Semi-Automatic Semantic Enrichment of Raw Sensor Data*

Nicolas Legeay[1], Mark Roantree[2], Gareth J.F. Jones[1], Noel E. O'Connor[1], and Alan F. Smeaton[1]

[1] Centre for Digital Video Processing & Adaptive Information Cluster, Dublin City University, Ireland
[2] Interoperable Systems Group, Dublin City University, Ireland - mark@computing.dcu.ie

Abstract. One of the more recent sources of large volumes of generated data is sensor devices, where dedicated sensing equipment is used to monitor events and happenings in a wide range of domains, including monitoring human biometrics. In recent trials to examine the effects that key moments in movies have on the human body, we fitted fitted with a number of biometric sensor devices and monitored them as they watched a range of different movies in groups. The purpose of these experiments was to examine the correlation between humans' highlights in movies as observed from biometric sensors, and highlights in the same movies as identified by our automatic movie analysis techniques. However, the problem with this type of experiment is that both the analysis of the video stream and the sensor data readings are not directly usable in their raw form because of the sheer volume of low-level data values generated both from the sensors and from the movie analysis. This work describes the semi-automated enrichment of both video analysis and sensor data and the mechanism used to query the data in both centralised environments, and in a peer-to-peer architecture when the number of sensor devices grows to large numbers. We present and validate a scalable means of semi-automating the semantic enrichment of sensor data, thereby providing a means of large-scale sensor management.

1 Introduction

We are currently witnessing a groundswell of interest in pervasive computing and ubiquitous sensing which strives to develop and deploy sensing technology all around us. We are also seeing the emergence of applications such as environmental monitoring and ambient assisted living which leverage the data gathered and present us with useful applications. However, most of the developments in this area have been concerned with either developing the sensing technologies, or the infrastructure (middlewear) to gather this data and the issues which have been addressed include things like power consumption on the devices, security

* This work is partly supported by Science Foundation Ireland under Grant No. 03/IN.3/I361

of data transmission, networking challenges in gathering and storing the data and fault tolerance in the event of network and/or device failure. If we assume these issues can be solved, or are at least will be addressed successfully in the short term, we are then left to develop applications which are robust, scalable and flexible, and at such time the issues of high-level querying of the gathered data becomes a major issue.

The problem we address in this paper is how to manage, in an efficient and scalable way, and most importantly in a way that is *flexible* from an application developer or end user's point of view, large volumes of sensed and gathered data. In this, we have a broad definition of sensor data and we include raw data values taken directly from sensor devices such as a heart rate monitor work by a human, as well as *derived* data values such as the frame or time offsets of action sequences which appear in a movie. In the case of the former there would be little doubt that the heart rate monitor readings are sensor values, whereas the latter still corresponds to data values, taken from a data stream, albeit with some intermediate processing (audio-visual analysis in this case). We now describe the motivation for our work.

1.1 Motivation and Contribution

To design a scalable system to manage sensor data, it is firstly necessary to enrich the data by adding structure and semantics in order to facilitate manipulation by query languages. Secondly, in order to improve efficiency the architecture should be suitably generic to make it applicable to other domains. Specifically, it should not be necessary to redesign the system or write new program code when new sensor devices are added. Finally, when the number of sensor devices increases to very large numbers, the system should be capable of scaling accordingly.

The contribution of the research work reported here is the development of an architecture that is both generic, and has the capability to scale to very large numbers. In this respect, our XSENSE architecture facilitates the addition of new sensor devices by requiring that the knowledge worker or user provides only a short script with structural information regarding the sensor output. Scalability is provided in the form of a Peer-to-Peer (P2P) architecture that classifies sensors into clusters, but otherwise contains no upper limit on the numbers of sensors in the network.

The paper is structured as follows: in §2, a description of sensor networks is provided and in particular the sensor network we use in our experiments, together with the issues involved in this particular domain; in §3, we describe our solution to problems of scale and processing by way of a processing architecture that transforms raw data and provides semantically rich files; in §4, we provide scalability by removing the centralised component and replacing it with a Peer-to-Peer Information System; in §5, we demonstrate workable query response times for distributed queries; a discussion on related research is provided in §6 and finally in §7, we offer some conclusions.

2 Sensor Network Background

In previous work [9], we reported a study conducted to investigate the potential correlations between human subject responses to emotional stimuli in movies, and observed biometric responses. This was motivated by the desire to extend our approach to film analysis by capturing real physiological reactions of movie viewers. Existing approaches to movie analysis use audio-visual (AV) feature extraction coupled with machine learning algorithms to index movies in terms of key semantic events – dialogs, exciting sequences, emotional montages, etc. However, such approaches work on the audio-visual signal only and do not take into account the visceral human response to viewed content. As such, they are intrinsically limited in terms of the level of semantic information they can extract. However, integrating and combining viewer response with AV signal analysis has the potential to significantly extend such approaches toward really useful semantic-based indexing.

