Tầng vận chuyển trong mạng Internet

Bởi:

unknown

Tầng vận chuyển trong mạng Internet

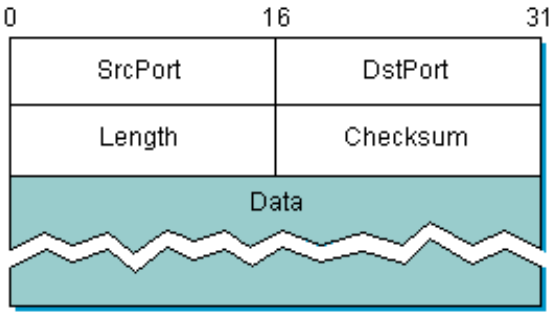
Trong Internet, tầng vận chuyển được thiết kế ra với ý đồ thực hiện các nhiệm vụ sau:

- 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.
- 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.

Tầng vận chuyển trong Internet cũng hỗ trợ hai phương thức hoạt động không nối kết và có nối kết với hai giao thức liên lạc tương ứng là UDP và 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, do đó 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. Khuôn dạng của chúng như sau:



Khuôn dạng của một segment UDP (H7.9)

Giải thích:

- 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: Tổng chiều dài của segment, tính luôn cả phần header.
- Checksum: Là phần kiểm tra lỗi. UDP sẽ tính toán phần kiểm tra lỗi tổng hợp trên phần header, phần dữ liệu và cả phần header ảo. Phần header ảo chứa 3 trường trong IP header: địa chỉ IP nguồn, địa chỉ IP đích, và trường chiều dài của UDP. Phương thức tính toán như sau:

```
u_short

cksum(u_short *buf, int count)

{

register u_long sum = 0;

while (count--)

{

sum += *buf++;

if (sum & 0xFFFF0000)

{

/* bit carry xuất hiện, vì thế gấp và cộng dồn nó lại */

sum &= 0xFFFF;

sum++;

}

}

return ~(sum & 0xFFFF);

}
```

Xem thông điệp là một chuỗi các số nguyên 16 bits. Cộng dồn các số nguyên này từng bit một. Kết quả cộng dồn cuối cùng chính là phần kiểm tra lỗi.

- Data: Phần dữ liệu hai bên gởi cho nhau.

UDP hoạt động không tin cậy cho lắm, 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.

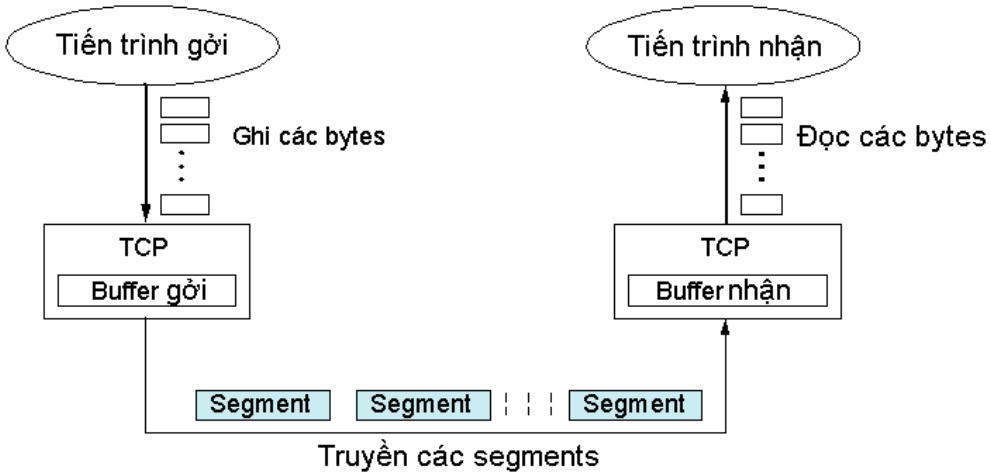
Giao thức TCP (Transmission Control Protocol)

Ngược với giao thức UDP, TCP là giao thức vận chuyển tinh vi hơn, dùng để 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 hai hướng đồng thời, nghĩa là mỗi một nối kết hỗ trợ hai luồng bytes chạy theo hai hướng. Nó cũng bào gồm một cơ chế điều khiển thông lượng cho mỗi luồng bytes này, để cho phép bên nhận giới hạn lượng dữ liệu mà bên gởi có thể truyền tại một thời điểm nào đó. TCP cũng hỗ trợ cơ chế đa hợp, cho phép nhiều tiến trình trên một máy tính có thể đồng thời thực hiện đối thoại với đối tác của chúng.

