Journal of Korea Multimedia Society Vol. 12, No. 4, April 2009(pp. 608-615)

에이전트 이동성을 이용한 효율적인 전자태그 데이터 전처리 가능한 분산 소프트웨어 도구

안용선[†], 안진호[††]

요 약

RFID 기술의 발전으로 태그 가격이 급속도로 낮아짐에 따라, 더 세밀하게 제품을 관리하기 위해 각 태그가 포장상자에만 부착되는 것이 아니라 개별 제품에 부착된다. 그러나 RFID 데이터를 처리하는 판독기와 미들웨어가 한정된 하드웨어 자원을 가지고 있기 때문에, 초대용량의 태그 데이터를 신속하게 처리하기 위한 방법들이 필수적이다. 본 논문에서는 이러한 요구조건을 효율적으로 충족시키는 새로운 이동 에이전트 기반 분산형 소프트웨어 도구를 설계하고 구현한다. 이 도구는 명시된 데이터 수집 정책을 포함한 이동 에이전트를 다수의 이동식 판독기들로 전송함으로써, 필요한 데이터가 제품의 운송 중 반복적으로 전 처리 될 수 있도록 하는 편리한 환경을 제공한다. 이러한 수행 형태는 목적지에 도착한 후 매우 많은 양의 태그 데이터를 고정식 판독기에서 처리하는 기존 방식에 비해, 판독기와 미들웨어에서 매우 높은 인식률로 태그 데이터를 처리하기 위해 요구되는 소요시간을 상당히 줄일 수 있다.

Distributed Software Tools Enabling Efficient RFID Data Pre-Processing Using Agent Mobility

Yong Sun Ahn[†], Jin Ho Ahn[††]

ABSTRACT

As RFID tag prices have rapidly been declining because of the advance of RFID technology, each tag is attached to an individual item, not a packing box only, for managing the item much more precisely. However, some mechanisms are essential to handle a very large amount of tag data quickly because readers and middlewares processing RFID data have limited hardware resources. In this paper, we design and implement a new mobile agent-based distributed software tools to satisfy this requirement efficiently. These tools provide a convenient environment enabling required data to be pre-processed repeatedly in transit by transferring a mobile agent including its specified data collection policy to numerous mobile readers. This behavior can significantly reduce the elapsed time required for processing huge volumes of tag data at the readers and middlewares with their very high recognition rates compared with the existing one to process the data by fixed readers after having arrived at the destination

Key words: RFID(RFID), MIDDLEWARE(미들웨어), MOBILE AGENT(이동 에이전트), SCALABILITY (확장성), DATA PRE-PROCESSING(데이터 전처리)

※ 교신저자(Corresponding Author) : 안진호, 주소 : 경기도 수원시 영통구 이의동 (443-760), 전화 : 031)249-9674, FAX : 031)249-9673, E-mail : jhahn@kyonggi.ac.kr
접수일 : 2008년 10월 7일, 완료일 : 2009년 2월 24일
[†] 준회원, 경기대학교 컴퓨터과학과 석사과정
(E-mail : altin@nate.com)

[††] 경기대학교 컴퓨터과학과 부교수
(E-mail : jhahn@kyonggi.ac.kr)
※ 본 연구는 경기도 지역협력연구센터인 경기대학교 콘텐츠융합소프트웨어연구센터의 지원으로 수행되었음(수행과제명: 대규모 RFID 서비스를 위한 단위 통합 프레임워크 개발 및 산업화)

1. 서 론

RFID(Radio Frequency IDentification) 기술은 태그(RFID Tag)에서 방출하는 전파를 판독기의 전파 수신기가 인식하여 객체를 식별하고 식별된 데이터를 미들웨어(RFID Middleware)로 전송하여 처리하는 기술을 말한다[1]. 이러한 기술은 공급 망(Supply chain)을 통해서 이동하는 제품들의 위치 정보와 상태 정보 등 다양한 정보를 실시간으로 파악하는 것을 가능하게 해주고 있다[2-3]. 특히 물류유통 분야에서의 RFID 기술은 산업의 큰 변화를 주고 있다. 창고로 입출고 되는 제품의 정보를 사람이 일일이 파악하지 않아도 자동으로 수집 및 집계가 가능하게 해주어 경제적으로 큰 도움을 주고 있기 때문이다.

