A Mediation Framework for Multimedia delivery

Onyeka Ezenwoye, Raimund K. Ege, Li Yang, Qasem Kharma
School of Computer Science
Florida International University
Miami, FL 33199
Telephone: +01 – (305) – 348 -1038
{oezen001, ege, lyang03, qkhar002}@cs.fiu.edu

ABSTRACT

We present a conceptual mediation framework that features three layers of mediators: *presence*, *integration*, and *homogenization* layers that work together in a peer-to-peer (p2p) manner to facilitate the delivery of multimedia data. On arrival of each request for data from a client[1], a *global-mediator* is elected from a group of *integration* layer mediators to service that request. Using distributed hash table (DHT), the *global-mediator* dispatches the request to other integrator mediators to track down the data sources. Upon receipt of the results, from the source(s), the *global-mediator* presents the data to the client via a *presence-mediator*. The *presence-mediator* may need to reformat the data to suit the execution context of the client. This mediation process is context-aware, adaptive and dynamically structured. Quality of service (QoS) factors are taken into consideration in the retrieval and presentation of data.

Keywords

Mediator, middleware, heterogeneous data sources, multimedia delivery.

1. INTRODUCTION

The proliferation of the internet has enabled access, at least on a physical level, to a multitude of disparate but often related information, while scaling geographical barriers. This information, in the form of multimedia data is stored on and access from various kinds of heterogeneous devices, recently more of which are mobile. Multimedia data requires special attention to throughput, timeliness and other quality of service factors. There is a need for architectures to deal with buffering and the intermittent connection associated with mobility. Our approach to enabling high quality access is to build a layered framework of mediators [18]. Lower-layer mediators connect to the actual data sources, while higher-layer mediators provide a logical schema of information to applications.

Mediators are typically employed in a situation where the client data model does not coincide with the data model of the potential data sources. They are facilitators that search for likely resources and ways to access them [17]. They provide a mapping of complex models to enable interoperability between client and source(s). Although many mediator systems have been proposed for a variety of applications, a major problem often encountered is how to seamlessly query and integrate data from heterogeneous data sources. Hence there is a need to formulate a mediator language that provides support for complex and semi-structured data types; a language that allows communication of knowledge between the mediator and source as well as the mediator and the client [3].

To overcome the problems posed by heterogeneity of data sources, the language of choice for our system is XML. XML is clearly today's standard of choice for the representation and exchange of structured data, particularly where that data must be read and interpreted by different applications running of different kinds of devices. XML and XML Schema provide a convenient, potentially human readable, easily extensible representation standard. Therefore, all data exchanged between mediators would be as XML.

In this paper we describe a three-layer architecture for multimedia mediation. The paper is organized as follows: Section 2 presents some related work and briefly covers some differences and similarities between our architecture and existing ones. Section 3 describes each layer and what functions are performed therein. We also discuss the different classes of mediators in those layers. Section 4 covers the election of global mediators to handle specific queries and a brief overview of the proposed election algorithm. Section 5 explains the various classifications of Distributed Hash Table algorithms and their use in looking up peers in p2p networks.

2. RELATED WORK

A lot of work has been done on mediation systems [4, 16, 19, 12, 9, 8, 7]. As stated in [9, 7], most of these architectures however are centralized, in that, there is a single mediator through which query decomposition, result integration and access to heterogeneous sources is achieved. Like our architecture, some [9, 16, 19] mediator architectures are distributed and mediators are able to access and communicate with each other. [19] is a two-tier mediation model that comprises a *homogenization* and *integration* layer with mediators in each that playing similar roles as in our architecture. [9] on the other hand does not have any restrictions on mediator functions as each mediator can play the role of homogenization and/or integration. There is also no

MUM 2004, October, 27-29, 2004 College Park, Maryland, USA.

