System Architecture Specification

(Architekturspezifikation)

Author: Isabel Schwalm

(TINF20C, SWE I Praxisprojekt 2021/2022)

Project: Websockets in a LwIP HTTP Server

Customer: Rentschler & Holder
Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 4: Laura Reeken, inf20051@lehre.dhbw-stuttgart.de
Benjamin Esenwein, inf20074@lehre.dhbw-stuttgart.de
Yannis Plaschko, inf20093@lehre.dhbw-stuttgart.de
Maximilian Meier, inf20084@lehre.dhbw-stuttgart.de
Lucas Kaczynski, inf20147@lehre.dhbw-stuttgart.de
Isabel Schwalm, inf20085@lehre.dhbw-stuttgart.de
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	02.11.2021	Isabel Schwalm	created
0.2	03.11.2021	Isabel Schwalm	Filled Introduction, defined Quality goals, started Architectural concept
0.3	05.11.2021	Isabel Schwalm	Update and modify Architectural concept, system and software environment
1.0	08.11.2021	Isabel Schwalm	Filling System design and technical concepts, review
1.1	05.05.2022	Benjamin Esenwein	Document format fix

Contents

1. Introduction

lwIP is a fully functional TCP/IP stack focusing on consuming as few resources as possible. In this project, the architectural deficiencies of the patch "#9525 (httpd: add websocket support)" [1] are to be fixed in coordination with the project community. In doing so, the goal is to get the patch through the approval process in the open source project.
For development and testing purposes, the latest version of lwIP must be installed and made executable on a Windows system. A demo server (in a virtual environment) will also be created. After the architectural flaws of the patch have been fixed, the design and implementation of a GUI-based test client can begin. The improved patch is to be both demonstrated and extensively tested with the test client to ensure full functionality.

1.1. Glossar

lwIP	lightweight IP
TCP/IP	Transmission Control Protocol/Internet Protocol
OSI model	Open Systems Interconnection model
IDE	Integrated development environment
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol

2. System Overview

The system will work as follows:
The first step of fixing the architectural deficiencies is to get lwIP running on Windows.
The user will test the WebSocket support by pressing a button on the GUI. This will request the API und the response will be shown in a understandable way to the user. The bidirectional communication will be tested.

2.1. System Environment

lwIP is suitable for embedded Systems with minimal RAM and ROM availability such as stm32. Nevertheless windows 32 support is still present.

2.2. Software Environment

To compile lwIP, the latest compatible directories must be loaded from the git repository:

- Contrib: STABLE-2_1_0_RELEASE: lwIP-contrib-STABLE-2_1_0_RELEASE.tar.gz
- LwIP: STABLE-2_1_0_RELEASE lwIP-STABLE-2_1_0_RELEASE.tar.gz
- Additional header files: lwIPcfg.msvc.h (save in contrib/ports/win32)

The program library WinPcap is used, consisting of a driver that provides hardware-near access to the network card and a collection of programs that provide convenient access to the individual layers of the OSI model relevant for networks.
The IDE Visual Studio (Community Edition) is used to compile the code. Alternatively CMake can be used.
The HTTP Server will be written in C. CLion and vsCode will be used as IDE.

2.3. Quality Goals

The following quality goals listed below should be achieved by the following architecture.

2.3.1. Usability

The test client should provide an intuitive GUI.

2.3.2. Maintainability

The patch will be maintained by Simon Goldschmidt.

2.3.3. Portability

A portable demo server is to be created. In addition, it should be possible to run lwIP under Windows without much effort.

2.3.4. Bug Fixing

Patch #9525 is to be brought through the approval process by Simon Goldschmidt. All bugs should be fixed. Other bugs can be reported.

2.4. Quality Concept

The following quality goals should be archived by the end of the project.

2.4.1. Usability concept

The patch should achieve its objectives effectively, efficiently, and satisfactorily.

2.4.2. Code Quality

In a software project, code quality is one of the most important aspects. Therefore, the developers in this project will be guided by Robert Martin's Clean Code goals [2]. This includes the following principles:

- Try to write code which can be understand without comments – only write necessary comments
- DRY – Don’t Repeat Yourself
- YAGNI – You ain’t Gonna Need It
- SOC - Separation of Concerns
- KISS – Keep It Simple, Stupid

In addition, as many as possible already existing resources (from forums, ...) are used.

3. Architectural Concept

3.1. Architectural Model

As usual with TCP/IP Protocols, lwIP is designed in a layered fashion. Each protocol can be implemented separately as its own module with a few functions acting as entry points to each protocol. To improve performance (processing speed and memory usage) a relaxed scheme for communication between the application and the lower layer protocols is used.
Modules of LwIP [3]:
TCP/IP protocols:

- IP (Internet Protocol) including packet forwarding over multiple network interfaces
- ICMP (Internet Control Message Protocol) for network maintenance and debugging
- IGMP (Internet Group Management Protocol) for multicast traffic management
- UDP (User Datagram Protocol) including experimental UDP-lite extensions
- TCP (Transmission Control Protocol) with congestion control, RTT estimation and fast recovery/fast retransmit

Support modules:

- Raw/native API for enhanced performance
- Optional Berkeley-like socket API
- DNS (Domain names resolver)
- SNMP (Simple Network Management Protocol)
- DHCP (Dynamic Host Configuration Protocol)
- AUTOIP (for IPv4, conform with RFC 3927)
- PPP (Point-to-Point Protocol)
- ARP (Address Resolution Protocol) for Ethernet

TCP/IP is a collection of protocols that enables communication between different Internet-enabled devices. TCP/IP defines how information is packetized, addressed, transmitted, routed and received.
TCP/IP is divided into four layers. The lowest layer is the link layer. It contains all hosts that a device can access within its network without going through a router. Data packages can be exchanged via the link layer within this local or virtual network.
One layer above is the internet layer. With this it is possible to transfer data packages from one network to another network. This process is called routing.
Located above the internet layer is the transport layer. This is where the host-to-host connections take place. These end-to-end message transfer services are independent of their underlying network.
The top most layer is the application layer. This is where process-to-process data exchange for applications takes place. Applications can exchange information through already established connections in the lower layers. The Hypertext Transfer Protocol (HTTP) and the File Transfer Protocol (FTP), for example, run via the application layer [4].

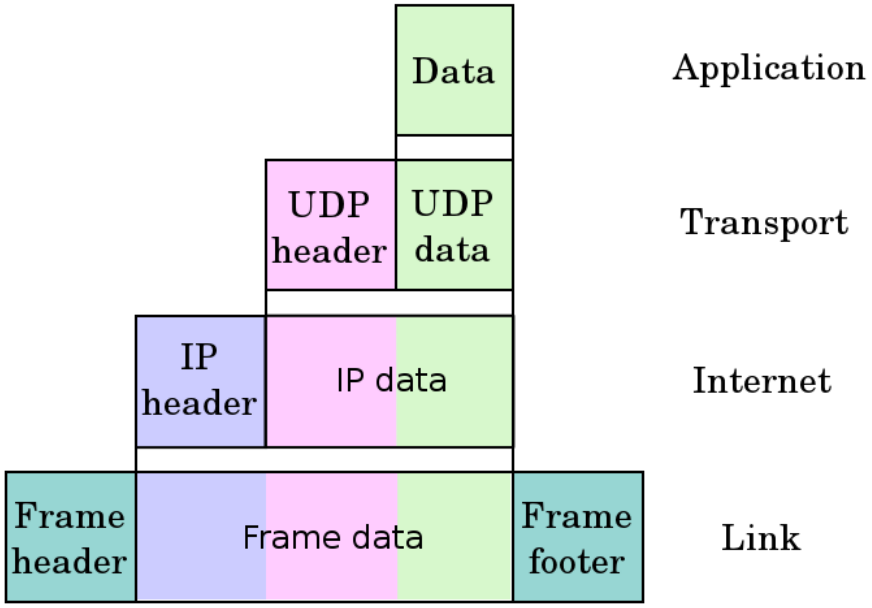

Figure 1: Structure of TCP/IP [5]

Also lwIP offers a simple HTTP-server, but there is currently no WebSocket support. Websockets is is a full-duplex technology that makes it possible to establish an interactive communication session between the user's browser and the server. With this API you can send messages to the server and receive event-driven responses without having to query the server for the response [5].

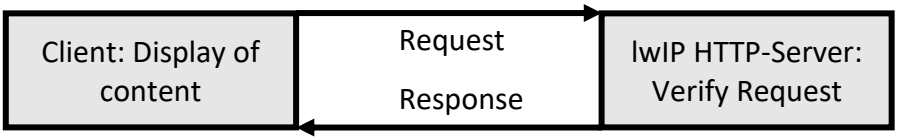

Figure 2: Communication between client and HTTP server

The connection between the server and the client remains open. Thus, the server can send information to the client without the client needing to request it.

