A REMOTE COMPUTING BROKER IN .NET

By

SENDHIL ANNAMALAI

B. E., Madurai Kamaraj University, India, 2000

A REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2003

Approved by:

Major Professor
Dr. Daniel Andresen
ABSTRACT

Global computing is the technique of applying resources from many computers in a network, at the same time, to a single problem. There is a lot of ongoing research in this field. .NET is an emerging technology that offers the basic infrastructure for integrating computers connected to the Internet into a distributed computational resource. This report aims at studying the feasibility of such a system and the design of a basic system for remote computing. This system is a .NET-based infrastructure for Internet computing. It is based on Internet software that is interoperable based on .NET enabled web technology. The architecture of the system requires participants to only a Web browser. The already existing architectures, techniques used to design the current system are discussed.
# TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................ III

ACKNOWLEDGEMENTS ............................................................................................... IV

DEDICATION .................................................................................................................. V

1. INTRODUCTION ........................................................................................................1

1.1 GLOBAL COMPUTING/GRID COMPUTING ......................................................... 1

1.2 JAVELIN .................................................................................................................. 2

   1.2.1 Advantages ................................................................................................. 2

1.3 GLOBUS ................................................................................................................. 3

1.4 REMOTE COMPUTING BROKER ....................................................................... 3

2. WEB SERVICES IN .NET ....................................................................................... 3

2.1 WHAT ARE WEB SERVICES? ............................................................................. 3

2.2 ADVANTAGE OF WEB SERVICES ................................................................. 4

2.3 HOW WEB SERVICE WORKS ........................................................................... 4

2.4 COMPARISON OF WEB SERVICES WITH WINSOCK AND DCOM ............ 5

2.5 UDDI (UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION) ...... 6

2.6 WEB REFERENCES .............................................................................................. 6

3. BACKGROUND ....................................................................................................... 8

3.1 DIME (DIRECT INTERNET MESSAGE ENCAPSULATION) ....................... 8

3.2 DIME RECORD FORMAT ................................................................................ 8
3.3 MIME (MULTIPURPOSE INTERNET MAIL EXTENSIONS) ......................... 9
3.4 DIFFERENCES BETWEEN DIME AND MIME ...................................... 9
3.5 WEB SERVICES ENHANCEMENTS ...................................................... 11
3.6 WEB SERVICE ENHANCEMENTS IN BROKER SERVICE ...................... 11
3.7 CONFIGURING WSE FOR DIME IN BROKER .................................... 12

4. IMPLEMENTATION ............................................................................. 13

4.1 ARCHITECTURE ............................................................................... 14

4.1.1 Detailed Architecture ..................................................................... 15

4.1.2 Class Diagram ................................................................................ 16

4.2 STORING HOST INFORMATION ......................................................... 17

4.3 DATA STRUCTURES USED ................................................................. 18

4.4 SCHEDULING ALGORITHM ............................................................... 19

4.5 REMOTE EXECUTION IN JAVELIN .................................................... 20

4.6 SECURITY ........................................................................................ 22

4.7 TECHNOLOGIES USED ................................................................. 23

4.8 LESSONS LEAR .............................................................................. 25

4.9 DIFFICULTIES FACED ..................................................................... 25

5. PERFORMANCE EVALUATION .......................................................... 26

6. CONCLUSION AND FUTURE WORK ............................................... 28

7. REFERENCES ...................................................................................... 29
LIST OF FIGURES

FIGURE 1: DISCOVERING A WEB SERVICE .............................................................. 7
FIGURE 2: DIME DATA RECORD FORMAT ............................................................. 9
FIGURE 3: WSE RUNTIME AND ASP.NET HANDLING INCOMING DIME

MESSAGE .................................................................................................................. 12
FIGURE 4: CONFIGURING WSE FOR DIME .......................................................... 13
FIGURE 5: ARCHITECTURE OF REMOTE COMPUTING BROKER ......................... 14
FIGURE 6: DETAILED ARCHITECTURE OF REMOTE COMPUTING BROKER ...... 15
FIGURE 7: REMOTE BROKER SERVICE – CLASS DIAGRAM ............................... 16
FIGURE 8: HOST INFORMATION DATA STRUCTURE .......................................... 19
FIGURE 9: STEPS IN REMOTE EXECUTION OF .NET BINARY ............................. 21
FIGURE 10: STEPS IN REMOTE EXECUTION OF APPLET ................................. 22
FIGURE 11: COMPUTATION TIME FOR NUMBER OF HOSTS-1 ............................. 26
FIGURE 12: COMPUTATION TIME FOR NUMBER OF HOSTS-2 ............................. 27
ACKNOWLEDGEMENTS

