Présentation du protocole SAMP

Thomas Boch [CDS]

Plan

- Présentation de SAMP
- Démonstration
- Implémenter SAMP dans votre application

SAMP

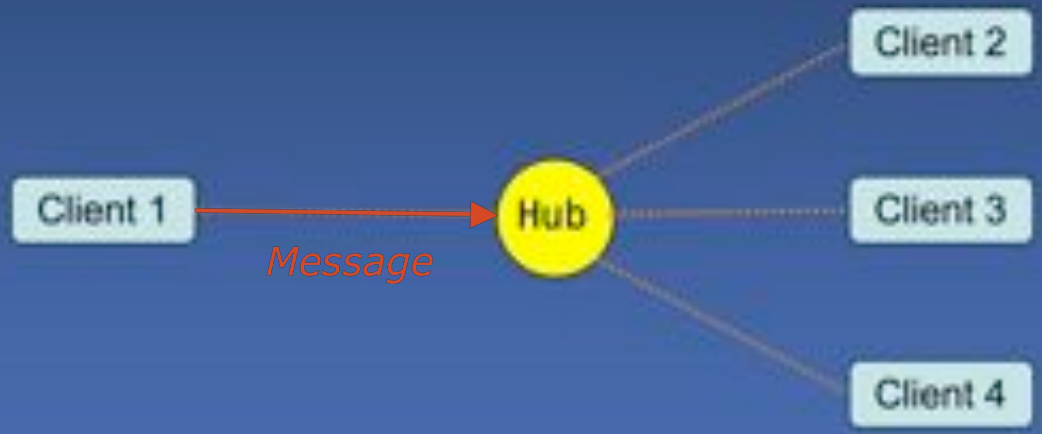
- SAMP = Simple Application Messaging Protocol
- Un protocole permettant la communication entre applications s'exécutant sur une même machine
  - Echange d'images, de tables, sélections d'objets, …
- Construit sur le succès et les principes de PLASTIC
  - Neutre par rapport aux langages
  - Multi-plateformes
  - Architecture similaire : "hub-based"
  - Adoption aisée par les développeurs d'applications

Hub-based architecture

- Hub : broker service routing messages between clients

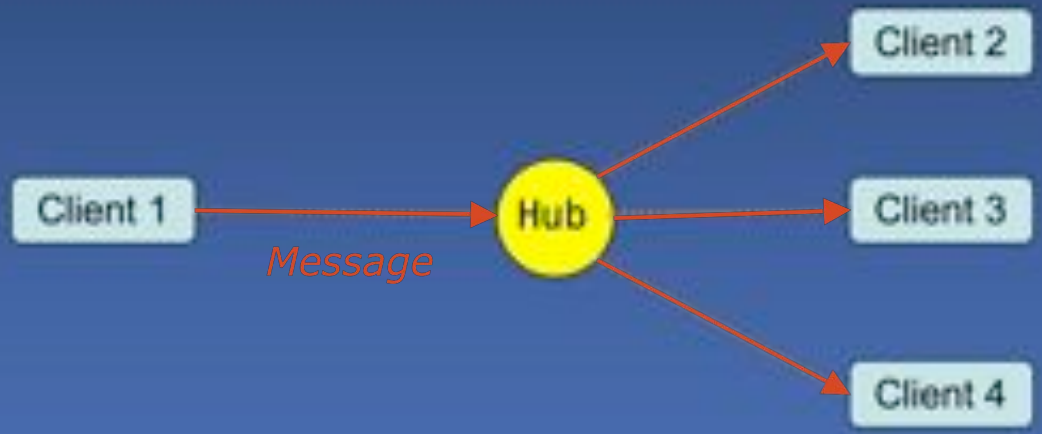
Hub-based architecture

- Hub : broker service routing messages between clients



Hub-based architecture

- Hub : broker service routing messages between clients



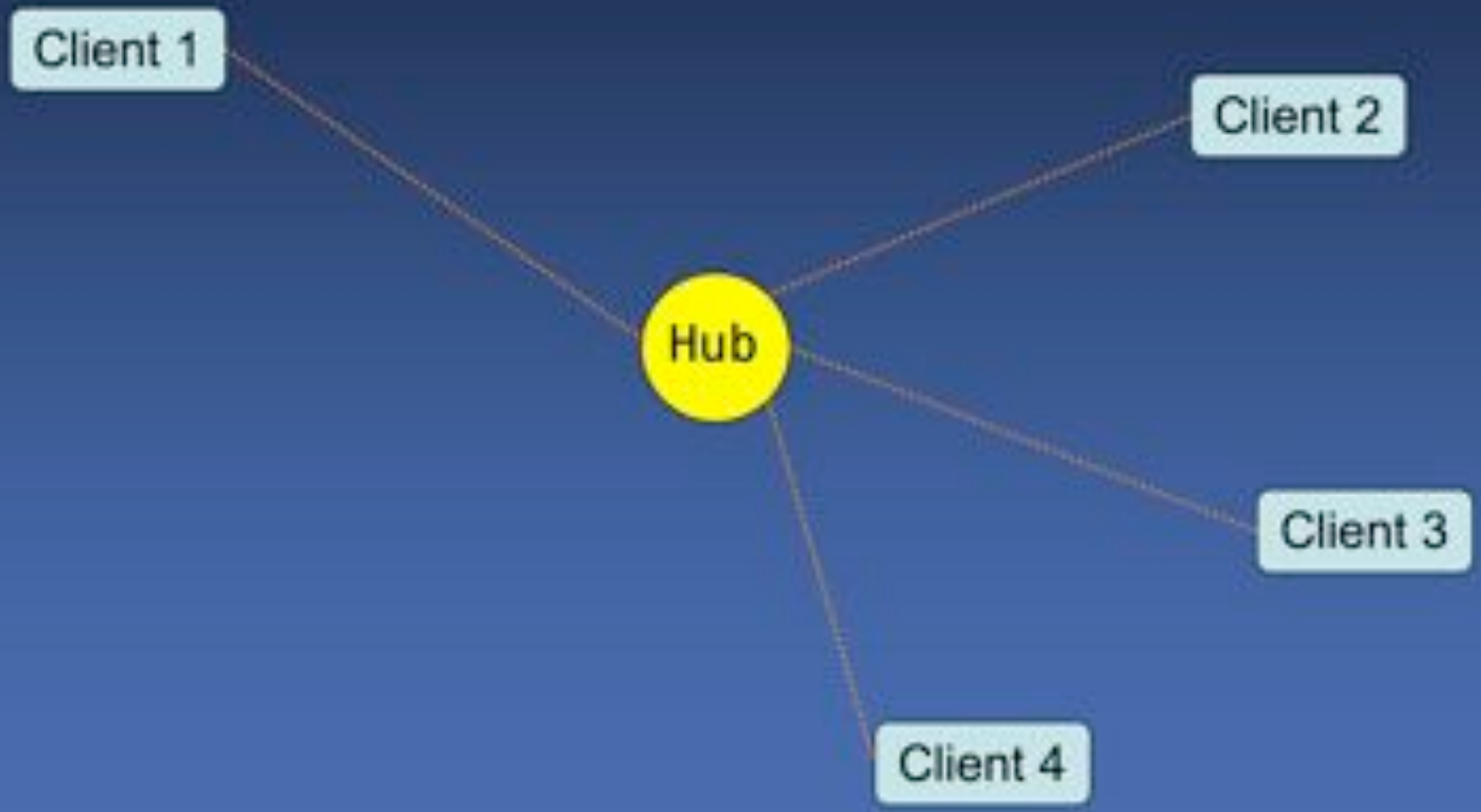
MType

- Message = MType + parameters
- MType : defines the Semantics of a Message and of its arguments and return values
  - Eg: 'display an image' --> *image.load(imname)*
  - Loose definition : exact behaviour to a given Mtype is specific of each SAMP client

