Chương 3: Nội dung

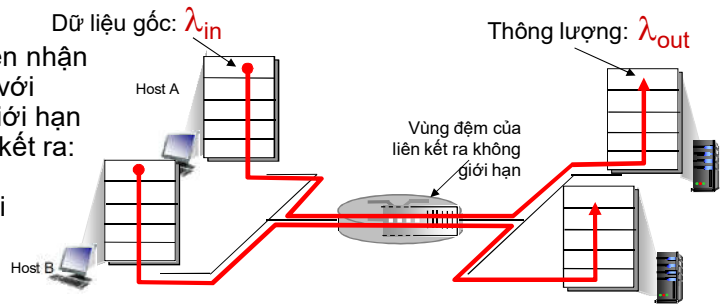
Các nguyên lý điều khiển tắc nghẽn

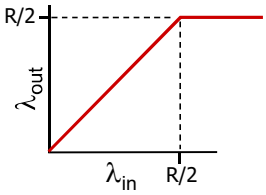
Tắc nghẽn:

- Có thể hiểu là: "quá nhiều nguồn cùng gửi quá nhiều dữ liệu với tốc độ quá nhanh tới *mạng*"
- Khác điều khiển luồng dữ liệu!
- Các biểu hiện chính:
  - Mất các gói tin (tràn bộ đệm tại các bộ định tuyến)
  - Trễ quá lâu (hàng đợi dài trong vùng đệm của bộ định tuyến)
- Là một trong mười vấn đề nan giải nhất của mạng!

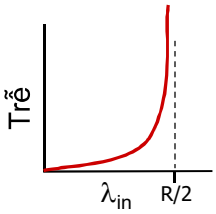
Các nguyên nhân/chi phí của tắc nghẽn: tình huống 1

- Hai bên gửi, hai bên nhận
- Một bộ định tuyến với vùng đệm không giới hạn
- Khả năng của liên kết ra: R
- Không có truyền lại





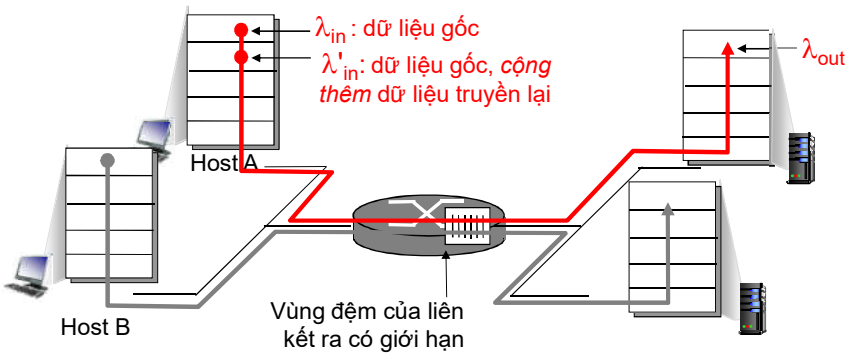
- Thông lượng lớn nhất trên mỗi kết nối: R/2



- Trễ lớn do tốc độ đến, λ_{in}, tiệm cận đến thông lượng tối đa

Các nguyên nhân/chi phí của tắc nghẽn: tình huống 2

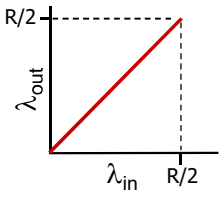
- Một bộ định tuyến, vùng đệm có *giới hạn*
- Bên gửi truyền lại gói tin bị timeout
  - Đầu vào tầng ứng dụng = đầu ra tầng ứng dụng: $\lambda_{in} = \lambda_{out}$
  - Đầu vào tầng giao vận bao gồm việc truyền lại $\lambda'_{in} \geq \lambda_{in}$

