UbiCore : XML 기반 RFID 미들웨어 시스템

(UbiCore : An Effective XML-based RFID Middleware System)

이 훈 순 [†] 최 현 화 [††] 김 병 섭 [†] 이 명 철 [†††]
(Hun-Soon Lee) (Hyun-Hwa Choi) (Byoung-Seob Kim) (Myung-Cheol Lee)
박 재 홍 [†] 이 미 영 [††] 김 명 준 [†††] 진 성 일 [††††]
(Jae-Hong Park) (Mi-Young Lee) (Myung-Joon Kim) (Sung-Il Jin)

요 약 유비쿼터스 컴퓨팅의 핵심 기술로 주목 받고 있는 RFID (Radio Frequency Identification) 기술의 발달로 인해 이를 이용하여 사람들에게 편의를 제공하는 응용들이 점점 늘어나고 있다. 이러한 응용들을 쉽게 개발하기 위해서는 다양한 센서를 관리하며, 센서를 통해 수집된 태그 데이타 스트림으로부터 의미있는 데이타를 추출하여 응용에 전달해주는 미들웨어 시스템이 필요하다. 본 논문에서는 UbiCore (Ubiquitous Core)라 불리는 XML 기반 RFID 미들웨어 시스템을 제안한다. UbiCore는 다음과 같은 특징을 가진다. 첫째, XQueryStream이라는 XQuery에 기반한 연속 질의 언어를 제공한다. 둘째, 프리 필터링과 중간 결과 재사용을 통해 스트림 데이타에 대한 질의 처리 속도를 향상 시킨다. 셋째, 연속적으로 생성되어 들어오는 실시간 데이타뿐 아니라 트리거에 기반하여 저장된 이력 데이타에 대한 질의를 지원한다. 넷째, 컨텍스트와 서비스의 연계 정보를 표현하기 위한 마크업 언어인 CSML(Context-driven Service Markup Language)을 제공한다.

키워드 : RFID 미들웨어, XML 데이타 스트림, 연속 질의, 컨텍스트-드리븐 서비스, 프리필터링, 트리거, 이력 데이타 질의, ALE

Abstract Owing to the proliferation of Radio Frequency Identification (RFID) technologies which is being watched as a core technology of ubiquitous computing, applications which offer convenience to people using RFID technologies are more and more increased. To easily develop these applications, a middleware system which acts as a bridge between RFID hardware and application is essential. In this paper, we propose a novel XML-based RFID middleware system called UbiCore (Ubiquitous Core). UbiCore has following features: First, UbiCore employs its own query language called XQueryStream (XQuery for Stream Data) which is originated from XQuery. Second, UbiCore has the preprocessing phase called pre-filtering prior to query evaluation and reuses the intermediate result of previous evaluation to speed up the processing of RFID tag data stream. Third, UbiCore supports query on both continuously generated stream data and archived historical data. And last, UbiCore offers a distinct markup language called Context-driven Service Markup Language (CSML) to easily specify the linking information between context and service

Key words : RFID middleware, XML data steam, continuous query, context-driven service, prefiltering, trigger, historical data query, ALE

· 본 연구는 정보통신부의 정보통신 차세대 핵심기술 개발 사업의 "차세대 인터넷 서버 기술 개발" 과제 중 "스마트 객체 처리 프레임워크 기술 개발" 세부 과제로 수행되었습니다. 본 연구의 수행에 참여하고 있는 모든연구원들에게 감사의 뜻을 전합니다.

† 비 회 원 : 한국전자통신연구원 인터넷서버그룹 연구원
hunsoon@etri.re.kr
powerkim@etri.re.kr
jaehong74@etri.re.kr
†† 정 회 원 : 한국전자통신연구원 인터넷서버그룹 연구원
hyunwha@etri.re.kr
mylee@etri.re.kr
††† 종신회원 : 한국전자통신연구원 인터넷서버그룹 연구원
mclee@etri.re.kr
joonkim@etri.re.kr
†††† 종신회원 : 충남대학교 전기정보통신공학부 교수
sijin@cnu.ac.kr

논문접수 : 2006년 8월 3일
심사완료 : 2006년 10월 30일

1. 서 론

최근 활발히 연구가 진행되고 있는 유비쿼터스 컴퓨팅의 핵심 기술로 주목 받고 있는 RFID(Radio Frequency Identification) 기술을 이용하여 편리한 서비스를 제공하는 응용들 - 물류[1], 소매업[2], 의료[3], 공장·가정·사무실 자동화, 보안, 재해·재난 방지, 재산 관리 등 - 이 점차 늘어나고 있다. 유비쿼터스 컴퓨팅 환경에서 센서를 통해 수집되는 정보들은 마치 물이 흐르는 것처럼 끊임없이 생성되며, 데이타 하나의 크기는 작지만 그 양은 굉장히 많고, 데이타의 양이 시간에 따라 변하며 실시간으로 처리되어야 가치가 있는 특징을 가진다. 이와 같은 특성을 가지는 센싱되는 정보로부터 필요한 것을 추출하여 서비스에 이용하기 위해서는 복잡하고 빠른 처리가 필요하다. 따라서 응용 서비스 개발자로 하여금 RFID 기술을 이용한 다양한 유비쿼터스 컴퓨팅 응용 서비스를 쉽게 구축할 수 있도록 하기 위해 RFID 태그가 부착된 객체(이하 스마트 객체)와 응용 서비스의 교량 역할을 해주는 미들웨어(이하 RFID 미들웨어)가 필요하다[4]. 그림 1은 전형적인 RFID 시스템의 구조를 보인다. RFID 시스템은 태그, 안테나, 센서, RFID 미들웨어, 그리고 기존 IT 비즈니스 시스템으로 구성되고, 유무선 통신망에 연동되어 사용된다.

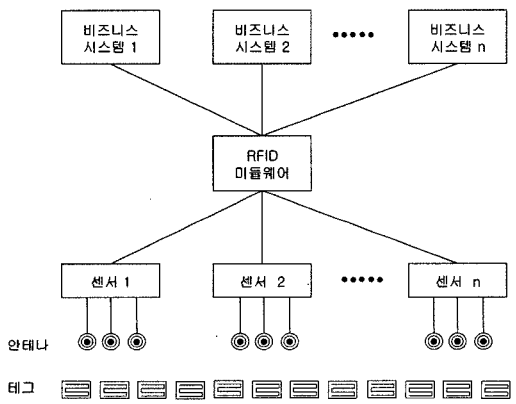

그림 1 RFID 시스템 구조

RFID 미들웨어는 다양한 센서를 관리하며, 센서로부터 다양한 프로토콜을 이용하여 데이타를 수집하고, 또한 수집된 가공되지 않은 데이타로부터 의미있는 정보를 추출하여 응용 서비스인 비즈니스 시스템에 전달한다. 근래에 SUN의 Java System RFID Software[5], Oracle의 Sensor Edge Server[6], CapTech의 TagsWare [7], ConnecTerra의 RFTagAware[8]와 GlobeRalger의 iMotion[9]와 같은 RFID 미들웨어 시스템 제품 및 솔루션들이 많이 개발되어 출시되고 있다. 하지만, 복잡한 비즈니스 로직의 처리나 다양한 형식을 가지는 센싱된 데이타의 처리에 대한 고려가 미진한 상태이다.

본 논문에서는 UbiCore(Ubiquitous Core)라 불리는 RFID 응용을 위한 XML[10]에 기반한 미들웨어(XML-compliant middleware) 시스템을 제안한다. UbiCore는 사용자에 의해 스키마 정의가 명시된 임의의 XML 문서 스트림 데이타를 처리할 수 있을 뿐 아니라 비즈니스 시스템에서 사용되는 임의 XML 형태의 결과를 생성할 수 있는 데이타 변환 기능을 가진다. 따라서 다른 기존 시스템들과 높은 상호 운영성을 가진다.

