UDC 681.3:519.7
Original scientific paper
Received: 01.07.2002.

EJB based component model for distributed load flow monitoring of multi-area power systems

Kannan Nithiyananthan and **V. Ramachandran**
Center for Professional Development Education, Anna University, Chennai 600025, INDIA
e-mail: knithiyananthan@hotmail.com

SUMMARY

The main objective of this paper is to develop component model architecture for load flow monitoring of multi - area power systems. A component which is based on a single-server serving multiple clients has been proposed enabling all neighboring power systems to have simultaneous access to the remote load flow server for obtaining continuous load flow solutions. An EJB (Enterprise Java Beans) based, distributed environment has been implemented in such a way that each power system client can access the remote load flow EJB server through JNDI (Java Naming and Directory Interface) naming service with its load flow data. The server computes the load flow and it provides the continuous automated load flow solutions to all the registered power system clients. Load flow EJB server inherently creates a new thread of control for every client request and hence a complete component based on distributed environment can be achieved.

***Key words**: client-server model, distributed computing, EJB, load flow monitoring, tunneling.*

1. INTRODUCTION

The Power System load flow solution by conventional client-server architecture is complicated, memory management is difficult, source code is bulky, and exception-handling mechanism is not so easy. In the conventional power system operation and control, it is assumed that the information required for monitoring and controlling of power systems is centrally available and all computations are to be done sequentially at a single location [1]. With respect to sequential computation, the server has to be loaded every time for each client's request and the time taken to deliver the load flow solution is also comparatively high [2, 3].

This paper outlines a new approach to develop a solution for load flow analysis by the way of distributed computing. An EJB based component model overcomes the difficulties associated with sequential computation and it is easy to implement. Enterprise load flow component models are pluggable, reusable and can simplify complexity in the areas of synchronization, scalability, load flow monitoring integrity, networking and distributed object frameworks. A load flow bean can be developed once and it can be deployed on multiple platforms without recompilation or source code modification.

EJB uses built in security facilities for authentication, authorization and for secure communication. Hence the distributed load flow monitoring through the EJB load flow server is safe and secure.

2. THE PROPOSED EJB ARCHITECTURE

In this proposed model, each power system client can access the remote load flow EJB server through the servlets based on data object serialization [3]. The load flow Server in turn computes and disseminates load flow solutions to all the power system clients simultaneously for every specific period of time based on the client's requirement. The various entities of proposed EJB model are a load flow EJB server, the load flow EJB container that runs with in the server, home objects,

remote EJBObjects, and load flow bean that runs within EJB containers, power system clients and JNDI Services. The relationship between the above entities of the proposed EJB model is shown in Figure 1.

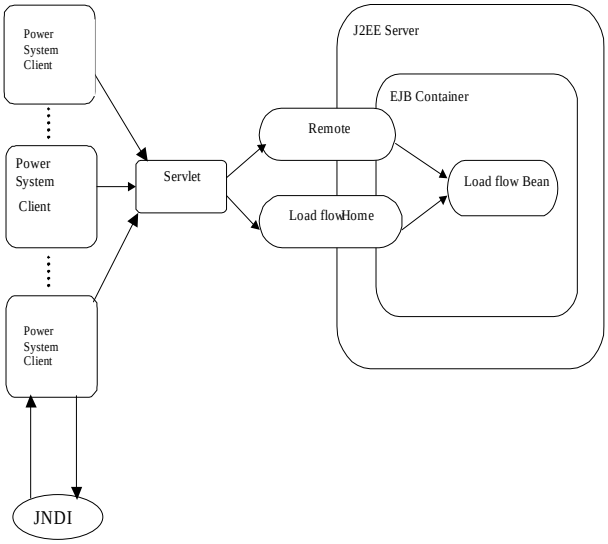

Fig. 1 Component model for Load flow Monitoring

2.1 Load flow EJB Server

The load flow EJB server provides an organized framework or execution environment in which EJB container can run. It makes available system services for multiprocessing, load balancing and device access for EJB containers. The J2EE platform enables a multi -tiered, distributed application model, the ability to reuse components, a unified security model, and flexible transaction control. Power system clients simultaneously access the load flow EJB server through JNDI naming service. Based on the client's requirement, the server communicates with the remote client, fetches the present load flow data, computes load flow solution and provides the result to that specific client. The process is simultaneously done for every registered client by generating a separate thread of control. The purpose of loading the server with load flow computational skill is that any further modification to the computation methodology would reflect appropriate results for all the remote clients.

2.2 Load flow EJB container

The load flow bean component lies inside the load flow EJB container. The load flow EJB container provides services such as load flow calculation management, versioning, scalability, mobility, persistence, and security to the components it contains. Since the EJB container handles all of these functions, development of load flow component is made easy and EJB container contains the load flow Bean, Home and Remote interfaces.

2.3 The load flow Bean component

The load flow Bean executes with in a load flow EJB container, which in turn executes within an EJB Server. A load flow EJB component is the type of EJB class, which is most likely to be the load flow computation logic. All the other classes in the EJB system support either client access or provide services to EJB component classes. In this proposed architecture, the load flow bean is a stateless session bean. A stateless load flow bean does not maintain a conversional state for a particular power system client. When a power system client invokes the method of a load flow bean, the bean's instance variable may remain in a particular state, but only for the duration of the invocation. When the operation of the method is over, the state is no longer retained. Stateless session beans can support multiple clients and it can offer better scalability for load flow monitoring application for large power system clients.

2.4 Load flow EJB Home and Remote interface

Load flow EJB component which is the Home Interface, defines the methods for creating, initializing and destroying the instances of the server. The home interface is a contract between a load flow EJB component class and its container, which defines construction, destruction, and looks up the EJB instances. A load flow EJB home interface extends the interface javax.ejb.EJBHome, which defines base-level functionality for a home interface and all methods in this interface must be RMI-compatible. The Remote Interface lists the load flow method available in the bean. This EJB object is the client's view of the enterprise bean and implements the remote interface. While the load flow bean defines the remote interface, the container generates the implementation code for the corresponding EJB object. Each time the power system client invokes the EJB object's method, the EJB container handles the request before delegating it to the load flow bean.

2.5 Power System Clients

Power system clients locate the specific load flow EJB container that contains the load flow bean through the Java Naming and Directory Interface (JNDI) service. They make use of the EJB container to invoke the load flow bean and get a reference to an EJBObject instance. When the client invokes a method, the

EJBObject instance receives the request and delegates it to the corresponding bean instance and also provides necessary wrapping functionality. In this proposed method, load flow monitoring by each client is achieved through an applet to servlet to EJB communication for every specific period of time. The applet to servlet communication is enabled via HTTP tunneling. The power system client applet opens a URL connection to the servlet, passing it the name and the port number of the remote host that can upload the load flow data to the EJB load flow server. The proxy servlet transforms objects into a stream of bytes (Byte Array Output Stream) which is sent as a load flow response, and reconstituted at the power system client at regular time intervals.

2.6 Java Naming and Directory Interface Service

Java Naming and Directory Interface (JNDI) adds value to load flow bean deployment by providing standard interface for power system clients. Naming service in JNDI is the entity that associates names with objects and it provides a facility to find an object based on its name. Directory service in JNDI is the naming service that has been extended and enhanced to provide directory object operations for manipulating attributes. JNDI is a unified system to access all sorts of directory service information such as security credentials, machine configurations and network address of the power system clients. JNDI is extensible and it insulates the application from protocols and from implementation details. The greatest use of JNDI service is to locate load flow beans' home objects. To acquire the reference of the load flow home object, the environment properties files or system files provide the details of the JNDI service provider to be used in the load flow bean deployment. The client then uses the environment properties employed in creating the initial context factory to look up the load flow object stored in the directory.

3. EJB DATA FLOW MODEL

EJB data flow model is formed according to the Ref. [4]. Power system clients use the Java Naming and Directory Interface to lookup load flow objects over a network. A remote power system client accesses the load flow bean through its remote and home interfaces. When the power system client performs a JNDI lookup for a home object, EJB container might use JNDI to return a RMI remote stub. The Remote stub is a proxy for the load flow home object, which is located elsewhere in the network and once the power system client has a stub, it can invoke a load flow method on the home object through the remote stub object. The EJB object that implements the remote and home interfaces is accessible from a client (the servlet acts as an EJB client) through the standard RMI APIs. The servlet communicates with the remote EJB container thus requesting that the load flow method and then it communicates with the load flow bean as shown in Figure 2. The load flow EJB container executes the load flow bean and sends the load flow solution back to each power system client via the servlet at specific intervals.

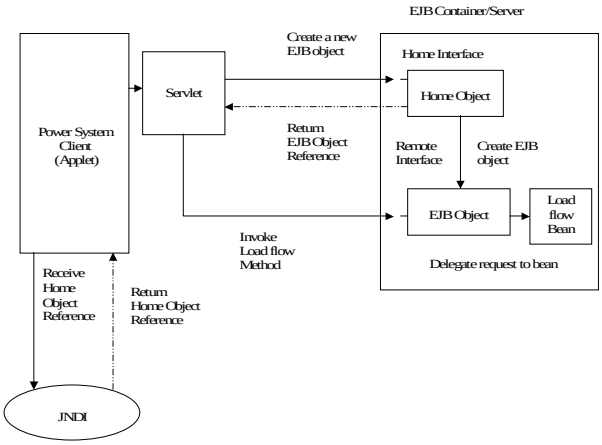

Fig. 2 Invoking a load flow method on the remote EJB Server

4. LOAD FLOW BEAN LIFE CYCLE

The following steps describe the life cycle of a load flow bean instance as shown in Figure 3 [4]:

- A stateless load flow bean instance's life starts when the container invokes newInstance() on the load flow bean class to create a new instance and the container calls setSession-Context() followed by ejbCreate() on the instance. The container can perform the instance creation at any time and there is no relationship to a client's invocation of the create() method.
- The session bean instance is now ready to delegate a load flow method call from any power system client.
- When the load flow bean is no longer needed, container invokes the ejb Remove(). This ends the life of the stateless load flow bean instance.

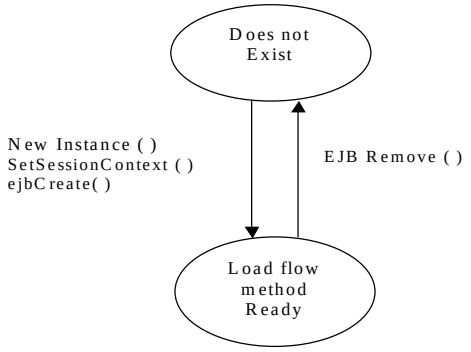

Fig. 3 Life cycle of EJB load flow Bean

5. DEPLOYING PROCEDURE TO BUILD THE PROPOSED LOAD FLOW MONITORING APPLICATION

In order to deploy the EJB component into the load flow server, the following steps are to be followed:

1. Start the J2EE deploytool window and select the new application.
2. Choose the corresponding enterprise archive file and type the application display name.
3. Start the New Enterprise wizard to package the load flow bean and type the JAR display name.
4. Add the loadflowint.class, loadflowHome.class and loadflow EJB.class to JAR dialog box.
5. In the General dialog box, choose the bean type as stateless session bean and choose appropriate interfaces in the Enterprise Bean class and enter the name of the Enterprise bean.
6. Open the deploy wizard and give the full path of client's jar file name which contains the stub classes and it will enable remote access to the load flow bean.
7. Enter the JNDI name and WAR context root and deploy the load flow bean.

6. RESULT

A complete component model for load flow monitoring by EJB based n-tier architecture has been implemented in Windows NT based HP workstations connected in an Ethernet LAN. The results are shown in a client applet as given in Figure 4.

