Design and Implementation of Generic TC/CORBA Gateway Using Object Pool Mechanism

Jeong-Hwan Kim[0], Dong-Hee Lee, and Jong-Tae Park
School of Electronics, Kyungpook National University
{jhkim, dhlee}@ain.knu.ac.kr, park@ee.knu.ac.kr

오브젝트 풀 메커니즘을 이용한 지네릭 TC/CORBA 게이트웨이 구현

김 정환[0], 이 동희, 박종태
경북대학교 전자공학과

Abstract

Interworking of legacy TC application and CORBA-based one is important to satisfy emerging market demands on the rapid introduction of new multimedia services and building of scalable, flexible, and interoperable service control architecture. For this, we develop generic TC/CORBA gateway on interworking between TC and CORBA system. Generic TC/CORBA gateway can be used for interworking in many fields of telecommunication service control such as INAP, MAP, other TC-based application parts, and mobility management of IMT-2000. Finally, we propose the object pool mechanism for efficient management of TC-user objects and it is helpful for enhancement of scalability and reduction of the response time of remote service operations.

I Introduction

The liberalization and deregulation of the public telecommunication put the network and service providers in the competitive market environment. Additionally, a rapid introduction of new multimedia services is inevitable in this environment in order to occupy the leading position in the telecommunication world. Therefore, it is strongly necessary that telecommunication service architecture should be more scalable, flexible, and interoperable.

Recently, distributed object-oriented technology is adopted in Intelligent Network (IN) [1]. In intelligent network, Service Switching Point (SSP) is more difficult to be changed than Service Control Point (SCP), and the number of SSP is greater than that of SCP. Therefore, CORBA-based IN implementation is tried from upper layer to lower [2]. Gradually, full CORBA implementation will be achieved. In the future telecommunication environment, mobile network is also expected to adopt distributed object-oriented technology for mobility management and call control [3].

Eventually, legacy TC systems and CORBA-based systems coexist in transition and interworking of them is indispensable. There are researches and specifications done by Object Management Group (OMG), Consortium of Telecommunication Information Networking Architecture (TINA-C), and Open Group [4, 5, 6]. Specially, OMG provides specification; "Interwoking between CORBA and TC system"[5]. Transaction Capabilities/Signaling System No. 7 (TC/SS7) is used for transmission of signaling messages created by Mobile Application Part (MAP) and Intelligent Network Application Part (INAP) for service control. CORBA is being widely used as the appropriate infrastructure in a value-added telecom network. As a solution, TC/CORBA gateway performs conversion between CORBA IDL and TC application part such as INAP and MAP.

Firstly, we develop TC ASN.1 to IDL translator, which translates TC ASN.1 specifications such as INAP and MAP into IDL. Secondly, we design and implement generic TC/CORBA gateway. It is based on interaction translation specification of OMG [5]. The building of generic TC/CORBA gateway requires TC/SS7 interfaces, TC-user facilities and generated IDL interfaces by a TC ASN.1 to IDL translator. In addition, it should support application location, dialog initiation, dialog maintenance and operation invocation [5].

The Generic TC/CORBA gateway supports time critical feature. In gateway operation, TC-user object is created by request from remote service operations. This method makes delay elements. We propose object pool mechanism to figure it out. This mechanism maintains the created TC-user objects in an object pool of CORBA domain. After this, object pool administrator manages binding of proxy objects at a gateway and TC-user objects in CORBA system. This mechanism enhances the response time characteristics and scalability. Generic TC/CORBA gateway can be used as an intermediate solution to the full CORBA environment with reasonable cost and short time for service development, deployment, and execution.

In the following section, we present related works, which include TC/SS7, IN, CORBA, and their integration. In section 3, we analyze generic TC/CORBA interworking in detail, and then, we describe implementation methods of generic TC/CORBA gateway and show some example codes. In section 4, we analyze scalability and performance with regard to the response time of remote service operations. And we propose object pool mechanism, which efficiently manages TC-user objects in CORBA domain. Finally, we end with conclusion and future works.

II Related Works

1 Transaction Capabilities of SS7

The SS7 is the signaling protocol standardized by the ITU-T[7] to complete a call between two end-subscribers (either fixed or mobile). SS7 is the signaling protocol standardized by the ITU-T [7].

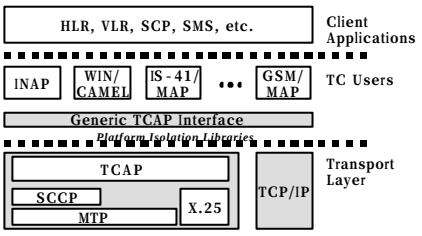

Figure 1. The TC applications and the TC/SS7 protocol suite

As shown in figure 1, the TC/SS7 protocol suite consists of the Message Transfer Part (MTP) that provides a connectionless, highly reliable datagram capability and the Signaling Connection Control Part (SCCP) that provides additional addressing capabilities. On top of this, the Transaction Capabilities Application Part (TCAP) is located For detail information of TCAP, see the reference [7, 8].

TC-based applications are defined as a collection of ASEs, which are defined as a set of TC functions for a specific purpose and are various from fixed, (e.g., INAP) [9] to mobile application, (e.g., MAP).

2 Common Object Request Broker Architecture

CORBA is a specification defined by OMG to provide a common architecture framework for distributed object-oriented applications. CORBA offers many advantages that make it an attractive choice as a basic technology that meet the requirements such as location transparency, in telecommunication area. In addition, CORBA supports a wide range of services, such as event service, naming service, lifecycle service, and the forthcoming messaging service [10].

3 CORBA and IN Interworking

There are some early works for interworking between CORBA based

IN applications and legacy IN infrastructure [1, 2, 4]. In particular, OMG [5] proposes the interworking of CORBA-based IN application entities (e.g., SCPs) with legacy IN application entities (e.g., SSPs) through a gateway mechanism. Furthermore, this gateway mechanism can be used as an adaptation unit for interworking legacy IN and TINA, which standardized by TINA-C and tries to evolve the IN by proposing an infrastructure that includes both the control and management.

III The Generic TC/CORBA Gateway

1 Generic TC/CORBA Interworking Architecture

The overall architecture of the generic TC/CORBA gateway is shown in figure 2.

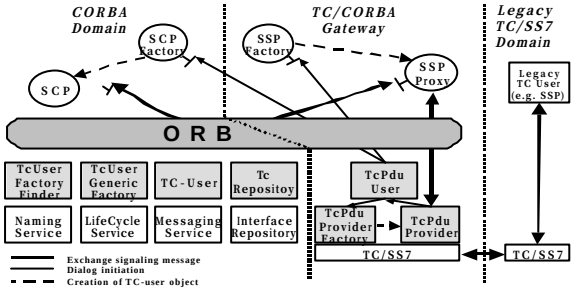

Figure 2. The generic TC/CORBA interworking architecture

The architecture consists of three divisions; CORBA domain, generic TC/CORBA gateway, and TC/SS7 domain. TC-user objects are usually located in CORBA domain. These TC-user interfaces inherited from TC-user facilities are implemented and executed in this domain. The generic TC/CORBA gateway consists of TC PDU oriented interface, which allows proxy objects to be implemented with the consistent prototype and proxy objects representing legacy TC-user applications. In TC/SS7 domain, typically legacy TC-based applications send requests and accept the results via the TC/SS7 protocol suite. For detail information of each interfaces, see the reference [5]

The generic TC/CORBA gateway supports four major interaction features as follows: application location (finding), dialog initiation, dialog maintenance and operation invocation. In general, application location and dialog initiation are provided by the CORBA naming service and the life cycle service. Dialog maintenance is provided by the base interfaces of all TC-user CORBA server objects and operation invocation is provided by TC-user interfaces, the ORB, the messaging service and optionally the interface repository and TC repository. TC facilities at a gateway are inherited from or directly use CORBA basic services. A CORBA-based TC-user object in CORBA domain communicates with a proxy object at the gateway. These facilities and interfaces are summarized in table 1.

