Three-Layered Mediator Architecture Based on DHT [*]

Raimund K. Ege, Li Yang, Qasem Kharma, Xudong Ni
Secure Software Architecture Laboratory
School of Computer Science
Florida International University
Miami, FL 33199, USA
{ege|lyang03|qkhar002|xni001}@cs.fiu.edu

Abstract

The interchange of data between client and heterogeneous sources requires an efficient and dynamic approach to mediation. Our framework features three layers of mediators: presence, integration, and homogenization. On arrival of a request for data exchange, a session initiation server forks to the mediator group to elect one mediator as global presence mediator that is responsible for data caching and service provision. Using Distributed Hash Table (DHT), the presence mediator dispatches the data stream request to other mediators and tracks down to the source, which is then integrated and connected to the client in a user-relevant way. Our mediation process is adaptive and dynamic upon the user request and takes QoS factors into consideration.

Keywords: *mediator, layered mediator, middleware, software architecture, XML, integration*

1. Introduction

Modern information systems use inherently complex data. The data is multi media, i.e. it is delivered as a continuous stream from a multitude of sources. Multimedia data requires special attention to throughput, timeliness, and other quality of service factors. Our approach to enabling high quality access is to build a layered framework of mediators.The lower-layer mediators reside on top of actual data sources, and maps the data source schemas to XML schema. The middle layer resolves the schema differences between the user needs and the source availability by providing a logical schema of information to application. The upper layer makes the data source seem ever-present to the user and communicates directly with the user and makes the multimedia streams available in the context of a web service.

Mediators are typically employed in a situation where the client data model does not coincide with the data model of the potential data sources. The mediator provides a mapping of complex models to enable interoperability between client and source(s). Many mediator systems have been proposed for access to heterogeneous databases to serve a variety of client types. A standard mediator language [6] proposal requires support for complex types and semi-structured data; abstract types with methods; the exchange of rules that allow the communication of knowledge between the mediator and source as well as the mediator and the client; and the exchange of metadata.

Also relevant to our research are main-memory database systems and constraints. A main-memory database system presents an in-memory set of data to a client and hides the complexities of secondary storage access [9]. Constraints allow the specification of facts that have to be considered in the context of many others. Constraint satisfaction is the process of considering all constraints to arrive at a state where all constraints are satisfied. Constraints have been used to support an architecture for user interfaces [5, 8], but also in mediator system [1] where they provide a more flexible and dynamic type mapping between source and client data model.

The new contributions of this research are the follows: providing XML based homogeneous interface and data integration among heterogeneous data sources in mediator architecture, implementing SIP protocol to support user mobility in mediator architecture, and using DHT to provide distributed data lookup in mediator middleware.

The remainder of the paper is organized as follows: Section 2 discuses our three-layer architecture. Section 3 explains the mediator components. Finally, Section 4 describes the communication between mediators using SIP and DHT.

[*]This material is based upon work supported by the National Science Foundation under Grant No. HRD-0317692.

2. Three layer architecture

The proposed three-layer mediator architecture is to handle requests from a client which can be any computing unit. These mediators will play intermediate roles between the client and the data sources, and these mediators help the client to establish streams to and from data sources.

The framework features three layers: it differentiates among homogenization / connector, integration and presence mediators. The homogenization, also called connectors, mediators connect to actual data sources; integration mediators collect data from various sources into a coherent logical schema; the top-level presence mediators are meant to enable caching and buffering capabilities. The top layer contains mediators whose task is to make the data source seem ever-present to the client. These "presence" mediators are located using the Session Initiation Server (SIS) which is a Session Initiation Protocol [13] like server and subsequently retrieve and cache the data stream needed by the client. For every request, one of the presence mediator will be elected to be the Global Mediator which communicates directly with the client. The middle layer contains the "integration" mediators. Their task is to resolve the schema differences between the client needs and the source availability. The bottom layer contains the "homogenization" mediators - connectors. Their task is to make a specific feed source available to the middle layer of integration mediators. Basic mapping of data representation and bit level compatibilities are handled here.

2.1. Presence

Our mediators will transfer and negotiate on four kinds of information: the schema of data/information stream, the operation specification, the quality of service information specified from user API, and the context information detected by the system. The request could be classified as those four kinds of data, and user perceived quality has to be mapped onto QoS parameters that will be supported by the different layers.

Because our system handles multimedia objects, it requires the integration of various services for multimedia object creation, storage, access, transfer and presentation [4]. In our system, we allow user to set or modify his requirements in the request.

