논문 99-24-9A-6 한국통신학회논문지 '99-9 Vol.24 No.9A

IMT-2000 기지국 제어기를 위한 실시간 운영체제의 성능비교

정회원 이 경 오*, 이 숙 진**, 염 헌 영***

Performance Comparison Among Realtime Operating Systems for IMT-2000 Base Station Controller

Kyung-Oh Lee*, Sook-Jin Lee Kim**, Heon-Young Yoem*** *Regular Members*

요 약

본 논문에서는 현재 널리 사용되고 있는 상용 운영체제들 중에서는 빠른 응답 시간을 보장하고 있는 실시간 운영체제 3가지를 선정하여 이 실시간 운영체제들이 IMT-2000 기지국 제어기에 호 처리를 위해 사용될 경우 어느 정도의 성능을 보일 것인지를 실험하였다. 이를 위해서 무선 호 처리가 어떠한 방식으로 이루어지는지 분석하여 시스템 성능에 영향을 크게 미치는 기본기능요소들을 도출하였으며 분석된 내용을 중심으로 가상의 호 처리 실험 공간을 구축하고 일정 시간 운영해봄으로써 시스템의 성능에 병목현상이 되는 부분이 있는지 또 어느 운영체제의 성능이 다른 운영체제에 비해 우수한 성능을 보일 것인지를 비교 분석하였다.

ABSTRACT

This paper presents a performance comparison of commercial realtime operating systems which is widely used and guarantees short response time for call processing in IMT-2000 base station. For this, we analyzed wireless call process and extracted primitives which have strong influence on the performance of the system are derived. Constructing and running a test-bed for call processing on the basis of the analysis, we tested if there's a bottleneck in the performance of the system and compared the performances among the commercial realtime operating systems.

I. 서 론

IMT-2000(International Mobile Telecommunications-2000)은 차세대 이동 멀티미디어 통신의 고품질 서비스를 목표로 하고 있으며 전세계 어느 곳에서나 하나의 단말기로 통신이 가능하도록 하는데 목적이 있다. IMT-2000은 현재 서비스되고 있는 각종 다양한 통신 시스템(전화, ISDN, B-ISDN, CDMA, PCS, CT2 등)들을 단일

* 선문대학교 컴퓨터정보학부 데이터베이스 연구실(leeko@rainbow.sunmoon.ac.kr),
** 한국전자통신연구원 이동통신기술연구단 (sjlee@etri.re.kr),
*** 서울대학교 전산과학과 분산시스템 연구실(yeom@arirang.snu.ac.kr)
논문번호 : 98282-0706, 접수일자 : 1998년 7월 6일

한 라디오 기간망(Radio Infrastructure)으로 통합하여 기존의 고정된 무선 통신 네트워크로부터 제공받던 서비스의 품질은 보장받으면서 동시에 사용자가 요구하는 고품질의 다양한 서비스를 제공할 수 있도록 해주는 제 3 세대 글로벌 시스템으로서, 2000년대에 서비스 개시를 목표로 ITU-T/R을 중심으로 여기에 요구되는 각종 표준화를 진행하고 있으며 세계 여러 나라에서 경쟁적으로 시스템 개발에 박차를 가하고 있다[3,5].

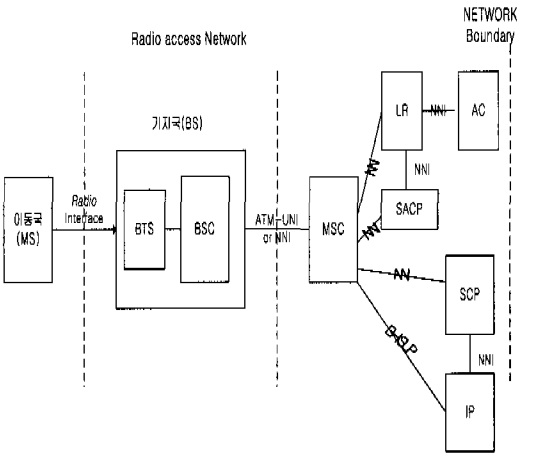
IMT-2000과 같은 무선 통신 환경에서는 사용자가 이동국(Mobile Station)을 이용하여 무선 호출을 발생시키게되면 이는 일단 기지국(Base Station)에 전달되게 되며 적절한 처리를 거쳐 ATM 등과 같은 고속의 유선 전송망을 통해 다른 지역의 MSC(Mobile Switching Center)로 전달되게 되어있다. 향후에는 현재보다 무선 통신 사용자의 수는 물론이요 각 사용자가 무선 통신을 하는 빈도가 더욱더 많아질 것으로 예상된다. 이렇게 많은 수의 통화가 폭주하게 되면 각 기지국에서 단위 시간당 처리할 수 있는 호의 수는 여러 가지 요인에 의해 제약을 받게된다. 이 중의 한가지는 기지국 제어기(BSC:Base Station Controller)의 처리 능력일 것이며 기지국 제어기의 처리 능력은 제어기 Hardware 성능과 운영체제(Operating System) 및 호 처리를 위한 여러 응용 프로그램에 의해 좌우될 것이다[4].

