SOAP ON THE POWER SYSTEM APPLICATIONS

Liao Jiaping  Lu Benjie
Huazhong University of Science and Technology

Abstract The potential requirement of hetero-system inter-operation exists widely in the power system applications. The common solutions are COM and CORBA. The main limitation of these methods is the inconsistency among the implements With the birth of the SOAP protocol, A new, much easier, much convenient method emerges in the implementation of the inter-operation in the power system application. This paper analyzes the model of the inter-operation in power system application, proposes a new method to utilize the SOAP in the power system. We can see the SOAP can greatly enhance the extensibility and compatibility and inter-operation of the power system application.

Keywords SOAP XML HTTP COM CORBA POWER SYSTEM

1. Introduction of SOAP

SOAP(Simple Object Access Protocol) was submitted to the W3C by Microsoft,IBM,Acba and Commerce One in May 2000. It has been accepted as a W3C note. SOAP extends the XML-RPC, a protocol which has gained good names in industry. SOAP includes the XML schema specification, which provides a mechanism to define data types much more freely than before. SOAP combines the HTTP and XML successfully. It is a scaleable communication protocol Its core spirit is transform the messages wrapped by XML format through HTTP. The communication end-users encode and decode the messages independently. There are not brand new things here, but the combination of HTTP and XML provides an new method to communicate. There are three main parts in the SOAP:

a. SOAP envelop. It defines a set of framework to express the messages including the contents and how to conduct them.
b. SOAP encode rule. It shows how to define the data types the application requires.
c. SOAP RPC It shows how to express the remote process call and answer.

SOAP is only two year old, it has shown a good future in the distribute compute domain because of its simplicity and expansibility and flexibility.

2. The distribute compute and the inter-operation in power application system.

In our country, especially in power-produce enterprise, industry automation has been developed to such an extent that the application software has been more and more complicate, and become an application cluster. As an instance, in the center controlling room in a cascade dispatch center, there are at least such applications: SCADA, monitor system, AGC (auto generation
control), AVC(auto voltage control), MIS (management information system), decision support system, reservoir dispatch system, EMS(energy management system) etc.

These applications exist in different hardware platform, supported by different operation system, implemented by different and independent process. They all read from and write to the database, and they often interact with one another. It is almost certain that there are a lot of redundant and reduplicate parts in the application cluster. These reduplicate parts not only waste the human resource and time and energy and money, but bring the redundant function problem, which make it more difficult to cooperate among the applications, to administrate, to upgrade the system, just like the redundant data in database does. So, it is very necessary to pursue the object that “one function module, one implement” not only in an individual application, but in an application cluster. In this case, RPC is very important role the system applications.

3. Traditional Implementation methods.

On Windows operation system, Microsoft provides DCOM (Distributed Common Object Model) as the solution of the distribute compute, its logic flow can be shown as follows:

The biggest limitation of DCOM is that it is limited in Windows system. Although some Unix systems declaim to support the DCOM and implement it in their system. We can find few DCOM implements in other operation systems. At the same time, because of the complexity of DCOM itself, DCOM programming needs high techniques while common programmers can hardly possess.

In unix system CORBA (Common Object Request Broker Architecture) is the most common method to implement distribute compute. CORBA is maintained and recommended by OMG(Object Management organization) and is supported by a lot of hardware and software manufactures. CORBA also has its own problem. The ORBs( Object Request Broker) are not the same among the separated manufactures. Although the inter-operation among them should be possible, according the specification of CORBA, the ORBs can communicate on the support of IIOP. But the specification is one thing, the reality is another. The workflow of CORBA is shown below:
Both the two methods (DCOM and CORBA) need a special TCP port to communicate. Without the permission of the administrator, although you know your calling is absolutely safe, the process can not penetrate the firewall in Internet.

4 Implement Distribute Compute with SOAP

On the framework of SOAP, the RPC can be shown as:

Compared to the complicate CORBA and DCOM technique, SOAP only utilizes two simple protocol: HTTP and XML, the former is responsible for data transport while the latter for encoding. HTTP is a simple communication protocol that it is implemented on almost all the operation systems. While XML is only a specification to express data independent to the operation platform. More and more tools can be used to encode and decode the XML documents. These things give a good chance to the SOAP.

Because of the simplicity, flexibility and expansibility of XML, SOAP inherits these virtues too. With SOAP, we can build a loosely coupled communication pipe in the internet quickly. Absolute different end-users, including Non-compatible DCOM and CORBA component, can exchange data, can provide or call services through this pipe. So, distribute compute can be implemented among hetero-structure systems through SOAP. Its expansibility is almost unlimited. The following graphic shows how to communicate between COM and CORBA.

5. Case study.

Here we will show how to implement inter-operation in power system applications by the examples of Reservoir Dispatch System and Management Information System (MIS).

