ETSI
TECHNICAL
REPORT

ETR 121

February 1994

Source: ETSI TC-NA

Reference: DTR/NA-070303

ICS: 33.080

Key words: UPT, architecture, interworking

Universal Personal Telecommunication (UPT); Architecture and functionalities for interworking

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

Contents

Foreword

This ETSI Technical Report (ETR) has been produced by the Network Aspects (NA) Technical Committee of the European Telecommunications Standards Institute (ETSI).

ETRs are informative documents resulting from ETSI studies which are not appropriate for European Telecommunication Standard (ETS) or Interim European Telecommunication Standard (I-ETS) status. An ETR may be used to publish material which is either of an informative nature, relating to the use or the application of ETSs or I-ETSs, or which is immature and not yet suitable for formal adoption as an ETS or an I-ETS.

This ETR defines the functional architecture and network interconnections for Universal Personal Telecommunication (UPT).

Blank page

1 Scope

This ETSI Technical Report (ETR) defines the functional architecture and network interconnections for Universal Personal Telecommunication (UPT). It gives a high-level description of the network functions required to support UPT, categorizing them into several classes (see Clause 4). It describes the architecture and possible interconnection scenarios and allocates the network functions to the functional entities of the architecture (see Clause 5). The UPT architecture is then applied to several example cases of call handling and service management (see Clause 6).

2 References

For the purposes of this ETR, the following references apply:

[1]	ETR 055-6: "Universal Personal Telecommunication (UPT); The service concept; Part 6: Subscriptions and service profiles".
[2]	ETR 065: "Universal Personal Telecommunication (UPT); Requirements on charging, billing and accounting".
[3]	ETR 066: "Universal Personal Telecommunication (UPT); Requirements on information flows and protocols".
[4]	ETR 067: "Universal Personal Telecommunication (UPT); Network considerations and requirements on dialling, routeing and numbering".
[5]	ETR 083: "Universal Personal Telecommunication (UPT); General UPT security architecture".
[6]	DTR/NA-070306: "Universal Personal Telecommunication (UPT); Management aspects".

3 Symbols and abbreviations

For the purposes of this ETR, the following abbreviations apply:

AP	Access Point
API	Access Point Identity
CCAF	Call Control Agent Function
CCF	Call Control Function
CMISE	Common Management Information Service Element
CS1	Capability Set 1
CS2	Capability Set 2
DDB	Distributed Database
GSM	Global System for Mobile communication
IN	Intelligent Network
O&M	Operations & Maintenance
ODP	Object-oriented Distributed Processing model
PSTN	Public Switched Telephone Network
PUI	Personal User Identity
SCEF	Service Creation Environment Function
SCF	Service Control Function
SCFh	home SCF
SCFo	local ("visited") SCF, originating side
SCFt	local ("visited") SCF, terminating side
SDF	Specialized Database Function
SDFh	home SDF
SDFo	local ("visited") SDF, originating side
SDFt	local ("visited") SDF, terminating side
SI	Service Identity
SMAF	Service Management Agent Function
SMF	Service Management Function
SMFh	home SMF

SMFo	local ("visited") SMF, originating side
SMFt	local ("visited") SMF, terminating side
SRF	Specialized Resources Function
SSF	Service Switching Function
TP	Transport Protocol
UPT	Universal Personal Telecommunication
UPTN	UPT Number

4 UPT network functions

4.1 Introduction

This Clause identifies UPT specific network functions, i.e. network functions required to support UPT which are not provided by current networks. These functions have been grouped into a number of classes, based on the aspect of UPT they are related to:

- functions related to the Access Point (AP);
- functions related to the Personal User Identity (PUI);
- functions related to service provider agreement;
- functions related to authentication;
- functions related to UPT call control;
- functions related to registration/deregistration;
- functions related to service profile management;
- functions related to services;
- functions related to charging;
- Operations & Maintenance (O&M) functions.

Each of these classes is described in a separate subclause (subclauses 4.2 to 4.11, respectively). Within a given class the functions are individually described and, where necessary, an example of a situation where it may be used is given, in order to illustrate the general description.

NOTE 1: The level of detail may vary greatly from one description to another: this fact simply reflects that some functions are easier to detail at this stage, while others will require further study before they can be more precisely described.

The definition of network functions required for UPT relies on some important facts about UPT, which are briefly summarized below:

- a unique UPT Number (UPTN) is assigned to every UPT user, who may render it public. This is the number dialled by other users when they wish to call the UPT user. It can also be used by network entities when they need to send a request to the UPT user's home Specialized Database Function (SDFh);
- every UPT user is also identified by a unique PUI, for security and management purposes. This identity is not public. If no UPT device is used, this identity is known to the UPT user, who will have to supply it every time that UPT user accesses the UPT service. If a UPT device is used, it may store the PUI, in which case the UPT user does not have to memorize it;
- there is a one-to-one relationship between the UPTN and the PUI of a given user;
- a UPT user can register for specific services on specific APs. These APs are identified by Access Point Identities (APIs). The correspondence between Service Identity (SI) and API may vary according to time of the day and other parameters;

NOTE 2: A routeing address can be derived from an API.

- the characteristics of an AP have an impact on the services which may be supplied on it. Therefore, it may be be useful to store the characteristics of APs (i.e. their terminal/network capabilities);

- some services may have specific requirements on the capabilities of the AP on which they are supplied. Therefore, it may also be useful to store the relationship between services and the capabilities required to supply them.

4.2 Class "access point"

The functions listed in this subclause are related to the APs used for access to the UPT service. These functions allow the input/output of an API from/to the user; they allow the verification of an API input from the user and the conversion of an API to a routeing address; they allow the storage of an association between an API and a service and the storage of an association between an API and a set of capabilities.

Request API: function to request and input an API from the UPT user. For example, this function can be used during registration procedures (InCall registration, OutCall registration or AllCall registration) to request the identity of the AP on which the user wishes to register.

Determine API: function to determine the API of the AP currently used.

Provide API: function to provide an API to the user. This function may be used by interrogation procedures to provide the user with the identity of the AP(s) that the user has associated to a given service.

Verify API: function to verify an API specified explicitly by the user. This verification mainly asserts that the format of the API given by the user is acceptable from the point of view of the network where the verification takes place. It is not, in general, a complete verification as the API may belong to another network, in which case there may not be sufficient information available to perform a complete verification.

Determine routeing address: function to derive a routeing address from an API.

Store API/routeing address: function to store an API/routeing address in connection with a PUI and a specific service. For example, this function may be used at the time when a service is activated by the user, to store the identity of the AP on which the service is to be provided.

Retrieve API/routeing address: function to retrieve a stored API/routeing address. This function may be used by service logic when the service is actually in action, to retrieve the identity of the AP on which the user requested the service to be provided. It may also be used by interrogation procedures if the user wishes to be reminded of the API that is associated with a given service.

Store access point capabilities: function to store relevant capability information on an AP of the visited network. This function may be used in conjunction with **store API/routeing address** to store additional information about the access which is to be used for a given service. This is subject to the ability of the network to supply such information, and is for further study.

Retrieve access point capabilities: function to retrieve stored capability information for an AP. This function may be used by service logic to determine the capabilities of the AP on which the service is to be supplied, before attempting to match them against the capabilities required by the service.

Store network capabilities: function to store capability information for the APs of a given network.

Retrieve network capabilities: function to retrieve capability information for the APs of a network. This function is used when the capabilities of a particular AP need to be determined. This happens when a UPT user registers to an AP; the UPT service needs to know about the capabilities of the AP to adapt the services it supplies.

4.3 Class "PUI"

The functions described in this subclause are related to the PUI of a UPT user. They allow the retrieval of a user's PUI (from that user's UPTN, or by querying it) as well as its verification. These functions are mostly used when access to the UPT service is requested by the user.

Determine PUI: function to derive a PUI from a UPTN. This function may, for example, be used when an incoming UPT call is to be handled by the UPT logic. The calling party dials the UPTN, whereas the service logic needs the user's PUI. This function provides the required mapping between the two.

Request PUI: function to request a PUI from the UPT user or from the user's UPT device. For example, all requests for UPT service from an unregistered terminal will start with this function, as the user cannot be identified until the UPT user's PUI is known.

Verify PUI: function to check if a PUI is still valid, if it is blacklisted etc. This function can be used immediately after the **request PUI** function, to check whether or not the UPT user is allowed access to the service.

4.4 Class "agreements"

The functions listed in this subclause are related to agreements between service providers and network operators. They allow the determination of a given UPT user's home service provider, of the originating network of an incoming call and the interrogation of agreements passed between operators and service providers.

Determine service provider identity: function to derive the identity of a user's home service provider from the UPT user's UPTN or PUI. This identity is required when checking for an agreement between the visited network operator and the home service provider (see the **check service provider agreement** function).

Check service provider agreement: function to verify whether there is an agreement between the UPT user's currently visited network operator (where this function is used) and the UPT user's home service provider. This function may be used when a UPT user attempts to place an outgoing call, to assess whether or not the UPT user should be allowed to do so.

Determine calling network identity: function to derive a network operator identity from a calling party address. This identity is required when checking for an agreement between the operator of the network where the request for UPT service originates and the corresponding UPT service provider (see next function).

Check network operator agreement: function to verify whether there is an agreement between the UPT user's home service provider (which uses this function) and the network from which a request for UPT service has been received.

The two functions **determine service provider identity** and **check service provider agreement** work in tandem, as do the two functions **determine calling network identity** and **check network operator agreement**. The two pairs of functions are duals of each other, the first one operating from the point of view of the visited network operator, and the second one operating from the point of view of the home service provider.

4.5 Class "authentication"

The functions described in this subclause are related to the authentication of the UPT user when requesting service. They allow storage, retrieval, input and validation of authentication information.

Store authentication information: function to store authentication information related to a UPT user.

Retrieve authentication information: function to retrieve authentication information related to a UPT user.

Request authentication information: function to request and input authentication information from the UPT user and/or the UPT user's UPT device. For example, all requests for UPT service from an unregistered terminal will use this function together with the **request PUI** function, as UPT service will not be provided to a user until the user is both identified and authenticated. The general principles and methods used for authentication in UPT are specified in ETR 083 [5].

Authenticate: function to validate the authentication information supplied by the UPT user. This information can be obtained by means of the **request authentication information** function.

4.6 Class "call control"

The functions listed in this subclause are related to UPT call handling. They allow the recognition, processing, routeing, indication and negotiation of UPT calls. They have been divided into two categories:

- the functions dealing with incoming UPT calls; and
- the functions dealing with outgoing UPT calls.

4.6.1 UPT incoming calls

Recognize UPT incoming call: function to recognize that a call is directed to a UPT user. This function needs to be provided by low-level network call handling mechanisms. Its role is to trigger the following high-level function:

> **process UPT incoming call:** function to provide service logic for handling a UPT incoming call: locate the UPT user; retrieve service profile information; check services to be supplied against the capabilities available; supply available services; etc. This function uses the functions related to services described in subclause 4.9. When it is done, it triggers the following low-level function:
>
> > **route UPT incoming call:** function to route a call to a UPT user. This function uses the routeing address supplied by the **process UPT incoming call** function to route the call to the UPT user. Another UPT specific aspect of this routeing function is that it may add an indication that the call to be routed is a UPT call in the signalling messages it sends.