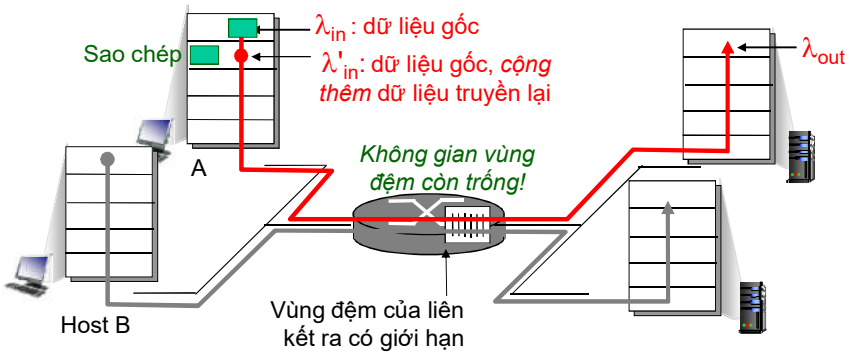


Các nguyên nhân/chi phí của tắc nghẽn: tình huống 2

Lý tưởng hóa: hiểu biết hoàn hảo

- Bên gửi chỉ gửi khi vùng đệm của bộ định tuyến sẵn sàng.

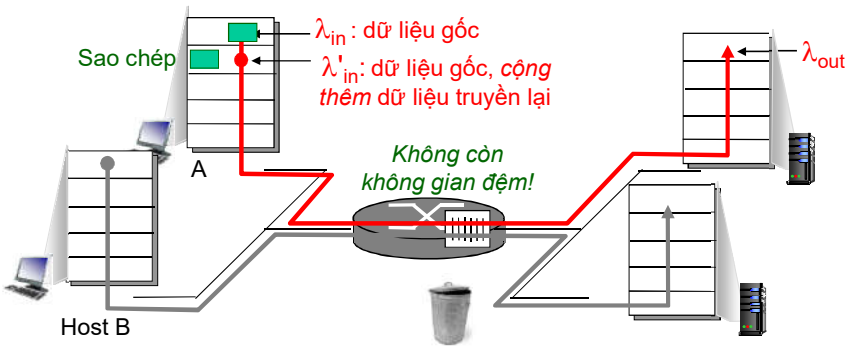




Các nguyên nhân/chi phí của tắc nghẽn: tình huống 2

Lý tưởng hóa: biết về sự mất mát

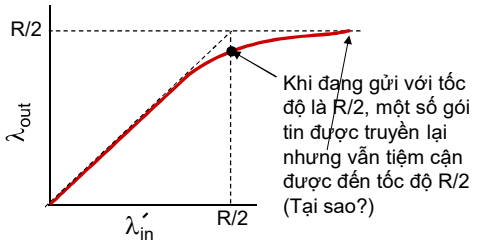
- Các gói tin có thể bị mất, bị bỏ rơi tại bộ định tuyến nếu vùng đệm của nó bị đầy
- Bên gửi chỉ gửi lại nếu gói tin *được biết* là đã bị mất

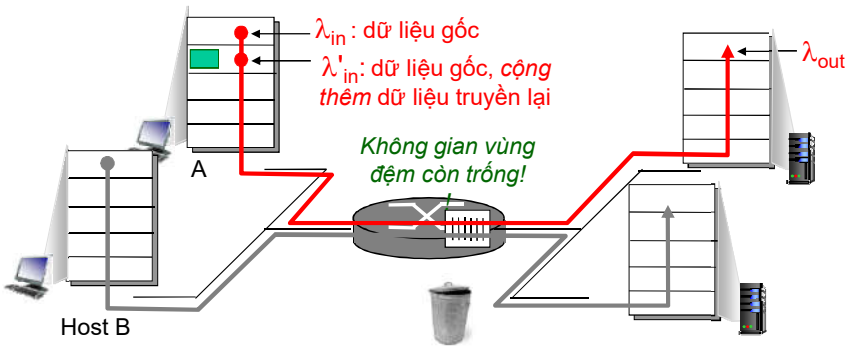


Các nguyên nhân/chi phí của tắc nghẽn: tình huống 2

Lý tưởng hóa: biết về sự mất mát

- Các gói tin có thể bị mất, bị bỏ rơi tại bộ định tuyến nếu vùng đệm của nó bị đầy
- Bên gửi chỉ gửi lại nếu gói tin *được biết* là đã bị mất

