# Presence and Awareness Services

Ramiro Liscano
Professor, SITE
U. Of Ottawa
rliscano@site.uottawa.ca

# Tutorial Outline

- Introduction to Presence and Awareness
- The SIP / SIMPLE Protocol
- Open Source Presence Platforms
  - Jabber.org
- The IM Standards Race
  - Jabber / XMPP vs SIP / SIMPLE
- Presence Policies.
- Private Presence Services and Other Info
  - MSN, ICQ, Yahoo Messenger, AOL.
  - PAM Forum

# Introduction to Contextual Services

# Contextual Services

- Always-On Connection
  - supports visibility for efficiency and innovation
- Connection to Human Work
  - sensing relevance and importance of activity
  - problem of privacy
    - relationships and mutual obligations
    - interruptability and intrusiveness
  - pervasive and mutually supporting services
    - location services
    - relationship services: personal calendar, organization chart, group management services
    - presence services
    - co-presence services
    - conference services
    - and probably many many more

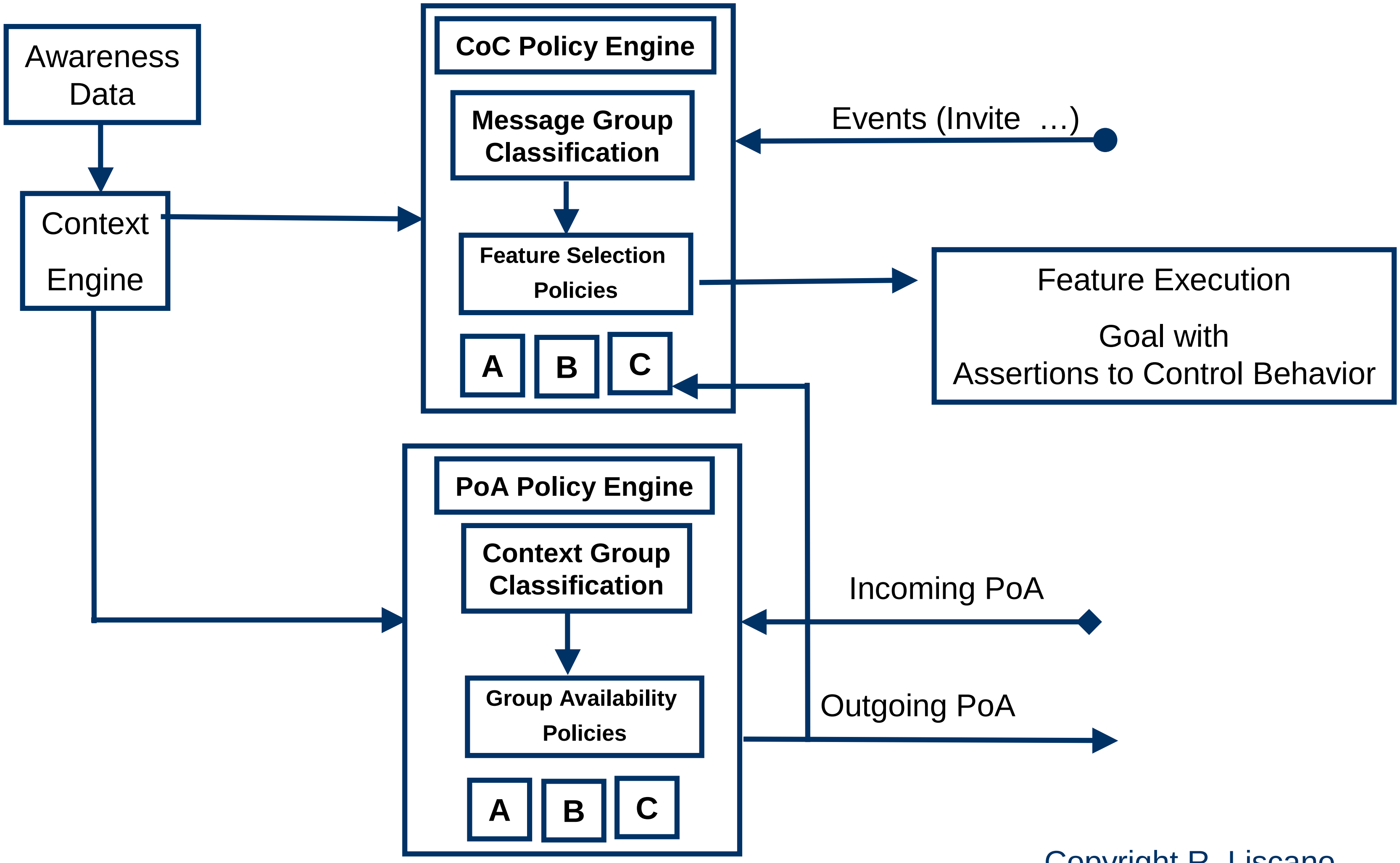
# CoC and PoA

- Always Connected Creates Tension
  - CoC -- control of communication
  - PoA -- projection of availability
- Challenge is required balance
  - between
    - fostering communication for innovation (and efficiency)
    - restricting communication for execution (and efficiency)
  - achieved by
    - sensing context
    - personalized policies

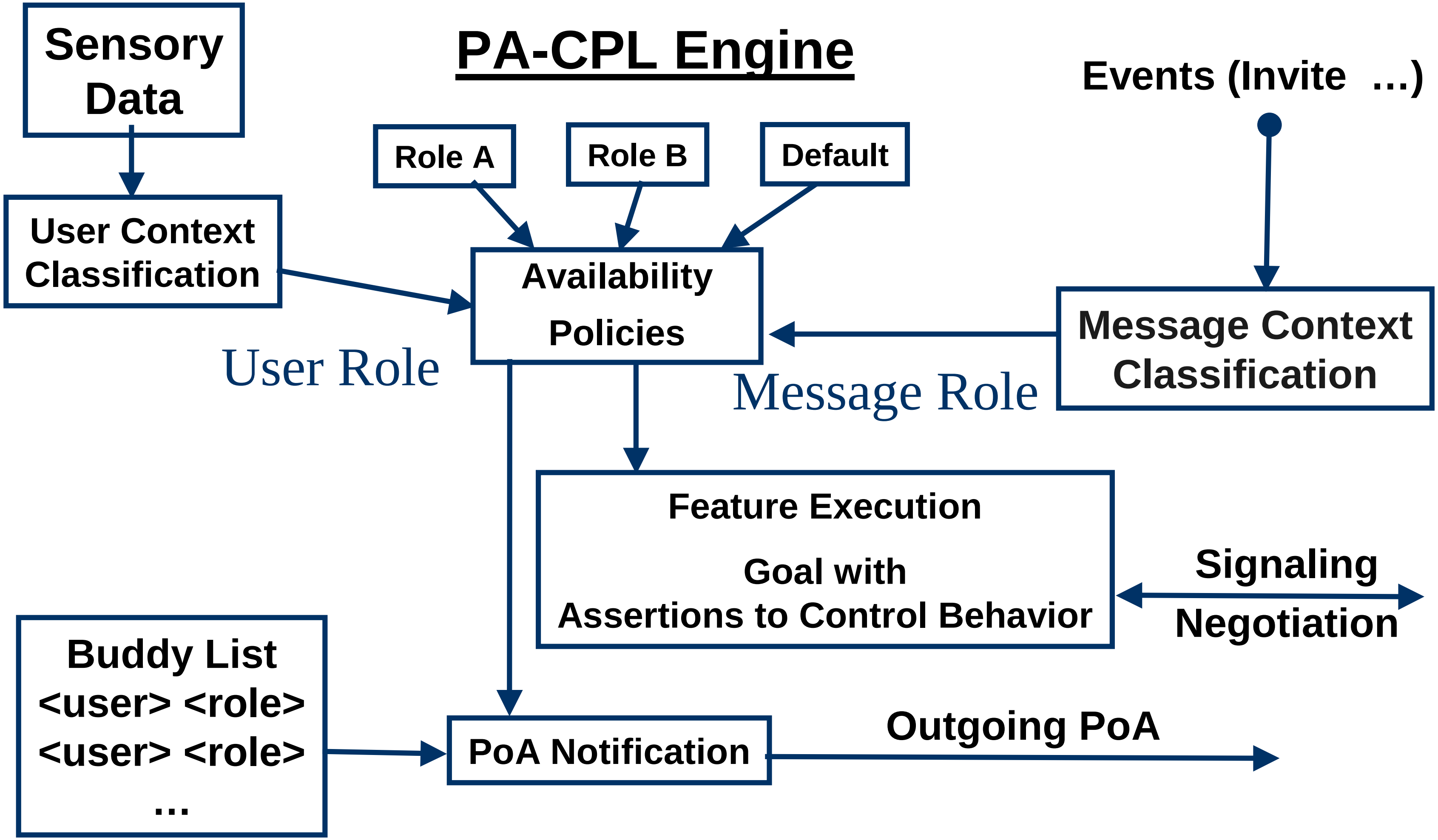
# Context-Aware Features

- Highly Personalized Call Control
  - personalization is about relationships
- Sharing of Awareness to Create Availability
  - awareness of current activities at the human level
- Coordination Support for Collaborative Activities
  - sensitivity to events (artifacts) in collaboration
  - alerting and notification services
- Mid-Call Services
  - adaptive sharing of resources
  - appropriate device behavior
    - ringer control for wireless devices
    - display sensitivity to screen capabilities

# Context-Aware Architecture


Awareness Data
Context Engine
CoC Policy Engine
Message Group Classification
Feature Selection Policies
A
B
C
Events (Invite …)
Feature Execution
Goal with Assertions to Control Behavior
PoA Policy Engine
Context Group Classification
Group Availability Policies
A
B
C
Incoming PoA
Outgoing PoA

