Proyecto NanoFiles

Pablo Belchi Corredor
Daniel Cascales Marcos

JOSE RUBÉN TITOS GIL

ÍNDICE

1. INTRODUCCIÓN

En este documento se especifica el diseño de los protocolos del servidor UDP. Se especifican los mensajes y estados entre cliente y directorio, a la par que se muestran autómatas que nos enseñan los diferentes estados que pueden tener. Por último veremos ejemplos reales de mensajes que se realizan entre el directorio y el cliente.

2. FORMATO DE LOS MENSAJES DE PROTOCOLO DE COMUNICACIÓN CON EL DIRECTORIO

Mensaje:

Sentido de la comunicación:
Descripción:

Ejemplo:

Mensaje: **login**

Sentido de la comunicación: Cliente-> Directorio
Descripción: Mensaje enviado por el cliente al servidor para iniciar sesión y registrar un nickname

Ejemplo:

```
operation: login\n
user: alicia\n
\n
```

Mensaje: **logout**

Sentido de la comunicación: Cliente-> Directorio
Descripción: Mensaje del cliente para cerrar sesión en el directorio.

Ejemplo:

```
operation: logout\n
user: alicia\n
\n
```

Mensaje: **ping**

Sentido de la comunicación: Cliente-> Directorio
Descripción: Mensaje que envía el cliente para verificar el estado del directorio.

Ejemplo:

```
operation: ping\n
protocolId: NanoFiles_v1\n
\n
```

Mensaje: **ping_ok**

Sentido de la comunicación: Directorio -> Cliente
Descripción: Mensaje que envía el directorio en respuesta a “ping” para confirmar su

estado.

Ejemplo:

```
operation: ping_ok\n
\n
```

Mensaje: **list_users**

Sentido de la comunicación: Cliente-> Directorio
Descripción: Lo envia el cliente para solicitar la lista de usuarios conectados

Ejemplo:

```
operation: list_users\n
\n
```

Mensaje: **user_status**

Sentido de la comunicación: Directorio-> Cliente
Descripción: Mensaje que envía el directorio para informar sobre el estado de un usuario.

Ejemplo:

```
operation: user_status\n
user: alicia\n
status: online\n
\n
```

Mensaje: **error**

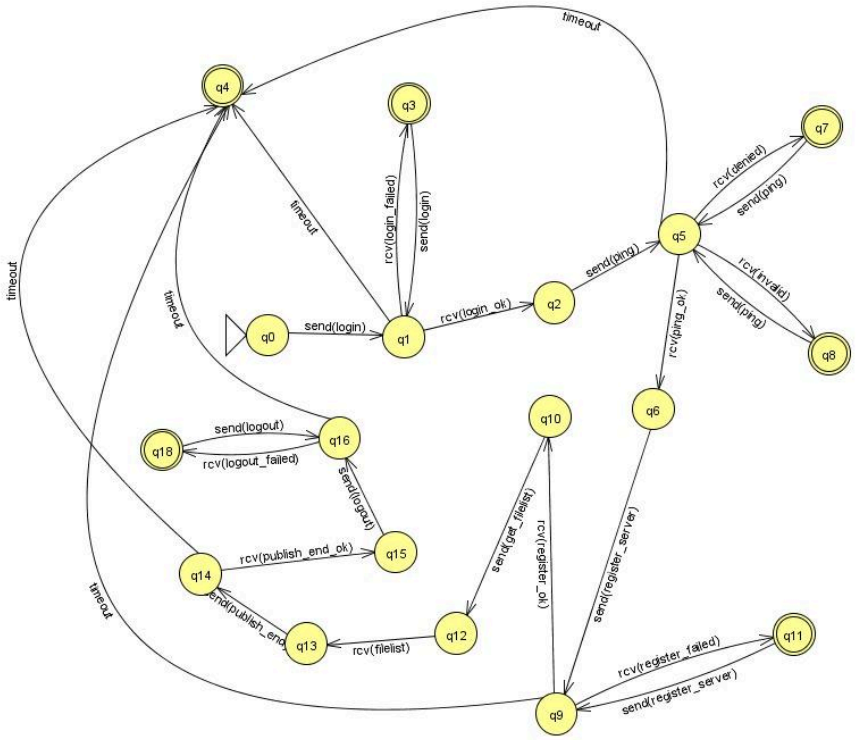
Sentido de la comunicación: Directorio-> Cliente
Descripción: Mensaje que envía el directorio para notificar que ha ocurrido un error en una operación.