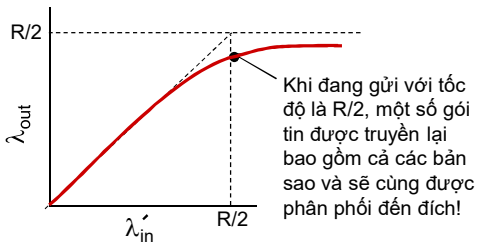


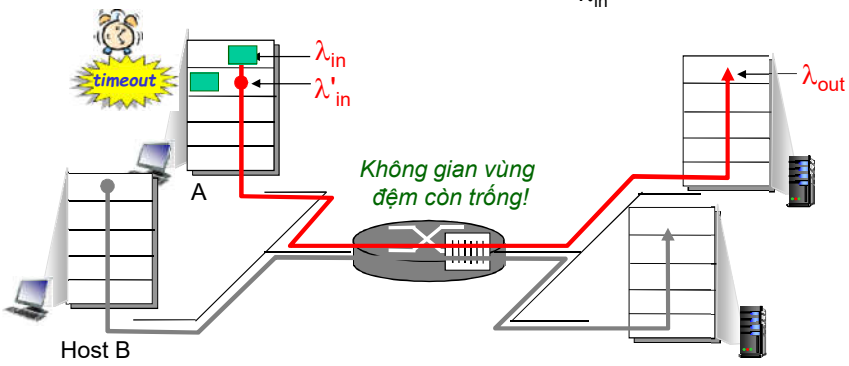


Các nguyên nhân/chi phí của tắc nghẽn: tình huống 2

Thực tế: các bản sao

- Các gói tin có thể bị mất, bị bỏ rơi tại bộ định tuyến nếu vùng đệm của nó bị đầy
- Nếu bên gửi timeout sớm, thì sẽ gửi đi *hai* bản sao của gói tin, và cả hai đều được phân phối đến đích

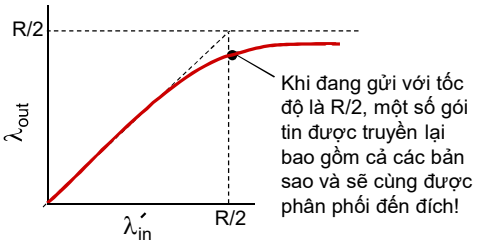




Các nguyên nhân/chi phí của tắc nghẽn: tình huống 2

Thực tế: các bản sao

- Các gói tin có thể bị mất, bị bỏ rơi tại bộ định tuyến nếu vùng đệm của nó bị đầy
- Nếu bên gửi timeout sớm, thì sẽ gửi đi *hai* bản sao của gói tin, và cả hai đều được phân phối đến đích



"Chi phí" của tắc nghẽn:

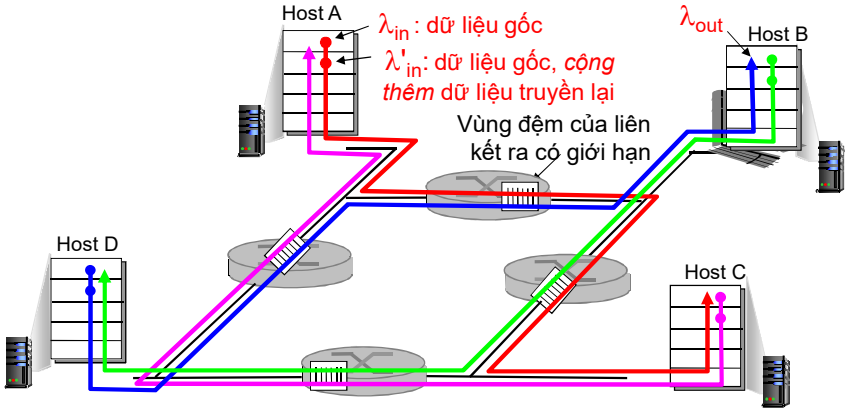
- Nhiều việc (truyền lại), với lưu lượng xác định
- Không cần thiết phải truyền lại: liên kết mang nhiều bản sao của gói tin
  - Làm giảm lưu lượng

Các nguyên nhân/chi phí của tắc nghẽn: tình huống 3

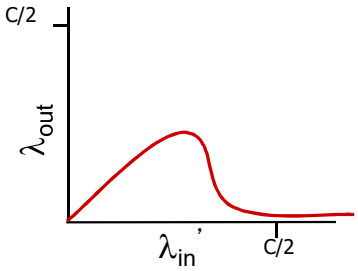
- Bốn bên gửi
- Nhiều đường đến đích
- timeout/truyền lại

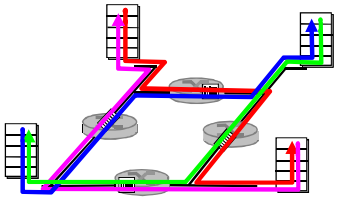
Hỏi: Điều gì sẽ xảy ra khi λ_{in} và λ_{in}' tăng lên?

