PARAMETERIZED ASSEMBLING MODEL OF COMMUNICATION SERVICE

Tianzhou Chen, Wei Hu, Zhaohui Wu
College of Computer Science, Zhejiang University, Hangzhou, P.R.China, 310027
{tzchen, ehu, wzh}@zju.edu.cn

ABSTRACT:
Protocol components and protocol component library are constructed by dividing protocols into components. Component character parameter describes the quantization character of certain components. Quantization-oriented assembling model is driven by component character parameter. This model provides a transparent layer between operating system and protocol component library and deals with the encapsulated components. Then specific communication service is offered to communication devices. In the architecture of quantization-oriented assemble model, independent sub-functions are implemented by different intelligent agent. By the collaboration of multiple sub-modules, assemble process driven by component character parameter is implemented.

1. INTRODUCTION
Traditional network protocols are divided into different layers and every layer is with responsibility for corresponding communication function. One protocol family is constructed by multi-protocol at different layers. Every protocol is responsibility for different function. The OSI[1] and the TCP/IP protocol family is the deputation of the hierarchal network protocol. But the component-orientation of the network protocol compartmentalizes the protocol by different compartmentalizing methods. Then it will encapsulate the modules and this makes the protocol components become the bottom compositions. Moreover all components can upgrade, dynamic load, even execute across different networks.

Traditional protocols are monolithic, so it is monolithic service that this architecture can provide. It cannot answer to the flexible applications. The services will never be changed except that the services are reconfigured after the communication devices turn off. By all appearances this fixed mode cannot fulfill the need of the modern communication to function and performance increasingly improving. People do many things to provide modular communication services to the communication devices [3] and bring forward the mixed partition method. It can provide more flexible architecture by allowing configure a set of protocol functions to get a protocol and to combine the pre-designed protocol component. DaCaPo (Dynamic Configuration of Protocols)[4] is also an architecture based on functions but it adopts the same protocol configuration not distinguishing each connection. AVCOA[5] is a implementation of the communication protocol system based on components. It uses components to implement some protocol functions. Just like X-Kernel, all components adopt the identical interfaces. These research works advance the development of the communication protocol based on module and components.

We can construct the protocol component library by encapsulating the communication protocols. The protocol component library is the software component library which is developed against the character of the need of the network communication. The library is used to manage maintain and search the protocol components and it will be improved. The implementation of the component library is based on the network-oriented mechanism of the software engineering and the special management mechanism. The protocol components can’t provide services to the communication devices solely. So we need one mechanism to assemble the objects divided to provide the communication service. Therefore to dynamic reconstruct protocol stack, we construct the assembling model of component protocol to reuse the component and customize the communication service. The model selects the modules according with the need from multifarious protocol components and then assembles these components according to the layers of the protocol stack.

The quantization-oriented assembling model driven by the component character parameter in this paper is different from the fixed layers in the protocol stack and the fixed function of each protocol existing. The quantization-oriented assembling model is designed to orient the specific application of the communication protocol. The users can create their own stack by using the tailor-made service providing by the model. They can select different protocol functions to process the assembling work. If the stack is possible to get, the users can gain the optimized services to their application.

We call the content of component/protocol/communication services service object. The requests of component/protocol/communication services sending by OS are called service request.

2. THE COMPONENT OF THE PROTOCOL
We adopt the OO (Object-Oriented) and method of
component to operate on the assembling model itself during the assembling of the communication service. And we will define the operations to get the flexible architecture.

In this architecture of the network protocol, the protocols and protocol stack exists when the communication devices start. And after the start, these protocols can’t be changed, so they only can provide fixed communication devices. It is so rigescent that it can’t adapt the practical need.

The concept of the component is the extension of the OO. It’s progress from modeling to the thinking to the software manufacture; that is to say, components can be used as assessors in the software manufacture of application domain.

It needs to construct the component-assembling model to assemble and reuse the components because the components can’t provide services themselves after the protocols encapsulated and the protocol component library constructed.

The whole protocol component architecture is composed of OS, component library, component development platform, assembling model. PAM (Parameterized Assembling Model) alternates with OS and component library and only has indirect relation with component development platform. The data which is fed back from PAM to the component development platform. The architecture is shown in figure 1.

