Web Services and SOA for Communication – Part II

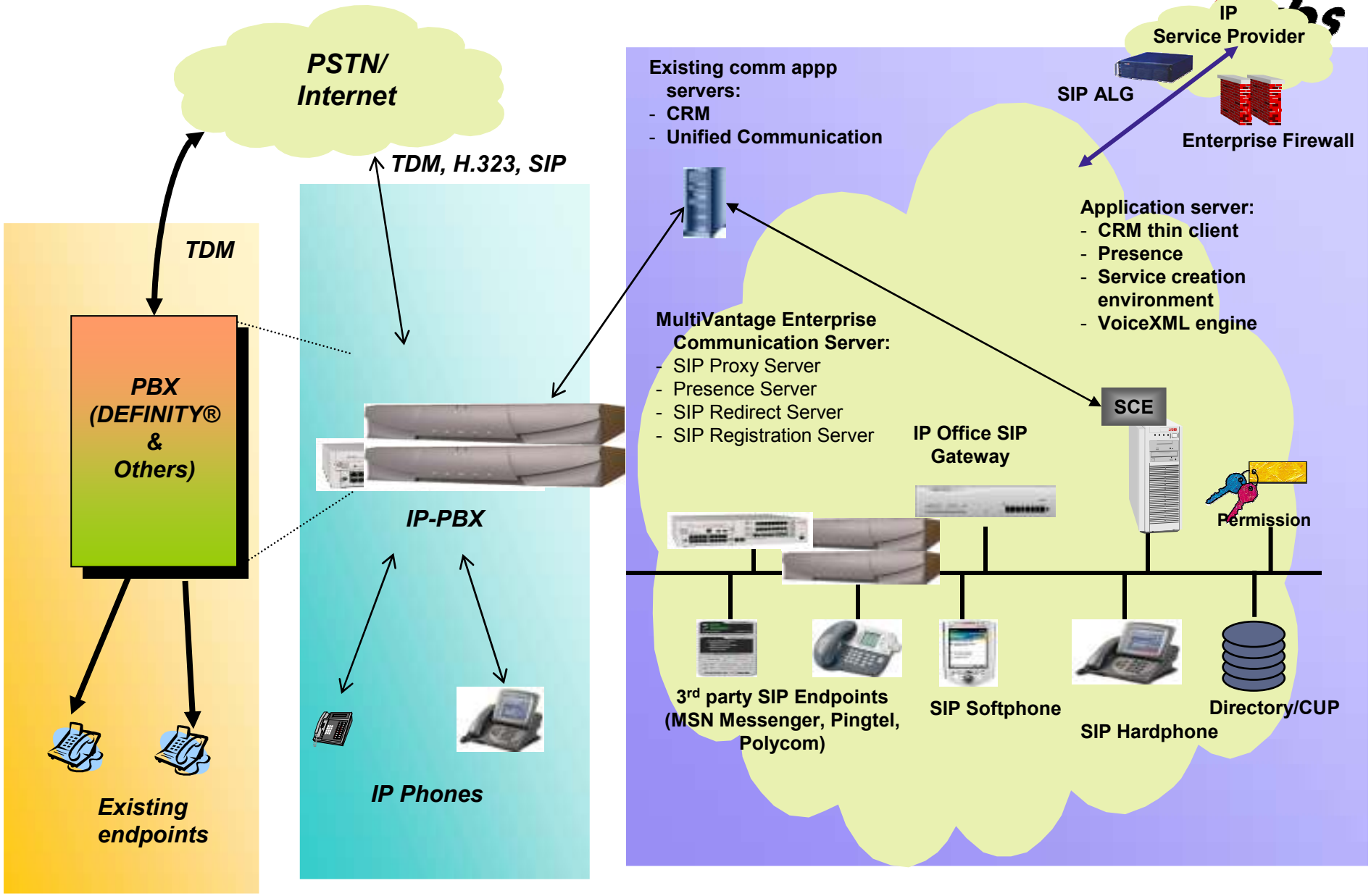
Wu Chou
Director, Avaya Labs Fellow
wuchou@avaya.com
06/15/07

IP Telephony | Contact Centers | Mobility | Services

PART 2

Web Services and SOA for Communication Enablement

Multimedia/Voice Communication over IP



Communication: Things to Improve/Wish

- Hard to integrate with business or personal applications (especially for non-telecom experts).
- A tightly coupled nutshell wrapped by many layers of rigid, low level and non-reusable protocols, since telephone was invented.
- In the era of Internet, communication should be enabled as service
- Need paradigm shift from object-oriented to service-oriented to support CEBP (Communication Enabled Business Process)
- Communication services should be stackable, e.g. add services on demand without re-working the whole communication protocol.

In the era of Internet, communication should not be lagging behind in non service-oriented paradigms.

