XML Query Result Size Estimation for Small Bandwidth Devices

Stefan Böttcher, Sebastian Obermeier, and Thomas Wycisk

University of Paderborn
Fürstenallee 11
33102 Paderborn, Germany
{stb,so,thwycisk}@upb.de

Abstract. Whenever mobile ad-hoc networks are used as a large data storage, a huge number of queries requesting the same information again and again can slow down the network and drain battery power. In this paper, we introduce an example application, the query classes that are required to be supported, and show up two possible caching methods. The two caching methods are based on the concept of query shipping and data shipping, respectively. Since our caching strategies can be used simultaneously, the decision for doing data shipping depends among other aspects on the overhead of transferred data. We explain why an XML Query Result Size Estimator can assist the application in the question of which mechanism should be used for a certain query, and point to other related estimation techniques.

1 Introduction

Today, ad-hoc networks are researched and used in different contexts for different purposes. Whenever database functionality is needed, the property that ad-hoc networks consist of multiple devices offers new possibilities in terms of interpreting the whole network as a distributed database. This gives each mobile device the possibility to own a local data store, where gathered data can be collected and offered to other devices. In addition, devices may search and query other data stores for certain data. However, besides the limited bandwidth of mobile ad-hoc networks, the limited battery power of devices causes the problem that frequently requested data may have a high effect on up-time of the ad-hoc network. In such a case, caching is considered as a good means to save bandwidth and energy. However, the success of caching depends among other issues on the used caching method. We investigate two possible caching strategies and show why a *Query Result Size Estimator* that gives estimations on the query result size is helpful for setting up a query that reduces the amount of transferred data.

The remainder of the paper is organized as follows. We introduce our application scenario and the supported query classes in Section 1.1. Furthermore, in Section 2, we describe the alternative querying technologies query shipping and data shipping, and explain why the query result size is important in order to decide between these querying technologies. Then, we list the requirements to a query size estimator (Section 3) and compare these requirements with existing estimators (Section 4). Finally, Section 5 concludes the paper.

R. Cooper and J. Kennedy (Eds.): BNCOD 2007, LNCS 4587, pp. 172–175, 2007.

1.1 Example Application Scenario

We focus on the following mobile auction application, where the mobile devices form a mobile ad-hoc network on a flea market. Each participant of the flea market may offer, buy, and search articles with his mobile device. Articles are either services or goods. Since the articles can contain details like descriptions, pictures, or videos, we choose the flexible XML format for representation. Whenever users search for items or bid, a multi-hop routing will enable a wide physical area to be covered.

When we look closer at the issued XPath queries in such a scenario, we can see that no arbitrary queries are executed. Instead, the application GUI supports the user when he starts searching for certain items. This means that only a small number of query classes following a few query templates must be supported in the context of this application. We identified the following query classes that are executed in a mobile ad-hoc flea market application:

Q_1**:** simple path expressions, e.g. searching for product title, current price, description, etc.

Q_2**:** simple path expressions containing existential filters, e.g. queries searching for items with a picture.

Q_3**:** queries containing category filters, e.g. filters selecting items of the category music or DVD.

Q_4**:** queries containing range filters, e.g. filters selecting items of a certain price range.

Q_5**:** queries containing functions, e.g. a substring function for returning items containing a certain keyword.

2 Query Shipping vs. Data Shipping

Whenever a user U of our mobile auction application issues an XPath query Q_x for a certain data store DS, the following two possibilities exist

Query shipping means the query Q_x is sent to DS via multiple hops. The result is calculated and sent back to U, who can directly display the result.

Data shipping means that the XML document owner splits the document into pairwise disjunct segments $S_1 \ldots S_n$, and informs all clients beforehand about this segmentation schema. When U issues the query Q_x, the client U itself identifies which XML segments are required to answer the query locally by U, and requests these segments. In this case, the network functions as a large file server, since U, and not the database, is responsible for the evaluation of Q_x.

Query shipping has the advantage that transferring a calculated result often involves less data transfer than transferring the data that is necessary to calculate the result locally.

The advantage of data shipping concerns the simple re-use of cached data. A node that routes a request for a certain XML segment must only compare its cached segments