Query Management in a Sensor Environment

Martin F O'Connor
Interoperable Systems Group
Dublin City University
moconnor@computing.dcu.ie

Vincent Andrieu
Interoperable Systems Group
Dublin City University
vandrieu@gmail.com

Mark Roantree
Interoperable Systems Group
Dublin City University
mark.roantree@computing.dcu.ie

Abstract

Traditional sensor network deployments consisted of fixed infrastructures and were relatively small in size. More and more, we see the deployment of ad-hoc sensor networks with heterogeneous devices on a larger scale, posing new challenges for device management and query processing. In this paper, we present our design and prototype implementation of XSense, an architecture supporting metadata and query services for an underlying large scale dynamic P2P sensor network. We cluster sensor devices into manageable groupings to optimise the query process and automatically locate appropriate clusters based on keyword abstraction from queries. We present experimental analysis to show the benefits of our approach and demonstrate improved query performance and scalability.

1. Introduction

The interest in and growth of Peer to Peer (P2P) systems in both academia and industry is on the increase. The application domain in which P2P networks initially achieved both success and notoriety was with file sharing systems such as Napster, Kazaa and Gnutella. However, the use of P2P systems have matured beyond this domain and is now being deployed in business (on demand video streaming systems), military (communications and reconnaissance ad-hoc mobile networks) and telephony (Skype).

An area of active research is the utilisation of P2P systems for the management of dynamic sensor networks. In this paper, we focus on the deployment of a P2P system as the underlying architecture for the XSense project. The goal of the XSense project is to construct and implement a large-scale database architecture with Query and Metadata services supporting an underlying arbitrarily large dynamic P2P sensor network. The Use Case presented in this paper is based on the CDVPlex Biometric Cinema [1]. In the CDVPlex study, new sensor technology and biometric measurement techniques developed by researchers in Health and Sports science were used to gather the physiological reactions of movie viewers as they watch a film.

1.1. Motivation and Contribution

The goal of the CDVPlex study was to detail the data gathering process, not to provide an analysis of the observed responses. Indeed the lack of a logical Data Management layer and Query service identified a gap that needed to be addressed. The motivation behind our work is the construction of a digital representation of a large cinema audience in order to test responses to various stimuli. It is envisaged that the sensor network will continue to grow and expand and thus requiring an architecture capable of supporting an arbitrarily large-scale sensor network. We use a logical P2P architecture to represent the human and sensor devices. What is needed to support the management of sensor data generated is the construction of a *virtual cinema* based upon a physical network of sensor devices. This virtual cinema will thus provide a digital interface for data and query management.

The contribution of this paper is the specification and development of a Data Management, Metadata and Query service for the virtual cinema on top of an underlying dynamic sensor network. We have built a prototype simulating a large cinema audience using a P2P architecture. We cluster sensor devices into manageable groupings to optimise the query process. Due to the fact that we have constructed the digital cinema, we can *move* individuals into different clusters in a virtual manner. This provides the ability for different users to form part of different study groups. We automatically locate appropriate clusters based on keyword extractions from queries. We describe our

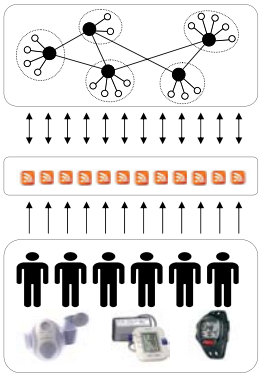

Figure 1. XSense Architecture Model.

