Design-Aspects for the Integration of CORBA-Based Value Added Services and Intelligent Networks

Frank Imhoff and Marcos dos Santos Rocha

Aachen University of Technology
Department of Computer Science, Informatik 4
Ahornstraße 55, 52056 Aachen, Germany

Abstract: This paper deals with the realisation of Value Added Services known from legacy Telecommunications networks on top of middleware platforms. These legacy services are predominantly based on ITU's Intelligent Network (IN) definitions. Due to the forthcoming network convergence to IP-based networks and the tremendous growth of telecommunications markets these services have to be distributed over a wide range of heterogeneous hardware platforms. Nevertheless, the interconnection to existing circuit-switched networks has to be taken into account. This paper discusses different design aspects for a gateway between CORBA and IN-based services and suggests a dynamic as well as a static approach. Ongoing studies will optimise and evaluate these approaches.

1 Motivation

The technical development and the liberalisation of the telecommunications markets showed that for the operators of predominantly circuit-switched telecommunication networks the provision of *Value Added Services* becomes more and more important. The implementation of such services is difficult. With the introduction of *Intelligent Networks* (IN) somewhat eased the problem but this can not be a solution to satisfy the demand of more and more complex services in the medium-term. *Voice-Data-Integration* and the use of the internet for voice communication (*Voice over IP*, VOIP) requires the introduction of new technologies.

An important aspect during the introduction of new technologies for an efficient implementation of Value Added Services is the independence of the underlying network and hardware platforms. On the one hand this ensures the independence from manufacturers and providers, and on the other hand the internet may be used as transport network [3]. Moreover, potential customers of complex and customised services (e.g. Call Center) want to administer these services themselves, and do not want to invest in expensive, proprietary hardware. Therefore, the use of an object-based software solution such as the Common Object Request Broker Architecture (CORBA) or the Telecommunication Information Network Architecture (TINA) seems inevitable.

2 The CORBA/IN-Gateway

To enable smooth integration of IN-based networks and CORBA the gateway has obviously a key position in the system between these islands. The success of the in

C. Linnhoff-Popien and H.-G. Hegering (Eds.): USM 2000, LNCS 1890, pp. 304-309, 2000.

troduced of CORBA-based Value Added Services depends predominately on the efficiency and functionality of such a gateway [2]. Therefore, to meet the common requirements of existing telecommunications systems such as reliability, scalability, or real-time capabilities become more and more important. First, however the gateway should enable the transparent communication between conventional *Service Switching Points* (SSP) and CORBA based *Service Control Points* (SCP) in both directions.

2.1 Translation Interaction

During the translation interaction, the information which is generated by the translation of the specification, but not transmitted explicitly in the message or the function calls needs to be also taken into account. This could, for example, be a TCAP-message (Transaction Capabilities Application Part) which is only marked by an *operation code* at run-time. The gateway may either statically or dynamically deliver the names of the IDL methods which correspond to this operation code. The timer-values which are allocated to certain operations are another example for the information needed at run-time are. Information such as the allocation of the *Operation Code Function Name* or the coding of the parameters of an operation (ASN.1 tags) can be directly defined and compiled in the code, as well as requested dynamically from a *TcRepository* at run-time. We discuss this important difference in the next sections.

Static vs. Dynamic Gateway

The most important question concerning the design of an IN-CORBA gateway is weather to follow a static or dynamic approach (see Fig. 1).

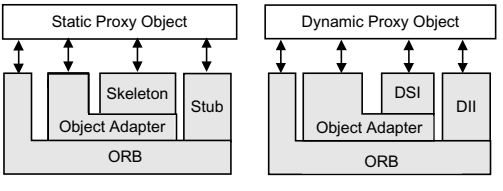

Fig. 1. Static vs. dynamic gateway architecture with integrated proxy

A static gateway implements proxy objects which are defined in their IDL interfaces at compile-time. That is, these objects implement only static interfaces referring to specific protocols, e.g. a variant of INAP. Almost all changes of these protocols lead to a recompilation of the objects. This makes sense if the interaction function is realised in the uppermost layer of INAP, since in this case the gateway only has to be realised for one single protocol. The proxy objects of a dynamic gateway use the functionality of the *Dynamic Skeleton Interface* (DSI) to handle all kinds of function calls. On the other hand the proxies use the *Dynamic Invocation Interface* (DII) in conjunction with the *Interface Repository* to transform an at compile-time unknown INAP message into a CORBA method. Thus, it is possible to create an appropriate call for each function known at runtime, and as a result a transparent communication is still granted. However, the dynamic realisation has its disadvantages. The static

gateway knows all the information it needs for the translation at compile time, and allows a much more efficient implementation, because no access to the interface and the TC repositories is necessary, and because additional effort for using the DII and DSI interfaces is required. These access times are difficult to predict. Only an implementation of both the static and the dynamic gateway would allow a realistic comparison. It is to be examined whether a mixed solution (dynamic and static) is of interest.

Translation BER ↔ CORBA-Any
The structure of an INAP message is specified in ASN.1 notation, at runtime they are coded into BER format [8, 9]. The gateway reads (binary) data in ASN.1/BER format and translates them into *CORBA-Any* structures and vice versa. This is realised on a very low level. To implement the dynamic gateway, it seems to be reasonable to use a tree as data structure for the internal encoding and decoding of ASN.1/BER data to reflect the hierarchic structure of the data. This also avoids the low level operations necessary for encoding and decoding of ASN.1/BER data are distributed over the whole program code. This tree structure is used in conjunction with CORBA-*DynAny*s.

Asynchronous Communication and Quality of Service
An operation is called synchronous if no other operations are allowed to be called before it is terminated. TCAP operations are asynchronous, i.e. they do not have the above mentioned limitations. However, for the time being CORBA offers only two function call modules - synchronous and deferred synchronous; the deferred synchronous modules can only be used in connection with the DII.

To convert TCAP operations to real asynchronous function calls, one will need the so called CORBA Messaging-Service, which will soon be officially specified by the OMG (CORBA 3.0 Specification). In addition to a real asynchronous function call, the *Messaging Service* offers *Quality of Service* (QoS) functionality, determining for example timeouts and priorities. However, because it is not intended for a series of CORBA implementations to offer the *Messaging Service*, asynchronous TCAP operations must first be converted to synchronous method calls within the gateway. Thus, an overlap of a given reply time for a method call will not be recognised. Yet, assuming that the present system is fast enough, this should not be a problem.

The Course of the Interaction
We will now explain the typical course of the interaction translation, using an example to show the communication between a legacy *Service Switching Point* (SSP) and a CORBA-based *Service Control Point* (SCP) (see Fig. 2). To simplify, we assume that the gateway objects are already initialised, i.e. the factories for *TcUser* and for the proxy objects have been started and are registered by *TcServiceFinder*. As this is a dynamic gateway, both the Interface Repository and the TcRepository are needed [1]. These are not shown in Fig. 2.

After the SCP has received an InitialDP message, it sends a message to the SSP. This is done through the direct call of the associated function by the proxy. The InitialDP is defined without a return value. If it deals with an operation defining a return value,

the proxy will code it in the ASN.1/BER format and passes it to the provider with a *return_result Operation.* Subsequently, the provider sends to the SSP either a Begin-PDU or an End-PDU containing the results. In the above example, the dialogue was initialised from the IN-side. It is foreseeable that the dialogues can also be initiated at the CORBA side, e.g. by the SCP.