[1] The use of the word client does not necessarily mean desktop PC. It could be any device with a digital heartbeat, mobile or immobility.

restriction as to the number of mediator tiers. [9] and [19] employ a similar integration process for homogenized sources [9]. Our architecture is a three layer model that consists of the *presence*, *integration* and *homogenization* layers. Our architecture does not only accommodate heterogeneous data sources but also with the aid of the *presence* layer mediators adapts to the heterogeneous nature of the client devices by taking into account various QoS issues of the client. [9] is a peer mediation system much like ours but unlike our model, it does not employ the use of the DHT in the distribution of source schema and peer lookup.

3. THREE LAYER ARCHITECTURE

The proposed three-layer mediation architecture is to handle requests (query or update) from a client which can be any special device or mobile computing unit.

The framework features three layers; *Presence*, *Integration* and *Homogenization*. A different class of mediator will be implemented within each layer (see Figure 1). A device may have all the three classes of mediators running on it at the same time. The mediators will transfer and negotiate on three kinds of information; the schema of the data stream, the type of operation required (e.g. query or update) and some quality of service (QoS) information specific to the client. The reason for exchanging QoS information is so that data streams can be tailored at the appropriate layers to suit the execution context of the client device.

3.1 Presence Layer

The primary functions performed in this layer are:

1. Attach required QoS parameters to queries.

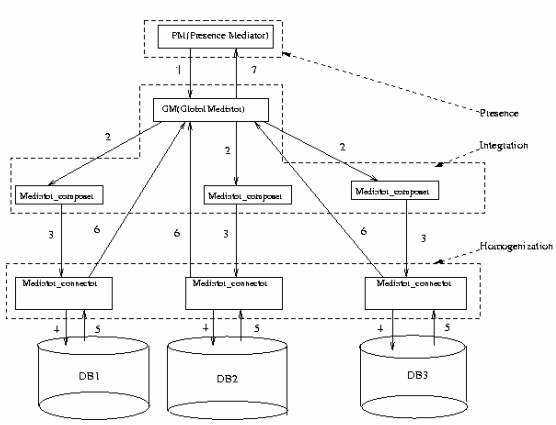
2. Election of global mediators to handle the requests.

3. Continuously advertise changes in QoS parameters to global mediators.

At the inception of a user request, the system would create a *presence-mediator* to handle that request. So there is one *presence-mediator* for each request, a *presence-mediator* cannot handle more than one request and the lifespan of an instance of a *presence-mediator* is dictated by the duration for which the request is valid. Upon receipt of a request, the *presence-mediator* would conduct an election to elect a *global-mediator* to serve that request.

A *presence-mediator* serves as a go-between for the client device and the *global-mediator* for that request, continuously monitoring the status of the device and for changes in its QoS parameters. The request's QoS specification is a translation of the perceived execution context on the client application.

QoS management is essential to efficiently access pertinent information at the required level of quality. This function attempts to meet the level of quality required by user.



1: query and/or client QoS information

2,3,4: query

5,6,7: query result

Figure 1. Three-Layer Architecture.

The continuous nature of the QoS management is especially important in the event that the client device is mobile. Resources are scarce on mobile devices and the availability of a resource may vary significantly and unpredictably during the runtime of an application. In the absence of resource guarantees applications need to adapt themselves to the prevailing operating conditions.

Presence mediators also have the job of converting the results of the request from XML to a format that is required for the particular client device.

In an ambulatory environment, for instance, a doctor might need to get critical information about a patient. The doctor, armed with a PDA with a wireless link, submits a query (Figure 2) to retrieve the patient's medical records.

```
<xs:schema  xmlns:xs =
"http://www.w3.org/2001/XMLSchema">
  <xs:element name = "query">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="patient_record" >
             <xs:element name="patient_id" >
             <xs:element name="name" >
             <xs:element name="date_of_birth">
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Figure 2. An example schema of a request for patient records.

Upon receipt of the query, the *presence-mediator* modifies the query by attaching the PDA's QoS parameters as illustrated in Figure 3.

