

The LiFT (Lightweight File Transfer) P2P Network

MANUEL ALEJANDRO GARCÍA TOVAR
VÍCTOR HUGO ISRAEL SEGUNDO OSORIO

ITESM CAMPUS GUADALAJARA
MAESTRÍA EN CIENCIAS COMPUTACIONALES

24 DE NOVIEMBRE DE 2017



Motivación y justificación

Motivación y justificación



- Estudiar la arquitectura de sistemas tipo **Napster** y **GNUnet**.
- Mejorar los problemas de **escalabilidad** conocidos de GNUnet.
- Mejorar los problemas de **disponibilidad** conocidos de Napster.
- Agregar soporte para compartir archivos con **sencillez** por medio de ligas (UFLs)
- Transferir archivos de forma rápida y segura.



Descripción del proyecto

Descripción del proyecto

- LiFT es un sistema P2P que permite a clientes exponer archivos desde su computadora **en una red local** y compartirlos de forma sencilla a través de enlaces (UFLs).
- Los UFLs ofrecen anonimidad dentro de la red LiFT. Solo el servidor conoce el cliente asociado al enlace UFL.
- Los enlaces UFL pueden ser compartidos como texto plano y utilizados desde un cliente LiFT para poder descargar el archivo de forma sencilla y rápida.

Responsabilidades de cada módulo

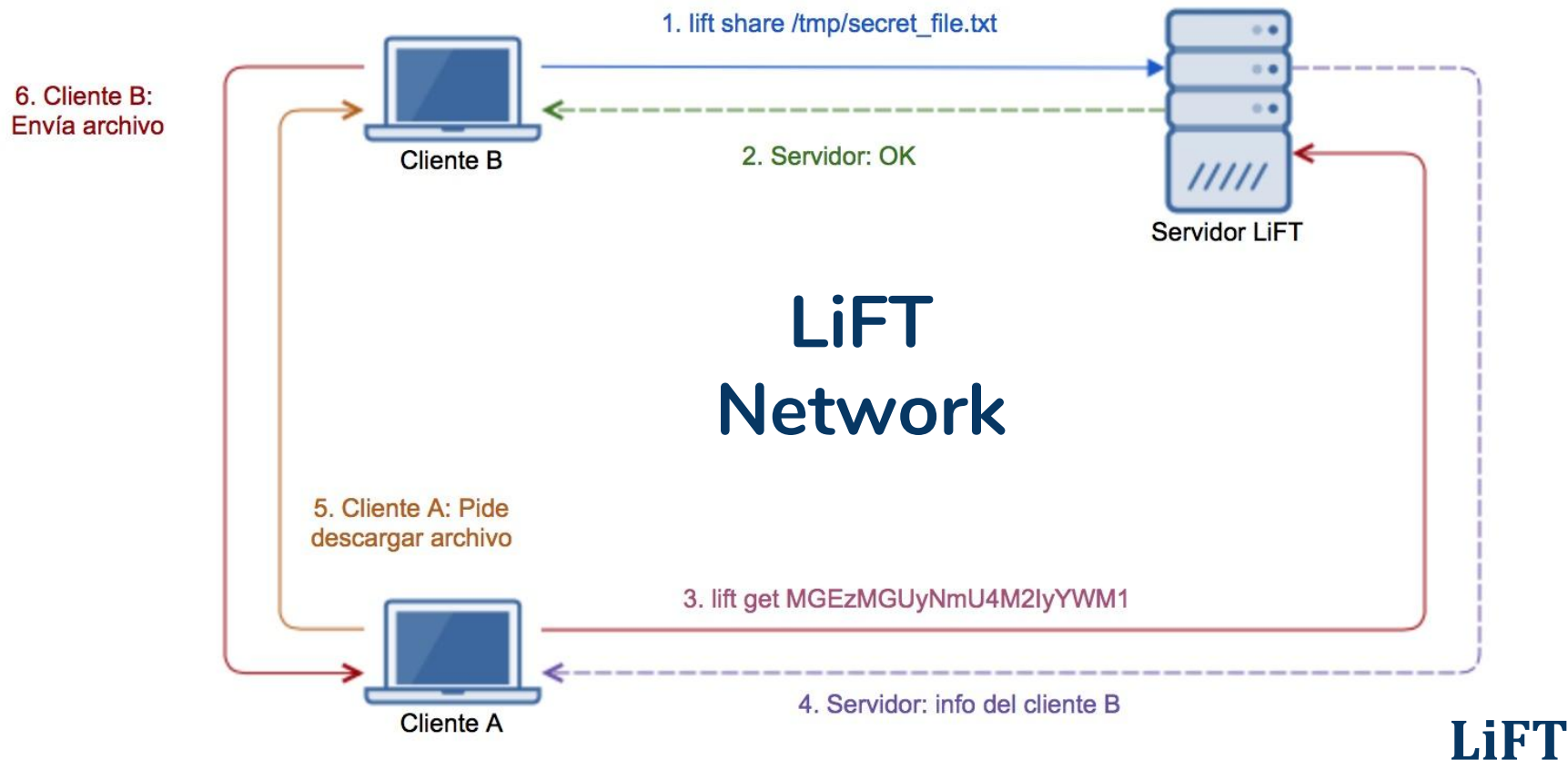
Cliente LiFT:

- Registrar archivos en el repositorio local para ser compartidos por medio de un demonio.
- Generar enlaces magnéticos (UFLs)
- Servir peticiones de otros clientes que han usado un UFL.
- Mandar una señal de vida hacia el server (HeartBeat Service)
- Descargar un recurso usando un UFL.

Server LiFT:

- Registrar clientes en la red LiFT.
- Capturar señales a desde los clientes para comprobar disponibilidad. (HeartBeat Service)

Diagrama de la red:



Caso de uso:

Compartir archivo

1. Se requiere compartir el archivo:
/home/bucaneer/S06E01_ThatShowAboutDragons.mp4
2. Se ejecuta el comando SHARE.

```
$ lift share /home/bucaneer/S06E01_ThatShowAboutDragons.mp4
```

```
MGEzMGUyNmU4M2IyYWM1Yj11Mj11MTI6MmNmMjRkYmE1ZmIw
```

3. El UFL generado se comparte con otros clientes.

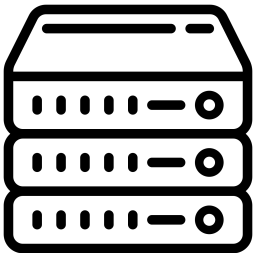


Caso de uso:

Servidor administra clientes en la red

1. Servidor escucha peticiones de nodos que quieren ingresar a la red.

```
{  
  "c_id": "89ad1060-48a0-4c76-b47f-ba42a95f",  
  "date_joined": "2017-11-21 17:24:10",  
  "ip_addr": "10.244.155.116",  
  "last_accessed": "2017-11-22 22:28:22",  
  "last_heartbeat": "2017-11-22 17:54:52",  
  "number_files_shared": "2",  
  "port": "24862"  
}
```



2. El servidor monitorea disponibilidad de los nodos en la red.
3. El servidor da de baja nodos en caso de desconexión.

Caso de uso:

Descarga de archivo usando UFL

1. El cliente solicita el recurso asociado al enlace UFL con el comando GET.

```
$ lift get MGEzMGUyNmU4M2IyYWM1Yj1lMj1lMTI6MmNmMjRkYmE1ZmIw
```

```
Getting file S06E01_ThatShowAboutDragons.mp4
```

```
Downloading: [=====> ] 40021 / 45463 bytes
```