For the study, we created a controlled cinema-like environment and instrumented both this and movie watchers, in a variety of ways. We also performed our AV analysis on all films watched by our viewers and synchronised these analysis results with the captured biometric responses. The instrumented environment, termed the "CDV*Plex*", was designed to replicate a true cinematic experience as closely as possible. It corresponded to an air-conditioned windowless room with comfortable seating for up to 4 people, in which a Dolby 5.1 surround sound system, DVD player and large-screen digital projector were installed.

We gathered a total of 6 biometric sensor data feeds from each of our participants, via 3 different sensor devices, as follows:

- **Polar S610i™ heart-rate monitor**. This consists of a fabric band which fits around a person's chest and detects and logs their *heartrate*, sampled every few seconds.

- **BodyMedia SenseWear®**. This sensor array is worn around the upper arm and measures and logs the following: *galvanic skin response*, a measure of skin conductivity which is affected by perspiration; *skin temperature*, which is linearly reflective of the body's core temperature activities; *heat flux* which is the rate of heat being dissipated by the body; *subject motion* using an in-built accelerometer.

- **Smart Chairs**. Each of the chairs used had a specially designed foam-based pressure sensor [2] integrated into its backrest to record changes in viewer *posture*.

The participants were 43 staff and student volunteers from across the university. In total, 37 full length feature films of 10 different genres (e.g. Action/Adventure, Animation, Documentary, Horror, etc.) were shown, resulting in over 500 hours of recorded biometric data from the set of sensors.

As outlined in [9], this gathered data, when combined with automatically detected movie events (see section 3.1) is potentially a hugely valuable resource for

modeling and integrating human responses with automatic content structuring and indexing. Unfortunately, the value of this resource is significantly reduced in the absence of a seamless and efficient means to perform semantic queries against this repository. In this paper, we report our work on using XML for the semantic enrichment of this gathered sensor data.

2.1 Describing the Network

In order to employ any data management utility for a large volume of information such as is the case here, a user needs a compact yet detailed description of the overall system components, in our case the sensors, their data, and how they inter-operate. In figure 1, the Sensor Network is represented as an UML class diagram. The principal class is the `Experiment` class, where one instance is created for each experiment. Each experiment requires a single movie (`Movie` class) and multiple viewers (`Viewer` class). The Movie has a set of static attributes associated with that movie, together with dynamic data (captured by the `Event` class) that is generated as the movie is processed by our video event detection software [5]. Each Viewer is associated with a `Person` class that captures static information about the viewer, and four other classes containing dynamic information: 3 types of `Sensor` class and a single `Feedback` class.

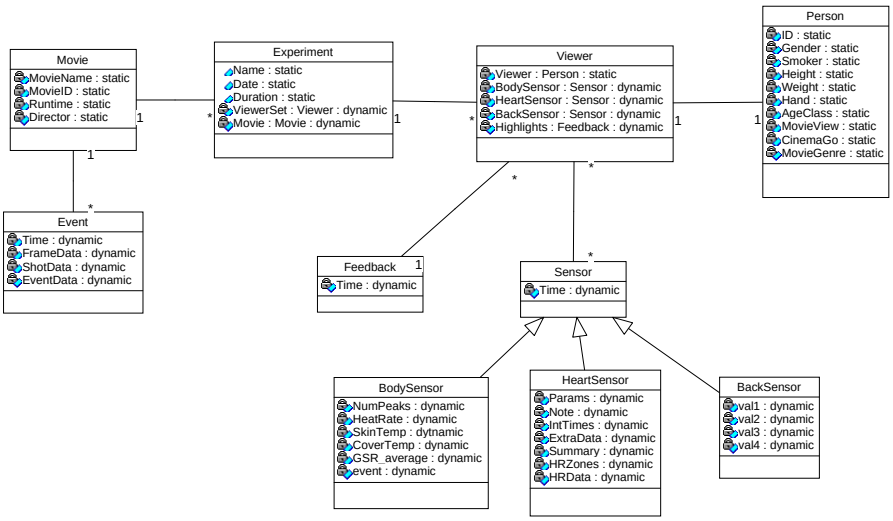

Fig. 1. Class Model for the Sensor Network

One property of these experiments that cannot be captured in Figure 1 is the Time dependency across all of the classes containing dynamic information.

All experiments are time-related and the classes `Sensor`, `Feedback` and `Event`, are bound together using the start time of the movie.

2.2 Calibration and Normalisation Issues

Sensor networks generally have two forms of data: static and dynamic. Static data is not generated by sensors or from the video analysis process. It refers to information regarding the movie, the experiment or an individual person (a viewer). Static data generated during experiments includes Personal Details, Movie Preferences and Movie Opinions. Dynamic data includes movie semantics regarding scenes, shots, events and Movie Highlights & Feelings. Sensor data involves the generation of biometric sensor data, heart rate and backrest pressure on the sensors in the chairs.

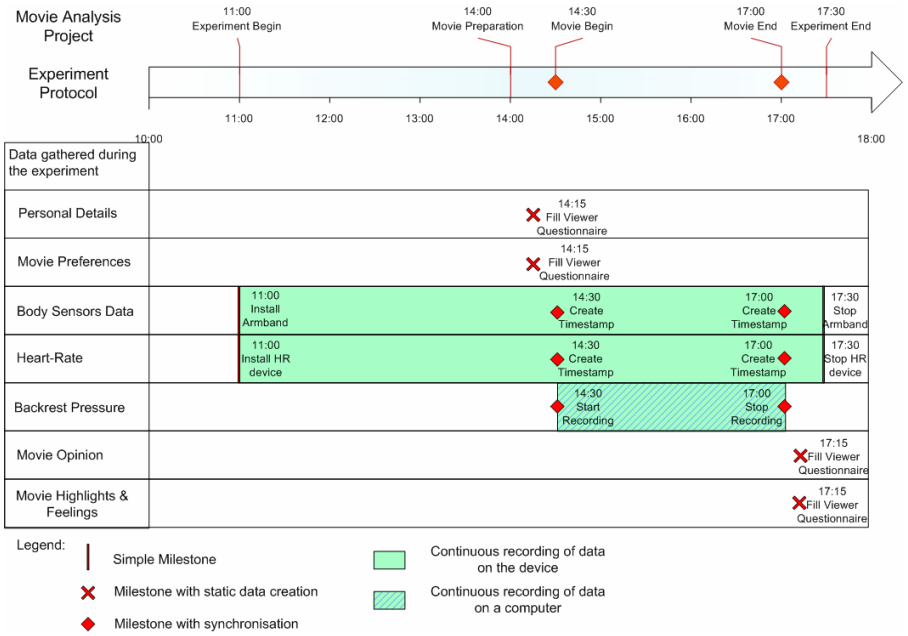

Fig. 2. Sensor Data Graph

Two important points are clear from figure 2: there is a strict timing relationship across sensor sources, and some experimental data will contain anomalies, eg. before watching the movies, participants logged up to 3 hours of biometric data in order to establish their baseline biometric values but the duration of this baseline varied considerably from person to person and from movie to movie. Thus, sensor output is influenced by the users' actions in many cases.

There is a synchronisation activiy carried out at the beginning and end of each movie showing movie for each of the sensors' measurements. A synchronisation process is also required to link information concerning viewers' reactions and movies' events. These are the events that have been identified using the semantic analysis of audio and video contents of the shots (described later in §3). Movie data includes all data related to a movie and is independent of Viewer data. Viewer data includes all data related to a viewer during an experiment for a movie and is generated during the experiments. They are three sources of sensors data: body sensor data generated by the armband, heart-rate given by the HR device and the backrest pressure as measured on the chairs. Thus, one of the issues for the sensor network is how to facilitate the calibration of sensor data by associating timing events across many of the sensor output streams.

3 XSENSE System Architecture

The XSENSE Architecture illustrated in figure 3 comprises six layers, with each pair of layers joined by a processor performing a key activity in the enrichment process. Layer 0 contains raw multimedia files that use processor P0 (described in the next section) to extract meaningful content. The files generated are textual and contain timing and event data that is quite similar to raw sensor data. At Layer 1, raw text files (both sensor and multimedia metadata files) are converted by Process P1 into basic XML files and using Process P2, stylesheets are automatically generated to enrich the basic XML files. By Layer 4, the output from the previous two layers are combined (by Process P3) to form semantically rich sensor data files. At this point, sensor files are autonomous with no relationship across files (indicating for example, that they were used in the same experiment or that their times are synchronised). The final process (P4) adds relationship semantics to link sensor files at the global layer.

3.1 Extracting Multimedia Semantics

In order to extract semantics from the video content, a process corresponding to P0 in figure 3, we employ a movie indexing framework capable of automatically detecting three different kinds of semantic events – dialogues between two or more characters, exciting sequences (e.g. car chases, fight scenes, etc) and emotionally laden musical sequences [18, 14, 13]. The approach relies on a series of AV feature extraction processes, designed to mimic well-known film creation principles. For example, when filming a dialogue sequence, the director needs to ensure that the audience can clearly interpret the words being spoken and uses a relaxed filming style with little camera movement, large amounts of shot repetition and clearly audible speech [15]. Conversely, when shooting an exciting part of a film, the director uses fast-paced editing combined with high amounts of movement [16]. Emotionally laden events, on the other hand, are shot with a strong musical soundtrack, usually combined with slower paced editing and filming style [15, 6].