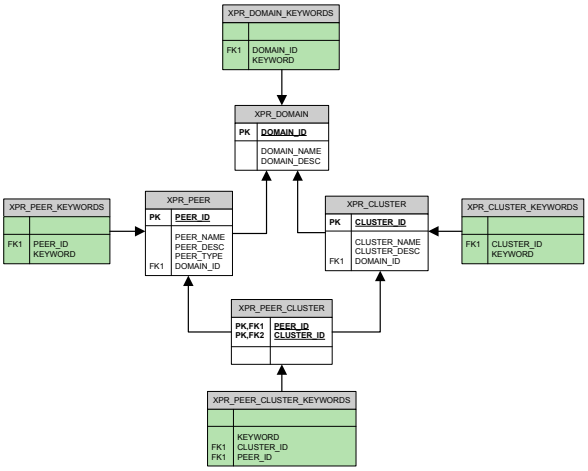

Figure 2. Repository Metadata for the XSense P2P Network.

implementation and provide detailed analysis of our experiments and a demonstration of improved results by tweaking the distribution of peers.

This paper is structured as follows: in §2 the XSense architecture is presented and described; in §3 the Query Management layer is detailed. In §4 our query processing strategy is presented and in §5 a detailed analysis of our experiments is provided. In §6 we present our related research and in §7 we provide our conclusions.

2. The XSense Architecture

In classic distributed database management systems, an efficient query service was made possible through the key features of central control and global knowledge (via a global schema). The goal of the XSense architecture is to facilitate an improved query service and minimise the cost of broadcast messages between peers while addressing the challenges of a large scale, decentralised, dynamic P2P network. The XSense architecture illustrated in Figure 1 consists of three logical layers: the Physical (or Human/Device) layer, the Virtual (or Logical) layer and the Query Management layer. Due to its significant content, the Query Management layer shall be described in §3.

2.1. Physical Layer

The Physical or Human/Device layer represents the physical sensor devices and the humans to which they are attached. In our Use Case based on the CDVPlex study, the sensors gather biometric data recording physiological reactions by human subjects as they are viewing movies in a large cinema. An in-depth description of the data gathering process and the sensors used may be found here [1].

2.2. Virtual Layer

The Virtual Layer is developed and implemented using JXTA [2]. JXTA is a P2P platform from Sun Microsystems providing a set of 6 core protocols that allow distributed client interoperability. Each peer in our P2P network represents one sensor device attached to one human. Multiple sensor devices may be attached to the same human.

In order to process the sensor data, it is necessary to enrich the data by adding structure and semantics to facilitate manipulation by query languages. The provision of a simple template for each sensor type mapping the raw sensor data to the enriched data format provides the necessary functionality enabling our virtual P2P network to be interoperable with any number and type of sensor devices.

3. Query Management Layer

The Query Management layer exploits and extends the XPeer architecture [3] with metadata repositories. The XPeer architecture defines a mediation system that allows querying a large scale P2P Database Management System by hiding the distribution, localisation and heterogeneity of data sources while providing reasonable query response times.

3.1. Metadata Management

The purpose of the metadata repositories is to store and maintain the metadata describing the role of each peer in the P2P network and the keywords associated with them. A UML diagram of the metadata repository

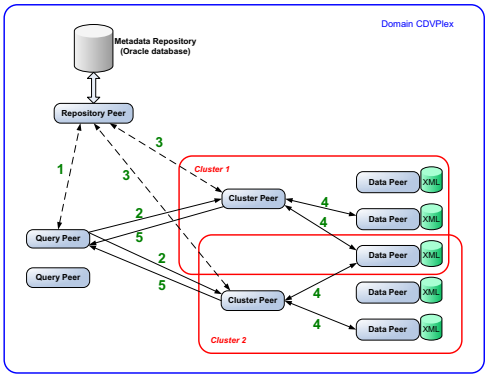

Figure 3. XSense Query Processor.

is presented in Figure 2. Each system has one or more Domain peers and each Domain peer may contain one or many Cluster peers. In a similar fashion, each cluster may contain one or many Data peers as described in [3]. The black nodes in Figure 1 illustrate the Cluster peers. Additionally, the system instantiates and maintains Query and Repository peers to provide both query and repository interfaces as required by the JXTA platform.