Indicate UPT incoming call: function to indicate to the called UPT subscriber that an incoming UPT call is present.

Negotiate UPT incoming call: function to negotiate with the UPT user acceptance of an incoming UPT call, and to handle the response. If the corresponding supplementary services are activated, this function may involve the presentation of the identity of the calling party, the authentication of the calling party and/or of the called party, the indication of charging rates for the call and the negotiation of the supplementary services to be provided for the call, if restrictions are imposed by the capabilities of the AP.

4.6.2 UPT outgoing calls

Recognize UPT outgoing call: function to recognize a UPT outgoing call request. This function may have to be provided by low-level network call handling mechanisms, in the case where a specific dialling prefix or number is defined for UPT outgoing calls. Its role is to trigger the following high-level function:

> **process UPT outgoing call:** function to provide service logic for handling a UPT outgoing call: retrieve service profile information; check services to be supplied against the capabilities available; supply actual services; etc. This function uses the functions related to services described in subclause 4.9.
>
> NOTE: There is no need for a function to route a UPT outgoing call, as the normal network mechanism is used.

4.7 Class "registration"

The functions listed in this subclause are related to UPT registration and deregistration. They allow the recognition and handling of registration and deregistration requests.

Recognize registration/deregistration: function to recognize a UPT registration or deregistration. This function may have to be provided by low-level network call handling mechanisms, in the case where a specific dialling prefix is defined for UPT registration or deregistration. Its role is to trigger the following high-level function:

> **process registration/deregistration:** function to provide service logic for handling UPT registrations and deregistrations: retrieve service profile information; perform housekeeping (time of day dependant routeing, etc.); activate default registration upon deregistration; etc. This function uses the functions related to services described in subclause 4.9.

4.8 Class "service profile"

The functions listed in this subclause are related to service profile management by the user. They allow the recognition and handling of service profile management requests, including functions to retrieve, modify and store the service profile.

Recognize service profile management: function to recognize a service profile management request from the user. This function may have to be provided by low-level network call handling mechanisms, in the case where a specific dialling prefix or number is defined for UPT service profile management. Its role is to trigger the following high-level function:

> **process service profile management:** function to provide service logic for handling service profile management requests from the user: retrieve service profile information; check that user is authorized to edit it; edit it; and store it back. This function uses the following functions:
>
> > **store service profile:** function to store the whole service profile of a given UPT user;
> >
> > **retrieve service profile:** function to retrieve the whole service profile of a given UPT user;
> >
> > **edit service profile:** function to interact with the user, for editing the user's service profile;
> >
> > **update service profile:** function to maintain up to date copies of the service profile which may be stored outside the home network, in a distributed scenario.

4.9 Class "service"

The functions listed in this subclause are related to the provision of UPT services. They allow the recognition of a request for UPT service, the recognition of the particular UPT service requested, and the verification of the user's profile and AP capabilities, with respect to that service.

Recognize UPT request: function to recognize a request for UPT service. This function needs to be provided by low-level network call handling mechanisms. Its role is to trigger the following high-level function:

> **recognize UPT service:** function to interact with the UPT user, to identify the type of service requested (outgoing call, registration, service management, etc.) and to run its service logic.

Retrieve service information: function to retrieve information on a specific service for a given UPT user.

Check service against profile: function to check the service requested against the corresponding information in the UPT user's service profile. If the user has not subscribed to the service, it cannot be provided.

Check service against capabilities: function to check the capabilities needed by the requested service against the capabilities of the user's current AP(s).

4.10 Class "charging"

The functions listed in this subclause are related to the charging process for the UPT service. They allow the collection of charging information, the transfer of this information between operators and service providers, and the provision of charging information to the UPT user.

Charge: this function covers the whole charging process, which includes the recording of charging information during the call, and the storage of this information after the end of the call. This complex function may need to be split into simpler functions (e.g. **charge determination, charge generation, charge registration**), but this has no impact on UPT architecture. More information can be found in ETR 065 [2].

Check credit against limits: function to check if the credit used by the user exceeds the amount authorized by the service provider. If the authorized credit has been exceeded, access to the UPT service is forbidden until some administrative action takes place. This credit checking may take place off-line, at regular time intervals, or in real-time, after each UPT call.

Transfer charging information: function to transfer collected charging information between network operators and service providers.

Provide charging information to user: function to indicate to the UPT user the cost of a call, either in real-time (i.e. during the call), or off-line (i.e. after the call is completed).

4.11 Class "O&M"

Functions related to operations and management aspects of the UPT service are described in DTR/NA-70306 [6].

5 UPT architecture and interworking

5.1 UPT functional architecture

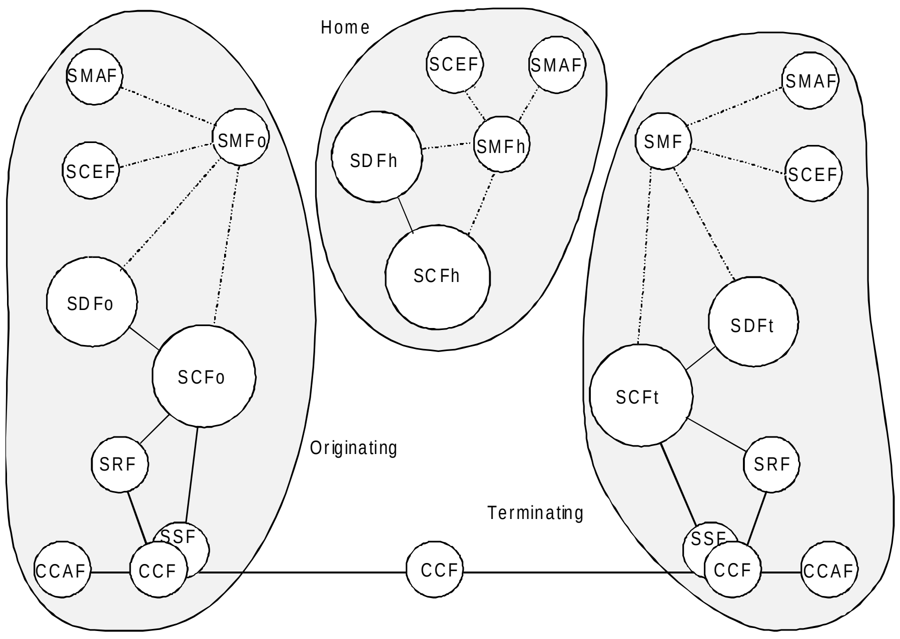

Figure 1: General UPT functional architecture

The functional architecture for UPT is given in figure 1. It is based on the standard Intelligent Network (IN) architecture. Apart from standard IN terminology, the following notations are used on the figure:

-	**SCFh**	home Service Control Function (SCF);
-	**SDFh**	home SDF;
-	**SMFh**	home Service Management Function (SMF);
-	**SCFo**	local ("visited") SCF, originating side;
-	**SDFo**	local ("visited") SDF, originating side;
-	**SMFo**	local ("visited") SMF, originating side;
-	**SCFt**	local ("visited") SCF, terminating side;
-	**SDFt**	local ("visited") SDF, terminating side;
-	**SMFt**	local ("visited") SMF, terminating side.

5.2 Interconnection scenarios

There are four possible scenarios for the interconnection between the networks of the UPT functional model:

- SDF - SDF;
- SCF - SDF;
- SCF - SCF;
- SCF - SCF + SDF - SDF.

These scenarios are described and discussed in this subclause.

5.2.1 SCF-SDF interconnection

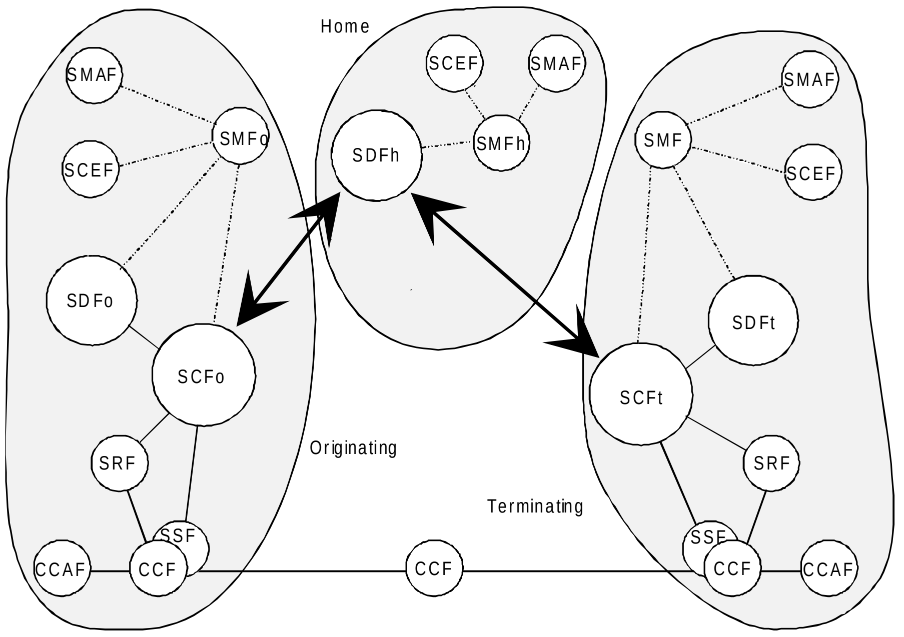

Figure 2: SCF-SDF interconnection scenario

This scenario is illustrated by figure 2. The interconnection takes place between SCFo and SDFh, and between SCFt and SDFh. The following consequences can be derived:

- the interconnection of networks takes place between SCF and SDF functional entities, as indicated by the arrows on the figure. The interface between SCF and SDF is specified in IN Capability Set 1 (CS1). Therefore, scenario can be used in the short term (UPT phase 1);

- no transparency to the access of service data is offered to the service logic. If SCFo requires "local" data, it needs to send a request to SDFo; but if "home" data is required, the request needs to be sent to SDFh;

- the interconnection of networks takes place over only one interface;

- functions related to the management of UPT data are spread over the SCF and the SDF entities. A change in the principles used may have an impact on functional entities and interfaces;

- this scenario does not lend itself well to the distribution of UPT data in the originating and terminating networks, since there are no direct links between the SDFs. For instance, if data were to be distributed from SDFh to SDFo, it would have to first transit through SCFo. Also, the SCF-SDF interface would have to be extended, since the request for updating data stored in SDFo would flow from the SDF to the SCF (only requests from SCF to SDF are currently specified). Therefore, this scenario is rather oriented towards a centralized approach of the management of UPT data (SDFh stores all data related to the UPT user);

- this scenario does not allow any distribution of the service logic to take place between the originating, home and terminating networks. SCFo always has full control of the UPT call, precluding complex services which require the co-operation of several SCFs;

- SDFh is more than a simple data repository. It shall contain a set of data management and security-related functions:

    - **access control:** this control can be based on the identity of the source of the request (an SCF entity from another network) and/or other security criteria;

    - **UPT user authentication:** SDFh performs the authentication of the UPT user;

- SDFo stores a list of agreements, which indicates the identity of all the service providers whose subscribers are allowed to access UPT service in SDFo's network;

- SDFo stores a list of service limitations resulting from agreements with service providers or network limitations;

- SDFo also stores information related to the management of the UPT service in its network, e.g. charging records which will be used later for accounting;

- the service profile of visiting UPT users is stored only in their home networks (SDFh). SDFo does not store a copy of it.

