임베디드 시스템을 위한 DHCP Client 구현 방안

김근경; 정기현; 최경희
아주대학교 컴퓨터시스템 연구실
amantecute@hanmail.net, khchung@madang.ajou.ac.kr, khchoi@madang.ajou.ac.kr

DHCP Client Implement of methods on Embedded System

Keun-Kyoung Kim[0] Ki-Hyun Jung; Kyung-Hee Choi,
Computer System Lab. Ajou University

요 약

본 논문에서는 ARM7TDMI CPU를 기반으로 한 네트워크 장비에 구현된 DHCP의 설계 및 구현에 대하여 논한다. 실시간 운영 체제인 uC/OS와 Watt TCP/IP 등의 소프트웨어가 탑재되어 있는 시스템에 DHCP를 구현함으로써, 네트워크 설정이 편리한 시스템이 되도록 하였다. 제안할 ' tiny-DHCP' 는 본래의 기능은 수행하면서, 그 크기를 축소한 것이다. 임베디드 시스템에서 더 경제적으로 구현할 수 있는 소형 DHCP인 ' tiny DHCP' 적용 방법에 관하여 논의하고자 한다.

1. 서 론

데이터 통신은 컴퓨팅의 기본 요소가 되었다. 인터넷망이 급속도로 발달한 지금, 원거리의 컴퓨터와의 정보 교환은 필수적이다. 사용자들은 그들의 통신 분제에 적합한 기술을 끊임없이 연구하고 있다. 그 결과 많은 프로토콜들의 표준화가 이루어졌다. 네트워크를 더욱 편리하게 사용하려는 요구가 증가하면서 DHCP의 사용이 급증하였다. DHCP(Dynamic Host Configuration Protocol)는 동적 호스트 설정 규약이다. DHCP는 호스트와 관련된 IP와 기타 정보를 설정할 수 있도록 지원하는 프로토콜이므로, DHCP는 사용자들이 ' 네트워크 설정' 을 몰라도 TCP/IP를 이용한 통신을 가능하게 해 준다. (참고문헌1) DHCP의 이러한 장점은 홈네트워크 시스템 환경에서도 중요한 역할을 수행한다.

최근 라우터를 비롯한 네트워크 장비들이 특정 작업을 수행하기 위해 설계되고, 저비용으로 구현하려는 임베디드 시스템이 대부분이다. 본 논문에서는 이러한 임베디드 시스템에 DHCP를 구현하고, 임베디드 시스템 목적에 부합하는 **tiny-DHCP** 구현 방안에 대해 논의하고자 한다.

2. 본 론

2.1 DHCP 동작원리

DHCP는 [그림1]과 같이 클라이언트와 서버로 구성된다.

클라이언트는 노트북이나 PC같이 IP를 요청하는 측이고, 서버는 IP를 적절하게 할당하는 측이다. (참고문헌1)

TCP/IP 인터넷에 접속된 컴퓨터는 데이터그램을 송수신하기 이전에 자신의 IP주소를 알고 있어야 한다. DHCP는 시스템 부팅시, 자신의 IP주소를 해결해 주는 프로토콜인 RARP를 사용하지 않고도 호스트가 자신의 IP주소를 결정할 수 있도록 한다.

일반 업무용 PC에서 DHCP를 사용하면 PC가 많은 네트워크에서는 관리자의 일이 많이 사라지게 된다

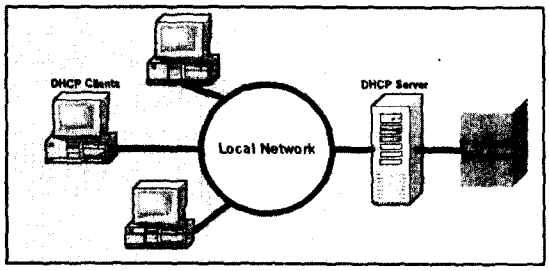

그림1. DHCP client/server 구성도

DHCP는 클라이언트와 server간에 lease(임대)를 확립하기 위하여 몇 개의 정의된 메시지를 주고 받는다. DHCP는 BOOTP 메시지 필드와 동일하다. 따라서 BOOTP와 DHCP 프로토콜은 호환이 가능하다.

