내장형 시스템을 위한 웹서버 구현

*이현숙[0],*이수정,*정휘석,*최경희,**정기현
*아주대학교 정보통신전문대학원, ** 아주대학교 전자 전기 공학부
{sugy[0], berdl, kitty}@cesys.ajou.ac.kr, {khchoi, khchung}@madang.ajou.ac.kr

Implementation Tiny WebServer for Embedded System

*Hyun-Suk Lee[0] ,*Soo-Juong Lee,*Hui-Seok Jung,*Kyung-Hee Choi, **Gi-Hyun Jung
*Graduate School of Information and Communication Ajou University, ** Division of Electric & Electronics Engineering Ajou University

요 약

내장형 시스템은 특정한 기능만을 수행하는 작은 시스템을 목적으로 개발되기는 하였으나 본래의 기능 외에 시스템을 관리하거나 시스템에 의해 수집된 정보를 사용자가 쉽고 편리하기 이용할 수 있도록 하기 위해 다른 기능이 추가 되곤 한다. 우리가 개발한 시스템 역시 시스템의 configuration을 원격에서 설정/ 변경하고 수집된 data를 관리하기 위한 목적에서 웹서비스 기능이 필요하게 되었다. 웹서비스를 제공하기 위해서는 반드시 웹서버가 있어야 하는데, 기존의 웹서버는 크고, 실시간 운영체제에서는 맞지 않기 때문에 우리의 시스템만을 위한 서비스를 제공하는 작은 웹서버를 구현하게 되었다.

1. 서 론

내장형 시스템(Embedded System)이란, 특정 작업을 수행하기 위해 설계된, 이용할 수 있는 자원이 한정되어 있는 저비용, 소규모의 시스템을 말한다. 내장형 시스템의 가장 대표적인 예로는 네트워크 장비를 들 수 있으며, 그 외로는 셋톱박스, PDA, 가전제품 등에 이르기까지 이용 범위가 매우 다양하다.

이러한 시스템들 중에는 필수 기능으로 네트워크 기능을 지원하기도 하지만, 일부는 시스템을 관리하거나 사용자의 편의를 제공하기 위해 네트워크 기능을 지원하기도 한다.

우리가 관심을 갖는 바는 후자로서, 내장형 시스템의 configuration 값을 수정하거나 설정하기 위해, 또는 시스템에 의해 수집된 자료를 사용자가 쉽게 이용할 수 있도록 하기 위해 네트워크를 지원하는 것이다.

요즘 인터넷 사용이 보편화 되었기 때문에, 많은 사용자들이 쉽게 이용할 수 있도록 하기 위해서는 웹과 연계하는 방법이 가장 효과적이라고 할 수 있다. 웹을 사용할 수 있도록 하기 위해서는 내장형 시스템이 네트워크 기능을 지원해야 함은 물론이고, 웹 서비스를 제공할 수 있도록 웹서버를 포함하고 있어야 한다. 기존에 사용되고 있는 안정성이 입증된 여러 웹서버들이 있지만, 이들은 많은 기능을 포함하고 있기 때문에 내장형 시스템에서 사용하기엔 너무 크다는 단점이 있다.

또한 우리가 만든 내장형 시스템은 실시간 운영체제(Real-Time OS)를 사용하고 있기 때문에, 기존의 웹서버를 그대로 사용하는 데는 무리가 있다. 따라서 우리는 실시간 운영체제를 탑재하고 있는 내장형 시스템에서 사용되는 작은 웹서버를 구현하게 되었다.

2.관련 연구

2.1 HTTP [1,2]

HTTP(Hyper Text Transfer Protocol)는 월드와이드웹(World Wide Web)에서 웹을 통해 글, 그림 데이터 등을 교환할 수 있도록 하는 프로토콜로, TCP/IP와 관련된 응용 프로토콜 가운데 하나이다.

웹서버는 HTTP 데몬을 포함하고 있는데, 이 프로그램은 클라이언트로부터의 요청을 기다리고 있다가, 요청이 들어오면 그것을 분석하고 처리한 후 필요하다면 결과 파일을 보내준다.

HTTP 프로토콜에서 중요한 것은 http 헤더로써 이는, 클라이언트 프로그램(웹브라우저)과 웹서버 간에 의사소통이 이루어지도록 한다.

HTTP 헤더에는 요구 헤더(Request Header)와 응답 헤더(Response Header)가 있다. 이 중 웹서버를 구현할 때에 꼭 필요한 몇 가지만 살펴보도록 한다.

① Request_Method URI HTTP_Protocol_version
- 클라이언트가 가장 먼저 보내는 헤더
- 가장 흔한 메소드는 GET 또는 POST
- URI : 요구하는 파일의 경로

② Accept: media-type; quality
- 클라이언트가 받아들일 수 있는 데이터의 종류를 웹서버에게 알려줌.
- media-type: 브라우저가 처리할 수 있는 미디어 타입 (Application, Audio, Image, Text, Video등.)

