• Week #9 (March 1 - 5)
• Quiz
• Week #10 (March 8-12)
• Parlay/OSA and CPL (Other examples of signaling protocol neutral service engineering technologies)
• Week #11 (March 15 - 19)
• QoS
• Week #12 (March 22 – 26)
• Web 2.0
• Week #13 (March 29 – April 2)
• UMTS and beyond
• Week #14 (April 5 - 9)
• Quiz
• Week #15 (April 12 – 16)
• Project reports + demos
Chapter IX

Parlay/OSA and CPL as examples of signaling protocol neutral approaches
Introduction

Signaling protocol neutral service engineering technology
• Service architecture applicable to NGNs using any signalling protocol
  – Next Generation signalling protocols
    • SIP
    • H.323
• Example already studied in this course: Web services
• Examples presented today:
  – Parlay/OSA
  – Call Processing Language (CPL)
• Example to study later in the course:
  – Web 2.0
Signaling protocol neutral architectures ...

1. Parlay

2. CPL
1. Introduction
2. Business model
3. Interactions
4. APIs
5. Case Study
6. Pros and cons
PARLAY forum
- Created in 1998 as close forum
- Open since 2000
- Include most major players from telecommunications and computer industries (e.g. Ericsson, Lucent, Siemens, IBM)
- Work initially done in collaboration with third generation partnership project (3GPP)
- API called Parlay / Open Service Access (OSA)
- Parlay forum now dismantled and work fully done in 3GPP
Introduction

PARLAY main goal: Open up telecommunication networks

• Enable new business models
• Use open information technology middleware
• Make telecommunication network capabilities available for application development
  – Two types of APIs
  – Services APIs
    • Expose the network capabilities (e.g. call control, presence)
  – Framework APIs
    • Make the use of the service APIs secure, accountable and resilient (e.g. security, registration, authentication)
The business model

• Introduction
  – TINA-C inspired business model
  – Terminology: Services mean network capabilities

• Roles
  – Client application
    • Consume/use the services (e.g. network capabilities)
    • Equivalent to end users in TINA-C.
  – Enterprise operator
    • The entity that subscribes to the services
    • Subscriber in TINA-C
  – Framework operator
    • Entity that handles the subscriptions
    • Equivalent to the retailer in TINA-C
General model

(1) Service Registration

Enterprise Operator (Subscriber)

(2) Service Subscription

Framework Operator (Retailer)

(3) Service Usage

Client Application (End User)
Commonly deployed model ...

- Enterprise operator
- Client application
- Framework
- services/APIs
- Third party domain
- Network operator domain
Interactions

Application and framework

Authentication
• Peer to peer model
• Allow framework to check that application is “who” it claims to be and application to check that framework is “who” it claims to be
• Usually used in only one direction (I.e. framework checking).

Authorisation
• Determination of what the application can do once authenticated

Discovery
• Once authenticated applications can get info on available APIs

Establishment of service level agreement
. Usually done off-line
Interactions

Services/ APIs and framework Registration / De-registration

• Allow services to register/de-register to/from the framework
Interactions (Taken from reference [2])

1: authentication
2: request Registration interface
3: register factory
4: discovery Service
5: request Discovery interface
6: discover Service
7: Select Service + sign SLA
8: create Service Manager
9: return Service Manager
10: return Service Manager
11: Use service

1 – 3 registration/discovery, 4-11 run time communications establishment ..
The APIs

- Framework APIs
- Service APIs
- Client Applications
- Parlay/OSA APIs

Resources interfaces

Figure 2 Parlay APIs interfaces
The APIs

Some common characteristics

Specifications include
- High level specification in UML (Universal Modelling Language)
- API specifications for several IT technologies
  - CORBA IDL
  - WSDL
  - Java

Two modes of communications
- Synchronous
- Asynchronous
Framework API: Make the use of the service APIs secure and resilient

- Trust and security management
- Event notification
- Service discovery
- Integrity management (e.g. load management)
- Service agreement
Framework API: Make the use of the service APIs secure and resilient

Trust and security management – Examples of methods:
- AbortAuthentication()
- AuthenticationSucceeded()
- Challenge()
- TerminateAccess()
- InitiateAuthenticationWithVersion()
Service API: Give access to network capabilities

- Call control
- User interactions
- Generic messaging
- Mobility
- Terminal capabilities
- Connectivity management
- Account management
- Charging service
- Data session control
- Presence and availability management
An example of Service API: Call control

- Generic call control service
- Multiparty call control service
- Multimedia call control service
- Conference call control service
The call control API

Call model

- Terminal
  - End point (Not covered in the current specifications)

- Address
  - Represents a party in a call (E.164 number, IP address)

- Call
  - Abstraction of the physical call that occurs in a network

- Call leg
  - Logical association between a call and a party involved in a call
The call control API

Generic call control
- Two party voice call only
- Remain in Parlay for historical reasons

Multiparty call control
- Establishment of calls with any given number of users
- Root of the inheritance tree

Multimedia call control
- Add multimedia (e.g. media negotiation) capabilities

Conference call control
- Add conferencing capabilities
Conferencing / multiparty sessions

Basis of a wide range of applications
- Voice/videoconferencing
- Multiparty gaming
- Distance learning
- And more …

Categorization schemes
- With / without sub-conferences
- Pre-arranged vs. ad hoc
- With / without floor control
  - Floor control: Who can be heard / seen
- Where the media is mixed (e.g Centralized vs. decentralized)
- Dial-in (Meet-me) vs. dial-out
Conferencing with Parlay ….