Trả lời: Nếu λ_{in}' (đỏ) tăng lên, thì tất cả gói tin màu xanh nước biển đang đến tại hàng đợi phía trên sẽ bị bỏ rơi, thông lượng màu xanh nước biển sẽ tiến đến 0



Các nguyên nhân/chi phí của tắc nghẽn: tình huống 3





"Chi phí" khác của tắc nghẽn:

- Khi gói tin bị bỏ rơi, thì bất kỳ luồng lưu lượng truyền nào cho gói tin đều là lãng phí!

Phương pháp tiếp cận hướng tới điều khiển tắc nghẽn

Hai cách tiếp cận chính hướng tới điều khiển tắc nghẽn:

Điều khiển tắc nghẽn end-end:

- Không có phản hồi rõ ràng từ mạng
- Tắc nghẽn được suy ra từ hiện tượng mất mát hoặc trễ quan sát được tại hệ thống đầu cuối
- Cách tiếp cận này được thực hiện bởi TCP

Điều khiển tắc nghẽn có hỗ trợ từ mạng:

- Các bộ định tuyến cung cấp phản hồi tới các hệ thống đầu cuối.
  - bit đơn chỉ thị tắc nghẽn (SNA, DECbit, TCP/IP ECN, ATM)
  - Tốc độ gửi được xác định rõ ràng

Case study: điều khiển tắc nghẽn trong ATM ABR

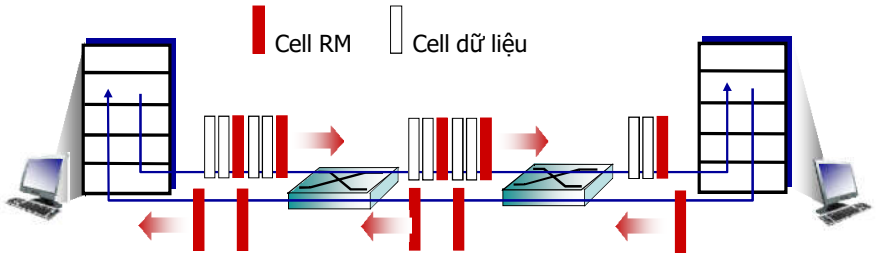
ABR: tốc độ bit có sẵn:

- "Dịch vụ mềm dẻo"
- Nếu đường dẫn phía bên gửi "dưới tải" thì:
  - Bên gửi nên dùng băng thông có sẵn
- Nếu đường dẫn bên gửi bị tắc nghẽn thì:
  - Bên gửi nên giảm để đảm bảo tốc độ là tối thiểu

Các cell RM (quản lý tài nguyên):

- Được gửi bởi bên gửi, xen kẽ với các cell dữ liệu
- Các bit trong cell RM được thiết lập bởi các switch (*"có hỗ trợ từ mạng"*)
  - *bit NI:* không tăng theo tốc độ (tắc nghẽn nhẹ)
  - *bit CI:* xác định tắc nghẽn
- Các cell RM được trả lại bên gửi từ bên nhận, với các bit còn nguyên vẹn

Case study: điều khiển tắc nghẽn trong ATM ABR



- Hai byte trường ER (explicit rate) trong cell RM
  - Switch bị tắc nghẽn có thể có giá trị ER thấp hơn trong cell
  - Bên gửi gửi với tốc độ được hỗ trợ lớn nhất trên đường truyền
- Bit EFCI trong các cell dữ liệu: được thiết lập là 1 trong switch bị tắc nghẽn
  - Nếu cell dữ liệu trước cell RM có EFCI được thiết lập, thì bên nhận thiết lập bit CI trong cell RM được trả về