유비쿼터스 컴퓨팅 환경에서 RFID 응용을 개발하는데 많은 잇점을 제공하는 UbiCore의 주요 특징은 다음과 같이 요약된다:

- 엑스쿼리(XQuery)[11]를 기반으로 RFID 데이타 스트림에 대해 효과적으로 사용자의 관심사항을 표현할 수 있는 기능을 추가한 XQueryStream이라 불리는 연속 질의 언어를 제공한다.
- 이전 질의 평가의 중간 결과를 효과적으로 관리하여 이미 평가된 데이타에 대한 불필요한 질의 조건 평가를 최소화함으로써 윈도우 질의 처리를 가속화시켰다.
- 끊임없이 생성되어 들어오는 스트림 데이타와 저장소에 저장된 이력 데이타를 모두 접근하는 하이브리드 질의(hybrid query)를 지원한다.
- 질의 처리 속도 향상을 위해 불필요한 데이타가 연속 질의의 대상이 되는 것을 방지하는 프리필터링(prefiltering)을 한다.
- 컨텍스트와 서비스 연계 정보 표현을 위해 CSML (Context-driven Service Markup Language)이라는 마크업 언어를 제공한다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구를, 3장에서는 UbiCore라 불리는 XML 기반 RFID 미들웨어 시스템에 대해, 4장에서는 UbiCore의 특징이 되는 내용에 대한 설계를, 그리고 5장에서는 구현 및 실험에 대해 기술한다. 마지막으로 6장에서 결론과 향후 연구에 대해 서술한다.

2. 관련 연구

근래에 몇몇 RFID 기반 미들웨어 제품 및 솔루션들이 개발되어 상품화되고 있다.

SUN의 Java System RFID Software[5]는 EPCglobal의 EPC Network Architecture[12]의 한 구성 요소인 SAVANT 규격을 기반으로 구현된 RFID 기반 소프트웨어(RFID infrastructure software)이다. SUN Java System RFID Software의 구성 요소 중 하나인 이벤트 관리기(event manager)는 다양한 센서로부터 오는 센서 데이타 스트림을 처리하는데, 리더 어댑터

(reader adapter), 필터(filter), 로거(logger), 엔터프라이즈 게이트웨이(enterprise gateway)로 구성된다. 필터들을 서로 연결하여 특정 마스트(mask) 조건을 만족하는 EPC 값을 얻을 수 있으며, 델타(delta)와 스무딩(smoothing) 질의를 할 수 있다. 뿐만 아니라 사용자 정의 필터를 개발할 수 있는 도구를 지원한다. 제공하는 필터는 EPC 태그 데이타와 같이 그 형식이 미리 정해진 것에 대해서만 사용이 가능하다.

Oracle의 Sensor Edge Server[6]는 센서 기반 서비스(sensor-based service)를 지원하기 위한 통합 플랫폼으로 데이타 수집, 이벤트 처리, 데이타 디스패칭 등의 기능을 지원한다. 미리 정의된 단순 필터(Pass-Thru Filter, Palet Pass Thru Filter, Shelf Filter, Check Tag Filter 등)들을 이용하여 센서 데이타로부터 원하는 데이타를 추출하게 하며, 좀 복잡한 로직을 가지는 필터를 사용하기 위해서는 사용자 정의 필터를 개발하여 추가해야 한다.

CapTech 사의 TagsWare[7]는 자바로 개발된 RFID 통합 툴킷으로, RFID 태그 데이타를 센서들로부터 응용에 전달하기 위한 링크(link), RFID 장비로의 표준 인터페이스를 지원하는 드라이버(driver), 그리고 응용에서 링크와 드라이버의 사용을 지원해주는 응용 기반 요소(application infrastructure element)로 구성된다. TagsWare는 RFID 리더로부터 수집된 잡음이 많은 원시 데이타로부터 태그 데이타를 추출하고, 스무딩하여 응용 시스템에 전달하기 위해 링크라는 컴포넌트가 체인으로 연결된 구조를 제공한다. TagsWare에서는 처리 가능한 데이타가 RFID 태그 데이타로만 한정되어 있다.

ConnecTerra의 RFTagAware[8]는 RFID 응용을 개발하고, 배치하고(deploying), 관리하기 위한 광범위한 소프트웨어 기반을 제공한다. 실시간 RFID 태그 데이타에 대해 필터링하기 위해 ALE(application level events) [13] API를 이용한다. ALE API만을 이용하므로 필터 표현 능력이 ALE의 ECSpec 에 의해 제약을 받으며, 처리 가능한 데이타가 EPC 체계를 따르는 데이타로만 한정된다.

RFID 태그 가격의 하락으로 인한 많은 태그의 사용은 네트워크 트래픽을 증가시키고, 결국 비즈니스 시스템을 마비시킬 수 있다. RFID 미들웨어가 데이타의 생성된 곳으로부터 가능한 가까이에서 불필요한 데이타를 제거할 수 있다면 전체 시스템의 성능이 향상될 것이다. 위에서 언급한 미들웨어 시스템들은 태그 데이타를 수집하고 단순 필터를 적용하여 정제한 후에 IT 비즈니스 시스템에 정보를 전달한다. 하지만 제공하는 필터들은 그 표현력이 약해서 사용자들이 그들이 원하는 바를 정확하게 표현할 수 없다. 따라서 복잡한 필터나 비즈니스 로직은 응용 개발자가 개발하여 추가하거나, 응용 프로그램에서 구현을 통해 표현해 주어야 한다. 또한, 센서 혹은 메모리 칩을 탑재한 RFID 태그가 이용되는 RFID 응용에서 태그가 생성하는 데이타는 객체에 대한 식별 정보뿐 아니라 온도, 습도, 가격, 위치, 색깔 등과 같은 특별한 정보를 포함할 수 있다. 이러한 경우에는 센싱되는 태그 데이타 스트림의 형식이 단순 고정된 것이 아니라 복잡 다양하게 된다. 하지만 위에서 살펴본 RFID 미들웨어 시스템들은 데이타 스트림이 복잡하고 다양한 형식을 가질 수 있다는 점을 간과하고 있다.

3. XML 기반 RFID 미들웨어 시스템

3.1 시스템 소개

UbiCore는 유비쿼터스 환경에서 산재해 있는 다양한 센서의 통합 관리, 연속적으로 생성되는 센서 데이타 스트림에 대한 고품질의 처리 및 처리 결과를 IT 비즈니스 시스템에 전달하는 것을 지원하는 XML 기반 RFID 미들웨어이다.

UbiCore는 다음과 같은 기능을 가진다.

- 다양한 센서 연동: UbiCore는 Alien 사의 ALR-9750와 ALR-9780[14], Matrics 사의 AR-400[15], Intermec사의 IF5[16], SAMSYS 사의 MP9320[17], ThingMagic사의 Mecury4[18], 그리고 다른 EPC 호환 RFID 센서들과 연동할 수 있다.
- 다양한 형식의 센서 데이타 스트림 처리 : UbiCore는 사용자로 하여금 XML 스키마[19]를 이용하여 센서 데이타 스트림의 구조를 표현하고, 이에 대한 XQuery에 기반한 연속 질의를 할 수 있게 한다.
- 센서 데이타 스트림에의 준실시간(near-real-time) 처리: 질의 처리 속도 향상을 위해 불필요한 데이타가 연속질의의 대상이 되는 것을 방지하는 프리필터링과 질의 평가시 중간 결과 재사용을 함으로써 입력 스트림 데이타가 준실시간으로 처리가 되도록 한다.
- 컨텍스트 드리븐 서비스 제공: UbiCore는 특정 조건(컨텍스트)이 만족되면 이에 해당되는 특정 서비스가 자동으로 수행되는 컨텍스트 드리븐 서비스를 지원한다. XQueryStream이라는 질의 언어를 이용하여 컨텍스트를 표현한다. XQueryStream은 단순 필터링 조건뿐 아니라 복잡한 비즈니스 로직까지도 표현할 수 있다. 또한 실시간 입력 스트림과 저장 데이타를 이용한 통합 질의를 지원한다. UbiCore에서 컨텍스트 조건이 만족될 때 호출할 수 있는 서비스로는 Web Services, Java Message Services, HTTP services가 있다. 컨텍스트와 서비스와의 연계 정보는 CSML 이라는 마크업 언어를 이용하여 표현된다.
- 센서 데이타 저장 관리: UbiCore는 의미있는 XML

태그 스트림 데이타를 관계형 데이타베이스 관리 시스템(Relational Database management System, 이하 RDBMS)에 저장하는 것을 지원한다. 또한, 저장된 데이타를 XQJ(XQuery API for Java)[20]와 XQuery를 이용하여 검색할 수 있게 한다.

