Peer to Peer Database Integration System

GRADUATE TECHNICAL REPORT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University-Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Chunhui Jin
Summer 2008

Committee Members

Dr. Longzhuang Li
Committee Chairperson

Dr. John Fernandez
Committee Member

Dr. Dulal Kar
Committee Member
Peer to Peer Database Integration System

GRADUATE TECHNICAL REPORT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University-Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Chunhui Jin
Summer 2008

Committee Members

Dr. Longzhuang Li
Committee Chairperson

Dr. John Fernandez
Committee Member

Dr. Dulal Kar
Committee Member
ABSTRACT

This project implements a peer to peer (P2P) database integration system focusing on fish data and some Gulf of Mexico Coastal Ocean Observing System (GCOOS) data sources. The system contains two parts, called Clawer and Integrator. Clawer is a Web based application can that retrieve fish and GCOOS data to a local temporary database. Integrator subsystem is a network of databases with different schemas that are connected by some rules. Each database in the system is called a node or peer, using Juxtapose (JXTA) technology to communicate with others. Each node can be queried in its schema for data, and the node can also fetch data from its neighbors if the particular coordination rule is correct. The application is able to extract data from remote Websites, and then use the existing source as a peer to share data by JXTA. For example, a user can get both fish and other information on the peer that only has fish data, when its neighbor peers have corresponding data sources. The two important characteristics of this project are no global schema, and only temporary local storage.
TABLE OF CONTENTS

Abstract ................................................................................................................................................. ii

Table of Contents ................................................................................................................................. iii

List of Figures ......................................................................................................................................... vi

1. Introduction and Background .......................................................................................................... 8
   1.1 Concept of Data Integration .......................................................................................................... 8
       1.1.1 Theory of Mediator-Based Data Integration System ........................................................... 11
       1.1.2 Query Processing .................................................................................................................. 12
   1.2 Peer to Peer Technology ............................................................................................................. 13
       1.2.1 Advantage of Peer to Peer Technology .............................................................................. 13
   1.3 Peer to Peer Technology in Data Integration ............................................................................. 14
       1.3.1 Problems of Current Data Integration .............................................................................. 14
       1.3.2 Peer to Peer Database Integration System ........................................................................ 16
   1.4 Some Previous Related Projects ............................................................................................... 16
   1.5 Data Used in the Project ............................................................................................................ 17

2. Peer to Peer Database Integration System ....................................................................................... 19
   2.1 Overview of this Project ............................................................................................................. 19
   2.2 Technologies Used in the Project ............................................................................................. 20
       2.2.1 JXTA ................................................................................................................................ 20
       2.2.2 MySQL and Apache Tomcat ............................................................................................... 22
       2.2.3 Jericho and HttpClient ....................................................................................................... 23
   2.3 GUIs and Related Functions ...................................................................................................... 23
       2.3.1 Peers Monitor GUI .............................................................................................................. 24
2.3.2 Queries GUI.................................................................26
2.3.3 Results GUI............................................................27
2.3.4 Statistics GUI..........................................................28
2.3.5 Clawer’s GUI............................................................29

3. System Design...............................................................32
3.1 Development Environment .............................................32
3.2 System Design Details ....................................................33
  3.2.1 Analysis Phase.......................................................33
  3.2.2 Design Phase ........................................................34
  3.2.3 Database Schemas..................................................36
  3.2.4 Implementation Phase.............................................37
    3.2.4.1 Detailed Implementation of Clawer.......................38
    3.2.4.2 Detailed Implementation of Integrator...................39

4. Evaluation and Results...................................................44
4.1 Clawer Evaluation.......................................................44
  4.1.1 FishBase Website Test..........................................44
  4.1.2 TABS Website Test..............................................45
4.2 Project Evaluation......................................................47
  4.2.1 Line Network.......................................................47
  4.2.2 Ring Network.......................................................49
  4.2.3 Tree Network.......................................................52
  4.2.4 Star Network.......................................................53
  4.2.5 Fully Connected Network......................................54
4.3 Some Improvements and Remains.................................................................55

4.4 Final Results........................................................................................................56

5. Future Work.............................................................................................................58

6. Conclusion..............................................................................................................59

Bibliography and References...................................................................................61

APPENDIX – Introduction to JXTA Protocols.........................................................63
LIST OF FIGURES

Figure 1.1 Data warehousing architecture ..............................................................9
Figure 1.2 Mediated schema ....................................................................................10
Figure 1.3 Partial stations or buoys in Gulf of Mexico ...........................................15
Figure 1.4 Architecture of peer to peer system .....................................................16
Figure 2.1 Project GUI A: peers tab ........................................................................24
Figure 2.2 Network management buttons ..........................................................26
Figure 2.3 Project GUI B: queries tab .....................................................................27
Figure 2.4 Project GUI C: results page .................................................................28
Figure 2.5 Project GUI D: statistics tab .................................................................29
Figure 2.6 Project GUI E: FishBase ......................................................................30
Figure 2.7 Project GUI F: TABS ...........................................................................31
Figure 3.1 Flowchart of Clawer ............................................................................33
Figure 3.2 Flowchart of Integrator .......................................................................34
Figure 3.3 Architecture of the peer .....................................................................35
Figure 3.4 Design of Integrator ............................................................................36
Figure 3.5 Format of coordination rule .................................................................42
Figure 4.1 FishBase test .......................................................................................44
Figure 4.2 TABS test A .......................................................................................45
Figure 4.3 TABS test B .......................................................................................46
Figure 4.4 Classifications of network topology ....................................................47
Figure 4.5 Line network A ....................................................................................48
Figure 4.6 Line network B ....................................................................................49
Figure 4.7 Ring network..................................................................................51
Figure 4.8 Tree network.................................................................................53
Figure 4.9 Star network..................................................................................54
1. INTRODUCTION AND BACKGROUND

The problem of combining heterogeneous data sources under a single query interface is not a new one. The rapid adoption of databases after the 1960s naturally led to the need to share or merge existing repositories. This merging can be done at several levels in the database architecture. One popular approach is data warehousing, where data from several sources are extracted, transformed, and loaded into a single source that can be queried with a single schema [Daswani 2003].

The recent trend in data integration has been to loosen the coupling between data. The idea is to provide a uniform query interface over a mediated schema. Some of the current work in data integration research concerns the semantic integration problem. This problem is not about how to structure the architecture of the integration, but how to resolve semantic conflicts between heterogeneous data sources [Alvarez 2005].

All above are mediator-based data integration methods; however, there is another way to do integration using peer to peer (P2P) technology. This project develops a P2P data integration application based on Juxtapose (JXTA) technology, a group of protocols supporting P2P.

1.1 Concept of Data Integration

Data integration is the process of combining data residing in different sources and providing the user with a unified view of these data. This process emerges in a variety of situations, both commercial (when two similar companies need to merge their databases) and scientific (combining research results from different bioinformatics repositories). Data integration appears with increasing frequency as the volume and the need to share
existing data explodes. It has been the focus of extensive theoretical work and numerous open problems remain to be solved [Daswani 2003].

There are several solutions which are discussed in the following sections:

a. Data Warehousing

Figure 1.1 shows the basic components of this method that can be perceived architecturally as a tightly coupled approach because the data reside together in a single repository at query time. Problems with tight coupling can arise with the "freshness" of data, for example, when an original data source is updated, but the warehouse still contains the older data and the extract, transform, load (ETL) process needs to be executed again. It is also difficult to construct data warehouses when you only have a query interface to the data sources and no access to the full data. This problem frequently arises when integrating several commercial query services like travel or classified advertisement Web applications [Jun 1998].

Figure 1.1 Data warehousing architecture [Jun 1998]
b. Mediated schema

Figure 1.2 shows the architecture of the mediated schema method. Any query is transformed into specialized queries over the original databases. This process can also be called as view based query answering because designers can consider each of the data sources to be a view over the nonexistent mediated schema. Formally such an approach is called Local As View (LAV) — where "Local" refers to the local sources/databases. An alternate model of integration is one where the mediated schema is designed to be a view over the sources. This approach called Global As View (GAV) — where "Global" refers to the global (mediated) schema — is often used due to the simplicity involved in answering queries issued over the mediated schema. However, the obvious drawback is the need to rewrite the view for mediated schema whenever a new source is to be integrated and/or an existing source changes its schema [Jun 1998].