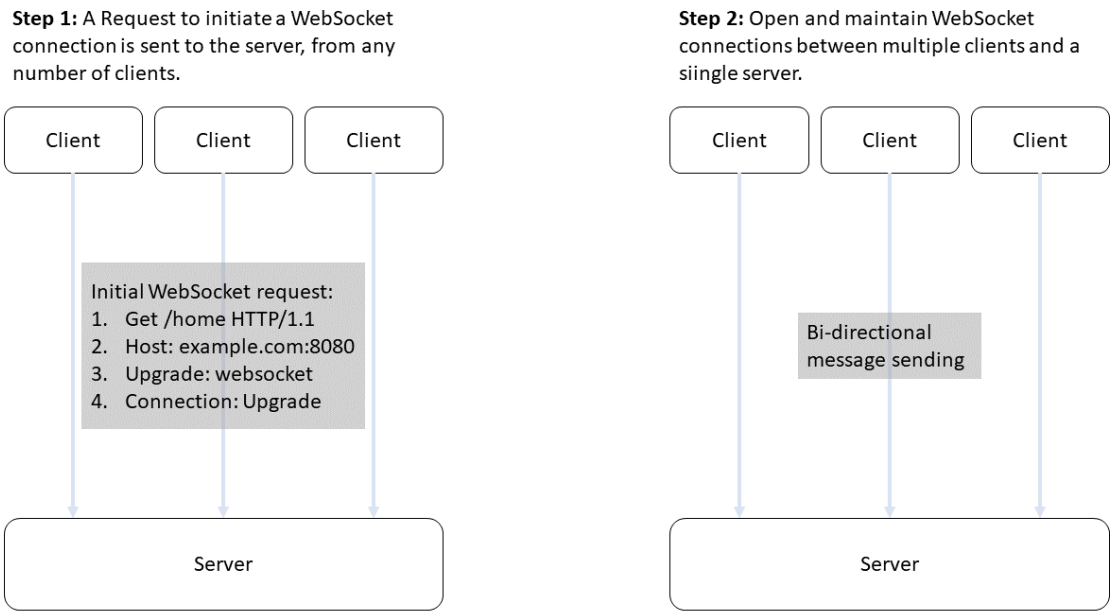

Figure 3. WebSocket Protocol.

TCP/IP used to run over http version 1.0. In the meantime, http version 2.0 has been released, which enables faster connections. A comparison of the two versions can be found in Table 1.

FEATURE	HTTP 1.0	HTTP 2.0
DATASTREAM	Multiple TCP connections are opened for different page elements (JS, CSS, image files).	A TCP connection is opened over which several page elements can be transferred in parallel.
COMPRESSION	Data is transmitted uncompressed.	Data is compressed into binary code and then transmitted.
PRIORITIZATION	Data packages are not prioritized.	Data packages are prioritized.

Table 1. Comparisson between http version 1.0 and 2.0.

4. Systemdesign

4.1. <MOD.001>: HTTP-Server

This module specifies the simple HTTP server which is offered by lwIP.

<MOD.002>	HTTP Server
System requirements covered:	/F40/
Service:	
Interfaces:	• WebSockets
External Data:	• User request
Storage location:	GitHub link!

4.2. <MOD.002>: Graphical User Interface

This module specifies and implements the graphical user interface and manages all possible in- and outputs.

<MOD.002>	Graphical User Interface
System requirements covered:	/NF30/
Service:	• Display a graphical interface to the user • Handle user input • Display exceptions, errors and warnings
Interfaces:	• User input
External Data:	• Output http-Server
Storage location:	GitHub link!

5. Technical Concepts

5.1. User Interface

One part of the project is to implement an GUI based Test client which will be used to demonstrate and test the functionality of patch #9525 (httpd: add websocket support).

5.2. Ergonomics

The graphical user interface will follow the standard ergonomic design patterns. It will be kept simple and designed for practicality.

5.3. Deployment

lwIP can be used after it has been installed on an appropriate (emulated) microcontroller. In addition, an installation routine for windows is to be created.

5.4. Exception Handling

The user needs to click one button. If the test client throws errors, meaningful error messages will be shown to the user.

5.5. Internationalisation

The software and all its components are written in English. The same goes for the instructions. Accordingly, the software can be used worldwide.

5.6. Testability

The software will be tested with the self-written test client.

5.7. Availability

The program and code can be cloned via GitHub.

6. Figures

7. References

[1] „lwIP - A Lightweight TCP/IP stack - Patches: patch #9525, httpd: add Websocket support [Savannah],“ 15 September 2021. [Online]. Available: https://savannah.nognu.org/patch/?9525.

[2] R. C. Martin, „Clean Code: A Handbook of Agile Software Craftsmanship,“ Prentice Hall, 2008.

[3] [Online]. Available: https://lwip.fandom.com/wiki/LwIP_Wiki. [Zugriff am 24 September 2021].

[4] „Elektronik Kompendium,“ [Online]. Available: https://www.elektronik-kompendium.de/sites/net/0606251.htm. [Zugriff am 08 October 2021].

[5] „MDN Web Docs mozilla,“ [Online]. Available: https://developer.mozilla.org/de/docs/Web/API/WebSockets_API. [Zugriff am 03 November 2021].

[6] „Wikipedia,“ [Online]. Available: https://en.wikipedia.org/wiki/File:UDP_encapsulation.svg. [Zugriff am 08 October 2021].