- 센서 데이타 접근 관리: UbiCore는 태그 데이타 스트림에 대해 XML 구조와 값에 기반한 접근 제어를 한다.
- 자바 API 제공: UbiCore는 네이티브 자바 인터페이스와 ALE[13] 표준 인터페이스를 응용 개발자에게 제공한다.

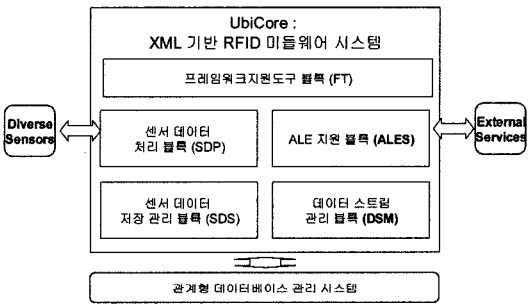

그림 2 UbiCore의 시스템 구성 요소

3.2 시스템 구성 요소

그림 2에 보이는 것처럼, UbiCore는 다음과 같은 구성요소를 가진다.

- 프레임워크 지원 도구 블록(Framework Toolkit Block, 이하 FT)은 사용자가 UbiCore를 쉽게 사용할 수 있게 해주는 GUI 기반의 도구이다. 또한 사용자는 FT를 통해 자원(CPU, memory 등)에 대한 모니터링을 할 수 있다.
- ALE 지원 블록(ALE Supporting Block, 이하 ALES)은 사용자가 EPC 데이타를 처리하는 미들웨어의 표준 인터페이스인 ALE 표준 규격[13]을 이용하여 UbiCore를 사용할 수 있도록 지원한다.
- 센서 데이타 처리 블록(Sensor Data Processing Block, 이하 SDP)은 사용자로 하여금 UbiCore에 접근하여 사용할 수 있게 하는 API를 제공한다. SDP는 다양한 센서로부터 센싱된 데이타를 처리하기 위해 연결과 센서 관리를 한다. 사용자로 하여금 센서들을 그룹핑하여 쉽게 관리할 수 있게 하는 센서 그룹과 여러 센서가 센싱한 동일한 형식을 가지는 데이타를 하나의 센서가 센싱한 데이타로 여기게 해주는 데이타 소스와 같은 논리 센서 개념이 편의를 위해 이용된다. SDP는 또한 컨텍스트 드리븐 서비스를 지원하기 위해 CSML을 처리한다.
- 데이타 스트림 관리 블록(Data Stream Management Block, 이하 DSM)은 XML 스키마로 정의된 형식을 가지는 거대한 XML 데이타 스트림에 대해 준실시간 필터링과 검색 기능을 지원한다.
- 센서 데이타 저장 관리 블록(Sensor Data Storage Management Block, 이하 SDS)은 필터링된 XML 데이타를 관계형 테이블로 변환하여 RDBMS에 저장하여 관리한다. 사용자가 하이브리드 질의를 사용할 수 있도록 하기 위해 SDS는 XQuery를 SQL로 변환하여 RDBMS에서 수행시키고, 관계형 결과를 XML 형태로 변환한다. 또한 SDS는 하이브리드 질의 처리를 위해 XML 트리거를 지원한다. 뿐만 아니라 SDS는 RDBMS를 이용하여 UbiCore에서 다루어지는 메타데이타를 관리한다.

3.3 내부 처리 흐름

센서 데이타를 위한 UbiCore의 내부 처리 흐름은 그림 3과 같다. 안테나와 센서를 통해 센싱되고 수집된 태그 데이타는 메시지로 변환되어 해당 센서와 연결된 RFID 미들웨어인 UbiCore에 전달된다. UbiCore에서는 메시지의 도착을 기다리고 있던 센서 어뎁터가 메시지를 받아서 UbiCore의 내부 형식으로 바꾸어서 입력 큐에 넣는다. 프리필터가 정의되어 있으면 연속 질의 평가의 대상이 될 수 없는 태그 데이타가 제거된다. 그런 이후에 연속 질의 처리 프로세서가 센싱되어 정제된 태그 데이타로부터 사용자 관심을 만족시키는 데이타를 추출한다. 추출된 데이타는 사용자에 의해 XQueryStream 질의문에 명시된 응용에 특화된 형식으로 변환되어 질의 결과 큐에 삽입된다. 서비스 호출자가 결과 큐로부터 데이타를 꺼내서 이를 인자로 사용하여 외부 비즈니스 서비스를 호출하거나 나중에 이용할 용도로 RDBMS에 저장한다.

4. UbiCore 주요 설계

본 장에서는 XML 기반 RFID 미들웨어 시스템으로써 UbiCore의 특징이 되는 설계 내용에 대해 기술한다.

4.1 연속 질의 언어 : XQueryStream

UbiCore에서는 XML 형태의 스트림 데이타로부터 사용자 관심 사항을 찾는 연속 질의를 표현하기 위해 XQuery를 확장한 질의 언어인 XQueryStream (XQuery for Stream Data)을 고안하여 사용한다. XQuery-Stream은 끊임없이 생성되어 들어오는 센서 데이타 스트림에 대해 질의 대상을 한정하기 위해 시간과 이벤트 수에 기반한 윈도우 정의가 가능하도록 했으며, RFID 태그 데이타 스트림에 대한 질의를 효과적으로 지원하기 위해 구조적 동일성에 기반한 집합 연산자인 `unionS`, `intersectS`와 `exceptS`, 시간 순서 함수인 `before()`와 `after()`, EPC 필드 추출 함수인 `epc-field()`, 그리고

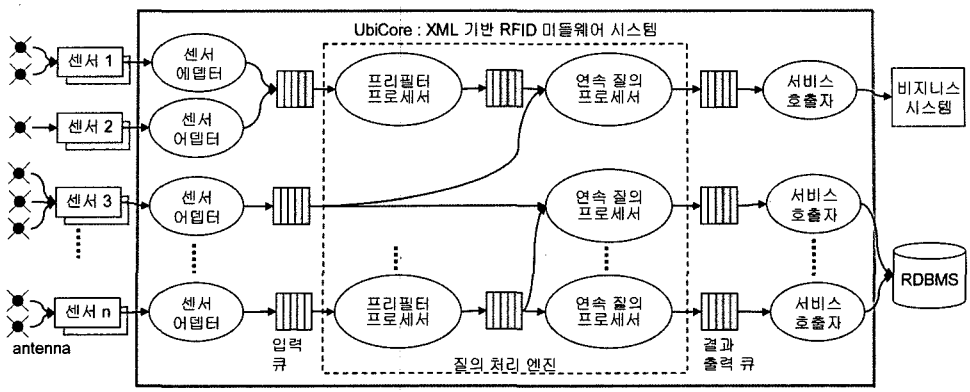

그림 3 UbiCore에서 센서 데이타에 대한 내부 처리 흐름

트리거 함수인 trigger() 등을 추가했다.

