실시간 운영체제 VxWorks 상에서 통신 미들웨어 TAO의 실시간성 지원에 대한 연구

A Research to support Real-time of TAO on VxWorks Real-time Operating System

임재석*, 손재열**, 이용태**, 이철훈*
충남대학교 컴퓨터공학과*,
(주)한화종합연구소**

Lim jae-seok*, Son jae-yeol**,
Lee yong-tae**, Lee cheol-hoon*
Choongnam Univ. Computer Engineering*,
Hanwha Corp.**

요약

분산 환경에서 이기종 시스템 간의 독립적이고 표준화된 환경을 지원하기 위해 RT-CORBA(Real-Time Common Object Request Broker Architecture) 기반의 오픈소스 TAO(The ACE ORB) 통신 미들웨어에 대한 연구가 활발해지고 있다. 실시간성 측면에서 TAO는 Windows나 Linux와 같은 범용 운영체제에서는 실시간성을 지원하지 않으며, VxWorks, LynxOS 등과 같은 실시간 운영체제에 의존하여 실시간성을 지원한다. 이에 본 논문에서는 실시간 운영체제인 VxWorks에 TAO 통신 미들웨어를 이식함으로써 TAO가 쓰레드 레벨의 실시간성을 지원할 수 있는 환경을 구축한다. 또한 TAO를 탑재한 범용 운영체제 Windows 및 Linux와의 통신을 통해 VxWorks로의 TAO 이식을 검증한다.

Abstract

To support distributed environment that is independent and standardized between heterogeneous system, a study on the TAO(The ACE ORB) communication middleware based on the RT-CORBA is highly being done. The TAO supports real-time features not on the general purpose operating system like Windows or Linux but on the real time operating system like Vxworks or LynxOS. In this paper, we build and construct environment that supports a thread-level real time feature by porting the TAO on VxWorks target. Also we validate the ported TAO by network communication with the TAO of Windows.

Ⅰ. 서론

최근 고성능의 마이크로 프로세서와 고속 통신망 기술이 결합하여 수많은 CPU를 상호 유기적으로 동작하는 분산 시스템에 대한 연구가 활발하다. 분산 시스템은 하나의 프로그램이 네트워크로 연결된 여러 대의 컴퓨터에 분산되어 실행되면서 마치 단일 시스템에서 동작하는 것처럼 구현된 시스템이다[1]. 그러나 분산 환경에서의 각 시스템은 이기종의 하드웨어와 운영체제를 사용하기 때문에 통신을 위한 표준 규격이 요구되었으며[2], OMG(Object Management Group)에서는 이러한 통신 미들웨어의 표준 규격인 CORBA를 제안하였다[3]. 특히 응용프로그램의 실시간성 지원을 위해 RT-CORBA(Real-Time CORBA)를 표준화하였으며, TAO는 RT-CORBA를 기반으로 실시간성을 제공하는 대표적인 통신 미들웨어이다[4].

TAO는 Windows나 Linux와 같은 범용 운영체제에서는 실시간성을 지원하지 않으며, VxWorks, LynxOS 등과 같은 실시간 운영체제에 의존하여 실시간성을 지원한다[4][5]. 본 논문에서는 이러한 통신 미들웨어 TAO의 실시간성 지원을 위해 실시간 운영체제 VxWorks 상에 TAO를 이식하기 위한 방안을 연구하고, VxWorks에 이식된 TAO와 범용 운영체제 Windows에 이식된 TAO와의 통신을 통해 VxWorks로의 올바른 TAO 이식을

검증한다.

본 논문은 2장에서 관련연구로 실시간 운영체제 VxWorks에 대해 간략히 소개하고, VxWorks에 이식할 통신 미들웨어 TAO에 대해 살펴본다. 3장에서는 VxWorks에 TAO를 이식하기 위한 방안에 대해 설명하고, 4장에서 TAO의 이식결과와 함께 VxWorks에 대한 TAO 이식을 검증하며, 5장에서 결론 및 향후연구과제에 대해 기술한다.

