Requirements and a Proposal for the Prevention of a Class of Service Interactions in Intelligent Networks

Dirk O. Keck

Institute of Communication Networks and Computer Engineering (IND)
Prof. Dr.-Ing. Dr. h.c. Paul J. Kühn, University of Stuttgart,
Pfaffenwaldring 47, D-70569 Stuttgart, Germany
keck@ind.uni-stuttgart.de

Abstract. Service interactions in Intelligent Networks are a major obstacle to the fast and cost-efficient introduction of new telecommunication services. In this contribution, a class of service interactions are addressed that result from deficiencies of the IN infrastructure to provide to a service the necessary information to carry out its task properly. A number of requirements are established that remove or at least reduce these deficiencies, and a proposal is made how these requirements could be realized. In this proposal, the concept of *Service Admission Control (SAC)* is introduced, which is based on a session concept represented by a dynamic *Session Configuration Profile (SessCoP)* accessible and modifiable by IN services that may influence a call relationship (session). This SessCoP allows the IN services to gather the information needed to carry out their tasks properly, that are not provided by the messages of the signalling protocols. The application of these concepts are explained by two brief case studies. This contribution reports on work in progress.

1 Introduction

1.1 The Intelligent Network

The Intelligent Network (IN) provides architectural support for the realization of new services in communication networks. It is based on the principle of separation between the basic functionality of the switches and the functionality of the services. This leads to a simplification and acceleration of service development, since it is not necessary to modify the software within the switches to introduce new services, and to a reduction of management effort, as the services are concentrated into service control points. For a more detailed introduction into the concept of the Intelligent Network, the reader is referred to [1, 2].

1.2 The Service and Feature Interaction Problem

The terms of service and feature interaction refer to situations where two or more communications services or service features (as the basic elements of a communications service) influence each other mutually. This results in a behaviour of at least one of the services or features, that is different from the behaviour of this service or feature without presence of the other service or feature. The term *interaction* does not imply that this behaviour is undesired; if this fact has to be expressed, the term *interference* should be used. [3,4] provide a concise introduction into this problem area.

1.3 Classification of Service Interactions

The cause of a service interaction is an important classification criterion [5]. In this paper, a slightly different taxonomy is used, based on a scheme sketched in [6]. This taxonomy is based on the observation that feature interactions emerge at different levels of abstraction in the view of a service, and that this fact presents a good taxonomy for classification of interactions with respect to dealing with them.

- *Logical Level.* The logical level is used to model concepts like a service's purpose and intention, focused on the service users' perspective. This level corresponds to the service plane in the IN conceptual model (INCM, [7]). At this level, the way how a service is provided, i. e. the assignment of the different functions to different functional entities, as well as the algorithm and the information used in order to provide the services are neglected.
- *Level of the Abstract Architecture.* On the level of the abstract architecture, a service is modelled in a network- or architecture-dependent way, i. e., the mapping of functionalities to functional entities, the algorithms used, and the information needed in order to provide a service are an issue. This level corresponds to the Global and Distributed Functional Planes in the INCM. The main ideas presented in this paper relate to this level of abstraction, more exactly to the way how the appropriate information about the state of the network and suitable mechanisms to get this information can be provided to a service.
- *Level of the Concrete Architecture.* The concrete architecture takes into account all the irregularities of real systems, like, e. g., physical limitations (memory, bandwidth), interworking with legacy systems and services, different signalling protocols and procedures. It corresponds to the Physical Plane in the INCM. Except for the aspect of interworking with legacy services, this level is not within the scope of this paper.

2 Interactions Caused by Insufficient Information

A significant number of service interactions are caused by insufficient information on the state of the network and of other services being involved in a call relationship. This may result in incorrect decisions of the services and in conflicts concerning the access to system resources. Interactions that belong into this class are addressed by this contribution.

2.1 IN Services in Present Systems

The design of IN services for today's networks is mainly based on the following design guidelines, as far as a treatment of service and feature interactions is concerned: (1) A proper interworking with network based services has to be guaranteed; in cases where this cannot be achieved, the activation of the IN service in combination with these network-based services is prohibited in order to avoid problems. (2) Interworking with the known set of other IN services present in the network has to be ensured by designing the services appropriately or, in difficult cases, prohibiting their activation if conflicting services are present. These quite simple guidelines reflect the limited abilities of the IN infrastructure to deal with interactions.

IN services of present implementations are based on the mechanisms provided by the IN capability set CS-1 (or CS-2)[1] and the digital exchanges using the ISUP (ISDN User Part) protocol. The mechanisms to obtain information on the network state and on other services (listed in Table 1) are in most cases sufficient to implement services that interwork with a number of traditional switch-based and ISDN supplementary services.