![Figure 1.2 Mediated schema [Jun 1998]]
1.1.1 Theory of Mediator-Based Data Integration System

Data integration systems are formally defined as a triple $<G, S, M>$ where $G$ is the global (or mediated) schema, $S$ is the heterogeneous set of source schemas, and $M$ is the mapping that maps queries between the source and the global schemas. Both $G$ and $S$ are expressed in languages over alphabets comprised of symbols for each of their respective relations. The mapping $M$ consists of assertions between queries over $G$ and queries over $S$. When users pose queries over the data integration system, they pose queries over $G$ and the mapping then asserts connections between the elements in the global schema and the source schemas [Bernstein 2002].

In GAV, the global database is modeled as a set of views over $S$. In this case $M$ associates to each element of $G$ a query over $S$. Query processing becomes a straightforward operation because the associations between $G$ and $S$ are well-defined. The burden of complexity is placed on implementing mediator code instructing the data integration system exactly how to retrieve elements from the source databases. If any new sources are added to the system, considerable effort may be necessary to update the mediator, thus the GAV approach should be favored in cases where the sources are not likely to change [Bernstein 2002].

On the other hand, in LAV, the source database is modeled as a set of views over $G$. In this case $M$ associates to each element of $S$ a query over $G$. Here the exact associations between $G$ and $S$ are no longer well-defined. As is illustrated in the next section, the burden of determining how to retrieve elements from the sources is placed on the query processor. The benefit of an LAV modeling is that new sources can be added
with far less work than in a GAV system, thus the LAV approach should be favored in cases where the mediated schema is not likely to change [Bernstein 2002].

1.1.2 Query Processing

The theory of query processing in data integration systems is commonly expressed using conjunctive queries. One can loosely think of a conjunctive query as a logical function applied to the relations of a database. If a tuple or set of tuples is substituted into the rule and satisfies it, then designer consider that tuple as part of the set of answers in the query. While formal languages like Datalog express these queries concisely and without ambiguity, common SQL queries are classified as conjunctive queries as well [Chin 2003].

In GAV systems, a system designer writes mediator code to define the query rewriting. Each element in the user's query corresponds to a substitution rule just as each element in the global schema corresponds to a query over the source. Query processing is simply expanding the subgoals of the user's query according to the rule specified in the mediator and thus the resulting query is likely to be equivalent. While the majority of the work is done beforehand by the designer, some GAV systems involve simplifying the mediator description process [Kementsietsidis 2003].

In LAV systems, queries undergo a more radical process of rewriting. This is because there is no mediator to align the user's query with a simple expansion strategy. The integration system must execute a search over the space of possible queries in order to find the best rewrite. The resulting rewrite may not be an equivalent query but maximally contained, and the resulting tuples may be incomplete. If the space of rewrites
is relatively small this is not a problem even for integration systems with hundreds of sources [Kementsietsidis 2003].

1.2 Peer to Peer Technology

P2P technology in computer networks uses diverse connectivity between participants in a network and the cumulative bandwidth of network participants rather than conventional centralized resources where a relatively low number of servers provide the core value to a service or application. P2P networks are typically used for connecting nodes via largely ad hoc connections. Such networks are useful for many purposes. For instance, file sharing containing audio, video, data or anything in digital format is very common, and real time data, such as telephony traffic, is also passed using P2P technology [Calvanese 2003].

A pure P2P network does not have the notion of clients or servers, but only equal peer nodes that simultaneously function as both “clients” and “servers” to the other nodes on the network. This model of network arrangement differs from the client-server model where communication is usually to and from a central server. A typical example of a file transfer that is not P2P is an FTP server where the client and server programs are quite distinct, the clients initiate the download/uploads, and the servers react to and satisfy these requests. However in a pure P2P network, each node can download from other nodes and also can be downloaded by other nodes [Serafini 2001].

1.2.1 Advantage of Peer to Peer Technology

An important goal in P2P networks is that all the peers provide resources, including bandwidth, storage space, and computing power. Thus, as nodes arrive and
demand on the system increases, the total capacity of the system also increases. This is impossible for client-server architecture with a fixed set of servers, in which adding more clients could mean slower data transfer for all users [Adjiman 2006].

The distributed nature of P2P networks also increases robustness in case of failures by replicating data over multiple peers, and in pure P2P systems, by enabling peers to find the data without relying on a centralized index server. In the latter case, there is no single point of failure in the system [Nejdl 2003].

1.3 Peer to Peer Technology in Data Integration

Unlike the other techniques, in P2P systems, data integration is not based on a global schema. Instead there is a network of peer nodes which can define mappings from themselves to other nodes and form a logical association between peers. This can be exploited during query evaluation, where queries can be passed along the logical associations, thus exploiting the entire network [Adjiman 2006].

1.3.1 Problems of Current Data Integration

In the modern world, organizations keep large databases for their own use, but it is often the case that information in one organization can be useful to another and vice versa and therefore, if the two organizations could share their data, it would mean that their “information power” would increase and benefit each other. However, the databases of the two organizations are typically heterogeneous which does not allow the data to be unified. For example, Figure 1.3 shows some observational stations or buoys in the Gulf of Mexico. Red points belong to Texas Coastal Ocean Observation Network (TCOON), blue ones belong to Texas Automated Buoy System (TABS), and other colors represent
other institutions. If one wants to compare data from both TCOON and TABS, it is very complicated. He has to login to both Websites, getting the same category data at the same time. Sometimes the data is huge, such as in TCOON water temperature is recorded every 6 minutes and in TABS is 30 minutes. Such a situation makes it hard to analyze the data.

Generalizing the above, it is not surprising that large systems need an integrated virtualized access to distributed information resources. The approaches to do this is to somehow form a mapping between schemas such that either the data that is defined in a schema can be materialized in the mapped schema or a query expressed to a schema can be translated to a query defined to the mapped schema. Most approaches have the notion of local and global schemas. Local schemas are the schemas of the sources and global schemas are the shared ones. But the problem is when database integration systems have many kinds of schemas and the number of schemas is large, designing a global schema looks very hard. Back to the Gulf of Mexico example, each institution collects data in a way suitable to its interest (different domains), resulting in the lack of a common or standard format adopted for date representation, and also the collaboration between organizations is immature. Therefore using the P2P technique (Global schema free) to build a data integration system is important and exceeding useful.

Figure 1.3 Partial stations or buoys in Gulf of Mexico [GCOOS 2008]
1.3.2 Peer to Peer Database Integration System

Figure 1.4 illustrates the architecture of a P2P integration system. In the graph, each block represents a peer node containing local schema and local data sources, and the sources could be from different domains. Within the peers, there are local mapping rules to map real data attributes to peer schema. Among the peers, there are other mapping standards for each two particular peers, such as the figure shows Peer 1 map to Peer 5, Peer 3 map to Peer 1 and so on.

![Figure 1.4 Architecture of peer to peer system](image)

**Figure 1.4 Architecture of peer to peer system**

1.4 Some Previous Related Projects

P2P Data integration systems increasingly attracted the attention of the database community during the last few years. Serafini proposed a model for data integration in a P2P environment [Serafini 2001], called the Logical Relational Model (LRM). Here, no global schema is assumed and coordination formulas allow propagating updates from one peer to another. In the Hyperion project [Arenas 2003], mapping tables are used for
information integration. Mappings are defined in each peer by a data curator. A different approach is followed by Halevy in the Piazza project [Halevy 2003a], where GLAV rules are defined in sets of peers denoted acquaintances. The PeerDB system [Siong 2003] optimizes network traffic using peer proximity criteria. The CoDB project [Franconi 2004a] is a relatively new system which the research can use as a suitable reference of this project. It also uses GLAV rules to define mappings between the peers in P2P network. The approach appears to be similar to Piazza in terms of query distribution and termination. The difference is coDB develops a particular rule between nodes, and there are some special parts in local database handling.

So far, the best two projects of P2P data integration systems are Piazza and coDB, both of which use peer schemas to describe each peer. The set of mappings of coDB defines its topology network. They both use corresponding search algorithms based on the Extensible Markup Language (XML). The coDB project uses JXTA for message transportation that is also XML based.

1.5 Data Used in the Project

This project focuses on the data sources of the Gulf of Mexico. There are two domains of data: fish and environmental data.

For fish data, the Websites includes FishNet (http://www.fishnet2.net/) and FishBase (http://www.fishbase.org/). Each site has its own schema, and the system designer analyzes each of them and develops a corresponding extractor to retrieve the data. Presently the FishNet Website has some problems to extract data, so committee
chair Dr Li offered me some fish data from Dr. Thomas Shirley who is an endowed chair at the Harte Research Institute.