Ⅱ. 관련연구

1. VxWorks

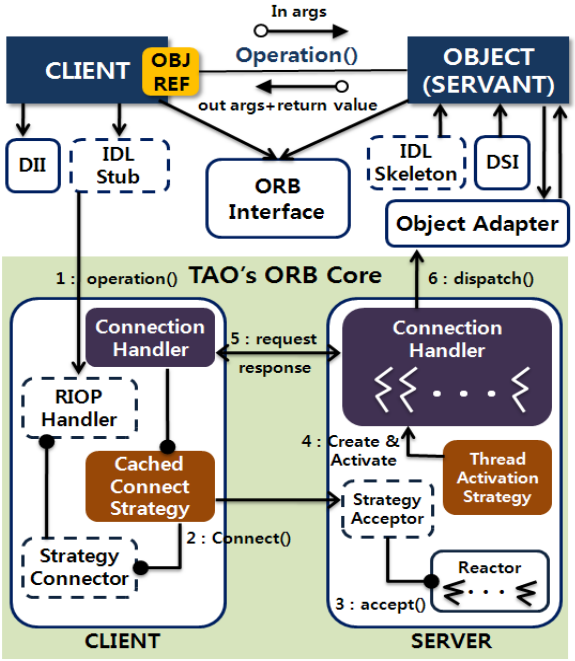
VxWorks는 Wind River사에서 제작하여 판매하는 상용 실시간 운영체제이다. 선점형 스케줄러 기반의 빠른 멀티태스킹 커널로, 빠른 인터럽트 응답시간과 확장된 ITC(Inter Task Communication)을 지원한다. 0부터 255의 총 256단계 우선순위를 사용하며, 동일 우선순위의 태스크 사이에는 선택적 라운드로빈 스케줄러를 사용하고, POSIX 인터페이스와 AMP(Asymmetric Multi-Processor)를 지원한다. 또한 VxWorks는 사용자 인터페이스를 위한 WindSh라고 부르는 셸을 지원한다. 셸은 심볼릭 또는 소스 수준의 디버깅 기능, 성능 모니터와 파일 시스템 입출력 등을 지원한다[6].

VxWorks는 x86 시리즈, MIPS, PowerPC, SH-4 등 많은 CPU에 이식되었으며, 우주 및 항공 산업, 국방산업, 산업용 로봇 등 실시간성을 위한 분야에 사용되고 있다.

2. TAO

TAO는 DOC(Distributed Object Computing)그룹에서 만들어진 DRE(Distributed Real-time and Embedded) 시스템을 위한 CORBA 기반의 분산 미들웨어 플랫폼으로, OS 독립적인 네트워크 환경을 구축해주는 프레임워크 ACE(Adaptive Communication Environment)를 사용한다[7]. 현재까지도 계속 버그를 수정 중에 있으며 2009년 4월 현재 최신 릴리즈 버전은 ACE+TAO 5.6.8(Latest Micro Release Kit) 이다. 각 버전별로 VxWorks를 지원함에 있어서 약간의 차이가 있는데, ACE+TAO 5.6.2버전 이상부터 VxWorks 5.X.X 및 6.X 버전 모두에 대해 지원을 한다[8][9].

TAO는 OS 이식성, 통신접속관리, 이벤트 디멀티플렉싱(Event demultiplexing), 이벤트 핸들러 디스패치(Event handler dispatching), 동기화(Synchronization), 고장탐지(Fault detection), 고장감내(Fault tolerance) 및 호스트와 호스트 간 지연 조절(Latency controlling) 등의 다양한 QoS를 위한 기능을 제공한다. 또한 재사용이 가능한 C++ wrapper facades와 다양한 플랫폼에서 수행 가능한 프레임워크 컴포넌트를 제공하고 있다. 그림 1은 TAO의 ORB 코어를 나타낸다.