Some Challenges in Web Service Enablement of Communication

–Services in communication are typically stateful services

–Require the establishment of explicit "*session*" association in order to maintain the service context and to transmit and exchange events

–Endpoints in communication can be both client and peer-to-peer server

–Asynchronous outbound operations and event notification

–Event sink interface definition/design/verification in service-oriented communication so that it can support two-way web service interaction

–…

Road to WIP

(Web Service Initiation Protocol)

An end-to-end web service and SOA based paradigm for communication over IP

Related Work

- SIP (Session Initiation Protocol): A set of operation protocols which are neither XML nor web service.
- XMPP/Jingle: It is based on a streaming XML extension of XMPP (eXtensible Messagine Presence Protocol) for real time peer-to-peer communication (e.g. GoogleTalk, Jabber, etc.). It is not web service based nor service oriented.
- Skype/QQ: Private protocols/methods for signaling and messaging
- WSIP: It is based on a two-stack approach (SIP+Web Service) that relies on SIP for low level signaling and web service for integration (not a full web service based communication protocol)

WS Based Session Management

- **Issues**
  - It must be transport protocol neutral and cannot relay on the lower level transport protocol to manage the session
  - The session association cannot rely on implicit off-line agreement
  - Must handle stateful session based services
- **Solution**
  - Introduce a generic Web service specification, WS-Session, for application service management
  - Treat application session as an independent service that can either work by its own or be integrated, composed, and stacked with other services
  - Separate the application session service management from transport protocol
  - Allow a client to establish explicit association and to create a session on the server through Web service

WS-Session (ECMA-366, ISO/IEC 25437)

```
<portType name="ApplicationSessionServicesPortType">
 <operation name="tns:StartApplicationSession">
 <input message="tns:startApplicationSession"/>
<output message="tns:startApplicationSessionPosResponse"/>
<fault name="StartFault" message="tns:startApplicationSessionNegResponse"/>
</operation>
<operation name="tns:StopApplicationSession">
<input message="tns:stopApplicationSession"/>
<output message="tns:stopApplicationSessionPosResponse"/>
<fault name="StopFault" message="tns:stopApplicationSessionNegResponse"/>
</operation>
<operation name="tns:ResetApplicationSessionTimer">
<input message="tns:resetApplicationSessionTimer"/>
<output message="tns:resetApplicationSessionTimerPosResponse"/>
<fault name="ResetFault"
message="tns:resetApplicationSessionTimerNegResponse"/>
</operation>
<operation name="tns:ApplicationSessionTerminated">
<output message="tns:applicationSessionTerminated"/>
</operation>
</portType>
```

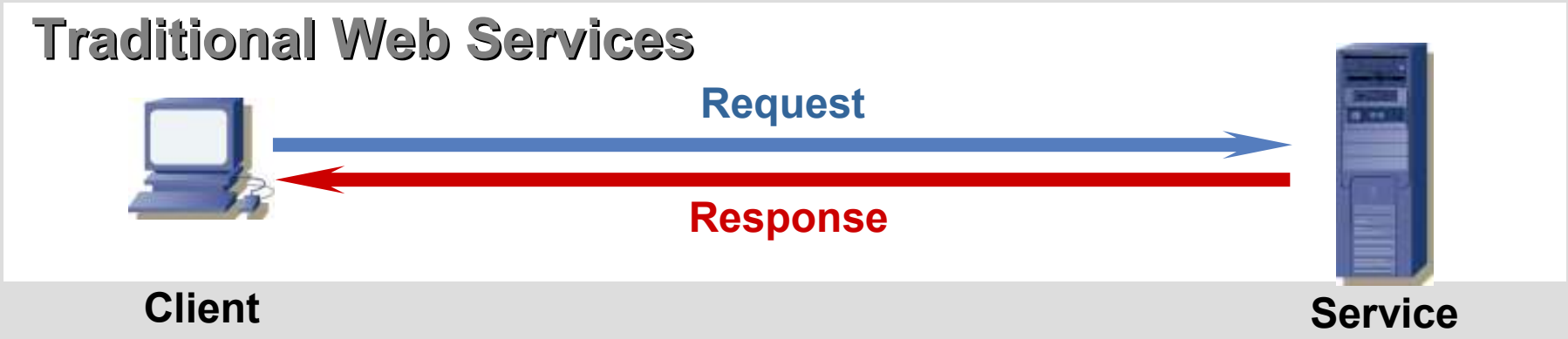
Web Service Session Management

- **WS-Session**
  - Defines specific session management WS operations
  - Defines SOAP binding and a successful session establishment will result a unique sessionID in the SOAP header.

```
<s11:Envelope><s11:Header>…</s11:Header>
<s11:Body>
<StartApplicationSessionPosResponse xmlns="http://www.ecma-
international.org/standards/ecma-354/appl_session" >
        <sessionID>${session_id}</sessionID>
        <actualProtocolVersion>http://www.ecma-
international.org/standards/ecma-
323/csta/ed3</actualProtocolVersion>
        <actualSessionDuration>-1</actualSessionDuration>
</StartApplicationSessionPosResponse>
</s11:Body>
</s11:Envelope>
```

Two-Way Web Services: Communication

applications motivate two-way web services



1. *Each endpoint needs to have both client and server interface (not a client-server interaction anymore).*
2. In order to support asynchronous messages proactively pushed to the client from the server, e.g. events, the client interface has to bear certain relationships with the corresponding server interface
3. General approach to interface design and verification based on operation reversal and XML type generalization ("Two-web services from Interface Design to Interface Verification", Li Li and Wu Chou, Proc. of ICWS'05)

Two-way WS for Communication Enablement

- **Two-way Web service Interaction**
  - Endpoints in communication are typically both client and peer-to-peer server and communicate in two-way, full duplex fashion
  - Outbound (asynchronous) operations and event notifications are critical to Web service based communication enablement
- **Interface design is key to sustain the two-way Web service interaction (extension from one-way request/response)**
  - The client interface must match the outbound operations of the server in a proper way to allow two-way web service interaction
  - The client needs to expose a proper event sink interface to receive events pushed from the server
  - What properties, relations and operations are required to enable two-way web service interaction between two web service endpoints?

Two-way Web Service Interface Design

- **Tightly Coupled (TC) Interface**
  - The client event sink contains a mirrored reversal of the server outbound operations
  - Strong type checking can be performed by both server and client
  - Tightly coupled and any change of the server operation will lead to the corresponding changes of all clients
- **Loosely coupled (LC) interface**
  - "Wrapped" message delivery model, and only needs to expose a small amount of operations to receive a variety of messages
  - Client and server are loosely coupled and each can evolve independently as long as thy maintain the loosely coupled relation
  - More suited for distributed processing
  - Strong type checking is not possible by the client

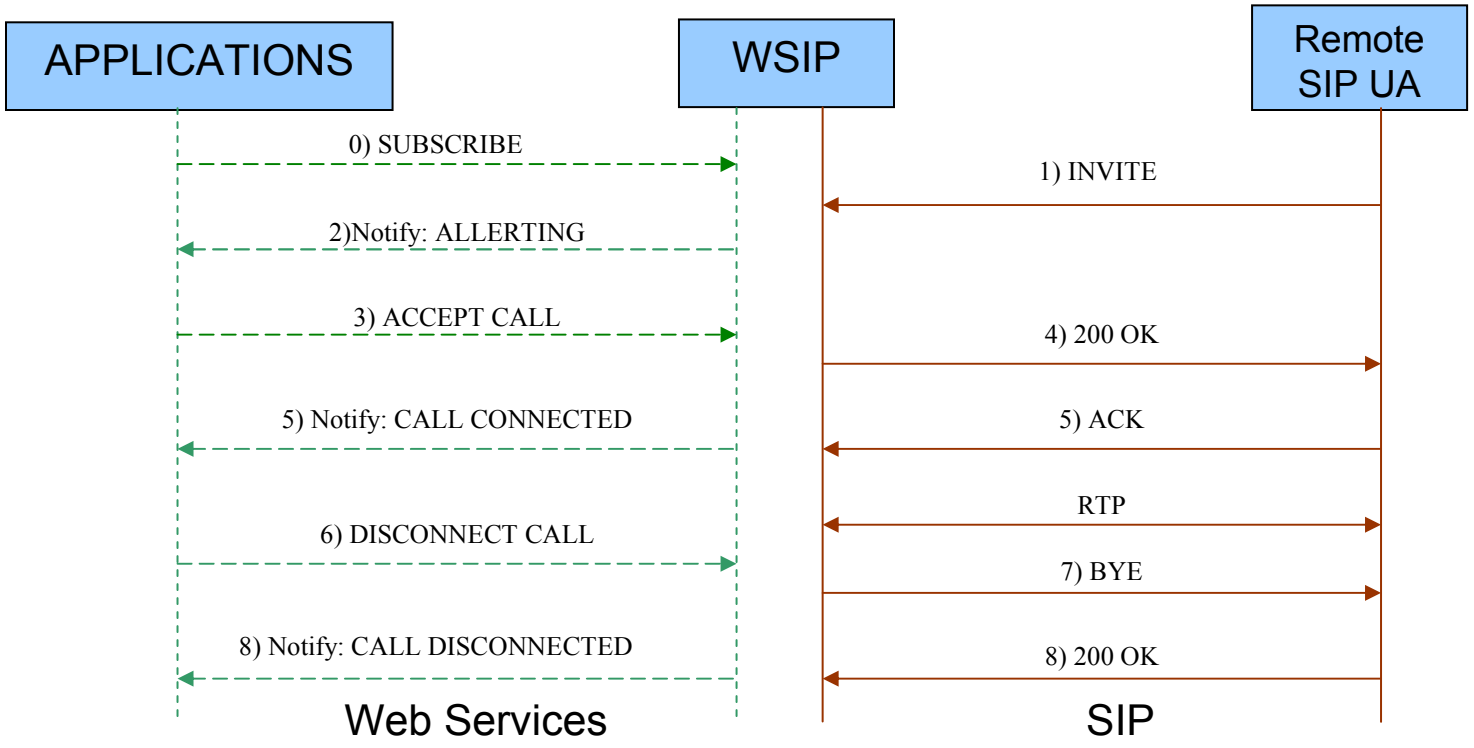
Combined interface: TC+LC design

References:

