실시간 RFID 미들웨어시스템에서의 동기화를 고려한 필터링관리 기법의 설계 및 구현

Design and Implementation of Filtering Management Scheme for Synchronization in the Realtime RFID Middleware System

박병섭
인하공업전문대학 컴퓨터시스템과

ByoungSeob Park(bspark@inhatc.ac.kr)

요약

본 논문은 대용량 데이터 처리를 위한 실시간 RFID 미들웨어 시스템에서 요구되는 태그 데이터의 필터링 엔진의 동기화를 고려한 필터링 관리기법을 다루고 있다. 응용인터페이스는 HTTP, XML, JMS, SOAP 등의 이는 다양한 프로토콜을 지원하여 다양한 플랫폼에서 본 미들웨어 시스템을 접근하도록 개발되었다. 일반적으로, 필터를 제어하는 클라이언트가 다수가 되는 환경에서 하나의 필터링 화일을 접근하면 동기화 문제가 발생한다. 본 논문에서는 필터 관리프로세스를 통해 동기화를 고려하면서 필터링을 관리하는 기법을 설계하고 구현하였으며, 이를 RFID 미들웨어의 동작 통해 검증한다.

■ 중심어 : | RFID미들웨어 | 필터링 | JMS | SOAP |

Abstract

We design a filtering management scheme with synchronization function under a realtime RFID middleware system for larger-scale data processing. The application interface(AI) is to support a various access protocol, HTTP, XML, JMS, and SOAP for the RFID applications. Generally, the synchronization problem is occurred in multiple accessing of clients for single filtering file. In this paper, we implement a filtering management scheme supporting the synchronization using the filter management process, and then demonstrate the RFID middleware filtering scheme.

■ keyword : | RFID Moddleware | Filtering | JMS | SOAP |

I. 서 론

최근에 RFID 기반 미들웨어 제품 및 솔루션들이 많이 개발되고 있으며, EPC 코드 등과 같은 간단한 형식의 데이터를 처리에서 대량의 데이터 처리 분야로 방향 전환이 이루어지고 있다. ALE(Application Level Event) 버전을 지원하는 미들웨어 시스템들이 국내외에서 연구 개발되어 발표되고 있다[1-3]. 일반적인 RFID 미들웨어 시스템의 기본 구조는 실시간 스트림 태그 데이터를 읽어내는 리더인터페이스(RI : Reader Interface), 수집된 데이터를 의미 있는 데이터로 가공/필터링하는 이벤트 관리자(EM : Event Manager), EM은 데이터를 필터링 조건에 따라 의미 있는 데이터로 가공하는 필터링 엔진, 필터링 결과 데이터를 RFID 응

* 본 연구는 2005년도 산업자원부 신기술실용화기술개발사업 연구비 지원에 의해 수행되었습니다.

접수번호 : #070416-001
접수일자 : 2007년 04월 16일
심사완료일 : 2007년 07월 02일
교신저자 : 박병섭, e-mail : bspark@inhatc.ac.kr

용으로 전달하기위한 HTTP, JMS(Java Messaging Service), SOAP(Simple Object Access Protocol), XML 기반의 응용인터페이스(AI : Application Interface)로 구성된다[4][5]. 본 논문에서는 인프라스트럭처의 확장성과 통합, 그리고 대용량의 데이터 처리와 통합프로세스관리를 지원하며, Java 기반의 XML을 지원하는 형태로 개발하여 보다 공통 데이터 관리 모델을 지향하는 미들웨어에 적용될 필터링 관리 기법에 대해 다루고자 한다.

현재 기 개발된 RFID 미들웨어들의 필터링 관리 기법들에 있어서 컨텍스트 정의 및 처리는 독자기술에 의한 기 정의된 필터(predefined filter)를 설계하여 적용하고 있다. 다음 표는 본 논문을 적용할 미들웨어를 기준으로 각 미들웨어서 사용하는 기능들을 중심을 정리한 것이다[2-4].

