một nút phải tiếp tục nhận dữ liệu sau khi đã gởi đi yêu cầu giải phóng nối kết (DISCONNECT REQUEST – DR), cho đến khi nhận được chấp thuận hủy bỏ nối kết của bên đối tác
- Sử dụng phương pháp hủy nối kết ba chiều cùng với bộ định thời (hẹn giờ)

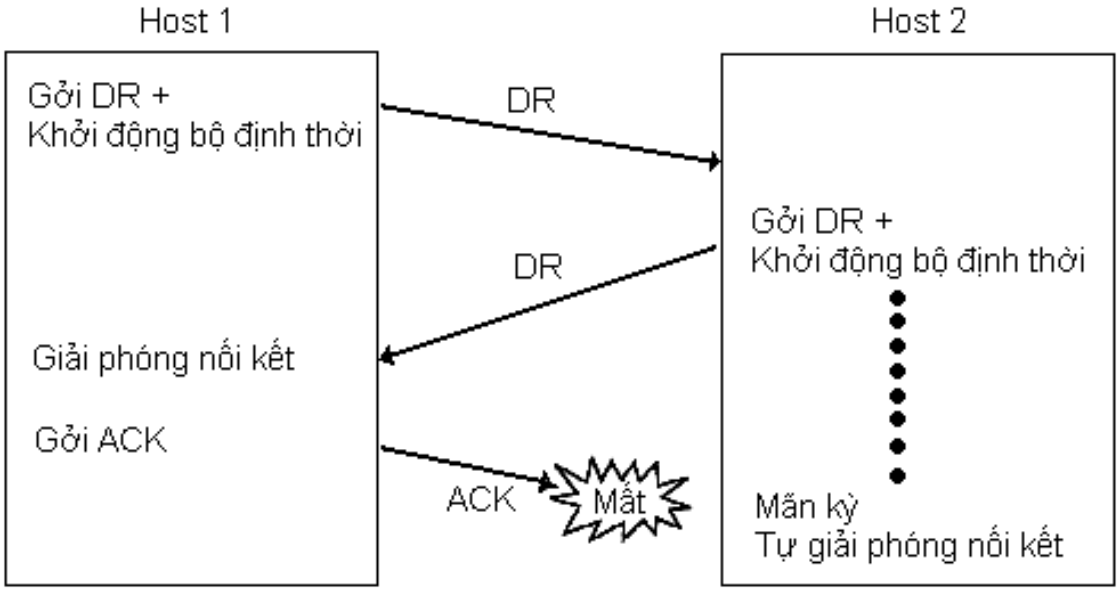
Giải phóng nối kết đồng bộ

- Bình thường



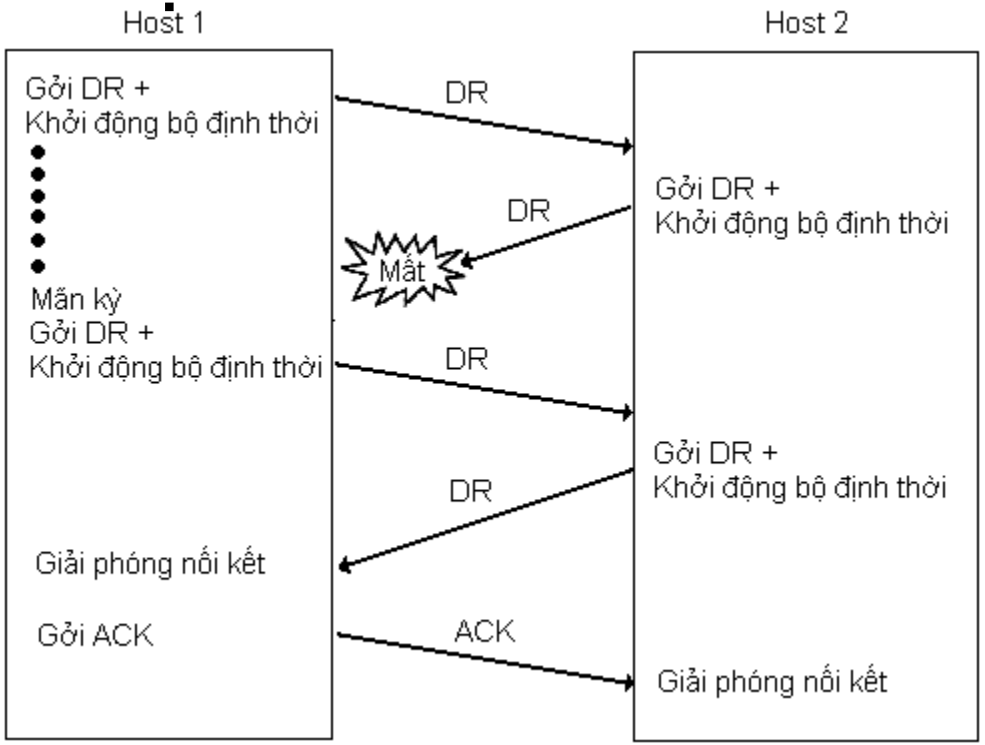
Giải phóng nối kết đồng bộ

- Khung ACK cuối cùng bị mất



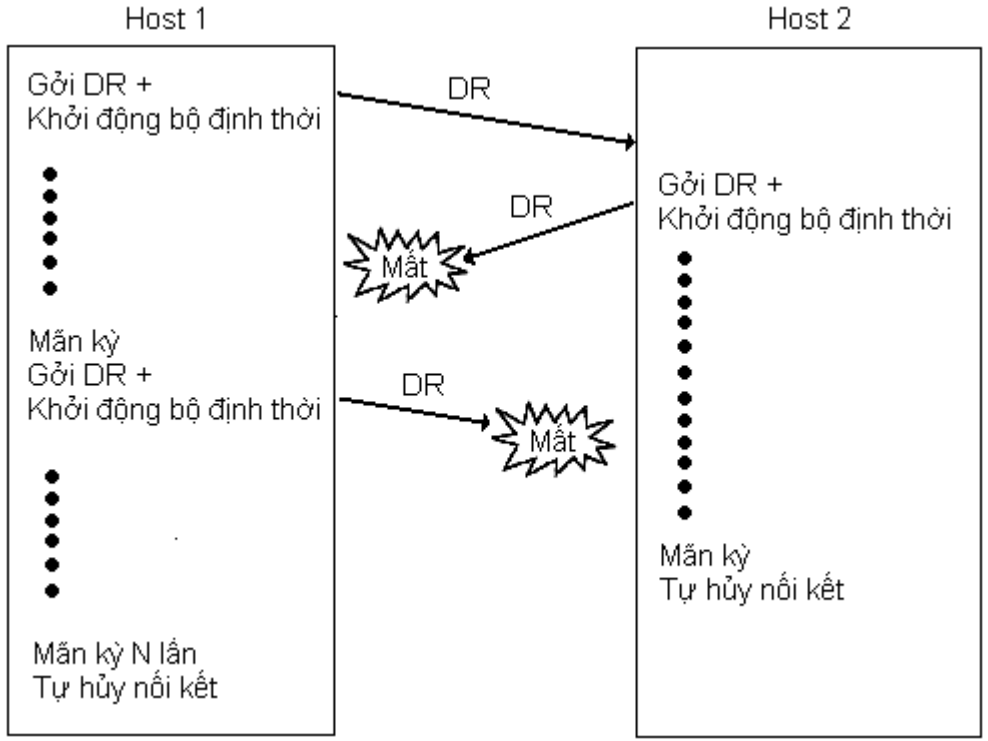
Giải phóng nối kết đồng bộ

- Trả lời bị mất



Giải phóng nối kết đồng bộ

- *Trả lời mất và các gói tin DR theo sau cũng bị mất*



Điều khiển luồng (Data flow)

- *[còn gọi là Điều khiển thông lượng]*
- Sử dụng giao thức cửa sổ trượt với kích thước cửa sổ của bên gởi và bên nhận là **khác nhau.**
- Cần phải có sơ đồ cung cấp buffer động:
  - Trước tiên, bên gởi phải gởi đến bên nhận một yêu cầu dành riêng số lượng buffer để chứa các gói tin đến.

Điều khiển luồng (Data flow)

- Bên nhận cũng phải trả lời cho bên gởi số lượng buffer tối đa mà nó có thể cung cấp.
- Mỗi khi báo nhận ACK cho một gói tin có số thứ tự SEQ_NUM, bên nhận cũng phải gởi kèm theo thông báo cho bên gởi biết là lượng buffer còn lại là bao nhiêu để bên gởi không làm ngập bên nhận.

