모바일 SoC 에서 실시간성을 요구하는 서비스를 위한 네트워크 프로토콜 스택의 설계 기법

김영만*, 김태훈*, 탁성우*
*부산대학교 컴퓨터공학과
e-mail : ssomai@gmail.com, ninkth@hotmail.com, swtak@pusan.ac.kr

A Design Mechanism of Network Protocol Stack for Supporting Real-time Service in Mobile SoC

Youngmann Kim*, Taehoon Kim*, Sungwoo Tak*
* Dept. of Computer Science & Engineering, Pusan University

요 약

최근 휴대폰, PMP 와 같은 모바일 장치를 개발하는 데에 그 성능과 저전력화 SoC 기술을 적용하고 있다. 또한 화상통화와 같은 영상 및 음성 멀티미디어 서비스가 확장되고 있다. 그러나 현재 모바일 SoC 기술에서 멀티미디어 서비스의 실시간 요구사항을 고려한 네트워크 프로토콜 설계에 대한 연구가 부족하다. 이에 본 논문에서는 실시간성 모바일 SoC 에서 실시간성을 제공하는 네트워크 프로토콜 스택을 설계하는 기법을 제안하고자 한다. 그리고 제안한 기법이 구현된 실시간 네트워킹 SoC 플랫폼의 성능을 실험하였으며, 그 결과 기존의 기법보다 더 좋음을 확인하였다.

1. 서론

SoC (System on Chip)는 CPU (Central Processing Unit)와 DSP (Digital Signal Processor), 그리고 FPGA (Field-Programmable Gate Array) 등 다양한 하드웨어 컴포넌트와 소프트웨어 태스크 등을 단일 칩에 통합하는 기술이다. 최근 휴대폰, PMP (Portable Multimedia Player)와 같은 모바일 장치를 개발하기 위해서 SoC 기술을 사용하는 경향이 두드러지고 있는데, 그 이유는 다음과 같다.

1. 고성능: FPGA 에 구현되는 하드웨어 태스크는 동일한 기능을 가진 소프트웨어 태스크보다 최대 100 배의 더 빠른 처리 성능을 보여줄 수 있을 뿐만 아니라, 소프트웨어 태스크를 처리하는 CPU 와 병렬 수행이 가능하다[1].
2. 저전력: 동일한 기능을 수행하는 데에 있어서 CPU 보다 더 빠른 시간 내에 처리할 뿐만 아니라, CPU 의 경우에는 캐쉬나 메모리로부터 명령어를 받아와서 수행하는 데에 있어서 더 많은 전력을 소모한다[2].
3. 소형화: 다양한 하드웨어 컴포넌트와 하드웨어 태스크, 그리고 서비스를 단일 칩에 통합할 수 있다.

이러한 SoC 기술을 기반으로 하여 네트워크 서비스의 질을 향상시키기 위해서 네트워크 프로토콜의 일부 코어 또는 전체 프로세스를 하드웨어 모듈로 구현하는 연구가 이루어지고 있다. [3] 그러나 기존 연구는 최근 휴대폰에서 제공하는 서비스인 화상통화와 같은 영상 및 음성 멀티미디어 서비스에서 요구하는 실시간 요구사항에 대해서 고려하지 않았다. 영상 및 음성 멀티미디어 서비스는 일정한 주기간격으로 패킷을 생성하기 때문에 반드시 주기 내에 패킷의 전송이 완료되어야 한다. 반면에 FTP (File Transport Protocol)과 같은 서비스에 의해서 비주기적으로 생성되는 패킷은 주기적으로 생성되는 패킷과는 달리 전송 마감시한이 없다. 그러나 QoS (Quality of Service)를 제공하기 위해 주기적으로 생성되는 패킷의 마감시한을 보장하는 한에서 가장 빠른 시간 내로 전송을 완료할 수 있어야 한다. 이에 본 논문에서는 모바일 SoC 에서 실시간성을 제공하는 네트워크 프로토콜 스택을 개발하고자 한다. 이를 위해서 다음과 같은 2 가지 기법을 제공하는 실시간 네트워킹 SoC 플랫폼을 제안한다.

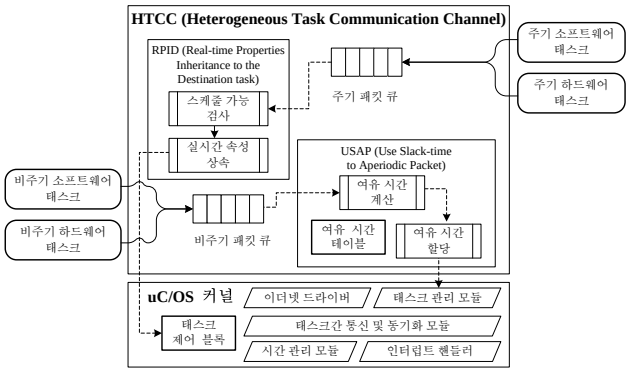
1. 하드웨어-소프트웨어 태스크간 통신 채널
2. 실시간 태스크 및 메시지 스케줄링

2. 실시간 네트워킹 SoC 플랫폼

하드웨어와 소프트웨어 간에 통신이 가능하게 하기 위해서는 동일한 성질을 가진 태스크로 보이도록 추상화할 필요가 있다. 이를 위해서 소프트웨어 태스크나 하드웨어 태스크가 동일한 통신 인터페이스를 사용할 수 있는 하드웨어-소프트웨어 태스크간 통신 채널인 HTCC (Heterogeneous Communication Channel)를 구현하였다. (그림 1)은 HTCC 의 구조를 보여준다. RTOS (Real-Time Operating System) 커널로는 uC/OS [4]

본 연구는 국토해양부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(07 국토정보 C04)에 의해 수행되었습니다.

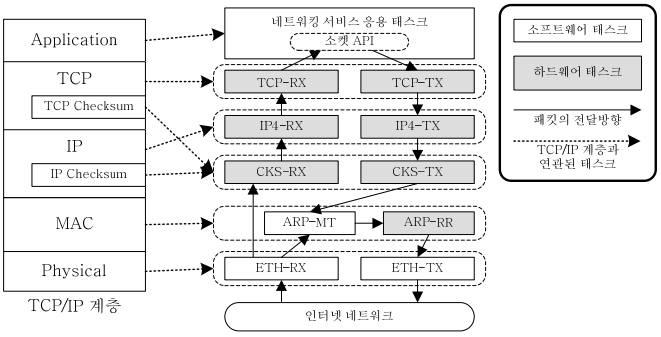
를 사용하였다. 그리고 주기적으로 생성되는 패킷의 마감시한을 보장하고 비주기적으로 생성되는 패킷의 전송처리시간을 최소화하기 위한 실시간 스케줄링 기법을 HTCC 에 구현하였다.



(그림 1) HTCC 와 스케줄링 기법

제안한 스케줄링 기법은 RPID 와 USAP 두 가지 모듈로 분할할 수 있다. 각 모듈은 서로 다른 대상을 스케줄링하지만, 서로의 단점을 상호 보완한다. RPID 는 주기적으로 생성되는 패킷의 실시간 속성 정보를 패킷을 처리하는 태스크의 실시간 속성 정보로 상속시키는 기법이다. 이를 통해서 주기적으로 생성되는 패킷은 태스크에 의해서 주기 내에 처리될 것을 보장받을 수 있다. USAP 는 주기 태스크와 주기 패킷의 처리 마감 시한을 보장하는 한도 내에서 비주기적으로 생성되는 패킷의 처리 시간을 최대한 단축 시키는 기법이다. 이를 위해서 주기 태스크와 주기 패킷의 처리 마감 시한까지의 여유시간에 대한 정보를 이용한다.

3. 실험



(그림 2) HTCC 및 스케줄링

본 논문에서 제안한 실시간 네트워킹 SoC 플랫폼은 ㈜휴인스에서 개발한 SoCMaster2 [5] 보드에서 구현하였다. FPGA 합성 툴은 Quartus 4.0 [6]을 이용하였으며, 주로 사용한 하드웨어 기술 언어는 Verilog HDL 이다. 그리고 소프트웨어 부분은 C 와 Assembly 를 사용했으며, C 와 Assembly 컴파일러로는 Arm Development Suite 1.2 [7]를 사용하였다. 그리고 네트워크 프로토콜로는 현재 가장 많이 사용되고 있는 TCP/IP 프르토콜을 사용하였으며, 이를 (그림 2)와 같이 태스크 단위로 분할하여, 실시간 네트워킹 SoC 플랫폼에 구현하였다. TCP/IP 프로토콜을 이용하는 주기 응용 태스크 집합의 실시간 속성은 <표 1>, 비주기 응용 태스크 집합의 실시간 속성은 <표 2>와 같다.