I sincerely thank Dr. Daniel Andresen, my major professor, for giving me encouragement, timely advice, guidance to complete this project. I also thank him for being flexible, adjusting during the course of this project. I would like to thank Dr. Gurdip Singh and Dr. Mitchel Neilsen for serving in my Committee

I would like to thank Ms. Delores Winfough for patiently helping me out in understanding the policies of the graduate school.

I thank Mr. Travis Bradshaw for helping and solving my day to-day problems with my experiments. I thank Mr. Yogesh Bhardwaj, Mr. Kiran Devaram, Mr. Srinivas Kolluri and Mr. Letchumanan Muthiah for giving me ideas and helping me in various stages of the project.

I would like to thank Mr. Ruchir Joshi for giving his system to do testing in the initial stages of the project and Mr. Shishir Joshi for helping me in formatting the report. I would like to thank my roommates for giving me moral support and all my friends who have been directly or indirectly helped me in the project.
DEDICATION

To my Parents
1. Introduction

The purpose of this project is to develop a .NET based Remote Computing environment. This will utilize the combined resources of many computers connected to the Internet, which forms a heterogeneous computing environment for running applications. The project shows the power that can be harnessed on Internet using unused CPU time. The first section will give introduction about global computing and already existing infrastructures Javelin and Globus. The second section will discuss about .NET and Web services in detail. The third section will discuss some background information needed for the project. The fourth section will discuss about the implementation. The fifth section will discuss the tests done to evaluate the solution. The final section will discuss the conclusion and make comments related to future work that could be done with.

1.1 Global Computing/Grid Computing

Global Computing is the cooperative use of geographically distributed resources unified to act as a single powerful computer. This approach is known by several names, such as Meta computing, scalable computing, global computing, Internet computing and more recently peer-to-peer or Grid computing.

Grid computing enables to compress time by expanding computing space. The first stage of grid computing, the cluster grid enables organizations to better utilize available computer resources. Every organization wants to better utilize computer resources, increase efficiency, solve problems more quickly, and get products to market faster. With enterprise grids and global grids, another benefit is the ability to increase access to resources and better control how resources are allocated, allowing for more efficient
deployment across heterogeneous environments, and enabling joint collaborative projects in dynamic, distributed, virtual organizations. Security remains an issue for the industry to better address. Security is less of an issue if it is within a department or within an open research community, where applications and data are not critical.

1.2 Javelin

Javelin is a Java-based infrastructure for parallel Internet computing. It utilizes the Internet’s vast growing computational capacity for very large, coarse grained parallel applications. It is designed for

- Obtaining the performance of massively parallel implementation,
- Provide a simple API allowing designers to focus on a recursive decomposition/composition of the parallelizable part of the computation.

1.2.1 Advantages

**Ease of use:** The key property of Javelin is ease of use, because it relies on the participation of thousands of users.

**Performance:** The performance benefits of massive parallelism are achieved without adulterating the application logic with interprocessor communication protocols. The resulting code performs well even when the set of processors change dynamically.

**Fault Tolerance:** The architecture scales to thousands of hosts and is fault tolerant, when hosts, in addition to failing, may dynamically disassociate from further participation in an ongoing computation.
1.3 Globus

The Globus Project is a research and development project focused on enabling the application of Grid concepts to scientific and engineering computing.

- Groups around the world are using the Globus Toolkit to build *Grids* and to develop *Grid applications*.

- *Globus Project research* areas include resource management, data management and access, application development environments, information services, and security.

- *Globus Project software development* has resulted in the Globus Toolkit, a set of services and software libraries to support Grids and Grid applications. The Toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability.

1.4 Remote Computing Broker

The system designed in this report demonstrates that a Grid computing system to perform parallel Internet computation is feasible in .NET. Our system performs remote computation of a .NET binary in a way, which is beneficial for the client.

