PUNTO DE ENCUENTRO VIRTUAL P2P CON ACCESO MÓVIL
ANEXO VI – "FALLOS ENCONTRADOS EN EL PROYECTO JXME Y ALTERNATIVAS DE SOLUCIÓN PLANTEADAS"

RICARDO ALBERTO CAMACHO GÓMEZ
LUIS ERNESTO GARCÍA MARTÍNEZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELEMÁTICA
LÍNEA DE ÉNFASIS EN INGENIERÍA DE SISTEMAS TELEMÁTICOS
POPAYÁN

TABLA DE CONTENIDO

LISTA DE TABLAS Y FIGURAS

ANEXO VI - FALLOS ENCONTRADOS EN EL PROYECTO JXME Y ALTERNATIVAS DE SOLUCIÓN PLANTEADAS

Descripción General

A través de la experimentación con las diferentes características que ofrece JXTA para dispositivos J2ME, se descubrieron algunos fallos en las operaciones relacionadas con la gestión de grupos y el manejo de pipes. Frente a estos problemas se implementaron soluciones para llevar a buen término toda la funcionalidad que se pensó en la fase de diseño para el Punto de Encuentro Virtual y por otro lado hacer un aporte a toda la comunidad desarrolladora que enfrenta los mismos problemas.

La descripción de los fallos encontrados se especifica a continuación, detallando tanto los resultados obtenidos con el fallo, como los resultados obtenidos con las correcciones.

1. LA COMUNICACIÓN JXME

A través de esta sección se dará una breve descripción de la estructura de los mensajes y de la comunicación de los peers J2ME con el host relay para dar al lector un ligero marco teórico que sirva para facilitar el entendimiento de los fallos (bugs) encontrados y las soluciones implementadas. Cabe anotar que resulta de gran utilidad leer este anexo en conjunto con el capítulo 4 de la monografía, en el cual se describen detalladamente todas y cada una de las operaciones en cuanto a la comunicación de los peers JXME con la red JXTA (Gestión de grupos, manejo de pipes, entre otros).

1.1 Los Mensajes JXME y la Interacción con el Host Relay

Los peers JXTA en general, sin importar si son móviles o no, se comunican entre sí a través de *mensajes*. Como los peers J2ME no se pueden comunicar directamente con otros peers en red por carecer de una IP oficial, soporte para almacenar advertisements, procesar mensajes XML como lo haría un peer JXTA convencional en Internet, entre muchas otras limitaciones, aparece un tercer actor, el *Host Relay*. Este es el encargado de entenderse con todos los peers J2ME de tal manera que puedan interactuar con la red JXTA como cualquier otra clase de peer en la red. Dada la situación de no poder invocar un servicio o hacer peticiones de comunicación directamente en un terminal móvil por

carecer de una dirección IP real y un puerto, la comunicación de los peers J2ME con el Host Relay se realiza a través de peticiones HTTP sucesivas, este procedimiento es conocido como "*polling*" o sondeo. Mediante este procedimiento, los peers J2ME envían y reciben *mensajes JXTA* para realizar las diferentes operaciones en red. Entre dichas operaciones están:

- Creación: de peers, peer groups y pipes.
- Búsqueda: de peers, peer groups y pipes.
- Ingreso a peer groups.
- Cierre de pipes.

Más adelante se dará una descripción a fondo de los fallos encontrados entre las operaciones que se acaban de enumerar. Por ahora se sigue con la descripción de los conceptos fundamentales a tener en cuenta para un mejor entendimiento de los fallos encontrados y las soluciones implementadas.

Continuando con la descripción de los mensajes JXME, estos mensajes están conformados por unidades llamadas *"Elementos"* (Ver Figura1.1).

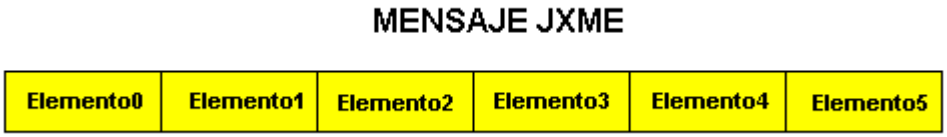

Figura 1.1 Mensaje JXME

1.1.1 Los elementos de un mensaje JXME

Los elementos son los encargados de llevar información específica del mensaje, como el tipo de mensaje, nombre del mensaje entre otros. Cada elemento a su vez está compuesto por cuatro campos de información como lo muestra la Figura 1.2.

