자바기반 WAP 상의 우선순위 트랜잭션 계층의 설계 및 구현

(Design and Implementation of a Java-Based WAP Transaction Layer with Priority Policy)

이 준 규 † 임 경 수 †† 안 순 신 †††
(Jungyoo Lee) (Kyungsoo Lim) (Sunshin An)

요 약 인터넷과 무선 이동통신 등이 실생활에 보편화되면서 두 기술을 접목한 WAP(Wireless Application Protocol)[1]이 등장하였고, 다수의 사용자 요구를 동시에 효율적으로 처리하면서 다양한 무선 단말기로부터 요청을 신속하게 처리할 수 있는 WAP 게이트웨이(WAP gateway)에 대한 연구가 여러 연구 그룹에서 수행되고 있으며, 이에 대한 효율적인 처리가 필수적이다. WAP 스택[2]은 이러한 연구 중의 가장 기본적인 기술이다. 본 논문에서는 Java의 기본적인 서비스인 멀티쓰레드를 이용하고, 컨테이너 유효 리소스의 효율적 관리를 위해 접속 풀링(connection pooling) 기능을 적용하여 WAP container를 구현하며, 우선순위 정책에 기반한 WAP Protocol의 WTP(Wireless Transaction Protocol)[3] 계층 및 UDP(User Datagram Protocol)[4] 계층을 설계하고 구현함으로 우선순위가 높은 트랜잭션들에 대해 기존 시스템보다 신속히 처리할 수 있는 기능을 구현하였고 모의 실험을 통해 그 효율성을 입증하였다.

키워드 : 무선 인터넷, WAP, WAP 게이트웨이, WTP

Abstract As the Internet and wireless mobile communication is being widespread, the WAP (Wireless Application Protocol) that merges these two technologies has emerged. Also, the research of the WAP gateway that enables efficient processing of multiple user demands concurrently and prompt response to requests from various wireless devices has been performed in many working groups. The WAP stack is the most fundamental technology among these researches. In this paper, we implement the WAP container utilizing the JAVA's multithreading and applying to the connection pooling technique to manage the available resource in the container efficiently. Also, we design and implement the WTP(Wireless Transaction Protocol) layer and the UDP(User Datagram Protocol) layer based on the priority policy. Our design and implementation showed the shorter waiting time than that of the existing FCFS(First-Come, First-Served) system in the transaction layer and its efficiency was proved through simulation.

Key words : Wireless Internet, WAP, WAP Gateway, WTP

1. 서 론

정보의 보고인 인터넷을 중심으로 한 가상공간의 형성으로 자유롭게 임의의 국한된 지역에 의존하지 않고 인터넷을 통한 정보의 수집 및 교환이 가능해지면서 정보의 대중화가 활발히 이루어 지고 있으며, 이는 데이타 커뮤니케이션 서비스, 개인정보 관리 서비스, 부가가치 정보 서비스, 전자 상거래 서비스등 새로운 문화창출이라는 모습을 보여주고 있다. 또한 이동 중인 사용자에게 통신을 가능하게 하는 무선 이동 통신의 급속한 발전으로 인하여 이러한 두 매체를 접목한 새로운 서비스 형태인 이동 통신 공중망을 이용하는 무선 인터넷이 많은 활성화를 보이는 추세를 보이고 있다. 따라서 이러한 정

• 고려대학교 특별연구비에 의하여 수행되었음
† 비 회 원 : 대신증권 전산개발팀 대리
jklee@daishin.co.kr
†† 학생회원 : 고려대학교 전자컴퓨터공학과
angus@dsys.korea.c.kr
††† 종신회원 : 고려대학교 전자컴퓨터공학과 교수
sunshin@dsys.korea.ac.kr
논문접수 : 2002년 3월 5일
심사완료 : 2002년 12월 18일

보를 무선 인터넷을 통하여 효과적으로 얻고자 하는 시스템들이 만들어지게 되었고 이를 겨냥한 기술의 필요성에 많은 관심이 증대되고 있다. 현재 무선 인터넷 환경과 무선 단말기는 전력 소모량, 메모리 크기, 화면 표시 능력, 전송 용량 및 속도, 안정성 등에 대한 많은 제한 사항을 가지고 있다. 이러한 제한 사항은 유선 인터넷 표준을 무선 인터넷 환경에서 그대로 사용하기에는 적합하지 않으므로, 기존 방식을 상당부분 수용하면서 새로운 무선 환경에 적합한 프로토콜의 규정이 요구되어 졌다. 기존의 인터넷과 무선망의 연동을 위한 연구 개발이 활발하게 이루어지는 가운데 유선으로만 제공했던 인터넷 서비스를 무선의 환경에서 가능하게 해주기 위한 무선 응용 프로토콜 개발이 주된 관심사로 떠오르게 되었고, 이러한 추세 속에 현재 WAP, ME, I-mode 등의 다양한 솔루션이 개발되었고 서비스 중에 있다.

이 가운데 WAP은 공개된 프로토콜로서 전세계적으로 가장 많이 사용되고 있는 프로토콜이며 가장 활발한 연구가 이루어지고 있다. 이동통신제공업체의 선두주자인 몇 개의 업체가 이동 디바이스에서 사용할 인터넷 프로토콜 표준을 제정하기 위해서 WAP 포럼을 구성하였고 이곳에서는 WAP 모델이 정의되었다. 이 모델은 클라이언트인 휴대용 단말기와 인터넷 서버 사이에 WAP Proxy라 불리는 WAP 게이트웨이를 두어 서로 다른 두개의 망 사이에 접목을 수행하는 역할을 하도록 한다. WAP 게이트웨이의 주요 역할은 서로 상이한 두 프로토콜인 WAP과 인터넷 TCP/IP를 무선 단말기와 Origin Server사이에서의 변환을 수행하여 준다. 즉, 모든 휴대용 단말기의 인터넷 서비스 요구는 WAP 게이트웨이를 경유하도록 되어 있고, 게이트웨이는 이를 HTTP 요청으로 변경하여 기존의 유선 네트워크를 통하여 서비스를 요청한다. 이어서 게이트웨이가 인터넷 서버로부터 요청에 대한 응답을 받고 다시 서비스를 최초 요청했던 휴대용 단말기로 WAP 프로토콜의 형태로 변형하여 전송함으로써 모든 과정이 이루어진다. WAP 스택은 기본 데이타를 프로토콜에 맞추어 전송하고 수신하는 매체의 역할을 한다.

현재 WAP과 관련되어 많은 연관된 연구가 이루어지고 있다. 대표적으로는 WAP 환경의 성능 향상에 관한 것과 WAP과 다른 프로토콜과의 연동에 대한 연구가 활발히 진행되어지고 있다. 또한 플랫폼에 대한 연구와 보안에 관련된 연구도 여러 방향으로 접근이 되고 있다. 많은 연구가 자바 기반에서 이루어 지고 있고 또한 많은 업체들이 이러한 연구동향에 맞게 개발 및 서비스를 제공한다. 이러한 추세에 비추어볼 때 향후 자바 기반의 무선 인터넷 시장의 활발한 발전이 이루어질 것으로 예상되며 본 연구에서는 이러한 연구동향에 맞추어 자바 기반의 WAP 프로토콜 스택을 설계 및 구현하고, 동시 전송에 대한 성능을 향상시키고자 자바의 멀티쓰레드 및 접속풀링기법을 사용하며 서비스를 받는 사용자의 차별화에 효과적으로 처리할 수 있는 방향에 대해 우선순위를 고려한 트랜잭션을 처리하는 WAP 스택을 설계하고 구성하고자 한다.

본 논문은 다음과 같은 구성으로 되어 있다. 2장에서는 무선 인터넷과 WAP에 관련된 연구를 살펴보고, 3장에서는 WAP의 트랜잭션 계층과 UDP에 관련된 주요한 구조를 설계하고 구현한다. 4장에서는 LAN과 WAN상에서 우선순위 정책에 따른 성능 평가를 하고, 마지막으로 5장에서는 결론 및 향후연구를 기술하며 끝을 맺는다.

2. 관련연구

2.1 WAP 스택

WAP 구조는 이동 통신 장치를 위한 응용프로그램 개발을 위해 확장 가능한 프로토콜 환경을 제공하고, 스택의 구조는 전체 네트워크의 계층화된 구조를 제공한다. 이는 단말기가 무선 네트워크를 통하여 WAP 게이트웨이에 접근할 수 있도록 좁은 대역폭에 알맞게 설계되었을 뿐만 아니라 각 국가 및 망 사업자들마다 다른 데이타 베어러(bearer)에 맞게 5개의 계층으로 설계 되었다. 각각의 계층은 상위 계층에 의해서 접근될 수 있으며 다른 서비스나 응용 프로그램에 의해서도 이용될 수 있다. 외부 응용프로그램은 WSP[5], WTP, WTLS(Wireless Transport Layer Security), UDP, WDP[6] 계층에 직접 연결될 수 있다.

WAP 기술은 WAP 포럼에 의해 제시된 것 이상으로 응용 프로그램과 서비스들을 위해서 유용하게 사용될 수 있다. 그림 1은 WAP을 이용한 서로 다른 구조의 프로토콜 스택이다.

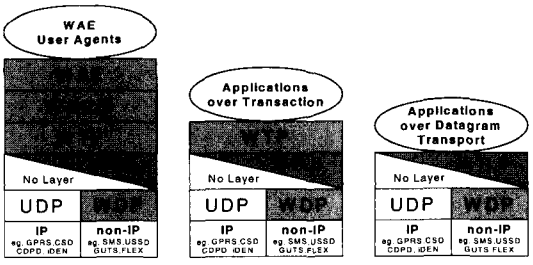


그림 1 WAP 스택의 예

WAP을 설계하고 구현함에 있어서 모든 계층을 구현함으로써 WAP Application을 사용할 수 있는 것이 아니며 각각 사용할 무선 네트워크 베어러의 특징 및 WAP Application의 사용목적 및 특징에 따라서 제시된 그림 1과 같이 여러 가지 형태로 스택의 구성이 가능하다. 왼쪽의 스택 구조는 WAP 응용의 가장 전형적인 형태이고 WAP의 모든 계층을 구현하며 WAE 사용자 에이전트가 이용하는 구조이다. 가운데 스택 구조는 트랜잭션을 요구하는 응용프로그램과 서비스에 의해 사용되는 경우이다. 마지막으로 오른쪽의 스택 구조는 단지 데이타그램 전송을 요청하는 응용프로그램과 서비스에 의해 사용되는 경우이다. 모든 구조에 공통적으로 보안에 관련된 WTLS 계층은 선택 사항이다.

2.2 자바 멀티 쓰레딩(Multi-Threading)과 접속 풀링

무선 인터넷이 급속도로 발전을 하면서 WAP 게이트웨이는 몇 천, 몇 만개의 서비스를 수행해야 하고 다수의 무선 단말기와 접속을 유지해야 할 필요가 있다. 무선 단말기의 접속하는 수가 증가할수록, 시스템의 리소스의 유효량은 감소하고, 어느 시점에서는 그 수행 속도가 현저히 감소할 것이다. 이러한 문제점의 해결방안으로 본 논문에서 설계하는 WAP의 컨테이너는 풀링 기능을 제공한다. 풀링의 개념은 이미 시스템 내의 비즈니스 객체와 데이타베이스의 접속을 분할 처리하기 위해 데이타베이스 접속 풀링[7]이라는 기술로 널리 사용되고 있었다. 이 기술은 연결 수와 리소스 소비량을 줄이며 그 수행속도를 향상시켰다. 이는 데이타베이스를 여러 번 새로이 접속하고 끊는 것보다 한번 접속하고 대기 상태에 머무르다가 재사용하는 것이 좀 더 적은 양의 리소스를 소비한다는 점에서 착안된 것이다. 접속 풀링이라는 메커니즘을 사용하여 전체 클라이언트와 WAP 게이트웨이 간의 접속을 맺고 끊을 때 효율성을 증가시키고자 한다. 그림 2는 WAP gateway에서의 접속풀링을 이용한 프로토콜스택의 생성 과정을 보여준다.

컨테이너를 설계하는데 있어서 반드시 사용하는 기술은 멀티쓰레드[8][9]이다. 이것은 하나의 프로그램 안에서 다중의 흐름, 쓰레드가 동시에 수행되는 것을 의미한다. 이것은 같은 데이타 공간을 공유하면서 수행되므로 멀티태스킹(multi-tasking)과는 다른 개념이다.

자바에서는 멀티쓰레드 기능을 가진 두 개의 클래스와 하나의 인터페이스를 제공한다. 첫번째 클래스는 Thread이고, 시스템 쓰레드를 휴지(pause), 재개(resume), 정지(stop)하는 기능이 있다. 다른 하나의 클래스는 다중의 쓰레드를 관리해주는 역할을 하는 ThreadGroup이 있다. 인터페이스로는 Thread 클래스를 직접적으로 상속 받지 않

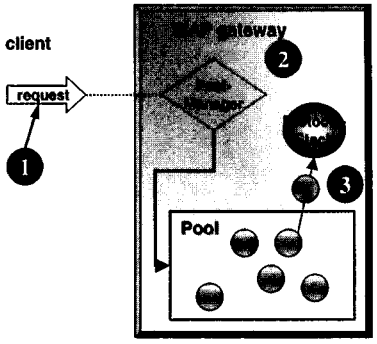


그림 2 WAP gateway의 접속풀링 개념도

은 객체들이 쉽게 쓰레드를 생성할 수 있도록 도와주는 Runnable이 있다.

3. 설계 및 구현

3.1 WAP Gateway

WapGateway 클래스는 여러 개의 클래스들의 조각틀로 구성된다. Listener 클래스는 주어진 포트로 단말기의 요청이 들어오기를 기다리는 쓰레드이다. PoolManager 클래스는 현재 연결된 접속 리스트를 관리한다. Listener는 단말기로부터 접속요청이 들어오면, PoolManager에게 메시지를 보낸다. 이때 PoolManager는 접속 요청을 받아들일 수 있는가를 조사한 후 더 이상 접속 요청을 수용할 수 없으면 단말기의 요청을 거절한다. PoolManager가 단말기와의 접속이 가능하면 접속에 대한 전반적인 처리를 담당하는 새로운 ProtocolStack 객체를 생성한다. ProtocolStack은 Thread 클래스로부터 상속 받은 클래스이고, 각각의 WAP 스택의 서비스들은 동시에 다중의 접속을 처리할 수 있다. 각각의 ProtocolStack 객체는 WAP 스택 객체에 메시지를 보내 실질적인 데이타 처리를 위한 작업을 수행한다.

WapGateway의 main() 함수는 컨테이너를 생성하고 동작시켜 주는 스탠드-어론 프로그램[10]의 시작점이다.

그림 3은 이러한 전체적인 구조를 구성하는 뼈대를 보여주고 있다. WAP gateway와 WAP client 양단에는 그림에서 보여지는 것처럼 동일한 프로토콜 스택이 올라가게 된다. 트랜스포트 계층에서는 IP 환경에서의 테스트를 하기위해서 UDP를 사용하게 된다. 트랜잭션 계층은 Initiator와 Responder로 구성이 된다. 그 위로 세션을 관리하는 계층이 있으며 각 계층은 이벤트큐를 통해서 각 계층간 통신이 이루어지게 된다. 그림에서 보여지는 WAP의 전체구조 중 본 논문에서는 WAP gateway의

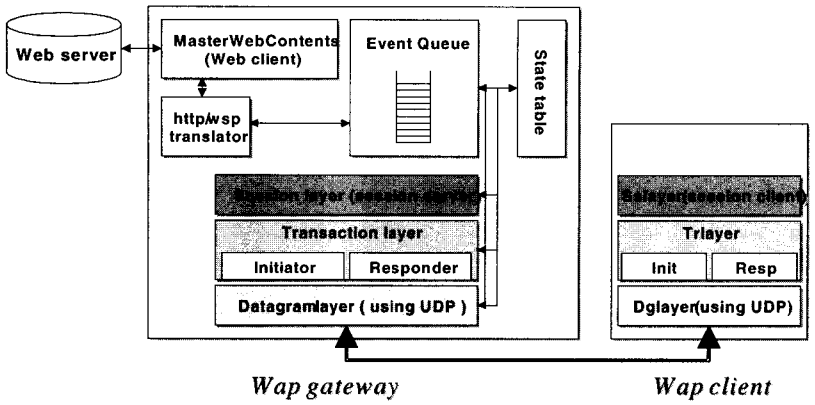


그림 3 WAP 기본 구조

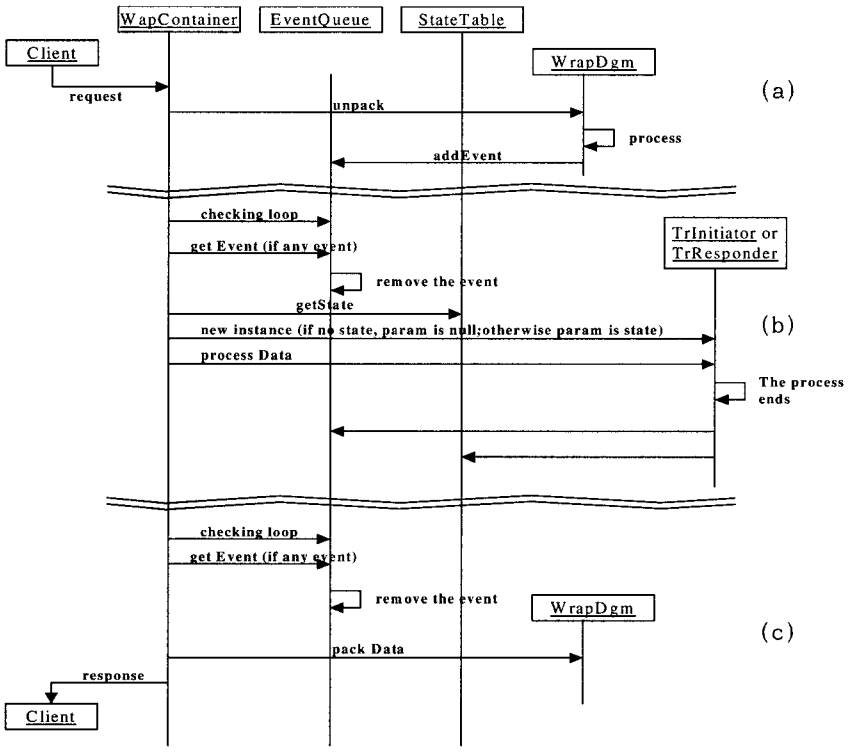


그림 4 패킷 송수신 순차 다이어그램

Session 계층 부분을 제외하고 설계 및 구현을 하였다. 이 부분에 대해서는 차후에 연구될 것이다.

그림 4는 컨테이너에서 패킷 송수신 순차 다이어그램이다. 이때 (a)는 WAP클라이언트가 메시지를 보내면 WAP 컨테이너는 패킷을 본래의 데이타를 가져오면서 새로운 이벤트가 발생되었다고 Event Queue에 등록을 시키는 과정이다. (b)는 컨테이너가 EventQueue를 조사하면서 등록된 이벤트가 있으면 StateTable의 등록된

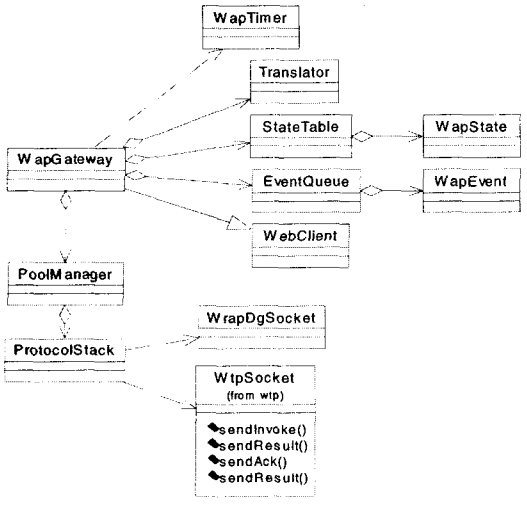


그림 5 게이트웨이 클래스 다이어그램

상태가 있는지 조사하고, 이 결과에 따라 이벤트와 상태를 같이 트랜잭션 처리루틴에 넘겨 처리를 하는 과정을 보여준다. 이때 추후에 다시 처리되어야 하는 이벤트와 상태가 존재한다면, EventQueue와 StateTable에 등록을 시킨다. 그리고 (c)는 컨테이너가 클라이언트에 보내는 이벤트가 있는지 조사를 한 후에 실질적인 데이타를 전송하기 위한 패킷에 실어서 보내는 과정을 그림상으로 표현하고 있다.

그림 5는 WAP 게이트웨이의 클래스 다이어그램이고, WapGateway의 *main*() 함수는 컨테이너를 생성하고 동작시켜 주는 스탠드-어론 프로그램의 시작점이다. WapTimer 클래스는 통신상의 시간초과를 처리하여 주고, Translator 클래스는 WAP과 WWW간의 컨텐츠를 변환시켜 준다. StateTable, EventQueue는 큐의 구조를 갖고 있으면서 통신상태(WapState)와 발생되는 이벤트(WapEvent)들을 관리한다. PoolManager와 ProtocolStack은 실질적인 WAP의 통신계층(UDP, WTP)을 관리하는 클래스이다.

3.2 Transport Layer(UDP)

WAP 프로토콜을 구현하면서 가장 하위 계층은 WDP [6]를 사용하지만, IP 계층 위에서 동작하는 WAP 게이트웨이는 UDP상에서 동작을 한다. 이 설계는 시뮬레이션용으로 사용할 스택을 구현하는 것을 목적으로 하기 때문에 Java를 이용한 구현에서 소켓용 UDP[10]를 이용한다. UDP는 IP 계층 위에서 동작하는 비연결-지향형 전송 프로토콜이며, 패킷을 기본 단위로 하고, 패킷을 생성하고 네트워크를 통해 원하는 목적지로 전송하는 과정을 계속해서 반복한다. 각각의 패킷은 서로 독립적이며 분리되어 전송되고, 매번 주소 체계를 포함하여야 한다. 자바에서 UDP를 구현하기 위해서는 기본적인 클래스 세가지를 사용하여 구현한다. DatagramPacket, DatagramSocket, InetAddress 등이다.

UDP 패킷을 전송하려면 목적지 주소와 실질적인 데이타로 구성된 DatagramPacket의 객체를 생성해야 한다. 그리고 나서 각각의 패킷에 내용을 채워넣고, 네트워크 전송을 위한 포맷으로 대치하여 전송한다. 패킷을 받는 경우에는 DatagramPacket 객체를 생성하고, 네트워크를 통해 들어온 패킷을 객체로 복사하여, 그 객체로부터 출발지 주소와 내용을 추출해 낸다. 여기서 WrapDgSocket은 DatagramPacket을 상속하여 제작함으로써 추후에 확장성에 대응한 설계를 한다.

3.3 Transaction Layer(WTP)

WTP는 IP 환경에서는 UDP 위에서, IP 환경이 아닌 곳에서는 WDP 위에서 수행되면서 인터넷 접속 전용 컴퓨터에서 실행하기에 적당한 트랜잭션 프로토콜이다. WTP는 보안 혹은 비보안 무선 데이타그램 네트워크 위에서 모두 동작한다. 교환의 기본 단위는 Byte stream이 아니라 Message이다.

TCP상에서는 트랜잭션을 이용한 요청-응답을 기본으로 하는 연결-지향형 서비스를 통해 효율성을 증가시킨다. 또한 무선환경에서는 비연결형서비스를 사용함으로써 연결시 발생할 수 있는 통신상 오버헤드 등의 과부하를 제거하였다.

프로토콜을 구현하기 위해서는 트랜잭션 내부의 PDU간에 기밀성을 제공하고, 전체적인 관리를 위해 생겨난 TID(Transaction Identifier)의 관리가 필요하다. 이때, TID는 송신지 주소, 수신지 주소, 수신지 포트번호 등과

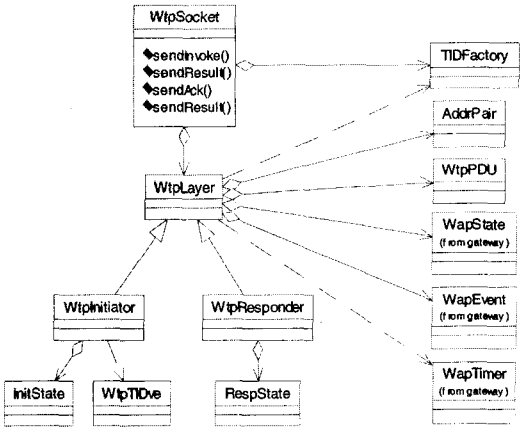


그림 6 WTP 클래스 다이어그램

함께 작용하여 같은 트랜잭션 내의 모든 작업들이 유일함을 증명해준다. TIDFactory 클래스는 WtpResponder 객체가 주로 사용하며 TID가 유효한지를 검사한다. TID 실증과정은 PDU 내부의 TIDnew 항목이 세트 되었을 때 실행된다.

클라이언트와 서버에서 보내는 모든 PDU에는 반드시 SendTID를 포함해야 하고, RcvTID는 처음 Invoke 메시지를 받으면서 생성된다. 그리고 RcvTID는 하나의 트랜잭션이 종료될 때까지 계속해서 사용된다. SendTID와 RcvTID는 XOR의 관계를 갖는다.

그리고 WtpPDU는 일반적인 특성의 Invoke, Result, Ack, Abort와 선택 사항인 Segmented Invoke, Segmented Result, Negative Acknowledge 등으로 각각 구분하여 처리한다. WapTimer는 트랜잭션 수행 중에 서로 간의 상태를 관리하기 위한 타이머의 역할을 한다.

그림 7은 Transaction Initiator의 상태와 Transaction Responder의 내부 상태의 흐름을 보여주는 상태 다이어그램이다. 여기서 각각의 상태 전이 이벤트는 매우 복잡한 관계로 그림 상에서 생략되었다. C0 서비스(Class 0 service or transaction)는 WtpInitiator에 의해 수행되고, 주로 확인을 받을 필요 없는(상태를 관리하지 않는) 메시지를 보낼 때 사용한다. C1 서비스는 상태 관리를 위해 타이머를 사용하고, Initiator와 Responder사이에 메시지를 주고 받는다. 에러 핸들링은 대부분 상태를 가지고 있는 WtpInitiator와 WtpResponder에게 WapEvent

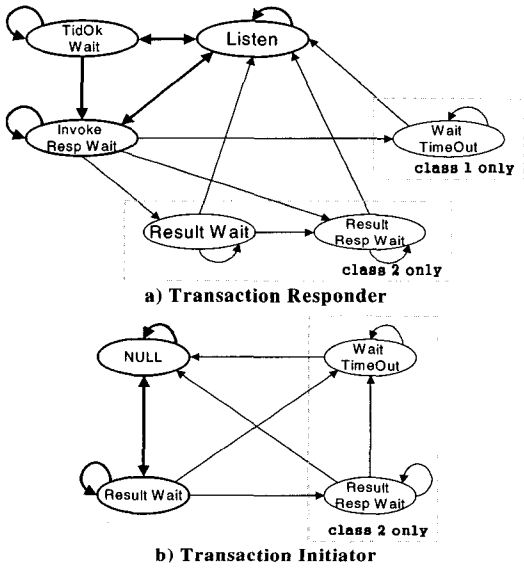


그림 7 트랜잭션 상태 다이어그램

를 넘겨주기 전에 이벤트와 상태정보를 비교, 분석하여 수행된다. 그렇지만 받은 PDU자체 내에 잘못된 헤더파일을 포함하고 있으면 WTP는 처리할 수 없다.

또한, 본 논문에서 주안점을 둔 각각의 PDU에 우선순위를 주기 위해 기본적인 PDU헤더 바로 뒤 1 byte를 우선순위 등급을 표시할 수 있도록 예약하여 사용하였다.

3.4 일반적인 큐와 우선순위 큐

Queue 인터페이스는 기본적인 기능들을 선언하였고, QueueImpl와 QueuePri 클래스는 Queue를 상속 받아서 실질적인 구현을 하였다.

Queue는 기초적으로 수행해야 할 add()와 remove() 함수를 포함하고 있다. Queue의 내부 변수로는 정보를 저장하는 Vector의 속성을 가진 변수를 가지고 있다. 여기서 add()는 구성요소를 첨가하고, remove()는 삭제하며, size()는 몇 개의 구성요소를 포함하고 있는지 조사하고, isEmpty()는 큐가 비어있는지를 조사한다. QueuePri와 QueueImpl의 공용 함수는 동일하며, QueuePri는 add() 함수 내에 삽입할 장소를 결정짓는 search() 함수가 첨가되어 있는 것이 다른 점이다. 이는 우선순위 큐가 Heap 자료구조로 이루어 졌기 때문에 해당되는 적합한 위치에 삽입을 결정짓는 역할을 한다. QueuePri와 QueueImpl의 기능상의 상이한 점은 그림 8을 통하여 알아볼 수 있다.

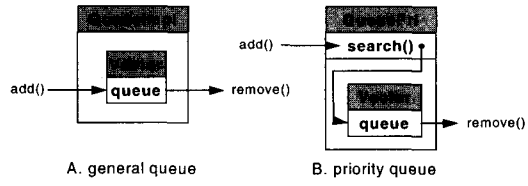


그림 8 큐의 내부 동작

4. 우선순위 큐의 성능 평가

실험은 LAN과 WAN 상에서 클라이언트와 게이트웨이간의 메시지를 보내고 받는 기본적인 실험을 하였다. 실험환경은 테스트용 WAP 게이트웨이가 설치되어있는 컴퓨터를 한국통신 ADSL 망위에서 작동하도록 하고, 클라이언트는 고려대학교 네트워크 연구실에 위치하였다.

테스트는 한번에 동시에 작동할 수 있는 Thread를 40개로 제한하고, 우선순위는 4개로 분리하였다. 이때 우선순위는 r=0가 가장 낮고, r=3이 가장 높다. 우선순위는 랜덤하게 발생하여 대략 같은 수의 비율로 발생하게 된다. 트랜잭션에 사용되는 메시지의 크기는 258 bytes로 고정시켰고, 4300개에서부터 수렴을 하여 그보

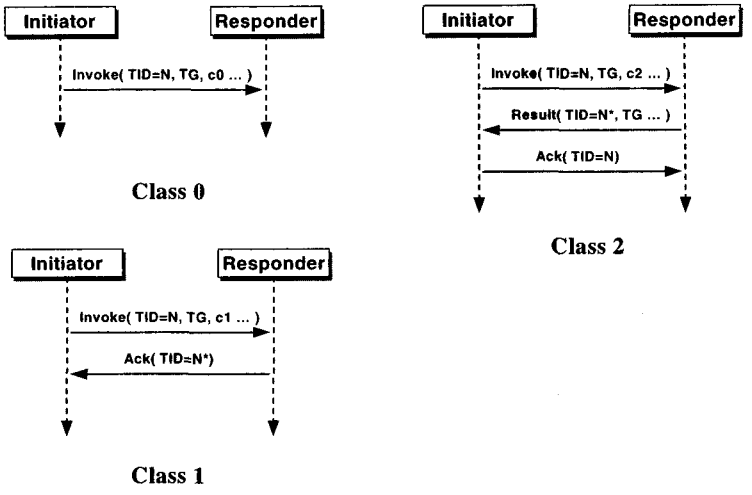


그림 9 기본 클래스 트랜잭션

다 많은 수인 6000개 트랜잭션을 각 클래스마다 수행하였고 트랜잭션은 세 가지(class 0, class 1, class 2)로 나누어 테스트를 하였다. 각 트랜잭션은 그림 9에 설명한다.

그림 10과 표 1은 각각 LAN과 WAN에서 처리시 대기 시간의 평균을 계산하여 통계를 낸 것이다. 각 그림과 표에서 볼 수 있듯이 기존의 FCFS 방법의 트랜잭션 처리방식보다 우선순위가 높게 부여한 트랜잭션의 처리가 WAN보다는 LAN일 경우 더욱 대기시간이 짧아짐을 볼 수 있다. 이는 LAN의 경우 1개의 홉만을 경유하지만 WAN의 경우에는 상대적으로 많은 수의 홉을 경유하기 때문에 응답시간이 느려진다. 또한 우선순위가 낮을 때 보다는 우선순위가 높게 책정되었을 경우 그 대기시간은 급격히 짧아짐을 볼 수 있다. 우선순위를 적용한 r0-r3까지의 전체 평균 대기 지연 시간과 FCFS로 처리하는 경우의 평균 대기 지연 시간이 거의 유사함을 알 수 있다. 그러나 이중 r1-r3까지의 평균 대기시간과 FCFS 평균대기시간을 비교하면 우선순위가 높게 책정된 경우 평균대기시간이 짧아지면서 다수의 트랜잭션의 지연 시간 감축효과를 가져온다. 반면 우선순위가 낮은 r0의 경우에는 대기시간이 많이 걸리는 문제점을 안고 있다. 이는 상대적으로 우선순위가 낮은 것에 대해서의 대안 방법의 필요를 알 수가 있으며 이러한 문제점이 해결된다면 전체 평균 대기 지연 시간이 더욱 짧아질 것이다. 이 부분에 대해서는 차후에 연구될 것이다.

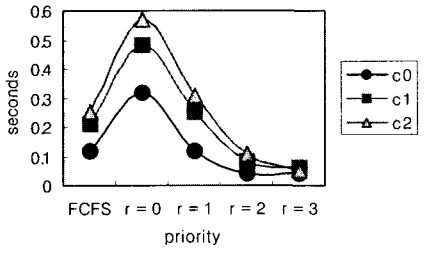


그림 10 LAN상에서의 대기 시간

표 1 LAN과 WAN상에서의 평균 대기시간

Queue type	Rate	Class 0		Class 1		Class 2	
		LAN	WAN	LAN	WAN	LAN	WAN
FCFS	None	0.12	30.64	0.21	45.19	0.25	53.77
Priority	0	0.32	120.20	0.48	164.18	0.57	221.57
	1	0.12	8.69	0.25	12.79	0.31	16.06
	2	0.04	3.14	0.08	4.71	0.11	5.63
	3	0.04	1.57	0.06	2.28	0.05	2.87

5. 결론 및 향후 연구

무선 인터넷의 급속한 발전으로 인하여 이에 관련된 기술들의 연구가 활발이 이루어 지고 있다. 그 중 무선 인터넷 프로토콜에 대한 다방면의 연구가 진행되고 있다. 본 연구에서는 WAP상에서 트랜잭션의 일괄적인 처리방식을, 중요한 트랜잭션을 우선시하여 처리하는 방법에 대해 자바 기반으로 설계 및 구현하였고, 우선순위가 높은 클래스를 갖는 트랜잭션은 FCFS로 설계한 것에 비해 빠르게 처리된다는 것을 보였다.

우선순위가 높은 클래스에 대해서는 서비스를 받기까지 대기하는 시간이 FCFS로 처리되는 것에 비하여 대기시간이 작은 장점을 갖고 있다. 하지만, 상대적으로 우선순위가 낮은 클래스는 많은 시간을 기다려야 하는 문제점이 존재한다. 향후 좀더 효율적인 우선순위 정책을 제공하여 전체 시스템에 대한 성능향상 방안이 연구되어야 한다.

본 연구에서는 추가적으로 Mobile IP와 같은 다른 종류의 네트워크 환경에서의 테스트를 통하여 다른 프로토콜 스택과 독립적으로 연동한다는 것을 증명하는 과제가 남아있으며, WSP 및 WTLS를 추가 제작하여 무선 통신의 안정성을 더해야 하는 과제가 남아있다. 또한 서비스별로 분류된 WAP 2.0 스펙에 대한 연구를 통해 새로운 안정적인 시스템의 모델을 제시하는 연구가 진행될 것이다.

참 고 문 헌

[1] WAP Forum, "WAP Architecture Specification," 30-April-1998.

[2] 이준규, 김동호, 김상경, 안순신, "자바 기반의 WAP 스택 설계", 정보과학회 학술발표 논문집, 2001년 4월.

[3] WAP Forum, "Wireless Transaction Protocol Specification," 19-February-2000.

[4] Merlin Hughes, Michael Shoffner, Derek Hamner, and Umesh Bellur, "Java Network Programming," MANNING, 1999.

[5] WAP Forum, "Wireless Session Protocol Specification," 4-May-2000.

[6] WAP Forum, "Wireless Datagram Protocol Specification," 19-February-2000.

[7] Richard Monson-Haefel, "Enterprise Java Beans," O'REILLY, 1999.

[8] Bill Venners, "Inside the JAVA 2 Virtual Machine," McGrawHill, 2000.

[9] Bolch, Greiner, de Meer, and Trivedi, "Professional Java Server Programming," WROX, 1998.

[10] Ted Neward, "Server-Based Java Programming," MANNING, 2000.

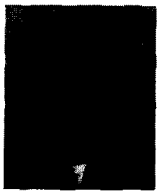
[11] WAP Forum, "Wireless Markup Language Specification," 19-February-2000.

[12] Ken Arnold, and James Gosling, "The Java Programming Language," Addison Wesley, 1998.

[13] WAP Forum, "Wireless Application Environment Specification," 29-March-2000.

이 준 규

1996년 경기대학교 전자공학과 졸업 1996년 ~ 1999년 한국소리마치 부설연구원 원구원. 2002년 고려대학교 대학원 전자공학과 석사. 현재 (주)대신증권 전산개발팀 대리. 관심분야는 분산처리, 멀티개스팅, 이동통신



임 경 수

2001년 순천향대학교 컴퓨터공학과 졸업. 2003년 고려대학교 대학원 전자공학과 석사. 현재 고려대학교 대학원 전자컴퓨터공학과 박사과정. 관심분야는 센서네트워크, 정보가전, QoS 라우팅, 이동통신

안 순 신

1973년 서울대학교 공과대학 졸업(B.S) 1975년 한국과학기술원 전기 및 전자과 졸업(M.S). 1979년 블란서 ENSEEIHT에서 공학박사 취득(ph.D). 1979년 3월~1982년 8월 아주대학교 전자과 교수 1991년 1월~1992년 2월 NIST(National Instiute of Standard and Technology) 방문연구원. 1982년~현재 고려대학교 전자공학과 교수. 관심분야는 컴퓨터 네트워크 및 분산 시스템