VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Hidajete Isa

WEB BASED GUI MANAGEMENT FOR FLEXINT22 SHDSL.BIS MODEM

Technology and communication

2011

FOREWORD

First of all I want to thank my family, especially my mother who has supported me from the beginning, her encouragement is what kept me motivated and moved forward. I also want to thank my friends who supported me.

Secondly I want to thank Pekka Lappalainen, R&D Manager, for giving me this opportunity to do the bachelor thesis at their department. I also would like to express my sincere gratitude to my instructor Tommi Lundell, SW Team leader, for his great help and guidance during the entire project. A big thank you goes to all the co-workers that helped with this project.

Finally, I wish to thank my supervisor Antti Virtanen, senior lecturer, for the valuable advice and support he has given me in the writing of this report.

15.2.2011 Helsinki

Hidajete Isa

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Hidajete Isa
Title	Web Based GUI Management for FlexiNT22 SHDSL.bis Modem
Year	2011
Language	English
Pages	51
Supervisor(s)	Antti Virtanen
Instructor	Tommi Lundell, SW Team Leader

The purpose of this thesis was to design and implement a prototype of a web based GUI management for FlexiNT22 SHDSL.bis modem at Nokia Siemens Networks BBA NBMS division.

Web based GUI management gives an administrator the ability to configure and monitor FlexiNT22 over the Internet using a web browser. The most direct way to accomplish this is to embed a web server (Embedded web Server) into the modem, and use that server to provide web-based management user interface constructed using HTML language.

The project involved familiarizing with the operations and characteristics of FlexiNT, searching for an appropriate web server, examining the features and the compatibility with the software, evaluation and implementing a demo version. The evaluation consists mainly of three parts: Surveying web servers and further choosing the most suitable web servers for the evaluation. In the third place, we were concerned with defining criteria of the embedded web server features for evaluation.

The project was carried out using KLone embedded web server, which is open source software. The study describes how to program HTML pages in C language and how to implement web pages. As result pages could be embedded into a single executable binary file that contained KLone's HTTP/S server.

Keywords	Embedded web server, web-based gui management, shdsl modem, html, C language

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Hidajete Isa
Opinnäytetyön nimi	FlexiNT22 SHDSL.bis modeemin web-hallintasovellus
Vuosi	2010
Kieli	Englanti
Sivumäärä	51
Ohjaaja	Antti Virtanen
Valvoja	Tommi Lundell, SW Team Leader

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa graafinen web-pohjainen käyttöliittymä prototyyppi FlexiNT22 SHDSL.bis päätelaitteeseen Nokia Siemens Networksin NBMS (Narrowband Multiservic) osastolle.

Web-hallintasovellus antaa järjestelmänvalvojalle mahdollisuuden määrittää ja valvoa FlexiNT22 modeemia internetin kautta käyttäen web-selainta.Toteuttaakseen tämän tarvitaan sulautettu Web-palvelin joka upotetaan modeemin sisään, ja käyttäen tätä palvelinta luodaan web-hallintasovellus HTML-kielellä.

Projekti jakautuu kahteen osaan: tutkiminen ja implementaatio. Tutkimukseen kuului perehtyminen FlexiNT:n toimintaan ja ominaisuuksiin, sopivan web-palvelimen etsiminen ja sen ominaisuuksien tutkiminen ja soveltuvuus ohjelmiston kanssa. Sulautettu web-palvelin on ideaali tähän projektiin. Valintamenetelmiin kuului valita kolme sopivinta palvelinta ja tutkia niiden ominaisuudet.
Implementaatioon kuului suunnitella ja toteuttaa toimiva web-hallintasovellus runko.

Työ toteutettiin käyttäen KLonen web-palvelinta, joka on avoimen lähdekoodin ohjelmisto. Tässä työssä kuvataan, miten ohjelmoidaan HTML-sivuja käyttäen C-kieltä ja miten ne toteutetaan. Lopputuloksena web sivut voidaan upottaa yhteen binääri ohjelmatiedostoon, joka sisältää KLone HTTP/S-palvelimen.

Asiasanat	sulautettu web-palvelin, web-pohjainen käyttöliittymä, shdsl modeemi, html, c-kieli

SYMBOLS AND ABBREVIATIONS

ASCII	American Standard Code For Information Interchange
ASP	Active Server Pages
ATM	Asynchronous Transfer Mode
BBA	Broadband Access
BSS	Business Support Systems
CGI	Common Gateway Interface
CLI	Command Line Interface
COSI	Common Operating System Interface
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CVS	Concurrent Versions System
DAA	Digest Access Authentication
DSLAM	Digital Subscriber Line Access Multiplexer
EFM	Ethernet in the First Mile
EFMC	EFM over Copper
EOC	Engineering Order Channel
EWS	Embedded Web Server
FE	Fast Ethernet
GE	Gigabit Ethernet

GPL	General Public License
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IIS	Internet Information Services
IP	Internet Protocol
ISP	Internet Service Providers
LMI	Local Management Interface
MAC	Media Access Control
MIB	Management Information Base
MII	Media Independent Interface
MIME	Multipurpose Internet Mail Extension
NBMS	Narrowband Multiservice
NMS	Network Management System
NVT	Network Virtual Terminal
OAM	Operation Administration and Maintenance
OS	Operating System
OSE	Operating System Embedded
OBS	Operations and Business Software

OSS	Operation Support Systems
PAM	Pulse Amplitude Modulation
PMA	Physical Medium Attachment
PMD	Physical Medium Dependent
PWE	Pseudo Wire Emulation
RAM	Random Access Memory
RCS	Revision Control System
ROM	Read Only Memory
RSTP	Rapid Spanning Tree Protocol
RTOS	Real Time Operating System
SHDSL	Single-pair High speed Digital Subscriber Line
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSI	Server Side Includes
SSL	Secure Socket Layer
STP	Spanning Tree Protocol
TC	Transmission Convergence
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TLS	Transport Layer Security

URL	Universal Resource Locator
UTP	Unshielded Twisted Pair
VCS	Version Control System
VFS	Virtual File System
VLAN	Virtual Local Area Network
XML	Extensible Markup Language

CONTENTS

FOREWORD

ABSTRACT

TIIVISTELMÄ

SYMBOLS AND ABBREVIATIONS

1 INTRODUCTION 12
1.1 Project description 12
1.2 Company Background 13
2 FLEXINT22 SHDSL.bis MODEM 14
2.1 Overview 14
2.2 General Features 14
2.3 Development Environment 16
2.4 Software Platform 16
2.5 Operating System 17
2.6 TDM Interface 18
2.7 Fast Ethernet Interface 18
2.8 SHDSL Interface 19
2.9 SNMP Protocol 19
2.10 Telnet Protocol 20
2.10.1 The Network Virtual Terminal (NVT) 20
2.10.2 Options and Option Negotiation 21
2.11 EFM – Ethernet in the First Mile 21
2.12 ATM – Asynchronous Transfer Mode 22
2.13 General Web Server 22
2.14 Embedded Web Server 23
2.15 HTTP – HyperText Transfer Protocol 24
3 WEB SERVER SELECTION 27
3.1 Requirements for Embedded Web Servers 27
3.2 Embedded Web Server Solution Survey 27

3.3 The most Appropriate Solution 29
3.3.1 KLone 29
3.3.2 Nichestack HTTPServer 33
3.3.3 GoAhead Web Server 34
4 WEB GUI DESIGN 35
4.1 Interface Design 35
4.2 Web page styles and formatting 37
4.3 Web Development Tools 37
5 WEB GUI IMPLEMENTATION 38
5.1 Interface Implementation 38
5.2 Client-Server Communication 38
6 RESULTS 45
6.1 Main Page 45
6.2 Configure page 45
6.3 Line page 47
7 CONCLUSION 48
8 REFERENCES 49

LIST OF FIGURES

Figure 1. FlexiNT22 applications. [Fig1] 14
Figure 2. FlexiNT22. [Fig2] 15
Figure 3. FlexiNT22 SW environment. [Fig3] 17
Figure 4. TCP/IP Protocol Suite. [Fig4] 20
Figure 5. ATM Cell. [Fig5] 22
Figure 6. HTTP/1.0 One TCP connection per request. [Fig6] 25
Figure 7. HTTP/1.1 Multiple requests per TCP connection. [Fig7] 26
Figure 8. Example page of running Klone application. 31
Figure 9. User interface design structure. 35
Figure 10. All menu buttons and menu buttons with sublinks. 36
Figure 11. Menu frame HTML source code. 36
Figure 12. Sequence diagram between client and server. 39
Figure 13. frame_identifier[ ] variable. 40
Figure 14. Showtop and showframe. 41
Figure 15. Finished interface with three frames. 42
Figure 16. Main page. 45
Figure 17. Configure page 46
Figure 18. Line configuration page 47

1 INTRODUCTION

1.1 Project description

Rapid technology development in the last decade has put tremendous impacts to the network industry. We have undergone 2G to 3G upgrade and now even the 4G is on the way. Network service providers are equipped with up to date solutions to secure high efficiency operation as well as the best network quality for the end users. Today most operators or enterprises are using SHDSL modems not only because of its multi built in feature of transferring high-speed data in both directions and moving data farther and faster than earlier solutions but also the ability to improving spectral compatibility to pre-existing and emerging services. One of the drawback of FlexiNT22 is that it doesn't support web GUI management which means an on-site visit seems to be a must regarding any configuration update. This can be translated as huge opex to the operators shoulder. This thesis project is conducted in co-operation with my work place NSN to provide a solution by adding the web GUI management feature onto the FlexiNT22 SHDSL.bis modem. The main focus includes choosing the most suitable embedded web server, create a simple process and design user friendly web GUI structure and implement a demo version sample.

1.2 Company Background

Nokia Siemens Networks is one of the largest telecommunications hardware, software and services companies in the world. Operating in 150 countries worldwide with more than 60 000 employees, it's headquarter is in Espoo, Finland. Nokia Siemens Networks was established in 2007 as the result of a joint venture 50/50 between Nokia Corporation and Siemens AG.

Nokia Siemens Networks provides communication services. It offers a complete portfolio of mobile, fixed and converged network technologies as well as professional services including consulting and systems integration, network implementation, network design, maintenance and care, and managed services, including operations support, network operations, infrastructure and third party management.

The main products of the company include Internet and mobile communications, services, radio access, broadband access, converged core and OSS/BSS solutions. Its major manufacturing sites are in China, Finland, Germany and India. About 1 billion people are connected through its networks. The customer base of Nokia Siemens Networks includes 1,400 customers in 150 countries (including more than 600 operator customers). [1]

2 FLEXINT22 SHDSL.bis MODEM

2.1 Overview

FlexiNT22 is a Network Terminal to support Carrier Class services over the local copper loop. The subscriber services that FlexiNT22 supports are Ethernet & TDM over SHDSL.bis.

FlexiNT22 works to the Central Office equipment which can be DSLAM for the termination of the SHDSL.bis lines. FlexiNT22 can also work in a box to box configuration acting as a network terminal and a line terminal. FlexiNT22 is flexible, providing different ways of deploying TDM and Ethernet services over SHDSL, suitable for mobile backhaul or the enterprise market.

SHDSL.bis line coding supports line speeds up to 5.7 Mbit/s over a single copper pair. FlexiNT22 supports up to 4 SHDSL.bis bonded lines delivering symmetrical bandwidth of 22.8 Mbit/s.

The Figure 1 below illustrates the basic applications with FlexiNT22s, and FlexiNT22 combined with DSLAM or used as standalone connection. [2][3]

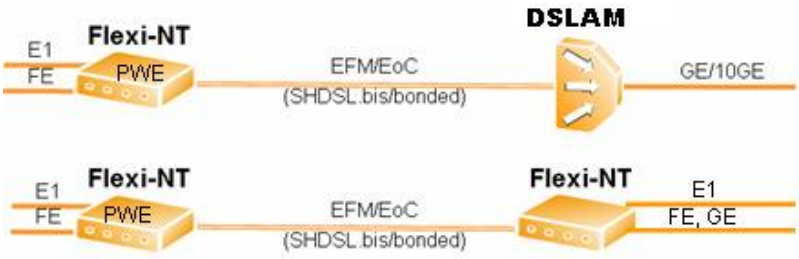

Figure 1. FlexiNT22 applications. [Fig1]

2.2 General Features

FlexiNT22 provides the following functionality:

• Support for up to 4 SHDSL.bis bonded lines

• 2 x 10/100Base-T Ethernet Interfaces

• 2 x E1/T1/J1 TDM interfaces

• Support for SHDSL.bis for ATM & EFM

• Support a wide range of Layer 2 Ethernet features

- Bridging : 2K MAC table

- VLAN tagging : 802.1Q

- VLAN Q in Q supported

- TLS supported

- STP, RSTP supported

- EFM OAM supported

• SNMP fault management

Figure 2 below shows FlexiNT22 interfaces.

Figure 2. FlexiNT22. [Fig2]

2.3 Development Environment

Tools for developing applications are a source code editor, version control system, a compiler and a debugger. FlexiNT22 is developed under linux environment (host) using RedHat Enterprise Linux platform where all application developing tools are installed. To access the developing tools from host are used X-terminals. An X-terminal is a display/input terminal for X Window System client applications.

Source code editor for writing application is used Sniff+ 3.2.1 editor which has supports to various version control tools such as RCS, CVS, ClearCase. For version control system (VCS) is used IBM's Rational ClearCase, it can handle large binary fills, large number of file, large repository sizes and versioning of directories. As for compiling is used gcc-compiler (GNU Compiler Collection). For debugging is used Trace32 tool with Lauterbach's Power Debug II Ethernet debugger.

2.4 Software Platform

FlexiNT22 has MPC8247 processor. The MPC8247 boot sets up the processor environment, configures Flash and RAM access areas. Boot should be OS independent so that it can be started without OS. Only dependency between boot and application SW is file system and possibly some reused low level device drivers can be same. The application SW is loaded by boot as one monolithic part to RAM and executed from there. The application part can be stored in compressed format in Flash file system.

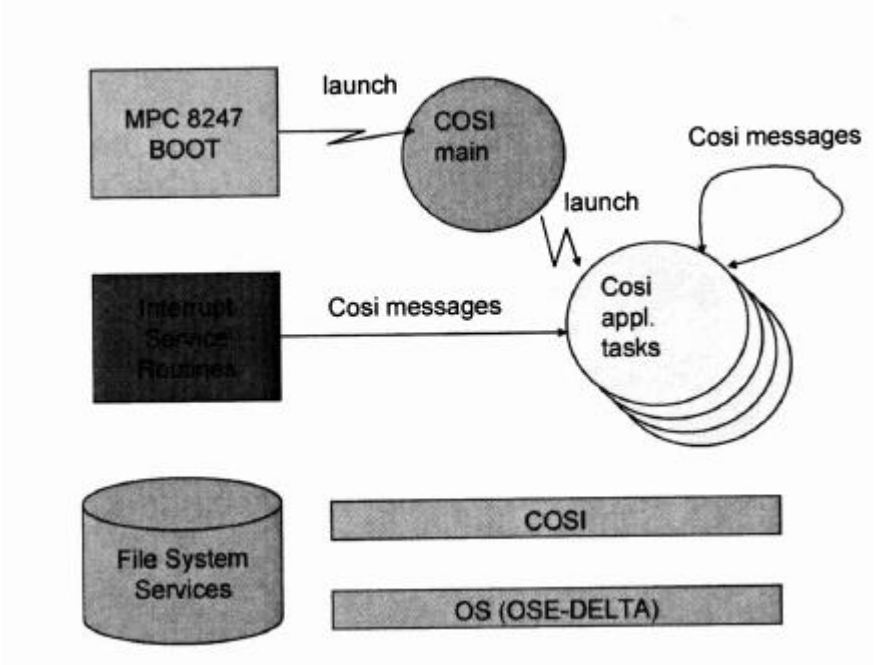

Figure 3. FlexiNT22 SW environment. [Fig3]

Operating system is OSE-Delta, with COSI interface on top of it. The application sybsystems are located in COSI tasks, which communicate with each other using OS message passing technique. [4]

2.5 Operating System

A real-time operating system (RTOS) is a program that must respond to external events within a limited time, and it must be a platform that supports real-time applications and embedded systems. When a user is waiting for the result of a compilation in a personal computer, no error will occur if the result appears a few seconds later than expected.

FlexiNT22 uses OSE (Operating System Embedded) Delta RTOS developed by ENEA, a Swedish based company. OSE Delta is especially designed for distributed and fault-tolerant real-time systems.

OSE uses signals in the form of messages passed to and from processes in the system. Messages are stored in a queue attached to each process. A 'link handler' mechanism allows signals to be passed between processes on separate machines, over a variety of transports.

2.6 TDM Interface

Time Division Multiplexing, a type of multiplexing that combines data streams by assigning each stream a different time slot in a set. TDM repeatedly transmits a fixed sequence of time slots over a single transmission channel. Within T-Carrier systems, such as T-1 and T-3, TDM combines Pulse Code Modulated (PCM) streams created for each conversation or data stream.

G.703 is a standard which originally described voice over digital networks. It's a ITU-T recommendation which is associated with the PCM standard. Voice to digital conversion according to PCM requires a bandwidth of 64 kbps (+/- 100 ppm), resulting in the basic unit for G.703. By multiplication this results in e.g. T1 (24 channels x 64 kbps + 8 kbps adding frame) 1544 kbps and E1 (32 time slots x 64 kbps(E0)) 2048 kbps.

G.704 (framing) is the framing specification for G.703. A carrier can 'steal' a 64kbps time slot (TS0) from a 2.048 Mbps line and use this to provide timing. The result is that 31 time slots are left for data, which equals in a bandwidth of 1.984 Mbps.

G.703 service is typically used for interconnecting data communications equipment such as bridges, routers, and multiplexers. It is transported over balanced (120 ohm twisted pair) or unbalanced (dual 75 ohm coax) cable. [5][6]

2.7 Fast Ethernet Interface

Fast Ethernet is a local area network (LAN) transmission standard that provides a data rate of 100 Mbits/s and is most referred to as 100BASE-T which is 10 times faster than the older Ethernet 10BASE-T specification.

Fast Ethernet is also known as IEEE 802.3u and consists of three separate specifications that describe different physical-layer transmission schemes for Fast Ethernet:

- The first specifications and the most popular is 100Base-TX, which operates over two pairs of copper wire known as Category 5 (CAT5) un-

shielded twisted pair (UTP), or over a shielded twisted pair (STP). One pair is for receiving data signals, and the other for transmitting data signals.

- The second specification, 100Base-T4, operates over four pairs of CAT3, CAT4 or CAT5 copper wires UTP with a signalling system that makes it possible to provide Fast Ethernet signals over standard voice-grade CAT3 UTP cable
- The third specification, 100Base-FX, operates over multimode fibre-optic cable and reaches distances up to two kilometres. [7]

2.8 SHDSL Interface

SHDSL - Single-pair High-speed Digital Subscriber Loop technology enables high speed, symmetrical data transport or simultaneous data and voice transport with N x 64 kbps at data rates up to 2312 kbps by fully exploiting current copper wire infrastructure, used for transmission of telephone conversations. SHDSL technology is especially suited for high bandwidth two-way, symmetrical video transmission for teleconferencing, video remote education and similar purposes.

SHDSL is standardized by ITU-T recommendation G.991.2 in February 2001 also known as G.SHDSL. After major updates to G.991.2 that were released in December 2003, G.991.2 is referred to by the standard's draft name of G.SHDSL.bis or just SHDSL.bis. The updated G.991.2 features:

- support for up to four copper pair connections
- extensions to allow user data rates up to 5696 kbit/s
- support for Dynamic Rate Repartitioning, allowing flexible change of the SHDSL data rate without service interruption. [8]

2.9 SNMP Protocol

SNMP (Simple Network Management Protocol) is a protocol for network management. It makes possible the exchange of management information between network devices. It is part of the TCP/IP protocol suite. SNMP enables network

administrators to manage network performance, find and solve network problems, plan for network growth. [9]

2.10 Telnet Protocol

Telnet is a network protocol used on the Internet. It is typically used to provide user oriented command line login sessions between hosts over a LAN or the Internet. The Telnet protocol is applied on a TCP connection to send data in AS-CIIformat coded over 8 bits between which the Telnet check sequences come. It therefore provides a communication orientated bi-directional system (half-duplex), coded over 8 bits. Telnet clients have been available on most Unix systems for many years, and are available for virtually all types of computers. [11]. Figure 4 below shows the TCP/IP Protocol suite.

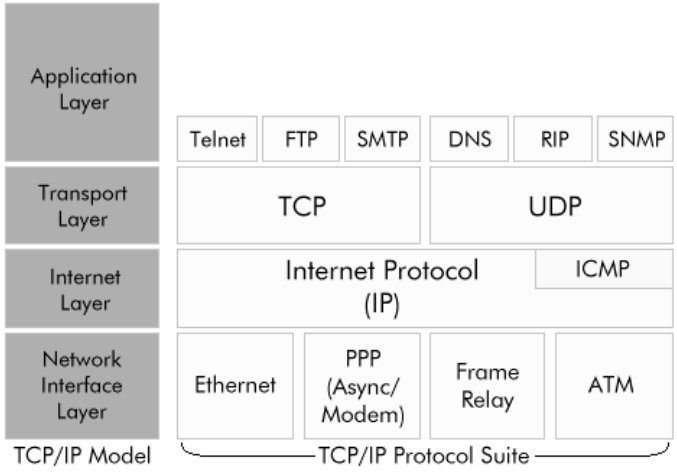

Figure 4. TCP/IP Protocol Suite. [Fig4]

2.10.1 The Network Virtual Terminal (NVT)

Telnet defines a standardized, fictional terminal called the *Network Virtual Terminal (NVT)* that is used for universal communication by all devices. A Telnet client takes input from a user and translates it from its native form to the NVT format to send to a Telnet server running on a remote computer; the server translates from NVT to anything representation the computer being accessed requires. The process is reversed when data is sent from the remote computer back to the user. This

system allows clients and servers to communicate even if they use entirely different hardware and internal data representations. [10]

2.10.2 Options and Option Negotiation

Having Telnet clients and servers act as NVTs avoids incompatibilities between devices, but does so by stripping all terminal-specific functionality to provide a common base representation that is understood by everyone. Since there are many cases where more intelligent terminals and computers may wish to use a more advanced communication feature or service, Telnet defines a rich set of options and a mechanism by which a Telnet client and server can negotiate their use. If the client and server agree on the use of an option it can be enabled; if not, they can always fall back on the NVT to ensure basic communication.

2.11 EFM – Ethernet in the First Mile

Ethernet in the First Mile (EFM), also known as IEEE 802.3ah standard is a new Ethernet technology targeted at access for copper and fibre media. EFM sets goals for a short reach option of at least 10 Mbps up to 750 m, and a long reach option of at least 2 Mb/s up to 2.7 km, but the standard does not limit systems to these rates. Bonding of multiple copper pairs allows much higher throughput, providing a viable alternative for end-users served only by copper.

Ethernet packets arrive through an MII (Media Independent Interface) interface. If incoming data does not fully fill the transport link, flags are added between frames. An optional aggregation layer breaks the packets into variable length fragments which are forwarded to the transmission convergence (TC) sub-layer are encapsulated with 64B/65B framing and transmitted via the (PMA/PMD) to the wire. The receiver side restores the packets and removes padding added at remote end to fill the TC frame.

A cross-connect layer connects between the Ethernet interfaces and physical links so that one physical link can carry signals from multiple Ethernet interfaces, or payload from one Ethernet interface can be carried over multiple physical links.[11]

2.12 ATM – Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) is a technology that provides a single platform for the transmission of voice, video and data at specified quality of service using cell relay technology.

ATM was standardized by ITU-T in 1987. It is based on packet-switching and is connection oriented transfer mode that uses asynchronous time division multiplexing techniques with the multiplexed information flow being organized into small blocks of fixed size, called cells. As shown in Figure 5 a cell consists of 53 bytes, the first 5 bytes contain header information such as connection identifier, while the remaining 48 bytes contain user information, or payload.

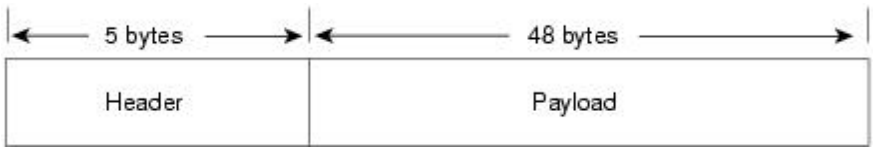

Figure 5. ATM Cell. [Fig5]

ATM networks are used in a variety of environments. For instance, it is widely used in the backbone of Internet Service Providers (ISP) and in campus networks to carry Internet traffic. [12]

2.13 General Web Server

Web servers, also known as HTTP servers, are computer software systems that store and transmit information to web clients. Generally, web browsers include Internet Explorer, Mozilla Firefox, Google Chrome, Apple Safari and Opera. A web server is the repository for web documents, storing the information and their display formats. A web server handles requests and passes documents back to the browser. The browser performs the more difficult work of presenting the text, displaying graphics, generating sound or video and running Java applets. When a browser receives a file from a web server, the server provides the MIME (Multipurpose Internet Mail Extension) type of the file. The browser uses the MIME type to establish whether the file format can be read by the browser's built-in capabilities or not, or whether a suitable plug-in application is required to read the

file. Web servers use HTML to represent hypertext documents on a web browser without a helper application. The web documents on the server can be static (e.g., stored as a read-only document) or created dynamically in response to parameters supplied by the clients. General web servers, which were developed for general purpose computers such as NT servers or Unix and Linux workstations, typically require megabytes of memory, a fast processor, a pre-emptive multitasking operating system, and other resources. [13]

2.14 Embedded Web Server

A web server can be embedded in a device to provide remote access to the device from a web browser if the resource requirements of the web server are reduced. Underlying this reduction of the resource requirements of the web server is typically a portable set of code that can run on embedded systems with limited computing resources, and which can be utilized to serve the embedded web documents to the web browsers. This type of web server is called an Embedded Web Server (EWS). By embedding a web server into a network device, it is possible for an EWS to provide a powerful web-based management user interface constructed using HTML, graphics and other features common to web browsers. EWSs are used to convey the state information of embedded systems, such as a system's working statistics, current configuration and operation results, to a web browser. EWSs are also used to transfer user commands from a web browser to an embedded system. The state information is extracted from an embedded system application and the control command is implemented through the embedded system application. In many instances, it is advisable for embedded web software to be a lightweight version of web software. Embedded environments are often more limited in resources than non-embedded systems, and hence increased efficiency of resource usage is in order. Also, by limiting the functionality of certain embedded applications, security and reliability can be enhanced. For network devices, such as routers, switches and hubs, it is possible to place an EWS directly into the devices without additional hardware. As more devices (such as home appliances, manufacturing devices and medical instruments) are connected to the Internet, I

believe that EWS technology will be essential to making Internet devices more manageable. [14]

2.15 HTTP – HyperText Transfer Protocol

The Hypertext Transfer Protocol (HTTP), which utilizes TCP/IP for its transport, is used by web clients and servers to exchange hypertext information between the two. HTTP is a client driven protocol, simply meaning that all communications are initiated with the client. To initiate a session, a web client makes TCP open requests to the well-known port number 80. Once the web server has accepted the request, it expects one of three basic operation commands (GET, HEAD and POST) and an object address. Object addressing is done using a Universal Resource Locator (URL). After processing the command, the server responds to the request with the header identifying the body content type, encoding, length and other supporting information, followed by the requested object body. Finally, the server closes the session by issuing a TCP close request.

Figures below show the difference between HTTP/1.0 and HTTP/1.1 when processing a TCP connection request. [15][16]

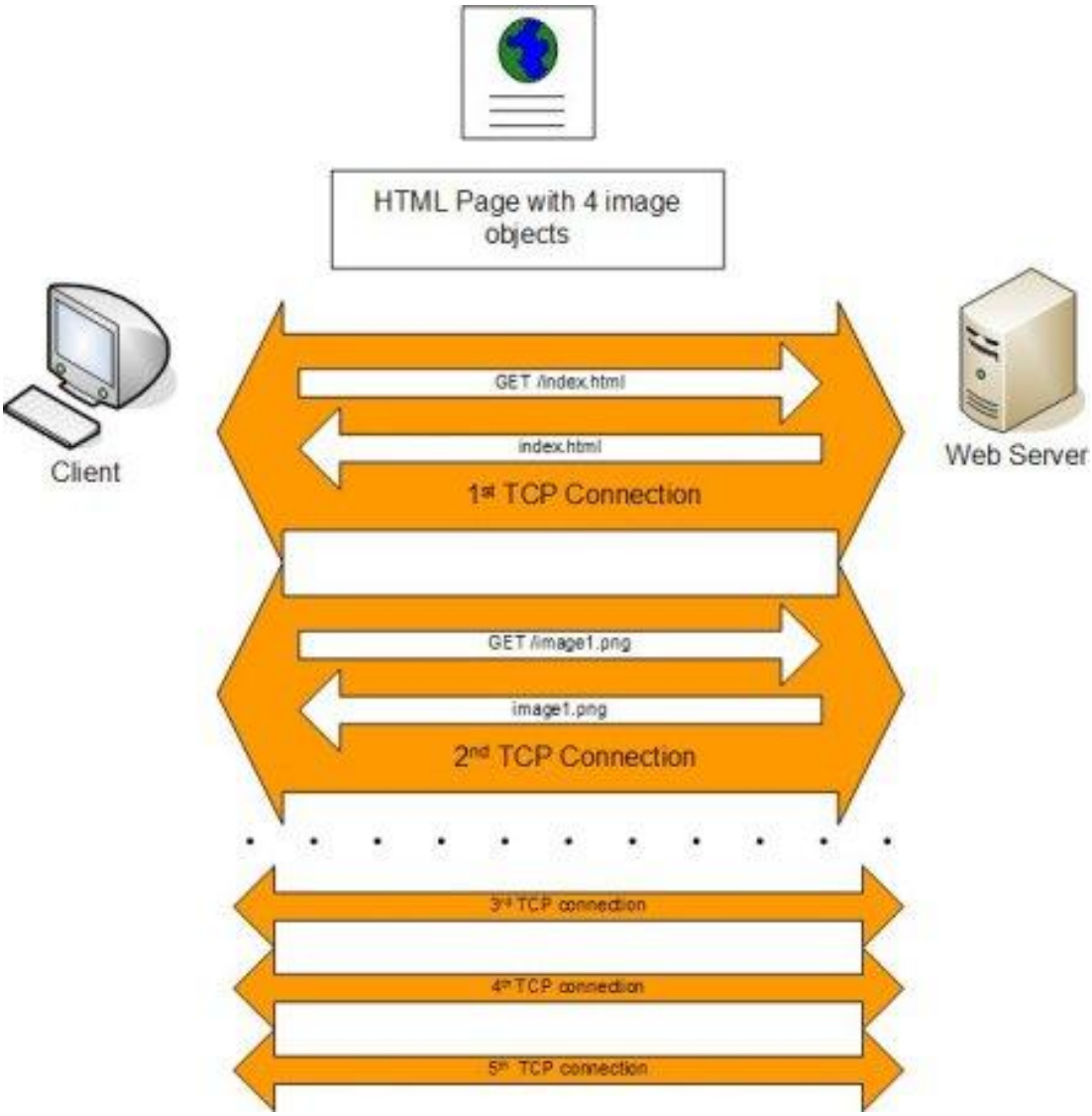

Figure 6. HTTP/1.0 One TCP connection per request. [Fig6]

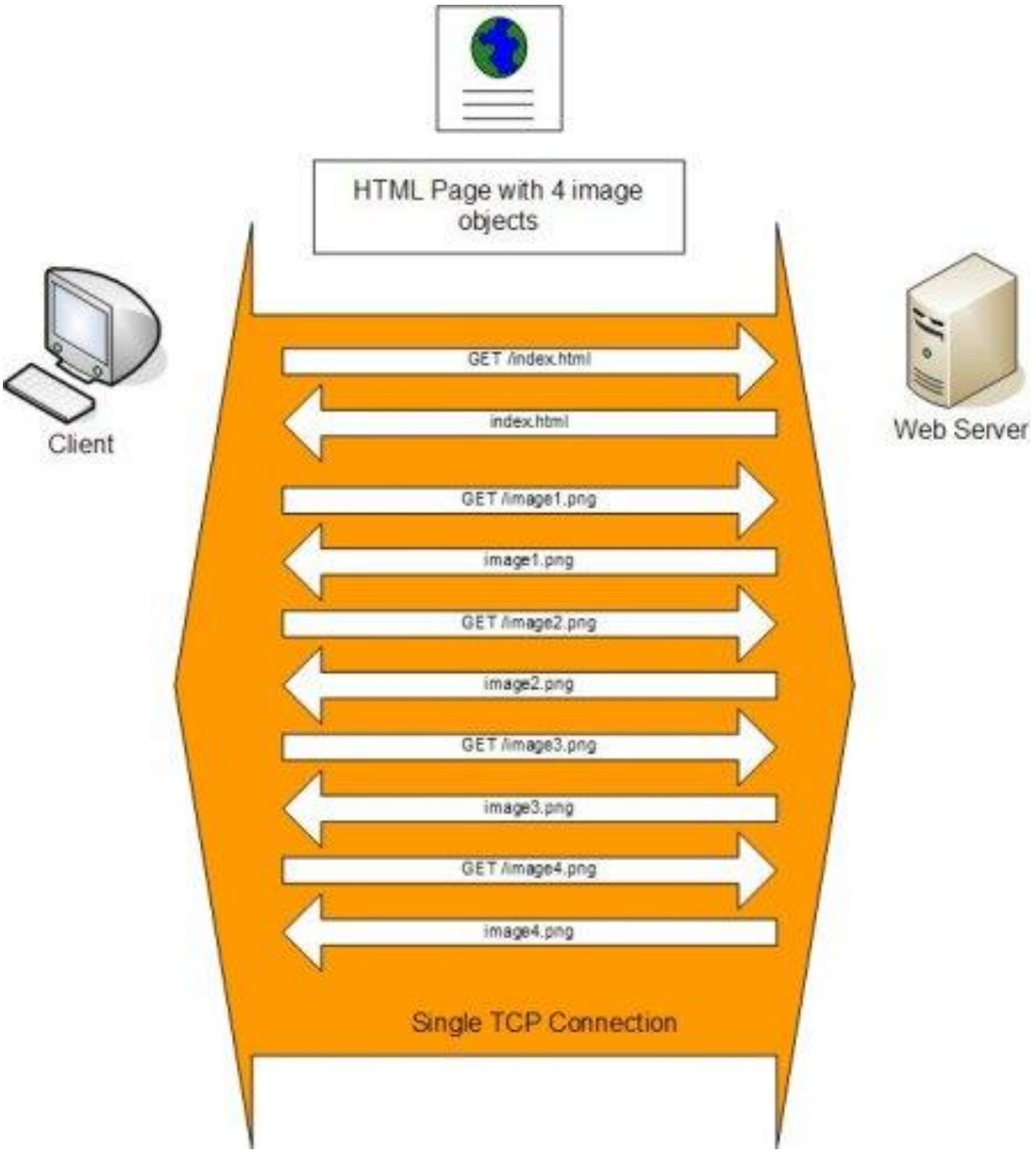

Figure 7. HTTP/1.1 Multiple requests per TCP connection**.** [Fig7]

3 WEB SERVER SELECTION

3.1 Requirements for Embedded Web Servers

Traditional web servers are designed to serve up static disk-resident web pages to end-users surfing the Internet, and are typically deployed on high-end Unix or Windows workstations. These technologies are unsuitable for the development of interactive GUIs because their common gateway interface (CGI), or similar application interfaces, require substantial development effort and do not facilitate rapid prototyping of the product's "look-and-feel." Furthermore, traditional web server implementations are unsuitable for embedded real-time software environments because they use megabytes of memory, consume significant CPU cycles, and require preemptive multitasking OSes.

The requirements for embedded web servers are quite different. The most important requirement for an embedded web server technology is its ability to rapidly develop and integrate a dynamic and interactive web based interface with the embedded firmware. CGI and other traditional technologies simply cannot meet these requirements. The embedded web server software itself must have a small memory footprint, both in code size and run-time memory usage. Since the web content will likely be stored in Flash memory, low-overhead data compression techniques are also important. In non-pre-emptive real-time environments, embedded web server software must be able to handle multiple concurrent HTTP requests without degrading system performance and without consuming more than a reasonable and bounded amount of CPU cycles at a time. [17]

3.2 Embedded Web Server Solution Survey

According to the study regarding embedded web servers, there are many commercial and open source embedded web servers' products on the market for web appliances. Based on the requirements of the study I have chosen some comparisons between various embedded web servers based on their features.

Table below shows features of selected embedded web servers.

Table 1. Comparisons between various embedded web servers.

Server	Memory footprint	Development language	Operating System support	Dynamic content	Authentication	Portability	Licence
Klone	110KB static 350KB dynamic	C	Linux, NetBSD, OpenSD, FreeBSD QNX, VxWorks, Darwin/MacOSX, Windows	C/C++ in usual HTML scripting style		yes	GPLv2 or commercial 900€
Embedthis AppWeb	400KB	C	Windows, Linux, FreeBSD, Mac OS X	Server-side JavaScrip, in-memory CGI, CGI/1.1 PHP	SSL, Basic	yes	GPL or commercial ~ 6300€
Nichestack HTTP Server	10-13KB	C	Any 16 or 32bit embedded	CGI SSI	Basic, Digest, Login	yes	4 800 €
RomPager	10KB ROM 4KB RAM	C	Portable to any OS	CGI	Basic	yes	basic ~7400€ stand ~14800€ advanced~22000€
GoAhead	60KB	C	Win32/WinCE, Embedded Linux, Linux, VxWorks, QNX, Lynx, Ecos, Novelle Netware, MAC OS X	ASP, In-proccess CGI, CGI, Embedded Javascript, open scripting architecture	Basic, SSL, Digest	yes	open source or commercial
Seminole	?	C++	eCos, VxWorks, POSIX, Win32, uCOS	Template system, CGI	SSL Digest	yes	basic 1500€ with application framework 2600€
Http web server	10KB ROM 4KB RAM	C	Portable to any OS	CGI	Basic	yes	basic ~3700€ stand ~ 7400€ pro ~ 22000€

Referring to the Table 1, here we have chosen 7 embedded web servers into our comparison which are KLone, Embedthis App Web, Nickestack HTTP Server, Rompager, Goahead, Seminole and HTTP web server. All the servers are developed and technically maintained and upgraded using C/C+ programming language with portability. Memory footprint is the measurement of consumption of memory for a running application. Larger programs often require big memory occupation. In case of a small memory constrain, we implement low memory footprint programs. Sometimes designers even sacrifice efficiency in order to downsize the footprint to fit into the available RAM. Looking at the Table 1, we find that KLone and Embedthis App Web require a relatively big memory to support its function with range from 110kb to 400kb. On the other hand, the remaining 5 serves have an average footprint of 10 to 20kb. KLone has the widest content coverage; Embedthis App Web and GoAhead both sharing a big content portfolio as well. At the end, we will also review the difference of each server from commer-

cial point of view. HTTP web server and Rompager offer us the most expensive license fee, topping 22000 Euros for its highest level service, Embedthis App Web together with Nichestack HTTP Server stay in the middle level with roughly 5000 Euros . KLone, Semonole and GoAhead have the cheap offer here ranging from 0 to 2000 Euros.

3.3 The most Appropriate Solution

Looking at the features of **Table 1** above, comparing the EWSs, the three most appropriate solutions for this study were chosen: KLone, Nichestack HTTP Server and GoAhead web server. We will take a short look on each web server for more details.

3.3.1 KLone

KLone is an open-source (GPL) Embedded Web Server. It is a fully-featured, multi-platform framework which allows both static and dynamic web pages to be written in C/C++ (with the usual scripting style: <% /* code */ %> mixing HTML and C/C++ code). The pages can then be embedded (in compressed and/or encrypted form) into a single executable binary file that also contains KLone's high-performance HTTP/S server. Given its nature, it can be linked natively to any C/C++ library (database, XML, graphics, etc.), without an intermediate layer, and it is especially suited for low-resource (embedded) systems.

KLone is open source software released under a double license: commercial and GPLv2. [18]

Here are some key features of KLone [18]:

· Multiplatform HTTP and HTTP/S

· Small memory footprint

· Multiple content suppliers (on-disk file system, embedded file system, CGIs)

· Open source or royalty-free commercial license

· Automatic setup and build framework

· Full source code available

· Dynamic page scripting in C/C++

Dynamic pages are written by mixing HTML and C/C++ code embraced within **<% code %>** tags.

During the build phase, the framework translates, compiles and links the pages to the web server providing fast script execution, tiny memory occupation, embeddable content and native usage of thousand of C/C++ libraries from within web scripts.

Example of running KLone application

Here the examples presented are taken from Koanlogics website, the reason is that during this testing phase I myself did not take screenshots of my testing procedure, hence I use these examples.

We create an empty directory and download a ready-made version from Koanlogic website and extract the files into this created directory:

Now we run make (GNU make must be used).

The framework will compile and create a simple application ready to be customized.

We run the daemon to see KLone in action:

```
$ ./kloned
```

Now we connect to the running webapp with the browser: http://localhost:8080/. The webapp directory holds the content of basic web application and each new web file created will be saved in this directory.

The page will look like in Figure 8.

Figure 8. Example page of running KLone application.

Adding new page with dynamic content [18]

The webapp directory contains all the files for creating a web application, this directory is automatically compiled, bundled into the KLone daemon executable with KLone HTTP server in it. To add a new page, it must be created inside the webapp directory and we have to run “make” again to compile, the same applies for each file modification or removel, “make” has to be done and compile so changes will take effect. If the server is running we kill it (*killall kloned)*, than rebuild the daemon (*make*) and restart it (*./kloned*).

As an example we will build a basic directory browser.

The page root.kl1 displays the content of the root directory of the computer running the KLone server. The kl1 extension is used by KLone server for web page that contains dynamic contents.

```
<%!
    #include <string.h>
    #include <sys/types.h>
    #include <dirent.h>
%>

<html>
<body>
<ul>
<%
    DIR *dirp;
    struct dirent *dp;

    dirp = opendir("/");
    while ((dp = readdir(dirp)) != NULL)
        io_printf(out, "<li>%s</li>", dp->d_name);
    closedir(dirp);
%>
</ul>
</body>
</html>
```

The output will be the list of files and directories of the computer. There are two blocks of C code enclosed in two different kinds of KLone blocks:

<%! %>. This is the declaration block, all #includes, global variables and function declarations go in here.

<% %> This is the block to be used inside HTML to generate dynamic output from C code. The function *io_printf(out, "
%s
", dp->d_name);* prints the list of directories to the client. [19]

The ***io_printf(out, …)*** function is a global variable defined within KLone which is most used in the implementation part. It is a function that is used to print data to the output stream that is to the browser. An example below is shown:

```
io_printf(out, “%s”, “this is a string”);

io_printf(out, “The value is %d”, 35);

const char *input_text = “hello!”;

io_printf(out, “text: %s”, input_text);
```

Three examples are shown above how the io_printf (out, …) function can be used.

KLone was chosen as the most suitable embedded web server for FlexiNT22 for the following reasons: as FlexiNT22 is a monolithic application KLones features fit the best for this solution, as the written pages in KLone are embedded into a single executable binary file, HTML pages can be written inside C/C++ code, it works on top of linux, and the CLI commands of the modem can be emulated to a test bench.

3.3.2 Nichestack HTTPServer

NicheStack HTTPServer is an embedded web server designed specifically to optimize size and performance without sacrificing important security features found in conventional web servers. It is easily integrated into any networked device architecture to dramatically simplify device deployment and remote management.

HTTPServer supports access to files stored in a local physical or virtual file system. These files may include embedded function calls for the creation of dynamic content which are executed as the file is converted to an HTTP stream.

HTTPServer provides features such as CGI and SSI, HTML compression through substitution and expansion of frequently repeated ASCII sequences, SSL and TLS, basic and digest authentication, includes NicheFile to enable file storage in RAM, ROM or Flash, dynamic content with SSI and CGI active hooks.

The optional HTML compiler is a valuable timesaving tool for developers that produces ready to use C code, linking variables to forms and dynamic HTML which is compatible with the run time systems of HTTPServer. The HTML compiler compresses standard HTML files and converts them into C language structures that are compatible with the VFS. [19]

Nichestack HTTPServer with its multifunction features is one of the candidates to be run, but because of the limited time of the project, and considering the amount

time taken for implementation, and not to forget the licence cost, it wasn't possible. That's the reasons it didn't make to final tests.

3.3.3 GoAhead Web Server

The GoAhead Web Server is a fully functional, small memory footprint, open source, standards-based embedded web server designed for cross-platform support. Its functionality includes Active Server Pages (ASP) for delivering dynamic HTML pages, in-process CGI, embedded JavaScript and an open, extensible scripting architecture, SSL 3.0 and DAA..

GoAhead Web Server supports HTTP 1.0, and some performance enhancements of HTTP 1.1 such as persistent connections. GoAhead Web Server, with free source code, has no proprietary lock-ins.

To run the GoAhead Webserver a TCP/IP stack, an event timer, and approximately 60KB of RAM are required.

An Active Server Page (ASP) is an HTML page that is processed on the web server before the page is sent to the user.

Server-side JavaScript is a very effective way to create dynamic web pages without having to recompile whenever you wish to change the data shown in the web pages.

The JavaScript is embedded in the web page, using Active Server Pages, and is run by the GoAhead Web Server before the web page is sent to the user's browser. The resulting page is small and transmits quickly to the user's browser even over slow modem links. The JavaScript implementation allows the creation of objects to represent device data for easy scripting access. [20]

GoAhead Webserver with its rich features and functionality would've been a good choice to run, but because of the lack of contact support it didn't make to final tests.

4 WEB GUI DESIGN

4.1 Interface Design

In this project it's important to have a structure of the web GUI and the main point is to make three working web pages using a test bench.

In this study a web based GUI is composed of three frames: the top frame, the menu frame and the info frame, which will be described below.

Figure 9 shows the structure of user interface.

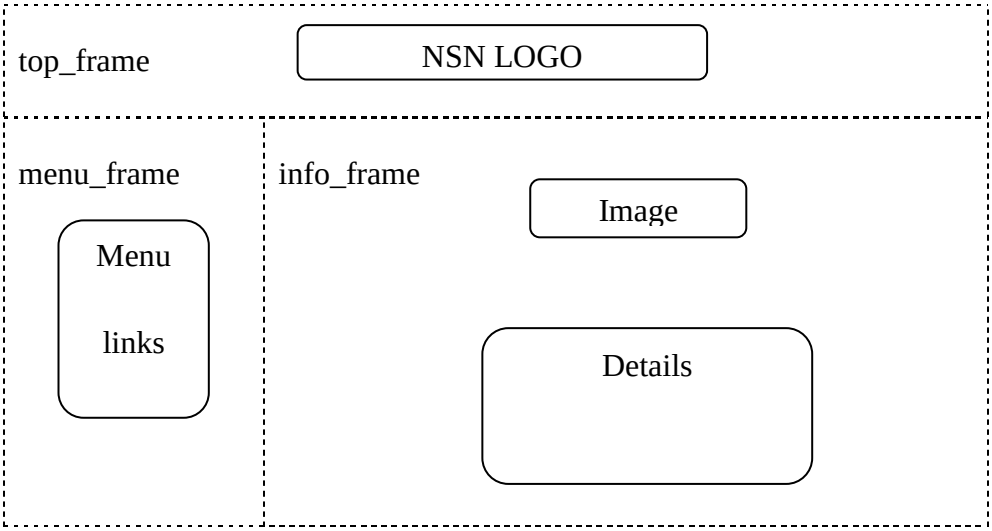

Figure 9. User interface design structure.

The **TOP FRAME** - This frame is designed to contain the official NSN LOGO and will be always within the browsers viewport regardless which link is clicked.

The **MENU FRAME** - This frame contains the menu links as shown in the figure below, which remains always within the browser viewport. Clicking on links in this frame will load new pages into the info frame. The buttons for the links were made with PhotoShop in 2 different colours, as default yellow buttons, and when clicked the button will change into orange colour or as to say highlighted as shown in Figure 10, and when *Configure* link is clicked it has sub links and they will be highlighted as shown in Figure 10.

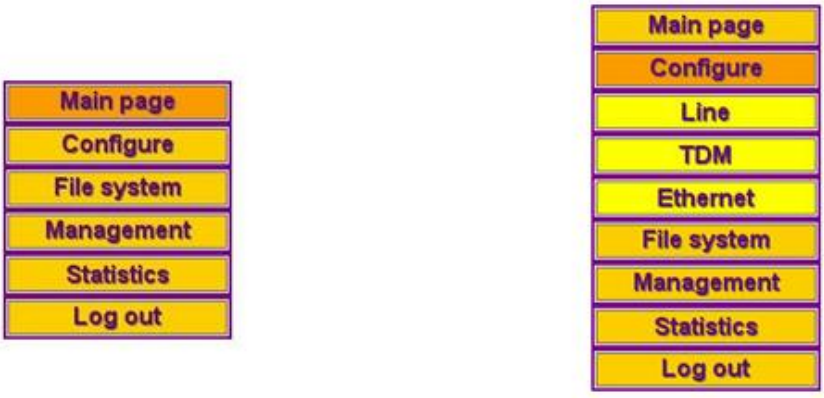

Figure 10. All menu buttons and menu buttons with sublinks.

The HTML code captured from web browser will look as shown in Figure 11.

```
<!-- MENU FRAME  -->
<HTML>
<BODY>
<a href ="index.kl1?frame=frame&menu=1&info=1" target="showframe" scrolling="no" noresize><img src=images/pic_main_orange.jpg alt="Main"></a><BR>
<a href ="index.kl1?frame=frame&menu=2&info=2" target="showframe" scrolling="no" noresize><img src=images/pic_configure_yellow.jpg alt="Configure"></a><BR>
<a href ="index.kl1?frame=frame&menu=10&info=10" target="showframe" scrolling="no" noresize><img src=images/pic_file_sys_yellow.jpg alt="File system"></a><BR>
<a href ="index.kl1?frame=frame&menu=11&info=11" target="showframe" scrolling="no" noresize><img src=images/pic_manag_yellow.jpg alt="Management"></a><BR>
<a href ="index.kl1?frame=frame&menu=12&info=12" target="showframe" scrolling="no" noresize><img src=images/pic_statistics_yellow.jpg alt="Statistics"></a><BR>
<a href ="index.kl1?frame=frame&menu=13&info=13" target="showframe" scrolling="no" noresize><img src=images/pic_log_out_yellow.jpg alt="Log out"></a><BR>
</BODY>
</HTML>
```

Figure 11. Menu frame HTML source code.

The coding will be explained in more detail in next chapter.