Hai đầu mút truyền dữ liệu với nhau như thế nào?

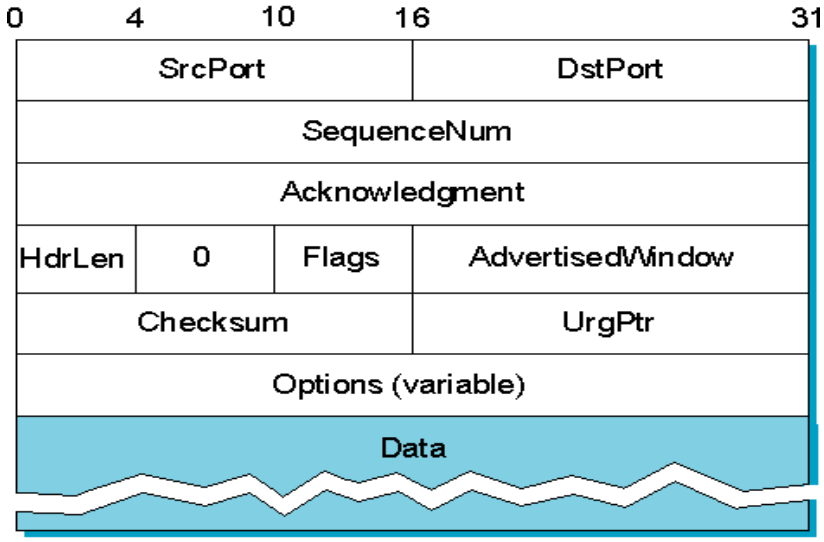
TCP là giao thức hướng byte, nghĩa là bên gởi ghi các bytes lên nối kết TCP, bên nhận đọc các bytes từ nối kết TCP đó. Mặc dù TCP mô tả dịch vụ mà nó cung cấp cho tầng ứng dụng là theo kiểu "luồng các bytes", nhưng tự thân TCP không truyền từng byte một qua mạng Internet. Thay vào đó, thực thể TCP trên máy nguồn trữ tạm đủ số bytes phát ra từ tiến trình gởi để tạo nên một gói tin có kích thước hợp lý rồi mới gởi gói tin đó đến thực thể TCP ngang hàng bên máy đích. Thực thể TCP bên máy đích sẽ bóc các bytes dữ liệu trong gói tin ra và đặt chúng vào buffer của nó. Tiến trình bên nhận từ đó có thể đọc các bytes từ buffer này tùy thích. Quá trình truyền nhận trên được mô phỏng trong Hình H7.10.



Cách thức TCP quản lý luồng các bytes (H7.10)

Khuôn dạng TCP Segment

Các gói tin được trao đổi bởi hai thực thể TCP trong Hình H7.10 được gọi là các *segment* (đoạn), do mỗi gói tin mang theo một đoạn của cả một luồng các bytes. Mỗi segment có một header như được chỉ ra trong Hình H7.11.



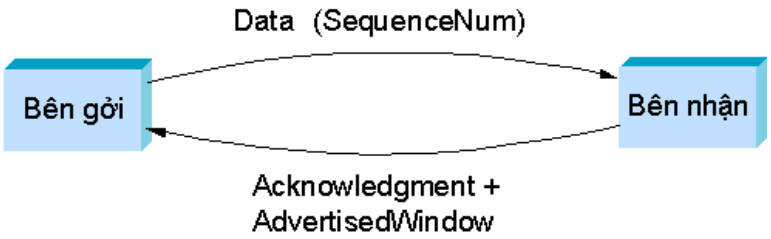
Khuôn dạng TCP header (H.11)