Li Li and Wu Chou, "Two-Way Web Service: from interface design to interface verification", *Proc. ICWS'05, vol. 2, pp. 525-532, July 2005*

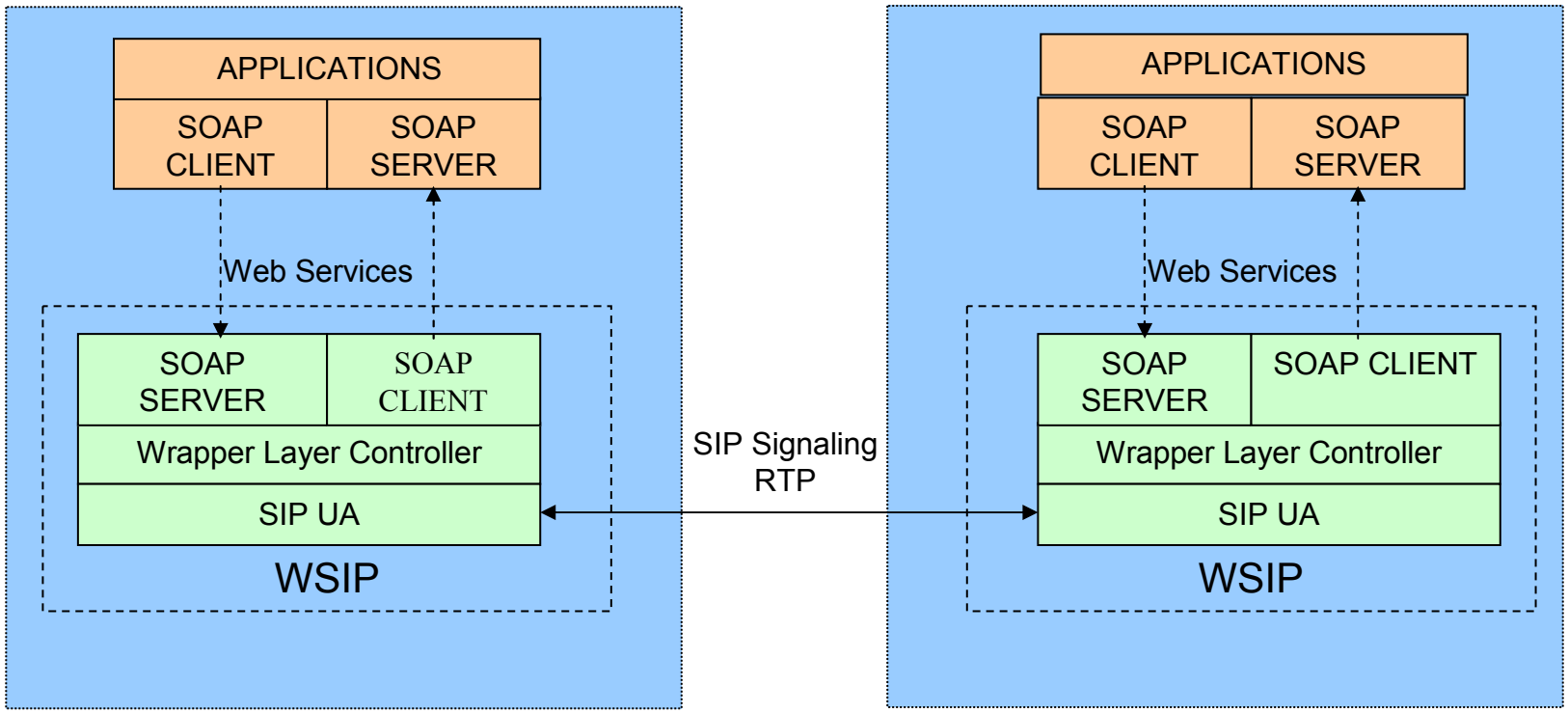
ECMA TR-90, ""Session Management, Event Notification, and Computing Function Services – an amendment to ECMA-348", Dec. 2005, ECMA International.