Data flow

Dấu … là mất thông điệp

	A	Thông điệp	B	Giải thích
1	→	<yêu cầu 8 buffers>	→	A muốn B cung cấp 8 buffers
2	← →	<ack = 0, buf = 4>	← →	B chỉ cấp cho A 4 buffers thôi
3	→	<seq = 0, data = m0>	→	A còn lại 3 buffers
4	→	<seq = 1, data = m1>		A còn lại 2 buffers
5	← →	<seq = 2, data = m2>	← → …	Thông điệp bị mất, nhưng A nghĩ nó còn 1 buffer
6	→ →	<ack = 1, buf = 3>	→ →	B báo nhận cho thông điệp 0 và 1, còn 3 buffers
7	←	<seq = 3, data = m3>	←	A còn lại 1 buffer

Data flow

Dấu … là mất thông điệp

8		<seq = 4, data = m4>		A không còn buffer nào và phải dừng
9		<seq = 2, data = m2>		Thông điệp thứ 2 của A mãn kỳ và được truyền lại
10		<ack = 4, buf = 0>		Mọi thứ đã được báo nhận, nhưng A vẫn nghẽn
11	←	<ack = 4, buf = 1>	←	A có thể gởi 1 gói tin thứ 5
12	← →	<ack = 4, buf = 2>	← →	B có thêm 1 buffer nữa
13	→	<seq = 5, data = m5>	→	A còn lại 1 buffer
14	←	<seq = 6, data = m6>	← ←	A nghẽn một lần nữa
15		<ack = 6, buf = 0>		A vẫn còn nghẽn
16	…	<ack = 6, buf = 4>		Khả năng dẫn đến deadlock

(thông điệp bị mất)

Tầng vận chuyển trong mạng Internet

- Nhiệm vụ
  - Đảm bảo việc phân phối thông điệp qua mạng.
  - Phân phối các thông điệp theo thứ tự mà chúng được gởi.
  - Không làm trùng lắp thông điệp.