```
<xs:schema  xmlns:xs =
"http://www.w3.org/2001/XMLSchema">
  <xs:element name = "query">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="patient_record" >
             <xs:element name="patient_id" >
             <xs:element name="name" >
             <xs:element name="date_of_birth">
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name = "qos">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="resolution" >
            <xs:element name="color_depth" >
            <xs:element name="bandwidth" >
             <xs:element name="power" >
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Figure 3. Modified schema of the request for patient records with QoS criteria.

3.2 Integration Layer

The mediators that comprise this layer are known as *mediator-composers*. These are the basic building blocks of the system. For a device to be considered a peer, it must implement a *mediator-composer*. It is this *mediator-composer* that upon receipt of a user request creates a *presence-mediator* for that particular request. Thus, the *presence* layer described in 3.1 only exists if there is at least one request being handled.

This layer is vital for the ability of the system to process queries. Because query processors may need to reformulate an initial query to enhance the chance of obtaining relevant data [1], this layer of mediators may need to translate the XML schema of the query into the schemas supported by other mediators. Because the mediators are p2p, each mediator will have specific knowledge about the supported data and schemas of its "neighbor" mediators. In the event that the *global-mediator*[2] (or other *mediator-composers*) has no knowledge about others' schema, the original schema is forwarded to its known peers (line 2 in figure 1) unaltered.

In other words, *mediator-composers* have the ability to re-construct XML schemas for requests. When a *mediator-composer* receives a request, it may need to simplify the request before forwarding it. If the *mediator-composer* has some knowledge about the request, it simplifies the request according to its knowledge. The *global-mediator* is informed whenever the schema of its request is altered. The *global-mediator* keeps track of where what is found with the use of a "mediated schema". This mediated schema will also be used to reassemble the results of the query that were obtained from different sources and further query.

The *integration* layer basically reformulates the client request into a set of queries over the data sources using the appropriate schemas. *Mediator-composers* set up contracts between multiple data sources in order to satisfy requests. They are in charge of finding systems that meet the specified QoS criteria.

3.3 Homogenization Layer

In our architecture, each *mediator-connector* (*homogenization* layer mediator) will be directly associated with a physical source. Note that (as stated in section 3) that a device can have all three types of mediators on it. A *mediator-connector* is implemented only if it is associated with a persistent data source.

Mediator-connectors do not change the XML for the request; they retrieve data from data sources and converts query result into a stream of XML data which is submitted to the *global-mediator* for the request (line 6 in figure 1).

Data from relational databases can be mapped to XML by table-based mapping. The advantage of this mapping is its simplicity: because it matches the structure of tables and result sets in a relational database. This type of mapping however has several disadvantages; primarily, it only works with a very small subset of XML documents. It also does not preserve physical structure (e.g. character and entity references, CDATA sections, character encodings, and standalone declaration) or document information (e.g. document type or DTD), comments, or processing instructions.

Because table-based mappings only works with a limited subset of XML documents, some middleware tools, most XML-enabled relational databases, and most XML-enabled object servers use a more sophisticated mapping technique called object-relational mapping. This models the XML document as a tree of objects that are specific to the data in the document; it then maps these objects to the database.

Most XML schema languages can be mapped to databases with an object-relational mapping. The exact mappings depend on the language.

4. GLOBAL MEDIATOR ELECTION

Once the *presence-mediator* (in the *presence* layer) receives a request for which it was instantiated, it creates an XML schema for this request coupled with some QoS criteria specific to its client device and that's best suited for that type of request. The *presence-mediator* then conducts an election to select a *mediator-composer* (in the *integration* layer) to be the *global-mediator* for that request. The election is conducted in order to find the best possible *mediator-composer* to serve the request. Selection criteria include but are not limited to available bandwidth, network traffic and load. The elected mediator will be the one that best meets the required QoS criteria and is able to carry out the search, cache, integrate and return the result. In explaining the mediator election, it is important to note that:

A peer knows at least one other peer otherwise the system isn't p2p.

