Lớp ứng dụng (Application Layer): là giao diện giữa các chương trình ứng dụng của người dùng và mạng. Lớp Application xử lý truy nhập mạng chung, kiểm soát luồng và phục hồi lỗi. Lớp này không cung cấp các dịch vụ cho lớp nào mà nó cung cấp dịch vụ cho các ứng dụng như: truyền file, gởi nhận E-mail, Telnet, HTTP, FTP, SMTP…

Chúng ta cùng nhau tìm hiểu một số giao thức ở lớp ứng dụng bao gồm:

Ở bài viết này tôi sẽ trình bày cho các bạn về HTTP và HTTPS rồi so sánh chúng. Các giao thức khác bạn có thể truy cập vào đường link mình gắn ở sau mỗi giao thức để tìm hiểu. cảm ơn bạn đã xem bài viết này.

* HTTP
* HTTPS
* FTP
* SMTP
* POP
* IMAP

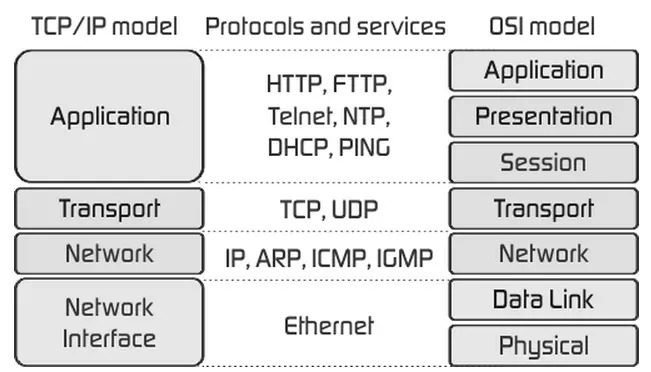
Bắt đầu thôi nào,

# HTTP

1. Giao thức HTTP là gì?

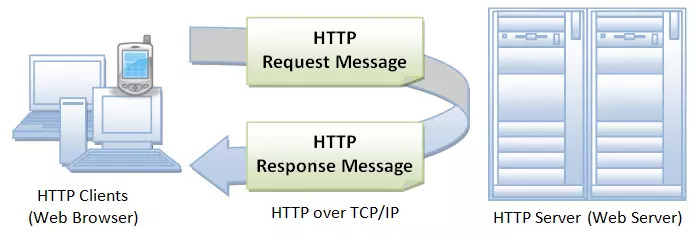
**HTTP** (**H**yper**T**ext **T**ransfer **P**rotocol – Giao thức truyền tải siêu văn bản) là một trong các giao thức chuẩn về mạng Internet, được dùng để **liên hệ thông tin** giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client), là giao thức Client/Server dùng cho World Wide Web – WWW

**Giao thức HTTP** là một giao thức tầng ứng dụng của **bộ giao thức TCP/IP** (các giao thức nền tảng cho Internet).



1. Cách hoạt động giao thức HTTP

Giao thức HTTP hoạt động dựa trên mô hình Client – Server. Thông thường khi các bạn lướt web, 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.



Ngoài ra, khi các hệ thống trao đổi dữ liệu với nhau, chúng cũng sử dụng giao thức này nhưng 2 bên đều là server.

1. HTTP – Requests

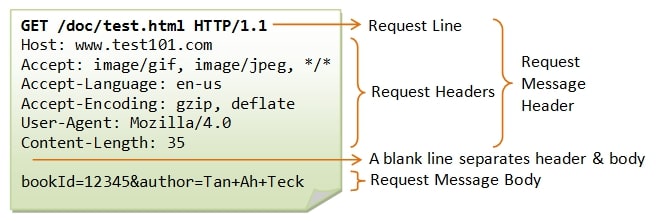
Một HTTP client (máy khách) gửi một HTTP request (yêu cầu) lên server (máy chủ) nhờ một thông điệp có định dạng như sau:

<method> <request-URL> <http-serverion>

<headers>

<body>

Ví dụ:



1. **Request Line**

Bắt đầu của HTTP Request sẽ là dòng Request-Line bao gồm 3 thông tin:

**Method**

Báo cho Server rằng hành động sẽ phải sử lý với thông tin được gửi từ client lên.

***Phương thức GET***

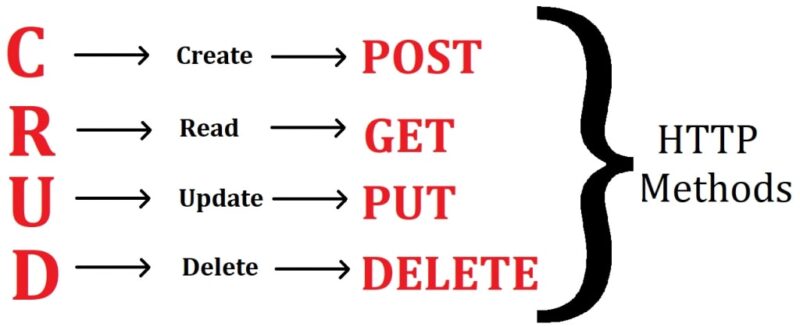
* Câu truy vấn sẽ được đính kèm vào đường dẫn HTTP request. Ví dụ: /?username=”tinohost”&pass=”tenmien”
* GET request có thể được cached, bookmark và lưu trong lịch sử của trình duyệt mà bị giới hạn về chiều dài (chiều dài của URL là có hạn).
* *Lưu ý:* Bạn không nên dùng GET request với dữ liệu quan trọng mà chỉ dùng để nhận dữ liệu, không có tính bảo mật.

***Phương thức POST***

* Câu truy vấn sẽ được gửi trong phần message body của HTTP request.
* POST không thể, cached, bookmark hay lưu trong lịch sử trình duyệt và cũng không bị giới hạn về độ dài.

***Các phương thức khác***

* HEAD: tương tự như GET nhưng chỉ gửi về HTTP header.
* PUT: tải lên một mô tả về URL định trước.
* DELETE: xóa một tài nguyên định trước.
* OPTIONS: trả về phương thức HTTP mà server hỗ trợ.
* CONNECT: chuyển kết nối của HTTP request thành một kết nối HTTP tunnel.



**Request URL**

Một URL (Uniform Resource Locator) được sử dụng để xác định duy nhất một tài nguyên trên Web. Một URL có cấu trúc như sau:

protocol://hostname:port/path-and-file-name

Trong một URL có 4 thành phần:

* Protocol: giao thức tầng ứng dụng được sử dụng bởi client và server
* Hostname: tên DNS domain
* Port: Cổng TCP để server lắng nghe request từ client
* Path-and-file-name: Tên và vị trí của tài nguyên yêu cầu.

**HTTP version**

HTTP version là Phiên bản giao thức HTTP đang sử dụng.

1. **Request header**

Tiếp theo dòng Request-Line là các trường Request-header, cho phép client gửi thêm các thông tin bổ sung về thông điệp HTTP request và về chính client. Một số trường thông dụng như:

* Accept loại nội dung có thể nhận được từ thông điệp response. Ví dụ: text/plain, text/html…
* Accept-Encoding: các kiểu nén được chấp nhận. Ví dụ: gzip, deflate, xz, exi…
* Connection: tùy chọn điều khiển cho kết nối hiện thời. Ví dụ: keepalive,Upgrade…
* Cookie: thông tin HTTP Cookie từ server
* User-Agent: thông tin về user agent của người dùng.

1. **Request Body**

Body là dữ liệu mà Client sẽ gửi lên Server,

Một request body có thể là một đoạn text thuần túy, HTML, XML, JSON, Javascript, hoặc một tập các cặp key-value dạng form-data.

Khi bạn mở một web, trình duyệt sẽ nhận payload dạng HTML, nó chính là giao diện mà chúng ta quan sát được trên trình duyệt đó.

Thông thường khi làm việc với các HTTP APIs chúng ta sẽ gửi và nhận các payload dạng JSON hoặc XML.

Không phải tất cả các message HTTP đều phải có payload: POST và PUT có thể có, còn với GET và DELETE thì có thể không có payload.

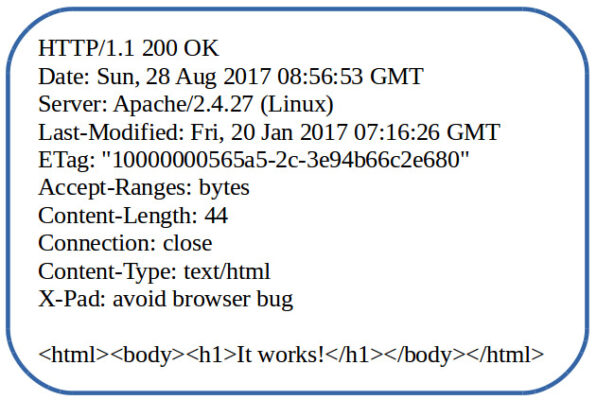
Sau khi request server sẽ xử lý và phản hổi (response) tới client theo 1 trong 3 phương pháp:

* Server phân tích request nhận được, maps yêu cầu với tập tin trong tập tài liệu của server, và trả lại tập tin yêu cầu cho client.
* Server phân tích request nhận được, maps yêu cầu vào một chương trình trên server, thực thi chương trình và trả lại kết quả của chương trình đó.
* Request từ client không thể đáp ứng, server trả lại thông báo lỗi.

