Instant Messaging and Presence

Kimmo Raatikainen

kimmo.raatikainen@cs.helsinki.fi

Lession Outline

- Overview
- A Model for Presence and Instant Messaging
  - Common Profile for Presence
  - Common Profile for Instant Messaging
- SIMPLE
- XMPP

Overview – 1/3

- Presence and Instant Messaging have recently emerged as a new medium of communications over the Internet.
- Presence is a means for finding, retrieving, and subscribing to changes in the presence information (e.g. "online" or "offline") of other users.
- Instant messaging is a means for sending small, simple messages that are delivered immediately to online users.

Overview – 2/3

- Applications of presence and instant messaging currently use independent, non-standard and non-interoperable protocols developed by various vendors.
- The goal of the IETF IMPP WG is to define a standard protocol so that independently developed applications of instant messaging and/or presence can interoperate across the Internet.

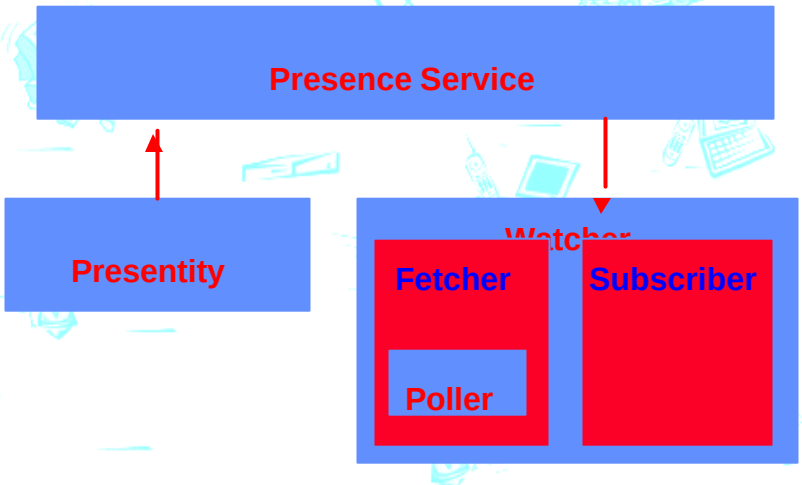
Overview – 3/3

- It is expected that Presence and Instant Messaging services will be particularly valuable to users over mobile IP wireless access devices.
- Indeed the number of devices connected to the Internet via wireless means is expected to grow substantially in the coming years.
- It is not reasonable to assume that separate protocols will be available for the wireless portions of the Internet.

Overview

- The model consists of a number of named entities that appear, in some form, in existing systems.
- The model defines two services: a ***Presence Service*** and an ***Instant Message Service***.
- The ***Presence Service*** serves to accept information, store it, and distribute it.
- The information stored is ***Presence Information***.
- The ***Instant Message Service*** serves to accept and deliver ***Instant Messages*** to ***Instant Inboxes***.

Presence Service



Presence Service – 1/3

- The ***Presence Service*** has two distinct sets of "clients".
- One set of clients, called ***Presentities***, provides ***Presence Information*** to be stored and distributed.
- The other set of clients, called ***Watchers***, receives ***Presence Information*** from the service.

Presence Service – 2/3

- There are two kinds of ***Watchers***, called ***Fetchers*** and ***Subscribers***.
- A ***Fetcher*** simply requests the current value of some ***Presentity***'s ***Presence Information*** from the ***Presence Service***.
- A ***Subscriber*** requests notification from the ***Presence Service*** of (future) changes in some ***Presentity***'s ***Presence Information***.
- A special kind of ***Fetcher*** is one that fetches information on a regular basis: a ***Poller***.

Presence Service – 3/3

- The ***Presence Service*** also has ***Watcher Information*** about ***Watchers*** and their activities in terms of fetching or subscribing to ***Presence Information***.
- The ***Presence Service*** may also distribute ***Watcher Information*** to some ***Watchers***, using the same mechanisms that are available for distributing ***Presence Information***.
- Changes to ***Presence Information*** are distributed to ***Subscribers*** via ***Notifications***.

Instant Message Service

- The ***Instant Message Service*** also has two distinct sets of "clients":
  - ***Senders*** and ***Instant Inboxes***.
- A ***Sender*** provides ***Instant Messages*** to the ***Instant Message Service*** for delivery.
- Each ***Instant Message*** is addressed to a particular ***Instant Inbox Address***, and the ***Instant Message Service*** attempts to deliver the message to a corresponding ***Instant Inbox***.

Protocols

- A ***Presence Protocol*** defines the interaction between ***Presence Service***, ***Presentities***, and ***Watchers***.
- ***Presence Information*** is carried by the ***Presence Protocol***.
- An ***Instant Message Protocol*** defines the interaction between ***Instant Message Service***, ***Senders***, and ***Instant Inboxes***.
- ***Instant Messages*** are carried by the ***Instant Message Protocol***.

