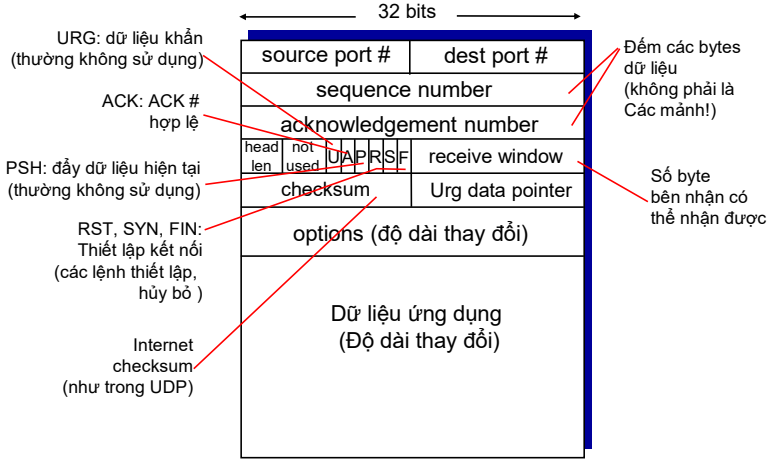
Chương 3: Nội dung

3.1 Các dịch vụ tầng giao vận
3.2 Ghép kênh và phân kênh
3.3 Vận chuyển không kết nối: UDP
3.4 Các nguyên lý truyền dữ liệu tin cậy
3.5 Vận chuyển hướng kết nối: TCP
3.5.1 Cấu trúc đoạn dữ liệu (segment)
3.5.2 Truyền dữ liệu tin cậy
3.5.3 Điều khiển luồng
3.5.4 Quản lý kết nối
3.6 Các nguyên lý điều khiển tắc nghẽn
3.7 Điều khiển tắc nghẽn TCP

Khái quát TCP [RFCs: 793,1122,1323, 2018, 2581]

- Điểm-tới-điểm:
  - Một bên gửi, một bên nhận
- Truyền *dòng byte* theo đúng thứ tự và truyền tin cậy:
  - Không có "ranh giới thông điệp"
- pipeline:
  - Điều khiển tắc nghẽn và điều khiển luồng TCP thiết lập kích thước cửa sổ
- Truyền dữ liệu song công (full duplex):
  - Luồng dữ liệu đi theo 2 hướng trên cùng một kết nối
  - MSS: maximum segment size (kích thước đoạn lớn nhất)
- Hướng kết nối:
  - Bắt tay (trao đổi các thông điệp điều khiển) khởi tạo trạng thái cho bên gửi và bên nhận trước khi trao đổi dữ liệu
- Điều khiển luồng:
  - Bên gửi không lấn át bên nhận

Cấu trúc TCP segment

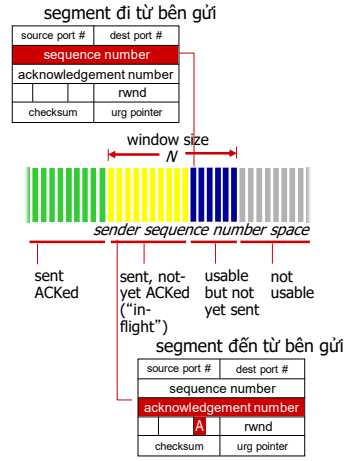


Số thứ tự và báo nhận ACK trong TCP

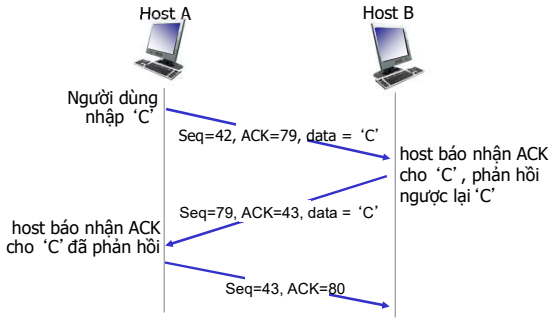
Số thứ tự:
- "Số" dòng byte của byte đầu tiên trong đoạn (segment) dữ liệu

Báo nhận:
- Số thứ tự của byte tiếp theo được mong đợi từ phía bên kia
- ACK tích lũy

Hỏi: Làm thế nào bên nhận xử lý được các segment không đúng thứ tự?
- Trả lời: TCP không đề cập, tùy thuộc vào người thực hiện



Số thứ tự và báo nhận ACK trong TCP



Kịch bản telnet đơn giản

TCP round trip time và timeout

Hỏi: Làm thế nào để thiết lập giá trị TCP timeout?