다음은 양단에서 주고 받는 메시지의 흐름을 순차적으로 표현한 그림이다.

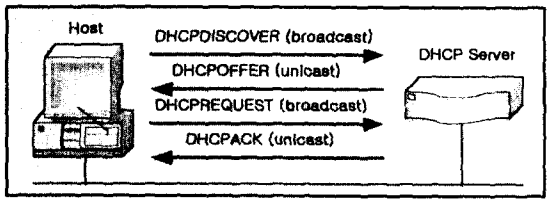

그림2. DHCP Messages

[그림2]는 DHCP 클라이언트가 DHCP server로부터 IP address를 요청할 때 일어나는 기본적인 단계를 보여준

다.

호스트인 클라이언트는 부팅시, DHCP 서버에게 DHCPDISCOVER 브로드캐스트 메세지를 보낸다. DHCP서버는 DHCPOFFER 단일전송(unicast)메세지로 클라이언트에게 IP 주소, MAC address, 도메인 네임, lease 등과 같은 정보를 제공한다.

DHCP 클라이언트는 여러 개의 DHCP 서버로부터 offers를 받을 수 있지만, 그 offers중 하나만 선택할 수 있다.(보통 클라이언트는 제일 먼저 도착한 offer를 받아들인다.) 클라이언트는 DHCPREQUEST 브로드캐스트 메시지로 제공된 IP 주소를 요청하는 메시지를 DHCP 서버에게 리턴한다. DHCP 서버는 DHCPACK 메세지를 클라이언트에게 보냄으로써, 클라이언트에게 할당된 IP 주소가 유효함을 입증한다.

만약에 설정값들이 유효하지 않은 DHCP 서버에 의해 클라이언트에게 보내졌다면,클라이언트는 DHCP서버에게 DHCPDECLINE 브로드캐스트 메시지를 리턴한다.

이런 과정을 거치고 나면 DHCP 클라이언트는 서버로부터 사용가능한 IP를 할당받게 된다. (참고문헌 2)

2.2 DHCP client 구현

DHCP 클라이언트가 정상적으로 IP address를 할당받기 위해서는, DHCP 서버에 dhcpd 데몬(daemon)을 띄워 놓아야 한다. (참고문헌 3) 그리고 할당받은 IP를 확인하기 위한 방법으로 (1)telned 응용프로그램 구현 (2)Sniffer 프로그램설치 (3)ping 프로그램 실행 등으로 DHCP의 성공적인 동작을 확인할 수 있도록 하였다.

DHCP를 동작시키기 위한 시스템 구조는 다음과 같다. 운영체제로는 uC/OS를 사용하고, 그 위에 TCP/IP및 UDP를 탑재하였다. 응용프로그램으로 DHCP를 구현하였다.

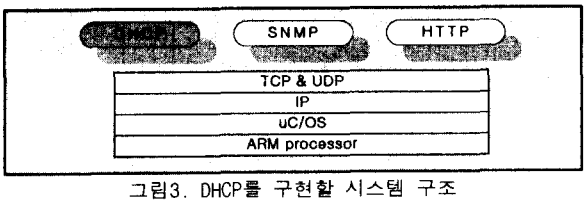

그림3. DHCP를 구현할 시스템 구조

ARM processor (참고문헌4)

ARM7은 RISC 칩 중에서 가장 널리 사용되고 있다. ARM의 가장 큰 장점은 저전력이라는 것이다. ARM은 비교적 저렴하면서도 가격에 비해 좋은 성능을 보이기 때문에 임베디드 시스템에 대단히 적합한 CPU라 하겠다. 또한, 국내 반도체 업계에서도 ARM코어를 사용한 칩을 생산하고 있으며, ARM사에서 개발환경과 도큐먼트를 제공한다는 것도 장점이다. 이런 이유로 DHCP를 구현하는 시스템에 ARM7TDMI chip을 선택하였다.