분 류	OAT System	SUN Java System RFID Software	Oracle Edge Server/ Sensor Data Hub	Ours
지원 리더기 종류	Matrics, Alien, ThingMagic, SAMSys, AWID	Alien, Matrics, Sensormatic, ThingMagic	Alien, Intermec, lightstick	Alien, Intermec(PDA형), Matrics, 국내B사
데이터 형식	EPC	EPC	EPC	EPC
컨텍스트 처리 (이벤트)	Predefined filter (SQL like)	Predefined filter	Predefined filter	Predefined filter
엔터프라이즈 연동	XML via File, JMS, http	File System, JMS, XML/HTTP/SOAP	Stream, JMS, Web Service, Http Post	HTTP, JMS, SOAP, XML
초당 데이터 처리	?	?	?	수천/sec

그림 1. 각 미들웨어 기능 비교표

일반적으로 미들웨어에서는 다양한 응용 API (Application Programming Interface)들을 지원하여 다양한 플랫폼에서 본 미들웨어의 설정을 접근할 수 있도록 한다. 그런데 필터를 제어하는 클라이언트가 다수가 되는 환경에서는 하나의 필터링 파일을 사용하면 동기화 문제가 생겨 파일을 손상될 수 있다. 본 논문의 목적은 이러한 동기화 기능을 가지며, RFID 미들웨어에서 적용 가능한 필터링 관리 기법 설계하고 이를 구현한다. 이러한 방법은 여러 클라이언트들이 동시에 접속시에도 미들웨어가 원할이 동작하도록 하는데 있다. 동기화 기능을 지원하는 필터링 관리 기법은 RFID 응용 시스템들이 여러 지역에 분산되어 수행됨을 기본으로 할 때 꼭 해결해야할 문제이다. 본 논문에서는 이 동기화를 지원하는 필터파일을 관리하는 프로세스(XML Process)를 하나 두고, 모든 필터의 제어는 이 프로세스를 통하여 작업하도록 구현한다.

본 논문은 서론에 이어 제2장에서는 RFID 미들웨어 구조에 대해 간략히 언급하고, 필터링 관리 시스템의 구조를 다룬다. 제3장에서는 XML 필터링 관리시스템의 개발내용에 대해 설명한다. 제4장에서는 필터링 관리 시스템의 테스팅을 수행한다. 그리고 마지막으로 5장에서 결론을 맺는다.

II. 필터링 관리 시스템 구조

1. 적용 미들웨어 구조

기개발된 RFID 미들웨어의 구조 및 데이타 흐름도는 [그림 2]와 같다. RI에서 리더를 통해 태그 데이터를 읽어 들이면, EM 모듈에서 필터링 및 라우트를 수행하고, 이런 필터링 된 데이터는 AI의 적절한 응용 API를 거쳐 응용프로그램으로 전달된다. 미들웨어서 필터링 관리는 AI에서 필터링 요청을 EM으로 전달하게 되며, 실제 필터링은 EM엔진에서 수행되는 구조이다.

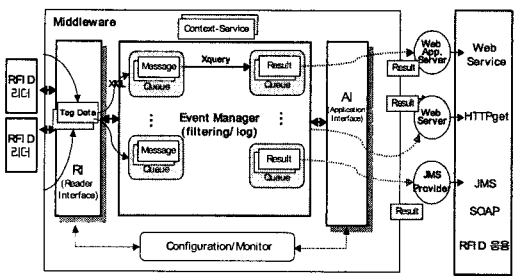

그림 2. RFID 미들웨어의 데이타처리 흐름도

2. XML 필터링 매니저

XML 필터링 매니저(XML Filter Manager)는 미들웨어에서 사용하는 필터를 관리하기 위한 프로그램이다. 그리고 다양한 프로토콜을 지원하여 다양한 플랫폼에서 본 미들웨어의 설정을 접근할 수 있도록 한다. 그

런데 필터를 제어하는 클라이언트가 다수가 되는 환경에서는 하나의 필터링 파일을 사용하면 동기화 문제가 생겨 파일을 손상될 수 있다. 그렇기 때문에 [그림 3]에서처럼 필터파일을 관리하는 프로세스(XML Process)를 하나 두고, 모든 필터의 제어는 이 프로세스를 통하여 작업함으로써 동기화 기능을 지원하면서 필터링 관리 기능을 제공할 수 있다. 그래서 다수의 프로세스로가 다양한 프로토콜을 통해서 필터 제어 명령을 내리고 하나의 프로세서에서 파일을 갱신하여 성공적인 필터 정보를 관리한다. 본 응용 API 구조는 HTTP, JMS, SOAP 등의 프로토콜을 통해 클라이언트가 필터링 결과를 엑세스할 수 있는 구조이다.