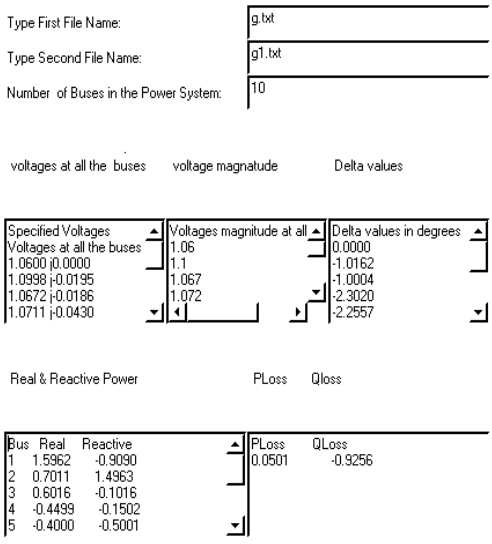

Fig. 4 Power System Load flow monitoring Applet

The above applet shows the load flow solution for a specific 10-bus power system client. When each power system client applet is loaded, it invokes the servlet via http tunneling and in turn accesses the load flow bean by its JNDI name, Web Context root. The EJB container runs the load flow bean automatically and the load flow solution is calculated and the result is sent back to the respective power system client. Using this approach, different power system clients can monitor continuous updated load flow solutions at regular time intervals.

7. CONCLUSION

An effective distributed component model has been developed to monitor the load flow of multi area power systems. This model attempts to overcome the overheads associated with sequential power system load flow computation. Although client-server architecture for load flow solution is well established, this paper emphasizes a unique methodology based on Enterprise Java Beans to serve a large number of clients in a distributed power system environment, across various platforms based on communication between virtual machines. A practical implementation of this approach suggested in this paper was assessed based on 6, 9, 10 and 13 bus sample systems. Accordingly the proposed model can be implemented for a large power system network spread over a large geographical area.

8. REFERENCES

[1] G. Bandyopandhyay, I. Senguptha and T.N. Saha, Use of client-server model in power system load flow computation, *IE(I) Journal-Electrical*, Vol. 79, pp. 199-203, 1999 .

[2] B. Qiu and H.B. Gooi, Web-based SCADA display systems (WSDS) for access via Internet, *IEEE Transactions on Power Systems*, Vol. 15, No. 2, 2000.

[3] G.P. Azevedo, B. Feijo and M. Costa, Control centers evolve with agent technology, *IEEE Transactions on Computer Applications in Power*, pp. 48-53, 2000.

[4] E. Roman, *Mastering Enterprise Java Beans and the Java 2 Platform, Enterprise Edition*, Wiley Computer Publishing, John Wiley & Sons, 2000.

[5] Enterprise Java Beans™ Specifications, version 1.0, http://java.sun.com/products/ejb/docs10.html.

[6] Enterprise Java Beans - Part 2, http://members.tripod.com/gsraj/ejb/chapter/ejb-2.html.

EJB ZASNOVANI KOMPONENTNI MODEL ZA MONITORING DISTRIBUIRANOG TOKA PUNJENJA VIŠEPOVRŠINSKIH ENERGETSKIH SUSTAVA

SAŽETAK

Osnovni cilj ovog rada je razviti arhitekturu komponentnog modela za monitoring toka punjenja višepovršinskih energetskih sustava. Predlaže se komponenta bazirana na jednom serveru koji opslužuje više klijenata omogućavajući svim susjednim energetskim sustavima simultani pristup udaljenom serveru toka punjenja za dobivanje stalnih rješenja toka punjenja. Zasnovana na EJB-u (Enterprise Java Beans), distribuirana okolina napravljena je na način da svaki klijent energetskog sustava može pristupiti udaljenom EJB serveru toka punjenja preko JNDI (Java Naming and Directory Interface) imeničkog servisa s podacima o toku punjenja. Server izračunava tok punjenja i daje stalna automatizirana rješenja toka punjenja svim registriranim klijentima energetskih sustava. EJB server toka punjenja omogućava kontrolu zahtjeva svakog klijenta, stoga se može postići potpuna komponenta zasnovana na distribuiranoj okolini.

***Ključne riječi**: klijent-server model, distribuirano proračunavanje, EJB, monitoring toka punjenja, prijenos podataka.*