# Personal Context-Aware Call Control



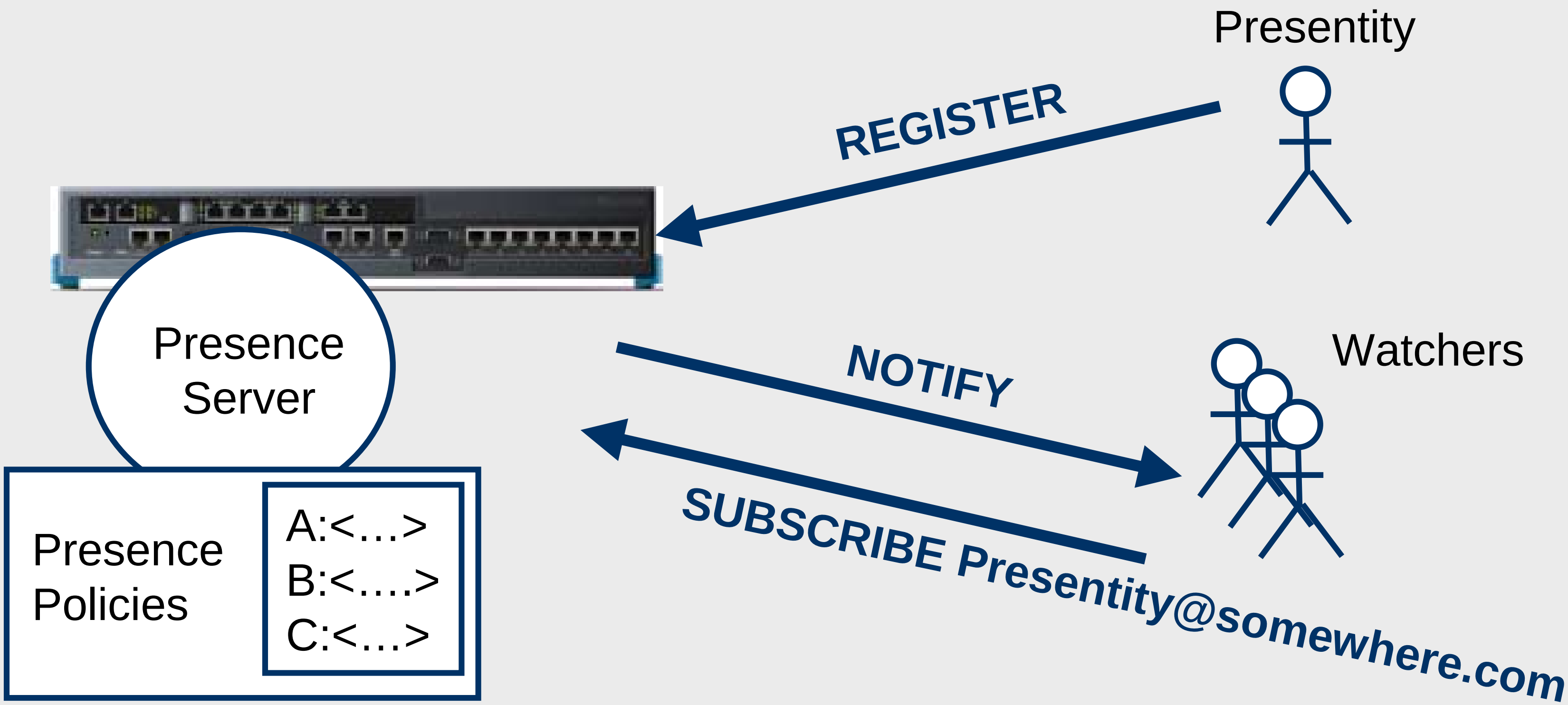
# Presence Communication Services

- Presence for network communications.
  - Is the concept of knowing whether a resource or person is registered with the system. We will keep away from monitoring systems that do not require user identification.
- Availability.
  - Is the concept of allowing other persons to contact you via a particular service.
  - Can also be extended to include common services not associated with a particular user.

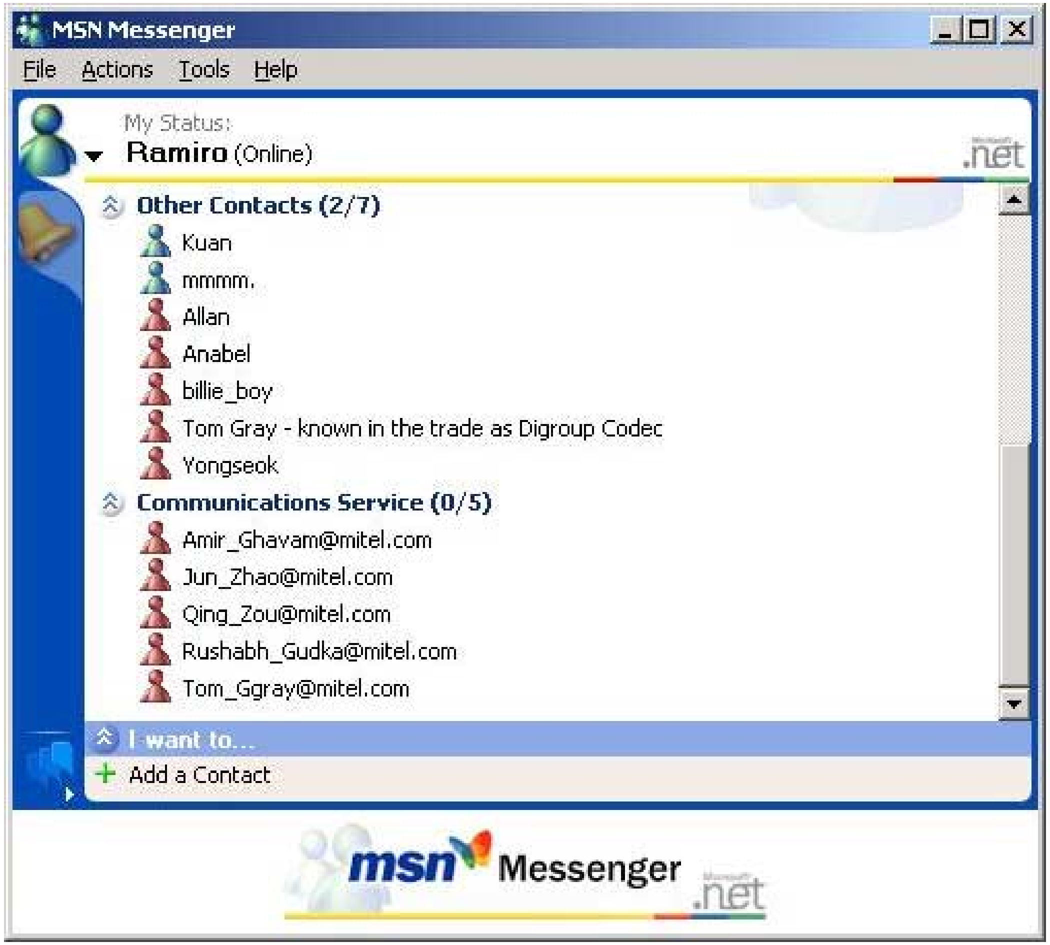
# Presence = Ad Hoc Communications

- Presence is knowledge of reachability, availability and status across all communications channels (e.g. networks, applications, transports - Internet, wireless, wireline).
- Facilitates Communication by Supplying “Contact” Information.
  - <contact priority="9">tel:09012345678</contact> (IMPP Example)
- Presence is not about communication protocols but availability protocols.

# Presentity and Watcher



# Basic Presence Features



My Status:
Ramiro (Online)

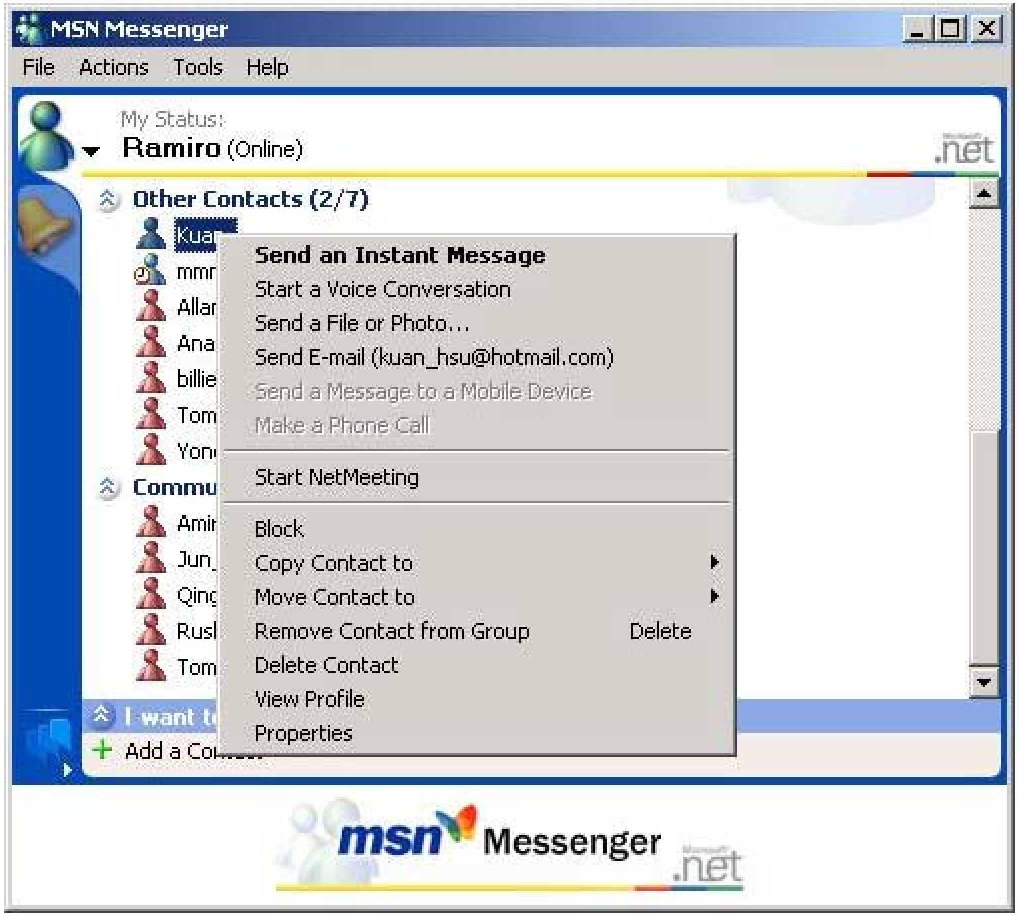
Other Contacts (2/7)

- Kuan
- mmmm.
- Allan
- Anabel
- billie_boy
- Tom Gray - known in the trade as Digroup Codec
- Yongseok

Communications Service (0/5)

- Amir_Ghavam@mitel.com
- Jun_Zhao@mitel.com
- Qing_Zou@mitel.com
- Rushabh_Gudka@mitel.com
- Tom_Ggray@mitel.com

# Communications Services



# SIP / SIMPLE Protocol

SIMPLE - IETF Applications Area Working Group

# SIP / SIMPLE Protocol

- SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE)
- IETF Applications Area Working Group
  - Focuses on the application of the SIP Protocol (RFC 3261) to the suite of services collectively known as instant messaging and presence.

# Primary work of SIMPLE

- A proposed standard SIP extension documenting the transport of Instant Messages in SIP.
- A proposed standard SIP event package and any related protocol mechanisms used to support presence.
- An architecture for the implementation of a traditional buddylist-based instant messaging and presence application with SIP, including new mechanisms for message confirmation delivery, indications for when a party is in the process of typing a message, secure buddylist manipulation operations, and the extension of the CPIM presence format to describe typical IM states.

# Components of SIP Simple

- Presence Event Package
  - draft-ietf-simple-presence-10.txt (July 2003).
- Extents the SIP Event Package RFC 3265
- The package is identified by:
  - Event: presence header field.

