APIs ouvertes pour la création de services de télécommunication

EFORT
http://www.efort.com

Les solutions orientées API désignent les approches qui se focalisent sur la définition d'interfaces de haut niveau pour la programmation de services de télécommunication. Ces interfaces permettent d'abstraire les ressources et de faciliter ainsi le développement des applications. Dans ce chapitre, sont décrites les architectures qui ciblent les services de télécommunications dans les NGN, à savoir, les architectures JAIN (paragraphe 1) de SUN Microsystems et l'architecture PARLAY/OSA (paragraphe 2) du groupe PARLAY et de l'ETSI.

1. JAIN

JAIN (Java APIs for Integrated Networks) est un ensemble d'interfaces de programmation qui fournit un cadre complet pour le développement des services qui s'exécutent aussi bien sur les réseaux en mode paquet que sur les réseaux en mode circuit. Conçu pour la plate-forme Java, cet ensemble d'interfaces vise à assurer la portabilité des services et à répondre aux besoins de convergence des réseaux. Le cadre JAIN repose sur le principe de séparation entre la programmation des services et celle du réseau. Ainsi, il se définit à travers deux couches : la couche application et la couche protocole. A cet effet, deux groupes de travail ont été mis en place, à savoir, le groupe application (AEG, Application Expert Group) et le groupe protocole (PEG, Protocol Expert Group).
Le groupe PEG a pour mission de normaliser les interfaces permettant l'accès aux protocoles de signalisation de différents types de réseaux. Ces protocoles sont TCAP, ISUP, INAP, MAP, H.323, SIP et MGCP. Ainsi, PEG offre la possibilité d'utiliser des piles de protocoles provenant de plusieurs constructeurs sans influencer le développement des services. Cette solution permet une portabilité maximum des composants développés au niveau de la couche application. Le groupe PEG définit uniquement les APIs des protocoles et ne propose pas d'implantation particulière. En effet, l'implantation d'une interface donnée se traduit par le développement d'un adaptateur spécifique pour chaque pile de protocole propre à un constructeur.
Le groupe AEG met l'emphase sur les API liées à la commande des appels (JCC, JAIN Call Control), sur la création de services et sur l'environnement d'exécution (JSC/SLEE, JAIN Service Creation/Servic Logic Execution Environment). En effet, de même que pour l'approche SoftSwitch, JAIN définit un modèle d'appel indépendant des protocoles de signalisation des différents réseaux. L'objectif suivi est de masquer les particularités des protocoles en offrant aux applications un modèle d'appel uniforme. Les entités de la couche application peuvent accéder aux adaptateurs de la couche protocole à travers les APIs définies à cet effet.
Les API JAIN sont conçues selon un modèle maître/asservi. En effet, une API est composée de deux interfaces : une interface du coté composant fournisseur de la capacité de l'API et une autre du coté composant utilisateur. La première est nommée « interface du fournisseur » (provider interface), la deuxième s'appelle « Interface d'écoute » (listener Interface). Dans ce modèle, les flux émis par le fournisseur et adressés à l'utilisateur sont transmis sous forme d'événements. Afin de recevoir ces événements, l'utilisateur doit enregistrer son interface d'écoute auprès du fournisseur à travers une méthode java qui est généralement de la forme « addToListener() ». Quant aux flux de l'utilisateur au fournisseur, ils s'effectuent par des invocations simples de méthodes Java définies dans l'interface fournisseur. L'utilisateur du service doit récupérer la référence de l'interface du fournisseur par d'autres moyens non spécifiés par JAIN. Par ailleurs, les interactions utilisateur-

fournisseur peuvent se dérouler d'une manière répartie en utilisant RMI (Remote Method Invocation) de Java.

1.1. Commande d'appel JAIN

L'API de commande d'appel (JCC) constitue le noyau de l'architecture JAIN. Le modèle d'appel qu'elle propose est inspiré du modèle de l'INAP-CS2.

En effet, JCC reprend tous les points de détection proposés dans le BCSM de CS2. Il offre en outre des capacités permettant la manipulation des appels multiparties et permet la gestion des canaux multimédias.

La figure 1 présente un exemple de service de traduction de numéro simple basé sur JCC et utilisant une entité SSP du réseau intelligent.