4.1.1 질의 규격

그림 4는 EBNF(Extended Backus-Naur Form)[21]로 표현된 XQueryStream 문법의 일부이다. XQueryStream 질의문은 질의 대상을 정의하는 부분 <QueryTarget>과 질의 조건을 나타내는 부분 <QueryBody>으로 구성된다. 사용자는 질의 대상 정의 부분 <QueryTarget>을 이용하여 질의 수행의 입력이 되는 데이타를 정의한다. 질의 조건 정의 부분 <QueryBody>은 XQuery 규격[11]을 따른다. <QueryTarget>은 "using" 이후에 소스 정의 부분 <SourceDefinition>이 하나 이상 온다. <SourceDefinition>는 소스 이름 <SourceName>, 소스 변수 이름 <SourceVariable>, 윈도우 정의 부분 <WindowDefinition>으로 구성된다. 만약 윈도우 정의가 명시되지 않은 경우에는 하나의 이벤트가 윈도우에 포함되는 디폴트 윈도우가 정의된 것으로 간주한다. 윈도우 정의 부분 <WindowDefinition>은 주어진 스트림에서 윈도우 영역 <WindowRange>과 건너뛰기 길이 <TumblingLength>, 그리고 윈도우 유형 <Unit> 정보를 가진다. XQueryStream을 이용하여 슬라이딩 윈도우를 포함하는 질의와 랜드마크 윈도우[22]를 포함하는 질의를 표현할 수 있다. 만약 <WindowRange>의 <From> 값이 -1 인 경우에는 랜드마크 윈도우임을 나타낸다. 윈도우는 주어진 기간인 <TumblingLength> 간격으로 계속 반복적으로 설정된다. <WindowRange>와 <TumblingLength>는 <Unit>에 설정된 값인 시간 혹은 사건에 기반하여 해석된다.

겹침이 있는 슬라이딩 윈도우를 이용하는 간단한 질의 예는 <예제-1>과 같다.

<예제-1 : 슬라이딩 윈도우 질의> 매 5분마다 지난 10분간 출입구(Gate1)을 지나간 사람 수를 출력하라.

```
using "Gate1" as $src within (0 to 10, 5, min)
<NumberOfVisitor> { count($src/tag) }
 </NumberOfVisitor>
```

4.1.2 확장된 주요 연산자와 함수

구조적 동일성에 기반한 집합 연산자

RF 센서의 센싱 영역 제약과 불안정성으로 인해 RFID 응용 개발자들은 특정 영역에서 데이타를 수집하기 위해 센서의 센싱 영역이 겹치도록 다수의 센서를 배치하여 다수의 센서로부터 수집된 태그 데이타를 취합하기를 원한다. 이러한 환경에서는 일부 태그 데이타들은 다수의 센서에 의해 중복 센싱되었을 것이므로 미들웨어가 응용에 전달하기 전에 중복되는 데이타를 제거해 주어야 한다. 만약 수집된 XML 형태를 가진 태그 데이타에 대해 노드 식별자에 기반한 집합 연산자를 이용하여 취합한다면, 동일한 태그 데이타 값을 가지더라도 노드 식별자가 다르므로 서로 다른 데이타로 간주되

```
<Query>             ::= <QueryTarget> <QueryBody>
<QueryTarget>       ::= "using" <SourceDefinition> ( "," <SourceDefinition> ) *
<SourceDefinition>  ::= <SourceName> "as" <SourceVariable>  (<WindowDefinition>
<WindowDefinition>  ::= "within" "(" <WindowRange> ( "," <TumblingLength> )?
                            ( <Unit> ) ? ")"
<WindowRange>       ::= <from> "to" <to>
<Unit>              ::= "event" | "min" | "sec" | "msec
<QueryBody>         ::= .. .. ..
```

그림 4 EBNF로 표현된 XQueryStream 문법의 일부

어 중복 데이타를 제거할 수 없을 것이다. 이러한 이유로 XQueryStream에서는 구조적 동일성에 기반하여 데이타의 중복 여부를 확인하는 집합 연산자인 unionS, intersectS, exceptS를 지원한다. 구조적으로 동일하다는 것은 노드 식별자는 달라도 현재 노드를 포함하여 자손 노드들이 모두 동일한 구조를 가지며 그 값이 같은 경우를 의미한다. 이들 연산자는 RFID 시스템에서 자주 사용되는 델타(delta)와 스무딩(smoothing)을 하는데 적합하다. 많은 RFID 응용에서 태그가 사라지거나 새로이 출현한 것을 체크하는데 유용한 델타 질의의 예는 <예제-2>와 같다.

<예제-2 : 슬라이딩 윈도우에서 델타 질의> 매 10분마다 선반에서 사라진 아이템을 반환하라.

```
using "ShelfStream" as $src_before within
      (0 to 10, 10 min),
     "ShelfStream" as $src_after within
      (10 to 20, 10 min)
let $bf := distinct-values($src_before/
           Observation/tag)
let $af := distinct-values($src_after/
           Observation/tag)
return
  <disappeared>
    { $bf exceptS $af }
  </disappeared>
```

시간 순서 함수

RFID 응용에서는 이벤트의 발생 순서나 그 간격이 중요한 경우가 있다. 이벤트의 순서에 기반한 질의를 지원하기 위해 시간 순서 함수인 before()와 after()가 추가되었다. 시간 순서 함수는 기준이 되는 이벤트와 시간 순서 간격에 대한 정보 - 예, 3초 뒤, 2개 사건 앞 등 - 를 인자로 받아서 시간 순서 조건을 만족하는 이벤트를 반환한다.

예를 들어, 물류 창고에 상품이 들어온 후에 분류하는데 5초가 걸린다고 하자. 만약 5초가 지났는데도 불구하고 동일한 상품이 물류 창고의 하적장에 있으면 물류 분류 시스템이 오동작을 일으키는 것으로 볼 수 있다. 물류 응용에서 이러한 경우를 탐지하는데 <예제-3>과 같은 질의가 이용될 수 있다.

<예제-3 : 시간 순서 함수를 가지는 질의> 동일한 물건이 동일한 장소에서 처음 보인 이후에 5초가 지나도 계속 보이는 경우에 물건의 정보를 반환하라.

```
using "ProductStream" as $src within ( 0
      to 15, 5 sec)
for $tg in $src/Observation/Tag
where after($tg, over, 5, sec)/Observation
      /Tag = $tg
return
   <SomeThingWrong>
      { $tg }
   </SomeThingWrong>
```

EPC 필드 추출 함수

EPC는 상품의 식별자로 이용되는 코드로 RFID 태그에 기록될 수 있다. EPC 값은 제조사, 상품 이름, 시리얼 번호 등의 상품에 대한 정보를 가지는 필드들로 구성되어있는데, 그 코드 체계에 따라 필드의 위치, 크기, 그리고 의미하는 바가 다르다[23]. EPC가 기록된 태그를 이용한 서비스를 제공하는 유비쿼터스 응용에서는 EPC 중 특정 필드 값을 알아야 하는 경우가 빈번하게 발생한다. UbiCore에서는 EPC 데이타를 쉽게 다룰 수 있도록 하기 위해 EPC 필드 추출 함수 epc-field()를 지원한다. epc-field() 함수는 EPC 값과 필드 이름을 인자로 받아서 주어진 EPC 값으로부터 특정 필드에 해당하는 값을 추출해낸다. <예제-4>는 EPC 필드 추출 함수를 이용하는 질의 예이다.