# Presence Event Package

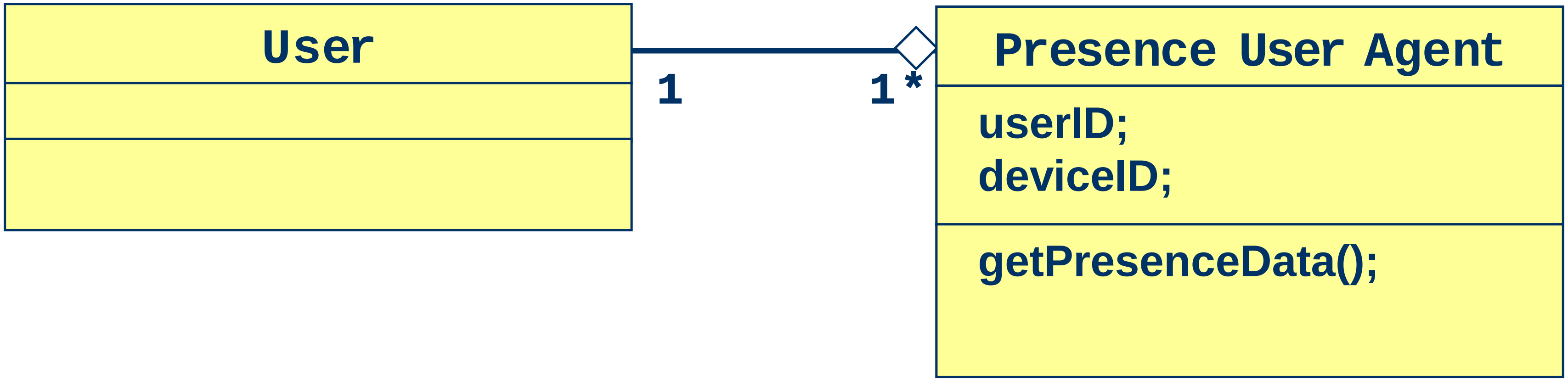
- Core extensions to SIP to support Presence and Awareness.
- Terminology
  - Presence is defined as the willingness and ability of a user to communicate with other users on the network.
  - Presentity & Watcher
  - Presence User Agent (PUA).
  - Presence Agent (PA).

# Presentity and Watcher

- Presentity
  - The entity that is projecting its presence information. More often this is a user identified with a SIP address.
- Watcher
  - The entity that is receiving presence information. More often this is a user identified with a SIP address.

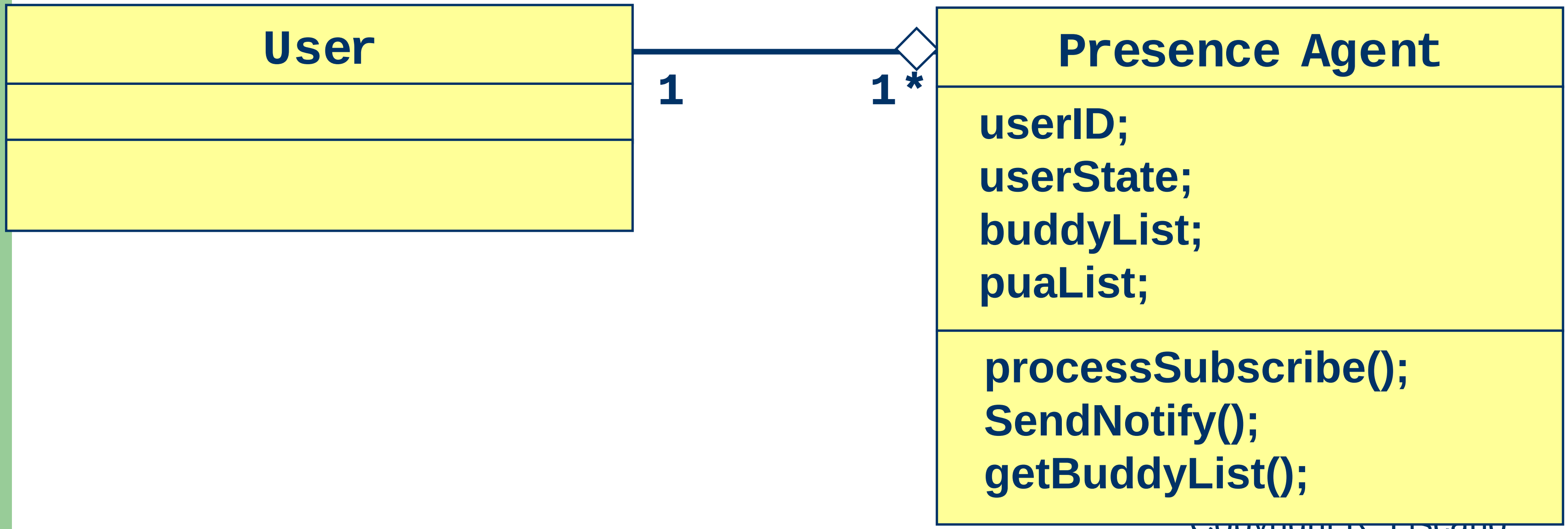
# Presence User Agent

- A Presence User Agent manipulates presence information for a presentity.



# Presence Agent

- A presence agent is a SIP user agent which is capable of receiving SUBSCRIBE requests, responding to them, and generating notifications of changes in presence state.



User

1

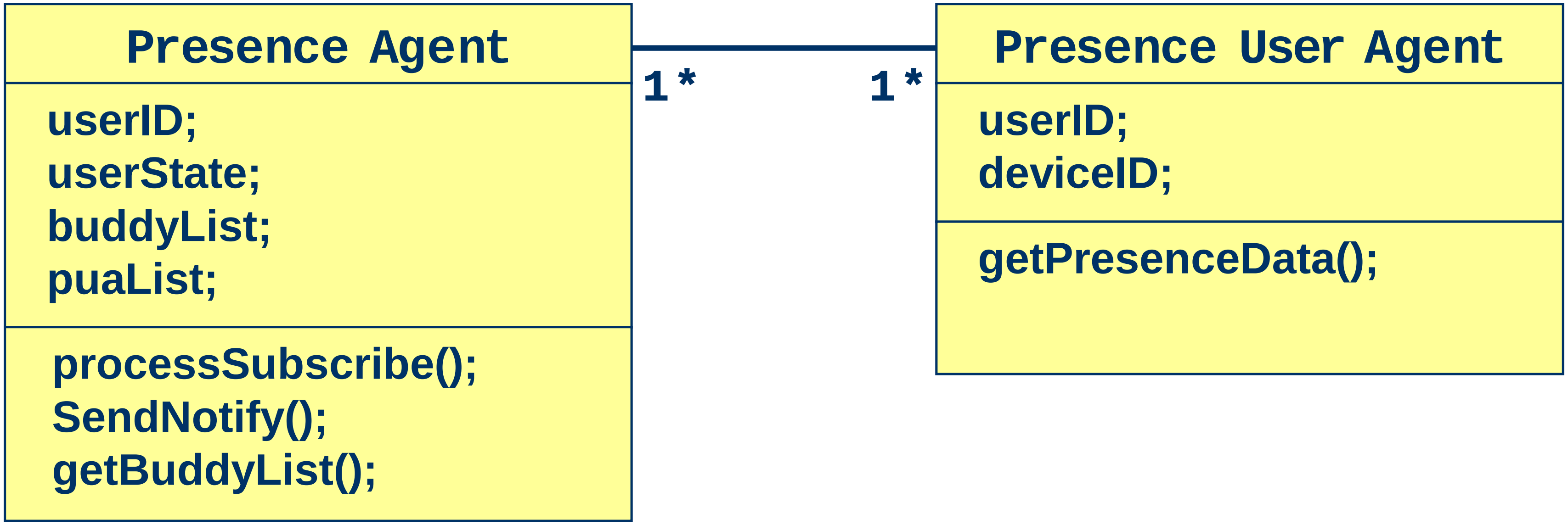
1*

Presence Agent

userID;
userState;
buddyList;
puaList;

processSubscribe();
SendNotify();
getBuddyList();

# PA & PUA Relationship



**Presence Agent**

userID;
userState;
buddyList;
puaList;

processSubscribe();
SendNotify();
getBuddyList();

1*

1*

**Presence User Agent**

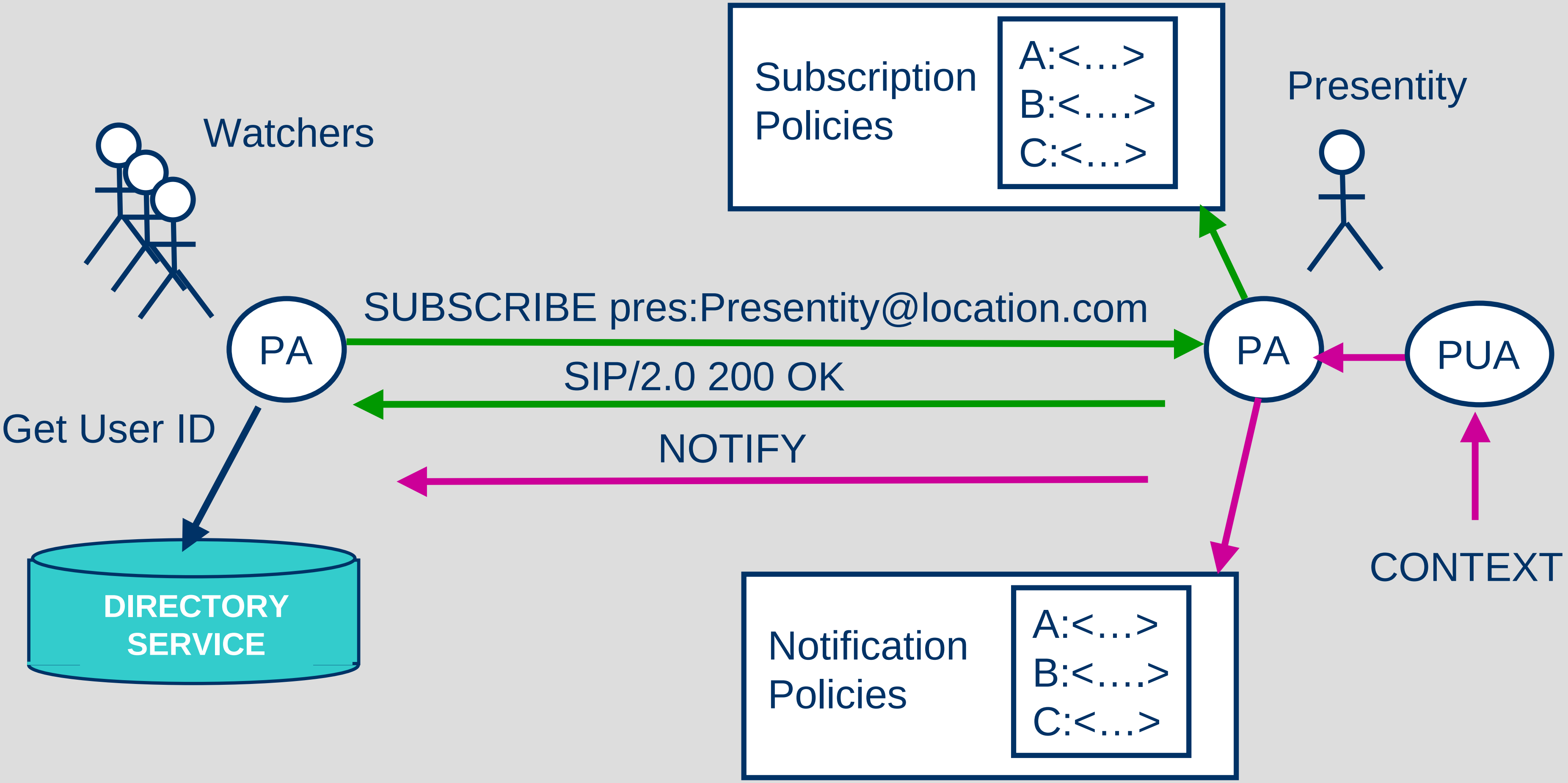
userID;
deviceID;

getPresenceData();

# Defined SIP Method Extensions

- SUBSCRIBE “I would like to monitor your presence.”
- NOTIFY “Hi! I’m online now and available to communicate”
- MESSAGE “I’m sending you an instant message.”

# Basic Messaging Model



# Presence Identity

- A Presentity is identified through a Presence URI:
  - pres:user@domain
- This is a protocol independent identifier:
  - that needs to be resolved to a protocol specific identifier: sip:user@domain

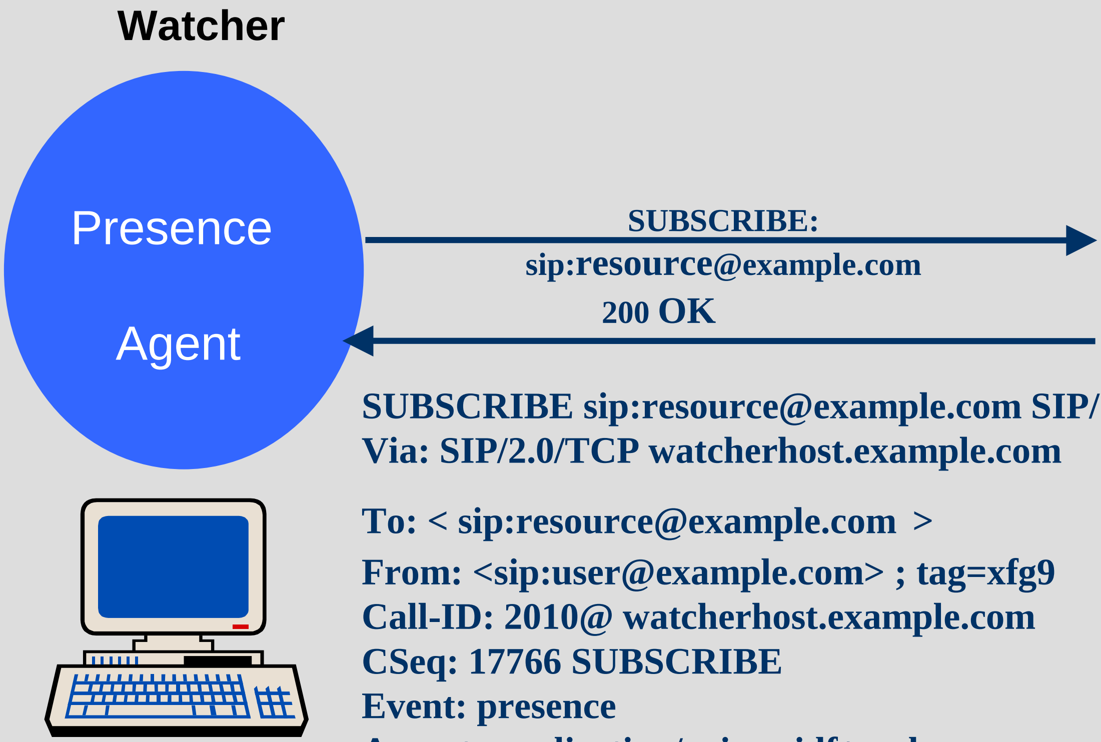
# Presence Agent Responsibilities

- Authenticate and Authorize the Subscription request.
- Can either:
  - Be Successful
    - 200 OK Returned
  - Be Rejected
    - 403 or 603 Returned
  - Be Pending (Authorization can not be determined)
    - 202 returned
- NOTIFY message is immediately sent.
  - Bogus Presentity state may be sent.
- A peer-to-peer dialog is established
  - Identified by the SIP tag: header field

# Time to Live

- Subscription Requests
  - Use a Expires: header field.
  - Can re-subscribed before timeout.
- Subscriber can Terminate
  - Sends a SUBSRIBE with Expires: 0
- PA can Terminate
  - Sends a NOTIFY with a Subscription-State header field of “terminate” (not specified up to user).

# Presence - SUBSCRIBE Message



```
SUBSCRIBE sip:resource@example.com SIP/2.0
Via: SIP/2.0/TCP watcherhost.example.com

To: < sip:resource@example.com  >
From: <sip:user@example.com> ; tag=xfg9
Call-ID: 2010@ watcherhost.example.com
CSeq: 17766 SUBSCRIBE
Event: presence
Accept: application/cpim-pidf+xml
Contact:< sip:user@ watcherhost.example.com>
Expires: 600
Content-Length: 0
```

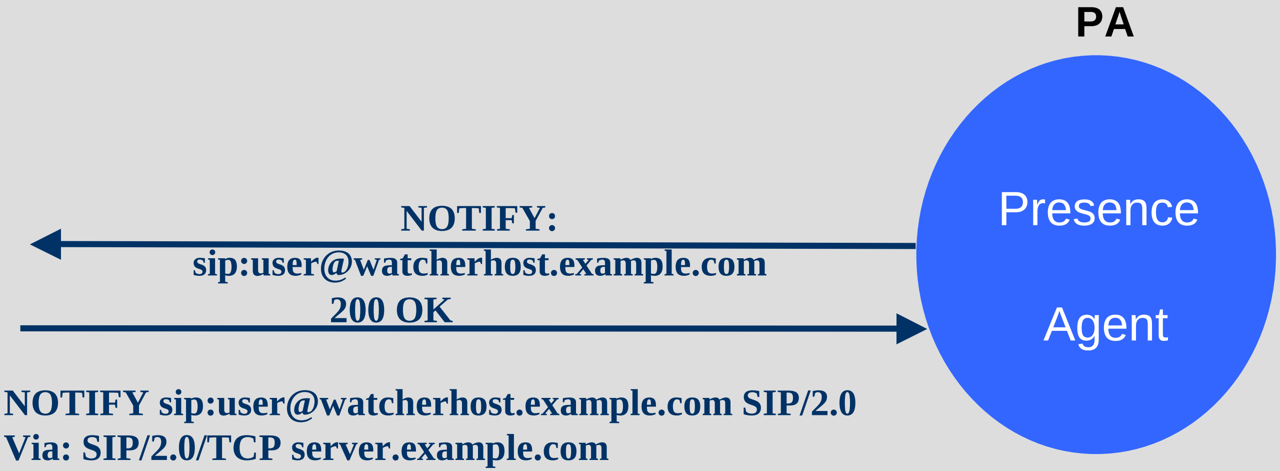
Note: SUBSCRIPTION requests may contain a filter document in the body.

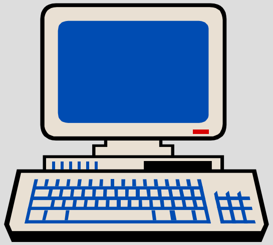
(Future RFP?)

# Presence - NOTIFY Message

Note: PA can send NOTIFY any time.

Rate: < 1 / 5 sec.

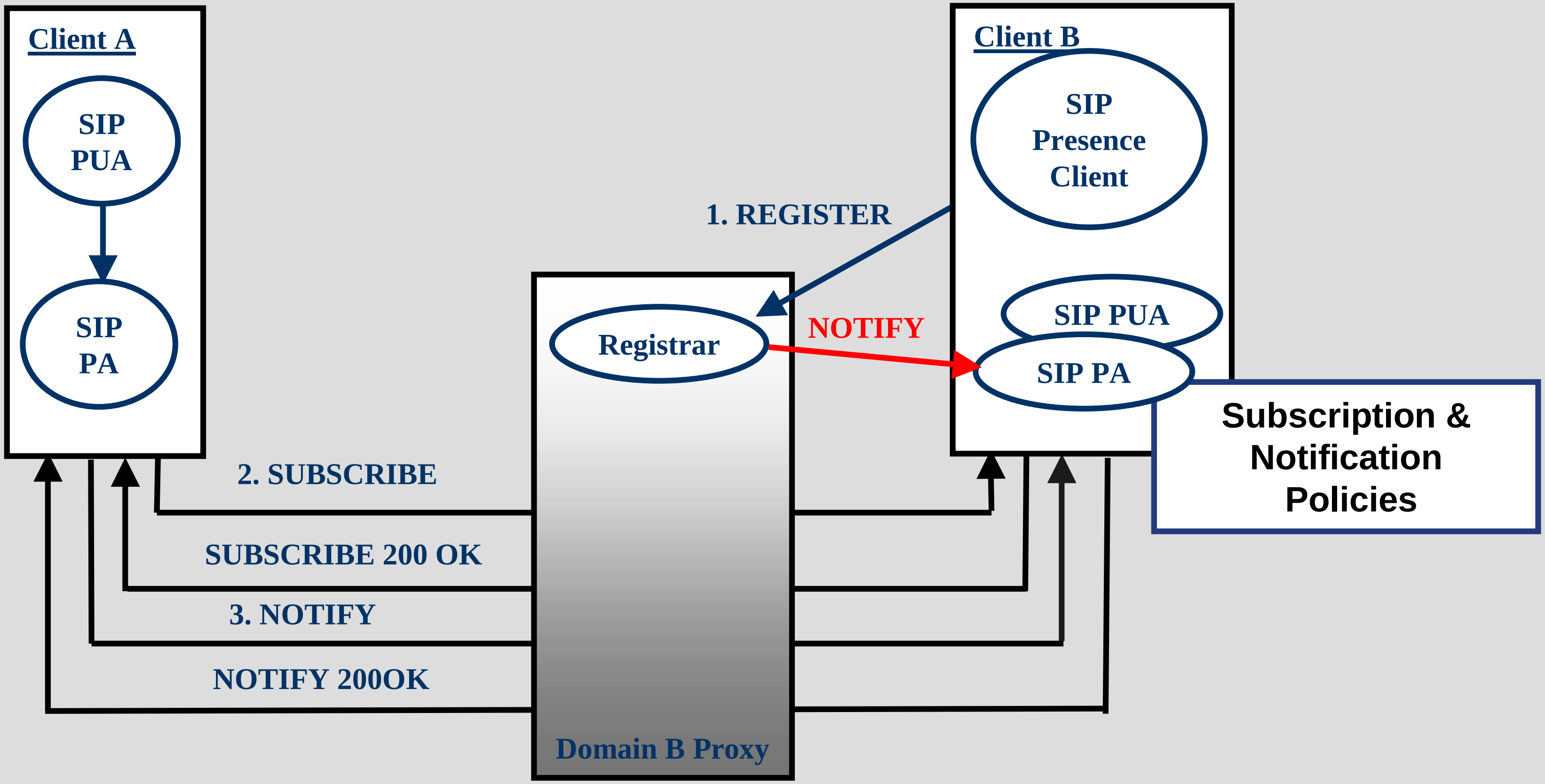




```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
Via: SIP/2.0/TCP server.example.com
From: <sip:resource@example.com>;tag=ffd2
To:< sip:user@example.com>;tag= xfg9
Call-ID: 2010@ watcherhost.example.com
Event: presence
Subscription-State: active;expires=599
CSeq: 8775 NOTIFY
Contact: sip:server@example.com
Content-Type: application/cpim-pidf+xml
Content-Length: ...

[PIDF Document … more to come on this]
```