Ejemplo:

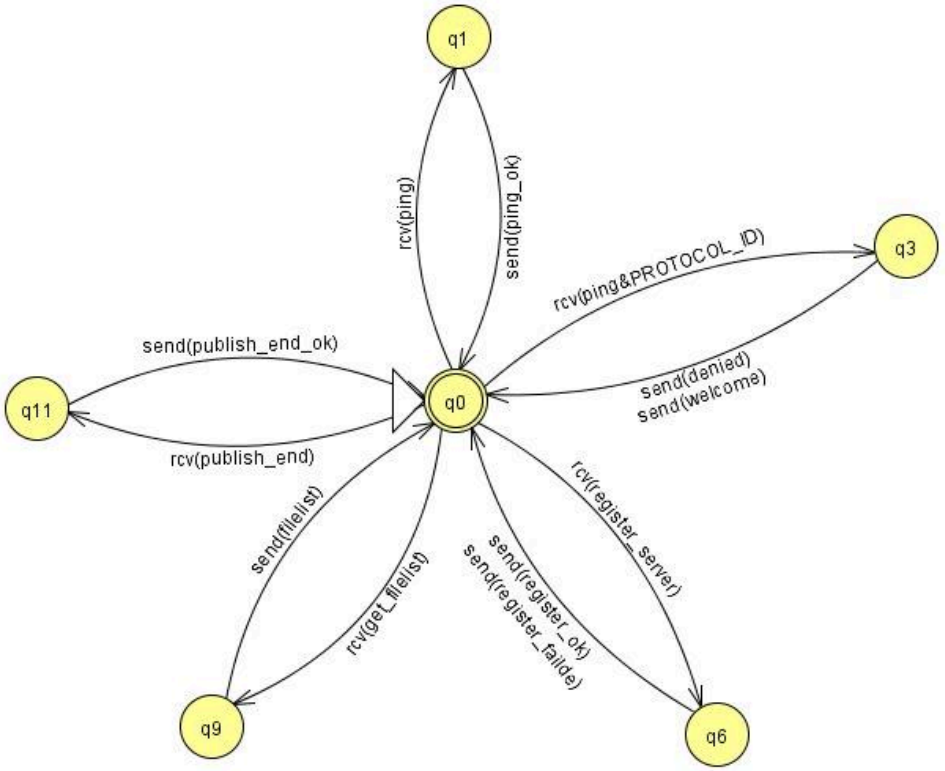
```
operation: error\n
message: Invalid operation\n
\n
```

4. AUTÓMATAS DE PROTOCOLO

Autómata rol Cliente->Directorio



Autómata rol Directorio->Cliente



5. EJEMPLO INTERCAMBIO DE MENSAJES

EJEMPLO 1:

CLIENTE: Inicia sesión

```
operation: login\n
user: alicia\n
\n
```

DIRECTORIO: Confirma inicio de sesión

```
operation: login_ok\n
\n
```

CLIENTE: Pide lista de usuarios

```
operation: list_users\n
\n
```

DIRECTORIO: Envía lista de usuarios

```
operation: user_status\n
user: alicia\n
status: online\n
user: bob\n
status: offline\n
\n
```

EJEMPLO 2:

CLIENTE: Inicia sesión

```
operation: login\n
user: alumno\n
\n
```

DIRECTORIO: Deniega.

```
operation: error\n
message: Invalid user\n
\n
```

CLIENTE: Intenta iniciar sesión de nuevo.

```
operation: login\n
user: alicia\n
\n
```

DIRECTORIO: Confirma inicio de sesión

```
operation: login_ok\n
\n
```

FORMATO DE LOS MENSAJES DE PROTOCOLO DE COMUNICACIÓN PEER TO PEER

Mensaje:

Sentido de la comunicación:
Descripción:

Ejemplo:

Mensaje: **OP_GET_CHUNK**

Sentido de la comunicación: Cliente->Servidor
Descripción: Solicita un trozo de fichero

Ejemplo:

```
operation: get_chunk\n
hash: [hash]\n
offset: [offset]\n
length: [length]\n
```

Mensaje: **OP_FILE_CHUNK**

Sentido de la comunicación: Servidor->Cliente
Descripción:El servidor envía el trozo de fichero solicitado

Ejemplo:

```
operation: file_chunk\n
data: [chunk data en bytes]\n
```

Mensaje: **OP_FILE_HASH**

Sentido de la comunicación: Cliente->Servidor
Descripción: Cliente pregunta si el servidor tiene el fichero

Ejemplo:

```
operation: file_chunk\n
data: [chunk data en bytes]\n
```

Mensaje: **OP_FILE_NOT_FOUND**

Sentido de la comunicación: Servidor->Cliente
Descripción: El servidor informa de que no tiene el fichero

Ejemplo:

```
operation: file_not_found\n
```

Mensaje: **OP_FILENAME_TO_SAVE**

Sentido de la comunicación:Cliente->Servidor
Descripción: Cliente indica el nombre con el que se debe guardar el servidor