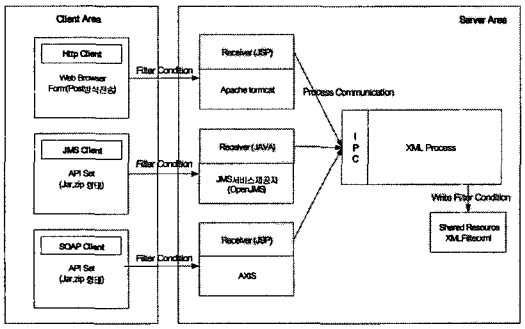

그림 3. XML 필터링 매니저

동기화를 제공하는 XML 필터링 매니저에 적용된 클래스의 그의 역할에 대한 내용이 [그림 4]에 나타나 있다.

영역	클래스명	역할
공통	FilterSet.java	필터들의 정보를 담는 클래스
	FilterMgr.java	필터 정보를 기록,삭제,읽기를 담당
	FilterServer.java	필터 정보를 관리하는 서버
JMS	Listener.java	JMS 서버 역할로 JMS 메시지를 받아서 처리
	ListenerBody.java	JMS 서버에서 실제 메시지 처리 클래스
	JMSController.java	JMS API를 가지고 있는 JMS 처리클래스
Http	jspform.jsp	JSP 인터페이스 제공 파일
	JSPController.jsp	JSP 인터페이스에서 데이터 받아 필터정보 기록,삭제 담당
SOAP	FilterMgr.jws	FilterMgr.java와 동일하나, SOAP을 위해 약간 변형
	SOAPController.java	SOAP API를 가지고 있는 SOAP처리 클래스

그림 4. XML 필터링 매니저의 클래스 구성

3. 관리 시나리오

3가지 프로토콜, JMS, SOAP, Http 프로토콜로 필터링 관리 시스템이 처리되는 과정을 표현하였다. 각 프로토콜들은 공통의 JMS 컨트롤러 모듈을 통해 각 루틴으로 분리된다. [그림 5]에서처럼 필터링 관리 프로세스를 통해 각 프로토콜별로 필터링 결과를 각 클라이언트에서 엑세스 가능하다.

III. XML 필터링 관리시스템 개발

본 시스템은 Java 1.5.0에서 제작하였고, 운영체제는 Window XP Professonal에서 실행하였다. 그리고 웹서버로 apache-tomcat-5.5.17을 사용하여 OpenJMS 0.7.7을 사용하였다. Axis는 axis-10을 사용하였다.

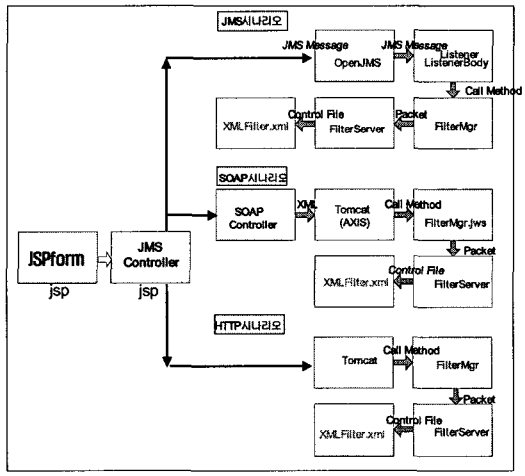

그림 5. XML 필터링 관리 시나리오

1. 실행 환경구축

(1) OpenJMS 설치

JMS를 이용하기 위해서 JMS서비스 제공자인 OpenJMS[6]를 설치해야 한다. 이것은 "http://openjms.sourceforge.net/downloads.html" 주소에서 다운로드를 받고 실행하면 된다. 설정과정은 다음과 같다; 특별히 설치를 위한 특수한 과정은 없고, 다운받은 파일의 압축을 풀고 설정을 하고 실행하면 된다. 다운 받은

OpenJMS파일의 압축을 풀면 bin, db, config, docs, lib, examples 디렉토리가 있다. bin과 config 디렉토리가 OpenJMS의 실행에 있어서 필수적이다.

(2) SOAP 설치

SOAP 설치를 위해, AXIS(Apache eXtensible Interaction System)[7] 공식 다운로드 사이트 http://xml.apache.org/axis/dist/1_0 사이트에서 xml-axis-10.zip 파일을 다운받는다. 설정과정은 다음과 같다; 먼저 다운받은 파일의 압축을 푼다. 그리고 다음 과정에 따라서 설정을 한다[그림 6].

2. 클래스 분석

XML 필터링 매니저는 앞 절에서 언급한 것과 같이 모든 프로토콜에서 사용하는 "공통" 클래스와 JMS를 사용한 인터페이스를 제공하는 "JMS" 클래스와 SOAP를 사용한 인터페이스를 제공하는 "SOAP" 마지막으로 웹을 통해 인터페이스를 제공하는 "HTTP"로 구성되어 있다.

Step1. 압축을 푼 디렉토리의 webapp/axis/WEB-INF/lib 디렉토리에 있는 모든 jar파일들을 아파치가 설치된 디렉토리의 common/lib 디렉토리로 복사한다

Step2. 압축을 푼 디렉토리의 webapp/axis 디렉토리를 아파치가 설치된 디렉토리의 webapps 디렉토리의 아래에 복사한다.

Step3. 아파치가 설치된 디렉토리의 conf/server.xml 파일을 열고 <Context Path="/axis" docBase="axis" debug="0" reloadable="true"> 를 추가한다. – 버전에 따라 생략가능

Step4. 자바 메일 패키지를 다운 받아 javamail-1.4.zip에 있는 mail.jar와 jaf-1_1fr.zip의 activation.jar를 아파치가 설치된 디렉토리의 common/lib 디렉토리에 복사한다(해당 두 zip파일은 시디로제공)

Step5. 아파치를 구동한다

그림 6. AXIS 설정순서

(1) 공통 클래스

공통 클래스는 FilterSet, FilterMgr, FilterServer로 각각 구성되어 있다. 각 클래스의 역할과 실행 및 세부 구현에 대해서 설명 한다.

① FilterSet

FilterSet 클래스는 필터들의 정보를 담는 클래스이다. 이 클래스는 Filter들을 하나의 객체에 저장하여 네트워크로 전송하기 위해 사용한다. 그러므로 등록하고자 하는 필터의 묶음이라고 할 수 있다. 즉, 필터링 기법에서 필터링을 위한 컨텍스트를 정의하고 이를 처리하는 부분이다. FIlterSet 클래스는 메소드는 없이 단순히 멤버들로 만 구성되어 있다. 맴버는 TagID, reader1, readerNum, maxDiscoveryDay…와 같이 등록하고자 하는 filter 규칙을 포함하고 있다. 그래서 사용시는 FIterSet 객체를 생성하고 맴버들을 직접 대입하여 Filter규칙을 설정한다.

② FilterMgr

FilterMgr 클래스는 필터 정보를 기록, 삭제, 읽기를 요청하는 API를 포함하고 있다.

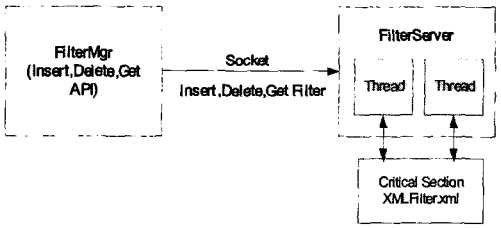

그림 7. FilterMgr과 FilterServer의 구조

[그림 7]과 같이 FilterMgr은 API를 가지고 있는 클래스로 실제 필터 정보를 기록하고 있는 XMLFilter 파일에 접근하는 FilterServer와 통신을 한다. XMLFilter는 여러 개의 파일이 동시에 접근할 수 있으므로 FilterServer라는 클래스가 임계영역을 설정하고, 접근을 제어한다. 그래서 FilterMgr에서 제공하는 함수를 가지고, XMLFilter의 동시 접근에 따른 파일 손상을 고려하지 않고, 간단히 파일에 필터정보 기록/삭제/읽기를 수행 할 수 있다. 다음 [그림 8]은 FilterMgr의 메소드들이다. 위의 함수들을 이용해서 FilterMgr에 필터정보를 설정하고, FM_XMLInsert, FM_XMLDelete, FM_GetXMLFile을 이용해서 필터정보를 제어할 수 있다.

메소드	역 할
setTagID	(필터설정) Tag ID를 설정한다
setReader1	(필터설정) setReader1를 설정한다
setReaderNum	(필터설정) ReaderNum를 설정한다
setMinDiscoveryDay	(필터설정) MinDiscoveryDay 를 설정한다
setMinDiscoveryTime	(필터설정) MinDiscoveryTime 를 설정한다
setMaxDiscoverDay	(필터설정) MaxDiscoverDay 를 설정한다
setMaxDiscoverTime	(필터설정) MaxDiscoverTime 를 설정한다
setAntenna	(필터설정) Antenna 를 설정한다
setRemoteAddr	(필터설정) RemoteAddr 를 설정한다
FM_XMLInsert	(API) 설정된 필터 정보를 기록한다(서버에 기록요청)
FM_XMLDelete	(API) 설정된 필터 정보를 삭제한다(서버에 삭제요청)
FM_GetXMLFile	(API) 필터 정보를 String으로 얻는다(서버에 요청)

