# Motivación:

El protocolo binario original es imposible de depurar al realizarse lecturas y escrituras por TCP a lo largo de todo el código.

El protocolo JiL-XML utiliza en parte el protocolo XML-RPC para definir la interacción entre JiL y Java ofreciendo las siguientes ventajas:

1) El protocolo es legible (simplifica la depuración)

2) El protocolo es fácilmente extensible.

3) Se separa completamente la implementación de los comandos de la comunicación. Esto simplifica portarlo a otros lenguajes/plataformas (JiL-c , JiL-PHP, acceso desde smartphones, JiL-matlab…… etc)

4) Permite modificar FÁCILMENTE el método de transporte (TCP, UDP, HTTP… o cualquier cosa que acabe en P)

# Mecanismo:

Cada mensaje del protocolo se codifica en XML de acuerdo a la convención XML-RPC.

Una vez extraído el comando XML se obtiene el método a ejecutar y se extraen los argumentos.

Se ejecuta el comando.

Se devuelve una respuesta que incluya el estado del comando o una **descripción textual comprensible del error que se ha producido.**

# Implementación:

Se comienza usando el protocolo XML-RPC original sobre HTTP en el puerto 8080.

# Modificaciones en JIl:

Se rescribe JiL Por completo aprovechando las ideas y el código ya existente.

# Métodos del protocolo:

## Connect()

Hay un pequeño intercambio de datos entre JIl y el cliente. En el intercambio el cliente y el servidor se identifican especificando las versiones del protocolo (para detectar posibles errores o incompatibilidades antes de continuar). Se indica también si es necesario hacer autenticación. Se incluye un identificador de sesión aleatorio.

EL CLIENTE ENVÍA:

<?xml version="1.0"?>

<methodCall>

<methodName>jil.connect</methodName>

</methodCall>

SI TODO VA BIEN EL SERVIDOR RESPONDE:

<?xml version="1.0"?>

<methodResponse>

<params>

<param>

<value>

<string>version</string>

<value>

</param>

<param>

<value>

<i4>SesionID</i4>

<value>

</param>

</params>

</methodResponse>

POSIBLES FALLOS:

[001]: “Warnig: User alredy connected” : No pasa nada, el usuario estaba conectado se queda todo igual.

[002]: “Too many users connected” : No te deja conectar, la conexión se cierra.

## Authenticate()

Se usa un usuario y una contraseña. El usuario y la contraseña no se envían en texto plano, se concatenan con el identificador y se aplica MD5, esto se envía al servidor para que lo compruebe y si concuerda con algún usuario autorizado se pasa al estado Authenticated.

## OpenVI()

Se envía la ruta del Vi que hay que abrir. Sólo funciona en estado Authenticado y si tiene éxito pasa al estado Opened. Devuelve la lista de identificadores y controles disponibles y sus tipos.

## Run()

Sólo funciona en el estado Opened. Devuelve un ACK.

## Stop()

Pasa al estado Opened y para el Vi.

## Close()

Pasa al estado Authenticated y cierra el Vi.

## Disconect()

Cierra el parser y devuelve un mensaje de despedida.

## Fallos:

**Si un método no funciona se devuelve un mensaje de error descriptivo usando la especificación del protocolo XML-RPC. Contiene un identificador de fallo (numérico) y un descriptor de fallo legible.**

SI FALLA EL SERVIDOR CONTESTA:

<?xml version="1.0"?>

<methodResponse>

<fault>

<value>

<struct>

<member>

<name>faultCode</name>

<value><int>ID de fallo</int></value>

</member>

<member>

<name>faultString</name>

<value><string>Descriptor de

fallo</string></value>

</member>

</struct>

</value>

</fault>

</methodResponse>

# Estados de JiL:

El servidor será una máquina de estados con los siguientes estados:

-Connected: Simplemente hemos empezado la conexión

-Authenticated: Ya hos hemos autenticado.

-Opened: Vi abieto y listo para usarse.