# Presence End to End Model



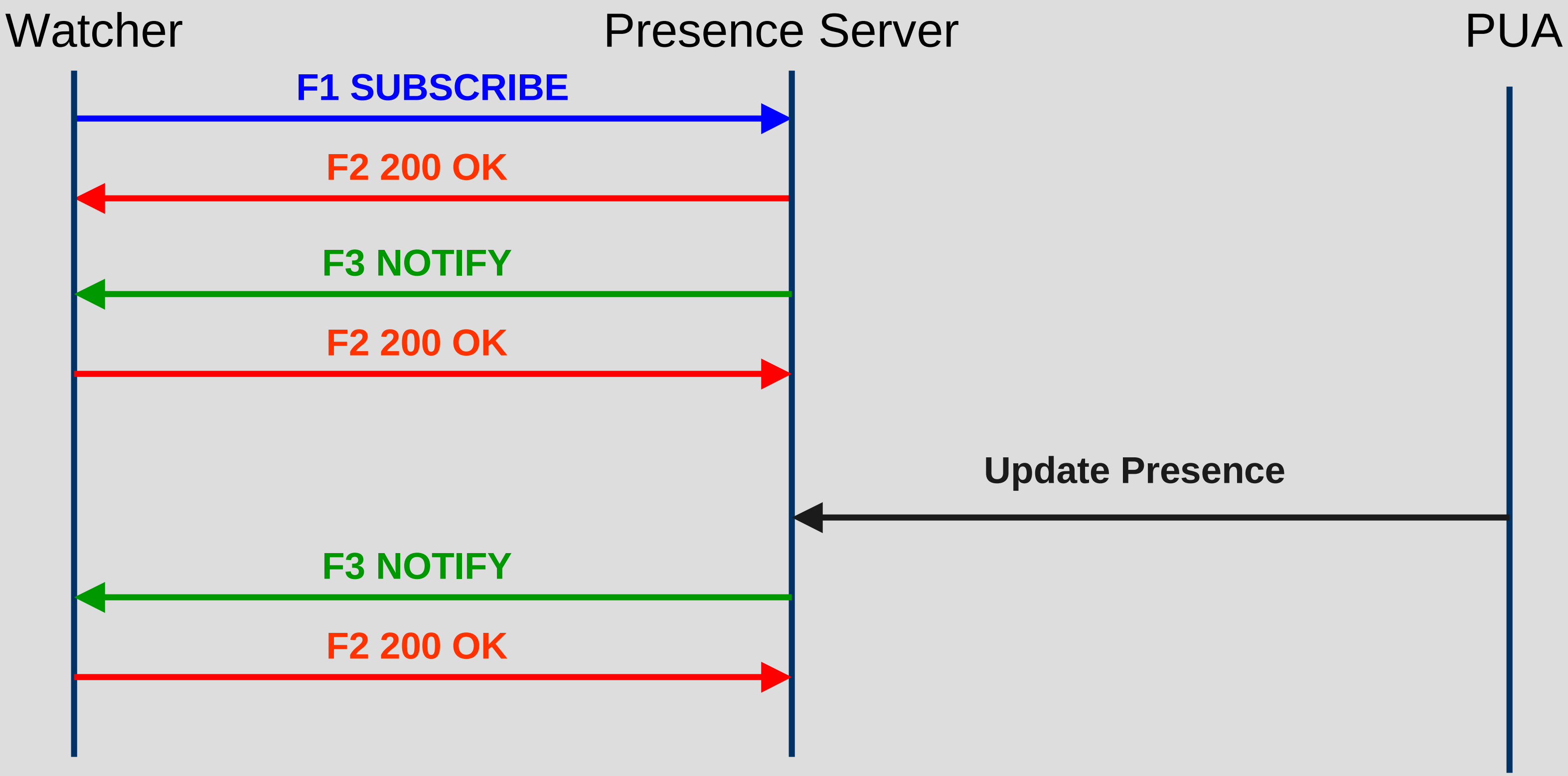
# Use of the Register Information

- The register is a useful way to determine contact information.
  - The address-of-record field in the REGISTER SIP message can be used for contact information.
    - Good way to leverage messages to be forced via a Proxy.
  - PA can be co-located with the register. Presence server-based model
  - A PUA can also use the registration information to change the state of the user. SIMPLE draft recommends the use of SIP caller preferences extension for this [

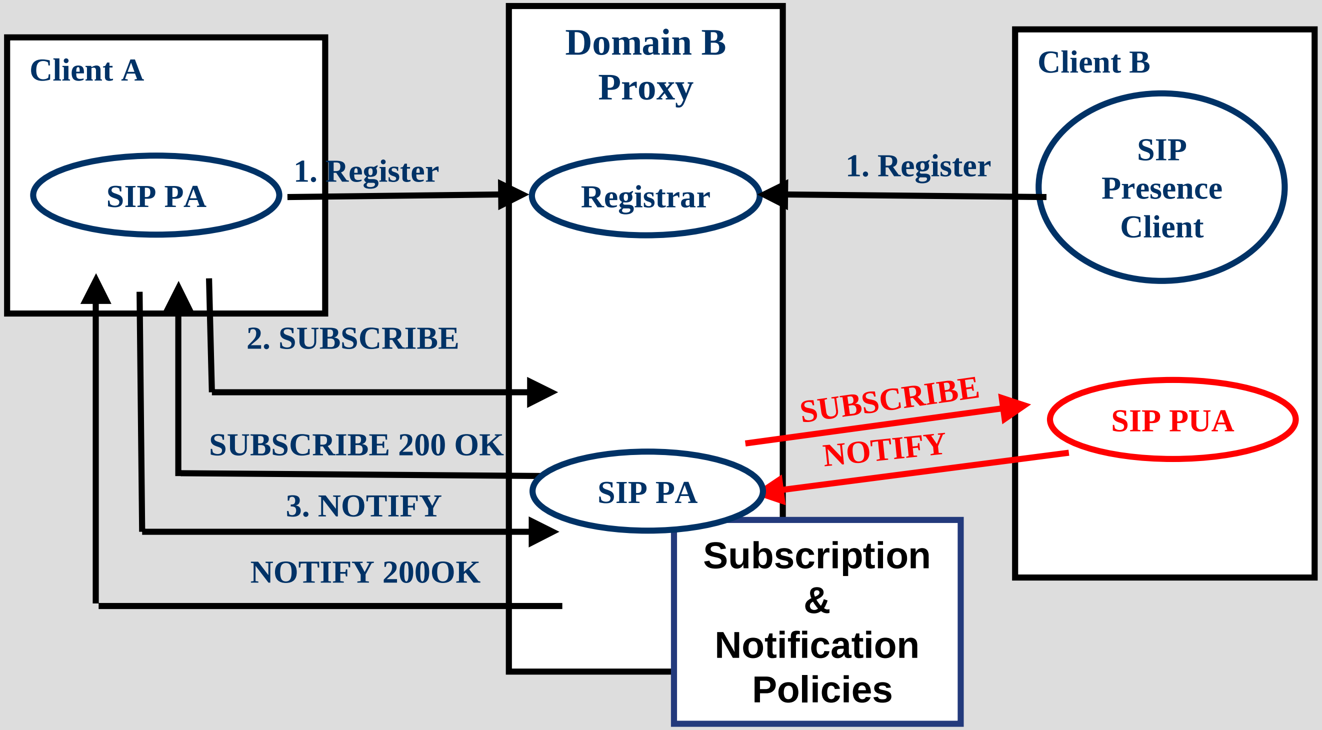
# SIMPLE Presence Server Model

- SIMPLE defines the use of 2 types of Presence servers.
  - Presence server that can act as a PA or a proxy server.
  - Edge Presence Server where the PA and PUA are co-located. The PA is aware of the presence information that is available from the PUA.

# SIMPLE Presence Server Model



# Presence Server Model



Subscription
&
Notification
Policies

# PA as a State Agent

- When the PA and PUAs are not co-located the PA collects presence state information from the PUAs and aggregates it into a presence document.
  - A centralized state agent is required to perform this aggregation.
  - The state agent is the presence server.
  - The PA subscribes to the PUAs to gather state information.
  - This is done with the watcherinfo event package. (IETF Draft – A watcher information event template-package for SIP, J. Rosenberg.)

# Presence Information Data Format

- The Presence Information Data Format (PIDF) is a an IETF draft that specifies a common presence data format.
  - draft-ietf-impp-cpim-pidf-08.txt (Expires Nov 2003).
- Based on a URL Namespace
  - urn:ietf:params:xml:ns:pidf

# Simple PIDF Presence

```
<?xml version="1.0" encoding="UTF-8"?>
     <impp:presence
xmlns:impp="urn:ietf:params:xml:ns:pidf"
         entity="pres:someone@example.com">
       <impp:tuple id="sg89ae">
         <impp:status>
           <impp:basic>open</impp:basic>
         </impp:status>
         <impp:contact priority="0.8">tel:+09012345678
         </impp:contact>
       </impp:tuple>
   </impp:presence>
```

# Presence Tuple

- Consists of:
  - <status> element,
  - Optional extension elements,
  - Optional <contact> element,
  - 1* Optional <note> elements,
  - Optional <timestamp> element.

# <status> element

```
<impp:status>
  <impp:basic>open</impp:basic>
</impp:status>
```

- The basic element is required that specifies an “open” or “closed” status referring to the user’s availability.
- May contain other extensions that give user location, or status information as defined by other URL namespaces.

```
<status>
         <basic>open</basic>
         <im:im>busy</im:im>
         <myex:location>home</myex:location>
</status>
```

# <contact> element

- Contains a URL of the contact address
- An optional priority over other contacts

```
<impp:contact
     priority="0.8">tel:+09012345678
 <impp:contact
priority="1.0">mailto:someone@example.com
</impp:contact>
</impp:contact>
```

# A PIDF Message Example

```
<presence xmlns="urn:ietf:params:cpim-presence:"
          entity="pres:shingo@jp.fujitsu.com"/>
  <tuple id="mobile-im">
    <status>
      <value>open</value>
      <value
        type="urn:ietf:params:cpim-presence:status-type:im">busy</value>
      <value
        type="urn:example-com:cpim-status-type:location"
        schema="http://www.example.com/impp/location.dtd">home</value>
    </status>
    <contact priority="2">im:shingo@mobilecarrier.ne.jp</contact>
    <note>Don't Disturb Please!</note>
    <timestamp>2001-10-27T16:49:29Z</timestamp>
  </tuple>
</presence>
```

# The CPIM Standard

- Common Presence and Instant Messaging (CPIM)
  - A proposed standard to deal with the contents of a presence document, i.e. that information that is carried in the body part of the NOTIFY message.
  - draft-ietf-impp-cpim-msgfmt-08.txt – July 2003.
  - Objective is to define a number of operations and criteria to be satisfied for interworking diverse instant messaging protocols.
  - Specifies little about user presence but more on message metadata.

# CPIM Message Format

m:  MIME headers for the overall message
s:  a blank separator line
h:  message headers
e:  encapsulated MIME object containing the message content
x:  MIME security multipart message wrapper

# Open Source Presence Platforms

Jabber.org

# Jabber.org

- Jabber Software Foundation www.jabber.org
- Open XML protocol for the real-time exchange of messages and presence between any two points on the Internet.
- Individual rather than organizational membership.
  - Prospective members are expected to have made significant contributions to Jabber.

