Chapter 7B: HTTP 1.1

Objective

This chapter goes into detail of using HTTP 1.1 for communicating with the Cloud

Introduction

HTTP came about in early 90's for static HTML pages. Evolved into dynamic HTTP (reading/writing databases and creating HTTP on the fly

Lots of infrastructure now exists so for IoT, it is what people used

70% of IoT traffic today is HTTP even though it is overkill (heavy, lots of overhead)

HTTP 1.1 came out in the late 90's and is still used for >50% of web traffic

HTTP 2.0 came out in the mid zeros. Unlike 1.1, it is binary rather than text based. It is supported in WICED too, but is different enough to be treated as a separate chapter.

HTTP 1.1 Protocol

HTTP 1.1 is:

- An application layer
- Single transaction – send request, receive response
- Stateless (cookies added later for some state information)
- Plain-Text (GET, POST, PUT)
- Client-server Protocol

The protocol is client makes a request, then the server responds.

Client Request Message Format

Request message is: Start Line, Headers, Body

Start Line

Start line is: Method, Resource path, Options, Version

Methods can be safe/unsafe, idemponent/non-idempotent

Methods are: GET/PUT/POST, etc.

Mostly IoT devices will use GET and POST

Resource Path: explain how it relates to WEB address

Options: follow path with "?" and separate with "&"

Headers

Headers are name/value pairs. Show examples. Host header is required in a request. If you have a body, must have Content-type and Content-length

Must send \r\n at end of every header and one line of just \r\n after the last header but WICED does this for you.

Body

Body is optional

MIME type (Multipurpose Internet Mail Extension) used to specify the type

IoT devices will usually use applicaton/json

Server Response Message Format

Server response message is: Start Line, Headers, Body – just like the request

After the response, the client can close the connection, or leave it open for another request. Server will close the connection after a timeout.

Start Line

Start Line is: Protocol, Status Code, Status Message

Status codes can be found on the Web (google “mozilla http status”)

Headers and Body

Same format as the request

Client for URLs (CURL)

CURL lets you test out HTTP requests before implementing in firmware. Will try in the examples.

Built into MAC, available in class material for Windows.

Talk through the example.

Representational State Transfer (REST) and RESTful APIs

Internet started out wild-wild-west (and is still somewhat that way). Thomas Fielding wrote a PhD dissertation proposing a REST design philosophy with 7 characteristics (listed in manual).

A RESTful API is a webserver that implements REST. Practically, that means (discuss items from manual)

Web APIs

There are lots of APIs out there and sites that list them for you.

Many APIs use either an option or a header for an API key to force you to register (and potentially pay) for using the API.

WICED HTTP 1.1 Client Library

WICED has several HTTP libraries in libraries/protocol:

- HTTP 1.1 Client, HTTP 1.1 Server, HTTP 2

To use the HTTP_Client library:

1. Talk through the list from manual
2. Discuss the callback function
    a. Arguments: http_client_t, http_event_t, and http_response_t
3. Use http_parse_header to look for headers
4. Parse the payload
5. Call the deinit function to free everything up
    a. Make sure remaining length is 0 before calling deinit
6. Call client deint after the server disconnects
    a. DON'T do this from inside the callback since it would remove a running thread.

HTTPBIN.ORG

Httpbin.org is a website that can be used to test requests. Use PUT, POST, GET, etc.

Note that you send PUTs to a resource called /put, GETs to a resource called /get, etc. so that you get an appropriate response. This is not typical.

Initial State

Initial state is a Web API IoT analysis platform (i.e. simple cloud server).

You send data to the cloud and can then display/analyze your data from their web platform

Send streams of data and collect streams in to buckets. Display data in tiles.

Discuss Elkhorn Creek example and why it is relevant to Alan

APIARY

Explain how APIARY can be used to test the format/syntax prior to putting into your FW.

Exercise(s)

1 ½ hours

Try accessing Httpbin.org using CURL and then using a snip from WICED.

Use Initial State virtual test server and then from WICED

Advanced: Send data from IoT device, graph data, use conversion API, etc.