2 The Interworking Between Legacy TC Application and CORBA-based TC-user Object

We describe the interworking procedures of the generic TC/CORBA gateway using an example of INAP TC-user as shown in figure 3.

(1) A *TcPduUser* is registered with a *TcPduProvider Factory* to request notification of PDUs with a particular (GT) and (AC) received from the TC/SS7 network.

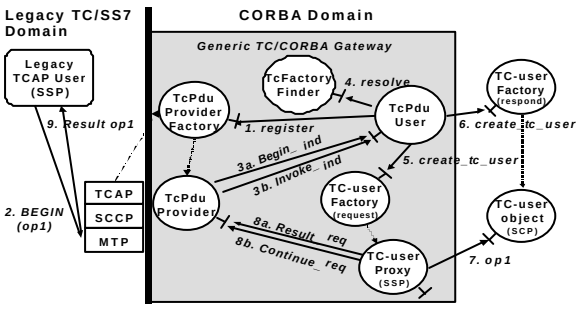

Figure 3. Interworking procedures

(2) As soon as a BEGIN PDU with a registered GT and AC is arrived at a gateway from TC/SS7 domain, the *TcPduProviderFactory* creates a suitable *TcPduProvider* object to interact with the TC/SS7 protocol stack. (3a) The *TcPduProvider* invokes a *begin_ind* operation on the registered *TcPduUser* object.. (3b) The following *invoke_ind* operation contains the remote operation in the received BEGIN PDU. (4) The *TcPduUser* resolves the GT and obtains the factory reference through *TcFactoryFinder* interface. (6) *TCPduUser* creates a proxy object representing the legacy SSP at a gateway via SSP creation factory, and then the responder interface is created by SCP creation factory. (7) Operations can be invoked on the created TC-user object using the CORBA Messaging Service. (8a) A SCP CORBA object returns the result of operation to a SSP proxy that calls *Result_req* operation and (8b) *Continue_req* operation on the TcPduProvider object at a gateway.

3 Implementation of TC-user in CORBA domain

TC-user applications are various from fixed to mobile, and even the same TC-user can be implemented in many different ways. In this section, we describe the common procedure for implementing TC-users.

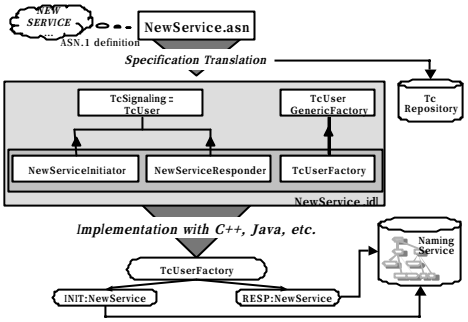

Figure 4. Steps for TC service implementation

This Procedure consists of three stages; TC ASN.1 definition, specification translation, and IDL implementation as shown in figure 4.

3.1 TC ASN.1 Definition for a New TC Service

It is only necessary in the case of a new service that has not been defined by ASN.1 constructs and/or Application Service Entities (ASEs). When a service has been already defined, jump to the next step.

3.2 Specification Translation (ST)

The TC ASN.1 definition for a TC-user is translated into CORBA IDL through ST according to the rule based on OMG work [5]. One TC ASN.1 definition is translated into three interfaces, initiator for a proxy object, responder for a TC-user object, and factory interfaces for creation of those two objects. Figure 5 shows ST of INAP, which fully complies with OMG ST rule.

```
core-INAP-CS1-IP-to-SCP-AC APPLICATION-CONTEXT
- - dialog initiated by IP with AssistRequestInstructions
INITIATOR CONSUMER OF {
...}
  ::= {ccitt(0) ... version1(0)};
SCF-SRF-activation-of-assist-ASE
  ::= APPLICATION-SERVICE-ELEMENT
- - consumer is SSF/SRF
CONSUMER INVOKES { assistRequestInstructions }
Timer-ASE ::= APPLICATION-SERVICE-ELEMENT
- - supplier is SCF
SUPPLIER INVOKES { resetTimer }
Specialized-resource-control-ASE
  ::= APPLICATION-SERVICE-ELEMENT
...
```

```
interface Core_INAP_CS1_IP_to_SCP_ACInitiator
                    :TcSignaling::TcUser {
    ... resetTimer( ... ) raises(... );
};
interface Core_INAP_CS1_IP_to_SCP_ACResponder
                    :TcSignaling::TcUser {
    ... assistRequestInstructions ( ... ) raises( ... );
};
interface TcUserFactory
                    :TcSignaling::TcUserGenericFactory {
  Core_INAP_CS1_IP_to_SCP_ACResponder
    create_Core_INAP_CS1_IP_to_SCP_ACResponder(...)
    raises ( ...),
  ...
  Core_INAP_CS1_IP_to_SCP_ACInitiator
    create_Core_INAP_CS1_IP_to_SCP_ACInitiator();
};
```