그러나 기술의 발전으로 RFID 장비의 소형화가 이루어지고 장비의 가격이 하락되어 기존에 포장 박스 단위로만 부착되었던 RFID 태그를 이제는 개별 제품 단위로 부착할 수 있게 되어 대용량의 태그 데이터 처리를 할 수 있는 판독기와 미들웨어의 필요성이 더욱더 커지고 있다[4-5]. 특히 물류 시스템에서 창고로 출입하는 트럭에 RFID 제품을 가득 싣고 들어온다면 출입문에 설치된 판독기는 수백에서 수천 개의 태그 데이터를 발생시킬 것이다. 더욱이 미들웨어와 판독기의 관계는 일대일의 관계가 아니라 하나의 미들웨어에 여러 대의 판독기가 연결되는 일대다의 관계로 되어있어 앞에서 말한 것처럼 짧은 시간에 많은 양의 RFID 태그 데이터가 발생된다면 미들웨어에 과부하가 걸리게 되어 미들웨어는 다른 판독기로부터 오는 데이터를 처리하는데 있어서 지연이 생길 수 있게 된다[6].

대용량의 태그 데이터를 처리하는 미들웨어의 부하를 줄이는 여러 가지 연구가 활발히 진행되고 있다. 특히 분산 시스템의 본질적인 특징 중 하나인 부하 균등(Load Balancing) 기술을 국내외에서 다양한 방법으로 적용하고 있다. 적응적 부하 균등(Adaptive Load Balancing) 기법은 미들웨어에 연결된 판독기로부터 전송되는 데이터양을 계산하여 처리 데이터양이 많은 미들웨어에 연결된 판독기를 부하가 적은 다른 미들웨어로 판독기를 재배치하여 미들웨어의 과부하를 피하려고 하는 접근 방법이다. 이와 유사한 다른 방법으로 이동 에이전트를 이용한 미들웨어의 부하 균등 기법도 제안되었다[7]. 이는 이동 에이전트(Mobile Agent)가 미들웨어들의 현재 부하 상태와 판독기에서 발생되는 데이터의 양을 수집하여 부하가 적은 미들웨어로 그 데이터를 전달하여 미들웨어의 과부하 상태를 해결하는 기법이다. 그러나 이러한 접근 방법은 판독기로부터 오는 부하를 분산시켜 주기는 하지만 실질적인 데이터 처리가 미들웨어에서 수행되고 있다[8-9]. 또한 판독기로부터 빠르게 수집되는 태그 데이터를 처리하기 위해 미들웨어에 내재되어 있는 데이터 스트림 관리(Data Stream Management System) 에 특성이 부여된 여러 개의 저장소를 생성하고 각 저장소와 데이터의 특성에 맞게 데이터를 저장해 필요시 해당 저장소에만 질의하여 원하는 결과를 빠르게 얻도록 하는 연구도 제안되었다[10]. 그러나 이 방법은 특정 저장소에 데이터가 집중되어 저장소의 효율성이 떨어질 수 있다. 미들웨어의 부하를 줄이기 위한 많은 연구가 진행되었지만 이러한 접근 방법은 궁극적으로 모든 데이터의 처리가 미들웨어에서 이루어지고 있어 실질적으로 미들웨어의 부담을 줄이지는 못했다.

본 논문에서는 판독기와 미들웨어의 부하를 줄이고 대용량의 RFID 태그를 처리할 수 있는 이동 에이전트 기반의 분산형 소프트웨어 도구의 설계 및 개발을 하였다. 이동 에이전트를 이용하여 멀리 떨어져 있는 이동식 판독기에 데이터 수집 정책(Data Management Rule)을 전송하여 데이터 처리를 가능하게 하고 이동시간을 데이터 수집과 처리시간으로 활용하여 목적지에 도착하였을 때 추가 작업 없이 미들웨어에 이동 에이전트를 통해 처리된 데이터를 즉시 제공함으로써 판독기와 미들웨어의 부담을 효율적으로 줄일 수 있다.

이후 2장에서는 구현된 시스템의 구조를 설명하며, 3장에서는 논문에서 제안하는 이동 에이전트 기반의 분산형 소프트웨어의 구조도와 주요 클래스 및 알고리즘에 대해 설명하고, 4장에서는 구현된 시스템의 기능설명을 하며, 5장에서는 구현된 시스템의 성능을 입증하기 위해 실시한 성능 평가에 대해 기술하며, 마지막으로 6장에서 결론을 맺는다.