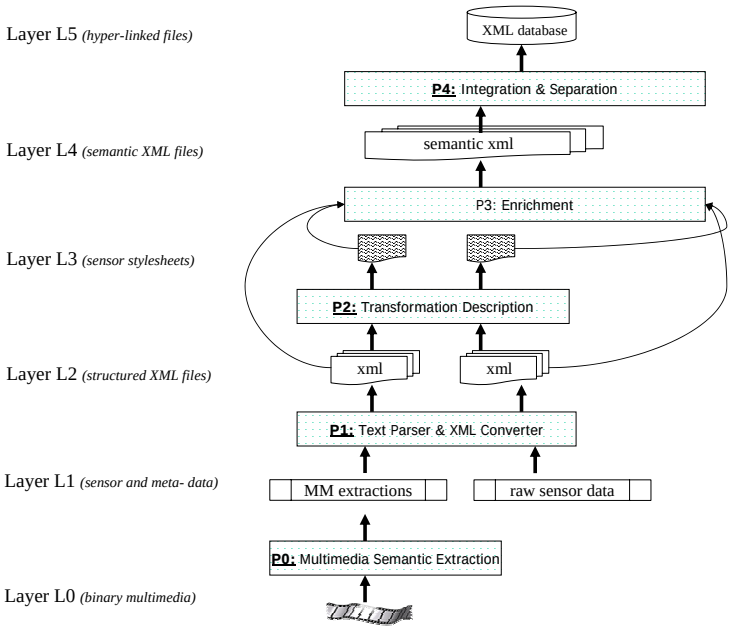

Fig. 3. XSENSE Processing Architecture

We extract a set of AV features to detect the presence of these characteristics. We characterise editing pace by detecting the rate of occurrence of *shot boundaries* using a standard colour histogram technique. To measure onscreen motion, we use MPEG-7 *motion intensity* for local motion (e.g. character movement) [17] as well as a measure of global *camera movement* [18]. A support vector machine based classifier is used to classify the audio track into one of: *speech*, *music*, *silence* and *other audio*. A set of Finite State Machines (FSMs) are then employed to detect parts of a film where particular combinations of features are prominent. For example, in order to detect dialogue events, we detect temporal segments which contain various combinations of speech shots, still cameras and repeating shots. Similar rules are employed for the other event types. An evaluation over ten films of very different genres and origins in [18], found that 95% of Dialogue events, 94% of Exciting events and 90% of Musical events as judged by multiple viewers were detected by the system, which indicates the usefulness of system for detecting semantically important sequences in movies.

3.2 Processors for Generating Sensor Semantics

Each of the layers in the XSENSE Architecture are joined by processors that operate to build the final sensor database. In this section, we provide a more detailed description of the functionality of each processor. The innovation in the

XSENSE architecture is its generic nature: it was designed to accommodate a heterogeneous collection of sensors. By providing basic scripts and the XSENSE Term Database (a small terminology database), XSENSE can integrate most or all sensor data formats.

Example 1. Raw Sensor Data
File : EVK_Action.evtkey
176 183 177 179 181
423 431 425 427 429

P1: Text to XML Conversion At layer 1, raw sensor files contain no semantic information and at this point, it is only possible to add structural information to the sensor file. Example 1 illustrates a subset of a multimedia event file. The output from this process is a basic XML file with structural semantics but not the real content semantics as required by a query processor. Firstly, a naming convention (located in the XSENSE Term Database) is applied to sensor data files to enable the parser to recognize the type of sensor file. For example, a file that contains action events is renamed to EVT_Action.events. Example 2 illustrates the same sensor data after structural information has automatically been added.

Example 2. XML Structure (output from P1)

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
    <event>
        <startShot>176</startShot>
        <endShot>183</endShot>
        <keyShot-group>
            <keyShot><value>177</value></keyShot>
            <keyShot><value>179</value></keyShot>
            <keyShot><value>181</value></keyShot>
        </keyShot-group>
    </event>
```

The XSENSE System uses ServingXML schema [11] to provide a generic process for incorporating new types of sensor devices. A ServingXML service describes the sequence of tasks that create a set of rules to coordinate the activities of the parser. The activity for the knowledge worker who is charged with managing and manipulating such sensor data is to provide a short script describing the file structure. The output for the next layer is a set of XML data files of the type shown in Example 2.

P2: Transformation Description The aim of this process is to create an XSLT stylesheet describing the transformations necessary to create a semantically enriched XML sensor file. Each sensor file has its own time format or

an implicit time interval (eg. a reading every second). In this situation, this XSLT stylesheet describes how to transform timing information into the system time format and how to normalize times. In XSENSE, there are currently four transformation categories.

- Semantic Transformation. Rules for file naming and deriving information from the content.
- Structural Transformation. Rules for changing entity or attribute names and for changing information groupings.
- Normalisation. Rules for content format normalization (i.e. date format), generating IDs and synchronising time information.
- File Transformation. Rules to merge several XML files, divide XML files, and generating new files.

The output from this process is the stylesheet for this sensor type. Once created, the stylesheet will be reused for all sensors of this type, eliminating P2 from the next iteration of the process.

P3: Semantic Enrichment The aim of the formatting process is to transform a basic XML file into a semantically enriched file by applying the XSLT stylesheet. Processors P1 and P2 generated the structured XML file and XSLT stylesheet respectively. The ServingXML service is updated by process P3 in order to facilitate the XSLT transformations and build the enriched data files. When a sensor that has been processed previously is detected by the system, the ServingXML service updates are not required.

Example 3. Enriched XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<events>
    <event type="Action">
        <startShot id="176"/>
        <endShot id="183"/>
        <keyShot id="177"/>
        <keyShot id="179"/>
        <keyShot id="181"/>
    </event>
```

Example 3 illustrates how the sensor data used in the prvious two examples can now be queried using the standard XPath language. The Saxon XSLT Processor [10] manages all transformations for this process.

P4: Integration and Separation At this point, XML files contain information relating to a single sensor file. The aim of this process is to add information from other sensors' files, to merge several XML files or to generate new files. Those transformations are once again described in XSLT stylesheets. This is necessary as most sensor networks contain sensors that while physically autonomous, operate in relation to other sensors to deliver the overall monitoring process.