a) translation example of ETSI INAP CS-1

Figure 5. The example of specification translation

3.3 IDL Implementation

Finally, we left object implementations of the generated IDL interfaces with programming language such as C++, Java, etc. and then, bind TcUserFactory object and two naming contexts under a specific GT and AC through naming service. In execution time, a TcUserFactory must create initiator and responder in order to start association.

IV Object Pool Mechanism for Efficient Management of TC-user Objects

When implementing a real time application in CORBA domain, response time should be considered because it affects call setup delay. In addition, scalability also should be considered to cope with the unpredictable requests efficiently. For this, OMG proposes two mechanisms; multiplexing mechanism for dialogs by use of a globally unique association ID (*see the case I in the figure 7*) and dialog flow control (*see the case II in the figure 7*) [5]. Despite of using OMG's mechanisms, there is a basic problem related to object lifecycle. So, a TC-user object, which has ever instantiated, should be reused for other requests. Therefore, we suggest object pool mechanism. The object pool contains a stack of instantiated objects. Once the object pool receives a call from a remote TC-user, it allows one of object instances in the pool to cope with the request, and after finishing, the object will be returned to the pool for later use (*see the case III in the figure 7*).

For implementing the object pool mechanism, we newly define *PoolAdmin* interface that provides basic functions to add, remove an object, and some additional facilities to manage an object pool as shown in figure 6. It defines a structure for each entry named *PoolEntry* that consists of TC-user object reference and its current state. In addition, it has an attribute called *PoolRepository* that consists of several *PoolEntries* as a repository of object pool.

```
struct PoolEntry {
        CORBA::Object object_id;
        CORBA::Boolean status;
}
typedef sequence<PoolEntry> PoolRepository;
...
interface PoolAdmin {
        readonly attribute PoolRepository pool_repository;
        Object Lookup() raises(...);
        void AddEntry(in short idx, in PoolEntry object) raises(...);
        void RemoveEntry(in short idx) raises(...);
        void Destroy() raises(...);
...
}
```

Figure 6. *PoolAdmin* interface

The case III in the figure 7 shows the operations of object pool mechanism. Firstly, it allows a factory to create objects and to register the references at *PoolRepository* in *PoolAdmin* interface. After that, every registered object should notify its current state, that is, busy or idle. Once a request from a TC-user is arrived at a gateway, the *PoolAdmin* checks whether an object pool associated with the request has been made or not. And if yes, a *PoolAdmin* returns one of object references whose state must be idle. Every request received from the distributed TC-users is allocated to the corresponding object in the pool in turn. When the number of TC-users is greater than the number of objects in the pool, the rest of requests should be buffered within the tolerable time limit, otherwise an exception would be thrown. In the case of receiving more than one request from a single TC-user at unit time interval, a *PoolAdmin* allows the object to multiplex these requests by allocating a globally unique association ID.

V Performance of Object Pool Mechanism

1 Response Time

The response time of TC-user consists of time to search for a factory, time to create a new object, waiting time, and service time. Waiting time is bounded to the overall system performance rather than object's, and service time depends on the capability of each object. Therefore, we assume that these factors are not varied according to object management strategies. On the other hand, factory searching time and object creation time is varied according to the strategies. Compared with factory searching time at a normal gateway, normal gateway, it takes less time to lookup a corresponding object pool at a gateway using object pool mechanism, and the latter doesn't have to consider the object creation time. Conclusively, object pool mechanism is helpful to reduce the response time because it can skip steps for searching a factory as well as creating a new object through reusing of the created objects in the object pool.

