SDT를 이용한 무선 트랜잭션 프로토콜의 프로토타입 구현*

정호원°, 오연주, 최윤석, 임경식
경북대학교 컴퓨터과학과
{won, kijo, yoonsuk, kslim}@ccmc.knu.ac.kr

An Implementation of the Wireless Transaction Protocol Prototype using SDT

Howon Jung°, Yeunjoo Oh, Yoonsuk Choi , Kyungshik Lim
Dept. of Computer Science, Kyungpook National University

요 약

정보통신 분야의 급격한 발달은 다양한 프로토콜을 지원하는 통신 시스템의 개발 주기를 더욱 빠르게 하고 있다. 따라서, 소프트웨어 개발자들은 공식적인 규격에 따른 소프트웨어의 설계와 실제 구현단계 이전에 프로토타입의 구현을 통해 시스템의 요구사항 분석 및 문제점을 조기에 발견하고, 소프트웨어의 개발 주기를 단축시키며, 한편으로는 개발 공정의 체계적 관리를 필요로 하게 되었다. 본 논문에서는 이에 대한 실질적인 해결방안을 제시하기 위해 통신 시스템의 명세서 작성 및 설계를 위한 표준화된 언어인 SDL(Specification and Description Language)을 이용해 WAP(Wireless Application Protocol)의 WTP(Wireless Transaction Protocol)를 설계하고, Telelogic사의 SDT를 이용하여 실행코드를 만들어 리눅스상에서 실제로 실행 가능한 프로토타입을 구현하였다.

1. 서론

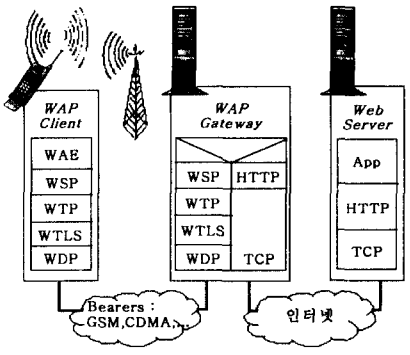
정보통신 분야의 급격한 발달로 인하여 다양한 프로토콜을 지원하는 통신 시스템의 개발 주기는 더욱 빨라지고 있으며 개발 여건 역시 더욱 복잡화되고 있다. 따라서, 소프트웨어 개발자들은 공식적인 규격에 따른 소프트웨어의 설계와 실제 구현단계 이전에 프로토타입의 구현을 통해 시스템의 요구사항 분석 및 문제점을 조기에 발견하고, 소프트웨어의 개발 주기를 단축시키며, 개발 공정의 체계적 관리를 필요로 하게 되었다. 한편 CCITT(현재 ITU-T)는 통신 시스템의 프로토콜을 기술하고 기능을 정의하며 시스템을 체계적으로 설계하기 위한 표준화된 형식언어인 SDL을 권고하였다[1,2,3].

논문 [7]에서는 소프트웨어 명세 언어인 SDL를 이용한 WAP 프로토콜 스택의 WTP 모델링 및 서비스 프리미티브와 SDL 명세에서의 시그널과의 관계 규정 등에 관한 연구가 있었다. 본 논문에서는 위 연구의 바탕 위에서 시스템의 체계적인 분석 및 프로토타입의 생성을 위하여 SDL 시스템을 객체지향 방법론에 입각하여 설계하였고, 클라이언트-서버 모델에 기반한 프로토타입을 개발하기 위한 전체 시스템 모델링, SDL 시스템과의 연결 및 WTP 서비스를 위한 API(application programming interface)의 정의를 하였으며, 결론 및 향후 연구 방향을 제시하고 있다.

2. 관련기술 개요

2.1 WAP과 WTP

무선 인터넷 서비스를 위한 WAP 모델은 <그림 1>과 같다. WAP 프로토콜 스택의 기본 모듈인 WTP 프로토콜은 데이타그램 서비스 상에서 동작하는 트랜잭션 지향 프로토콜로 상호적인 "브라우징"(요구/응답) 응용 소프트웨어에 대해 비 신뢰적 단방향 요구(클래스 0), 신뢰적 단방향 요구(클래스 1), 신뢰적 양방향 요구/응답(클래스 2)의 차별화된 트랜잭션 서비스를 제공한다[4,5,6,7].

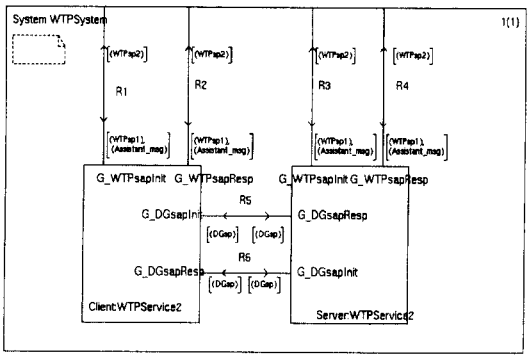


<그림 1> WAP 모델과 프로토콜 스택

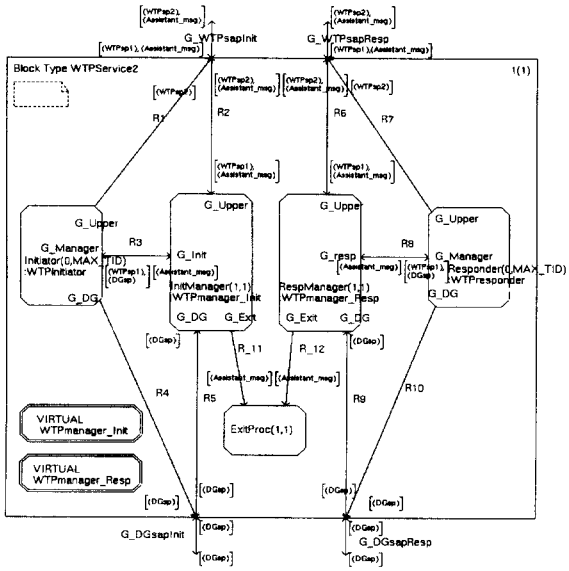
2.2 SDL(Specification and Description Language)

SDL은 통신 시스템을 명세, 기술하기 위한 표준화된 형식언어이다. SDL은 시스템과 그 환경과의 상호 동작을 표현하는데 유용하며, 개괄적이고 총괄적으로 시스템을 나타낼 수도 있고, 세부적으로 자세히 설계할 수도 있다. SDL 관련 지원도구는 일반적으로 명세 편집기능, 시뮬레이션 기능, 검증 및 목적코드 자동생성기능 등을 지원한다. 특히 목적코드의 자동생성은 시스템의 분석 단계에서 실제 시스템에서 동작하는 프로토타입의 구현을 가능하게 하며, 이는 프로토콜에 대한 이해와 요구사항의 분석 및 문제점의 조기발견 등을 가능하게 하여 프로그램의 목표를 신속하고 정확하게 이해할 수 있도록 한다[3,7,8].

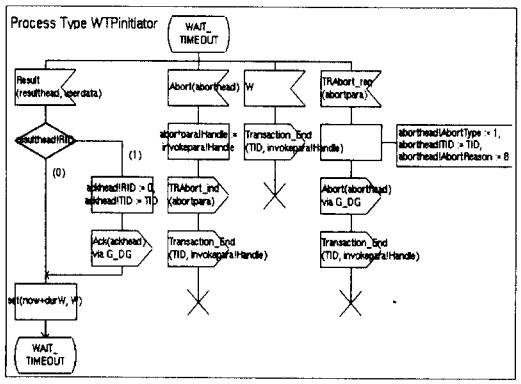
*본 연구는 한국과학재단 목적기초연구(1999-2-303-004-3) 지원으로 수행되었습니다.



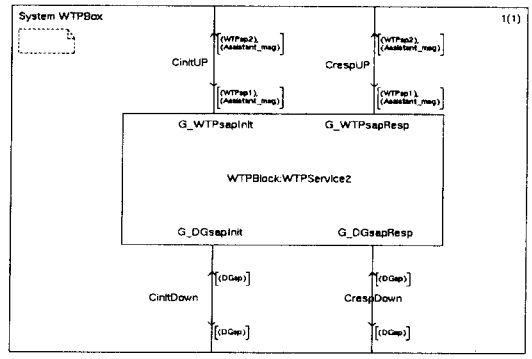
<그림 2> 클라이언트-서버 모델의 시스템 다이어그램



<그림 3> 블록 타입 다이어그램



<그림 4> 프로세서 타입 다이어그램



<그림 5> 프로토타입 개발을 위한 시스템 다이어그램

3. SDL을 이용한 시스템 설계

<그림 2>는 SDL 시스템을 클라이언트-서버 모델로 구성한 예이다. 이러한 시스템 형태는 WTP 프로토콜 스택에 대한 분석 및 이해를 도와주며, WTP 프로토콜의 동작이 적절히 이루어지는지를 SDL 관련 지원 도구를 통해 시뮬레이션하고 검정할 수 있게 해준다. 시스템 내의 클라이언트와 서버 블록은 모두 WTP서비스(WTPService2) 타입을 상속받으며 이 블록 타입의 형태는 <그림 3>과 같다. 블록 내에는 5개의 프로세서가 있다. 요구관리자(InitManager)와 응답관리자(RespManager) 프로세서는 WTP의 멀티트랜잭션을 관리하기 위해 존재하며, 요구자(Initiator)와 응답자(Responder) 프로세서는 개개의 트랜잭션을 처리한다. 각각의 프로세스 타입은 <그림 4>와 같이 SDL 프로세서 정의의 심볼들로 나타낼 수 있다. 프로세서는 상태(state)와 메시지, 테스크(task)를 가지며, 메시지의 입력에 따라 테스크를 처리하고 상태를 변화시키는 형태로 진행된다. 또한 타이머를 정의하고 사용할 수 있으며, 프로세스 간 및 프로세서와 외부와의 통신을 메시지로 정의하여 사용한다. 시스템을 이루는 주요 블록과 프로세서는 타입의 형태로 되어있어 재사용이 가능하다. <그림 5>는 블록 타입을 이용하여 시스템 다이어그램을 다시 구성한 예이다. 이 시스템 다이어그램은 <그림 2>의 시스템 다이어그램을 통해 분석되고 개발된 WTP 블록 타입을 이용해 프로토타입에 사용될 목적코드를 생성하는 데 사용될 수 있다.

4. 클라이언트-서버 모델을 위한 프로토타입 구성

<그림 6> SDT 환경함수의 역할

Telelogic사의 SDT는 SDL 관련 지원도구중 하나로 프로토콜의 구현을 위한 목적코드를 생성하는데, 이는 하나의 시스템 형태로 제공되며 SDL 시스템과 환경함수로 나누어진다. SDL 시스템은 <그림 6>과 같이 환경함수를 통해 외부환경(파일시스템, 네트워크, 다른 프로세서 등)과 통신을 하게된다.

WTP 프로토콜의 프로토타입을 생성한다는 것은 프로그래머가 응용프로그램 블록에서 프로토타입이 제공하는 API를 이용해 통신 프로그램을 작성할 수 있도록 개발환경을 제공하는 것이다. 이를 위해서 우선 프로그래머에게 제공할 프로그래밍 모

델을 정립하여야 하며 본 논문에서는 일반적으로 가장 널리 사용되는 클라이언트-서버 모델을 사용하였다.

<그림 7>과 <그림 8>은 클라이언트와 서버의 구성도이다. 두개의 구성도는 기본적으로 UDP 소켓을 이용하여 실제 데이타의 전송을 담당 송수신 모듈이 있는 데이타그램 블록과 API를 통해 프로그래머가 응용프로그램을 개발할 수 있는 응용프로그램 블록, 그리고 WTP 프로토콜을 제공하는 WTP 블록으로 나누어져 있다. SDT는 <그림 5>를 통해 WTP 블록 내에 있는 SDL 시스템과 환경함수를 만들어 준다. SDL 시스템은 하나의 독립된 프로세서로 동작하게 되는데, 프로토타입 내의 모듈로 넣기 위해서 스레드로 만들고, 응용프로그램 및 송수신과의 메시지 교환을 위해 환경함수에서는 시스템에서 제공하는 IPC(inter process communication)을 이용하게 된다. 구성도에 따른 전체적인 시스템은 라이버러리의 형태로 프로토타입을 사용하여 프로그램을 만드는 사용자에게 제공된다.

<표 1>은 프로토타입이 제공할 API를 정의하고 있다. API의 정의는 프로그래머가 어떤 형태로 통신 프로그램을 작성해야 할것인지에 대한 기본적인 모델을 제시하게 된다. 본 논문에서는 클라이언트-서버 모델의 통신 프로그램을 작성하는데 용이하도록 소켓 개념을 도입하였다. 가상 소켓을 위한 구조체인 struct WTPsocket을 정의하였으며, clientWTP() 함수와 serverWTP() 함수는 이 구조체를 반환한다. 소켓 프로그램과 같이 모든 메시지 처리 함수는 이 소켓 구조체를 참조하게 되어 있다. TCP/UDP 소켓 프로그램은 read()와 write()함수만으로 통신이 가능하지만, WTP 프로토콜은 사용자에게 다양한 서비스를 제공하기 때문에 API 함수의 정의도 다양하다.

5. 결론 및 향후과제

본 논문에서는 무선 인터넷 서비스를 위한 무선 응용 프로토콜 스택 중 트랜잭션 처리를 담당하는 WTP를 SDL을 이용하여 설계하였고, SDT를 이용하여 클라이언트-서버 프로토타입을 구성하였다. 또한 이를 실제로 이용하기 위한 API를 정의하고, WTP 스택을 라이브러리로 만들었다.

TTCN 관련 지원 도구인 ITEX를 이용한 프로토콜의 검증 및 완성된 코드의 적합성 검정을 위한 자동화된 테스트 케이스의 생성에 관한 연구와 WSP(Wireless Session Protocol)와 WTP 스택을 통합한 SDL 시스템의 설계 및 프로토타입의 구현은 다음 연구 과제로 계속 추진할 예정이다.

6. 참고 문헌

[1] Shari Lawrence Pfleeger, Software Engineering Theory and Practice, Prentice Hall, 1997.

[2] 한국전자통신연구원, 정보통신 프로토콜공학, 1994년 8월.

[3] 김경민, 남영호, 박재홍, 노철우, 이병길, 한운영, "SDT를 이용한 WLL 프로토콜 검증," 한국정보과학회, 1999년도 춘계 학술발표논문집, Vol. 26, No. 1, 1999년 5월.

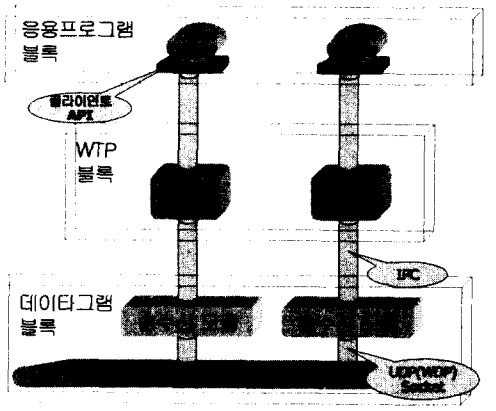
[4] Wireless Application Protocol Architecture Specification, WAP Forum, Nov. 8. 1999. URL: http://www.wapforum.org/

[5] Wireless Transaction Protocol Specification, WAP Forum, Nov. 8. 1999. URL: http://www.wapforum.org/

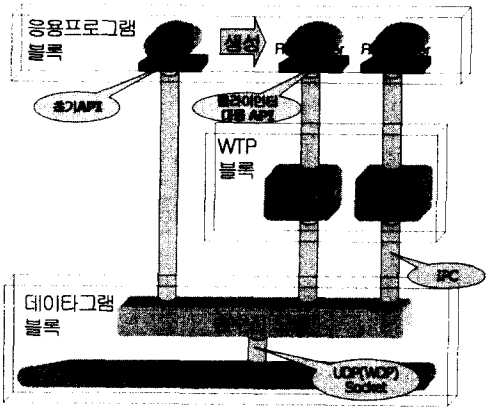
[6] 김기조, 최윤석, 최은정, 임경식, "무선 응용 프로토콜 기술," 정보처리학회지, 한국정보처리학회, 2000년 5월.

[7] 이해동, 최윤석, 임경식, "SDL을 이용한 무선 트랜잭션 프로토콜의 설계 및 구현," 제10회 통신 정보 합동 학술대회 (JCCI2000) 발표논문집, 2000년. 5월.

[8] ITU_T Recommendation Z.100: Specification and Description Language SDL



<그림 7> 클라이언트 시스템 구성도



<그림 8> 서버 시스템 구성도

```
[클라이언트 API]
   WTPsocket clientWTP(char * clientIP,
          u_short clientPort, char * serverIP,
          u_short serverPort);
   void clientListen(WTPsocket sd);

[서버 API]
   WTPsocket serverWTP(char * serverIP,
          u_short serverPort);
   WTPsocket serverAccept(WTPsocket server_sd);

[공용 API]
   int invoke(WTPsocket sd, int handle, int class,
          int acktype, char * pData, int size);
   int read_invoke(WTPsocket sd, int *handle, int *class,
          int *acktype, char * pData, int size, int *msgType);
   int result(WTPsocket sd, int handle,
          char *pDAta, int size);
   int getConform(WTPsocket sd, int handle);
   int read_result(WTPsocket sd, int handle, char *pData,
          int size, int *msgType);
   void wtp_ack(WTPsocket sd, int handle);
   void wtp_abort(WTPsocket sd, int handle);
   void closeWTP(WTPsocket sd);
```

<표 1> WTP 서비스 제공을 위한 API 정의