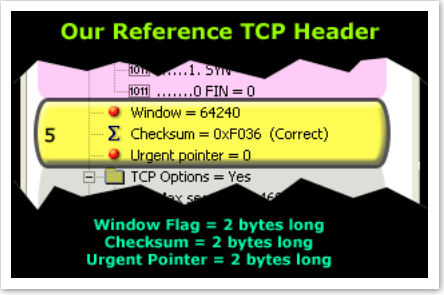
TCP Window Size, Checksum & Urgent Pointer

Giới thiệu

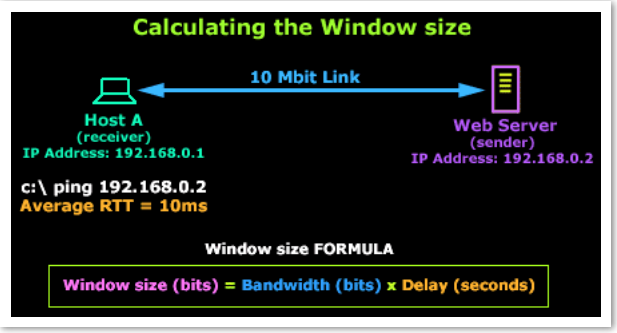
Bài này tôi sẽ giới thiệu một vài trường rất thú vị được sử dụng trong giao thức vận chuyển TCP. Chúng ta có thể thấy được làm thế nào để TCP giúp điều khiển dữ liệu được truyền đi trên mỗi Segment, đảm bảo không có lỗi trên từng Segment và cuối cùng, cắm cờ "khẩn cấp" cho dữ liệu của chúng ta để đảm bảo nó có được quyền ưu tiên khi đến được với người nhận.



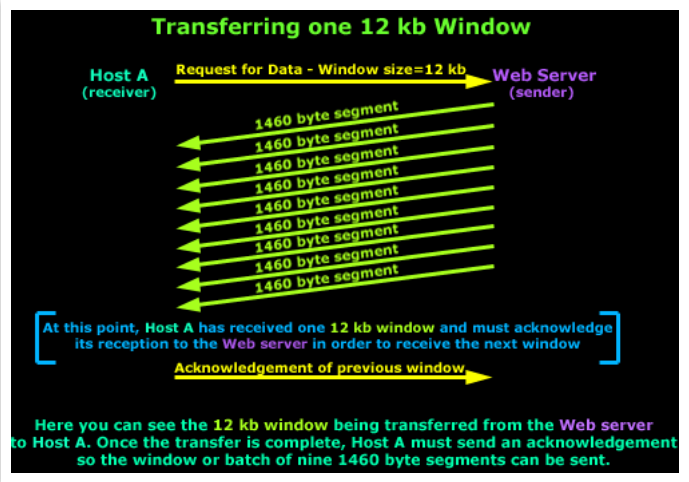
Các trường mà chúng ta đang chuẩn bị phân tích ở đây chiếm tổng cộng khoảng 6 bytes trong TCP Header.
Những giá trị này, như hầu hết các trường trong Header của giao thức, đều giữ nguyên kích thước bất kể dữ liệu của ứng dụng có lớn như nào đi chăng nữa.

The Window Flag

Window Size được xem như là 1 trong những trường quan trọng nhất trong TCP Header. Trường này được sử dụng bởi người nhận để chỉ ra rằng người gửi chỉ được gửi lượng dữ liệu là bao nhiêu. Bất kể người gửi hoặc người nhận là ai, thì trường này luôn tồn tại và được sử dụng.
Bạn sẽ nhận ra rằng bài này phần nhiều sẽ nói về trường Window Size, bởi vì trường này là 1 trường rất quan trọng. Trường Window Size là chìa khóa cho việc truyền dữ liệu 1 cách hiệu quả và điều khiển luồng.
Trường Window Size sử dụng đơn vị đo là byte. Khi nhìn vào hình ở trên, thì ta có thể thấy được 64240 tương đương với 64240 bytes, hoặc khoảng 62.7 kb.
62.7 kb chính là lượng dữ liệu mà người nhận có khả năng nhận được, trước khi chuyển cho người gửi (Server) một giá trị Window Size mới. Khi lượng dữ liệu được truyền bằng giá trị Window Size hiện tại, người gửi sẽ mong đợi 1 giá trị Window Size mới từ người nhận, cùng với 1 báo nhận cho giá trị Window Size vừa nhận được.
Quá trình trên là cần thiết để duy trì việc truyền dữ liệu 1 cách hoàn hảo và đạt hiệu quả cao. Tuy nhiên chúng ta nên lưu ý rằng trường Window Size được chọn không phải là giá trị ngẫu nhiên ở trong mọi trường hợp, mà nó có thể được tính toán sử dụng những công thức đặc biệt như trong ví dụ dưới đây:



Trong ví dụ này, máy A kết nối tới 1 Server qua 1 đường truyền 10 Mbits. Qua công thức, để tính toán ra giá trị Window Size tốt nhất chúng ta cần những thông tin sau: **băng thông** và **độ trễ**. Chúng ta biết được băng thông của đường truyền: 10,000,000 bits (10 Mbits), và chúng ta có thể dễ dàng tìm ra độ trễ bằng cách 'ping' thử từ máy A tới Server mà cho chúng ta kết quả như trong hình là 10 ms hay là 0.01 s.
Sau đó chúng ta có thể sử dụng thông tin này để tính ra được giá trị Window Size hiệu quả nhất (WS):
WS = 10,000,000 x 0.01 = 100,000 bits hoặc là (100,000/8)/1024 = 12.5 kbytes
Với băng thông 10Mbps và độ trễ 0.01s cho chúng ta Window Size vào khoảng 12 kb hay là 1460 bytes Segment:



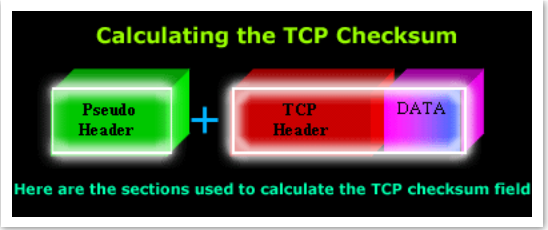
Nó sẽ mang lại thông lượng tối đa cho 1 đường LAN 10 Mbps, ngay cả khi độ trễ cao đến 10 ms bởi vì thông thường hầu hết các mạng LAN chỉ có độ trễ khoảng vài ms. Khi băng thông thấp hơn, thì độ trễ phải cao hơn mới cho ra được Window Size tương tự, bởi vậy 1 giá trị Window Size khoảng 12 kb cũng có thể hoạt động tốt ở tốc độ thấp hơn.

Windowing - một hình thức điều khiển luồng

Ngoài việc là nhân tố chính trong việc truyền dữ liệu 1 cách hiệu quả thì Windowing cũng là 1 hình thức của điều khiển luồng (người nhận có thể chỉ định với người gửi lượng dữ liệu mà người nhận có thể chấp nhận là bao nhiêu).
Sự thật là trong hầu hết các trường hợp, 62 kbytes là giá trị mặc định được sử dụng cho Window Size. Ngoài ra, mặc dù Window Size đã được chọn bởi người nhận là 62 kbytes, đường truyền liên tục được theo dõi sự mất mát gói tin hay độ trễ trong quá trình truyền dữ liệu, dẫn đến sự tăng nhẹ hay giảm nhẹ Window Size ban đầu để tối ưu hóa việc sử dụng băng thông và thông lượng dữ liệu.
Cơ chế tự động sửa lỗi này đảm bảo rằng cả 2 máy đều sẽ cố gắng để liên kết với nhau 1 cách tốt nhất có thể, nhưng không phải lúc nào cũng luôn thành công. Điều này thường là lý do tại sao người dùng có thể tự thay đổi Window Size cho đến khi tìm thấy giá trị tốt nhất và điều này phụ thuộc vào đường truyền giữa các máy và độ trễ của nó.
Trong trường hợp Window Size bằng 0, thì remote TCP sẽ không thể gửi dữ liệu nữa. Nó phải đợi cho đến khi không gian bộ đệm sẵn có và nhận được 1 gói thông báo Window Size khác 0.
Cuối cùng, đối với những người hay làm việc với Router của Cisco, bạn có thể cấu hình Window Size đối với những Router có IOS v9 trở lên. Router với các phiên bản từ 12.2(8)T trở lên hỗ trợ Window Scaling, 1 tính năng được tự động kích hoạt cho Window Size trên 65,535 và giá trị tối đa là 1,073,741,823 bytes.
Đối với Server: Window Size lớn hơn = nhiều bộ nhớ hơn.
Bây giờ chúng ta sẽ phân tích tại sao Window Size ảnh hưởng đến những Web Server mà có hàng ngàn khách kết nối tới chúng và yêu cầu dữ liệu.
Khi 1 khách hàng truy cập đến 1 Web Server, Server được yêu cầu dự trữ 1 lượng nhỏ bộ nhớ (RAM) dành cho phiên làm việc của khách hàng. Lượng bộ nhớ cần thiết là tương tự với Window Size, và như chúng ta đã thấy, giá trị này phụ thuộc vào băng thông và độ trễ giữa client và Server.
Để cho rõ ràng hơn, chúng ta cùng xem qua ví dụ sau:
Nếu bạn có 1 Web Server phục vụ 10,000 khách hàng trên 1 mạng LAN với băng thông 100 Mbits và độ trễ 0.1 giây và muốn tối đa hiệu suất trong việc truyền file, theo công thức của chúng tôi, bạn sẽ cần phải phân bổ 1 cửa sổ khoảng 1.25 MB cho mỗi khách hàng, hoặc 12 Gigs bộ nhớ cho tất cả các khách hàng của bạn. Tất nhiên giả sử là tất cả 10,000 khách hàng kết nối tới Web Server của bạn cùng lúc.
Để hỗ trợ việc truyền 1 tập tin lớn ở cả 2 hướng (từ máy chủ cho khách hàng và ngược lại), Server của bạn sẽ cần: [(100,000,000 x 0.1) 10,000 x 2] = hơn 24 Gigs bộ nhớ chỉ dành riêng cho socket buffers!
Vì vậy, bạn có thể thấy tầm quan trọng của nó là để cho các khách hàng không sử dụng giá trị Window quá cỡ. Trên thực tế, các tiêu chuẩn TCP hiện tại yêu cầu người nhận phải có khả năng chấp nhận giá trị full window của dữ liệu tại mọi thời điểm. Nếu máy nhận vượt quá không gian bộ đệm của nó, nó sẽ phải bỏ đi gói tin đang đến. Người gửi sẽ nhận ra sự mất mát gói tin và gọi cơ chế kiểm soát tắc nghẽn TCP mặc dù mạng không bị tắc nghẽn.

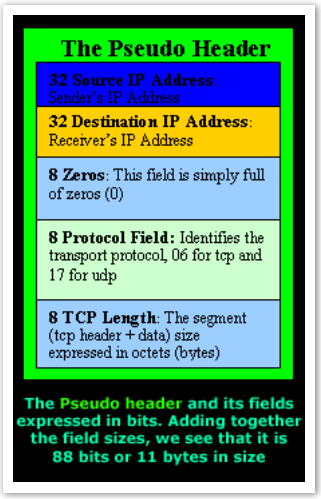
Checksum Flag

Trường TCP Checksum được tạo ra để đảm bảo rằng dữ liệu chứa trong TCP Segment đến đích chính xác và không có lỗi. Sẽ có người băn khoăn rằng làm thế nào để TCP sẽ đảm bảo Segment này đến đích chính xác -> là do có 1 ít thông tin được sử dụng ngoài TCP Header để tính toán Checksum và nó sẽ bao gồm 1 phần của IP Header.
Mẫu thông tin được thêm vào này được gọi là Header "giả" (Pseudo Header). Bây giờ, chúng ta hãy xem hình ảnh của các bộ phận được sử dụng để tính toán TCP Checksum:

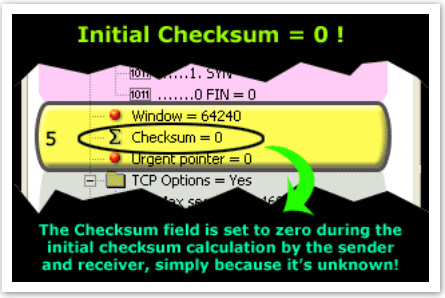


Mô hình trên cho ta thấy có Header "giả", tiếp theo là TCP Header và dữ liệu chứa trong Segment. Tuy nhiên, hãy nhớ rằng Header "giả" được bao gồm trong việc tính toán Checksum để đảm bảo Segment này đã đến được với người nhận chính xác.
Bây giờ chúng ta sẽ đi vào phân tích Header "giả".