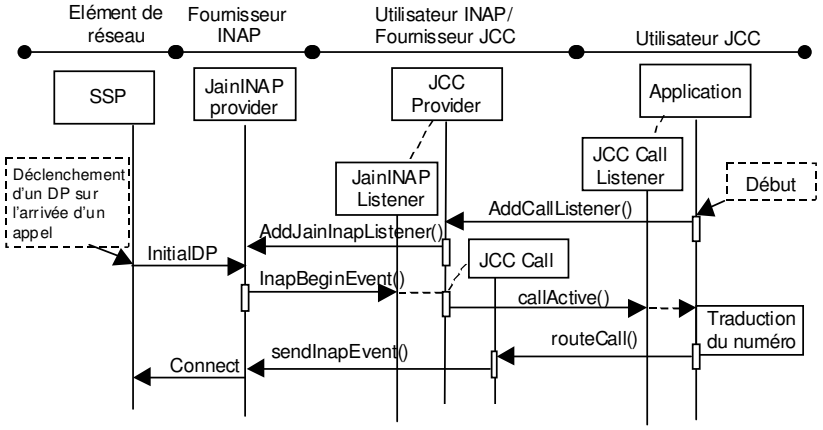

Figure 1 : Service de traduction de numéro utilisant JAIN-INAP et JCC

Afin de recevoir des notifications d'appels, l'application de traduction de numéro s'enregistre auprès du fournisseur JCC en utilisant la méthode addCallListener(). A cet effet, l'application crée un écouteur d'appel « CallListener » et un filtre d'événements précisant les propriétés des appels pour lesquels elle souhaite être notifiée. Le fournisseur JCC cherche le (ou les) fournisseur(s) INAP approprié(s), crée un « listener » d'évènements INAP et s'enregistre auprès de ce (ou ces) fournisseur(s). Des TDP (Trigger Detection Point) doivent être armés au préalable au niveau de l'entité SSP, et ce d'une manière cohérente avec les événements que les fournisseurs INAP sont capables de notifier. Ainsi, à l'arrivée d'un appel à l'entité SSP, cette dernière émet la requête INAP « InitialDP » au fournisseur approprié. Le fournisseur INAP signale l'événement à l'utilisateur correspondant (ici le listener créé par JCC Provider) en invoquant la méthode InapBeginEvent(). Le fournisseur JCC intercepte l'événement et crée un objet[1] représentant l'appel à traiter. Cet objet contient des informations concernant l'état de l'appel et permet de maintenir la cohérence de ces informations par rapport à l'état réel de l'appel dans le réseau (ici dans l'entité SSP). Ensuite, l'événement est notifié à l'application par invocation de la méthode callActive() du listener déjà créé par l'application. A cette étape, la traduction du numéro s'effectue et l'application décide de router l'appel vers un numéro physique. Pour ce faire, l'application invoque la méthode routeCall() de l'objet représentant l'appel au sein du fournisseur JCC. A la réception de l'invocation, JCC agit comme utilisateur du fournisseur JAIN-INAP et invoque la méthode sendInapEvent(). Cette méthode contient en paramètres l'objet de type opération « connect » de JAIN-INAP qui contient l'adresse destination. C'est ainsi que l'exécution du

[1] Les objets dans JAIN sont exclusivement des objets Java. Un objet peut implanter une ou plusieurs interfaces JAIN.

service se termine par l'envoi du message INAP « connect » à destination de l'entité SSP qui prend en charge la reprise et le routage de l'appel.

JAIN définit donc des APIs de bas niveau pour l'abstraction des protocoles, et des APIs de haut niveau pour l'abstraction du traitement des appels. Cette architecture présente néanmoins deux principales limites. D'une part, l'architecture JAIN est fortement dépendante de la plate-forme Java. D'autre part, elle propose uniquement des services de téléphonie. L'architecture PARLAY proposée par le groupe Parlay décrite dans le paragraphe suivant constitue une solution qui permet de dépasser ces limites.

2. PARLAY/OSA

Le groupe PARLAY a été créé en 1998 conjointement par BT, Microsoft, Nortel, Siemens et Ulticom. L'objectif de ce groupe est de spécifier un ensemble d'interfaces ouvertes (API) permettant aux opérateurs ainsi qu'aux fournisseurs tiers de construire des applications qui reposent sur la commande en temps réel des ressources des systèmes de télécommunication. Les interfaces spécifiées par PARLAY sont indépendantes des technologies et des langages de programmation ; elles comprennent des définitions de méthodes, d'évènements, de paramètres et de leur sémantique. L'architecture PARLAY possède trois caractéristiques essentielles qui la distinguent de la solution JAIN. Ces caractéristiques sont :

- La fourniture d'interfaces standards et indépendantes de toute technologie et de langage de programmation.
- L'offre d'un cadre permettant l'accès sécurisé des fournisseurs situés à l'extérieur du domaine administratif de l'opérateur qui contrôle le système de télécommunications[2].
- La prise en compte, en plus de la téléphonie, d'une large gamme de services, tels que services de localisation, services de messagerie, services multimédia, etc.

Les APIs publiées par PARLAY sont spécifiées en langage UML (Unified Modeling Language) et sont organisées en paquetages d'une manière hiérarchique. Quant à l'architecture, elle est définie à travers six blocs conceptuels, à savoir, le service, l'application cliente, le cadre d'utilitaires (framework) et les fonctions d'administration liées à chacun de ces blocs (Figure 2).

[2] Il faut cependant de noter que JAIN propose dans le cadre des spécifications JAIN Parlay une implémentation d'un sous-ensemble de PARLAY et ce afin de permettre à des fournisseurs tiers d'accéder aux services de l'infrastructure.

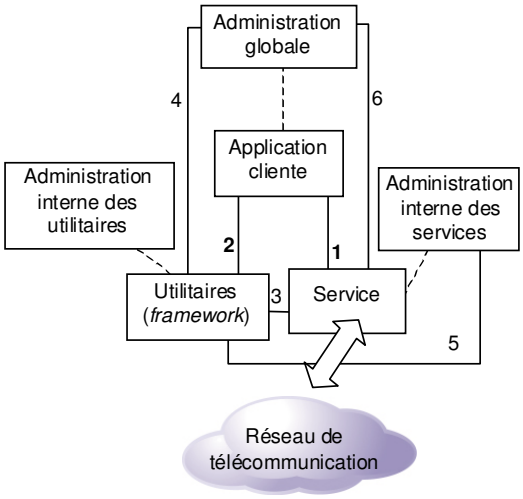

Figure 2 : Les Interfaces Parlay

L'interface N°1 se positionne entre l'application cliente et le service. Cette interface couvre les fonctions de commande des services. L'application cliente contient la logique du service global à offrir. Le bloc service offre l'ensemble des capacités que les ressources du réseau sont capables de fournir. De la même façon que pour JAIN, l'interface N°1 se définit par un couple d'interfaces : l'interface utilisateur et l'interface fournisseur. L'interface utilisateur est nommée « interface d'application » (Application Interface) ; celle relative au fournisseur est appelée « interface de service » (Service Interface). Contrairement à JAIN, l'architecture PARLAY ne propose pas d'interfaces de protocoles.

L'interface N°2 définit les interactions entre l'application cliente et le bloc utilitaire (framework). En effet, le framework offre l'ensemble des fonctionnalités nécessaires à la sécurité et à la gestion des interfaces de service. Les fonctionnalités offertes couvrent la gestion de la sécurité d'accès, la gestion de l'intégrité, la gestion de la souscription et la découverte des services. En d'autres termes, le framework assiste les applications dans leur accès et dans l'utilisation des capacités offertes par le bloc service.

