실시간 운영체제를 위한 IPv4 TCP/IP 구현*

양회권[0], 최인범, 김용희, 이철훈
충남대학교 컴퓨터공학과
{hkyang[0], ibchoi, yhkim, chlee}@ce.cnu.ac.kr

The Implementation of IPv4 TCP/IP for Real Time Operating System

Hui-Kwon Yang[0], In-Bum Choi, Yong-Hee Kim, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.

요 약

정보화가 가속되면서 시스템의 네트워크 기능은 필수적 요소가 되었다. 범용 운영체제와 달리 경량의 시스템에 탑재되던 실시간 운영체제들은 TCP/IP 프로토콜을 기반으로 한 네트워크 기능이 추가됨으로써 정보가전의 시대를 앞당기고 그 적용 범위 또한 확대하는 효과를 가져왔다. 본 논문에서는 본 연구팀이 개발한 실시간 운영체제인 *i*RTOS™에 IPv4 TCP/IP를 구현하였다[1].

1. 서 론

실시간 운영체제는 멀티태스킹을 지원하며 우선순위 기반의 선점 또는 비 선점형 스케줄러를 제공한다. 또한, Windows, Unix등 범용 운영체제와 달리 시스템의 한정된 자원(Resource)을 이용하여 논리적 정확성과 시간적 정확성(Determinism)을 보장해 주어야 한다. 이러한 실시간 운영체제의 중요성과 국산화의 필요성이 부각됨에 따라, 본 연구팀은 선점형 우선순위 기반 스케줄링(Preemptive Priority-based Scheduling)을 지원하는 *i*RTOS™라는 실시간 운영체제 커널을 개발하였다. 아울러 정보화 시대에 이르러 컴퓨팅 시스템에 대한 네트워크 관련 기능이 요구됨에 따라 본 연구팀은 *i*RTOS™를 위한 IPv4 TCP/IP 프로토콜 스택을 설계 및 구현하였다. 본 연구에서 TCP/IP 처리를 위한 태스크의 설계 및 제한된 자원의 효율적 활용을 위한 자원 관리 방법 그리고 소켓 응용프로그램의 호환성을 제공하기 위한 표준 소켓 API의 구현 방안에 대해 고려하였다[2].

본 논문의 구성은 2 장에서 기반연구로서 *i*RTOS™의 스케줄링, 메모리 관리, 태스크간 통신(Inter-Task Communication) 기법등에 대해 소개하고, IPv4 TCP/IP 프로토콜, 표준 소켓 API등에 대해 기술한다. 3 장에서는 TCP/IP 프로토콜 스택 구현 사항, 4 장에서는 테스트 환경 및 결과를 보이고, 5 장에서는 결론 및 향후 연구 과제를 기술한다.

2. 관련 연구

2.1 *i*RTOS™

2.1.1 스케줄링

*i*RTOS™는 선점형 우선순위 기반 멀티 태스킹을 지원하는 실시간 운영체제로서, 256 단계의 우선순위를 제공한다. 서로 다른 우선순위의 태스크들 사이에는 선점형을 동일한 우선순위의 태스크들에는 라운드-로빈(Round-Robbin)방식의 스케줄링을 적용한다. 선점형 스케줄링이란 프로세스의 우선순위에 기반하여 가장 높은 우선순위를 가지는 프로세스의 상태가 실행준비 상태(Ready State)로 되는 즉시 CPU 사용권을 부여하는 것으로서 응답성(Responsiveness)이 중요시되는 시스템에 적합하다[1][2][3].

2.1.2 동적 메모리 관리

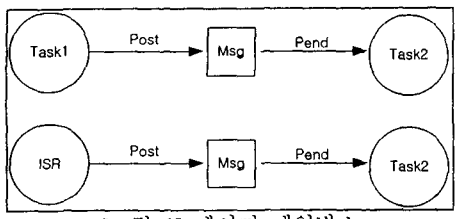
*i*RTOS™는 기본적으로 가변크기 메모리 할당을 위한 힙(Heap)을 제공하는데, 가변크기 메모리를 사용하면 외부 단편화(External Fragment)로 인한 문제와 적당한 메모리 블록을 찾기 위한 시간이 일정하지 않기 때문에 시간적 정확성을 보장하지 못할 수 있다. *i*RTOS™는 힙 기반의 고정크기 메모리 풀(Memory Pool)을 제공하여 외부단편화를 해결하고 시간결정성을 보장하도록 하였다[1][2]

2.1.3 태스크간 통신

태스크간 통신을 위해 메시지 메일박스와 메시지 큐가 제공된다. 아래 [그림 1]은 메시지 메일박스의 동작을 도식화한 것이다. 태스크 또는 ISR은 메시지 메일

* 본 연구는 산업자원부의 지역전략산업 석,박사 연구인력 양성사업의 지원으로 수행된 것임.

박스나 메시지 큐를 통해 다른 태스크로 메시지를 전달한다.



[그림 1] 메시지 메일박스

*i*RTOS™는 메시지 전달 방법으로 값 또는 메시지 스트림에 대한 포인터를 전달하는 방법과 메시지 스트림을 복사하여 전달하는 방법을 제공한다. 전자의 경우 데이터를 보호하기 어렵지만 전달이 빠르고, 후자의 경우 데이터 보호의 장점이 있다[1].

2.2 IPv4 TCP/IP 프로토콜

IPv4 TCP/IP 프로토콜은 32bit 주소체계를 가지는 인터넷 표준으로서 네트워킹의 주류를 이루고 있다. 크게 Ethernet, IP, TCP/UDP 계층으로 구분할 수 있으며 ARP, ICMP 및 기타 보조 프로토콜들이 정의 되어있다. 본 연구에서는 다음 6 가지 프로토콜만을 구현하였다.

- Ethernet
- IP
- TCP
- UDP
- ARP
- ICMP

2.3 표준 소켓 API

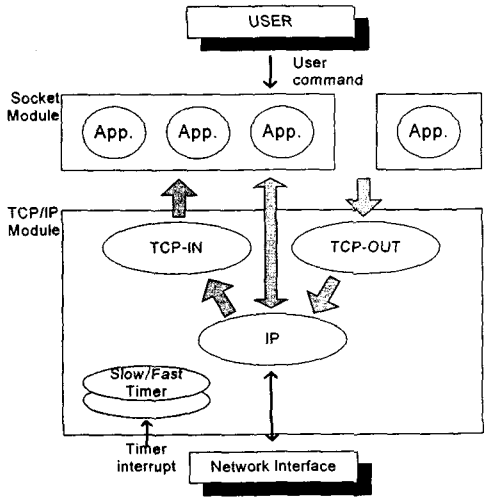
일반적으로 네트워크 응용프로그램은 TCP/IP프로토콜 스택의 내부 구조와 관계 없이 POSIX에서 제안한 표준 소켓 API에 맞추어 작성된다. 소켓 API에는 다음의 유형이 있다.

- 소켓관련 API
  socket(), bind(), connect(), listen(), accept(), send(), recv() ...
- 주소관련 API
  ntop(), pton(), ntoa() ...
- 기타 유틸
  htonl(), htons(), ntohl(), ntohs() ...

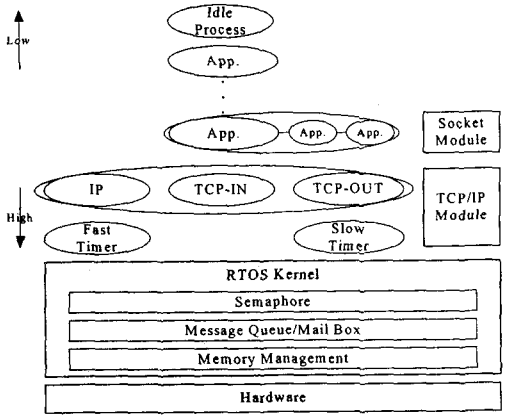
3. TCP/IP 프로토콜 스택 구현

3.1 TCP/IP 태스크의 설계

본 연구에서 효율적인 네트워킹을 위해 IP, TCP-IN, TCP-OUT, Fast-timer, Slow-timer 5 개의 TCP/IP 태스크를 설계하고, 중요도에 따라 크게 세 계층으로 우선순위를 나누었다. 특히, 패킷을 처리하는 태스크들 사이의 중요도는 패킷의 송수신에 의해 좌우되므로, 동일한 우선순위를 주어 라운드-로빈 방식으로 처리될 수 있도록 하였다. UDP패킷의 처리는 간단하기 때문에 태스크로 만들지 않고 모듈의 형태로 제공하여 IP 태스크 또는 응용프로그램에서 호출하여 사용할 수 있도록 하였다. 다음 [그림 2]는 TCP/IP 태스크들의 상호 작용을 나타낸 것이고, [그림 3]은 시스템 내의 우선순위 계층을 나타낸 것이다.



[그림 2] TCP/IP 태스크의 상호작용



[그림 3] TCP/IP 태스크의 우선순위 분포

3.2 자원사용의 최소화

실시간 운영체제가 탑재되는 시스템은 대부분 자원이 한정되어 있기 때문에 자원사용을 최소화 하기위한 고려가 필요하다. 본 연구에서는 [그림 4]와 같이 TCP/IP에서 사용하는 테이블과 메모리 풀의 크기를 정의할 수 있도록 함으로써 시스템의 자원에 따라 환경설정이 가능하도록 하였다.