1. A *mediator-composer* can be elected to serve as *global-mediator* for more that one request at the same time.

2. Each request is serviced by only one *global-mediator*.

3. In the event that a *global-mediator* fails, another one is elected.

[2] When a mediator-composer is elected to serve a request, it becomes the global-mediator for that request

As of this writing, our election algorithm of choice is the ring algorithm. Our ring algorithm is based on the ring algorithm [15] with some modifications.

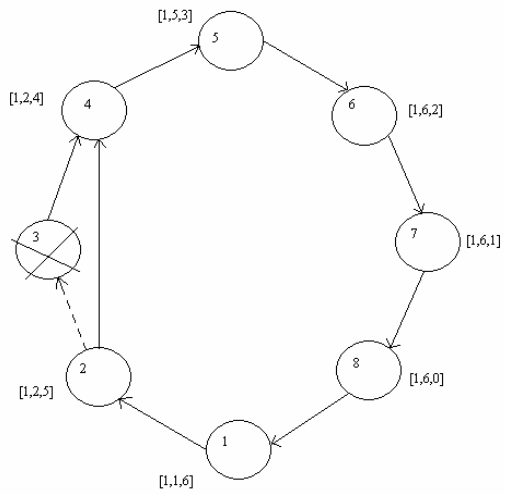


Figure 3: The Ring Election Algorithm.

For the purpose of clarity, we will use Figure 3 above to illustrate the algorithm. It is also important to note that for the purpose of brevity, the following description has been simplified. The election message is a 3-tuple [i, j, k], where i = election initiator; j = best candidate so far; k = hop count. The hop count is used to make sure that the election message does not travel perpetually because a p2p network may not actually be physical ring.

When there is a need to elect a *global-mediator*, the peer that is initiating the election (node 1) sends an election message to its successor. In Figure 3, node 1 sets i and j to 1, thereby making itself a candidate for the election. k = 6 for the purpose of illustration. By setting j to 1, we mean that node 1 attaches to the election message its status information. This could be information such as load and bandwidth. It is possible for the initiating node to be elected because (as stated in section 3.2), a *mediator-composer* must reside on the physical device for that device to be considered a peer. It is this *mediator-composer* that creates the *presence-mediators* (section 3.1) to handle requests. A *presence-mediator* - in its quest to find the best suitable peer (*mediator-composer*) to act as *global-mediator* for the query - may end up electing the same *mediator-composer* that created it. Thus electing its own peer.

Upon receipt of the election message, the receiving node compares the status information contained in the election message with its own status information. If it determines that its status is superior to that which is contained, it replaces the status information with its own (e.g. node 2 in figure 3). It checks to see that the k > 0 then forwards the message to its successor after reducing the hop count by 1. If its successor is down (e.g. node 3 in Figure 3), the message is sent to the next successor. The election only terminates in these three cases:

1. If a node's only successor is down, the message is sent to i and j is elected.

2. If k = 0, the election message is sent to i and j is elected.

3. If a node's successor is i (a ring), j is elected.

In our example in Figure 3, node 6 is elected as *global-mediator*.

5. PEER LOOKUP

After electing the *global-mediator*, the *global-mediator* will coordinate with other *mediator-composer(s)* and/or *mediator-connector(s)*, in order to serve the request. Interaction between mediators is P2P in which peers share distributed files. The most recent lookup algorithms for P2P system are based on distributed hash table (DHT). In general, these algorithms routing time complexity is O(log N) where N is the number of nodes (peers) in the system. [2] classifies DHT algorithms into three categories:

1. Skiplist-like routing algorithm:

Chord algorithm [14] is an example of skiplist-like routing algorithm. In Chord, every node in the system maintains information about O(log N). The hash function assigns an m-bit identification key using SHA-1 as a base function to map the IP address. The nodes in the system are arranged in an identifier circle. Each node on this circle maintains a finger table containing the IP addresses of n+2i-1 successors where n is the node ID and 1<= i <= m. In other words, this finger table maintains the IP addresses of halfway, quarter-of-the-way, eighth-of-the-way, and so forth. As a result this algorithm can find the required node in O(log N) time.