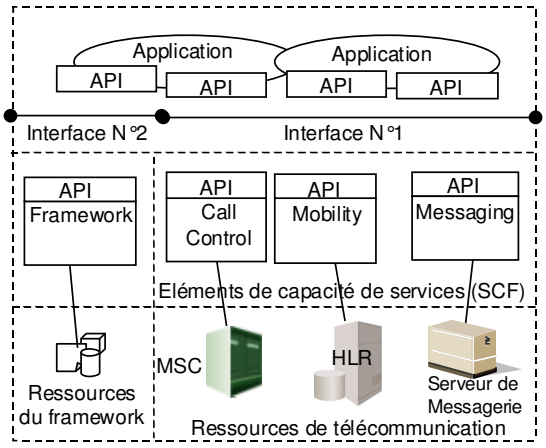

Figure 3: L'architecture Parlay

Les interfaces 3 et 5 sont utilisées par les développeurs de services. L'interface N°5 permet l'enregistrement par la gestion des capacités nouvellement déployées. Elle offre aussi des fonctions de gestion d'intégrité utilisables par les administrateurs des services. L'interface N°3 permet de réaliser d'une manière automatique les fonctions offertes par l'interface N°5.

Ces deux interfaces autorisent également le déploiement de capacités développées par des parties tierces.

Enfin, les interfaces N°4 et N°6 permettent de supporter de manière normalisée l'accès à des fonctions de gestion pour l'administration des services et des utilitaires. En revanche, la gestion interne de ces derniers ainsi que la gestion des applications ne sont pas couvertes par les spécifications PARLAY.

Les interfaces N°1 et N°2 sont mise en œuvre par un Parlay Gateway dans les solutions constructeurs. Le Parlay Gateway assure la traduction entre les interfaces Parlay et les protocoles de télécommunication pour utiliser les interfaces offertes par les éléments de réseau. Parmi ces protocoles figurent INAP, CAP, MAP, ISUP, SIP et H.323.
Le Parlay Gateway est considéré par les constructeurs de télécommunication tels qu'ALCATEL ou ERICSSON comme une application présente sur leur offre SCP Réseau Intelligent (Figure 4). Le SCP dispose de tous les protocoles cités ci-dessus, pour interagir avec les éléments de réseaux GSM, RTCP et NGN. L'application Parlay Gateway offre par ailleurs une interface IT aux serveurs d'application contenant les applications Parlay. Parmi les interfaces IT possibles figurent CORBA (Common Object Request Broker Architecture), Java RMI (Remote Method Invocation), HTTP (HyperText Transport Protocol), SOAP (Simple Object Access Protocol) et DCOM (Distributed Object Component Model).
De nouveaux fournisseurs proposent des Parlay Gateways indépendants (Figure 5). Ces fournisseurs ne vendent pas d'équipements de télécommunication et leur produit Parlay Gateway est conçu à partir de piles (stack) protocolaires permettant la communication avec les éléments de réseau à travers des protocoles standards. Parmi ces fournisseurs figurent Infitel, Aepona, IpGen et Incomit.

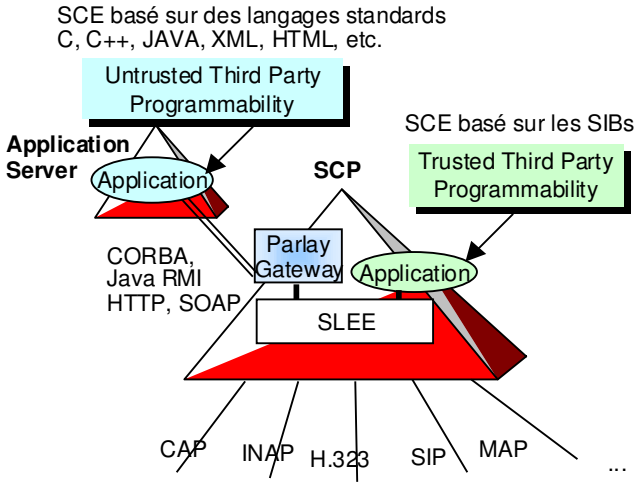

Figure 4 : La solution Parlay des fournisseurs de télécommunication

L'autre composant important de l'architecture Parlay est le serveur d'application (Application Server). Ce dernier fournit un environnement de création de service standard avec des langages de programmation tels que JAVA, C++, HTML, XML, etc. Par ailleurs, le serveur d'application est un client Parlay qui peut invoquer des méthodes publiées par le framework et les services Parlay (interfaces N°1 et N°2)

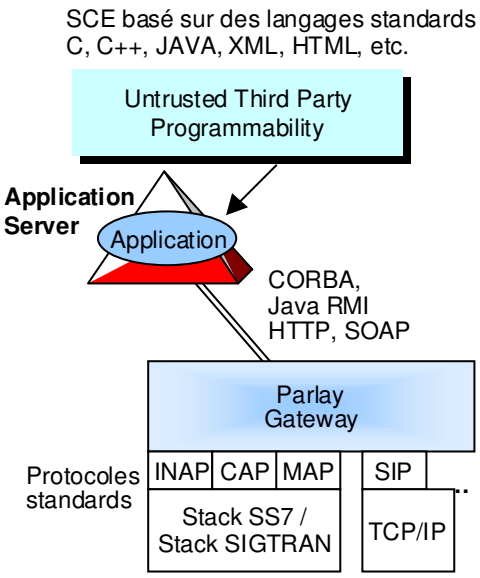

Figure 5 : La solution Parlay des fournisseurs indépendants

2.1. Les services de PARLAY/OSA

Comme il a été déjà précisé, outres les services de commande des appels de téléphonie (Call Control), PARLAY couvre de nombreux autres services. Il s'agit précisément des services de messagerie générique (Generic Messaging), de mobilité (Mobility), d'interactions avec l'usager (User Interaction) et des services de connectivité (Connectivity Management). Lorsque le groupe PARLAY a travaillé conjointement avec l'ETSI, l'architecture a pris le nom d'OSA (Open Service Architecture) et a été enrichie par trois autres services. En effet, le groupe 3GPP de l'ETSI a orienté les spécifications OSA vers le contexte de l'UMTS et a introduit le service de « session de données » (Data Session), le service de gestion de compte (Account Management) et le service de facturation (Charging). Par ailleurs, l'architecture OSA traite le mapping entre les interfaces des services qu'elle propose et les protocoles de télécommunication tels que CAP, MAP et SIP. D'un point de vue architectural, chaque service est conceptualisé par un bloc fonctionnel nommé « élément de capacité de services » (SCF, Service Capability Feature).

La SCF de commande d'appels (CC SCF, Call Control SCF) est très semblable à celle de JAIN. En effet, il est possible de considérer JAIN JCC comme étant une implantation de la SCF CC en utilisant le langage JAVA.
La SCF interactions avec l'usager (UI SCF, User Interaction SCF) est conçue afin d'enrichir la SCF CC avec des capacités de collecte d'informations utilisateur (e.g., collecte de digits DTMF) et d'envoi d'annonces (annonces vocales ou SMS).
La SCF de mobilité (Mobility SCF) est applicable aux utilisateurs des réseaux RTCP, IP et mobiles (GSM, GPRS, UMTS). En effet, cette capacité permet d'obtenir des informations sur l'état et la localisation géographique d'un ou de plusieurs utilisateurs. Les applications peuvent demander de recevoir des informations de localisation périodiquement ou à chaque changement de localisation. Ce type de capacité permet la réalisation de services à forte valeur ajoutée. A titre d'exemple, une application de réservation de taxi peut utiliser ce service afin de servir une demande après avoir identifié le taxi disponible, qui est actuellement le plus proche possible de la localisation du demandeur.
La SCF de messagerie générique (Generic Messaging SCF) est une fonction qui s'appuie sur des serveurs de messagerie électronique. En effet, La SCF GM peut communiquer avec un agent de messagerie pour émettre un message, consulter une boîte aux lettres ou être notifiée de l'arrivée d'un message. Cette capacité offre l'API nécessaire pour accomplir l'ensemble des fonctions de manipulation de messages vocaux ou messages électroniques.

La SCF de session de données (Data Session SCF) remplit les mêmes fonctions que la SCF CC mais avec la particularité de traiter une signalisation qui se rapporte à des sessions de données. C'est une SCF introduite spécialement pour le réseau GPRS.
La SCF gestion de compte et la SCF taxation sont définies dans le but de faciliter les services prépayés qui sont de plus en plus utilisés dans les systèmes mobiles. Ainsi, ces capacités permettent, d'une part, d'effectuer la taxation en temps réel et, d'autre part, d'autoriser les usagers à gérer leurs comptes. La SCF de taxation est aussi utilisée pour facturer l'usage des applications et permet par le biais de mécanismes de réservation le partage d'un même compte.
Enfin, la SCF gestion de connectivité (Connectivity Management SCF) est utilisée par les réseaux d'entreprises de type VPN (Virtual Private Network). L'opérateur offre les services de transport et permet aux dépositaires de VPNs, à travers l'interface de cette capacité, de superviser et de configurer leur réseau privé virtuel. Cette SCF est essentiellement utilisée pour la supervision de la QoS réalisée par le fournisseur et pour l'établissement de liens entre les sites du VPN selon une QoS donnée.

Structure d'une SCF

Tous les éléments de capacité de services (SCF) OSA ont une structure similaire. OSA définit deux classes d'objet du côté réseau :

- **Ip<Interface>** est une interface qui fournit des méthodes afin de contrôler les ressources dans le réseau. <Interface> est le nom de l'interface.
- **Ip<Interface>Manager** est une interface de gestion fournissant les méthodes d'initialisation et de gestion d'une instance d'une Ip<Interface>. Elle permet aussi de souscrire à des événements particuliers.

Afin que le réseau puisse émettre des notifications à l'application cliente, cette dernière implante deux classes d'objet pour chaque SCF :

- **IpApp<Interface>** est une interface de l'application cliente supportant les méthodes lui permettant de recevoir les résultats ainsi que les notifications de l'interface Ip<Interface>.
- **IpApp<Interface>Manager** est une interface sur laquelle l'application cliente peut recevoir des résultats ainsi que des notifications de l'interface Ip<Interface>Manager.

2.2. SCF Call Control

Il existe trois variantes de la SCF Call Control : La SCF Generic Call Control (GCC), la SCF Multiparty Call Control (MPCC) et la SCF Multi-Media Call Control (MMCC). La SCF GCC permet l'établissement d'appel téléphonique entre deux participants. La SCF MPCC enrichit la SCF GCC avec la gestion de leg. Elle permet l'établissement d'appel téléphonique multiparties. Un nombre de legs strictement supérieur à 2 peuvent donc être simultanément connectés. La SCF MMCC enrichit la SCF MPCC avec la capacité multimédia pour établir des sessions multimédia (voix et vidéo), multiparties.

Le **modèle d'appel** adopté par les SCFs Call Control distinguent les objets suivants :

- Un objet **call**. Un appel (call) : Il s'agit d'une relation entre un nombre de participants. C'est la vue qu'a l'application d'un appel physique dans le réseau.
- Un objet **call leg** : Une branche d'appel (call leg) représente une relation logique entre un appel et une adresse. Dans un appel classique impliquant deux participants, il y a toujours deux call legs impliqués : l'un représente l'appelant et l'autre, l'appelé. Un appel multiparties implique plus que deux call legs.
- Un objet **address** : Une adresse représente un participant dans l'appel (e.g., par un numéro de téléphone dans le cas du RTCP ou du réseau GSM ou par une adresse IP dans le cas d'un réseau s'appuyant sur le protocole IP.

Il existe deux façons pour une application de contrôler un appel :

- L'application peut demander d'être notifiée de l'occurrence d'appels vérifiant certains critères. Lorsque de tels appels sont observés dans le réseau, l'application est notifiée et peut alors contrôler l'appel. C'est le cas par exemple pour des appels numéro vert. Lorsqu'un usager tente d'établir un appel numéro vert, il émet une demande au réseau qui notifie l'application numéro vert après avoir analysé que le préfixe du numéro est "0800". L'application peut alors traduire le numéro vert en numéro physique et contrôler l'appel déjà créé dans le réseau pour le router vers cette destination physique.
- L'application peut créer un nouvel appel directement. Par exemple une application Click-to-Dial crée directement un appel entre deux participants, i.e., l'appelant ayant cliqué sur un bouton d'une page WEB et une destination qui correspond par exemple à un centre d'appel.

La suite décrit en particulier les interfaces des SCFs GCC et MPCC.
Les interfaces de la SCF Generic Call Control sont ipCallControlManager et ipAppCallControlManager

ipCallControlManager

ipCallControlManager représente l'interface de gestion de la SCF GCC. Le programmeur d'application l'utilise afin de créer des objets "call", de souscrire à des événements ou d'appliquer un contrôle de la charge (Figure 6).

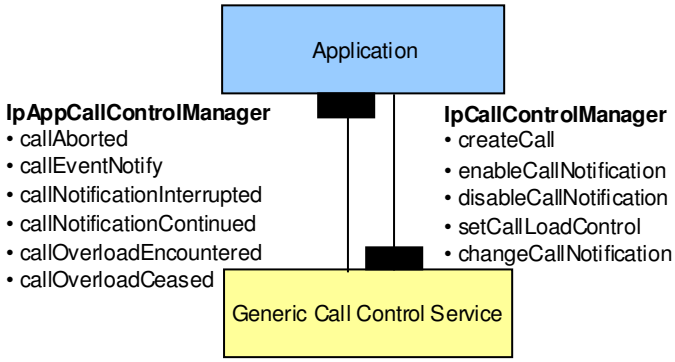

Figure 6 : Interfaces ipCallControlManager et ipAppCallControlManager

La liste des méthodes supportées par l'interface ipCallControlManager est décrite ci-dessous.

createCall : Cette méthode permet de créer un nouvel objet "call". Un objet "call" peut être créé de deux manières : par l'application (e.g., application click-to-talk) à travers l'envoi explicite d'une méthode createCall ou par la SCF GCC elle-même dans le cas où l'appel a été initié par un appelant particulier sur le réseau (e.g., application numéro vert ou prepaid). Dans ce dernier cas, la SCF crée implicitement un objet "call".

enableCallNotification (eventCriteria) permet d'activer la remontée d'événements afin que les événements qui vérifient les critères spécifiés par la méthode soient notifiés à l'application (e.g., être notifié pour tout appel devant être établi vers une destination dont le numéro commence par 0800). C'est la première étape par laquelle doit passer l'application afin de recevoir des événements relatifs à des appels établis dans le réseau et pour lesquels la SCF GCC a créé implicitement un objet "call". Cette méthode ne s'applique pas si l'objet « call » a été établi par l'application. Le Parlay Gateway recevant une méthode enableCallNotification sur l'interface ipCallControlManager doit programmer le réseau intelligent afin que le SSP puisse notifier le Parlay Gateway lorsqu'un appel 0800 est reçu par le SSP. Si le réseau sous-jacent est le RTCP supportant le Réseau Intelligent, le parlay Gateway traduit la méthode enableCallNotification en une requête délivrée au SMP (Service

Management Point) afin que ce dernier puisse armer les points de détection appropriés sur le SSP (Figure 7).

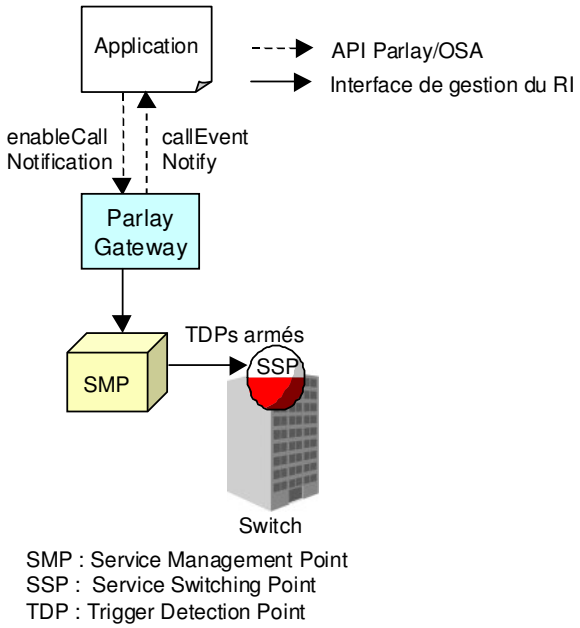

Figure 7 : Souscription d'événements

disableCallNotification permet à l'application de désactiver la remontée d'événements.

setCallLoadControl impose ou désactive le contrôle de charge pour des appels effectués vers une destination particulière. Il s'agit d'une fonction équivalente à l'opération CallGap du protocole INAP ou CAP.

changeCallNotification (eventCriteria) permet à l'application de changer les critères d'événements positionnés initialement avec enableCallNotification.

ipAppCallControlManager

L'interface ipAppCallControlManager permet à l'application de recevoir des réponses aux demandes qu'elle a transmis sur l'interface ipCallControlManager. Les méthodes supportées par cette interface sont les suivantes:

callAborted : Cette méthode indique à l'application que l'objet "call" (instancié sur le Gateway Parlay) a été interrompu anormalement. L'application ne pourra plus communiquer avec cet objet.

callEventNotify (eventInfo) notifie l'occurrence d'un événement relatif à un appel.

callNotificationInterrupted indique à l'application que la remontée de notifications a été momentanément suspendue (e.g., suite à la détection de fautes).

