제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

실시간 운영체제상에서의 TCP/IP 프로세스 스케줄링

양희권, 박희상, 이철훈
충남대학교 컴퓨터 공학과
e-mail : hkyang@ce.cnu.ac.kr

TCP/IP Process Scheduling based on Real-Time OS

Hui-Kwon Yang, Hee-Sang Park, Cheol-Hoon Lee
Dept. of Computer Engineering, Chung-nam University

요 약

실시간 운영체제는 시간 결정성이 보장되는 운영체제로서, 주로 적은 자원과 적은 전력을 사용하는 임베디드 시스템(Embedded System)에 사용된다. 인터넷 정보화 시대가 가속화되면서 일반 컴퓨터 뿐만 아니라 거의 모든 기기들에서 네트워크 기능은 필수적 요소가 되었고, 이들 시스템을 운용하는 운영체제에서의 네트워크 기능에 대한 요구는 계속 증가하고 있다. 본 논문은 네트워크 응용 프로그램을 처리하기 위해 TCP/IP 모듈(Module)을 기능에 따라 몇 개의 프로세스로 구분하고 각각의 프로세스가 효율적으로 동작하도록 스케줄링하는 내용에 대해 기술하고 있다.

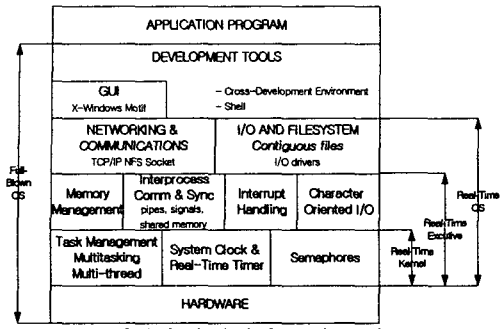
1. 서론

인터넷 정보화 시대에 들면서 인터넷의 적용 범위가 컴퓨팅 환경에만 국한되지 않고, 소형 단말기 등 각종 정보가전 제품으로 확대되고 있다. 정보가전에서의 네트워크 기능은 몇 개의 모듈만으로 구성할 수 있으나, 운영체제 없이 이들을 수행하기에는 작업분배 및 자원관리 등에 많은 문제가 발생하게 된다. 실시간 운영체제는 Unix, Linux, Windows등 데스크탑 PC 및 중·대형 컴퓨터용 운영체제와 비교하여 시간 결정성을 보장한다는 점과 상대적으로 실행 이미지가 작다는 점에서 큰 차이가 있으며, 주로 내장 제어 시스템과 같은 특수 목적으로 사용되어 왔다. 정보가전 기기는 대체적으로 제어 시스템의 크기가 작고 적은 전력을 소모하기 때문에 범용 운영체제보다는 실시간 운영체제가 적합하다. 따라서 이들 시스템을 운용하는데 사용될 실시간 운영체제에 있어 네트워크 기능은 필수 구현 요소이다. 이에 본 연구에서는 자체기술로 개발된 실시간 운영체제인 iRTOS™ 에 TCP/IP 표준에 의거하여 네트워크 프로세스를 설계, 구현하였다. 본 논문은 2 장에서 실시간 운영체제를, 3 장에서 TCP/IP 프로세스를, 4 장에서 프로세스 스케줄링을, 5 장에서 시스템 구현 및 실험 결과를, 6 장에서 결론 및 향후 연구과제에 대해 기술하고 있다.

2. 실시간 운영체제

실시간 운영체제는 시간 결정성이 보장된 실시간 시스템의 개발, 운영에 사용되는 운영체제를 말한다. [그림 1]은 각 유형의 실시간 운영체제의 규모 및 구성요소를 도식화 한 것이며, 이러한 실시간 운영체제가 가져야 할 특징은 다음과 같다[2].

첫째, 다중 프로세스를 지원하며, 선점 가능해야 한다.
둘째, 프로세스간의 우선 순위를 보장하여야 한다. 이는 프로세스의 데드라인을 만족시킬 수 있는 방법을 마련하기 위해 필요하다.
셋째, 프로세스간의 동기화를 지원해야 한다. 프로세스들 사이에 자원의 공유와 통신을 위해 동기화가 필요하다.
넷째, 운영체제의 행동이 명확해야 한다. 여기서 행동이라는 것은 시간적인 측면을 말하는 것으로, 인터럽트 지연시간, 시스템 콜의 처리시간 등이 포함된다.



[그림 1] 실시간 운영체제의 기능 및 구성요소

■ 스케줄링

실시간 운영체제의 스케줄링은 선점형으로 대표될 수 있다. 각 프로세스는 고유한 우선순위를 가지며 가장 높은 우선 순위를 가지는 프로세스가 보다 높은 우선순위를 가지는 프로세스가 생성되기 전까지 실행상태를 유지할 수 있게 된다. 여기에 더하여 iRTOS™ 에서는 특수한 경우를 위해 우선순위의 중복을 허용하여 정의된 타임 슬라이스(Time Slice) 동안 차례로 실행될 수 있도록, 라운드-로빈(Round-Robin) 방식을 병행할 수 있도록 하였다.

3. TCP/IP 프로세스

본 논문에서 TCP/IP 프로세스는 IP, TCP-IN, TCP-OUT, 응용, Slow/ Fast Timer 프로세스 등 모두 6 가지로 구분하였다. '응용 프로세스'는 TCP/IP 의 일부 모듈을 수행하는 측면에서 TCP/IP 프로세스로서 간주하였다. '응용 프로세스'를 제외한 나머지 5 가지 프로세스들은 시스템이 초기화 되면서 각각 1 개씩 생성되어 운영되며, 'Slow/Fast Timer 프로세스'를 제외한 나머지 프로세스들 사이에서는 메시지 전달에 의해 통신이 이루어지고 이로 인해 활성화 된다. 프로세스 간의 메시지 전달은 운영체제에서 제공되는 IPC(Inter Process Communication)에 의하는데, 여기에서는 메일박스(Mailbox) 및 메시지 큐(Message Queue)를 통한 메시지 전달 방법을 사용하였다. 이들 TCP/IP 프로세스들이 수행하는 개략적인 기능은 다음과 같다.

3.1 IP 프로세스

- 패킷 수신

네트워크 인터페이스로부터 데이터가 수신되었다는 인터럽트를 받아 패킷의 헤더정보를 분석하여 ARP, ICMP, TCP, UDP 등으로 패킷을 구분하여 ARP, ICMP 패킷은 해당 모듈을 통해 직접 처리하고, TCP 패킷의 경우 'TCP-IN 프로세스'로 메시지를 전달하여 작업을 넘겨준다. UDP 패킷의 경우, UDP관련 프로세스가 따로 존재하지 않기 때문에 해당되는 소켓을 찾아 '응용 프로세스'로 메시지를 전달한다.

- 패킷 송신

'TCP-OUT 프로세스'로부터 새로운 패킷이 생성되었다는 메시지가 도착하면 CPU를 넘겨받아 IP 헤더 등을 추가하고 패킷이 네트워크 인터페이스를 통해 전송될 수 있도록 하는 작업을 수행한다.

3.2 TCP-IN 프로세스

'IP 프로세스'로부터 TCP 패킷이 수신되었음을 알리는 메시지를 받게 되면, 오류 검사, 분해된 패킷의 조립 등을 거쳐 해당 패킷과 관련한 소켓을 찾아 '응용 프로세스'로 메시지를 전달하여 최종적으로 '응용 프로세스'로 하여금 데이터를 가공, 처리하도록 한다.

3.3 TCP-OUT 프로세스

'응용 프로세스'로부터 외부로 전송할 데이터가 있음을 알리는 메시지가 도착하면 TCP 윈도우 사이즈등을 고려하여 데이터를 분해하고 헤더를 추가하여 TCP 패킷을 생성한 후, 'IP 프로세스'로 메시지를 전송한다.

