

Trabajo de NanoFiles

Realizado por Christian Matas Conesa

1.Introducción:

En este pdf se encuentra como está diseñado el programa nanofiles en cuanto a funcionamiento mostrando el diseño de los protocolos(formato + ejemplos y sus respectivos automatats), como han sido implementando los mensajes(cada uno con su respectivo constructor y tratado de diferente forma dependiendo a sus necesidades), seguido de las trazas de el proceso de conexión entre dos peers junto a la descarga de un archivo y por último una conclusión sobre el proyecto.

2.Diseño de protocolos:

a) Formato de los mensajes del protocolo de comunicación entre cliente y servidor de ficheros:

Formato: FilehashMessage

operation: download\n

filehash: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n

\n

Formato: FileNotFoundMessage

operation: filenotfound\n

\n

Formato: FileFoundMessage

operation: serverdata\n

fileData: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n

\n


```

        assert (operation == DirMessageOps.OPCODE_LOGIN);

        opcode = operation;
    }

```

-OneParameter:

```

public DirMessage(byte operation, String nick) {
    assert (opcode == DirMessageOps.OPCODE_REGISTER_USERNAME);

    opcode = operation;

    userName = nick;
}

```

-ThreeParameter:

```

public DirMessage(byte operation, String nick, String port, String files) {
    assert (opcode == DirMessageOps.OPCODE_SERVE_FILES);

    opcode = operation;

    userName = nick;

    this.port = port;

    this.files = files;
}

```

Mensajes TCP:

-TwoParametersBothStrings:

```

public PeerMessage(String args0, String args1) {
    switch (args0) {
        case PeerMessageOps.OP_DOWNLOAD:
            this.operation = args0;

            this.hash = args1;

            break;

        case PeerMessageOps.OP_SERVERDATA:
            this.operation = args0;

            this.file = args1;

            break;
    }
}

```

```

        default:
            break;
    }
}

```

Dependiendo del OP_CODE se creará de tipo DOWNLOAD O de SERVERDATA cada uno dependiendo de la necesidad del programa.

-TwoParametersOneStringAndOtherbyte:

```

public PeerMessage(String args0, byte[] args1) {
    this.operation = args0;
    this.fileData = args1;
}

```

4. Mejoras adicionales implementadas:

- Explorar ficheros remotos mediante nickname (browse nick) - 1 punto
- Fgstop - 0.5 puntos
- Bgserve - 1 puntos
- Queryfiles - 0.5 puntos
- Mantener actualizados ficheros y servidores - 0.5 puntos

5. Trazas de Wireshark:

-192.168.1.128 es el Peer Servidor

-10.0.2.15 es el Peer Cliente

1	0.000000000	10.0.2.15	192.168.1.128	TCP
2	0.000514766	192.168.1.128	10.0.2.15	TCP
3	0.000535766	10.0.2.15	192.168.1.128	TCP
4	3.900372015	fe80::16f2:3151:9a7...	ff02::fb	MDNS
5	3.900426645	10.0.2.15	224.0.0.251	MDNS
6	12.545589225	10.0.2.15	192.168.1.128	TCP
7	12.545894605	192.168.1.128	10.0.2.15	TCP
8	12.547141483	192.168.1.128	10.0.2.15	TCP
9	12.547147753	10.0.2.15	192.168.1.128	TCP
10	23.076256112	10.0.2.15	192.168.1.128	TCP
11	23.076421063	192.168.1.128	10.0.2.15	TCP
12	23.077625555	192.168.1.128	10.0.2.15	TCP
13	23.077631725	10.0.2.15	192.168.1.128	TCP

74	50608	→	10000	[SYN]	Seq=0	Win=64240	Len=0	MSS=1460	SACK_PERM=1	TSval=2650313861	TSrc=0	WS=128
60	10000	→	50608	[SYN, ACK]	Seq=0	Ack=1	Win=65535	Len=0	MSS=1460			
54	50608	→	10000	[ACK]	Seq=1	Ack=1	Win=64240	Len=0				
102	Standard query 0x0000 PTR _pgpkey-hkp._tcp.local, "QM" question											
82	Standard query 0x0000 PTR _pgpkey-hkp._tcp.local, "QM" question											
90	50608	→	10000	[PSH, ACK]	Seq=1	Ack=1	Win=64240	Len=36				
60	10000	→	50608	[ACK]	Seq=1	Ack=37	Win=65535	Len=0				
80	10000	→	50608	[PSH, ACK]	Seq=1	Ack=37	Win=65535	Len=26				
54	50608	→	10000	[ACK]	Seq=37	Ack=27	Win=64214	Len=0				
90	50608	→	10000	[PSH, ACK]	Seq=37	Ack=27	Win=64214	Len=36				
60	10000	→	50608	[ACK]	Seq=27	Ack=73	Win=65535	Len=0				
104	10000	→	50608	[PSH, ACK]	Seq=27	Ack=73	Win=65535	Len=50				
54	50608	→	10000	[ACK]	Seq=73	Ack=77	Win=64164	Len=0				

6. Cambios respecto a la primera entrega

- En la terminal en la que se ejecuta el servidor se muestran las peticiones de los usuarios.
- Implementado fgstop.
- Implementado bgserve.
- Implementado queryfiles.
- Mantener actualizados ficheros y servidores implementado.
- Error al descargar ficheros de gran tamaño solucionado.

7. Conclusión:

Como conclusión este proyecto consideramos que ha sido uno de los más útiles que hemos recibido a día de hoy, debido a que su utilidad es bastante amplia y necesaria en el mundo de la programación. La parte a realizar ha sido bastante difícil de ejecutar debido a que, al ser escrito la mayor parte del código por otra persona, tuvimos que dedicar bastante tiempo a su comprensión lo cual es lógico encontrarse en un futuro ante un proyecto escrito por otras personas, por lo que lo vemos muy buena idea para ir adaptándonos al mundo laboral.