PCS 정보단말기용 소형 운영체제에 적합한 다중 메시지 처리 API의 설계 및 구현

김정수, 염근혁
부산대학교 컴퓨터공학과

Design and Implementation of Multiple Message Handling API for PCS Operating Systems

Jung Soo Kim, Keunhyuk Yeom
Dept. of Computer Engineering, Pusan National University

요 약

최근에는 Non-PC 디바이스들을 위한 Embedded OS들이 PCS, PDA, 핸드헬드(Handheld) 컴퓨터, 터미널, 산업용 제어기 및 다른 소형 컴퓨터에서 부터 인터넷 TV, 디지털 셋탑 박스, 웹폰, 인터넷 디바이스, 및 Mobile Computing 분야에 이르기까지 다양한 분야에서 그 요구가 급증하고 있다. 현재 이러한 응용들은 사용자의 응답시간이 빠른 표준 API(Application Programming Interface)의 사용을 필요로 한다. Window System의 Application을 개발하기 위해서는 보통 Win32 API를 사용한다. 그러나, Win32 API에서 제공하는 자료형과 관련 함수들이 크기가 크고 속도가 느려서 소형의 적은 메모리와 빠른 속도를 필요로 하는 PCS 단말기에는 적합하지가 않다. 따라서 기존의 API의 기능을 충실히 수행하면서 PCS에 최적화된 API에 대한 연구가 필요하다. 본 논문에서는 PCS 정보단말기용 소형 운영체제에 적합한 API를 GWES(Graphics, Windowing, and Events Subsystem) 모델을 기반으로 설계하였으며 사용자 응답시간 지연을 해결하기 위해서는 Event 처리를 최소화하는 다중 메시지 처리 방식을 개발하였다.

1. 서론

소프트웨어 시스템들은 상호 복잡하게 얽혀있어, 이해하기가 어렵고 비결정적인 요소(Nondeterministic factor)들이 많아 어려움이 많다. 이러한 시스템의 대표적인 것으로 Embedded System이 있다[1].

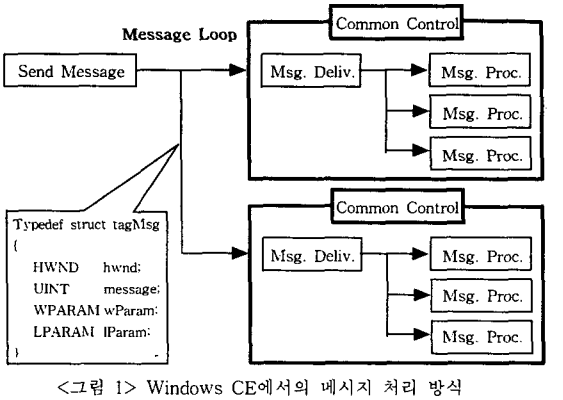
Mobile Computing 환경을 중심으로 한 다양한 분야에서 신뢰성을 가진 Embedded System의 개발에 대한 연구와 개발이 이루어지고 있으며, 그 개발에 있어서 Window System을 지원하기 위한 API들에 대한 연구는 중요한 역할을 담당하고 있다[5].

API란 윈도우하에서 수행되는 프로그램을 만들기 위해 사용될 수 있는 모든 것 즉 파일, 데이터 타입, 데이터 스트럭쳐, 메시지, 함수 등을 총괄하여 부르는 말이다. 따라서 API는 윈도우 응용 프로그램을 만들 수 있는 도구(Tool)들의 집합이다. API중에서도 가장 중요하고 분량이 방대한 부분은 라이브러리 함수이며 윈도우즈의 기능은 바로 이 라이브러리 함수를 통해서 이루어진다[4]. 본 논문에서는 PCS 정보단말기용 소형 운영체제에 적합한 API를 GWES(Graphics, Windowing, and Events Subsystem) 모델을 기반으로 설계하였으며 사용자 응답시간 지연을 해결하기 위해서는 Event 처리를 최소화하는 다중 메시지 처리 방식을 개발하였다.

2. 관련 연구

2.1 Windows CE

소형 윈도우 운영체제로는 마이크로소프트사의 Windows CE를 들 수 있다. Windows CE는 기존 PC의 Windows 98이나 Windows NT 수준의 윈도우 응용 프로그램들을 지원하며, 소형 컴퓨터 시스템(예, HPC)등에 주로 사용되고 있다. Windows CE는 가장 널리 사용되는 프로그래밍 모델인 Win32 API를 지원함으로, 각각의 그래픽 장치 인터페이스(Graphics device interface)에 대한 개별적인 컨트롤들과 이들을 지원해 주기 위한 GWES(Graphics, Window Manager, Event Manager)구조를 가지고 있다[3].



<그림 1> Windows CE에서의 메시지 처리 방식

Windows CE에서는 <그림 1>과 같이 메시지 스트럭쳐를 메시지 루프를 통해서 Common Control들에게 전달되며, Common Control들은 받은 메시지 스트럭쳐의 정보를 이용하여 필요한 동작을 수행하게 된다. 그러나, Windows CE에서 제공하는 Win32 API는 상대적으로 저속의 CPU를 사용하는 소형 PCS 운영체제에서 사용하기에는

크기가 크고 처리 속도가 느려서 최적의 환경을 필요로 하는 PCS 시스템에는 부적합하다. 이러한 Embedded System에서 API의 속도는 전체 시스템의 성능에 영향을 주므로 최적화된 API에 대한 연구가 필요하다.

2.2 OS9

실시간 마이크로 커널 운영체제로는 마이크로웨어사의 OS9을 대표적인 기술로 들 수 있는데, OS9은 크기가 작고 실시간 처리 기능을 제공하고 있어 현재 소형 가전 기기용으로 많이 채택되고 있다. OS9은 자체적인 MAUI(Multimedia application user interface) 지원하고 있으나, 윈도우 기능 등의 화려한 사용자 인터페이스와 메시지 처리 방식을 포함한 API Programming Style의 지원이 부족하다[6].

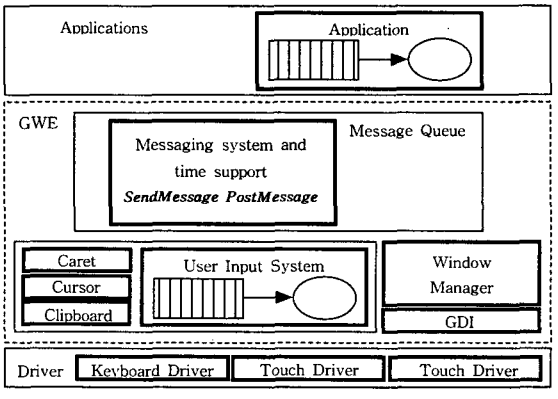
2.3 QNX

QNX의 경우는 POSIX기반의 X Library들을 지원한다. QNX는 Photon이라는 microGUI가 매우 작은 크기로 안정적인 GUI 환경을 제공하고 있다. 그러나, X Library를 기반으로 하는 QNX의 경우도 위젯이라는 데이터 스트럭처를 이용하여 Windows CE와 유사한 메시지 처리방식을 가지고 있다.[6] 따라서, 위젯의 데이터 스트럭처와 콜백 평션을 이용할 경우에도 저속의 CPU를 사용하는 소형 PCS 운영체제에서 사용하기에는 크기가 크고 처리 속도가 느려서 Embedded System에 최적화된 메시지 처리 방식이 필요하다.

3. PCS 소형 운영체제에 적합한 API에 대한 연구

3.1 GWES(Graphics, Windowsing, and Events Subsystem)

기존의 Window System에서는 Graphics, Windowing System과 Event System이 독립적으로 존재하였으나 PCS 소형 운영체제에서는 Event 처리의 효율성을 위해서 이를 통합해서 운영한다[3].



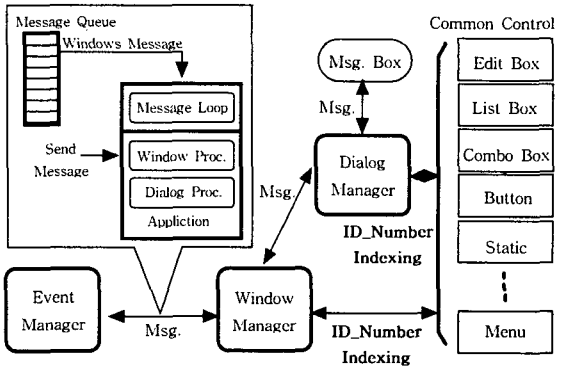
<그림 2> GWES의 내부 구조

GWES는 <그림 2>에서와 같이 사용자와 어플리케이션 그리고 오퍼레이팅 시스템간의 그래픽 사용자 인터페이스이다. GWES는 정보를 어플리케이션에서 오퍼레이팅 시스템으로 운송하는 메시지들로서 키보드 입력의 번역, 스타일러스의 이동 그리고 컨트롤 선택들에 의해서 사용자 입력을 처리하고, 디스플레이 장치나 프린터로 디스플레이 되어진 윈도우들, 그래픽 텍스트들을 생성하거나 관리하면서 사용자에 대한 출력을 처리한다[1].

GWES는 모든 윈도우들, 다이얼로그 박스들, 컨트롤들, 메뉴 그리고 Window 사용자 인터페이스에 의해 만들어진 자원들을 지원한다. 이러한 인터페이스들은 사용자가 메뉴의 선택 명령, 버튼을 누름, 박스의 체킹, 그리고 여러 가지 다른 컨트롤들의 조작에 대해서 응용프로그램을 제어하는 것을 허용한다. GWES는 사용자에게 커서, 텍스트 그리고 아이콘의 형태로 정보를 제공한다.

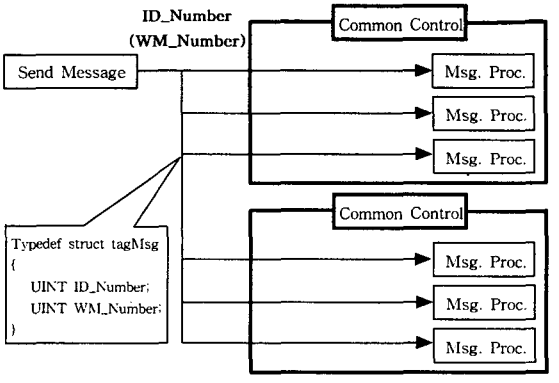
3.2 GWES의 메시지 다중 처리 방식

기존의 GWES는 메시지를 기반으로 하기 때문에 사용자가 입력하는 메시지와 운영체제로부터 들어오는 메시지 그리고 네트웍을 통해서 들어오는 메시지등 메시지들의 처리가 큐에 쌓이게 되면 느린 CPU와 적은 메모리상에서 속도가 느려지게 된다[2]. 메시지 처리 방식이 리소스들을 많이 차지하므로 본 논문에서는 입력된 Event들에 대한 메시지 처리에 대해서 다중 처리 방식을 사용하였다.



<그림 3> 다중 메시지 처리 방식을 사용하는 Window system

다중 메시지 처리 방식은 기존의 시스템들이 GWES 내에서 하나의 메시지 구조를 통해서 통신을 하던 것을 <그림 3>과 같이 윈도우 메니저, 이벤트 메니저 그리고 그 외의 서브시스템들 사이에서는 기존의 메시지 처리 방식을 사용하고, Common Control들과 윈도우 메니저, 다이얼로그 메니저 사이에서는 <그림 4>에서와 같이 각각의 Common Control들에 대한 ID_Number(Identification Number)와 그 내부의 동작들을 지시하는 WM_Number(Window Message Number)들을 이용해서 메시지를 전달하는 방식을 사용한다.



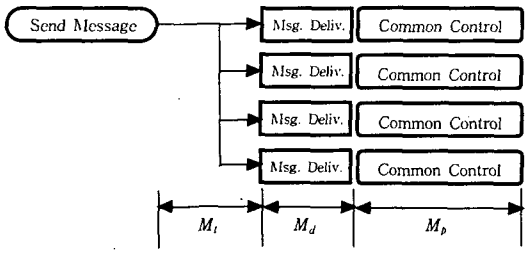
<그림 4> ID_Number Indexing을 가지는 메시지 처리 방식

이러한 다중 메시지 처리 방식은 GWES내에서 메시지 전달의 속도를 향상시키고 기존의 Common Control들이 가지고 있던 메시지 처리 루틴을 제거함으로서 메시지 처리 속도를 줄일 수 있다.

PCS 소형 운영체제의 GEWS에서 제안된 다중 메시지 처리 방식을 사용함으로서 사용자가 선택을 하고, 명령을 수행하고, 입력과 출력을 수행하는 표준적인 방법으로 제공되는 컨트롤들, 메뉴들, 다이얼로그 박스들 그리고 자원들에 대해서 기존의 방법보다 성능 향상을 가져올 수 있으므로 최적화된 API를 지원할 수가 있다. 이러한 다중 메시지 처리 방식은 메시지 처리 방식과 API Library들을 하나로 통합된 GWES에서 효율적이다.

3.3 메시지 지연시간 분석

GWES에서 메시지 전송 부분을 도식화하면 <그림 5>과 같이 메시지 전송(Transmission), 전달(Delivery), 그리고 처리(Process)에서 지연 시간(Delay time)이 존재한다[6].



<그림 5> 메시지 지연시간 분석

메시지가 전달되어 실행될 때까지 걸리는 시간 M_s (Message send delay)를 <식 1>과 같이 정의 할 수가 있다.

$$M_s = M_t + M_d + M_p \qquad \text{<식 1>}$$

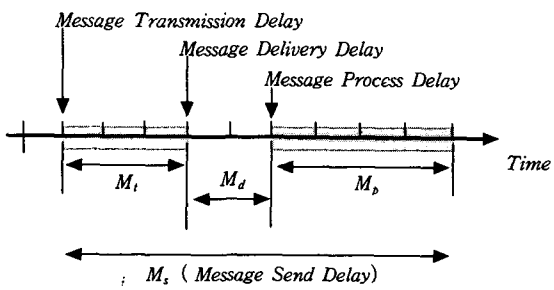
M_s : *Message Send Delay*

M_t : *Message Transmission Delay*

M_d : *Message Delivery Delay*

M_p : *Message Process Delay*

M_s를 도식화하면 <그림 6>과 같다.

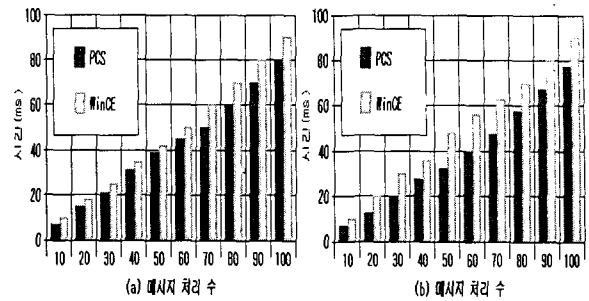


<그림 6> Message Send Delay

기존의 메시지 처리 방식이 하나의 메시지를 처리하는 데에는 M_s 만큼의 시간이 소비되지만 새로이 제안된 다중 메시지 처리 방식에서 제안한 ID Number Indexing방식을 적용 할 경우에는 기존의 메시지 구조를 보내는 대신에 그보다 더 적은 정보량을 가지는 ID Number를 전송하게 되므로 M_t의 시간을 줄일 수 있고, M_d의 시간이 소비되지 않으므로 전체 메시지 지연 시간을 줄일 수가 있다.