The largest one is the environmental data domain which has lots of Websites. For example, the GCOOS has more than twenty branch Websites, each of which contains many kinds of attributes. Because it is too big for this project, the system implementer only focuses on two of them, TCOON (http://lighthouse.tamucc.edu/) and TABS (http://tabs.erg.tamu.edu/).

So far, the developer has introduced the background of P2P data integration system. In the next chapter, the author will give an overview of this project, and some technologies used in it. There also have some GUI figures within their functions to be shown.
2. Peer to Peer Database Integration System

This project shares the idea of the Piazza and coDB systems. The vision of the Piazza peer data management system (PDMS) project is to provide semantic mediation between an environment of autonomous and heterogeneous peers, each with its own schema [Franconi 2004b] [Halevy 2003a]. The coDB project introduces a general logical and computational characterization of P2P database systems [Franconi 2004a]. Rather than requiring the use of a single, uniform, centralized mediated schema to share data between peers, both projects allow peers to define schema mappings between pairs of peers (or among small subsets of peers). In turn, transitive relationships among the schemas of the peers are exploited so the entire resources of the PDMS can be used [Halevy 2003a].

The original Piazza and coDB systems are limited in the fact that they do not allow complex mapping rules (i.e., mappings can only support one domain); they only support acyclic networks, and Piazza does not allow for dynamic networks (i.e., networks where peers may join or leave anytime) [Halevy 2003b]. Another problem is both systems have the inconsistency problem, sometimes they return unexpected results.

2.1 Overview of this Project

Basically, the project includes two steps. One is the implementation of extracting data from remote Websites on demand and storing them in a temporary database on a peer. The other is to share the stored data with other peers using the idea of coDB and Piazza. And the designer has addressed most of the problems described in last paragraph.
To make it easier to understand, the author from now on calls the first part of the project “Clawer” and the second part “Integrator”. The basic function of the project is: user can use Clawer to extract fish and environmental data from source Websites, and then use Integrator to share data with other users in P2P network. For example, a user has some fish data, and wants to see the environmental conditions as well. He can send a particular query to other peers on the network, and then peer(s) who has the data receives this query and sends the result back to the user. The program can be run on multiple PCs and it is network independent.

The project partially solved the previous projects’ problems. It can handle the inconsistency when the peer network is not a cycle, and supports complex mapping rules with different domains. Also, it can handle the dynamic networks.

2.2 Technologies Used in the Project

When implementing the project, the designer finished the Integrator part using JXTA protocols and MySQL. The first idea is to add the Clawer part into the Integrator, but later during the development, the implementer gave up this idea because Clawer is Web based application using Java Server Pages (JSP) and Tomcat. Clawer and Integrator have totally different base, it is meaningless to combine them.

Before introducing the GUI part, the author would like to have a brief talk about the technologies that used during the implementation.

2.2.1 JXTA

JXTA is an open network computing platform designed for P2P programming. It provides a common set of open protocols backed with open source reference
implementations for developing P2P applications. The JXTA protocols standardize the manner in which peers can discover each other, self-organize into peer groups, advertise and discover network resources, communicate with each other and monitor each other.

There are six protocols [JXTA 2007]: (Details will be introduced in Appendix)

- Peer Discovery Protocol (PDP)
- Peer Information Protocol (PIP)
- Peer Resolver Protocol (PRP)
- Pipe Binding Protocol (PBP)
- Endpoint Routing Protocol (ERP)
- Rendezvous Protocol (RVP)

The designer uses PDP, PRP and PBP to finish the P2P Integrator’s design. PDP is used to find other peers on the network and organize a peer group, PRP is in charge of sending queries and PBP is used to build virtual channels for information transportation, such as binding queries with pipes. Also, the implementer bind both ends of a pipe with the endpoints of two acquainted peers using ERP, for example a TCP port.

Using JXTA technology, the designer is able to write applications that can (1) find other peers on the network with dynamic discovery across firewalls and NATs, (2) easily share resources with anyone across the network, (3) find available content at network sites and (4) create a group of peers that provide a service [JXTA 2007]. In this project, the implementer first puts all the nodes into the same peer group which is the precondition of P2P computation.

The JXTA network consists of a series of interconnected peers. A peer may be any type of device and each peer provides a set of services and resources which it makes
available to other peers [JXTA 2007]. In this project, the service of Integrator part is the interactive program which can fulfill temporary database integration.

JXTA uses sockets and pipes to send messages to one another. JXTA sockets are bidirectional connections used for applications to communicate reliably. Here “reliably” means the messages are surely sent and received by sockets, but the “unreliable” is that messages could be altered. Pipes are an asynchronous and unidirectional message transfer mechanism used for service communication [JXTA 2007]. The project uses pipes to send queries, receive results and so on.

All JXTA advertisements and messages are XML based, which means in this project the mapping rules between peers is an XML file. Otherwise the rules are not able to be transmitted in the JXTA P2P network. Also JXTA supplies secure login, user can add this function before starting the JXTA application, and then the program will require username and password to enter. The default is empty.

2.2.2 MySQL and Apache Tomcat

The MySQL database has become the world’s most popular open source database because of its consistent fast performance, high reliability and ease of use. It's used on everywhere by individual Web developers as well as many of the world's largest and fastest-growing organizations.

Apache Tomcat is an implementation of the Java Servlet and JSP technologies. It is the “ASP in Java world”.

This designer uses Java and Java Script language to accomplish the Clawer part of the project, and so MySQL and Tomcat are the best choice. Also they are open source and totally free.
2.2.3 Jericho and HttpClient

For the Clawer part, extracting data from Websites is easy, but remove the noise of the particular Web pages is not that simple. Each Website has its own style, and the noise concludes tags, table, picture, ads and so on. Here writing wrappers for every Websites is not a smart move. Fortunately Java has a third party open source package – Jericho HTML Parser that can make the job easier. Jericho is a java library allowing analysis and manipulation of parts of an HTML document, including server-side tags, while reproducing verbatim any unrecognised or invalid HTML. It also provides high-level HTML form manipulation functions.

Another big problem when the author was developing the project is how to handle PHP Websites. The URL does not change when submitting queries, so at first the author did not know how the variables are passed and how to extract the required data. The solution is HttpClient, which are standards based, pure Java, implementation of HTTP versions 1.0 and 1.1 and full implementation of all HTTP methods.

All the details will be explained in next chapter, and of course, both Jericho and HttpClient are open source and free.

2.3 GUIs and Related Functions

In the following sections, the designer uses some GUIs of the project to explain the functions that the project supports. First is the Integrator’s GUI, and then is the Clawer’s GUI. Some demos for Integrator part had been shown in the proposal paper and the following parts are the final ones.
2.3.1 Peers Monitor GUI

Figure 2.1 shows what the Peers tab of this project look like. The status box on the upper left shows five aspects of a peer: peer name, link direction, availability for input, availability for output and peer status.

At the bottom of the tab, there are two groups of buttons: Peer Management and Network Management. Peer management buttons help in discovering the connecting peers. Search Peers button can discover all existing peers on the P2P network. Build Pipes button is able to construct a pipe between current peer and the peer chosen in the status box. Flush Messages button tests the pipe to see if it is successfully built or not.

For example, Figure 2.1 shows the current peer name is Node-0. By clicking on Search Peers button the peer can find another peer (i.e., Node-1), and then the user can click on the other peer management buttons to build the pipe and test the pipe.

![Figure 2.1 Project GUI A: peers tab](image)

Network Management buttons help in initializing the P2P network. Read Rules button is proposed to read the file that contains the mapping rules between peers. Publish Rules button publishes the rules to related peers, making them acquainted with the
current peer, and then establishes the pipes. **Initialization** button is able to make peers ready for query or allow them to be queried by other peers.

Figure 2.2 is an instance showing how the indicators within the status box are going to change after clicking on related buttons. When the user first clicks **Read Rules** button, Figure 2.2 (a) and (b) show *Node-0* discovers *Node-1* and vice versa. **Links** gives the direction of query. When the *In* lights are green that means both peers are available to receive advertisements. Afterwards, the user clicks **Publish Rules** button on *Node-0*’s panel to publish the mapping rules to *Node-1*. Finally by clicking on **Initialization** button, Figure 2.2 (c) and (d) illustrate that both peers *Out* lights are green because the output pipe has been built successfully. The mapping rules are accepted by *Node-1* and the pipe is available to send or receive messages. The **Ready** light of *Node-1* is green means that it is ready to be queried based on the rules.
2.3.2 Queries GUI

This interface and its functions are the core of the Integrator part. Figure 2.3 illustrates the Queries tab. The left section has three boxes: Available Schemas shows the temporary database schema of a peer, Schema relations contains the table names, and Relation attributes gives the column names of each table. The right section is the query part, including query box, two additional query options and Processed Queries box.