WSIP – A new converged communication paradigm over IP



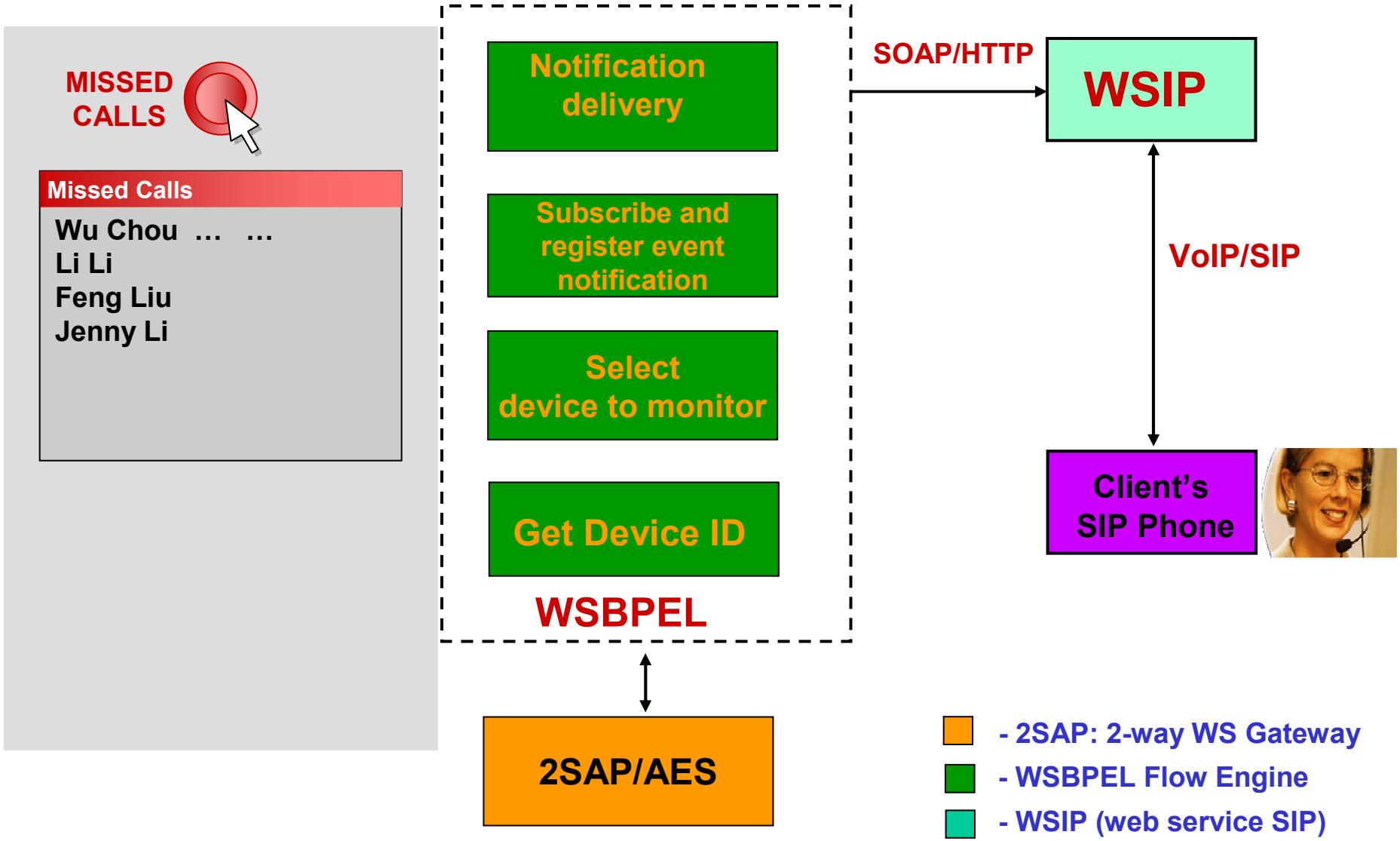
An example of call flow

WSIP – A new converged communication paradigm over IP



The architecture of WSIP

Example: Missed-calls application using an open standard engine WSBPEL (Web Service for Business Process Execution Language)



Things to Improve from WSIP

- **WSIP** – Web service enablement of SIP endpoint to integrate SIP based VoIP communication business transaction

- WSIP is a dual-stack (Web Services + SIP) approach
  - Rely on SIP for low level signaling to set up the call
  - Rely on SIP signaling to negotiate the media transmission
  - Need to maintain both web service stack and SIP stack
  - Web service is for service layer abstraction and integration
  - Not a full SOA communication paradigm yet