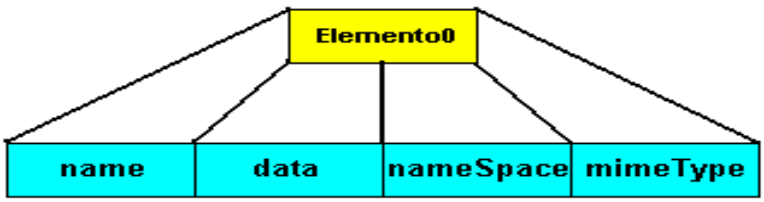

Figura 1.2 Campos de Información de un Elemento

- **Name:** Campo que describe el nombre del elemento.
- **Data:** Este parámetro contiene los datos que el elemento lleva, estos datos son transportados a través de toda la red JXTA.
- **NameSpace:** Describe el nameSpace usado por el elemento, los mensajes JXTA usan el namespace "jxta". Los mensajes en JXME usan un namespace privado. Si el nameSpace es *null*, se usa el namespace por defecto "".
- **MIME Type:** Describe el tipo de MIME de los datos que lleva el mensaje. Si se pone null en este campo null, automáticamente se asume el mime type por defecto que es *"application/octet-stream"*.

1.2 Pruebas Realizadas y Detección de Fallos

Para el análisis de pruebas es necesario tener muy claro los conceptos de mensaje, elemento y los campos de información de los elementos pues mediante el análisis de estos podemos verificar el funcionamiento correcto tanto de la aplicación que se ejecuta sobre el móvil como de la respuesta del Host Relay ante las peticiones realizadas por los peers J2ME.

1.2.1 Gestión de grupos

En la gestión de grupos se pueden invocar tres posibles operaciones: Crear Peer Group, buscar Peer Group e ingresar a un Peer Group.
A continuación se especifican los pasos necesarios para la gestión de grupos y las respuestas esperadas de cada operación.

- **Fallo: Inactividad al ingresar a un grupo diferente al grupo universal (NetPeerGroup).**

  DESCRIPCIÓN:

  Inicialmente, todas las operaciones sobre grupos se pudieron realizar con éxito, excepto el Ingreso a peer groups (*join*), este fue el primer fallo encontrado.
  Cuando un peer se conecta a la red JXTA, automáticamente entra a pertenecer al NetPeerGroup, grupo universal al que todos los peers JXTA pertenecen, aquí se podían invocar todas las operaciones disponibles desde los peers J2ME y todo

funcionaba correctamente, sin embargo al unirse a un grupo diferente, ya no había respuesta de la red JXTA frente a los mensajes enviados desde el terminal J2ME, el peer permanecía sondeando el relay para enviar o recibir mensajes sin respuesta alguna.

A continuación se describen los mensajes de petición y respuesta entregados por la aplicación en el proceso de ingreso a un grupo.

Elemento Número	**Nombre del elemento**	**Datos del elemento**
0	Request	Join
1	Id	urn:jxta:uuid-E25359ECA0524ED28C2EF85FE16C004E02
2	Arg	Null
3	requestId	2
4	EndpointDestinationAddress	http://127.0.0.1:9700/EndpointService:uuid-E25359ECA0524ED28C2EF85FE16C004E02/urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000E05/urn:jxta:uuid-E25359ECA0524ED28C2EF85FE16C004E02
5	EndpointSourceAddress	jxta://uuid-59616261646162614A787461503250334C7A8F4C38FA4CDD810C910A56CFF27603

Tabla 1. Mensaje enviado para unirse a un grupo

El mensaje representado en la tabla 1, corresponde al mensaje que el peer envía como petición para unirse a un nuevo grupo, en él se pueden detallar los siguientes elementos:

- Request: indica la clase de petición que se le está solicitando al relay, en este caso *Join,* indicando una petición de ingreso a un grupo.
- Id: especifica el identificador del grupo (groupId) al cual se desea ingresar.
- Arg: este argumento identifica el password de acceso al grupo, si el grupo es privado. En casop de que el grupo sea público, este argumento es null. Nota: hasta el momento la versión del relay que funciona para JXME no soporta la creación o ingreso a grupos con membresía, es decir, a grupos que requieran password.
- requestId: este número especifica el número de peticiones que se han hecho al host relay.

- EndpointDestinationAddress: especifica la dirección del relay al cual se hace la petición.
- EndpointSourceAddress: especifica la dirección del peer que hace la petición de ingreso al peer group.

Una vez enviado el mensaje de petición de ingreso a un peer group, el peer queda a la espera de un mensaje de respuesta exitosa. A continuación, en la tabla 2 se muestra este mensaje de respuesta.

Elemento Número	**Nombre del elemento**	**Datos del elemento**
0	Response	Success
1	requestId	1
2	EndpointDestinationAddress	http://127.0.0.1:9700/ EndpointService:jxta-NetGroup/ urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000E05/ urn:jxta:jxta-NetGroup
3	EndpointSourceAddress	jxta://uuid-59616261646162614A787461503250334C7A8F4C38FA4CDD810C910A56CFF27603

Tabla 2. Mensaje de respuesta del éxito de unirse a un grupo

De este mensaje se pueden ver los siguientes elementos:

- response: este elemento especifica el tipo de respuesta obtenido frente a la petición enviada. En este caso, se recibió "success" lo cual indica una respuesta exitosa.
- requestId: este número especifica el número de peticiones que se han hecho al host relay.
- EndpointDestinationAddress: especifica la dirección del relay al cual se hace la petición.
- EndpointSourceAddress: especifica la dirección del peer que hace la petición de ingreso al peer group.

Luego de esto, es necesario crear una nueva instancia de la clase PeerNetwork para este nuevo grupo, mediante la siguiente declaración:

PeerNetwork.createInstance(String peerName, String peerGroupId);

Esta instrucción hace que en cada nueva petición el elemento *EndpointDestinationAddress* cambie, de tal forma que ahora los mensajes van dirigidos al nuevo grupo al cual se ha unido el peer.

Este cambio es visible en la observación del elemento *EndpointDestinationAddress* que aparece en cada mensaje enviado al relay.

Los datos del elemento *EndpointDestinationAddress* son los siguientes:

- URL del Relay
- Dirección del EndpointService
- PeerGroupId

La sintaxis en código fuente es:

EndpointDestinationAddress = relayUrl + "/EndpointService:" + trimmedGID + "/" + PROXY_SERVICE_NAME + "/" + groupId;

El cambio se puede observar comparando esta dirección antes y después de unirse al grupo:

- **Antes de unirse al grupo:**

  EndpointDestinationAddress:http://127.0.0.1:9700/EndpointService:jxta-NetGroup/urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000E05/urn:jxta:jxta-NetGroup

- **Después de unirse al grupo:**

  EndpointDestinationAddress:http://127.0.0.1:9700/EndpointService:uuid-E25359ECA0524ED28C2EF85FE16C004E02/urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000E05/urn:jxta:uuid-E25359ECA0524ED28C2EF85FE16C004E02

  El segmento con letra azul: la URL del host relay con el puerto del servicio (9700).

  El segmento con letra roja: indica el groupId pero con otra cabecera (EndpointService:).

  El segmento con letra verde: indica el nombre del servicio Proxy del relay

  El segmento con letra café: indica el peerGroupId, que como puede observarse ha cambiado una vez el peer se ha unido al nuevo peerGroup.

Hasta aquí el peer se ha logrado unir al nuevo peer group. El fallo fue descubierto cuando se pretendía hacer enviar cualquier mensaje en este nuevo grupo, ya sea para crear una nueva pipe, enviar un mensaje o buscar un recurso cualquiera y no había respuesta. El peer quedaba sondeando el relay indefinidamente y nunca recibía respuesta alguna. A continuación se analiza el fallo y se documenta la solución y aporte en este sentido.

ANÁLISIS DEL FALLO Y SOLUCIÓN:

De acuerdo a los resultados obtenidos, se sospechó que no se estaba sondeando adecuadamente el host relay por parte de la aplicación móvil. Esta opción fue descartada cuando se observó en la ventana de mensajes de consola, incorporado en el entorno de desarrollo, que la aplicación J2ME estaba realizando correctamente sus funciones, los mensajes de sondeo eran correctos y se estaban enviando a intervalos adecuados.

Por otra parte, se pensó que el problema estaba directamente en el Host Relay, que éste recibía todas las peticiones pero no estaba en capacidad de enviar las respuestas por alguna razón. Por lo cual se examinó en el código fuente del relay el módulo Proxy, lo cual corresponde al archivo (net\jxta\impl\proxy\ProxyService.java). Y se encontró que el servicio Proxy del relay no estaba siendo instanciado para los grupos diferentes al NetPeerGroup, por lo cual se agregaron las siguientes líneas al método handleJoinRequest de ProxyService.java, se recompiló la clase y se agregó a la librería correspondiente (/lib/jxta.jar).

```
   private synchronized void handleJoinRequest(Requestor requestor,
                                               String grpId,String passwd) {
..
.
          try {
        // Fork this proxy into the new grp.
        p = new ProxyService();
        p.init(g, assignedID, implAdv);
        p.startApp(null);
       } catch(Exception e) {
```

```
            requestor.notifyError(e.getMessage());
            return;
        }..
        ... }
```

Finalmente, al realizar las pruebas correspondientes con las operaciones convencionales de creación, búsqueda y selección de PIPES, GROUPS y PEERS dentro del nuevo grupo, éstas funcionaban correctamente.

Por otra parte, cabe anotar que en el proceso de ingreso a un grupo, se deben seguir los siguientes pasos:

1. Búsqueda de grupos disponibles, para obtener el identificador del grupo (peerGroupId).
2. Envió de petición de ingreso al servidor relay (join)
3. Instanciar un objeto de la clase PeerNetwork usando la versión del constructor de la clase PeerNetwork que recibe dos argumentos:

```
PeerNetwork.createInstance(String peerName, String peerGroupId);
```

1.2.2. Manejo de Pipes

- **Fallo 1: No existe diferencia entre los tipos de pipes declarados por JXTA, JxtaUnicast y JxtaPropagate**

DESCRIPCIÓN:

Teóricamente JXTA ha definido tres tipos de pipes, en resumen se tienen:

- ***JxtaUniCast:*** son pipes unidireccionales para comunicación punto a punto.
- ***JxtaPropagate***: son pipes unidireccionales para comunicación punto a multipunto.
- ***Secure Unicast:*** son pipes del tipo ***JxtaUnicast*** con la adición de una capa de seguridad SSL, usada para comunicaciones seguras.

Una prueba inicial consistió en crear una pipe de tipo *JxtaUnicast,* cuyo pipeId fuera conocido por todos los demás peers, de tal manera que todos estaban sondeando dicha pipe en espera de mensajes. Se ha definido que las pipes *JxtaUnicast* están destinadas a las comunicaciones punto a punto, pero sucedió que al enviar un mensaje hacia dicha pipe, todos los peers que la estaban sondeando recibieron el

mensaje. En ese momento surgió la cuestión, entonces en qué se diferencian las pipes *JxtaUnicast* de las *JxtaPropagate?,* si tanto con las primeras como con las segundas se pueden realizar las mismas operaciones.

ANÁLISIS DEL FALLO Y SOLUCIÓN:

Este fallo fue publicado en la lista de bugs (fallos) del proyecto JXTA y se discutió al respecto con varios desarrolladores de la comunidad JXTA en el canal #jxta en el IRC concluyendo finalmente que todas las pipes funcionan de la misma manera. A pesar de todo, esto no afectó notablemente el desarrollo del proyecto.

- **Fallo 2: El método close() de la clase PeerNetwork no responde.**

DESCRIPCIÓN:

Se supone que este método sea invocado para cerrar las pipes abiertas en el proceso de lectura de datos desde un peer fuente. Pero dado que no funcionó, ni hasta el momento ha dado resultado, se tuvieron que implementar soluciones alternativas para evitar seguir leyendo sobre una pipe de lectura abierta previamente. Este inconveniente no permite que un peer, por ejemplo, inicie sesión en un grupo y luego al unirse a otro grupo deje de recibir mensajes provenientes de la pipe anterior, lo que en una aplicación como el Punto de Encuentro Virtual P2P con acceso móvil resultaría en un problema, ya que los mensajes que se generen por la aplicación de mensajería dentro de un grupo generan tráfico innecesario al peer aumentando sus costos al utilizar la aplicación.

ANÁLISIS DEL FALLO Y SOLUCIÓN:

En el diseño de la aplicación se definió que el usuario debe elegir el grupo al cual desea unirse después de haberse conectado a la red JXTA. Habiendo ingresado al grupo el peer crea dos pipes, una para recibir mensajes privados y otra para enviar y recibir mensajes públicos de todo el grupo. Una vez haya ingresado al grupo ya no puede cambiarse, si lo desea hacer debe cerrar la aplicación e ingresar al otro grupo que se desee. De esta manera se crean nuevas pipes con las cuales operar, sin estar ahora ligados a leer sobre las pipes anteriores.

Por otra parte, hay que tener en cuenta que las pipes continúan existiendo dentro de un cierto tiempo, por lo cual seguramente aún existan en un próximo ingreso, pero

ya no estén asociadas a los mismos peers. Para solucionar esto, al iniciar la sesión en un grupo, el peer recién llegado envía un mensaje de multicast al grupo informando del identificador (pipeId) de pipe privada y su nombre, de la misma forma ellos responden enviando los mismos datos y de esta manera todos los peers cuentan con información actualizada sobre los contactos en línea.