Tầng vận chuyển trong mạng Internet

- Hỗ trợ những thông điệp có kích thước lớn.
- Hỗ trợ cơ chế đồng bộ hóa.
- Hỗ trợ việc liên lạc của nhiều tiến trình trên mỗi host.

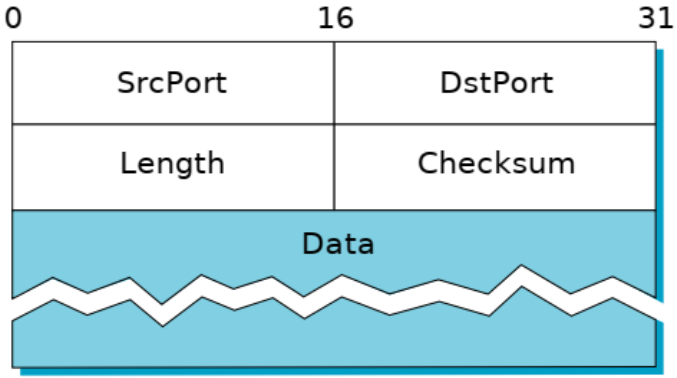
- Hỗ trợ hai phương thức hoạt động
  - Không nối kết (UDP)
  - Có nối kết (TCP)

Giao thức UDP (User Datagram Protocol)

- UDP là dịch vụ truyền dữ liệu dạng không nối kết.
- Không có thiết lập nối kết giữa hai bên truyền nhận.
- Gói tin UDP (segment) có thể xuất hiện tại nút đích bất kỳ lúc nào.
- Các segment UDP tự thân chứa mọi thông tin cần thiết để có thể tự đi đến đích.

Giao thức UDP (User Datagram Protocol)

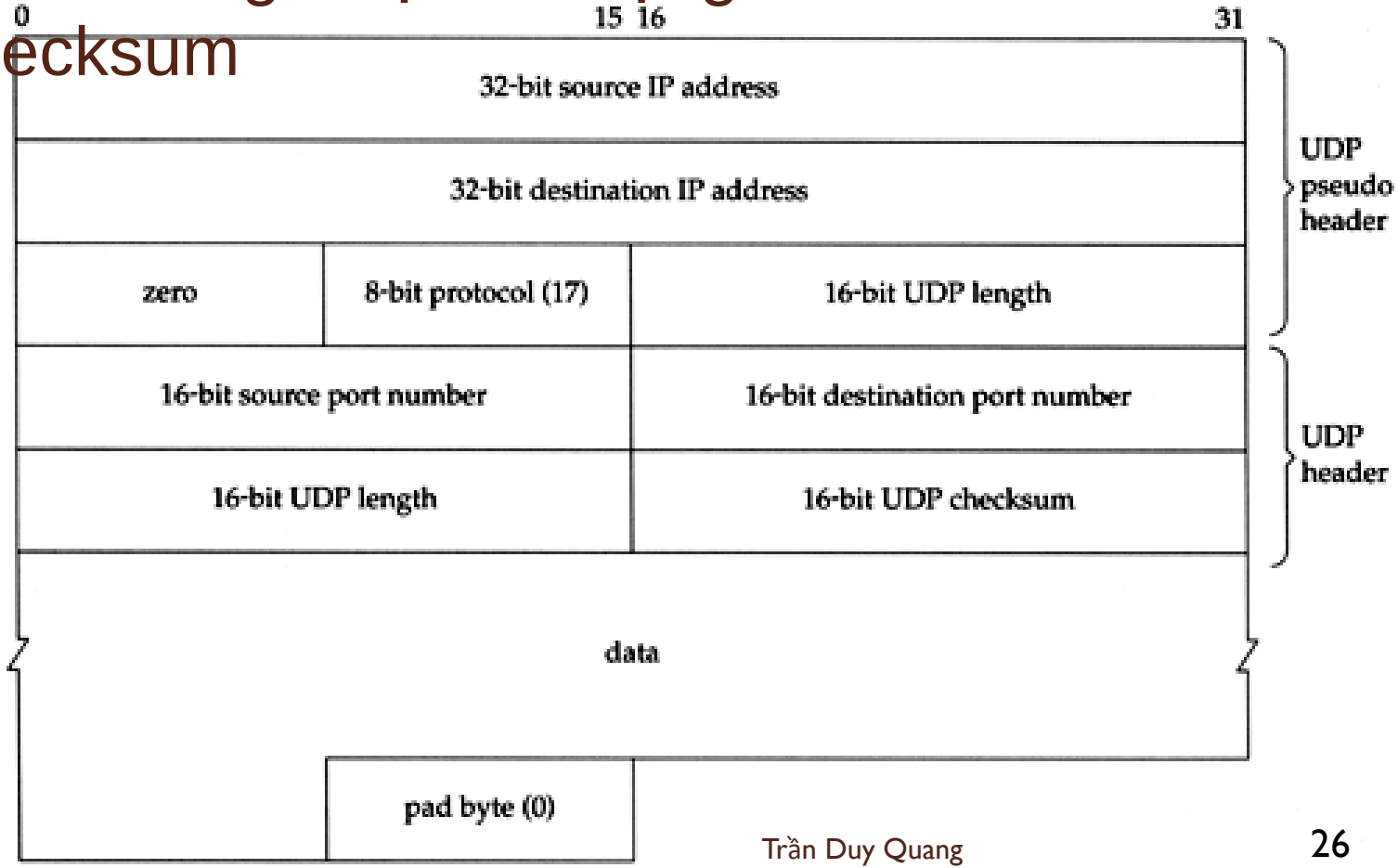
- **SrcPort**: Địa chỉ cổng nguồn, là số hiệu của tiến trình gởi gói tin đi.
- **DstPort**: Địa chỉ cổng đích, là số hiệu của tiến trình sẽ nhận gói tin.
- **Length (UDP total length)**: Tổng chiều dài của segment, tính luôn cả phần header.