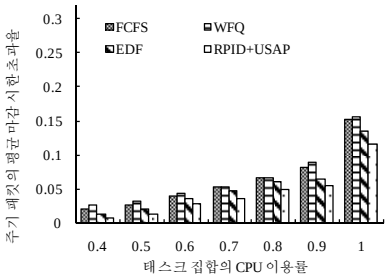
<표 1> TCP/IP 프로토콜을 이용하는 주기 태스크 집합

태스크	주기(ms)	수행시간(ms)
영상 패킷 전송 태스크	30	3.57
음성 패킷 전송 태스크	20	1.32
주기 패킷 전송 태스크	25	0.50

<표 2> TCP/IP 프로토콜을 이용하는 주기 태스크 집합

태스크	도착률(per second)	수행시간(ms)
비주기 패킷 전송 태스크	50	0.50
비주기 패킷 수신 태스크	50	0.50
영상 패킷 수신 태스크	33	2.31
음성 패킷 수신 태스크	50	0.78
주기 패킷 수신 태스크	40	0.50

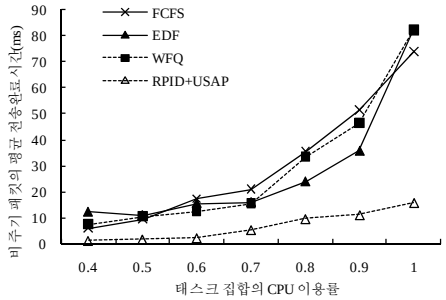
제안한 스케줄링 기법과의 성능을 비교하기 위해서 메시지 스케줄링 알고리즘으로써 잘 알려진 FCFS (First Come First Service)와 WFQ (Weighted Fair Queue), 그리고 실시간 스케줄링 알고리즘인 EDF (Earliest Deadline First)도 함께 구현하였다. 스케줄링 기법간 비교 성능으로써 태스크 집합의 CPU 이용률 별로 주기 패킷의 마감시한 초과율과 비주기 패킷의 응답시간을 실험하였다. 다양한 CPU 이용률에서 실험하기 위해서 더미 태스크를 추가하였다. 그리고 각 CPU 이용률 별로 총 10,000 초 동안 수행시켰다.



(그림 3) 주기 패킷의 평균 마감시한 초과율

(그림 3)에서 볼 수 있듯이 RPID+USAP 은 다른 스케줄러를 사용했을 때보다 최소 4.7%, 평균 43.7%만큼 주기 패킷의 마감시한 초과율이 낮아졌다. 패킷의 주기 속성을 고려하지 않는 FCFS 나 WFQ 는 모두 가장 높은 마감시한 초과율을 기록했는데, 그 중에서 영상/음성 패킷의 마감시한이 초과비율이 81.4%였다.

비주기 패킷의 평균 전송 완료시간에서도 RPID+USAP 은 다른 스케줄러보다 빠른 전송 완료시간을 보여준다. 이것은 주기 태스크와 주기 패킷의 마감시한을 보장하는 한에서 얻어지는 여유시간을 충분히 활용하였기 때문이다.



(그림 4) 비주기 패킷의 평균 전송완료시간

4. 결론

본 논문에서는 네트워크 및 멀티미디어, 그리고 응용 서비스로 구성된 모바일 SoC 에서 각 서비스가 요구하는 실시간성을 만족시키기 위한 HTCC 와 실시간 스케줄링 기법을 제안하고, 구현하였다. 제안한 실시간 스케줄링 기법은 주기적으로 생성되는 패킷의 처리 마감시한은 보장하고, 비주기적으로 생성되는 패킷은 최대한 빠르게 처리하도록 스케줄링한다. 그리고 실제로 네트워크 프로토콜 스택의 구현 기법의 성능을 분석하기 위하여 현재 가장 많이 사용되고 있는 TCP/IP 프로토콜을 태스크 단위로 분할하여 SoC 플랫폼에서 각각 하드웨어 및 소프트웨어 태스크로 구현한 후, 실험을 통해서 기존의 스케줄링 알고리즘보다 성능이 좋음을 확인하였다.

참고문헌

[1] J.P. Durbano, F.E. Ortiz, J.R. Humphrey, P.F. Curt, and D.W. Prather, "FPGA-Based Acceleration of the 3D Finite-Difference Time-Domain Method," IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 156-163, 2004.

[2] D. Desmet; P. Avasare; P. Coene; S. Decneut; F. Hendrickx; T. Marescaux; J.-Y. Mignolet; R. Pasko; P. Schaumont; D. Verkest: Design of Cam-E-leon: A Run-time Reconfigurable Web Camera, (accepted), Lecture Notes in Computer Sci-ence, Springer - Verlag..

[3] D.W. Kim, W.O. Kwon, K. Park, and S.W. Kim, "Internet Protocol Engine in TCP/IP Offloading Engine," International Conference on Advanced Communication Technology, Vol. 1, pp. 270-275, 2008.

[4] J.J. Labrosse, Micro C/OS-II. CMP Books, Kansas, 2002.

[5] Quartus development tool, Quartus development software handbook, Altera, available at http://www.altera.com/literature/lit-qts.jsp

[6] ADS C compiler 1.2, ADS (ARM Developer Suite) documentation, ARM, available at http://www.arm.com/documentation/Software_Development_Tools/index.html