Alternative Methods of Intelligent Network Service Implementation in IP Telephony

MIROSLAV VOZNAK, JAN RUDINSKY
CESNET, z.s.p.o.
Zikova 4, Prague 6
CZECH REPUBLIC
miroslav.voznak@vsb.cz, rudinsj@fel.cvut.cz

Abstract: - Supplementary services in traditional telephony were provided by Intelligent Network (IN) concept developed by ITU. The services were introduced in groups starting with Capability Set 1 (ITU-T Q.121x) to CS-4 (Q.124x). An important fact is that the recommendations, instead of providing exact service specification, give only definition of elementary principles that can be used to build services. IP telephony can be considered missing a global mechanism for supplementary service provision. This paper describes proposal, specification and implementation of DDDS (Dynamic Delegation Discovery System) application for supplementary services provision in IP telephony.

Key-Words: - Intelligent Network; Supplementary Services; Next Generation Networks; ENUM

1 Introduction
A concept of Next Generation Network (NGN) was born at ITU as a convergence of traditional telephony with packet networks. NGN represents unified, high-speed packet network that provides number of services with guaranteed quality [1]. IP telephony and IMS architecture for mobile networks can be considered as implementations of NGN concept [2]. The primary goal of any telecommunication network is to provide basic services such as telephony. The network in addition provides supplementary services (e.g. call forwarding, screening, alternative charging, etc.) that increase user comfort and cause intense market competition. Supplementary services in traditional telephony were provided by Intelligent Network (IN) concept developed by ITU. One way to implement the services is by convergence with IN and traditional telephony network. Another more recent activity by IETF is represented by direct implementation of the services by protocol SIP [3]. Both solutions are either complex and require changes to be made at client/server side or implement only a relatively small subset of services. An alternative method can be the utilization of existing IP mechanisms. An observation has been made that number of IN supplementary services essentially convert identifiers of calling parties. Therefore ENUM (E164 Numbering Mapping), as a mechanism that translates telephony and VoIP identifiers, could be used for the service implementation in IP telephony [4].

2 ENUM
ENUM has been proposed as a translation mechanism between E.164 telephony numbers and identifiers used in the Internet (URIs) [5]. It thus enables calls originating in the telephony network to be connected to endpoints located in the Internet. The capability of string translation and route selection based on priority designates ENUM to be a mechanism representing IN services. However analysis of ENUM exposes two constraints. First, the input is represented by a single telephone number in the form according to ITU-T E.164. But there are services that require processing of non-digit input (such as password). Second limitation is the processing of single input. But many services require more than one input parameter. Such an example is Abbreviated dialing, where calling party number is needed in addition to abbreviated number in order to convert it into called party identifier.

3 DDDS Application Design
Because ENUM mechanism is limited in the capability to implement maximum number of IN services, new DDDS application is proposed and specified in this text. ENUM mechanism can be described as an application of Dynamic Delegation Discovery System (DDDS). DDDS application represents an abstract algorithm operating on a database with rewrite rules used by the application for string conversion. In order to design an alternative DDDS application to ENUM several parameters need to be defined: Algorithm, Database and Application specific parameters.

3.1 Algorithm
General DDDS algorithm was already specified in RFC 3402. The service is provided as a string processing defined by the algorithm, depicted on Fig. 1. The input is initially converted into a database search key used later to query the database. The key is then matched to
database records in order to retrieve rewrite rules for input string conversion. In case a rule is not final, it is applied on the initial input and the search is repeated. Once the terminal rule is reached it is applied on the input string to produce output string for further call processing.

3.2 Database
The database contains rewrite rules for string conversion. DDDS specification does not imply any specific database, however a DNS-based hierarchical system has been proposed in RFC 3403. Properties of well known DNS and its scalability make it desirable storage for application rewrite rules. The rules are stored in format of NAPTR resource records.

3.3 Application Specific Parameters
DDDS application implements the algorithm and makes use of selected database. The application has to specify four parameters:
- Unique input
- First well known rule
- Select database
- Output
- Unique input

3.3.1 Unique Input
As the application is aimed at identifier translation following input types encoded in UTF-8 are proposed:
- Non-digit string
- Single E.164 number
- Couple of E.164 numbers

An input string must be unique to identify single search path and it has to include complete information to reach desired output. The unique input is used to create a search key and to apply rewrite rule upon to obtain application output. Input can be divided into fixed and dynamic part. Compulsory fixed part is known prior to service execution, while dynamic part changes with every call. Thus the fixed part can be stored in the database as well as used during the search process. Both fractions need to be separated by a delimiter. It should not be any character that appears neither in fixed nor in dynamic input and it should not play primary role in regular expressions (as it is processed by regexp.). Therefore "&" is selected as delimiter. The unique input has been defined by ABNF:

uniq-input = fixed-part delim-char dyn-part
delim-char = ";"
fixed-part = uri / e164number
dyn-part = uri / e164number
uri = <URL as defined in RFC 2396>
e164number = <E.164 phone number>

3.3.2 First Well-known Rule
Initial FWKR is applied on the input and provides a database search key. The specification of the rule conforms to rewrite-rules and is represented in the form of regular expression below. FWKR represents an association between the unique input fixed part and the operator domain for service provider identification. When applied the rule provides domain name search key for DNS query.

\!^\([^\&]\*)&!\1.OPERATOR\_DOMAIN!