callNotificationContinued indique à l'application la reprise de la remontée d'événements.

callOverloadEncountered indique que le réseau a détecté une surcharge et a pu imposer automatiquement un contrôle de charge pour des appels dont la destination appartient à une plage d'adresse particulière ou pour des appels effectués vers une destination particulière.

callOverloadCeased indique que le réseau a détecté une cessation de surcharge et a automatiquement supprimé le contrôle de charge pour des appels dont la destination appartient à une plage d'adresse particulière ou pour des appels effectués vers une destination particulière.

ipCall

L'interface ipCall (Figure 8) de la SCF GCC permet le routage, la supervision, la libération et la facturation de l'appel. Cette interface ne permet pas de contrôler les legs de l'appel. Ce contrôle du leg n'est possible qu'avec les SCFs MPCC et MMCC.

routeReq (targetAddress): Cette méthode asynchrone requiert le routage de l'appel (et donc des parties rattachées à l'appel) vers une destination particulière, à travers un nouveau « call leg » (créé implicitement). La SCF GCC ne considère que des appels à deux participants. L'objet call est donc constitué implicitement de deux « call legs ». Un « call leg » est routé à travers la méthode routeReq.

release requiert la libération de l'appel et des objets associés. L'appel sera aussi libéré dans le réseau.

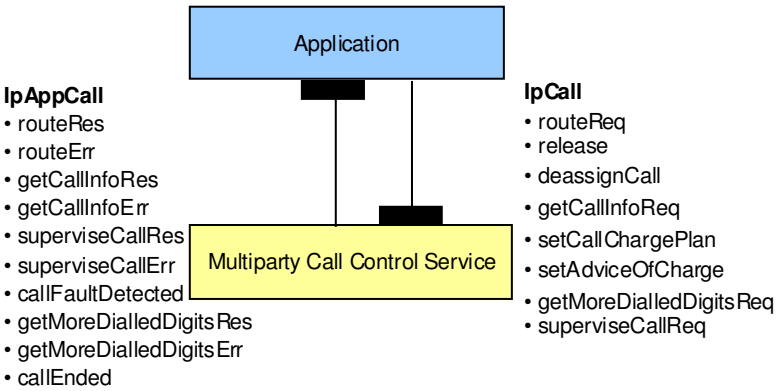

Figure 8 : Interfaces ipCall et ipAppCall

deassignCall permet à l'application de demander à ce qu'elle soit dissociée de l'appel. Cela permet de laisser l'appel actif dans le réseau ; par contre l'objet "call" sur le Gateway Parlay est supprimé afin qu'aucun contrôle ne soit appliqué dorénavant par l'application sur l'appel.

getCallInfoReq demande la fourniture à un moment approprié d'informations d'appel (e.g., à la fin de l'appel dans un but de comptabilité).

setCallChargePlan permet d'affecter un plan de facturation pour l'appel. Le plan de facturation doit être positionné avant que l'appel ne soit routé vers une destination donnée. En fonction de l'opérateur, la méthode peut être utilisée pour changer le plan de facturation pour des appels en cours.

setAdviceOfCharge permet l'envoi d'informations advice of charge (AOC) à des terminaux ayant la capacité de recevoir ces informations.

getMoreDialledDigitsReq permet à l'application de demander des digits supplémentaires.

superviseCallReq demande la supervision de l'appel. L'application peut affecter une durée pour un appel. Si une application invoque cette fonction avant d'appeler la fonction routeReq() ou la fonction user interaction, la mesure du temps commence dès que l'appelé décroche ou des que le système de messagerie vocale répond à l'appel.

ipAppCall

L'interface ipAppCall est fournie par l'application afin de recevoir les réponses aux requêtes émises sur l'interface ipCall ainsi que les rapports d'état.

routeRes : Cette méthode asynchrone indique que la demande de routage de l'appel (routeReq) a été exécutée correctement.

routeErr indique que l'exécution de la requête routeReq a produit une erreur (e.g., le réseau n'a pas pu router l'appel, les paramètres de la demande étaient incorrects, la demande a été refusée, etc.)

getCallInfoRes retourne un rapport d'information d'appel. Elle peut être utilisée à des fins de taxation.

getCallInfoErr informe l'application que la requête getCallInfoReq est erronée ou que son exécution a produit une erreur.

superviseCallRes retourne un rapport de supervision d'appel à l'application.

superviseCallErr informe l'application d'une erreur dans la supervision de l'appel.

callFaultDetected indique à l'application qu'une faute a été détectée dans le réseau. L'appel a pu être terminé ou non. Le système supprime l'objet « call ». C'est pourquoi l'application n'a plus de contrôle sur le traitement de l'appel. Aucun rapport ne sera retourné à l'application.

getMoreDialledDigitsRes retourne à l'application les digits collectés.

getMoreDialledDigitsErr informe l'application d'un problème lors de la collecte de digits.

callEnded() indique à l'application que l'appel a été libéré dans le réseau. Toutefois l'application peut recevoir des information additionnelles sur cet appel (e.g. getCallInfoRes). L'application doit se dissocier de l'objet « call » après avoir reçu la méthode « callEnded ».

2.3. Exemple du service Click-to-Dial

Le diagramme de séquence montre une application Click-to-Dial créant un appel entre deux participants A et B (Figure 9). Cette séquence peut être effectuée une fois que l'internaute a accédé à une page WEB et sélectionné sur cette page un nom ou une organisation à appeler.

1: Ce message est utilisé afin de créer un objet implantant l'interface IpAppCall. Cela permettra à l'application de recevoir des réponses à ces requêtes émises sur l'interface ipCall.

2: Ce message demande à ce que l'objet implantant l'interface IpCallControlManager crée un objet implantant l'interface IpCall.

3: Considérant que les critères permettant la création de l'objet implantant l'interface IpCall sont satisfaits (e.g. valeurs de contrôle de charge n'ont pas dépassé un seuil), l'objet est créé.

4: Ce message est utilisé pour router l'appel vers le participant A (il s'agit de l'internaute qui dispose d'un moyen pour participer à l'appel, tel qu'un téléphone fixe, un téléphone mobile, un téléphone IP, etc.). Dans le message, l'application demande de recevoir une réponse lorsque le participant A décroche.

5: Ce message indique que le participant A a répondu.

6: Ce message relaye le message précédent (5) à l'application.

7: Ce message est utilisé pour router l'appel vers le participant B (appelé). Dans ce message, l'application demande de recevoir une réponse lorsque le participant B répond (décroche).

8: Ce message indique que le participant B a répondu. L'appel est constitué de deux participants et un circuit de parole est établi entre ces derniers (dans le cas de téléphones IP, des canaux RTP seront établis entre les participants).

9: Ce message relaye le message précédent à l'application.

10: Puisque l'application ne souhaite plus contrôler l'appel, celle-ci s'en dissocie. L'appel reste actif dans le réseau, mais il n'y a plus de communication entre l'objet "call" et l'application.

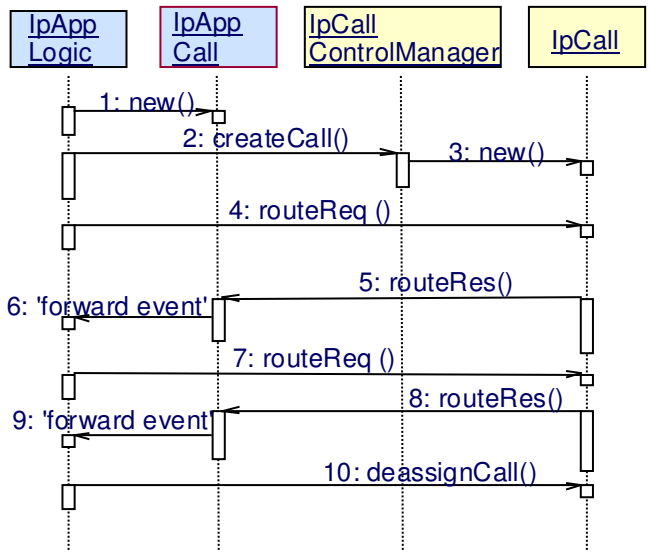

Figure 9 : Flux d'information pour l'exécution du service Cick-To-Dial

2.4. Exemple du service traduction de numéro

Le diagramme de séquence à la figure 10 montre une application de traduction de numéro (e.g., application libre appel ou numéro vert).

1: Ce message est utilisé par l'application afin de créer un objet implantant l'interface IpAppCallControlManager.

2: Ce message est émis par l'application afin d'activer la remontée de notifications relatives à des événements d'appel. Comme le diagramme de séquences décrit une application de traduction de numéros, seuls les événements relatifs à de nouveaux appels dont le numéro appartient à une plage de valeurs seront notifiés.

Lorsqu'un nouvel appel qui satisfait aux critères d'événement définis au message 2 arrive, un message est envoyé à l'objet implantant l'interface IpCallControlManager (non montré sur la 10). Considérant que les critères de création de l'objet implantant l'interface ipCall sont satisfaits, d'autres messages (non montrés) sont alors utilisés afin de créer l'appel et l'objet « call leg ».

3: Ce message est utilisé afin de passer l'événement de nouvel appel à l'objet implantant l'interface IpAppCallControlManager.

4: Ce message est utilisé afin de relayer le message 3 à l'application.

5: Ce message est utilisé par l'application afin de créer un objet implantant l'interface IpAppCall.

6: Ce message invoque la fonction de traduction de numéro.

7: Le numéro traduit est inséré dans le message 7 afin de router l'appel vers la destination.

8: Ce message retourne le résultat du routage de l'appel.

9: Ce message est utilisé afin de relayer le message précédent à l'application.

10: L'application ne souhaitant plus contrôler l'appel, s'en dissocie. L'appel reste actif dans le réseau, mais il n'y a plus de communication entre l'objet "call" et l'application.

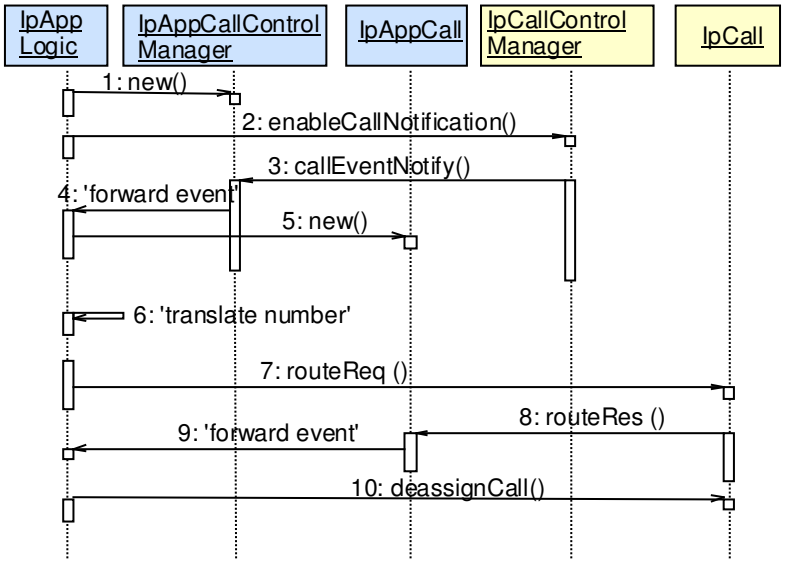

Figure 10 : Flux d'information pour l'exécution d'une application de traduction de numéro

2.5. Exemple d'application Parlay

Afin d'illustrer les interactions entre les applications et les services, l'exemple suivant décrit une application qui utilise la SCF de messagerie et la SCF de mobilité. Il s'agit d'un service de réservation de taxi permettant à un utilisateur d'être mis en relation avec le taxi disponible, le plus proche de sa localisation. Aussi et afin d'évaluer le pourcentage d'usager non servi, la centrale de réservation désire recevoir un message électronique contenant les détails de l'appel et ce à chaque appel non abouti. Préalablement et avant de décrire les échanges qui se déroulent à chaque appel, les hypothèses suivantes sont considérées:

- L'application dispose de la liste des numéros de téléphones mobiles des chauffeurs de taxis concernés par le service ainsi que de l'adresse e-mail du service de réservation.
- La phase d'authentification et d'accès est déjà effectuée à l'aide du « framework Parlay ».
- Le service dispose d'un numéro d'accès unique connu par les usagers du service.
- L'agent de messagerie de l'application est prêt à envoyer des messages.