2. Tree-like algorithms:

Tree-like algorithms, such as Pastry [13], Tapestry [6], and Kademlia [10], use structured prefix to maintain the location of nodes. Each node maintains IP addresses of some other nodes in its leaf.

3. Routing in Multiple dimensions:

CAN [11] is an example of routing in multiple dimensions. Each node in CAN maintains chunk of DHT called zone. These zones are distributed in d-dimension. In addition to storing a chunk of DHT in the zone, each zone maintains information about its neighbors in the d-dimension. The routing time complexity for this algorithm is O(d N1/d).

The reader can observe that these algorithms are similar in the following aspect:

1. With the use of DHT, each node maintains information about its neighbors only, not all the nodes in the system

2. Their time complicity for routing is O(log N) for most of them.

These algorithms however differ in many aspects, but the most important difference is how each algorithm determines "neighbor". In general, each node should maintain minimum knowledge about other nodes in the systems. A hash function, i.e. SHA-1, maps keys onto values where values could be file names, IP addresses, or any naming to be looked up. In our case, we are interested in mapping XML schema tags and we will use Cord algorithm [14] because of its performance [5] in comparison to other algorithms.

In order for a new node to join the system and become a peer, it will send a "join" message [14]. After locating the position of this node, the new node will join the P2P system as a composer.

This new peer may have an associated *mediator-connector* if there is an associated database.