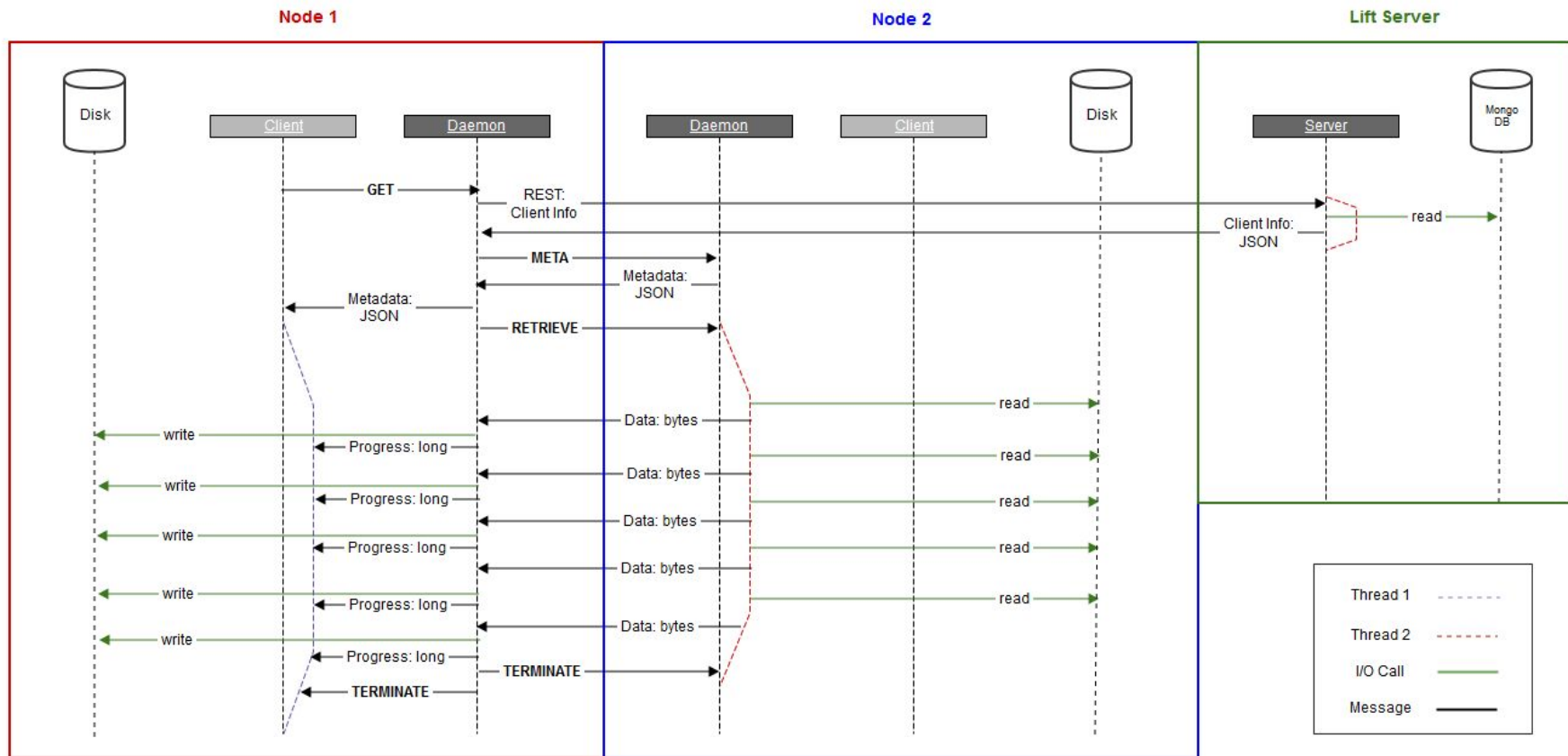
```
File downloaded to local repo: /tmp/S06E01_ThatShowAboutDragons.mp4
```

2. El archivo se descarga en el repositorio local del cliente

3. Metadatos asociados al archivo (última fecha de acceso, número de descargas etc) son actualizados en el servidor.



Operación GET entre dos nodos:





Demostración de uso

Instalación y configuración

1. Descargar los archivos desde el repositorio de Github (El enlace se provee al final de la presentación)
2. Configurar el directorio donde LIFT escribirá metadatos y el directorio donde se descargarán los archivos:

```
bash-4.1$ pwd  
/scratch/vsegundo/lift/lift
```

```
bash-4.1$ cat lift-config.properties  
lift.build.version=1.0  
lift.build.date=Saturday, November 18th, 2017.  
lift.daemon.hostname=localhost  
lift.shared.dir=/scratch/vsegundo/lifthome/shared  
lift.config.dir=/scratch/vsegundo/lifthome/config
```

```
lift.server.hostname=http://slc07grj.us.oracle.com  
lift.server.port=8181
```

```
lift.daemon.heartbeat=20
```

Instalación y configuración (cont)

3. Definir la variable de ambiente LIFT_HOME. Esta variable debe hacer referencia al directorio en donde se encuentra la instalación de LIFT.

```
bash-4.1$ export LIFT_HOME=/scratch/vsegundo/lift/lift
```

```
bash-4.1$ env | grep LIFT_HOME  
LIFT_HOME=/scratch/vsegundo/lift/lift
```

Nodo A: Lanzar el daemon.

```
-bash-4.2$ ./daemon-start.sh &  
[1] 24155
```

```
-bash-4.2$ ps -eaf | grep daemon-start.sh  
vsegundo 24155 10181 0 10:57 pts/32 00:00:00 /bin/bash ./daemon-start.sh
```

Nodo A: Compartir un archivo y generación de UFL

```
-bash-4.2$ ./lift.sh share /scratch/vsegundo/lifthome/zoom_0.mp4
```

```
-bash-4.2$ ./lift.sh files
```

LOCATION	FILE ID	DATE ADDED	SIZE	HITS
/scratch/vsegundo/zoom_0.mp4	27226fef1af	17 hours ago	53.7 MiB	0
/scratch/vsegundo/Tarea3.jar	e0c59336c8be	17 hours ago	18.1 KiB	0
/scratch/vsegundo/lifthome/zoom_0.mp4	ac7af94ad994	16 hours ago	68.0 MiB	0
/scratch/vsegundo/lifthome/bears1.jpg	d3c7f6691e9f	13 hours ago	333.8 KiB	0

```
-bash-4.2$ ./lift.sh ufl ac7af94ad994
```

```
UFL is: OD1hZDEwNjAtNDhhMC00Yzc2LWI0N2YtYmE0MmE5NWY6YWw3YWY5NGFkOTk0
```

```
-bash-4.2$ █
```