2. 구현된 시스템 구조

2.1 이동식 판독기와 미들웨어 구조

구현된 시스템은 이동식 판독기 부분(Movable

Reader Module)과 미들웨어 부분(Middleware Module)로 구성되어 있다. 이동식 판독기 부분은 이벤트 관리자[11], 에이전트 관리자[12], 판독기 관리자, 저장소 그리고 RFID 판독기로 구성되어 있으며, 미들웨어 부분은 일반적인 RFID 미들웨어에 에이전트 관리자를 추가 하였다.

이동식 판독기 부분의 에이전트 관리자는 미들웨어에서 전송된 에이전트를 수신하여 에이전트로부터 RFID 데이터 수집 정책을 판독기 관리자에게 전달해주는 역할과 목적지에 도달 하였을 때, 미들웨어로 에이전트를 되돌려 보내는 역할을 수행한다. 판독기 관리자는 에이전트 관리자가 전달해주는 RFID 데이터 수집 정책을 기반으로 연결된 RFID 판독기의 데이터 수집 환경을 변경하는 역할을 하며, 이벤트 관리자는 RFID 판독기에 의해서 수집된 데이터의 처리를 담당하고 있다. 이렇게 처리된 데이터는 저장소에 저장된다. 미들웨어 부분의 에이전트 관리자는 이동식 판독기에게 현재 고정식 판독기에서 적용중인 RFID 데이터 수집 정책을 에이전트를 통해서 전달하는 역할과 이동식 판독기로부터 에이전트가 전송되면 처리된 데이터를 미들웨에게 전달하는 역할을 수행한다.

2.2 데이터 흐름도 설명

그림 1은 논문에서 제안하는 분산형 소프트웨어의 영역별 데이터 흐름도이다. 판독기 부분의 에이전트 관리자는 미들웨어에서 전송된 에이전트로부터 RFID 데이터 수집 정책을 전송 받아 판독기 관리자에게 전달하고, 판독기 관리자는 전달받은 RFID 데이터 수집 정책을 기반으로 이동식 판독기 부분에 연결되어 있는 RFID 판독기의 환경을 변경한다. 이렇게 판독기의 환경 설정을 변경하는 이유는 목적지에 설치된 고정식 판독기와 동일한 환경을 제공하여 이동 중 데이터의 전처리를 가능하게 하기 위해서이다. 이렇게 정책이 적용된 판독기는 변경된 수집 정책을 바탕으로 태그를 인식한다. 이때 판독기는 태그를 인식하여 처리되지 않은 데이터(Raw Data)를 이벤트 관리자에게 전달한다. 이벤트 관리자는 미들웨어에서 수행되어야하는 중복 데이터 제거, 데이터의 집계와 같은 데이터 처리 과정을 수행하여 처리된 데이터(Processed Data)를 만든다. 이렇게 생산된 데이터는 저장소에 저장되어 목적지에 도착하였을 때 이동 에이전트가 미들웨어로 가지고 돌아간다.

3. 제안하는 분산형 소프트웨어

3.1 분산형 소프트웨어 구조도

그림 2는 논문에서 구현한 분산형 소프트웨어의 구조도를 나타내고 있다. 미들웨어에서 생성된 에이전트가 데이터 수집 정책을 가지고 떨어져있는 각각의 이동식 판독기 부분으로 이동하면 이동식 판독기에 설치된 RFID 판독기에 데이터 수집 정책이 적용된다. 적용이 끝난 각각의 RFID 판독기는 태그의 수집 및 처리를 시작하고 완료가 되면 에이전트가 처리된 데이터를 가지고 미들웨어로 모여지는 구조를 가지고 있다.

3.2 주요 클래스 다이어그램

그림 3은 본 논문에서 구현된 주요 클래스인 Agent 클래스의 다이어그램을 표시한 그림이다. Agent 클래스는 기본적인 에이전트 이동을 위한 함

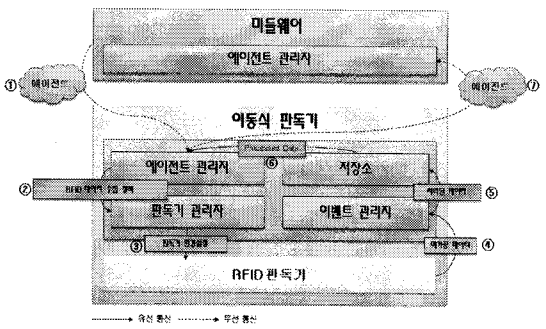

그림 1. 영역별 데이터 흐름도

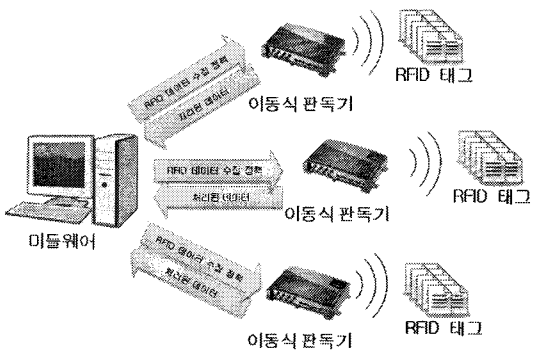

그림 2. 분산형 소프트웨어 구조도

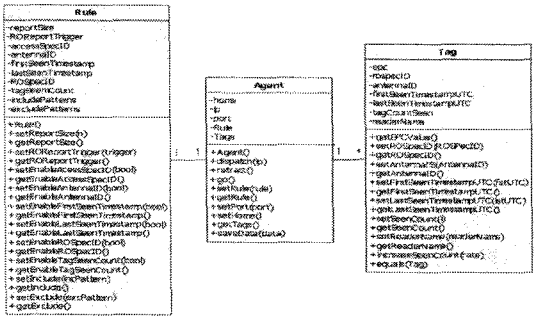

그림 3. Agent 클래스 다이어그램

수인 dispatch(), retract(), go(), setPort, setHome()가 정의 되어 있으며 미들웨어로부터 이동식 판독기로 데이터 수집 정책을 전달하기 위한 함수인 setRule()과 getRule()이 정의 되어있다. 또한 이동식 판독기로부터 미들웨어에게 처리된 태그 데이터를 전달할 수 있는 getTags()와 saveData()가 정의 되어있다.

Agent 클래스가 가지고 있는 Rule 클래스는 미들웨어에서 정의된 데이터 수집 정책의 정보를 저장하고 있으며 Tag 클래스는 판독기가 인식한 태그의 정보를 객체화하여 저장할 수 있도록 Agent 클래스에 포함되어 있다.

3.3 주요 알고리즘

3.3.1 태그 보고서 선택

그림 4는 에이전트가 미들웨어로부터 가지고온 데이터 수집 정책을 이동식 판독기에 적용하는 부분이다. 미들웨어에서 요구하는 정보만을 수집하기 위해 보고서(Tag Report)에 포함될 정보를 설정한다[13]. 5~6 줄은 판독기가 보고하는 보고서 당 몇 개의 태그를 포함 시킬 것인가를 정의하며, 8 줄은 보고서의 보고시기를 정의한다. 11~22 줄은 보고서에 포함 시킬 태그의 정보를 정의하는 부분으로 안테나 ID, EPC, 관측 시간 등을 설정할 수 있다.

3.3.2 에이전트 생성

그림 5는 미들웨어에서 에이전트를 생성하는 코드의 일부분이다. 3~4 줄은 에이전트 생성 시 미들

```
1       private SetReaderConfig createSetReaderConfig() {
2               SetReaderConfig src = new SetReaderConfig();
3
4               // set up RO reporting
5               ROReportSpec rrs = new ROReportSpec();
6               rrs.setN((short) DMR.getReportSize());
7               // report at end of rospec
8               rrs.setROReportTrigger((byte) DMR.getROReportTrigger());
9
10              // TagReport Content Selector
11              TagReportContentSelector trcs = new TagReportContentSelector();
12              trcs.setEnableAccessSpecID(DMR.getEnableAccessSpecID());
13              trcs.setEnableAntennaID(DMR.getEnableAntennaID());
14              trcs.setEnableChannelIndex(true);
15              trcs.setEnableFirstSeenTimestamp(DMR.getEnableFirstSeenTimestamp());
16              trcs.setEnableInventoryParameterSpecID(true);
17              trcs.setEnableLastSeenTimestamp(DMR.getEnableLastSeenTimestamp());
18              trcs.setEnablePeakRSSI(true);
19              trcs.setEnableROSpecID(DMR.getEnableROSpecID());
20              trcs.setEnableSpecIndex(true);
21              trcs.setEnableTagSeenCount(DMR.getEnableTagSeenCount());
22              rrs.setTagReportContentSelectorParam(trcs);
23
24              AccessReportSpec ars = new AccessReportSpec();
25              ars.setAccessReportTrigger((byte) 0);
26
27              return src;
28      }
```