③ Content-Type
- 헤더 뒤에 전송될 정보의 종류

④ Content-Length
- 헤더 뒤에 전송될 정보의 크기

⑤ 헤더의 마지막에는 반드시 공백 라인을 두어 헤더의 끝임을 알려 준다.

2.2 uC/OS [4]

uC/OS에 대한 자세한 설명은 생략하고, uC/OS가 가지는 내장형 시스템에 알맞은 특징을 중심으로 살펴보도록 한다.

uC/OS 코드의 대부분은 ANSI C로 되어 있고, 프로세서에 따라 달라져야 하는 최소한의 코드만이 어셈블리어로 되어 있기 때문에, 다양한 프로세서에 쉽게 포팅할 수 있다.

내장형 시스템을 위해 설계되었기 때문에 롬(ROM)에 내장할 수 있으며, 시스템의 목적에 맞는 서비스만을 포함하도록 함으로써 커널 사이즈를 줄일 수 있다. 또한 각 태스크는 자신만의 스택을 갖는데, 이 스택의 크기는 태스크마다 다르게 설정할 수 있으므로 이것을 잘 이용하여 응용 프로그램에 의해 사용되는 메모리의 크기를 줄일 수 있다.

최대 56(64-reserved 태스크 8)개의 태스크를 관리할 수 있으며, 항상 우선 순위가 가장 높은 태스크를 실행하는 선점형 멀티태스킹 커널이다.

또한 커널은 메시지 메일박스(Message MailBox), 메시지 큐(Message Queue), 세마포어(Semaphore), 고정 크기 메모리 블록 관리(fixed-sized memory partitions), 시간 관리 등의 시스템 서비스를 제공함으로써 응용 프로그램의 작성을 용이하게 한다.

3. 구 현

3. 1 내장형 시스템의 내부 구조

다음 그림1은 내장형 시스템에서 uC/OS 커널 내부에서 inet 데몬과 웹서버의 구조를 보여 주고 있다. 이 그림에는 표현되어 있지 않으나, 두 태스크 외에 시스템의 원래 기능을 수행하는 여러 개의 태스크들이 존재한다.

3.2 시스템의 동작 과정

uC/OS에서는 모든 프로그램이 하나의 태스크로 동작하기 때문에 아이넷 데몬(Inet Daemon)은 물론 웹서버도 하나의 태스크로 작성되어야 한다. 실시간 운영체제에서는 각 태스크에게 부여하는 우선 순위에 의해 성능이 크게 좌우되는데, 우리 시스템에서는 아이넷 데몬, 시스템의 주요 기능을 하는 태스크들, 그 다음 웹서버의 순서로 순위를 부여하였다.

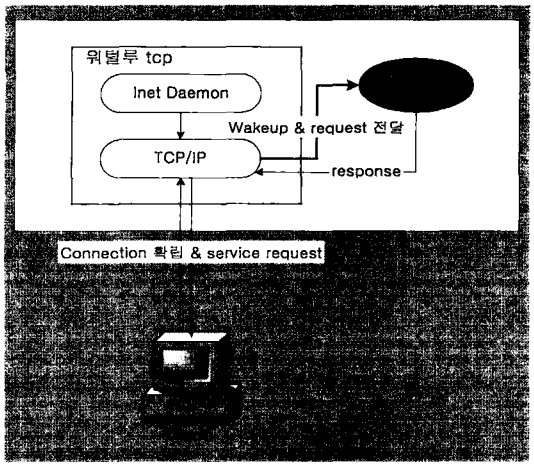

그림 1 내장형 시스템의 내부 구조

uC/OS 커널은 부팅하게 되면, 우선 순위가 높은 태스크들을 순서대로 생성하며 호출하게 된다. 이 과정을 통해 각 태스크들은 준비 작업을 하게 된다. 웹서버는 이 과정을 통해 자신의 리스닝 소켓을 초기화하는 등 필요한 준비 작업을 하고 커넥션이 확립되기를 기다리며 블록된다. 웹서버를 블록킹 시키는 것은, 웹서버가 주기적으로 돌면서 커넥션이 확립된 것이 있는 지를 확인하는 폴링 방식을 사용할 경우 자원이 낭비되며, 이로 인해 우선순위가 낮은 태스크의 수행이 방해될 수 있기 때문이다. uC/OS에서 세마포어를 제공하기 때문에 이것은 쉽게 구현할 수 있다.

부팅 후에는 태스크가 생성될 때 정해진 특성에 따라 커널에 의해 불려진다. inet 데몬은 주기적으로 자주 불려지며, 들어온 패킷이 있는 지를 확인하고 있다면

TCP/IP 프로토콜에 의해 패킷을 처리하게 된다. 이 과정에서 웹서버에 대한 커넥션이 확립되면 uC/OS에서 제공하는 함수를 통해 블록되어 있는 웹서버를 깨우게 된다.

깨어난 웹서버는 HTTP1.0 프로토콜에 의해 메시지를 주고 받으며, 클라이언트가 요청한 서비스를 제공한다.

3.3 웹서버의 동작

그림2는 웹서버의 main 함수의 flow를 보여주고 있다.

우리가 사용하는 시스템은 파일 시스템이 없기 때문에, HTML 파일은 물론 이미지 파일도 메모리에 저장하고 있어야 한다(프로그램의 컴파일 시에 함께 컴파일 되어야 한다.).

만약 클라이언트가 요청한 파일이 html이나 이미지 파일이라면, 요청한 파일의 내용이 저장되어 있는 메모리를 읽어서 그 내용을 http 프로토콜에 맞도록 구성하여 전송한다. html 파일이나 이미지 파일을 저장할 때에는 파일의 내용 뿐만 아니라 파일의 이름, 크기 등을 함께 저장 해야 하는 번거로움이 있다.

따라서 이러한 환경에서는 관리자의 편의를 위해서나 다양한 기능을 제공하기 위해서 CGI를 이용하는 편이 더 편리할 수 있다. CGI를 이용할 경우 관리자가 파일을 별도의 공간에 저장할 필요가 없기 때문이다. 일반적인 시스템에서는 사용자가 abc.cgi를 요청하게 되면 웹서버가 abc.cgi를 찾아서 실행하게 되지만, 우리의 시스템은 파일 시스템이 없기 때문에 웹서버가 처음 실행될 때 약간의 추가 작업이 필요하다.

웹서버는 자신이 제공할 수 있는 모든 cgi에 대해 파일 이름과 그것이 요청되었을 때 호출해야 하는 함수 이름의 짝으로 구성된 리스트를 생성하여 별도로 관리하여야 한다. 즉 사용자가 abc.cgi를 요청하면 abc() 함수가 호출될 수 있도록 (abc.cgi, abc)로 구성된 리스트를 가지고 있어야 한다. 이러한 방법은 파일 시스템이 없는 내장형 시스템에서는 반드시 필요한 과정이다. 이러한 점이 다소 불편해 보이긴 하지만, 내장형 시스템에서 제공하는 페이지의 양이 많지 않기 때문에 크게 문제 되지는 않는다.

4. 결론 및 향후 과제

본 논문에서는 파일 시스템이 구현되어 있지 않은 내장형 시스템에서 웹서버를 구현하는 것에 대해 기술하였다. 구현 후 실험을 통해 html, 이미지 등이 있는 파일과 cgi파일을 요청하였을 때, 웹서버가 제대로 동작하고 있음을 확인하였다.

파일 시스템의 부재라는 특성 때문에 일반 웹서버에는 없는 추가 작업이 필요하였으나, 이는 서비스를 제공하는 관점에서는 동일하다고 볼 수 있으며, 파일 시스템만 추가된다면 일반 웹서버와 동일하게 동작할 수 있을 것이다.

이 미니 웹서버는 다른 내장형 시스템에서도 그대로 사용할 수 있을 것이다.

현재 구현된 웹서버의 기능을 확장하기 위해서는 Get 메소드 뿐만 아니라 Post 메소드도 지원할 수 있도록 수정해야 할 필요가 있다.

그리고 현재 웹서버는 한번에 한명의 사용자만을 처리할 수 있는데, 성능 향상을 위해서는 여러 명의 사용자를 동시에 서비스 할 수 있는 방법을 추가해야 한다.

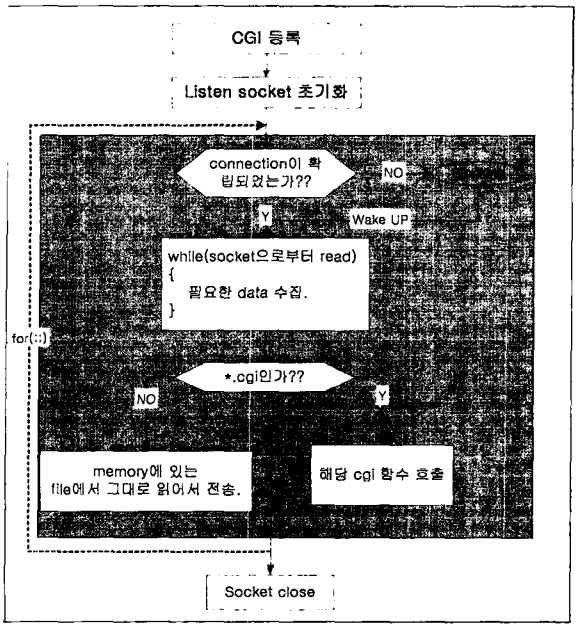

그림 2 웹서버의 main flow

5. 참고 문헌

[1] RFC-1945 " Hypertext Transfer Protocol -- HTTP/1.0"

[2] RFC-2616 " Hypertext Transfer Protocol -- HTTP/1.1"

[3] RFC-2169 " A Trivial Convention for using HTTP in URN Resolution"

[4] Jean J.Labrosse, MicroC/OS-II The Real-Time Kernel

[5] RFC-2854 " The 'text/html' Media Type"