3.2. System Configuration

A *peer* is a node in the JXTA network. A JXTA peer in the Java implementation is associated with one Java Virtual Machine (JVM). Having only one peer per device, such as a PC, is the default scenario. Multiple JVMs are required to create multiple peers on a single PC. However, even a relatively small number of peers result in memory issues on a modern PC. Thus, in order to optimise the management of peers on a single machine in the Virtual layer, each peer is implemented as a java thread and not a JVM process.

The current version of the XSense prototype contains 268 peers and all CDVPlex peers may be launched simultaneously.

- 221 Data peers where each Data peer represents a single sensor output (as an XML document).
- 39 Cluster peers. The breakdown is as follows: 33 movie clusters (one for each movie) and one cluster for each of the following; body sensors, smart chair sensors, heart rate sensors, chest belt sensors, female viewers and male viewers.
- 6 Repository peers (arbitrary number).
- 2 Query peers (to simulate 2 clients).

It should be noted at system startup, the Query peers and Cluster peers must access the metadata repository upon instantiation to retrieve information about themselves because initially all peers are generic.

4. Query Processing

The Query Processing strategy used in the XSense project is illustrated in Figure 3. Data peers (and indeed Cluster peers) can be associated with one or more keywords. A query may also contain one or more keywords. Thus a Query peer receives a query and associated keywords to target the relevant cluster(s). Query processing proceeds as follow (refer to Figure 3 for an illustration of each step):

1) The Query peer accesses the repository for the IDs of the Cluster peer(s) matching the query keyword(s).
2) The Query peer queries those Cluster peers.
3) The Cluster peer requests (from the repository) the IDs of its Data peers matching the query keyword(s).
4) The Cluster peer queries those Data peers.
5) The Cluster peer aggregates the Data peers results and sends the entire result back to the Query peer.

In our current implementation, the keyword(s) are identified and stripped from the query itself. Although the static encoding of keywords within the metadata repository is sufficient at this stage in order to produce a working prototype for the XSense architecture, it is evident a dynamic encoding and keyword matching algorithm is required for a more robust P2P query processor. In [4] the authors present an approach for approximate XML query processing (supporting top-k queries) over distributed dynamic collections of XML data based on a *clustered* path index. The use of clustered path indexes to partition similar data on the same superpeer would facilitate reduced communications costs between peers, while achieving a higher pruning degree during query processing and eliminate the need for the static encoding of keywords. An adaptation of cluster path indexes to enable dynamic keyword encoding and its application for the XSense project remain as future work.

5. Experimental Evaluation

In this section we provide experimental analysis for our methods. The principle questions we tried to answer were:

1) Is the XSense architecture suitable for managing dynamic data sources in a large P2P network?
2) Does our approach provide improved query response times?
3) Does our approach adapt favorably with regard to scalability?

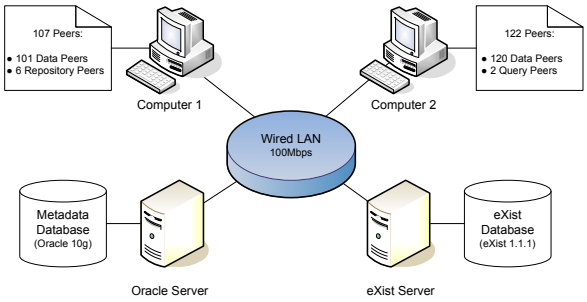

Figure 4. Network Configuration of the Chaotic P2P Network.

The P2P network is implemented using the JXTA platform in Java and runs on two 3.2GHz Pentium IV machines running Windows XP Professional, each with 1GB of RAM. The Java Virtual Machine hosting the P2P network is Sun JVM version 1.5. The Oracle 10g database (hosting the metadata repositories) runs on a Fedora 7 platform with a Pentium IV 3.2GHz CPU and 1GB of RAM. The open source eXist XML database [5] (hosting the underlying data for all Data peers) runs on a similar hardware and software specification as the Oracle server.