Quality of Service management is essential to efficiently access pertinent information at the required level of quality. This function intends to meet the level of quality required by user. In a distributed multimedia system, it is necessary to take both users requirements and Quality of Service (QoS) provided by the system into consideration. System-level QoS factors may include: delay needed to transfer objects, the quality of information provided, e.g. image resolution and colors, as well as financial costs attached to document delivery such as the costs charged by a library to obtain a copy of a journal article.

The context information detected by the system is important for QoS Management, since resources are scarce on mobile devices and the availability of resources may vary significantly and unpredictably during the runtime of an application. In the absence of resource guarantees, applications adapt themselves to the prevailing operating conditions. For example, if communication bandwidth is scarce, a doctor information application on a mobile computer or PDA, that receives data via some wireless link, might display text and low-resolution pictures, instead of video clips. The following is the example for XML schema of request from user.

```
 <xs:schema  xmlns:xs = "http://www.w3.org/2001/XMLSchema">
  <xs:element  name = "request">
   <xs:complexType>
    <xs:sequence>
     <xs:element  name = "patient_id"/>
     <xs:element  name = "name"/>
     <xs:element  name = "age"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 <xs:element  name = "Action"/>
 <xs:element name = "QoS">
  <xs:complexType>
    <xs:sequence>
     <xs:element  name = "color"/>
     <xs:element  name = "delay"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element  name = "system"/>
 <xs:element  name = "bandwidth"/>
</xs:element>
```

This example shows the XML schema that governs a typical request for patient data that might be issued from a handheld device as a doctor attends to a patient in an ambulatory environment. The requests QoS specification must be translated to an application QoS specification. For instance, the "QoS" and "bandwidth" fields are user requirements which will be mapped into parameters that can be interpreted by the system.

2.2. Integration

The global mediator receives the client request based on an XML schema and translates it into schemas supported by the underlying layers. The data integration system reformulates the client request into a set of query over the data sources and then executes them. Then, each source needs to be mapped to relevant parts of this unified schema. The single schema of the integrated system is called the "mediated schema". Having a mediated schema facilitates the formulation of queries to the integrated system. The users simply pose queries in terms of the global mediated schema, rather than directly in terms of the source schema. The system then intelligently processes the query, reading data across the network and responding to data source sizes, network conditions, and other factors. The functionality is similar with element patterns on matching data among the different sources.

The Global Mediator dispatches the user query to other mediators which coordinate with the global mediator in order to serve the request. These mediators are also called mediator-composers: they set up a contract between the performances of multiple data sources in order to satisfy the users QoS requirements; they are in charge of finding a system configuration that supports the requested QoS and can be considered as similar to resource allocation in distributed systems. QoS negotiation leads to a commitment from the overall components concerning the quality level that will be offered. Different types of commitment can be provided: guaranteed, best-effort or stochastic. In the first case, the different components must reserve the corresponding resources. The communication between mediators is 2-way and request/response based. The request is short, and the response is in terms of delivering a stream of data in XML form. Every mediator is currently being implemented as a Java Servlet that runs on an application server (e.g. Tomcat). It provides the following methods (from the HttpServlet class documentation):
- *doGet*: to request data, i.e. an instance of an XML schema, from the mediator.
- *doPut*: to send data, i.e. an instance of an XML schema, to the mediator.
- *init* and *destroy*: to manage resources that are held for the life of the servlet.
- *getServletInfo*: to provide information about a servlet and to return its XML schema.

The communication among mediators is based on http methods: *get* and *put* to accomplish the two-way communication, and query to query the XML schema from Schema DB. Each mediator can run servlets independently, without affecting the other's function. As an example, consider Figure 1: when a client intends to query data from DB_i, the get method is employed to get data from the Global Mediator. After the translation of the query information, the Global Mediator dispatches the queries to lower level mediators and obtain data form base Database by get method. Then, the data is integrated and sent to the Global Mediator. What returned to the client is an XML stream of data that contains any other kind of data including multimedia. On the other hand, when a client intends to update the data in DB_i, the client uses the put method to inform Global Mediator of update action. The Global Mediator sends the update command to lower level mediators which use put methods to update the data information in the databases. Also, Global Mediator can query the current schema form the Schema DB via the getServletInfo method.

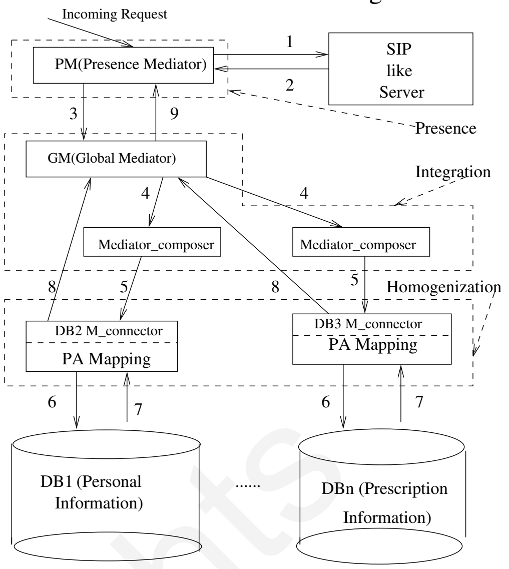

Figure 1: Three-Layer Architecture

2.3. Homogenization

The homogenization layer, which is represented by mediator-connectors, converts physical data, such as data from SQL queries, ftp gets, http gets, into a stream of XML data. XML is clearly todays standard of choice for the representation and exchange of structured data, particularly where that data must be read and interpreted by different applications written by different groups. XML and XML Schema provide a convenient, potentially human readable, easily extensible representation standard. Therefore, we opt to use XML in communication among the mediators. Data from relational databases can be mapped to XML by table-based mapping or an object-relational (object-based) mapping. There is an obvious table-based mapping between the XML document and relational data. The advantage of this mapping is its simplicity, and it is easy to write code based on this mapping; code which is fast, scales well, and is quite useful for certain applications, such as transferring data between databases one table at a time. The mapping has several disadvantages; primarily, it only works with a very small subset of XML documents. In addition, it does not preserve physical structure (such as character and entity references, CDATA sections, character encodings, or the standalone declaration) or document information (such as the document type or DTD), comments, or processing instructions. Because table-based mappings only work 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, called an object-relational mapping. This models the XML document as a tree of objects that are specific to the data in the document, then maps these objects to the database. The XML mapping and conversion tool, which lets the user to interactively create a data transformation, will allow user to map and integrate some types of data to XML data in a user-friendly command-line interface, used in XMLconversion product. Such XML conversion gives users a simple, consistent approach to creating XML from other types of data or updating other types of data from XML input. Users can use it across multiple databases. XML schema is an emerging standard from W3C. XML schema is a language for defining the structure of XML document instances that belong to a specific document type. XML schema can be seen as replacing the XML DTD syntax. XML schema provides strong data typing, modularization and reuse mechanisms not available in XML DTDs. Most XML schema languages can be mapped to databases with an object-relational mapping. The exact mappings depend on the language. DDML, DCD, and XML Data Reduced schemas can be mapped in a manner almost identical to DTDs. The mappings for W3C Schemas, Relax, TREX, and SOX appear to be somewhat more complex. In the case of W3C Schemas, a complete mapping to object schemas and then to database schemas is available.

3. Mediator Model

Our architecture differentiates between two kinds of mediators: mediator-composer and mediator-connector. The client first connects to a special kind of mediator-Composer called Global Mediator as described in the next section. The hierarchy of the mediators is dynamically built based on which mediator is elected to be the Global Mediator (Section 4.1) and how the Distributed Hash Table (DHT) is built (section 4.2). At the lowest level of the mediator hierarchy, mediator-connectors are located. Data sources can be accessed through mediator-connectors only.

3.1. Mediator-Composer

Interoperation with the diversity of available sources requires a variety of functions. The mediator group has to accommodate multiple types of modules, and allow them to coordinate as required. For instance, facilitators will search for likely resources and ways to access them [18]. Query processors will reformulate an initial query to enhance the chance of obtaining relevant data [2, 19]. Text associated with images can be processed to yield additional keys[10]. The QoS specification from client is taken into consideration; finally, further integration makes the results relevant to the client. The mediator-composers have the same interfaces and capabilities, so each one would deal not only with client users (translation functionality occurring in presence layer) but also with other mediators (integration functionality occurring in integration layer), providing or giving response. The elected Global mediator has the responsibility of caching data and providing service.

In other words, mediator-composers have the ability to construct XML schemas for requests. When mediator-composer receives a request, it will either simplify and forward the request or just forward the request. If the mediator-composer has some knowledge about the request, it simplifies the request according to its knowledge. For instance, if a mediator-composer receives a request to retrieve ”name” and this mediator knows that the ”name” is composed of ” first-name” and ”last-name”, it replaces the ”name” with ”first-name” and ”last-name”. Besides, it may add some QoS parameters. Then, it will forward the request to the next mediator. When it receive the response, it will integrate the ”first-name” and ”last-name” into ”name”

3.2. Mediator-Connector

Selection and filtering of data is a function which is best performed at the source since one does not want to ship large amounts of unneeded data to the client or the mediators [17]. Making data accessible may require a wrapper/connector at or near the sources, so the access can be performed using standard interfaces. In our architecture, each mediator-connector will be directly associated with a physical source, and is to provide transparent translations from different data formats to XML format, and to provide transparent communications protocol interoperability between components and persistent data storage.

There are two differences between mediator-connectors and mediator-composers. First, mediator- connectors do not change the XML for the request. Mediator-connectors analyze and provide the request data from the data source to the mediator-composer which request this data. The second difference is that mediator-connectors are responsible to retrieve data from data source, reformat the data to XML format, create a small chunk of the data to be submitted, and format the stream of data to be submitted.

3.3. Mediator Connections

Every mediator has a composer stream or connector stream. The stream is a Java interface that has at least a read or write method. Mediators can communicate via streams, one mediator writes to a stream, the other reads from the same stream. The stream of every mediator-composer represents the abstraction of mediator and the stream of every mediator-connector represents the abstraction of resources. The stream is semi-structured and can be instantiated into events or messages, a kind of XML object. And, the XML object can be passed among mediators by reading or writing stream instance. A mediator searches its DHT to determine the next destination of a request. Searching the DHT yields the IP address of the next mediator, so the mediator can create a port to communicate with the destination mediator using transport protocol such as UDP. Then, the senders writer submits stream of messages, and the destinations reader receive the stream messages. When a mediator-connector receives a request, it will send each data on separate port; for instance, if the request contains images and audio, the mediator-connector creates a port to submit the images and another one to submit the audio file.

4. Session Initiation

Our goal is to arrive at a simple and unifying mediator concept that captures all three layers. The relationship among mediators in the same layer is peer-to-peer, and once one mediator receives a request from a client, the mediator sends an INVITE message to a mediator through a session initiation server (SIS). After the SIS, which is a SIP-like server, provides an ID of a mediator to the client who sent a request, this mediator will become the global mediator for the incoming request and take the responsibility of a presence mediator (cache data, integrate data, return service).

4.1. Global Mediator Election

The global mediator is elected dynamically upon the request from the client. A user (client) sends a request to session initiation server via a INVITE command to know which Mediator-Composer can serve as its global mediator. SIS then broadcasts to the group of the available Mediator-Composers via a FORK command to all registered mediators. A register mediator is a mediator server that is available and has some knowledge about some data sources. In other words, when a mediator is started, it sends a registration message which includes the IP address, and name or ID to SIS. The first mediator that responds (RESPOND) to the forked message will be the global mediator which also plays the role of presence mediator for that request and to serve the incoming request. Any response to the forked message after RESPOND the IP to the client will be ignored. Therefore, the global mediator is elected and the architecture is dynamically formed. Although SIS seems to be a central failure point, it is not a big issue since any SIP server can play its role. Using a SIP-like architecture is to support user mobility and real-time multimedia streaming[7].

4.2. Lookup Algorithm

After electing the global mediator which is responsible for receiving and responding for the user request, the global mediator will coordinate with other mediator(s), either composer(s) and/or connector(s), in order to serve the request. This coordination is similar to peer-to-peer (P2P) system in which nodes share distributed files. The most difficult challenge in P2P is how data can be found in a large, scalable P2P system without having central failure server [3].

The most recent lookup algorithms for P2P system are based on distributed hash table (DHT). In general, these algorithms routing complexity is $O(logN)$ where N is the number of nodes in the system. [3] classifies DHT algorithms into three categories:

Skiplist-like routing algorithm: Chord algorithm [16] is an example of skiplist-like routing algorithm. In Chord, every node in the system maintains information about $O(logN)$. The hash function assigns 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 $n + 2^{i-1}$ successors where n is the node ID and $1 \leq i \leq m$. 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 $O(logN)$.

Tree-like algorithms: Tree-like algorithms, such as Pastry [15], Tapestry [11], and Kademlia [12], use structured prefix to maintain the location of nodes. Each node maintains IP addresses of some other nodes in its leaf.

Routing in Multiple dimensions: CAN[14] 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 $O(dN^{1/d})$.

