An Approach to Query Decomposition for Reader Level Filtering in RFID Middleware

M. Ashad Kabir\textsuperscript{1}, Jun Han\textsuperscript{1}, Wooseok Ryu\textsuperscript{2}, Bonghee Hong\textsuperscript{2}

\textsuperscript{1}Faculty of Information & Communication Technologies
Swinburne University of Technology
Melbourne, Australia

\textsuperscript{2}Department of Computer Engineering
Pusan National University
Busan, Republic of Korea
Outlines

- Introduction
  - RFID System Overview & Standard
  - RFID Tag Memory
- Motivation
- Query Analysis based on Standard
- Problem Analysis
- Query Decomposition Technique
- Experimental Evaluation
- Conclusion
Introduction

RFID System Overview

EPCglobal Standard*

Applications

ALE Interface

Filtering & Collection (ALE)

Reader Protocol

http://www.epcglobalinc.org/
**RFID System Overview**

- Process real time event data

---

**Tags**

**Readers**

**RFID Middleware**

**Enterprise Applications**

---

**Data filtering and aggregation**

**Data routing and integration**

---

**Process Management**

**Reader Coordination**

---

**Environments**

- Distribution center
- Retail
- Manufacture
- Stores
RFID Tag Memory

**Application of User-Memory**

- Cold Chain Management
  - temp, humidity, etc
- National Drug Code
  - Lot/batch number, expire date, etc
- Telecommunication industries
  - Version level, country of origin, etc
- Air transport industry
  - Passenger name, flight number, etc
- Animal tracking system
  - Date of birth, last vaccination, etc
Motivation

**Reader Level Filtering (RLF)**

- *Smart Reader*
  - Provides sophisticated filtering, smoothing and other functionality

- EPCglobal proposed Reader Protocol (RP) standard
  - Which specifies interaction between Reader and Middleware with concept of RLF

**Why necessary/required?**

- In RFID environment M/W need to process huge amount of RFID Streaming data
  - Wal-Mart generates as much data in three days as is contained in the entire U.S. library of congress

- RLF reduce M/W load by diminishing
  - Network traffic
  - Processing time
  - Memory requirements
  - Computational complexity
Motivation

RFID Middleware should support
- Various clients
- Multiple queries
- Several readers

M/W load depends on:
- No. of query conditions (ECSpec)
- No. of Tags information send by readers

Reader load depends on:
- No. of query conditions (TagSelector)
- No. of Tags on interrogation zone

Tags with User Memory
Motivation

Load reduction scenario

- Considering
  - Reader level data filtering
  - Pallet contains 100 items
    - Monitor – 20, CPU – 60, Keyboard – 20
  - Application only interested information about Keyboard
- Reduce 80% network traffic as well as amount of Data

\[
\text{Reducing traffic} = \frac{N - I}{N} \times 100\%
\]

Where

- \( N \) is total amount of data
- \( I \) is amount of interested data
Query Analysis

Query Model (for Reader) – Reader Protocol Standard

IF ((Value bitand Mask) = (F_value bitand Mask))
THEN FieldMatchesTheFilter ()
**Query Analysis**

**Query Model (for Middleware) – ALE standard**

- **Filter Specification (FS)**
  - INCLUDE or EXCLUDE
  - Represents some area of tag memory
  - INCLUDE – pass tag if tag-value match the pattern in PatList
  - EXCLUDE – pass tag if tag-value doesn’t match all the patterns in PatList

- **Pattern List**
  - e.g. [10-18], &FF=F6

- **Reader Specification (RS)**
  - R1
  - R2
  - ... Rn

- **Reader ID**

Query for Middleware, e.g. ECSpec
### Application Query Examples

<table>
<thead>
<tr>
<th>Query</th>
<th>Tag field name</th>
<th>Filter condition</th>
<th>Pattern list</th>
<th>Reader name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>lotNumber</td>
<td>EXCLUDE</td>
<td>4, [10-15]</td>
<td>R1, R2, R3, R4</td>
</tr>
<tr>
<td></td>
<td>destCode</td>
<td>INCLUDE</td>
<td>[0A-15]</td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td>lotNumber</td>
<td>EXCLUDE</td>
<td>[10-13]</td>
<td>R1, R2</td>
</tr>
<tr>
<td></td>
<td>destCode</td>
<td>INCLUDE</td>
<td>[06-10], &amp;F7=14</td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td>lotNumber</td>
<td>EXCLUDE</td>
<td>[6-14]</td>
<td>R1, R2, R3, R4</td>
</tr>
<tr>
<td></td>
<td>destCode</td>
<td>INCLUDE</td>
<td>[0B-0F], 14</td>
<td></td>
</tr>
</tbody>
</table>
Simple approach of Reader Level Filtering

- Pushing all queries to reader

<table>
<thead>
<tr>
<th>Query</th>
<th>Logical Reader</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q₁</td>
<td>entry_door, exit_door</td>
</tr>
<tr>
<td>Q₂</td>
<td>entry_door</td>
</tr>
<tr>
<td>Q₃</td>
<td>dock_door</td>
</tr>
</tbody>
</table>