These mechanisms provide valuable support for dealing with interactions, but there are some severe restrictions, making it difficult to develop a general method for interaction resolution or prevention on top of these mechanisms:

- For an IN service triggered during a call, it is impossible to obtain information if there is another IN service active on the same call relationship, if the latter service has been triggered later. Furthermore, the information which IN service has been triggered previously can be obtained only indirectly by evaluating the *serviceInteractionIndicator* information element of the INAP. In cases where more than one service have been triggered previously, this may be difficult if not impossible.
- The information that may be exchanged between IN services triggered by different SSP is limited to the capabilities of the ISUP indicators and other information elements.
- The actions that can be carried out for dealing with interactions are limited to disabling the common activation of two services – as far as one service has the possibility to recognise that the other one is active, or to influence a service's behaviour based on the information obtained by this service through ISUP/INAP information elements.

The items presented above demonstrate how limited the capabilities of the IN with respect to dealing with interactions are. During the remainder of this paper, the problems are analysed and a number of requirements for capabilities that would improve the situation will be established. Afterwards, a proposal is made how these requirements could be realized in a real network by an extension to the IN concept called *Service Admission Control (SAC)*.

[1] The mechanisms of CS-2 with respect to obtain information on the network state are in principle the same.

Table 1. Mechanisms to deal with interactions provided by CS-1 (or CS-2)

MECHANISM	DESCRIPTION
Mechanisms of IN CS-1 to obtain information on the network state	
Detection Points	Provide information to a service about the point during call processing that has been reached.
INAP Information Elements (Q.1218)	*Information on Directory numbers and parties:* dialledDigits, calledPartyNumber, callingPartyNumber (+ additional information), originalCalledPartyID, additionalCallingPartyNumbers
	Information on call forwarding treatment: forwardCallIndicators, redirectingPartyID, redirectionInformation
	Possibility to propagate ISUP indicators to IN services: serviceInteractionIndicator
Control/Monitor relationship	A concept to guarantee that only one service may actively influence a basic call state model at any given time; the number of services that are notified is not limited.
StatusReportRequest/ StatusReport	Request the SSF to report the busy/idle status of a physical termination resource immediately/on change/on a specific status (this message is not available in ETSI Core INAP).
Mechanisms of the ISUP to transport information on services between SSPs	
ISUP Service Interaction Indicators	Fixed set of indicators (in forward and backward direction) that allow to influence the reaction on or the treatment of specific events or switch-based services.
Information Elements	Practically identical to the IEs of INAP concerning directory numbers, parties, and call forwarding treatment.

2.2 Analysis of the Problems and Examples

The design of the messages and information elements of ISUP and INAP are based on traditional two-party calls, enhanced with relatively simple call forwarding and number translation services. The concepts that are introduced by new and complex IN services extend the paradigm of traditional services by a number of aspects described below. In order to allow a proper and correct reaction of the service, a view onto these extensions must be obtainable by the services.

Directory numbers and their mapping to physical lines. The traditional *one-to-one* relationship between directory numbers and physical lines is enhanced significantly by IN services. In many services, one directory number is used to reach multiple destinations, possibly in a time- and origination-dependent way. This introduces a *one-to-many* relationship between directory numbers and physical terminations. Conversely, there is a *many-to-one* relationship introduced by services like MSN (Multiple Subscriber Number). *Many-to-many* relationships exist for services like call centers that work for several companies. A number of services need this information in order to be able to react appropriately. Furthermore, there can be a theoretically unlimited[2] number of participants (directory numbers) involved in a call relationship, like in the example described below. The limitation of the information elements conveyed by the signalling messages may lead to an information loss in such cases.

Multiple Call Forwarding serves as an example for this problem. In this example, the information transfer in a chain of call forwarding steps is studied. A calls B, who forwards to C. C forwards to D, etc. If this scenario is closely examined, the result is that the information on the redirection by party C is lost in the IAM received by E, i. e. party E cannot reconstruct the sequence of forwarding steps that has led to this call. It is obvious, that each service that requires complete information on a call's history cannot work properly. The information obtained by A on the history of this call with the CPG (Call ProGress) / ACM (Address Complete Message) ISUP signals is even more limited.

Resources. Usually, it is assumed by IN services that a resource is available for use by a service when it is available in the network. For example, a specific key on the terminal's keypad or the hook-flash signal is used without knowing whether there is another service that reacts on the same stimulus. As a second example, an announcement requiring user action can be mentioned, where it is assumed that a human user is on the phone who is able to perform the required action. If this is not the case (voice mail, "do not disturb" services with PIN entry), a problem is predetermined.