# Popular Jabber Clients

- Exodus for Windows (http://exodus.jabberstudio.org/)
- Gabber for Linux (http://gabber.sourceforge.net/)
- There are many other Jabber clients to choose (http://www.jabber.org/user/clientlist.php)
- Jabber servers (http://www.jabber.org/network/)

# Jabber basics

- Jabber uses a client-server architecture, not a direct peer-to-peer architecture.
  - SIP / SIMPLE is primarily a peer-to-peer protocol.
- A Jabber client connects to a Jabber server on a TCP socket over port 5222.
  - This connection is "always-on" for the life of the client's session.
  - If you are behind a firewall you might not be able to use some of the clients.
    - WebClient at http://webim.jabber.com/
- When a client connects to a server, it opens a one-way XML stream from the client to the server, and the server responds with a one-way XML stream from the server to the client

# Jabber basics

- Jabber's architecture is modeled after that of e-mail. Each user connects to a "home" server, which receives information for them, and the servers transfer data among themselves on behalf of users.

```
<message from='juliet@capulet.com' to='romeo@montague.net'>
  <body>Wherefore art thou, Romeo?</body>
</message>
```

# Role of a Jabber Server

- Jabber server plays three primary roles:
  - Handling client connections and communicating directly with Jabber clients.
  - Communicating with other Jabber servers.
  - Coordinating the various server components associated with the server such as registration, authentication, presence, contact lists, offline message storage.

# Role of Jabber Clients

- The only things a Jabber client must do are:
  - Communicate with the Jabber server over TCP sockets.
  - Parse and interpret well-formed XML "chunks" over an XML stream.
  - Understand the core Jabber data types (message, presence, and iq).

# Jabber ID (JID)

- The JIDs are formed of a domain, node, and resource in the following format:
  - [node@]domain[/resource]
- Domain Name is the primary identifier. It represents the Jabber server to which the entity connects.
- The Node is the secondary identifier. It represents the "user".
- Resource is an optional third identifier. Used to identify specific objects that belong to a user. Resources enable a single user to maintain several simultaneous connections.

# Jabber Protocol

- Based on the IETF XMPP Draft
  - draft-ietf-xmpp-im-11(November 2003).
  - Meets the requirements of RFC 2779 but does not leverage SIP.
  - Basic elements are </message>,</presence>, and </iq>

# Message Element

- A message may possess the following attributes:
  - to: Specifies the intended recipient of the message.
  - from: Specifies the sender of the message.
  - id: An optional unique identifier for the purpose of tracking messages.
  - type: An optional specification of the conversational context of the message.
  - body: The textual contents of the message.
  - subject: The subject of the message.
  - thread: A random string that is generated by the sender and that MAY be copied back in replies; it is used for tracking a conversation thread.
  - error: An error code.

# Message Element Example

```
<message from='juliet@capulet.com‘
   to='romeo@montague.net'>
     <body>Wherefore art thou, Romeo?</body>
</message>
```

# Presence Element

- A presence MAY possess the following attributes:
  - to: Specifies the intended recipient of the presence.
  - from: Specifies the sender of the presence.
  - id: A unique identifier for the purpose of tracking presence.
  - type: Describes the availability state, subscription request, presence request, or error.
    - These can be unavailable, subscribe, subscribed, unsubscribe, unsubscribed, probe, error.
  - show: Describes the availability status of an entity or specific resource.
    - away - Entity or resource is temporarily away.
    - chat - Entity or resource is free to chat.
    - xa - Entity or resource is away for an extended period.
    - dnd - Entity or resource is busy.
    - status - An optional ascii description of availability status.
    - priority - A non-negative integer representing the priority level of the connected resource, with zero as the lowest priority.
    - error – an error code.

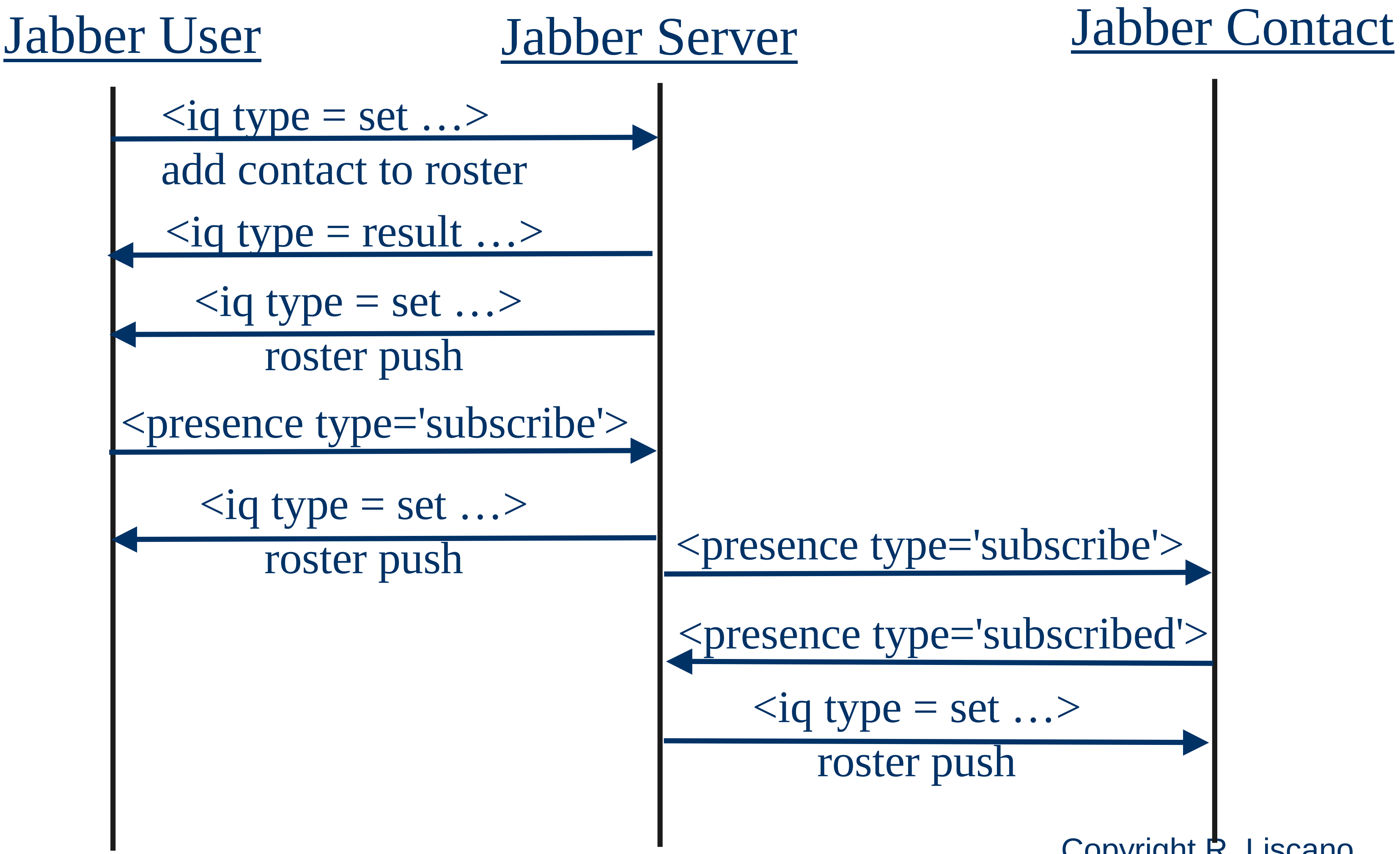
# Presence Element Example

```
<presence>
   <show>dnd</show>
   <status>Busy fighting the Romans</status>
</presence>
```

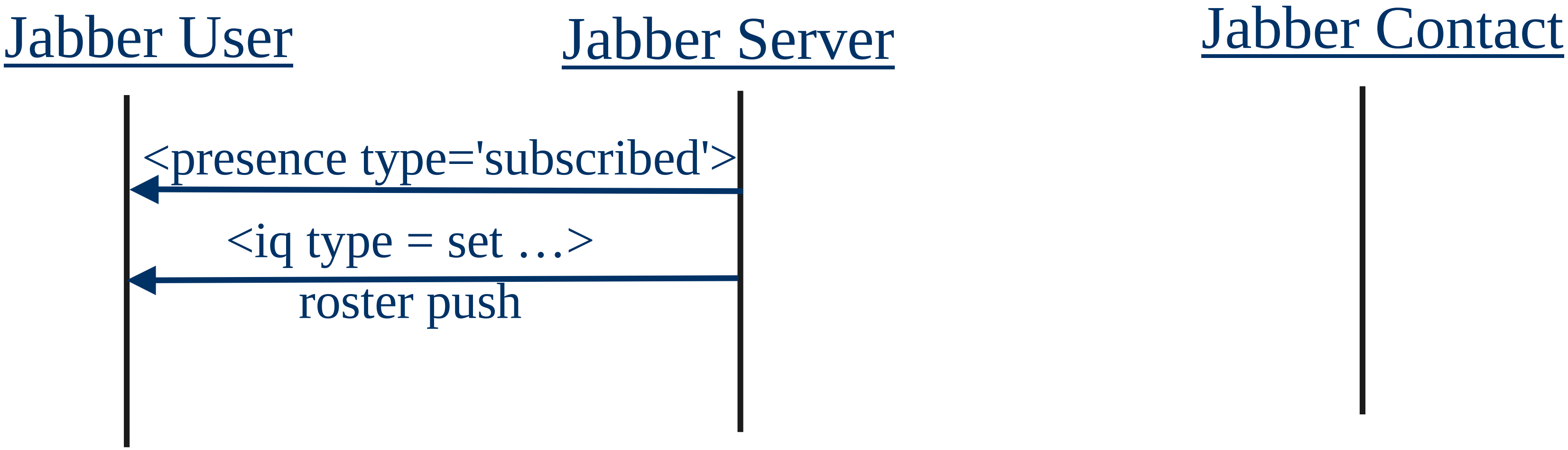
# Info / Query Element

- Info/Query, or IQ, is a simple request-response mechanism and MAY possess the following attributes:
  - to: Specifies the intended recipient of the IQ.
  - from: Specifies the sender of the IQ.
  - id: An optional unique identifier for the purpose of tracking the request-response interaction.
  - type: The required 'type' attribute specifies a distinct step within a request-response interaction.
    - get - A request for information.
    - set - Message contains data intended to provide required data, set new values, or replace existing values.
    - result - Message is a response to a successful get or set request.
    - error - An error has code.

# Subscription Process



# Subscription Process (cont.)



Jabber User

Jabber Server

Jabber Contact

<presence type='subscribed'>

<iq type = set …>

roster push

JABBER USER'S ROSTER:

<item jid='contact@host‘
subscription='to‘
name='contact'/>

CONTACT'S ROSTER:

<item jid='jabberuser@host'
subscription='from'/>

# Supported Gateways

- The following four gateways are the ones most commonly requested:
  - AIM Transport (also handles ICQ)
  - ICQ Transport
  - MSN Transport
  - Yahoo Transport

# Jabber Enhancement Proposals

- The Jabber Software Foundation tries to define a number of other relevant presence enhancements that include a total of 94 more features. Some that are of interest are:
  - JEB 0020 Feature Negotiation. http://www.jabber.org/jeps/jep-0020.html
  - JEB 0030 Service Discovery http://www.jabber.org/jeps/jep-0030.html
  - JEB 0050 Adhoc Commands http://www.jabber.org/jeps/jep-0050.html

# Recent XMPP Drafts

- The most recent XMPP Internet-Drafts are:
  - draft-ietf-xmpp-core-12 (text | HTML) -- XML streams, SASL, TLS, and core XMPP data elements.
  - draft-ietf-xmpp-im-11 (text | HTML) -- extensions to XMPP for standard instant messaging and presence functionality.
  - draft-ietf-xmpp-nodeprep-02 (text | HTML) -- a stringprep profile for XMPP node identifiers.
  - draft-ietf-xmpp-resourceprep-02 (text | HTML) -- a stringprep profile for XMPP resource identifiers.
  - draft-ietf-xmpp-e2e-03 (text | HTML) -- end-to-end object encryption in XMPP.
  - draft-miller-xmpp-cpim-00 (text | HTML) -- mapping XMPP to CPIM (obsolete, needs to sync up with recent IMPP work).