▶▶ 그림 1. TAO의 ORB 코어

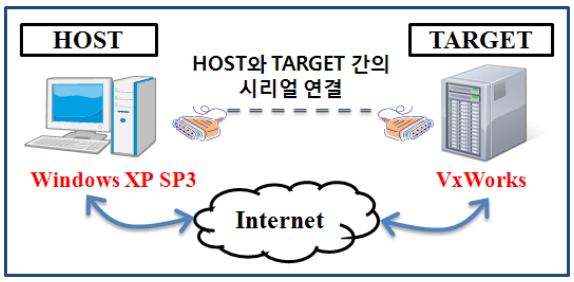
TAO는 RT-CORBA를 기반으로 운영체제의 실시간 스케줄링 기법에 의존하여 실시간성을 지원한다. TAO의 실시간 스케줄링 서비스는 오프라인(off-line) 스케줄링과 온라인(on-line) 스케줄링으로 구분되는데, 스케줄링 서비스는 모든 IDL(Interface Definition Language) 연산의 실행 가능성 여부를 분석한 뒤 그 결과를 기반으로 충분한 CPU 자원을 가지고 연산이 수행될 수 있는지 여부를 분석하게 된다. 또한 스케줄링 서비스는 실행 시에 TAO의 실시간 ORB endsystem이 우선순위에 접근할 수 있도록 우선순위를 제공하는 역할을 수행한다[10].

Ⅲ. TAO 이식을 통한 실시간성 지원

TAO는 Windows, Linux, VxWorks 등 다양한 플랫폼에 이식하기 위해 각 플랫폼에 맞는 빌드환경을 제공한다. VxWorks의 경우 크로스 컴파일(Cross-Compiling)을 한 후, 생성된 라이브러리를 VxWorks에 이식해야 하기 때문에, Windows를 HOST로, VxWorks를 TARGET으로 하여 TAO를 VxWorks에 이식함으로써 실시간성을 지원하는 방안에 대해 설명한다.

1. TAO 이식을 위한 H/W 및 S/W 구성

TAO를 VxWorks에 이식하기 위한 전체적인 하드웨어 구성을 그림 2에 나타내었다.



▶▶ 그림 2. TAO 이식을 위한 하드웨어의 구성

HOST는 Windows XP 서비스팩3을 운영체제로 한 Intel® Core2DUO 3.0GHz CPU를 사용하였고, TARGET은 PPC604기반의 SVME/DMV-183 BSP 지원하에 VxWorks5.5.1 버전을 사용하였다. 표 1은 TAO 이식을 위해 필요한 소프트웨어와 버전을 나타낸다.

표 1. TAO 이식을 위한 소프트웨어의 구성

소프트웨어 이름	버전
Microsoft Visual Studio	VC7 이상
Tornado	2.2.1(VxWorks5.5.X)
Cygwin	1005.25.0.0
ActivePerl	5.10.0.1004-MSWin32-x86
ACE+TAO	ACE Version 5.6.4
	TAO Version 1.6.4

2. TAO 이식을 위한 환경구성

2.1 Tornado의 환경구성

Tornado 2.2.1은 기본적으로 make 3.74 버전을 제공한다. 하지만 TAO의 빌드를 위해서는 make 3.80 GVK Patches 버전의 make가 필요하다. Wind River 사에서 해당 패치파일을 제공하지만 소스코드 형태로 되어 있기 때문에 MSVC 4.0을 이용해 make 3.80 GVK Patches를 컴파일 한 후 이를 사용한다. 또한 make의 새로운 규칙과 패턴을 적용하기 위해 depend.tcl 파일을 make.exe와 동일한 폴더에 복사하며, 라이브러리 생성에 필요한 호스트 독립적인 규칙과 의존적인 규칙을 정의한 파일을 업데이트 한다.

2.2 TAO의 환경구성

VxWorks TARGET을 위한 ACE+TAO 빌드는 Windows HOST에서 이루어진다. 즉, Windows HOST에서 우선적으로 ACE+TAO 빌드를 통해 얻어진 dll 파일을 가지고 VxWorks TARGET을 위한 빌드를 수행하는 것이다. ACE+TAO는 각각의 플랫폼을 위한 헤더파일을 정의하였으며, 컴파일러가 이를 선택해서 빌드할 수 있도록 CONFIG.H 헤더파일을 생성한다. CONFIG.H 파일에는 이식될 TAO가 가지는 여러 가지 기능을 선언할 수 있으며, 이를 표 2에 나타내었다.

표 2. platform_macros.GNU 파일 구성의 예

CONFIG.H 파일의 내용
#define ACE_HAS_IP_MULTICAST #if defined (_MSC_VER) || defined (__BORLANDC__) /* Windows를 위한 헤더파일 선언 */ # include "ace/config-win32.h" #else /* VxWorks를 위한 헤더파일 선언 */ # include "ace/config-vxworks5.x.h" #endif

또한 make 명령어와 함께 명시해야 할 옵션을 정의하는 파일인 platform_macros.GNU를 생성해야 한다. 여기에는 특정 플랫폼, 동일 플랫폼의 버전 혹은 컴파일러에 따라 적절한 Makefile 구성정보를 기술하며, 표

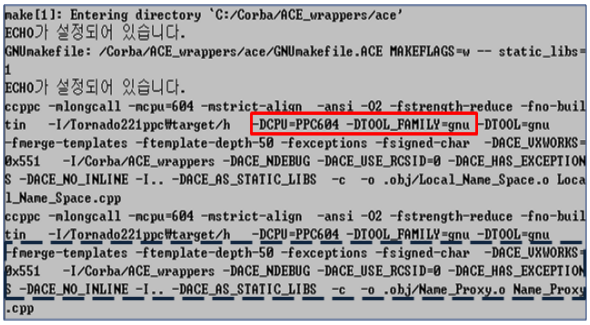
3에서 platform_macros.GNU 파일 구성의 예를 보이고 있다.

표 3. platform_macros.GNU 파일 구성의 예

platform_macros.GNU 파일의 내용
CPU=PPC604
debug=0
optimize=1
QtReactor=0
XtReactor=0
partial_links=1
include $(ACE_ROOT)/include/makeinclude/platform_vx
works5.5.x.GNU
ACE_COMPONENTS=FOR_TAO

3. VxWorks로의 TAO 이식

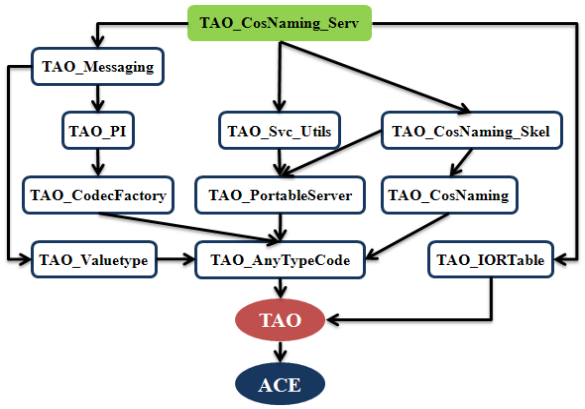
TAO 이식을 위한 환경구성을 완료한 후, Windows HOST에서 VxWorks TARGET을 위한 라이브러리의 빌드를 수행한다. 모든 라이브러리에 대해 make의 빌드 옵션은 정적 라이브러리에 대해서는 'static_libs=1'을, 동적 라이브러리에 대해서는 ' shared_libs=1'이다.



▶▶ 그림 3. VxWorks로의 이식을 위한 TAO 빌드

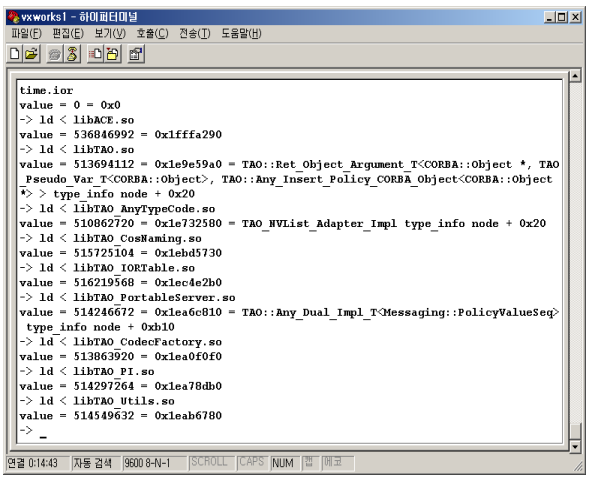
그림 3은 make명령을 수행한 후 TAO 라이브러리가 빌드되는 과정을 나타낸다. 실선으로 표시된 사각형 안의 내용에서 보는 바와 같이, platform_macros.GNU로부터 설정된 TARGET CPU, 디버깅 도구 등을 바탕으로 빌드가 수행된다. 또한 점선으로 표시된 사각형의 내용과 같이 CONFIG.H에서 정의한 옵션을 토대로 ACE와 관련한 빌드옵션이 추가되어 빌드가 수행된다.