Voice Mail (VM) and Credit Card Calling (CCC), an example presented in [5], demonstrate a conflict in the use of the #-key, which is used by the CCC service for ending the actual call and placing a new one, and by the VM service to trigger a specific action. Clearly, if both services react on the key, this will lead to a severe and confusing interference. In section 4.2, this examples serves as a case study.

Enhanced Concepts. Traditionally, a call relationship is terminated, if all participating parties are on hook again. Some services enhance this concept by introducing a relationship between subsequent calls, like, e. g., recall of the last caller, where it may lead to severe interactions if the second call (the recall) is treated as independent from the previous one, as it is explained in the following example.

Calling Line Identification Restriction (CLIR), Automatic Recall and Itemized Billing, an example by *B. Cohen*, are services that exhibit an interference although individual pairs of these services are free of interactions. Party A, with CLIR active, has called B, with Automatic Recall. B then recalls A, and can find A's directory number on its itemized phone bill, although A has subscribed to

[2] In real implementations, there are, of course, limitations that are introduced by mechanisms like counters (e. g., call forwarding counters), or by physical limitations, but these limitations are caused by design decisions, not by logical restrictions.

CLIR which suppresses the presentation of A's identity (directory number). This interaction is caused by the fact, that the recall is treated as an independent, new call, and not as a consequence of the previous call, where CLIR has been active. It could be resolved (prevented) by providing additional information on the attributes of the last caller's number to the Automatic Recall services (in this case, that the number must not be presented). Of course, this attribute must then be recognized by the Itemized Billing feature.

Charging. The traditional charging concept is based on two major principles: (1) The caller is charged and (2) the tariff can be determined by the directory number that has been dialled. This concept has been extended relatively early by concepts like reverse charging (toll-free numbers), split-rate and premium-rate services, which are, however, in principle built using the same mechanisms.

Advanced IN services introduce the need for more flexible models, like a generalized split charging model (where all participants pay for a call), a third party payment possibility, where a party pays for a call in that it is not directly involved, and the possibility for one subscriber to pay for several IN services he uses during the course of the same call, and to be informed online what amount he is charged for these services.

The above list of abstract concepts extended by IN services does not claim to be complete, but gives an idea where a number of problems that are listed under the generic term service interaction, have their origins.

3 Requirements for the Prevention of these Interactions

3.1 Principles of Service Creation for IN

The IN philosophy follows the guideline of independent and parallel service development (service creation) and also concepts of service privacy in a competitive multi-provider environment. A conventional treatment of service interactions, i. e., taking the implementation of the new service and checking its behaviour in the presence of all other services would clearly offend this philosophy, let alone that this would end up in a complexity problem. Therefore, an important general requirement for prevention and resolution methods is that they must support this independent development of new services.

3.2 Functional Requirements

In the previous chapter, a number of weaknesses of the IN concept concerning the possibility to exchange information between individual services are identified as obstacles to react properly on the threat of service interactions. In this section, requirements for the elimination of these flaws are introduced using a stepwise approach with three levels of increasing functionality and complexity. On the first level, two options are described to notify a service of other services active on the same call relationship.

Level 1a: When an IN service is initially triggered, it has to be informed which other services are already active on this call. This requirement is fulfilled to a significant extent in today's systems by the mechanisms presented in 2.1. Provided this knowledge is available, the IN services have to be designed for proper interworking with known other services (IN and switch-based).

Level 1b: It is possible to obtain information on other services active on this call not only when a service is triggered initially, but also during service processing. This requirement is not fulfilled by today's signalling protocols.

In the above two requirements, it is up to the services to react appropriately on each individual other service, of course only within the limitations of network support. This contradicts to the requirement of independent service development, but in a single-provider environment, the importance of this fact is (still) limited. The following option provides a solution to this problem, the concept of a service-independent Service Template that contains static information on a service, a so-called Service Profile.

Level 2: There exists a Service Profile based on a general template where each service can obtain generalized information on the "internal structure" of other services and react properly on the different alternatives found in the template. The contents of these templates are generated during service creation. It is assumed that each service knows what other services are active (like in levels 1a and b), but they do not need special knowledge of the services, as they can access this knowledge in the form of the static Service Profile.

This template based approach is the first step towards a generalized Service Admission Control (SAC) concept that allows to decide whether a service may be executed or not, or executed in a modified or limited manner, based on the service's requirements expressed in the Service Profile. The problem of such an approach based on a static Service Profile is the fact, that information about what the service could do in principle can be obtained, but not about what the service has done actually. This is an obstacle especially for complex services with many alternative execution paths. This limitation directly leads to the following level 3.

Level 3: The concept of a static Service Profile is replaced by a dynamic Session Configuration Profile (SessCoP), where the actions of a service on a call (session) are recorded and the actual state of this session can be retrieved. It is obvious that this session profile has to be accessible by all services participating in the session.

The requirement outlined on level 3 introduces the Service Admission Control (SAC) based on the Session Configuration Profile (SessCoP), a powerful means to react properly on the actions carried out by other services involved in the same call relationship. During the rest of this contribution, it is described how this requirement can be realized within the existing IN infrastructure and exploited in order to prevent (or resolve) service interactions.

The requirements of level 3 in the previous section is built on top of the concept of a session. Here, an informal definition of this concept is given.

> **Definition (Session):** A session consists of a set of parties that take part actively or passively (e. g., as forwarding parties) in a call relationship. These parties must be related to each other by the existence of (active or suspended) connections or their establishment (successful and unsuccessful) and teardown processes. Thus, the set of participating parties may change over time. A party is involved at each point in time in at most one session; this implies the necessity to be able to join two or more sessions into one. A session ends, if all participating parties are idle and no knowledge (memory) of this session exists in the network that is able to influence the future. Service interactions, as far as they are treated in this paper, are always taking place within one session[3].

3.3 Non-Functional Requirements

The functional requirements presented above are complemented by several other non-functional ones, a selection of which is listed below:

- In order to guarantee acceptance of the method, it has to be integrated seamlessly into the established SIB-based service creation methodology in a way that is as transparent to the service developer as possible.
- The impact on the performance of the services (in terms of answer time) as well as the impact on the computational complexity of the services has to be kept reasonably low.
- As far as the robustness is concerned, there must be useful default behaviours for cases where the session information is obviously inconsistent or non-existent.

3.4 Other Issues

Some other issues that have to be taken into consideration are interconnection of networks belonging to different operators (a major issue for a de-regulated telecommunication market), and the inclusion of services provided by alternative mechanisms like service nodes, and customer premises equipment. These topics exceed the scope of this paper.

4 A Proposal for a Methodology

In this section, it is examined whether and how the concepts and requirements for a Service Admission Control mechanism based on a dynamic Session Configuration Profile that have been introduced in the previous section can be transferred into a real IN infrastructure.

[3] The author has no knowledge of interaction examples that occur across session boundaries if a session is defined in the above sense.

4.1 Necessary Basic Mechanisms

Notification of other services on initial trigger. For the proposal made on levels 1a and b, the notification of other services (triggered after the first service) can be achieved using a unique Service Identifier (Service Key), that is conveyed as a new information element by the ISUP messages. During the call establishment phase, there is always an exchange of ISUP messages between all participating parties, making it easy to transport the new information element without introducing additional messages. The Service Identifier (or, if more services are active, the list of identifiers) are sent to the services as an additional information element of the Initial Detection Point (IDP) INAP message.

More difficulties result from services that are triggered during the stable phase of a call (by so-called mid-call triggers), or during the release phase. In these situations, a message has to be sent that would not be needed otherwise, possibly a Call Progress Message (CPG)[4].

Notification of other services during service processing. The method described on level 1b requires a means to notify a service during service processing in addition to the initial notification of its activation. Service processing in the IN usually takes place in a stimulus-response manner with short response times. The stimuli are the detection point messages emitted when the DPs are reached during call processing. Under the assumption of short response times, it is sufficient to provide the Service Identifiers of new services active on the actual call together with the event detection point messages sent to a service.

Static Service Profile. In an IN infrastructure defined according to IN CS-1, it is possible to implement this Service Profile described on level 2 as a static data structure stored in a Service Data Function (SDF) database. There, it can be accessed easily by the services. Performance and database load implied by such a solution is an issue that has to be investigated in further research.

Dynamic Session Configuration Profile (SessCoP). This concept, presented on level 3, is the most complex one and makes up the key issue in the Service Admission Control concept presented in this paper. It is far more difficult to implement than the concepts of levels 1a,b and 2 due to the dynamic evolving nature of the SessCoP. It requires the possibility to uniquely identify a session, and means to locate, access and modify the session configuration profile. The management of the session configuration profiles is also an issue. Proposals for a realization of these mechanisms are presented in the following paragraphs.

The *Session Identifier (Session ID)* is the means to access the SessCoP of a session. Since it is not necessary for the first IN service triggered in a session to take into account interaction with other services (except for conventional switch-based or ISDN services), a network wide (or worldwide) unique Session Identifier

[4] The CPG is used, e. g., in conference calling services to notify participants of actions carried out by other participants, like leaving a conference, isolating a party, etc.

is created and assigned when the first IN service is triggered and is propagated to the other parties involved in this session using the mechanisms described in a previous paragraph. The Session Identifier contains information that is sufficient to locate the SessCoP.

The *Session Configuration Profile (SessCoP)* is represented by a data structure containing generic information that is able to reflect the manipulations an IN service is able to carry out. Table 2 displays a list of information needed for the profile, without claiming that this information is sufficient for the intended task.

Table 2. Information of the dynamic SessCoP

Participants	O-/T-BCSMs with physical line IDs as identification Dialled Number history (the way to obtain the term. physical line ID) Attributes (terminating, non-terminating) Number Attributes (time-/origination dependent, CUG, IN number)
Resources	Keys, Hook-flash signal, Voice Channel (in/out/both), ...
Topology	Connections, Connection Attributes (active, suspended)
Charging	Information on charging (for further study)

SessCoP Manipulation: In order to maintain the information stored in a SessCoP, it is necessary to provide a complete set of operations to manipulate this information. This includes operations to manipulate the individual data members, to clear a SessCoP, to join two profiles and to divide two profiles. The operations needed depend of the contents of the profile and are still under research.

Management of the SessCoPs: For each session a SessCoP is created and maintained. As this profile is accessible only by IN-services, i. e. SCPs, it is an issue how to remove profiles whose sessions have ceased to exist. This information is not necessarily propagated to the SCPs, because synchronization only takes place at armed detection points. However, this is an important issue with respect to security concerns, like the protection of personal data.

There are a number of alternatives to keep this information up do date. First, it is possible to design the IN services appropriately, so that all IN services notify the SCP on release. This possibility creates a large amount of signalling traffic (INAP), which makes this possibility quite expensive in terms of signalling, but also guarantees that the SessCoPs are removed correctly. The second alternative is an aging mechanism, which may lead to inconsistencies because there is no limit to the maximal duration of a session. If this aging mechanism is combined with a check of the parties that belong to a session, possibly by a kind of periodic “keep-alive message”, this problem could be circumvented. The third possibility is to monitor for all physical line IDs to which session they belong, and if there is no physical line ID assigned to a specific session, it can be removed. This would lead to a quite complicated and error prone algorithm.

Impact on Service Creation: The maintenance of a SessCoP adds a significant amount of complexity to IN service logics. It cannot be avoided that also already existing services will be affected and have to be adapted. However, a large part of this additional complexity can be hidden from the service designer by an appropriate design of the SIB library.

4.2 Case Studies

The case studies described in this section are coarse grained and simplified with respect to the full amount of signalling messages, information elements and internal processing required. The intention is to illustrate the method described in this paper, not to give a complete and unambiguous specification of it. The contents of the SessCoPs printed in *italics* are displayed as far as they are relevant for the actual case study.

Voice Mail (VM) and Credit Card Calling (CCC). In this case study, it is shown how the interaction in the example explained in 2.2 can be resolved by SAC mechanisms using the dynamic SessCoP presented in this contribution. Both services are realized using the Specialized Resource Function (SRF). In Fig. 1, the message exchange during service processing is sketched together with the contents of the SessCoP. Party A calls the CCC number. A session identifier (ID) is assigned. The credit card number and the voice mail service number are entered by the user and the appropriate event notifications are requested by the CCC service, the Session ID is returned. The SessCoP is updated by a *UpdateSessCoP* message. Afterwards, the VM service number is connected and the service is triggered, this time with a session ID. The VM service tries to allocate the necessary resources. The conflict concerning the use of the #-key is detected and reported by a *SessCoPReply* message to the service logic. In this situation, different reactions are imaginable:

1. The VM service is ended with an announcement *"Failed to allocate the #-key resource"*,
2. The VM services request another unreserved key, for example the $\star$-key, which has to be announced to the user, of course.
3. The user of the two services may actively select (by using his display or an announcement) to which of the services the #-key signal will be propagated.
4. There is a concept of a "context" of a service, and a signal is propagated to the service whose context is active, similar to the service architecture described in [8].

In the example shown in Fig. 1, the second alternative has been chosen. The VM service can be quit either by going on hook or by sending a #-key signal to the CCC service. If the latter case takes place, the reserved resources are released. If the CCC call is ended by going on hook, the session is finished. Now the cleanup mechanisms described in 4.1 come into action.

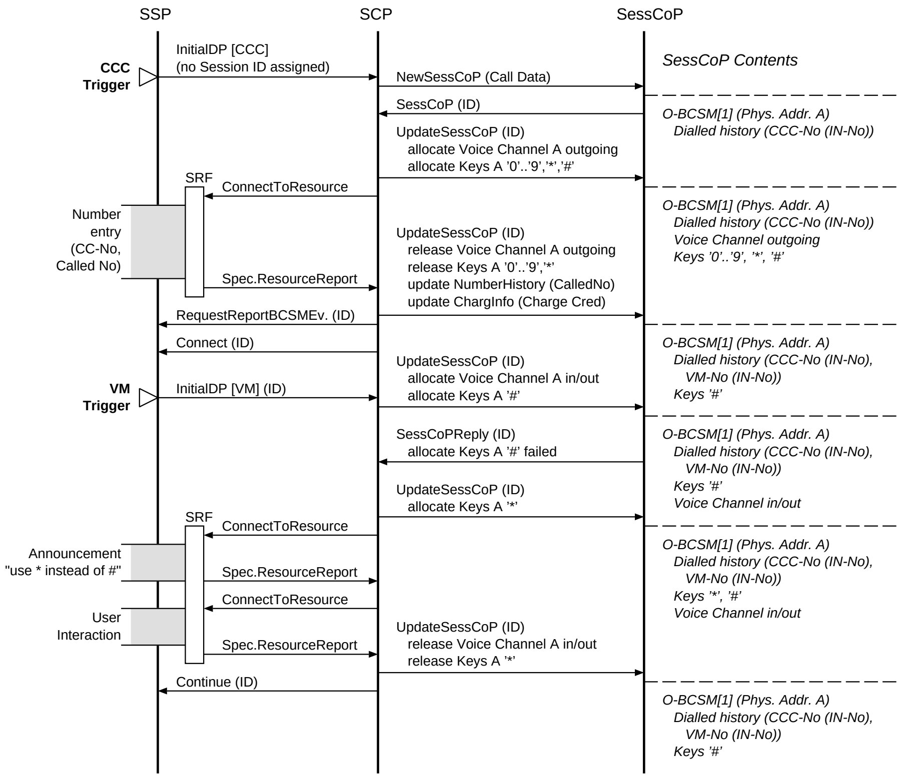

Fig. 1. Interaction of *Credit Card Calling (CCC)* and *Voice Mail (VM)* services.

Dial-In conference with Advanced Originating Call Screening. In this example, it is explained, how a service called Advanced OCS[5] does behave together with a conference call service realized as an IN service on a specialized resource (conference bridge). The conference call service is implemented in a way that a new party that calls any of the parties already involved in the conference is automatically forwarded to the conference bridge. Fig. 2 displays a part of the interaction of AOCS and Conference Calling. For clearness sake, the SSPs of D and A and the SCPs of the both services are drawn separately. Party D (with B on its screening list) calls A, AOCS is triggered and the session ID-AOCS is assigned. When the Conference-Forwarding service of A is triggered, it is detected that there is already a session present (ID-Conf). The both session are joined, and ID-AOCS and ID-Conf reference this joined session. However, one of the IDs (ID-AOCS) is elected for future use, the other one is no longer propagated. After the Answer message (ANM) has been returned to D's SSP, the SessCoP is

[5] An Advanced OCS is an originating call screening service that does a second verification before the alerting state is reached or the call is connected (both detection points are available only in IN CS-2).

checked again using D's screening list, recognizing that B is taking part in the Conference. Therefore, D's AOCS service logic releases immediately.

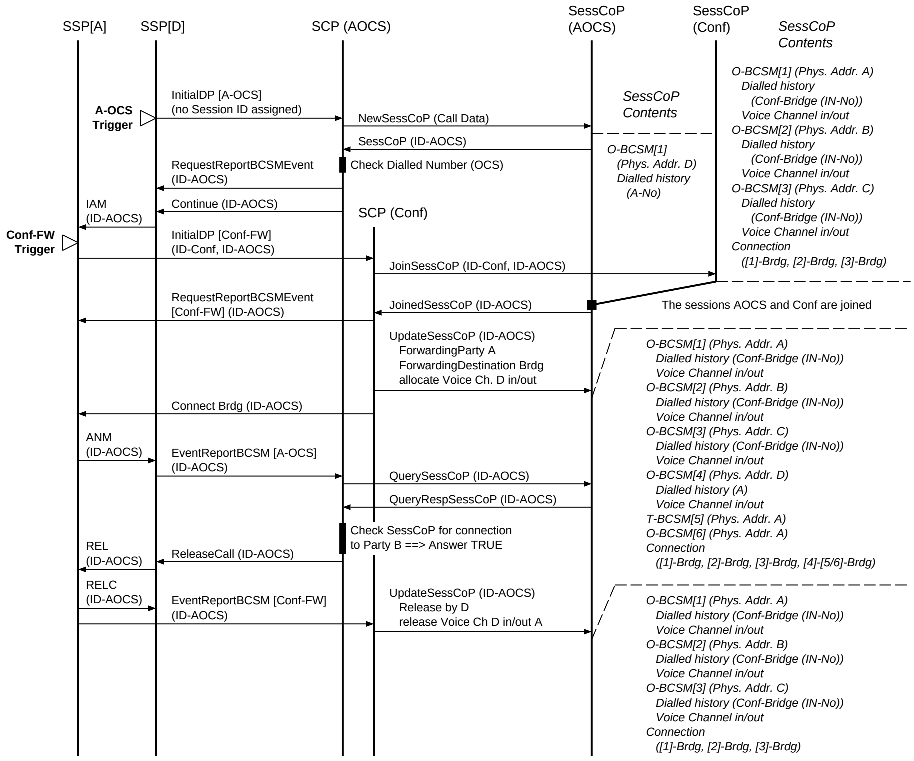

Fig. 2. Interaction of *Advanced Originating Call Screening (AOCS)* and *Conference Calling* services.

4.3 Technical and Realization Aspects

Session ID Assignment. For the assignment of the Session IDs there are basically two alternatives, an assignment in the SSP or an assignment in the SCP.

It is probably more convenient to assign a Session ID in the SCP after the first service has been triggered, and the Session ID is returned together with the first SCP-SSP message sent after the trigger message. This strategy would assign to the SCP the responsibility to maintain the SessCoP, and the Session ID would contain an address of this SCP, together with an ID to identify the profile on this SCP. This strategy is used in the above case studies. However, it leads to problems if the initial trigger is a notification, i. e. has no reply (used in services like televoting). This serious restriction can be circumvented if the assignment of a Session ID is requested explicitly in cases where it has not been already assigned. In these cases, the SSP has to request the assignment of a Session ID using a request as initial trigger, while otherwise a notification is sufficient if a Session ID is already present.

If the Session ID is assigned in the SSP prior to the processing of the trigger, this problem does not occur, but it is probably more complicated to create a unique Session IDs here, as there are much more SSPs than SCPs, which makes the creation more difficult to maintain.

SessCoP Location. The Session Configuration Profiles are data structures, that have to be accessible by the Service Control Points (SCPs) of the network. For their location, there exist different options.

A straightforward solution is a location in the SCP where the first service in a session is triggered. In an IN, there are typically few SCPs, and the probability is high that all IN services that influence one session are provided by the same SCP. This would reduce the inter-SCP information exchange necessary to access the SessCoPs and to keep them consistent. In cases where SessCoPs are joined, it may be necessary to transfer the SessCoP information to another SCP and to leave a reference to the joined SessCoP instead.

An alternative is the introduction of an independent and dedicated node to maintain the session configuration profiles. However, there is communication necessary for each access, which impacts the performance of such a solution.

A distribution of the information to the SSP would bring advantages for the actuality of the SessCoPs' information, as the actual state of the participants could be accessed easier, but the communication overhead due to the distributed nature of the information on the relatively slow signalling network would be significant. However, this restriction does not hold for an IN architecture based on the B-ISDN.

Performance. Service processing in the IN is a time critical task, and it can be taken for granted that the proposed methodology influences this service processing time. There is a direct relationship between the performance and the delays that are introduced by the access and processing of the service configuration profiles as well as an indirect relationship caused by the higher complexity of the service logics. These important issues need more investigation in further research.

5 Related Work

In contrast to the application of a *Feature Interaction Manager (FIM)* as presented in [9], in the SAC approach presented here the knowledge about the resolution of the conflicts between the services is not concentrated to one functional entity – the FIM – but distributed in a generalized form to the services. The *table driven approaches* described by Homayoon and Singh in [10] and by Schessel in [11] decide in the SSP on the activation of features based on static decision tables, in a feature individual way.

In the approaches described above, only interactions of services triggered in the same call state model can be handled properly, while in the SAC approach presented here these interactions as well as interactions between services triggered in different SSPs, but in the context of the same session, are addressed, as the decisions are made during service processing, not in the SSP.