그림 4. ReportSpec 적용 알고리즘

```
1       private void createAgent() {
2               // Agent 생성 및 판독기로 전송
3               Agent agent = new Agent();
4               setRule(agent);
5               InetAddress inet;
6               try {
7                       inet = InetAddress.getLocalHost();
8                       agent.setHome(inet.getHostAddress());
9                       str = JOptionPane.showInputDialog("목적지 정보 입력(ip:port)",
10                                                        "localhost:7201");
11                      info = str.split(":");
12                      agent.setPort(Integer.parseInt(info[1]));
13                      agent.dispatch(info[0]);
14
15              } catch (UnknownHostException e) {}
16              .....
17      }
```

그림 5. 에이전트 생성 알고리즘

웨어의 데이터 수집 정책을 설정해주는 부분이며, 7~8 줄은 데이터를 처리한 후 에이전트가 되돌아갈 수 있도록 목적지 미들웨어의 정보를 설정한다. 9~13 줄은 에이전트를 전송하기 위한 목적지의 정보(IP, PORT)를 입력받아 목적지로 이동(Dispatch) 하는 부분이다.

4. 구현된 시스템의 기능

4.1 이동식 판독기 부분

이동식 판독기 부분은 설치된 판독기의 정보를 표시하는 판독기 정보 표시 영역(Reader View), 미들웨어로부터 전달받은 RFID 데이터 수집 정책을 표시하는 에이전트 정보 표시 영역(Agent View), RFID 판독기로부터 전달받은 태그 정보를 표시하는 태그 정보 표시 영역(Tag List), 시스템의 이벤트를 표시하는 콘솔 정보 표시 영역(Console View)으로 구성되어 있다. 그림 6은 이동식 판독기의 실행화면으로 연결된 판독기의 정보와 RFID 데이터 수집 정책, 판독기로부터 보고된 태그 목록을 표시하고 있다.

4.1.1 판독기 정보 표시 영역

현재 동작중인 RFID 판독기의 정보를 표시하는 부분으로 RFID 판독기의 이름과 판독기의 상태(Running, Suspended, Stoped), 설치된 전파 수신기 수, 판독기의 IP와 PORT 정보를 표시한다. 다수

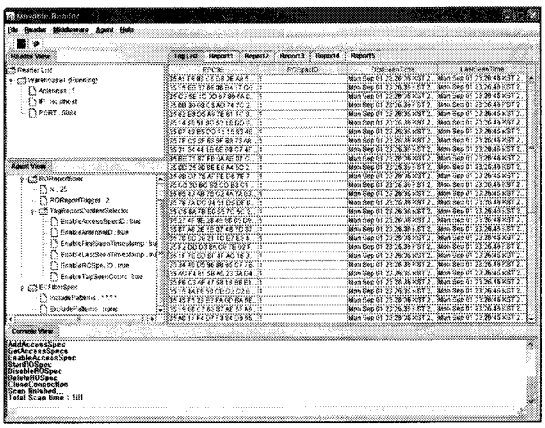

그림 6. 이동식 판독기 실행화면

의 판독기일 경우에는 트리(Tree) 형태로 표현되어 원하는 판독기의 정보를 쉽게 볼 수 있도록 구성하였다.

4.1.2 에이전트 정보 표시 영역

미들웨어로부터 전달받은 에이전트가 표시되면 목적지 미들웨어에서 적용 중인 데이터 수집 정책을 확인할 수 있다. 데이터 수집 정책은 ROReportSpec, ECFilterSpec 등이 표시되어 적용할 정책에 자세한 내용을 확인할 수 있다. 또한, 여러 개의 에이전트가 서로 다른 데이터 수집 정책을 가지고 올 경우 원하는 수집 정책을 선택하여 판독기마다 필요한 수집 정책을 선택적으로 적용 가능하도록 구현되어 있다.

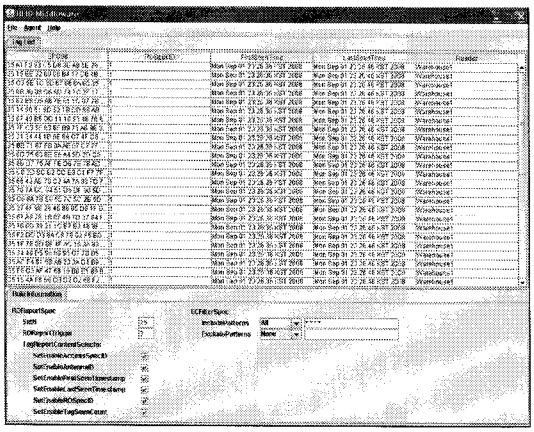

그림 7. 미들웨어 실행화면

4.1.3 태그 정보 표시 영역

판독기로부터 미들웨어로 보고될 태그의 내용이 표시가 된다. 판독기가 보고한 다수의 보고서들과 중복된 데이터를 제거한 최종 보고서를 확인할 수 있다. 표시되는 정보로는 EPC, 관측 시간, ROSpecID 등이 표시가 된다.

4.2 미들웨어 부분

그림 7은 미들웨어 부분의 실행 화면이다. 미들웨어에서는 이동식 판독기로 전달할 RFID 데이터 수집 정책을 정의할 수 있는 데이터 수집 정책 정보(Rule Information) 부분과 판독기로부터 전달받은 처리된 태그 정보를 표시하는 태그 정보 표시 영역(Tag List) 부분으로 구성되어 있다.

4.2.1 데이터 수집 정책 정보 표시 영역

RFID 데이터 수집 정책을 정의하는 부분으로 보고서 당 포함되는 태그의 개수와 판독기의 보고서 보고시기를 정의하며 보고서에 포함 시킬 태그 정보를 정의하는 ROReportSpec과 불필요한 데이터의 제거 및 처리하기 위한 IncludePattern과 ExcludePattern을 정의하는 ECFilterSpec으로 구성되어 있으며 이는 에이전트 생성 시 에이전트에 포함된다.

4.2.2 태그 정보 표시 영역

판독기로부터 전송된 처리된 데이터를 확인할 수 있는 부분으로 수집된 태그의 EPC96, ROSpecID, 관측 시간, 관측된 판독기 이름의 정보를 표시하고 있다.

5. 성능평가

5.1 성능평가 환경

성능평가에 사용되는 RFID 판독기는 Rifidi 사(社)에서 개발한 에뮬레이터(Emulator)를 사용하였다. 이는 EPCglobal 표준에 맞추어 개발되어 실제 RFID 판독기와 같은 역할을 해주며 또한, 임의의 가상 태그를 생성시킬 수 있어 실제 개발환경과 유사한 조건을 제공해준다. 성능평가에 사용되는 고정식 판독기의 인식 속도는 유통물류산업에서 요구하는 고정식 판독기의 인식 속도인 초당 500개(일반적으로 200tags/sec, 최대 500tags/sec)의 태그를 인식하도록 설정하여 제안 방법과 비교하였다[14]. 또한 전파를 인식하는 기술적 특성으로 인해 실제 상용 환경에서 사용되는 RFID 판독기들의 인식률이 100%가 되지 못하고 있다. 그리하여 실제적인 실험 환경을 고려하기 위해 고정식 판독기의 인식률을 일반적인 상용 환경에서의 인식률인 80% 부터 99% 까지 변화시키며 500개의 태그를 완전히 인식하는데 소요되는 시간을 측정하였다[2]. 이를 통해 태그를 모두 인식 후 미들웨어에서 불필요한 데이터의 제거 및 처리를 마치고 처리된 데이터를 응용 프로그램에게 제공하는데 까지 걸리는 시간을 측정하여 본 논문에서 제안하고 있는 에이전트 기반의 이동식 판독기를 사용하였을 때와 비교하여 성능평가를 실시한다. 성능평가에 사용되는 미들웨어와 이동식 판독기의 환경은 표 1과 같다.