5.1. Benchmarking the Virtual P2P Network

Before meaningful analysis may be performed, it is necessary to evaluate and benchmark the P2P network in its default chaotic state. In this experiment, there are no clusters. There is no repository. It is a pure P2P network with no notion of Super peers, only equal Data peers. When a query is initiated, the entire P2P network is queried and all Data peers (body sensors, heart rate monitors, smart chair sensors and chest belt sensor peers) will process the query. The P2P network consists of 221 Data peers, 6 Repository peers and 2 Query peers and is illustrated in Figure 4. It should be observed that the Repository peers are required to inform the Data peers of their role at launch time, but they are not used by the Query Processor.

Each query has been performed 10 times and the average response times are displayed in Table 1.

#	Query	Times
Q1	//@deviceId/string()	7.36s
Q2	//measurement/@name/string()	9.08s
Q3	//startDateTime	10.43s
Q4	//interval	12.65s
Q5	//interval[MinHR='66']	7.78s

Table 1. Query Response times in the Chaotic P2P Network.

5.2. Using the XSense architecture

In this experiment, the P2P network is configured according to the XSense architecture: Data peers are grouped into one or more clusters, with associated keywords, and Repository peers act as an interface (as required by JXTA) to the Oracle database maintaining the cluster configuration. Using the XSense architecture it is not necessary to query the entire P2P network but only the relevant clusters (by choosing relevant keywords). On the other hand, the Cluster peer(s) will have to communicate with a Repository peer to identify the relevant Data peers and this will incur an overhead. The cost of this overhead shall be examined shortly.

The XSense P2P network has 221 Data peers, 6 Repository peers, 2 Query peers and 39 Cluster peers.

5.2.1. Evaluating Queries in the XSense P2P Network. We report on the same queries and their performance over a cluster of sensors: all body sensors, all heart rate monitors, all smart chair sensors and all chest belt sensors. In each case, a single cluster is queried. Figure 5 displays a table listing a number of XPath queries processed on different sensor clusters and their response times.

The query response times are largely proportional to the number of Data peers to be queried: we obtained the fastest results for the Chest Belt cluster (10 Data peers) and the slowest for Body Sensor (75 Data peers) and Heart Rate (70 Data peers) clusters. Depending on the targeted cluster size, the query response times are 3 to 15 times faster using the XSense architecture. The communication between Cluster Peers and the Repository Peer is very fast (0.053 seconds on average) and is performed only once per query. Thus, the cost incurred of communication with the Repository peer is very small while cluster targeting facilitates

Cluster Type (DataPeers)	#	Query	Times
Body Sensor (75)	Q1	//@deviceId/string()	2.653s
	Q2	//measurement/@name/string()	2.699s
	Q3	//startDateTime	3.291s
	Q4	//interval	3.262s
Heart Rate (70)	Q5	//@deviceId/string()	2.340s
	Q6	//measurement/@name/string()	2.463s
	Q7	//startDateTime	3.775s
	Q8	//interval[MinHR='66']	2.666s
	Q9	//interval[MinHR='66']/MaxHR	2.323s
	Q10	//interval[MaxHR/number()>200]	2.292s
Smart Chairs (33)	Q11	//@deviceId/string()	1.583s
	Q12	//measurement/@name/string()	1.770s
Chest Belt (10)	Q13	//@deviceId/string()	0.517s
	Q14	//measurement/@name/string()	0.454s
	Q15	//Params	0.564s
	Q16	//Params/Length	0.485s
	Q17	//VO2max	0.501s

Figure 5. Query Response Times using the XSense architecture.

a significant improvement in performance times; the XSense architecture permits faster query processing.

