무선 인터넷 서비스를 위한 트랜잭션 프로토콜의 구현과 성능평가

(Implementation and Performance Evaluation of Transaction Protocol for Wireless Internet Services)

최 윤 석 † 임 경 식 ††
(Yoonsuk Choi) (Kyungshik Lim)

요 약 본 연구에서는 무선 인터넷 서비스를 위한 전송계층 프로토콜인 TCP, T/TCP와 WTP의 특징을 비교·분석한다. 우선 무선구간에 가장 적합한 WTP를 제한된 자원을 가진 무선 단말기상에 탑재하기 용이하도록 코루틴 모델을 기반으로 하나의 쓰레드로 구현하고 이를 Nokia, Kannel 그리고, WinWAP의 기존 구현물과 상호 동작시켜 호환성을 검사한다. 그리고, 무선환경에서의 패킷손실을 잘 표현할 수 있는 길버트(Gilbert) 모델을 기반으로 구현물의 트랜잭션 성공률(throughput) 및 수행시간(system response time)을 측정하여 기존의 TCP, T/TCP와 비교한다. 그 결과, WTP는 트랜잭션 성공률과 수행시간에 있어서 기존의 프로토콜에 비해 높은 성능을 보였다. 특히, 연속적인 에러가 발생하며 패킷손실률이 비교적 높을 때, WTP는 T/TCP와 TCP에 비해 매우 높은 트랜잭션 성공률을 나타냈으며, 10배 이상 빠른 수행 시간을 보였다. 이는 WTP가 다른 프로토콜에 비해 적은 개수의 패킷으로 하나의 트랜잭션을 수행하고 패킷 손실로 인한 타임아웃 발생 시, 타이머 값을 exponential backoff를 적용하지 않고 일정한 값을 유지하기 때문이다. 또한, 무선환경에 최적화된 WTP의 재전송 횟수를 결정하기 위한 실험을 통해, 가장 적절한 재전송 횟수가 5~6회임을 알 수 있었다.

키워드 : 코루틴 모델, 프로토콜 구현, 성능평가

Abstract In this paper, we design and implement Wireless Transaction Protocol(WTP) and evaluate it for wireless transaction processing in mobile computing environments. The design and implementation of WTP are based on the coroutine model that might be suitable for light-weight portable devices. We test the compatibility between our product and the other products such as Nokia, Kannel and WinWAP. For the evaluation of WTP, we use an Internet simulator that can arbitrary generate random wireless errors based on the Gilbert model. In our experiment, the performance of WTP is measured and compared to those of Transmission Control Protocol(TCP) and TCP for Transactions. The experiment shows that WTP outperforms the other two protocols for wireless transaction processing in terms of throughput and delay. Especially, WTP shows much higher performance in case of high error rate and high probability of burst errors. This comes from the fact that WTP uses a small number of packets to process a transaction compared to the other two protocols and introduces a fixed time interval for retransmission instead of the exponential backoff algorithm. The experiment also shows that the WTP performance is optimized when the retransmission counter is set to 5 or 6 in case of high burst error rate.

Key words : TCP, T/TCP, WTP, WIM

† 비 회 원 : 경북대학교 컴퓨터과학과
yoonsuk@ccmc.knu.ac.kr
†† 종신회원 : 경북대학교 컴퓨터과학과 교수
kslim@knu.ac.kr
논문접수 : 2001년 12월 24일
심사완료 : 2002년 5월 2일

1. 서 론

인터넷 서비스를 위한 전송계층 프로토콜을 개발하기 위해서는 기존의 유선망과는 다른 무선 인터넷 환경의 여러 가지 제약조건을 고려해야 한다. 첫째, 무선 링크는 매우 빈약하고 높은 패킷 손실률을 가진다. 따라서,

응용계층에게 만족할 만한 성능을 제공하기 위해서는 전송계층에서 적절한 에러 복구 방법을 지원해야 한다. 둘째, 무선 링크는 제한된 대역폭과 오랜 지연시간을 가지며, 망 자원이 가변적으로 변화한다. 그러므로, 전송계층에서는 이러한 변화를 고려한 손실 복구 방법을 제공해야 한다[1].

현재 Transmission Control Protocol(TCP), TCP for Transacion(T/TCP)과 Wireless Application Protocol (WAP) 포럼에서 제안한 Wireless Transaction Protocol (WTP)가 무선 전송계층 프로토콜로 연구되고 있다[2-5]. 본 논문에서는 이러한 전송계층 프로토콜을 비교·분석하고 그 중 현재 가장 널리 사용되며, 무선구간에 가장 적합한 WTP를 분석하고 이를 구현하기 위한 모델을 제시한다. 간헐적인 패킷 손실 및 연속적인 패킷 손실이 일어나는 다양한 무선환경에서 시뮬레이션을 통해 WTP 구현물의 성능 및 트랜잭션 수행시간을 측정하여 기존의 TCP, T/TCP와 비교한다. 또한, 트랜잭션 수행시간과 성능을 고려해서 무선환경에 가장 적합한 WTP 재전송 횟수를 제안한다.

본 논문의 2장에서는 연구 배경을 살펴보고 3장에서는 WTP의 구현 모델 및 요소 기술에 대해 기술한다. 4장에서는 다양한 시뮬레이션 환경에서 성능 분석을 하고 마지막으로 5장에서는 결론 및 향후 과제를 제시한다.

2. 연구배경

무선 인터넷 서비스를 제공하기 위해서는 낮은 대역폭, 데이타 전송 지연과 불안정한 접속 등 무선망의 다양한 문제점과 브라우징을 위한 트랜잭션(요구/응답) 위주의 무선 인터넷 서비스를 고려해서 전송계층 프로토콜을 설계해야 한다. 이를 위해 현재 연구중인 전송계층 프로토콜 중 TCP, T/TCP와 WTP를 무선환경을 기반으로 비교·분석한다.

첫째, TCP 프로토콜은 유선망에서 가장 널리 사용되는 데이타 전송 프로토콜로서 망 폭주에 의한 종단간 지연 및 패킷손실에 적응할 수 있는 메카니즘을 사용하여 종단간의 신뢰성있는 전송을 보장한다. TCP는 기본적으로 망 폭주에 의한 패킷 손실이 발생하지 않는 동안 계속적으로 전송률을 증가시킨다. 그러나, 무선구간에서 패킷 손실이 발생할 경우, 이를 망 폭주에 의한 패킷 손실로 간주하고 망 폭주 회피 알고리즘이나 fast retransmission을 구동해서 윈도우 크기를 줄임으로써 종단간 전송 성능을 감소시키게 된다[6]. 둘째, T/TCP 프로토콜은 TCP의 신뢰성 있는 데이타 전송 서비스를 제공하면서 WWW와 같은 트랜잭션 위주의 응용을 지원하기 위해 기존 TCP를 확장했다. 기존의 TCP와 달리, T/TCP는 명시적인 연결 설정 및 종료 과정을 단순화하여 전송되는 세그먼트의 수를 감소시키고 응답 시간을 줄인다. 그러나, T/TCP는 망 폭주 제어나 패킷 손실 복구와 같은 대부분의 TCP 기능을 그대로 사용함으로써 무선환경에는 적합하지 않다[2,3]. 마지막으로, WTP는 데이타그램 서비스(UDP) 상에서 동작하면서 상호적인 브라우징(요구/응답) 응용 소프트웨어가 필요로 하는 서비스를 제공하는 트랜잭션 지향 프로토콜이다. WTP는 제한된 성능을 갖는 클라이언트 환경에서도 쉽게 구현될 수 있고 보안 또는 비보안 무선 데이타그램 네트워크 상에서 효율적으로 동작하도록 설계되었다[5].

표 1 TCP, T/TCP와 WTP의 주요기능 비교

주요기능 \ 프로토콜	TCP	T/TCP	WTP
망 폭주 제어	○	○	×
명시적인 연결(3WHS) 설정 및 종료	○	×	×
트랜잭션 식별자 확인	×	○	○
분할 및 재결합	○	○	○
차별화된 서비스	×	×	○
사용자 확인	×	×	○
트랜잭션 취소	×	×	○

표 1은 TCP, T/TCP와 WTP의 주요기능을 보여준다. T/TCP는 트랜잭션 식별자 확인을 통한명시적인 연결설정 및 종료과정을 제외하면 TCP와 거의 유사하다. 그에 비해, WTP는 하나의 홉으로 이루어진 무선구간에서 패킷 손실을 망 폭주가 아닌 무선링크의 특성상 발생하는 것으로 간주하고 패킷 손실이 일어날 경우, 망 폭주 제어를 수행하지 않고 재전송만 함으로써 성능을 향상시킨다. 또한, T/TCP처럼 트랜잭션 식별자 확인을 통해 명시적인 연결 설정 및 종료를 하지 않음으로써, 트랜잭션 처리를 위해 전송되는 메시지의 수와 지연시간을 줄일 수 있다. 그리고, WTP는 무선 인터넷 서비스를 위해 기존의 TCP와 T/TCP에서 없던 다음과 같은 새로운 기능을 추가했다[5].

- 차별화된 트랜잭션 서비스 제공

WTP는 클래스 0, 1, 2의 세 가지 트랜잭션을 정의하고 있다. 클래스 0 트랜잭션은 응용 계층에게 비신뢰적 푸시 서비스를 지원하기 위해 신뢰성 없는 단방향 데이타그램 서비스를 제공하며, 클래스 1 트랜잭션은 응용

계층에게 신뢰적 푸시 서비스를 지원하기 위해 신뢰성 있는 단방향 데이타그램 서비스를 제공한다. 클래스 2 트랜잭션은 가장 빈번히 사용되는 서비스이며, 신뢰성 있는 양방향 데이타그램 서비스를 제공한다. 이러한 클래스 2 트랜잭션의 동작과정을 살펴보면, 먼저 WTP 요구자는 요구 메시지(Invoke PDU)를 응답자에게 전달하고, 이 메시지를 수신한 응답자는 결과 메시지(Result PDU)를 전송함으로써 묵시적으로 응답한다. 이렇게 함으로써 전송되는 메시지의 수를 줄여 전송 성능을 향상시킬 수 있다.

• 사용자 확인(User Acknowledgement)

사용자가 수신된 메시지에 대하여 응답하기 위하여 사용된다. WTP응답자는 U/P 필드가 설정된 요구 메시지를 받으면 사용자에게 알리고 응답 타이머를 구동시킨 후, 사용자로부터 응답을 기다린다. 만약 사용자가 응답하기 전에 요구 메시지가 재전송 되면, 응답자는 메시지를 무시하고 응답 타이머를 재구동시킨다. 정해진 시간이 지난 후에도 사용자로부터 응답이 없으면 트랜잭션은 취소된다.

• 트랜잭션 취소(Transaction Abort)

트랜잭션 취소는 존재하지 않는 파일의 요청과 같이 사용자 프로그램이 서비스를 제공할 수 없거나 사용자의 강제적 트랜잭션 종료에 의해 발생할 수 있다. 또한, WTP 프로토콜에서 에러가 발생할 경우, 수행중인 트랜잭션은 취소될 수도 있다. 즉, WTP가 수신한 데이타의 오류로 인해 서비스를 거절하거나 요구하는 기능이 내부적으로 구현되어 있지 않을 경우, WTP는 해당 트랜잭션을 취소시킨다.

앞에서 살펴보듯이, WTP는 무선 인터넷 서비스를 위한 가장 적합한 프로토콜이다. 다음에는 이러한 WTP를 설계 및 구현하고 다양한 무선환경에서 시뮬레이션을 함으로써 TCP, T/TCP 프로토콜과의 성능을 비교한다.

3. 무선 트랜잭션 프로토콜의 구현

3.1 WAP 프로토콜 구현 모델(WIM)

3.1.1 코루틴 모델

그림 1은 서버 모델(server model)의 코루틴(coroutine) 구현 방식을 개략적으로 표현한 그림이다. 서버 모델은 프로토콜의 각 계층을 하나의 프로세스로 구현하는 형태인 단일 프로세스 서버 모델(single-process server model)과 다수의 프로세스로 구현하는 형태인 다중 프로세스 서버 모델(multiplrocess server model)로 구분할 수 있다. 서버 모델에서는 프로토콜 계층을 "서버"라는 실행 단위로 구성하며, 각 서버는 단일 프로세스 또는 다중 프로세스로 구성할 수 있다. 각 서버들은 상·하위 입력버퍼 큐에 메시지 입력이 있거나, 타이머와 같은 이벤트가 발생할 경우에 수행된다. 수행이 완료된 서버는 메시지의 입력이나 새로운 이벤트가 발생하기 전까지 수행이 정지된다[7-9].

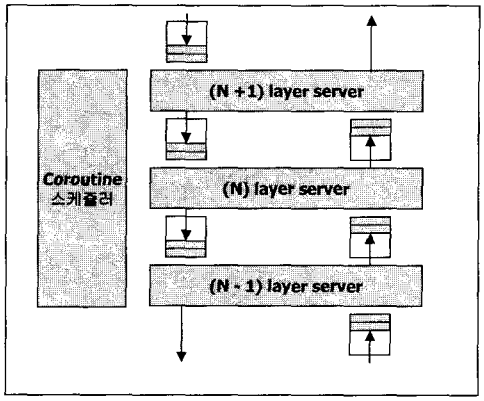

그림 1 서버 모델 - Coroutine 구현 방식

서버 모델을 실제 구현함에 있어서는 세 가지 방식이 있다. 첫째, 프로세스 구현 방식은 서버를 운영체제에서 제공하는 프로세스로 구현한다. 둘째, 쓰레드 구현 방식은 서버를 쓰레드로 구현한다. 마지막으로, 코루틴 방식은 서버가 코루틴이 되며 응용 계층에서 구현된 스케줄러에 의하여 코루틴을 수행하는 모델이다. 또한, 코루틴이 서버가 되므로 운영 체제에서 프로세스나 쓰레드를 지원하지 않는 경우에도 구현이 가능하며, 메시지 전달 방식으로 계층간 통신을 함으로써 모듈화, 은닉화 및 명확한 인터페이스를 제공하므로 프로토콜 공학적 관점에서 이상적인 접근 방법이다.

3.1.2 WAP Implementation Model(WIM) 설계

WIM은 WAP 프로토콜 스택을 구현하기 위한 모델이며, 그림 2는 WIM을 도식화한 것이다. WIM은 코루틴 구현 방식을 기반으로 하며, 크게 엔진 관리 영역과 계층 엔진 영역으로 나뉜다. 엔진 관리 영역에는 메시지 재전송과 같은 작업을 수행하기 위한 타이머 시스템, 프로토콜 계층간 메시지 전달을 위한 버퍼 시스템, 그리고, 계층 엔진의 호출 및 스케줄링의 역할을 담당하는 엔진 관리 시스템이 있다. 각 시스템들의 기능은 API를 통하여 계층 엔진 영역에 제공된다. 계층 엔진 영역은 WAP 프로토콜을 실제로 구현하는 부분이다. 각 계층간의 인터페이스는 프리미티브로 결정되며, 내부 동작은 프로토콜 동작을 기술한 상태 테이블을 기반으로 구현된다. 각 프로토콜 계층에서는 계층간 인터페이스를 정

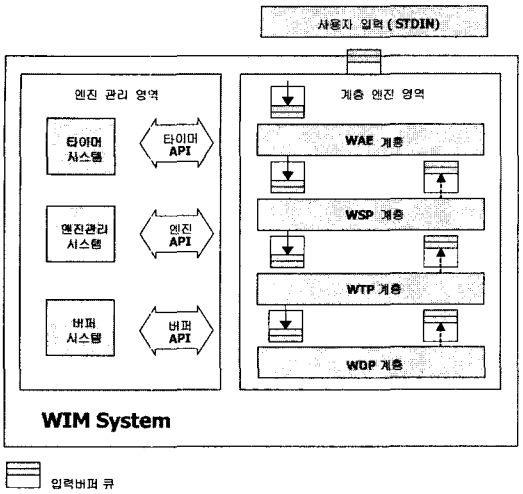

그림 2 WIM 구조

의하기 위하여 프리미티브를 버퍼 시스템에서 정의하는 버퍼 메시지로 구조화하고, 버퍼 I/O 함수를 사용하여 다음에 수행할 계층에 메시지를 전달한다. 각 프로토콜 엔진은 타이머 시스템, 엔진 관리 시스템 및 버퍼 시스템에서 제공하는 API들을 사용하여 구현된다.

표 2는 프로토콜 계층 엔진을 관리하기 위한 엔진 관리 시스템의 주요 모듈이다. 프로세스의 메인은 각 엔진의 버퍼 큐를 관리하며, 큐에 메시지가 들어오면 virtual_select 함수내의 메시지 분석 메커니즘에 의하여 해당하는 엔진을 호출하며, 엔진이 처리를 마치면 제어는 다시 메인함수로 넘어간다. 메인함수는 버퍼 큐에 입력이 없는 동안 블로킹되어 있으며, 입력이 발생하면 블로킹에서 깨어나게 된다. 입력의 발생은 크게 다음의 세 부분에서 일어나게 된다. 첫째, 표준 입력(Standard Input)을 통한 사용자 입력이 발생할 경우, 메인함수는 WAE 엔진을 호출하게 된다.

둘째, 네트워크를 통해 데이타를 수신하는 경우, WDP

표 2 엔진 관리시스템의 주요 모듈

```
while(1){
        layer = virtual_select(msg_queue);
        switch(layer) {
           case WAE: wae_engine(); break;
           case WSP: wsp_engine(); break;
           case WTP: wsp_engine(); break;
           case WDP: wsp_engine(); break;
        }
 }
```

계층에서는 소켓을 통한 입력으로 발생하고 메인함수는 WDP 엔진을 호출하게 된다. 마지막으로, 각 엔진간 또는 타이머에 의한 처리 결과를 다른 엔진의 입력버퍼 큐에 전달하고, 파이프에 다음에 처리할 계층에 대한 정보를 입력한다. 메인함수는 파이프의 입력을 분석하여 해당하는 엔진을 호출한다.

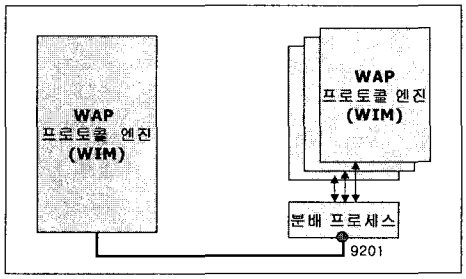

그림 3 WAP 클라이언트 및 서버

그림 3은 WAP 프로토콜 엔진이 클라이언트와 서버 사이에서 상호 동작하는 방법을 나타낸다. WAP 프로토콜 엔진은 단일 프로세스 형태인 WIM으로 구현되며, 클라이언트와 서버에서 동일하게 사용된다. 그러나, 서버는 WAP 프로토콜 엔진 외에 분배 프로세스를 추가적으로 두고있는데, 이 분배 프로세스는 WAP 포럼에서 정의한 관용 포트(Well Known Port)를 열고 클라이언트의 접속을 기다리며, 해당 클라이언트가 접속하면 WAP 프로토콜 엔진을 수행할 프로세스를 생성시킨다. 또한, 분배 프로세스는 클라이언트의 소켓 주소 및 포트 값으로 클라이언트를 구분하여 각 클라이언트에 대응하는 WAP 프로토콜 엔진 프로세스에게 데이타그램을 분기시켜주는 역할을 한다.

3.2 WIM 주요 시스템 및 API

3.2.1 엔진 관리 시스템

엔진 관리 시스템은 엔진 수행의 우선 순위, 스케줄링 알고리즘과 입력 메시지 유무에 따라서 수행될 엔진을 결정하고 수행 권한을 부여한다. 엔진 관리 시스템에 계층 프로토콜 엔진을 등록할 때, 엔진의 우선 순위와 스케줄링 알고리즘이 결정되며, 설정된 우선 순위와 알고리즘에 따라서 엔진 관리 시스템은 각 엔진의 실행 순서를 결정한다. 스케줄링 알고리즘은 라운드 로빈(Round Robin) 방식으로 각 엔진에 대한 우선 순위를 동등하게 주고, 버퍼 큐에 메시지가 있는 엔진을 호출하는 방식으로 동작한다. 엔진 관리 시스템에서 제공하는 API는 표 3, 표 4와 같다.

표 3 WIM_Start API

API	int WIM_Start(WIM_argument wim_arg);		
설 명	WIM 시스템을 시작		
인 자	Parameter WIM_argument ※N-Null은 Null이 아닌 값이어야 하며, Null은 Null 값이 가능한 파라미터를 나타낸다.	Meaning 시스템 구동	Others N-NULL
리턴값	• 0 시스템 시작 실패. • 1 프로토콜의 정상적인 종료.		

표 4 WIM_Stop API

API	void WIM_Stop(int stop_option);		
설 명	WIM 시스템을 종료		
인 자	Parameter stop_option	Meaning 시스템 종료	Others N-NULL
리턴값	• 0 정상 종료. • 1 즉시 종료.		

3.2.2 버퍼 시스템

각 엔진에는 두 개의 입력버퍼 큐가 할당되며, 큐는 메시지의 우선 순위에 따라서 처리된다. 우선 순위가 낮은 메시지는 다른 메시지가 처리되는 동안, 우선 순위가 높아지는 방식으로 구현되어 모든 메시지가 처리되도록 한다. WIM 시스템에서는 시스템 시작 시에 입력버퍼 큐를 결정해 고정된 크기의 큐를 사용하며, 메시지의 타입은 void 형으로 선언하여 저장할 데이타 구조에 독립적이다. 버퍼 시스템은 프로토콜 계층간에 메시지를 전달할 때, 메시지를 일시적으로 저장한다. WIM_msg_ send 함수는 버퍼 시스템에 메시지를 저장하며, 인자에 명시된 메시지를 처리할 계층 엔진이 호출 되도록 파이프에 관련 정보를 입력한다. 엔진 관리 시스템은 이러한 정보를 이용하여, 엔진 스케줄링 방식에 따라 계층 엔진을 호출하여 저장된 메시지가 처리되도록 한다. WIM_ msg_rec 함수는 현재 수행 중인 계층 엔진의 식별자 값과 버퍼 큐 정보를 사용하여 버퍼 시스템으로부터 버퍼 메시지를 출력한다. 아래 표 5, 표 6은 버퍼 시스템이 제공하는 API이다.

표 5 WIM_msg_send API

API	int WIM_msg_send(int source, int target, void *m_pMsg, int priority);		
설 명	버퍼 메시지를 버퍼 시스템에 전달		
인 자	Parameter source target *m_pMsg priority	Meaning 송신계층 수신계층 버퍼메시지포인터 메시지 우선 순위	Others N-NULL N-NULL N-NULL N-NULL
리턴값	• >=0 버퍼 큐에 입력 성공. • -1 버퍼 큐에 입력 실패.		

표 6 WIM_msg_recv API

API	int WIM_msg_recv(int recv_layer, Queue Direction direction, void * m_pMsg);		
설 명	버퍼 시스템으로부터 버퍼 메시지를 출력		
인 자	Parameter recv_layer direction *m_pMsg	Meaning 메시지 출력 계층 버퍼 큐 지정 출력메시지	Others N-NULL N-NULL N-NULL
리턴값	• >=0 버퍼 큐에서 출력 성공. • -1 버퍼 큐에서 출력 실패.		

3.2.3 타이머 시스템

기존의 시스템 타이머는 일정 시간이 지난 후, 타이머 시스템에서 정의한 시그널이 발생하여, 시그널 핸들러로 등록된 함수가 수행되는 방식으로 동작한다. 그러나, WIM에서는 그림 2에서와 같이 계층 엔진이 수행되는 동안, 타이머 시스템에 메시지를 등록하며, 정해진 시간이 지나면 등록된 메시지를 해당 계층 엔진의 입력버퍼 큐에 전달하는 방식으로 동작한다. 이러한 방식은 시스템의 시그널에 의한 시스템 함수들의 예외 동작을 방지해주는 역할을 한다. 타이머가 전달하는 메시지는 기존의 엔진에서 사용되는 WAP 메시지가 사용된다.

그림 4는 타이머의 동작을 도식화한 그림이다. 타이머 테이블은 타임아웃 시 각 계층 엔진으로 전송해야 할 WAP 메시지의 포인터를 저장할 수 있다. 현재의 인덱스는 0.1초 간격으로 타이머 테이블을 한 칸씩 이동하며 해당 타이머 테이블에 프로토콜 계층 엔진으로 보내야 할 메시지가 있는지를 검사한다. 이때, 메시지가 있으면 그 메시지를 해당하는 엔진으로 보내게 된다. 예를 들어, WTP 계층에서 0.4초 후 재전송을 하기 위해서 다음과 같이 타이머를 구동시키면, WIM_timer_start(0.4초 후 타임아웃, 반환될 WAP 메시지);, 그림 4와 같이 현재의 인덱스에서 4칸 다음의 위치에 WAP 메시지가 달릴 것이고 0.4초 후에 타임아웃이 발생하여 해당 메시지를 WTP 입력버퍼 큐로 넘겨준다. 타이머 시스템 API는 표 7, 표 8과 같다.

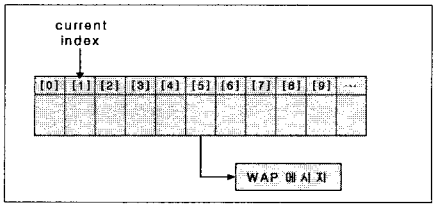

그림 4 타이머의 동작 원리

표 7 WIM_timer_start API

API	int WIM_timer_start(int tv_sec, int tv_usec, int target, int direction, void * m_pMsg);		
설 명	타이머 시스템에 서비스를 등록		
인 자	Parameter	Meaning	Others
	tv_sec,tv_usec	타이머 만기 시간	N-NULL
	target	수신 계층	N-NULL
	direction	버퍼 큐 지정	N NULL
	*m_pMsg	전달될 메시지	N-NULL
리턴값	· >=0 타이머 시작, 타이머의 식별자 값. · -1 타이머 시작 실패.		

표 8 WIM_timer_stop API

API	int WIM_timer_stop(int m_tmIndex);		
설 명	타이머 시스템에 등록된 타이머를 해제		
인자	Parameter	Meaning	Others
	m_tmIndex	타이머식별자	N-NULL
리턴값	· >=0 타이머 해제 성공. · -1 타이머 해제 실패.		

3.3 WTP 프로토콜 엔진 구현

3.3.1 WTP 엔진 구성도

그림 5는 WTP 프로토콜 엔진 인터페이스를 도식적으로 표현한 것이다. 다른 계층 엔진들도 이와 유사한 형태로 구현된다. WTP 엔진은 상·하위 계층으로부터 전달되는 메시지를 저장하기 위하여 상·하위 입력버퍼 큐를 가진다. wtp_engine 함수는 WAPQueueDirection 인자를 통해서 메시지가 입력된 큐를 식별한다. 또한, 이 값을 이용하여 엔진 내부에서는 메시지를 전달할 큐를 선택할 수 있다. wtp_recvmsg_up 함수는 상위 입력버퍼 큐로부터 메시지를 가져오고 wtp_recvmsg_down 함수는 하위 입력버퍼 큐로부터 메시지를 가져온다. 또한, 상·하위 계층으로 메시지를 전달할 경우, wtp_sendmsg_up, wtp_sendmsg_down 함수를 사용한다. wtp_loopback 함수는 프로토콜 수행 중에 현재 계층으로 메시지를 다시 전달한다. 위와 같은 함수들을 이용하여 WTP 계층의 상·하위 계층으로 메시지가 전달되며, 인접 계층에서도 이러한 함수들을 이용하여 WTP 계층에 메시지를 전달한다. 이렇게 전달된 메시지를 기반으로 WTP의 상태 테이블에 정의된 동작 메커니즘에 따라서 프로토콜이 동작하게 된다.

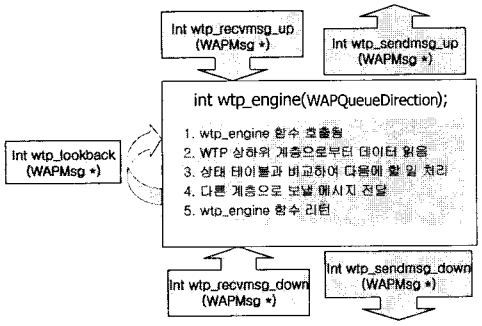

그림 5 WTP 엔진 구성도

3.3.2 WTP 동작 구현

가. 메시지 버퍼

인접한 계층간에 메시지를 전달하기 위한 인터페이스 역할을 하는 프리미티브는 실제 구현 부분에서는 WAP 메시지로 구조체화되어 해당 계층으로 전달된다. 표 9는 각 프로토콜 계층간 메시지를 전달하기 위한 버퍼 시스템의 WAP 메시지를 나타낸다. service type은 WAP이 제공하는 서비스 종류를 의미하며, 서버의 입력포트에 의해 결정된다. event type은 프리미티브의 네 가지 이벤트 형식인 Request, Indication, Response와 Confirm의 정보를 포함하며, 일반적으로 각 계층 엔진은 서비스 프리미티브와 이벤트 종류를 함께 정의한다. timer_event는 프로토콜 계층에서 타이머의 종류를 등록할 때 이용하며, para_type은 계층사이에서 메시지를 주고받을 때 파라미터 종류를 정의한다. 실제 파라미터는 _Para라는 union으로 정의된 구조체에 저장하며, pdu는 각 계층에서 생성한 프로토콜 데이타 유닛을 저장한다.

표 9 계층간 메시지 전달을 위한 구조체

```
typedef struct _WAPMsg{
        short service_type;
        short event_type;
        short timer_event;
        short para_type;
        union _Para para;
        WAPPDU pdu;
}WAPMsg;
```

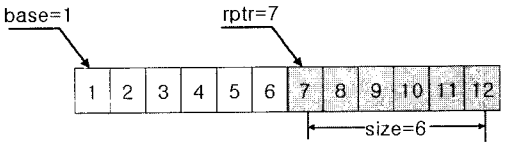

그림 6 프로토콜 데이타 유닛 구성

프로토콜 데이타 유닛은 클라이언트와 서버 사이에서 프토토콜이 상호 동작하기 위한 정보들을 형식화한 데이타 단위이다. 그림 6은 프로토콜 데이타 유닛을 실제 메시지로 만드는 과정을 도식화한 것이다. 먼저, 최상위

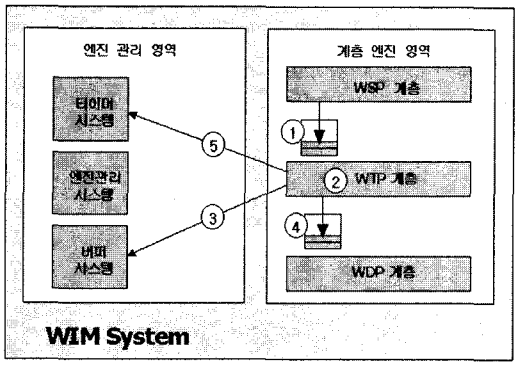

그림 7 TR-Invoke.req 이벤트 처리과정

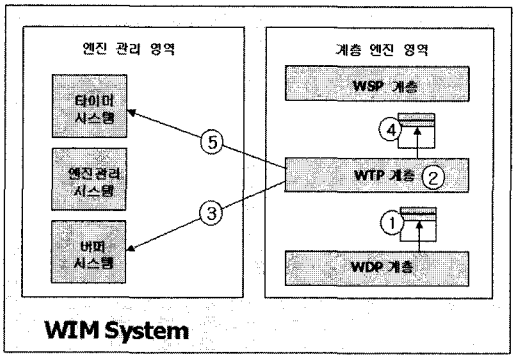

그림 8 TR-Invoke.ind 이벤트 처리과정

계층은 각 계층의 프로토콜 데이타 유닛 및 사용자 데이타의 크기를 고려하려 메모리를 할당한다. WTP 상위계층은 보내고자 하는 데이타를 메모리의 끝에서부터 데이타를 저장하고 WTP는 앞에 남은 부분에 WTP의 헤더를 붙인 후에 이를 WDP를 이용하여 전송하게 된다. 이렇게 함으로써, 각 계층은 별도의 메모리를 할당할 필요없이 초기에 생성된 버퍼에 필요한 자료를 복사함으로써 메모리 할당 및 복사의 횟수를 줄여 프로토콜 데이타 유닛의 처리 속도를 향상시킬 수 있다.

나. WTP 엔진 동작 메카니즘

그림 7은 새로운 트랜잭션을 초기화하기 위한 TR-Invoke.req 이벤트가 발생할 경우, WTP 계층에서 동작하는 과정을 WIM 시스템을 기반으로 도식화한 그림이다. 먼저, WTP는 버퍼 시스템을 통해 자신의 입력버퍼 큐로부터 TR-Invoke.req 이벤트의 발생을 확인한다(①). 입력버퍼 큐로부터 WAP 메시지를 읽어온 후, 새로운 트랜잭션을 초기화하기 위한 작업을 수행한다. 즉, 새로운 트랜잭션 식별자를 생성하고 WTP 컨트롤 블록을 초기화한다. 또한, 상위계층으로부터의 WAP 메시지를 기반으로 WDP 계층으로 전달할 파라미터를 메시지화하고 상위계층의 데이타를 기반으로 WTP 프로토콜 유닛 데이타를 생성한다(②). 그리고, 이를 버퍼 시스템을 이용해서 WDP 계층으로 전달한다(③,④). 마지막으로, 전송한 데이타에 대한 신뢰성을 보장하기 위해 타이머 시스템을 통해 재전송 타이머를 구동시킨다(⑤). WTP 상위계층으로부터의 다른 이벤트 처리과정도 이와 유사하게 이루어진다.

그림 8은 새로운 트랜잭션의 요구에 대한 응답을 위한 TR-Invoke.ind 이벤트가 발생할 경우, WTP 계층에서 동작하는 과정을 WIM 시스템을 기반으로 도식화한 그림이다. 먼저, WTP는 버퍼 시스템을 통해 자신의 입력버퍼 큐로부터 TR-Invoke.ind 이벤트의 발생을 확인한다(①). WTP 계층에서는 메시지 및 트랜잭션 식별자에 대한 유효화 검사를 한후, WTP 컨트롤 블록을 초기화하고 하위계층으로부터의 WAP 메시지를 기반으로 WSP 계층으로 전달할 파라미터를 메시지화한다(②). 그리고, 이 메시지를 버퍼 시스템을 이용해서 WSP 계층으로 전달한다(③,④). 마지막으로, 사용자의 응답을 기다리기 위해 타이머 시스템을 통해 사용자 응답 타이머를 구동시킨다(⑤). WTP 하위계층으로부터의 다른 이벤트 처리과정도 이와 유사하게 이루어진다.

3.4 WTP API 구현

응용 프로그래밍 인터페이스(Application Programming Interface)는 복잡한 프리미티브 구조체에 직접 접근하지 않고, 응용 계층에 편리한 개발 환경을 제공하기 위하여 사용된다. WTP 응용 함수는 서비스 프리미티브의 특성과 계층간에 필요한 정보를 고려하고 함수의 이름은 프리미티브의 특성을 잘 나타낼 수 있게 설계하였다. 표 10은 각각 WTP 응용 프로그래밍 인터페이스 함수를 나타낸다.

표 10 WTP 응용 프로그래밍 인터페이스 함수

서비스 프리미티브	WTP 응용 함수	설명
	trMessage_read	WTP 엔진으로부터의 메시지를 분석하고 발생한 이벤트 종류를 반환
TR_Invoke.req	trInvoke_send	새로운 트랜잭션 초기화
TR_Invoke.ind	trInvoke_read	요구 메시지의 수신
TR_Result.req	trResult_send	결과 메시지의 송신
TR_Result.ind	trResult_read	결과 메시지의 수신
TR_Invoke.res TR_Result.res	trAck_send	수신한 메시지에 대해 응답 메시지의 송신
TR_Invoke.cnf TR_Result.cnf	trAck_read	응답 메시지의 수신
TR_Abort.req	trAbort_send	수행중인 트랜잭션의 취소
TR_Abort.ind	trAbort_read	트랜잭션 취소 메시지의 수신

3.5 WTP 호환성 검사

생성된 WTP 구현물이 프로토콜 표준을 따르는지 확인하기 위하여 기존 구현물과의 호환성을 검사하였다. 클라이언트의 호환성 검사를 위해 Nokia, Kannel의 게이트웨이와 상호동작 시키고 서버의 경우에는 Nokia, WinWAP 브라우저를 사용한다. 호환성 검사 항목은 WTP 스펙에서 제공하는 Protocol Implementation Conformance Statement(PICS)를 기준으로 한다. 표 11은 호환성 검사 결과를 나타내며, WTP의 기능을 기준으로 다른 구현물과의 호환성 여부를 보여준다.

표 11 클라이언트 및 서버 PICS (M: Mandatory, O: Optional)

WTP Function	Req.	Imp.	Nokia	Kannel	WinWAP
Transaction Class 0	M	O	O	O	O
Transaction Class 1	M	O	O	O	O
Transaction Class 2	M	O	O	O	O
User acknowledgement	M	O	O	O	O
Concatenation	O	×	·	·	·
Separation	M	O	O	O	O
Retransmission until acknowledgement	M	O	O	O	O
Transaction abort	M	O	O	O	O
Asynchronous transaction	O	O	O	O	O
Transaction identifier verification	M	O	O	O	O

4. 무선 트랜잭션 프로토콜의 성능평가

4.1 시뮬레이션 환경

4.1.1 시뮬레이션을 위한 테스트 베드

그림 9는 시뮬레이션을 위한 테스트베드 구축환경을 나타낸다. WAP 클라이언트와 서버는 리눅스 운영체제에서, 프로그래밍 언어는 C 언어를, 컴파일러는 공개용 컴파일러인 GNU C 컴파일러를 사용하였다. RADCOM 사의 인터넷 시뮬레이터는 펜티엄 Ⅲ 450MHz 급의 윈도우즈 NT 플랫폼 컴퓨터에 설치되어, 무선 링크를 시뮬레이션 가능하게 한다. 인터넷 시뮬레이터를 이용해서 무선 구간에 맞는 패킷 손실률, 링크 성능, 패킷의 순서, 지연 및 지터를 설정할 수 있다. 본 실험에서는 20개의 클라이언트와 1개의 서버 시스템을 가정하며, 각 클라이언트는 실제 하나의 프로세스로 수행된다. 실험을 위한 에러 모델은 길버트 모델을 기반으로 하고, 링크의 성능은 144kbps(IS-95C)로 설정하며, RTT는 200ms로 한다. 하나의 홉을 가진 무선 링크를 고려하여 패킷의 순서 및 지터는 무시한다. 이러한 환경하에서, 연속적인 및 간헐적인 패킷손실이 일어날 때, FreeBSD상의 TCP, T/TCP와 구현된 WTP의 성능 및 트랜잭션 평균 수행시간을 측정한다.

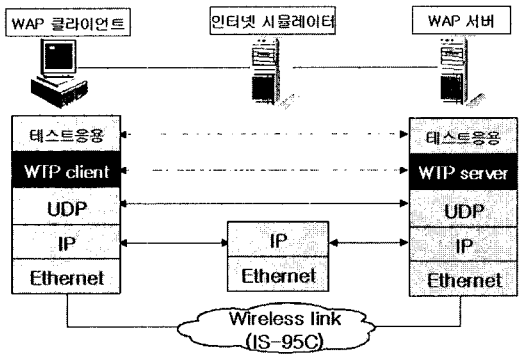

그림 9 시뮬레이션을 위한 테스트베드

4.1.2 에러 모델

실험을 위한 에러 모델은 무선환경의 간헐적인 패킷 손실과 연속적인 패킷 손실 모두를 잘 표현할 수 있으며, 비교적 정확한 확률 모델로 평가받는 길버트 모델을 사용한다[10,11]. 그림 10에서와 같이, 길버트 모델에서는 패킷의 손실 여부를 2-state 마코프 체인(Markov chain)으로 설명한다. 임의의 패킷이 무사히 수신자 측에 도달하는 경우를 하나의 스테이트(state-0)으로 보고, 반대로 패킷이 손실되는 경우는 다른 스테이트(state-1)로 본다. 이때, 패킷 손실 특성은 각 스테이트 간의 전이 확률 p와 q로 표현할 수 있다. n번째 패킷이 어느 스테이트에 있는지를 표현하는 확률 변수를 X_n이라 하면, $X_n=0$은 n번째 패킷이 수신자 측에 도착한 경우이고, $X_n=1$은 n번째 패킷이 전송 혹은 수신 과정에서 손실된 경우를 의미한다. 전이 확률 p, q는 확률 변수 X_n을 이용하여 다음과 같이 표현할 수 있다.

$$p = \Pr[X_{n+1}=1 | X_n=0],$$

$$q = \Pr[X_{n+1}=0 | X_n=1]$$

연속된 패킷들이 같은 스테이트를 지속적으로 유지할 확률은, 전이 확률 p, q에 대한 기하 분포(geometric

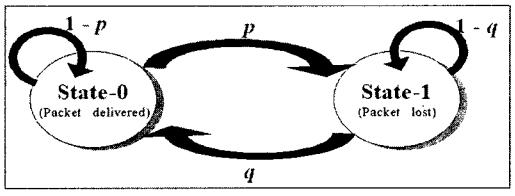

그림 10 길버트 모델

distribution)를 보인다[10]. 따라서, 임의의 패킷이 state-0 또는 state-1에 있을 확률은 기하 분포의 특성에 따라, 다음과 같이 계산된다.

$$\Pr[X_n=0]=\frac{q}{p+q} \quad (1)$$

$$\Pr[X_n=1]=\frac{p}{p+q} \quad (2)$$

무선 구간을 고려해서 패킷 손실률이 50%이하, 연속적으로 손실되는 패킷의 평균 개수가 2~10개라고 가정한다면, (2)의 식으로부터 무선구간에서의 p와 q를 구할 수 있다. q 값은 길버트 모델에서 state-1의 평균 거주 시간(Mean Residence Time)을 이용하여 구할 수 있다. 평균 거주 시간(MRT)이란 일단 손실상태(state-1)에 들어갔을 때, 평균적으로 머무는 시간을 의미하고, 이 값을 확률 계산식으로 유도하면, $\frac{1}{q}$이 된다[10-12]. 연속적으로 손실되는 패킷의 평균 개수가 2~10개이므로, $2\le\frac{1}{q}\le 10$ 이고, 따라서 q 값은 $0.1\le q\le 0.50$의 영역을 가지게 된다. 본 실험에서는 q값이 0.1, 0.3과 0.5일 때 패킷 손실률(Frame Error Rate)이 1%~50% 구간에서 프로토콜의 성능을 측정한다.

4.1.3 트래픽 생성 모델

표 12는 실험을 위한 트래픽 생성 모델을 보여준다. 트래픽 생성 모델은 크게 클라이언트와 서버로 나뉘며, 각각 사용자 행동과 생성되는 트래픽을 양을 나타낸다. 사용자의 행동은 on-off 모델을 따르며, 트랜잭션 사이의 간격은 파레트 확률분포(pareto distribution)를 따르며, 세션사이의 간격은 지수분포(exponential distribution)를 세션 수행시간은 파레트 확률분포를 따른다[13]. 사용자의 요구 메시지와 결과 메시지는 실제 WAP 트래픽의 분석 결과를 기반으로 각각 유니폼 확률분포(uniform distribution)와 파레트 확률분포를 따른다[13,14].

표 12 트래픽 생성 모델

Process	Parameters	Distribution
Request size	min = 36bytes max = 80bytes	uniform
Result size	shape = 1.2 scale = 101bytes max = 1500bytes	truncated pareto
Transaction inter arrival time	shape = 1.5 scale = 17sec max = 46800sec	truncated pareto
Session inter arrival time	lambda = 0.011 (mean = 90sec)	exponential
Session duration time	shape = 1.5 scale = 260sec max = 900sec	truncated pareto

4.2 결과 분석

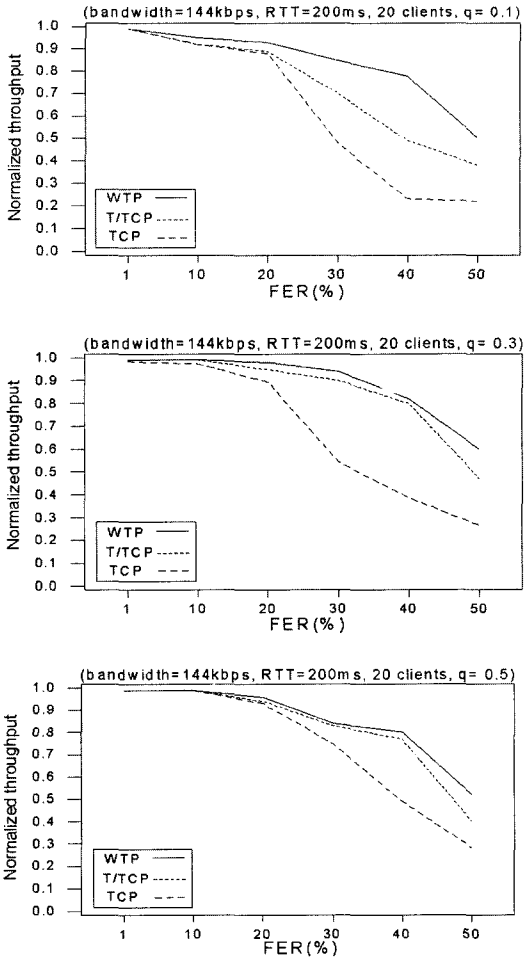

그림 11 TCP, T/TCP와 WTP의 성능 비교

[정의]

- 발생한 트랜잭션의 개수 : N_a
- 성공한 트랜잭션의 개수 : N_s
- Normalized throughput : N_s/N_a(트랜잭션 성공률)

그림 11은 현재 무선망인 IS-95C 망으로 가정해 대역폭을 144kbps로 설정하고, RTT가 200ms일 때, TCP, T/TCP와 WTP의 성능을 비교한 그래프이다. q 값이 각각 0.1, 0.3, 05 일 때, 패킷 손실률이 변함에 따라 Normalized throughput을 측정한다. 즉, 연속적인 패킷 손실 및 간헐적인 패킷손실이 발생하는 다양한 환경에서 트랜잭션 성공률을 측정하였다. 모든 경우, 패킷 손실률이 증가함에 따라 트랜잭션 성공률이 감소하며,

WTP는 TCP와 T/TCP에 비해 높은 성공률을 보인다. 특히, q값이 0.1일 때 즉, 연속적인 에러가 발생하는 경우, WTP는 TCP와 T/TCP에 비해 월등히 높은 성공률을 보인다. 또한, T/TCP도 TCP에 비해 높은 성공률을 보인다. 이는 하나의 트랜잭션을 수행하는데 필요한 패킷의 개수와 깊은 관련이 있다. 그림 12와 같이 하나의 트랜잭션을 수행하는데 TCP, T/TCP와 WTP는 각각 9, 6, 3개의 패킷을 필요하다. 그러므로, WTP는 에러률이 높은 무선구간에서 다른 프로토콜에 비해 높은 트랜잭션 성공률을 보인다. 그래프에서 보듯이, WTP는 패킷 손실의 연속성에 상관없이 높은 트랜잭션 성공률을 보이며, 연속적인 패킷 손실이 발생할 경우도 간헐적인 패킷 손실이 발생하는 경우와 같이 높은 성공률을 나타낸다. 이는 연속적인 패킷 손실이 많이 발생하는 무선환경에 적합하다는 것을 의미한다.

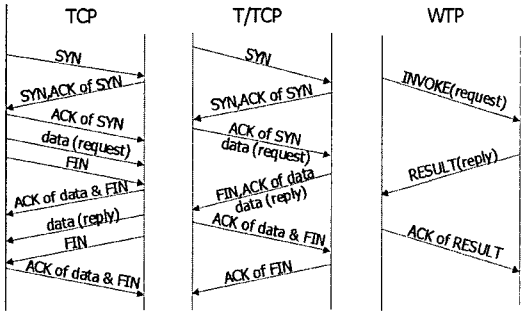

그림 12 TCP, T/TCP와 WTP의 트랜잭션 수행비교

그림 13은 앞의 시뮬레이션과 같은 환경에서 TCP, T/TCP와 WTP의 시스템 응답시간을 비교한 그래프이다. 이때, 시스템 응답시간은 트랜잭션 평균 수행시간을 나타내며, 트랜잭션이 성공하거나 실패하는데 걸리는 시간을 의미한다. TCP와 T/TCP는 패킷 손실률이 증가함에 따라 응답시간이 증가하며, 특히 트랜잭션 성공률이 급격히 감소하는 구간인 패킷 손실률이 20~40%에서 응답시간이 급격히 증가한다. 이에 비해, WTP는 패킷 손실률이 50%인 구간에서 다소 증가하지만 대체적으로 낮은 응답시간을 보인다. 이는 패킷손실로 인한 재전송 시 재전송 타이머 값의 설정으로 인해 발생한다. TCP와 T/TCP의 경우, 패킷손실로 인한 재전송이 발생할 경우 exponential backoff를 적용하지만 WTP는 RTT를 기반으로 일정한 값을 유지한다.

