Supporting Multimedia Applications With NetTraveler

Juan Pablo Carvajal Barreto M.E.
Advisor: Manuel Rodriguez Martinez Ph.D.

Computing and Information Sciences and Engineering Ph.D. Program
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681-5000
juanpc@ece.uprm.edu

Abstract

In recent years, several new architectures have been developed for supporting multimedia applications such as digital video and audio. However, quality of service (QoS) is an important element that is still missing from these architectures. This work deals with the development of a prototype for supporting multimedia applications using NetTraveler architecture. NetTraveler is a Database Middleware Systems, will be used to establish dynamic federations (similar to workgroups) of machines. On a given federation, local data sources will be able to interoperate with each other, and with data sources that might be located on remote geographic sites connected via a wide-area network. NetTraveler will be designed to cope with dynamic wide-area environments where data sources go off-line, change location, have limited power capabilities, and form ad-hoc federations of sites that work together to complete a given task.

1. INTRODUCTION

Multimedia applications such as digital video and audio often have stringent quality of service (QoS) requirements. For a network to deliver performance guarantees it has to make resource reservation and excise network control. In the past several years, there have been much discussion and research and much new architecture have been proposed. In our case, we apply the NetTraveler architecture how a solution for this problem.

The emergence of wide-area networks such as the Internet has provided users with access to vast amounts of rich data sets that are located on data sources distributed over these networks. For the past twenty years, researchers in the area of Distributed Database Systems have concentrated on the problems of heterogeneous data integration [8], distributed query processing [6], distributed transaction processing [4], and data dissemination systems [2]. Database Middleware Systems were developed to integrate heterogeneous collections of data sources distributed over a computer network. Typically, Database Middleware Systems follow an architecture centered on a *data integration server*, which provides client applications with a uniform view and a uniform access mechanism to the data available in each source. Database gateways [3] and Mediator systems [9] are the best known examples of database middleware.

The next wave of information systems to be deployed over wide-area networks will feature data sources residing on handheld devices (e.g. Personal Digital Assistants-PDAs), mobile laptop computers, embedded systems, sensors and other types of small-sized devices. In this scenario, we will find data sources that contain very diverse data sets such as MP3 files, movies, Animated GIF images, etc. To efficiently harvest these data sets, it will be necessary to integrate them into a coherent information system, which must also incorporate the more traditional data sources (e.g. relational databases) that are residing on enterprise servers. However, the existing database middleware solutions for data integration are inadequate for these new environments because they are designed for rather static systems based on enterprise server machines that are always on-line on a fixed-location, with continuous power sources, administered by teams of professionals, and organized into slow-changing federations of sites.

For our purposes we will develop a strategy using NetTraveler, a database middleware system to integrate data sources residing on PDAs, mobile laptops, embedded devices and enterprise servers. NetTraveler is designed to cope with dynamic wide-area environments where data sources are detected, and form ad-hoc federations of sites that work together to complete a given task and then go about their business independently. In NetTraveler, handheld and mobile devices will be treated as *bonafide* data sources, capable of delivering content to other sites in the system such as Web servers and Data Warehouses. NetTraveler could be used to establish dynamic federations (similar to workgroups) of machines. On a given federation, local data sources will be able to interoperate with each other,

and with data sources that might be located on remote geographic sites connected via a wide-area network.

The central paradigm upon which NetTraveler is based is the idea that data and supporting environments set up on a given computing device should accompany the user as he/she moves around his/her circle of influence. Moreover, these devices should be able to adapt to changes in the environment to provide users with access to newly available resources. Existing computing systems are based on location-dependent configurations which tend to isolate resources and produce "Islands of Information" that force people to plan around their computer infrastructure. At times it can be very frustrating to have applications, databases, and other tools at work or school which are unavailable when we get home or go on a business trip. Likewise, it can be disappointing to have critical personal data, multi-media files or personal memorabilia (e.g. digital photos of a wedding) which cannot be accessed if we are not using our home computer. Thus, NetTraveler is designed so that your data and supporting environment can "follow you", and become available at different locations where you need access to it.

2. RESEARCH PLAN

The research plan seeks improve the service of multimedia application helped with the NetTraveler, a database middleware system to integrate diverse data sources on dynamic wide-area environments. Figure 1 depicts the kind of environment that motivates the need for NetTraveler. In these kinds of environments there is wide-spread variability in terms of data source capabilities, data models, networks connectivity, and application requirements. Data sources might be Web servers running on enterprise server machines, Database Management Systems running on high-end workstations, personal accounting data and multimedia files stored on laptops, or PDAs.

This project contains a series of activities that will generate original contributions in the areas of Distributed Database Systems, Middleware Systems, and Computer Networks. The major contributions of the proposed activities can be summarized as follows:

- Development of the NetTraveler middleware architecture to integrate data sources residing on handheld devices, laptops, workstations, sensors, embedded systems, and enterprise servers.
- Development of a secure, power-aware query execution model that provides Quality of Service (QoS) to the sources.
- Implementation of a working prototype for support multimedia applications whit NetTraveler. This prototype will be an open source system implemented in Java.