Despite of the role of *global-mediator* in the presentation, all *mediator-composers* need to cooperate in order to find the connector(s) (*mediator-connectors*) to the desired data source(s). To find the connectors(s), the route from the *global-mediator* through composers can be found using DHT instead of having a central repository of the connectors' XML schemas. All messages between mediators are in XML format and each composer maintains some XML schema which will be used to decompose/compose the XML request in order to match a XML schema that is stored in a connector. The hash function maps the XML tags or elements onto keys which will be distributed over the peers. Assume the following is a valid XML schema:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/XMLSchema">
  <xs:element name="PatientXrays">
    <xs:complexType>
     <xs:attribute name = "ssn" type="xs:string">
     <xs:attribute name="fullname" type="xs:string">
     <xs:attribute name="xray" type="xs:image">
   </xs:complexType>
  </xs:element>
</xs:schema>
```

The hash function maps ssn, fullname[3], and xrays onto keys which will be distributed over the peers (*mediator-composers*). *Mediator-composers* will generate the XML tree for the XML schemas which have been sent from *mediator-connectors* to be mapped. *Mediator-composers* hash the nodes, which correspond to elements in the XML schema in the corresponding tree and distribute the generated key with the element to a node in the system which maintains the range of that key.

The *mediator-composers* decompose the incoming request or simplify the incoming request by adding subtree(s) to the original request until all the tree leaves represent connectors. In decomposing a query, if the *global-mediator* which has been elected to handle this request cannot solve the FullName for instance, it will forward the request to one of its neighbors. Eventually, one of the composers will decompose the FullName into FirstName and LastName. After that, all the leaves in the tree can be directed to the corresponding connector using the DHT.

6. CONCLUSION AND FUTURE WORK

The interchange of data between client and heterogeneous sources requires an efficient and dynamic approach to mediation. The framework described in this paper features three layers of mediators: *presence*, *integration*, and *homogenization*. On arrival of a request for data, a *mediator-composer* is elected as *global-mediator* that is responsible for data caching and service provision. The *global-mediator* dispatches the data stream request to other *mediator-composers* in order to track down the adequate sources. The results are then integrated and sent back to the user in a way that best suits the execution context of the user device.

The advantage of our mediation process is its adaptive and dynamic nature. The framework is designed to uniquely determine how to fulfill each query while taking properties of delivery into consideration. The presence-mediator takes into account the heterogeneous nature of client devices and is meant to tailor the query formulation and presentation of results to suit the execution context of the client. This is especially important for mobile devices give their limited resources. Our mediation architecture is a work-in-progress and there are many research issues that will encountered during the course of this project, they include but not limited to, defining of communication protocols with specific focus on QoS, how to deal with real-time data and mobility (e.g. temporary loss of connectivity in mobile devices, failure of the global-mediator), security issues involved with the distribution and access of data across a p2p network and how to intelligently decompose and integrate XML schemas while avoiding loss of information.

[3] Some elements in the client XML schema might need to be decomposed; i.e. the fullname could be decomposed into lastname and firstnamae, and that way multiple composers may cooperate to serve the client.

7. REFERENCES

[1] Arens, Y., Knoblock, C. and Shen, W. Query Reformation for Dynamic Information Integration. *Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies*, 1996;6(2-3):99-130.

[2] Balakrishnan, B., Frans Kaashoek, M., Karger, D., Morris, R. and Stoica, I. Looking up data in P2P systems, *Communications of the ACM,volume 46 , Issue 2*, February 2003.

[3] Buneman, P., Raschid, L. and Ullman, J. Mediator Languages – a Proposal for a Standard, *Report of DARPA I3/POB working group*, University of Maryland, 1996.

[4] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman,Y. D., Vassalos, V., and Widom, J. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117 - 132, 1997.

[5] Gummadi, K., Gummad, R., Gribbl, S., Ratnasam, S., Shenke, S., and Stoica, I. The Impact of DHT Routing Geometry on Resilience and Proximity. In *Proceedings of SIGCOMM'03*, August 25–29, 2003, Karlsruhe, Germany.

[6] Hildrum, K., Kubiatowicz, J., Rao, S., and Zhao, B. Distributed Object Location in a Dynamic Network. In *Proceedings of 14th ACM Symposium. on Parallel Algorithms and Architectures (SPAA)*, August 2002.

[7] Josifovski, V., and Risch, T. Comparison of Amos II with other Integration Projects, Technical Report, EDSLAB/IDA, Linköping University, April 1999

[8] Karjalainen, M. Integrating Heterogenous Databases with the Functional Data Model Approach, http://www.cs.chalmers.se/~merjaka/report04d.pdf, January, 2004

[9] Katchaounov, T. *Query Processing for Peer Mediator Databases*, Doctoral thesis, Uppsala University, 2003

[10] Maymounkov, P., and Mazieres, D. Kademlia: A peer-to-peer information system based on the XOR metric. In Proceedings of the 1st International Workshop on Peer-to-Peer Systems, Springer-Verlag version, Cambridge, MA, Mar. 2002.

[11] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, San Diego, CA, August 2001.

[12] Risch, T., Josifovski, V., and Katchaounov, T. AMOS II Concepts, http://www.dis.uu.se/~udbl/amos/doc/amos_concepts.html, June 23, 2000

[13] Rowstron, A., and Druschel, P. Pastry, Scalable, distributed object location and routing for large-scale peer-to-peer systems, In *Proceedings of the 18th IFIP/ACM Int'l Conf. on Distributed Systems Platforms*, Heidelberg, Germany, pages 329-350, Nov. 2001.

[14] Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM*, San Diego, August 2001.

[15] Tanenbaum, A. S., Van Steen, M. *Distributed Systems: Principles and Paradigms*. Pearson Education, September 2001, page 263.

[16] Tomasic, A., Raschid, L., and Valduriez, P. Scaling access to heterogeneous data sources with DISCO. *IEEE Transactions on Knowledge and Data Engineering*, *10*: 808 – 823, 1998.

[17] Wiederhold, G., and Genesereth, M. The Conceptual Basis for Mediation Services. *IEEE Expert*, Vol.12 No.5, Sep.-Oct. 1997, pages 38-47

[18] Wiederhold, G., Mediators in the Architecture of Future Information Systems, *IEEE Computer*, 25(3):38–49, Mar. 1992

[19] Yan L., Tamer Özsu, M., Liu, L., Accessing Heterogeneous Data Through Homogenization and Integration Mediators, In *Proceedings of the Second IFCIS International Conference on Cooperative Information Systems*, pages 130-139, June 24-27, 1997