|  |
| --- |
| **Redes de Computadoras – Curso 2013** |
| **Capa de Aplicación – Obligatorio 1**  Grupo 50  Leticia Arias – 1641989  Sebastián Piazza – 4504299  Santiago Máximo – 4647623 |
| **(2013/09/08)** |

Índice

1 Introducción 3

1.1 Objetivo del trabajo 3

1.2 Descripción general del problema 3

2 Máquinas de estado 4

2.1 client – máquina de estados 4

2.2 tracker – máquina de estados 5

3 Pseudocódigos 6

3.1 client - pseudocódigo 6

3.2 tracker - pseudocódigo 7

4 Diccionario de implementación 8

# Introducción

El presente documento incluye:

* decisiones de diseño
* máquinas de estado
* pseudocódigo
* documentación de la implementación

Junto con este documento se entrega:

* los archivos correspondientes a la implementación de la aplicación propuesta
* scripts para la construcción del ejecutable (makefile)

## Objetivo del trabajo

* aplicar los conceptos teóricos de capa de aplicación, mediante el desarrollo de una aplicación que funciona en red
* familiarizarse con el uso de las API de sockets en C/C++, fundamental en el desarrollo de aplicaciones para redes de computadoras

## Descripción general del problema

Se desea implementar un **SISTEMA P2P** para **COMPARTIR ARCHIVOS**.

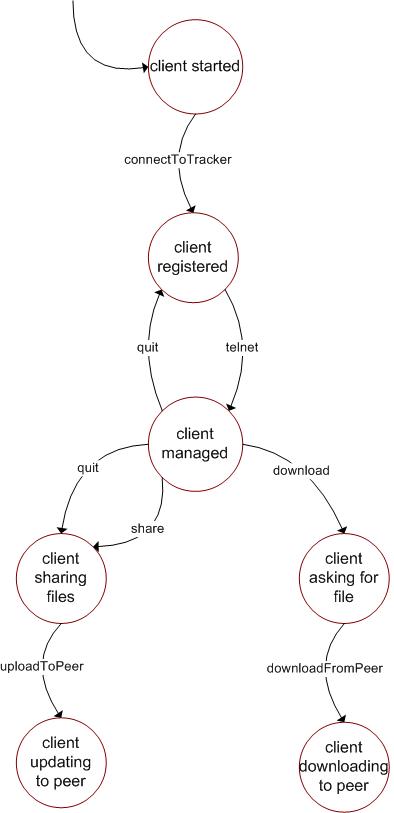
Un mismo archivo puede estar disponible en varios clients. Para identificar los archivos se usará hash MD5.

La solución utilizará un protocolo de transporte confiable, orientado a conexión (TCP).

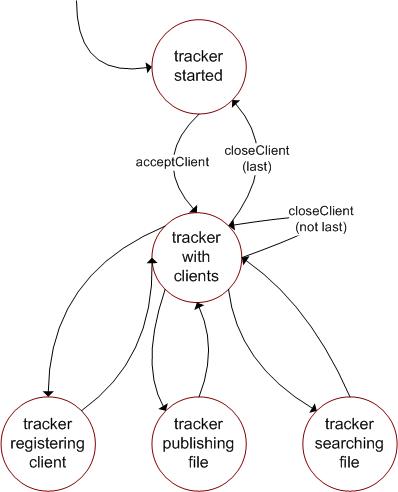
|  |  |  |
| --- | --- | --- |
| **tracker** | **registra** la ubicación de los archivos compartidos | |
| **client** | **transferencia** de **archivos** | |
| funciona **a la vez** como **uploader** y **downloader** | |
| single-thread, atiende múltiples conexiones usando la llamada poll del sistema  (atiende varias descargas a la vez) | |
| uploader | coloca archivo en lista local de archivos compartidos |
| escucha por pedidos en una dirección dada (IP-puerto) |
| registra su dirección y sus archivos compartidos en el tracker |
| downloader | consulta al tracker por el archivo y obtiene la o las direcciones de los uploaders de ese archivo |
| realiza el pedido del archivo a un uploader |
| uploader | inicia la transferencia del archivo al downloader |

