
Mi-Joung Choi

DP&NM Lab.
Dept. of Computer Science and Engineering
POSTECH, Pohang Korea
Tel: +82-562-279-5659
Email: mjchoi@postech.ac.kr
Contents

• Introduction
• Embedded Web Server
  – General Web Server & Embedded Web Server
  – Web-MUI & EWS Web-MUI
  – Advantages of EWS
• EWS Requirements
• Design
• Implementation
• POS-EWS Performance Evaluation
• POS-EWS Optimization
• Related Work
• Conclusions & Future work
Introduction

• Increase of the Network device
• Evolve World-Wide Web Web Technology
• Apply Web technology to network element management for monitoring and control
• The most direct way is to embed a Web server into network devices
• This can provide Web-based Management User Interface to manager
Introduction - cont’d

• Web-based management user interfaces through EWS have many advantages: Ubiquity, User-friendliness, Low-development cost, High maintenance

• Embedded Web Servers have consideration: Low resource usage (CPU usage, memory footprints)

? A lightweight and efficient EWS is essential for Web-based network element management.
Web Server & Embedded Web Server

• Web Server (World Wide Web Server)
  – Also known as HTTP Server
  – Designed to communicate with Web clients using HTTP
  – The repository of Web documents whose types are HTML and any application data with MIME type
  – Web documents are either static or generated dynamically
  – Typically runs on general purpose computers

• Embedded Web Server (EWS)
  – A Web server which runs on embedded systems with limited computing resources
  – Provides a Web browser interface between Web client and embedded system applications
WebMUI & EWS-WebMUI

• WebMUI (Web-based Management User Interface)
  – Provides a Web browser user interface for management
  – Provides static, dynamic and interactive content of management information of systems and networks
  – Can be used to configure, monitor and control managed systems via Web browser

• EWS-WebMUI (WebMUI through EWS)
  – EWS makes it possible for a Web browser to connect directly to the embedded system
  – Can be achieved by embedding three components into an embedded system: a Web server, Web documents and management applications
Advantages of EWS

• Provides enhanced user interface
  – Ubiquitous management
  – User-friendly interface via standard Web browsers

• Low development cost
  – No porting & distribution efforts for user applications
  – Platform independent graphical user interface
  – Short development time (short time-to-market)

• High maintenance
  – Web documents and associated programs can be easily modified
EWS Functional Requirements

- **HTTP Engine**
  - must deal with HTTP packets

- **File System**
  - Store Web documents and provide Web clients with them

- **Security**
  - limit access to sensitive information or configure & control

- **Powerful application interface**
  - mechanisms for the Web server to interact with embedded applications
EWS Non-Functional Requirements

- Low resource requirements
  - must use as little RAM, ROM and CPU as possible
- High reliability
  - highly reliable like one of the embedded system components
- High portability
  - portable on various RTOS and embedded systems
## EWS Architecture

![EWS Architecture Diagram]

1. **Web Documents (html)**
2. **RTOS**
3. **Management Application** (Configure, Monitor and Control)
4. **Embedded Application**
5. **Embedded System**
   - **VFS**
   - **Configuration**
   - **Security**
   - **Application Interface**
   - **HTTP Engine**

---

Design & Implementation of an EWS
Design Issues

- Protocol Consideration
  - Explicit cache control
    - For static Web page, caching is desirable, eliminating requests for redundant information
    - Dynamically-generated Web documents must not be cached in order to retrieve up-to-date information
    - HTTP/1.1 allows server to control the Web cache
  - Persistent TCP connection
    - TCP implementations maintain connection state information for two minutes after connection closed
    - HTTP/1.1 allows for a single persistent TCP connection between the browser and server

- Application Interface
  - CGI (Common Gateway Interface)
  - SSI (Server Side Include)
POS-EWS Features

- HTTP/1.1 compliant
  - Cache control
  - Persistent TCP connection
- Single thread based on extended architecture
  - Simple scheduler
  - Multiple finite state machines
- Virtual File System
  - Limited set of read-only file interface
  - Compression at compile time & Decompression at run time
- Web compiler
  - To build up virtual file system
  - Efficient server side include
- Cookie: state management
- OS: Xinu, pSOS, CPU: MPC 860
Design & Implementation of an EWS

POSTECH DP&NM Lab.

POS-EWS Finite State Machine

Set Up Initial State

Listen Connection

Parse Request Header

Map URL to Web Document

Close Connection

Wait new Request

Create Response

Application Interface

Send Response

Read Web Document

Check Authentication

OK

Accepted

HEAD Request

Non-Head Request

Time-out

HTTP/1.0

HTTP/1.1

Dynamic Info

Static Info

Fail

Success

Fail

Success

SSI or FORM.
POS-EWS Web compiler

Web Document \rightarrow Web Compiler \rightarrow ROM File

Executable Image \rightarrow C Compiler

Management Application Code

Web Server Code

```
struct vf {
    char * data = sysname_html;
    char date[]="1999:06:12";
    int size = 78;
    struct sc_list *head = &sysname_html_sysname;
} sysname_html;

struct sc_list {
    int start = 44;
    int end = 55;
    (char *) (*fptr)() = sysname;
    struct sc_list *next = (struct sc_list) NULL;
}sysname_html_sysname;

char sysname_html_html = "<HTML> ……"
```
Embedded Management Architecture

Embedded System

Web Browser

RTOS

RTOS

Management Application
(Configuration, Monitor and Control)

Embedded Application

VFS

Configuration

Security

Application Interface

EWS

HTTP Engine

SNMP Agent

Java SNMP Manager

Web Documents (html, Java applets)
POS-EWS Application Example

Web Browser

Java

SNMP Mngr

POS-EWS

Embedded Application

Web Document (html, Java)

SNMP Agent

(a) Design & Implementation of an EWS
POS-EWS Performance Evaluation

• Performance Metrics of General Web Server
  – Requests per second
  – Throughput in bytes per second
  – Round-trip time
  – Error rate

• Performance Metrics of EWS
  – Code size : 30 Kbytes
  – Run-time memory : 64Kbytes
  – CPU usage : Lowest priority
  – Maximum user connectivity (capacity)
POS-EWS Optimization

- Implement as a Finite State Machine (FSM) supporting single thread
- Keep TCP connection open and reuse by the Keep-Alive option
- Reduce the processing time using Web compiler through preprocessing
- Compress the documents for saving ROM resources
- Implement binary encoding of headers instead of ASCII
## Related Work

<table>
<thead>
<tr>
<th>Company &amp; Product</th>
<th>OS supported</th>
<th>CPU supported</th>
<th>HTTP code size (version)</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agranat Systems, EmWeb</td>
<td>No OS</td>
<td>Any CPU with a C compiler</td>
<td>25kbytes (1.1)</td>
<td>O</td>
</tr>
<tr>
<td>AllegroSoft, RomPager</td>
<td>Any RTOS, No OS</td>
<td>Any processor with an ANSI-C Compiler</td>
<td>10-40 kbytes (1.1)</td>
<td>O</td>
</tr>
<tr>
<td>Spyglass, MicroServer</td>
<td>LynxOS, QNX,OS-9, pSOS, VxWorks</td>
<td>Any CPU with a C compiler</td>
<td>35-110 kbytes (1.1)</td>
<td>O</td>
</tr>
<tr>
<td>Magma, Lava</td>
<td>Any RTOS</td>
<td>Any CPU with a C compiler</td>
<td>15-40 kbytes (1.1)</td>
<td>O</td>
</tr>
<tr>
<td>Quiotix, QEWS</td>
<td>pSOS, LynxOS, VxWorks</td>
<td>Any CPU with a C compiler</td>
<td>45-50 kbytes (1.0)</td>
<td>O</td>
</tr>
<tr>
<td>Web Device, Pico Server</td>
<td>LynxOS, Nucleus Plus, pSOS, VxWorks</td>
<td>Any CPU with a C compiler</td>
<td>15-30 kbytes (1.0)</td>
<td>O</td>
</tr>
<tr>
<td>POSTECH, POS-EWS</td>
<td>Real-time Xinu, pSOS</td>
<td>Any CPU with a C compiler</td>
<td>30kbytes (1.1)</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
<th>SSI</th>
<th>VFS</th>
<th>Compiler</th>
<th>Compression</th>
<th>Security (encode)</th>
<th>Cookie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Proprietary</td>
<td>Basic + Digest</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>None</td>
<td>Basic + Digest + SSL</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Proprietary</td>
<td>Basic + SSL</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GZIP</td>
<td>Basic</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ZIP-like</td>
<td>Basic + Digest + SSL</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GZIP/ CSS-Style</td>
<td>Basic + Digest (Base64)</td>
<td>O</td>
</tr>
</tbody>
</table>
Conclusions & Future work

- EWS-WebMUI provides ubiquitous, simple but powerful, user-friendly management user interface
- POS-EWS: HTTP/1.1 Compliant, Efficient embedded Web Server
- SNMP can be easily integrated with EWS
- Provide easy integration mechanisms to embedded management applications
- This “webification” of network devices has generated new philosophy for network element management

- Smaller and more powerful POS-EWS
- Port POS-EWS to other CPUs and embedded OSs
- POS-EWS on a chip
- Network Management of EWS-equipped Network Devices