<예제-4 : EPC 필드 추출 함수 epc-field()를 이용한 질의> 매장에서 삼성 전자 (CompanyPrefix=5)가 생산한 냉장고(ItemReference=123)가 있는 곳의 위치를 반환하라.

```
using "ShelfStream" as $src
for $obs in $src/Observation
where epc-field($obs/Tag, CompanyPrefix)
      ="5"
and epc-field($obs/Tag, ItemReference)
    ="123"
return
   <SamsungRefrigerator>
     { $obs/Location}
    </SamsungRefrigerator >
```

트리거 함수

실시간 스트림 데이타와 과거 이력 데이타 모두를 대상으로 질의하는 하이브리드 질의를 지원하기 위해서는 질의 대상이 되는 스트림 데이타뿐 아니라 이력 데이타를 지칭할 수 있어야 한다. 이를 위해 trigger() 함수를 지원한다. trigger() 함수는 트리거 이름을 인자로 받아들여 이력 데이타인 RDBMS에서 수행된 트리거의

결과들 중 현재 질의와 연관있는 이력 데이타를 구별한다. 하이브리드 질의 관련 좀 더 상세한 내용은 4.3에서 설명한다. <예제-5>는 trigger() 함수를 이용하는 질의 예이다.

<예제-5 : trigger() 함수를 이용한 질의> 고객들의 평균 구매액보다 20%를 초과하여 구매하는 고객이 있으면 고객 식별자 정보를 반환하라.

```
using "PurchaseListStream" as as $src
for $cur in $src/Customer
for $avg in trigger(AveragePurchase)
where $cur/TotalPrice >= $avg + $avg *0.2
return $cur/ID
```

4.2 간단하고 효과적인 윈도우 질의 처리 가속화 방법

겹침이 있는 슬라이딩 윈도우를 가진 연속 질의에서 겹치는 영역에 있는 데이타는 여러 번 질의 수행의 대상에 포함될 수 있다. 그림 5는 윈도우 바인딩의 예를 나타내는데 구간 (t+4, t+5) 사이에 들어오는 데이타는 WB1, WB2, WB3, WB4, WB5의 5번 윈도우 평가 결과에 포함되어 질의 평가에 이용된다. 동일한 데이타에 대해 동일한 질의를 여러 번 평가하는 것은 컴퓨팅 자원의 낭비이므로, 이에 대한 특별한 고려가 필요하다.

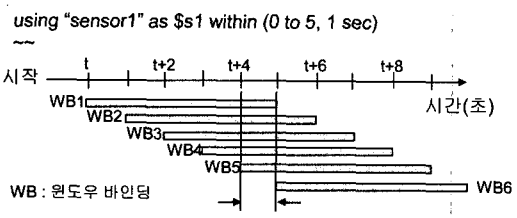

그림 5 윈도우 바인딩의 예

UbiCore의 질의 처리 엔진(XQueryStream Processing Engine)에서는 윈도우 연산을 포함하는 연속 질의 수행 시에 이전 평가의 중간 결과를 저장하여 다음 번 평가 시에 이용함으로써 이미 평가된 데이타에 대해 불필요한 질의 평가를 줄인다. 이를 통해 전체 시스템의 성능 향상 효과를 얻을 수 있다.

그림 6은 UbiCore에서 윈도우 연산을 포함하는 연속 질의 처리시 이전 질의 평가 중간 결과를 이용하는 연속 질의 처리 절차이다. 입력 큐에 있는 센서가 센싱한 데이타에 대해 주어진 윈도우 정의를 만족하는 스트림 데이타를 바인딩하는 윈도우 평가가 이루어진다. 윈도우 평가 결과는 센서가 센싱하여 전달해준 입력 데이타 중 이번에 질의 조건 평가의 대상이 되는 데이타이다. 윈도우 평가 결과에 대해 윈도우 정의를 이용하여 이전 윈도우 평가 결과에 포함되었던 것과 새로이 이번에 윈도우 평가 결과에 포함된 것의 두 부분으로 나눈다. 시간에 기반한 윈도우의 경우에는 이번 질의의 윈도우 종료 시간에서 건너띄기만큼을 뺀 시간 이후에 들어온 데이타가 새로이 들어온 데이타가 되고, 이벤트에 기반한 윈도우의 경우에는 최근에 들어온 건너띄기만큼의 데이타가 새로이 들어온 데이타이다. 데이타 분리 결과 이전 윈도우 평가 결과에 포함되었던 데이타에 대해서는 반복적으로 수행해야 하는 질의 조건 평가 중 일부를 수행하지 않도록 하기 위해 이전 윈도우 평가 결과에 대한 질의 조건 중간 평가 결과로부터 질의 중간 평가 결과를 추출한다. 새로이 윈도우 평가 결과에 포함된 데이타와 추출한 이전 질의 중간 평가 결과를 이용하여 질의 조건을 평가하여 새로운 결과를 생성한다. 이때 다음 번 질의 평가시 이용할 목적으로 이번 질의 조건 평가에 사용된 질의 중간 평가 결과 - 예, 경로식 평가 결과, 집합 함수 수행 결과 등 - 를 메모리 상에 저장하여 기억한다. 또한 생성된 새로운 질의 처리 결과는 출력 큐에 넣는다. 연속 질의이므로 윈도우 평가부터 결과를 구성해서 출력 큐에 넣는 과정이 연속 질의가 철회될 때까지 반복 수행된다.

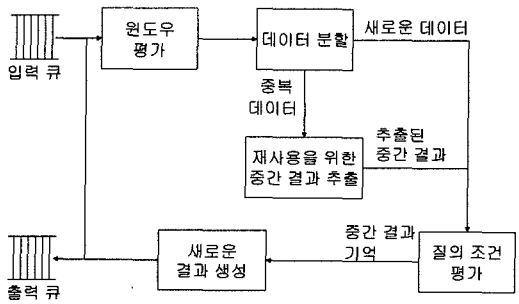

그림 6 윈도우 연속 질의 처리 절차

4.3 하이브리드 질의

근래에 많은 응용들에서 연속적으로 생성되어 들어오는 스트림 데이타와 RDBMS에 저장되어 있는 이력 데이타 모두에 대한 질의가 실시간으로 처리되는 것을 필요로 한다. 이러한 질의를 하이브리드 질의라고 한다. 이와 같은 하이브리드 질의를 다루기 위해서는 시스템이 데이타베이스에 있는 영구(persistent) 데이타와 유무선을 통해 흘러가는 임시(transient) 데이타에 대한 통합 질의 처리를 지원할 수 있어야 한다. 하이브리드 질의를 지원하는데 있어 중요한 문제점은 저장소에 있는 이력 데이타를 꺼내오는데 소요되는 I/O 비용이다. 컴퓨팅 능력과 네트워크 대역폭이 아무리 증가한다고 해도 이력 데이타에 접근하는 비용과 네트워크에 흘러

가는 데이타를 접근하는데 드는 비용의 차이는 무시할 수 없다. 이러한 I/O 비용은 다수의 하이브리드 질의를 수행하는 질의 처리 엔진의 능력을 감소시키는 효과를 나타낼 수 있다. I/O 비용을 최소화하기 위해 저장되어 있는 이력 데이타에 대한 수정없이 평가 횟수와 질의를 위해 접근하는 이력 데이타의 크기를 줄이는 것이 필요하다. 이러한 문제를 해결하기 위해 UbiCore에서는 XML 트리거를 이용하여 하이브리드 질의 중 이력 질의 부분을 미리 독립적으로 수행하여 그 결과를 캐싱해 놓고 이에 대해 질의를 한다. XML 트리거는 연관된 이력 데이타가 변경될 때마다 캐싱된 결과를 변경한다. 그림 7은 하이브리드 질의 처리의 개념을 설명하고 있다. XML 트리거는 이력 데이타가 저장된 RDBMS에서 SQL 트리거로 변경된다. RDBMS로부터 추출된 SQL 트리거의 결과는 XML 문서 형태로 변환된 후 연속 질의 실행 공간에 캐싱되어 통합 질의 처리에 이용되어야 한다.