그림 8. FilterMgr의 메소드

③ FilterSerever

FilterServer는 필터정보를 저장하는 XMLFilter. xml 파일을 여러 개의 프로세스가 동시에 접근할 때의 파일 손상을 막기 위해서, 소켓을 사용해서 여러 개의 요청을 받아들이고, 파일 접근시 임계영역을 설정해서 동기화를 지원하는 파일관리 서버이다. [그림 7]과 같이 FilterMgr을 이용해서 접속을 하면 쓰레드를 하나 생성한다. 그리고 개별 쓰레드는 삽입요청, 삭제요청, 읽기 요청에 따라 파일에 Lock을 걸고 파일입출력 작업을 수행한다.

(2) JMS 클래스

Java Message Service를 이용한 인터페이스를 제공하기 위한 클래스로 JMSController와 Listener, Listener Body 3개로 구성되어 있다. 본 XML 필터링 매니저에서 JMS 서비스 제공자는 OPENJMS로 사용하였다. [그림 9]을 보면 전체 처리 과정이 나타나 있다. JMSController에서 API를 이용해서 Filter에 대한 연산 요청을 한다. 이 요청은 OPENJMS의 testQueue에 쌓이게 되고, JMS Listener에서 읽어 와서, 앞 절에서 소개한 FilterMgr을 이용해서 실제 XMLFilter. xml파일에 저장하게 된다. 그리고 필터정보 요청연산의 경우 FilterMgr을 통해서 필터정보를 읽어 와서 InterMiddlewareQueue에 쌓게 되고 JMSController에서 최종적으로 String으로 XMLFIlter. xml의 내용을 받을 수 있다.

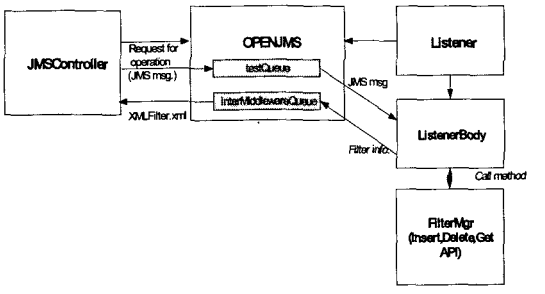

그림 9. JMS 인터페이스

① JMSController

JMSController Class는 JMS 인터페이스를 위해 API를 제공하는 클래스이다[그림 10]. JMSController은 JMS서비스 제공자에 접속을 해서, Filter 쓰기요청, 삭제요청, 읽기 요청을 할 수 있다.

메소드	역 할
JMS_Connect	JMS 서버에 접속한다
JMS_XMLInsert	필터 정보를 저장한다(FilterSet 이용)
JMS_XMLDelete	필터 정보를 삭제한다(FilterSet 이용)
JMS_GetXML	필터 정보를 String으로 얻는다

그림 10. JMSController의 메소드

② Listener

JMS 서비스 제공자에 연결하여 Listener 연결을 생성한다. 그리고 ListerBody를 생성해서 비동기적으로 메시지를 처리한다.

③ ListenerBody

실제적인 메시지 처리 부분 클래스이다. Listener에서 비동기적인 메시지 처리를 위해 해당 클래스를 사용한다. ListnenerBody는 testQueue에서 JMSController가 요청한 메시지를 받아서 메시지 타입에 따라 처리한다. ListenerBody 내부적으로 FilterMgr 클래스 객체를 소유하고, 이를 이용해서 필터 연산 요청을 수행한다.

(3) SOAP 클래스

AXIS 에서 SOAP을 이용하여 Filter를 관리하는 클래스이다. SOAP 클래스는 FilterMgr.jws와 SOAP Controller로 구성되어 있다. [그림 11]은 SOAP을 이용하여 Filter 연산을 수행하는 구조이다. SOAP의 통신 방식인 XML 방식을 이용해서 아파치상에서 구동되는 AXIS에 데이터를 전송한다. 이는 XML로 직접 통신이 가능하고, 제공하는 API로 통신도 가능하다. 그러면 AXIS에 있는 FilterMgr에 연결되어서 처리 된다.

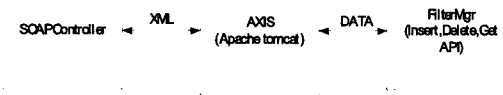

그림 11. SOAP 인터페이스

① SOAPController

SOAPController Class는 SOAP 프로토콜을 이용한 인터페이스를 위해 API를 제공하는 클래스이다. SOAPController을 이용해서 AXIS에 접속을 해서, Filter 쓰기요청, 삭제요청, 읽기 요청을 할 수 있다. 아래 [그림 12]에 제시된 3개의 API를 이용해서 SOAP을 이용한 필터정보 제어를 할 수 있다.

메소드	역 할
SOAP_XMLInsert	필터 정보를 저장한다(FilterSet 이용)
SOAP_XMLDelete	필터 정보를 삭제한다(FilterSet 이용)
SOAP_GetXML	필터 정보를 String으로 얻는다

그림 12. SOAPController의 메소드

② FilterMgr (jws)

FilterMgr(jws) 클래스는 필터 정보를 기록, 삭제, 읽기를 요청하는 API를 포함하고 있다. 이는 FilterMgr (java)와 동일한 역할을 하지만, AXIS 환경을 위해서 약간의 변화가 있다. 그래서 FilterMgr.java 파일과 FilterMgr.jws 파일은 구분된다. 그러므로 설치 시 두 파일을 구분해서 사용해야 한다. 이상을 요약 하면 FilterMgr.java는 HTTP와 JMS를 위해서 사용되고, FilterMgr.jws는 SOAP을 위해서 사용된다.

IV. XML 필터링 관리시스템 API 테스팅

1. JMS

(1) ClassPath의 설정

JMS프로토콜을 사용하기 위해서는 먼저 서버 측에서 FilterServer와 Listener와 OpenJMS가 실행되어야 한다. 그리고 클라이언트 측에서 API(JMSController)를 사용해서 처리할 수 있다.

(2) JMS Listener 실행

JMS Listener가 실행되기 위해서는 먼저 OpenJMS가 실행되어야 한다. OpenJMS의 실행이 성공되면 다음에는 Listener을 실행해야 한다. 이때 Listener를 실행하기 전에 한 가지 설정을 해야 한다. testQueue는 JMS서비스 제공자를 통해서 사용할 메시지 queue로 앞 절의 (2) 설정 부분에서 설정한 Queue이다. 그러므로 JMS 실행 시 꼭 설정되어 있어야 한다. 이는 Config 파일을 변경하여 임의로 바꿀 수 있다.

OpenJMS를 실행하고 jndi.properties을 수정하여 Listener까지 실행이 완료되면 마지막으로 filterServer를 작동해야 한다.

(3) JMS API 사용 예시

위의 과정을 성공적으로 완료 하였으면, 아래와 같은 소스로 JMS를 이용한 필터정보를 관리 할 수 있다. [그림 13]에는 JMS API를 사용하여 필터정보를 관리하는 방법을 보여주고 있다. 먼저 FIlterSet 클래스를 이용해서 필터정보를 등록하는 부분이 있다. 그리고 (1)이라고 표시한 곳에 먼저 JMS에 연결한다. 이때 "testQueue"를 사용하였다. 이는 JMS의 Listener 클래스와 OpenJMS에서 사용하는 testQueue와 이름이 같다.

그리고 (2)에서는 실제 filter 정보를 삽입하는 방법이다. 생성한 filterSet을 보내주면 된다. (3)에서는 (2)와 같은 방법으로 filterSet에 설정된 filter 조건을 삭제한다. 그리고 (4)에는 XMLFIlter.xml에 들어있는 모든 정보를 String으로 불러온다. 그리고 마지막 (5)는 JMS 연결을 종료하는 함수이다. 이와 같이 위의 과정을 이용해서 JMS를 이용해서 필터 조건을 제어할 수 있다.

```
package xml;

import java.net.*;
import java.io.*;

class testJMSAPI
{
        public static void main(String args[])
        {
                JMSController Controller = new JMSController();
                /* 필터 정보 등록 */
                FilterSet filterObj = new FilterSet();
                filterObj.tagID = "0012";
                filterObj.reader1 = "pos";
                filterObj.antenna = "m2";
                String          xmlData = "";
                /* JMS 접속 */
                if( Controller.JMS_Connect("testQueue") ) (1)
                {
                        System.out.println("접속성공");
                }
                else
                {
                        System.out.println("접속실패");
                }
                /* JMS Insert */
                Controller.JMS_XMLInsert(filterObj); (2)
                /* JMS Delete */
                Controller.JMS_XMLDelete(filterObj); (3)
                /* JMS Get XML */
                xmlData = Controller.JMS_GetXML(); (4)

                if( xmlData != null )
                {
                        System.out.println(xmlData);
                }
                else
                {
                        System.out.println("  데  이  터
수신실패");
                }
                /* JMS exit */
                Controller.JMS_Exit(); (5)
        }
}
```