Pseudo Header (Header "giả")
Header "giả" là 1 sự kết hợp của 5 trường khác nhau, được sử dụng trong quá trình tính toán Checksum. Điều quan trọng cần lưu ý là các Header giả không được truyền cho người nhận, chỉ đơn giản là tham gia vào việc tính toán Checksum.
Sau đây là 5 trường được định nghĩa bởi TCP RFC:



Khi Segment này đến đích và được xử lý thông qua các lớp OSI, một khi tới tầng vận chuyển, người nhận sẽ tái tạo lại Header "giả" để tính toán lại TCP Header Checksum và so sánh kết quả với giá trị được lưu trữ trong Segment đã nhận được.
Nếu chúng ta giả định Segment bằng 1 cách nào đó có thể tìm đường tới 1 máy có địa chỉ không chính xác, khi Header "giả" được tái tạo, địa chỉ IP sai sẽ được chèn vào trường Destination Address và kết quả Checksum được tính toán không chính xác.
Bây giờ bạn biết được làm thế nào trường Checksum đảm bảo rằng máy chính xác sẽ nhận được gói tin, hoặc sẽ nhận được gói tin đó mà không có bất kỳ lỗi nào.
Cuối cùng, trong quá trình tính toán TCP Header Checksum, trường này được thiết lập là 0 như hình dưới đây. Hành động này chỉ được thực hiện trong quá trình tính toán Checksum ở trên 2 đầu bởi vì nó không được biết vào thời điểm đó. Một khi giá trị được tính toán, sau đó nó sẽ được chèn vào trường, thay thế giá trị khởi đầu (0).

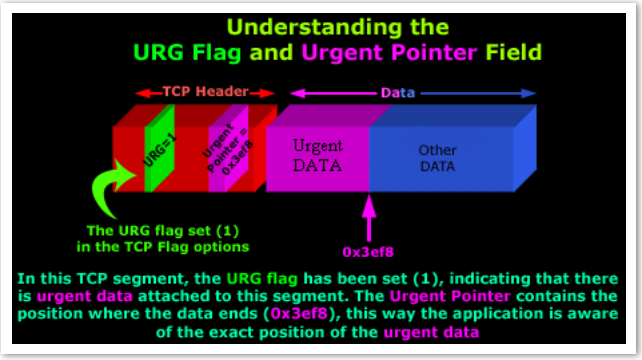


Người gửi chuẩn bị các Segment để gửi đến người nhận. Checksum được thiết lập là 0, trên thực tế là 4 số 0 (hex) hoặc 8 số 0 (0000 0000) nếu là hệ nhị phân, vì Checksum là 1 trường 8 bits.
Checksum sau đó được tính toán bằng cách sử dụng Header "giả", TCP Header và cuối cùng là dữ liệu được gán với Segment cụ thể. Kết quả sau đó được lưu trữ trong trường Checksum và Segment này được gửi.
Segment này đến với người nhận và được xử lý. Khi nó đến tầng thứ 4 của mô hình OSI, trường Checksum một lần nữa được thiết lập lại là 0. Người nhận sau đó sẽ tạo ra Header "giả" riêng của mình cho Segment vừa nhận được bằng cách nhập địa chỉ IP của mình vào trường Destination Address và sử dụng TCP Header và dữ liệu để tính toán Checksum mới.
Nếu tất cả được thực hiện thành công, kết quả sẽ giống với 1 trong các trường Checksum mà Segment có khi nó được chuyển đến. Khi điều này xảy ra, gói dữ liệu sau đó sẽ tiếp tục được xử lý và dữ liệu được giao cho các ứng dụng đang

chờ nó.
Tuy nhiên, nếu Checksum là khác nhau, sau đó gói tin sẽ bị drop và 1 thông báo sẽ được gửi đến người nhận tùy thuộc vào TCP Stack được thực hiện trên hệ điều hành của người nhận.

The Urgent Pointer

Cờ Urgent Pointer trong TCP Flag cho phép chúng ta đánh dấu 1 Segment của dữ liệu là khẩn cấp, trong khi trường Urgent Pointer xác định chính xác nơi mà các dữ liệu khẩn cấp kết thúc.
Để hiểu rõ điều này, hãy nhìn vào mô hình sau:



Bạn cũng cần biết rằng Urgent Pointer cũng được sử dụng để tấn công các máy chủ từ xa. Từ 1 số nghiên cứu, chúng ta có thể thấy rằng 1 số ứng dụng nhất định để bảo vệ hệ thống của bạn từ những nỗ lực tấn công, sẽ không thể ghi lại các cuộc tấn công nếu như cờ URG được bật.

Lược dịch từ bài gốc: Click Here

Bài tiếp theo: Analysing TCP Header Options

Người đăng: Unknown
Nhãn: TCP Protocol

Không có nhận xét nào:

Đăng nhận xét

Nhập nhận xét

Bài đăng Mới hơn Trang chủ Bài đăng Cũ hơn

Đăng ký: Đăng Nhận xét (Atom)