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

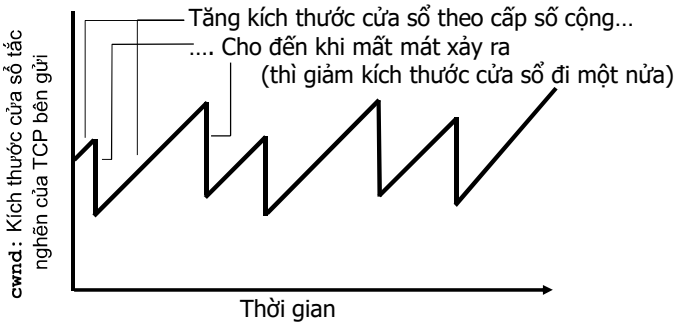
3.6 Các nguyên lý điều khiển tắc nghẽn

3.7 Điều khiển tắc nghẽn TCP

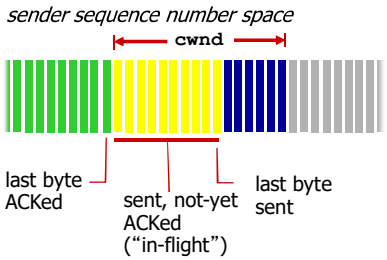
Điều khiển tắc nghẽn trong TCP: Tăng theo cấp số cộng Giảm theo cấp số nhân

- *Cách tiếp cận:* Bên gửi tăng tốc độ truyền (kích thước cửa sổ), thăm dò băng thông sử dụng, cho đến khi có mất mát xảy ra
  - *Tăng theo cấp số cộng:* tăng **cwnd** theo 1 MSS mỗi RTT cho đến khi phát hiện mất mát
  - *Giảm theo cấp số nhân*: giảm **cwnd** đi một nửa sau khi phát hiện có mất mát

Thăm dò băng thông



Chi tiết điều khiển tắc nghẽn trong TCP



- Bên gửi giới hạn việc truyền:

$$\texttt{LastByteSent} - \texttt{LastByteAcked} \leq \texttt{cwnd}$$

- **cwnd** thay đổi, có chức năng nhận biết tắc nghẽn trên mạng

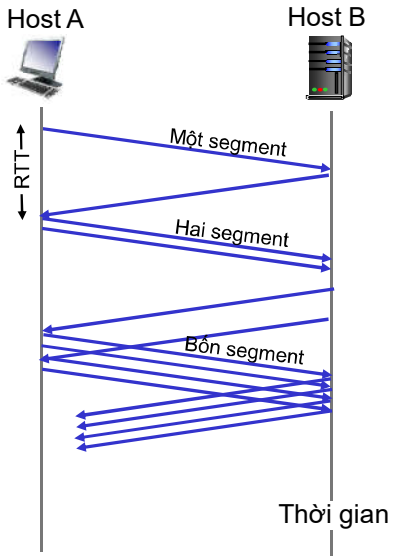
Tốc độ gửi của TCP:

- *Được hiểu là:* gửi cwnd byte, chờ một RTT cho ACK, sau đó gửi nhiều byte hơn

$$\text{Tốc độ} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

TCP khởi động chậm

- Khi kết nối bắt đầu, tăng tốc độ lên theo cấp số nhân cho đến khi có sự kiện mất mát đầu tiên xảy ra:
  - Khởi tạo **cwnd** = 1 MSS
  - Tăng gấp đôi **cwnd** cho mỗi RTT
  - Thực hiện tăng **cwnd** cho mỗi ACK nhận được
- *Tổng kết:* tốc độ khởi đầu là chậm nhưng sau đó tăng lên theo cấp số nhân



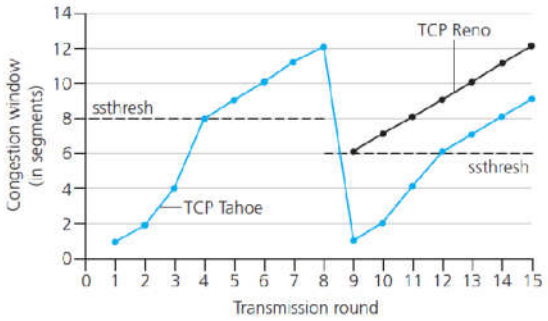
Phát hiện và phản ứng lại khi có mất mát

- Mất mát được xác định khi bị timeout:
  - **cwnd** được thiết lập lại là 1 MSS;
  - Cửa sổ sau đó sẽ tăng theo cấp số nhân (như trong khởi động chậm) tới ngưỡng, thì sẽ tăng tuyến tính
- Mất mát được xác định khi thấy 3 ACK trùng lặp: TCP RENO
  - Các ACK trùng lặp xác định khả năng truyền các segment của mạng
  - **cwnd** giảm đi một nửa kích thước cửa sổ, sau đó tăng tuyến tính
- TCP Tahoe luôn đặt **cwnd** là 1 (khi có timeout hoặc 3 ACK trùng lặp)

Hiện thực trong TCP

Hỏi: Khi nào nên chuyển từ tăng theo cấp số nhân sang tăng tuyến tính?