5.2.2. Querying Dynamic Clusters. In this section, we report on queries executed over more dynamic and smaller clusters. The logical P2P architecture and metadata service provided by the XSense architecture facilitate the construction of dynamic clusters at query time based on keyword extraction. This enable queries in the virtual cinema that would not be possible at the physical level. If the queries were posed at the physical level, all Data peers would have to be queried. By querying a dynamically constructed cluster, only those Data peers whose keyword matches the query keyword (as described in §4) are actually queried. Figure 6 reports on queries executed over two small clusters, this time grouped by film type and not by sensor device. The second column, CP(DP), shows the number of Cluster peers receiving the query and the number of Data peers processing the query. The response times are more than an order of magnitude faster than that provided by the chaotic P2P network in which all Data peers are queried.

Keyword	CP(DP)	#	Query	Times
`Shrek`	1(5)	Q19	//@deviceId/string()	0.266s
		Q20	//measurement/@name/string()	0.269s
		Q21	//title	0.297s
`Harry.*`	3(14)	Q22	//@deviceId/string()	0.438s
		Q23	//measurement/@name/string()	0.484s
		Q24	//title	0.469s

Figure 6. Query Response Times over Dynamic Clusters.

5.3. Network Tweaking for Performance Enhancement

We analysed query performance at each stage of the query processing strategy as outlined in §4 and identified a potential bottleneck at step 4 (illustrated in Figure 3). When the Cluster peer queries the Data peers, the slowest stage of the entire query process occurs between the *beginProcessingXMLQuery* checkpoint and the *XMLQueryProcessed* checkpoint. On one hand, this is to be expected as querying the Data peers constitutes the bulk of the query operation. However, for the purposes of our experiments the P2P network is constructed such that the underlying data belonging to all Data peers is stored in one eXist database. Thus, a Data peer must query the eXist database in order to retrieve its data. This centralised server approach for maintaining the Data peer's data is a bottleneck when the number of data peers queried is large. In a real world scenario each Data peer will manage its own data. In order to evaluate the impact of having only one data store (eXist XML database) in our P2P network, we performed another set of experiments using three eXist XML databases as data stores. Due to a practical limitation on the number of machines available, the two new eXist XML databases were installed on `Computer 1` and `Computer 2` (illustrated in Figure 4). An analysis of the results (which are not displayed due to lack of space) revealed the gap between the *beginProcessingXMLQuery* and *XMLQueryProcessed* checkpoints was reduced significantly (by two orders of magnitude). These results confirm that many of the limitations we encountered in our experiments were due to processing power and resource limitations. We simulated a large P2P network using only two machines. In our experiment, `Computer 1` and `Computer 2` hosted 268 peers (Data peers, Cluster peers, Repository peers and Query peers) as well as the eXist XML databases. In a real world scenario, Data peers will manage their own data, reside on many machines and will far outnumber (relatively speaking) Cluster peers, and thus the XSense architecture will reap performance gains from real large scale distributed and parallel processing of queries in a P2P network.

6. Related Research

In this section, we examine related research under two different criteria: the ability to provide generic frameworks for sensor management and querying and the ability to generate meaningful semantics for complex sensor data (eg. multimedia).

In [6], they propose Virtual XML, an approach to an XML-based virtualization of data resources. They describe how to support the processing of non-XML data without explicit conversion to XML. The key contribution of their work is that no conversion of sensor data is necessary as they create a view definition to interpret the raw data. However, the template system they provide has not been applied to any domain (rather they provide some use-case descriptions) and no query evaluations are possible.

In [7], the authors process and query the raw data streams and avoid conversion to XML. Their Semantic Streams model contains two fundamental elements: *event streams* and *inference units*. Event streams represent real world events recorded by sensors and have properties such as time, location and speed. Inference units are processes that operate on event streams. They infer semantic information from events and either generate new event streams or add the information to existing events as new properties. Their usage of

constraints (eg: Quality of Service) on the data streams provides for a useful query mechanism for approximate queries. However, the focus of this work is on semantic interpretation facilitating declarative queries of sensor data.

In [8], they employ the concept of proximity queries whereby network nodes monitor and record interesting events in their locality. While their results are positive in terms of cost, queries are still at a relatively low level (no common format for query expression) and query processing is not designed for large scale distributed networks.