In our sample domain, events files of different types are merged and sorted according to the time they occur in the movie. Sensor files are edited to ensure that links referring to the same experiment, heart monitor, film event etc. are preserved. This process ensures that users need not be aware of implicit relationships across the sensor network: instead they are explicitly created inside sensor documents. This provides an added advantage when sensor networks are very large and require a distributed architecture to manage scalability. In the Appendix, we provide a subset of the files created for the sensor data used in the examples in this section. This file is automatically created after user input (script data) to the ServingXML process.

3.3 Building Sensor and Movie Databases

The sensor network comprises 33 experiments, covering 29 movies, and a total of 171 sensor outputs, creating 316Mb of raw data. The XSense architecture requires that sensor files follow a specific naming convention and that binary files be converted to text files.

When all of the sensor data for a particular experiment is generated, this causes the creation of an experiment XML file (by processor P4). This experiment xml file contains timing information for both the movie and experiment. However, some information such as movie id, title and language are entered manually before they are stored in the XML (eXist) database. Table 1 presents the times for the creation of the database (approx 1 hour and 50 minutes).

	Input	Output	Time
P1 P3	171 sensor files (316MB) 171 raw XML files	171 raw XML files 188 enriched files (651MB)	 23 min
P4 (incl. manual editing)	188 enriched files (651MB)	221 linked files (651MB)	1h 27min (+ 27 min)
Total	171 sensor files (316MB)	221 linked files (651MB)	1h 50min

Table 1. Creation of the Experiment database

Movie Database. The analysis of the 29 films generated 268 semantic files, however, processor 4 merges all XML documents generated for the same film. Unlike sensor files from experiments, movie analysis files are small (less than 1MB) and can be merged. The generated movie file must be edited to add the movie id, the movie title and language before files are inserted into the eXist XML database. Table 2 presents the timing for each processor. As both processes run in parallel, the movie database is created before the sensor database.

	Input	Output	Time
P1 P3	268 text files (2.3MB) 268 raw XML files	268 raw XML files 326 enriched files	 8 min
P4 (incl. manual editing)	326 enriched files (44.2MB)	29 movie files (24.7MB)	36 min (+ 30 min)
Total	268 text files (2.3MB)	29 movie files (24.7MB)	44 min

Table 2. Creation of the Movie database

4 Managing a Large Sensor Network

As many sensor networks comprise very high volumes of sensors and data, a centralised approach to data management would gradually reduce in performance. In prior work [1], we developed a distributed information system architecture that operated over a Peer-to-Peer (P2P) network. The XPeer architecture was effectively a logical IS architecture that provides the scalable benefits of P2P systems with the information management functionality of traditional database systems. In this section, we describe how XSENSE was extended with P2P concepts to facilitate scalable data management for data generated for a larger sensor network.

4.1 XSENSE-Peer Architecture

Using XSENSE-Peer, a sensor is modelled as a *peer*. Peers sharing common information are grouped into a cluster and managed by a super-peer. In the CDVPlex context, the definition of a cluster is flexible and might contain all heart-monitor data, or all data for a specified experiment, or all experiments that contained no gender mix among the users, for example. In §5, we will describe the results from using a number of clusters that were formed in this way. Clusters of peers are referenced by domains. Queries are sent to chosen domains and then routed using super-peers to the appropriate clusters. In this example, we have a single domain, CDVPlex, and all clusters belong to this domain.

System overview In XSENSE-Peer, each peer makes its sensor data available in the form of a service. A special client peer called a `Query Peer` reads XPath queries, sends them to `Cluster Peers` and retrieves responses. A `Repository Peer` receives metadata queries from both `Query Peers` or `Cluster Peers`. A `Cluster Peer` receives XPath queries from `Query Peers` and passes the query to all `Data Peers` in its cluster. It aggregates responses and returns the result back to the `Query Peer`. Figure 4 provides an overview of the query process.

1. The Query Peer passes keywords to the Repository Peer requesting ids of suitable Cluster Peers.
2. The query is sent to the ClusterPeers.