본 논문에서는 현재 널리 사용되고 있는 상용 운영체제들 중에서 빠른 응답 시간을 보장하고 있는 실시간 운영체제 3가지를 선정하여 실험을 실시하였다. 또 무선 호 처리가 어떠한 방식으로 이루어지는지 분석하여 많이 사용되거나 시스템 성능에 영향을 크게 미치는 기본 기능요소(Primitive)들을 도출하였으며 분석된 내용을 중심으로 가상의 호 처리 실험 공간(Test-Bed)을 구축하고 일정 시간 운영해봄으로써 시스템의 성능에 병목현상이 되는 부분이 있는지 또 어느 운영체제의 성능이 다른 운영체제에 비해 우수한 성능을 보일 것인지를 비교 분석하였다.

현재 IMT-2000 관련 기술들은 선진국뿐만이 아니라 국내에서도 많은 연구 개발이 이루어지고 있다. 시스템 개발 초기 단계에서 어떠한 운영체제를 사용하는 것이 시스템 전체의 성능을 초대로 발휘할 수 있게 할 것인지를 판단하여 적합한 운영체제를 선정하는 일은 매우 중요한 일이겠지만 또한 실험과정에서 얻어진 기술과 정보 및 시스템 성능 평가 기술은 향후 실제로 운영되는 시스템을 개발하고 구축하데 있어서 발생할 수 있는 시행착오를 최소화하고 보다 빠른 시간 안에 시스템을 개발할 수 있도록 하는 기술적 토대가 될 수 있을 것이다.

본 논문은 2장에서 실험을 위한 시스템을 모델링하고 3장에서는 실험의 방법론에 대해 언급하고 있으며 4장에서 각 시스템에 대한 실험 내용과 실험 결과를 종합적으로 분석한 내용이 기술되어 있고 마지막으로 5장에서 결론을 맺고 있다.

Ⅱ. 시스템 모델



MS : Mobile Station, BTS : Base Station Transceiver, Subsystem, BSC : Base Station Controller, MSC: Mobile Switching Center, UNI: User Network Interface, B-ISUP:B-ISDN User Park, LR : Location Registration, AC : Authentication Center, SACP : Service Access Control Point, SCP : Service Control Point, IP : Intelligent Peripheral

그림 1 IMT-2000의 표준 모델의 구조

IMT-2000의 표준 모델은 그림 1과 같이 크게 MS(Mobile Station), BTS(Base Transceiver Subsystem) 및 BSC(Base Station Controller) 그리고 교환기능을 하는 ATM-SW(스위치)와 호 처리 및 가입자 정보를 처리하는 망 장치로 구성되어 있다[4]. MS에서 전화를 걸게 되면(발신의 경우) 이 신호는 BTS에 전달되게 되고 BTS는

BSC와 필요한 메시지를 교환하게 되며 다시 BSC는 MSC와 정해진 메시지를 교환한 후 전화 통화를 할 수 있는 상태에 들어가게 된다. 전화를 받는 착신 호의 경우도 유사한 방법을 통해 이루어지나 MSC에서 BSC, 그리고 BTS로 발신의 역순서로 거치는 과정을 밟게 된다.