2. Web Services in .NET

2.1 What are web services?

Web Services describe a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for
describing the services available and UDDI is used for listing what services are available. A Web service is a class that resides on a Web server and services requests, just like an ASP application. The difference is that the Web service doesn’t furnish HTML pages to the client. Instead, it behaves like a function: clients call the function by name, pass arguments (if there are any), and get back a result.

### 2.2 Advantage of Web Services

There is always a need to create some code with functionality that could be invoked by another machine. A few technologies evolved to assist developers, including DCOM from Microsoft, CORBA from the Object Management Group, and, later, RMI from Sun Microsystems. The disadvantage of these technologies is that the systems we are communicating with have to use the same platform.

In XML Web services, the method of invoking functionality on another machine is done through plain text using standard W3C protocols. This is a step backward, binary protocols can be much faster than an XML Web service. This disadvantage, though, is more than made up for by the ease with which different platforms can communicate with one another.

### 2.3 How Web service works

The client sends an XML document in a format called the Simple Object Access Protocol (SOAP) over HTTP. The server has a listener waiting for a SOAP packet. When it receives one, it opens it up, and if everything is in order, it executes the native code. It then takes the results from that code, packages it into another SOAP packet, and returns it to the client. SOAP supports platform and language interoperability through XML.
2.4 Comparison of Web Services with Winsock and DCOM

Winsock

Using Microsoft Winsock control, writing a basic client/server application that leveraged TCP is not a problem, but the efforts will be duplicated for each application server. It’s not an easy way of accomplishing distributed application programming because:

- Server code is applied specifically to the client that it is built for.
- Maintenance is a hassle.
- Firewalls can obstruct, if the transmission is not formatted in an HTML-like format.
- Distribution of the client application is a nightmare. Any time changes needed to be done on the client, the effort would have to be replicated on the server.

The need for a protocol, which is TCP compliant, is standard across all platforms, and overcomes the problems associated with historical ways of communicating with remote applications, is solved by SOAP. It improves the efficiency and decreases the complexity.

DCOM

Remote Procedure Call (RPC) is intended to allow an individual to run processes on distributed systems. The primary downfalls of RPC are security and configuration. The attacks that might be possible when allowing distributed RPC communication are Port mapping, Spoofing, Traffic sniffing and stealing and Denial-of-service attacks. With SOAP, developers can focus more on architecting and developing scalable solutions and concentrate less on network configuration issues. SOAP acts as a service-level provider, allowing you to expose only the methods that you want to expose over a secure, standard
protocol. Special ports do not need to be opened for access as they do with RPC. The developer is in complete control.

2.5 UDDI (Universal Description, Discovery and Integration)

One of the major building blocks required for successful Web services is UDDI. It creates a standard interoperable platform that enables companies and applications to quickly, easily, and dynamically find and use Web services over the Internet. UDDI also allows operational registries to be maintained for different purposes in different contexts. It is a cross-industry effort driven by major platform and software providers, as well as marketplace operators and e-business leaders within the OASIS standards consortium.

The UDDI project takes advantage of World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) standards such as Extensible Markup Language (XML), and HTTP and Domain Name System (DNS) protocols. Additionally, cross platform programming features are addressed by adopting early versions of the proposed Simple Object Access Protocol (SOAP) known as XML Protocol messaging. The UDDI protocol is the building block that will enable businesses to quickly, easily and dynamically find and transact with one another using their preferred applications.

2.6 Web References

The Web Reference files describe to the .NET Framework and to the developer much of the information required to use the Web service. There are three files to be aware of: the Discovery File (DISCO), the WSDL file, and the Reference Map. Figure-1 shows how to discover a Web Service.
Figure 1: Discovering a Web Service

The Discovery File

The Discovery (DISCO) File’s responsibility is to provide links to other libraries that further illustrate the Web service, such as the WSDL and ASMX files. The WSDL file is discussed in the following section, while the ASMX file is nothing more than the Microsoft-specific file extension of the compiled Web service or multiple Web services.

The WSDL File

The WSDL file contains information such as the schema, the message names and type, the SOAP ports, the operation names, the binding names, the binding transport and the service name.

The Reference Map

The Reference Map is a list of files that consist of the URLs referenced by the consumer application. This allows you to close an application after referencing a Web service and still maintain a reference. It is a late-bound reference because the application will not be
able to determine if the interface of the Web service changes after the initial reference until the consumer is executed, the Web service is called, and the client fails.