Giải thích:

- 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)

- Các trường *Acknowledgement*, *SequenceNum* và *AdvertisedWindow* tất cả được sử dụng trong giải thuật cửa sổ trượt của TCP. Bởi vì TCP là giao thức hướng byte, nên mỗi byte của dữ liệu sẽ có một số thứ tự; trường *SequenceNum* chứa số thứ tự của byte đầu tiên của một dãy các bytes chứa trong một segment. Các trường *Acknowledgement* và *AdvertisedWindow* được dùng để thông báo tiến độ nhận các bytes trong luồng dữ liệu và khả năng tiếp nhận chúng. Để đơn giản hóa vấn đề, chúng ta bỏ qua sự thật là dữ liệu có thể chạy theo hai chiều, ở đây ta đưa ra sơ đồ như sau: bên gởi sẽ gởi một segment dữ liệu, byte dữ liệu đầu tiên trong segment đó sẽ có số thứ tự là *SequenceNum*; bên nhận sẽ báo nhận bằng các trường *Acknowledgement* và *AdvertisedWindow*.



Gởi dữ liệu và báo nhận (H7.12)

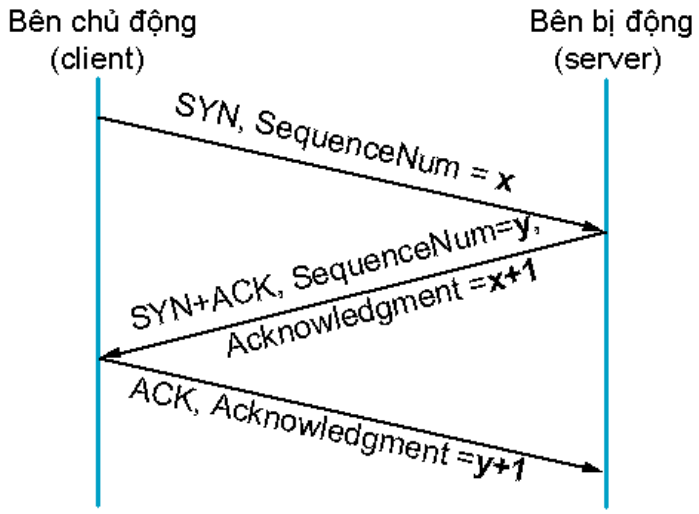
Cách thức hai bên sử dụng các trường trên như thế nào sẽ được trình bày trong phần điều khiển luồng dữ liệu.

- Trường *Flags* dài 6 bits được sử dụng để chứa thông tin điều khiển giữa hai bên sử dụng giao thức TCP. Một bit trong trường này là một cờ, cụ thể như sau: *SYN, FIN, RESET, PUSH, URG, ACK*. Hai cờ *SYN* và *FIN* được dùng để thiết lập và giải phóng nối kết. Cờ *ACK* được đặt mỗi khi trường *Acknowledgement* là hợp lệ. Cờ *URG* được dùng để đánh dấu segment này chứa dữ liệu khẩn cấp. Khi cờ này được đặt, trường *UrgPtr* sẽ chỉ ra nơi bắt đầu của dữ liệu không khẩn cấp (dữ liệu khẩn cấp luôn nằm ở đầu của phần dữ liệu). Cờ *PUSH* báo hiệu cho bên nhận rằng bên gởi đã dùng thao tác PUSH, tức là 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. Cuối cùng, 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.
- Trường *Checksum* được sử dụng chính xác giống như trong giao thức UDP.
- Do header của TCP có độ dài thay đổi, nên trường *HdrLen* sẽ chỉ ra độ dài cụ thể của phần header này.

Bắt tay trong TCP

TCP sử dụng giao thức bắt tay 3 chiều.