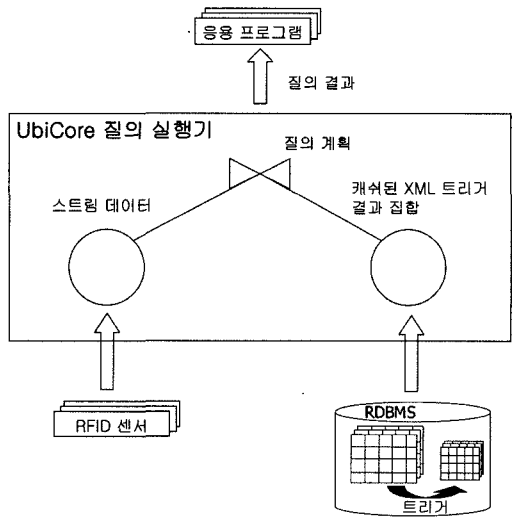

그림 7 하이브리드 질의처리 개념도

4.4 프리필터링

센싱 기술이 발달함에 따라 최근에 개발된 'smart' 센서들은 하드웨어적으로 특정 조건을 만족하는 데이타만을 수집할 수 있으며, 필터링, 스무딩, 리포팅 등의 기능을 제공한다[24]. 하지만 모든 센서가 이러한 기능을 가진 것은 아니다. 어떤 태그들이 센서의 RF 영역에 있는지에 대해 리포트만 하는 'dumb' 센서들도 있다. 만약 응용에 사용된 모든 센서들이 이러한 필터링 기능을 가지고 있는 것이 아니라면, 응용 개발자는 항상 센서 자체에 필터링 기능이 없는 'dumb' 센서들로부터 생성되는 스트림 데이타로부터 관심 사항을 만족하는 것을 검색하기 위한 필터링 조건을 포함하는 연속 질의문을 작성해야 한다. 하나의 센서에서 센싱한 데이타가 하나 이상의 연속 질의에 스트림 데이타 형태로 입력되어 처리되는 환경에서, 센서가 센싱한 데이타 중 연속 질의의 대상이 될 수 없는 일부 데이타가 연속 질의에 전달되기 전에 걸러진다면 연속 질의 처리를 위해 해야 하는 일의 양이 줄어들 것이다. 이러한 이유로 UbiCore에서는 센서로부터 얻어지는 스트림 데이타 중 불필요한 데이타가 연속 질의의 대상이 되는 것을 방지하기 위해 연속 질의를 평가하기 전에 소프트웨어 필터를 통해 전처리하는 단계를 가진다. UbiCore에서 연속 질의 대상이 될 수 없는 데이타를 미리 제거하기 위해 제거될 데이타의 특성을 표현한 것을 프리필터라 한다. UbiCore에서 프리필터는 연속 질의 언어인 XQueryStream의 문법(Syntax)을 빌려서 사용하고 그 의미(semantic)를 달리 해석한다. 질의문이 연속 질의로 사용되면 질의 조건을 만족하는 것이 결과로 반환되지만, 질의문이 프리필터로 사용되면 질의 조건을 만족하는 것들이 제거(filter-out) 된다. 프리필터링을 위한 질의는 스트림 형태로 입력되는 데이타의 형식을 정의한 스키마 - 데이타 소스 스키마 - 에 타당한(valid) 문서를 생성해야 한다는 제약이 있다. 그림 3에서 보이는 것처럼 프리필터링 단계를 가지는 질의 처리 엔진은 프리필터 프로세서와 연속 질의 프로세서로 이루어져 있다. 프리필터 프로세서는 연속 질의의 대상이 되지 않는 데이타들을 제거함으로써 스트림 데이타를 정제한다. 연속 질의 프로세서는 프리 필터를 통해 정제된 스트림 데이타로부터 사용자의 관심사항을 추출하여, 사용자가 원하는 형태로 가공하여 질의 결과 큐에 전달한다.

하나의 데이타 소스는 다수의 연속 질의의 입력이 될 수 있다. 하지만 단지 하나의 프리필터만을 가진다. 프리필터링은 하나의 연속 질의 처리 관점에서 보면 그 효과가 작을(미미할) 수 있다. 하지만 연속 질의의 입력 데이타가 감소되게 되고, 이 데이타는 여러(수십 혹은 수백) 개의 연속 질의의 입력으로 사용될 수 있으므로 시스템 전체 측면에서는 커다란 성능 향상이 있을 수 있다.

4.5 컨텍스트-서비스 연계 표현 : CSML(Context-driven Service Markup Language)

UbiCore에서는 RFID 응용 개발자에게 개발한 응용 서비스를 특정 컨텍스트와 연계시켜서 자동으로 구동할 수 있는 기능을 제공한다. 즉, 응용 개발자는 응용 서비스를 개발하고 이 서비스가 수행되어야 하는 상황을 XQueryStream으로 표현한 후에 서비스와 상황 연계 정보를 미들웨어 UbiCore에 등록한다. 그러면 UbiCore가 특정 상황이 만족되었을 때 해당 서비스를 자동으로

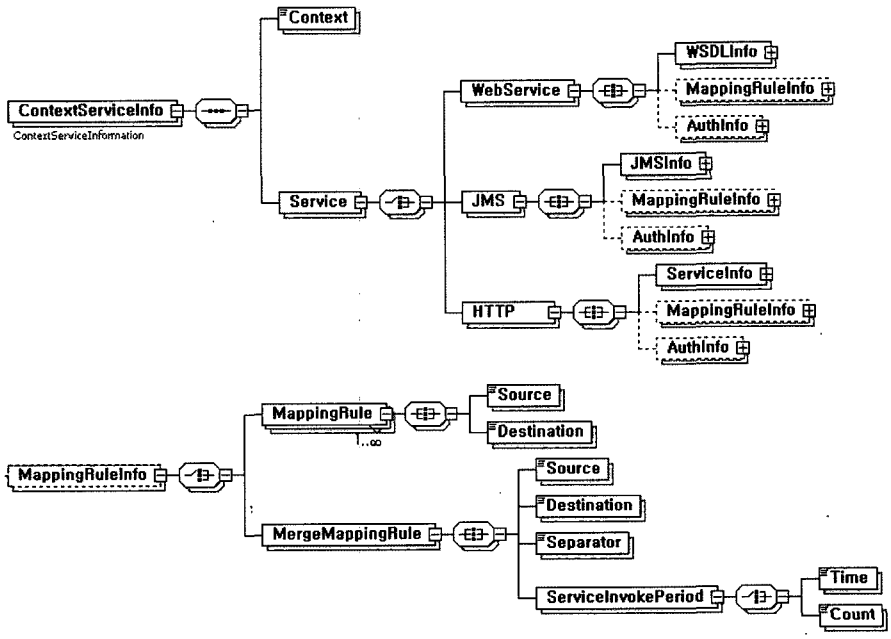

그림 8 컨텍스트와 서비스 연계 정보 구조

수행시킨다.

컨텍스트와 서비스 연계 정보는 서비스가 수행되어야 하는 상황(context), 특정 상황에 연계되어 수행 되어야 하는 서비스, 컨텍스트와 서비스를 연계하기 위한 매핑 정보, 서비스 호출 주기 및 권한 정보로 구성된다. UbiCore 사용자는 특정 조건과 응용 서비스간의 연계 정보를 CSML 이라는 마크업 언어를 이용하여 쉽게 표현할 수 있다. 컨텍스트와 서비스의 연계 정보를 표현하는 CSML은 XML 형태의 마크업 언어로 그림 8과 같은 구조를 가진다.