3. Background

3.1 DIME (Direct Internet Message Encapsulation)

Direct Internet Message Encapsulation is the method prescribed by WS-Attachments for sending and receiving SOAP messages with additional attachments, like binary files, XML fragments, and even other SOAP messages. Based on a specification submitted by Microsoft and IBM to the Internet Engineering Task Force (IETF), DIME is designed to encapsulate a SOAP message and its related attachments in a MIME-like way. DIME messages can be sent using standard transport protocols, like HTTP, TCP, and UDP. DIME supports streaming data. DIME can even be used without SOAP, although there is limited ability for DIME to describe the contents of its messages in this instance.

3.2 Dime Record Format

The DIME record has a Header part and Data part. The Header includes the version of DIME that is being used, type information about the data in the record's payload, an ID to uniquely identify each record, and support for adding any optional information that may be transmitted with a particular record. The type, ID, optional data, and payload for a record can all be of varying lengths, so the beginning of the record header is a fixed-length portion that includes the version, flags, and the lengths of the four variable-length portions of the record. The record format is shown in Figure 2.
3.3 MIME (Multipurpose Internet Mail Extensions)

MIME is an official Internet standard, which specifies how messages should be formatted so that they can be exchanged between different email systems. MIME is a very flexible format, permitting one to include virtually any type of file or document in an email message. Specifically, MIME messages can contain text, images, audio, video, or other application-specific data.

3.4 Differences between DIME and MIME

DIME is similar to MIME for Web Services, but there are some key differences between the two specifications. DIME is designed to be a fast and efficient protocol to parse. The length and type of attached data are defined in a few simple header fields. MIME is a widely used protocol developed to enable the transmission of attachments with e-mail via mail transport protocols like SMTP. Like DIME, the MIME multipart protocol (RFC
2387) also allows you to encapsulate multiple attachments of different media types in a single message.

In MIME special strings are used to separate multiple message attachments and allow us to include additional metadata in the message by the creation of custom message headers. Though these features make MIME very flexible, they also tend to make it a less efficient protocol. While potentially less flexible than MIME, DIME prescribes a simpler message format, which in turn provides for faster and more efficient message processing. For example, all of the data in a MIME message must be read and interpreted to determine simple things like the number of attachments included in the message. However, when using DIME, a parser can simply use the data in the record headers to quickly walk through and count the number of records in the message without having to read any record data. Also, in a DIME message, the only metadata included in the record headers specifies the length and encoding of the message header fields and payload. Any additional message information can be included in the attached SOAP messages.

The benefits of DIME over MIME fall into two main categories:

- **Performance:** DIME does not require any sort of encoding of binary data, and the DIME approach to specifying Data Record lengths, as opposed to specifying separator strings, makes parsing faster, and memory allocation more efficient.

- **Simplicity:** DIME is designed for simplicity whereas MIME is designed for flexibility. This means that it will be easier for tools for XML Web Services to be developed that support DIME, while decreasing the risk of interoperability issues.
MIME approach to sending SOAP messages with attachments is a technically plausible solution. However, supporting multiple specifications for achieving the same results is costly and potentially confusing.

### 3.5 Web Services Enhancements

The major players in XML Web Services (including Microsoft, IBM and Verisign) have proposed new specifications that will improve interoperability in the enterprise, in areas that are crucial for Web services. They include security, reliable messaging and sending attachments. To support these new proposed standards, Microsoft has released Web Services Enhancements (WSE) 1.0, which is made up of a set of classes that implement these new protocols, as well as a set of filters hosted by Microsoft ASP.NET that intercept incoming and outgoing SOAP messages and interpret or generate the SOAP headers to support the requested functionalities. WSE 1.0 provides support for WS-Security, Web Services Security Addendum, DIME, WS-Attachments, WS-Routing and WS-Referral.

### 3.6 Web Service Enhancements in Broker Service

WSE v1.0 supports DIME with SOAP attachments using ASP.NET, and reading and writing DIME messages to and from an I/O stream. The WSE runtime implements a DIME-compliant message parser that is able to translate the records of an incoming DIME message for SOAP messages with attachments. The parser extracts the primary SOAP message from the first DIME record and any encapsulated files from successive records as attachment objects. After being extracted from DIME, the primary SOAP message is then passed to the WSE message pipeline where the series input filters
evaluate the SOAP message for any other WSE-supported headers. Figure-3 shows how an incoming DIME message is handled by the WSE runtime and ASP.NET [3].