그림 13. JMS API(JMSController이용) 사용 예

2. SOAP

SOAP 프로토콜을 사용하기 위해서는 먼저 서버 측에서 아파치와 AXIS 설정이 완료되고 아파치가 구동되고, axis에 FilterMgr.jws가 복사되어 있어야 한다. 그리고 FilterServer가 작동해야 한다. 그 후 클라이언트 측에서 XML을 이용한 API(SOAPController)를 사용해서 처리 할 수 있다.

(1) FilterMgr.jws의 설치

SOAP을 이용해서 FilterServer에 접근해서 필터정보를 관리하기 위해서는, FilterMgr을 설치하기만 하면 된다. 설치는 다음과 같이 매우 간단하다. 아파치가 설치된 디렉토리의 webapps/axis 디렉토리가 있다.(2-2 SOAP의 설치에서 만들었다) 이곳에 FilterMgr.jws를 복사하면 모든 작업이 완료된다. 위와 같이 FilterMgr.jws을 설치하였으면 다음으로는 filterServer를 작동해야 한다.

(2) SOAP API 사용 예시

위 (1)의 과정을 성공적으로 완료 하였으면, 아래와 같은 소스로 SOAP를 이용한 필터정보를 관리 할 수 있다. 아래 [그림 14]에는 SOAP API를 사용하여 필터정보를 관리하는 방법을 보여주고 있다.

```
package xml;

import java.io.*;

class testSOAPAPI
{
        public static void main (String args[]) {
                FilterSet ObjSet = new FilterSet();
                SOAPController Controller = new SOAPController();
                String xmlData = "";
                ObjSet.tagID = "SOAP";
                ObjSet.reader1 = "Allien";
                ObjSet.readerNum = "3";

                if( Controller.SOAP_XMLInsert(ObjSet) == true )
(1)
                {
                        System.out.println("삽입성공");
                } else
                {
                        System.out.println("삽입실패");
                }

                if( Controller.SOAP_XMLDelete(ObjSet) == true )
(2)
                {
                        System.out.println("삭제성공");
                } else
                {
                        System.out.println("삭제실패");
                }


                if( (xmlData = Controller.SOAP_GetXML()) != null
) (3)
                {
                        System.out.println(xmlData);
                }
                else
                {
        System.out.println("GetXML실패");
                }
        }
}
```

그림 14. SOAP API(SOAPController이용) 사용 예

먼저 FIlterSet 클래스를 이용해서 필터정보를 등록하는 부분이 있다. 그리고 (1)이라고 표시한 곳에 먼저 SOAP으로 필터정보를 삽입하는 방법이 있다. 설정된 FilterSet을 가지고 삽입을 요청한다. (2)에서는 (1)와 같은 방법으로 filterSet에 설정된 filter 조건을 삭제한다. 그리고 (3)에는 XMLFIlter.xml에 들어있는 모든 정보를 String으로 불러온다. 이와 같이 위의 과정을 이용해서 SOAP를 이용해서 필터 조건을 제어할 수 있다.

(3) 클라이언트 연동 화면

[그림 15] 화면은 클라이언트에서 필터링 조건에 따라 미들웨어에서 응용으로 필터링/가공된 데이타 결과

를 전달받은 상황을 보여준다.

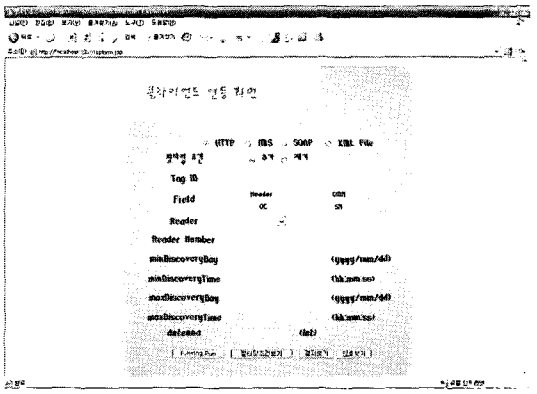

그림 15. 클라이언트 API 설정 및 필터링 조건 입력 화면

[그림 16] 화면은 SOAP 프로토콜을 사용하여 필터링 결과를 서버에 접속하여 클라이언트에서 본 화면이다. 이러한 필터링결과는 위에서 기술된 여러 가지 응용 접근 프로토콜을 사용하여 클라이언트에서 읽어 올 수 있으며, 또한 필터링 결과는 암호화가 가능하여 암호를 입력해야만 클라이언트에서 볼 수 있도록 설정할 수 도 있다.