Griffeth's and Velthuijsen's *negotiating agents* approach [12] is based on a negotiation process based on a general hierarchy of goals common to all services. One service makes a proposal how to proceed based on its own goals and preferences. Another service checks whether this proposal can be accepted with respect to its own goals, or whether a counterproposal has to be made. Although this approach aims at interactions on the logical level, there are some structural similarities to the SAC approach in a sense that there is a service independent representation of the results of the services' actions on a session. However, in the SAC approach, no abstract goals representing intentions for the future, but only concrete facts representing actions that have taken place in the past, are available. Thus, no negotiation process takes place, but only a simple decision based on the SessCoP data. It cannot be guaranteed, that interactions that manifest itself as logical contradictions in the services' intentions are detected and resolved.

6 Conclusions

The results and proposals presented in this contribution result from studies of service and feature interactions in Intelligent Networks. The significant progress that has been achieved in the area of detection of feature interactions during the last years has not been complemented by comparable success in the resolution and prevention of these situations. The reason for this fact is, in the author's opinion, twofold. It is relatively easy to detect logical and intentional contradictions between services provided there is a suitable notation for these issues, but it is difficult to resolve them, like in the negotiating agents approach. But even if these problems can be solved, there remains a significant number of interaction cases, where it is clear how the services should react in principle, but it cannot be done or it is done in a wrong way as it is not enough information available to carry out the task correctly.

In this paper, we focus on the latter type of interactions. The concept of *Service Admission Control* is introduced and a methodology to resolve and prevent unwanted interactions is presented that is based on a dynamic *Session Configuration Profile* for each session that is influenced by at least one IN service. This profile can be accessed and modified by all IN services participating in a session. They can gather all the information needed to react properly that is not available in the ISUP and INAP signalling messages, provided that this information is contained in the session configuration profile's data model, and they can use resource reservation mechanisms that are not supported by the underlying network.

As this contribution reports on work in progress, there are many open issues for further research. There is at the moment only a vague idea of the necessary contents of the Session Configuration Profile, as well as of the methods needed to access and manipulate these contents. Furthermore, it has to be investigated to what degree the service design will be complicated by taking into account the maintenance of a SessCoP. On the side of non-functional properties, all the allocation alternatives and the performance impact of the proposal have to be studied thoroughly.

References

1. T. Magedanz and R. Popescu-Zeletin, *Intelligent Networks: Basic Technology, Standards and Evolution*, International Thomson Computer Press, 1996.
2. I. Faynberg, L. R. Gabuzda, M. P. Kaplan, and N. J. Shah, *The Intelligent Network Standards: their application to services*, McGraw-Hill, 1997.
3. E. J. Cameron and H. Velthuijsen, "Feature Interactions in Telecommunications Systems", *IEEE Communications Magazine*, vol. 31, no. 8, pp. 18–23, August 1993.
4. N. D. Griffeth and Y. J. Lin, "Extending Telecommunications Systems: The Feature-Interaction Problem", *IEEE Computer*, vol. 26, no. 8, pp. 14–18, August 1993.
5. E. J. Cameron, N. D. Griffeth, Y. J. Lin, M. E. Nilson, W. K. Schnure, and H. Velthuijsen, "A Feature Interaction Benchmark for IN and Beyond", in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., May 1994, pp. 1–23.
6. P. Combes, M. Michel, and B. Renard, "Formal Verification of Telecommunication Service Interactions using SDL Methods and Tools", *Proceedings of the 6th SDL Forum 1993 (SDL '93)*, pp. 441–452, 1993.
7. "Principles of Intelligent Network Architecture", ITU-T Recommendation Q.1201, Geneva, 1992.
8. P. K. Au and J. M. Atlee, "Evaluation of a State-Based Model of Feature Interactions", *Feature Interactions in Telecommunication Networks IV*, pp. 153–167, June 1997.
9. M. Cain, "Managing Run-Time Interactions Between Call-Processing Features", *IEEE Communications Magazine*, vol. 30, no. 2, pp. 44–50, February 1992.
10. S. Homayoon and H. Singh, "Methods of Addressing the Interactions of Intelligent Network Services With Embedded Switch Services", *IEEE Communications Magazine*, vol. 26, no. 12, pp. 42–70, December 1988.
11. L. Schessel, "Administrable Feature Interaction Concept", *Proceedings of the XIV International Switching Symposium (ISS)*, pp. B6.3–B6.3, October 1992.
12. N. D. Griffeth and H. Velthuijsen, "The Negotiating Agents Approach to Runtime Feature Interaction Resolution", in *Feature Interactions in Telecommunications Systems*, L. G. Bouma and H. Velthuijsen, Eds., May 1994, pp. 217–235.