Figure 2.3 shows the temporary database is db_0. The database has five tables: eng, ibci, ibe, met and vet, and in met table there are metid, data, and time and other attributes. At query box, the system user is able to input formatted queries and click Send Query button to send a query out to other peers. Processed Queries box can store the history queries. When user chooses one of the past queries and clicks Show Details button, the system will send out this particular query again, which can save user’s time.
2.3.3 Results GUI

After the query is sent to other peers, there will be results returning back to the local peer. Figure 2.4 shows the GUI of the results which has three boxes. The first is called Query Info containing the start and finish time of the query. The second box contains the Query Instance Information, such as the particular query sentence and its translated SQL sentences. The Results for this Query Instance box includes consonant results from all requesting peers in order.

In Figure 2.4, the third box shows the result in two sections. The first part of data is received from local peer Node-0, and the second part is from Node-1. The program allows the user to query based on two additional options: Execute Query Locally and Force Local Updates at Nodes. If the user selects the former one, the system will send the query to local peer also, otherwise the query will be only sent to other peers on the P2P network. And the results page will not contain the results from local peer anymore. The latter option is designed for real time database update. If the program at Node-1 drops the
temporary database suddenly while the system is running, when the user at Node-0 selects this option and does the same query, he will not get any results from Node-1 because the database at Node-1 has been removed.

Figure 2.4 Project GUI C: results page

2.3.4 Statistics GUI

Figure 2.5 shows the Statistics tab of the system. On the left section, there are several buttons. **Show Local Status** can display the local peer information in the box on the right. The user is able to check other peers’ status using **Collect Network Status** button. Moreover, tester can save information that is displayed in the box and all the links' information into an XML file. On the right section, there are some tabs containing useful related information.
2.3.5 Clawer’s GUI

The main function of Clawer is to extract the related data from a Website and store them in a temporary database. The program is able to gain information from FishBase, Fishnet, TCOON and TABS. Because this part of project is Web based, so the author can combine them together using several JSP pages. Here using FishBase and TABS pages as example.

Figure 2.6 (a) Shows the FishBase page. In the left part is the **IBE Claw Option**, **IBE** means *Information by Ecosystem*. First part is **Table Name**, here is **ibe**. Afterwards is the **Place Name**, right now the program has three ecosystems to choose. On the FishBase Website, there are many ecosystems all over the world, but the project focuses on Gulf of Mexico, therefore the developer only includes Gulf of Mexico, Mississippi and Caribbean Sea. The next is a list box of all the fish families defined by FishBase Website. The user can choose one or more the shrink the result. Because for each ecosystem, the
data is large containing fish from all the families, this option is designed for experienced fish researchers to get the particular fish data by family.

When user chooses the place and click the **Start Claw** button, the system will extract the real time data from FishBase Website, show the extraction data on the page and store them into temporary database. Figure 2.6 (b) shows the result of Gulf of Mexico from the Website.

![FishBase Data Claw System](image)

**Figure 2.6 Project GUI E: FishBase**

Figure 2.7 (a) shows the TABS data claw page. First part is *Buoy Data*, showing the result. *Tabs Claw Option* has the similar boxes as FishBase. User can choose which buoy from the *Buoy* drop down list, and choose the *Start Date* and *End Date*, later choose which type of data, TABS contains three types of data: Velocity, Meteorological and System Data. Finally by clicking the **Start Claw** button, the program will start working and soon return the result. Figure 2.7 (b) shows the velocity data of buoy D from 10/11/2008 to 10/15/2008.
In Clawer pages, user can choose more than one options is the list box. Also there is an option *clean table*, once user tick it, the program will use new data overwrite the old one. If not, the program will append the new data to the old one.

So far the author finishes introducing the GUI part of this project. In the next chapter, the developer will focus on the internal section of the project and explain all the problems he met and how he solved them.
3. System Design

The purpose of this project is to build a P2P network for several peers, each of which can extract data from Websites of different domains and store into temporary database, and then each peer using particular mapping rules to share data with others. This design makes the network a cycle, which may bring the inconsistency to the system. Eventually, the developer makes a new design: adding a coordination peer to the network, which has no data, this peer can query all the data of different domains from all the peers, and other peers can also communicate with each through this special peer.

This chapter firstly introduces the development environment of the project, then the analysis, design and implementation phases using software engineer’s point of view. After that are the temporary database tables and schemas. Finally the author shows the encountered and how he solved them.

3.1 Development Environment

The designer uses Java and Java Script as the developing language, using Java SE Development Kit (JDK) 1.6, Netbeans IDE 6.1, Apache Tomcat 5.5 and MySQL Server 5.0 to be the developing tools. The basic idea of this application is P2P technology, and the foundation to create P2P network is JXTA. The current version of JXTA is JXTA for Java SE (JXSE) 2.5. Besides, there are some third party open source Java packages that used in the project: Common HttpClient 3.1 and Jericho HTML Parser 2.6. Finally, the developer uses a browser plug-in called firebug to view the parameters.


3.2 System Design Details

In the previous chapter, the author talked about the two parts in the project: Clawer and Integrator. The following sections are to document the analysis, design and implementation phases of the two parts, and how to combine both as a whole system. Also the DB schemas will be given, too.

3.2.1 Analysis Phase

First of all, the program needs to be run on multiple machines, assuming each machine has one user. The author gives the definition of a peer in this particular project: user + program + temporary database. Figure 3.1 shows the flow chat of Clawer: the dashed part is the peer based on the definition. User visits the Websites through Common HttpClient, getting the required data and wrapping the data using Jericho HTML Parser. Right now the data is well formatted and stored in tables, which can be sent back to the User and to the Temporary Database.

![Figure 3.1 Flow chart of Clawer](image)

Figure 3.1 Flow chart of Clawer

Second, when each peer has its temporary data, the Integrator can work to share the data among all the peers. Of course the Coordination Peer is different, the system treats it as a special peer, which can either have data or not. Figure 3.2 shows the Flow
chart of Integrator: the graph has five peers, four regular and one special peer. The solid lines and peers construct a “star” network, coordination peer can get all the data from this kind of network and also it is the “agency” allowing other peers to communicate. All the solid lines and dashed lines build a fully connected network, more complex than ring network. But this network could return inconsistent result, which is the major problem will be discussed later.

![Flow chart of Integrator](image)

**Figure 3.2 Flow chart of Integrator**

After introducing how the two parts work, the next part is the design of them, includes the detailed components of a peer, the Clawer and Integrator.

### 3.2.2 Design Phase

Figure 3.3 gives the inside components of a peer. On the left in the rectangle, a user needs a *User Interface* to control the peer, and underneath is the *Database Manager* controlling the *Wrapper* to wrap the temporary database. Also the *Database Manager* can use *JXTA Layer* open a port communicating with other peers (users).
Figure 3.3 Architecture of the peer [Franconi 2004b]

Figure 3.4 gives the design of Integrator part. Basically, the project deals with data from two big domains: GCOOS and Fish. Actually inside each the detailed schemas are very complicated. Later the author will give some of the attributes of those schemas which are too complicated to build a global schema. At this situation, the major advantage of this project is distinct: the user does not need to create “huge” global schema, instead he can easily create some mapping rules to realize the data integration through P2P network. In Figure 3.4, there are five peers or nodes. Peer 0 is the coordination peer, Peer 1 and Peer 2 contain the fish data, Peer 3 and Peer 4 have the GCOOS data. The solid lines mean that Peer 0 can send query to all other peers to get data from every domain. The dashed lines show that pairs of peers from same base domain or not can send and retrieve data from each other. For example, FishBase and FishNet can communicate with each other, same as TCOON and TABS. Moreover, even FishBase and TCOON can send query to each other. The return could or could not make sense, depends on the queries.

Coming to the design of Clawer, it is pretty much like the analysis part. Through Figure 3.1, the developer clearly knows what is going to be done step by step. First build the framework of the JSP pages, then for each Website write a Java file to contain the login function and particular wrapper function. Of course the implementer should analyze
the Websites before writing the wrapper, here “analyze” is nothing but know the schemas and the Web page noise for each Website. Second, write some base function that each Java file can call, such as the common login function, and the regular parser function.

Figure 3.4 Design of Integrator

3.2.3 Database Schemas

After analyzing all the Websites, the developer figured out the schema for every Website. Before talking about the implementation phase, here the author gives the DB schemas for all the peers based on Figure 3.4.

FishBase:

It has two tables on the Website.

Table 1: ibe (information by ecosystem)

- id
- species
- name
- family
- habitat
- length
- trophic
- condition
- place

Table 2: ibci (information by country/island)
id, type, family, species, condition, other, name, place.

FishNet:

id, family, depth, range, reference, chapter, noc (number of chapter)

TABS:

It has three tables based on the Website.

Table 1: ven (velocity data)

id, data, time, east, north, speed, dir, waterT, buoy.

Table 2: met (meteorological data)

id, data, time, east, north, ailT, atmPr, gust, comp, tx, ty, par, relh, wSpeed, wDir, buoy.

Table 3: eng (system data)

id, data, time, vBatt, SigStr, comp, nping, ADCP_Volt, ADCP_Curr, vBatt_Sleep, Buoy.

TCOON:

id station (id), name station (name), battery voltage (bat), air temperature (atp), water temperature (wtp), wind speed (wsd), wind gust (wgt), wind direction (wdr).

Among all the schemas, only the type id in all the tables is int, others are defined as varchar (data type in MySQL).

3.2.4 Implementation Phase

The Clawer is a Web based JSP application which can be run under the environment of Tomcat and MySQL. The Integrator is a JXTA based P2P database integration program that can be run under the environment of MySQL. The connection
between both components is the temporary database, where Clawer stores into and Integrator reads from.

### 3.2.4.1 Detailed Implementation of Clawer

To begin with, there are several steps the writer needs to finish.

**Step 1:** creates the public framework for JSP pages including tables, list boxes, drop down lists and messages, these files are `index.jsp`, `ibe.jsp`, `ibci.jsp` and `tabs.jsp`. And also use a `global.css` file to control the format of pages.

**Step 2:** writes `SysConst.java` containing all the static variables such as table names, URLs and so on. Now construct some base classes using default constructors to maintain the retrieved data, in the Clawer they are called `CountryCheck.java` and `FishEco.java`.

**Step 3:** starts writing the core of the Clawer: `GetIBEData.java`, `GetIBCIData.java`, `GetTABSData.java` and other files. These files have the functions to visit corresponding Web pages and claw the real time data.

Here the developer met a major problem, for TABS Website, the language is PHP, which means when doing the query on that Website, the URL does not change. In this situation, user does not know which parameters the current Web page sends and receives and can not get the data. Before this happens, developer is familiar with HTTP Web extraction because the change of URLs clearly reflects the parameters’ change. It takes about one month to solve this PHP extraction problem, and the solution is the third party open source Java package – Common HttpClient. This package has three methods can help the Clawer passing parameters to the TABS Website to request data. They are `GetMethod`, `PostMethod` and `HttpMethodParams`. Right now the problem seems to be
solved, but there is another problem, what exactly the parameters are. To get this job done, the implementer needs a plug-in for Web browser. If using IE, there has HttpWatch plug-in, and if is Firefox, there is a firebug plug-in. Firebug is utilized because it is free. Right now the readers can understand why in DB schema section, the table names are ven, eng and met, those are the real parameters transferring between TABS Website. By using all have learnt above, developer writes his own common function getContentByPostMethod() in HttpClientUtil.java file, this function reads the post URL and some parameters then return the content of the post URL, called post data.

So far the Claws can retrieve the original data from either HTTP or PHP Websites. Next work is to remove the noise and only keep the useful data. To get it done, the implementer needs to write particular wrapper for each Website, and that is not hard but too inefficient. Fortunately, there is another third party package called Jericho, which is an html parser. It does a good job on removing the useless tabs in the source code of Web pages, and also it is very easy to use. User only needs to transfer the original content to the Jericho format, called Segment, and then can use any Jericho functions to modify the segment. For example, List dataRow = segment.findAllElements("pre") gets all the data between tabs <pre> and </pre> and store it into the list dataRow. That’s pretty neat.

Moreover, there are some other java files in the Claws that make it work better. But they are not the core so the author right here does not need to take time to introduce them.

3.2.4.2 Detailed Implementation of Integrator

Compared with Claws, Integrator is more complex. The beginning is similar, AllVariables.java and Constants.java contain all the static variables and constants in the
Integrator. I18n.java guarantees the program can be run on Java platforms are not English (i18n is an abbreviation for internationalization with the starting letter i, has 18 characters in the middle, and ends with the letter n.). CoordinationRules.java defines the XML format of the mapping rules. Peers.java gives the definition of a peer including its id, inlink, outlink, adv, pipes, listeners and so on. Query.java indicates the format of a query in this project containing query instances, time, id, path and other information.

After the preparation, the developer needs to initialize the P2P network first. To test the network and nodes (peers) are ready or not, user needs to run the functions in InitializedNetAdvImpl.java and NodereadyAdvImpl.java. The former can test if the network is ok and the latter tests the nodes. Also most of the definitions of the functions are in the prototype files: InitializedNetAdv.java and NodereadyAdl.java.

After the initialization, each peer on the network generates the GUI and wraps the temporary database. PeersManagementPanel.java, SendQueryPanel.java, QueryResultsJFrame.java and StatisticsPanel.java develop the GUI discussed in Chapter 2. DBField.java, DBScheme.java and DBTable.java implement the functions to make the Integrator identify the tables, schemas and data field.

Next part is the core of the Integrator, which is sending conjunctive queries to peers and getting the results back. To begin, each query is made up with clauses. The author uses RelationClause.java to construct a clause that contains head name, head value, clauseID, and answers. This file also gives the function how to merge the answers and remove the duplicates. The Vlink.java connects the query with the link paths (pipes). The ComparisonClause.java controls the clause that have comparisons, for example, Q(x) :- fish(x,y,z) is a relation clause but Q(x) :- fish(x,y,z); x > 50 is a comparison
clause. So far the author defines all the necessary components of a query.

*CQToSchema.java, ConjQueryVar.java* and *ConjunctiveQuery.java* define the functions, includes the definition of a conjunctive query, construct query with relation and comparison clauses from strings, tests if the query is valid or not, transfer a query into schemas. For instance, peer 1 sends a query to other peers, because the schema is different, the function can translate the peer 1’s query using peer 2’s schema, and transfers the peer 1’s SQL sentence to peer 2’s format. To do this, the mapping rule is the precondition which has been done during the initialization process.

In statistics GUI, there are some functions: *LocalStatVars.java* controls the status of the local peer, *LinkStat.java* and *QueryStat.java* manages status of links and query, includes the time and bandwidth. *Statistics.java* has some other functions collecting other information of the system.

Another very important file that the Integrator uses is the coordination rule file, the *rules.xml*. Figure 3.5 is an example of this standard format which includes:

- File header info: XML version, document type and particular protocol.
- Network ID info: the content between <topology ID> and </topologyID>.
- Current node info: the content between <node> and </node>, includes current nodeID and nodename.
- Mapping rules info: the content between <cr> and </cr>, includes target nodeID, nodename and the particular rule.

In this example, the source node is Node-0 and the target node is Node-1. nodeID is generated by the program, but using fixed nodeID can reduce the inconsistency of the system. Apparently the table in Node-0 is fishbase and Node-1 has fishnet table. The rule
here is mapping all the attributes of fishbase to fishnet. After that, when a conjunctive query is sent from Node-0 to Node-1 using fishbase’s schema, the Integrator system can translate the query using Node-1’s schema and generate a new SQL sentence to make the query look like a “local query” running on Node-1. When Node-1 computes the result, the program does the previous job reversely, sending the result back to the Node-0 and be shown on the panel using Node-0’s schema. One more thing, the mapping rule uses the initial of attributes in tables for reference, and the program use these initials to do the simple semantic mapping. For example, if the column name in fishbase table is a_fish1, and the same thing in fishnet table is a_fish2, the program can treat them like the same semantic attributes and then map these columns when doing the query.

<figure> 

The author also defines a file called c2c_en.properties, which is a support file for I18n.java. In this file contains all the names of the list boxes, buttons, tabs and panels in Integrator program. The purpose is when the project is running on Java platform with other languages, and there will be no confusion code on the screen.
At last, the author gives the main function of the Integrator part. There are
*Main.java*, *Query.java*, *OutputpipeEventsHandler.java*, *InputpipeEventsHandler.java*,
*JXTAInit.java*, *JXTAClient.java* and some other files in the root folder to connect all the
software modules and implement all basic functions, such as initialize the application and
create pipes. *JXTAInit.java* concludes the function to call all the GUI parts.
*JXTAInit.java* has the ability to create a DOS window while the Integrator is running, all
the message is shown in this window which is good for debugging. *Main.java* can call
functions in *JXTAInit* and *JXTAInit* to start the program. Other files can handle the
error events while the system is being initialized.

So far, the developer has finished talking about the detailed design of the whole
project. Next chapter is the evaluation and result and the system will be tested under the
particular P2P networks with different mapping rules to see if the results are the expected
ones or not.
4. Evaluation and Results

4.1 Clawer Evaluation

To make sure that the whole project works well, the precondition is the Clawer returns the correct real time result. Therefore the author decides to test the Clawer individually first to see if the retrieval data matches the Websites’.

4.1.1 FishBase Website Test

Both Website and Clawer chooses the Gulf of Mexico Ecosystem. In Figure 4.1 the left is the result from FishBase Website, and the right part is the Clawer result. At this particular case, Clawer returns 984 results, and it shows 50 results per page. Figure 4.1 also shows the first row and last row are matched, which means the Clawer for FishBase Website can get the right real time data.
4.1.2 TABS Website Test

In Figure 4.2, the upper part is the query interface of TABS Website. Here the test condition is querying the *Velocity Data* from *Buoy B*, two days before 11/04/2008 and two days after it. The lower part is the Clawer, the author chooses *Buoy B*, from 11/02/2008 to 11/06/2008, and the data type is *Velocity Data*. 

![Select Buoy](image1)

**Select Buoy:**

**Select Data Type:** Velocity Data  Meteorological Data  System Data

**Select Start Date:**

**Include in Data Set:**

Output Format:

**Submit Query**  **Reset Fields**

**Tabs Claw Option**

<table>
<thead>
<tr>
<th>Buoy</th>
<th>Start Date</th>
<th>End Date</th>
<th>Data Type</th>
<th>Clean Table</th>
<th>Will Clean Clawed Data If Checked?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>November 2008</td>
<td>2008</td>
<td>Velocity Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 4.2 TABS test A](image2)

Figure 4.3 shows the results from both TABS Website and Clawer. It is clear that Clawer returns 240 columns and all the data is correct.
Figure 4.3 TABS test B

Clawer also works well with FishNet and TCOON data. Here the author just omits the graphs of them. And also when the Clawer shows the result, at the same time it stores all the data to temporary database. After that, user on a peer can use Integrator to read the database and send the queries to other peers.
The next evaluation is test the whole project. After each peer retrieves corresponding data to its temporary database, the system starts running. The network built in the system is a topology P2P network, which has several classifications. To test the communication ability of this project, the developer constructs some type of topology networks including Line, Star, Tree, Ring and Fully Connected. Figure 4.4 gives the five types of networks. The author records the running time for each one, which is based on the average of ten queries. The final result shows which type of network is the most suitable one for this project.

![Figure 4.4 Classifications of network topology](image)

**4.2 Project Evaluation**

During all the tests, the system concludes five nodes (peers): Node-0 to Node-4. The data contained in turn are: coordination peer (N0), FishBase (N1), FishNet (N2), TABS (N3) and TCOON (N4). All the mapping rules are two ways because in this project the P2P network supports bidirectional communication.

**4.2.1 Line Network**
From left to right there are Node-0, 1, 2, 3 and 4, and the coordination rule is created only for neighbor peers. The particular rule here is: N0 maps to N1 and vice versa, the same for N1 and N2, N2 and N3, N3 and N4. During the test, N0 can get data only from N1 and N1 only see N0’s data based on the rule. In Figure 4.5 the left result box shows the result when N0 queries the species, name, family and habitat to N1. In the Query Instance Info box shows the SQL sentence and the execution time is 1.187 second. And Result for this Query Instance box shows clearly that local peer returns 0 result because it has no data, and all the 224 results comes from N1.

Figure 4.5 Line network A

Now based on the previous query, we can narrow the results by adding some additional conditions. For example, when the user wants to know what kinds of fish live
in the bottom of sea, he just needs to add \textit{habitat = 'demersal'}. The right result box in Figure 4.5 shows the result containing only demersal fish.

Not just coordination peer can do the query, every peer on the network can request data if the rules allow it to do. For example, in Figures 4.6 the left box shows the result when N1 sends query to N2 to get the FishBase data. Here \textit{Local Peer} is N1, the result is from N2 not N0 because N0 has no data. And right box illustrates the result when N2 sends query to N3 to get the TABS data.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{line_network_b.png}
\caption{Line network B}
\end{figure}

The advantage of Line Network is that the result is stable and consistent, but the flaw is that each node can only get data from its neighbor peers. This conclusion leads the author to try to connect the head and the tail of the Line Network, which is discussed in the next evaluation: Ring Network.

\subsection*{4.2.2 Ring Network}

This type of the network is similar as the Line Network, only the difference is that the network is a cycle. Compared to the last example, the only change is the developer
changes the rule. To test the consistency, the tester lets all the peers have FishBase data, but from different places, N0-Texas, N1-Florida, N2-Louisiana, N3-Mississippi, N4-Alabama. And the mapping rule is: N0 maps to N1, N1 to N0, N1 to N2, N2 to N1, N2 to N3, N3 to N2, N3 to N4, N4 to N3, N4 to N0 and N0 to N4. Now the user on N0 sends a query to the whole network and the result is what the author expects.

Figure 4.7 shows the result returns data from all the places, the reader may say that it is inconsistent because N0 only maps to N1, so basically user on N0 can only get the data from N1. By taking a look at the results carefully, the author finds that all the results are coming from N1, looks like it is not correct, but actually it is correct. It reflects N0 retrieves all the data even only maps to N1. Here is the explanation: N0 sends query to N1, N1 returns the results. At the same time, because the network is a cycle and mapping rule says N1 maps to N2, so N1 sends the same query to N2, N2 returns result to N1, N1 then sends the N2’ result back to N0 and N0 does not know this part of result comes from N2. Same process for N3 and N4, but it takes longer time. Right now another question is that the rule also specifies that N4 maps to N0, which means N4 should get the result from N0 and sends back to N3, N3 to N2, N2 to N1, finally N1 to N0. The program has this result, too. The last graph in Figure 4.7 shows that N1 also returns the result of Texas, N0 has the Texas data, which proves that in this particular ring network, the return result is consistent. One more thing to mention, user on peer 1 to peer 4 also can do the query, too. For example, if N1 sends the same query, the all the results will be returned from N2 because the rule only maps N1 to N2. The average query time for Ring Network is around 3.5 seconds. Next the user will continue testing the Tree Network.
<table>
<thead>
<tr>
<th>Taxon Family</th>
<th>Genus</th>
<th>Species</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyprinodontiformes</td>
<td>Fundulidae</td>
<td>Aplocheilus vittatus</td>
<td>Texas</td>
</tr>
<tr>
<td>Mugiliformes</td>
<td>Mugilidae</td>
<td>Mugil cephalus</td>
<td>Texas</td>
</tr>
<tr>
<td>Lamniformes</td>
<td>Alopiidae</td>
<td>Alopias superciliosus</td>
<td>Texas</td>
</tr>
<tr>
<td>Lamniformes</td>
<td>Alopiidae</td>
<td>Alopias vulpinus</td>
<td>Texas</td>
</tr>
<tr>
<td>Clupeiformes</td>
<td>Clupeidae</td>
<td>Alosa chrysocloris</td>
<td>Texas</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Centrarchidae</td>
<td>Ameca miniatus</td>
<td>Texas</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameiurus nebulosus</td>
<td>Texas</td>
</tr>
</tbody>
</table>

**Received From: Local Peer; at: Mon Nov 10 11:46:54 CST 2008**

<table>
<thead>
<tr>
<th>Taxon Family</th>
<th>Genus</th>
<th>Species</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perciformes</td>
<td>Pomacentridae</td>
<td>Abudelfa's taxables</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Pomacentridae</td>
<td>Abudefduf taurus</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Centracanthidae</td>
<td>Acantharchus porroide</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Centracanthidae</td>
<td>Acantharchus sexfasciatus</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Centracanthidae</td>
<td>Acantharchus cheliparum</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Acanthopagrus</td>
<td>Acanthopagrus randalli</td>
<td>Florida</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ostariophysi</td>
<td>Ostariophysi nebulosus</td>
<td>Florida</td>
</tr>
</tbody>
</table>