Ejemplo:

```
operation: filename_to_save\n
filename: [nombre_archivo]\n
```

Mensaje: **OP_UPLOAD_TO_FILE**

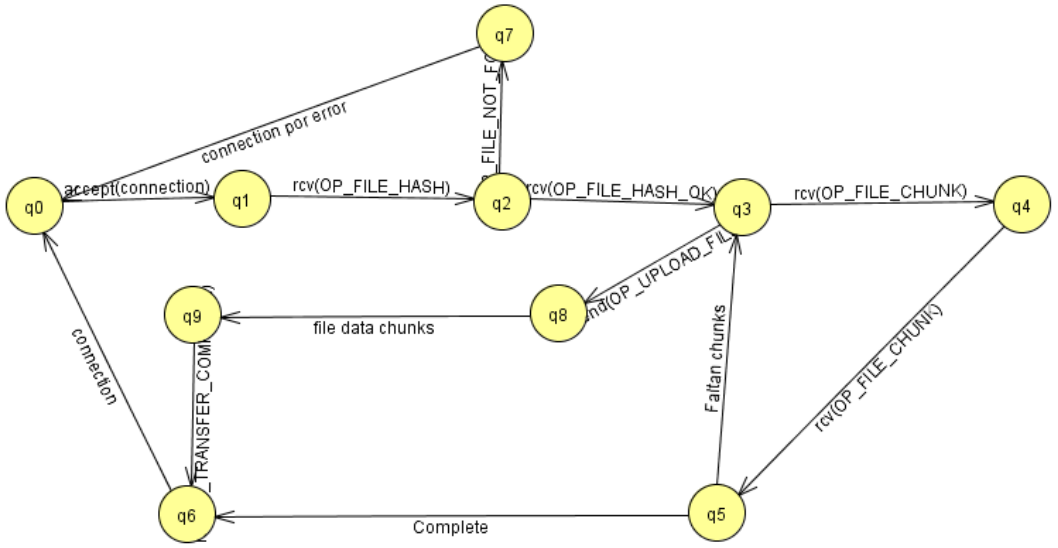
Sentido de la comunicación: Cliente-> Servidor
Descripción: Cliente envía contenido del fichero para que el servidor lo guarde

Ejemplo:

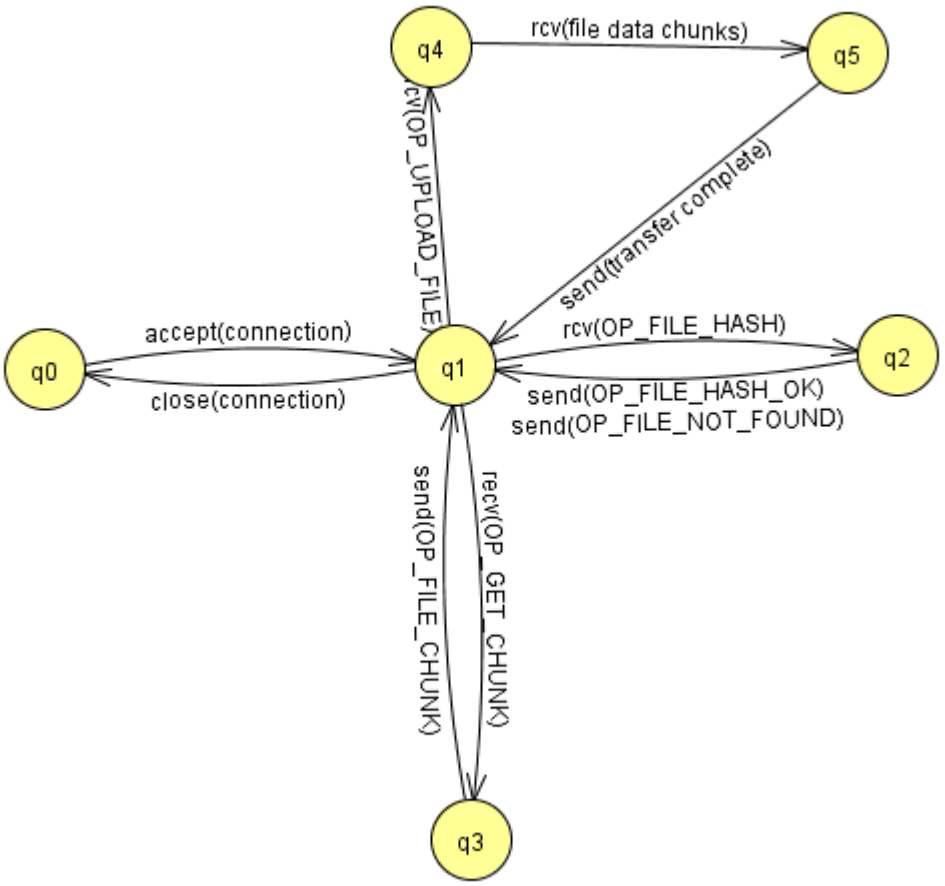
```
operation: upload_file\n
data: [contenido del archivo en bytes]\n
```

AUTÓMATAS

Autómata rol Cliente->Directorio



Autómata Peer to Peer



CONCLUSIONES

Pablo Belchí Corredor: Este proyecto me ha enseñado a trabajar con sockets UDP y TCP. Lo que me ha hecho poder comparar ambos protocolos y entender sus diferencias de forma práctica. También me ha encantado poder diseñar nuestros propios protocolos de comunicación.

Daniel Cascales Marcos: Este proyecto aunque me haya parecido de gran dificultad, he aprendido mucho y me ha ayudado a aprender mucho sobre los protocolos, como diseñarlos, implementarlos, etc.