Nodo B: Descargar archivo remoto desde A

```
bash-4.1$ ./lift.sh get ODlhZDEwNjAtNDhhMC00Yzc2LWl0N2YtYmE0MmE5NWY6YWY5NGFkOTk0
Downloading: [=====] 68.0 MiB / 68.0 MiB
Download complete!
File location : /scratch/vsegundo/lifthome/shared
bash-4.1$ ls /scratch/vsegundo/lifthome/shared
zoom_0.mp4
```

Nota: Se asume que el daemon LIFT se encuentra corriendo en el Nodo B.

LiFT

Otros comandos

```
bash-4.1$ ./lift.sh --help

Usage:  lift COMMAND

A P2P client for lightweight file transfer

Options:

    --help    Print usage

Commands:

    add        Add a file to local repository
    files      List files
    get        Get the file from remote repository
    id         Show the user GUID in the network
    rm         Remove a file from local repository
    share      Share a file and generate it's UFL
    ufl        Generate the file's UFL
    version    Show the Lift version information
```


Comandos: id y version

```
bash-4.1$ ./lift.sh id
User GUID is: 554ea528-7a40-4c98-a0ee-3502bab6


You are [ CONNECTED ] to the Lift network.
```

```
bash-4.1$ ./lift.sh version
Client:
  Version:      1.0
  Built:        Saturday, November 18th, 2017.

Server:
  Version:      1.0.0
  Built:        Thu Nov 23 12:21:07 PST 2017
```



Resultados y entregables



Tecnologías que se usaron

1. Cliente: Programa escrito en **Java** que maneja la interacción con el servidor usando tecnología **RESTful**; y la transferencia de recursos usando **sockets TCP**.
2. Servidor: Programa escrito en **Python** que atiende las solicitudes RESTful. Los datos de los enlaces quedarán registrados en una base de datos no relacional como **MongoDB**.

Retos

- ❑ Control de creación de nuevos threads con diversas responsabilidades entre el cliente en línea de comando, el daemon del lado del cliente y el daemon del lado del servidor remoto.
- ❑ Sincronización de threads a través de memoria compartida y semáforos.
- ❑ Creación y destrucción de los sockets en el orden y momento adecuados.
- ❑ QoS para transferir los datos de forma rápida y notificar en tiempo real al cliente del proceso de descarga.
- ❑ Soporte multiplataforma (Linux, MacOS, Windows)

Trabajo futuro

- ❑ Realizar validación entre el cliente y el servidor para validar que no exista un firewall.
- ❑ Habilitar LiFT para funcionar sobre Internet, cuando un cliente está detrás de NAT.
- ❑ Trabajar en un cliente gráfico.
- ❑ Optimizar el servidor (agregar balanceo de carga, replicación) para soportar gran cantidad de peticiones simultáneas.

Código fuente y binarios

Para acceder al código fuente del cliente y del servidor, visite el siguiente repositorio de Github:



<https://github.com/israel-segundo/sistdist-lift>

Referencias

Rusty, E. (2013) Java Network Programming. O'Reilly Media, Inc.

Leuf Bo (2002) Peer to Peer: Collaboration and Sharing over the Internet. Addison-Wesley Professional.

Vu Q.H., Lupu M., Ooi B.C. (2010) Architecture of Peer-to-Peer Systems. In: Peer-to-Peer Computing. Springer, Berlin, Heidelberg.

Coulouris, George (1988). Distributed Systems: Concepts and Design. Addison-Wesley.

GNUTELLA 0.6 RFC. Recuperado el 20 de octubre de 2017 de:

<http://www.infosys.tuwien.ac.at/Teaching/Courses/NetworkServices/Gnutella/Development-GnutellaProtocol-v0.6-200206draft.txt>

Ivkovic, Igor (2005). Improving Gnutella Protocol. Cornell University. SWAG. Recuperado el 20 de octubre de 2017 de:

<https://www.cs.cornell.edu/people/egs/615/gnutella.pdf>

Lanham, Nick et al. (2003). Making Gnutella-like P2P Systems Scalable. Cornell University. Recuperado el 20 de octubre de 2017 de:

<https://www.cs.cornell.edu/people/egs/cornellonly/syslunch/fall03/gnutella.pdf>