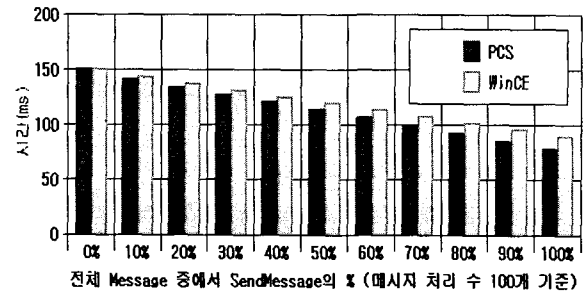
4 성능 평가

본 논문에서 구현한 PCS 정보단말기용 소형 운영체제에 적합한 API와 기존의 Windows CE용 API의 메시지 처리에 대한 성능을 실험 평가를 통하여 비교한다. 실험은 같은 하드웨어 환경에서 메시지를 처리하는 테스트 프로그램을 수행하는 방식으로 성능을 측정하였다. <그림 7>에서 (a)는 ID_Number와 WM_Number 2개를 사용할 때(ListBox, ComboBox등의 경우)이며 (b)는 ID_Number만으로 인덱싱 할 때(PushButton, EditButton등의 경우)를 실험하였다.



<그림 7> Windows CE와 PCS용 API의 성능 비교 1

실험 결과와 같이 메시지 처리 횟수가 적은 경우에도 성능 향상을 볼 수 있으며 메시지 처리 횟수가 많아 질 수록 기존의 API 보다 새로 제안된 API의 메시지 지연 시간이 줄어드는 결과를 볼 수 있다.



<그림 8> Windows CE와 PCS용 API의 성능 비교 2

<그림 8>은 한 쓰레드 내에서 발생하는 전체 메시지들 중에서 SendMessage에 대한 비중에 따라서 실험한 것이다. 여기서 SendMessage가 전체 메시지들 중에서 차지하는 비중은 프로그램의 종류에 따라 다르지만, 윈도우 시스템에서 GUI 쓰레드에 대한 우선순위가 높고, 이러한 GUI를 제어할 때 사용하는 SendMessage는 사용자 응답시간에 영향을 주게 되므로 전체 시스템 메시지 처리에 대해서 성능 향상의 결과를 보여준다.

5. 결론

본 논문에서는 PCS 정보단말기용 소형 OS에 적합한 API를 설계할때 GWES에 다중 메시지 처리 방식을 사용하여 최적화된 API 구조를 지원하여 성능향상을 보았다. 향후 연구 방향으로는 같은 쓰레드 내의 Common Control들에 대해 적용되는 다중 메시지 처리 방식을 확장하여 다른 쓰레드들 간의 최적화된 메시지 처리 방식에 대한 연구가 필요하다.

참고문헌

[1] John Murray, "Inside Microsoft Windows CE", Microsoft Press, 1998

[2] Kanevsky, A., et. al., "Design of a Quality of Service (QoS) Driven Adaptive Computing", MITRE Technical Report 98B002, January 1998

[3] Kimberly Gregory, "Embedded Development with Microsoft Windows CE", Microsoft Embedded Review, March 1998

[4] Microsoft, "Microsoft Windows CE Platform SDK version 2.0", http://www.microsoft.com/windowsce, 1998

[5] Yangmin Seo, Jungkeun Park, Seongsoo Hong, "Supporting Preemptive User-Level Threads for Embedded Real-Time Systems", Technical Report SNU-EE-TR-1998-1 Seoul National University, August 1998

[6] 신현식, 김태웅, 서울대학교, "실시간 시스템의 개관", 한국정보처리학회지, 1998.7