Query Subscription
Query Conversion
Query Propagation
**Problem Analysis**

**Query execution in Reader**
- **Overlap filter-condition (EXCLUDE)**
  - Field-name: *lotNumber*
  - DataFormat: *decimal*
  - FieldSize: 4 bits

If tag-value match the pattern then drop the tag

<table>
<thead>
<tr>
<th>Pattern space</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ 2 8 9 11 ]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source 1</th>
<th>Source 2</th>
<th>Source 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ 9 ]</td>
<td>[ 9 ]</td>
<td>[ 2 ]</td>
</tr>
<tr>
<td>[ 8 ]</td>
<td>[ 8 ]</td>
<td>[ 2 ]</td>
</tr>
<tr>
<td>[ 2 ]</td>
<td>[ 2 ]</td>
<td>[ 2 ]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Space: [0-15]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ 0 7 15 ]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query for Reader</th>
<th>No. of TagSelectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>7</td>
</tr>
<tr>
<td>S₂</td>
<td>4</td>
</tr>
<tr>
<td>S₃</td>
<td>9</td>
</tr>
</tbody>
</table>

Total = 20

<table>
<thead>
<tr>
<th>Problem</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>M/W receives duplicate data</td>
<td>Impose M/W load and Network traffic</td>
</tr>
<tr>
<td>Reader need to process many filtering conditions (TagSelector)</td>
<td>Impose Reader load and reduce performance</td>
</tr>
</tbody>
</table>
Problem Analysis

- **Query execution in Reader**
  - Overlap filter-condition (INCLUDE)
    - Field-name: `destCode`
    - DataFormat: `hex`
    - FieldSize: **16 bits**

  If tag-value match the pattern then **pass the tag**

<table>
<thead>
<tr>
<th>Query for Reader</th>
<th>No. of TagSelectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q₁</td>
<td>12</td>
</tr>
<tr>
<td>Q₂</td>
<td>11</td>
</tr>
<tr>
<td>Q₃</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>28</td>
</tr>
</tbody>
</table>

- **Problem Effect**
  - M/W receives duplicate data: Impose M/W load and Network traffic
  - Reader need to process many filtering conditions (TagSelector): Impose Reader load and reduce performance
Summary

**Summary of the Problems**

- Middleware receives duplicate data
  - Impose MW work load and network traffic between MW and Reader

- Reader need to process many filter conditions
  - Impose Reader work load and reduce performance

**Requirements**

- Appropriate *schema of duplicate data reduction* is required

- Need to *minimize number of filter conditions* in MW and Reader level
**Basic Idea & Approach**

- **Query Decomposition**
  - Identify overlap query conditions
  - Divide *overlap queries* into *sub-queries* for Reader and MW level processing
    - Using *Splitting* and *Merging* techniques

![Diagram of Query Flow and Data Stream](image)

*Figure. Flow of Data and Query*
**Approach**

**Splitting technique** for overlap Filter Condition (EXCLUDE)

- Intersection of pattern-lists: [10-13] – for Reader-level processing

<table>
<thead>
<tr>
<th>Query for Reader</th>
<th>No. of TagSelectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>7</td>
</tr>
<tr>
<td>S₂</td>
<td>4</td>
</tr>
<tr>
<td>S₃</td>
<td>9</td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query for Reader</th>
<th>No. of TagSelectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁₂₃</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>4</td>
</tr>
</tbody>
</table>

- Reduce no. of TagSelectors and duplicate data

**Notification cycle**

**Splitting Technique**

**Reader Level Processing**

**MW Level Processing**
**Merging technique** for overlap Filter Condition (INCLUDE)

- Union of pattern-lists: [06-15], 1C – for *Reader-level processing*

<table>
<thead>
<tr>
<th>Query for Reader</th>
<th>No. of TagSelectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>12</td>
</tr>
<tr>
<td>$S_2$</td>
<td>11</td>
</tr>
<tr>
<td>$S_3$</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>28</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query for Reader</th>
<th>No. of TagSelectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{123}$</td>
<td>17</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>17</strong></td>
</tr>
</tbody>
</table>

**Reduce no. of TagSelectors and duplicate data**

**Merging Technique**

**Reader Level Processing**

**Notification cycle**
Experimental Evaluation

**Investigate readers with filtering functionality**

- Alien – ALR9800, Intermec – IF5, MATRICS – AR400, ThingMagic – Mercury4

![Screen capture of Alien HyperTerminal with command output]

- Two tags are in reader vicinity
- Setting filtering condition
- One tag satisfies the filtering condition
Experimental Evaluation

- Conduct experiment with RP compliant virtual reader – certified by EPCglobal*
- Data is generated by the Tag Data Generator
- Evaluate MW to Reader network traffic & MW performance

* http://www.epcglobalinc.org/certification/sw_cert/
Conclusion

- Exploit Reader filtering capability to reduce MW workload
- Analyzed Query model for MW and Reader according to standard specification
- Illustrate the problem of reader level query execution caused by the simple approach
- Propose **Query Decomposition** approach to resolve the problems
  - Splitting & Merging techniques
- Finally performed simulation to validate the approach
Thank you

akabir@swin.edu.au