CSML에서 서비스를 제공해야 하는 상황인지를 인식하기 위한 것뿐 아니라 서비스 제공을 위한 입력으로 필요한 동적인 값들을 얻어오는 것을 하나의 컨텍스트(상황) 정보 <Context>로 표현한다. 이는 UbiCore에서 연속 질의를 나타내기 위한 질의 언어인 XQuery-Stream을 이용하여 표현된다. 서비스 정보 <Service>는 명시된 컨텍스트가 만족되었을 경우에 호출할 서비스에 대한 정보를 표현한다. 현재 UbiCore에서 이용할 수 있는 연동 서비스는 Web Services, Java Messaging Services, HTTP Services가 있다. 컨텍스트와 서비스를 연계하기 위한 매핑정보 <MappingRuleInfo>는 컨텍스트의 결과로부터 서비스 호출을 위한 인자를 만들기 위한 규칙 정보로 <MappingRule>과 <MergeMappingRule> 중 하나로 표현된다.

CSML의 한 예가 그림 9에 나타나 있다. 예제 CSML은 데이타 소스 이름이 ShelfSensor인 센서들로부터 연속하여 입력되는 XML 문서 스트림 데이타에 Sensor/Observation 값이 들어오는 상황이 발생하면 Observation 내에 들어있는 Tag의 값을 추출하여 결과를 만들고, 그 결과를 http://129.254.175.223:8080/Bean1.wsdl에 기술되어 있는 웹서비스 중에서 서비스 이름이 ProductMgmt이고 포트 이름이 PortName1이며 오퍼레이션 이름이 makeProductList인 서비스를 호출하라는 것이다. 서비스 호출시 질의 결과를 makeProductList의 products라는 인자의 입력으로 사용한다.

5. 구현 및 실험

UbiCore의 프로토타입을 J2SE 5.0 JDK 환경에서 JAVA 로 구현했다. XML 형태의 스트림 데이타으로부터 사용자 관심 사항을 추출하기 위해 DOM(document object model) 파싱 후 질의 처리에 이용했다. J2SE 5.0 JDK 에서 제공하는 DOM 패키지는 다중 쓰레드 환경에서 안전(multi-thread safe)하지 않아서 다중 쓰레드 환경에서 안전한 DOM 을 자체 구현하여 이용했다. 현재 4장에서 설명한 윈도우 질의 평가 가속화 부분과 하이브리드 질의 부분은 구현이 완료되지 않은 상태이다.

UbiCore의 성능을 계량화를 위한 척도로 초당 처리량과 응답시간을 이용했다. 응답 시간은 데이타가 UbiCore에 들어온 시간과 서비스에 전달된 시간 사이의

```
<?xml version="1.0" encoding="UTF-8"?>
<ContextServiceInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="csml.xsd" Name="">
  <Context><![CDATA[using "ShelfSensor" as $sour1
          for $b in $sour1//Sensor/Observation
           return <RESULT> {$b//Tag} </RESULT>]]>
  </Context>
  <Service>
    <WebService>
      <WSDLInfo>
        <WSDL-URL>http://129.254.175.223:8080/Bean1.wsdl</WSDL-URL>
        <ServiceName>ProductMgmt</ServiceName>
        <PortName>PortName1</PortName>
        <OperationName>makeProductList</OperationName>
      </WSDLInfo>
      <MappingRuleInfo>
        <MappingRule>
          <Source Value="/" Kind="dynamic"/>
          <Destination Name="products" Type="string"/>
        </MappingRule>
      </MappingRuleInfo>
    </WebService>
  </Service>
</ContextServiceInfo>
```

그림 9 CSML 예

시간차를 이용해서 측정했다.

UbiCore의 성능 실험 환경은 그림 10에서 보이는 것처럼 네 가지 컴포넌트 즉, 리더 에뮬레이터, UbiCore 미들웨어, 주기억 장치 상주 데이타베이스 시스템(main memory resident DBMS, 이하 MMDBMS), 성능 모니터로 구성된다. 리더 에뮬레이터는 다수의 RFID 리더 장비를 에뮬레이션하여 SGTIN-64 형식의 EPC 데이타를 가진 스트림 데이타를 UbiCore 미들웨어로 전송한다. UbiCore 미들웨어는 태그 스트림 데이타 중 컨텍스트를 만족하는 데이타와 성능 관련 시간 정보인 태그가 UbiCore에 들어온 시간과 서비스에 전달되기 바로 직전의 시간을 MMDBMS에 저장한다. 성능 모니터는 MMDBMS에 성능치 수집을 위한 질의를 수행하여 성능 측정치를 계산하여 화면에 출력한다.

성능 측정을 위해 조건을 만족하는 데이타를 MMDBMS에 로깅하는 컨텍스트-서비스를 200개 등록해 놓고, 리더 에뮬레이터를 이용하여 200개의 가상의 리더를 엔진에 연결시킨 후에 각 리더가 초당 생성하는 태그 데이타 수를 변경하면서 실험했다. 컨텍스트로는 80%, 85%, 90%, 95% 필터링을 하는 질의를 각각 50개씩 사용했

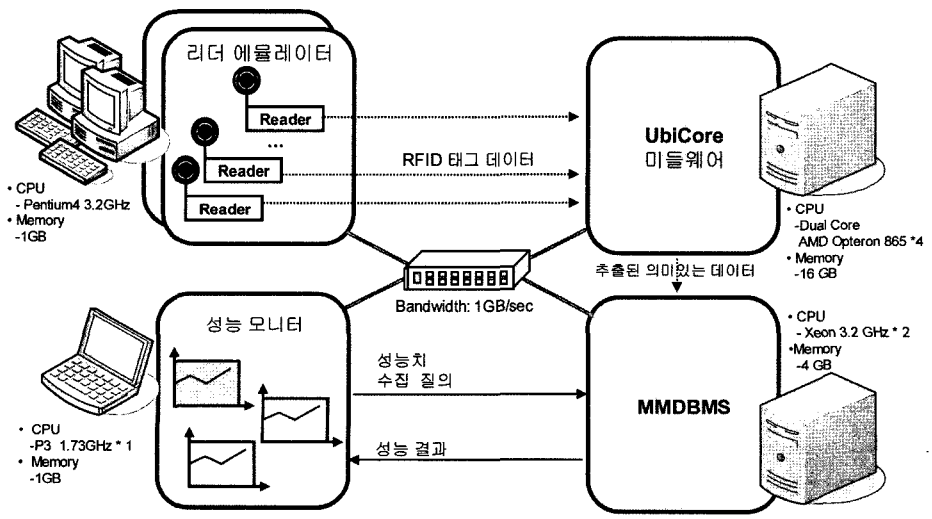

그림 10 성능 실험 환경

다. 대략 하나의 리더가 초당 80개의 태그 데이타를 생성할 때 평균 응답 시간 100ms 이내에서 항상 100%의 처리율을 보였다. 실험 결과는 현재 UbiCore가 200개의 리더와 200개의 컨텍스트-서비스가 등록된 환경에서 대략 16,000건의 태그 데이타를 준실시간으로 처리한다고 볼 수 있는데 이는 UbiCore 미들웨어가 탑재된 시스템의 사양에 비해 만족할만한 성능은 아니다.

UbiCore의 동작을 프로파일링 해보면 연속 질의 처리를 위해 데이타를 얻어오는 부분에서 많은 CPU 자원을 사용하고 있었다. 이는 데이타 입력 여부를 폴링(polling)에 기반하여 처리하고 있기 때문인데, 폴링 대신 wait-notify 개념을 적용하면 성능이 나아질 것이다. 하지만 항상 wait-notify 개념이 적용된 것이 좋은 것은 아니다. 데이타가 드물게 오는 경우에는 wait-notify 개념을 이용하는 것이 좋고, 데이타가 많이 생성되는(busty) 환경에서는 폴링이 더 나을 수도 있다. 또한, '플래폼에 독립적이다'는 이유로 UbiCore 전체를 JAVA를 이용하여 구현했는데, 많은 CPU 작업을 필요로 하는 부분은 JAVA 가 아닌 C로 구현함으로써 더 나은 성능을 낼 수 있을 것이다.

6. 결론 및 향후 연구

유비쿼터스 컴퓨팅 환경에서 스마트 객체를 활용하는 응용 서비스를 개발하기 위해 추가적으로 요구되는 공통의 작업은 미들웨어가 처리하도록 하고, 응용 개발자는 개발하고자 하는 응용 서비스의 본래 기능만을 고려함으로써 보다 양질의 서비스를 조기에 제공할 수 있으며, 새로운 환경으로의 적응을 용이하게 한다. 본 논문에서는 이러한 미들웨어로 XML 기반 RFID 미들웨어 시스템인 UbiCore를 제안했다. UbiCore는 XQuery를 RFID 스트림 데이타 처리에 특화시킨 XQueryStream이라는 질의 언어를 가지며, XML 트리거에 기반하여 실시간 스트림 데이타뿐 아니라 저장소에 저장된 이력 데이타에 대해서도 질의를 할 수 있다. 또한, CSML이라는 마크업 언어를 이용하여 XQuery로 표현된 컨텍스트와 서비스의 연계 정보를 쉽게 표현할 수 있다. 뿐만 아니라, RFID 태그 데이타 스트림을 빠르게 처리하기 위해 연속 질의 평가 이전에 프리필터링이라는 전처리 단계를 두어 불필요한 데이타가 연속질의의 대상이 되는 것을 방지했으며, 이전 질의 평가의 중간 결과를 효율적으로 관리하고 이를 이용한 질의 평가 방법을 제안했다.

현재 성능에 대한 고려가 덜된 UbiCore 2.0 버전의 개발이 완료되었고, 성능 병목 지점을 파악하고 제거하여 성능을 향상시키기 위해 다양한 환경에서 시험과 프로파일링을 하고 있다. 뿐만 아니라 다중 질의 처리, 모니터링 그리고 스케쥴링에 관한 연구가 진행되고 있다. 또한, UbiCore가 분산 RFID 미들웨어 시스템으로도 동작할 수 있게 하는 연구도 진행될 예정이다.

참 고 문 헌

[1] Intel, "Building the Digital Supply Chain: An Intel Perspective," Intel Corp., Jan. 2005.

[2] METRO, "The METRO Group Future Store Initiative," http://www.futurestore.org

[3] Siemens to Pilot RFID Bracelets for Health Care. http://www.infoworld.com/article/04/07/23/HNrfidimplants1.html, Jul. 2004.

[4] 원종호 외, "유비쿼터스 컴퓨팅 환경을 위한 RFID 기반 센서 데이타 처리 미들웨어 기술 동향", 전자통신동향분석, 한국전자통신연구원, 19권 5호, 2004년 10월.

[5] Sun Java System RFID Software, version 2.0, http://www.sun.com/software/products/rfid/index.xml

[6] Enterprise Information Architecture for RFID and Sensor-Based Services, Oracle White Paper, Feb. 2006.

[7] CapTech, "TagsWare: Agile RFID Solutions," http://www.tagsware.com

[8] ConnecTerra, "RFTagAware," http://www.connecterra.com

[9] GlobeRanger, "iMotion," http://www.globeranger.com

[10] Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation 04 Feb. 2004.

[11] XQuery 1.0 : An XML Query Language. W3C Candidate Recommendation, 3 Nov. 2005.

[12] The EPCglobal Architecture Framework. EPCglobal Final Version of 1 Jul. 2005.

[13] The Application Level Events (ALE) Specification, Version 1.0. EPCglobal Proposed Specification Version of 11 Feb. 2005.

[14] Alien Technology. "RFID Readers," http://www.alientechnology.com/products/rfid_ readers.php

[15] Symbol, AR400 RFID Reader from Symbol. http://www.symbol.com/AR400/

[16] Intermec Technologies, "IF5 Fixed Reader," http://intermec.com/eprise/main/Intermec/Content/Products/Products_ShowDetail?Product=RFID2_IF5

[17] SAMSys Technologied, "MP9320 RFID Reader," http://www.samsys.com/default.php?alpha=products&beta=uhf&gamma=mp9320

[18] ThingMagic, "Mercury4 RFID Reader," http://www.thingmagic.com/html/prod-merc4.htm

[19] XML Schema Part 0, 1, 2, W3C Recommendation, 28 Oct. 2004.

[20] Jan-Eike Michels, "XQuery API for Java(XQJ) 1.0," JSR 225 EG Draft Specification, Mar. 2006.

[21] ISO/IEC 4977, Information technology - Syntatic metalanguage - Extended BNF, Dec. 1996.

[22] S. Chandrasekaran, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah. "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," In Proc. of Conference on Innovative Data Systems Research, Jan. 2003.
[23] EPC Tag Data Standards Version 1.1. Standard Specification, 01 Apr. 2004.
[24] Reader Protocol 1.0. Working Draft, EPCGlobal Inc, Mar. 2005.

이 훈 순

1997년 충남대학교 컴퓨터과학과 이학사 1999년 충남대학교 대학원 컴퓨터과학과 이학석사. 1999년~현재 한국전자통신연구원 선임연구원. 관심분야는 데이타베이스, 데이타 스트림 처리, 자료저장시스템

최 현 화

2000년 충남대학교 컴퓨터공학과 학사 2002년 포항공과대학교 대학원 컴퓨터공학과 공학석사. 2002년~현재 한국전자통신연구원 연구원. 관심분야는 XML, 데이타베이스, 데이타 스트림 처리, 데이타 마이닝

김 병 섭

1997년 전북대학교 정보통신공학과 학사 1999년 전북대학교 정보통신공학과 공학석사. 1999년~현재 한국전자통신연구원 선임연구원. 관심분야 XML, 데이타베이스, 데이타 스트림 처리, 정보통합

이 명 철

1999년 충남대학교 컴퓨터공학과 공학사 2001년 충남대학교 대학원 컴퓨터공학과 공학석사. 2001년~현재 한국전자통신연구원 선임연구원. 관심분야 XML, 데이타베이스, 데이타 스트림 처리, 정보통합

박 재 홍

2000년 충남대학교 컴퓨터공학과 공학사 2003년 충남대학교 대학원 컴퓨터공학과 공학석사. 2004년~현재 한국전자통신연구원 연구원. 관심분야는 SOA, EDA, Webservice, 데이타 스트림 처리

이 미 영

1981년 서울대학교 식품영양학과(계산통계학 부전공) 이학사. 1983년 서울대학교 계산통계학과 이학석사. 2005년 충남대학교 컴퓨터공학과 공학박사. 1983년~1985년 한국전기통신연구소 연구원. 1988년~현재 한국전자통신연구원 책임연구원. 관심분야는 데이타베이스시스템, XML 문서 관리, 정보 통합, 데이타 스트림 처리

김 명 준

1978년 서울대학교 계산 통계학과 학사 1980년 한국과학기술원 전산학과 이학석사. 1986년 프랑스 Nancy 제 1대학교 응용 수학 및 전산학과 이학박사. 1980년~1981년 아주대학교 종합연구소 연구원. 1981년~1986년 프랑스 Nancy 전산학 연구소(CRIN) 연구원. 1993년 프랑스 Univ. of Nice Sophia-Antipolis 방문 교수. 1986년~현재 한국전자통신연구원 디지털홈연구단 인터넷 서버그룹 책임연구원/그룹장 관심분야는 데이타베이스, 분산시스템, 인터넷서비스

진 성 일

1978년 서울대학교 계산 통계학과 학사 1980년 한국과학기술원 전산학과 이학석사. 1994년 한국과학기술원 전산학과 이학박사. 1987년~1989년 Northwestern 대학 전산학과 객원교수. 1983년~현재 충남대학교 전기정보통신공학부 교수. 관심분야는 데이타베이스, 멀티미디어시스템, 소프트웨어공학, 시뮬레이션모델링