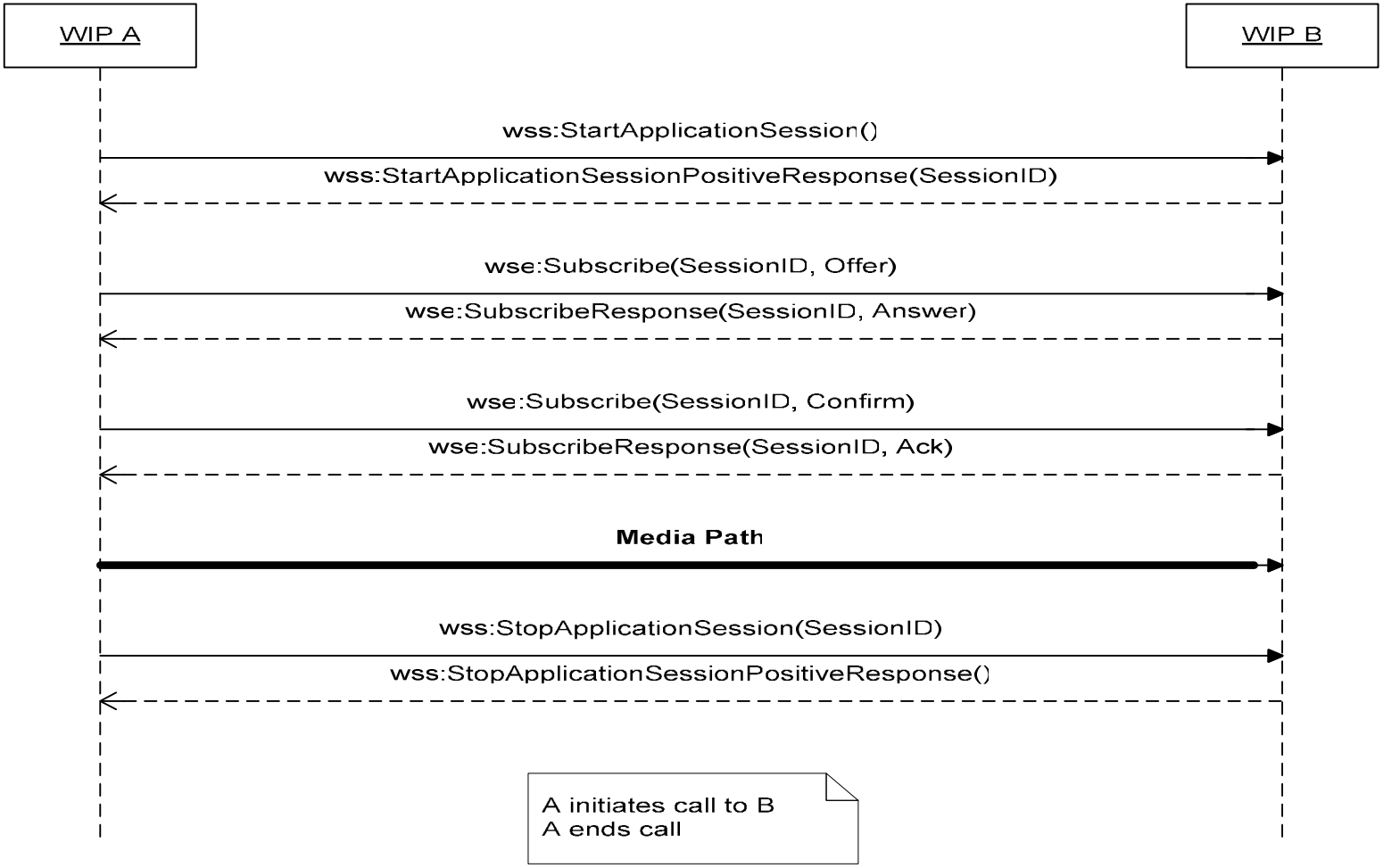
We need to advance more towards a full Web service based communication protocol for communication over IP.

WIP (Web Service Initiation Protocol)

- WIP (Web Service Initiation Protocol): Full Web service enabled communication protocol based on a single stack of web services for communication enablement
- WIP protocol stack is based on WS-Session, WS-Addressing and WS-Eventing
- In WIP, *signaling for media negotiation and control is modeled by a special web service event subscription between event sink and event source*

(Note: This approach solves the critical problem of a full web service based signaling for media negotiation and control. Together with WS-Session, two-way web services, etc., a web service and SOA based paradigm of communication over IP is realized by WIP."WIP: Web Service Initiation Protocol for Multimedia and Voice Communication over IP", Wu Chou, Li Li, Feng Liu, Proc. ICWS'06)

WIP Call Flow



WIP (*Web Service Initiation Protocol*)

- **A full featured Web Service and SOA based protocol for multimedia and voice communication over IP**
  - Fully web service based communication protocol without dependency on non-web service methods for session establishment and signaling
  - Two-way web service based communication service enablement
  - Extensible for peer-to-peer communication and for service extension and integration with switching environment/services, e.g. CSTA
  - New SOA communication endpoints that enable voice and multimedia communication as services
  - Extensible with web service protocols to achieve end-to-end security and reliable messaging
  - Agnostic to the transport protocols, e.g. communication services can be enabled through HTTP, JMS, etc.