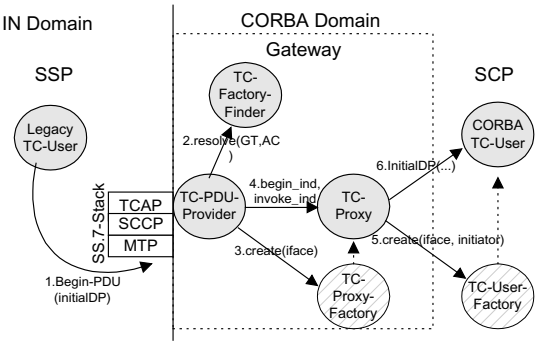

Fig. 2. Example of an interaction: InitialDP

2.2 Integration of the CORBA-Based Application Layer

The interfaces of the CORBA *TcUsers* (see Fig. 2) have been defined in the course of the specification translation. Principally, they could have been implemented directly from the CORBA-based application. Since one could be unwilling to give out for new CORBA applications the interfaces which are defined for an SS.7-based communication, the *TcUser* will become an adapter beforehand. Considering the communication between traditional *Service Switching Function* (SSF) and CORBA-based *Service Control Function* (SCF), it seems to make sense to merge the *TcUser* with the implementation of a CORBA-based *Global Service Logic Program* (GSL). However, whether this can be realised without Threads has to be investigated in more detail.

3 Performance Analysis

First simulations have been done on two PCs (Pentium III 500 MHz, 128 MB RAM) connected over a shared Fast-Ethernet LAN. The first computer was running the simulation manager, the SSP simulators, the *TcPDU* providers, and the *TcFactory Finder*. The remaining objects were running on the second computer. It is important to note that in this first approach each SSP simulator communicates with exactly one *TcPDU* provider creating its own *TcProxy*.

Several simulation runs have been performed to measure the performance of both the dynamic and the static gateway. Additionally, a simulation manager was used to start multiple SSP simulators simultaneously. After an SSP simulator is started, it starts sending *freephone service* (IN-Service "tollfree") requests to the gateway as of *InitialDP* operations encapsulated in TCAP messages. In the next step the gateway translates these messages to CORBA method calls and are forwarded to a SCP simu-

lator. The SCP simulator responds to each *freephone service* request by calling two methods on the *TcProxy*, which translates these method calls in two INAP operations *Connect* and *RequestReportBCSM*. The gateway then sends these operations to the corresponding SSP as a TCAP message. Altogether the gateway translates three INAP operations while handling a *freephone call*.

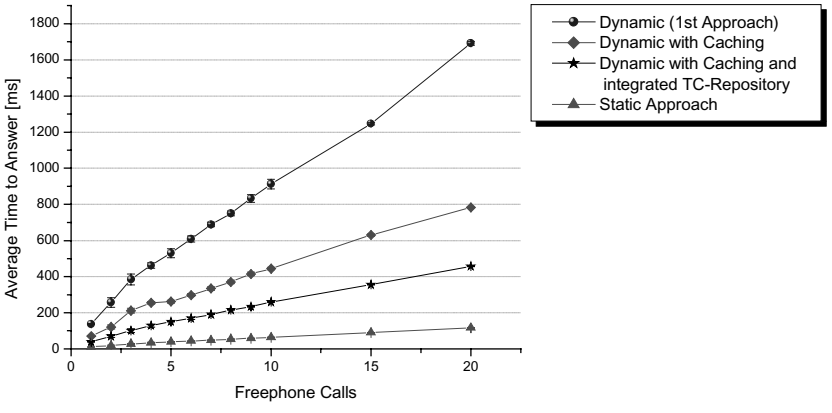

Fig. 3. Static versus dynamic gateway performance including optimisations

Fig. 3 shows that the dynamic translation is much more time consuming than the static one. The reasons for the higher complexity of the dynamic approach can be identified as follows:

- The dynamic marshalling and unmarshalling of CORBA operation parameters and results are very expensive operations. As a CORBA object is not able to implement dynamic and static interfaces at the same time, the *TcProxy* must dynamically unmarshall all the operation parameters and results it receives, even if the signature of some operations is already known at compile-time (e.g. general TCAP message indications).
- The creation of dynamic CORBA invocations is more expensive than using precompiled stubs.
- The *Interface Repository* as well as the *TcRepository* may very short become a bottleneck for the gateway. This happens especially if these objects do not use threads, as concurrent requests may always block each other. To reduce the number of accesses to the *Interface Repository*, the *TcProxies* store information about queried interfaces in a cache (see Fig. 3). Another possible approach would be to have the *TcRepository* actually doing the entire translation of INAP operation parameters, results and errors.
- The dynamic construction of data types being unknown at compile-time is a very powerful CORBA feature using the so called *DynAny* (dynamic any) class. Unfortunately, most applications usually do not make use of this class. Especially the translation of long BER encoded ASN.1 “SEQUENCE OF” data sets (e.g. SEQUENCE OF OCTETS) may be very slow.

4 Conclusion and Further Work

As the discussion so far has shown, the higher flexibility of dynamic gateways has its price. Comparing the results with mandatory requirements of telephony systems probably no current CORBA implementation is able to realize complex telecommunications services. For example, 2.0 seconds is the maximum time for the SCP to respond to a message received over the *Common Channel Signaling Network* (CCS) [7]. It takes into account the time it takes the SCP reply to reach the sending node, i.e. the SSP. Yet, future optimizations may enable the use of dynamic gateways described above at least for simple IN-based services.

As the performance of a CORBA application always depends on the Object Request Broker (ORB), it would be interesting to compare the performance of a MICO based gateway to a gateway based on different ORB, for example ORBacus. Nevertheless, the use of threads will improve the overall performance of the gateway. We foresee further performance improvements especially in the dynamic gateway implementation. The key point is the implementation of the *TcRepository*. On the other hand, the static gateway seems to have almost achieved an optimum. Additionally, it would be interesting to analyze the gateway's behavior after distributing the *TcProxies* over several computers and to investigate different load balancing strategies. It seems to be reasonable to combine static and dynamic *TcProxy* objects in one gateway, to get the benefits of both approaches, i.e. flexibility and performance. The integration of both gateway types will be as next step.

5 References

[1] S. Mazumdar, N. Mitra, "Design of a ROS-CORBA Gateway for inter-operable intelligent networking applications", Proceedings of IS&N Conference, Lake Como, Sept. 1997

[2] H.A. Berg, S. Brennan, "CORBA and Intelligent Network Interworking", Proceedings of IS&N'98, 1998

[3] F. Imhoff, A. Küpper: Intelligent Network Functionality in an IP based Environment. In: Fifth International Conference on Information Systems Analysis and Synthesis, Orlando, July/August 1999

[4] Object Management Group, "Interworking between CORBA and Intelligent Networks Systems Request for Proposal ", OMG DTC Document ftp://ftp.omg.org/pub/docs/, 1997

[5] AT&T, IONA Technologies, GMD Fokus, Nortel, Teldec DCU, "Interworking Between CORBA and TC Systems", OMG DTC Document ftp://ftp.omg.org/pub/docs/, 1998

[6] EURESCOM, "CORBA as an Enabling Factor for Migration from IN to TINA: A EURESCOM-P508 Perspective", OMG DTC Document ftp://ftp.omg.org/pub/docs/, 1997

[7] The Open Group, "Inter-domain Management: Specification Translation", Project P509, 1997

[8] S. Mazumdar, N. Mitra, "ROS-to-CORBA Mappings: First Step towards Intelligent Networking using CORBA", Proceedings of the Conference on Intelligence in Services and Networks, Lake Como, Mai 1997.

[9] T. Seth, A. Broscius, C. Huitema, H. P. Lin, "Performance Requirements for TCAP signaling in Internet Telephony", Internet Draft, Internet Engineering Task Force, http://www.ietf.org/draft-ietf-sigtran-tcap-perf-req-00.txt , August 1999