Figure 3: WSE runtime and ASP.NET handling incoming DIME message

In our system the client sends the binary file as a DIME attachment to the Broker Service. The following steps have to be followed before calling the Broker Service. The client creates an instance of the Broker Service. The Web Services Enhancements package allows it to create a Dime Attachment. Then the dime attachment is added with the SOAP request to the server. In the Broker Service the record contents are extracted from the binary stream

3.7 Configuring WSE for DIME in Broker

After installing WSE on the ASP.NET Web server, few additional DIME-specific configurations have to be made to the ASP.NET application in order to use attachments with DIME.
In the Broker service project

- References to the Microsoft.Web.Services.dll assembly need to be added to the project.
- Add a new type to the soapExtensionTypes element by adding a new add element in the Web.config file for the project as follows:

```xml
<configuration>
  <system.web>
    ...
    <webServices>
      <soapExtensionTypes>
           priority="1" group="0" />
      </soapExtensionTypes>
    </webServices>
  </system.web>
</configuration>
```

**Figure 4: Configuring WSE for DIME**

### 4. Implementation

The goal of this project is to find the feasibility of developing a global computing system in .NET and implement a part of it. Studying the systems that are already available is essential. Javelin has many broker systems, which coordinate between one another to get a task done. Such a system is used for huge parallel processing application. The scope of this project is to get the work done from the host with a single broker.
4.1 Architecture

The system has a three participating elements: broker, clients and hosts. The overall Architecture is given in Figure 4.

Client

A client is a process that seeks computing resources. Clients register with the broker, if they have any task to be done.

Host

A host is a process that offers computing resources at certain cost. Hosts register with the broker if they have resources that they can offer to the world.

Broker

A broker is a process that coordinates between the client and host according to the supply and demand for computing resources. The broker assigns tasks to hosts, which runs it and sends back the results through the broker. In our system, any computer in the Internet can become a client or host after registering with the broker.

Figure 5 : Architecture of Remote Computing Broker
4.1.1 Detailed Architecture

The Clients and Hosts can get access to the system by using the Web forms accessible to them thru a browser. The requests made are handled by VB.NET code behind files. For submission of files or for the Hosts to register the Broker Service is called. The Broker submits the job to the Host Service and gives back the result. The detailed system architecture is given in Figure-6.

Figure 6: Detailed Architecture of Remote Computing Broker
4.1.2 Class Diagram

This section gives the classes used in the system and the associations between the classes. For each ASPX form, there’s a code behind file, namely .aspx.vb, which consists of a class for that ASPX form. Along with this, there are the classes for the C# Web Service.

We have two web services.

**BrokerService.asmx:** It consists of all the web methods to perform the functionalities for the Users and Hosts.

**HostService.asmx:** This is the Web Service which all the hosts who want to participate in the system offer. Figure 7 shows the class diagram for the system.

![Class Diagram](image)

**Figure 7:** Remote Broker Service – Class Diagram
Broker Service can connect to the Oracle database using ADO.NET API for storing and retrieving data about host. This information will be useful when the host crashes. The Login class makes connection to the database directly without going through the web service.

4.2 Storing Host Information

A host that wants to participate in the system can send the information about it. This information will be useful for the broker to schedule a job. The Information that a host sends is

Bandwidth

The range of frequencies, expressed in Kilobits per second, that can pass over a given data transmission channel within a network. The bandwidth determines the rate at which information can be sent through a channel - the greater the bandwidth, the more information that can be sent in a given amount of time. It is measured in Kilo bits-per-second in our system. The bandwidth is taken into consideration when there are two hosts with the same configuration or same cost of computation. The job will be assigned on the host that has a better bandwidth.

Cost of Computation

If a machine wants to participate as a host then there should be an economic incentive for it. Each and every host has some cost for 1hr of computation. If the cost is less then, the clients which need the computation to be done with lesser cost will be scheduled.
**Configuration**

Configuration is the speed of the processor of the host. It is submitted as the next data.

The standard configuration of the system is 1 GHZ. The configuration varies from 1GHZ – 5 GHZ.

**Web Reference URL**

It specifies the literal string for the address of the Web service that the Host machine runs for getting the binary file from the broker.

### 4.3 Data Structures Used

All the information that the host sends and the Client Information when the client registers will be stored in the database for future use. They are also stored in data structures. This section discusses the about the data structures which are used in the system.