The reader can observe that these algorithms are similar in the following aspects:
- Each node maintains information about its neighbors only, not all the nodes in the system.
- They are using a DHT instead of maintaining of central server.
- Their time complexity for routing is O(log N) for most of them. However, these algorithms differ in many aspects, but the most important different is how each algorithm defines "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 lookup. In our case, we are interested in mapping the XML schema tags and we will use Chord algorithm [16].

4.3. Putting it all together:

The scenario how the system will work is as following:

- Adding a new mediator:
  - send a REGISTER message to SIS to announce its availability
  - build its DHT, maintain some knowledge about its neighbors, and let its neighbors know about its services
- When a client wants a service
  - send a REGISTER message to SIS
  - send an INVITE message to SIS
  - SIS forks the INVITE message
  - SIS returns the IP address of the first mediator responding to the INVITE message
  - this mediator will be the global mediator and the client establish a direct connection with it
  - the client send the request to the global mediator
- When a global mediator receives a request
  - build the XML schema for the request
  - lookup the IP address(es) for coordinating mediator(s) which can handle the XML schema data or apart of it (tags). Coordinating mediators either is the neighbor of global mediator or contains the requested XML schema data.
  - establish a connection with the coordinator(s)
  - if the coordinator is a connector mediator, get data
  - if the coordinator is a composer mediator, the coordinator lookup data in similar manner (loop) with global mediator does
  - integrate collected data with every level
  - return the response.

5. Summary

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 exchange, a session initiation server forks to the mediator group to elect one mediator as global presence mediator that is responsible for data caching and service provision. With the help of the session initiation server, the presence mediator dispatches the data stream request to other mediators and tracks down to the source, which is then integrated and connected to the client in a user-relevant way. The advantage of our mediation process is its adaptive and dynamic nature, not only may the user request change rapidly, but also properties of delivery, such as QoS factors, are taken into consideration.

References

[1] C. Altenschmidt, J. Biskup, J. Freitag, and B. Sprick. Weakly constraining multimedia types based on a type embedding ordering. In *the 4th International Workshop on Multimedia Information Systems*, volume 1508, pages 121–129, Berlin, 1998. Springer-Verlag.

[2] Y. Arens, C. Knblock, and W.-M. Shen. Query reformation for dynamic information. 1985.

[3] H. Balakrishnan, M. F. Kaasoek, D. Karger, R. Morris, and I. Stoica. Looking up data in P2P systems. *Communication of the ACM*, 46(2), February 2003.

[4] Berra, P.B, G. F., R. Mehotra, and O. Sheng. Introduction multimedia information systems. *IEEE Transactions on Knowledge and Data Engineering*, 5 No 4, August 1993.

[5] A. Borning. Graphically defining new building blocks in TingLab. *Human-Computer Interaction*, 2(4):269–295, 1986.

[6] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. pages 505–516, 1996.

[7] H. S. E. Wedlund. Mobility support using SIP. In *Second ACM/IEEE International Conference on Wireless and Mobile Multimedia(WoWMoM'99)*, Seattle, Washington, August 1999.

[8] R. K. Ege. Defining constraint-based user interfaces. *IEEE Database Engineering, Special Issue on Whatever Happened to Semantic Modeling*, 11(2), 1988.

[9] R. K. Ege. Reading large volumes of java objects from database. In *Proceedings of Technology of Object-Oriented Languages and Systems(TOOLS USA) Conference*, Santa Barbara, CA, August 2000. IEEE Computer Society Press.

[10] E. J. Gugliemo and N. C.Rowe. Natrual language retrieval of images based on descriptive captions, May 2000.

[11] K. Hildrum, J. Kubiatowicz, S. Rao, and B. Zhao. Distributed object location in a dynamic network. In *Proceedings of the 14th ACM Symposium on Parallel Algorithms and Architectures(SPAA)*, 1997.

[12] P. Maymounkov, Mazieres, and 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*, Cambridge, MA, March 2002. Springer-Verlag.

[13] M.Handley, ACIRI, and H. Schulzrinne. Technical report.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. a scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, San Diego, CA, 2000.

[15] A. Rowstron, Druschel, and 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*, pages 329–350, Heidelberg, Germany, November 2001.

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

[17] G. Wiederhold. Mediation to deal with heterogeneous data sources. Jan. 1999.

[18] G. Wiederhold and M. Genesereth. The conceptual basis fo mediation services. *IEEE Expert*, 12 No. 5:38–47, Sep.-Oct. 1997.

[19] W.W.Chu and Q.Chen. A structured approach for cooperative query answering. *IEEE Transactions on Knowledge and Data Engineering*, 6 No. 5, October 1994.