# The IM Standards Race. Let’s not forget Presence along the way.

This race is clearly among the IETF XMPP (Jabber) and PIDF (SIP/SIMPLE).

# Jabber and SIP

- "SIP is really good at rendezvous, call control, mobility support -- none of which exists in Jabber/XMPP. Jabber is really good at carrying blocks of data between Point A and Point B, through firewalls, and with applications attached to the transport, like recording applications and so on. By putting them together to take advantage of Jabber for point-to-point messaging, it's really the ideal combination."
  - Jonathan Rosenberg, Dynamicsoft's chief scientist.

# Comparison Chart

| SIP / SIMPLE | Jabber / XMPP |
|---|---|
| Peer-to-peer (New networking paradigm) | Client server and Server to Server (Networking based on email model) |
| Control protocol for sessions. (NATs and Firewalls are a challenge) | Data protocol over XML. |
| SIMPLE messages require SIP headers<br>IM “Hello” (268 bytes)<br>Presence Msg (602) | Jabber messages independent of session<br>IM “Hello” (100 bytes).<br>Presence Msg (67) |

# The Future of Jabber and SIP

1. XMPP/Jabber disappears from the marketplace; this seems unlikely given the huge installed base of Jabber commercial and open source servers and the rapid growth of XMPP/Jabber.
2. SIMPLE gains enough market adoption (especially on Microsoft platforms) to coexist with XMPP/Jabber; gateways and server-side interfaces provide connectivity between the two protocols.
3. XMPP/Jabber continues its rapid growth and SIMPLE does not take off, effectively consigning SIMPLE to the sidelines; SIP devices treat XMPP as just another data transport protocol and use SIP to negotiate XMPP sessions for IM functionality.
4. XMPP/Jabber supplants SIP itself through the ability to negotiate media sessions using any XMPP device; this is possible but less likely, given the large number of SIP devices.

Taken from The Instant Messaging Standards Race: Comparing XMPP/Jabber and SIP/SIMPLE, Jabber White Paper, May 2003.

# Designing a Presence Service

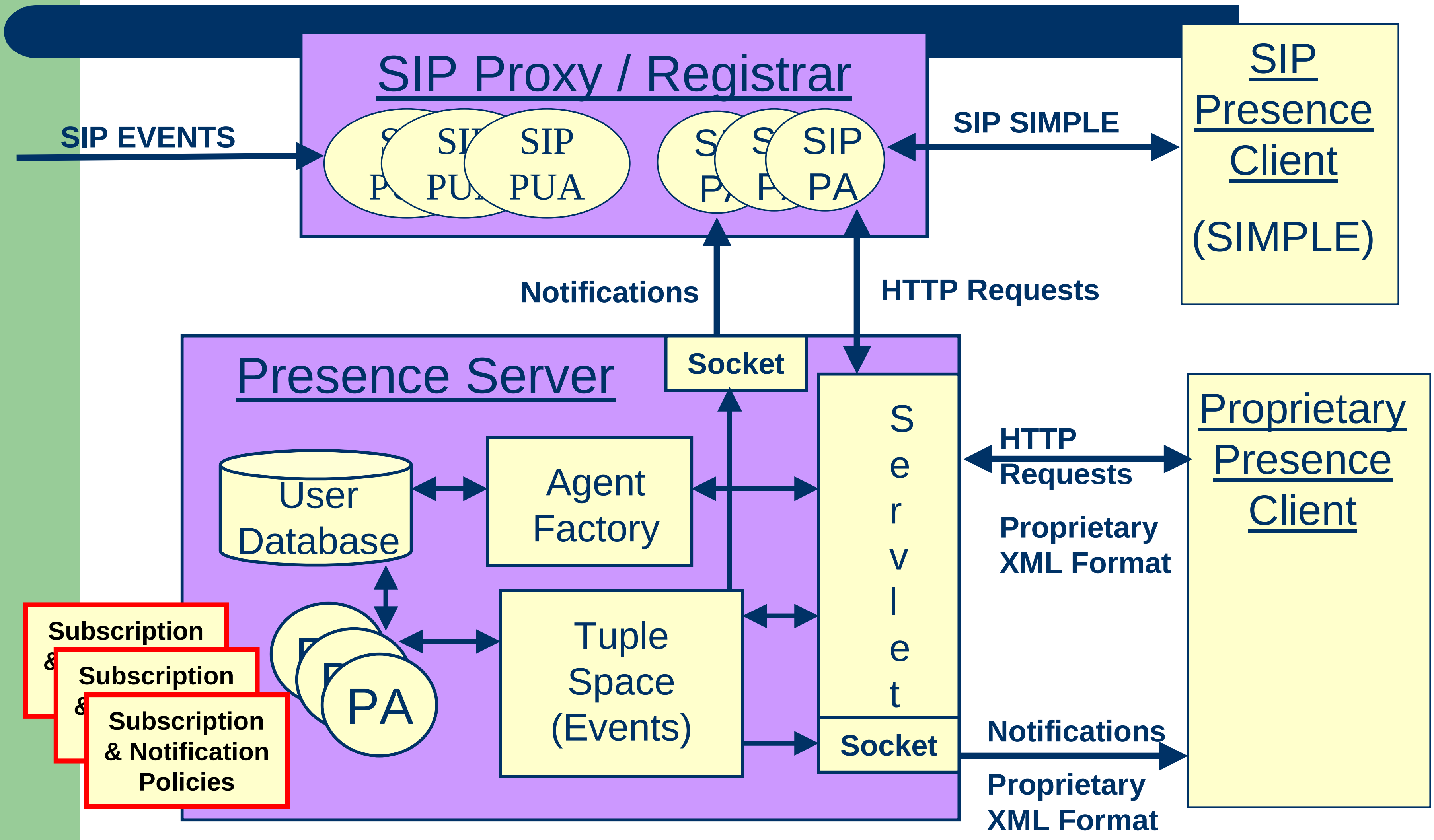
# Approaches

- Start with existing server.
  - Jabber presence framework
  - Open protocol, several implementations.
- Develop own server.
  - Can be customized to include special features.
  - For example the inclusion of presence and availability policies.
  - Support for group-based presence.

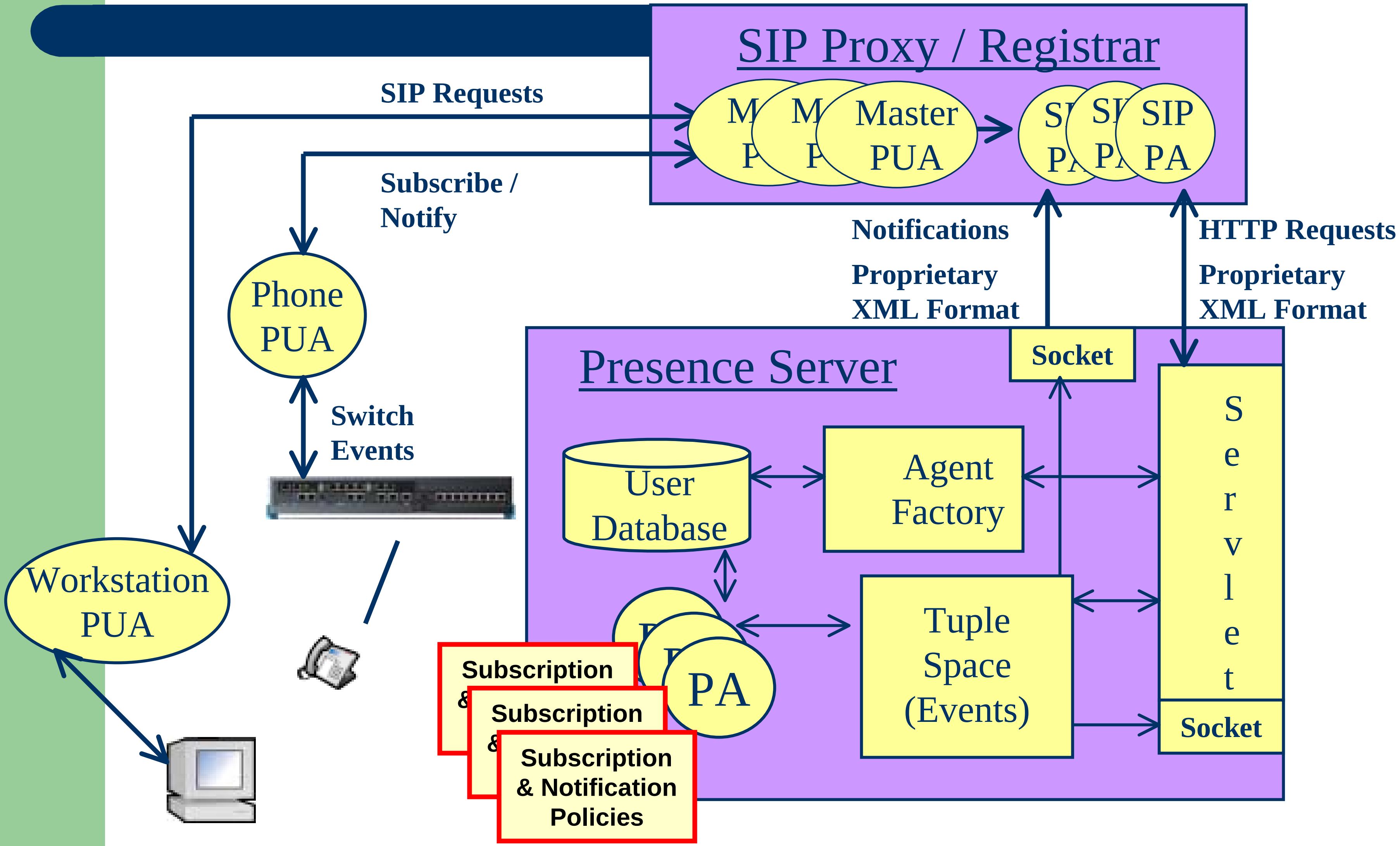
# Required Components

- Presence Protocol
  - Proprietary, SIP / SIMPLE, or Jabber
- Presence Server that supports
  - User Management: Login, Resource Specification, Activation.
  - Integration of Periphery Services: Location, Directory, etc…
  - Description of Resources or Communication services.
  - Support for Presence Agents.
  - Support for Presence User Agents to monitor user state.
- An event and notification component.
- A Presence and Availability Policy Language.
- Group Management Service.

