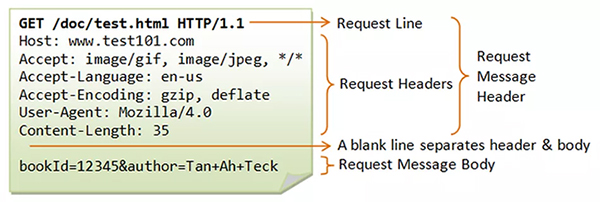
**Phần 1: Giao thức HTTP:**

* Khái niệm: **HTTP** là từ viết tắt của Hyper Text Transfer Protocol nghĩa là Giao thức Truyền tải Siêu Văn Bản được sử dụng trong www. **HTTP** là 1 giao thức cho phép tìm nạp tài nguyên, chẳng hạn như HTML doc. Nó là nền tảng của bất kỳ sự trao đổi dữ liệu nào trên Web và cũng là giao thức giữa client và server.
* **HTTP** là một giao thức lớp ứng dụng được xây dựng dựa trên TCP, sử dụng mô hình giao tiếp giữa máy khách (client) và máy chủ (server). Máy khách và máy chủ HTTP giao tiếp thông qua các tin nhắn yêu cầu và phản hồi. 3 loại thông báo HTTP chính là GET, POST và HEAD.

**+ HTTP GET:** Tin nhắn được gửi đến máy chủ chỉ chứa một URL. Không hoặc nhiều tham số dữ liệu tùy chọn có thể được thêm vào cuối URL. Máy chủ xử lý phần dữ liệu tùy chọn của URL, nếu có và trả kết quả (một trang web hoặc thành phần của trang web) cho trình duyệt.

**+ HTTP POST:** Tin nhắn đặt bất kỳ tham số dữ liệu tùy chọn nào trong phần thân của thông báo yêu cầu, thay vì thêm chúng vào cuối URL.

**+ HTTP HEAD:** Các yêu cầu hoạt động giống như GET. Thay vì trả lời với đầy đủ nội dung của URL, máy chủ chỉ gửi lại thông tin tiêu đề (có trong phần HTML).



+ Trình duyệt bắt đầu giao tiếp với máy chủ HTTP bằng cách bắt đầu kết nối TCP đến máy chủ. Các phiên duyệt web sử dụng cổng máy chủ 80 theo mặc định, mặc dù các cổng khác như 8080 đôi khi cũng được sử dụng thay thế. Sau khi phiên được thiết lập, bạn kích hoạt gửi và nhận tin nhắn HTTP bằng cách truy cập trang web.

* Các thành phần chính của HTTP:

+ **HTTP Request Method**: Là phương thức để chỉ ra hành động mong muốn được thực hiện trên tài nguyên đã xác định.

+ **HTTP – Responses**

**Phần 2: Giao thức websocket**

* **Khái niệm:** Socket là điểm cuối end-point trong liên kết truyền thông hai chiều (two-way communication) biểu diễn kết nối giữa Client – Server. Các lớp Socket 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.
* **Cách thức hoạt động socket:** kết nối giữa client và server thông qua TCP/IP và UDP để truyền và nhận giữ liệu qua Internet. Giao diện lập trình ứng dụng mạng này chỉ có thể hoạt động khi đã có thông tin về thông số IP và số hiệu cổng của 2 ứng dụng cần trao đổi dữ liệu cho nhau. 2 ứng dụng cần truyền thông tin phải đáp ứng điều kiện sau thì socket mới có thể hoạt động:

+ 2 ứng dụng có thể nằm cùng trên một máy hoặc 2 máy khác nhau

+ Trong trường hợp 2 ứng dụng cùng nằm trên một máy, số hiệu cổng không được trùng nhau.

* **Khái niệm Websocket:** là công cụ hỗ trợ việc kết nối qua lại trên internet giữa client và server. Giúp diễn ra nhanh chóng và hiệu quả hơn thông qua việc sử dụng TCP socket. Không chỉ sử dụng riêng cho ứng dụng web, Websocket có thể áp dụng cho bất kì ứng dụng nào khác cần có sự trao đổi thông tin trên Internet. Giao thức WebSocket ra đời nhằm mục đích khắc phục nhược điểm độ trễ khá cao của HTTP – HyperText Transfer Protocol – Giao thức truyền tải siêu văn bản.
* **Ứng dụng Websocket:** Ứng dụng của Websocket được dùng trong các 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. Websocket còn được dùng trong game online hoặc các ứng dụng chat.

**Phần 3: So sánh HTTP và Websocket**

* **Giống nhau:** Điểm giống nhau là cả WebSocket và HTTP đều là những giao thức hỗ trợ việc truyền tải thông tin giữa máy chủ và máy trạm. Mục đích của các nhà phát triển khi tạo ra WebSocket là nhằm để khắc phục được nhược điểm độ trễ cao của HTTP.
* **Khác nhau:**

|  |  |  |
| --- | --- | --- |
|  | **HTTP** | **Websocket** |
| **Quy trình vận hành** | **HTTP là giao thức 1 chiều dựa theo giao thức TCP, bạn có thể tạo ra các kết nối dựa vào request HTTP, sau khi kết nối được thực hiện xong và được phản hồi lại, quá trình sẽ kết thúc và đóng lại.** | **WebSocket là một giao thức truyền tải 2 chiều giữa máy chủ và máy khách. Dữ liệu có thể truyền 2 chiều giữa máy khách – máy chủ hoặc máy chủ – máy khách dựa trên những kết nối đã được thiết lập.** |
| **Ứng dụng vào thực tiễn** | **HTTP được ứng dụng và sử dụng vào các dịch vụ ứng dụng RESTful đơn giản, chủ yếu là nhận thông tin 1 chiều về server để xử lý => phản hồi => đóng kết nối.** | **Hầu hết các ứng dụng theo thời gian thực – real-time đều đang ứng dụng WebSocket để truyền và nhận dữ liệu trên 1 kênh liên lạc duy nhất.** |
| **Ứng dụng dựa vào mức độ ưu tiên** | **Khi muốn giữ lại một kết nối trong khoảng thời gian cụ thể hoặc bạn chỉ có nhu cầu sử dụng kết nối để truyền dữ liệu, HTTP sẽ chính là lựa chọn ưu tiên hơn.** | **Đối với các ứng dụng cần phải thường xuyên hoặc liên tục cập nhật thông tin, WebSocket sẽ là ưu tiên hàng đầu vì WebSocket có kết nối nhanh hơn, độ trễ thấp hơn so với HTTP.** |

**Phần 4 Kết luận**

Từ các thông tin trên ta có thể thấy rằng WebSocket sẽ có lợi thế rất nhiều so với HTTP trong các trường hợp thao tác, dịch vụ của bạn đang thực hiện cần phải có độ trễ thấp, đặc biệt là các tính nhắn từ client đến server. Nhưng WebSocket chỉ là một phần mở rộng cho HTTP nên trong thực tế WebSocket sẽ sử dụng cho các ứng dụng được xây dựng bằng JavaScript và cần phải cập nhật dữ liệu liên tục theo thời gian thực. Ví dụ thực tế như các game online. Vì vậy, sẽ không có một khuôn mẫu điển hình nào để bạn chỉ chọn 1 trong 2 phương thức trên. Điều bạn cần làm sẽ là xác định phương thức nào sẽ phù hợp với dự án của bạn hơn.