Referenced Document: SRAD: Diamond Core

Some Advantages of WIP

- End-To-End Message Level Security Over IP

  End-to-end message level security over IP though WS-Security, which is open standard based and interoperable

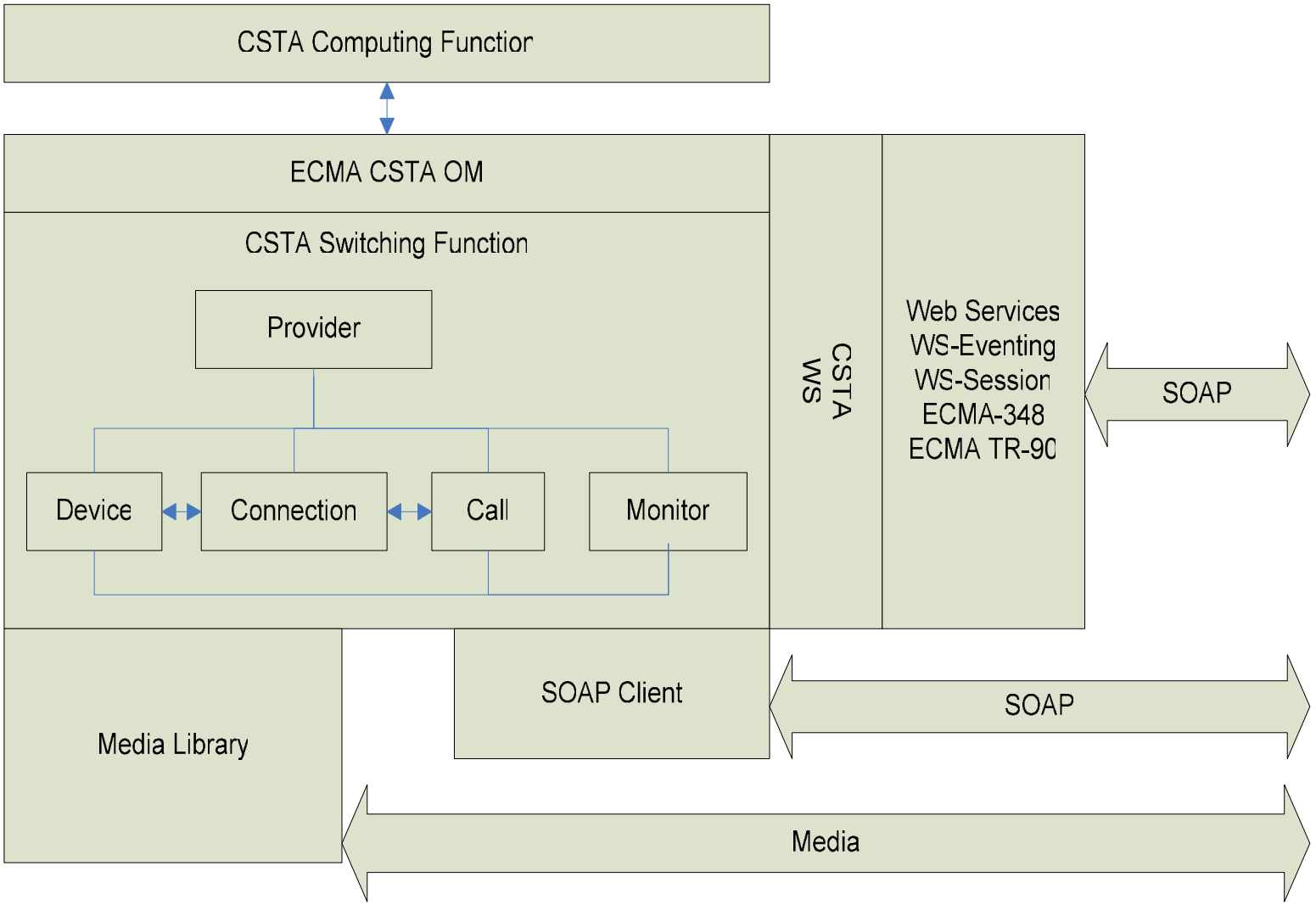
  (Transport layer security (e.g. TLS, SSL, etc.) is only hop-by-hop and cannot extend over unfriendly intermediate proxy, a major security issue for VoIP)

- Stackable and Extensible SOA Service Endpoints

  It can dynamically add and reassemble new services on WIP endpoint in an interoperable and declarative SOA fashion. It is service grid ready for scalable solutions.

  (e.g. a stackable framework to add/assemble services from simple P2P to switching P2P, CSTA, etc. that is declarative without ad-hoc or private structural change to WIP protocol.)

WIP Endpoint Architecture



Some WIP Call Control Features

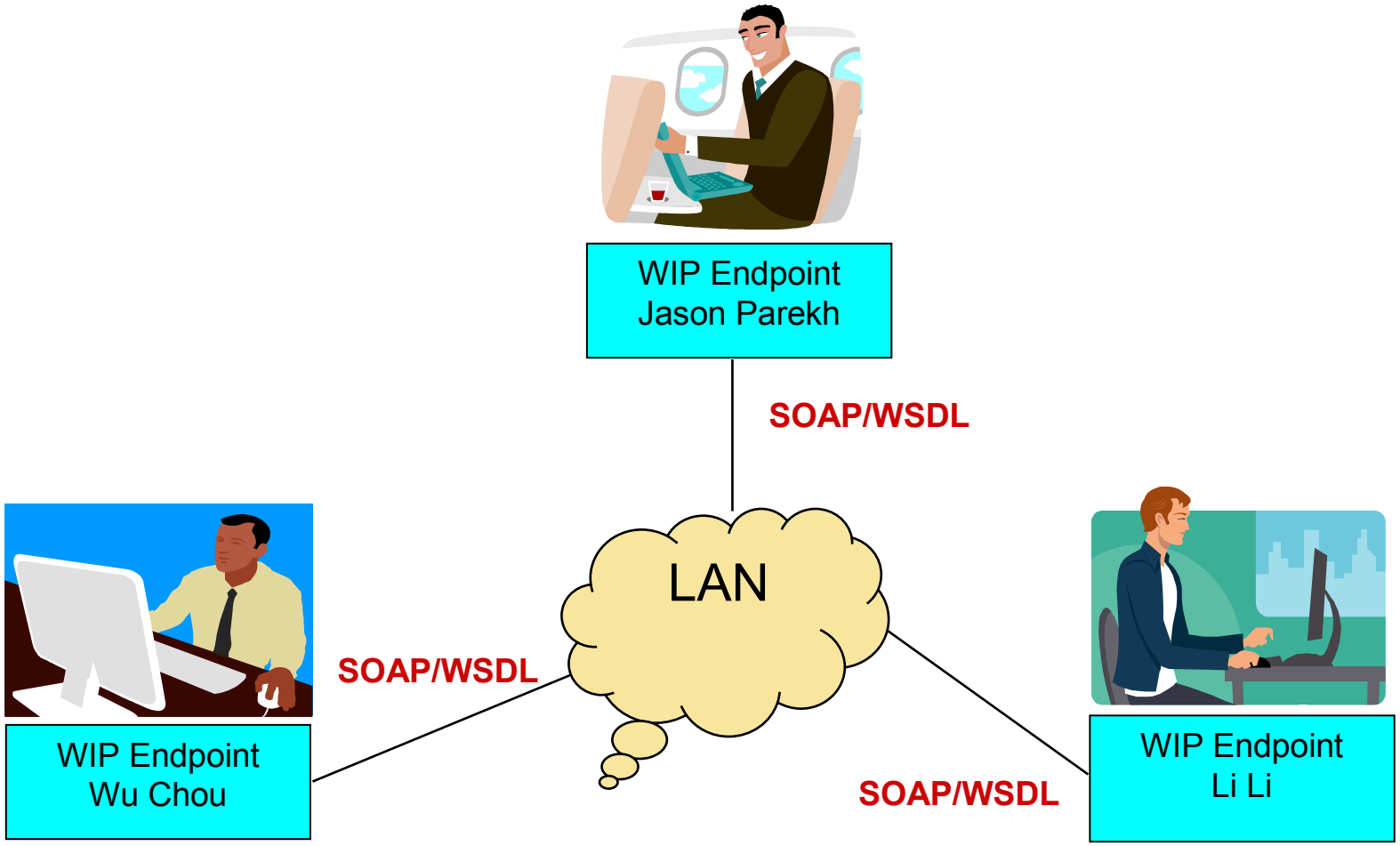
- Switching Function: ProviderFactory, Provider, Device, Logical and Physical Elements, Call, Connection, Monitor
  - Multimedia device: audio, video
- Call Control functions and events
  - MakeCall, AnswerCall, ClearCall, ClearConnection, SingleStepTransfer, MonitorStart, MonitorStop
  - DeliveredEvent, EstablishedEvent, OfferedEvent, OriginatedEvent, FailedEvent, TransferedEvent, ConnectionClearedEvent, ServiceInitiatedEvent

Performance Evaluation

Averaged over 9 trials, Performed on a Pentium-M 1.60Ghz/512MB and Pentium 4-M/768MB

	Client (round trip) (ms)	Server (processing time) (ms)
Start Application Session	44.67 (16.45)	< 10
Subscribe to Session	45.67 (10.07)	< 10
CSTA Monitor Start	30 (7.07)	< 10
Subscribe Offer	59 (23.64)	5.56 (16.67)
Subscribe Confirm	85 (29.03)	7.78 (8.33)
Total	256.11 (86.27)	13.34 (25)

WIP Communication Diagram



Intelligent Communication for Enterprise

Productivity, Efficiency, Usability through

Open interface
Modality
Mobility
Extensibility
Intelligent Network and Endpoints

The full potential of Intelligent Communication still remains to be uncovered.

Discussion and Q/A