This scenario suffers from some defects, but seems attractive for the short term, since it only requires features specified in IN CS1 and uses very simple principles (centralized data management and centralized service logic).

5.2.2 SCF-SCF interconnection

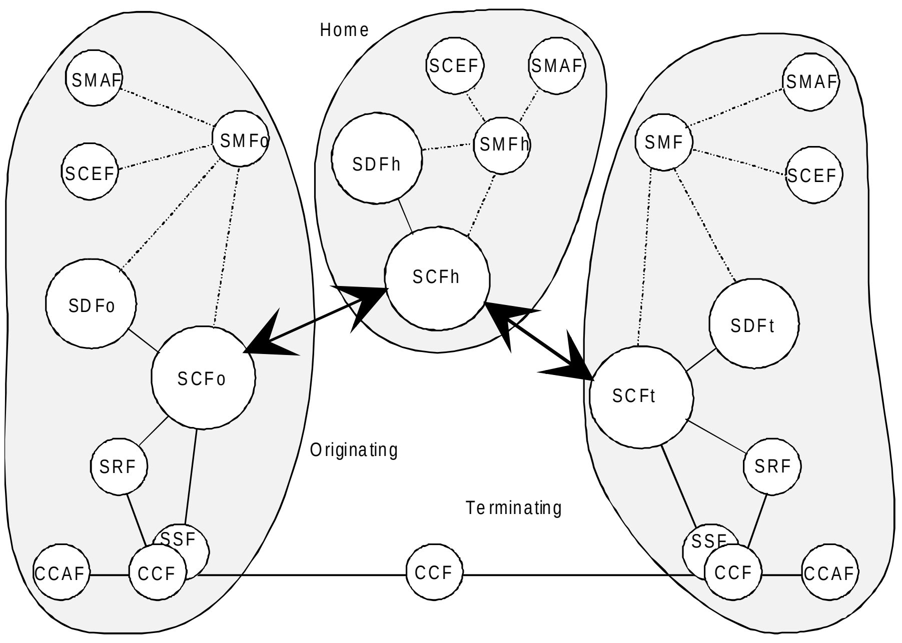

Figure 3: SCF-SCF interconnection scenario

This scenario is illustrated by figure 3. The interconnection takes place between SCFo and SCFh, and between SCFt and SCFh. The following consequences can be derived:

- the interconnection of networks takes place between SCF and SCF functional entities, as indicated by the arrows on the figure. The interface between SCF and SCF is not specified in IN CS1. Therefore, this scenario cannot be used in the short term (UPT phase 1) and will have an impact on the specification of IN Capability Set 2 (CS2) and beyond, if chosen;

- no transparency to the access of service data is offered to the service logic. If SCFo requires "local" data, it needs to send a request to SDFo; but if "home" data is required, the request needs to be sent to SCFh. This may induce extra complexity in the service logic;

- the interconnection of networks takes place over only one interface;

- functions related to the management of UPT data are spread over the SCF and the SDF entities in all networks involved. A change in the principles used for managing UPT data may have an impact on functional entities and interfaces;

- this scenario does not lend itself well to the distribution of UPT data in the originating and terminating networks, since there are no direct links between the SDFs (this is the same as in the case of the SCF-SDF scenario). For instance, if data were to be distributed from SDFh to SDFo, it would first have to transit through both SCFh and SCFo, which, depending upon the physical implementation, would require extra processing and, possibly, extra signalling. Therefore, this scenario is strongly oriented towards a centralized approach of the management of UPT data (SDFh stores all data related to the UPT user);

- this scenario lends itself well to the distribution of UPT service logic between the originating, home and terminating networks. Because the interconnection of the networks takes place over an SCF-SCF link, it is possible for SCFo to invoke service logic contained in SCFh and/or SCFt. This distribution of service logic enables the provision of complex services (which require the co-operation of several SCFs);

- data exchanges between the originating and the home network (for instance, when SCFo issues a request for data stored in SDFh) need to transit through SCFh;

- SDFh may be a simple data repository. Most functions related to the management of UPT data (access control, etc.) are provided by SCFh, and therefore allow for a very simple SDFh;

- SDFo stores a list of agreements, which indicates the identity of all the service providers whose subscribers are allowed to access UPT service in SDFo's network;

- SDFo stores a list of service limitations resulting from agreements with service providers or network limitations;

- SDFo also stores information related to the management of the UPT service in its network, e.g. charging records which will be used later for accounting;

- the service profile of visiting UPT users is only stored in their home network (SDFh). SDFo does not store a copy of it.

This scenario seems an attractive solution for the long term, if distribution of service logic is seen as a prime requisite, while distribution of data is not. It cannot, however, be used in the short term, since it requires an interface which is not specified in IN CS1.

5.2.3 SDF-SDF interconnection

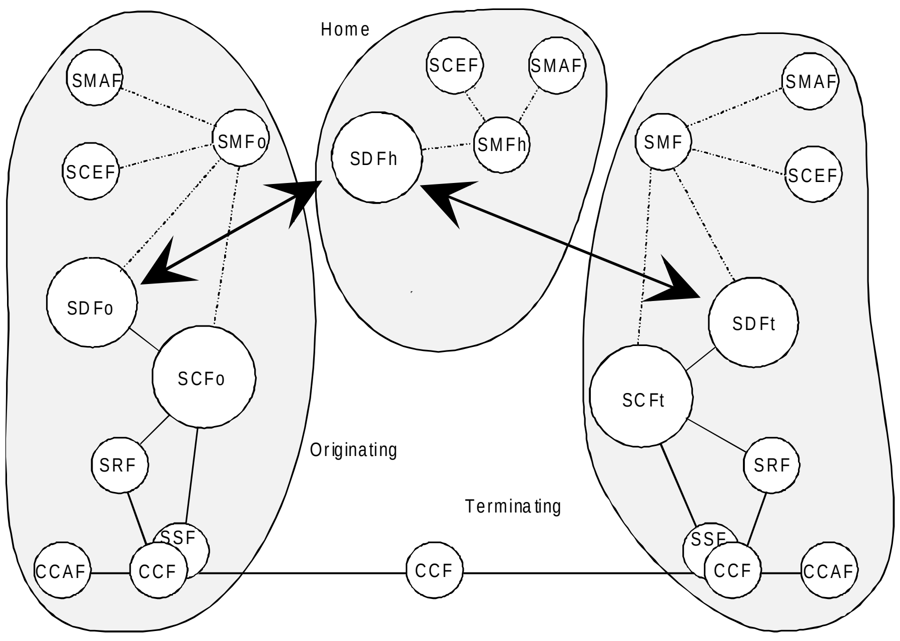

Figure 4: SDF-SDF interconnection scenario

This scenario is illustrated by figure 4. The interconnection takes place between SDFo and SDFh, and between SDFh and SDFt. The following consequences can be derived:

- the interconnection of networks takes place between SDF functional entities, as indicated by the arrows on the figure. The interface between SDF and SDF is not specified in IN CS1. Therefore, this scenario cannot be used in the short term (UPT phase 1) and will have an impact on the specification of IN CS2 and beyond, if chosen;

- the service logic may be offered transparently to the access of service data. Independant of where the data is located, SCFo will query it from SDFo and likewise, SCFt will query from SDFt. The SDFo and SDFt are then responsible for locating the data requested by the SCF entities, which may be in another network. Such a separation extends the ideas of the IN model to a distributed environment: the logic of the service is decoupled from the data for the service;

- this scenario does not allow any distribution of the service logic to take place between the originating, home and terminating networks. SCFo always has full control of the UPT call, precluding complex services which require the co-operation of several SCFs;

- the interconnection of networks takes place over only one interface;

- the SCFh entity of the general UPT model is not used. All "home" functionalities (storage of subscriber information and user profile, authentication algorithms and keys, etc.) are grouped in SDFh;

- all the functions related to the management of UPT data are in the SDF entity, which could be given a modular structure. If necessary, this structure can then be modified without any impact on other functional entities and interfaces;

- SDFh is more than a simple data repository. It shall contain a set of data management and security-related functions:

  - **access control:** this control can be based on the identity of the source of the request (an SDF entity from another network) and/or other security criteria;

  - **data integrity:** if data stored in SDFh is also stored in SDFo and/or SDFt (shadowing technique) and an update is performed in SDFh, the data stored in SDFo/SDFt also needs to be updated. Hence, all data transfers need to be registered. The SDFh has a co-ordination role over SDFo and SDFt, which act as local storage of data maintained by SDFh.

This scenario seems an attractive solution for the long term, if distribution of UPT data is seen as a prime requisite, while distribution of service logic is not. It cannot, however, be used in the short term, since it requires an interface which is not specified in IN CS1.

5.2.4 SCF-SCF + SDF-SDF interconnection

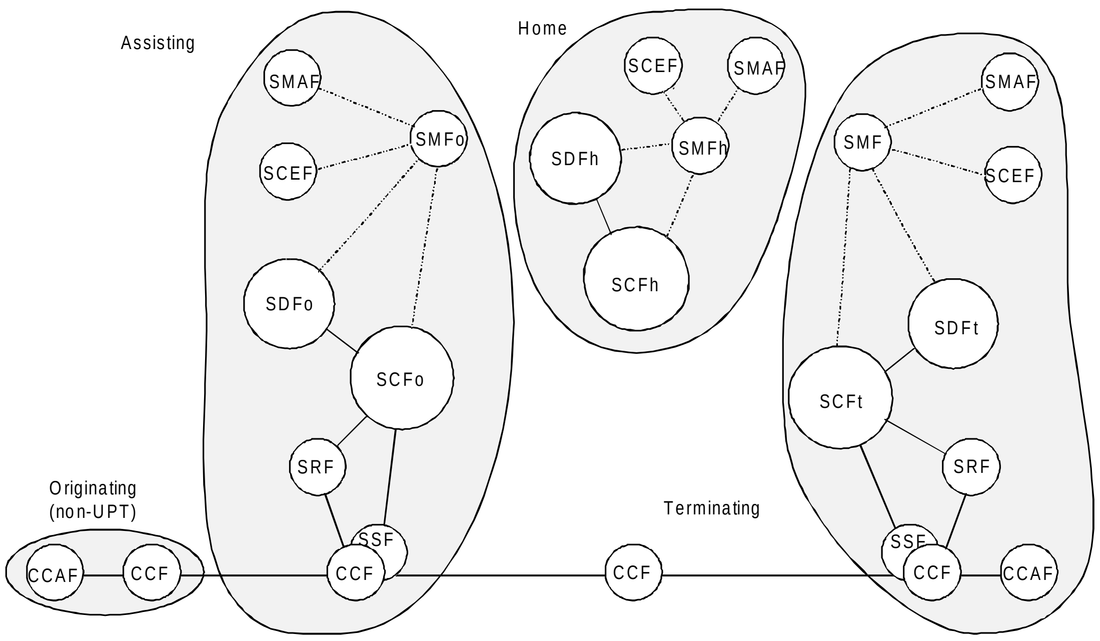

Figure 5: SCF-SCF + SDF-SDF interconnection scenario

This scenario is illustrated by figure 5. The interconnection takes place between SCFo and SCFh and between SCFt and SCFh, as well as between SDFo and SDFh and between SDFt and SDFh. The following consequences can be derived:

- the interconnection of networks takes place between SCF/SCF and SDF/SDF links, as indicated by the arrows on the figure. Neither of these interfaces are specified in IN CS1. Therefore, this scenario cannot be used in the short term (UPT phase 1) and will have an impact on the specification of IN CS2 and beyond, if chosen;

- the service logic may be offered transparently to the access of service data. Independent of where the data is located SCFo will query it from SDFo, and, likewise, SCFt will query from SDFt. The SDFo and SDFt are then responsible for locating the data requested by the SCF entities, which may be in another network. Such a separation extends the ideas of the IN model to a distributed environment: the logic of the service is decoupled from the data for the service;

- this scenario lends itself well to the distribution of UPT service logic between the originating, home and terminating networks. Because the interconnection of the networks takes place over an SCF-SCF link, it is possible for SCFo to invoke service logic contained in SCFh and/or SCFt. This distribution of service logic enables the provision of complex services (which require the co-operation of several SCFs);

- the interconnection of networks takes place over two interfaces. Therefore, this scenario is more complex than the three previous ones. However, it may be seen as the superposition of the SCF-SCF and of the SDF-SDF scenarios: the SCF-SCF interface is used to provide distribution of service logic and operates independently of the SDF-SDF interface, which is used to provide distribution of data. In fact, evolution from either the SCF-SCF or the SDF-SDF scenario to the SCF-SCF + SDF-SDF one can be achieved while preserving backward compatibility;

- both the SCFh and SDFh entities of the general UPT model are used. "Home" functionalities are distributed over both entities;

- all the functions related to the management of UPT data are in the SDF entity, which could be given a modular structure. If necessary, this structure can then be modified without any impact on other functional entities and interfaces;

- SDFo stores a list of agreements, which indicates the identity of all the service providers whose subscribers are allowed to access UPT service in SDFo's network;

- SDFo stores a list of service limitations resulting from agreements with service providers or network limitations;

- SDFo also stores information related to the management of the UPT service in its network, e.g. charging records which will be used later for accounting.

While being the most complex, this scenario seems the most general solution for the long term, since it allows for the distribution of both the service data and the service logic. It cannot, however, be used in the short term, since it requires an interface which is not specified in IN CS1.

5.2.5 Evolutionary path

As a conclusion to the description of the interconnection scenarios, an evolutionary path may be roughly sketched:

- SCF-SDF scenario for the short term (UPT phase 1), offering only a centralized approach to both service logic and data handling;

- SCF-SCF or SDF-SDF scenario for the middle term, depending on which of the two types of distribution (service logic or data) is seen as the most useful for this timescale;

- SCF-SCF + SDF-SDF scenario for the long term, offering full distribution of service logic and data, and thereby allowing complex services to be offered in an optimal way.

5.3 Allocation of network functions to functional entities

Based on the general functional architecture for UPT, this subclause attempts to allocate the UPT network functions described in Clause 4 to the functional entities: Call Control Function (CCF); Service Switching Function (SSF); SCFo; SDFo; SDFh; SMFo; SMFh; and Call Control Agent Function (CCAF). Some functions require the co-operation of several entities or may be provided by different entities. This situation is always explicitly described in a note which states more precisely what is meant by this multiple allocation. Only three of the scenarios described in the previous subclause have been considered so far: SDF-SDF; SCF-SDF; and SCF-SCF.

Table 1: CCF

Class	**Function**	**Functional Entity**		
		SDF-SDF scenario	**SCF-SDF scenario**	**SCF-SCF scenario**
Call control	Route UPT incoming call	CCF	CCF	CCF
Charging and accounting	Charge	CCF (NOTE 2)	CCF (NOTE 2)	CCF (NOTE 2)
	Provide charging information to user	CCF / SCFo / SMFh (NOTE 1)	CCF / SCFo / SMFh (NOTE 1)	CCF / SCFo / SMFh (NOTE 1)

NOTE 1: This function may be implemented by any three functional entities, depending on the type of information provided to the user. Real-time charging information can be provided by the CCF (by emitting a pulse every time a charging unit is spent) or by the SCFo (by emitting voice messages to indicate specific charging conditions). Off-line charging information can be provided by the SMFh (by listing the costs incurred so far since the last bill, etc.).

NOTE 2: The charging information produced by the CCF entity may be transferred to SMFo and/or SMFh if no credit limit checking is to be performed or if only off-line credit limit checking is required. If real-time credit limit checking is required, this information alsoneeds to be transferred to SDFh. Also, as indicated in subclause 4.10, this function may be split further to refine the description of the charging process. Several allocation scenarios are then possible, but they have no impact on the allocation of UPT functions. See ETR 055-6 [1] for more information.

Table 2: SSF

Class	**Function**	**Functional Entity**		
		SDF-SDF scenario	SCF-SDF scenario	SCF-SCF scenario
Access point	Determine API	SSF	SSF	SSF
Call control	Recognize UPT incoming call	SSF + SCFo (NOTE 1)	SSF + SCFo (NOTE 1)	SSF + SCFo (NOTE 1)
	Recognize UPT outgoing call	SSF + SCFo (NOTE 1)	SSF + SCFo (NOTE 1)	SSF + SCFo (NOTE 1)
Registration	Recognize registration / deregistration	SSF + SCFo (NOTE 1)	SSF + SCFo (NOTE 1)	SSF + SCFo (NOTE 1)
Service profile	Recognize service profile management	SSF + SCFo (NOTE 1)	SSF + SCFo (NOTE 1)	SSF + SCFo (NOTE 1)
Services	Recognize UPT request	SSF (NOTE 2)	SSF (NOTE 2)	SSF (NOTE 2)

NOTE 1: These functions are implemented by the two functional entities, working in collaboration: the SSF triggers on a specific prefix or number and sends a Provide Instruction request to the SCFo, which then determines which service to activate (incoming call, outgoing call, etc.), depending on the prefix or number used.

NOTE 2: This function only recognizes the fact that a request for UPT service exists. The exact nature of the request is determined later by the SCFo, using the **recognize UPT service** function, which interacts with the user to determine which service is requested. Therefore, only the SSF is involved.

Table 3: SCFo

Class	Function	Functional Entity		
		SDF-SDF scenario	SCF-SDF scenario	SCF-SCF scenario
Access point	Request API	SCFo	SCFo	SCFo
	Provide API	SCFo	SCFo	SCFo
	Verify API	SCFo	SCFo	SCFo + any other SCF (NOTE 6)
	Determine routeing address	SCFo	SCFo	SCFo + SCFh + SCFt (NOTE 7)
PUI	Request PUI	SCFo	SCFo	SCFo / SCFt (NOTE 8)
Authentication	Request authentication information	SCFo	SCFo	SCFo / SCFt (NOTE 8)
	Authenticate	SCFo + SDFo/SDFh (NOTE 5)	SCFo + SDFh (NOTE 1)	SCFo + SDFh (NOTE 1)
Call control	Recognize UPT incoming call	SSF + SCFo (NOTE 2)	SSF + SCFo (NOTE 2)	SSF + SCFo (NOTE 2)
	Process UPT incoming call	SCFo	SCFo	SCFo + SCFh + SCFt (NOTE 7)
	Negotiate UPT incoming call	SCFo	SCFo	SCFo + SCFh + SCFt (NOTE 7)
	Recognize UPT outgoing call	SSF + SCFo (NOTE 2)	SSF + SCFo (NOTE 2)	SSF + SCFo (NOTE 2)
	Process UPT outgoing call	SCFo	SCFo	SCFo + SCFh (NOTE 7)
Registration	Recognize registration/deregistration	SSF + SCFo (NOTE 2)	SSF + SCFo (NOTE 2)	SSF + SCFo (NOTE 2)
	Process registration/deregistration	SCFo	SCFo	SCFo + SCFh (NOTE 7)
Service profile	Recognize service profile management	SSF + SCFo (NOTE 2)	SSF + SCFo (NOTE 2)	SSF + SCFo (NOTE 2)
	Process service profile management	SCFo / SMFh (NOTE 3)	SCFo / SMFh (NOTE 3)	SCFo + SCFh (NOTE 7)
	Edit service profile	SCFo / SMFh (NOTE 3)	SCFo / SMFh (NOTE 3)	SCFo / SMFh (NOTE 3)
Services	Recognize UPT service	SCFo	SCFo	SCFo
Charging	Provide charging information to user	CCF / SCFo / SMFh (NOTE 4)	CCF / SCFo / SMFh (NOTE 4)	CCF / SCFo / SMFh (NOTE 4)

(continued)

Table 3 (concluded): SCFo

NOTES to table 3.

NOTE 1: The authentication of the UPT user is performed by SDFh, upon request from SCFo.

NOTE 2: See NOTE 1 of table 2.

NOTE 3: The user has two methods to perform service profile management:

- interact with the same terminal used for making and receiving calls. The exchange of information related to the UPT procedure flows through the CCAF, CCF, SSF, SCF and SMF entities, in this order. Due to limitations imposed by the terminal this method cannot be used for the more complex management procedures;

- interact with a terminal dedicated to service management (the Service Management Agent Function (SMAF)). The exchange of information related to the UPT procedure then flows through the SMAF, SMF, SCF and SSF entities, in this order. For technological reasons, this method is in general more powerful, more flexible and more user-friendly than the previous one. However, it may not be available to all users at all times, since it requires access to some specific equipment.

Therefore, both SCFo and SMFh will need to have the corresponding functions.

NOTE 4: See NOTE 1 of table 1.

NOTE 5: The authentication of the UPT user is performed by SDFo or SDFh, upon request from SCFo. Which of the two SDFs does the authentication depends on the degree of distribution of the information.

NOTE 6: In this scenario, a more complete verification of the API may be performed, since SCFo may send a verification request to a SCF in the API's network.

NOTE 7: This function is normally performed by SCFo, but extra service logic may be supplied by SCFh and/or SCFt, if specific services have been activated by the UPT user.

NOTE 8: This function is performed by SCFo when interaction with the originating user is required, and by SCFt when interaction with the terminating user is required.

