30.10.97

Environment for Designing and Testing of IN-Services

(for ICIN'98 / Service specification, creation and validation)

Tim Welsch, Hermann Wietgrefe, Klaus Jobmann

Institut für Allgemeine Nachrichtentechnik
(Institute for Communications, University of Hannover)
Universität Hannover, Appelstr. 9A, 30167 Hannover, Germany
tel +49 / 511 / 762-2814, fax +49 / 511 / 762-3030
email: welsch@ant.uni-hannover.de

Abstract:

The use of Intelligent Networks (IN) comes with two major advantages. First there are standardized interfaces between important network functions implemented which set the hope to become independent of the switch manufacturers. The second advantage is the very quick and easy providing of new IN-Services (INS) to the subscribers which is based on the use of tested, reuseable Service Independent Building Blocks (SIB). For institutions like providers or universities who do not own a running IN it is very difficult to deploy their new service-idea, but this step is important for study, examination, demonstration or education purpose. The presented developing environment is intended to fill this gap. It includes different tools for the designing and the practical testing of INS: a SIB-Editor, an IN-Service-Editor, a Service-Control-Function and two different Service-Switching-Functions. All descriptions which are used for SIBs and INSs are following the ITU Q.1200-Recommendations as far as possible.
The SIB-Editor-Tool manages a SIB library within a database. Althought the SIBs which have been defined at the Capability Set (CS) 1 are already available in the environment, the SIB-Editor provides the capability of creating new SIBs with new functions or modifying the existing ones. In the environment a SIB is based on three descriptions:

1. a verbal description of its function only as an information to the human user,
2. a formal description which defines SIB names, Call Instance Data (CID), Service Support Data (SSD) and SIB endpoints for the logical program flow which is important for the

runtime-operation on the Service Control Function (SCF) and the Global Service Logic (GSL) respectively and

3. an implementation part, which contains the raw function of the SIB in pascal language.

The SIB editor can compile the SIB's description to an executable program file which will be used by the SCF at operation time.

The IN-Service-Editor-Tool manages the chain of SIBs which actual forms the service. Every service can be loaded from and stored into the enviroments database. The INS editor provides a graphical desktop as user interface to present a service. The SIBs appear as object-boxes which can be dragged and dropped to get a clear and structured representation of the service. SIBs that are stored in the library can be added to the desktop and the actual chaining is done by mouse interaction with linking a SIB‘s logical endpoint with the startpoint of another SIB. The Basic Call State Mashine (BCSM) includes the interface-functionality of a switch and the Service Switching Point (SSP) respectively. This BCSM function is also represented on the desktop as a box with startpoints and endpoints which are called Points Of Return (POR) and Points Of Initiation (POI). The allocation of SSD-parameters is possible for every seperate SIB. During our examinations we found that it is insufficient to rigidly use only a comparison between a CID-Input name and a CID-Output name for transferring the data values to the next SIB. For this reason we provide a mask to every single instance of CID-Input parameters which can select any existing CID-parameter at the current service for masking.

The Service-Control-Function-Tool (SCF) works simular to an operation system of a Service Control Point (SCP) and makes a service available. The user can launch the various services that have been stored into the database. On a console window he can follow the resource allocation of this service or analyse the error message in case of a malfunction. At service launching time it is necessary that the SIBs needed for the selected service are available as an application running on the system. After starting a SIB executable it registers automatically at the SCF via a specialised software interface. It can be gathered from the ITU-Recommendations that it is sufficient to provide a function only one time at most on a physical point which is valid especially for the SIB functions. So we decided to share the SIB resource by entrusting the SCF and the GSL respectively to control the SIB activation requests by the various services. We also took a look at the data handling between GSL and SIBs: ITU determines to pass a pointer from the controlling GSL to the SIB which points to the real CID value that has to be transferred. Such a pointer grants read/write access on the CID. We prefer to distinguish between read and write because of the two following reasons: First our environment is used for development and may work with new, untested SIBs which are not necessarily error free. To enclose errors in services it is usefull to guarantee that a