TAO의 동적 라이브러리를 VxWorks에 이식할 때는 라이브러리간의 의존성을 고려해야 한다. C++ 객체지향 기반의 TAO 내에 정의된 여러 클래스들은 서로 상속관계에 있기 때문에 라이브러리가 이식될 때 해당 라이브러리에 대한 생성자가 호출된다. 이 때 필요한 클래스 생성자가 존재하지 않으면 라이브러리 이식이 실패하기 때문에 반드시 라이브러리 의존성에 따라 순차적으로 라이브러리를 이식해야 한다. 그림 4는 TAO_CosNaming_Serv 라이브러리에 대한 의존성을 도식화 한 것이다.



▶▶ 그림 4. TAO_CosNaming_Serv 라이브러리의 의존성

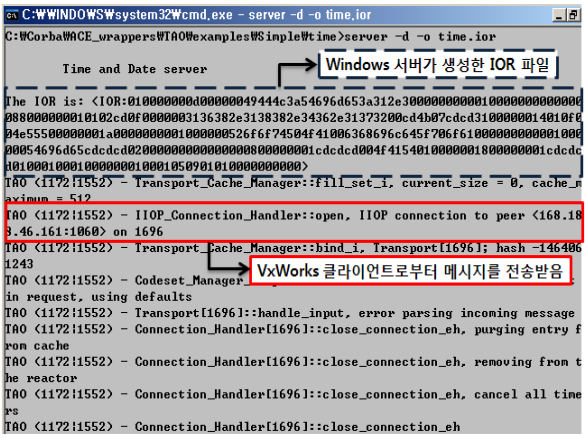
VxWorks로의 이식을 위한 TAO 라이브러리가 모두 빌드가 되면 VxWorks의 사용자 인터페이스인 WindSh 셸의 'ld'함수를 통해 TAO 라이브러리를 이식한다. 그림 5는 VxWorks를 위해 생성된 TAO 라이브러리를 의존성을 고려한 순서대로 이식하는 모습을 보인다.



▶▶ 그림 5. VxWorks로의 TAO 라이브러리 이식

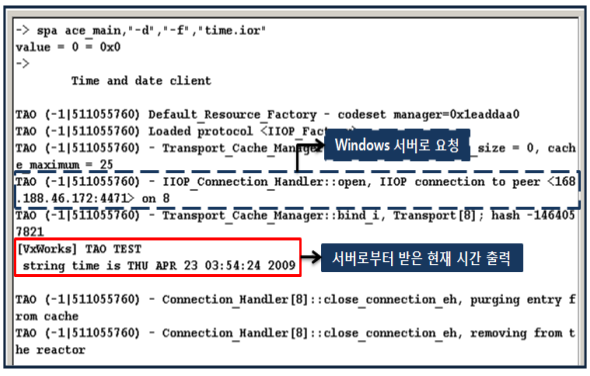
Ⅳ. TAO 이식결과 검증 테스트

VxWorks 상에 TAO가 올바르게 이식되었음을 검증하기 위해 Time Stamp 예제를 통해 Windows 운영체제와의 시간 동기화 테스트를 진행하였다. Windows 서버 프로그램이 통신을 위한 IOR(Interoperable Object Reference)파일을 생성하여 클라이언트 요청을 기다리고 있다가 요청이 들어오면, 요청한 클라이언트의 IP정보와 Port번호를 출력한 후, TAO 통신 메커니즘의 절차를 순서대로 출력한다. 클라이언트의 요청을 모두 처리한 서버는 해당 연결을 종료하고 다시 요청대기 상태로 돌아가 요청을 기다리게 된다. 그림 6은 Windows에서 실행된 Time Stamp 서버의 모습이다.



