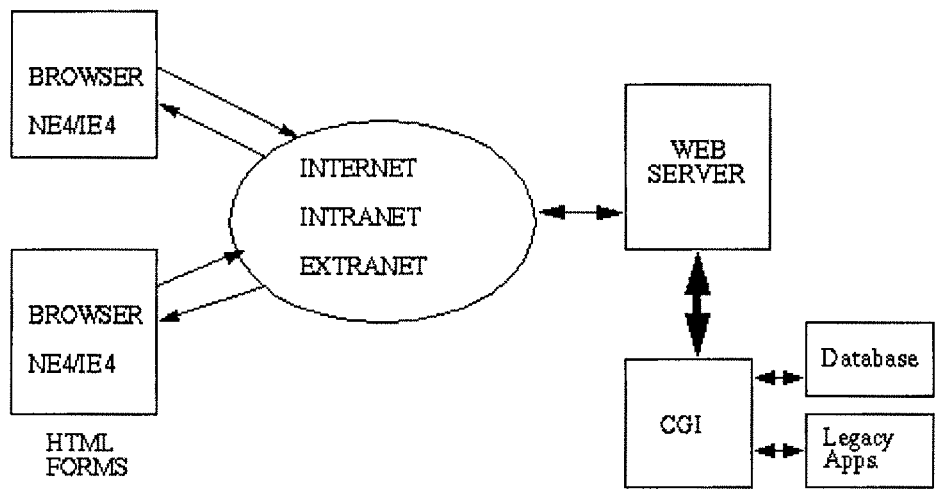
11 Contribution

Briefly, three tasks were accomplished to show an improved performance over the current HTTP/CGI based applications. Writing a dummy object that will register with the Application Server and will provide some fake services. The Java client that will communicate with the application server using RMI to ask for a service from the registered object was implemented. The service provided by the application server is a class that that writes to the file system.



HTTP/CGI based web applications are reported to have poor performance, because of different resons.

1. HTTP is connectionless.
2. CGI programs are costlier as a new process is started everytime.
3. The CGI's can't maintain a persistent connection to the database.
4. HTTP is stateless, and then the state management required for complicated applications becomes very tedious.

Steps considered

1. Write the middleware (App. Server)
2. Write a servant object, which will register with app. server and provide fake services.
3. Write the client side (Java Applet/or even Java Application).

Writing some sort of client<-> server communication over a TCP/IP socket. A simple multithreaded, server was written in Java which would listen on a TCP/IP port e.g. 4000 and then write a Java client which should open a socket connection with this server on port 4000 to demonstrate the communication. This communication is done over the network using RMI protocol.

b. The application server, its functionality would be much like a multithreaded echo server but instead of just echoing back the characters it provides some fake services like, write a bunch of files to the server

Separate Java classes were written to implement the actual service and then these classes should register themselves with the application server.

The client side in this case is a Java applet, but in this step it opens a socket connection with the application server, instead of invoking a CGI as any web application would do.

In response to the user actions in the GUI, the applet should send some data in a specific format back to the application server, now depending on the received data the application server should decide to call the appropriate service.

This shows that there is only one instance of JVM(Java Virtual Machine) running and as the app. server is multithread, it will serve multiple clients at the same time.

** This approach is faster. No more processes are started for every invocation of CGI, reduces load on the web server, reduces n/w traffic as the application server sends only raw data getting rid of HTML tags.

This application server could be enhanced with unlimited possibilities,

like state management, user data caching, load balancing, and middleware to some RDBMS.