태그 데이터 처리를 최소화하기 위한 RFID미들웨어의 필터링 알고리즘

손성원*, 김용진*, 유민수*
*한양대학교 전자컴퓨터통신공학과
e-mail : {swson, yjkim, msryu}@rtcc.hanyang.ac.kr

A Filtering algorithm of RFID middleware for minimizing tag data handling

Sung-Won Son*, Yong-jin Kim*, Min-Soo Ryu*
*Dept of Electronics & Computer Engineering, Han-yang University

요 약

RFID 시스템의 적용범위가 증가함에 따라 실시간으로 처리해야 하는 태그의 정보도 급격하게 증가하고 있다. 따라서 불필요한 태그정보를 제거하고 정제하여 클라이언트로 전달하는 RFID 미들웨어의 역할이 증가하고 있다. 미들웨어에서 태그정보를 필터링 하기 위한 Brute force 방법은 대용량의 태그정보의 필터링에는 비효율적이며 미들웨어의 성능을 저하시키는 원인이 된다. 본 논문은 빠른 시간에 효과적인 필터링을 위해 미들웨어 표준인 EPCglobal ALE의 ECSpec에 정의된 필터 패턴들 사이의 연관 관계를 구성하여 패턴의 비교 횟수를 감소시키고, 표준에서 정의한 EPC 태그형식의 특징을 해싱 알고리즘에 적용하여 빠른 검색을 수행함으로써 효율적인 RFID 미들웨어의 필터링 알고리즘을 구현하였다.

1. 서론

RFID는 사물에 부착되어 있는 태그(Tag)의 정보를 리더를 통해 인식하는 객체인식기술 중의 하나이다.[1]

RFID 시스템의 적용범위의 급격한 증가에 따라 실시간으로 처리해야 할 태그정보의 양도 늘어나고 있다. 따라서 대용량의 태그정보를 처리하기 위한 미들웨어의 중요성이 부각되고 있다.

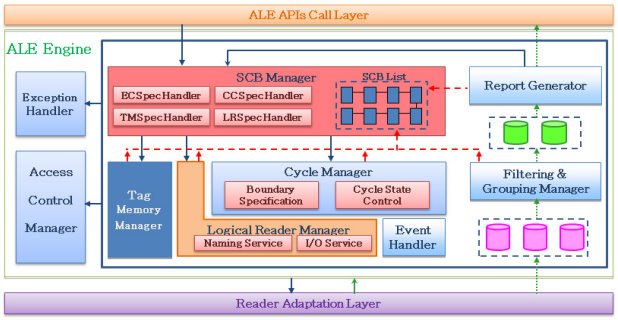
본 논문은 EPCglobal의 ALE (Application Leve Event)[2] 표준을 기반으로 하는 미들웨어를 설계하고 태그정보 처리를 위한 고속 필터링 알고리즘을 방법을 제안한다.

본 논문은 구성은 다음과 같다. 2장에서는 설계한 미들웨어에 대하여 소개하고, 3장에서는 본 논문에서 제안하는 필터링 방법, 4장에서는 알고리즘의 성능평가와 마지막 5장에서는 결론 및 향후 연구 방향을 기술한다.

2. RFID 미들웨어 설계

2.1 ALE 엔진 구조

그림 1은 본 논문에서 제안하는 RFID 미들웨어 시스템의 아키텍처를 보여주고 있다. ALE 엔진의 주요 컴포넌트로는 Event Cycle Specification을 관리하는 SCB Manager, 이벤트를 발생시키는 Cycle Manager(CM), 리더를 정의하고 리더들을 그룹화 하거나 추상적인 이름을 정의 하는 Logical Reader Manager(LRM), 태그정보를 필터링 및 그룹핑등 사용자의 요구에 맞게 태그를 처리하는 Filtering & Grouping Manager(FGM), 리포트 작성과 작성된 리포트를 클라이언트로 전송하는 Report Generator(RG)가 있다.



(그림 1) ALE 엔진

2.2 ALE 엔진 동작

기본적인 ALE 엔진의 실행은 다음과 같다. 클라이언트가 ALE 인터페이스를 통해 ECSpec을 전달한다. SCB Manager는 이 ECSpec을 등록하고 Cycle Manager를 생성하고 이벤트 발생을 위해 ECSpec의 정보를 분석한다. 클라이언트는 리더를 사용하기 위한 LRSpec(Logical Reader Specification)을 LRM에 등록하고 리더들을 대기 상태로 만든다.

클라이언트의 실행 명령에 의해 CM은 LRM을 통해 태그정보를 FGM으로 전달한다. FGM은 ECSpec에 정의된 패턴정보로 태그정보를 필터링 및 그룹핑을 하고, 결과를 RG로 전달을 한다. RG는 ALE에 정의된 ECReports 형태로 정보를 가공하여 클라이언트로 전송하게 된다.

3. 고속 필터링 기법

3.1 비교 횟수를 줄이는 필터링 기법

ECSpec의 ECReportSpec들에 정의된 필터패턴 사이에는 포함관계가 성립될 수 있다. 포함관계가 성립된 필터패턴에서는 하나의 패턴을 만족시키는 태그정보는 그 패턴을 포함하는 모든 리포트에 포함을 시킨다. 따라서 태그정보가 하나의 필터패턴과 일치하게 되면, 그 태그 정보는 다른 패턴과 비교를 하지 않는다. 태그를 패턴과 비교를 할 때는 패턴의 범위가 작은 패턴을 먼저 검색한다. 범위의 대소 관계는 '*' > [Low value -Hi value] > 단일 값과 같다. 표 1과 같이 두 개의 ECReport는 다음과 같은 포함관계를 가진다.

1.1.[1-5] ⊃ 1.1.[3-4]

2.1.[1-2] ⊃ 2.1.1

<표 1> ECReport

리포트 이름	패 턴	
ECReport1	1.1.[1-5]	2.1.[1-2]
ECReport2	1.1.[3-4]	2.1.1

포함관계를 형성한 패턴들의 정보는 표2 와 같다.

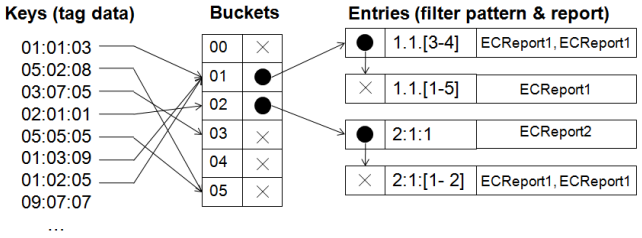
<표 2> 포함관계를 형성한 패턴 리스트

패턴	리포트 이름
1.1.[3-4]	ECReport1, ECReport2
1.1.[1-5]	ECReport1
2.1.1	ECReport1, ECReport2
2.1.[1-2]	ECReport2

포함관계가 이루어진 패턴과의 검색에서 포함된 태그정보는 표2에서 각 패턴에 포함된 리포트에 모두 삽입이 된다. 예를 들어 1.1.[3-4]의 조건을 만족하는 태그는 ECReport1과 ECReport2에 모두 삽입된다.

3.2. 해싱(Hahsing) 기반 패턴 검색 방법

해싱은 데이터의 규모에 관계없이 삽입, 탐색, 삭제 조작을 실질적으로 *O(1)*에 할 수 있는 알고리즘이다. 해싱을 패턴 검색에 적용함으로써 태그와 일치하는 패턴검색을 효율적으로 할 수 있다. 그림 2는 표 2에 구성된 패턴 리스트를 해싱을 이용해 패턴검색을 하는 예를 보여주고 있다. 해쉬 테이블의 키 값은 serial number값의 첫 번째 필드 값이 된다. 이 필드의 값이 [a-b], '*'일 경우 개별적인 키 값으로 취급한다. 해시 테이블의 값은 리스트로 구성되며 리스트의 끝까지 검색을 한다.



(그림 2) 해싱 기반 필터링 예시

4. 실험 및 평가

본 논문에서 제안한 알고리즘을 테스트하기 위해서 가상적으로 태그 정보를 생성하고 미들웨어에 전송한다. 실험은 일반적인 Brute force방법과, 포함관계를 이용한 필터링, 포함관계에 해싱을 적용한 필터링으로 구분하여 성능 비교를 수행하고 평가한다.

4.1 구현 환경

- CPU : Intel Core2 Quad 2.5Ghz
- Memory : 3GB
- OS : WinXP SP3
- 개발 언어 : Java

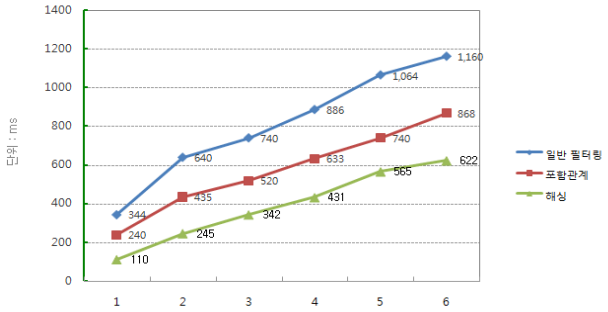
실험은 표3과 같이 3개의 ECReportSpec을 사용하였다.

<표 3> ECReportSpec 필터 패턴 정의

Report1	urn:epc:pat:gid-96:1.1.[1-5000]
	urn:epc:pat:gid-96:2.[2-3].[1-5000]
	urn:epc:pat:gid-96:[3-5].*.*
Report2	urn:epc:pat:gid-96:1.1.[1-5000]
Report3	urn:epc:pat:gid-96:2.[2-3].[1-5000]
	urn:epc:pat:gid-96:3.[1-5].*

4.2 실험결과

표 4는 만개부터 6만개 까지의 태그 정보를 생성하여 세 가지 방법으로 필터링한 실행 시간을 보여준다.



(그림 3) 필터링 속도 비교

5. 결론

본 논문에서는 RFID 필터링의 중요성에 대해서 인식하고 효율적인 필터링 방안에 관하여 연구하였다. EPCglobal ALE를 분석하여 필터 패턴들의 포함관계 처리에 대한 문제를 인식하고 필터 패턴들을 포함관계를 형성하여 태그정보를 필터패턴에 비교하는 횟수를 줄이고, 해싱 알고리즘을 검색 방법에 적용함으로써 고속 필터링 방법을 제안하였다. 실험을 통하여 제안한 알고리즘이 일반 Brute force 방법 보다 속도면에서 우수함을 알수 있었다.

참고문헌

[1] 유승화, "RFID기술현황 및 활용분야", 정보과학지, 제23권 제7호. pp.64-70, 2005.7.

[2] EPCglobal, "The Application Level Events(ALE) Specification, Version 1.1

[3] EPCglobal, http://www.epcglobalinc.com