The **INFO FRAME** - This frame can be scrolled and contains the main content pertaining to whatever link the user clicks on in the menu frame on the left. By default when loaded this frame will show Main page that contains at the top an image of the modem and below is created a table that contains the main information about the modem. This applies to the other links when clicked, the page will be loaded in this frame with corresponding information.

4.2 Web page styles and formatting

The pages were created using plain HTML including frames, tables, forms with style sheets (CSS) according to NSN styles (fonts and colours), but without using any JavaScript or ASP scripts, as they are not suitable with the software.

4.3 Web Development Tools

CoffeeCup HTML Editor was chosen as the main development tool for html coding. This is due to fact that it's easy to learn and use, has built in tags which made it less consuming time and the other feature that makes it easier to read is the Preview tab, it's next to the Code Editor tab, we are able to see how the page will look like in the browser.

5 WEB GUI IMPLEMENTATION

5.1 Interface Implementation

Implementation involves coding the various parts of the application. All the modems CLI commands were collected and put them in a C file test bench as functions where they are called later from the index.kl1 file that is associated with the KLone server. The coding of pages was written in Notepad++ which is a powerful text editing tool, as for compiling and testing was done under UNIX environment using Putty an SSH client and nano text editor.

As explained above in chapter 4.1 Interface Design, Figure 9, we see that the main window is divided in three frames: top, menu and info frames. To create these frames are used functions, which contain mix of C and HTML code. These functions are inside the index.kl1 file and each step will be explained in detail how the pages are created by using examples of coding.

5.2 Client-Server Communication

After introduced all the tools and steps in chapter 4 and as explained in chapter 3 KLones functionality, the next step is to open a web browser, type the IP address and port number, the default port number is 80 so it doesn’t need to be specified in the address bar, but if the port number is other than the default port than the URL address should be like http://localhost:8080 where 8080 is the specified port number. Figure below shows a sequence diagram of client-server communication that will be explained in more details below.

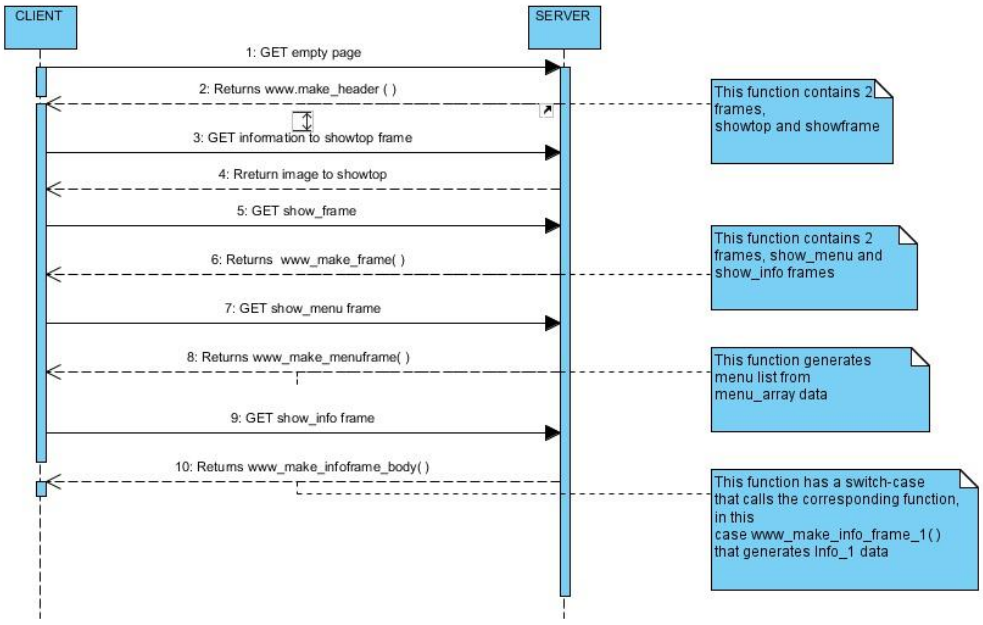

Figure 12. Sequence diagram between client and server.

When we open the browser and write the IP address of the modem, first the client requests an empty page.

Server returns the www_make_header( ) function, below will be shown main parts of the function.

This function generates two pages (frames) named showtop and showframe as shown in Figure 14. As the function name says, this function creates the main window puts the HTML tags in it, and divides the page in two frames, which are showtop and showframe. Below will be shown an example how the frame is created using variable *frame_identifier[ ]*, which is defined in the global data section and has the values {frame, frame, menu, info}. This variable table is used to identify each frame names in a logical way as shown in the Figure 13 below . The top part with orange colour has the value “frame” the same goes for the others as seen in the in Figure 13. The output of the index.kl1 will be as index.kl1?frame=frame, meaning that two frames are created as mentioned above showtop and showframe, can be seen in the code below.

Figure 13. frame_identifier[ ] variable.

```
void www_make_header(void)
{
     /* Here the function creates the HTML tags */
      io_printf(out, "<HTML>\n");
      io_printf(out, "<HEAD>\n");
      io_printf(out, "<META HTTP-EQUIV=\"refresh\">\n");
      io_printf(out, "<META HTTP-EQUIV=\"Content-Type\" CON-
      TENT=\"text/html;
             charset=utf-8\">\n");
      io_printf(out, "</HEAD>\n");

      io_printf(out, "<frameset rows=\"20%,*\" border=\"0\">\n");
      io_printf(out, "<frame src=\"images/nsn_logo_top.jpg\"
      name=\"showtop\"
             scrolling=\"no\" noresize name=TOP>\n");
      io_printf(out, "<frame src=\"index.kl1?%s=%s\"
      name=\"showframe\"
             scrolling=\"no\" noresize name=showFRAME>\n",
             frame_identifier[0],frame_identifier[1]);
      io_printf(out, "</frameset>\n");
      io_printf(out, "</HTML>\n");

      return;
}
```

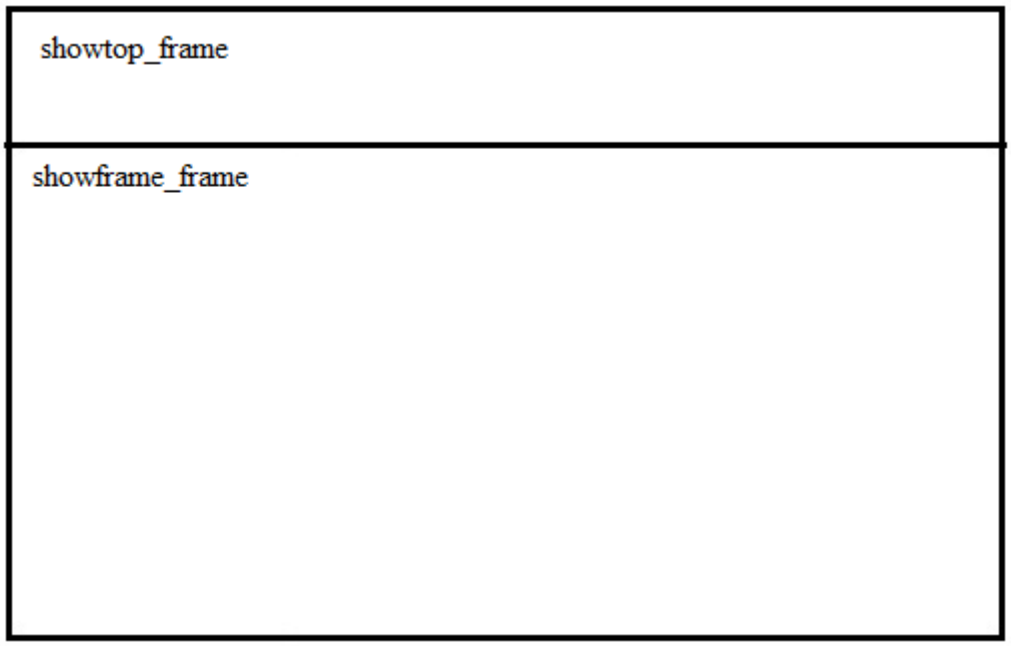

Figure 14. Showtop and showframe.

Next the client requests the information for showtop frame. Server returns the image nsn_logo_top.jpg and puts it in showtop_frame, as shown in the code above.

Next the client requests the information for "showframe" frame. Server returns the contents of the www_make_frames( ) function, part of code below shows how the frames are created.

This code generates two other frames inside showframe: ***showmenu*** and ***showinfo*** frames as shown in Figure 15. In addition to to *frame_identifier[ ]*, are added *menu_identifier[ ]* and *info_identifier[]* which are also defined in the global data section, to identity the two other frames that are created. The variable *menu_identifier[ ]* is used to identify the menu navigation links according to menu level, and *info_identifier[ ]* is used to identify which page will be called according to parameter given, for example the output of showmenu is index.kl1?frame=menu&menu=1&info=1 and showinfo is index.kl1?frame=info&menu=1&info=1. As yet the output is not complete and doesn't show any data, but is further below explained until the final part is reached.

```
io_printf(out, "<frameset cols=\"176,*\">\n");
io_printf(out, "<frame src=\"index.kl1?%s=%s&%s=%s&%s=%s\"
          name=\"showmenu\">\n",frame_identifier[0],frame_identif
          ier[2],menu_identifier[0],menu,info_identifier[0],info)
          ;
io_printf(out, "<frame src=\"index.kl1?%s=%s&%s=%s&%s=%s\"
          name=\"showinfo\">\n",frame_identifier[0],frame_identif
          ier[3],menu_identifier[0],menu,info_identifier[0],info)
          ;
io_printf(out, "</frameset>\n");
```

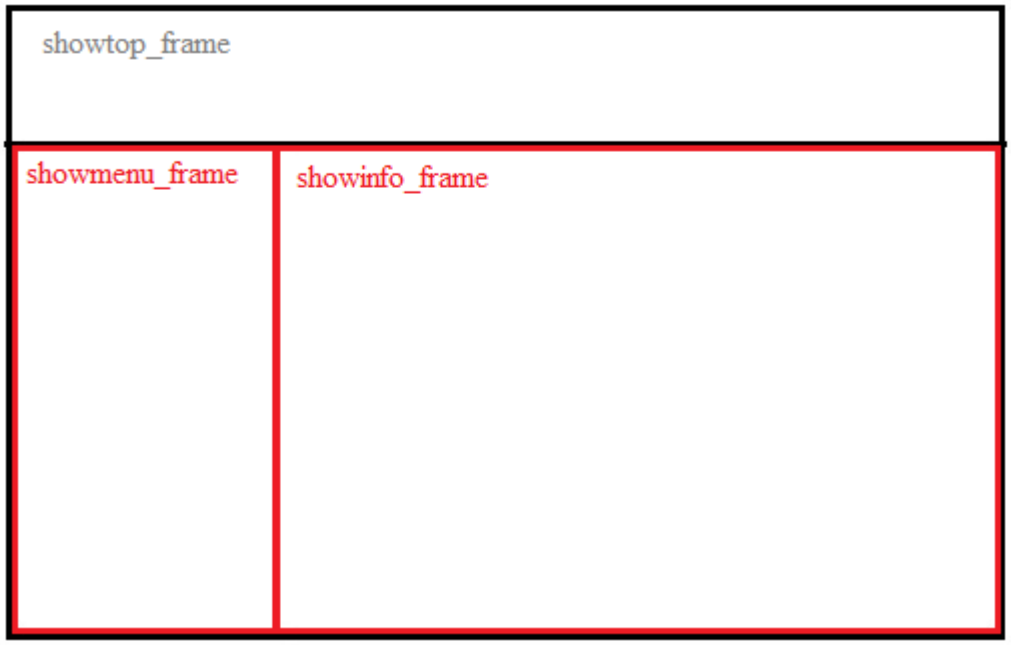

Figure 15. Finished interface with three frames.

Next the client requests the information of showmenu frame ⇔ src=index.kl1?frame=menu&menu=1&info=1 which calls the www_make_menuframe( ) function. Server returns the contents of the www_make_menuframe( ).

This function generates the menus from the menu_array data table (menu_array table is defined in the global data section), it puts the image on each menu link and highlights the clicked link. Figure 10 above in chapter 4.1 Interface Design shows the generated menus.

Level mentioned in the comment blocks, describes navigation menu as tree view, “0” means main menu link, like ”Main”, “1” means sub link, like “Line”, “2” means sub link of sub link, like “TDM0”.

```
/* explanation of array numbers " " as in order: "LEVEL" "Visible
name","menu args","info args", "default MENU image", "active MENU
image"*/

const char *menu_array[][6] = {
{"0","Main","1","1","images/pic_main_yellow.jpg","images/pic_main_
orange.jpg"},
{"0","Configure","2","2","images/pic_configure_yellow.jpg","images
/pic_configure_orange.jpg"},
{"1","Line","3","3","images/pic_line_yellow.jpg","images/pic_line_
orange.jpg"},
{"1","TDM","4","4","images/pic_tdm_yellow.jpg","images/pic_tdm_ora
nge.jpg"},
{"2","TDM0","5","5","images/pic_tdm0_yellow.jpg","images/pic_tdm0_
orange.jpg"},
{"2","TDM1","6","6","images/pic_tdm1_yellow.jpg","images/pic_tdm1_
orange.jpg"},
};
```

Next is shown part of the code how menu frame is called.

The output values of showframe will be as follow: index.kl1?frame=frame&menu=1&info=1 which loads the default main page, and on each menu link will be put a corresponding picture as defined in menu_array data table.

```
io_printf(out, "<a href =\"index.kl1?%s=%s&%s=%s&%s=%s\" tar-
get=\"showframe\"
          scrolling=\"no\"          noresize><img          src=%s
          alt=\"%s\"></a><BR>\n",frame_identifier[0],frame_identi
          fier[1],menu_identifier[0],menu_array[temp][2],info_ide
          ntifier[0],menu_array[temp][3],temp_select_pic,
          menu_array[temp][1]);
```

Next and the last step, the client request the information for the showinfo frame ⇔ src= index.kli?frame=info&menu="1"&info="1" which calls the www_make_infoframe_body ( ) function.

Server returns the contents of the www_make_infoframe_body( ) as shown below main parts of the code. This function creates the HTML head tags for each page and it will load as well the stylesheet and background image, this part will not change, but it will be part of every page that is generated as a whole, so at the end all the functions will combine and at the client side we will see the HTML source code.

```
io_printf(out, "<HTML>\n");
io_printf(out, "<HEAD>\n");
```

```
io_printf(out,    "<link    rel=\"stylesheet\"    href=\"style.css\"
type=\"text/css\">\n");
io_printf(out, "</HEAD>\n");
io_printf(out,                      "<BODY                      back-
ground=\"images/texture_0005_rgb.jpg\">\n");
```

After the showinfo frame head tags are created the next part is the main contents of each page, as shown above highlighted text (index.kli?frame=info&menu="1"&info="1") determines which page will be loaded by switch-case statement as below.

The switch-case statement selects what page will be loaded depending by the parameter given by client which is info=”x” where “x” is a changeable value. As mentioned above index.kli?frame=info&menu="1"&info="1" in this example case 1 will be printed with the www_make_info_frame_1 function that has the contents of *Main page*. Same goes for the two other functions, www_make_info_frame_2 has the contents of *Configure* page and www_make_info_frame_3 that has the contents of *Line* page.

```
switch (tempi){
case    1:    www_make_info_frame_1(frame_string,    menu_string,
info_string);
      break;
case    2:    www_make_info_frame_2(frame_string,    menu_string,
info_string);
      break;
case    3:    www_make_info_frame_3(frame_string,    menu_string,
info_string);
      break;
      }
/* Here we close the body and html tags (appears on every page) */
io_printf(out, "</BODY>\n");
io_printf(out, "</HTML>\n");
```

6 RESULTS

6.1 Main Page

The Main Page is shown first when using a web browser. The currently shown page is shown highlighted on the list on the left on the menu page. Clicking an item on the menu list (Configure, File System, Management, Statistics, Log out) takes you to the corresponding page. Figure 16 shows the main page.

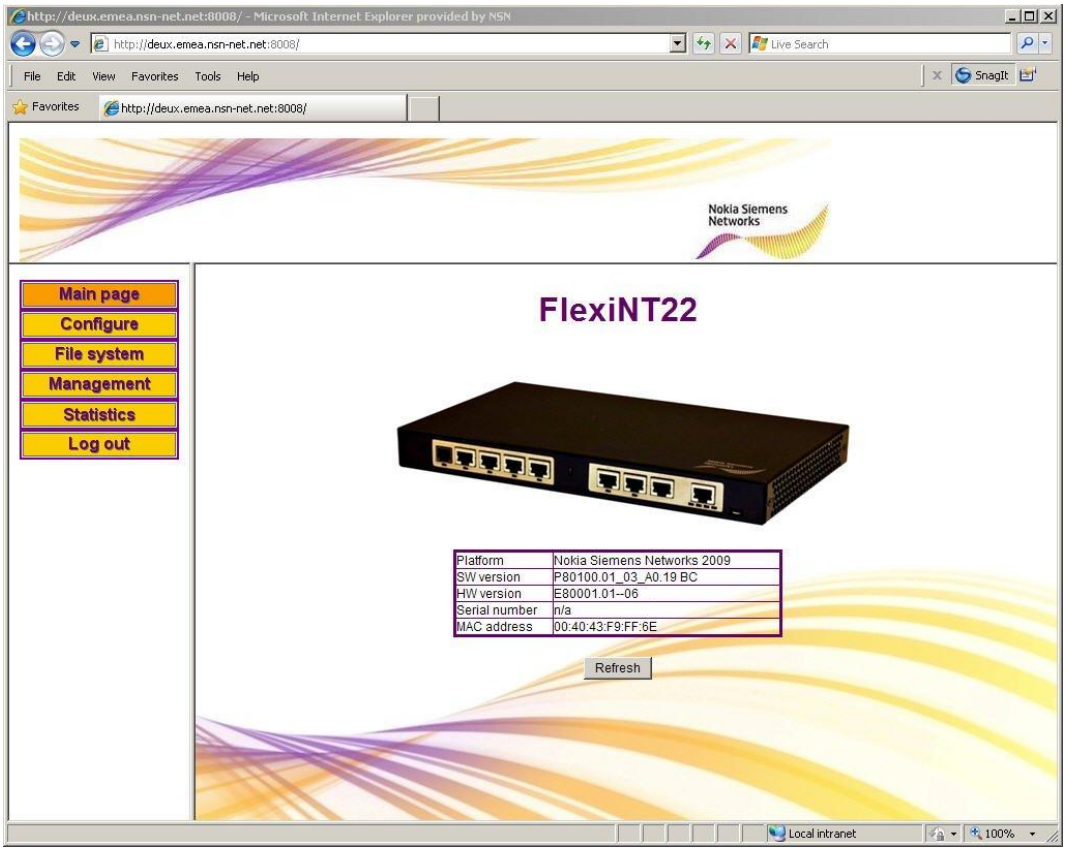

Figure 16. Main page.

The **Main Page** shows the information on Platform, Software and Hardware versions, serial number and MAC address of the device.

6.2 Configure page

The Configure page has three subpages: ***Line, TDM and Ethernet.*** The purpose of the study is to make the Line page work with the line modes as shown in Table 2 below:

Table 2. Line modes.

LINE MODES		
LINE	FRAMING	DUAL-BEARER
CO Manual CP CO Auto	ATM EFM	ON OFF

CO Auto mode is not implemented in this study. Figure 17 below shows conf status page.