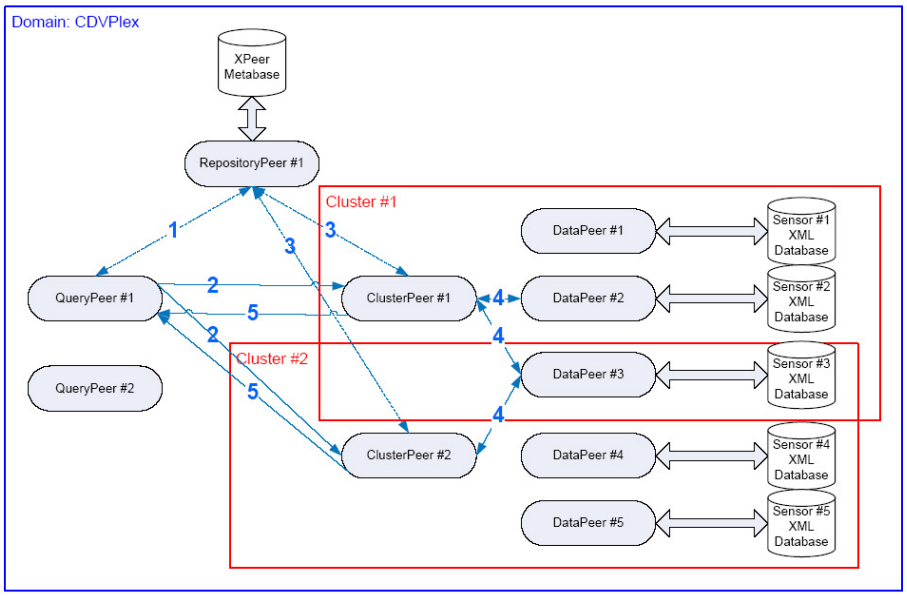

Fig. 4. P2P Query Processing

3. Cluster Peers request the ids of the Data Peers within their respective cluster from the Repository Peer. This step can be eliminated by loading this information at startup but we chose to leave it in place for our experiments. This was to enable a system where sensors can come and go in a random fashion.
4. Data Peers send results to their Cluster Peers
5. Cluster Peers send the final result to the QueryPeer.

5 Experiments

In this section, we report on a series of queries and their performance over a P2P network of sensor documents. Experiments were run on 3.2GHz Pentium IV machines, each with 1GB of RAM using the Windows XP Pro operating system and the P2P Query Processor was implemented using Java Virtual Machine (JVM) version 1.5. Each peer required a separate JVM (even on the same machine) to emulate a properly distributed environment and experiments were run three times with their times averaged to produce the final query response time.

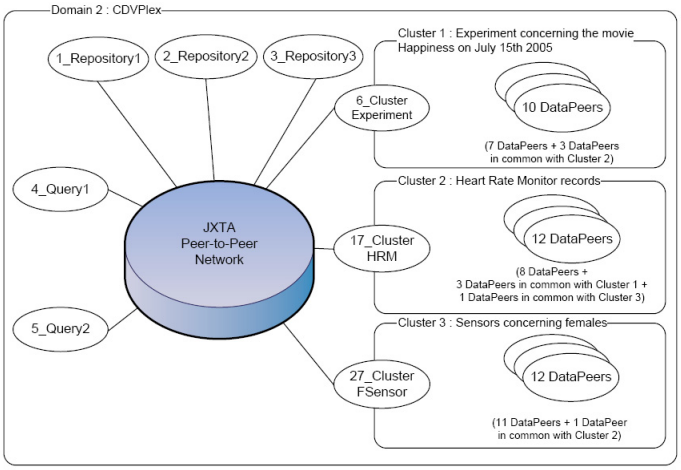

Fig. 5. JXTA Peers Network

5.1 Implementation with JXTA

JXTA provides the platform to implement the logical XSENSE-Peer layer. It defines an API of P2P protocols based on exchanges of XML messages to manage the basic services of a P2P application. The 6 protocols are:

- Peer Discovery Protocol: a searching mechanism with a local cache.
- Peer Resolver Protocol: to process request/response over the entire network.
- Peer Information Protocol: to provide peer status.
- Peer Membership Protocol: to manage the authentication of a peer.
- Pipe Binding Protocol: to manages pipe creation and connection.
- Endpoint Routing Protocol: to define the physical routes between peers.

The main components of JXTA are:

- Peer: a single JXTA instance.
- PeerGroup: a logical clustering of peers.
- Service: a function running inside a PeerGroup on some or all of its peers.
- Pipe: a virtual communication channel for document and data exchange.
- Advertisement: an XML document that describes any resource in a P2P network (peers, peergroups, pipes, services, etc).
- Rendezvous Peer: a special peer to caches advertisements and make them available within a PeerGroup.

With these features of JXTA, it was relatively easy to map the logical XSENSE-Peer structure to the implementation model required to run a series of experiments.

#	XPath Queries	C	P	Time
Q1	//base-uri()	3	34	8.672s
Q2	//experiment/title	1	10	2.856s
Q3	//experiment/@movieId/string()	1	10	4.131s
Q4	//experiment//timestamp	1	10	2.329s
Q5	//experiment[@movieId='tt0147612']/title	3	34	6.699s
Q6	//experiment[@movieId='tt0147612']/title	1	10	3.279s
Q7	//experiment[title='Happiness'] //timestamp[@name='ExperimentBegin']//text()	3	34	9.493s
Q8	//experiment[title='Happiness'] //timestamp[@name='ExperimentBegin']//text()	1	10	3.362s
Q9	//document-uri(sensorData[descendant::interval /startDateTime<'2005-08-01T00:00:00'])	3	34	9.652s
Q10	//sensorData[@type='HRSensor' and @personCode='14']//intervals	3	34	8.179s
Q11	//sensorData[@type='HRSensor' and @personCode='14']//intervals	3	20	5.953s
Q12	//sensorData[@personCode='28']/@type/string()	1	12	7.913s
Q13	//sensorData[@type='HRSensor']/@deviceId/string()	3	34	7.621s
Q14	//sensorData[@type='HRSensor']/@deviceId/string()	1	12	4.087s

Table 3. P2P Sensor Network Response Times

Peer Network The metabase contains a description of peers, their clusters and domains and their keywords. In this experiment, the keyword `cdvplex` is common to all peers and for example, all Data Peers linked to heart-rate monitors files have got the keyword `HRSensor`. The network is comprised of 250 peers (221 sensor files and 29 movie files) including the 3 clusters using the 30 DataPeers shown in figure 5. Clusters are not disjoint: there are 3 heart-rate monitor files in the Happiness experiments which also belong to the hear-rate monitor cluster.