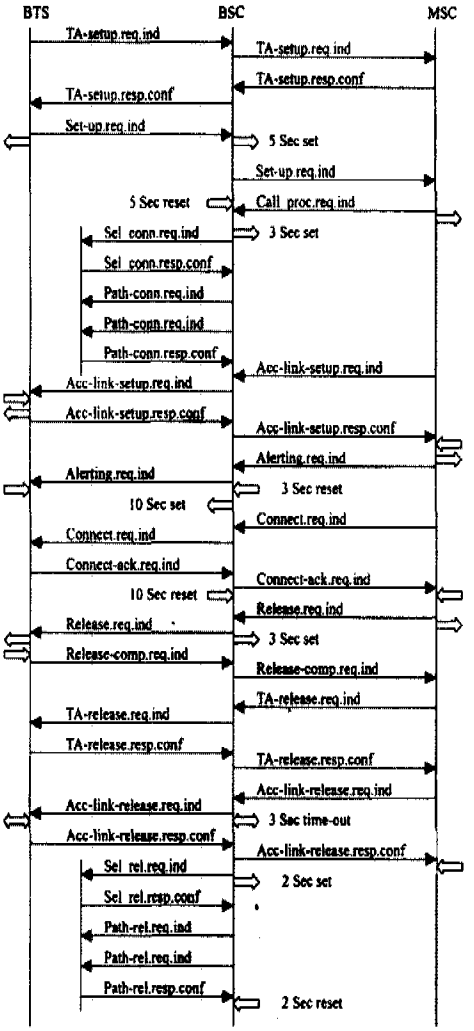


그림 2. 발신 호의 호 제어 절차

그림 2에는 발신 호의 경우 호 제어 절차가 묘사되어 있고 그림 3에는 착신 호의 경우 호 제어 절차가 묘사되어 있으며 한국전자통신 연구원의 CDMA 호 흐름(Call Flow)을 참조하여 작성된 것이다.

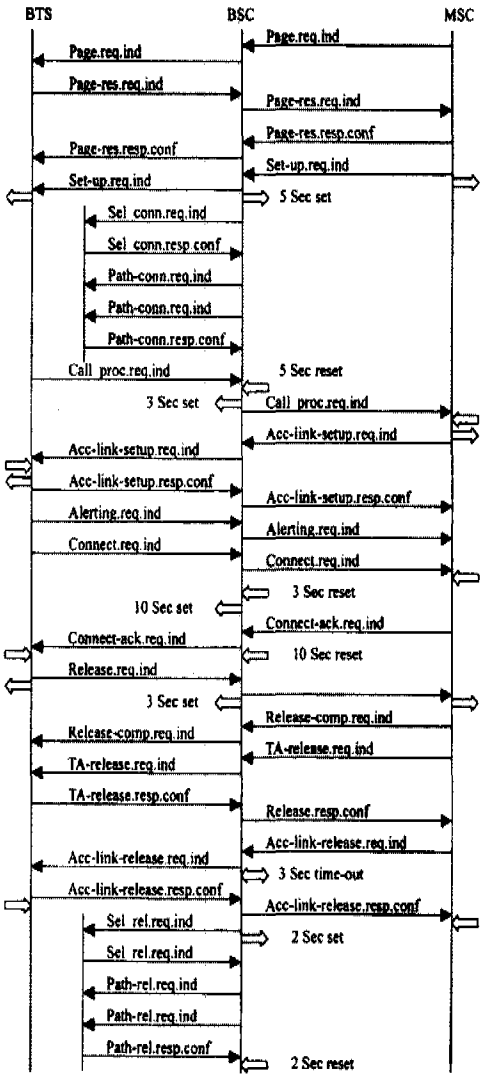


그림 3. 착신 호의 호 제어 절차

여기서 각 화살표에는 그 메시지의 내용이 무엇인지를 표시하는 이름표가 붙어 있는데 BTS, BSC 그리고 MSC에서는 받은 메시지의 종류에 따라 사전에 정해진 루틴을 수행하게 된다. 만약 중간에 네트워크의 상태가 좋지 않거나 시스템에 과부하가 걸리면 정해진 시간 안에 원하는 메시지가 도달하지 않을 수가 있다. 이 경우 Time-Out이 발생하고 해당 호는 비정상적인 상태라고 판단하여 즉시 처리를 종료하고 시스템을 빠져나가게 되며 이를 Dropping이라고 부른다. 예를 들어 BSC는 BTS로부터

Set-up.req.ind라는 메시지를 받으면 5초 Timer를 설정하고 MSC로 Set-up.req.ind라는 메시지를 전송하게 된다. 만약 5초 이내에 MSC로부터 Call-proc.req.ind라는 메시지가 BSC로 전달되게 되면 5초 Timer는 이제 필요 없어지게 되고 다음 수행을 계속하게 되지만 5초가 지나도 원하는 메시지가 도착하지 않으면 이 호는 비정상적인 종료를 하고 시스템을 빠져나가게 되는 것이다.

BTS, BSC 그리고 MSC 간에 메시지를 송수신 하면서 일어나는 일을 간단히 정리하면 그림 4와 같다. 일단 메시지가 도착하기를 기다리고 있다가 메시지가 도착하면 메시지의 종류에 따라 특정한 루틴을 실행하고 다른 편으로 메시지를 송신하고 다시 메시지가 도달하기를 기다리는 상태가 되는 것이다. 여기서 메시지의 크기는 ATM의 Cell 크기와 동일한 53 Byte로 크기가 고정되어 있다.

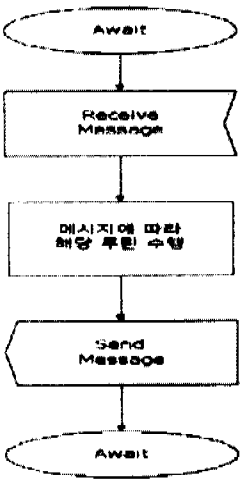


그림 4. BTS, BSC, MSC 의 메시지 이벤트 처리 과정

Ⅲ. 프로그램 모델

본 논문에서는 2장에서 언급한 시스템 모델을 상용 실시간 운영체제 상에서 구현하였을 경우 어느 정도의 호 처리 성능(단위 시간당 처리 가능한 호의 수)을 보일 것인지를 분석하였으며 이번 장에서는 IMT-2000 기지국 제어기 시스템의 성능 분석을 위해 어떠한 방법으로 프로그램을 구현할 것인지를 설명하고 있다.

1. 호 발생

단위 시간당 발생하는 호의 수는 포아송 분포를 따르며 가장 많은 호를 처리해야 하는 경우(최번시) 시간당 80,000호를 평균적으로 처리해야 한다고 가정한다. 즉 최번시 호 발생의 평균 간격은 3600/80000=0.045초가 된다. 일단 통화가 성립하게 되면 통화 시간은 평균 100초인 지수분포를 따른 다고 가정한다. 그때그때 호가 발생되어야 할 시점을 계산하는 것이 아니라 미리 적절한 수의 난수를 생성하여 Look-up Table에 입력해 두고 이를 참조하여 호를 발생하도록 하였다. 지수분포를 따르는 호 발생 간격의 난수는 다음 식으로부터 구해진다[2].

$$x=F^{-1}(y)=-0.045\ln(1-y),\ y\sim U(0,1) \quad (1)$$

식 (1)에서 0.045는 호 발생 평균 시간을 나타내며 y가 0부터 1사이의 Uniform한 분포를 따르게 되면 x는 지수분포를 따르게 된다. 이와 유사한 식으로 통화 시간을 나타내는 난수들을 구할 수 있으며 이 역시 Look-Up Table에 저장된다. 즉 0과 1사이에 난수를 발생시켜 식 1의 y에 대입하면 지수분포를 따르는 값들을 생성할 수 있다는 것을 의미한다.

2. 프로그램의 구조

2장에서 언급한 시스템의 구조를 살펴보면 한번에 동시에 두 프로세서가 통신을 하는 경우가 없으며 BSC를 중심으로 메시지가 BTS나 MSC 쪽으로 송수신 되는 것을 볼 수 있다. 따라서 BSC를 하나의 프로세서로 모델링하고 MSC와 BTS를 하나로 묶어 한 개의 프로세서로 모델링하여 이 두 개의 프로세서가 교신을 한다고 가정하더라도 무리가 없다.

BSC와 MSC는 ATM과 같은 유선 통신망을 통해 메시지를 주고받고 BSC와 BTS는 E1 트렁크 및 LAN 상에서 메시지를 주고받는다. 그러나 본 연구에서는 실험의 편의성을 위해 이들 세 프로세서가 한 프로세서 안에 존재하는 프로세스들로 가정하였으며 다시 이를 위에서 언급한 바와 같이 BSC를 위한 프로세스 하나와 MSC와 BTS 역할을 하는 하나의 프로세스가 있다고 가정하고 이 두 프로세스가 IPC(Inter Process Communication)를 통해 메시지를 주고받는다고 가정하였다. IPC는 실제 네트워크를 통

해 메시지가 전달되는 것보다 빠른 시간 안에 수행되므로 이를 보정하기 위하여 메시지를 받은 후 처리하는 루틴의 수행시간을 적정량만큼 증가시켰다.

일단 하나의 호가 발생하면 이 호는 수 십개의 메시지를 주고받은 후에 통화 개시가 가능하고 일단 통화를 개시하면 평균 100초 동안 통화를 하고 통화를 종료하는 절차에 들어가게 된다. 짧은 시간 내에 많은 호가 발생할 수도 있기 때문에 모든 호를 순차적으로 처리하는 것은 문제가 있다. 따라서 한 호가 발생하여 처리되고 있는 도중에 다른 호를 병렬 적으로 처리해야만 한다. 이를 위해서 본 연구에서는 한 호당 하나의 프로세스를 할당하여 이를 처리하게 하는 방식을 취하고 있다. 그런데 한 호가 처리되는 과정을 하나의 프로세서에 있는 두 개의 프로세스간의 IPC를 통해 구현하고 있으므로 한 호당 2개의 프로세스가 할당되어야 한다. 시간당 80000호를 평균적으로 처리하기 위해서 필요한 프로세스의 수는 약 2200개가된다. 호를 제어하기 위해 사용하는 시간을 무시하면 시스템은 M/M/m/m인 m-server loss system으로 볼 수 있다. 즉, 호는 0.045의 평균을 가지는 지수 분포로 시스템에 도착하고 시스템의 서버는 평균 100초의 지수분포로 호를 처리하며 서버가 다 차 있으면 호가 block되는 시스템으로 생각되어 진다. 이 경우에 호가 block될 확률 - m개의 서버가 다 차 있을 확률은 아래의 식에서 구할 수 있다[1].

$$p_m = \frac{(\lambda/\mu)^m m!}{\sum_{k=0}^{m}(\lambda/\mu)^k/k!} \qquad (2)$$

시스템이 호를 정상적으로 제어하기 위해서는 호가 block될 확률을 3% 이하로 유지하는 것이 필요하며 그 경우에 위 식에 의해서 약 2190개 이상의 서버가 필요하다. 그런데 이러한 프로세스들을 전체적으로 스케줄을 하기 위한 프로세스가 최소한 한 두개가 더 필요하고 하나의 서버에 대해 2개의 프로세스가 필요하므로 필요한 프로세스의 총수는 약 4500개로 잡을 수 있다. 이러한 프로세스들은 미리 실험을 개시하기 전에 모두 만들어 두었다가 호가 발생하면 한 쌍씩 할당하여 호를 처리하게 하는 Static 방법과 호가 발생하면 프로세스를 2개를 새롭게 생성하여 호 처리를 맡긴 후 호가 종료되면 삭제하는 Dynamic 방법이 있으며 그림 5에는 Static 방법에 대한 개념도가 예시되어 있다.

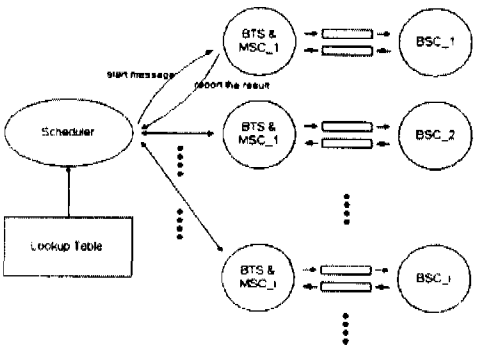


그림 5. Scheduler와 Process 간의 호 처리 개념도

Ⅳ. 실험 결과 및 분석

우리가 실험대상으로 선정한 RTOS A와 RTOS B는 host 장비에서 시스템을 개발하여 OS의 kernel과 응용 프로그램을 Target 장비로 이전하여 실행하는 Embedded OS 방식이며 RTOS C는 host 장비에서 개발과 운영을 동시에 할 수 있는 Stand Alone형 OS이다. 실험에는 Intel의 80486 CPU와 64MB의 주기억장치가 장착된 동일한 사양의 개인용 컴퓨터가 사용되었고 시스템 개발에는 C언어가 이용되었다. 3장에서 설명한 것처럼 모든 호는 사전에 발생시간(혹은 발생 간격)을 미리 Look-up Table에 기록해두고 있다가 Scheduler Task 혹은 Thread가 이를 처리하는 방식을 취하고 있으며 시스템이 안정상태(Steady State)에 들어간 후의 성능을 측정하기 위해 처음 2000개의 호가 발생하여 처리된 후부터 성능을 측정하였고 마지막 2000호의 처리 역시 성능 측정에서는 제외되었다.

RTOS A와 B에서 사용되는 Process는 Task라고 불리며 이는 일반 상용 OS에서 Process와 Thread의 중간 정도의 크기와 특성을 갖고 있으며 RTOS C에서는 Process와 Thread 모두를 지원하고 있다.

2장에서 호 처리의 절차를 분석한 결과 가장 중요한 Primitive는 메시지 송수신, Timer 관련,

그리고 Context Switching임을 알 수 있었다. Context Switching은 메시지의 교환이 일어날 때마다 최소한 한번(실제는 2~4번)이 일어나며 전체 성능에 가장 큰 영향을 준다. 표 1에는 각 Primitive를 측정한 결과를 보여주고 있다. 이 표를 살펴보면 Context Switching 시간은 RTOS C가 가장 짧고 메시지 처리 시간은 RTOS A가 그리고 Timer 관련 기능은 RTOS A가 가장 빠른 것을 알 수가 있다.

표 1 OS별 단위 성능 요소 측정 (단위 : μs)

	RTOS A	RTOS B	RTOS C
Context Switching	15.43	14.76	5.7
Message Passing (Send & Receive)	8.83	29.68	24.7
Timer 읽기	0.27	15.11	57

다음으로는 Static 방법론과 Dynamic 방법론의 실험을 실시하는데 정상적인 상황에서는 시스템의 부하가 적어 Time-out에 의한 Dropping이 거의 발생하지 않기 때문에 각 메시지를 받았을 때 처리하는 루틴을 수행횟수(시간)를 증가 시켜가면서 실험을 하였다. 그림 6에서 가로축은 메시지 처리 시간을 몇 배로 증가시켰는지를 보여주고 있으며 세로축은 Dropping 율과 Blocking 율을 합하여 서비스를 정상적으로 받지 못하는 하는 호의 비율을 나타낸 것이다. 여기서 Blocking이란 호가 발생하였는데 이미 정해진 숫자의 Task(혹은 Thread)가 모두 작업 중이기 때문에 Task를 할당받지 못하여 서비스를 받지 못하는 경우를 나타내며 이 경우 이 호는 시스템을 빠져나가게 된다. 그림 6에서 Dynamic 기법은 처리 시간을 약 5.4배 한 후부터 비정상 종료가 생기기 시작하고 Static은 약 5.5배한 경우 발생하는 것을 볼 수 있고 약 6배를 하는 경우 전체 호의 70%가 비정상 종료하는 것을 볼 수 있다. 전반적으로 Static 기법이 Dynamic 기법에 비해 우수한 성능을 보이지만 그 차이는 예상보다는 적었으며 이는 프로세스를 생성하고 삭제하는 Overhead가 전체에 비해서 크지 않기 때문으로 해석된다. 그림 5는 RTOS A에 대해 실험한 결과이며 3가지 OS가 비슷한 양상을 보였기 때문에 한 그림만을 예시하였다.

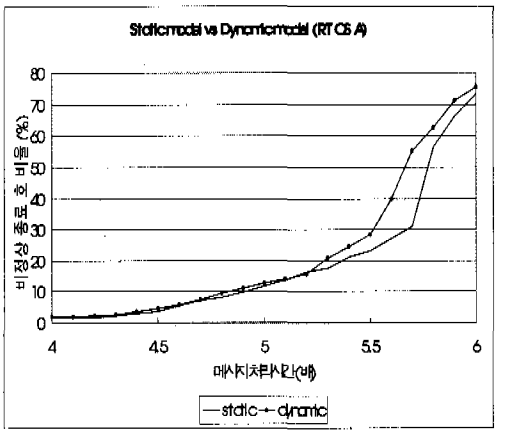


그림 6 Static 방법과 Dynamic 방법 비교

그림 7에는 3가지 OS에 대해 비정상적으로 종료한 호의 비율을 보여주고 있으며 위의 실험과 같은 방법으로 시스템의 부하를 증가시켜 가면서 실험하였다. 이 실험 결과에 의하면 RTOS C가 가장 우수한 성능을 보이고 있는데 이는 표 1에서 본 바와 같이 RTOS C의 Context Switching 시간이 다른 OS에 비해 매우 적은데 기인한다.

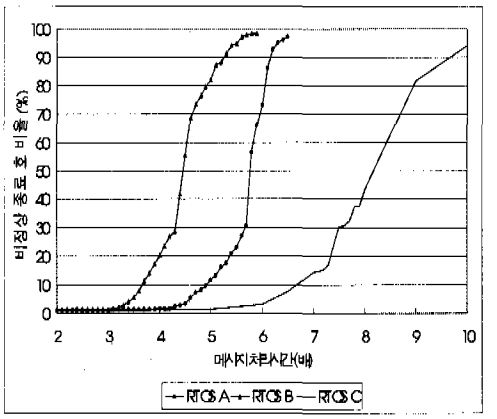


그림 7 OS별 비정상 종료 호 비율 - 메시지 처리시간 변화

일단 호가 발생하여 통화가 개시되면 이 통화 시간(Holding Time) 동안에는 음성 신호 이외의 제어 신호는 전혀 발생하지 않고 통화가 종료되는 시점부터 제어 신호가 발생하게 된다. 즉

이 통화시간 동안에 시스템은 아무런 일도 하지 않는 유휴 상태에 들어간다. 따라서 시스템 부하를 최대화한 상태에서의 성능비교를 위해 이 통화시간을 0으로 하여 실험을 실시하였으며 그림 7에 결과가 나타나 있다. 시스템 부하를 높이는 방법으로 호 발생간격을 줄여 즉 호의 Inter-arrival time을 짧게 하여 단위 시간당 발생하는 호의 수를 증가 시켜가면서 성능을 측정하였다. 일반적으로 무선 전화 시스템에서 비정상적으로 종료하는 호가 3%이내이면 큰 문제가 없는 것으로 판단하는데 RTOS A의 경우는 호 발생 간격 평균이 0.022초, RTOS B의 경우 0.028초, 그리고 RTOS C의 경우 0.015초보다 짧은 경우 3% 이상의 호가 비정상적으로 종료됨을 볼 수가 있다. 역시 RTOS C가 가장 우수한 성능을 보이는 것을 알 수 있다.

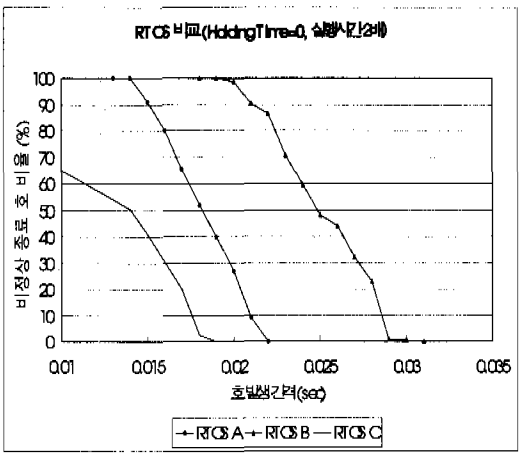


그림 8 OS별 비정상 종료 호 비율 - 호 발생 간격의 변화

그림 9에는 호가 발생하였는데 처리할 Task나 Thread가 없어서 처리를 아예 못하고 시스템을 빠져나가게 되는 경우(Blocking)와 호 처리 과정에서 시스템 과부하 등의 이유로 정해진 사건(Event)이 정해진 시간 내에 발생하지 않았기 때문에 비정상 종료하는 경우(Dropping)를 비교하여 설명하고 있다. 초기에는 Dropping이 거의 일어나지 않고 있다가 메시지 처리시간이 어느 값보다 크게 증가시키면 Dropping이 급속하게 일어나고 반대로 Blocking은 급속하게 줄어드는 것을 볼 수 있다. 호가 정상적으로 처리하게 되는 경우 평균 100초 동안 통화를 하기 때문에 최대의 호가 처리되고 있는 상황에서는 적어도 한 호가 종료될 때까지 새로운 호에게 서비스가 개시될 수 없다. 그러나 호가 비정상적으로 종료하게 되면 해당 Process 혹은 Thread는 이제 새로운 호에게 즉시 할당될 수가 있게 되므로 Blocking 비율과 Dropping 비율은 반비례한다고 해석할 수 있다.

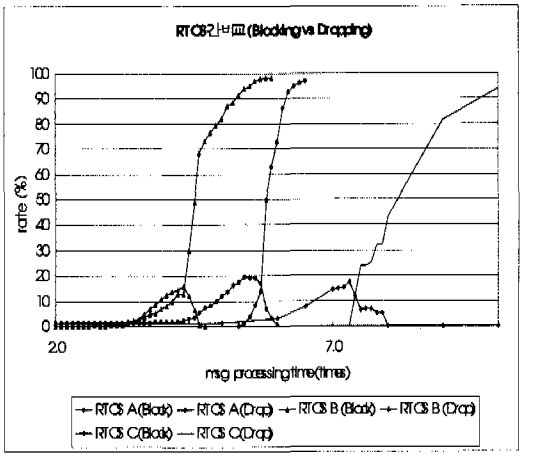


그림 9 OS별 Dropping/Blocking 호 비율 - 메시지 처리시간 변화

V. 결 론

실시간 운영체제에 관해서는 많은 연구[6,7]가 이루어졌고 상업용 제품도 많이 나와 있으나 이와 같은 상용 제품을 이용한 시스템 구현시의 기술적 난점이나 해결책 등에 대한 심도 있는 연구가 공개되어 있지 않은 관계로 필요한 기술 자체에 대한 접근이 어려운 상태이며 특히 IMT-2000과 관련해서는 현재 기술적 연구와 구현이 진행중인 상태이므로 이와 관련된 특화된 정보에 대한 접근이 쉽지 않다는 문제점이 있다. 본 논문에서는 IMT-2000 기지국의 기능 및 제공 서비스에 대한 요구사항의 분석, 기지국 제어에서 요구되는 실시간적 특성에 대한 연구, 현재 상업용으로 제공되고 있는 다양한 실시간 운영체제들의 특성을 비교/검토하여 기존의 실시간 운영체제를 적용하여 기지국 제어 S/W를 구현하는 것이 적합한지를 논의하였다. 이를 위해 본 논문에서 3개의 상용 실시간 운영체제를 대상으로 기지국의 Test-Bed가 될 수 있는 가상 기지국 제어 시스템을 개발하였으며

이 가상 시스템을 운영하여 성능이 어는 정도가 나오는 지를 측정하였다. 또한 이 과정에서 각 OS의 개발 환경과 개발 도구는 잘 구비되었는지를 검토하였다.

실험 결과 전체적인 성능에서는 RTOS C가 가장 우수한 성능을 보이고 있는 것을 알 수 있었으나 아직 시스템이 불안정하다는 단점이 지적되었다. 또 RTOS의 특성상 난수 발생 함수가 다른 OS에 비해 질이 떨어져 실험에 다소 어려움을 겪었다. 호 발생 간격이나 호 유지 시간 등을 생성시키기 위하여 난수를 발생시켰으나 Seed 값에 따라 성능에 2~4% 차이를 보이는 것을 확인할 수 있었고 오차를 최소화하기 위해 다양한 Seed 값을 이용하여 반복 실험을 실시하였다. RTOS C가 우수한 성능을 보인 것은 Context Switching 시간이 타 OS에 비해 짧았기 때문이다. 만약 다른 OS에서 Context Switching Time을 줄일 수 있다면 오히려 RTOS C가 가장 나쁜 성능을 보이게 될 수도 있을 것이다.

현재의 시스템 및 프로그램 모델에서는 한 호당 하나 혹은 한 쌍의 Task 혹은 Thread가 할당이 되어 일을 처리하는 방식을 취하고 있으나 이 구조는 필연적으로 많은 Context Switching이 요구되지만 이 방식은 프로그램을 작성하기 쉽다는 장점이 있으며 시간당 80000호를 처리하는데는 문제가 없는 것으로 판단된다. 그러나 호 처리 능력을 높이기 위해서는 몇 개의 Task가 많은 호를 Multiplexing하여 Context Switching 횟수를 줄이는 방법을 생각할 수 있을 것이다.

참 고 문 헌

[1] Leonard Kleinrock, *Queueing Systems, John Wiley & Sons, Inc.*, pp208-210, 1975

[2] Robert V. Hogg and Elliot A. Tanis, *Probability and Statistical Inference*, pp105-106, *Prentice Hall*, 1993

[3] 장석진, 김응배, 한기철, "ETRI의 IMT-2000 연구 개발 현황", *한국통신학회지* 14권 11호, pp42-51, 1997년 11월

[4] 위규진, "ITU-R의 IMT-2000/FPLMTS 표준화 동향", *한국통신학회지* 14권 11호, pp13-21, 1997년 11월

[5] M.H. Callendar, "Future Land Mobile Telecommunication Systems", *IEEE Personal Commun.*, vol.1, No.4, pp.18-22, 1995

[6] Darren Cathey, "RTOS Benchmarking, Real-Time Programming", Second Quarter 1993 Edition

[7] JohnBoyd and Roger Theyyunni, "Development of a Real-Time Simulation System", Embedded Systems Programming, pp48-56, 1996

이 경 오(Kyung-oh Lee) 정회원

1989년 2월 : 서울대학교 계산통계학과 졸업

1994년 2월 : 서울대학교 전산과학과 석사

1999년 2월 : 서울대학교 전산과학과 박사

1999년 3월~현재 : 선문대학교 컴퓨터정보학부 전임강사

<주관심 분야> 실시간 운영체제, VOD, 멀티미디어

이 숙 진(Sook-jin Lee) 정회원

1990년 경북대학교 전자공학과

1990년 ~ 현재 한국전자통신 연구소 선임연구원

현재 충남대 전자공학과 석사 과정

<주관심 분야> CDMA 이동통신 시스템/무선멀티미디어 시스템 트래픽 제어

염 헌 영(Heon-young Yeom) 정회원

1984년 2월 : 서울대학교 계산통계학과 졸업

1986년 2월 : Texas A & M 전산학과 석사

1992년 2월 : Texas A & M전산학과 박사

1992년 2월~1993.6 : 삼성데이타시스템 선임연구원

1993. 7 ~ 현재 서울대학교 전산과학과 부교수

<주관심 분야> 분산시스템, 멀티미디어, 데이터베이스