3.3.3 Select Database
As previously mentioned, proposed DDDS application makes use of DNS to store and query the NAPTR records. The records include regular-expression based rewrite-rule set for string translation. These rules applied to the unique input provide either consecutive search key or final output. The use of the rules represents the service provision in one of three forms:
- Translation of involved party identifier
- Verification of identifier presence
- Output selection for routing purpose

Each rule composes of priority, flags, rewrite rule and service parts. Priority identifies the order in which rules are processed and it can be used for output selection to
route the call. Flag indicates, whether a rule is terminal or not and what the following process should be. Rewrite regular expression specifies the translation of string identifier. The regular expression part of NAPTR record for the proposed DDDS application has the ABNF form below. It is composed of two parts. First is regular expression ("ere") that identifies part of the input string intended for subsequent processing. The second part is substitution string ("repl") that contains selected input and an additional part. Application output string is a result of applying the substitution on unique input string.

subst-expr =delim-char ere delim-char repl delim-char
flags
delim-char ="/" / "!" / <char except 'POS-DIGIT' and 'flags'>
ere = <POSIX Extended Regular Expression>
repl = *(string / backref)
string = *(anychar / escapeddelim)
anychar = <any character other than delim-char>
escapeddelim = "\" delim-char
backref = "\" POS-DIGIT
flags = "i"
POS-DIGIT = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"

Finally service field of NAPTR identifies meaning of the record, which has to be unique for every record within a domain (represented by call-party identifier). DDDS application defines two services kinds, where "type" represents a service identifier as specified in ITU recommendation:

- E2E+type = E.164 numbers translation
- E2U+type = E.164 number-to-URI translation

3.3.4 Output
Output of the application represents the unique input processed by the rewrite rules. Depending on the service type result can be either E.164 number or URI. However some services require an information whether an input string is present in the database instead. Therefore an error channel is added. DNS provides three response types (below), where the first type is used for return of E.164 number or URI and name error type is used for service check of string presence.

- Record corresponding to queried name and type
- Name error
- Data not found error

3.3.5 DDDS Application Specification
Specification and Description Language, the graphical representation (SDL/GR) is used for formal specification DDDS application and its behavior. The application is realized by a tree structure composed of "system" as root, "block" as middle layer and "process" as lowest functional element, see Fig. 2. The system represents application interaction with environment through input and output channels. The blocks model functional elements with simplest interface to each other. The processes define system behavior and are composed of Extended Finite State Machines (EFSM).

![Figure 2. Tree structure of application.](image)

The system element makes border between the application and the environment. The application is a stand alone process with start up parameters as input channel and a return value as the output. Part of the system are three blocks. Input processing block checks the input parameters such as service type or connection identifiers. Record look up block creates the unique input, applies the first well known rule onto input to produce a search key and finally queries the rewrite rules from the database. Rule application block then applies the acquired rules on unique input to produce output. The output is checked on format and returned by system. Each block contains three processes, whose interaction defines the system behavior.

4 Implementation
The application is implemented for functionality verification in IP telephony environment [6], [7]. Out of numerous IP telephony software, Asterisk PBX is selected as a test platform for following reasons [8].

![Figure 3. IN-based NP architecture](image)
has many programmable interfaces that can be used for application implementation. BIND name server is chosen as DNS representative in connection with MySQL database as resource record storage. The architecture is shown on the picture. DDDS application is implemented by shell script (Appendix: DDDS application shell script) and it is portable to other platforms. The script is called by SHELL() function in Asterisk configuration file extensions.conf. The function receives input that contains script path and DDDS application input parameters (such as call party identifiers). It returns the output (identifier), where to connect the call or information about accepting or rejecting the call. Here is an example of Free Phone service:

```
exten => _800XXXXXX,1,NoOp(FPH service called, A=${CALLERID(num)}, B=${EXTEN})
exten => _800XXXXXX,n,Set(Result=${SHELL(/opt/asterisk/bin/ddds_application \ E2E+FPH ${CALLERID(num)} ${EXTEN})})
exten => _800XXXXXX,n,GotoIf($"${Result}" != "")?default,${Result},1
exten => _800XXXXXX,n,Macro(incept,1)
```

The format of NAPTR resource record stored by BIND name server and including rewrite rules can have following form:

```
$ORIGIN blue.comtel.cz.
800111112 NAPTR 10 100 "U" "E2E+FPH" "!(^800)(.*)(&.*$)!222\2!" .
```

DDDS application implementation is verified on Free Phone service. The output from Asterisk follows:

```
-- Executing [800111112@default:1]
NoOp("SIP/602336334-084cac8b", "FPH service called, A=602336334, B=800111112") in new stack
-- Executing [800111112@default:2]
Set("SIP/602336334-084cac8b", "Result=222111112") in new stack
-- Executing [800111112@default:3]
Gotol($"SIP/602336334-084cac8b", "1?default,222111112,1") in new stack
-- Goto (default,222111112,1)
```

## 5 Conclusion

It is shown that a number of supplementary services realized in the traditional telephony by Intelligent Network can be transferred to IP telephony. Basically these services process identifiers of calling parties for routing purposes and user verification. This can be realized by ENUM that enables telephony (E.164 numbers) and VoIP (URIs) identifiers manipulation. However limitation of ENUM mechanism leads to proposal of DDDS application capable of implementing more supplementary services, while keeping the advantages of the mechanism. The proposed DDDS application is defined in the form of parameters: unique input, first well known rule, rewrite rules and output. The application process is formally specified using SDL Graphical Representation that defines structural components and behavior. The application functionality is verified by implementation in IP telephony system. DDDS application is build as shell script called by Asterisk PBX connected to BIND name server. A Freephone service validates the functionality of proposed application with positive result.

### Acknowledgement

This research has been supported by the “Optical Network of National Research and Its New Applications” (MSM 6383917201) research intent of the Ministry of Education of the Czech Republic.

### References:

6. Voznak M., Rezac F., The implementation of SPAM over Internet telephony and a defence against this attack, 32nd International Conference on Telecommunications and Signal Processing, AUG 26-27, 2009 Dunakiliti, HUNGARY