SIB with only CID-Input property can not override the CID. The second problem may occur later on, when a third party manufacturer builds single SIBs which implementations are usual hidden. Then it may be interesting to define the read and write access seperatly to protect some user relevant CID like PIN or like previous used services tracked down by the previous routing number. To make this interface save we implemented a further stage in the GSL which can generate a copy of an input CID and publishes the corresponding pointer to the SIB. On the other hand it copies the possible modified SIB-Output value back to the real CIDs within the bounds of a regular CID-Output property.

For testing purpose of the launched services a SSF is required and for the signaling between SSF and SCF the Intelligent Network Application Protocol (INAP) is recommended. Because we wanted a flexible realisation the enviroments Functions are not bound to dedicated points. The SSFs may run on an other physical point as where the SCF runs. Here is the need to establish universal signaling links between the enviroments points. We studied varios possibilities for linking the points regarding to the use as a laboratory environment and regarding to the available hardware that consists most of PC with MS-DOS or MS-Windows as operating system. We found out that the IPX protocol do the job best with least expense. Moreover we are free to use the TCP/IP later on with the same hardware to involve points which use unix as operating system. The INAP messages are directly transmitted via IPX running on the institutes ethernet LAN. An incoming INAP message to the SCF which contains a valid service request from any SSF invokes the GSL. The GSL immediatly begins to call the corresponding SIBs in their chained order. All important GSL actions are displayed on the SCF console window to track down service malfunctions. As the logical program control reaches an POR of the BCSM-SIB the GSL origins an INAP message back to the SSF containing a possibly changed routing address.

We have developed two kinds of SSFs. The first one is a compact software tool which works like a virtual PABX/Switch. This tool can be used to check the routing capabilities of an launched service but voice-sessions etc. are not supported. On the desktop some subscriber lines are displayed with allocated preset routing numbers. The user can enter any sequence of dialled digits to request the service. The line to which the call is re-routed by the service starts flashing to show the routing result. The other SSF includes real PABX-Systems. Every switch-system consist of an PC with an network interface card (NIC) for signaling and one telecommunication card which provides four ISDN-BRI lines. It is possible to link two switches by crossconnecting one BRI line. The remaining lines are equipped with standard Euro-ISDN-telephon-sets. The switch software supports normal internal calls with no SCF interaction. IN calls are recognized by a table with valid IN triggers which is stored in the SSF. Such an IN trigger originates an INAP message to the SCF and the actions described above. As the new routing number is handed back to the switch the call and the speech

channel respectivley is forwarded to the next switch for example. The users can hook off, dial and talk normaly over this connection which is sufficient to show practically, how INS work and how they appear to the subscribers. Also the major part of the Service-Data-Function (SDF) is contained in the environment because Borlands pascal compiler includes an SQL-Database-Engine that provides easy access via SQL to vairous database formats. To get SDF functionality within a SIB there are just a few pascal/SQL commands at the SIB's implementation part necessary.

Within this environment we deployed services like Virtual-Number, Login-To-Virtual-Number and an Extended-Televoting with routing for percentage winners. All these services have been attended with small dialog-management-tools to show how easy the service provider can manage the service.

The level of development of this environment allows the fast implementation of new ideas for IN-Services. A new service can be testet and demonstrated on a small PABX system with real speech connections. The environment is also meant as a base for further studies on SIB interaction, IN-Service interaction and new definitions for SIB interfaces. Therefor we have to improve or add some monitoring and logging tools for the different internal GSL actions and states. Another topic can be a study about the sufficiency of ITU's CSs and which extensions to the INAP are necessary to implement a new CS. We are also working on the INAP implementation to a small existing Special-Resource-Function (SRF) that is needed for all services with audio input or output like speech and DTMF recognition and playing announcements to the subscribers.