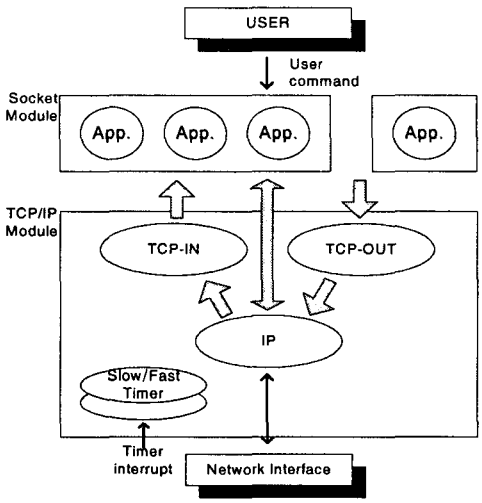
3.4 응용 프로세스

네트워크 응용프로그램 및 소켓 모듈을 수행하는 프로세스로서 하나의 네트워크 응용프로그램이 실행되면 단독으로 수행되거나(클라이언트 프로그램), 하나이상의 자식 프로세스들이 독립적으로 수행되는데(서버 프로그램), 사용자의 명령(Command) 입력 또는 데이터의 도착을 알리는 메시지가 'IP 프로세스'또는 'TCP-IN 프로세스'로부터 도착하는 경우에 CPU를 넘겨받아 작업을 수행할 수 있게 된다. 작업 수행 도중 전송해야 할 데이터가 발생하면 TCP '응용 프로세스'의 경우 'TCP-OUT 프로세스'로 메시지를 전달하여 데이터를 전송작업을 수행하도록 하고, UDP '응용 프로세스'는 'IP 프로세스'로 직접 메시지를 전달한다.

3.5 Slow / Fast Timer 프로세스

각각 1 sec, 10μ sec 의 지연시간(delay time)을 갖는 이들 Timer 프로세스는 ARP 테이블의 관리 및 TCP 흐름제어에 이용되는 것으로 정확도가 보장되어야 하기 때문에 TCP/IP 프로세스들 가운데 가장 우선하여야 한다. 이들 프로세스는 다른 TCP/IP 프로세스들과의 통신을 하지 않으며, Timer Interrupt에 의해서만 활성화 되어 동작한다.

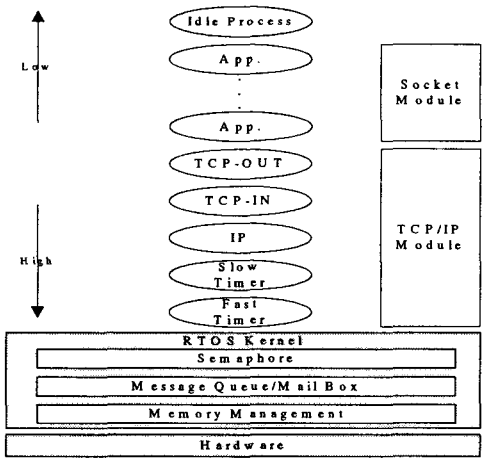
다음의 [그림 2]에 TCP/IP프로세스들 간의 실행의 흐름이 어떻게 진행되는지를 도식화 하였다. 그림에서 보듯이 각 프로세스들은 메시지 전달 또는 일부 인터럽트 등에 의해 프로세스를 실행할 수 있는 기회를 부여 받게 된다.



[그림 2] TCP/IP 프로세스 실행 흐름

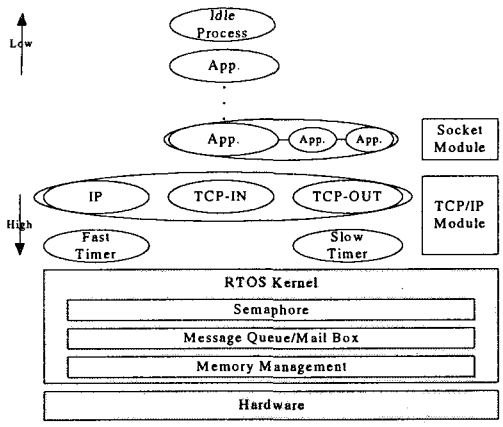
4. 프로세스 스케줄링

[그림 3]에서는 프로세스들의 중요도에 따른 우선순위를 개략적으로 나타내고 있다. 일반적으로 커널 수준의 프로세스들에 가장 상위 우선순위를 배정하고, TCP/IP 프로세스 및 동등 수준의 프로세스들에 그 다음 단계의 우선순위를 배정하며, 충분한 하위 수준의 우선순위를 응용 프로세스들이
할당받을 수 있도록 하고 휴지상태(Idle state)에 수행될 'Idle 프로세스'를 위해 최하위의 우선순위를



[그림 3] 프로세스의 우선순위

할당한다. 실시간 운영체제는 선점형 스케줄링을 수행하기 때문에 모든 프로세스들이 고유의 우선순위를 가질 수 있기 때문에 [그림 3]에서 볼 수 있듯이 모든 프로세스들이 수직적 계층 구조를 나타내는 것이 일반적이다. 여기에서 TCP/IP 부분만을 놓고 보면 실행상의 문제가 발생할 소지가 있다. 예를 들어, 연속적으로 패킷이 수신되는 상황을 가정해보자. 커널 수준의 프로세스와 일정시간마다 수행되는 'Slow/Fast Timer프로세스'가 수행된 다음엔 언제나 'IP 프로세스'가 CPU를 차지하게 되고, 보다 하위 수준의 우선순위를 갖는 나머지 프로세스들은 CPU를 차지 하지 못하게 되는 경우가 발생할 수 있다. 물론, 'IP 프로세스'로부터 전달될 메시지를 위한 버퍼가 부족하게 되면, 이후 수신되는 패킷에 대한 처리가 일단 유보되고 다른 프로세스에게 CPU를 할당받을 기회가 생길 수는 있다. 하지만, 'TCP-IN 프로세스' 또는 '응용 프로세스'가 메시지를 수신하여 메시지를 전달할 만한 버퍼가 확보되기 전까지 네트워크 인터페이스와 'IP 프로세스' 사이에서 발생하는 인터럽트/거부 동작이 불필요하게 반복되는 낭비를 초래하게 된다. 뿐만 아니라, 이후의 각 프로세스들에서도 같은 상황이 반복되기 때문에 시스템의 자원 낭비는 생각보다 커질 수가 있다. 때문에 TCP/IP 프로세스들에 대한 스케줄링 정책은 다소 달리 해야 할 필요가 있다.'Slow/Fast Timer 프로세스'는 다른 프로세스들과의 연관성이 떨어지고 정확한 시간에 바로 처리되어야 하기 때문에 TCP/IP 프로세스들 중 가장 높은 우선순위를 주되 'Fast Timer 프로세스'와 'Slow Timer 프로세스'간에도 한 수준 정도의 차이를 주어도 된다. 그러나, 각각의 주기(Delay)가 다르고 수행시간이 짧기 때문에 동등한 수준의 우선순위를 배정해도 된다. IP/ TCP-IN/ TCP-OUT 프로세스들은 동등한 우선순위를 부여하여 라운드-로빈(Round-Robbin)방식으로 처리되도록 하면 위와 같은 상황에서 발생하는 시스템의 낭비를 최소화 할 수 있게 된다. 실시간 운영체제에서 시스템내의 모든 프로세스는 실행시간에 대한 시간 결정성(Determinism)이 보장되어야 하는데, 이렇게 되면 TCP/IP 프로세스들의 시간 결정성이 깨지는 것처럼 보인다. 그러나, IP/ TCP-IN/ TCP-OUT 세 개의 프로세스는 각기 독립적인 프로세스가 아닌 상호 유기적인 프로세스들로서 네트워크를 수행하기 위한 하나의 프로세스로서 생각한다면, 네트워크에 대한 시간 결정성(Determinism)이 보장된다고 볼 수가 있다. 또한 '응용 프로세스'는 각기 생성시에 프로그래머에 의해 지정된 고유의 우선순위를 가지게 되며, 서버 프로그램과 같이 여러 개의 자식 프로세스를 생성하는 경우 부모 프로세스와 동일한 우선순위를 상속 받아 수행되도록 하였는데 이 역시도 부모-자식 프로세스를 동일한 목적으로 수행되는 하나의 프로세스로 간주할 수 있다. 결과적으로 다음의 [그림 4]에서와 같이 우선순위를 배정함으로써 시스템이 보다 안정되는 효과를 얻을 수 있다. [표 1]은 구현된 TCP/IP 프로세스들의 우선순위를 표로서 나타낸 것이다. '응용 프로세스'의 경우 응용 프로그램 프로그래머에 의해 31 부터 62 사이(응용프로그램의 우선순위 범위)에서 고유한 우선순위를 배정할 수 있도록 하였다.



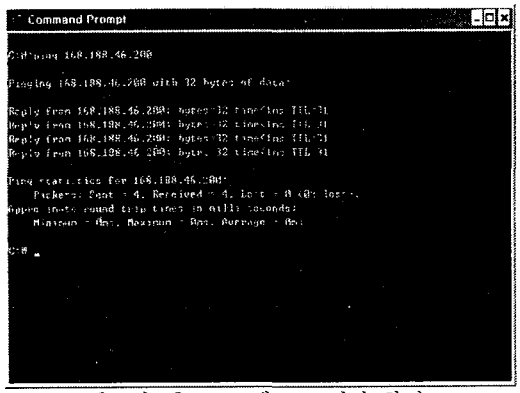
[그림 4] 개선된 프로세스의 우선순위

프로세스	우선순위	프로세스	우선순위
Fast Timer	11	IP	12
Slow Timer	11	TCP-IN	12
		TCP-OUT	12
		App.	31 - 62

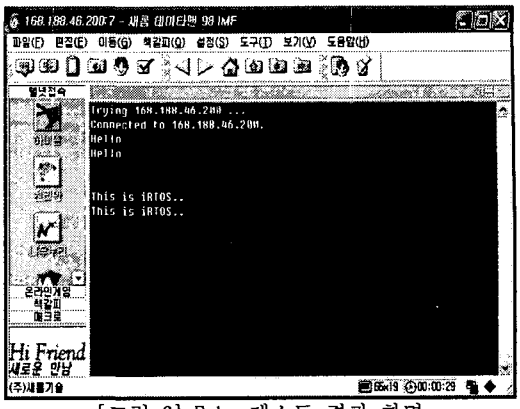
[표 1] TCP/IP 프로세스 우선순위 예

5. 시스템 구현 및 결과

실시간 운영체제인 iRTOS™ 을 기반으로 TCP/IP 스택을 구현한 실행이미지는 커널과 TCP/IP 스택을 포함하여 약 100Kbyte이다. 테스트 환경은 3COM™ 의 3C509b NIC를 탑재한 IBM 호환 PC를 타겟(Target) 시스템으로 설정 하고 네트워크로 연결된 또 다른 IBM 호환 PC에서 핑(Ping)테스트[그림 5]와 통신 프로그램을 이용한 에코(Echo) 테스트[그림 6]를 수행하였다. TCP/IP 프로세스 전반에 걸친 테스트라고 볼 수는 없지만 [그림 5]와 [그림 6]에서의 간단한 테스트 결과로 미루어 TCP/IP 프로세스에 대한 스케줄링이 제대로 동작되는 것을 알 수 있다.



[그림 5] Ping 테스트 결과 화면



[그림 6] Echo 테스트 결과 화면

6. 결론 및 향후 연구 과제

본 논문은, 실시간 운영체제 기반의 TCP/IP 스택을 처리하기 위한 프로세스의 기능별 분류 및 스케줄링 기법에 대해 기술하였다. 이를 통해 인터넷 정보 가전 등에 적용할 수 있게 되었다. 본 논문에서 기본적인 테스트 결과를 설명하였는데, 향후 Data Traffic 처리 등의 테스트를 통한 안정성에 관한 연구를 계속할 방침이다.

참고문헌

[1] http://www.inestech.com
[2] Jean, J. Labrosse, "μ C/OS The Real-Time Kernel", R&D Publications, 1992.
[3] D. E. Comer "Operation System Design Vol 1 : THE XINU APPROACH", 1988.
[4] D. E. Comer"Internetworking With TCP/IP Vol 2 : Design, Implimentation, and Internals", 1998.
[5] W. Richard Stevens, "UNIX NETWORK PROGRAMMING", 1997.
[6] RFC 791, 792, 793, 826 외