In [9], they provide for semantic clusters within their sensor network. They generate metadata either statically based on application semantics, or dynamically using inference rules and thus, support query processing. However, they observed the computational cost of the metadata generation process increases significantly as the number of sensor devices increased and thus the current solution does not scale well.

7. Conclusions

This paper presents the XSense architecture to support the construction of a virtual cinema based upon a physical network of sensor devices. The logical P2P network represents the human and physical devices. We provide a Query Management layer clustering devices into manageable groupings to optimise query processing. These clusters are dynamically constructed based on keyword abstractions, allowing us to virtually move humans and devices into different study groups as required. We provide an implementation and detailed analysis of our experiments, demonstrating improved query performance and scalability with our approach.

We have two principle directions for future research. The first, as previously mentioned in §4 is the development of a dynamic keyword encoding and keyword matching algorithm and in particular, an examination of the suitability of clustered path indexes to achieve this end. The second direction focuses on the inherent tree-like properties of clusters of sensors residing inside specified domains. In particular, we will investigate optimisation of distributed queries by extending techniques previously developed for XML trees [10].

Acknowledgment

This work is funded by the Irish Research Council for Science Engineering and Technology grant no. CNRS PICS/2006/04

References

[1] S. Rothwell, B. Lehane, C. H. Chan, A. F. Smeaton, N. E. O'Connor, G. Jones, and D. Diamond, "The CDVPlex Biometric Cinema: Sensing Physiological Responses to Emotional Stimuli in Film," in *Proc. International Conference on Pervasive Computing (Pervasive)*, Dublin, Ireland, May 2006. [Online]. Available: http://doras.dcu.ie/364

[2] L. Gong, "Project JXTA: A Technology Overview," Apr 2001. [Online]. Available: http://aot.ce.unipr.it/documentation/jxta/TechOverview.pdf

[3] Z. Bellahsène and M. Roantree, "Querying Distributed Data in a Super-Peer Based Architecture," in *Proc. International Workshop on Database and Expert Systems Applications (DEXA)*, Zaragoza, Spain, Sep 2004, pp. 296–305.

[4] G. Koloniari and E. Pitoura, "A Clustered Index Approach to Distributed XPath Processing," in *Proc. International Conference on Data Engineering (ICDE)*, Cancún, México, Apr 2008, pp. 1516–1518.

[5] W. Meier, "eXist: An Open Source Native XML Database," in *Proc. Web-Services, and Database Systems, (NODe)*, Erfurt, Germany, Oct 2002, pp. 169–183.

[6] K. H. Rose, S. Malaika, and R. J. Schloss, "Virtual XML: A Toolbox and Use cases for the XML World View," *IBM Systems Journal*, vol. 45, no. 2, pp. 411–424, Jan 2006.

[7] K. Whitehouse, F. Zhao, and J. Liu, "Semantic Streams: A Framework for Composable Semantic Interpretation of Sensor Data," in *Proc. Wireless Sensor Networks, Third European Workshop, (EWSN)*, Zurich, Switzerland, Feb 2006, pp. 5–20.

[8] Y. Kotidis, "Processing Proximity Queries in Sensor Networks," in *Proc. Workshop on Data Management for Sensor Networks, in conjunction with VLDB, (DMSN)*, Seoul, Korea, Sep 2006, pp. 1–6.

[9] H. Kawashima, Y. Hirota, S. Satake, and M. Imai, "MeT: A Real World Oriented Metadata Management System for Semantic Sensor Networks," in *Proc. Workshop on Data Management for Sensor Networks, in conjunction with VLDB, (DMSN)*, Seoul, Korea, Sep 2006, pp. 13–18.

[10] M. F. O'Connor, Z. Bellahsène, and M. Roantree, "An Extended Preorder Index for Optimising XPath Expressions," in *Proc. Third International XML Database Symposium, (XSym)*, Trondheim, Norway, Aug 2005, pp. 114–128.