표 1. 성능평가 환경

구분	미들웨어	이동식 판독기
운영체제	Windows XP Pro Service Pack 3	
구현언어	Java Development Toolkit v6.0	
메모리	2GB	1GB
CPU	Intel(R) Core(TM)2 Quad CPU 2.40GHz (4 CPUs)	Intel(R) Core(TM)2 CPU 1.66GHz (2 CPUs)
네트워크	Realtek RTL8169/8110 Family Gigabit Ethernet NIC	RT73 USB Wireless LAN Card

5.2 성능평가 분석

본 논문에서 제안하고 있는 소프트웨어 도구의 우수성을 증명하기 위해, 일반적인 처리 방법인 고정식 판독기를 이용한 방법과 본 논문에서 제안하고 있는 이동 에이전트 기반 판독기를 이용한 방법의 처리시간을 비교하였다. 에뮬레이터에 의해 임의로 생성된 RFID 태그를 100개부터 800개까지 100개 단위로 증가시켜, 이를 완전히 인식하고 처리를 완료하는데 소요되는 시간을 측정하였다. 즉, 고정식 판독기를 이용한 방법의 처리시간은 목적지에 도달한 시점부터 태그 인식을 시작하고, 미들웨어에 데이터를 전송하여 처리를 완료하는데 까지 소요된 시간이다. 이동 에이전트 기반 판독기 방법의 처리시간은 목적지로 가는 도중 이미 태그의 인식과 처리가 완료되었기 때문에 태그의 인식과 처리시간은 고려하지 않고 이동 에이전트가 미들웨어에 전송되는데 소요되는 시간만을 고려한다. 에이전트 전송시간은 에이전트 직렬화(Serialization) 시간 및 역직렬화(Deserialization) 시간을 모두 포함한다.

실험 결과 그림 8과 그림 9와 같이 태그의 수가 약 200개 이하의 경우에는 에이전트 사용에 따른 직렬화와 역직렬화에 의해서 발생하는 비용 때문에 기존의 방법이 우수했지만, 약 200개를 초과하면서부터 에이전트 기반 판독기로 처리하는 방법이 더 우수함을 알 수 있다. 또한 상용 RFID 판독기에 적용되는 인식률을 고려한 결과, 500개의 태그를 처리하는데 있어서 99%의 매우 높은 인식률을 가진 RFID 판독기를 사용했을 때는 1005(ms)의 처리시간이 소요되었지만 논문에서 제안하고 있는 에이전트 기반 판독

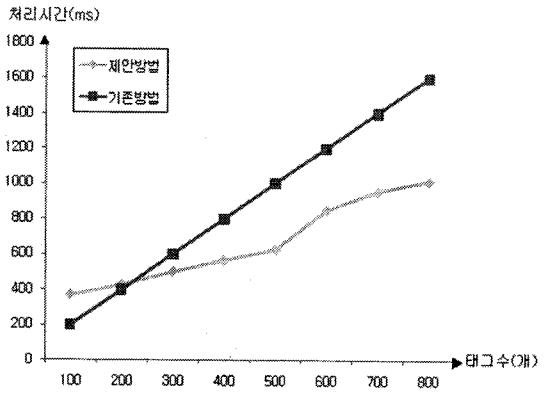

그림 8. 고정식 판독기와 이동 에이전트 기반 판독기의 처리시간

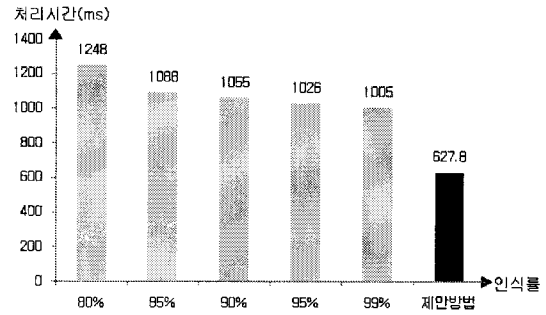

그림 9. 이동 에이전트 기반 판독기와 고정식 판독기의 인식률에 따른 처리시간

기 방법으로 처리한 경우는 627.8(ms)이 소요되어 약 378(ms)의 시간을 절약할 수 있었다. 본 논문에서 제안하는 방법은 목적지에 도달했을 때부터 미들웨어로 데이터가 도달되어 처리되는 데까지 걸리는 지연시간을 상당히 줄일 수 있다. 이러한 결과는 대용량의 태그를 인식해야 할 경우 제안하는 방법이 더욱 효율적인 방법으로 사용될 수 있음을 보여준다. 이는 태그의 인식과 처리를 목적지에 도달하는 시점부터 수행하는 것이 아니라 운송 중에 데이터 수집 정책을 포함한 에이전트를 통하여 RFID 태그 데이터의 전처리를 가능하게 하였기 때문이다.