Reservoir Dispatch System is the core application in an Reservoir Dispatch Center. Its main function is to calculate the water dispatch among the cascade reservoirs. During the process of calculating, it will take account into the predict of rivers flow, the real-time river flow, the attribute of the cascade reservoirs, the attribute of power units, the predict of power load, the navigation
request, the irrigation request or other constraints about water flow. It is a very complicate system. While Management Information System is rather simple. It emphasizes on the enterprise information management, transaction management. But when it publishes its data to the Internet or acts as decision support system, there is potential requirement to call the calculating model in the Reservoir Dispatch System

From the viewpoint of implementation, Reservoir Dispatch System may run in Unix workstation or High-level PC. The calculate model should be designed as a COM component or a CORBA component because of its complexity. The whole system will be a typical Client/Server model. While Management Information System is a typical WEB application. The user operates it through a web browser. As common ways, inter-operation between the two system is conducted by DCOM or ORB. These methods are not only complicate but also have many constraints. When one hand changes, the other may stop running.

Here we propose to implement the inter-operation by SOAP. We can find that this method has great compatibility because of the flexibility of SOAP. At the same time, the complexity of encode has been sharply cut down.

Here we suppose the function we will call is DispatchFun with parameters DispatchParam:

1). Envelope the call with SOAP format. Envelope the namespace the method locates, the name of the method, and the parameters into XML documents according the SOAP specification:

```xml
<SOAP:Envelope
 xmlns:SOAP="urn:schema-XMLSOAP-org:SOAP.v1">
 <SOAP:body>
   <m:DispatchFun>
     <para>DispatchParam</para>
   </m:DispatchFun>
 </SOAP:body>
</SOAP:Envelope>
```

During the process of envelope, we can use any tools we like, for example, the SOAP toolkit or even the text editor.

2). Send the XML document to the remote server.

```javascript
oHTTP=MS.XMLHTTP;
oHTTP.open (www.test.com.cn) // url of remote server
oHTTP.setRequestHeader("Content-Type","text/XML")
oHTTP.setRequestHeader("SOAPMethodName" DispatchMethod) //set the header of the request
oHTTP.send(XMLDOcument)
```

Now, the HTTP request is:

```
POST / HTTP/1.1
HOST:www.test.com.cn
Content-Type: text/XML
Content-Length: nnnn
<SOAP:Envelope
 xmlns:SOAP="urn:schema-XMLSOAP-org:SOAP.v1">
 <SOAP:body>
   <m:DispatchFun>
     <para>DispatchParam</para>
   </m:DispatchFun>
 </SOAP:body>
</SOAP:Envelope>
```
3) Server gets the request through HTTP:
   oXML=MS.XMLDOM
   oXML.load(request)
4) Extract the method and parameters from the request:
   sMethodName=Request.ServerVariables("HTTP_SOAPMethodName")
   param=XMLToValue(oXML)
   SOAPCALLBACK(param)
5) Implementation of SOAP service.
   In the function SOAPCALLBACK we can implement the service arbitrarily, including creating COM components, creating CORBA objects, visiting the DBMS, calling other services etc. In fact, here we can do anything we like. There is not limitation at all.
6) The server envelopes the results according the SOAP specification and then returns to the client.
   DomServer.CreateResponseXML(NameSpace, MethodName, retVal)
   Response.ContentType="text/XML"
   Response.write DomServer.XML
   The results will be:
   <SOAP Envelope XMLns: SOAP="urn:schema-XMLSOAP-org:SOAP.v1">}
   <SOAP :body>
   "m: DispatchFunResponse>
   ……
   </m:DispatchFunResponse>
   </SOAP :body>
   <SOAP Envelope>
   If there is something wrong during the calling, return error information to the clients, of course, in SOAP format.
   <SOAP Envelope XMLns: SOAP="urn:schema-XMLSOAP-org:SOAP.v1">}
   <SOAP :body>
   <SOAP:Fault>
   <SOAP:faultcode>200 </SOAP:faultcode>
   <SOAP:faultstring> SOAP must understand error </SOAP:faultstring>
   <SOAP:runcode>1 </SOAP:runcode>
   </SOAP:fault>
   <SOAP: body>
   <SOAP: Envelope>
   7) The client gets the results:
   ret=oHTTP.responseXML
   First check the error information:
   oSOAP.selectSingleNode("faultcode").nodeTypedValue
   oSOAP.selectSingleNode("faultstring").nodeTypedValue
oSOAP.selectSingleNode("detail").nodeTypedValue
If everything is OK, then get the final result:
oRet=XMLToValue(oSOAP)
This is a whole SOAP calling work flow.

6. Conclusion

From the case we can see SOAP make a full use of expansibility of XML specification. It is quite easy to implement distribute compute through SOAP. It can resolve the collision among the different models, such as DCOM and CORBA. SOAP provides a platform to enable the distribute compute through loosely coupled communication pipe. In power application system, SOAP enhances the expansibility, improves the performance and inter-operation.

Reference: