내장형 실시간 응용 개발을 위한 도구에서 타겟 서버의 구현

임채덕, 공기석, 김흥남
한국전자통신연구원 컴퓨터.소프트웨어기술연구소
{cdlim, kskong, hnkim}@etri.re.kr

Implementation of the Target Server in the Development Tool for Embedded Real-Time Applications

Chaedeok Lim, Kisok Kong, Heung-Nam Kim
ETRI-Computer & Software Technology Laboratory

요 약

셋탑 박스와 같은 내장형 타겟 시스템의 자원 제약으로 인해 내장형 실시간 응용을 효율적으로 개발하기 위해서는 원격 호스트 상에서 동작하는 응용 개발 도구가 필요 시 된다. 본 논문에서는 호스트 상의 원격 디버거, 대화형 쉘, 자원 모니터와 같은 응용 개발 도구들의 타겟 접근을 최소화하고 통신 채널을 하나로 유지하기 위해 호스트와 타겟 간의 중개자 역할을 하는 타겟 서버를 설계 구현한 내용에 대해 기술한다. 우리가 개발한 타겟 서버는 호스트-타겟 통신 방식, 실행 모듈 형식에 독립적인 구조이며, 도구의 확장성을 고려하여 Open API 를 제공한다.

1. 서론

최근 새로이 저가형이면서 특정 응용 전용인 32 비트 내장형 마이크로프로세서의 출현과 통신 단말기나 가전 제품에 다양한 기능을 추가하기 위한 내장형 실시간 시스템[1] 응용 개발 수요가 증가하고 있다. 더구나 내장형 응용들이 점점 복잡해지고 있으며 이를 위한 개발 환경[2]은 타겟 시스템의 자원이 매우 제한적이며 호스트와 타겟 간의 통신 부담이 크다는 점에서 기존의 소프트웨어 개발 환경과 다르다.

호스트와 타겟 간의 통신 부담과 타겟 자원이 제한적이라는 문제를 해결하기 위해서, 개발 환경에 있어서 기존의 타겟에 의존적인 개발 도구들을 호스트로 이동시킬 필요성이 대두된다. 이들 도구를 호스트로 이동시킴으로써 호스트와 타겟 간에 정보를 주고 받기 위한 부가적인 시간과 대역폭을 감소시킬 수 있다. 또한, 타겟은 메모리와 같이 응용에 필요한 자원들이 더 넉넉해지게 된다. 이는 도구들이 차지하는 비중과 상당히 큰 심볼 테이블이 호스트에 위치하는 데서 비롯된다.[3]

호스트 상의 도구를 지원하기 위한 구조는 그림 1 과 같이 두 가지 경우가 가능하다. 그림 1 의 (a)는 각각의 도구가 타겟과 연결하여 타겟 자원에 접근하는데, 이 경우는 각 도구가 동시에 접근할 경우 타겟 시스템의 통신 부담이 매우 커진다. 그리고, 도구를 사용하는 응용 개발자가 타겟 시스템에 심각한 오류를 발생시켰을 경우에, 다른 도구를 사용하는 개발자도 타겟 시스템을 사용할 수 없게 된다. 서로 다른 도구일 지라도 타겟 시스템의 심볼 테이블과 같이 중복된 데이터를 각 도구가 가지고 있어야 하고 경우에 따라서는 중복 데이터에 대한 일관성도 문제가 될 수 있다. 또한 호스트 상에 새로운 도구를 추가할 경우에도 통신 방식, 타겟 시스템에 대해 상세히 파악해야 하므로 도구 개발자에게 큰 부담을 줄 수 있다. 그런데 그림 1 의 (b)와 같은 경우는 타겟 시스템과 호스트 상의 도구들 사이에 중개자 역할을 하는 구성 요소를 두는 구조이다. 이 중개자는 호스트와 타겟의 통신을 호스트와 타겟 간의 통신 채널을 하나로 유지하면서 중재하고, 도구들이 심볼 테이블을 공유할 수 있도록 심볼 테이블을 관리하고, 타겟에 있는 도구 전용 메모

리를 관리하는 등의 일을 처리 한다. 우리는 통신 문제와 타겟 자원을 효율적으로 사용할 수 있는 구조는 (a)보다는 (b) 구조라고 판단되어 (b) 구조의 중개자(타겟 서버라 명명)의 프로토타입을 개발하였다.

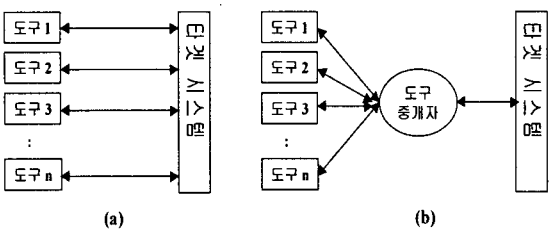

그림 1. 호스트 상의 도구가 타겟에 접근하는 방법

본 논문에서는 타겟 서버의 설계와 구현에 대하여 기술한다. 다음 절에서는 타겟 서버가 동작하게 될 내장형 실시간 응용 개발 환경에 대하여 소개한다. 제 3 절에서는 타겟 서버의 구조에 대해 기술하고, 제 4 절에서는 구현과 그 결과를 보여주고, 마지막 절에서 결론을 맺는다.

2. 내장형 실시간 응용 시스템 개발 환경

그림 2 는 우리가 개발한 내장형 실시간 응용 시스템을 위한 통합 개발 환경(IDE: Integrated Development Environments)이다. IDE 메인 프레임은 각 도구들을 제어하기 위한 Control View 를 제공하기 위한 것이며, 크로스 컴파일러, 원격 디버거, 대화형 쉘, 자원 모니터, 타겟 서버 관리 창을 선택하기 위한 메뉴 바를 가지고 있다.

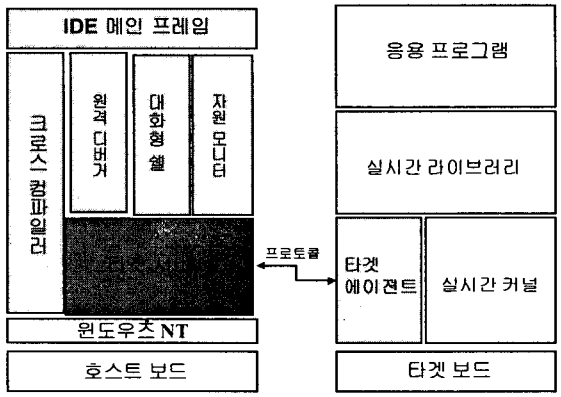

그림 2. 실시간 응용 개발 환경

크로스 컴파일러는 GNU C 언어 컴파일러를 기본으로 하고 있으며, gcc, make, ld, as 등과 같은 바이너리 유틸리티도 지원한다.

원격 디버거는 GNU 소스 레벨 디버거를 확장한 버전이다. 호스트 상에서 원격 디버거를 이용하여 개발자는 타겟의 실시간 시스템의 태스크를 생성하거나 디버깅할 수 있다.

대화형 쉘은 호스트 상에서 타겟 시스템의 모든 실행 시간 유틸리티들을 제공 받을 수 있는 도구이다. 즉, 타겟의 실행 시간 시스템 함수를 호출할 수 있고, 응용 프로그램 실행 모듈을 호출할 수 있으며, 응용 프로그램의 변수 값을 알아보거나 입력할 수 있으며, 새로운 변수를 생성하거나, 메모리에 있는 내용을 확인하고 수정하거나, C 연산자를 이용 계산을 할 수 있다. 또한 필수적인 디버깅 기능을 사용할 수 있다.

자원 모니터는 GUI 를 통해 시스템 정보를 제공해 준다. 메모리 할당 정보, 태스크, 메시지 큐, 세마포어 등 모든 시스템 오브젝트 정보를 쉽게 파악할 수 있다. 이들 정보는 정기적 혹은 개발자 요구에 따라 갱신된다.

타겟 에이전트는 타겟 서버와 함께 개발자가 통신 방식과 타겟 자원에 독립적일 수 있게 하는 IDE 의 주요 구성 요소이다. 호스트 상의 모든 개발 도구들은 타겟 서버를 통하여 타겟에 있는 타겟 에이전트를 통해서만 정보를 주고 받을 수 있다.

타겟 서버는 타겟과 도구들 사이에서 중개자 역할을 담당한다. 즉, 도구들은 타겟 서버가 제공하는 인터페이스를 통해 타겟 시스템에 다양한 요구를 하게 된다. 타겟 서버는 다양한 요구를 해석하여 적절한 처리를 수행한다. 타겟 서버가 타겟과 접속하기 위해서 타겟 에이전트와 연결하여야 하므로 이들 간에도 인터페이스가 존재한다. 이 인터페이스는 통신 방식에 무관하다.

타겟 시스템에는 실시간 운영체제 위에 개발하고자 하는 내장형 응용 프로그램의 개발 버전이 탑재되어 있다.

3. 타겟 서버의 구조

타겟 서버는 그림 2 에서 보여 주듯이 호스트 상에 위치하면서 타겟의 타겟 에이전트와 연결하여 응용을 개발함에 있어서 타겟 관리자와 같은 역할을 한다. 타겟 서버의 설계 원칙은 다음과 같다.

- 타겟 접근의 최소화
- 하나의 타겟에 하나의 통신 채널만 존재
- 도구 확장의 간소화

타겟 접근을 최소화 하기 위해 타겟의 심볼 테이블, 도구가 사용하는 타겟 메모리, 응용 프로그램의 실행 모듈에 대한 관리를 호스트 상에서 할 수 있도록 한다. 하나의 타겟에 하나의 통신 채널 만을 유지하는 것은 여러 개의 서로 다른 도구들이 하나의 타겟을 공유하게 하도록 하기 위함이다. 이미 상용화된 도구나 새로운 도구를 IDE 에 추가하고자 할 경우에 쉽게 확장할 수 있도록 타겟 서버는 Open API 를 제공하고 있다.

위 원칙을 만족시키기 위한 타겟 서버의 구조는 그림 3 과 같다. 각 구성 요소에 대해 간략히 설명하면 다음과 같다.

- Open C API: 도구들은 반드시 이 API 를 통해

서만 타겟 서버에 요구할 수 있다. 타겟 서버의 시작과 종료, 실행 시간 타겟 시스템 정보, 이벤트 관리, 디버깅 지원, 실행 모듈 관리, 심볼 관리, 도구가 사용하는 타겟 메모리 관리 등에 관한 기능을 가지고 있다.

- 심볼 테이블 관리자: 타겟의 시스템 심볼 테이블과 타겟에 로딩된 모든 실행 모듈들의 서브루틴과 변수, 모듈 id에 대한 정보를 관리 한다. 원격 디버거와 대화형 쉘과 같은 도구들이 서브루틴을 호출하는 것은 심볼 테이블 관리자를 통해서 가능하다. 특히, 로드된 모듈에 대한 심볼릭 disassembly 및 특정 태스크의 서브루틴 호출을 심볼릭하게 추적할 필요가 있는 소스 레벨 원격 디버깅에 유용하다.
- 실행 모듈 관리자: 실행 모듈 관리자는 크게 호스트에 있는 실행 파일을 타겟에 로딩/언로딩하는 기능, 로딩된 모듈 리스트를 호스트 상에서 관리하는 기능을 지원 한다. 로더는 COFF, ELF, a.out 과 같은 여러 형식을 지원하기에 용이한 DLL 구조를 갖고 있다. 타겟 서버 구동 시에 해당 파일 형식을 지원해주는 DLL이 타겟 서버 데몬에 연결된다.
- 타겟 메모리 관리자: 타겟 메모리 관리자는 도구들이 내장형 응용을 로딩 요구할 때와 같이 타겟 시스템의 메모리의 할당을 요구할 수 있다. 실시간 운영체제 구동 시에 도구들이 사용할 수 있는 타겟 메모리 크기를 운영체제 설정 파일에 세팅할 수 있고, 타겟 메모리 관리자는 처음에는 그 크기만 관리하다가 다 쓰게 되면 타겟 시스템에 추가 요청하여 관리해 준다.
- 통신 벡엔드 관리자: 통신 벡엔드 관리자는 벡엔드 관리자, 이더넷, 시리얼과 같은 통신을 지원해주는 각각의 벡엔드들로 이루어져 있다.
- 벡엔드 프로토콜: 타겟 에이전트와 통신하기 위해 필요하다. 이 프로토콜은 이더넷, 시리얼 통신에 공통적으로 적용된다.

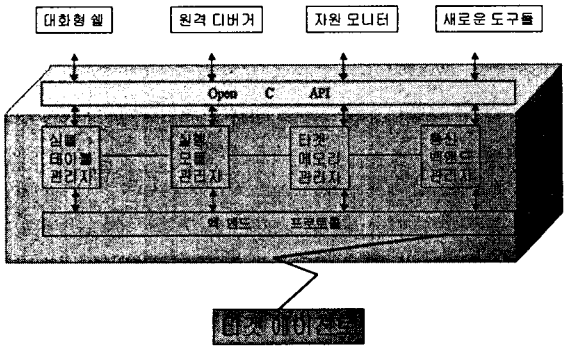

그림 3. 타겟 서버의 구조

4. 타겟 서버의 구현

타겟 서버의 프로토타입은 윈도우 NT에서 마이크로소프트 비주얼 C++/C로 구현하였다. 통신 방식은 NT에 포팅된 ONCRPC[6]를 사용하여서 이더넷 방식을 지원하고 있고, 실행 파일 형식은 COFF 만을 지원하고 있다. 타겟 보드는 StrongARM 보드를 사용하고 있고 타겟의 실시간 운영체제는 VxWorks[4][5]를 사용 시험하였다.

그림 4는 타겟 서버의 구현 결과의 한 예로 결과로 타겟 서버를 설정하여 구동시키는 창이다. 타겟 서버 데몬이 구동되면 호스트 상의 여러 도구들은 타겟 서버를 통해 타겟에 탑재될 내장형 응용 프로그램을 개발할 수 있다.

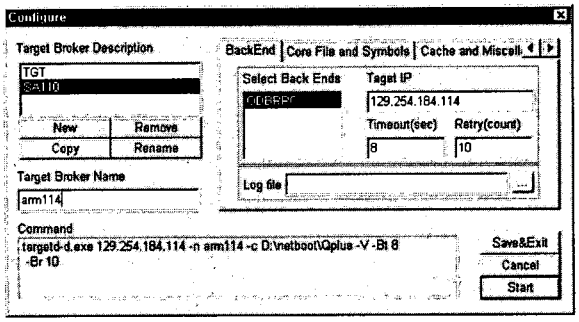

그림 4. 타겟 서버 설정 창

5. 결론

우리가 개발한 IDE는 내장형 실시간 응용을 개발하고자 할 때 제한된 호스트와 타겟 간의 통신, 제한된 타겟 시스템의 자원, 도구들 간의 빈약한 공조 체제 등의 문제점을 제거하기 위해 확장한 호스트 중심 개발 환경이다.

본 논문에서는 IDE의 주요 요소인 타겟 서버의 프로토타입에 대해 기술하였다. 타겟 서버는 대화형 쉘, 원격 디버거, 자원 모니터와 같은 도구들이 타겟에 접근할 수 있도록 해주는 중개자 역할을 한다. 특히, 도구의 확장성을 고려한 Open API 구현과 다양한 실행 모듈 및 통식 방식 지원을 위해 DLL 형식으로 구현했다는 장점이 있다.

현재의 타겟 서버는 통신 방식은 이더넷, 실행 모듈 포맷은 COFF 만을 지원하고 있다. 앞으로 시리얼 통신 방식과, ELF 실행 파일 포맷을 우선적으로 지원하여 확장해 나갈 것이다.

참고문헌

[1] Phillip A. Laplante, Real-Time Systems Design and Analysis, pp. 315-326, IEEE Press, 1997.
[2] C. M. Krishna, Kang G. Shin, *Real-Time Systems*, pp. 176-183, McGraw-Hill, 1997.
[3] Microtec, Spectra Backplane Concepts, Microtec, 1997.
[4] WindRiver, "VxWorks 5.3.1 Reference Manaual", 1996.
[5] WindRiver, "Tornado API Guide 1.0.1", 1997.
[6] W. Richard Stevens, Unix Network Programming, Prentice Hall, 1994.