In the architecture, OS answers for sending the requests of the protocol components, and it must support the dynamic loading and dynamic uninstalling of the protocol components. Protocol OS is the source of the service requests. The component library implements the control of the user’s privilege. Simultaneously the library responses the requests to the components, implements the quick search mechanism and exactly comprehension mechanism of the requested components. While the main function of the assembling model is responsibility with the alternation between the OS and the component library; simultaneously, assembling model monitors the performance of the components, and records the data as the character parameters to feed back to the component library; tracks the different services and protocols in the system.

OS only sends out the requests and don’t know how to get the components or the services, and these processes are doing by the assembling model. Generally, assembling model will deal with the components if it can get the requested components or the components of the requested services. The OS will get the service object if above is true. Otherwise this request will be a request “out of all reason” (just that the requested components don’t exist in the component library), and OS has to give up its requests and continues its operations. In any case OS won’t care about the implementation course; it only sends the requests and receives the results.

The PAM can not only select the needed modules from the components and assemble them by the protocol layers but also customize the communication services and provide the assembling methods by the quantization-analysis fashion. It’s shown in figure 2. The users can get the optimized services to their applications if the requested protocol stack is feasible. This is the base springboard to bring forward the concept of the PAM.

The assembling model itself can be transferred to component and encapsulated by the idea of the component-orientation and OO and for unifying the standards of the whole component-orientation architecture. The components got after the component and encapsulated, just like the common components, can upgrade, dynamic load, even execute across different networks. These components also locate at the remote component library. It is in favor of the upgrading and managing the components after the assembling model of components and encapsulated. And we can effective utilize the flexibility of the components when using the components of the assembling model. These assembling model components can get different status by the applications. The flexible services based on the components and dynamic loadings are finer than the old static stack.

### 3. ARCHITECTURE OF PAM

At some time we will assemble the components to get the protocol stack, protocols and services. When assembling, we will adopt the PAM to complete the job. The PAM must do such things:

- Communication between PAM and OS/component library, between communication devices and component library.
- To test and initialize the service objects
- To assemble the service objects
- To manage the service objects

We bring forward the architecture of the PAM and use the different intelligent agent to implement the absolute functions. It’s shown in figure 3.

1. User. User illustrates the need aim service. These aim services will be embodied by different parameters. The requests will be encapsulated into the aim service’s illuminate sending to the communication module.
2. Communication Manager (CM). This module answers for the communication among the users and the other modules. CM will be executing if the component library is executing. CM will exist in the OS of the communication devices after the communication devices start.

3. Assembling Manager (AM). AM will answer for assembling the components to responsible protocols and corresponding the components. AM completes the assembling using the resources got from the assembling pool. AM will control the sequence of the assembling events.

Assembling Pool (AP): AP will be created after the AM is created. It includes the components to compose the service objects predefined by the component library. AM will control the sequence of the components to complete the assembling work. At the same time for the requirement of the assembling service.

4. Test Manager (TM): Test the assembled service objects by using the typical testing data. TM will test if the assembling work is done; test if the results of assembled service objects are satisfied the requests. The two tests will be done separately. Moreover the testing is not necessary but it’s important to build more reliable services.

5. Monitor Manager (MM): MM will answer for monitoring and recording the parameters of the components, and feed back to the component library in due course. At one time when OS submits new requests, MM will encapsulate the requests and the component parameters and submit to the component library.

6. Supervise Manager (SM): SM will control and maintain the communication service objects. The main target of the SM is to track and manage the protocol and protocol stack. It’s different from the MM. The main target of the MM is to monitor the components. It can provide the customized service by the assembling work cooperating with these different parts.

The PAM must provide quantization guidelines to decide how to response the requests, so we need the component character parameters.

4. COMPONENT CHARACTER PARAMETER

The PAM needs series evaluating parameters of service request objects to dynamic select and response the requests of the OS. So we bring forward the “component character parameter” in this paper to describe the idiosyncratic components by quantity. The MM gains the domain of the parameters of the request components by the dynamic component ability evaluating algorithm using the component character parameter. The component character parameter will be used to evaluate the components.

The component character parameter includes two contents: parameters of service resource and Quantity of the service.

Parameters of service resource(Ψ[β], β shows the parameters in the set) shows the requirement of the component to the system resource, mostly: execution states, utility of the CPU, locality of the memory and so on.

Quantity of the service(Ψ[β], β shows the parameters in the set) includes the parameters called QoS (Quality of Service) and some other parameters about components at least, mostly definitions.

Moreover this information can be expanded and the amounts of these parameters are limited. The character parameters will be initialized at the start of the system and support upgrading. The upgrading is controlled by the MM, SM, and CM. Initially the component character parameters will be set when dividing the protocols and can be set to zero or set to some especial initial value based on the dividing methods.

So we can define the components as following:

\[
\text{Protocol component} \quad \{ \Phi[\alpha]; \Psi[\beta] \}
\]

The protocol will be defined as a row vector:

\[
P = \begin{bmatrix} A_1; & A_2; & \cdots; & A_n \end{bmatrix}
\]

\[
S \ P = \begin{bmatrix} A_{11}; & A_{12}; & \cdots; & A_{1n} \\ A_{21}; & A_{22}; & \cdots; & A_{2n} \\ \vdots \\ A_{n1}; & A_{n2}; & \cdots; & A_{nn} \end{bmatrix}
\]
In this matrix the row respond to a protocol $P_i$. These parameters and matrix are maintained by the MM and SM.

5. QUANTIZATION-ORIENTED ASSEMBLING MEANS

OS sends the component requests to the PAM. OS encapsulate the basic description of the service object in the request. If it needs a new service object, there is the basic description of the performance. If it’s a replacement of a service object, OS only submit the requests.

The MM and SM can touch off the OS to send the component request by evaluating the component character parameters. The MM and SM tracks and records the component character parameters. The MM and SM will gather and record the related component character parameters when they execute normally such as the performance while high load, the performance while low load, reliability, the utility of the CPU, the locality of the memory and so on. The component library will adjust the component character parameters of the components after these parameters will be fed back to it.

The MM or SM will inform the OS that the OS should change some components if it find a parameter or several parameters ($\Psi[\beta_i]$) low than critical values when it monitor the component character parameters.

The MM or SM gets the parameters of the new components by operating the parameter matrix. The $\Phi[a_i]$ decides what services the communication devices can gain. The MM and SM gets $\Psi[\beta_i]$ by function $Func_c(SP)$, and gets the old $\Psi[\beta_i]$ by function $Func_c(SP)$. Given the new requirement $SP^*$ from OS, the MM and SM gets the new component by some ways, such as the following:

$$\sigma = \gamma \times \Phi[a_i] + [\Psi[\beta_i] - Func_c(SP^*)]^2$$

The $\gamma$ is the weigh value of the service resource parameters. So we can get the lower requirement to new components. It can carry its point if we use the other ways such as least square method and so on.

In order to guarantee better service quality, we can establish different PRI for different requests, and offer the right value that respond in the course of calculating.

The CM encapsulate requests and component character parameters from MM and SM and sends out these requests to the component library.

The CM of the component library end receives these requests, and the component character parameters encapsulated will be drawn out and then given to the component library. The component library carries on search the suited components then sends the messages and the components if found. Otherwise it only sends the messages.

After the CM of the communication device end receives the message, it will tell operating system the result of search. Then the assembling will go on.

Assembling can go on in two ends: the component library end or the communication device end. No matter in which position to assemble, its course is similar. The AM begins to assemble after it receives the code that the component library transmits to it. Here, the SM will offer protocol stack figure and protocol figure to support the assembling work. The SM will deal with the situation of monitoring the protocol stack and protocol rank, and provide the data of quoting in the new protocol stack figure and protocol figure to the AM. The service object assembled will be sent to the TM to be tested. After above-mentioned work finishing, the service object assembled successfully will be assigned to operating system to carry out. And the SM will monitor the results of the service objects, so we can get the probability of assembled successfully.

As we see during the course of assembling work, the component character parameters are to exist as the reason to the assembling work.

6. CONCLUSION

After the implementation of component of the protocol, the application mode is provided by the PAM; the users decide which protocols are their needs; so a new application protocol stack is constructed based on the two facts. The PAM can complete the functions that it can select required modules from different components and assemble to get new functions by protocol layers. The more important thing is that the PAM is driven by the component character parameters, and can give the assembling methods in the form of quantitative analysis to provide the customized communication services. The component character parameters clearly describe the communication device’s requirement to the service objects, and this make the service objects can be specified further. The service provided by the PAM is more specified and more adaptive to the requirement of the communication devices.

REFERENCES