**Received From: Node: 1; at: Mon Nov 10 11:46:54 CST 2008**

<table>
<thead>
<tr>
<th>Taxon Family</th>
<th>Genus</th>
<th>Species</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adiadiidae</td>
<td>Adiadiidae</td>
<td>Adiadius minutus</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Mugiliformes</td>
<td>Mugilidae</td>
<td>Mugil cephalus</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Clupeiformes</td>
<td>Clupeidae</td>
<td>Alosa labrosa</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Centrarchidae</td>
<td>Ambloplites aurora</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameiurus nebulosus</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes baeri</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameiurus nebulosus</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes caurinus</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes vavas</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes meridiana</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameiurus nebulosus</td>
<td>Louisiana</td>
</tr>
</tbody>
</table>

**Received From: Node: 1; at: Mon Nov 10 11:46:55 CST 2008**

<table>
<thead>
<tr>
<th>Taxon Family</th>
<th>Genus</th>
<th>Species</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lamniformes</td>
<td>Alopiidae</td>
<td>Alopias superciliosus</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Lamniformes</td>
<td>Alopiidae</td>
<td>Alopias vulpinus</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Clupeiformes</td>
<td>Clupeidae</td>
<td>Alosa labrosa</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes baeri</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes caurinus</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes vavas</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes meridiana</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes caurinus</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes vavas</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes meridiana</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes vavas</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Cypriniformes</td>
<td>Cyprinidae</td>
<td>Campostoma anomalum</td>
<td>Mississippi</td>
</tr>
</tbody>
</table>

**Received From: Node: 1; at: Mon Nov 10 11:46:55 CST 2008**

<table>
<thead>
<tr>
<th>Taxon Family</th>
<th>Genus</th>
<th>Species</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perciformes</td>
<td>Centrarchidae</td>
<td>Ampholatilus pupfish</td>
<td>Alabama</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameca miniatus</td>
<td>Alabama</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameiurus nebulosus</td>
<td>Alabama</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes baeri</td>
<td>Alabama</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes caurinus</td>
<td>Alabama</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes meridiana</td>
<td>Alabama</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes caurinus</td>
<td>Alabama</td>
</tr>
<tr>
<td>Cypriniformes</td>
<td>Cyprinidae</td>
<td>Campostoma olympias</td>
<td>Alabama</td>
</tr>
<tr>
<td>Cypriniformes</td>
<td>Cyprinidae</td>
<td>Campostoma paucigranulata</td>
<td>Alabama</td>
</tr>
</tbody>
</table>

**Received From: Node: 1; at: Mon Nov 10 11:46:56 CST 2008**

<table>
<thead>
<tr>
<th>Taxon Family</th>
<th>Genus</th>
<th>Species</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyprinodontiformes</td>
<td>Fundulidae</td>
<td>Fundulus grandis</td>
<td>Texas</td>
</tr>
<tr>
<td>Mugiliformes</td>
<td>Mugilidae</td>
<td>Mugil cephalus</td>
<td>Texas</td>
</tr>
<tr>
<td>Lamniformes</td>
<td>Alopiidae</td>
<td>Alopias superciliosus</td>
<td>Texas</td>
</tr>
<tr>
<td>Lamniformes</td>
<td>Alopiidae</td>
<td>Alopias vulpinus</td>
<td>Texas</td>
</tr>
<tr>
<td>Clupeiformes</td>
<td>Clupeidae</td>
<td>Alosa chrysocloris</td>
<td>Texas</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes clara</td>
<td>Texas</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocoetes clara</td>
<td>Texas</td>
</tr>
</tbody>
</table>

51
4.2.3 Tree Network

The tester makes the coordination peer the root of the tree. The rule is: N0 maps to N1, N1 to N0, N0 to N2, N2 to N0, N1 to N3, N3 to N1, N2 to N4 and N4 to N2.

Figure 4.8 illustrates that the result is correct and consistent, containing Texas data from N0 (local peer), Florida data from N1, Louisiana data from N2, Mississippi data from N3 (sending result back to N1, actually shown from N1) and Alabama data from N4 (sending result back to N2, actually shown from N2). The average query time for Tree Network is approximate 3.1 seconds.

<table>
<thead>
<tr>
<th>Received From: Local Peer; at: Tue Nov 11 04:12:44 CST 2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibci.b_type</td>
</tr>
<tr>
<td>Cyprinodontiformes</td>
</tr>
<tr>
<td>Mugiliformes</td>
</tr>
<tr>
<td>Lamniformes</td>
</tr>
<tr>
<td>Lamniformes</td>
</tr>
<tr>
<td>Clupeiformes</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Received From: Node-1; at: Tue Nov 11 04:12:45 CST 2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibci.b_type</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Received From: Node-1; at: Tue Nov 11 04:12:45 CST 2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibci.b_type</td>
</tr>
<tr>
<td>Lamniformes</td>
</tr>
<tr>
<td>Lamniformes</td>
</tr>
<tr>
<td>Clupeiformes</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
<tr>
<td>Perciformes</td>
</tr>
</tbody>
</table>
4.2.4 Star Network

In this particular network, coordination peer (N0) is the central of the network, mapping to all other peers and all other peers map back to N0. There is no connection among other peers. Due to the rule, only N0 can retrieve the results from the network and it should be consistent, and other peers can only get data from themselves. Figure 4.9 reflects the expected result. The results show that Texas data from N0, Florida data from N1, Louisiana data from N2, Alabama data from N4 and Mississippi data from N3. The reason N3’s data is before N4’s is the messages are being sent and received parallel in P2P network, not in order, therefore which result is going to return first is not predictable. Base on the design of this project, the fastest result is shown in the front. The average time of Star Network is around 2.9 second.
<table>
<thead>
<tr>
<th>ibci_b_type</th>
<th>ibci_c_family</th>
<th>ibci_d_species</th>
<th>ibci_h_place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyprinodontiformes</td>
<td>Fundulidae</td>
<td>Adinia xonica</td>
<td>Texas</td>
</tr>
<tr>
<td>Mugiliformes</td>
<td>Mugilidae</td>
<td>Agonostomus monticola</td>
<td>Texas</td>
</tr>
<tr>
<td>Lamineformes</td>
<td>Alopidae</td>
<td>Alopia superciliosus</td>
<td>Texas</td>
</tr>
<tr>
<td>Lamniformes</td>
<td>Alopidae</td>
<td>Alopia vulpinus</td>
<td>Texas</td>
</tr>
<tr>
<td>Clupeiformes</td>
<td>Clupeidae</td>
<td>Alcasa chrysochiroides</td>
<td>Texas</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Cirrhidae</td>
<td>Amblycirrhitus pinos</td>
<td>Texas</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ibci_b_type</th>
<th>ibci_c_family</th>
<th>ibci_d_species</th>
<th>ibci_h_place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perciformes</td>
<td>Pomacongridae</td>
<td>A. feduf saxatilis</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Pomacongridae</td>
<td>A. feduf taurus</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Centrachidae</td>
<td>Acantharchus pomotis</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Chaenopsidae</td>
<td>Acanthemblemaria a</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Chaenopsidae</td>
<td>Acanthemblemaria c</td>
<td>Florida</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Chaenopsidae</td>
<td>Acanthemblemaria s</td>
<td>Florida</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ibci_b_type</th>
<th>ibci_c_family</th>
<th>ibci_d_species</th>
<th>ibci_h_place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acipenseriformes</td>
<td>Acipenseridae</td>
<td>Acipenser oxyrinchus</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Mugiliformes</td>
<td>Mugilidae</td>
<td>Agonostomus monticola</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Clupeiformes</td>
<td>Clupeidae</td>
<td>Alcasa alabamica</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Centrachidae</td>
<td>Ambloplites arilomus</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameiurus nebulosus</td>
<td>Louisiana</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocrypta beanii</td>
<td>Louisiana</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ibci_b_type</th>
<th>ibci_c_family</th>
<th>ibci_d_species</th>
<th>ibci_h_place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perciformes</td>
<td>Centrachidae</td>
<td>Ambloplites rupestris</td>
<td>Alabama</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameiurus brunneus</td>
<td>Alabama</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameiurus nebulosus</td>
<td>Alabama</td>
</tr>
<tr>
<td>Siluriformes</td>
<td>Ictaluridae</td>
<td>Ameiurus serracanthus</td>
<td>Alabama</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocrypta beanii</td>
<td>Alabama</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocrypta bifascia</td>
<td>Alabama</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ibci_b_type</th>
<th>ibci_c_family</th>
<th>ibci_d_species</th>
<th>ibci_h_place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lamniformes</td>
<td>Alopidae</td>
<td>Alopia superciliosus</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Lamniformes</td>
<td>Alopidae</td>
<td>Alopia vulpinus</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Clupeiformes</td>
<td>Clupeidae</td>
<td>Alcasa alabamica</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocrypta beanii</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocrypta clara</td>
<td>Mississippi</td>
</tr>
<tr>
<td>Perciformes</td>
<td>Percidae</td>
<td>Ammocrypta meridiana</td>
<td>Mississippi</td>
</tr>
</tbody>
</table>