Giao thức UDP (User Datagram Protocol)

- **Checksum**: Là phần kiểm tra lỗi tổng hợp trên *phần header ảo*, phần header và phần dữ liệu.
  - *Phần header ảo* (12 byte) chứa các trường: địa chỉ IP nguồn, địa chỉ IP đích (lấy từ IP header), Protocol và UDP total length.

Các trường được sử dụng để tính UDP checksum



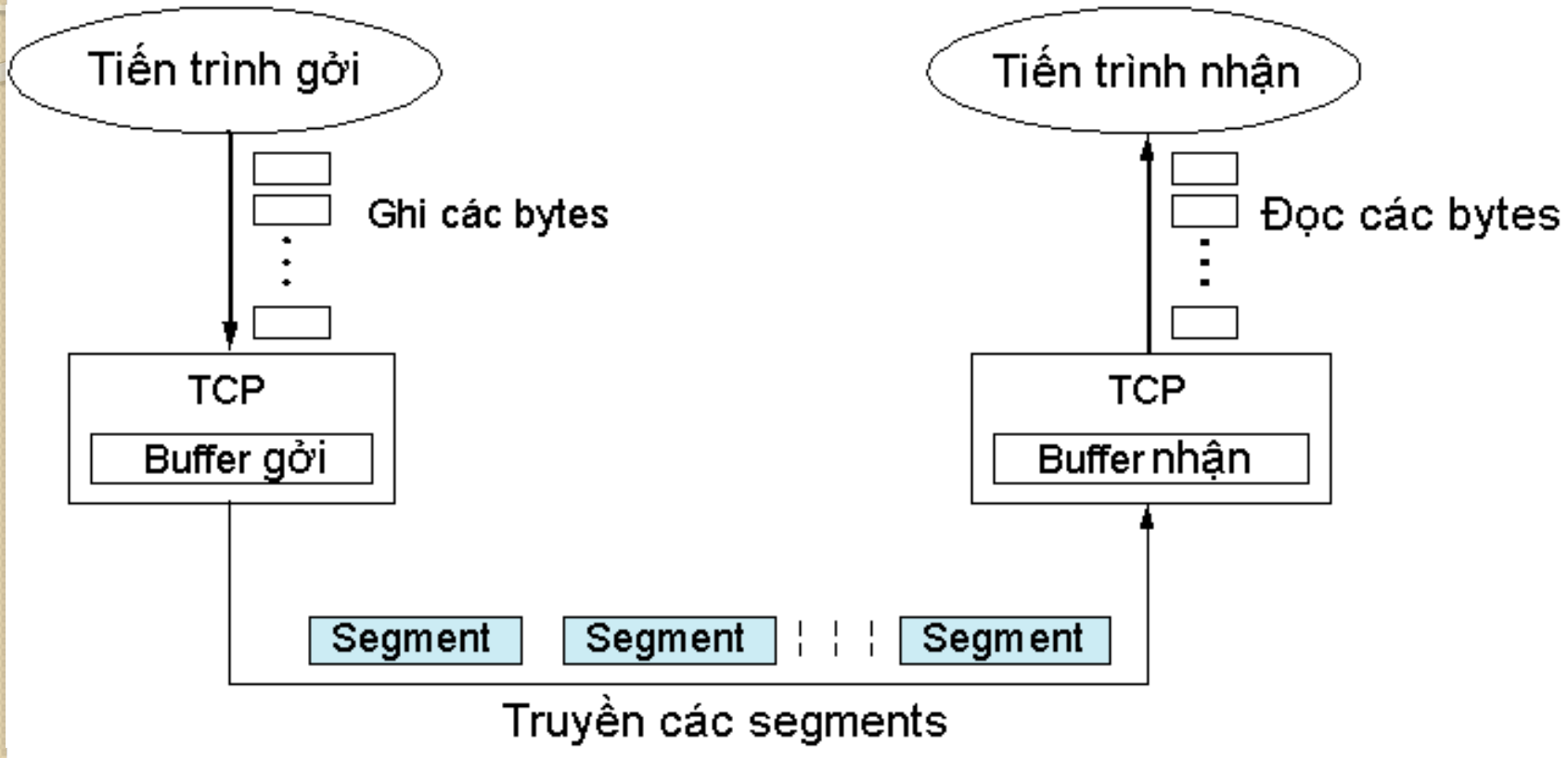
Giao thức UDP (User Datagram Protocol)

- **Data**: Phần dữ liệu hai bên gởi cho nhau.
- UDP hoạt động dạng không tin cậy, vì: Không có báo nhận dữ liệu từ trạm đích; không có cơ chế để phát hiện mất gói tin hoặc các gói tin đến không theo thứ tự; không có cơ chế tự động gởi lại những gói tin bị mất; không có cơ chế điều khiển luồng dữ liệu, và do đó có thể bên gởi sẽ làm ngập bên nhận.
- Nếu cần xử lý lỗi, ứng dụng tự xây dựng giao thức bên trên UDP.

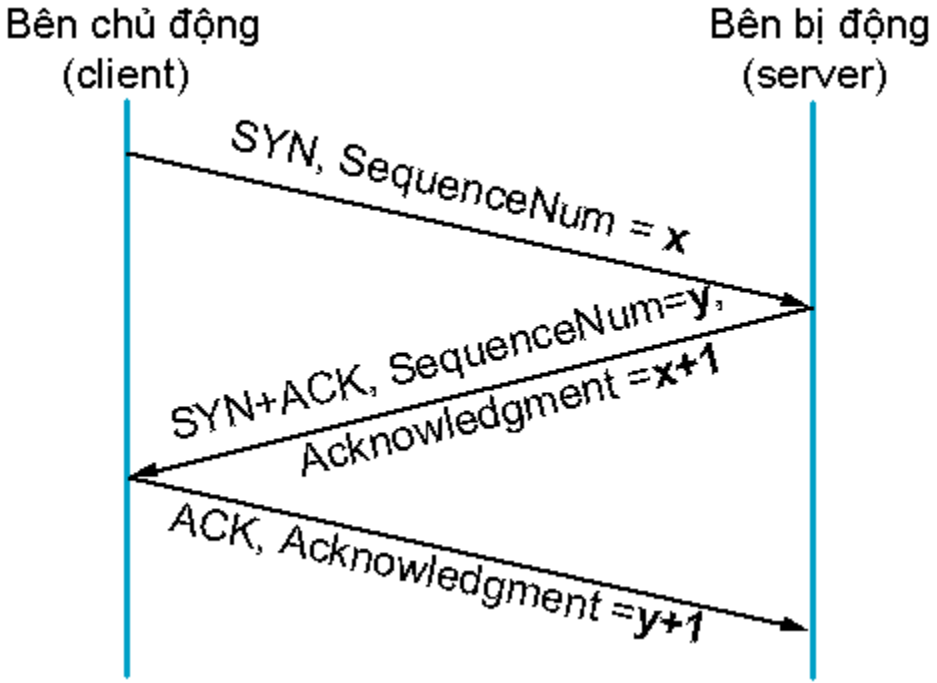
Giao thức TCP (Transmission Control Protocol)

- TCP là giao thức cung cấp dịch vụ vận chuyển tin cậy, hướng nối kết theo kiểu truyền thông tin bằng cách phân luồng các bytes.
- TCP là giao thức truyền song công, hỗ trợ cơ chế đa hợp.
- TCP là giao thức hướng bytes.

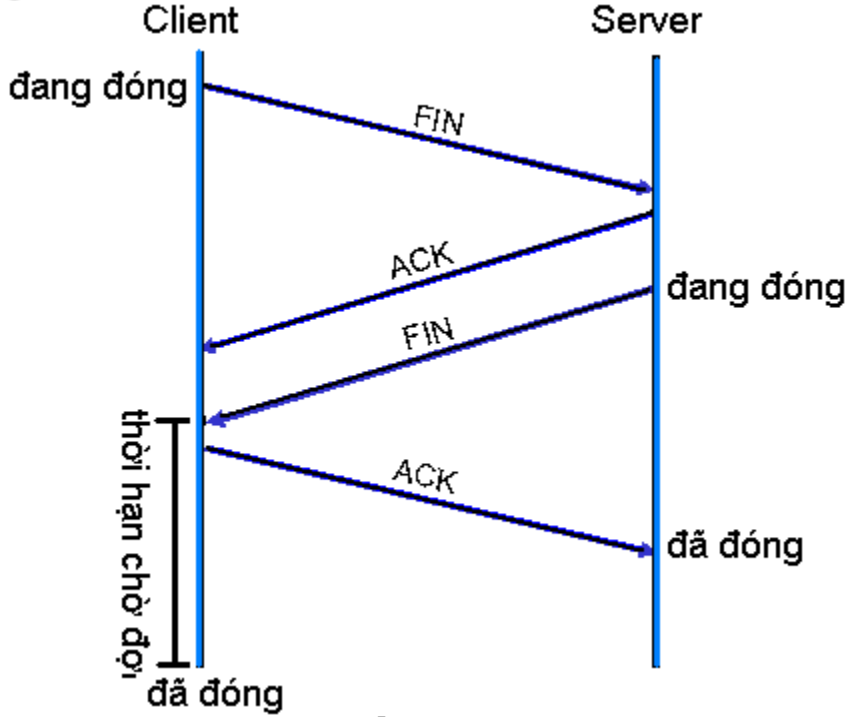
Giao thức TCP (Transmission Control Protocol)



Giao thức TCP (Transmission Control Protocol)



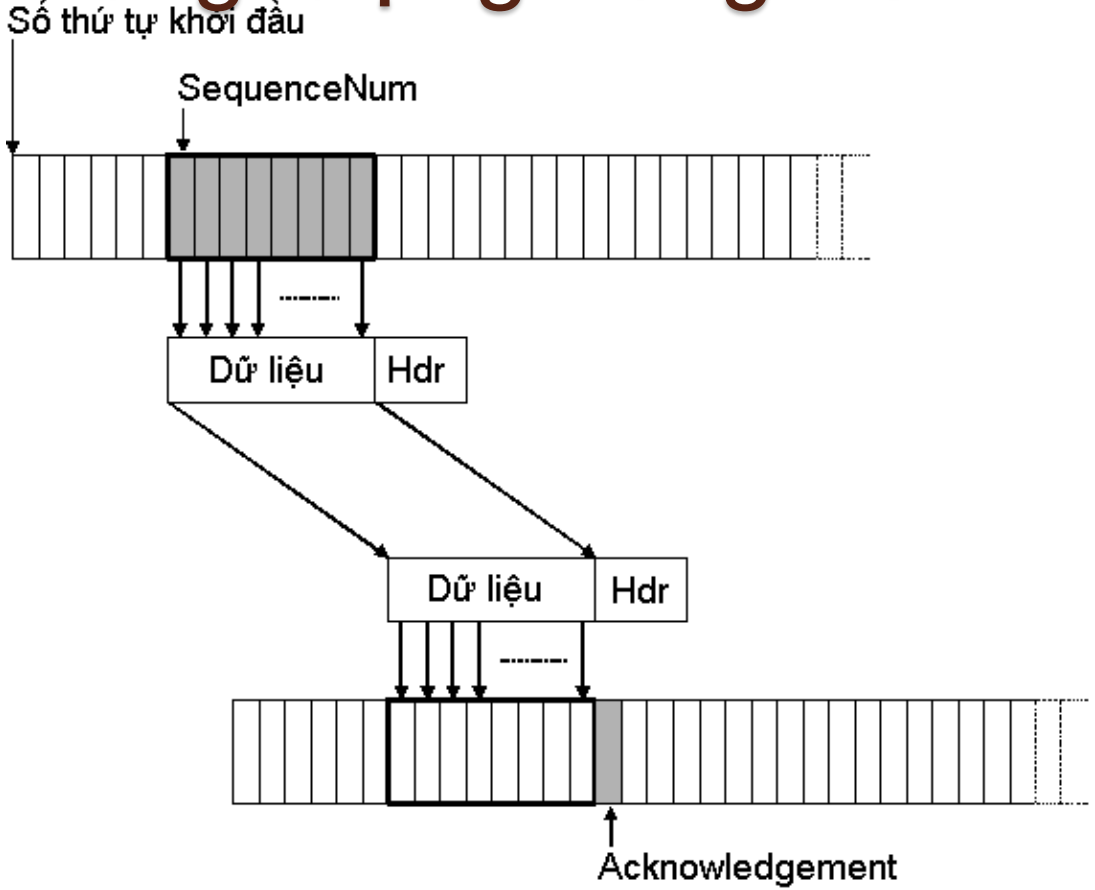
Bắt tay trong TCP



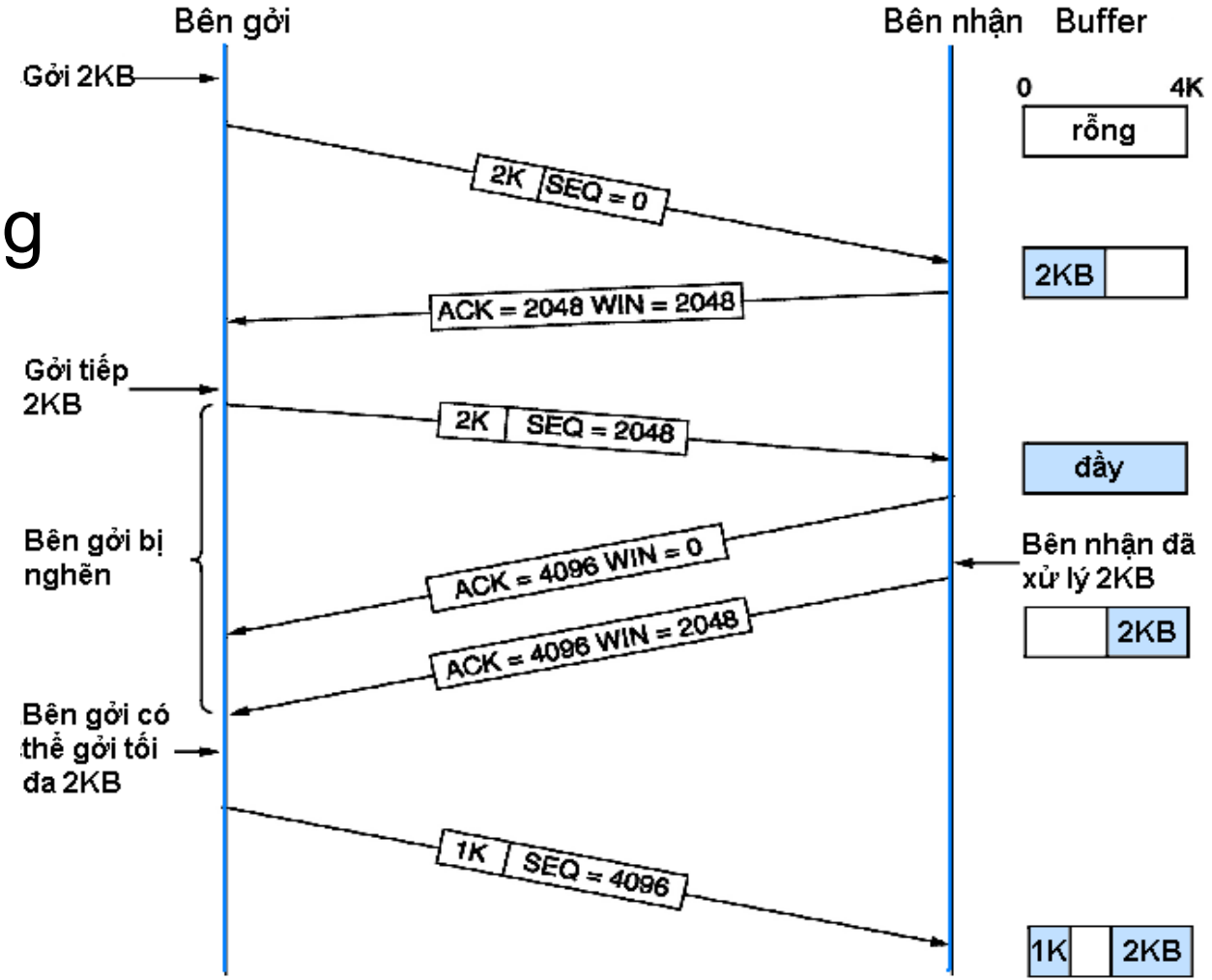
Hủy bắt tay trong TCP

Điều khiển thông lượng trong TCP

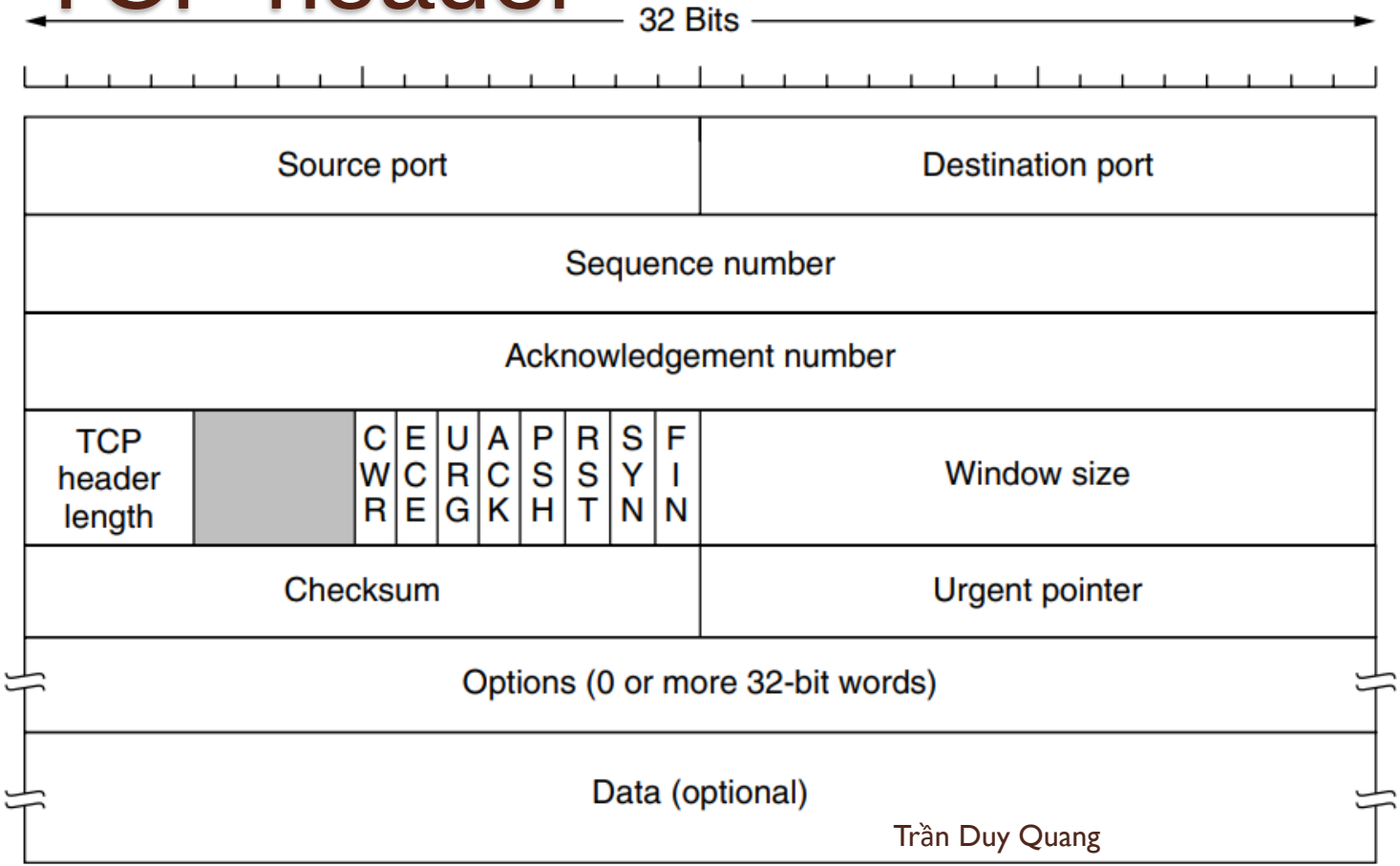
- Là giao thức truyền hướng bytes
- Mỗi lần truyền đi một Segment (chứa một dãy các byte)



- Sử dụng giao thức cửa sổ trượt TCP



TCP header


32 Bits
Source port
Destination port
Sequence number
Acknowledgement number
TCP header length
CWR
ECE
URG
ACK
PSH
RST
SYN
FIN
Window size
Checksum
Urgent pointer
Options (0 or more 32-bit words)
Data (optional)

Giao thức TCP (Transmission Control Protocol)

- Các trường **SrcPort** và **DstPort** chỉ ra địa chỉ cổng nguồn và đích, giống như trong UDP. Hai trường này cùng với hai địa chỉ IP nguồn và đích sẽ được kết hợp với nhau để định danh duy nhất một kết nối TCP. Nghĩa là một kết nối TCP sẽ được định danh bởi một bộ 4 trường **(Cổng nguồn, Địa chỉ IP nguồn, Cổng đích, Địa chỉ IP đích)**

Giao thức TCP (Transmission Control Protocol)

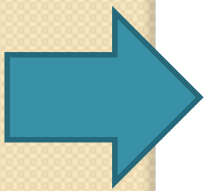
- **Acknowledgement**, **SequenceNum** và **AdvertisedWindow (Window size)** tất cả được sử dụng trong giải thuật cửa sổ trượt của TCP.
- **SequenceNum**: chứa số thứ tự của byte đầu tiên của một dãy các bytes chứa trong một segment (TCP là giao thức hướng byte).
- **Acknowledgement:** báo nhận, nếu cờ ACK = 1 thì đây là số thứ tự byte kế tiếp (theo thứ tự dãy byte) mà bên nhận sẵn sàng nhận.

Giao thức TCP (Transmission Control Protocol)

- **AdvertisedWindow (Window size):** kích thước cửa sổ nhận.
- **HdrLen (TCP header length)**: 4 bit, kích thước header tính bằng word (32 bit).
- **Reserved**: 4 bit, dự phòng, nên chứa 0.
- **Flags** (8 bits): Một bit trong trường này là một cờ: CWR, ECE, URG, ACK, PUSH, RESET, SYN, FIN.

Giao thức TCP (Transmission Control Protocol)

- Cờ **ECE** (ECN-Echo): bên nhận dùng tín hiệu này để thông báo về tắc nghẽn để bên gởi gởi chậm lại.
- Cờ **CWR** (Congestion Window Reduced): khi bên gởi đã gởi chậm lại, bên gởi dùng CWR để thông báo cho bên nhận và bên nhận có thể dừng gởi tín hiệu ECN-Echo (tham khảo RFC 3168).



Giao thức TCP (Transmission Control Protocol)

- Cờ **URG** được dùng để đánh dấu segment này chứa dữ liệu khẩn cấp. Khi cờ này được đặt, trường **UrgPtr (Urgent pointer)** sẽ chỉ ra nơi bắt đầu của dữ liệu không khẩn cấp (dữ liệu khẩn cấp luôn nằm ở đầu của phần dữ liệu) (RFC-793).

Giao thức TCP (Transmission Control Protocol)

- Cờ **ACK** được đặt mỗi khi dùng đến Acknowledgement (nếu không thì Acknowledgement bị bỏ qua).
- Cờ **PUSH** báo hiệu bên gởi đã *không chờ nhận đủ các bytes để lấp đầy một segment*, trong buffer gởi dù có bao nhiêu bytes dữ liệu cũng được bên gởi đóng vào segment và gởi đi.

Giao thức TCP (Transmission Control Protocol)

- Cờ **RESET** được dùng để thông báo rằng bên nhận đã bị rối (ví dụ như nó đã nhận một segment mà đáng lẽ ra không phải là segment đó), vì thế nó muốn hủy bỏ nối kết.
- Hai cờ **SYN** và **FIN** được dùng để thiết lập và giải phóng nối kết.