- The first cluster contains Data Peers with sensor documents for the experiment using the film "Happiness". Peers in this cluster share the keyword "experiment".
- The second cluster concerns DataPeers with heart-rate monitor documents. Peers in this cluster share the keyword "HRM".
- The third cluster groups DataPeers linked to sensor files concerning female viewers. Those peers contain the keyword "female".

Comment on Results Table 3 illustrates the performance for a set of 14 queries run over the P2P network. Columns C and P represent the number of Clusters and Data Peers respectively. Unlike more low-level approaches to P2P searching, we use a metabase that contains information and location of domains and clusters. Thus, we route our queries to specified clusters of sensors. In general, we discovered that a single peer responded with sensor data inside 200ms to 300ms. When we scaled to larger numbers of peers, the times rose

linearly. Table 3 demonstrates that locating the correct peers does not cause a high overhead on the system unlike conventional large scale architectures, where it is not gneerally guaranteed that all sensors can be reached. Furthermore, traditional sensor networks (as will be shown later in our discussion on related research) tend to employ low-level query languages as these networks contain a semantic deficit between query language and representation of data. This is not the case with our network as our broad range of queries demonstrate.

Our experiments indicate that if cluster sizes are kept reasonably small there is no need to integrate the sensor data sources. However, if clusters are very large, then a single peer should be be used to integrate and cache sensor data. This will consifderably reduce the query response times.

6 Related Research

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

While sensor networks are now quite popular, there is not a great deal of published work in the area of semantic enrichment for sensor networks, nor in the area of evaluating performance over enriched sensor sources. Instead, most of the published work covers the use of XML for reasons of interoperability [8] or evaluates query processing using simple methods, operating at the network level [4].

In [8], they provide a template for incorporating non-XML sources into an XML environment. They tackle the same issue as that faced by knowledge workers in sensor networks: converting data to a usable format. Their approach is similar to XSENSE in that they use their Data Format Description Language to generate the XML representation of data. This is similar to our usage of ServingXML to meet the same goals although their approach requires a lot more user input as descriptions can often be quite lengthy. 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. On the negative side, they provide only a template system that has not been applied to any domain (instead they provide some use-case descriptions) and no query reponse times are possible. In this respect, their query optimiser will face problems when converting between the view definition and the physical data.

In [12] and [4], they process and query the raw streams and avoid conversion to XML. This has its benefits as the construction times for XML repositories (both centralised and distributed) are often reported to be quite large, and we have reported similar issues here. In [12], their approach is to enrich raw data into semantic streams and process these streams as they are generated. Their usage of constraints on the data streams provides a useful query mechanism with possibilities for optimisation. However, this work is still theoretical and contains no evidence of experiments or query times. In [4], they employ the concept of

proximity queries where 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 it is difficult to see how this type of proximity network can be applied in general terms due to the complexity of the technologies involved.

In [3], they provide semantic clusters within their sensor network. This is a similar approach to our work, where we cluster related groups of sensors. They also adopt a semi-automated approach and are capable of generating metadata to describe sensors and thus, support query processing. However, their object-oriented approach is likely to lead to problems with interoperability and this could be exacerbated through the lack of common query language. While this can be addressed with a canonical layer (probably using XML) for interoperability, it is likely to have performance related issues.

7 Conclusions

As sensor networks become more pervasive, the volumes of data generated will pose challenges to managing the stream of data values in an efficient, scalable, and yet useful manner. This data will include data readings from sensing devices, such as the biometric sensing devices we have used in this paper. It will also include data values derived from analysis of raw sensed data, such as the highlights and event detection results from an audio-visual data stream which we have illustrated in this paper through our analysis of movie video. While many differing solutions exist for efficiently managing the raw data values, we are more concerned in the work reported here on allowing semantic enrichment of the raw data to take place and to facilitate subsequent high level queries. To address this we have presented and used the XSENSE architecture, extended with Peer-2-Peer concepts and implementation, to realise scalable management of sensor data at a semantic level. We have demonstrated this in operation using a dataset of 33 experiments covering 29 movies and a total of 171 sensor outputs, realising a data volume of 316 MB of raw data. Our implementation has been tested using a collection of different *semantic* query types and response time performance, on a standard desktop machine, has been good.

There are many directions in which we intend to pursue further work. As clusters of sensors inside specified domains have similar properties to tree-based systems, we intend to optimise distributed XSENSE queries by extending techniques previously developed for XML trees [7]. We also intend to examine issues of real-time ingestion of sensor data, and real-time querying. Finally, we will also address questions of how to handle instances of sensors dropping out of the sensing process, and then returning later, usually done in order to save power consumption on the device.