▶▶ 그림 6. Windows의 Time Stamp 서버

. 실행 옵션 '-d' 는 디버깅 모드로 실행함을 의미하고, '-o time.ior' 은 클라이언트와의 통신을 위해 서버가 생성할 IOR 파일의 이름이다.



▶▶ 그림 7. VxWorks의 Time Stamp 클라이언트

그림 7은 VxWorks에 클라이언트 라이브러리를 적재하고 실행하는 모습이다. 실행 옵션 '-f time.ior' 은 서버에 의해 생성된 IOR 파일의 정보를 해석하여 통신하겠다는 의미이다. 서버로부터 받은 메시지를 해석하여 점선으로 그려진 박스에 보이는 것처럼 서버의 IP주소와 연결포트를 출력하고, 현재 시간을 출력한다.

Ⅴ. 결론 및 향후연구과제

본 논문에서는 통신 미들웨어 TAO의 실시간성 지원을 위해 실시간 운영체제 VxWorks 상에 TAO를 이식하기 위한 방안을 연구하고, 올바른 TAO 이식을 검증하였다.

VxWorks에 TAO를 이식하기 위해 Cygwin, ActivePerl을 설치하였으며, Tornado가 제공하는 GCC 2.96 컴파일러 및 링커를 사용하기 위한 Makefile 옵션을 VxWorks 5.5.1버전에 적합하도록 수정하였다. 또한 Cross-Compile Make 옵션을 사용하기 위해 platform_macros.GNU 파일을 생성하고 VxWorks5.5.1을 위한 GNU Makefile을 포함하도록 하였다. TAO 빌드를 통해 얻어진 동적 라이브러리간의 의존성을 파악하여 순차적으로 VxWorks에 이식하고, Windows상의 TAO와의 시간 동기화 예제를 통해 이식된 TAO의 기능을 검증하였다.

향후 연구과제로 TAO의 버전이 높아짐에 따라 발생하는 VxWorks로의 TAO 이식에 대한 호환성 문제를 해결하기 위한 연구가 필요하다. 또한 TAO에 사용되는 모든 라이브러리를 VxWorks에 이식할 경우 약 80MB 크기의 메모리 공간이 필요한데, 이는 자원제약이 심한 임베디드 시스템의 특성을 고려할 때 심각한 문제를 초래할 수 있기 때문에 TAO 라이브러리를 임베디드 시스템에 사용 가능하도록 모든 라이브러리의 의존성을 파악하여 필요한 라이브러리만을 이식할 수 있는 방안에 대한 연구가 필요하다.

▌참 고 문 헌▐

[1] George Coulouris, Jean Dollimore, Tim Kindberg,

"Distributed Systems Concepts and Design", Addison-Wesley, 2000.

[2] Ross J. Anderson, pp115-133, "Security Engineering: A Guide to Building Dependable Distributed Systems", WILEY, 2008.

[3] OMG, "CORBA(Common Object Request Broker Architecture) Specification, Version 3.1", 2008.

[4] Douglas C. Schmidt, David L. Levine, and Sumedh Mungee, "The Design of the TAO Real-Time Object Request Broker", Computer Communications, Vol.21, No.4, pp294-324, 2000.

[5] Douglas C. Schmidt, Fred Kuhns, "An Overview of the Real-time CORBA Specification", Computer, Vol.33, No.6, pp56-63, June 2000

[6] Glenn Seiler, "Wind River Linux and VxWorks Real-Time Capabilities : A Comparison", Wind River System, Inc., 2007.

[7] Stephen D.Huston, James CE Johnson, Umar Syyid, pp32-52, "ACE Programmer's Guide", 2003.

[8] http://www.theaceorb.nl/en/acetao.html, TAO commercial support

[9] DOC Group, "ACE-INSTALL Document", Jul 2006.

[10] OMG, "Realtime CORBA Joint Revised Submission", ORBOS/99-02-12 ed., Mar 1999.