3.3 연산의 간소화

패킷의 처리에 있어 잦은 에러 확인과 통계정보에

대한 처리는 연산의 효율을 떨어뜨릴 우려가 있다. 이에 본 연구에서는 에러 확인 과정을 간소화하거나 일부 생략하고 패킷 통계정보에 대한 처리를 생략하였다.

```
#define ARP_TSIZE       8
#define SOCKET_MAX      16
#define NETBUFS         16
#define MAXNETBUF       EP_MAXLEN
```

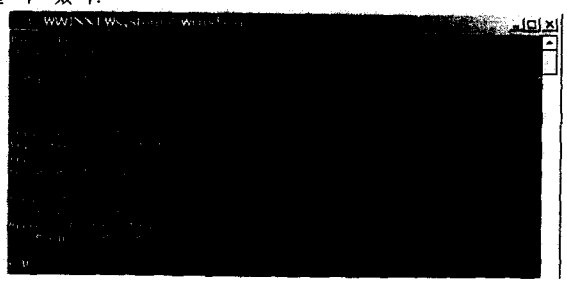
[그림 4] 자원 사용의 최소화를 위한 설정

3.4 표준 소켓 API

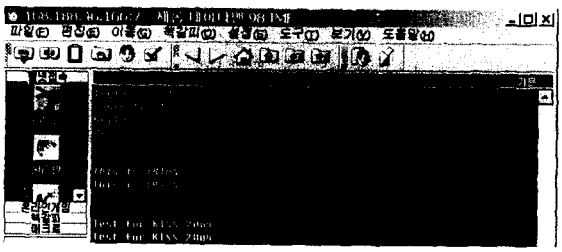
네트워크 응용프로그램의 호환성을 제공하기 위해 POSIX에 기술된 표준에 따라 소켓 API를 구현하였다. socket(), bind(), connect(), listen(), accept(), send(), recv() 등 기본 API와 htonl(), htons(), ntohl(), ntohs() 등 바이트 순서를 위한 API를 구현하였다. 현재 AF_INET프로토콜 패밀리와 STREAM방식과 DATAGRAM방식 소켓만을 지원한다[5][6][8].

4. 테스트 환경 및 결과

본 연구는 ARM-920T CPU가 탑재된 S3C2800 보드 기반으로 SDT2.51 ARM용 통합개발환경을 사용하여 진행하였다. 커널은 본 연구팀에서 자체 개발한 *i*RTOS ™를 이용하였으며 네트워크 어댑터는 Crystal社의 cs8900A 를 사용하였다. [그림 5]은 TCP/IP 스택을 로드한 상태에서 시스템에 Ping 요청(ICMP)을 수행한 결과이며, [그림 6]은 시스템의 에코 서버(Echo Server: TCP) 프로그램으로 에코 테스트를 수행한 결과 화면이다. [그림 7]은 시스템의 TFTP서버(UDP)를 테스트한 화면이다. 약 3Mbps의 속도를 나타낸다. 세가지 테스트를 통해 프로토콜 스택이 정상적으로 동작함을 확인할 수 있다.



[그림 5] Ping 테스트



[그림 6] Echo Test

[그림 7] TFTP 테스트

5. 결론 및 향후 과제

본 논문에서는 실시간 운영체제를 위한 IPv4 TCP/IP 스택의 구현에 대해 기술하였다. 실시간 운영체제 및 대상 시스템의 특성에 의해 시스템의 성능을 향상시키기 위한 여러 가지 사항들이 고려되었으며, 몇 가지 테스트를 통해 정상적으로 동작됨을 확인할 수 있었다.

향후, 다양한 네트워크 응용프로그램과 성능 분석 도구를 이용한 성능 테스트 및 개선이 진행되어야 하며, 나아가 차세대 네트워크 프로토콜인 IPv6 를 위한 TCP/IP 스택의 구현에 대한 연구가 진행되어야 한다.

6. 참고문헌

[1] http://www.inestech.com
[2] C.M.Krishna, Kang G.Shin, "Real-Time Systems", McGraw-Hill, 1997
[3] Jean, J. Labrosse, "μC/OS The Real-Time Kernel", R&D Publications, 1995.
[4] D. E. Comer "Operation System Design Vol 1 : THE XINU APPROACH", 1988.
[5] D. E. Comer"Internetworking With TCP/IP Vol 2 : Design, Implimentation, and Internals", 1998.
[6] W. Richard Stevens, "UNIX NETWORK PROGRAMMING", 1997.
[7] RFC 791, 792, 793, 826 외
[8] IEEE Std 1003.1-2001 "Open Group Technical Standard Base Specifications, Issue 6