Formats – 1/3

- The model defines the ***Presence Information*** to consist of an arbitrary number of elements: ***Presence Tuples***.
- Each such element consists of a ***Status*** marker, an optional ***Communication Address***, and optional ***Other Presence Markup***.

Formats – 2/3

- A ***Communication Address*** includes a ***Communication Means*** and a ***Contact Address***.
- One type of ***Communication Means***, and the only one defined by this model, is ***Instant Message Service***.
- One type of ***Contact Address***, and the only one defined by this model, is ***Instant Inbox Address***.

Formats – 3/3

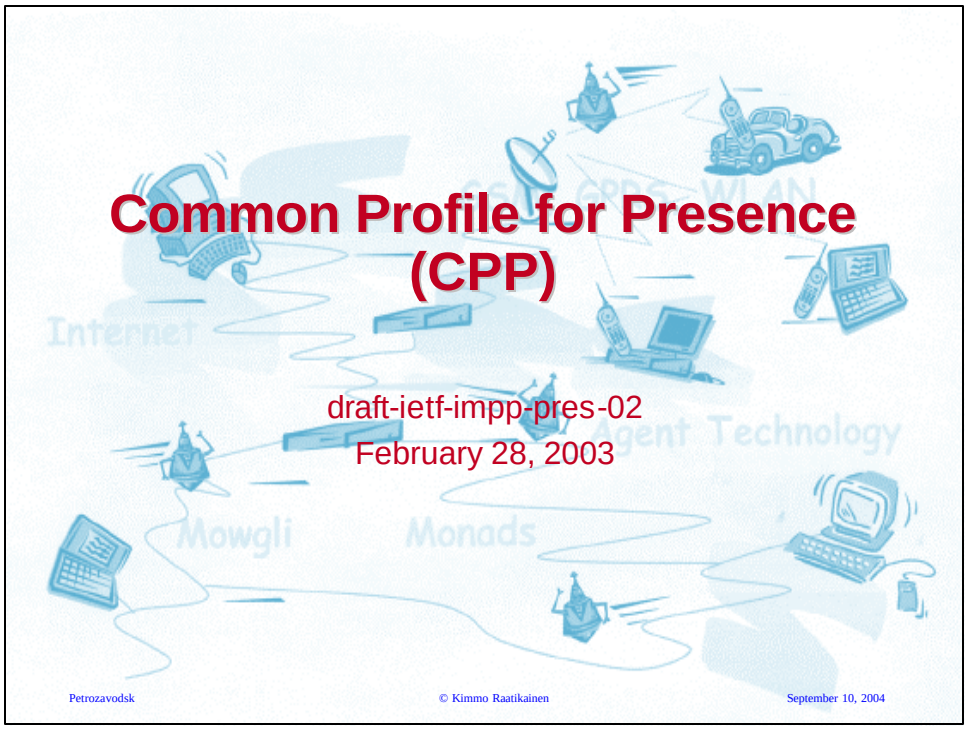
- ***Status*** is further defined by the model to have at least two states that interact with Instant Message delivery:
  - ***Open***, in which Instant Messages will be accepted, and
  - ***Closed***, in which Instant Messages will not be accepted.
- The model allows Status to include other values, which may be interpretable by programs or only by persons.
- The model also allows Status to consist of single or multiple values.

Principals and Their Agents – 1/2

- Principals are the people, groups, and/or software in the "real world" outside the system that use the system as a means of coordination and communication.
- It is entirely outside the model how the real world maps onto Principals – the system of model entities knows only that two distinct Principals are distinct, and two identical Principals are identical.

Principals and Their Agents – 2/2

- A Principal interacts with the system via one of several user agents:
  - Inbox User Agent; Sender User Agent; Presence User Agent; Watcher User Agent
- A user agent is purely coupling between a Principal and some core entity of the system
  - Instant Inbox; Sender; Presentity; Watcher



Subscribe Operation

- When an application wants to subscriber to the presence information associated with a ***Presentity***, it invokes the ***subscribe*** operation.
- The subscribe operation has the following attributes:
  - watcher, target, duration, SubscriptID and TransID.
  - The 'watcher' and 'target' identify the ***Watcher*** and ***Presentity***, respectively.