**HTTP – Responses**

HTTP response là bản tin trả về từ server sang client, trong đó sẽ có các trường thông tin mà request yêu câu.

Định dạng gói tin HTTP response cũng gồm 3 phần chính là: Status line, Header và Body.



**Response Status**

Gồm 3 trường là phiên bản giao thức (HTTP version), mã trạng thái (Status code) và mô tả trạng thái (Status text):

* Phiên bản giao thức (HTTP version): phiên bản của giao thức HTTP mà server hỗ trợ, thường là HTTP/1.0 hoặc HTTP/1.1
* Mã trạng thái (Status code): mô tả trạng thái kết nối dưới dạng số, mỗi trạng thái sẽ được biểu thị bởi một số nguyên. Ví dụ: 200, 404, 302,…
* Mô tả trạng thái (Status text): mô tả trạng thái kết nối dưới dạng văn bản một cách ngắn gọn, giúp người dùng dễ hiểu hơn so với mã trạng thái. Ví du: 200 OK, 404 Not Found, 403 Forbiden,…

Một số loại thông dụng mà server trả về cho client như sau:

***1xx: information Message***

Các status code này chỉ có tính chất tạm thời, client có thể không quan tâm.

***2xx Successful***

Khi đã xử lý thành công request của client, server trả về status dạng này:

* 200 OK: request thành công.
* 202 Accepted: request đã được nhận, nhưng không có kết quả nào trả về, thông báo cho client tiếp tục chờ đợi.
* 204 No Content: request đã được xử lý nhưng không có thành phần nào được trả về.
* 205 Reset: giống như 204 nhưng mã này còn yêu câu client reset lại document view.
* 206 Partial Content: server chỉ gửi về một phần dữ liệu, phụ thuộc vào giá trị range header của client đã gửi.

***3xx Redirection***

Server thông báo cho client phải thực hiện thêm thao tác để hoàn tất request:

* 301 Moved Permanently: tài nguyên đã được chuyển hoàn toàn tới địa chỉ Location trong HTTP response.
* 303 See other: tài nguyên đã được chuyển tạm thời tới địa chỉ Location trong HTTP response.
* 304 Not Modified: tài nguyên không thay đổi từ lần cuối client request, nên client có thể sử dụng đã lưu trong cache.

***4xx Client error***

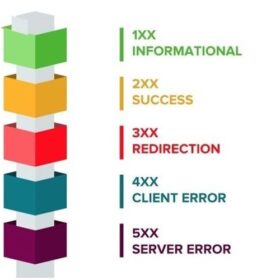
Lỗi của client:

* 400 Bad Request: request không đúng dạng, cú pháp.
* 401 Unauthorized: client chưa xác thực.
* 403 Forbidden: client không có quyền truy cập.
* 404 Not Found: không tìm thấy tài nguyên.
* 405 Method Not Allowed: phương thức không được server hỗ trợ.

***5xx Server Error***

Lỗi của server:

* 500 Internal Server Error: có lỗi trong quá trình xử lý của server.
* 501 Not Implemented: server không hỗ trợ chức năng client yêu cầu.
* 503 Service Unavailable: Server bị quá tải, hoặc bị lỗi xử lý.



***Response Header***

Header của gói tin response có chức năng tương tựn như gói tin request, giúp server có thể truyền thêm các thông tin bổ sung đến client dưới dạng các cặp “Name:Value”.

***Response Body***

Là nơi đóng gói dữ liệu để trả về cho client, thông thường trong duyệt web thì dữ liệu trả về sẽ ở dưới dạng một trang HTML để trình duyệt có thể thông dịch được và hiển thị ra cho người dùng.

Hoặc trả về dạng JSON, XML khi giao tiếp bằng API.

**Công cụ kiểm tra API với giao thức HTTP**

Để kiểm tra giao thức HTTP và làm việc với các API chúng ta có thể sử dụng công cụ như:

***Postman***

Postman là một công cụ cho phép chúng ta thao tác với API, phổ biến nhất là REST. Postman hiện là một trong những công cụ phổ biến nhất được sử dụng trong thử nghiệm các API. Với Postman, ta có thể gọi Rest API mà không cần viết dòng code nào.

Postman hỗ trợ tất cả các phương thức HTTP (GET, POST, PUT, PATCH, DELETE, …). Bên cạnh đó, nó còn cho phép lưu lại lịch sử các lần request, rất tiện cho việc sử dụng lại khi cần.

***Soap UI***

Soap UI là công cụ phổ biến để kiểm tra API Nó cho phép bạn kiểm tra APU Rest và Soap một cách dễ dàng. Công cụ này tạo ra các thử nghiệm nhanh chóng với các thao tác đơn giản, dễ thực hiện.

# HTTPS

1. Giao thức HTTPS là gì?

HTTPS viết tắt của Hyper Text Transfer Protocol Secure (giao thức truyền tải siêu văn bản bảo mật) là phiên bản an toàn của HTTP, giao thức mà nhờ đó dữ liệu được gửi giữa trình duyệt và trang web bạn đang kết nối. Chữ ‘S’ ở cuối HTTPS là viết tắt của “Secure” (Bảo mật)

HTTPS cũng có những hoạt động giống như HTTP. Tuy nhiên, chúng được bổ sung SSL và TSL. Giao thức này đảm bảo không có bên thứ 3 đăng nhập được và đánh cắp thông tin. Dù cho bạn đang tìm kiếm thông tin bằng máy tính công cộng hay máy cá nhân thì HTTPS cũng sẽ đảm bảo thông tin liên lạc của bạn với máy chủ được bảo mật.



1. Bảo mật PKI – Public Key Infrastructure

Cả hai giao thức TLS và SSL đều sử dụng hệ thống PKI (Public Key Infrastructure, hạ tầng khóa công khai) không đối xứng. Một hệ thống không đối xứng sử dụng hai “khóa” để mã hóa thông tin liên lạc, khóa “công khai” và khóa “riêng”. Bất cứ thứ gì được mã hóa bằng khoá công khai (public key) chỉ có thể được giải mã bởi khóa riêng (private key) và ngược lại.

Một hệ thống PKI hoàn chỉnh bao gồm các thành phần sau:

* Khóa công khai – Public Key
* Khóa bí mật – Privte Key
* Ủy quyền thẻ chứng thực – Certification Authority (CA)
* Ủy quyền thẻ đăng ký – Registration Authority (RA)
* Ủy quyền xác nhận hợp lệ – Validation Authority (VA)
* Thẻ chứng thực số – Digital certificates (DC)
* Thẻ chứng thực bị CA thu hồi – Certificate revocation list (CRL)
* Hệ thống phân phối thẻ – Certificate Distribution System (CDS)

1. Chứng chỉ SSL là gì

Khi yêu cầu kết nối giao thức HTTPS với trang web, đầu tiên trang web sẽ gửi chứng chỉ SSL tới trình duyệt của bạn. Chứng chỉ này chứa khóa công khai cần thiết để bắt đầu phiên bảo mật. Dựa trên trao đổi ban đầu này, trình duyệt và trang web sẽ bắt đầu giao thức SSL handshake (giao thức bắt tay). Giao thức SSL handshake liên quan đến việc tạo bí mật chia sẻ để thiết lập kết nối an toàn duy nhất giữa bạn và trang web.

Khi sử dụng chứng chỉ SSL đáng tin cậy trong quá trình kết nối HTTPS, người dùng sẽ thấy biểu tượng ổ khóa trong thanh địa chỉ của trình duyệt. Khi một chứng chỉ Extended Validation Certificate được cài đặt trên một trang web, thanh địa chỉ sẽ chuyển sang màu xanh lá cây.



# Sự khác nhau giữa giao thức HTTP và HTTPS

|  |  |  |
| --- | --- | --- |
| TIÊU CHÍ | HTTP | HTTPS |
| Chứng chỉ SSL | Không có SSL | Được bổ sung chứng chỉ SSL đảm bảo thông tin được bảo mật, tránh rò rỉ. |
| Port | Port 80 | Port 443 bảo vệ dữ liệu đang truyền đi |
| Mã hóa | Không có mã hóa bảo mật | Sử dụng SSL/TSL tiêu chuẩn, bảo mật |
| Mức độ bảo mật | Dữ liệu không được xác thực bảo mât | Xác thực đích danh của website. Giúp quá trình trao đổi thông điệp diễn ra an toàn |

HTTP kết nối sẽ đơn giản hơn HTTPS nhưng phần bảo mật sẽ không được đảm bảo, dễ bị tấn công.

# Kết luận

Giao thức HTTP và HTTPS và MQTT là các giao thức sử dụng rất rộng dãi khi làm các thiết bị IOT. Khi lập trình với các dòng chip như ESP8266, ESP32, các dòng SIMCOM, …. Vậy nên các bạn chú ý thật kỹ phần lý thuyết trên.

# TÀI LIỆU THAM KHẢO

[1] Nguyễn Thúc Hải, Mạng máy tính và các hệ thống mở, NXB Giáo Dục, 2002.

[2] TS. Cao Thành Nghĩa, Bài giảng Kỹ thuật mạng máy tính, Tài liệu giảng dạy.

[3] https://en.wikipedia.org/

[4] https://khuenguyencreator.com/