**Figure 4.9 Star network**

### 4.2.5 Fully Connected Network
This type of network is built based on the Ring network, but it is more complicated. The rule here maps N0 to N1, N2, N3, N4, maps N1 to N2, N3, N4, maps N2 to N1, N3, N4, maps N3 to N1, N2, N4, and maps N4 to N1, N2, N3. The test can not be done because the rule is too complicated and the P2P network sometimes even can not be initialized. And the results are inconsistent, which contains duplicate or unexpected data. For example, N2 only has the Louisiana data, but the result shows N2 has Alabama data for some queries. The reason will be discussed next paragraph. In a word, due to the evaluation the Fully Connected Network is not appropriate for this project.

4.3 Some Improvements and Remains

As mentioned before, none of the previous projects of P2P data integration systems supports dynamic networks, which means in such systems, if one data source is being added to or dropped from the system, still the query returns the same results. For example, N0 sends query to all the peers and returns the expected results. Suddenly N3 is not connected to the system, at this time N0 sends the same query again, but the result also contains the data from N3. This is not a “real time” system. So far the project implementer has solved this problem. All the peers in the system are dynamic, if one is dropped, the system can run also well without that peer, returning the expected result.

Another problem is the initialization problem. Sometimes nodes in the P2P network cannot be all initialized in the beginning. For example, maybe N1 and N2 are ready but N3 and N4 still not ready. The solution is simple, the user just clicks the Initialization button again until all the nodes are ready. The reason I think is the JXTA communication problem, sometimes the coordination rule cannot be propagated properly.
Another problem is the inconsistent result which both coDB and Piazza have. During the development period, the developer has kept trying to solve this problem. Unfortunately, the problem has only been partially solved. When the peer network of the project is not a Fully Connected Network, the query returns consistent result, but once rule is too complicated such as all the tables of all the peers are mapping or being mapped to each other, some unexpected results occur. For instance, the result could return same fish data from N2 and N3, but N3 does not have fish data at all, or the query returns the same data from one peer twice. After trying all the methods to solve, the designer also went to JXTA forums looking for helps, but the problem is still there. The author believes that the there is no problem with the source code so eventually it might be the matter of JXTA itself. When the P2P network is fully connected, and also the mapping rules are too complex, at this time there are too many pipes at one time, so there could be some conflicts when data is being sent through the pipes. The author guesses the messages could be transmitted more than one pipe, even this message should not be in that pipe, resulting in such kind of problem. Another reason leads the developer into this conclusion is the talk with other JXTA developers on the forums. JXTA was published in 2001, and it was very popular before 2004. After that, the Web page was not updated frequently and the JXTA forum has only a few people discussing very beginner’s issues. This is not normal, later someone told me that JXTA is not mature enough for business development, it has some bugs and it is not stable. That is why JXTA has existed for nearly 8 years, but almost none business application uses this technology to accomplish. The author’s opinion is that JXTA still has a long way to go. It does has some very good advantage such like P2P based and network independent, but unstable is its death-would.
4.4 Final Results

Based on the previous evaluation, the author decided to use the Star Network with a proper coordination rule to finish the final test. The result is acceptable, the coordination peer returns the data with various domains from different peers, and other peers also return the correct results. The detailed results will be presented in the final oral defense.
5. Future Work

The whole project is not mature so far, the author knows that there are some updates that should be done in future.

First is the temporary database problem, the original idea for this project is to not use any local storage to contain the data from remote Websites. That is, the data is stored in data stream all the time, both for Clawer and Integrator. But during the development, when using data stream method to implement the Integrator’s communication functions, something went wrong while messages are being transmitted in pipes, and the result returns nothing. The implementer so far does not know how to solve this problem, and therefore a compromising way comes out: using a temporary database.

Second is the inconsistency. The author mentioned it could be the JXTA problem but it is also could not be. The developer will continue to search for the solution.

The third is that the project should support many domains but right now it can only support several domains because the peer schemas will be complicated and the mapping rule is going to be coupled and may not be used to initialize the P2P network.

Above are the three major problems remaining. Hopefully the author can find the solutions by sustaining work in the future.
6. Conclusion

At present, the Internet has numerous data sources and many huge online databases have query interfaces on their Websites. There is no uniform interface and it is nearly impossible to design such an interface. Is there a simple way to get various data by not visiting all the Websites? The answer is data integration. Some previous projects are done by data warehousing or mediator-based methods. The burgeoning way right now is peer to peer (P2P) data integration.

This project is a P2P application that can integrate the data of various domains from different remote Websites. It has two parts, the extraction part (Clawer) and the integration part (Integrator). In the project’s view, each peer should have a user, an interface, a temporary database. User on a peer uses Clawer to extract data from Websites and store into the temporary database, later uses Integrator to share data with users on other peers. This project does not design the global schema for the data of different domains, but creates a coordination mapping rule for all the peers, letting them communicate with each other using this particular rule. The rule is stored in an XML file, and can only be published by a special peer (coordination peer). A coordination peer
usually contains no data, the task of this peer is to spread the rule to others and initialize the P2P network.

Through the evaluation, the coordination peer can return expected results from different domains by sending queries to all other peers. Moreover, by modifying the rule, the project can also allow other peers sending queries and show their results.

Right now the project contains two domains: fish and GCOOS, focusing on Gulf of Mexico area. For example, user A extracts Texas fish data from FishBase Website by using Clawer, user B has TABS data, user C contains TCOON data and fish data of Mississippi. Using this project, user A can get fish data from Mississippi and TABS data by sending two queries: first one to user C and the second to user B. The advantage of the project is the user does not need to worry about the global schema, and he can have data from different domains by sending several queries, which can bring more convenience to the scientific researchers. Also this project has some general meaning, that is, based on the P2P technology, implementing data integration applications will be easier and the use of these applications will be more flexible.

This project of course has some flaws, and the final goal is to handle the inconsistency and to use no temporary storage at all. The author will keep doing the research and trying to find some solutions in the future.
BIBLIOGRAPHY AND REFERENCES


APPENDIX – INTRODUCTION TO JXTA PROTOCOLS

JXTA defines a series of XML messages, or protocols, for communication between peers. Peers use these protocols to discover one another, advertise and discover network resources, and communication and route messages.

There are six standard JXTA protocols:

- **Peer Discovery Protocol (PDP)** — used by peers to advertise their own resources (e.g., peers, peer groups, pipes, or services) and discover resources from other peers. Each peer resource is described and published using an advertisement.

- **Peer Information Protocol (PIP)** — used by peers to obtain status information (uptime, state, recent traffic, etc.) from other peers.

- **Peer Resolver Protocol (PRP)** — enables peers to send a generic query to one or more peers and receive a response (or multiple responses) to the query. Queries can be directed to all peers in a peer group or to specific peers within the group. Unlike PDP and PIP, which are used to query specific predefined information,
this protocol allows peer services to define and exchange any arbitrary information they need.

- **Pipe Binding Protocol (PBP)** — used by peers to establish a virtual communication channel, or pipe, between one or more peers. The PBP is used by a peer to bind two or more ends of the connection (pipe endpoints).

- **Endpoint Routing Protocol (ERP)** — used by peers to find routes (paths) to destination ports on other peers. Route information includes an ordered sequence of relay peer IDs that can be used to send a message to the destination. (For example, the message can be delivered by sending it to Peer A which relays it to Peer B which relays it to the final destination.)

- **Rendezvous Protocol (RVP)** — used by edge peers to resolve resources, propagate messages, and advertise local resources. Used by rendezvous peers to organize with other rendezvous peers, share the distributed hash table address space, and propagate messages in controlled fashion (message walkers).