References

1. Bellahsène Z. and Roantree M. Querying Distributed Data in a Super-Peer based Architecture, in: Proc. 15th International Conference on Database and Expert Sys-

tem Applications, LNCS 3180, pp. 296-305, 2004.
2. Dunne, L.E., Brady, S., Smyth, B. and Diamond, D. Initial development and testing of a novel foam-based pressure sensor for wearable sensing. *Journal of NeuroEngineering and Rehabilitation*, 2:1, pp.4-, 2005.
3. Kawashima H., Hirota Y., Satake S., and Imai M. MeT: A Real World Oriented Metadata Management System for Semantic Sensor Networks. 3rd International Workshop on Data Management for Sensor Networks (DMSN), pp. 13-18, 2006.
4. Kotidis Y. Processing Proximity Queries in Sensor Networks. 3rd International Workshop on Data Management for Sensor Networks (DMSN), pp. 1-6, 2006.
5. Lehane B and O'Connor N, Movie Indexing via Event Detection, WIAMIS06 - 7th International Workshop on Image Analysis for Multimedia Interactive Services, Incheon, Korea, 19-21 April 2006.
6. Lehane, B., O'Connor, N.E., Smeaton, A.F. and Lee, H. A System For Event-Based Film Browsing. In Proceedings TIDSE 2006 – 3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment. Springer Lecture Notes in Computer Science (LNCS) Vol 4326, pp. 334-345, 2006.
7. O'Connor M., Bellahsene Z. and Roantree M. An Extended Preorder Index for Optimising XPath Expressions. *Proceedings of 3rd XML Database Symposium*, LNCS Vol. 3671, Springer, pp 114-128, 2005.
8. Rose K., Malaika S., and Schloss R. Virtual XML: A toolbox and use cases for the XML World View. *IBM Systems Journal* 45:2, pp. 411-424, 2006.
9. Rothwell, S., Lehane, B., Chan, C.H., Smeaton, A.F., OConnor, N.E., Jones, G.J.F. and Diamond, D. The CDVPlex Biometric Cinema: Sensing Physiological Responses to Emotional Stimuli in Film. in: Proc. Pervasive 2006 - the 4th International Conference on Pervasive Computing, 7-10 May 2006, Dublin, Ireland.
10. Saxon Project.
    at: URL saxon.sourceforge.net, 2007.
11. ServingXML Project.
    at: URL servingxml.sourceforge.net, 2007.
12. Whitehouse K., Zhao F., and Liu J. Semantic Streams: a Framework for Composable Semantic Interpretation of Sensor Data. *3rd European Workshop on Wireless Sensor Networks (EWSN)*, LNCS 3868, pp. 5-20, 2006.
13. Lehane B., and O'Connor N. Movie Indexing via Event Detection *7th International Workshop on Image Analysis for Multimedia Interactive Services, Incheon, Korea, 19-21 April, 2006*
14. Lehane B. and O'Connor N. Action Sequence Detection in Motion Pictures *The International Workshop on Multidisciplinary Image, Video, and Audio Retrieval and Mining 2004*
15. Bordwell B., and Thompson K., Film Art: An Introduction *McGraw-Hill 1997*
16. Dancyger K. The Technique of Film and Video Editing. History, Theory and Practice *Focal Press, 2002*
17. Manjunath B.S., Salember P., and Sikora T. Introduction to MPEG-7, Multimedia content description language *John Wiley and Sons ltd 2002*
18. Lehane B., and O'Connor N. and Murphy N. Dialogue Scene Detection in Movies *International Conference on Image and Video Retrieval (CIVR), pp 286-296, Singapore, 20-22 July 2005*

Appendix A. Semantic File for Film Happiness.xml

```
<movie movieId="tt0147612" langCode="eng">
    <title>Happiness</title>
    <movieStructure>
        <scene id="0">
            <cluster id="0">
                <shot id="0">
                    <keyFrame id="0"/>
                </shot>
                ...
            </cluster>
            ...
        </scene>
        ...
    </movieStructure>
    <events>
        <event id="0" type="Dialogue">
            <startShot id="0"/>
            <endShot id="3"/>
            <keyShot id="2"/>
            <keyShot id="1"/>
            <keyShot id="99999999"/>
        </event>
        ...
        <event id="18" type="Action">
            <startShot id="176"/>
            <endShot id="183"/>
            <keyShot id="177"/>
            <keyShot id="179"/>
            <keyShot id="181"/>
        </event>
        ...
    </events>
    <shots>
        <shot id="0">
            <startFrame id="0"/>
            <keyFrame id="0"/>
            <endFrame id="30"/>
            <startTime>0</startTime>
            <endTime>1001</endTime>
            <motionIntensity>0</motionIntensity>
            <percentCameraMovement>0.2</percentCameraMovement>
            <percentSilence>0</percentSilence>
            <percentSilenceMusic>1</percentSilenceMusic>
            <percentSpeech>0</percentSpeech>
            <percentMusic>0</percentMusic>
            <percentOtherAudio>0</percentOtherAudio>
        </shot>
    </shots> </movie>
```