Component Model for Multi-area Power Systems On-line Dynamic Security Analysis

K. Nithiyananthan and V. Ramachandran

***Abstract*—The main objective of this paper is to develop a component model architecture to construct a distributed environment through which the contingency selection for voltage security analysis, line overloads and the reactive power limit violations of multiple power systems can be monitored and controlled. A component, which is based on single-server serving multiple clients, has been proposed. It enables all neighboring power systems can have simultaneous access to the remote contingency server at any time, with their respective data and is able to get the contingency ranking based on their performance indices. An EJB (Enterprise Java Beans) based distributed environment has been implemented in such a way that each power system client can access the remote contingency EJB server through JNDI (Java Naming and Directory Interface) naming service with its system data. The server conducts the contingency analysis and it provides the continuous automated critical contingency ranking list based on performance index to all the registered power system clients. EJB server inherently creates a new thread of control for every client request and hence a complete component based distributed environment has been achieved.**

***Index Terms*—Contingency analysis, distributed computing, EJB, client-server model, tunneling.**

I. Introduction

THE KEY aspect of the new development in power system on-line control is the enhancement of the security of the power system in order to maintain a high reliability of electric power supply. Security of the power system requires the proper integration of both automatic and manual control functions. Both the steady state and dynamic state emergency conditions of a power system operating problem have to be characterized by keeping the system operating optimally based on voltage, real and reactive power. During the normal state in order to forestall an emergency and reassure that all loads to be satisfied, it is essential to conduct security analysis to know whether the system is vulnerable to electrical disturbance. The first function of the security analysis is to determine whether the normal system is secure or insecure by contingency evaluation and the second function is to determine whether the preventive action should be taken when the system is insecure.

The power system on-line contingency analysis 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 contingency ranking list is also comparatively high [2], [3]. This paper outlines a new approach to develop a solution for on-line contingency 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 contingency component models are pluggable, reusable and solve complexity such as in the area of synchronization, scalability, data security and integrity, networking and distributed object frameworks.

Modularity and reusability are the primary advantages of a component model. A component model is proposed to provide a clear specification of the inputs needed from other components of the power systems. The proposed multi -component application allows for the integration of components implement in different platforms. A component can be easily moved to a remote location without recompiling other parts of the application, in particular other components that interact with it directly. Hence component model is proposed even though it has low execution speed.

A Load flow bean and contingency bean can be developed once and they can be deployed on multiple platforms without recompilation or source code modification. Although the contingency code is easy to replicate at the users end, the new or updated contingency logic can be implemented very easily without any recompilation. All the Power System clients in the distributed environment are able to use the updated contingency logic with out any interruption.

EJB uses built in security facilities for authentication, authorization and for secure communication. Hence the distributed on-line dynamic security through the contingency EJB server is safe and secure.

II. The Proposed EJB Architecture

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

Manuscript received December 11, 2002; revised July 2, 2003.

K. Nithiyananthan is with the Department of Electrical and Electronics Engineering, College of Engineering, Guindy, Anna University, Chennai–25, India (e-mail: knithiyananthan@hotmail.com).

V. Ramachandran is with the Department of Computer Science and Engineering, College of Engineering, Guindy, Anna University, Chennai–25, India (e-mail: rama@annauniv.edu).

Publisher Item Identifier S 1682-0053(03)0192

EJB objects, load flow bean and contingency bean that run within EJB containers, power system clients and JNDI services. The relationship between the above entities of the proposed EJB model is shown in Fig. 1.

A. Contingency EJB Server

The contingency EJB server provides an organized framework or execution environment in which the 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 contingency EJB server through JNDI naming service. Based on the client's requirement, the server communicates with the remote client, fetches the present contingency data and computes the load flow after removal of the faulty line. Output of the load flow is used by the server to compute the performance Index and provides the contingency ranking 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 at all the remote clients.

B. Contingency EJB Container

The load flow bean and contingency bean components are stored inside the Contingency EJB container [4]. The contingency EJB container provides services such as contingency 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 and contingency component is made easy. EJB container contains the load flow bean, home and remote interfaces as the load flow component and the contingency bean, Home and Remote Interfaces as the contingency component.

C. The Load Flow and Contingency Components

The load flow bean executes with in a contingency 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 load flow computation logic. contingency component is also the type of EJB class, which is likely to be contingency logic. All the other classes in the EJB system support either client access or provide services to EJB component classes. In this proposed architecture, load flow bean and contingency bean are stateless session beans. A stateless load flow bean and contingency 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 contain a state, but only for the duration of the invocation. When the method is finished, the state is no longer retained. Stateless session beans can support multiple clients and it can offer better scalability for dynamic security analysis for large power system clients.

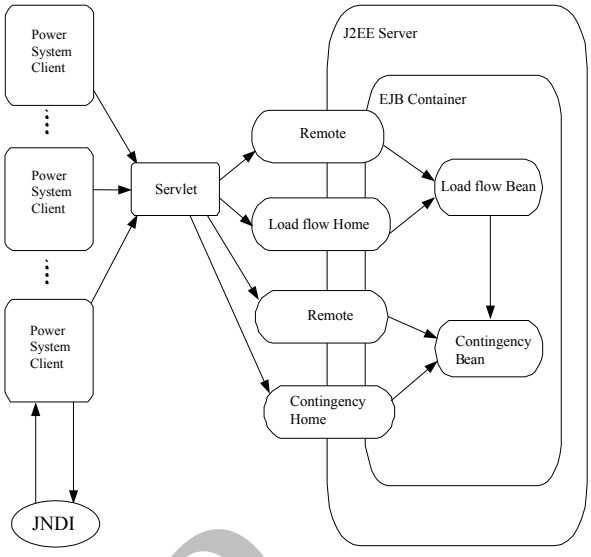

Fig. 1. Component model for dynamic security.

D. EJB Home and Remote Interface

Load flow EJB component and contingency component, which have the Home Interface that defines the methods for creating, initializing and destroying the instances of the server. The home interface is a contract between an EJB components and its container, which defines construction, destruction, and looks up of EJB instances. An 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 respective remote Interface lists the load flow method available in the load flow bean and as well as methods available in the contingency bean. The EJB object is the client's view of the enterprise bean and implements the remote interface. While the load flow bean and contingency bean define 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.

E. Power System Clients

Power system clients locate the specific contingency EJB container that contains the load flow bean and contingency bean through the Java Naming and Directory Interface (JNDI) service. They make use of the EJB container to invoke load flow bean and contingency bean to 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, dynamic security analysis by each client is achieved through a servlet to EJB communication for every specific period of time and applet to servlet communication is enabled via HTTP tunneling. The proxy servlet transforms objects into a stream of bytes (Byte Array Output Stream) that are sent as contingency request and reconstituted at the power system client at specific interval of time. Power system client applet opens a URL connection to the servlet, passing it by name, port number of the remote host and it can upload the contingency data to the EJB contingency server.

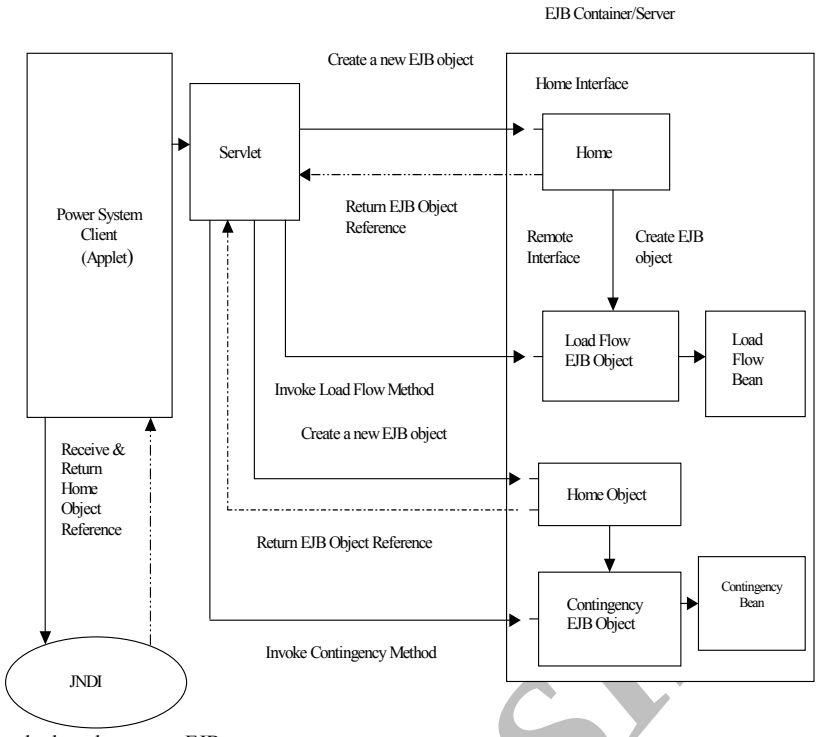

Fig. 2. Invoking a load flow method on the remote EJB server.

F. Java Naming and Directory Interface Service

Java Naming and Directory Interface (JNDI) adds value to load flow bean, contingency bean deployment by providing standard interface for power system clients [5]. Naming service in JNDI is the entity that associates names with objects and it provides a facility to find an object based on name. Directory service in JNDI is a 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 bean and contingency bean home objects. To acquire the reference of the load flow home object declaratively specifying environment properties files or system files which detail the JNDI service provider used in the load flow bean, contingency bean deployment respectively. The client then uses the environment properties employed in creating the initial context factory to lookup the load flow and contingency objects stored in the directory.

III. EJB Data Flow Model

Power system clients use the Java Naming and Directory Interface to lookup load flow and contingency objects over a network [6]. A remote power system client first 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 are accessible from a client through the standard RMI APIs. It communicates with the remote EJB container thus requesting that the load flow method and then communicates with the load flow bean and with the contingency data as shown in Fig. 2. Contingency EJB container executes the load flow bean and sends the load flow solution to the contingency bean, which in turn computes the performance index for each removable contingency's. Critical contingencies are ranked, based on their performance index are sent back to the each power system client via a servlet at specific intervals.

IV. Load Flow and Contingency Bean Life Cycle

The following steps describe the life cycle of a load flow bean and contingency bean instance as shown in Fig. 3.

- A stateless load flow bean and contingency 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 the load flow and contingency method calls from any power system client.
- When the container no longer needs the instance the container invokes ejbRemove() on it.

This ends the life of the stateless load flow bean and contingency bean instance.

V. Deploying Procedures to Build Proposed On-linr Dynamic Security Analysis Application

In order to deploy the EJB component into the contingency server the following steps are to be followed.

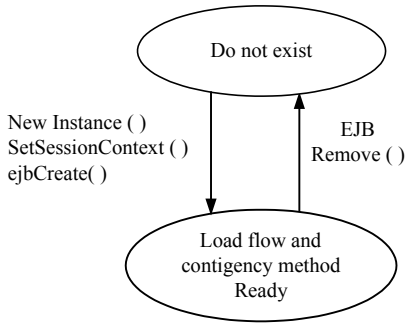

Fig. 3. Life cycle of EJB load flow bean and contingency bean.

1. Start the deploy tool 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 loadflowEJB.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. Enter the name of the Enterprise bean.
6. Repeat steps 3, 4 and 5 and create the contingency component, which contains the contingency bean.
7. Open the deploy wizard and give the full path name of client's jar file name which contains the stub classes and it will enable remote access to the load flow bean and contingency bean.
8. Enter the JNDI name and WAR context root and deploy the contingency application.

VI. RESULTS

A complete component model for on–line dynamic security analysis 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 Fig. 4.

The above applet shows the contingency ranking for a specific 8-bus power system client and the load flow result with a line is removed. When each power system client applet is loaded, it invokes the servlet via http tunneling and in turn the servlet accesses the load flow bean by its JNDI name, Web Context root. EJB container runs the load flow bean automatically and sends the load flow solution as input to the contingency bean. In turn the contingency bean calculates the performance index based on voltages, real and reactive power. Finally a critical contingency list has been sent back to respective power system client. The client then uses the environment properties employed in creating the initial context factory to look up the load flow and contingency objects stored in the directory.

VII. CONCLUSION

An effective RMI based distributed model has been developed to do security analysis of multi-area power systems. It has been tried out to overcome the overheads associated with sequential power system contingency evaluation through this model. Although a 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 large power system network spread over a large geographical area.

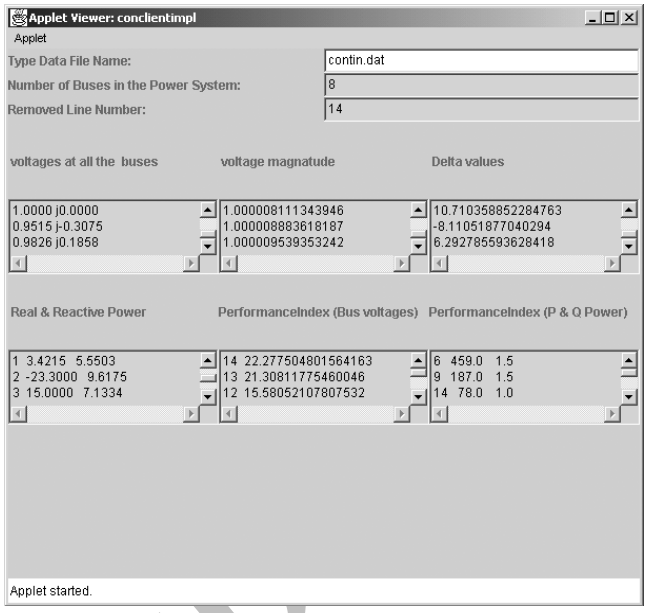

Fig. 4. Applet with contingency ranking list based on performance index.

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, no. 4, pp 199-203, Feb. 1999.

[2] B. Qiu and H. B. Gooi, "Web–based SCADA display systems (WSDS) for access via Internet," *IEEE Trans. on Power Systems*, vol. 15, no. 2, pp. 681-686, May 2000.

[3] G. P. Azevedo, B. Feijo, and M. Costa, "Control centers evolve with agent technology," *IEEE Trans. on Computer Applications in Power*, vol. 13, no. 3, pp 48-53, Jul. 2000.

[4] —, *Enterprise Java Beans TM Specifications Version 1.0*, http://java.sun.com/products /ejb/docs10.html.

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

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

K. Nithiyananthan received the B.E. degree in Electrical and Electronics Engineering and the M.E. degree in Power System Engineering from the Faculty of Engineering and Technology, Annamalai University, Chidambaram, India, in 1998 and 2000, respectively. He is currently working as a Teaching\Research Associate in the Department of Electrical and Electronics Engineering, College of Engineering, Guindy, Anna University, INDIA. His research interests include power systems analysis and modeling, distributed computing and Internet technologies.

V. Ramachandran received his M.E. and Ph.D. in Electrical Engineering from College of Engineering, Guindy, Anna University, Chennai, India. He is currently a Professor of Computer Science and Engineering in College of Engineering, Guindy, Anna University, India. His research interests include power systems reliability engineering, network security, component technologies and soft computing.