내장형 실시간 소프트웨어의 원격디버깅을 위한 디버그에이전트의 설계 및 구현

공 기 석°, 손 승 우, 임 채 덕, 김 흥 남
{kskong, swson, cdlim, hnkim}@etri.re.kr
한국전자통신연구원 컴소연 실시간시스템지원도구연구팀

A Design and Implementation of Debug Agent for Remote Debugging of Embedded Real-time Software

Kisok Kong, Seungwoo Son, Chaedeok Lim, Heung-Nam Kim
Real-time System Tool Research Team, Real-time Computing Dept., CSTL, ETRI

요약

인터넷에 접속되는 다양한 정보기기 등의 내장형 시스템에서 사용되는 실시간 소프트웨어를 개발하기 위해서는 자원이 풍부한 호스트 컴퓨터에서 동작되는 강력한 개발도구의 지원이 필수적이다. 디버그에이전트는 타겟시스템의 실시간 OS 상에서 실행되는 하나의 태스크로서, 디버거로 대표되는 호스트시스템의 각종 도구들과 디버그 프로토콜에 의해 정의된 메시지를 주고 받으며 이 도구들에서 내리는 명령을 타겟 상에서 수행하는 역할을 담당한다. 본 논문에서는 디버그에이전트를 포함하는 실시간 소프트웨어 개발 환경에 대해 살펴보고, 원격디버깅을 지원하는 디버그에이전트의 구조와 기능, 디버그 프로토콜에 대하여 기술한다.

1. 서론

최근 인터넷의 폭발적 성장과 함께 나타난 두드러진 현상은 컴퓨터뿐만 아니라 다양한 정보기기 들이 인터넷에 직접 접속 되기 시작했다는 것이다. 전화기, TV, PDA, 공작기계 등은 물론이고 냉장고, 전자레인지 등의 가전제품 들의 인터넷 접속까지 논의 되고 있는 단계에 이르고 있다. 이러한 기기들의 특징은 마이크로프로세서를 내장하여 여러 개의 실시간 응용프로그램을 수행한다는 것이다.

과거에 실시간 응용 프로그램의 개발은 운영체계(OS)의 지원 없이 주로 어셈블리언어를 사용하여 장치제어에서부터 사용자 인터페이스에 이르기까지 전체를 구현하였으나 인터넷 접속 기능 등 응용에서 요구하는 기능이 더욱 복잡하고 다양해지면서 실시간 OS의 지원이 없이는 개발이 불가능하게 되었다. 또한 고급언어를 사용하여 응용 프로그램을 작성하는 것이 일반적으로 되었는데, 내장형 (embedded) 시스템은 메모리나 디스크 등 자원이 한정되어 있어서 호스트 컴퓨터에서 주로 개발을 하여 타겟시스템의 실행코드를 생성해내는 교차개발 (cross development) 방법이 많이 사용되고 있다 [1][2]. 고급언어를 사용한 실시간 응용프로그램의 교차개발을 위해서는 호스트 컴퓨터 상에서 실행되는 다양한 개발도구의 지원이 필수적이다. 여기에는 디버거와 대화형 쉘 (interactive shell), 자원 모니터 (resource monitor) 등이 포함된다.

디버거는 도구 중 가장 중요한 것이라고 할 수 있는데, 타겟시스템에서 수행되는 프로그램을 호스트 컴퓨터에서 소스레벨 디버깅하는 *원격디버깅 (remote debugging)* 을 실현하기 위해서는 타겟시스템 내에 이를 지원해주는 태스크가 존재해야 하며 본 논문에서는 이를 *디버그에이전트 (debug agent)* 라고 정의한다. 호스트시스템에서 수행되는 심볼릭 디버거와 타겟상의 디버그 에이전트 간에는 원격디버깅을 위한 프로토콜이 존재하여 디버깅 명령 및 각종 디버깅 정보를 주고 받는데 사용된다 [3].

본 논문에서는 현재 ETRI에서 수행하고 있는 조립형 실시간 OS 개발과제의 하나로 개발중인 디버그 에이전트의 구조와 기능에 대하여 기술한다. 2장에서는 클라이언트 서버 구조로 되어있는 내장형 실시간 응용 프로그램의 개발환경에 대하여 살펴본다. 3장에서는 디버그 프로토콜에 대해 소개한 후 디버그에이전트의 구조와 각 모듈별 기능을 설명하고, 4장에서 결론을 맺는다.

2. 내장형 실시간 소프트웨어의 개발환경과 디버그에이전트

다음 그림 1은 클라이언트/서버 구조되어있는 실시간 소프트웨어의 개발환경을 보여준다 [4].

호스트시스템에 있는 타겟서버는 타겟시스템과 도구들 사이에서 일종의 중계자와 같은 역할을 한다. 도구들은 타겟서버에서 제공하는 인터페이스를 통해 타겟서버에게 다양한 요구를 하고, 타겟서버는 이를 디버그에이전트에게 전달하여 도구에서 원하는 기능을 수행하도록 한다. 타겟서버와 디버그에이전트간의 인터페이스를 위하여 프로토콜이 정의되어있다.

디버거는 호스트시스템과 상이한 CPU를 갖는 타겟보드상에서 수행되는 목적프로그램을 실행할 때 발생되는 오류를 찾기 위하여 목적프로그램을 타겟 보드에 적

재한 후 소스 라인단위로 실행하면서 흐름을 추적하고 원하는 곳에서 실행을 정지하는 등의 원격디버깅 기능을 갖는다.

대화형 쉘은 사용자가 실시간 OS를 조사하고 감시하는 기능, 대화식으로 응용프로그램을 작성하고 시험할 수 있는 기능, 메모리에 목적프로그램을 적재하거나 제거할 수 있는 기능을 제공한다.

자원 모니터는 타겟시스템의 OS, 태스크, 메모리, 스택 등에 관한 정보를 제공해 주고 모니터링할 수 있도록 한다.

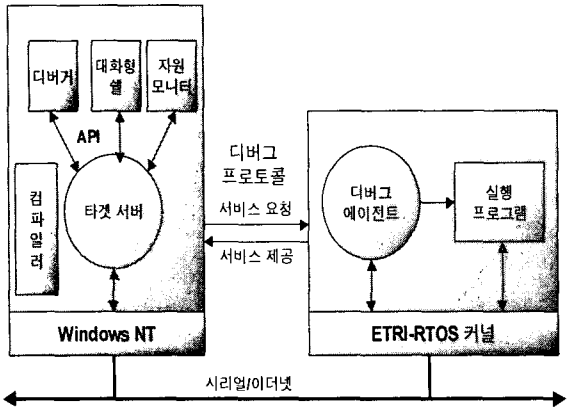

그림 1 실시간 소프트웨어 개발환경

디버그에이전트는 호스트 상의 도구들이 타겟서버를 통하여 원하는 기능을 수행할 수 있도록 타겟에서 대자로서의 역할을 한다. 디버그에이전트에서 수행하는 일을 간단히 정리하면 다음과 같다.

- 원격 디버깅 지원 기능: 정지점의 설정/해제, 타겟시스템의 메모리, 레지스터의 내용 조회 및 변경, 목적 프로그램의 실행을 제어
- 타겟에 적재된 오브젝트 코드내의 함수 호출
- 호스트시스템에 이벤트의 통지 (예외상황, 태스크의 종료)
- 가상 I/O 기능

타겟서버와 디버그에이전트는 ETRI에서 자체 정의한 *EDB (ETRI Debug) 프로토콜*을 사용하여 통신한다. EDB는 애플리케이션 레벨 프로토콜인데, 타겟과 호스트시스템의 OS에 대해 독립성을 갖도록 하였고, 타겟의 자원사용을 최소화 할 수 있도록 설계되었다 [5][6]. 하위레벨 통신 프로토콜로는 호스트-타겟 간 네트웍 프로그램의 편의성을 높이기 위하여 SUN의 RPC/XDR [7]을 사용한다. 패킷 전송에는 UDP/IP를 사용한다.

3. 디버그 에이전트의 구조 및 기능

ETRI-RTOS 상에서 수행되는 내장형 실시간 소프트웨어의 교차개발을 지원하기 위한 디버그 에이전트의 내부 모듈의 구성이 그림2에 나와있다.

타겟서버의 백엔드 모듈로부터 전달받은 EDB 프로토콜 메시지에는 다음과 같은 내용들이 포함된다.

- 호스트 내의 개발도구들 (디버거, 대화형 쉘, 자원모니터 등) 로 부터의 요청
- 타겟시스템상의 컨텍스트의 생성, 소멸, 재시작 요청
- 타겟시스템으로 부터의 이벤트
- 타겟 메모리에 적재된 오브젝트 모듈의 실행 및 종료 인지
- 타겟에 적재된 오브젝트 코드내의 함수 호출
- 호스트상의 도구들이 요청하는 타겟의 메모리, 레지스터의 상태정보
- 타겟시스템에 콘솔이 없을 경우 그 출력 값을 호스트상의 터미날로 출력할 수 있도록 가상 I/O로의 출력 요청

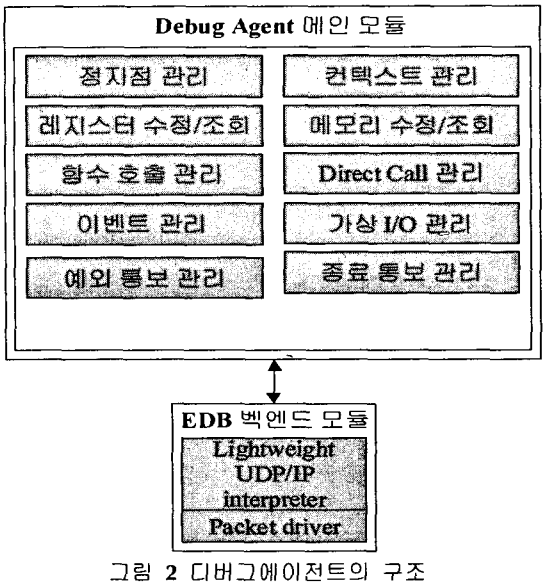

그림 2 디버그에이전트의 구조

그림 2에 나타난 각 모듈들은 EDB 프로토콜에 의해 전달된 이러한 요청들을 처리한다. 각 모듈별로 기능을 살펴보면 다음과 같다.

(1) 메인 모듈: 디버그 에이전트를 초기화하고 설정된 각 모듈을 초기화하여 타겟 서버의 요청시 응답할 수 있는 준비를 한다.

(2) 정지점 관리모듈: 호스트의 디버거에서 설정한 정지점 (breakpoint)을 처리하기 위하여 정지점들을 이중 연결 리스트 (doubly linked list)로 관리한다.

(3) 컨텍스트 관리모듈: 새로운 컨텍스트를 생성, 소멸, 재개, 중지 하는 등의 타겟상의 컨텍스트를 관리하는 서비스를 제공한다.

(4) 레지스터 수정/조회 모듈: 타겟의 레지스터 값을 얻어내고 새로운 값을 설정하는 RPC 함수호출 서비스를 제공한다.

(5) 메모리 수정/조회 모듈: 메모리 값에 대한 체크썸 (checksum) 계산, 메모리 읽기/쓰기, 메모리 채우

기, 메모리 보호, 메모리 이동 등의 기능을 수행한다.

(6) 함수 호출 관리모듈: 타겟 서버에서 요청한 함수를 수행하는 태스크를 생성하고 실행시킨다.

(7) Direct Call 모듈: 타겟상의 함수를 호출하지만 태스크를 생성하지 않고 디버그에이전트의 컨텍스트 내에서 수행한다.

(8) 이벤트 관리모듈: 이벤트포인트를 추가하고 삭제하는 기능을 수행한다.

(9) 가상 I/O 관리모듈: 타겟의 가상 I/O 채널에 데이터를 기록하는 서비스를 제공한다. 이는 타겟으로 부터의 출력 값을 호스트 도구에서 제공하는 가상 터미날로 전송하기 위한 것이다.

(10) 예외통보 관리모듈: 예외상황 (exception)을 호스트로 알리는 기능을 수행한다.

(11) 종료통보 관리모듈: 컨텍스트의 종료를 호스트에 알리는 기능을 수행한다.

(12) EDB 백엔드모듈: 물리적인 네트웍을 통하여 UDP/IP 패킷을 주고받는다. 타겟은 가용자원이 제한적이므로 경량 UDP/IP 해석기를 이용하여 타겟의 메모리 요구를 최소화한다.

타겟서버가 RPC를 통하여 보낸 EDB 메시지를 디버그에이전트가 처리하는 과정의 한 예를 다음 그림 3에서 보이고 있다.

4. 결론 및 향후과제

본 논문에서는 인터넷에 접속되는 정보기기등의 내장형 시스템에서 사용되는 실시간 소프트웨어의 개발을 위한 지원도구환경에 대하여 살펴보고 효율적인 교차개발을 지원하는 디버그 에이전트의 구조와 기능에 대하여 제시하였다.

최근의 내장형 소프트웨어에서 요구하는 매우 복잡한 기능을 지원하기 위해서는 강력한 개발도구의 지원이 필수적이다. 또한 호스트시스템에서 수행되는 도구들의 기능을 타겟시스템과 효율적으로 연결시키기 위해서는 플랫폼 독립적인 프로토콜을 갖고 수행되는 디버그 에이전트의 개발이 요구된다. 본 논문에서 제시된 디버그 에이전트는 타겟시스템의 제한된 자원을 효율적으로 이용하는 구조로 되어있어 다양한 실시간 시스템의 개발에 사용될 수 있을 것이다.

앞으로의 연구과제로는 EDB 프로토콜을 보다 효율적으로 설계하는 것을 들 수 있다. 현재 28개로 정의되어 있는 EDB 프로토콜의 갯수를 줄이고 특정 하드웨어나 실시간 OS 의존적인 부분을 보다 일반적인 형태로 수정하는 연구가 필요하다. 현재는 StrongARM 보드 [8]에 대해서 개발이 진행중이나 앞으로는 다양한 타겟 보드를 대상으로 디버그 에이전트의 이식성을 높이기 위한 시도가 요구된다.

또한 호스트, 타겟시스템 간에 주고 받는 메시지의 양을 줄이기 위한 연구도 중요하다. 보다 효율적인 메시지 형태를 갖게 되면 개발도구의 응답속도를 높일 수 있다.

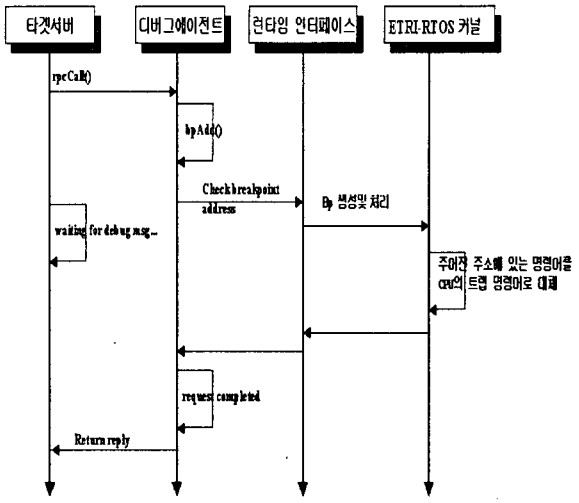

그림 3 정지점의 설정 과정

참 고 문 헌

[1] Jack G. Ganssle, "Debuggers for Modern Embedded Systems," in Embedded Systems Programming, Nov. 1998.

[2] 이은향 외 3 인, "교환 소프트웨어의 교차 디버깅을 위한 디버그 서버 구현," in JCCI'96 Proceedings, 광주, 1996.4.

[3] Jonathan B. Rosenberg, "How Debuggers Work," John Wiley & Sons, 1996.

[4] WindRiver, "Tornado User's Guide," 1995.

[5] WindRiver, "Tornado API Guide 1.0.1," 1997.

[6] Microtec, "Spectra Boot and VRTX Real-Time OS," 1996.

[7] W. Richard Stevens, "Unix Network Programming," Prentice Hall, 1994.

[8] Intel, "StrongARM EBSA-285 Evaluation Board," Oct. 1998.