Examples of methods …
- CreateConference ()
  - Parameters include the number of sub-conferences
- CheckResource ()
- ReserveResources ()
- FreeResources ()
- PartyJoined ()
- SplitSubconference ()
- MergeSubconference ()
- FloorRequest ()
A case study on PARLAY/OSA and SIP: Run For Your Life game (Described in detail in reference [3])

• 1 - Introduction
• 2 - Game
• 3 - Architecture
• 4 - Mapping
Introduction ...

Run-For-Your-Life

• Built from scratch in Ericsson Research lab in Montreal Canada
• Demonstrated at several trade shows (e.g. ICIN 2001, Parlay Munich meeting, Parlay Hong Kong meeting)
• Objectives assigned to the game design
  – Extensive usage of call control capabilities
  – Have fun …
Introduction ...

Objective of the case study ...
Aim at helping in tackling two issues:

1. PARLAY Call Control APIs that cannot be mapped onto SIP
   - What are they?
   - What is the impact on service creation?

1. SIP semantics that are not visible in PARLAY APIs as per today’s specification
   - What are they?
   - What is the impact on service creation?
A multiparty cooperative game

- Group of people trapped in a house with several rooms set to burn/explode in a given time
- Can escape only if password is found
- Letters making the password scattered in selected rooms of the house
- People ending up in the same room can exchange hints about the password via audio and chat

Game can be assimilated to a conference with as sub-conference people ending up in a same room

Requiring a set of well defined conferencing functionality

- Conferencing
- Sub-conferencing
The game ...

Roch H. Glitho
Architecture ...

GAME SERVER

PARLAY / SIP GATEWAY

- Signaling Control Unit
- Media Control Unit

TCP/IP

PARLAY

GAME client

SIP

RTP

TCP/IP

GAME client

Roch H. Glitho
Architecture ...

Signaling Control Unit

- Parlay handler
- PARLAT/SIP Glue
- SIP Handler

MEGACO/H248

Media Control Unit

- Media Manager
- Media Handler
- Media Handler
- Media Handler
- RTP Handler

Roch H. Glitho
Figure 5 - Mapping for dial-in
The mapping ...

PARLAY Call Control Services that cannot be mapped onto SIP
- There seems to be none
- However the mapping can be done in several ways in some cases

SIP semantics that are not visible in PARLAY APIs as per today’s specification
- There exist a few (e.g. Possibility of a caller to state for instance that the call should not forwarded)
- PARLAY may be extended to cater to these features
Pros and cons ...

**Pros**
- PARLAY/OSA allows the creation of a wide range of services including services that combine different types of network capabilities (e.g. call control, mobility, presence)
- Parlay allow the creation of services that span several network technologies (e.g. Sip, H.323)

**Cons**
- The level of abstraction is still low
  - 3N+1 calls were required to create a conference call in older versions of Parlay – The number is now N+1
- Parlay is not easy to grasp by people with no circuit switched telephony/IN background
  - Call leg concept
The Call Processing Language

1. Introduction
2. Requirements
3. Constructs
4. Example
5. Pros and cons
Introduction ...

Specificities:
- Only architecture that aims at service creation by end-users
- Prime target: Un-trusted parties
  - Direct use
  - Use via a graphical user interface
    - Higher level of abstraction
    - Mapping done by middle ware
Targeting end-users has a few consequences:
  – Stringent language requirements
  – Need to upload scripts to servers
    • REGISTER has been proposed for SIP
    • No mechanism has been proposed for H.323
Requirements on language (From the RFC).

- Lightweight, efficient easy to implement
- Easily verifiable for correctness
- Executable in a safe manner
- Easily writeable and parsable
- Extensible
- Signaling protocol independence
Constructs for an XML Based CPL ...

Switches
– Choices the script can make
  • Address, string, time, priority

Signaling operation
– Cause signalling events in underlying protocol
  • Proxy, redirect, reject

Location modifier
– Add/remove location
Telecommunication Services Engineering Lab

Simplified example from the RFC ...

Location: sip:john@example.com

Proxy

Time switch:
Mon. – Fri
From 9am to 5pm

Sat-Sun
From 9am to 10pm

01.01.03-15.01.03

Otherwise

Location: secretary@office.com

Location: private@home.com

Location: voicemail@office.com

Redirect

Roch H. Glitho
Pros and Cons ...

Highly suitable for service creation by end-users

- End-users familiar with scripts / XML
- End-users unfamiliar with scripts / XML (via GUI)
- Offer required security

However:

- Very few end-users are interested in creating service
- CPL is highly unsuitable for service creation by providers / third parties
  - Range of services that can be created is limited
  - More powerful tools exist
- Service logic and service data need to reside in the same script
To probe further ...

PARLAY:


CPL RFCs