**Host Information**

The host Information is stored in a list. The host information is global to all the clients. The information stored is Host No, Bandwidth, Configuration, Cost, Web Reference URL, Host Busy and Busy Time. Each host has a queue in which there are clients, if they need to be serviced by it. The Client Information maintained in the queue is Client No, Input, Attachment, Estimated Time and Dead Line. The Host Information structure is shown in Figure 8.
Figure 8: Host Information Data Structure

**Cost List**

This is the list of hosts available sorted according to Cost of computation and bandwidth.

**Configuration List**

This is the list of hosts available sorted according to Configuration of the host and bandwidth.

**Host Status**

It stores the status of the Host if it is busy or not.

### 4.4 Scheduling Algorithm

The main functionality of the broker is to schedule the job that the client submits to it, in a host and get back the result. The client specifies if the computation needs to be done in a cost effective or Time Effective Way.
If condition is cost then,

- The status of the first host in the cost list is checked. If it is not busy then there are no hosts waiting in the queue of that host, so we can assign the host.
- If the host is busy then the Estimated time it will be busy is got. If the busy time combined with the estimated time of this job is less than or equal to the deadline, then it is assigned to this host.
- If the above conditions are not met then the next host is considered till the job is scheduled.

If the condition is Time then the same steps are followed except that the order of hosts is taken from the Configuration list.

**4.5 Remote Execution in Javelin**

The design of Javelin and our system is more or less similar. In this section we will look at the remote execution of applet in the case of Javelin and also remote execution of .NET binary. First let us look at the steps involved in the remote execution of .NET binary. Figure 9 shows the steps involved.

1. The client logs into the system and send the .NET binary to the broker.
2. The broker uses its scheduling algorithm and schedules the work in a host and forwards the binary to the host when it is free.
3. The host does the execution of the binary and sends the result back to the broker.
4. The broker forwards the result to the client.
Figure 10 shows the steps involved in the remote execution of an applet. The steps are:

1. The applet and an embedding HTML page is uploaded by the client on an HTTP server.
2. The client registers the corresponding URL with a broker.
3. The host registers with the broker its intention to execute tasks, and retrieves the URL of a task.
4. The host downloads the HTML page from the HTTP server, and executes its embedded applet.
5. The host stores the result at the server site. If communication between cooperating hosts is required, messages are stored at, and retrieved from, the server site.
6. The client retrieves the result.
Since it is a system that is going to be available on the Internet, the users should be authenticated properly. Only registered users are allowed to access the system. The Client or Host has to register with the broker first. Every time they want to use the system they have to login to the system. The IIS server has been configured to enable Secure Sockets Layer (SSL) for 128-bit encryption of data. The web browsers communicate with the broker over a secure connection. All the data being sent is encrypted at one side, transmitted, and decrypted at the other side prior to any processing.
4.7 Technologies Used

The technologies used in the project and the reasons why we have chosen them is given in the following section.

**Visual Studio.NET 1.1**

The front end Interface for the system is developed using ASP.NET. ASP pages provide standards-based database connectivity and the ability to customize content for different browsers. ASP also provides error-handling capabilities for Web-based applications. Sessions were required to be maintained for accessing authenticated pages. .NET Framework form based security supports authorization for users who are registered with the system to have access to the Broker Service. VB.NET is the code behind for the aspx pages in the server. The advantage of using VB.NET on the server is that the application running on the server is compiled and runs much faster. The Web Services which the Broker and Host offer have been coded in C#.

**Advantages of Visual Studio .NET**

- The .NET framework is programming-language-neutral. The .NET environment includes the Common Language Runtime or CLR and the Common Intermediate Language, also known as Microsoft Intermediate Language, abbreviated MSIL or IL, depending on the time of day. It makes it possible to combine C#, C++ or Visual Basic .NET (VB.NET) programs as if they were all written in the same language.

- Microsoft gives all the tools you needed to create "platform-independent, standards-based, language-neutral" .NET Web services.

- The API which it offers is vast
• All Visual Studio.NET Languages have been integrated in one IDE. It offers
dynamic help which makes code development easy

**IIS Server 5.0 and 6.0**

IIS Server is used as the web server to host the Web Applications and Web Services.
Since we are developing the System for .NET IIS was the obvious choice.

**WSE 1.0**

Web Services Enhancements (WSE) is a solution for developing secure, robust and
scalable Web services using Microsoft .NET. WSE v1.0 supports DIME with SOAP
attachments using ASP.NET, and reading and writing DIME messages to and from an
I/O stream.

**SOAP**

SOAP is an XML based lightweight protocol for exchange of information in a
decentralized, distributed environment. Data encoded in a SOAP message can be used in
a variety of situation such as Message Passing and Remote Procedure Calls. SOAP can
be potentially used in combination with a variety of other protocols. SOAP itself does not
define any programming model or implementation specific semantics. It instead defines a
simple mechanism by providing a modular packaging model and encoding mechanisms
for encoding data within modules

**Oracle**

Oracle is a robust relational database management system. It is available in on the CIS
System.
4.8 Lessons Learned

By designing and implementing the system in .NET, I learned about Web Application development and writing Web Services in .NET. Programming in different Languages for the same project was a new experience. I was able to compare the features of both languages. A better understanding about an already existing system and features of distributed computing was done. I learned how to manage and schedule jobs on many hosts. I was able to understand the primitives for Synchronization and manage threads. I was able to know about SOAP and how to use SOAP with attachments.

4.9 Difficulties Faced

Deciding on what technology to use for transferring the .NET binary took time. Winsock programming was considered initially. Then the option using Microsoft Message Queuing (MSMQ) was evaluated. Then we came across DIME, the protocol that is specially used for sending and receiving SOAP messages along with additional attachments, like binary files, XML fragments. There number of threads running in the broker will be equivalent to the number of jobs submitted. Since there are many threads there were some synchronization problems. Since .NET programming is in its beginning stage there is not much of help for beginners to program.
5. Performance Evaluation

This section explains about the testing done on the system. The system configuration of the systems used for testing is

**Aspweb:** 298 Mhz Dual Pentium 3 Processor, ½ gigabyte RAM, O/S: Windows 2000

**Aspnet**: 3 Ghz Pentium 4 Processor with 2 gigabyte RAM, O/S: Windows 2003

Testing of the system has been done on various loads. Different number of files has been submitted at the same time and the performance of the system has been noted. An API was written. It can be called with number of threads as the input and it creates that many numbers of threads. Each Thread submits a job to the broker. The size of the file submitted was 7KB. Up to 30 jobs were submitted at the same time. First the number of host was kept to be 1 and then the number of host was increased to 2. Stress testing was done and results are shown below. Figure 11 shows the performance of the system when the number of hosts is 1.

![Figure 11: Computation Time for Number of Hosts-1](image-url)
Figure 12 shows the performance of the system when the number of hosts is 2. We are not able to notice a significant increase in performance.

![Figure 12: Computation Time for Number of Hosts-2](image)

In both of the above cases we are able to note that performance increases linearly. In the case of single host it takes 900 seconds for 30 jobs, and in two hosts the performance has not drastically increased. The reason is that one of the systems (aspweb) has limited resources. We are calculating the time taken for 30 jobs as a whole. Though the other system has completed executing all the jobs assigned, the slower machine would not have completed its computation by then.

**Fault Tolerance:**

Fault Tolerance is the ability of a system to respond gracefully to an unexpected hardware or software failure. Fault Tolerance is taken care in two ways in our system. If the broker goes down the host information stored in the data structures is lost and the
system cannot resume operation until it has the details about the entire host. We have stored the information about the host in database and the broker can retrieve the values from it. The host may go down when it is performing a job and the broker will keep waiting forever if this is not taken care. The broker waits for a response from the host for estimated time. If it does not receive a response the job is scheduled in another host.

**Scalability:**

A scalable network system would be one that can start with just a few nodes but can easily expand to many nodes. The system we have developed is very scalable because, any number of hosts can be in the system. Hosts can participate by registering and unregistering with the broker. It is limited only by the capacity of broker to store the information about it.

### 6. Conclusion and Future work

In this report we have implemented a remote computing system in .NET. It demonstrates the feasibility of developing a distributed computing system. The capability to do distributed computation can be incorporated into the system, if the client is capable enough to decompose the jobs and then integrate the results it gets. By this way we can use it for distributed computations. Now the system has one broker that manages all the hosts. One can improve the system by adding more brokers. This will enhance the performance of the system and increase fault tolerance.
7. References