uC/OS-II

운영체제로 사용한 uC/OS는 영상, 의료, 음향기계를 비롯한 네트워크 장비, 고성능 전화 설비, ATM등에 적용되고 있다. 임베디드 시스템은 특정한 작업을 수행하기 위해서 설계되고, 특성상 실시간으로 동작하여야 하므로 실시간 운영 체제인 uC/OS를 채택하였다.

TCP/IP - Watt32s-2.1-dev4 (참고문헌5)

이 프로그램은 모든 메뉴얼을 제공하진 않지만, 소형 시스템에 구현이 적합한 프로그램이다. 짜여진 프로그램의 함수들이 비교적 잘 정리되어 있어 프로그램을 구현하기가 용이하다. 이 소스는 개발 중이기 때문에 추가, 변경될 수 있다.

[그림4]는 DHCP를 구현한 하드웨어 구조이다. CPU로는 ARMTDMI를, 메모리로는 EPROM과 SDRAM을 사용하였다. DHCP 클라이언트 프로그램은 EPROM에 탑재되어 있고, SDRAM은 system 메모리로 사용된다. 시리얼 디바이스로 MAX 232, LAN 디바이스로는 LXT 972가 설계되었다.

ARM에서 송수신되는 패킷들은 Ethernet Controller 블록에서 관장한다. Tx 버퍼와 Rx 버퍼는 각각 256 bytes이다. EPROM은 4Mbit, SDRAM은 각각 8Mbyte 크기를 가진다.

ARMTDMI는 33MHz 클럭을 사용한다. (참고문헌6)

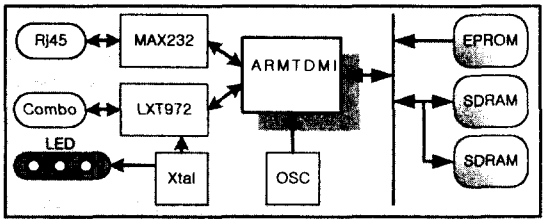

그림4. ARMTDMI CPU를 기반으로 설계한 하드웨어 구조

2.3 DHCP Client 소프트웨어 Porting

Porting이란 어떤 프로그램을 원하는 환경에 맞추어 코드를 수정, 변경, 추가하는 과정이다. 여기에서는 uC/OS와 탑재된 TCP/IP및 UDP에 대하여 응용프로그램인 DHCP 코드를 어떻게 적절히 구현했는지 살펴보겠다.

우선 DHCP에서 쓰는 data type을 uC/OS에서 사용하는 data type과 맞추어야 한다. 그리고 DHCP에서 주고받는 메시지들은 ' timer' 를 사용한다. 이 timer는 그것을 위한 함수를 생성하여 적용할 수도 있지만, 본 연구에서는 uC/OS에서 제공하는 OSTime을 적절하게 활용하였다.

여기서는 (1)DHCP가 동작하기 전에 선행되어야 하는 작업, 부팅시 제일 먼저 전송되는 (2)DHCPDISCOVER 메시지 생성, (3)각각의 메시지들을 수신, 전송하는 루틴, (4)정상적인 IP할당을 확인하기 위한 코드 순으로 기술하겠다.

DHCP가 동작하기 전에 선행되어야 하는 작업부터 살펴보자.

uC/OS는 OSInit()에서 uC/OS의 모든 변수들과 data structure를 초기화한다. 그러므로, 다른 어떤 service를 받기 전에 OSInit()를 호출해야 한다.

OSInit()에서는 OSTaskIdle()을 생성한다. OSTaskIdle의 우선 순위는 OS_LOWEST_PRIO이므로 어떤 다른 task가 생성된다면 새로 생성된 task를 언제든 수행할 수 있다. DHCP의 IP address 할당을 확인시켜줄 inetd와 telnetd을 위한 task를 생성하기위해 OSTaskCreate() 함수를 사용한다. (참고문헌7)

OSTaskCreate() 함수는 TASK를 만들어 등록시키는 함수로써, address와 몇가지 argument를 필요로 한다. Task가 생성될때는 stack도 함께 생성된다. OSInitTask()란 함수는 OSTaskCreat()와 OSTaskCreateExt()에 의해 불려지고 TASK의 스택 프레임을 초기화한다. TASK가 생성되면 TASK의 시작번지(task), 포인터(pdata), TASK의 우선순위(prio)를 OSTaskCreate() 또는 OSTaskCreateExt()로 돌려주는 함수이다. OSInitTask()에서는 DHCP handler인 _dodhcp()를 부른다. 핸들러가 불리고 나면 DHCP가 동작하기 시작한다. DHCP의 동작은 DISCOVER 메시지를 주변 서버에게 브로드캐스트함으로써 시작된다. Client는 부팅

시 67 port로 UDP를 사용하여, DHCPDISCOVER 메시지를 네트워크에 뿌린다.

DISCOVER의 패킷에는 DHCP_MAGIC_COOKIE 넘버를 채우는 필드가 있다. 이것을 잘 맞추어 정의해야 한다. 이 값이 잘못 정의되어 있으면 option 필드가 제대로 생성되지 않는 것을 확인하였다. (본 연구에서는 11개의 option이 사용되었다.)

Client에서 보내고 받는 메시지는 [표1]과 같이 나누어 볼 수 있다. (참고문헌8) [표1]에서 보면, client에서 보내는 packet 은 DISCOVER와 REQUEST 메세지이다.

DHCP는 메시지를 전송할 때, 공통적으로 sock_fastwrite()를 사용한다. sock_fastwrite()에서는 UDP type인지 check한 후, udp_write()를 통해 Ethernet address를 set, UDP header와 IP header를 생성, Check sum까지 수행한다. 실질적인 전송은 _eth_send()함수에서 수행한다.

Sending Packet from Client	Received Packet from Server
DISCOVER	OFFER
REQUEST	ACK / NACK

표1. Client에서 본 DHCP packet 분류

메시지를 전송할 때는 dhcp_out이란 버퍼를 사용한다. 여기에 패킷 필드의 각종 정보를 하나씩 채우게 된다. 전송하는 메시지의 타입은 클라이언트에서 결정하고, 받는 메시지의 타입은 서버에서 결정하여 전송한다. 서버로부터 packet들을 받으면 DHCP handler에서 sock_wait_input() 함수를 부른다. 이 함수는 packet을 받을 때 timer를 세팅한다. Timer는 OSTime과 OS_TICKS_PER_SEC으로 이루어진 set_timeout()을 활용하였다. OSTime은 OSTimeTick()에서도 사용되는데, 이것은 Power up된 이후의 시간을 나타내는 값이다. (참고문헌7) DHCP 메세지를 받은 경우, dhcp_timeout을 사용한다. dhcp_timeout 시간마다 들어온 packet이 있는지 sock_dataready()를 호출하여 확인한다. 그러다가 들어온 패킷이 발견되면, udp_read()에서 버퍼로부터 데이타를 읽어들인다. 이 버퍼를 검사한 결과, 그 패킷이 OFFER메시지라면 그에 해당하는 루틴으로 가고, DHCPACK라면 DHCPACK를 처리하는 과정을 거치게 된다.

각 메시지의 패킷 필드를 채우는 과정에서, 패킷 크기만큼 지정해주고 한꺼번에 데이터를 가지고 올 수 있는 ' 캐스팅' 이 지원되지 않는다. 그래서 각 필드를 하나씩 채우도록 수정하였다.

put_string()함수와 할당받은 IP가 담겨있는 dh_yiaddr 필드를 적절하게 연산하여, 어떤 IP address를 할당받았는지 바로 확인할 수 있도록 하였다.

2.4 실험 결과

모니터 프로그램과 앞서 언급한 telnet과 ping프로그램, sniffer로 동적 IP를 성공적으로 할당받았는지 확인하였다.

또한, dhcpd서버에 보면, /var/state/dhcp/dhcpd.leases라는 파일이 있다. 이곳에는 서버가 할당한 IP들이 남겨진다. IP를 할당받은 후 이 파일을 확인해보면, 그동안 서버가 할당해준 IP가 시간순서대로 쓰여있는 걸 확인할 수 있을 것이다.

DHCP 클라이언트가 적정한 IP를 할당받았다고 생각되면 ping프로그램으로 응답이 있는지 검사해본다.

이렇게 하여 구현된 DHCP Client가 dhcpd.conf파일에 있는 ' range' 값 중 하나를 할당받아서 정상적인 네트워크를 수행할 수 있다는 것을 입증하였다. 할당받은 IP로 어플리케이션인 telnet 접속을 시도한 결과, DHCP가 정상적으로 동작하고 있음을 다시 확인하였다.

Sniffer 프로그램은 패킷의 모든 필드를 표시한다. 이 프로그램을 사용하면 DHCP의 각 패킷의 데이터의 흐름을 한번에 알 수 있다. Sniffer를 사용하여 DHCP option 필드에 이상이 있다는 것을 발견할 수 있었고, range를 변화시켜 가면서 DHCP 클라이언트가 IP를 정상적으로 서버에게서 할당받고 있음을 확인할 수 있었다.

3. 결 론

DHCP의 IP address 할당과정을 보면 몇 개의 state를 거치게 된다. 이 중 RENEW state와 REBIND state에서는 DHCP *lease-time*의 50% 경과, 87%r 경과를 알려주는 두 개의 timer를 사용한다. 각각의 시간이 경과되면 DHCP 클라이언트는 server에게 REQUEST 메시지를 보내어 그 lease의 계속 사용 여부를 확인하고, 데이타를 갱신한다. (참고문헌9)

이러한 과정은 DHCP를 더 효율적이고 합리적이게 만들어 주긴 하지만, DHCP 본래 목적인 IP할당은 이미 이루어져있는 상태에서 발생하는 일들이다.

본 연구에서 제시하는 ' tiny-DHCP' 는 (1)DHCP 본래의 기능은 수행하면서도 (2)가격은 저렴하고, (3)크기는 축소한 동적 IP 할당 프로토콜이다. 특정 작업만을 수행하고, 저비용으로 구현되어야 하는 임베디드 시스템에서는 위에서 언급한 -RENEW와 REBIND state 과정을 생략한- ' tiny DHCP' 를 구현하면 시스템의 경제성을 보장할 수 있으리라 기대된다.

4. 참고문헌

[1] Neall Alcott, " DHCP for Windows 2000" , p.76, 2000
[2] http://www.cisco.com, " Cisco IOS DHCP Server" , p.1~p.5, 2001
[3] http://www.isc.org/products/DHCP/
[4] Steve Furder, " ARM System Architecture" , p.21, 1996
[5] ftp://dorm.rutgers.edu /pub/msdos/wattcp
[6] Samsung Electronics, " S3C4510B RISC Microcontrollers User' s Manual" , Revision1, p.1-1~p.1-5, 2000
[7] Jean J.Labrosse, " Micro C/OS-II The Real-Time Kernel" ,p.133, 1999
[8] R. Droms, " RFC 1541" , 1997
[9] R. Droms, " RFC 2131" , 1993

5. 부 록

[1] OSTaskIdle () : 이 함수는 항상 ready상태에 있다.
[2] OSTaskCreateExt() : OSTaskCreate()의 확장형.
[3] OSTime : Power up된 이후의 시간을 나타내는 값.