L'application maintient régulièrement la position géographique de l'ensemble des taxis. A cet effet, elle utilise la fonctionnalité de « notifications lors d'un changement de localisation» de la SCF mobilité. Ceci signifie qu'à chaque changement de localisation d'un véhicule, l'application est notifiée (*triggedLocationReportingStartReq()* et *triggedLocationReport()*). Ainsi, une carte qui reflète les positions réelles des véhicules est maintenue par l'application.

Lorsqu'un client appelle le service, l'application reçoit une notification par la SCF CC (*enableCallNotification()* et *callEventNotify()*). Elle effectue alors une interrogation à la SCF mobilité pour déterminer la localisation du client (*locationReportReq()* et *locationReportRes()*). Une fois cette dernière identifiée, l'application cherche d'après sa connaissance de la position des taxis le numéro de mobile du chauffeur le plus proche et actuellement disponible, et route l'appel vers ce numéro (*routeReq()*). Le routage de l'appel est effectué en demandant d'être notifié si le correspondant est occupé, absent ou non joignable. Si l'un de ces deux états se présente (*routeRes()*), l'application transmet un e-mail au centre de réservation en y incluant toutes les informations de l'appel (*putMessage()*).

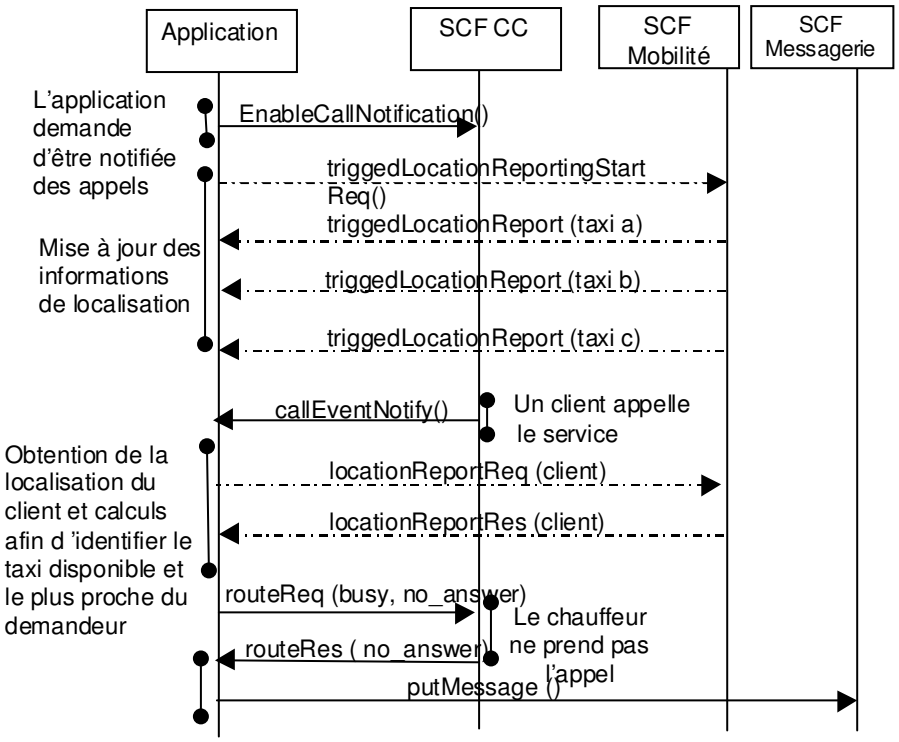

Figure 11 : Exemple d'application Parlay : Réservation de taxi

Le diagramme de séquences présenté dans la figure 11 ne permet pas de montrer les détails des interfaces utilisées. Par exemple, pour la commande d'appels, les interfaces *IpCallControlManager* et *IpCall* sont celles utilisées du coté du service. Du coté des applications, les interfaces correspondantes sont *IpAppCallControlManager* et *IpAppCall*. L'échange *routeReq()*/*routeRes()* se fait entre *IpCall* et *IpAppCall*.

3. Conclusion

PARLAY offre un cadre assez riche pour la construction des applications à forte valeur ajoutée dans le contexte des réseaux de nouvelle génération. Il permet ainsi d'assurer une certaine indépendance par rapport aux technologies et de combiner des fonctionnalités de plusieurs services. Aussi, le cadre PARLAY est facilement extensible et permet de suivre rapidement les évolutions technologiques. L'architecture OSA est un exemple d'extension de PARLAY qui a eu pour objectif de l'adapter à la 3G. Enfin, le framework offre des facilités qui

déchargent les SCF des procédures communes et supporte l'accès aux services pour des fournisseurs tiers.

Références

- ETSI ES 201 915, Open Service Architecture / Parlay

Part 1: "Overview";
Part 2: "Common Data Definitions";
Part 3: "Framework";
Part 4: "Call Control SCF";
Part 5: "User Interaction SCF";
Part 6: "Mobility SCF";
Part 7: "Terminal Capabilities SCF";
Part 8: "Data Session Control SCF";
Part 9: "Generic Messaging SCF";
Part 10: "Connectivity Manager SCF";
Part 11: "Account Management SCF";
Part 12: "Charging SCF".
URL : http://www.parlay.org