# A Presence Framework



# Presence User Agents



# Presence Policies

# An Availability Policy Language

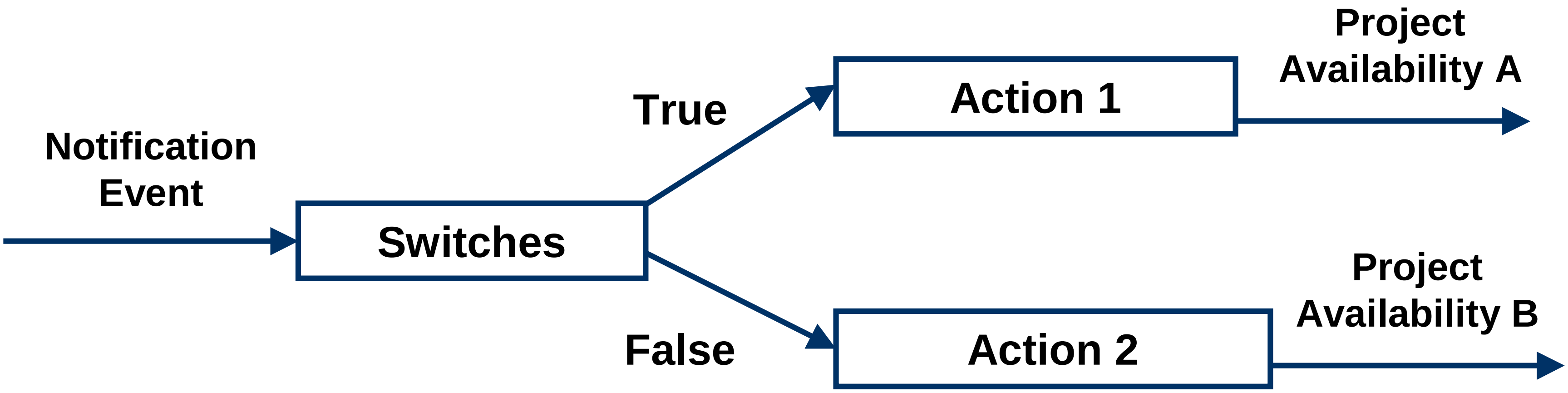
- There are 2 types of policies that manage availability:
  - Subscription policies
  - Notification policies
- Subscription Policies
  - Determine who can view your availability.
    - Fairly simple rules – if “watcher” == “Ramiro” -> OK.
    - Generally default to an “open” state, because availability is really managed by the notification policies.
- Notification Policies
  - Determine under what conditions you are available.

# Specifying Policies using CPL

- Call Processing Language (CPL)
  - A Language used to describe and control Internet Telephony Services.
  - Designed to be implemented on either Network Servers or User Agent Servers.
  - Describes Services as a Decision Graph based on XML Syntax.
  - Ideal for specifying COC-P.
- Advantages
  - Verifiable: Well-formed and executable.
  - Guaranteed to complete: No loops.
  - Safe to execute: Restrained environment.
  - Standardized Representation:
  - Readable and Producible: human or machine.
  - Protocol Independent:
  - Ease of Transport: File representation.

# Projection of Availability Policies

- Manage if people will see your availability.



# Presence and Availability Policy Language (PAPL)

- Based on CPL style structure with several “new” context-based switches.
  - Role, location, time/date, activity, relationship.

```
<papl id='10558' name='Hospital' priority='3'>
    <notification-event>
    <role-switch> <role name='Nurse'>
    <location-switch><location is='Hospital'>
    <availability value='available' status='current activity’
        location='current location'>
    <notify/>
    …
</papl>
```

# PAPL Example – Role Switch

```
<papl id='16571' name='test1'
priority='1'>
  <notification-event><role-switch>
    <role name='personal, test'>
    <availability value='available'
        status='current activity'
        location='current location'>
<notify/></availability></role>
</role-switch></notification-event>
</papl>
```

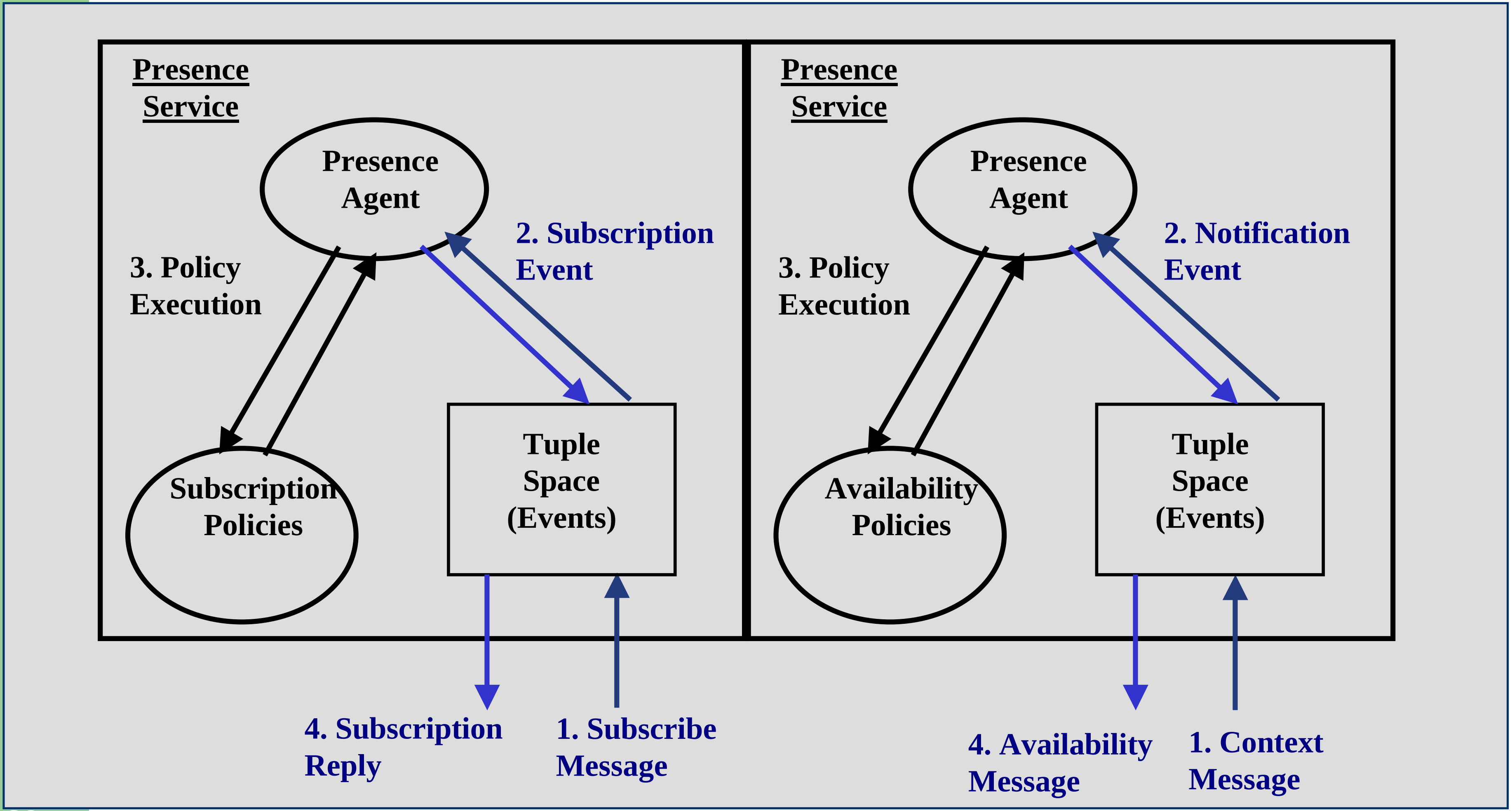
# PAPL Example – Watcher Switch

```
<papl id='19160' name='test3'
priority='2'>
  <notification-event>
    <watcher-switch>
    <watcher name=‘watcher@example.com'>
    <availability value='available'
      status='current activity'
      location='current location'>
   <notify/></availability></watcher>
   </watcher-switch>
   </notification-event>
</papl>
```

# PAPL Example – Subscription

```
<papl id='31383' name='subscribe policy'
priority='1'>
  <subscribe>
  <watcher-switch>
  <watcher name='Mary@mite.com'>
    <role-switch>
     <role name='personal, test'>
     <subscribe-reply reply='confirmed'>
  <send-reply/></subscribe-reply></role>
  </role-switch></watcher>x
  </watcher-switch></subscribe>
</papl>
```

# PAPL in Action



# Other Presence and IM Protocols

Consists Primarily of
MSN Messenger, AOL,
ICQ,

# Other IM & Presence Protocols

- Several other proprietary IM and presence protocols exist that consist of:
  - MSN Messenger (http://messenger.microsoft.com/)
  - AOL Instant Messaging (http://aim.aol.com/)
  - ICQ (http://web.icq.com/)
  - Yahoo Messenger (http://messenger.yahoo.com/)
- Will not cover details but present valuable pointers.

# MSN Messenger

- Proprietary protocol from Microsoft
  - Uses only TCP sockets. With the exception of "client to client" connections such as file transfer and voice chat, all connections are outgoing. The port number should always be 1863, but the server can specify another port.
- Some MSN un-official docs
  - http://www.hypothetic.org/docs/msn/index.php
  - http://www.venkydude.com/articles/msn.htm
  - http://mono.es.gnome.org/imsharp/tutoriales/msn/book1.html

# AOL Instant Messenger - AIM

- AIM uses the OSCAR (Open System for Communication in Realtime) protocol.
  - Used also by ICQ.
  - OSCAR Specification
    - http://iserverd.khstu.ru/oscar/
  - Java implementation of AIM
    - http://jaimlib.sourceforge.net/
  - AIM 1999 Specification:
    - http://jaimlib.sourceforge.net/doc/toc.txt
  - ICQ Protocol Site:
    - http://www.d.kth.se/~d95-mih/icq/

# Projects supporting all IM Protocols

- JBuddy
  - http://www.zionsoftware.com/products/jbuddy/index.html

# PAM Forum

- Presence & Availability Management (PAM) forum originally sponsored by Novell, Lucent & Ericsson.
  - Is now part of the Softswitch Consortium.
  - Open API specification for Presence
  - Not as open as when it was PAM Forum.
- PAM is the collection of all information about a person’s or entity’s ability and willingness to communicate
  - Presence = location + status + disposition (i.e. “in a meeting”)
  - Availability = presence + preferences + context (i.e. “customer”)
- http://www.pamforum.org

# PAM APIs

- Specifies the following capabilities and interfaces:
  - Identity management (IM):
    - manage end-user or entity names, aliases, groups and sets of attributes associated with identities.
  - Agent management (AM):
    - manage agent (that models a hardware or software device) names, communication capabilities and sets of attributes associated with agents
  - Agent assignment (AA):
    - manage the relationship between identities and the agents assigned to them.
  - Presence management (IP, AP):
    - agent and identity presence management interfaces

# PAM APIs (cont.)

- Availability management (AV):
  - Manage the preferences specified for the availability of an identity. All preference management occurs within a specific context.
- Event management (EV, EA):
  - manage the registrations and notifications of interest in events.
- Identity and Agent type management (TX):
  - defines the type schema for identities and agents specifying the attributes associated with the type
- Capability management (CX)
  - define capabilities and assign to agent type. Examples are voice, IM, SMS, WAP, etc.

# Concluding Remarks

- Presence and Awareness is creating a new form of Call Control.
  - Availability for both Call Control and Presence is one and the same. What will these policies look like?
  - There are many new features that range from collaborative applications to wireless SMS.
  - What forms of Feature Interactions will occur?
- There are many existing protocols with SIP / SIMPLE and Jabber / IMPP as principal contenders.