그림 14는 q값이 0.3일 때, WTP의 재전송 횟수에 따라 트랜잭션 성공률과 시스템 응답시간을 비교한 그래

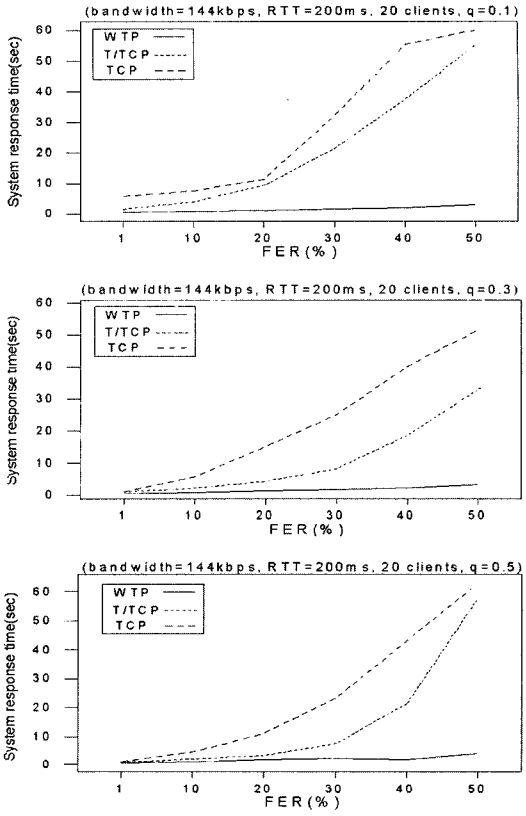

그림 13 TCP,T/TCP와 WTP 수행 시간 비교

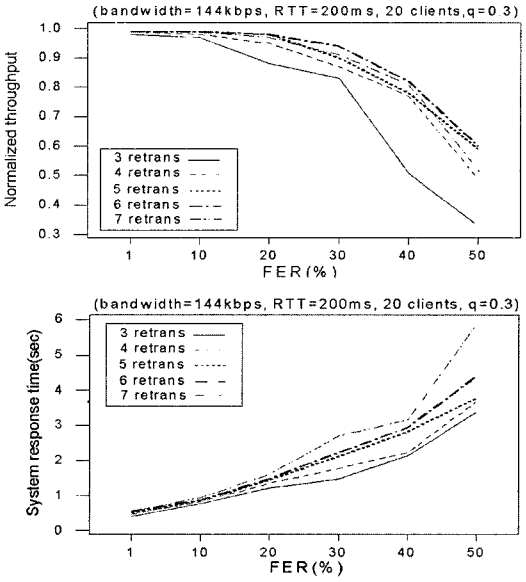

그림 14 재전송 횟수에 따른 성능 및 수행시간

프이다. 대체적으로, 재전송 횟수가 증가함에 따라 높은 트랜잭션 성공률과 응답시간을 보여준다. 그러나 재전송 횟수가 7번일 경우, 응답시간의 급격한 증가로 인해 무선환경에 사용하기에는 적당하지 않다. 따라서, 트랜잭션 성공률과 응답시간을 고려해서 가장 적절한 재전송 횟수는 5~6번이다.

5. 결 론

현재, 무선 인터넷 서비스에 대한 요구가 급증하고 있으며, 이를 수용하기 위해서 낮은 대역폭, 데이타 전송 지연과 불안정한 접속 등 다양한 문제점이 있는 무선환경에 적합한 전송계층 프로토콜이 필요하다. 따라서, 본 연구에서는 무선 전송계층 프로토콜로 연구중인 TCP, T/TCP와 WTP의 특징과 장·단점을 분석했으며, 그 중 무선환경에 가장 적합한 무선 트랜잭션 프로토콜을 구현했다. 무선 트랜잭션 프로토콜은 컴퓨팅 환경이 열악한 무선 단말기상에 탑재하기 용이하도록 코루틴 모델을 기반으로 하나의 쓰레드로 구현하였다. 이를 위해, 코루틴 스케줄러에 해당하는 엔진 관리 시스템과 계층 프로토콜이 동작하는데 필요한 타이머 및 버퍼 시스템으로 엔진 관리 영역을 구성하였고 계층 엔진 영역에서 실제 WTP를 구현하였다. 이렇게 구현된 WTP를 다양한 무선환경에서 시뮬레이션함으로써 기존의 프로토콜과의 성능을 비교·분석하였다. 패킷 손실률이 낮은 유선구간에서는 각 프로토콜의 성능은 유사했으나 패킷 손실률이 높은 무선구간에서는 WTP가 TCP나 T/TCP에 비해 높은 성능을 보였다. 특히, 연속적인 패킷 손실이 발생할 경우, TCP, T/TCP는 급격히 성능이 감소하는데 비해, WTP는 간헐적인 패킷 손실이 발생할 때와 같이 높은 성능을 보였다. 또한, WTP의 재전송 횟수에 따른 성능과 시스템 응답시간을 측정하였다. 이 실험을 통해, 무선환경에서 WTP의 가장 적절한 재전송 횟수가 5~6회임을 확인할 수 있었다.

본 실험에서는 현재 무선망인 IS-95C를 기반으로 실제 WAP 트래픽에 근거한 작은 크기의 트랜잭션에 대해 성능 및 시스템 응답시간을 측정하였다. 향후, IMT-2000과 같은 차세대 무선망을 고려해서 스트리밍 서비스와 같은 사용자 요구 데이타가 큰 트랜잭션에 관한 지속적인 연구가 필요하다.

참 고 문 헌

[1] Wen-Tsuen Chen and Jyh-Shin Lee, "Some Mechanisms to Improve TCP/IP Performance over Wireless and Mobile Computing Environment," IEEE 7th International Conference on Parallel and Distributed Systems, pp. 437-444, Japan, July 2000.

[2] R. Braden, "Extending TCP for Transactions - Concepts," RFC 1397, November 1992.

[3] R. Braden, "T/TCP-TCP Extensions for Transactions Functional Specification," RFC 1644, July 1994.

[4] WAP Forum, "Wireless Application Protocol Architecture Specification," July 12, 2001. URL: http://www.wapforum.org/

[5] WAP Forum,"Wireless Transaction Protocol Specification," February 19, 2000. URL:http://www.wapforum.org/

[6] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithm," RFC 2001, January 1997.

[7] L. Svobodova, "Implementing OSI systems," IEEE Journal on Selected Areas in Communications, vol. 7, pp. 1115-1130, September 1989.

[8] Douglas C. Schidt, "Transport System Architecture Services for High-Performance Communications Systems," IEEE Journal on Selected Areas in Communications, vol. 11, no. 4, pp. 489-506, May 1993.

[9] 한국전자통신연구원, *정보통신프로토콜공학*, ISBN 89-86328-22-4, January 1998.

[10] 오연주, 백낙훈, 이형호, 임경식, "이동 컴퓨팅 환경에서 확률모델을 이용한 FEC기반의 적응적 오류 복구 알고리즘", SK Telecommunications Review, 제10권, 제6호, pp. 1193-1208, 2000년 12월.

[11] Jean-Chrysostome Bolot, Hugues Crepin, Andres Vega Garcia, "Analysis of Audio Packet loss in the internet," Proceedings of Network and Operating System Support for Digital Audio and Video NOSSADV'95, pp. 163-174, Durham, NH, April 1995.

[12] J-C. Bolot, S. Fosse-Parisis, D. Towsley, "Adaptive FEC-Based Error Control for Internet Telephony," Proceedings of IEEE INFOCOM'99, New York, USA, March 1999.

[13] Yingxin Zhou and Zhanhong Lu, "Utilizing OPNET to Design and Implement WAP Service," OPNETWORK 2001, August 2001.

[14] Thomas Kunz, et al., "WAP traffic: Description and Comparison to WWW traffic," Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, pp. 11-19, Boston, USA, August 2000.

최 윤 석

1999년 경북대학교 컴퓨터과학과(이학사). 2002년 경북대학교 컴퓨터과학과(이석사). 2002년 ~ 현재 경북대학교 컴퓨터 과학과 박사과정. 관심분야는 이동 컴퓨팅, 무선 TCP, 무선 트랜잭션 프로토콜, 실시간 전송계층 프로토콜

임 경 식

1982년 경북대학교 전자공학과(공학사). 1985년 한국과학기술원 전산학과(공학석사). University of Florida 전산학과(공학박사). 1985년 ~ 1998년 한국전자통신연구원 책임연구원, 실장. 1998년 ~ 현재 경북대학교 컴퓨터과학과 조교수. 관심분야는 이동 컴퓨팅, 무선 인터넷, 홈 네트워킹, 컴퓨터통신