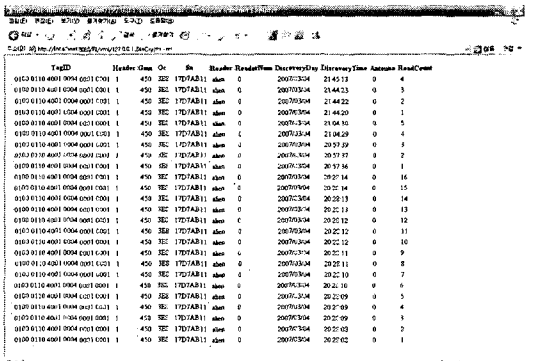

그림 16. 클라이언트로 전송된 필터링 결과 화면

V. 결 론

XML 필터링 매니저(XML Filter Manager)는 기 개발된 RFID 미들웨어에서 사용하는 필터를 관리하기 위한 필터링 엔진 접근 모듈이다. 개발 접근 방법은 HTTP, JMS, SOAP 등 다양한 프로토콜을 지원하여 다양한 플랫폼에서 본 미들웨어의 설정을 접근할 수 있도록 하였다. 그러나 필터를 제어하는 클라이언트가 다수가 되는 환경에서는 하나의 필터링 파일을 사용하면 동기화 문제가 생겨 필터링 처리 시 문제가 야기될 수 있다. 따라서 본 논문은 동기화기능을 지원하면서 필터 파일을 관리하는 프로세스(XML Process)를 하나 두고, 모든 필터의 제어는 이 프로세스를 통하여 작업함으로써 동기화 기능을 갖는 필터링 관리 기법을 설계 및 구현하였다. 구현한 필터링 관리 시스템을 RFID 미들웨어와 클라이언트를 연동하여 실험 및 테스팅을 수행하였으며, 실험결과 RFID 미들웨어와 필터링 관리 프로세스가 적절히 동작함을 확인할 수 있었다.

참 고 문 헌

[1] K. Traub, S. Bent, T. Osinski, S. N. Perertz, S. Rehlinh, S. Rosenthat, and B. Tracey, *The Allication Level Event(ALE) Specification*, ver1.0, 2005.

[2] 홍연미, 조윤상, 변지웅, 노영식, 박상열, 오상현, 변영철, "ALE 기반 RFID 미들웨어 시스템 설계", 한국콘텐츠학회 2006년 추계학술대회논문집, 제4권, 제2호, pp.469-475, 2006.

[3] 이훈순, 최현화, 김병섭, 미명철, 박재홍, 이미영, 김명준, 진성일, "UbiCore : XML 기반 RFID 미들웨어 시스템", 한국정보과학회논문지 : 데이타베이스, 제33권, 제6호, pp.578-589, 2006.

[4] 석수욱, 박재관, 홍봉희, "RFID 미들웨어에서 이벤트 필터링을 위한 질의 색인 기법", 제32회 통신학회추계학술발표회논문집, Vol.32, No.2, 2005.

[5] T. S. Lopez and D. Y. Kim, "A Context Middleware Based on Sensor and RFID Information," Proc. of IEEE PerCom'07, pp.331-336, 2007.

[6] http://openjms. sourceforge.net

[7] http://xml.apache.org/axis/dist/1_0

저 자 소 개

박 병 섭(ByoungSeob Park) 종신회원

- 1989년 2월 : 충북대 컴퓨터공학과(공학사)
- 1992년 2월 : 서강대학교 컴퓨터학과(공학석사)
- 1997년 2월 : 서강대학교 컴퓨터학과(공학박사)
- 1997년 4월 ~ 2000년 2월 : 국방과학연구소 선임연구원
- 2000년 3월 ~ 2002년 8월 : 우석대학교 컴퓨터교육과 교수
- 2002년 9월 ~ 현재 : 인하공업전문대학 컴퓨터시스템과 교수

<관심분야> : RFID/USN, Mobile-IPv6