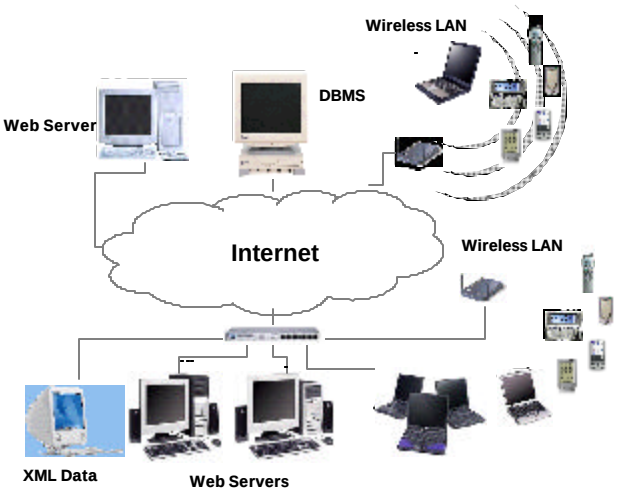

Figure 1. Wide-Area Dynamic Environment

2.1 NetTraveler Architecture

NetTraveler will be implemented using various technologies such as Virtual Private Network (VPN) software, Java, XML, servlets, Java Server pages, Database servers, and ASP. The purpose of each of the components is as follows:

- **Information Gateway** – provides access to other members of the system to the data sets maintained by a given source site. This element has a role equivalent to a database gateway or wrapper. This element will be run at the device hosting the data source, or at another computing unit that is physically close to it.
- **Data Broker** – works on behalf of a client site (which can be a data source site that become a client) to find the data and query processing capabilities necessary to solve a given query. This element has responsibilities such as query plan generation, query coordination, and acquisition of final results. The Data Broker can access data sources that lay outside its local execution environment. VPN technology will be used to enable this operation over wide-area environments.
- **Data Processing Server –** provides a commodity service for query execution. This is an optional element that will be most likely used in environments that require parallel processing capabilities, or which have many low-powered devices. The Data Processing Server should be

implemented on a multi-processor server or a cluster of workstations.

- **Registration Server –** this server is used to coordinate access and maintain bookkeeping necessary to detect the data sources. Register Servers located at cooperative local execution environments will work together to keep track of data sources and route data requests as appropriate.

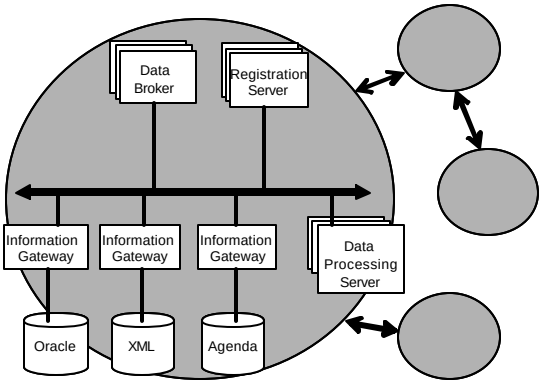

Figure 2. NetTraveler Architecture

Figure 2 depicts the different elements contained in the architecture of NetTraveler. The data sources are represented as cylinders, while each circle represent a *local execution environment* which can be composed of a combination 10/100/1000 Ethernet, and wireless networks (i.e. IEEE 802.11b). The fundamental organizational unit in NetTraveler is the **ad-hoc federation of peer sites**, which is a collection of computing devices that have agreed (or are configured) to work together as a group to exchange data, and share computational resources. A federation can span one ore more local execution environments. Federations are ad-hoc because they can be formed or dissolved over time based on the decisions taken by the member data sources. As a special case, a permanent federation is one that is never dissolved and corresponds to the type of federations assumed by existing middleware solutions. NetTraveler exhibits Peer-to-Peer behavior in the sense that a given data source site can be serving data to one or more sites, while it retrieves data from others (not necessarily at the same time).

We will implement a prototype for NetTraveler using the Java programming language. The Registration Server will be implemented as servlets, while the rest of the components will be implemented as stand-alone multi-threaded Java servers. We will assess if more components are needed or if some should be eliminated. For example, the functions performed by the Registration Server might be given to the Data Broker. The design choice was not to do that in order to keep the implementation of the Data Broker simple and manageable. However, performance considerations could change that.

2.2 Power-Aware Query Execution and Performance Guarantees

Most execution engines found in Database Middleware Systems are based on iterators [5]. In this model, the plan to solve a query is represented as a tree of operators, where each operator is represented by a node in the tree. Typically, leaf nodes represent operators that extract the data from the data sources, while the root represents the operator that delivers the results to the client. The intermediate nodes represent operation such as selection, projections, join, sorting and aggregation. Typically, an edge in the tree represents a network connection between the sites that will execute the operators in adjacent nodes. Most execution engines are "connection-driven" meaning that the state of query execution on a given edge depends on the state of the network connection. A failure in this connection often requires the query to be re-started, losing most if not all of the previous work.

We propose to develop a query execution engine that can provide Quality-of-Service (QoS) semantics to the data sites involved in query execution. In this context, QoS refers to guarantees in terms of resource sharing, reliability, performance and security. In our approach, the peer site submitting a query request will be able to indicate the kind of services it expects. For example, a PDA using 3G technology might submit a query and then ask to be "called back" when the results are ready, thus avoiding the need of wasting power and air time waiting for results that might take minutes to get produced. Similarly, a laptop user might ask for the query results to be sent to his computer at work. The process to generate query plans and execute them will be carried out in stages, where some query planning (optimization) follows execution, followed by more planning and so on. Thus, the system will be able to make gradual decision on how to solve the query and leverage on current system conditions. This differs from existing optimization frameworks where a query plan is generated a priori and latter executed, with somewhat limited possibilities for change [10] or adaptation [1].

In the execution model for NetTraveler, the Data Broker will come up with an initial plan that to carry out a first set of the operations needed to solve the query. The Data Broker will then find the data sources and other sites (e.g. Data Processing Server) that will cooperate in the process. The Data Broker

will use the data source profiles to determine which source to ask for help. This will permit finding a good set of sites. For example, PDAs might not be asked to participate in a complex join operation, but a laptop computer might get the call. The plan generated by the Data Broker will specify the actions to be taken, and the options to deliver the results to the next consumer of the data. For example, a plan might instruct a sensor to aggregate its current readings, and then send them to either one of two Data Processing Servers where the results will be further processed. Thus, the execution engine for NetTraveler will be "site-driven", allowing connections between sites sharing an edge in the query plan to come and go, to accommodate requirements such a power consumption and network connectivity costs. As the execution of the query unfolds, the Data Broker coordinating the operation will monitor progress and throw in the next set of actions to be taken. This process will continue until the query is finally solved.

We will implement the NetTraveler execution engine in Java, and will used XML to exchange the plans and control information between per sites. Plan generation algorithms will be developed, and a query planner will be implemented as part of the Data Broker. The execution engine will also feature code shipping capabilities like MOCHA[7].

3. SUMMARY

We will implementation the prototype for support multimedia applications whit NetTraveler. In our project, we development the following steps:

- Build client applications capable of receive and send the information from and to application server.
- Establish the communication protocol between client and server application.
- Develop the server application entitled of process the client information request and send the results to the client.
- Define the communication protocol between application server and the information sources.
- Design and build a Database Middleware System to integrate the information from data sources distributed over a computer network.

Notice that Database Middleware differs from Network Middleware such as CORBA, RMI, .NET and RPC since the latter are used as an infrastructure layer to provide applications access to the network. Database Middleware is at a higher layer, and can leverage on the Network Middleware for connectivity purposes.

References

[1] Avnur, R. and Hellerstein, J.M. Eddies: Continuously Adaptive Query Processing. in Proc. of ACM SIGMOD Conference, Dallas, TX, USA, 2002, 261-272.

[2] Bestavros, A. Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information Systems. in Proc. of ICDE Conference, New Orlean, LA, USA, 1996, 180-187.

[3] Oracle Corporation. Oracle Transparent Gateways. URL:http://www.oracle.com/gateways/html/transparent.html

[4] Delis, A. and Roussopoulos, N. Management of Updates in the Enhanced Client-Server DBMS. in Proc. 14th ICDCS Conferece, 1994, 326-334.

[5] Grafe, G. Query Evaluation Techniques for Large Databases. ACM Computer Surveys, 25 (2). 73-170.

[6] Haas, L.M., Kossmann, D., Wimmers, E.L. and Yans, J. Optimizing Queries Across Diverse Data Sources. in Proc. 23rd VLDB Conference, Athens, Greece, 1997, 276-285.

[7] Rodriguez-Martinez, M. and Roussopoulos, N. MOCHA: A Self-Extensible Middleware System for Distributed Data Sources. in Proc. ACM SIGMOD Conference, Dallas, Texas, USA, May 2000, 213-224.

[8] Roth, M.T. and Schwarz, P. Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources. in 23rd VLDB Conference, Athens, Greece, 1997.

[9] Tomasic, A., Rashid, L. and Valduriez, P. Scaling Heterogeneous Databases and the Design of DISCO. in Proc. 16th ICDCS Conference, Hong Kong, 1996.

[10] Urhan, T., Franklin, M.J. and Amsaleg, L. Cost Based Query Scrambling for Initial Delays. in Proc. ACM SIGMOD Conference, Seattle, Washington, USA, 1998, 130-141.