Prefetching and Caching in Localized Internet of Things Environment using Publish Subscribe Model

By Poorn Pragya (pp2157@rit.edu)

Advisor: Dr. Mohan Kumar

Introduction

• In this project, we use publish-subscribe model to prefetch and cache data for real time IoT applications.
• Prefetching information on anticipation can significantly improve the performance of applications and overcome delay caused by lower bandwidths.

Objective

• Develop a publish subscribe scheme for prefetching data items for real time IoT applications.
• Demonstrate improved performance of application through implementation for a sample application.

Background

• Publish Subscribe Model[5] is an already existing messaging pattern where senders of the messages can send different class of messages.
• The receivers can express interest in receiving a certain type of messages by senders.
• Prefetching[3] is technique to fetch relevant data from other devices before there is actual need of that data.

Design and Implementation

The application developed to demonstrate the above features is Pub-Sub Market Place
• a peer to peer application based on Publish Subscribe Model.
• relies on the fact that NOT all nodes are within the each other’s range all the time with good bandwidth conditions.
• it is necessary to prefetch and cache published items before the publisher goes out of network.
• It is also necessary to prefetch before subscribers starts viewing items to improve efficiency.

Users who use the app and act as subscribers can subscribe to the items that they intend to buy.

Users who act as publishers can publish can fruit items: bananas, mango, apple, strawberry, blueberry items

Prefetching

• C – Prefetch class (Yes / No)
• X – Input Variables (Item, Time)
• P(X) – Probability of input
• P(C) – Probability of class
• P(X | C) – probability of prefetch and its input variable
• P(X | C) – conditional probability of input with respect to prefetch

P(X | C) is calculated as follows:

P(X | C) = P(X | C) * P(C) / P(X)

Caching

LRU based scheme: Packet size and oldness of packet are used for cache replacement. Data Structure - Hash table backed by doubly Linked List.

### RESULTS

<table>
<thead>
<tr>
<th>Publishe r 1</th>
<th>Publishe r 2</th>
<th>Total # of items</th>
<th># of published items</th>
<th>Subscriber 1</th>
<th>Subscriber 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>With Prefetching</td>
<td>Without Prefetching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cache Hits</td>
<td>Fetch Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>12.26</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>6</td>
<td>25.26</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>15</td>
<td>10</td>
<td>30.56</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>20</td>
<td>9</td>
<td>67.43</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>25</td>
<td>12</td>
<td>78.47</td>
<td>7</td>
</tr>
</tbody>
</table>

Total Fetch Time for all 75 items in ms

213.98 331.16 292.05 430.06

Conclusion and Future Work

• As shown in the performance results, using prefetching mechanism speeds up the efficiency of the application in terms of run-time and improves the cache hit ratio.
• It would interesting to see how catch hit ratio varies when different regression models are used for prefetching and to verify if prefetching can be further improved.

REFERENCES