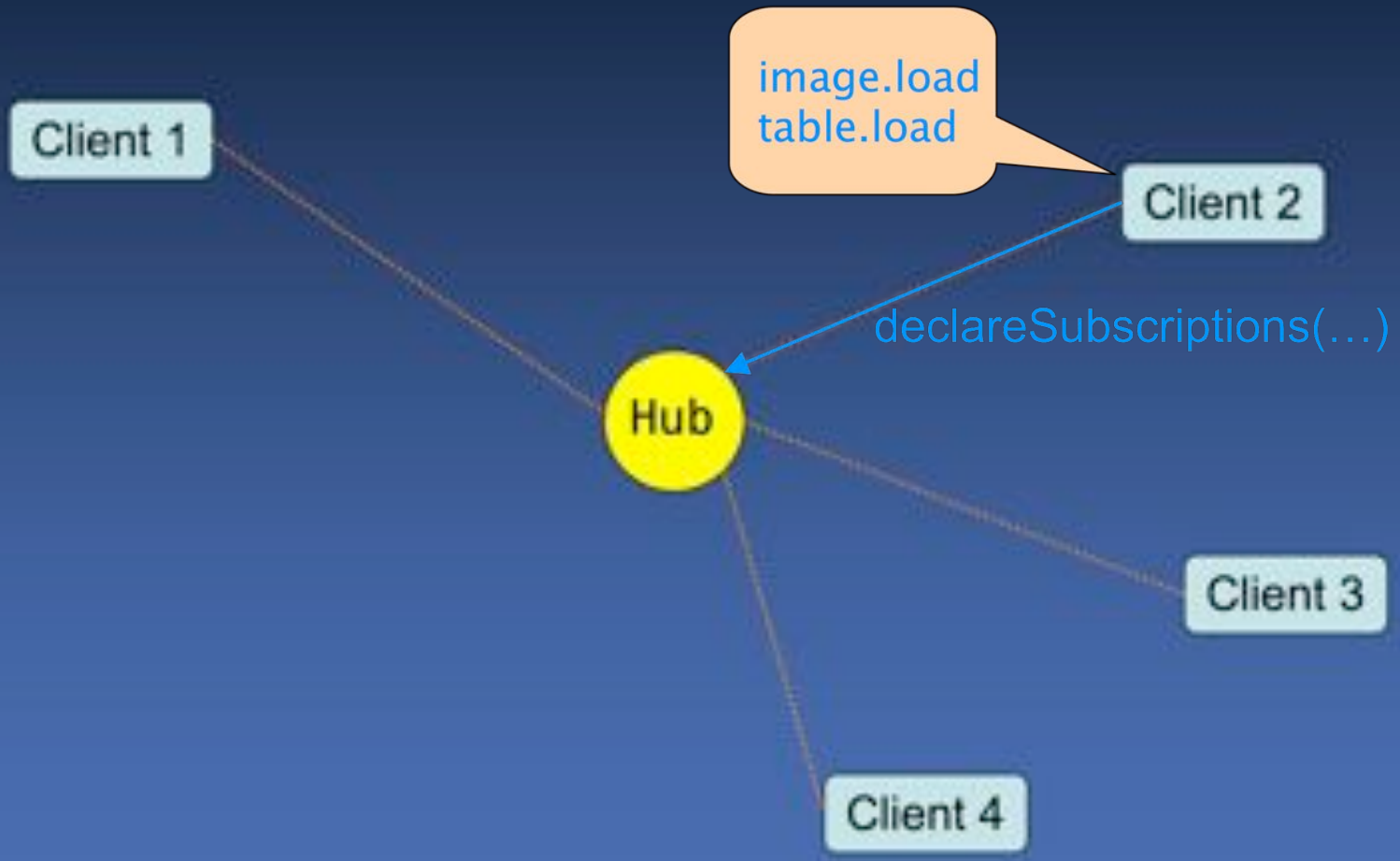
Message subscription (1/2)

- Each SAMP client subscribes to the MTypes it wants to receive
- A Message is delivered only to the clients subscribed to the corresponding MType

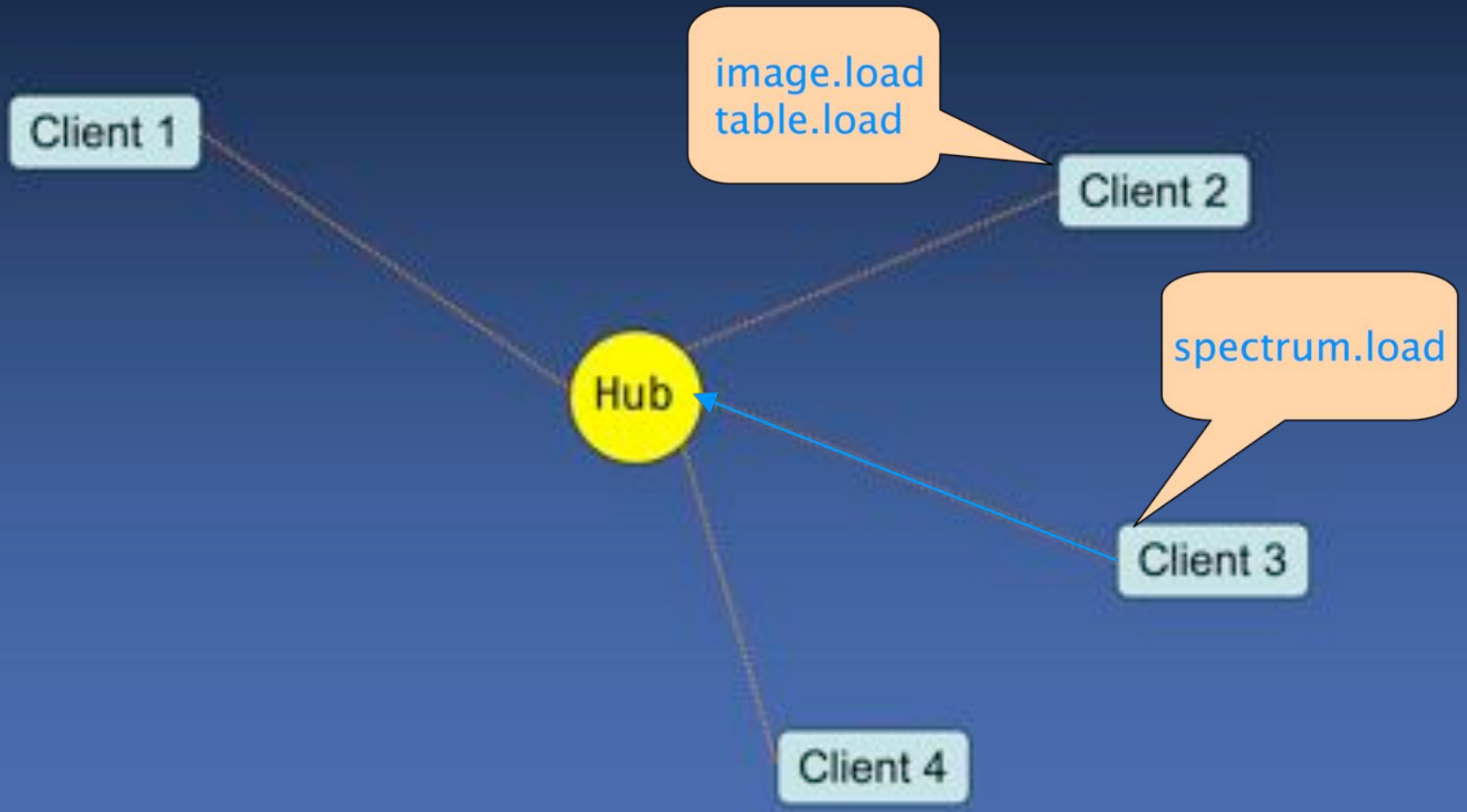
Message subscription (2/2)



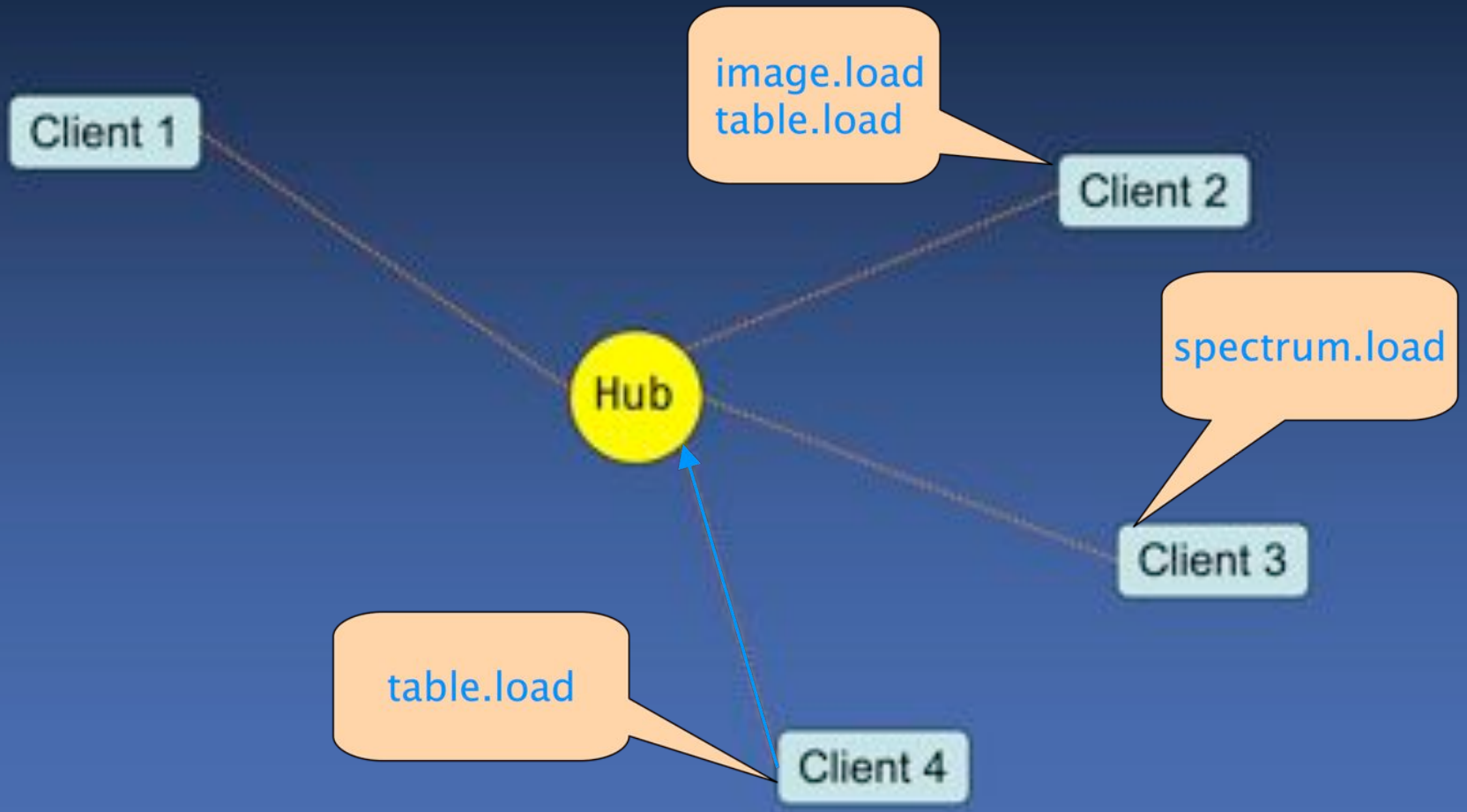
Message subscription (2/2)



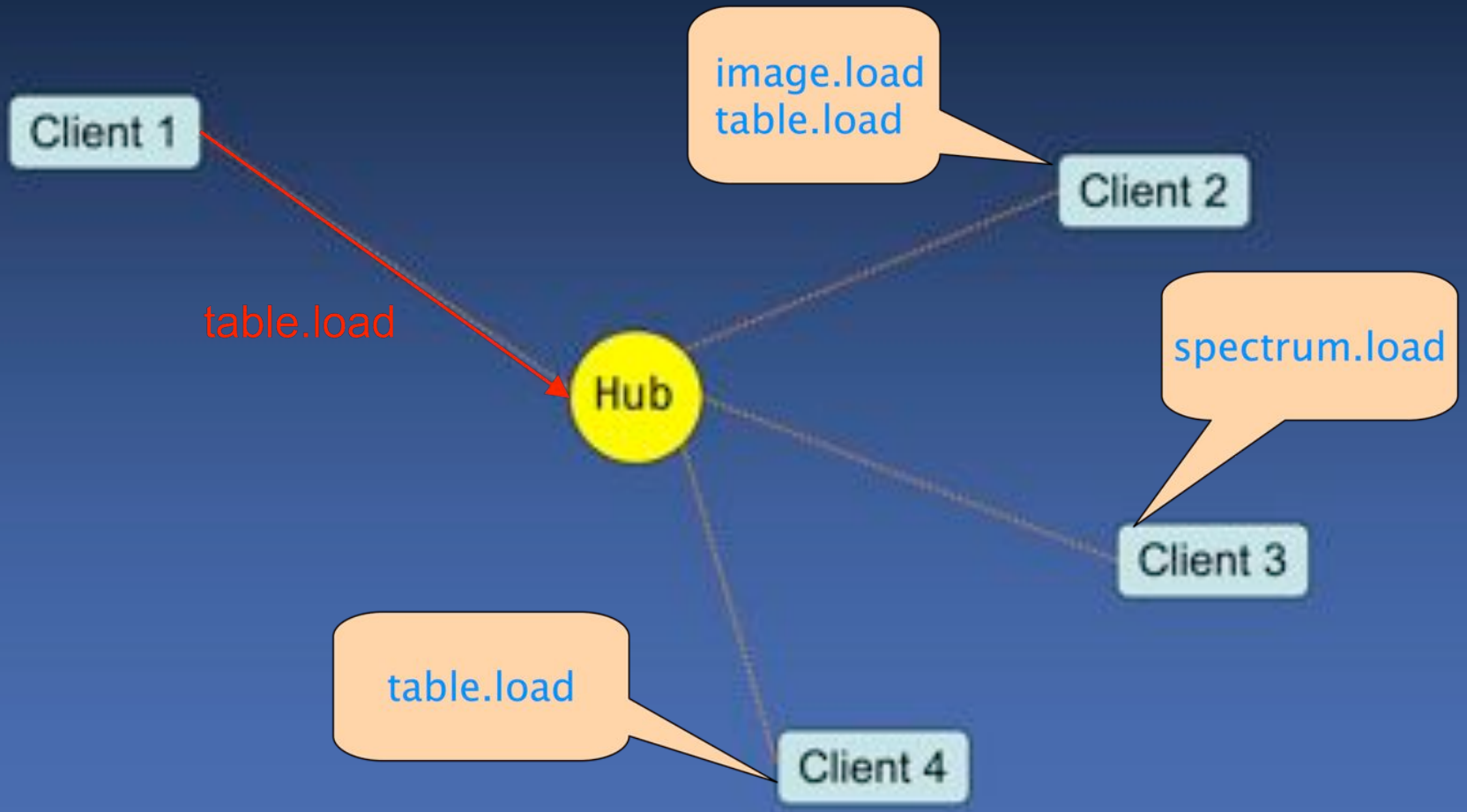
Message subscription (2/2)



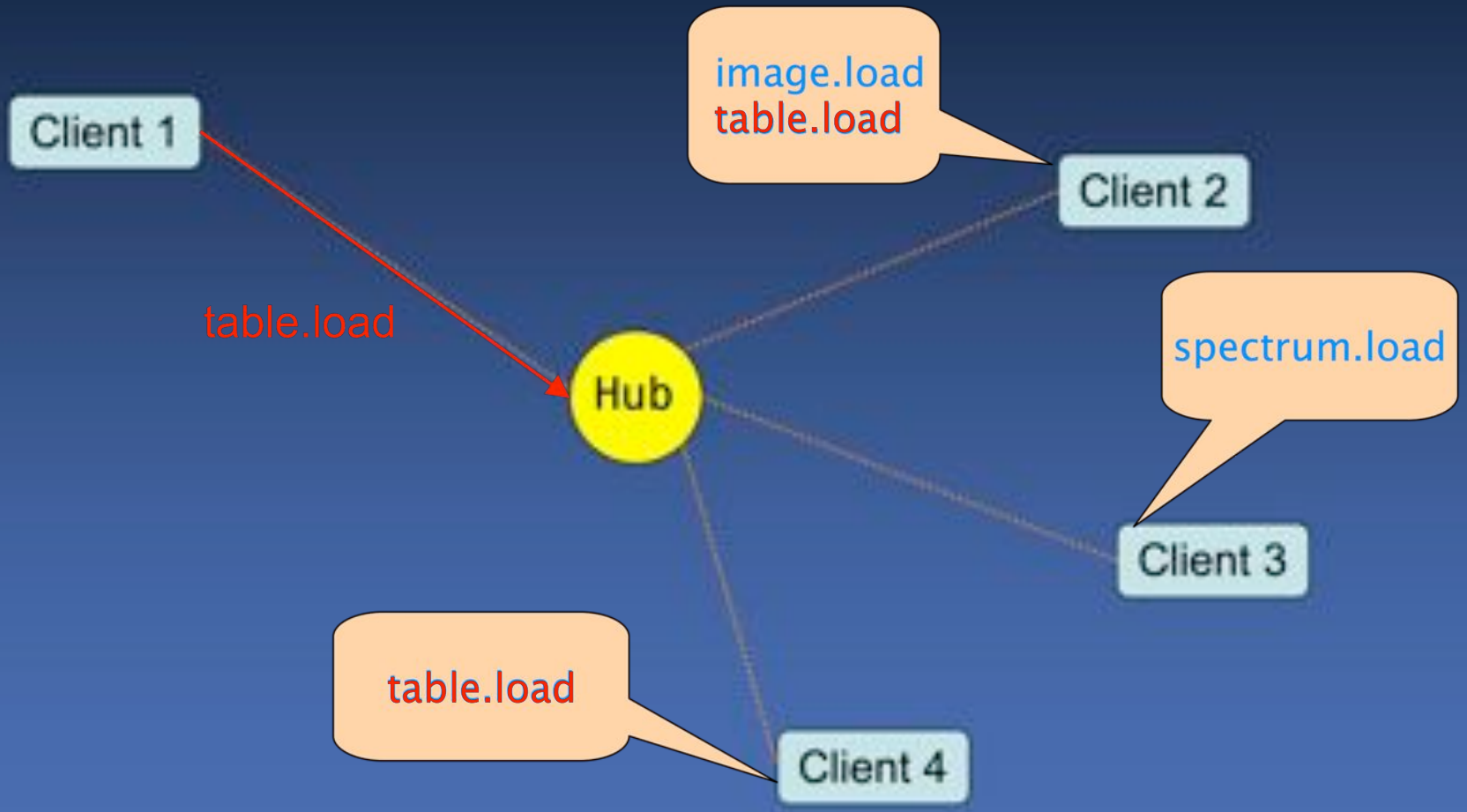
Message subscription (2/2)



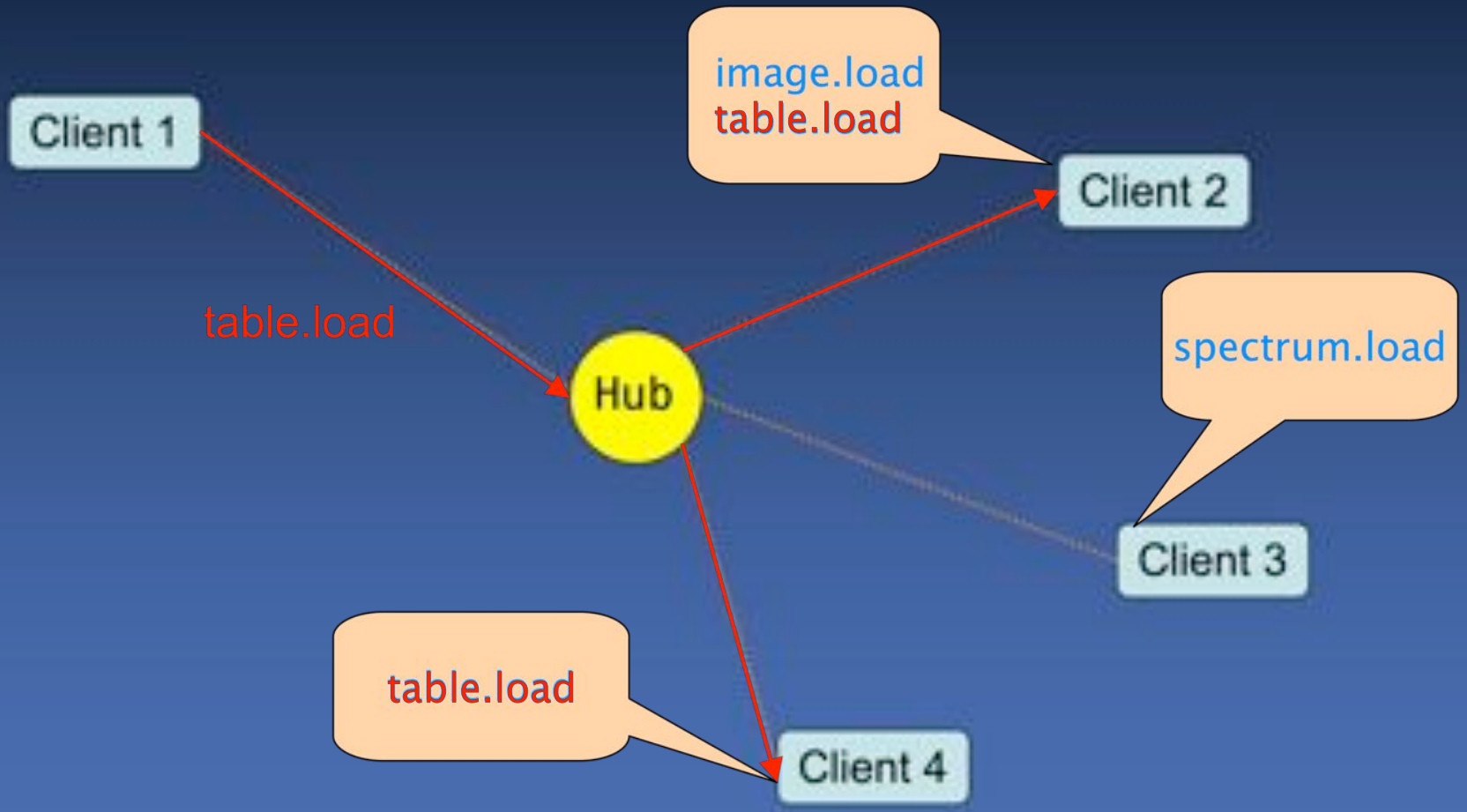
Message subscription (2/2)



Message subscription (2/2)



Message subscription (2/2)