Table 4: SDFo & SDFh

Class	Function	Functional Entity		
		SDF-SDF scenario	**SCF-SDF scenario**	**SCF-SCF scenario**
Access point	Store API/routeing address	SDFo / SDFh	SDFh	SDFh
	Retrieve API/routeing address	SDFo / SDFh	SDFh	SDFh
	Store access point capabilities	SDFo / SDFh	SDFh	SDFh
	Retrieve access point capabilities	SDFo / SDFh	SDFh	SDFh
	Store network capabilities	SDFo	SDFo	SDFo
	Retrieve network capabilities	SDFo	SDFo	SDFo
PUI	Determine PUI	SDFh	SDFh	SDFh
	Verify PUI	SDFh	SDFh	SDFh
Agreements	Determine service provider identity	SDFo	SDFo	SDFo
	Check service provider agreement	SDFo	SDFo	SDFo
	Determine calling network identity	SDFh	SDFh	SDFh
	Check network operator agreement	SDFh	SDFh	SDFh
Authentication	Store authentication information	SDFo / SDFh	SDFh	SDFh
	Retrieve authentication information	SDFo / SDFh	SDFh	SDFh
	Authenticate	SCFo + SDFo/SDFh (NOTE 2)	SCFo + SDFh (NOTE 1)	SCFo + SDFh (NOTE 1)
Service profile	Store service profile	SDFo / SDFh	SDFh	SDFh
	Retrieve service profile	SDFo / SDFh	SDFh	SDFh
	Update service profile	SDFo + SDFh	N/A	SDFh
Services	Retrieve service information	SDFo / SDFh	SDFh	SDFh
	Check service against profile	SCFo	SDFh	SDFh
	Check service against capabilities	SCFo	SDFh	SDFh
Charging	Check credit against limits	SDFh / SMFh (NOTE 3)	SDFh / SMFh (NOTE 3)	SDFh / SMFh (NOTE 3)

NOTE 1: See NOTE 1 of table 3.

NOTE 2: See NOTE 5 of table 3.

NOTE 3: This function may be implemented by two functional entities, depending on the type of credit limit checking required. If real-time checking is needed, then SDFh performs the checking. If off-line credit checking is needed, then SMFh performs the checking.

Table 5: SMFo & SMFh

Class	Function	Functional Entity		
		SDF-SDF scenario	**SCF-SDF scenario**	**SCF-SCF scenario**
Service profile	Process service profile management	SCFo / SMFh (NOTE 1)	SCFo / SMFh (NOTE 1)	SCFo / SMFh (NOTE 1)
	Edit service profile	SCFo / SMFh (NOTE 1)	SCFo / SMFh (NOTE 1)	SCFo / SMFh (NOTE 1)
Charging	Check credit against limits	SDFh / SMFh (NOTE 3)	SDFh / SMFh (NOTE 3)	SDFh / SMFh (NOTE 3)
	Transfer charging information	SMFo + SMFh	SMFo + SMFh	SMFo + SMFh
	Provide charging information to user	CCF / SCFo / SMFh (NOTE 2)	CCF / SCFo / SMFh (NOTE 2)	CCF / SCFo / SMFh (NOTE 2)
NOTE 1: See NOTE 3 of table 3. NOTE 2: See NOTE 1 of table 1. NOTE 3: See NOTE 3 of table 4.				

Table 6: CCAF

Class	Function	Functional Entity		
		SDF-SDF scenario	**SCF-SDF scenario**	**SCF-SCF scenario**
Call control	Indicate UPT incoming call	CCAF	CCAF	CCAF

Others functions:

- no UPT-specific functions have yet been identified for SCFt, SDFt, SMFt, Specialized Resources Function (SRF), Service Creation Environment Function (SCEF) and SMAF.

5.4 Interworking with networks without UPT functionality

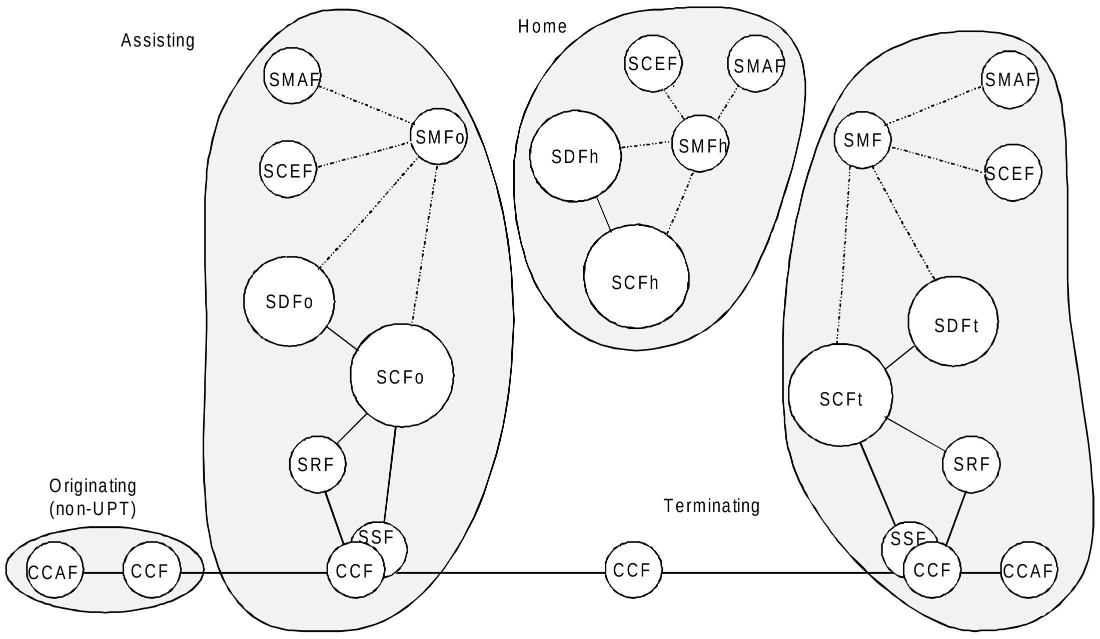

Figure 6: Interworking with networks without UPT functionality

Although the ultimate aim is to provide UPT functionalities in all networks, it is recognized that this will not be feasible in the near future, and especially not in the short term. UPT networks (i.e. networks which provide UPT functionalities) and networks without UPT functionality will, therefore, coexist for some time. On the other hand, even in the short term, user mobility should not be restricted to a small set of countries (for example, those which have an IN-structured network) or to certain types of terminals (for example, Public Switched Telephone Network (PSTN) terminals but not Global System for Mobile communication (GSM) terminals).

Therefore, some networks will not be able to offer full UPT support, but it may still be desirable that UPT users be able to use part of the functionalities of UPT when they visit these networks. This is possible, if the aid of another network with UPT capabilities (the assisting network) is sought. The corresponding architecture is shown in figure 6. A UPT user visiting the network without UPT functionality could then use the UPT facilities of the assisting network to access the UPT service. Three methods are possible:

a) the network without UPT functionality only accepts incoming calls to UPT users who have previously registered from another network on an AP of the network without UPT functionality;

b) the network without UPT functionality accepts incoming calls to UPT users. In addition, a UPT user can perform a registration for incoming calls and other simple service profile management operations;

c) full UPT service is available to the UPT user, by dialling a special "UPT access" number supplied by the network without UPT functionality.

Also, numbering and routeing aspects may have implications on the way of interworking with networks without UPT functionality. See ETR 067 [4].

6 Applications of the functional architecture

This Clause applies the functional architecture and the interconnection scenario defined in Clause 5 to several cases of use of the UPT service: UPT incoming and outgoing calls, UPT registration and service management (see also ETR 066 [3]). It also studies the situation where the UPT user is visiting a network without UPT functionality and requires the assistance of a UPT network.

6.1 Call handling

6.1.1 Outgoing UPT call originating in a UPT network

Since the originating network is a UPT network, the call can be handled locally (by SCFo) and optimal routeing can be achieved. Two different scenarios are examined separately in this subclause: SDF-SDF interconnection and SCF-SDF interconnection. Subclauses 6.1.2 to 6.1.5 only present the SDF-SDF scenario.

6.1.1.1 SDF-SDF interconnection

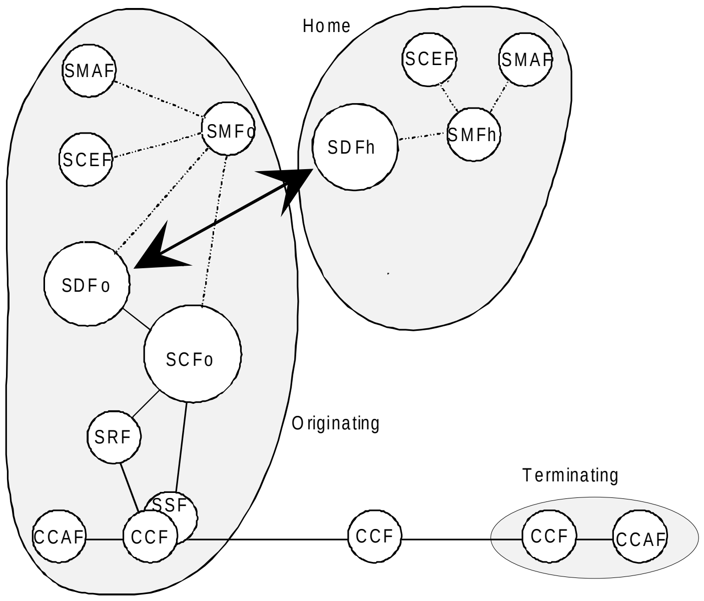

Figure 7: Outgoing UPT call, with data query between the originating and home networks

Two cases need to be considered in this scenario:

- the case where data needs to be queried by the originating network from the home network (figure 7), in which case communication is required between the two networks, over the SDF-SDF interface;
- the case where all the data required is already present in SDFo (figure 8), and where no communication with the home network is required. This happens when data has previously been copied from the home network to the originating network.

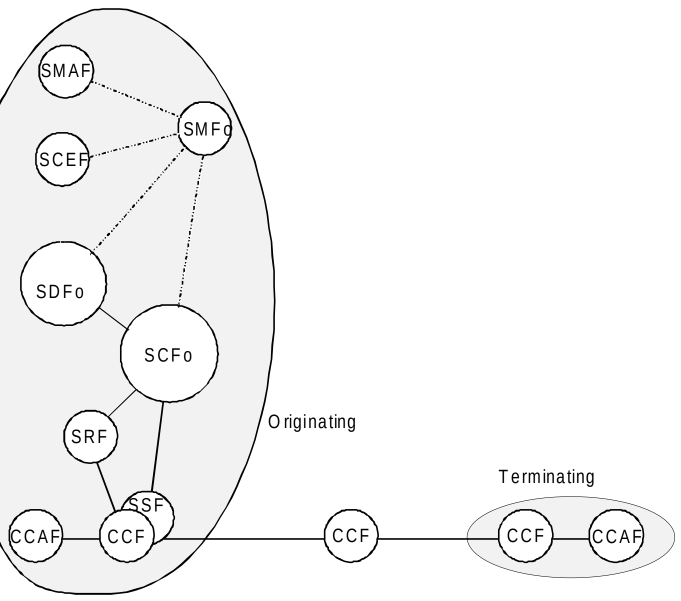

Figure 8: Outgoing UPT call, when data has previously been transferred from the home to the originating network (all the data is available in SDFo)

The same discussion may be repeated for most of the call cases presented (subclauses 6.1.3, 6.1.4, 6.1.5 and 6.2.1). The only figure which is presented is the case where the home network is involved(the case where it is not needed, because data has been copied into the originating network, is omitted).