- Dài hơn RTT
  - nhưng RTT thay đổi
- *Quá ngắn:* timeout sớm, không cần truyền lại
- *Quá dài:* phản ứng chậm với các segment bị mất

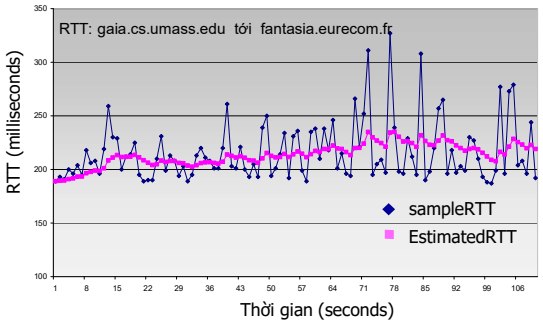
Hỏi: Ước lượng RTT như nào?

- **SampleRTT**: thời gian đo được từ khi truyền segment đến khi nhận được ACK
  - Bỏ qua việc truyền lại
- **SampleRTT** có thể thay đổi, cần giá trị RTT ước lượng "mượt hơn"
  - Tính trung bình một vài độ đo gần đây, không chỉ **SampleRTT** hiện tại

TCP round trip time và timeout

EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT

- Giá trị đặc trưng: **α =** 0.125



TCP round trip time và timeout

- Khoảng thời gian timeout: **EstimatedRTT** cộng với "hệ số dự trữ an toàn"
  - Nếu có biến thiên lớn trong **EstimatedRTT**, thì hệ số dự trữ an toàn phải lớn hơn
- Ước lượng sự biến thiên của SampleRTT từ EstimatedRTT:

DevRTT = (1-β)*DevRTT +
β*|SampleRTT-EstimatedRTT|

(Giá trị đặc trưng: β = 0.25)

TimeoutInterval = EstimatedRTT + 4*DevRTT

RTT ước lượng — "hệ số dự trữ an toàn"

Chương 3: Nội dung

3.1 Các dịch vụ tầng giao vận

3.2 Ghép kênh và phân kênh

3.3 Vận chuyển không kết nối: UDP

3.4 Các nguyên lý truyền dữ liệu tin cậy

3.5 Vận chuyển hướng kết nối: TCP

- 3.5.1 Cấu trúc đoạn dữ liệu (segment)
- 3.5.2 Truyền dữ liệu tin cậy
- 3.5.3 Điều khiển luồng
- 3.5.4 Quản lý kết nối

3.6 Các nguyên lý điều khiển tắc nghẽn

3.7 Điều khiển tắc nghẽn TCP

Truyền dữ liệu tin cậy trong TCP

- TCP tạo dịch vụ rdt trên dịch vụ không tin cậy của IP
  - Truyền segment theo kiểu pipelining
  - ACK tích lũy
  - Dùng bộ định thời cho việc truyền lại
- Việc truyền lại được kích hoạt bởi:
  - Các sự kiện timeout
  - ACK bị trùng lặp

Hãy bắt đầu xem xét bên gửi TCP theo cách đơn giản:

- Bỏ qua trùng lặp ACK
- Bỏ qua điều khiển luồng, điều khiển tắc nghẽn

Các sự kiện của TCP bên gửi:

Dữ liệu nhận từ ứng dụng:

- Tạo segment với số thứ tự
- Số thứ tự là số dòng byte của byte dữ liệu đầu tiên trong segment
- Khởi tạo bộ định thời nếu chưa chạy:
  - Chú ý bộ định thời của segment chưa được báo nhận muộn nhất
  - Hết thời hạn: **TimeOutInterval**

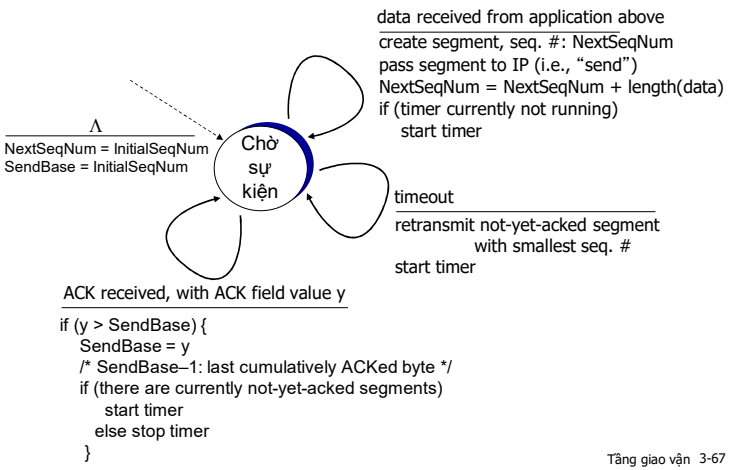
Timeout:

- Truyền lại segment bị timeout
- Khởi tạo lại bộ định thời

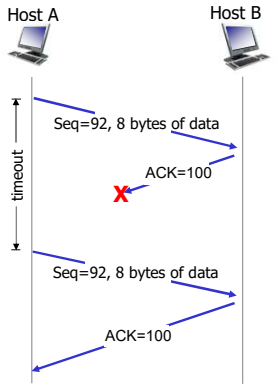
ACK đã nhận:

- Nếu ACK báo nhận cho các segment chưa được báo nhận trước đó, thì:
  - Cập nhật lại các segment đã được báo nhận
  - Khởi tạo bộ định thời nếu vẫn còn các segment chưa được báo nhận

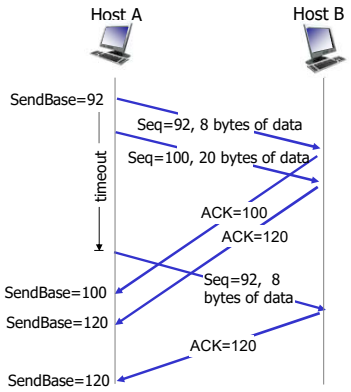
TCP bên gửi (đơn giản hóa)



TCP: Các tình huống phải truyền lại

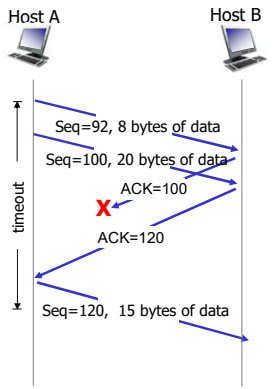


Tình huống mất ACK



Tình huống timeout sớm

TCP: Các tình huống phải truyền lại



Tình huống ACK tích lũy

Tạo ACK trong TCP [RFC 1122, RFC 2581]

Sự kiện tại bên nhận	*Hành động của TCP tại bên nhận*
Segment đến đúng thứ tự với số thứ tự mong muốn. Tất cả dữ liệu đến đã được báo nhận	ACK bị trễ. Chờ 500ms cho segment tiếp theo. Nếu không có segment tiếp theo thì gửi ACK
Segment đến đúng thứ tự với số thứ tự mong muốn. Một segment khác đang chờ ACK	Gửi ngay một ACK tích lũy, báo nhận ACK cho cả hai segment đến đúng thứ tự
Segment đến không đúng số thứ tự, số thứ tự lớn hơn mong đợi. Phát hiện có khoảng trống	Gửi ngay *ACK trùng lặp*, chỉ ra số thứ tự của byte mong đợi tiếp theo
Segment đến lấp đầy hoặc một phần khoảng trống	Gửi ngay ACK, với điều kiện là segment bắt đầu ngay tại điểm có khoảng trống

Truyền lại nhanh trong TCP

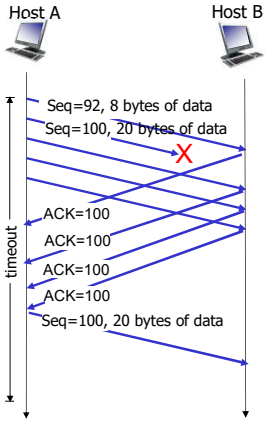
- Chu kỳ time-out thường tương đối dài:
  - Trễ dài trước khi gửi lại gói tin đã bị mất
- Phát hiện các segment bị mất qua các ACK bị trùng lặp.
  - Bên gửi thường gửi nhiều segment song song
  - Nếu segment, có thể sẽ có nhiều ACK bị trùng lặp.

Truyền lại nhanh trong TCP

Nếu bên gửi nhận được 3 ACK trùng lặp cho cùng một dữ liệu ("Ba ACK trùng lặp"), thì sẽ gửi lại segment chưa được báo nhận có số thứ tự nhỏ nhất

- Có thể là đã bị mất segment chưa được báo nhận, nên không cần phải đợi đến timeout

Truyền lại nhanh trong TCP



Truyền lại nhanh sau khi bên gửi nhận được 3 ACK trùng lặp