Trả lời: khi **cwnd** đạt đến 1/2 giá trị của nó trước khi timeout.



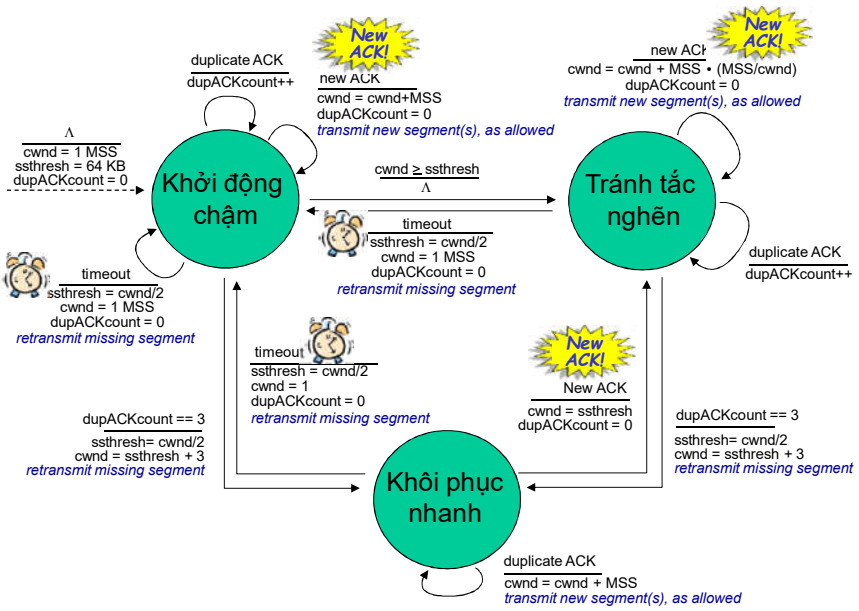
Cài đặt:

- Biến **ssthresh**
- Với mỗi sự kiện mất mát, **ssthresh** sẽ được đặt bằng 1/2 **cwnd** ngay trước khi có mất mát xảy ra

Tổng kết điều khiển tắc nghẽn trong TCP

- Khi **cwnd** dưới **ssthresh**, bên gửi đang trong giai đoạn khởi động chậm, kích thước cửa sổ tăng nhanh theo cấp số nhân.
- Khi **cwnd** trên **ssthresh**, bên gửi đang trong giai đoạn tránh tắc nghẽn, kích thước cửa sổ tăng nhanh theo cấp tuyến tính.
- Khi có 3 ACK trùng lặp xảy ra, **ssthresh** = **cwnd/2** và **cwnd** = **ssthresh**.
- Khi timeout xảy ra, **ssthresh** = **cwnd/2** và **cwnd=1 MSS**.

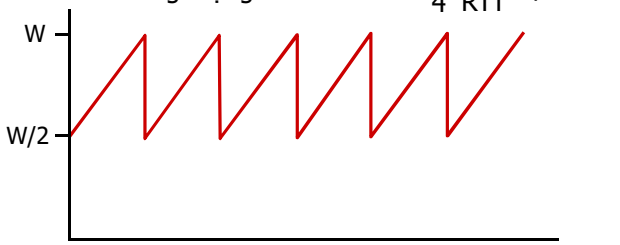
Tổng kết điều khiển tắc nghẽn trong TCP



Thông lượng của TCP

- Thông lượng trung bình của TCP được xác định qua kích thước cửa sổ và RTT như thế nào?
  - Bỏ qua khởi động chậm, giả sử dữ liệu luôn luôn được gửi
- W: kích thước cửa sổ (được tính bằng byte) khi có mất mát xảy ra
  - Kích thước cửa sổ trung bình (số byte trong lưu lượng) là ¾ W
  - Thông lượng trung bình là ¾ W trên RTT

$$\text{Thông lượng TB của TCP} = \frac{3}{4}\ \frac{W}{RTT}\ \text{bytes/sec}$$



TCP trong tương lai: TCP qua "đường truyền rộng và dài"

- Ví dụ: Các segment dài 1500 byte, RTT là 100ms, muốn đạt được thông lượng là10 Gbps
- Yêu cầu lưu lượng với kích thước cửa sổ là W = 83,333 segment
- Thông lượng của xác suất mất đoạn là L [Mathis 1997]:

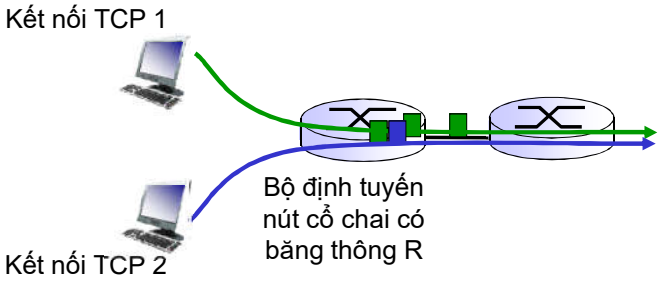
$$\text{Thông lượng TCP} = \frac{1.22 \cdot MSS}{RTT\sqrt{L}}$$

➜ Để có được thông lượng là 10 Gbps, cần tỷ lệ mất mát là $L = 2 \cdot 10^{-10}$ *– một tỷ lệ mất mát rất nhỏ!*

- Các phiên bản mới của TCP dành cho tốc độ cao

Tính công bằng trong TCP

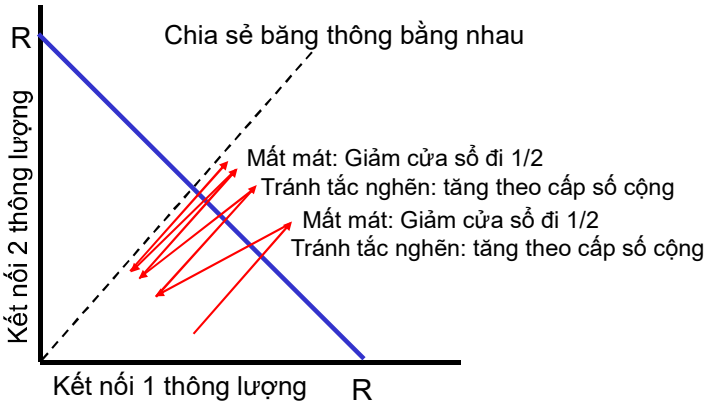
Mục tiêu: Nếu K phiên làm việc trong TCP chia sẻ cùng liên kết nút cổ chai có băng thông là R, thì mỗi phiên nên có tốc độ trung bình là R/K



Tại sao TCP là công bằng?

Hai phiên làm việc cạnh tranh nhau:

- Tăng theo cấp số cộng làm tăng lưu lượng liên tục
- Giảm theo cấp số nhân làm giảm lưu lượng tương ứng



Tính công bằng (tiếp)

Tính công bằng và UDP

- Các ứng dụng đa phương tiện thường không dùng TCP
  - Không muốn tốc độ bị chặn do điều khiển tắc nghẽn
- Thay bằng dùng UDP:
  - Gửi audio/video với tốc độ ổn định, chịu mất mát gói tin

Tính công bằng và kết nối song song trong TCP

- Ứng dụng có thể mở nhiều kết nối song song giữa hai host
- Các trình duyệt web làm theo cách này
- Ví dụ: liên kết có tốc độ R hỗ trợ 9 kết nối:
  - Ứng dụng mới yêu cầu 1 TCP, có tốc độ R/10
  - Ứng dụng mới yêu cầu 11 TCP, có tốc độ R/2

Chương 3: Tổng kết

- Các nguyên lý của các dịch vụ tầng giao vận:
  - Ghép kênh, phân kênh
  - Truyền dữ liệu tin cậy
  - Điều khiển luồng
  - Điều khiển tắc nghẽn
- Hiện thực trên mạng Internet:
  - UDP
  - TCP

Tiếp theo:

- Kết thúc các vấn đề liên quan đến "phần cạnh" của mạng (tầng ứng dụng và tầng giao vận)
- Chuẩn bị đi vào "phần lõi" của mạng