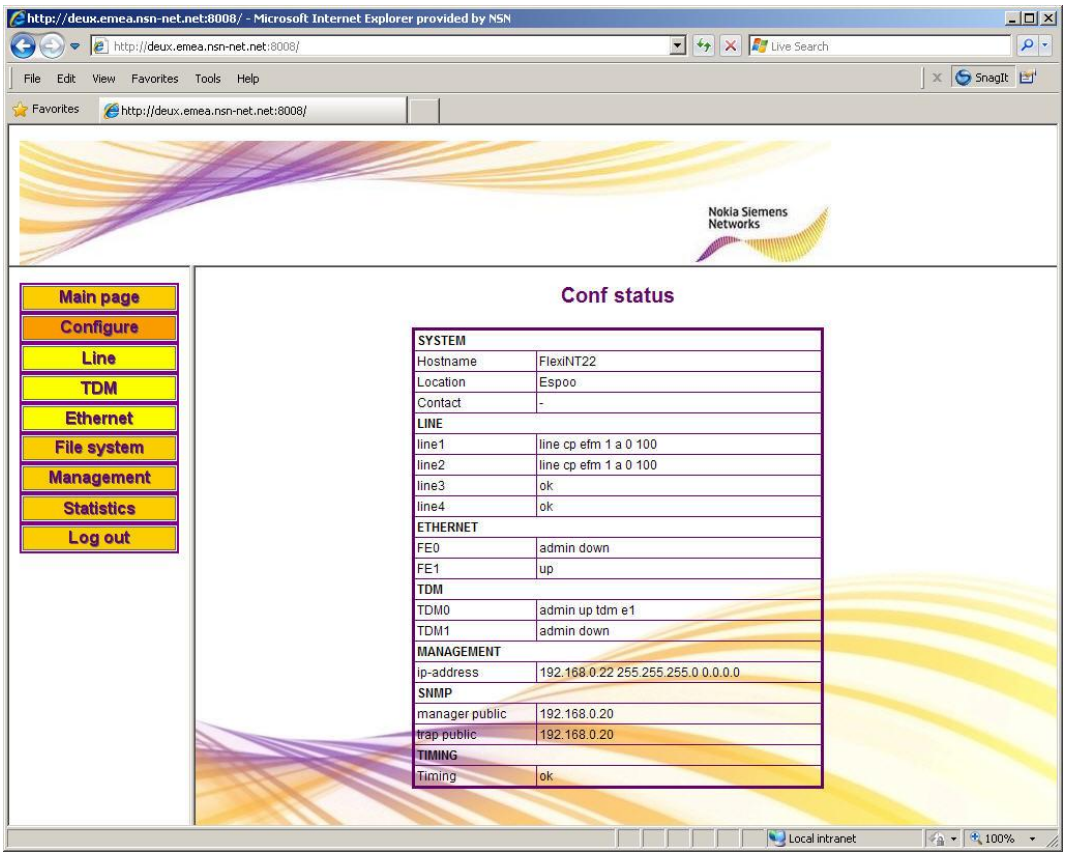

Figure 17. Configure page.

The configure page shows the current configuration status of the device.

- System status including Hostname, Location and Contact information
- Line status including line1 to line 4
- Ethernet status including FE0 and FE1
- TDM status including TDM0 and TDM1

- Management status including IP address
- SNMP status including manager public address and trap public address
- Timing status

6.3 Line page

On the Configure Line page we are able to configure the line mode choosing between “CO(Central Office) Manual”, “CP(Customer Premises)” and “CO Auto” (not implemented), between framing modes “ATM” or “EFM”, and choosing dual-bearer ON or OFF, which are shown on the first table. After choosing the desired configuration mode, by clicking on the “SET CONF” button, the changes will apply and on the bottom page will appear the configuration form according to line mode choice, in this case as shown with radio buttons chosen “CO Manual”, “ATM”, “DB-OFF”. Choosing between line and framing modes and applying the changes with “SET CONF” button. Figure 18 shows Line configuration page.

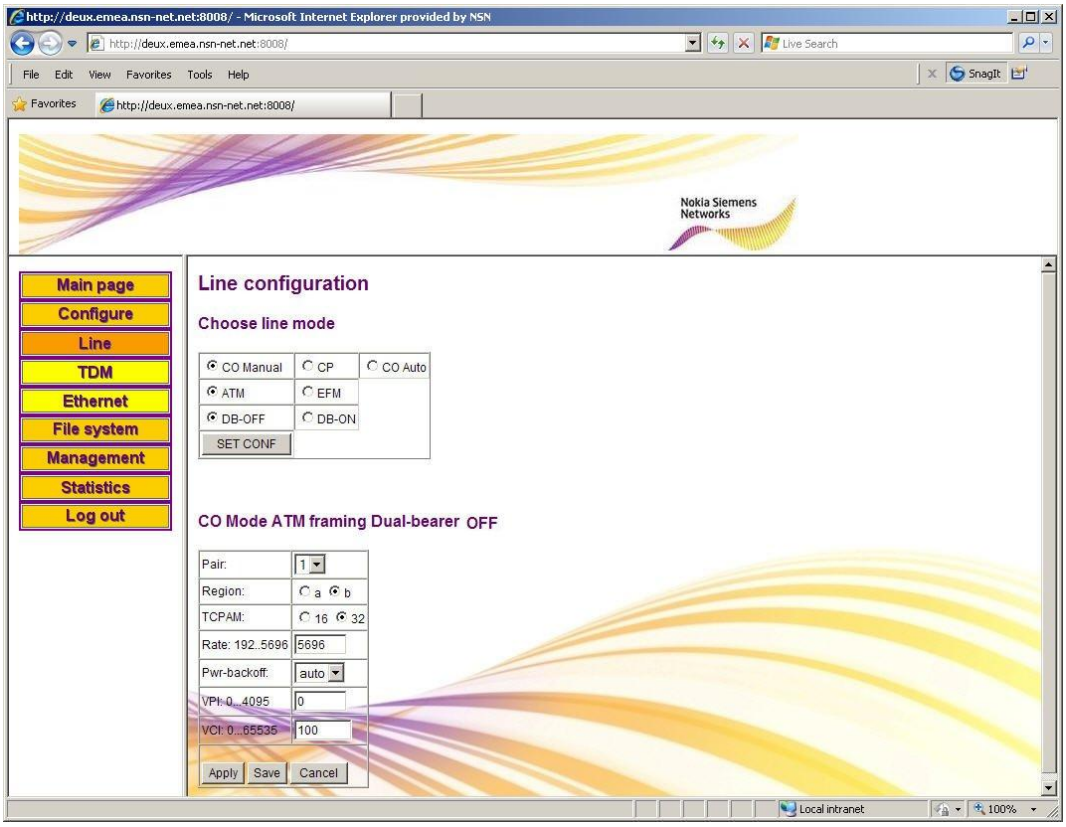

Figure 18. Line configuration page

7 CONCLUSION

The purpose of this thesis project was to design and implement a prototype of a web based GUI management for FlexiNT22 SHDSL.bis modem for NSN BBA NBMS division. Requirements from the company have been gathered and taken into account in making the web GUI design.

The main focus included in choosing the most suitable embedded web server, is to create a simple process and design user friendly web based GUI structure.

This thesis project has been for many parts very educational and rewarding. After a considerable amount of time and hard work a web based GUI management using KLone embedded web server has been successfully completed, and the objectives have been met. Nevertheless, this thesis writing process has been time consuming and very challenging. The main reason for this is because, during this journey lots of personal changes happened and unfortunately sometimes the motivation disappeared completely.

Despite the hardships encountered I greatly enjoyed all the challenges that this project brought up. The programming of web features, and rebuilding the server side software, is not something one often implements. All of these experiences made me a more capable c programmer and a better html designer.

For the future study other functions need to be developed; such as http authentication, file system, management and statistics.

8 REFERENCES

[1] NSN official website. [referred 8.1.2010] Available on the Internet: <URL: http://www.nokiasiemensnetworks.com>

[2] FlexiNT22, Rel 1.0 User Manual. Unpublished. [Accessed 1.11.2008] NSN intranet.

[3] Datasheet FlexiNT22 R1. Unpublished. [Accessed 1.11.2008] NSN intranet.

[4] Aalto, Matti 2007. *FlexiNT SW Architecture*. Unpublished. [Accessed 10.11.2008] NSN intranet.

[5] ITU-T Recommendation G.703 (11/2001), Physical/electrical characteristics of hierarchical digital interfaces

[6]ITU-T Recommendation G.704 (10/98), Synchronous frame structures used at 1544, 6312, 2048, 8448 and 44 736 kbit/s hierarchical levels

[7] IEEE 802.3u Standard Part3: Carrier Sense Multiple Access with

Collision Detection (CSMA/CD) access method and Physical Layer specifications Section Two

[8] ITU-T Recommendation G.991.2 (12/2003), Single-pair high-speed digital subscriber line (SHDSL) transceivers

[9] Mauro Douglas R. & Schmidt Kevin J. (2001). Essential SNMP. O'Reilly.

[10] Douglas E. Comer. (2006). Vol 1. 5th Ed. Internetworking with TCP/IP: Principles, protocols, and architecture. Chapter 24. Pearson Prentice Hall.

[11] IEEE Std 802.3ah -2004 Part3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and Physical Layer specifications

[12] Abhijit S. Pandya, Ercan Sen (1999). ATM Technology for Broadband Telecommunications Networks. CRC Press

[13] Nancy J. Yeager, Robert E. McGrath 1996, Web Server Technology: The Advanced Guide for World Wide Web Information Providers. Morgan Kaufmann.

[14] Sean K. Patterson, Senior Software Engineer, Spyglass Inc. Embedded Web server aids monitoring [referred 5.1.2010] Available on the Internet:. <URL: http://www.eetimes.com/electronics-news/4039713/Embedded-Web-server-aids-monitoring>

[15] Douglas E. Comer, (2006). Vol 1. 5th Ed. Internetworking with TCP/IP: Principles, protocols, and architecture, Chapter 27. Pearson Prentice Hall.

[16] Mani Subramanian. (2000). Networks Management: Principles and Practice. Addison-Wesley.

[17] GoAhead Software, Functionality Overview of an Open Source Embedded Web Server, (2000).

[18] KoanLogic embedded software engineering. [referred 15.10.2009] Available on the Internet: <URL: http://www.koanlogic.com/klone/>

[19] Interniche Technologies, inc. [referred 15.10.2009] Available on the Internet: <URL: http://www.iniche.com/webport.php>

[20] GoAhead Software. [referred 15.10.2009] Available on the Internet: <URL: http://www.goahead.com/products/webserver/default.aspx>

[Fig1] *Nokia Siemens Networks, Backhaul Technologies*. [referred 2.2.2009] Unpublished. NSN intranet

[Fig2] Wilton Paul, Frischholz Gerhard 2008, *FlexiNT22 – Network Terminal for SHDSL bis* (power point slide). [referred 2.2.2009] Unpublished. NSN intranet

[Fig3] Aalto, Matti 2007. *FlexiNT SW Architecture*. [Accessed 10.11.2008] Unpublished. NSN intranet.

[Fig4] http://www.ml-ip.com/assets/images/tcpip-layers.gif

[Fig6] http://lbdigest.com/wp-content/uploads/2008/05/httppersistence.png

[Fig7] http://lbdigest.com/wp-content/uploads/2008/05/httppersistence11.png