

**МИНИСТЕРСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ ПО СВЯЗИ И  
ИНФОРМАТИЗАЦИИ  
САНКТ-ПЕТЕРБУРГСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНАЦИЙ  
им. проф. М.А. БОНЧ-БРУЕВИЧА**

---

*А.А. Зарубин*

**МЕТОДИЧЕСКОЕ ПОСОБИЕ**  
**по проведению**  
**практических занятий**  
**по дисциплине**  
**«Концепции предоставления**  
**современных инфокоммуникационных**  
**услуг»**  
**200900 – Сети связи и системы коммутации**

Санкт-Петербург

2002

УДК 621.391.18:658.512.22

План УМД кафедры СКИРИ на 2002/2003 учебный год

МЕТОДИЧЕСКОЕ ПОСОБИЕ

по проведению

практических занятий

по курсу

Концепции предоставления современных инфокоммуникационных услуг

для студентов, обучающихся специальности

2009 – Сети связи и системы коммутации

Автор: А.А. Зарубин

В данном методическом пособии излагаются основы концепции предоставления дополнительных услуг связи, приводятся примеры реализации элементов услуг при помощи технологии OSA.

Рецензенты: к.т.н., доцент В.И. Исаев

к.т.н., с.н.с. А.Л. Суховицкий

Издание утверждено на заседании кафедры систем коммутации и распределения информации \_число месяц\_ 2002 г. Протокол № \_номер протокола\_.

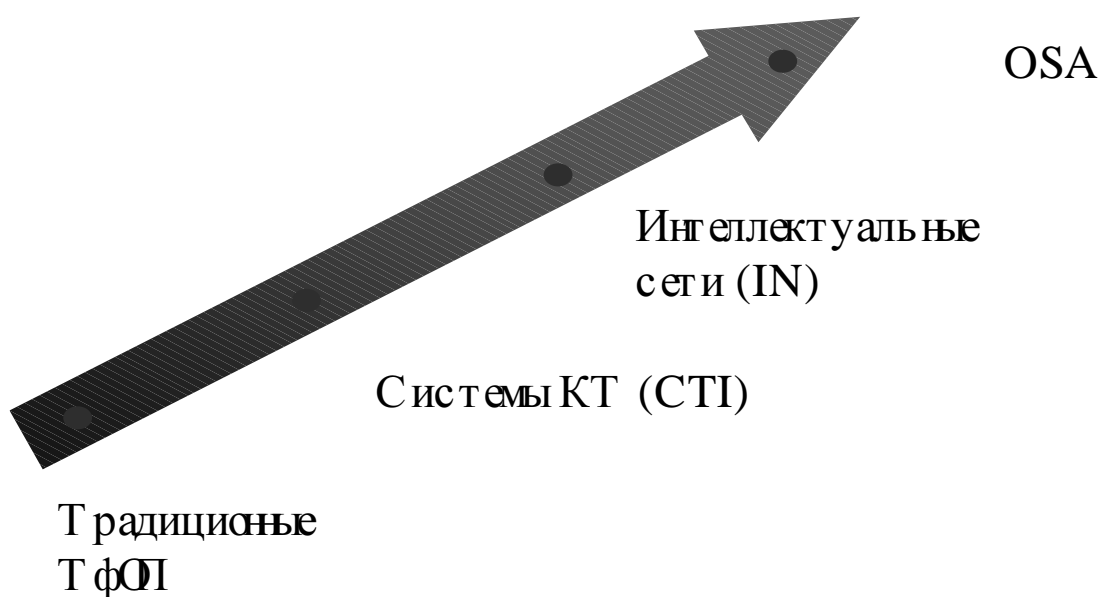
## **Содержание**

- 1 Концепция OSA, Parlay API
- 2 Элементы концепции OSA при использовании Parlay API
- 3 Основные интерфейсы Parlay
- 4 Реализация элементов услуг на Parlay API
- 5 Примеры услуг реализованных с применением Parlay API

## Введение

В 60-х годах XX века телекоммуникационные операторы развитых стран столкнулись со снижением темпов роста доходов от предоставления традиционного набора услуг. Обычные телефонные соединения уже не могли удовлетворить потребности абонентов и клиентов делового сектора, в результате чего началось активное развитие систем предоставления дополнительных услуг.

Спрос на подобные услуги постоянно растет, а на протяжении более чем 40 лет возникло несколько, в чем-то пересекающихся, концепций их предоставления. Рис. 1 демонстрирует последовательность развития систем предоставления услуг.



*Рис. 1. Развитие систем предоставления услуг*

Первоначально дополнительные услуги реализовывались с использованием возможностей существовавших в то время ТфОП. Однако с усложнением предоставляемых услуг, уже в конце 60-х годов возникли первые системы интегрирующие телефонные сети и компьютерные технологии. Эта концепция получила название компьютерной телефонии.

Системы КТ не лишены недостатков. Разработка и внедрение новых услуг может занимать продолжительное время, а богатые сетевые ресурсы используются не настолько, насколько могли бы быть задействованы.

Следующим шагом в развитии идей предоставления услуг стала концепция Интеллектуальных сетей. Концепция IN предполагает разделение всех функций создания, модификации, предоставления услуг и эксплуатационного управления ими на несколько блоков, взаимодействие между которыми обеспечивается через стандартные интерфейсы.

Все элементы этой архитектуры находятся в собственности и используются одним оператором, а механизмы предоставления услуг располагаются в области сети. Однако и такой подход имеет несколько существенных недостатков для применения на современном рынке. Так как один оператор ответственен за создание и работу всех приложений, трудно достичь необходимой гибкости, путь ввода новых приложений в эксплуатацию все ещё занимает продолжительное время.

По информации зарубежных аналитиков поддержка величины среднего дохода оператора в расчете на одного абонента на постоянном уровне в течении 2000 – 2007 годов будет возможна только с постоянным вводом в эксплуатацию новых инфокоммуникационных услуг включающих в себя услуги передачи данных, мобильности, универсальной обработки сообщений и электронной коммерции.

Предлагаемое пособие посвящено рассмотрению концепции, которая может реализовать единые средства и способы управления различными сетями и позволить использовать уже существующие и вновь разрабатываемые средства связи, исключив недостатки предыдущих способов предоставления инфокоммуникационных услуг.

Эта концепция получила название OSA, что означает Open Service Architecture или Open System Access, последнее соответствует официальным рекомендациям. Идеи, заложенные в концепцию OSA, уже начинают использоваться при переходе к NGN – сетям следующего поколения.

## **1. Концепция OSA, Parlay API**

Современный рынок телекоммуникаций находится в состоянии жесткой конкуренции и динамичного развития, преимущество имеют те операторы, которые могут предоставить своим пользователям множество разнообразных, нужных услуг.

В настоящее время известно несколько концепций, позволяющих оператору предоставлять своим абонентам дополнительные услуги, к ним относятся CTI и IN.

Компьютерная телефония, как процесс интеграции внешних автономных или находящихся в составе сети компьютерных систем с телефонной сетью и узлами коммутации, зародилась в конце 60-х годов, а в середине 70-х у инженеров Bell Labs возникла идея интеллектуальных сетей. С того времени эти два направления предоставления услуг постоянно развиваются и широко применяются в мире телекоммуникаций.

В начале 90-х годов возникла необходимость в технологии, которая реализовала бы единые средства и способы управления различными сетями и позволила бы использовать уже существующие и вновь разрабатываемые средства связи. В работу над созданием такой технологии включилось множество компаний-производителей телекоммуникационного оборудования и операторов связи.

На протяжении нескольких лет различными организациями предлагалось несколько вариантов решения этой задачи, пока, наконец, в 1998 году не был сформирован консорциум Parlay Group, занимающийся созданием спецификаций открытого API (интерфейса прикладного программирования), позволяющего управлять сетевыми ресурсами и получать доступ к сетевой информации.

Кроме этого консорциума разработкой подобных технологий занимались и другие организации, а в начале 2001 года, с созданием спецификаций Parlay API версии 2.1, ETSI, 3GPP и Parlay Group объединили свои усилия, в результате чего были созданы полностью согласованные спецификации Parlay 3.0, ETSI 1.0 и 3GPP 4.2.

Идея концепции, которую реализует Parlay API, заключается в следующем. Элемент управления сетью программируется через открытый интерфейс. Сам интерфейс реализован таким образом, что программа приложения (услуги) не зависит от протоколов и технологий, применяемых внутри самой сети, на столько, на сколько это возможно и необходимо.

Сервера, на которых находятся программы приложений (услуг) и сеть, через которую абоненту предоставляется услуга, могут принадлежать различным операторам. Их можно определить как «оператор (провайдер) услуг» и «сетевой оператор».

Концепция получила название OSA, в различных источниках эта аббревиатура раскрывается как Open Service Architecture или Open System Access, последнее соответствует рекомендациям ETSI.

Кроме Parlay Group работы в этом направлении ведутся также группой компаний во главе с Sun Microsystems, их разработка получила название JAIN – Java Advanced Intelligent Network.

Спецификации JAIN более широкие, они охватывают интерфейсы, не рассматриваемые в спецификациях Parlay. Кроме того, JAIN это попытка практической реализации концепции OSA, спецификации же Parlay API создавались независимыми от технологий их реализации. Технология JAIN Service Provider API (SPA) позиционируется как Java-реализация Parlay API, однако, изначально основанная на спецификациях Parlay, JAIN SPA, тем не менее, имеет некоторые отличия (применение Event Listeners вместо Call Backs и другие), но описание подобных тонкостей выходит за рамки пособия.

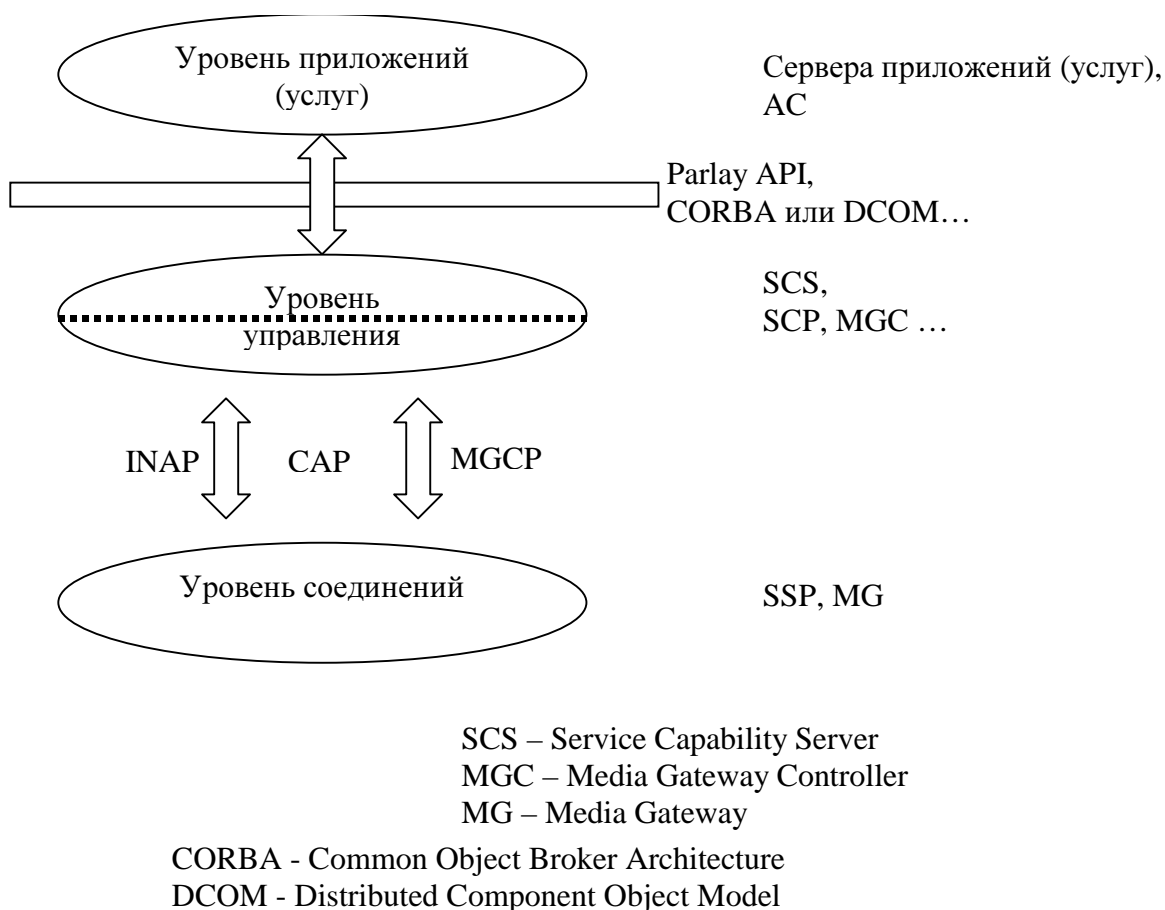


Рис. 2. Концепция OSA

В упрощенном виде схема концепции Parlay/OSA представлена на рис. 2. Рис. 3 демонстрирует разделение сети построенной с применением технологий OSA на области управления между различными операторами.

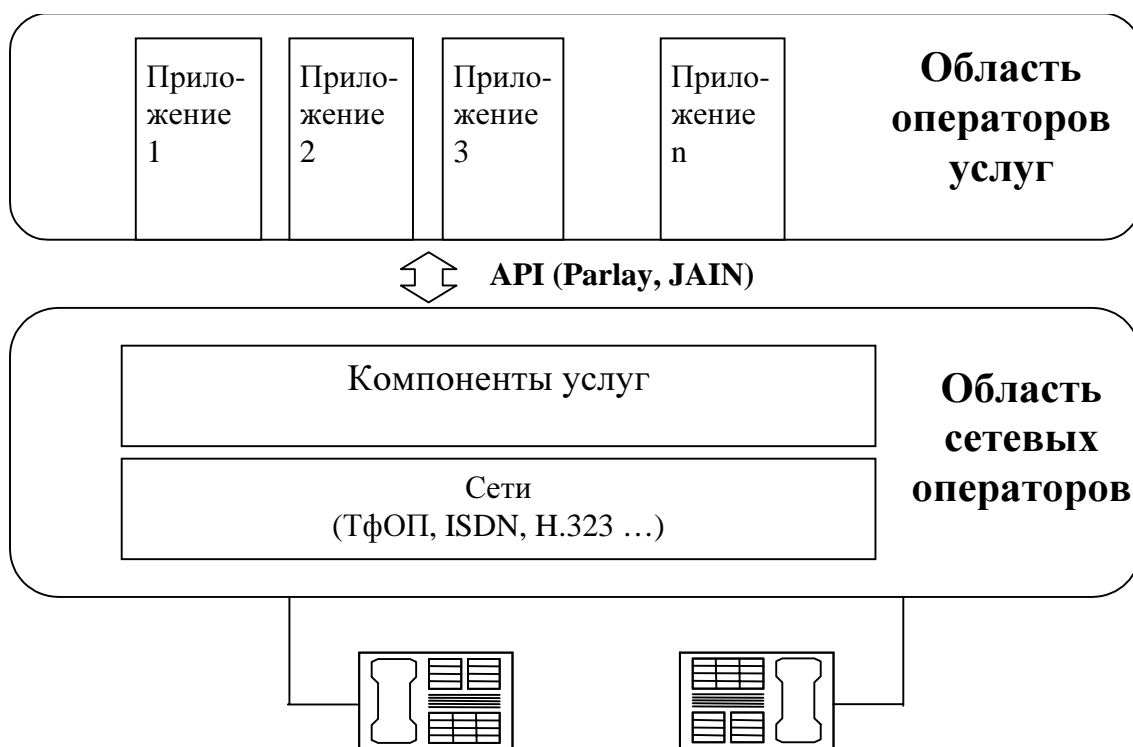


Рис. 3. Области управления элементами концепции OSA

Создатели технологии и разработчики оборудования утверждают, что применение концепции OSA принесет выгоду всем сторонам телекоммуникационного рынка. Приводятся такие доводы:

- Сетевые операторы получают рост трафика, как следствие ввода новых услуг, взаимодействие с операторами услуг станет проще;
- Применение Parlay API позволит операторам услуг предоставлять пользователям новые, ранее не существовавшие возможности;
- Станет легче создавать комплексные услуги, использующие ресурсы различных сетей;
- Гораздо меньше материальных затрат потребуется для того, что бы организовать предоставление услуг, уменьшится время, необходимое для ввода востребованных услуг в эксплуатацию;
- Разработка приложений будет проще и доступна большему количеству компаний;
- Все перечисленное пойдет на благо конечного пользователя.

Теперь обратим внимание на основные характеристики и особенности Parlay API.

Позволяя серверам приложений (услуг) управлять сетью извне, он, тем не менее, обеспечивает безопасность сети, это достигается через аутентификацию приложений, применение межсетевых экранов.

Так же API расширяем, например, по сравнению с версией 2.1, третья версия обладает возможностями получения информации о терминале пользователя, работой с голосовой почтой и E-mail, взимания оплаты с конечных пользователей и др. Однако при этом отсутствует совместимость этих версий, этот недостаток будет устранен в дальнейшем развитии Parlay. Как уже говорилось, сервера приложений могут находиться вне области сетевого оператора. Это обеспечивается с помощью технологий построения распределенных информационных систем, таких как CORBA или DCOM. Parlay – объектно-ориентированный API, CORBA (и DCOM) же позволяет вызывать методы у объектов, находящихся в сети где угодно, как если бы все они были локальными объектами.

Читателям, знакомым с идеей Softswitch концепция Parlay/OSA должна показаться знакомой. Действительно, если рассматривать Softswitch как универсальное устройство управления различными сетями, то он, в свою очередь, может управляться серверами приложений через Parlay API.

На рис. 4 представлена упрощенная архитектура Parlay API.

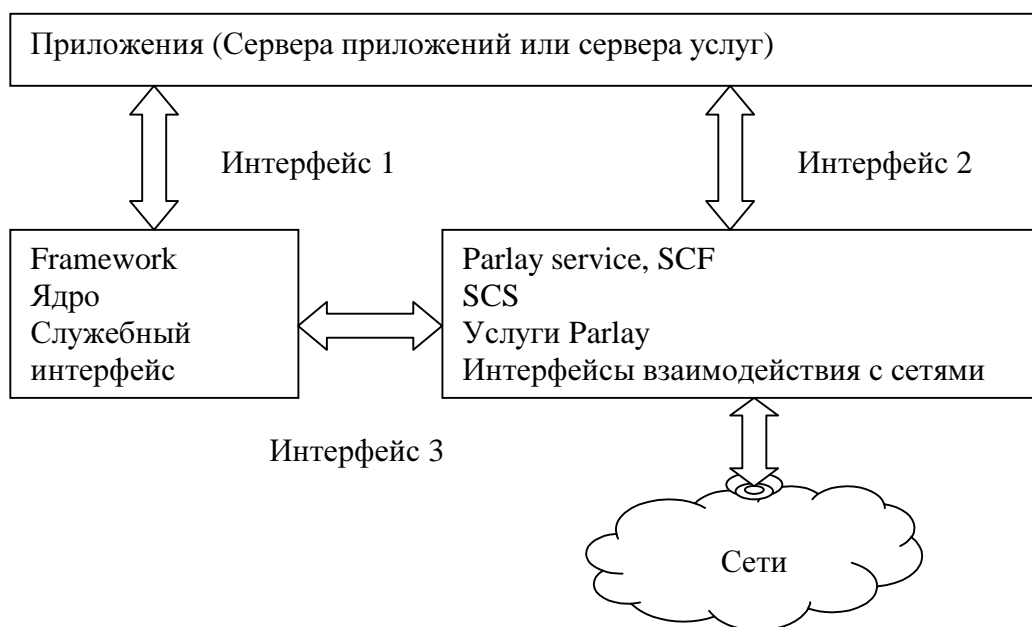


Рис. 4. Упрощенная архитектура Parlay API

В документах ETSI, касающихся OSA, говорится: «Спецификации OSA определяют архитектуру, позволяющую [...] использовать функциональность (возможности) сети

через открытые стандартные интерфейсы. Функциональность (возможности) сети представляется как SCF – Service Capability Feature или Службы Parlay. OSA Framework (Ядро или служебный интерфейс) является основным компонентом для обеспечения работы приложений и служб Parlay. [...] Концепция OSA определяет набор SCF». При этом спецификации не определяют преобразование методов (программных функций) Parlay API в сообщения протоколов INAP, CAP и др.

Интерфейс 1 обеспечивает основные механизмы взаимодействия приложений с Framework, таких как аутентификация приложения, уведомление приложения о существующих SCF и т.п.

Интерфейс 2 позволяет приложению использовать конкретные возможности сетей.

Интерфейс 3 служит для обеспечения возможности динамического подключения новых служб Parlay (SCF). В спецификациях OSA сказано, что это необходимо для предоставления служб Parlay (SCF) различными операторами.

Под интерфейсами здесь понимаются наборы классов данного API, содержащие различные методы (программные функции).

Интерфейсы реализуются определенным функциональным элементом сети, называемым Service Capability Server (SCS) – сервер обеспечения возможности реализации услуг. Иногда в литературе его называют «gateway».

В свою очередь, приложения располагаются на так называемых серверах приложений – Application Server (AS), соединенных с SCS через IP-сеть (здесь же может применяться и межсетевой экран).

## **2 Элементы концепции OSA при использовании Parlay API**

Итак, было дано общее описание Parlay-подобных технологий. Теперь рассмотрим сервера обеспечения возможности реализации услуг (Service Capability Server), их место в модели Parlay/OSA.

Рис. 5 с незначительными изменениями встречается во множестве зарубежных статей и докладов, посвященных Parlay/OSA, он хорошо иллюстрирует отношения между AS, SCS, SCF и элементами сетей.

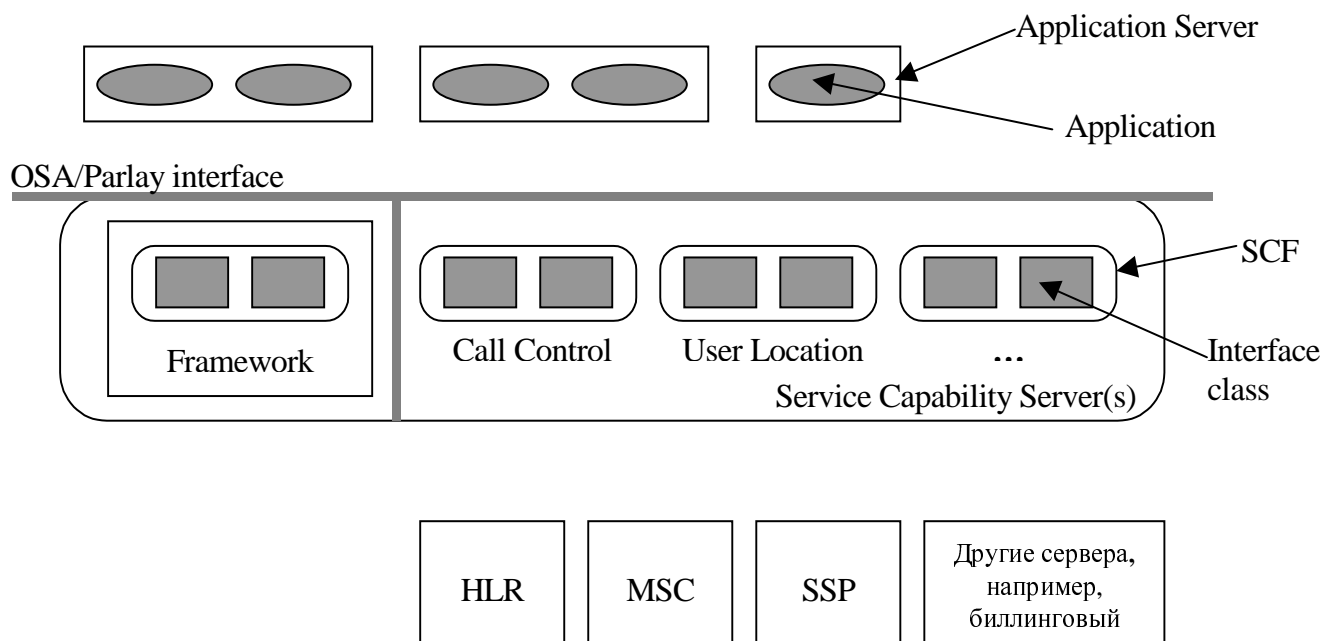


Рис. 5. Элементы концепции OSA

Будучи для серверов приложений одним логическим элементом, Service Capability Server физически может быть разнесен по сети или же находится на одном узле.

Некоторые из множества вариантов физического построения SCS представлены на рис. 6.

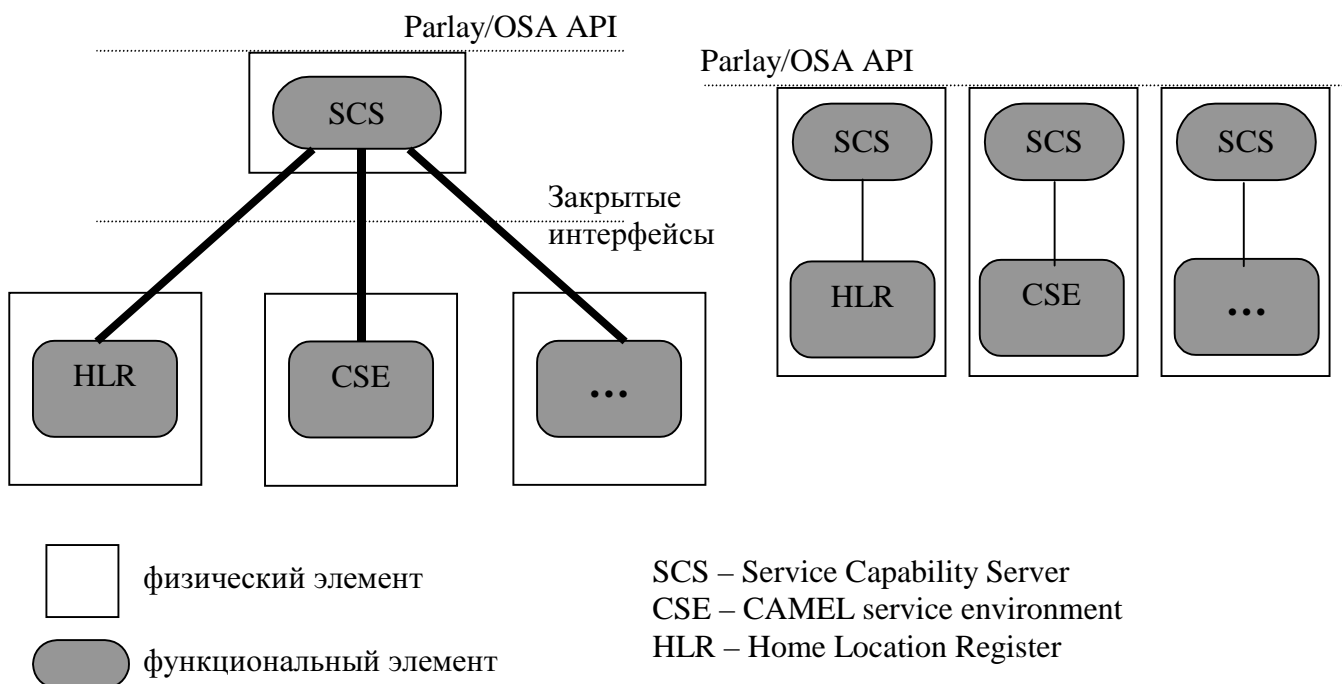


Рис. 6. Построение SCS

Наиболее вероятен вариант, когда SCS располагается на SCP (или CSE – узле, соответствующем SCP в концепции интеллектуальных сетей) и взаимодействует со всеми остальными элементами сети по стандартным протоколам сигнализации.

В любом случае такая система будет содержать только один интерфейс Framework, который может быть реализован как отдельный физический элемент сети или физически представлять единое целое с другими SCF. С точки зрения приложений построение SCS не важно, в любом случае приложению будет предоставлен список всех возможностей (SCF) через Framework и доступ к ним.

Все возможности доступные приложениям, использующим Parlay API, разделены на несколько SCF, каждому SCF соответствует определенный документ в спецификациях Parlay/OSA. Кратко рассмотрим интерфейсы третьей версии API.

### *Framework*

Основной интерфейс, ядро концепции, присутствует всегда. Надежности его работы должна быть не менее 99.999%. Для обеспечения определенных функций во Framework реализованы интерфейсы, называемые “Framework Interfaces”, они включают:

- Trust and Security Management. Средства (интерфейсы или наборы программных функций для) обеспечения безопасности. Т.е. механизмы для первоначального взаимодействия приложений с SCS и аутентификации приложений;
- Service Access. Средства обеспечения авторизации для доступа приложений к конкретным службам Parlay (SCF);
- Service Discovery. Средства информирования приложений о доступных возможностях сети;
- Service Registration. Средства автоматической регистрации (подключения и внесения в общий список) служб Parlay;
- Event Notification. Средства уведомления приложений о различных событиях;
- Integrity Management.
  - Load Management. Управление нагрузкой (загрузкой приложений и SCFs);
  - Fault Management. Обнаружение ошибок, сбоев;
  - Heartbeat Management - средства определения работоспособности приложений;
  - OAM – интерфейс используется для согласования (запроса) системных даты и времени. Приложение запрашивает у SCS.

- Service Subscription - интерфейс, применяемый, когда приложению оператора услуг, перед тем как воспользоваться службами Parlay, необходимо подписаться на них (абонировать их) у оператора Framework.

### *Call Control*

Семейство функций управления вызовами. Начиная от установления простейшего соединения до управления мультимедийной конференц-связью.

### *User Interaction*

Интерфейс используется для получения информации от пользователя, проигрывания ему подсказок.

### *Mobility SCF (User location / User status)*

Набор методов данного интерфейса может применяться для получения информации о местоположении пользователя и состоянии его терминала.

### *Terminal capabilities*

Применяется для получения информации о возможностях терминала пользователя.

### *Data Session Control*

Интерфейс используется для организации и управления сеансами передачи данных, в т.ч. предопределения возможных объемов передачи данных.

### *Generic Messaging*

Интерфейс для работы с электронной и голосовой почтой, используется для организации приема, передач и хранения информации.

### *Connectivity Management*

Методы интерфейса применяются для обеспечения реализации функций качества обслуживания в пакетных сетях.

### *Account management*

Интерфейс применяется для работы со счетами пользователей, например, проверки баланса или получения истории транзакций.

### *Content based charging*

Методы интерфейса также используются для работы со счетами пользователей, позволяют осуществлять резервирование необходимой суммы при предоставлении конкретной услуги, установления (продления) времени её предоставления, проводить операции дебета/кредита со счетом пользователя.

## **3 Основные интерфейсы Parlay API**

Теперь более детально рассмотрим SCF, составляющие Parlay API второй версии и включенные в третью.

### **3.1. Framework - ядро**

Этот интерфейс (или SCF) обеспечивает возможности, необходимые для того чтобы интерфейсы услуг были открытыми, безопасными, гибкими в использовании и вводе в эксплуатацию, а также управляемыми. Framework - часть Parlay API, обеспечивающая функциональность, независимую от вида предоставляемой услуги.

#### **3.1.1 Trust and Security Management - Средства обеспечения безопасности**

Этот интерфейс обеспечивает первый контакт приложения и SCS, аутентификацию приложений, возможность выбора приложением необходимых ему SCF.

Процесс взаимодействия приложения и Framework делится на три этапа – подключение (initial contact), аутентификация и доступ к SCF.

#### **Initial contact**

При первом подключении приложения инициируется процесс аутентификации, необходимый для обеспечения возможности дальнейшего взаимодействия приложения и SCS.

#### **Аутентификация**

Для того чтобы приложение могло взаимодействовать с различными интерфейсами должна пройти его аутентификация с framework. API поддерживает различные способы аутентификации.

#### **3.1.2 Service Access**

После аутентификации приложение получает возможность взаимодействовать с различными SCF. Для использования же определенного SCF, приложение должно быть авторизовано через установление соглашения между оператором приложения и оператором framework (SCS).

### 3.1.3 Service Discovery

Сначала приложение должно получить информацию о том, какие типы SCF поддерживаются и каковы их характеристики. Эти данные приложению дает framework. Если приложение находит необходимый набор SCF, оно может подписаться на них через Subscription Interfaces (см. Service Access).

### 3.1.4 Service Registration

Перед тем как работа приложений с определенным SCF становится возможной, SCF необходимо зарегистрироваться на framework. Для каждого SCF определен его service type и определенные параметры. Framework же управляет списком возможных типов и зарегистрированных на нем SCF.

### 3.1.5 Event Notification

Этот механизм необходим для возможности уведомления приложений о событиях, относящихся к базовым SCFs.

### 3.1.6 Integrity Management

#### *Load Management*

Определяет, как должен поступить framework, если изменится уровень загрузки (например, возрастет). Для некоторых приложений это может быть попытка обеспечить обслуживание «любой ценой», а для других – отключение, пока уровень загрузки не придет в норму. Policy load management соотносится с уровнем QoS, на который подписано приложение.

#### *Heartbeat Management*

Служит для инициализации наблюдения за работоспособностью приложений.

#### **OAM**

Интерфейс используется для запроса приложениями системной даты и времени. Т.о. приложения и framework могут синхронизировать время для лучшего взаимодействия.

### 3.1.7 Service Subscription

Эти средства framework могут использоваться, когда для работы с определенным SCF приложению надо сначала подписаться на него. Подписка представляет собой соглашение между оператором приложений и оператором framework.

## 3.2. Call Control – управление вызовами

### 3.2.1 Generic Call Control

GCC содержит основные функции управления вызовами для API.

GCC обеспечивает максимум две стороны (leg) вызова, большее число обеспечивается MultiParty GCC.

Приложение может получить контроль над вызовом двумя способами. Первый заключается в том, что приложение получает от сети уведомления о различных событиях (сначала запросив это) и в случае определенного события получает управление этим вызовом. Другой способ – приложение само генерирует вызовы и управляет ими.

### 3.2.2 MultiParty Call Control

Обладает функциональностью GCC, кроме того, позволяет создавать и управлять вызовами с числом сторон (leg) более двух.

### 3.2.3 MultiMedia Call Control

Функциональность MultiParty Call Control расширяется мультимедийными возможностями. Приложение может устанавливать медиаканалы с заданными характеристиками, следить за установлением и разрушением медиаканалов, разрешать или запрещать их установление (медиаканал это однонаправленный медиапоток, ассоциированный с одной из сторон вызова).

### 3.2.4 Conference Call Control

Расширяет возможности ММСС. Используя ССС приложения могут управлять частями конференций. Часть конференции – это группа сторон вызова, принадлежащих

одному конференц-вызову. Стороны внутри части конференции (subconference) имеют однонаправленные соединения (медиаканалы) друг с другом.

Т.о. приложение может: создавать новые части конференций, делить часть конференций на более мелкие, перемещать стороны вызовов между частями конференций, осуществлять слияние частей конференций, получать список всех частей конференций в вызове.

Данный SCF может работать по H.323, управлять MSC, заведовать «conference policy», т.е. правилами обработки видео, аудио и текстовых сообщения.

ССС позволяет приложениям резервировать ресурсы для конференций, освобождать их, искать ресурсы по заданным критериям.

Как и в предыдущем случае, приложение может быть уведомлено о начале (запросе) конференции и получить управление или само инициировать конференц-вызов.

### **3.3. User Interaction – Взаимодействие с пользователем**

Этот интерфейс включает Generic User Interaction и Call User Interaction.

#### **3.3.1 Generic User Interaction**

GUI используется приложениями для взаимодействия с пользователями. Это взаимодействие включает посылку и прием информации от пользователя, например, проигрывание подсказок.

#### **3.3.2 Call User Interaction**

Используется для приема и передачи информации пользователю, к которому установлено соединение. Этот интерфейс используется совместно с Call Control.

### **3.4. Generic Messaging**

GM используется приложениями для приема, хранения и отправки сообщений. С помощью этого интерфейса можно обеспечить работу электронного почтового ящика или голосовой почты.

### **3.5. Mobility (strap. User location / User status)**

Mobility это набор интерфейсов (SCF или Services), которые позволяют приложениям работать с параметрами мобильности пользователя. Этот набор включает:

User Location Service, User Location Camel Service, User Location Emergency Service, User Status Service.

#### 3.5.1 User Location Service

Интерфейс обеспечивает определение географического положения и состояния пользователя фиксированной, мобильной или сети IP-телефонии.

#### 3.5.2 User Location Camel Service

Обеспечивает получение информации о местоположении, основанной на сетевых данных. Эта информация может быть получена быстрее, чем географические координаты через UL.

С помощью ULC можно запросить данные такие данные, как номер VLR, Location Area Identification, Cell Global Identification и т.п.

#### 3.5.3 User Location Emergency Service

ULE используется для обработки экстренных вызовов. В случае такого вызова сеть может определить местоположение пользователя самостоятельно (без запроса со стороны приложения) и сразу послать приложению, ответственному за обработку экстренных вызовов, эту информацию. Если по каким-то причинам сеть не может быстро определить местоположение пользователя, ему будет сообщен номер пользователя.

#### 3.5.4 User Status Service

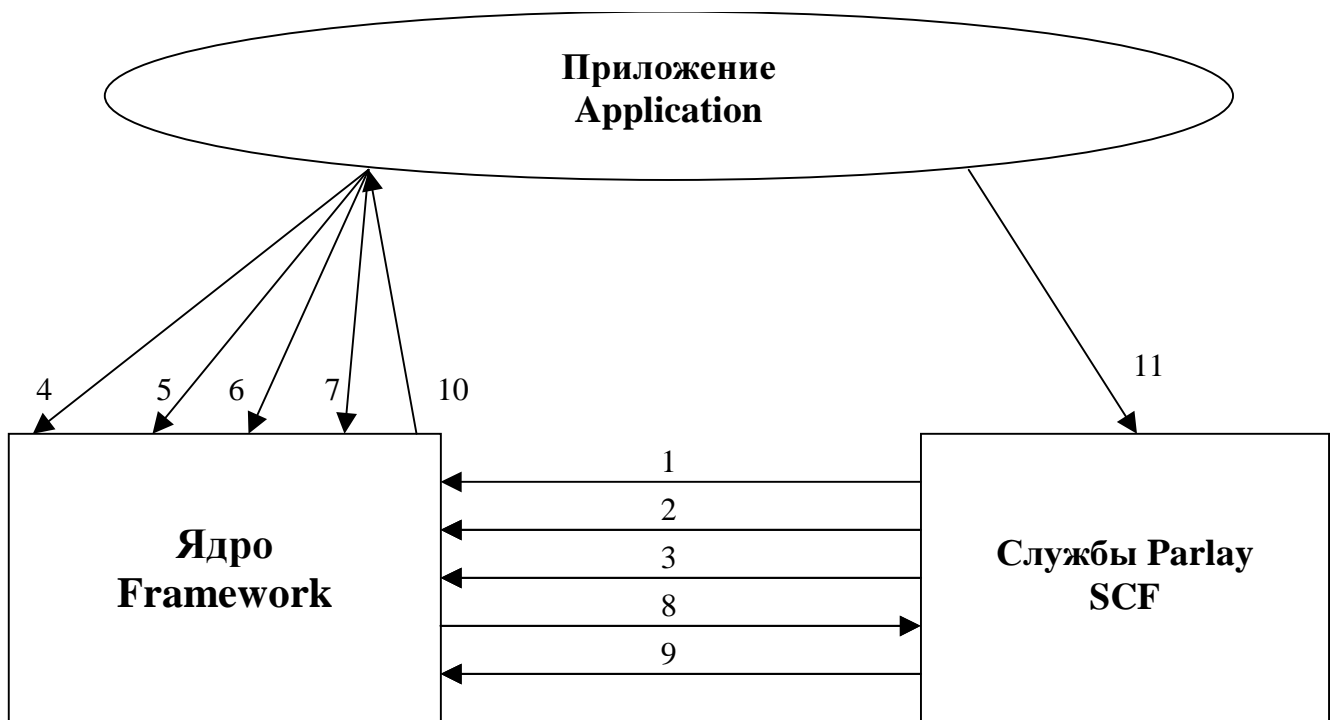
Интерфейс обеспечивает функции определения состояния пользователя мобильной, фиксированной или IP-сети, такие как доступен, недоступен или занят.

### **3.4. Реализация элементов услуг на Parlay API**

Выше мы, в общем, рассмотрели отдельные SCF Parlay. Теперь обратим внимание на то, как приложения взаимодействуют с SCS через SCF.

Далее предлагается схематичное описание цикла взаимодействия Framework, SCS (здесь в смысле реализующего один отдельный SCF) и приложения.

Предположим, что в нашей системе (сети) появляется новый элемент – часть SCS (далее просто SCS), реализующая определенный SCF. Пусть это MultiParty Call Control SCF.



- |                                   |                              |
|-----------------------------------|------------------------------|
| 1: Authentication                 | 6: Discover Service          |
| 2: Request Registration Interface | 7: Select Service & sign SLA |
| 3: Register factory               | 8: Create Service Manager    |
| 4: Authentication                 | 9: Return Service Manager    |
| 5: Request Discovery Interface    | 10: Return Service Manager   |
|                                   | 11: Use Service              |

*Рис. 7. Схема простого взаимодействия Framework, SCS и приложения*

На первом этапе во Framework должен быть зарегистрирован новый SCF. Для этого появившийся SCS связывается со Framework и проводит регистрацию, как это показано на рис. 7 п.п. 1 и 2. Далее, п.п. 3, MultiParty Call Control SCS публикует тип своего SCF и некоторые другие параметры (возможности), например максимальное число сторон, поддерживаемое данной реализацией МРСС.

Далее происходит установка т.н. Service Agreement, т.е. определяются условия, при которых приложениям будет разрешено использовать данный SCS. Например приложению может разрешаться работать с числом сторон не более определенного.

Теперь приложение может соединиться со Framework и использовать Discovery Interface для определения поддерживаемых системой SCF (п.п. 4 и 5). Пусть максимальный запрос приложения это SCF МРСС с поддержкой четырех сторон. Тогда оно запросит у Framework список всех SCS, реализующих МРСС SCF с поддержкой минимум четырех сторон. Framework поверит установленные Service Agreement на предмет допустимости обслуживания этих требований приложения и, если все в порядке,

сообщит ему список всех удовлетворяющих запросу SCS. Например, в этом списке будет только два SCS, поддерживающих MPCC. Один с максимальным числом сторон, равным восьми, а другой с поддержкой максимум шести сторон. Приложение выберет один из SCS, возможно, исходя из того, что дешевле.

В п.п. 8 Framework передаст выбранному SCS запрос на обслуживание приложения и параметра с которыми это обслуживание должно будет происходить (например, разрешение максимум четырех сторон). SCS сообщит Framework подтверждение, а тот, в свою очередь, приложению (п.п. 9). Приложение обратится к SCS (п.п. 10), с этого момента оно может использовать MultiParty Call Control SCF.

Остается добавить, что шаги 1 – 4 в некоторых случаях могут отсутствовать. Так же ничто не мешает использованию firewall, SSL, IPSec и подобных технологий.

Как же будет реализована с применением Parlay API конкретная услуга? Предположим, что нам необходимо создать приложение, которое уведомляет группу пользователей о доступности кого-либо из этой группы или о том, что они находятся рядом.

В упрощенном варианте взаимодействие такого приложения и SCS будет состоять из нескольких этапов:

1. Приложение должно связаться со Framework и, в общем случае, аутентифицироваться, чтобы получить доступ к интерфейсам реализующим Mobility SCF и Call Control;
2. Теперь пользователь группы может подключиться к приложению через Internet или WAP, чтобы настроить его;
3. Когда настройка будет произведена, приложение свяжется со службой Parlay реализующей Mobility SCF и запросит, чтобы ему (приложению) приходили уведомления, когда определенные пользователи становятся доступны (например, включают свои мобильные терминалы) или уведомления о местоположении пользователей группы;
4. В свою очередь, SCS настраивает на сети соответствующие триггерные точки;
5. Каждый раз, когда пользователь из данной группы активирует свой терминал, SCS информируется об этом сетью;

6. SCS уведомляет об активации определенного пользователя соответствующее приложение;
7. В случае активации пользователя группы, приложение уведомит об это других пользователей текстовым сообщением, например, ICQ, SMS или послав вызов с помощью Call Control, тоже может происходить при близком местоположении пользователей;
8. Пользователь может запросить у приложения произвести конференц-вызов доступным пользователям;
9. Приложение использует Charging SCF для работы с пользовательскими счетами, чтобы определить достаточно ли для этого средств на счетах;
10. Наконец, приложение свяжется со службой Parlay реализующей Call Control для установления конференц-вызова.

Рассмотрим подробнее часть приведенного выше примера.

Рис. 8 показывает, какие методы необходимо задействовать приложению, чтобы получать от SCS периодическое уведомление о местоположении пользователя.

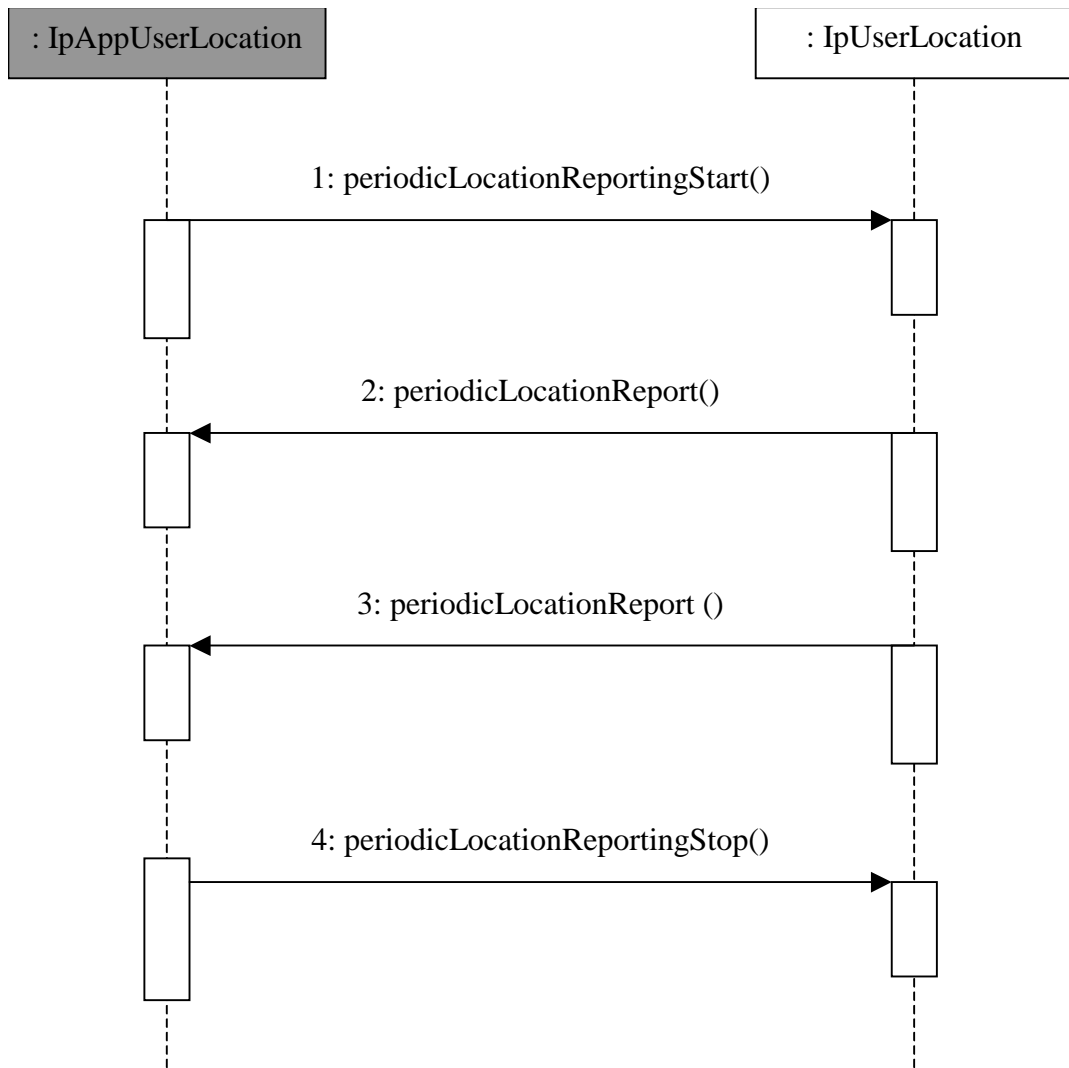


Рис. 8. Методы, применяемые для определения местоположения пользователя

Обмен данными между приложением и SCS изображен в виде диаграммы последовательностей (UML).

Для общего понимания вопроса можно представить, что интерфейсы, закрашенные серым цветом, относятся к стороне сервера приложений, остальные к стороне SCS. Однако в реальности все функции интерфейсов Parlay выполняются на SCS, применяется механизм callback-функций.

На данной диаграмме показано применение интерфейсов Mobility SCF IpAppUserLocation и IpUserLocation. В них входят методы (программные функции) periodicLocationReportingStart(), periodicLocationReport() и periodicLocationReportingStop().

Сообщение (запрос, метод) 1 используется для запуска периодического уведомления приложения о местоположении пользовательского терминала. Сообщения 2 и 3 несут данную информацию, интервал между ними настраивается одним из параметров

сообщения 1. Сообщение 4 является запросом на прекращение периодического обновления.

Часть программы приложения будет выглядеть примерно следующим образом:

```
periodicLocationReportingStart(параметры: идентификатор пользователя, интервал уведомлений и др.);  
periodicLocationReport(параметры: переменная, содержащая данные о местоположении и др.);  
.  
.  
.  
periodicLocationReportingStop(параметры);
```

Теперь рассмотрим комплексный случай, близкий к полной реализации услуги.

Рис. 9 демонстрирует процесс обмена сообщениями при использовании услуги, которая позволяет проиграть в заданный момент подсказку пользователю. Сюда не включен процесс взаимодействия со счетом пользователя.

Интерфейсы, закрашенные серым цветом, реализованы на стороне приложения, остальные на стороне SCS.

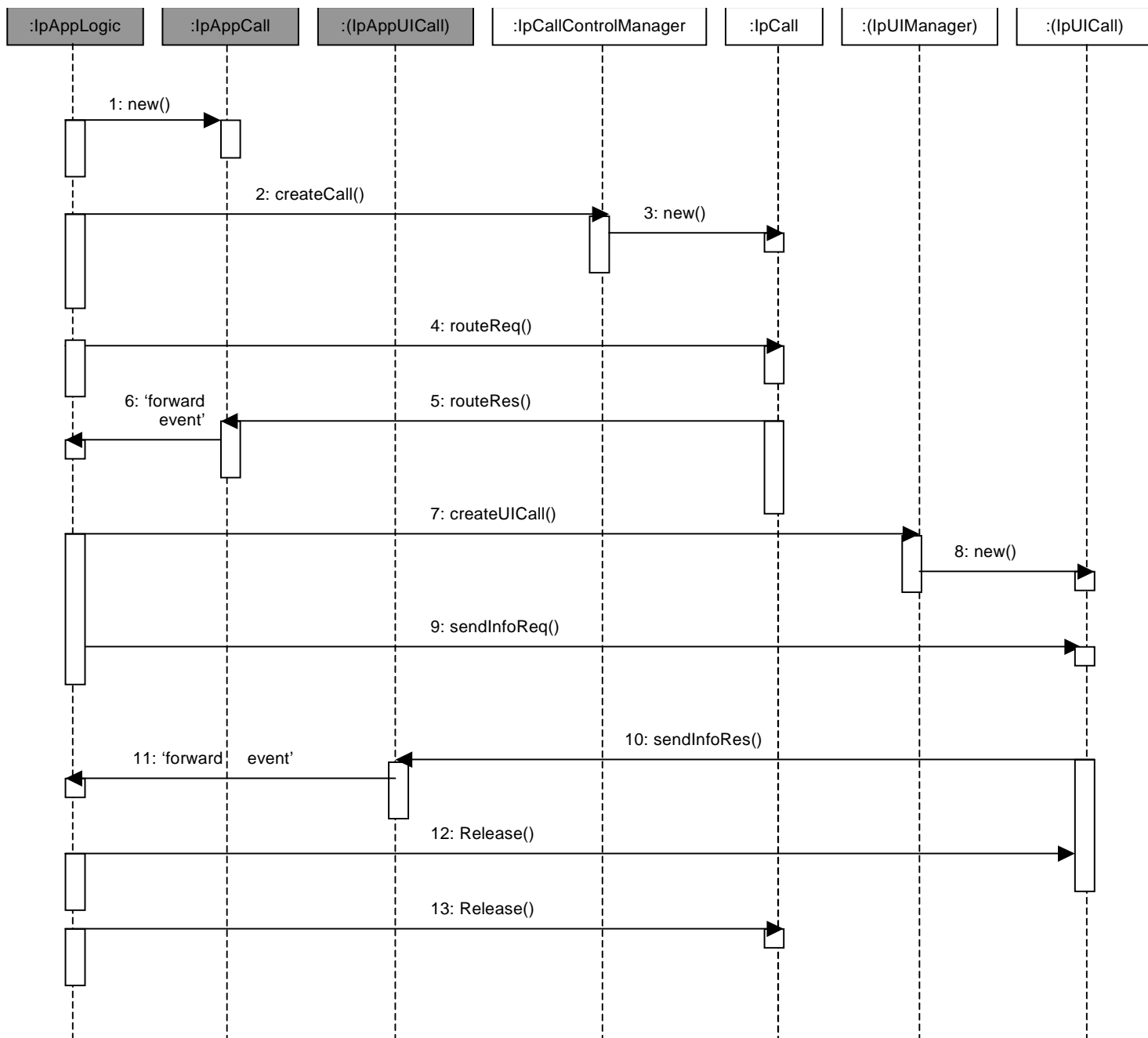
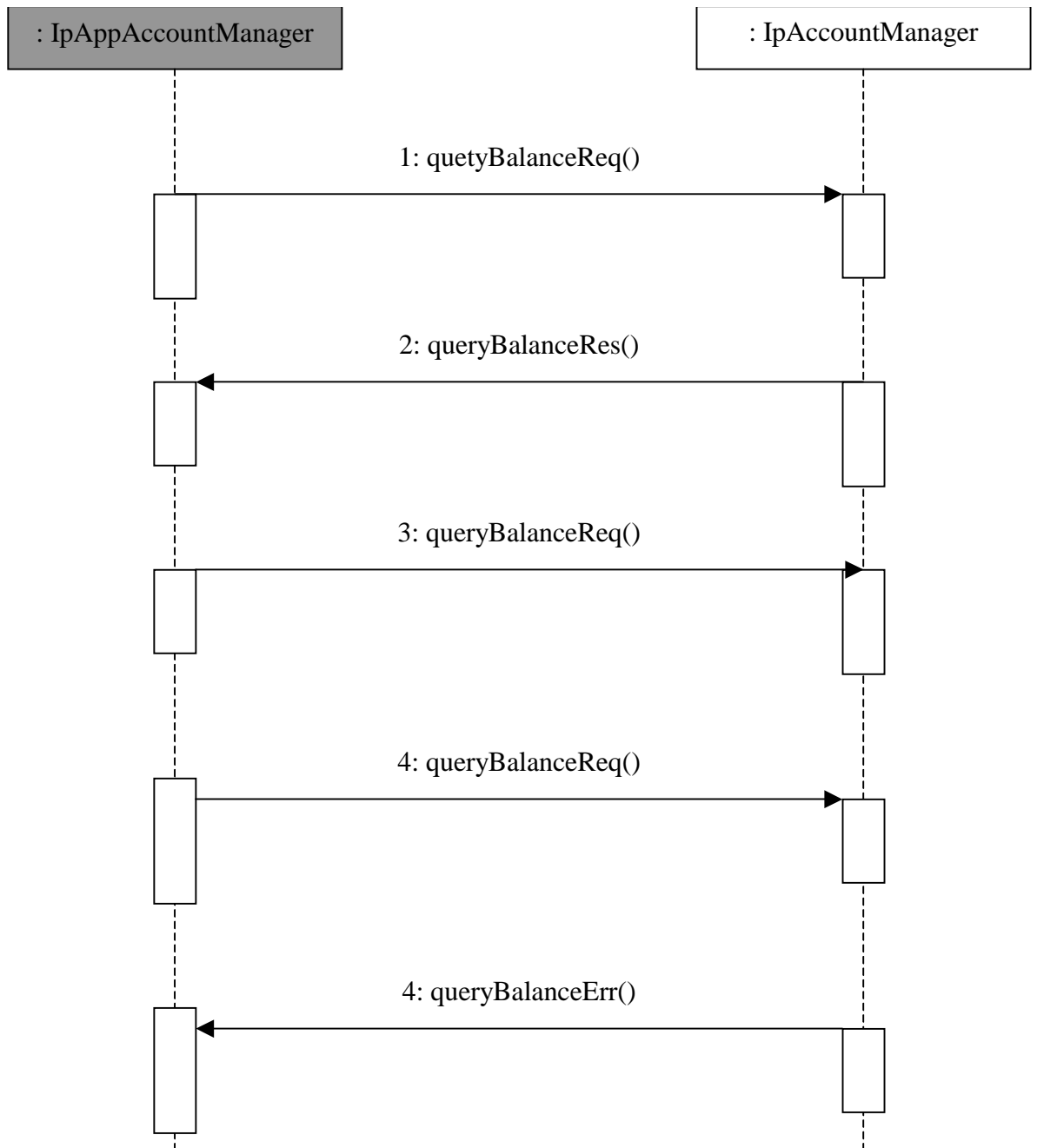


Рис. 9. Обмен сообщениями при использовании услуги напоминания

Сообщение 4 маршрутизирует вызов к пользователю, заказавшему услугу. Сообщение 5 – информация об ответе на вызов. Сообщение 9 вызывает проигрывание пользователю напоминания, а 10 – подтверждение о выполнении. Названия методов дают представления об их назначении, полное же их описание со всеми параметрами приводится в спецификациях Parlay API.

Ещё один пример, демонстрирующий работу Parlay API и взятый непосредственно из спецификаций, относится к Account management SCF. Диаграмма представлена на рис. 10.



*Рис. 10. Пример обмена сообщениями при использовании приложением Account management SCF*

Пример упрощен, в нем не приводятся некоторые служебные сообщения, необходимые для функционирования алгоритма. Итак, на этой диаграмме сообщений 1 используется приложением, чтобы запросить информацию о балансе счета одного или нескольких пользователей, а сообщение 2 передает запрошенные данные. Сообщения 3, 4 и 5 демонстрируют случаи неуспешных запросов, вследствие неправильных параметров сообщения (метода) и, соответственно, ошибки в сети (например, отсутствие данного абонента).

На рис. 11 можно видеть пример обмена сообщениями для Charging SCF.

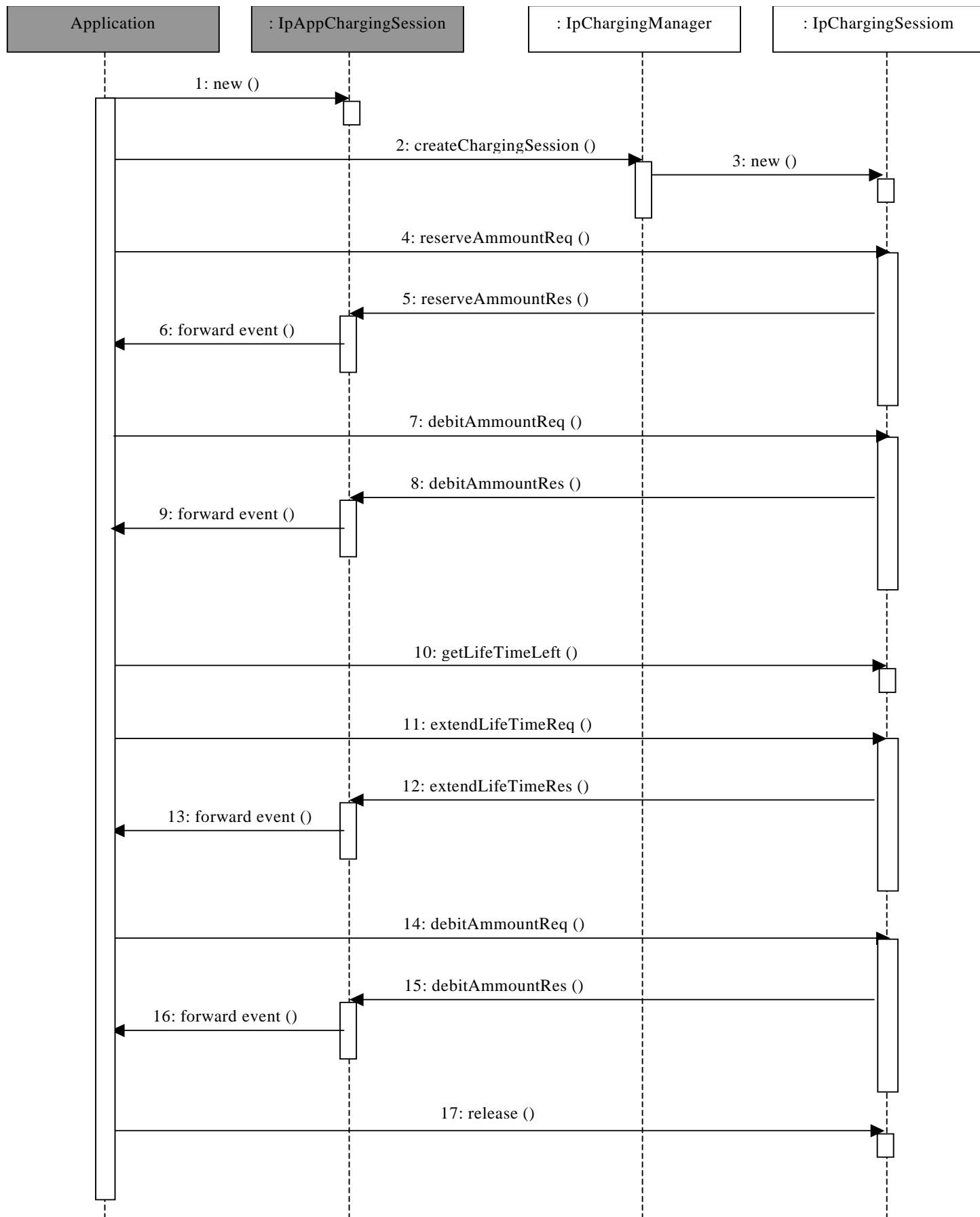


Рис. 11. Пример обмена сообщениями при использовании приложением Charging SCF с применением резервирования

Исходные предположения для этого примера следующие. Приложению необходимо предоставить пользователю поток видеоинформации в течении 10 минут, общей стоимостью \$2.00. Сначала требуется узнать, есть ли у пользователя на счету необходимые средства – это можно выполнить через Account management SCF, и не представлено на диаграмме 11. Далее приложение «резервирует» у сети необходимую сумму со счета пользователя. После предоставления услуги сумма списывается со счета.

Рассмотрим рис. 11 подробнее. Сообщения 1, 2, 3 нужны для обеспечения функционирования алгоритма. Сообщение 4 запрашивает у сети резервирование определенной суммы. В нашем случае \$2.00. Предположим, что все необходимые для этого требования соблюдаются, тогда сообщение 5 подтвердит резервирование данной суммы. После предоставления части (пусть половины) услуги, приложение может взять за это плату. Сообщение 7 списывает \$1 с зарезервированной суммы. Сообщение 8 является подтверждением. Сообщением 10 приложение проверяет, будет ли оставшееся время, выделенное для резервирования больше чем время, необходимое для предоставления оставшейся части услуги. Пусть оставшегося времени резервирования недостаточно. Тогда сообщение 11 будет запросом на продление времени резервирования. Если это допустимо, то сообщение 12 – подтверждение. Когда услуга будет предоставлена полностью, приложение снимет оставшийся \$1, это выполняется сообщением 14. Сообщение 15 – подтверждение. 17 – служебное сообщение. Сообщения 6, 9, 13, 16 – для коммуникаций внутри приложения.

Для расчетов с пользователем приложение может применять более простой алгоритм, не применяя резервирование. Как это происходит, представлено на рис. 12.

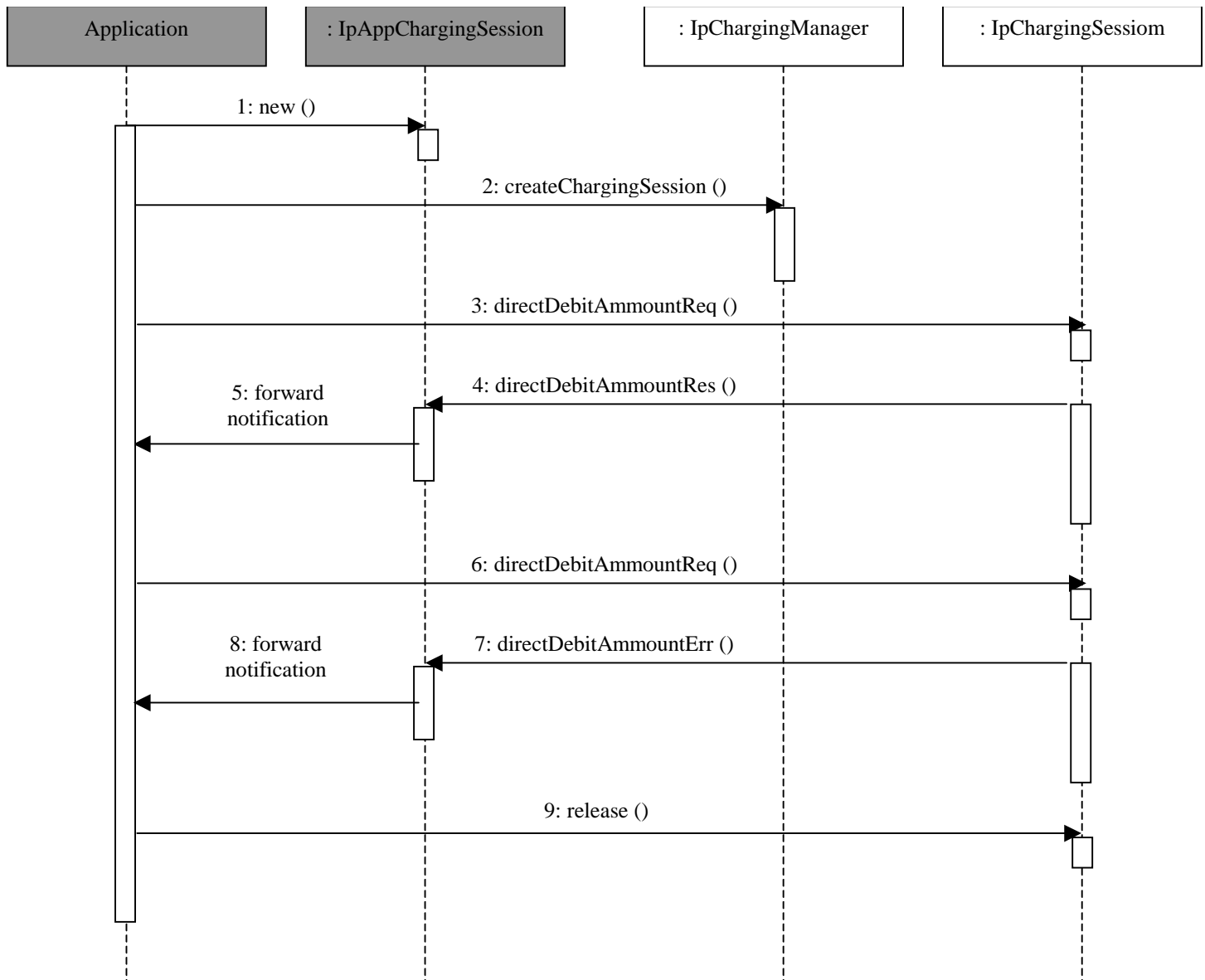


Рис. 12. Пример обмена сообщениями при использовании приложением Charging SCF без применения резервирования

Предположим, что приложение действует на WAP-шлюзе и за каждый запрошенный адрес (URL) с пользователя взимается \$0.01. Так как, в этом случае, потеря подобной суммы (\$0.01) не страшна, то можно не делать предварительного резервирования.

Приложение посылает запрос на списание \$0.01 со счета пользователя за каждый URL, а когда сеть возвращает ошибку, указывающую об окончании средств на счету, пользователь отключается.

Сообщения 1, 2, 9 и 5, 8 – служебные. Сообщением 3 приложение запрашивает списать со счета пользователя определенную сумму (\$0.01), сообщение 4 – подтверждение платежа. Далее эта последовательность сообщений типа 3 и 4 может повторяться неопределенное время. Сообщение 6 – то же, что и сообщение 3, однако в ответ на него приходит отказ, после чего предоставление услуги прекратится.

Наконец рассмотрим, вероятно, самый простой, пример использования приложением Call Control SCF. Диаграмма обмена сообщениями (вызова методов) представлена на рис. 13.

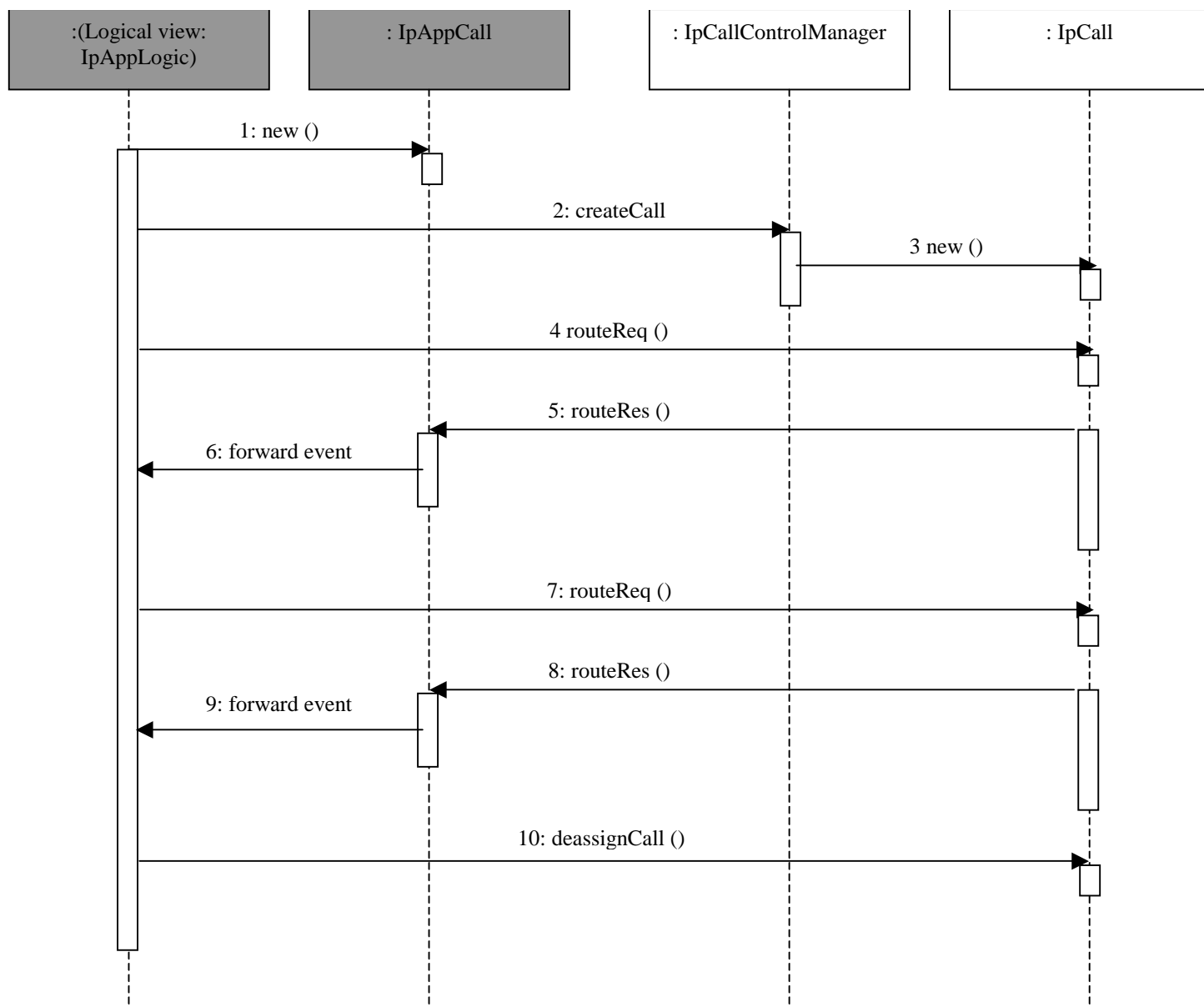


Рис. 13. Диаграмма обмена сообщениями при использовании приложением Call Control SCF для установления соединения между двумя пользователями

В этом примере приложение создает соединение между абонентами «А» и «Б». Предполагается, что один из них воспользовался, например, Web-интерфейсом для того, чтобы выбрать имя того, с кем хочет установить соединение и, тем самым, активировал работу приложения.

Остановим внимание на сообщениях, необходимых для общего понимания процесса. Сообщение 4 указывает сети, что созданный вызов необходимо маршрутизировать к

пользователю «А». Также, приложение требует подтверждения от сети, когда пользователь «А» ответит. Сообщение 5 указывает на то, что пользователь ответил. Сообщение 7 указывает сети, что созданный вызов необходимо маршрутизировать к пользователю «Б». В этом случае приложение тоже требует подтверждения от сети, когда пользователь «Б» ответит или произойдет неуспешный вызов. Сообщение 8 указывает на то, что «Б» ответил. Теперь у вызова две стороны и между ними автоматически устанавливается речевое соединение. Сообщением 10 приложение полностью прекращает контроль над вызовом, он продолжается в сети.

Мы рассмотрели простейшие элементы реализации услуг. Примеры реализации множества других элементов содержатся в спецификациях Parlay/OSA.

Консорциум Parlay Group включает в себя около 60 компаний, занимающихся разработками в области открытых интерфейсов. Среди них такие известные, как Alcatel, AT&T, BT, Cisco Systems, Ericsson, Fujitsu, Hewlett-Packard, IBM, Lucent, Nokia, Siemens, Sun Microsystems и др.

На момент подготовки пособия было известно, что уже тремя компаниями производятся сервера приложений, а одиннадцатью SCS (Parlay/OSA Gateways). Кроме того, на тот момент, уже было разработано более 20 различных комплексных приложений.

## **5 Примеры услуг реализованных с применением Parlay API**

В качестве ознакомления приведем несколько примеров существующих услуг реализованных с применением Parlay API.

Компания Net4Call.

### **1. Услуга “800”**

Услуга позволяет операторам направлять вызовы с префиксом 800 к определенным организациям или людям. При этом оплата за вызов снимается с вызываемой стороны. Вызовы могут генерироваться из фиксированной, мобильной или IP-сети. Net4Call предлагает использование удобного пользовательского интерфейса на WWW, через который заказчики услуги 800 могут посмотреть свои настройки, управлять подсказками и т.п. Маршрутизация вызова может происходить в зависимости от префикса номера вызывающего абонента, времени суток и, например, положения на местности. Могут применяться фильтры по номерам вызывающих абонентов, IP-адресам, доменным именам

и местоположения вызывающего. Возможно последующее применение Call Back при невозможности ответа вызываемого номера.

## 2. Phone (Click-to-dial), Conference

Услуга позволяет своим пользователям генерировать вызовы, пользуясь Web интерфейсом. При этом могут использоваться адресные книги, хранящиеся на ПК, службы каталогов с доступом по LDAP. Вызовы могут устанавливаться между различными сетями. Возможна реализация конференций. Также могут быть использованы автоматический дозвон, проигрывание звуковых файлов пользователю на удержании и т.п.

## 3. FindMe

Услуга предоставляет пользователю возможность практически всегда быть достигаемым средствами связи. Может интегрироваться с услугой 800 или click-to-dial. В зависимости от вызывающего абонента, вызов может быть игнорирован, маршрутизироваться на голосовую почту или какой-либо номер, в зависимости от времени суток и т.п. Пользователи могут сам настроить эту услугу через Web.

## 4. Info-Now

Услуга позволяет операторам предоставлять пользователям ту информацию, на которую они подписались, и в то время, на которое настроили эту услугу через Web.

Компания AePONA.

## 1. Close Friends

Услуга настраивается через Web или WAP. При её активизации происходит проверка, как близко с пользователем услуги находятся терминалы его “друзей”. Находящимся неподалеку рассылаются короткие сообщения. Возможна рассылка разных сообщений разным пользователям и разным группам пользователей. Это настраивается. Услуга может использоваться в составе других, более обширных услуг. Другой вариант – пользователю по запросу может быть выслан список находящихся рядом банкоматов и т.п.

## 2. E-Mail Alert

Также возможна настройка через Web. Пользователю посылается, выбираемым им способом, информация о том, что в его почтовый ящик поступило письмо (например, по SMS). Владельцем почтового сервера, естественно, может быть не только оператор сотовой связи (для примера с SMS), а любая организация.

### 3. Advanced Call Manager

Позволяет пользователю контролировать все входящие звонки. В зависимости от таких параметров как время суток, номер вызывающего абонента и т.п. вызов может быть отклонен, переадресован на голосовую почту и т.п. Похожа на FindMe Net4Call.

### 4. My Radio

Настраивается пользователем, например, через Web. Заключается в том, что когда по определенному радио играет какая-либо мелодия, пользователь может позвонить на определенный номер (или, вероятно, послать SMS) и узнать подробную информацию о данной мелодии. Другой вариант – когда на радио играет определенная мелодия, пользователю посылается сигнал (SMS, вызов), естественно, что пользователь должен сначала настроить услугу на определенную мелодию. Следующая возможность – пользователю посылается сообщение не о мелодии, а о специфичной для данной области нахождения пользователя информации – погоде, новостям. Сигнал может подаваться пользователю не во время, а немного до того как начнет передаваться информация или мелодия.

Компания Arrium.

#### 1. Fuzion-UM

Услуга интегрированного окружения сообщениями. Заключается в том, что предоставляется единое хранилище для информации (почтовый ящик), доступ к которому может быть осуществлен через e-mail, факс, web или голосовую почту (телефон). Есть уведомления о приходе сообщений, фильтрация сообщений, мультязычность. Применяются решения email to voice, voice to email. Услуга может комбинироваться с Fuzion-VA.

#### 2. Fuzion-VA

Приложение включает такие услуги, как Personal Number, Personal Assistant, Intelligent Prompting, Advanced Call Screening, Find-Me и Call-Blasting. Personal Assistant

имитирует секретаря и решает что делать с входящими вызовами, настраивается пользователем. One Number & Find Me – единый пользовательский номер, по которому его почти всегда можно найти – в реальности список номеров работы, дома, мобильных терминалов с приоритетами генерации вызовов. Intelligent Prompting – в случае недоступности пользователя, вызывающему может проигрываться конкретная подсказка - 'Jane is out of the office. I am trying to connect to her home number' или 'John is in a meeting until 15:00' и т.п.

### 3. Fuzion-UC

Интегрированная услуга на базе двух предыдущих.

Все возможные услуги обычно комплектуются системами помощи и подсказок.

Остается добавить, что развитие Parlay API непрерывно продолжается и скоро должна появиться четвертая версия спецификаций открытых интерфейсов, совместимая с предыдущими. Предпринимаются попытки сделать создание приложений доступным для как можно большего числа разработчиков, хотя бы и ценой уменьшения возможностей (т.н. Parlay X). Все большее число компаний обращает внимание на концепцию OSA и включается в работу с ней.

По прогнозам аналитиков наибольший пик активности по внедрению систем, поддерживающих открытые интерфейсы Parlay операторами связи должен придтись на 2003-2005 гг. После чего концепция, идее которой посвящена это пособие, перестанет быть интересной новинкой, а станет обычным, может быть даже обязательным элементом механизма предоставления разнообразных инфокоммуникационных услуг и конвергенции разнородных сетей.

### **Контрольные вопросы**

1. Каковы причины возникновения концепции OSA?
2. Приведите примеры технологий реализующих идеи, заложенные в концепцию OSA.
3. Перечислите назначение основных интерфейсов Parlay API.
4. Перечислите основные элементы входящие в состав сети построенной с применением технологии Parlay API. Изобразите их на схеме.
5. Перечислите все интерфейсы Parlay API третьей версии.
6. Каково назначение и функции интерфейса Framework?
7. Каково назначение и функции интерфейса Call Control?
8. Каково назначение и функции интерфейса Mobility?

9. Приведите пример цикла взаимодействия Framework, SCF и приложения.
10. Приведите пример реализации элемента услуги. Изобразите диаграмму, поясните назначение основных методов.
11. Какие услуги реализованные с применением технологии Parlay API или JAIN вы знаете? Попробуйте сконструировать новую услугу.

Ссылки:

1. Гольдштейн Б.С., Фрейнкман В.А. Call-центры и Компьютерная телефония. – СПб.: БХВ – Санкт-Петербург, 2002.
2. Гольдштейн Б.С., Ехриель И.М., Рерле Р.Д. Интеллектуальные сети. – М.: Радио и связь, 2000.
3. Гольдштейн Б.С., Пинчук А.В., Суховицкий А.Л. IP-Телефония. – М.: Радио и связь, 2001.
4. Ard-Jan Moerdijk, Lucas Klostermann. Opening the networks with Parlay/OSA APIs: standards and aspects behind the APIs.
5. Zygmunt Lozinski. Parlay Member Meeting, Hong-Kong, WG4. Developer Realization, 06 February 2002.
6. Richard Stretch. The OSA API and other related issues, BT Technol J, Vol.19, №1, January 2001.
7. Parlay/OSA White Paper - Ericsson Mobility World.
8. Richard Stretch, 3rd Party Service Control.
9. Marc LeClerc. Building Parlay/OSA based Applications for Telecomm Networks, 2002.
10. Zygmunt Lozinski. Parlay Report, 09 July 2002.
11. Спецификации Parlay API.
12. Сайты упомянутых компаний.