6.1.1.2 SCF-SDF interconnection

No data distribution takes place in this interconnection scenario, so that communication between the originating and the home network is required in all cases (figure 9).

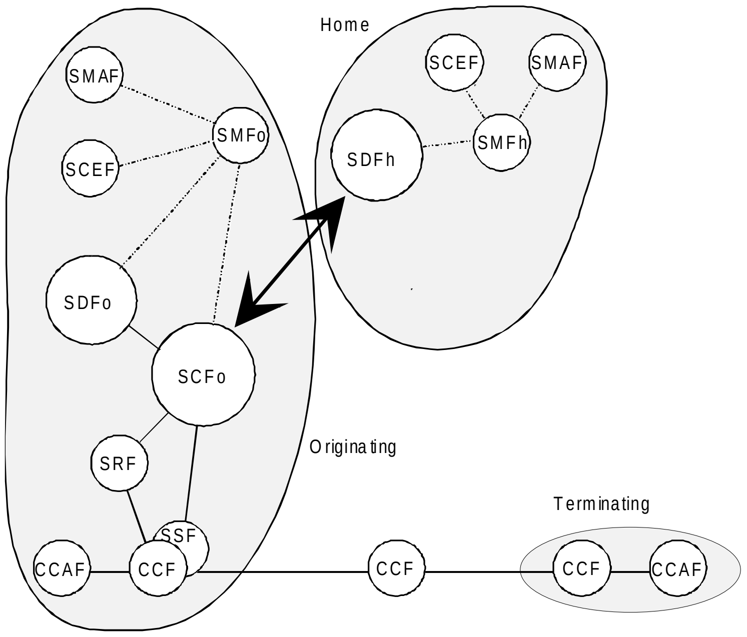

Figure 9: Outgoing UPT call, with SCF-SDF interconnection

6.1.2 Outgoing UPT call originating in a network without UPT functionality

As discussed in subclause 5.4 on the interworking with networks without UPT functionality, this scenario requires some co-operation between the originating and the assisting networks. The difference between this scenario and the call from a UPT network is that routeing may not be optimal and that tromboning can take place if the UPT user is calling someone who is also in the originating network. Figure 10 illustrates this case.

NOTE: As for subclause 6.1.1.1, data may have been transferred previously from the home network, in order to avoid querying SDFh during every call establishment phase. The only difference here is that the assisting network plays the part played by the originating network in 6.1.1.1, i.e. SDFo belongs to the assisting network.

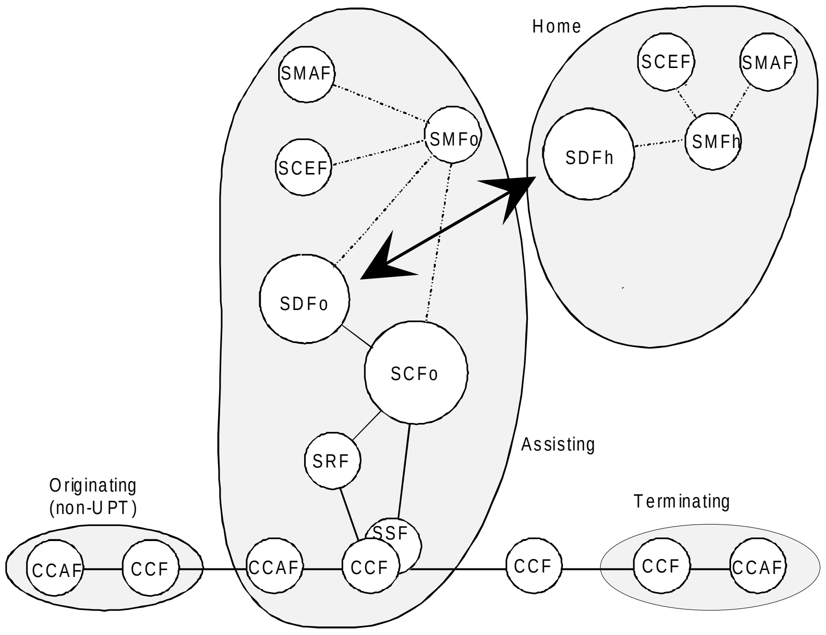

Figure 10: Outgoing UPT call originating in a network without UPT functionality, in the SDF-SDF interconnection scenario

6.1.3 Incoming UPT call originating in a UPT network

Since the originating network is a UPT network, the call can be handled locally (by SCFo) and optimal routeing can be achieved. Figure 11 illustrates the two interconnection scenarios.

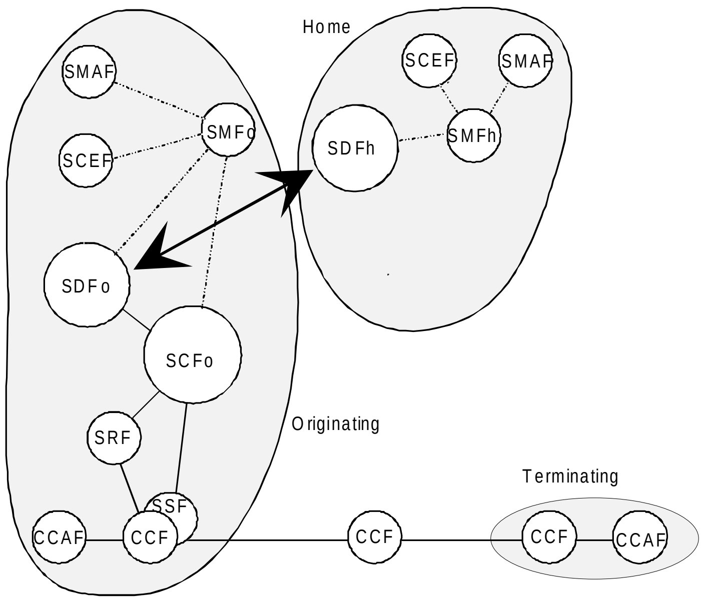

Figure 11: Incoming UPT call originating in a UPT network, in the SDF-SDF interconnection scenario

6.1.4 Incoming UPT call originating in a network without UPT functionality

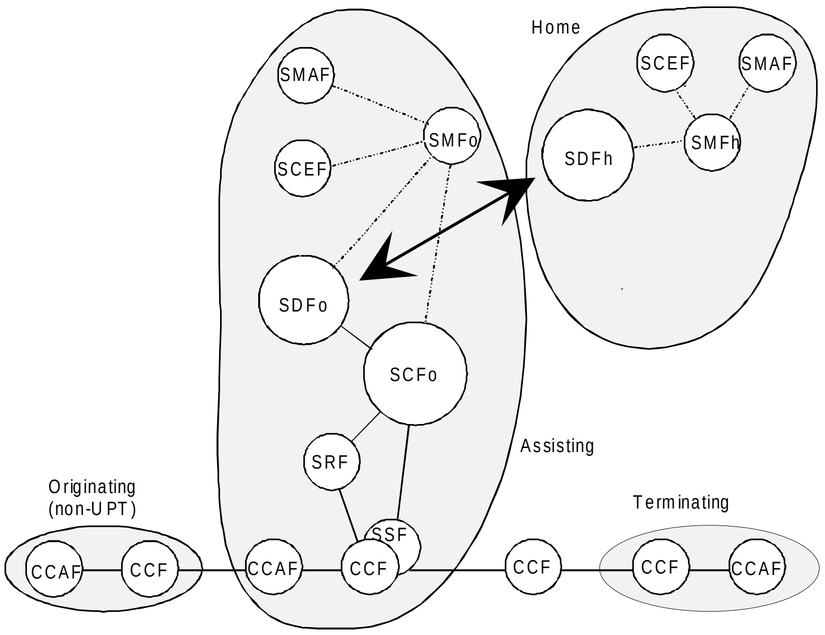

Figure 12: Incoming UPT call originating in a network without UPT functionality, in the SDF-SDF interconnection scenario

6.1.5 UPT user to UPT user call

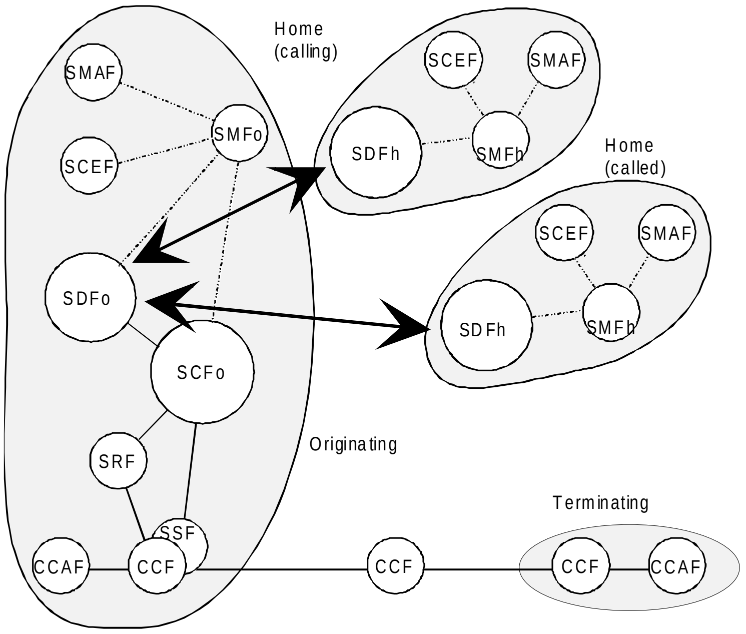

Figure 13: UPT to UPT user call, in the SDF-SDF interconnection scenario

6.2 Registration and service management

6.2.1 From a UPT network

Three networks are involved in this scenario. Call handling is performed locally (by SCFo), but registrations and the service profile are modified in the home network (SDFh). Also, any data which may have been transferred to a previously visited network shall be cancelled, under the supervision of SDFh (figure 14).

NOTE: Registration from a third network is not taken into account in the figure and is for further study.

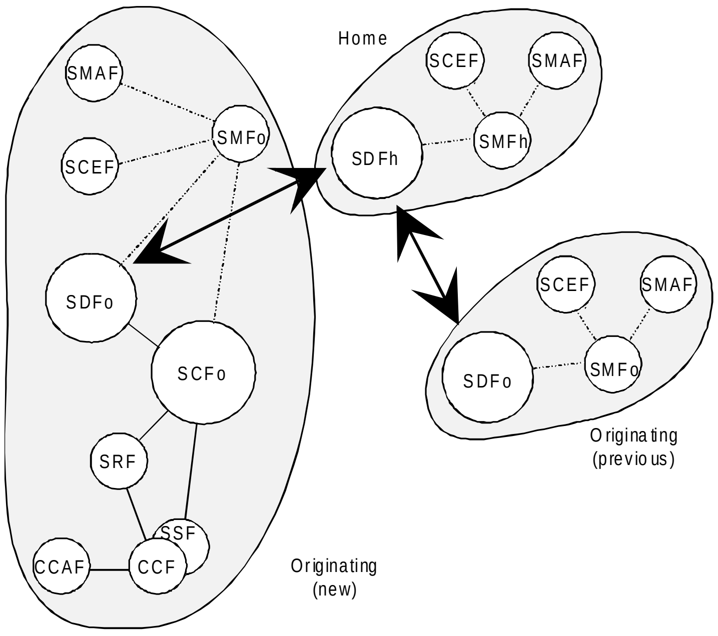

Figure 14: Registration from a UPT network, in the SDF-SDF scenario

6.2.2 From a network without UPT functionality

Three networks are involved in this scenario: call handling is not local, as it is performed by the assisting network (where the SCFo is located). The branch of the call going from the originating to the assisting network is not seen as a UPT call and, therefore, is subject to different charging principles (it may be free). For further study.

6.2.3 Using IN service management procedures

Only one network is involved in this scenario. Moreover, not all entities in this network are used, as only SMAF, SMFh and SDFh are used to let the UPT user interact with the network in order to query and modify the UPT user's service profile. For further study.

7 Handling of UPT data

7.1 Introduction

An important aspect for the provision of the UPT service is the need for a database in which subscriber-related and user-related information can be stored. This database shall also be able to store location information about UPT users and to retrieve it efficiently. In the short term, this database could be very simple by restricting it to only one entity, the SDFh, thereby resulting in a centralized mode of operation in which queries about a given user can only be answered by its SDFh.

In the long term, however, a more efficient way of handling queries is required, resulting in the need for a better distribution of the information in the network. Therefore, long term UPT architecture should specify a Distributed Database (DDB) for the management of UPT data. Annex A to this ETR sets a list of requirements on this DDB and indicates a number of problems to be studied on this subject.

It is the purpose of this subclause to study handling of UPT data: it describes a functional model which can be used to describe and compare different scenarios for the architecture of the UPT DDB. This model is used to describe two interconnection scenarios introduced in subclause 5.2 in more detail, from the point of view of UPT data handling.

7.2 UPT data handling functional model

The functional model presented in this subclause allows for a refinement of the general description of the UPT data handling functionality given above, by splitting further this functionality into separate functions. These functions can then be allocated to the entities of the UPT functional model to compare and evaluate scenarios.

A basic requirement is that no knowledge of the internal structure of the DDB should be required in the UPT service logic. The logical interface between the two should be designed to "hide" the structure of the DDB from the UPT service logic. To provide such a generic interface, two functions are defined (figure 15):

- **DDBin:** function to receive requests directed to UPT data handling from the UPT service logic. This function is responsible for hiding all details of the internal workings of the data handling part of UPT from the service logic part;

- **DDBout:** function to return the results of requests which have been processed by UPT data handling to the UPT service logic.

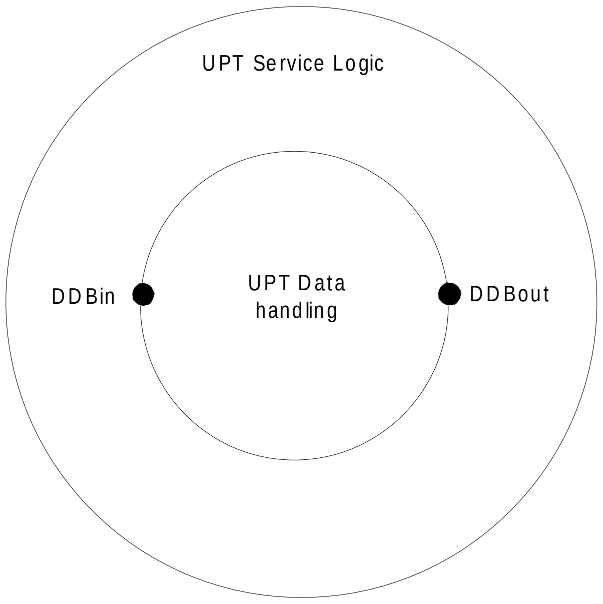

Figure 15: The DDBin and DDBout functions

Another important requirement is that the DDB be able to operate in a multi-operator environment. In order to take this into account, the logical interface between components of the DDB belonging to different operators is allocated to two functions (figure 16):

- **GWin:** function to receive requests from components of the DDB belonging to other operators. This function is responsible for authenticating the origin of the request, for checking that it is authorized and for passing it on to the relevant entity in its network;

- **GWout:** function to send requests to components of the DDB belonging to other operators.

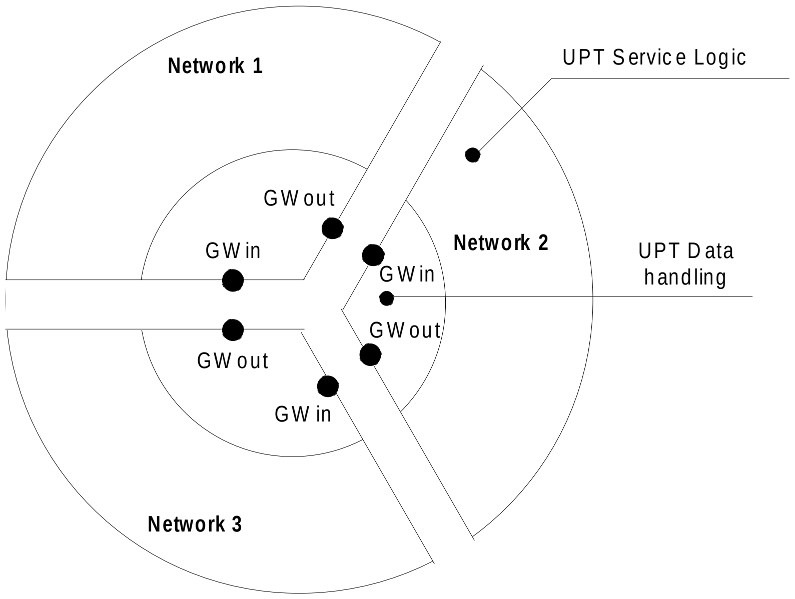

Figure 16: The GWin and GWout functions

The main requirement on the DDB is that it should be able to store and to locate UPT data. Two functions are defined for this:

- **DDBstore:** function to store UPT data. This function encompasses all the usual of functions a database management system: it allows the storage and retrieval of data, it ensures the integrity of the data stored, etc. Several instances of this function exist, on different physical nodes of the DDB;

- **DDBlocate:** function to determine which physical node of the DDB contains the data to be manipulated by a given request. The result it provides may be either the identity of another node in the same network, or the identity of another network. In the first case, the request is forwarded to the **DDBstore** function of the node where the data is stored, while in the second case it is forwarded to the **GWout** function.

Figure 17 gives a complete picture of the functional model for UPT data handling, and illustrates the processing of a request involving two networks.

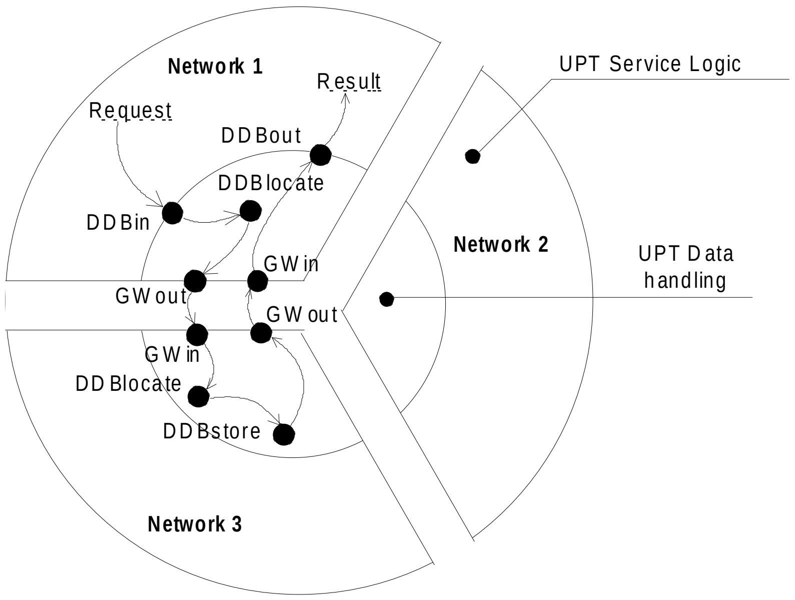

Figure 17: The UPT data handling functional model

The request originates in the service logic for network 1. It is submitted to the DDB by means of the **DDBin** function. The **DDBlocate** function handles it, and in this case, determines that the data is to be found in network 3. The request shall be forwarded to network 3, after all authentication and authorization checks are performed. This requires the co-operation of the **GWout** function of network 1 and the **GWin** function of network 3. Inside network 3, the **DDBlocate** function determines where the data can be found and forwards the request to the appropriate node, where the **DDBstore** function actually processes the request.

The path followed by the result of the request is the following: it is sent back to network 1 through a **GWout**/**GWin** tandem, and it is then passed on to the service logic by means of the **DDBout** function.

7.3 UPT data handling scenarios

The aim of the UPT data handling functional model is to allow the description and comparison of different scenarios for handling UPT data. This is achieved by allocating the functions described in the previous subclause to the entities of the general UPT functional model described in subclause 5.1. Two of the interconnection scenarios described in subclause 5.2 are considered: the SDF-SDF; and the SCF-SDF scenarios.

7.3.1 SDF-SDF interconnection

The allocation of the data handling functions to the functional model is given in figure 18.

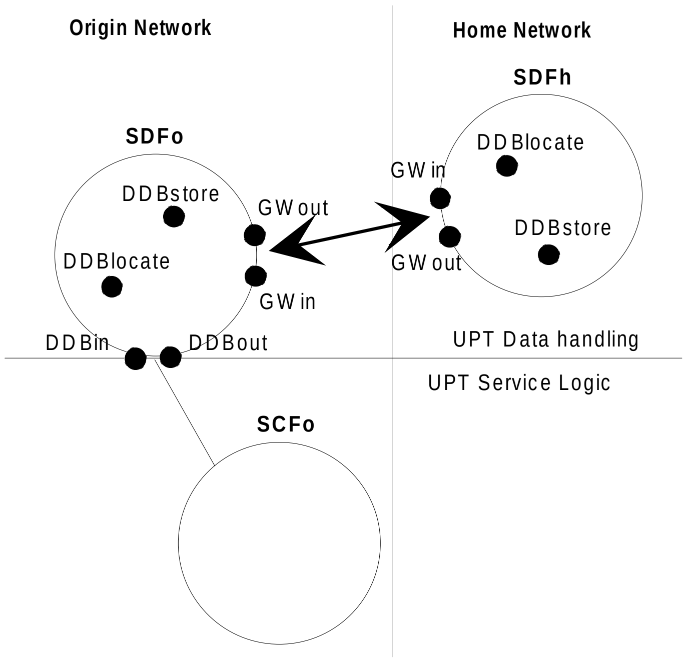

Figure 18: UPT data handling with an SDF-SDF interconnection

7.3.2 SCF-SDF interconnection

The allocation of the data handling functions to the functional model is given in figure 19.

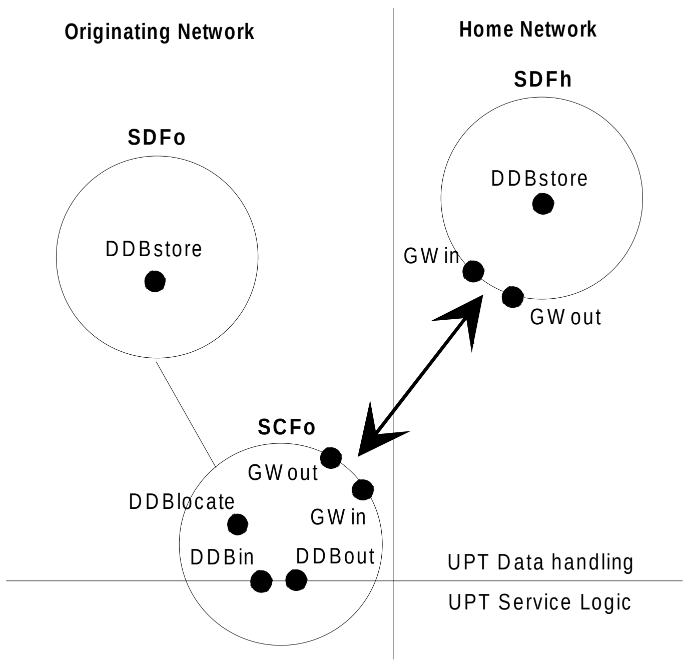

Figure 19: UPT data handling with an SCF-SDF interconnection

7.3.3 Comparison of the two scenarios

For further study.

Annex A: Requirements and study items for a UPT distributed database

A.1 Requirements

Generic interface: the interface between service logic (embodied by the SCF) and the DDB (embodied by the SDF) should be generic. No knowledge of the DDB structure should be needed when accessing it from the outside.

Access time and Performance: queries and updates to the DDB should be fast.

High mobility of data: it can be expected that mobility of data within the DDB will match that of UPT users. Therefore, it may be quite high in some cases, and the DDB should be able to handle this.

Minimum impact on the network: information exchanges between nodes of the DDB should be minimized, so as not to have a serious impact on the network.

Reliability: it should be possible, in the event of the failure of a node in the DDB, to back up to a safe state with minimal disruption of the service.

Protection: the DDB should be proof against unauthorized access.

Integrity of data: the DDB should ensure the integrity of the data it stores.

Gathering of statistics: the DDB should be able to gather statistics on its operation in order to provide data to the UPT service management.

Inter-Networking: the DDB should be able to operate in a multi-network and multi-operator environment. In particular, there is a strong need to keep a control over the ownership of data which may be distributed over several networks, only one of which really owns it.

A.2 Study items

A.2.1 Review existing architectures

A review should be conducted of existing or proposed architectures and protocols. Candidates in this domain are the Directory (X.500), with its extensions for distributed operation, the Common Management Information Service Element (CMISE) protocol with Transport Protocol (TP), the Object-oriented Distributed Processing (ODP) model, and possibly others. These three architecture/protocols sets have been standardized by international standardization bodies, or are in the process of being standardized. Therefore, they are available for re-use by systems like UPT. It is important that we should review them to assess whether they fit our needs, whether some modifications and extensions would be required, or whether they are not applicable at all to ETSI's work.

Criteria for such an evaluation could be:

- operation in a multi-operator environment;
- control on the ownership of data;
- high mobility of data;
- treatment of general security and reliability requirements.

A.2.2 Data model

The architecture of the DDB may not be influenced only by performance requirements. The nature and the structure of the data that it needs to store will also have an influence on it. Therefore, it is important that this nature be studied in detail and that a data model for UPT be evolved.

This data model may be simply described using plain English in a first stage, but a more formal method will certainly help in later stages. Entity-relationship modelling or one of the numerous object-oriented techniques may be used to this end.

A.2.3 Passive versus active DDB

The part the DDB plays in the overall UPT service logic may be seen in two different ways. It may either act as simple repository of information (even if it is one with a complex internal structure, since finding the information within it may involve the co-operation of several of its nodes), or it may take an active part in the logic of the service, in addition to storing data.

The difference between these two approaches can be illustrated with an example related to call handling. If UPT user A calls UPT user B, the service logic needs to first determine where the call to user B can be presented (that is, find where user B has made registrations) and then establish the call. In the passive approach, SCF_A sends a request to SDF_A, which then propagates it inside the DDB, until the location of B's registration is found (possibly in SDF_B). The information is returned to SCF_A which may then establish a dialogue with SCF_B and initiate the call set-up phase.

In the active approach, the two phases can be combined and the request sent by SCF_A to SDF_A may contain call control information which may be passed directly by SDF_B to SCF_B to initiate the call set-up phase.

A.2.4 Point of entry

When the UPT service logic needs to submit a request to the DDB, it shall send it to one of its nodes. The choice made for this point of entry into the DDB may have an important impact on the delay required to process the request. Several possibilities can be envisaged:

- **local request submission:** requests formulated by a given SCF are always sent to its associated SDF, which acts as an entry point into the DDB. The SCF has no knowledge of the internal structure of the DDB and needs not be aware of the existence of any other SDF entity. The only one it deals with is its local one;

- **remote request submission:** requests which can in advance be said to require the help of the SDFh are sent directly to this entity, whereas others are submitted locally. The SCF is required to have a little bit more knowledge of the structure of the DDB, but performance may be improved.

A.2.5 Structure

Using IN principles, the DDB will be made up of several SDF entities, each having a link with the rest of the service logic (through the SCF-SDF interface) and several links with the rest of the DDB (through the SDF-SDF interface). In order to design algorithms for querying, updating or exchanging data items within this collection of SDF entities, it could be useful to organize it into some structure.

It has been proposed[1)], for example, to use a hierarchical structure for the database nodes, in order to minimize the number of nodes to be queried before a given item of data can be found. ETSI should review this proposed structure and possibly identify and evaluate other structures as well.

A.2.6 Multi-operator environment

The ability to operate in a multi-operator environment has been listed as a requirement for the DDB. This puts severe constraints on its design, particularly the need to define precisely how data is copied from one network to another. What rights does an operator give on its' data when allowing it to be copied to another

---

1) Masanobu Fujioka, Sei-ichiro Sakai and Hikaru Yagi: "Hierarchical and distributed information handling for UPT", IEEE Network Magazine, November 1990.

operator's part of the DDB? Can the data be further copied to a third network? A check for an agreement between this third operator and the owner of the data will probably be required first. Perhaps such an operation should simply be forbidden.

Also, the overall structure of the DDB needs to be specified (i.e. how each operator's DDB fits into the overall scheme of the UPT DDB), but some degree of freedom needs to be left to each operator on how to organize that operator's part of the UPT DDB. Therefore, the inter-network part of the structure is an important part of the specification, while the intra-network part may not be. What is important for both parts is that the logic of the operations be consistently applied throughout.

A.2.7 High mobility of data

One of the requirements is that the DDB be able to handle a relatively high degree of mobility for the data it stores. Therefore, allowing the data to be moved from one node to another as the user changes registrations ensures better performance when the user requires access to the UPT service. As a consequence, the DDB should be able to efficiently move the items of data it stores from node to node and to keep track of them.

Classical DBB designs[2)], which are mainly distributed extensions of the relational model, may not be able to cope with such a requirement. Innovative solutions will be required in this domain.

A.2.8 Separation of user location and data location

It may be important that a rigid relation is not established between the current location of a user's registrations and the node(s) which store the user's profile data. If such a principle is adopted, the study of the DDB may be split into two parts:

1) study the problem of tracking and maintaining data consistent within the DDB;

2) study the problem of placing the data within the DDB, in order to optimize the way in which it handles the requests it receives from the rest of the service logic.

The first part is concerned with the algorithms used within the DDB to insert, to modify or to delete an item of data, as well as to create, move or delete a copy of an item of data. This problem is a fairly general one, and much could be gained by re-using existing protocols. While designing an architecture and a set of algorithms, it could be useful to consider what is available for re-use, in part or in whole (see subclause 6.1).

The second part is concerned with the heuristics for the placement of data within the DDB: when should a copy of an item of data be created; where should it be stored, when should it be moved to another node; when should it be deleted; etc. The advantage is that these heuristics need not be the same in all networks, provided that the logic of the basic algorithms is the same. Therefore, each operator may adapt the basic heuristics in order to optimize the operation of the operator's network.

---

2) See for example: Özsu and Valduriez: "Principles of distributed database systems", Prentice-Hall, 1991.

A.2.9 Management versus operation

Subclause A.2.8 describes the operations involving the creation of a new copy of a data item, etc. as a set of heuristics implemented in the database in order to optimize the performance of the DDB. It may, however, be argued that, due to their nature and to the timescales involved (weeks of statistics on the habits of a user may be needed before the system can decide that it should create a new copy of some item of data) these actions belong more to the management domain than to the operational domain.

The DDB would then be approached in a different light, it would perform only the basic work of storing data, retrieving it and maintaining it up to date, but it would be the task of another part of the system to analyse statistics and to take decisions as to which items of data should be duplicated or moved, in order to improve the performance of the DDB.

NOTE: Such an approach precludes real-time adjustments to the operation of the DDB, which are a possibility if the operational approach is used. Such real-time response to the activity of UPT users could be useful during rush hours or rush periods where a peak of activity may be generated by mass movements.

A.2.10 Consistency versus inconsistency

The main task of the DDB is to store one or more copies of the data item it has been entrusted with and to keep them consistent independent of where they are stored. In some cases this task may require a significant amount of signalling between database nodes if all the copies of a given item of data are to be kept synchronized (especially in the case of frequently updated data items, such as those pertaining to registrations).

Therefore, it may be beneficial in some cases to trade-off consistency for improved performance, i.e. some item of data will not be updated at the same time that other copies of it are altered, but only at the time it is needed. If several alterations of its copies have taken place between two uses of the data item, several updates will then have been saved by only updating it at the time it is actually used.

Such cache schemes, which accept a degree of inconsistency in the data stored by the DDB, require further study.

History

Document history	
February 1994	First Edition
February 1996	Converted into Adobe Acrobat Portable Document Format (PDF)