- *Bước 1*: Cient (bên chủ động) gởi đến server một segment yêu cầu nối kết, trong đó chứa số thứ tự khởi đầu mà nó sẽ dùng (*Flags = SYN, SequenceNum = x*).
- *Bước 2*: Server trả lời cho client bằng một segment, trong đó báo nhận rằng nó sẵn sàng nhận các byte dữ liệu bắt đầu từ số thứ tự x+1 (*Flags = ACK, Ack = x + 1*) và cũng báo rằng số thứ tự khởi đầu của bên server là y (*Flags = SYN, SequenceNum = y*).
- *Bước 3*: Cuối cùng client báo cho server biết, nó đã biết số thứ tự khởi đầu của server là y (*Flags = ACK, Ack = y+1*).

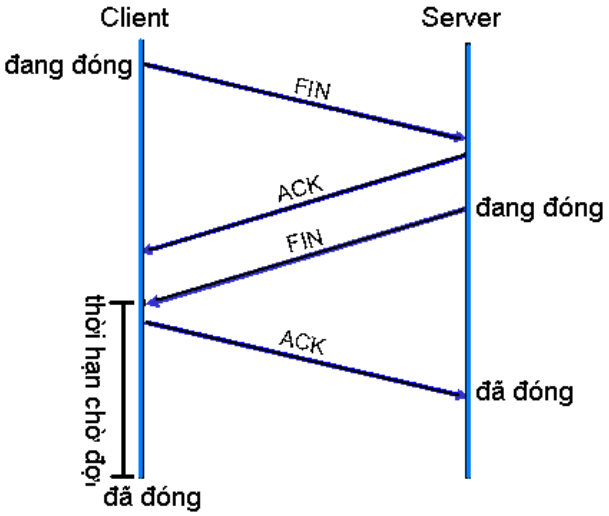


Bắt tay 3 chiều trong TCP (H7.12)

Hủy bắt tay trong TCP

Việc hủy bắt tay trong TCP được thực hiện qua 4 bước:

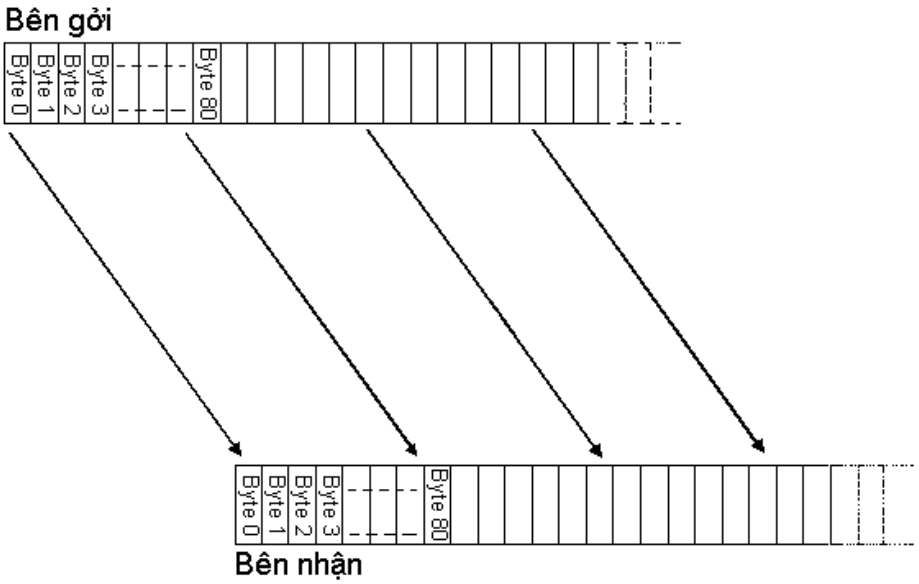
- *Bước 1*: Cient (bên chủ động) gởi đến server một segment yêu cầu hủy nối kết (*Flags = FIN*).
- *Bước 2*: Server nhận được một segment FIN, sẽ trả lời bằng một segment ACK. Sau khi đã hoàn tất hết mọi thứ để đóng nối kết, server sẽ gởi cho client tiếp một segment FIN.
- *Bước 3:* Client nhận được FIN sẽ trả lời ACK sau đó nó sẽ chuyển sang trạng thái chờ đợi có định hạn. Trong thời gian chờ đợi này, client sẽ trả lời ACK cho mọi khung FIN. Hết thời gian chờ đợi, client sẽ thật sự đóng nối kết.
- *Bước 4*: Server khi nhận được ACK sẽ thật sự đóng nối kết.



Hủy nối kết trong TCP (H7.13)

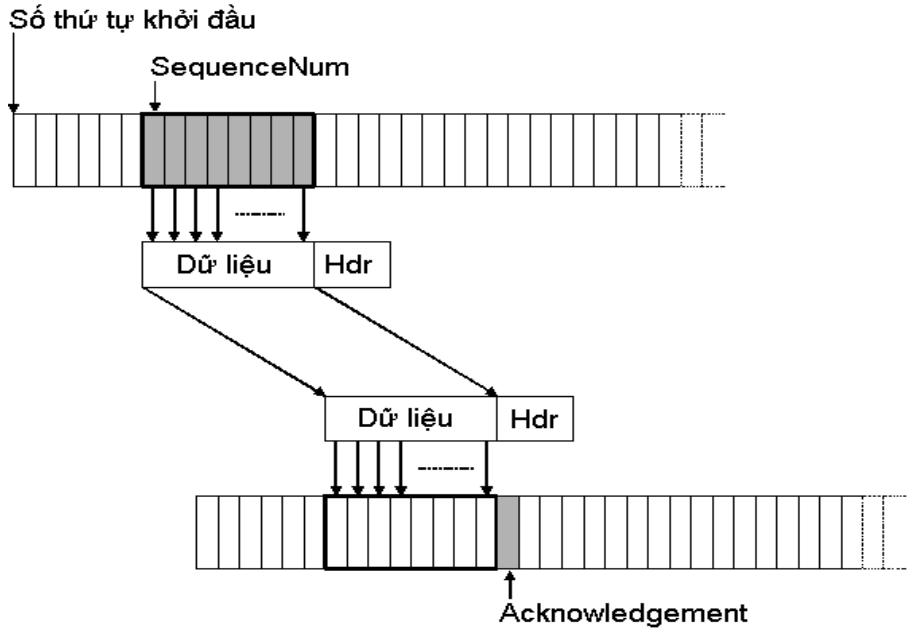
Điều khiển thông lượng trong TCP

TCP dùng phương pháp điều khiển thông lượng "cửa sổ trượt với kích thước cửa sổ động". Nhắc lại rằng TCP là giao thức hướng bytes. Ta có thể tưởng tượng hình ảnh sau: tiến trình bên gởi ghi ra một luồng các bytes, tiến trình bên nhận đọc vào một luồng các bytes.



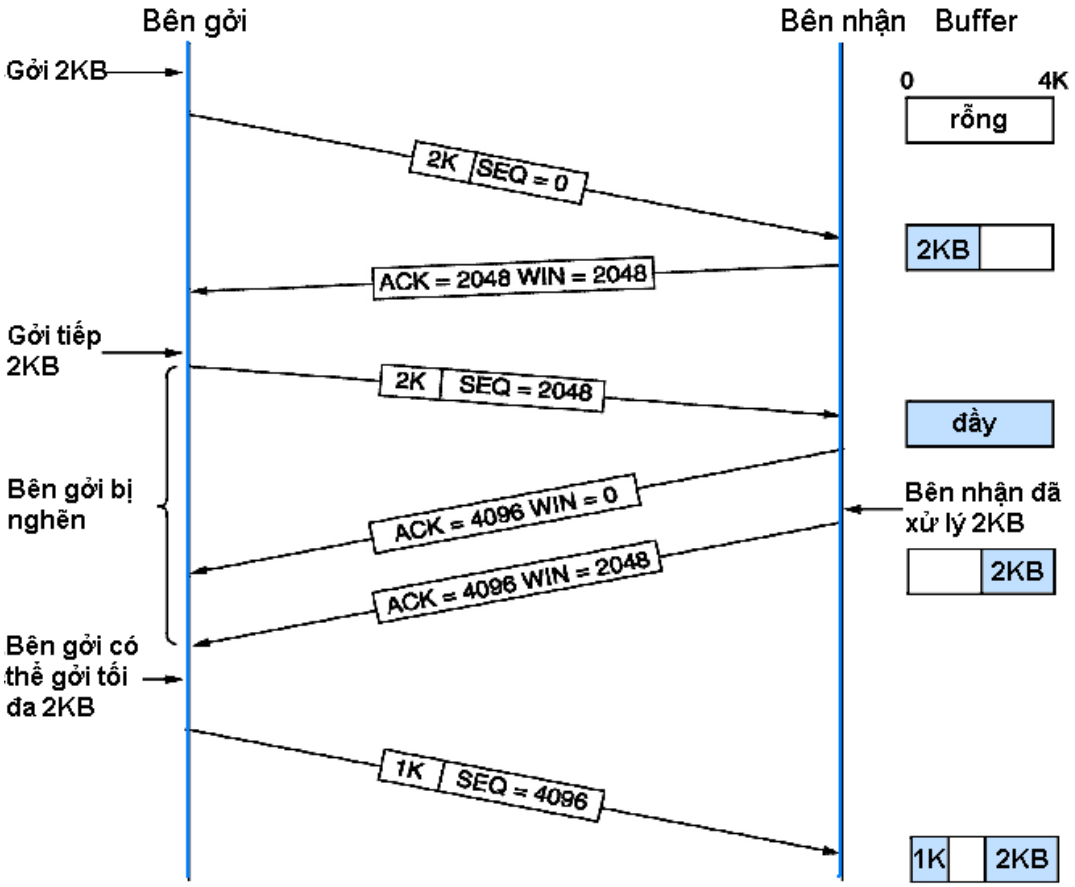
Truyền nhận theo luồng bytes (H7.14)

Như đã nói, TCP không truyền nhận dữ liệu cho ứng dụng từng byte một mà nó trữ tạm các bytes trong buffer đến khi đủ đóng gói thành một segment thì mới truyền đi.



Các luồng bytes được phân đoạn như thế nào (H7.15)

Byte đầu tiên của mỗi luồng bytes sẽ được đánh số bằng "số thứ tự khởi đầu", và số này được dàn xếp trong giai đoạn bắt tay 3 chiều. Trường *SequenceNum* trong TCP segment chứa số thứ tự của byte đầu tiên nằm trong segment đó. Cũng như trong giao thức cửa sổ trượt, khi bên nhận nhận được *n* bytes trong một segment, bắt đầu từ byte thứ *SequenceNum*, nó sẽ báo nhận tốt *n* bytes này và chờ nhận tiếp từ byte thứ *Acknowledgement* (Acknowledgement = SequenceNum + n).



Ví dụ về điều khiển thông lượng trong TCP (H7.16)

Do kích thước cửa sổ là động, nên trong mỗi khung báo nhận của mình, bên nhận đính kèm theo thông báo về kích thước cửa sổ sẵn dùng của nó (lượng buffer còn trống), đó chính là trường *AdvertisedWindow* trong TCP segment. Lần sau, bên gởi sẽ không được gởi lượng bytes vượt quá *AdvertisedWindow*.

Trong ví dụ trên, lúc đầu bên nhận có kích thước buffer là 4 KB. Bên gởi đặt số thứ tự khởi đầu là 0, sau đó truyền 2 KB. Buffer bên nhận còn lại 2 KB rỗng, do đó nó báo nhận "*Acknowledgement = 2048, AdvertisedWindow = 2048"*. Bên gởi gởi tiếp 2 KB, khi đó buffer bên nhận bị đầy, nó liền báo nhận "*Acknowledgement = 4096, AdvertisedWindow = 0"*. Không còn buffer nhận, nên bên gởi sẽ tạm thời bị nghẽn. Sau khi bên nhận xử lý xong 2 KB, nó liền báo "*Acknowledgement = 4096, AdvertisedWindow = 2048*". Lúc này bên gởi có thể gởi tiếp tối đa là 2 KB, nhưng nó chỉ còn 1 KB dữ liệu để gởi mà thôi.