2 Scalability

It is also necessary that we consider how many objects in a pool should be created to cope with the overall requests properly. For this, the average arrival rate of requests, λ and the average service time, E(T) are should be considered in order to determine the number of object instances in the pool.

The number of object instances in the pool is

$$M = l \cdot E(T)$$

The above expression is equal to Little's formula [11] and we consider CORBA domain where TC-user applications are located as a queuing system. Thus, M states the number of waiting object instances in the system. Accordingly, when M object instances in the pool are created, the number of object instances in the queue will be minimized. Furthermore, M can be easily adjustable according to the variation of request arrival rate. Consequently, object pool mechanism can be more scalable in a system expecting unpredictable requests

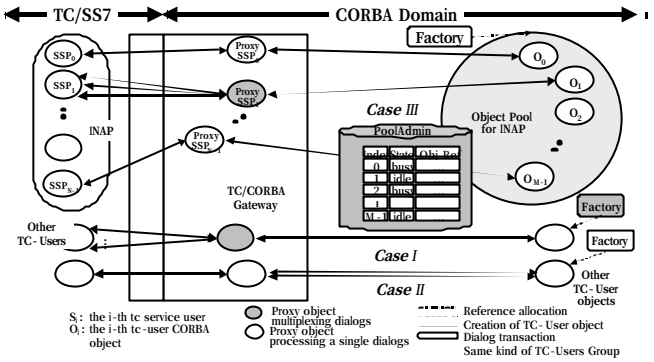

Figure 7. Interaction mechanisms using standard, dialog flow control, object pool technique

VI Conclusion

Based on the OMG work on interworking between CORBA and TC systems, we have developed generic TC/CORBA gateway, which can be used to build flexible, scalable, and interoperable service architecture for service control and management of IN, IMT-2000, and TINA. We introduced the use of object pool mechanism for the purpose of efficient management of TC-user objects. This mechanism makes a gateway more scalable and less response time in that it saves steps for searching an associated factory and creating a new object through the factory. It takes a little to lookup one of the instantiated object references in a pool, but less than time to perform those steps.

We will adopt the upcoming Messaging Service to our implementation. In addition, the optimized lookup algorithm will be developed to enhance object pool mechanism. Further more, we will develop the full CORBA architecture using a gateway interworking between TC/CORBA, CORBA/CORBA.

References

[1] Subrata Mazumdar and Nilo Mitra, "ROS-to-CORBA Mappings: First Step towards Intelligent Networking using CORBA," IS&N'97.

[2] Helge Armand Berg and Stephen Brennan, "CORBA and Intelligent Network(IN) Interworking," IS&N'98.

[3] Dong-Hee Lee, Moon-Sang Jeong, and Jong-Tae Park, "CORBA-based Integrated Control and Management for IMT-2000 Global Roaming Service," NOMS'2000.

[4] TINA-C, Business Cases and Critical points for TINA-IN Version 1.0, Oct, 1998.

[5] OMG telecom/98-10-03, Revised final RFP on "Interworking between CORBA and TC Systems.

[6] The Open Group, "Preliminary Specification Inter-Domain Management: Specification Translation" X/Open Document Number: P509.

[7] ITU-T Rec. Q.771 , "Signalling System No. 7 - Functional Description of Transaction Capabilities".

[8] ITU-T Rec. X.880 (1994) | ISO/IEC 13712-1:1995, "Information technology -Remote Operations: Concepts, model and notation".

[9] ITU-T Rec. Q.1218 " "Interface Recommendation for Intelligent Network CS-1", Geneva, 1995.

[10] OMG orbos/98-05-05, CORBA Messaging.

[11] Alberto Leon-Garcia, "Probability and Random Processes for Electrical Engineering," ISBN 0-201-12906-X.