# Máquinas de estado

## client – máquina de estados



## tracker – máquina de estados



# Pseudocódigos

## client - pseudocódigo

|  |  |
| --- | --- |
| **client** | |
| **parámetro** | **valor por defecto** |
| ipClient | 127.0.0.1 |
| portClient | 5555 |
| ipTracker | 127.0.0.1 |
| portTracker | 5556 |
| portConsola | 6666 |
| listaLocalFiles | |
| **startClient**(ipClient, portClient, ipTracker, portTracker, portConsola)  **share**(filename)  **download**(filename)  **downloadFromPar**(filename, ipPar, portPar)  **uploadToPar**(filename, ipPar, portPar)  **closeConsola**()  **//**closeClient() | |

**startClient**(ipClient, portClient, ipTracker, portTracker, portConsola)

tracker\_socket=***connectToTracker***(ipClient, portClient, ipTracker, portTracker)

client\_listen\_socket=***listenToPares***(ipClient, portClient)

***addToPollStructure***(client\_listen\_socket)

consola\_listen\_socket=***listenToConsola***(ipClient, portClient)

***addToPollStructure***(consola\_listen\_socket)

loop

Evento en client\_listen\_socket

Si tengo un pedido de conexión de un par

par\_socket=***acceptPar***(ipPar, portPar)

***addToPollStructure***(par\_socket)

Evento en consola\_listen\_socket

Si tengo un pedido de conexión desde la consola

consola\_socket=***acceptConsola***()

***addToPollStructure***(consola\_socket)

Evento en consola\_socket

Case share: **share**(filename)

download: **download**(filename)

show share:

show downloads:

show uploads:

quit: **closeConsola**()

Evento en par\_socket

Case download: **downloadFromPar**(filename, ipPar, portPar)

upload: **uploadToPar**(filename, ipPar, portPar)

end loop

## tracker - pseudocódigo

|  |  |
| --- | --- |
| **tracker** | |
| **Parámetro** | **Valor por defecto** |
| ipTracker | 127.0.0.1 |
| puertoTracker | 5556 |
| listaRemoteFiles | |
| **strackTracker**(ipTracker, puertoTracker)  **registerClient**(ipClient, puertoClient)  **publishFileFromClient**(filename, ipClient, puertoClient)  **searchRemoteFile**(filename)  **closeClient**(ipClient, puertoClient) | |

**startTracker**(ipTracker, portTracker)

tracker\_listen\_socket=***listenToClients***(ipClient, portClient)

***addToPollStructure***(tracker\_listen\_socket)

loop

Evento en tracker\_listen\_socket

Si tengo un pedido de conexión de un client

client\_socket=***acceptClient***(ipClient, portClient)

***addToPollStructure***(client\_socket)

Evento en client\_socket

Case register: **registerClient**(ipClient, puertoClient)

publish: **publishFileFromClient**(filename, ipClient, puertoClient)

search: **searchRemoteFile**(filename)

close: **closeClient**(ipClient, puertoClient)

end loop

# Diccionario de implementación

|  |  |  |
| --- | --- | --- |
|  | **Tipo** | **Descripción** |
| **ClientItem** | clase | modela la representación de los clientes en el tracker |
| host, ip(, lista de MD5) |
| **¿?** | ¿? | lista de clientes con clave |
| map<string,client\*>clients |
| la clave es el puerto concatenado con la ip |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

(Formato Tablas)

|  |  |
| --- | --- |
| **Título Ver. 1** | xxx |
| **…** | xxx |
| **Título Ver. 2** | xxx |

|  |  |  |
| --- | --- | --- |
| **Título Hor. 1** | **…** | **Título Hor. 2** |
| xxx | xxx | xxx |
| xxx | xxx | xxx |

|  |  |  |
| --- | --- | --- |
| **Cód.** | **Parámetros** | **Descripción** |
|  |  |  |
|  |  |  |

(Notas)

|  |  |
| --- | --- |
| Notas: |  |
| 1 | Nota 1 |
| 2 | Nota 2 |
| 3 | Nota 3 |

|  |  |
| --- | --- |
| **Def1** | |
|  | Explicación Def1 |
| **Def2** | |
|  | Explicación Def2 |

|  |  |
| --- | --- |
| **Abrev1** |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |