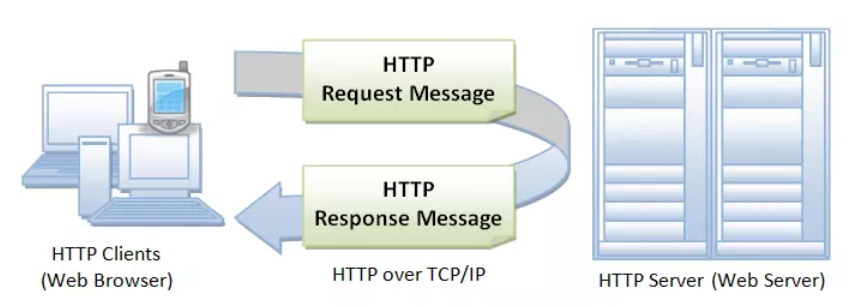
**Phần 1: Tìm hiểu về giao thức HTTP**

1. **Khái niệm giao thức HTTP**

* HTTP (HyperText Transfer Protocol - Giao thức truyền tải siêu văn bản) là một trong các giao thức chuẩn về mạng Internet, được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client), là giao thức Client/Server dùng cho World Wide Web – WWW
* HTTP là một giao thức ứng dụng của bộ giao thức TCP/IP (các giao thức nền tảng cho Internet).

1. **Cách thức hoạt động của HTTP**



* HTTP hoạt động dựa trên mô hình Client – Server. Trong mô hình này, các máy tính của người dùng sẽ đóng vai trò làm máy khách (Client). Sau một thao tác nào đó của người dùng, các máy khách sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi câu trả lời từ những máy chủ này.
* HTTP là một stateless protocol. Hay nói cách khác, request hiện tại không biết những gì đã hoàn thành trong request trước đó.
* HTTP cho phép tạo các yêu cầu gửi và nhận các kiểu dữ liệu, do đó cho phép xây dựng hệ thống độc lập với dữ liệu được truyển giao.

1. **Các thành phần chính của giao thức HTTP**

**HTTP – Requests**

* HTTP Request Method: Là phương thức để chỉ ra hành động mong muốn được thực hiện trên tài nguyên đã xác định.
* Cấu trúc của một HTTP Request:
* Một Request-line = Phương thức + URI–Request + Phiên bản HTTP . Giao thức HTTP định nghĩa một tập các giao thức GET, POST, HEAD, PUT ... Client có thể sử dụng một trong các phương thức đó để gửi request lên server.
* Có thể có hoặc không các trường header
* Một dòng trống để đánh dấu sự kết thúc của các trường Header.
* Request Header Fields: Các trường header cho phép client truyền thông tin bổ sung về yêu cầu, và về chính client, đến server. Một số trường: Accept-Charset, Accept-Encoding, Accept-Language, Authorization, Expect, From, Host, …
* Tùy chọn một thông điệp

**HTTP – Responses**

Cấu trúc của một HTTP response:

* Một Status-line = Phiên bản HTTP + Mã trạng thái + Trạng thái
* Có thể có hoặc không có các trường header
* Một dòng trống để đánh dấu sự kết thúc của các trường header
* Tùy chọn một thông điệp

Mã trạng thái: Thông báo về kết quả khi nhận được yêu cầu và xử lí bên server cho client.

**Phần 2:** **Tìm hiểu về giao thức websocket**

1. **Khái niệm Socket**

* Socket là gì? – Socket là một điểm cuối (end-point) của liên kết giao tiếp hai chiều (two-way communication) giữa hai chương trình chạy trên mạng. Nghĩa là một socket được sử dụng để cho phép 1 process nói chuyện với 1 process khác.
* Các lớp Socket được sử dụng để tiến hành kết nối giữa client và server. Nó được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới.

1. **Cách thức hoạt động Socket**

* Socket giúp lập trình viên kết nối các ứng dụng để truyền và nhận dữ liệu trong môi trường có kết nối Internet bằng cách sử dụng phương thức TCPIP và UDP.
* Khi cần trao đổi dữ liệu cho nhau thì 2 ứng dụng cần phải biết thông tin IP và port bao nhiêu của ứng dụng kia.
* Có rất nhiều dạng socket khác nhau phụ thuộc vào sự khác biệt giữa cách truyền dữ liệu (protocol). Dạng phổ biến nhất là TCP và UDP.

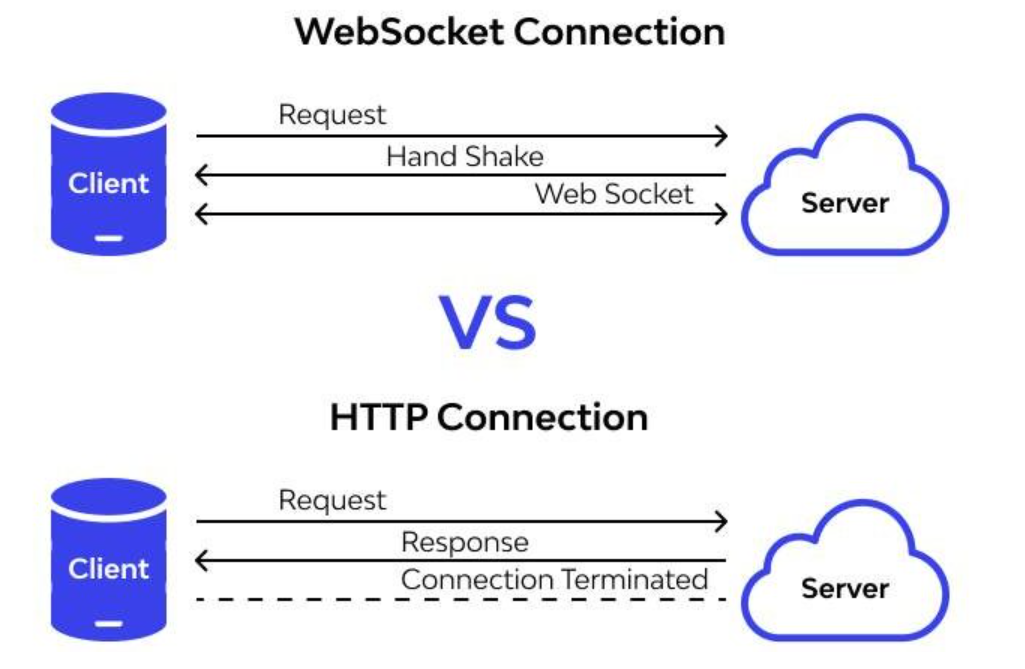
1. **Khái niệm về Websocket**

* Websocket là giao thức hỗ trợ giao tiếp hai chiều giữa client và server để tạo một kết nối trao đổi dữ liệu. Giao thức này không sử dụng HTTP mà thực hiện nó qua TCP. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, lập trình viên vẫn có thể đưa chúng vào bất kì loại ứng dụng nào.

1. **Ứng dụng Websocket**

* WebSocket cung cấp giao thức giao tiếp hai chiều mạnh mẽ. No có độ trễ thấp và dễ xử lý lỗi. Websocket thường được sử dụng cho những trường hợp yêu cầu real time như chat, hiển thị biểu đồ hay thông tin chứng khoán.

**Phần 3: So sánh HTTP và Websocket**



* WebSocket là một giao thức hoạt động hai chiều có hiệu suất hoạt động nhanh hơn, còn HTTP là giao thức hoạt động một chiều dựa trên TCP có hiệu suất hoạt động chậm hơn.
* WebSocket hỗ trợ truyền dữ liệu liên tục hỗ trợ quá trình tạo các ứng dụng thời gian thực, còn HTTP không liên tục được sử dụng để phát triển các ứng dụng RESTful/SOAP.
* WebSocket sử dụng kết nối TCP và luôn duy trì hoạt động cho đến khi có một bên ngừng kết nối, còn HTTP được thiết kế một kết nối riêng biệt có khả năng tự động ngắt kết nối sau khi yêu cầu được hoàn tất.

**Phần 4: Kết luận**

**Nên sử dụng WebSocket trong trường hợp :**

Có thể nói rằng, WebSocket là một công nghệ “xịn” có độ trễ rất thấp và phù hợp cho những dự án, những tác vụ cần phải nhanh ví dụ như:

* Ứng dụng theo thời gian thực
* Game Online
* Ứng dụng chat

**Ứng dụng theo thời gian thực**

* Tất nhiên, ứng dụng được ưu tiên hàng đầu để sử dụng các công nghệ tân tiến, có mức độ cập nhật dữ liệu nhanh chóng từ máy trạm đến máy khách và ngược lại. Các ứng dụng thời gian thực sẽ sử dụng WebSocket để hiện thị ở máy khách một cách liên tục nhờ vào các máy chủ phu hỗ trợ. WebSocket sẽ liên tục truyền/ đẩy dữ liệu đi trong cùng 1 kết nối đã mở. Vì thế, tốc độ của ứng dụng được cải thiện hơn rất nhiều.
* Một số ví dụ điển hình như: Sàn giao dịch Bitcoin, chứng khoán,… Những trang web, ứng dụng về các ngành này sẽ cần phải cập nhật thông với tốc độ tối đa nhất nhằm hỗ trợ người dùng có thể nhanh chóng bán ra hoặc mua vào.

**Game Online**

* Trong các trò chơi điện tử, đặc biệt là các game online, việc kết xuất hình ảnh và cập nhật dữ liệu trong thời gian thực là một trong những yếu tố “sống còn”. Ví dụ, bạn đang chơi một trò chơi bắn súng sinh tồn, bạn thấy một khẩu súng “xịn” trước mắt, bạn nhặt khẩu súng lên để hạ những người chơi khác. Nhưng độ trễ không được tối ưu sẽ khiến bạn bị người chơi khác bắn hạ khi bạn còn chưa kịp làm gì.

**Ứng dụng chat**

* Với các ứng dụng chat, nhắn tin, độ trễ sẽ không cần thiết phải tối ưu hoá tốt nhất trong thời gian thực. Tuy nhiên, những ứng dụng nhắn tin, chat luôn có một độ trễ nhất định có thể chấp nhận được.
* Nếu bạn đang phát triển các ứng dụng nhắn tin hay làm việc nội bộ như Bitrix24, bạn nên tham khảo công nghệ WebSocket để ứng dụng vào phần mềm của mình nhé!