Delivery Patterns

- Notification
- Asynchronous Call/Response
- Synchronous Call/Response

Delivery Patterns

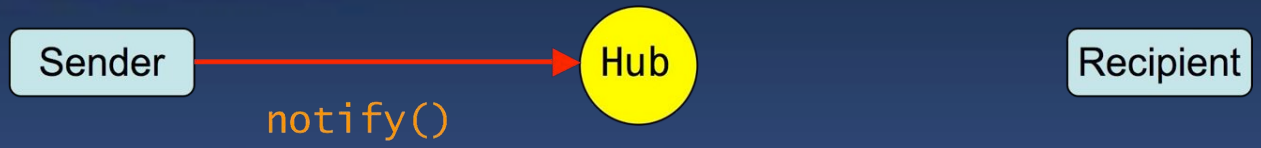
- Notification

- Asynchronous Call/Response

- Synchronous Call/Response

Delivery Patterns

- Notification

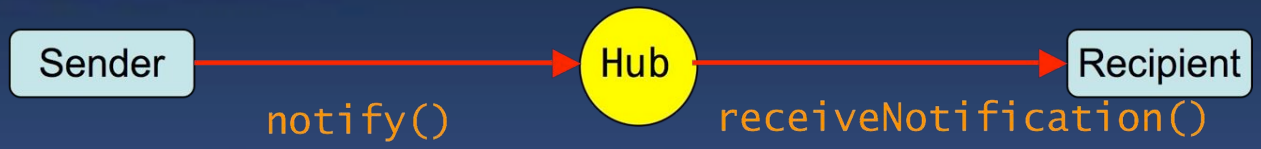


- Asynchronous Call/Response

- Synchronous Call/Response

Delivery Patterns

- Notification

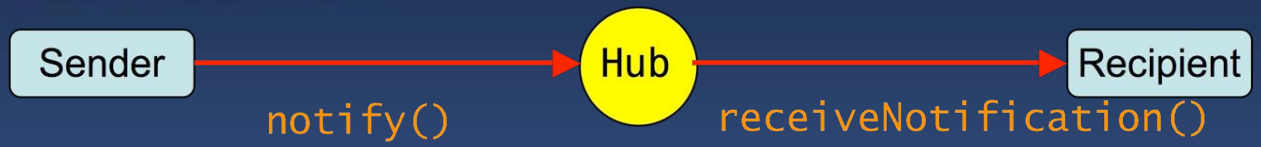


- Asynchronous Call/Response

- Synchronous Call/Response

Delivery Patterns

- Notification

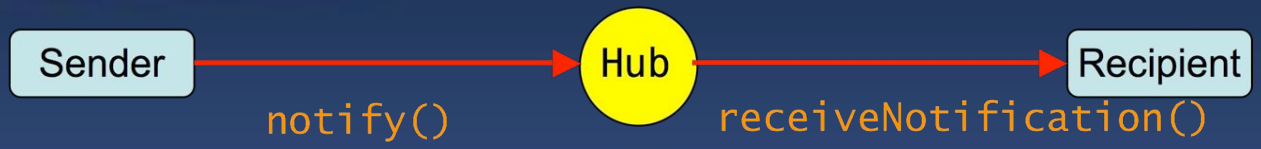


- Asynchronous Call/Response

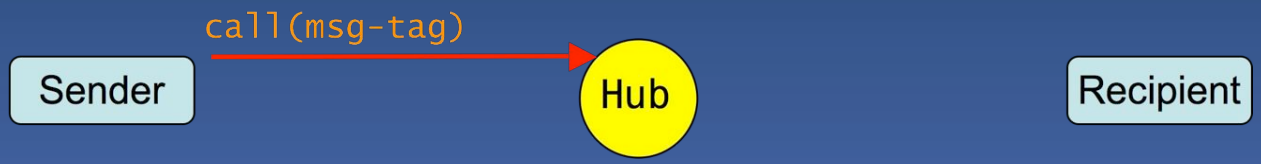
- Synchronous Call/Response

Delivery Patterns

- Notification



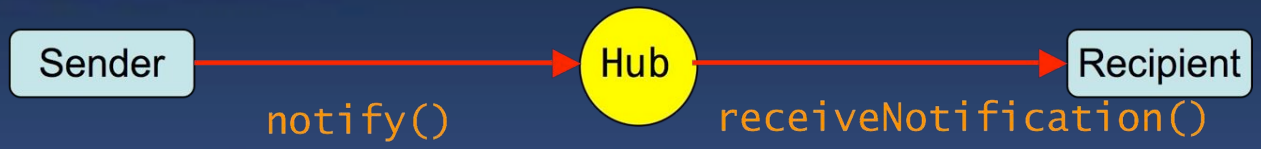
- Asynchronous Call/Response



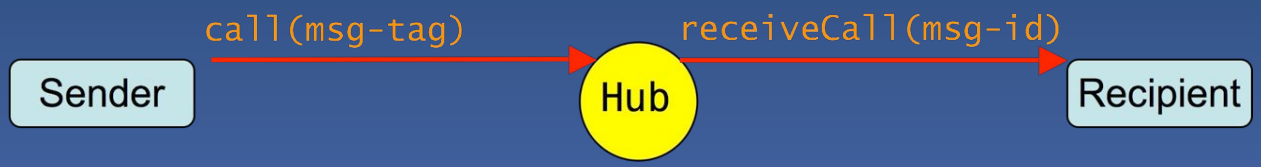
- Synchronous Call/Response

Delivery Patterns

- Notification



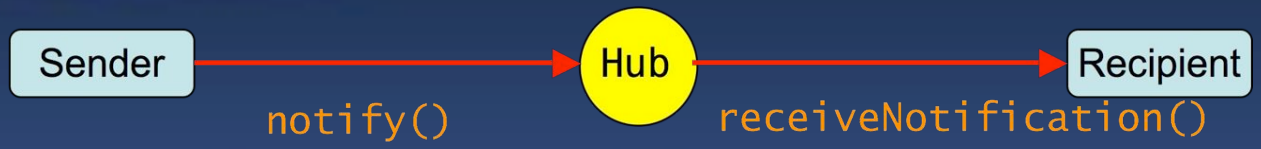
- Asynchronous Call/Response



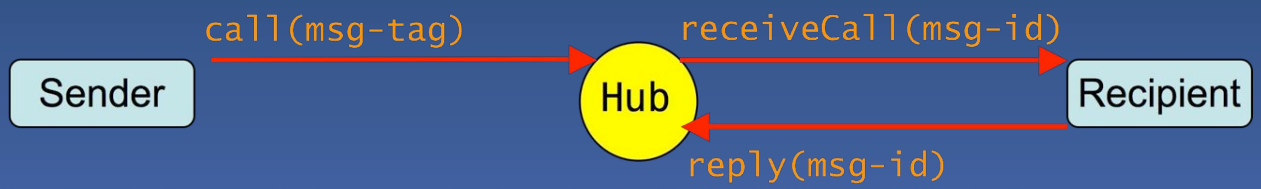
- Synchronous Call/Response

Delivery Patterns

- Notification



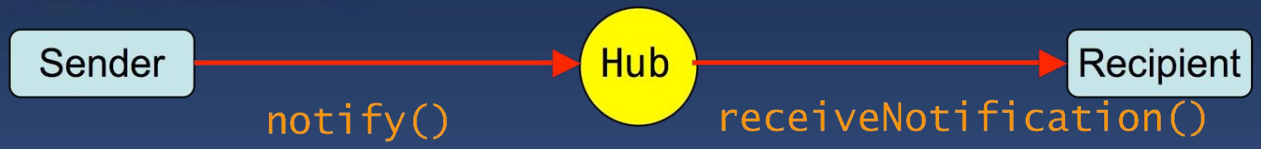
- Asynchronous Call/Response



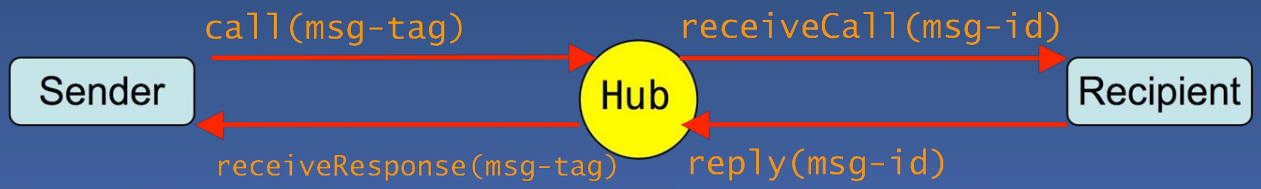
- Synchronous Call/Response

Delivery Patterns

- Notification



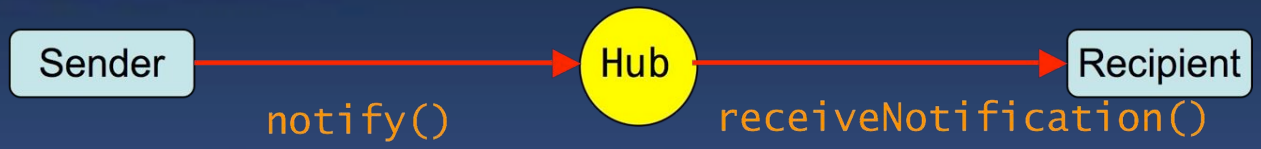
- Asynchronous Call/Response



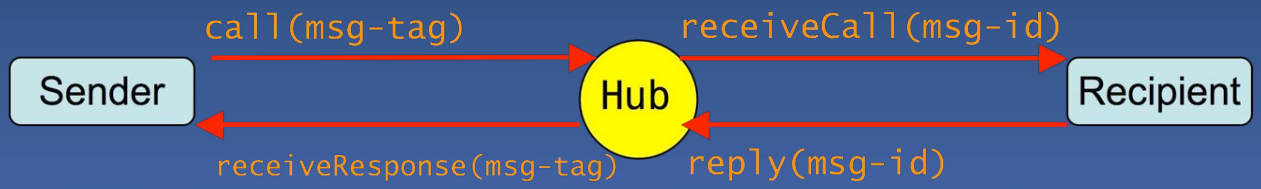
- Synchronous Call/Response

Delivery Patterns

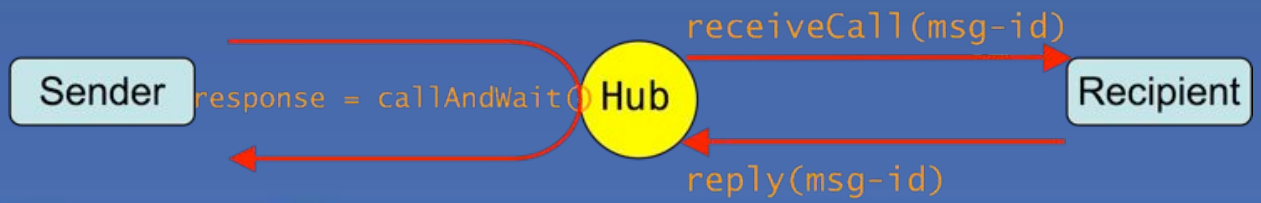
- Notification



- Asynchronous Call/Response



- Synchronous Call/Response



Abstract APIs

- Provides a high level description of the messaging protocol
  - Independent from the transport mechanism
  - Stable core, robust to changes
- Includes
  - Hub discovery mechanism
  - SAMP data types (string,list,map) + scalar type encoding convention (SAMP int, SAMP float, SAMP boolean)
  - Hub API
  - Client API

Hub API

- Operations that a hub must support
  - register(), unregister()
  - declareMetadata(map metadata), getMetadata(...)
  - declareSubscriptions(map subscriptions), getSubscriptions(client-id)
  - getRegisteredClients()
  - getSubscribedClients(list mtypes)
  - notify(...), notifyAll(map message)
  - call(...), callAll(map message)
  - response = callAndWait(map message)
  - reply(...)

Client API

- Operations which may be called (by the hub) on a callable client
  - receiveNotification(…)
  - receiveCall(…)
  - receiveResponse(…)

Profiles

- A profile is a set of rules defining how abstract interfaces (APIs) are mapped to specific network operations
- Gives rooms for other Profiles if needed in future

Standard Profile

- Relies on XML-RPC transport layer
- Mapping rules from abstract APIs to Standard profile
  - Hub discovery : lockfile in a well-known location
  - Data type mappings
  - API mapping
    - Hub/Client methods prefixed with *samp.hub/samp.client*
    - New method setXmlrpcCallback() to inform the hub of the XML-RPC endpoint of the client
    - New method ping() to ping the hub, and checks whether it is responding. Callable by non-registered applications

SAMP client sample session

- Reading lock file
- Registering with the hub
- Declaring metadata
- Sending a notification
- Declaring supported MTypes
- Declaring XML-RPC callback (make the client callable)
- Processing a message

SAMP lockfile

```
# SAMP lockfile written at 2008-05-
  16T22:26:23+0000
# Hub implementation by Alasdair Allan
  <alasdair@babilim.co.uk>
# Required keys:
samp.secret=HyPOflAQVZZxqmHla26W
samp.hub.xmlrpc.url=http://10.37.129.2:8001/
samp.profile.version=1.0
```

XML-RPC request for registering

```
<?xml version='1.0'?>
<methodCall>
   <methodName>samp.hub.register</methodName>
   <params>
      <param><value>
         <string>HyPOflAQVZZxqmHla26W</string>
      </value></param>
   </params>
</methodCall>
```

XML-RPC response from the hub

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param><value><struct>
      <member>
        <name>samp.private-key</name>
        <value><string>client-key:1a52</string></value>
      </member>
      <member>
        <name>samp.hub-id</name>
        <value><string>client-id:0</string></value>
      </member>
    </struct></value></param></params>
</methodResponse>
```

Roadmap

- En Proposed Recommendation dans les prochaines semaines
- RECommendation IVOA début 2009

Démonstrations

- Interactions entre Aladin et TOPCAT
- SkyView --> Aladin
- Contrôle d'applications SAMP depuis IDL

SAMP pour le développeur

- Retour d'expérience sur l'implémentation de SAMP dans Aladin
- Etat des lieux de l'existant
  - Hubs
  - Applications compatibles SAMP
- SAMP dans votre application
  - Toolkits clients
  - Pré-requis pour rendre votre application compatible SAMP

SAMPified Aladin

Thomas Boch - Réunion annuelle ASOV -
12-13 novembre 2008

SAMPified Aladin

Thomas Boch - Réunion annuelle ASOV -
12-13 novembre 2008

SAMPified Aladin

- Beta version of a SAMP compatible Aladin available:
  - http://cdsweb.u-strasbg.fr/~boch/SAMP/Aladin.jar
  - More robust version on Aladin official download page end of november

SAMPified Aladin

- Beta version of a SAMP compatible Aladin available:
  - http://cdsweb.u-strasbg.fr/~boch/SAMP/Aladin.jar
  - More robust version on Aladin official download page end of november
- Provides same features as PLASTICized Aladin
  - Load FITS images
  - Load VOTables
  - Select sources
  - Highlight source

SAMPified Aladin

- Beta version of a SAMP compatible Aladin available:

image.load.fits

coord.pointAt.sky

table.load.votable

table.load.fits

table.highlight.row

table.select.rowList

SAMPified Aladin

- Beta version of a SAMP compatible Aladin available:
  - http://cdsweb.u-strasbg.fr/~boch/SAMP/Aladin.jar
  - More robust version on Aladin official download page end of november
- Provides same features as PLASTICized Aladin
  - Load FITS images
  - Load VOTables
  - Select sources
  - Highlight source
- Includes JSAMP hub developed by M.Taylor

SAMP from a developer's point of view

- SAMP could be named *PLASTIC 2*
  - Evolution, not revolution
  - Concepts are similar:
    - Hub-based
    - Publish-subscribe architecture
  - Migrating from PLASTIC to SAMP is fairly easy
- SAMP is a **better** PLASTIC
  - Not Java centric
  - Separation of registration, metadata declaration, declaration of supported messages
  - Named parameters allows easier extension of existing messages (optional parameters)
  - Different delivery patterns are clearly defined

Supporting both SAMP and PLASTIC

Supporting both SAMP and PLASTIC

Supporting both SAMP and PLASTIC

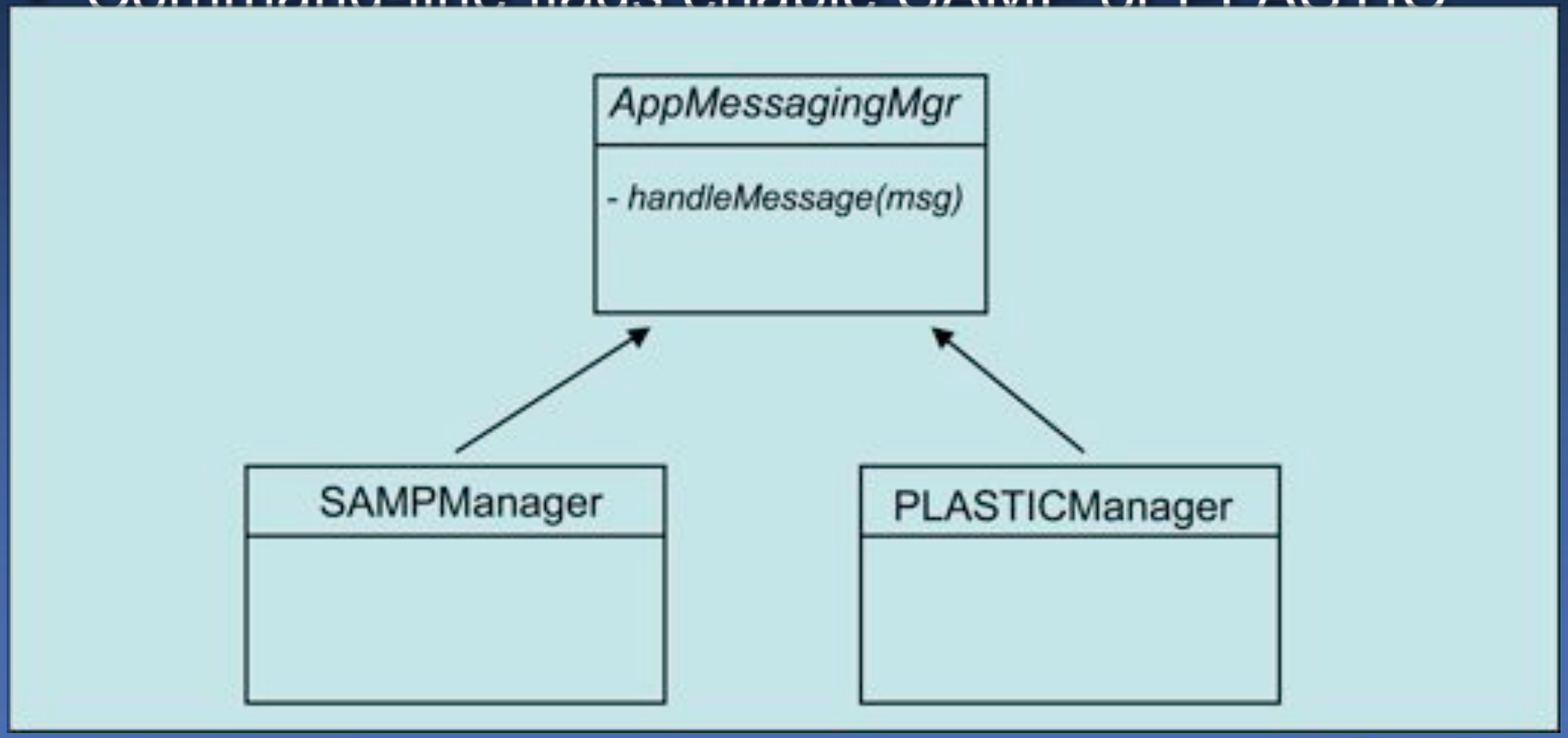
- Command-line flags enable SAMP or PLASTIC mode
  - *java -jar Aladin.jar [-samp|-plastic]*

Supporting both SAMP and PLASTIC

- Command-line flags enable SAMP or PLASTIC mode
  - *java -jar Aladin.jar [-samp|-plastic]*
- Require to refactor existing code
  - Abstract application messaging

Supporting both SAMP and PLASTIC

- Command-line flags enable SAMP or PLASTIC



Supporting both SAMP and PLASTIC

- Command-line flags enable SAMP or PLASTIC mode
  - *java -jar Aladin.jar [-samp|-plastic]*
- Require to refactor existing code
  - Abstract application messaging
  - Abstract message definition

Supporting both SAMP and PLASTIC

- Command-line flags enable SAMP or PLASTIC mode
  - *java -jar Aladin.jar [-samp|-plastic]*
- Require to refactor existing code
  - Abstract application messaging
  - Abstract message definition
- Supporting SAMP and PLASTIC in the same application session is trickier
  - Translation at super-hub level ?

Hubs SAMP disponibles

- JSAMP (Mark Taylor)
  - Développé en Java
  - http://deployer.astrogrid.org/software/jsamp/index.html
  - Licence LGPL
- SAMPy (Luigi Paioro)
  - Développé en Python
  - http://cosmos.iasf-milano.inaf.it/luigi/projects/vo/samp
  - Licence GPL
- Hub Perl (Alasdair Allan)

Applications compatibles SAMP

- TOPCAT
- Aladin
- SkyView

- A venir :
  - VOSpec
  - SPLAT-VO
  - DS9
  - Plugin Firefox SAMP
  - Google Sky, WWT ?
  - … votre application

Toolkits clients

Toolkits clients

- Java :
  - JSAMP
    - Gére la connexion au hub, la réception et l'envoi des messages XML-RPC

Toolkits clients

```
// Construct a connector
  ClientProfile profile = StandardClientProfile.getInstance();
  HubConnector conn = new HubConnector(profile)

  // Configure it with metadata about this application
  Metadata meta = new Metadata();
  meta.setName("Foo");
  meta.setDescriptionText("Application that does stuff");
  conn.declareMetadata(meta);

  // This step required even if no custom message handlers added.
  conn.declareSubscriptions(conn.computeSubscriptions());

  // Keep a look out for hubs if initial one shuts down
  conn.setAutoconnect(10);

  // Broadcast a message
  conn.getConnection().notifyAll(new Message("stuff.event.doing"));
```

Toolkits clients

- Java :
  - JSAMP
    - Gére la connexion au hub, la réception et l'envoi des messages XML-RPC
- Python :
  - SAMPy

Toolkits clients

- IDL :
  - A venir
  - Limité à l'envoi de messages dans un premier temps

Autres langages

- Pour envoyer un message via SAMP, il faut pouvoir :
  - Lire et parser un fichier local (`$HOME/.samp`)
  - Construire un document XML (XML-RPC)
  - Faire une requête HTTP POST
  - Parser un document XML
- Pour être en mesure de recevoir des messages (callable client), il faut en plus pouvoir :
  - Lancer un serveur HTTP sur un port dédié

Liens

- Document SAMP :
  http://www.ivoa.net/Documents/latest/SAMP.html
- Liste préliminaire de Mtypes :
  http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/SampMTypes
- IVOA Applications mailing-list :
  http://ivoa.net/forum/apps/
- SAMP mailing-list :
  http://ivoa.net/forum/apps-samp/