Response Operation

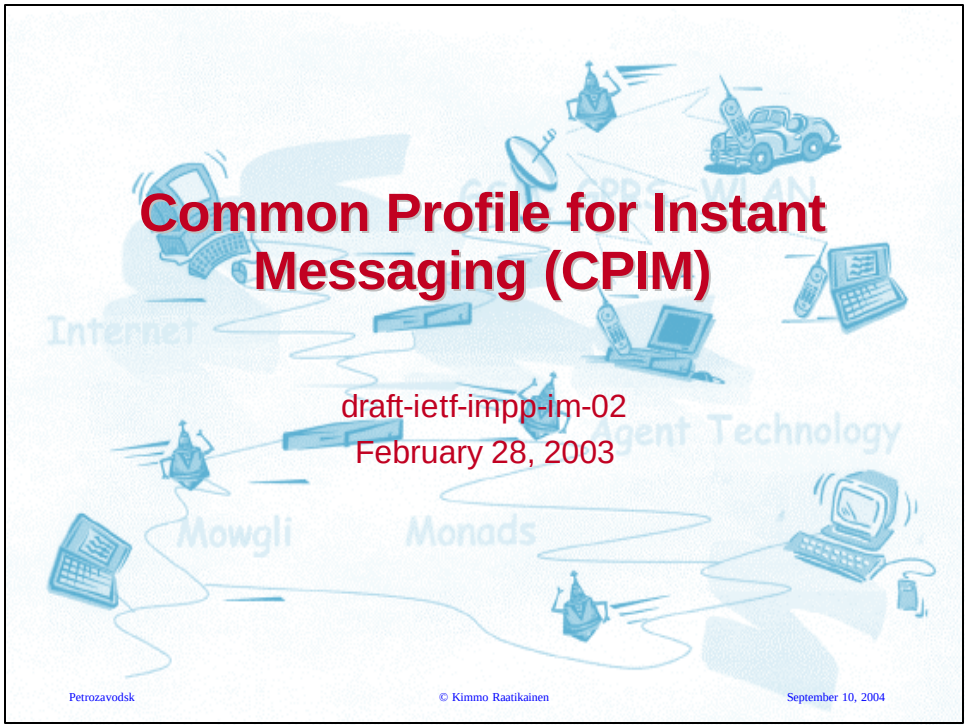
- Upon receiving a subscribe operation, the service immediately responds by invoking the ***response*** operation containing the same TransID.
- The response operation has the following attributes:
  - status, TransID, and duration.
  - 'status' indicates whether the subscribe operation has succeeded or failed.
  - The 'duration' attribute specifies the number of seconds for which the subscription will be active.

Notify Operation

- If the response operation indicates success, the service immediately invokes the ***notify*** operation to communicate the presence information to the ***Watcher***.
- The notify operation has the following attributes:
  - watcher, target, and TransID.
  - The values of 'watcher' and 'target' are identical to those given in the subscribe operation that triggered this notify operation.
  - The TransID is a unique identifier for this notification.

Unsunscribe

- The application may prematurely cancel a subscription by re-invoking the subscribe operation (as described above) with a duration of 0 and the same SubscriptID as the original subscribe operation
- Note that a notify operation will be invoked when a subscription is prematurely canceled in this fashion; this notification may be discarded by the watcher.



Common Profile for Instant Messaging (CPIM)

draft-ietf-impp-im-02
February 28, 2003

Message Operation – 1/2

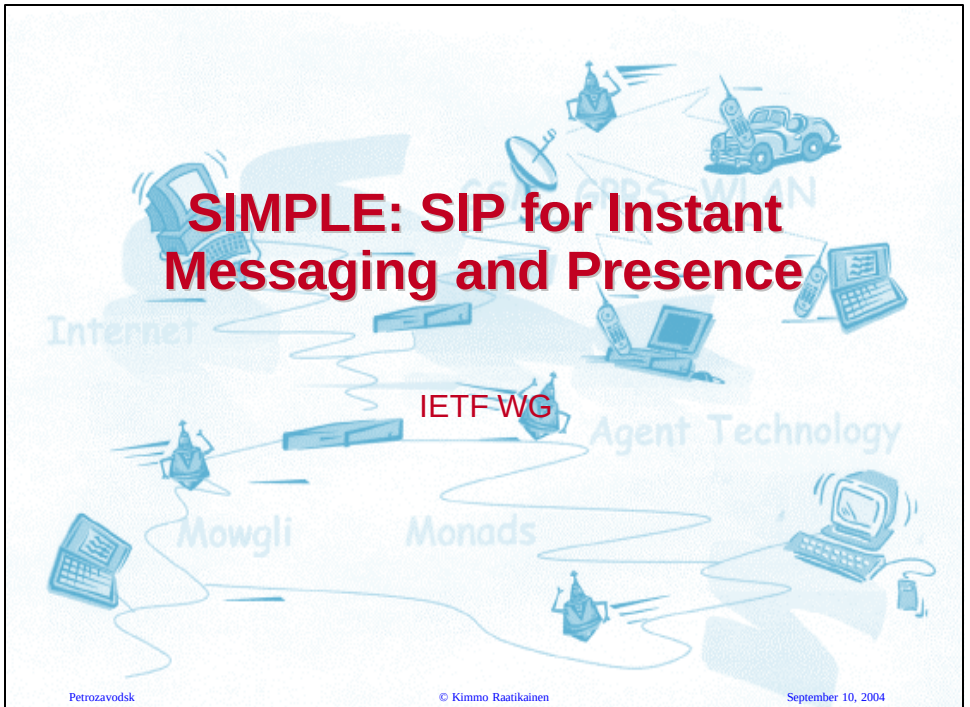
- When an application wants to send a message to an ***Instant Inbox***, it invokes the ***message*** operation,
- The message operation has the following attributes:
  - source, destination, MaxForwards and TransID.
  - 'source' and 'destination' identify the originator and recipient of an instant message, respectively, and consist of an ***Instant Inbox*** identifier.
  - The MaxForwards is a hop counter to avoid loops through gateways. Its initial value is set by the originator.

Message Operation – 2/2

  - The TransID is a unique identifier used to correlate message operations to response operations; gateways should be capable of handling TransIDs up to 40 bytes in length.
- The message operation also has some content, the instant message itself, which may be textual, or which may consist of other data.
- The specification assumes that instant messaging protocols provide reliable message delivery:
  - there are no application-layer message delivery assurance provisions in this specification.

Response Operation

- Upon receiving a message operation, the service immediately responds by invoking the ***response*** operation containing the same transaction-identifier.
- The response operation contains the following attributes:
  - TransID and status.
  - The TransID is used to correlate the response to a particular instant message.
  - Status indicates whether the delivery of the message succeeded or failed.



SIP as a Presence Protocol

- This is accomplished through a concrete instantiation of the general event notification framework defined for SIP, and as such, makes use of the ***Subscribe*** and ***Notify*** methods defined there.
- SIP is particularly well suited as a presence protocol.
- SIP location services already contain presence information, in the form of registrations.

Event Package

- This event package is based on the concept of a **presence agent**, which is a new logical entity that is capable of accepting subscriptions, storing subscription state, and generating notifications when there are changes in presence.
- This event package is also compliant with the Common Presence Profile (CPP) framework.

Presence User Agent (PUA)

- A Presence User Agent manipulates presence information for a presentity.
- Multiple PUAs per presentity are allowed.
- This means that a user can have many devices, each of which is independently generating a component of the overall presence information for a presentity.
- PUAs push data into the presence system, but are outside of it, in that they do not receive SUBSCRIBE messages, or send NOTIFY messages.

Presence Agent (PA) – 1/2

- 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.
- A presence agent must have knowledge of the presence state of a presentity.
- This means that it must have access to presence data manipulated by PUAs for the presentity.

Presence Agent (PA) – 2/2

- A PA is always addressable with a SIP URI that uniquely identifies the presentity:
  - sip:joe@example.com
- There can be multiple PAs for a particular presentity, each of which handles some subset of the total subscriptions currently active for the presentity.
- A PA is also a notifier (RFC 3265) that supports the presence event package.

Presence Server

- A presence server is a physical entity that can act as either a presence agent or as a proxy server for SUBSCRIBE requests.
- When acting as a PA, it is aware of the presence information of the presentity through some protocol means.
- When acting as a proxy, the SUBSCRIBE requests are proxied to another entity that may act as a PA.

Edge Presence Server

- An edge presence server is a presence agent that is co-located with a PUA.
- It is aware of the presence information of the presentity because it is co-located with the entity that manipulates this presence information.

XMPP Core

- The Extensible Messaging and Presence Protocol (XMPP) is an open XML protocol for near-real-time messaging, presence, and request-response services.

Server

- A server acts as an intelligent abstraction layer for XMPP communications.
- Its primary responsibilities are to manage connections from or sessions for other entities and to route appropriately-addressed XML data "stanzas" among such entities over XML streams.
- Most XMPP-compliant servers also assume responsibility for the storage of data that is used by clients.
- Compliant server implementations MUST ensure in-order processing of XML stanzas received from connected clients, servers, and services.

Client

- Most clients connect directly to a server over a TCP socket and use XMPP to take full advantage of the functionality provided by a server and any associated services.
- Multiple resources (e.g., devices or locations) MAY connect simultaneously to a server on behalf of each authorized client
  - with each resource connecting over a discrete TCP socket and differentiated by the resource identifier of a JID
- The port registered with the IANA for connections between a Jabber client and a Jabber server is 5222.

Gateway

- A gateway is a special-purpose server-side service whose primary function is to translate XMPP into the protocol(s) of another messaging system, as well as to translate the return data back into XMPP.
- Examples are gateways to SIMPLE, Internet Relay Chat (IRC), Short Message Service (SMS), SMTP, and foreign instant messaging networks such as Yahoo!, MSN, ICQ, and AIM.