6. 결 론

본 논문에서는 대용량의 태그 데이터를 처리하기 위해서 이동 에이전트 기반 판독기 및 미들웨어의 처리 능력 향상을 위한 소프트웨어 도구를 개발하였다. 이동 에이전트를 통해 목적지 미들웨어에서 요구하는 RFID 데이터 수집 정책을 이동식 판독기들에게 전달하여 목적지에 있는 고정식 판독기들과 동일한 환경을 제공해 주며, 이를 통해 제품의 이동시간 중 데이터 전처리를 가능하게 하였다. 또한 제품의 이동시간 동안 태그 데이터 인식과 처리 작업을 반복적으로 수행할 수 있어, 기존 처리방식보다 신뢰성 있는 데이터를 미들웨어에게 제공해 줄 수 있다. 대용량의 태그 데이터를 처리하는데 있어서, 본 논문에서 제안하는 이동 에이전트 기반 판독기의 효율성을 입증하기 위해 고정식 판독기 방법과의 성능평가를 실시하였으며, 그 결과 약 200개 이상의 태그 데이터를 처리하는데 있어서 제안하고 있는 방법이 보다 효율적이었으며, 상용 환경의 특성까지 고려한다면

더욱 우수함을 입증하였다.

참 고 문 헌

[1] 최길영, 성낙선, 모희숙, 박찬원, 권성호, "RFID 기술 및 표준화 동향," 전자통신동향분석, 제22권, pp. 29-37, 2007.

[2] B. Glover and H. Bhatt, *RFID Essentials,* O'Reilly Media Publishers, N.Sebastopol, Calif., pp. 1-139, 2006.

[3] F. Armenio, H. Barthel, L. Burstein, P. Dietrich, J. Duker, J. Garrett, B. Hogan, O. Ryaboy, and S. Sarma, "The EPCglobal Architecture Framework Final Version 1.2," pp. 24-26, 2007.

[4] 손명식, 조병록, "RFID Tag 기술," 고분자과학과기술, 제17권, 제1호, pp. 4-17, 2006.

[5] 김선진, 박석지, 구정은, 김내수, "RFID/USN 산업동향 및 발전전망," 전자통신동향분석, 제20권, 제3호, pp. 43-55, 2005.

[6] J. F. Cui and H. S. Chae, "Mobile Agent based Load Balancing for RFID Middlewares," *Proceedings of the International Conference on Advanced Communication Technology*, pp. 973-978, 2007.

[7] W. T. Yeung, "Load Balancing using Mobile Agent Approach," M.S. Thesis of the Chinese University of Hong Kong, 2000.

[8] M. K. Kona and C. Z. Xu, "A Framework for Network Management using Mobile Agents," *In Proceedings of the First Workshop on Internet Computing and E-Commerce (ICEC'01)*, 2001.

[9] Cisco Distributed Director, http://www.csico.com/warp/public/cc/pd/cxsr/index.html

[10] 한수, 박상현, 신승호, "RFID 입력 데이터 스트림에 대한 다중 버프 기반의 고속 데이터 처리 알고리즘," 한국 컴퓨터정보학회 논문지, 제13권, 제22호, pp. 79-85, 2008.

[11] EPCglobal, "The Application Level Events (ALE) Specification, Version 1.1 Part I: Core Specification," pp. 17-20, 2008.

[12] P. M Reddy, "Mobile Agents Intelligent Assistants on the Internet," Resonance, Vol.7, No.7, 2002.

[13] Rifidi Technology, Rifidi User Guide, http://www.rifidi.org/documentation.html, 2007.

[14] 이용한, "주요 산업별 표준적용 모델(템플릿) 및 ROI 분석 툴 개발 - 최종보고서," 산업 자원부 한국유통물류진흥원, pp. 38-39, 2007.

안 용 선

2007년 경기대학교 전자계산학과 졸업 학사

2007년 3월~현재 경기대학교 전자계산학과 석사과정

관심분야 : RFID, 미들웨어, 에이전트 시스템

안 진 호

1997년 2월 고려대학교 컴퓨터과학과 학사

1999년 2월 고려대학교 컴퓨터과학과 석사

2003년 2월 고려대학교 컴퓨터과학과 박사

2003년 3월~현재 경기대학교 정보과학부 컴퓨터과학과 부교수

관심분야 : 분산시스템, p2p, 그룹통신, 이동 에이전트 시스템