Desarrollo y comparacion de algoritmos de coordinación de sistemas

En este informe se revisarán los algoritmos de exclusión mutua centralizada y descentralizada utilizados en el desarrollo de una biblioteca online al igual que una revisión de los clientes y como estos cargar y descargan archivos para ellos.

Para comenzar con la revisión de lo realizado en este proyecto y el cómo fue realizado debemos primero separar el desarrollo del proyecto en 4 partes, ClienteDownloader, ClienteUploader, los Nodos centralizados y los Nodos descentralizados.

Empezando por el ClienteUploader el cual es el encargado de cargar los libros a los nodos. Para este proceso se le pide al cliente que elija la opción de carga siendo sus opciones: “Centralizado” y “Descentralizado”, dependiendo de la opción que elija diferentes funciones serán utilizadas, seguido de esto se le pide al cliente que ingrese el nombre del libro a cargar.

```
fmt.Println("Eliga una opcion: ")
fmt.Println("1. Cliente uploader")
fmt.Println("2. Cliente downloader")
fmt.Println("----------------------")

fmt.Scanln(&op_cliente)

if strings.Compare(op_cliente, "1") == 0 {
    fmt.Println("----------------------")
    fmt.Println("Va a subir un libro, escoga el algoritmo que utilizara: ")
    fmt.Println("1. Centralizado")
    fmt.Println("2. Distribuido")
    fmt.Println("----------------------")

    fmt.Scanln(&op_algo)
    if strings.Compare(op_algo, "1") == 0 {
        op_algo = "Centralizado"
    } else if strings.Compare(op_algo, "2") == 0 {
        op_algo = "Distribuido"
    }
    SubirLibro(op_algo)
```

Figura 1: Cliente Uploader

El ClienteUploader posterior a esta acción se preocupa de separar el archivo (En estos casos serán archivos pdf) en Chunks de 250 kb para luego enviarlos a un Nodo aleatorio. Toda esta acción es independiente de la elección que el cliente haga, ya que de cualquier manera los Chunks deben ser enviados a un Nodo aleatorio el cual será el encargado de trabajar la propuesta y repartir los Chunks utilizando el algoritmo que el cliente eligió.

El código del paso de Chunks de Cliente a Nodo es bastante lineal, recibe los Chunks uno por uno esperando que le llegue el último Chunk. Cuando esto pasa el Nodo este revisa que algoritmo el ClienteUploader eligió.

Si la elección elegida por el cliente fue la del algoritmo centralizado entonces el nodo realiza una propuesta rápida y la envía al NameNode

```
if strings.Compare(message.Algo, "Centralizado") == 0 {
    fmt.Println("Entre a centralizado")
    res := propuestaDataNode(int(message.NumPartes))
```

```
s.mut.Lock()
prop, _ := c.AceptarPropuesta(context.Background(), &res2)
fmt.Println(prop.Nombre)
response, _ := RepartirChunks(prop)
fmt.Println(response)
s.mut.Unlock()
```

Figura 2 y 3: Propuesta generada y revisada

El algoritmo centralizado está hecho de manera que si la propuesta inicial no es aceptada entonces el NameNode creará una nueva propuesta y la escribirá en el log al igual que devolver la nueva propuesta para que el nodo envíe los Chunks a sus respectivos DataNodes.

Autor: Jorge Juan, Maximiliano Ojeda Campus: San Joaquín Fecha: 02/12/2020 Email: Jorge.juan@sansano.usm.cl, Maximiliano.ojeda@sansano.usm.cl

```
if flag1 == true && flag2 == true && flag3 == true {
    respuesta = *message
} else {
    fmt.Println("Entra a propuesta")
    intero, err := strconv.Atoi(message.Total)
    if err != nil {
        log.Fatalf("Error en transformar strings en int: %s", err)
    }
    aux := propuesta(intero)
    respuesta = PropuestaRespuesta{
        Nombre: message.Nombre,
        Total:  message.Total,
        Nd1:    aux.Nd1,
        Nd2:    aux.Nd2,
        Nd3:    aux.Nd3,
        Intn1:  aux.Intn1,
        Intn2:  aux.Intn2,
        Intn3:  aux.Intn3,
    }
}
escribirLog(respuesta)
s.mut.Unlock()
return &respuesta, nil
```

Figura 3: Revisión propuesta

La función escribirlog(propuesta) genera o edita el archivo log.txt el cual contiene todos los libros que están en los nodos además de cuantos Chunks tiene cada libro y donde puede ser encontrado cada uno de los Chunks. Por ejemplo, para el Libro mujercitas tenemos la siguiente distribución:

```
Mujercitas 9
Mujercitas_0 :9001
Mujercitas_3 :9001
Mujercitas_6 :9001
Mujercitas_1 :9002
Mujercitas_4 :9002
Mujercitas_7 :9002
Mujercitas_2 :9003
Mujercitas_5 :9003
Mujercitas_8 :9003
```

En la cual se puede apreciar como el NameNode guarda el registro luego de haber aceptado la propuesta.

Finalmente, el Nodo que envió la propuesta recibe la aprobación o la nueva propuesta y procede a enviar los Chunks correspondientes a los Nodos que la propuesta dice. En este caso no se mostrará todo el código de envío de Chunks ya que utiliza varias líneas de código para separar los mensajes y realizar las conexiones que ayudan al envío del código, pero no son el envío en sí, pero las funciones son las siguientes:

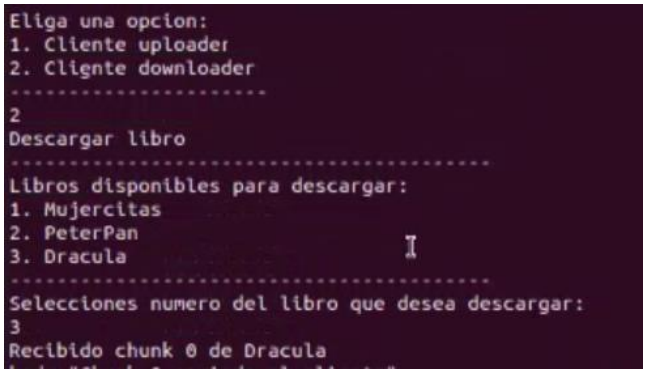
```
358 > func RepartirChunks(message *PropuestaRespuesta) (*Message, error) {···
486   }
487
488 > func (s *Server) EnviarChunksEntreNodos(ctx context.Context, message *Chunk) (*Message, error) {···
502   }
503
```

En el caso del envío de paquetes con el algoritmo descentralizado, este se llevó a cabo con la reutilización de gran parte del código como lo es la generación de propuesta y la aceptación de esta al igual que el uso de las funciones repartir Chunks y Enviar Chunks ente Nodos que pueden ser observadas en la imagen anterior, lo que si cambia en este algoritmo es la utilización de las variables mux y estado sirven para los estados de los Nodos.

```
type Server struct {
    dn1 []int
    dn2 []int
    dn3 []int
    estado string
    mux sync.Mutex
}
```

En la realización del algoritmo distribuido se utilizaron estos estados para la implementación del algoritmo de Richard & Agrawala. Debido a la reutilización de código en las funciones las imágenes son redundantes.

La última parte del proyecto es la del ClienteDownloader el cual simplemente se comunica con el NameNode solicitándole el nombre de los libros disponibles para después ingresar la opción del libro que desea para luego pedir el Chunk a cada uno de los Nodos correspondientes en el log y uniéndolos todos en el pdf.



En las siguientes imágenes se puede apreciar como se ve el uso de los algoritmos en las terminales.

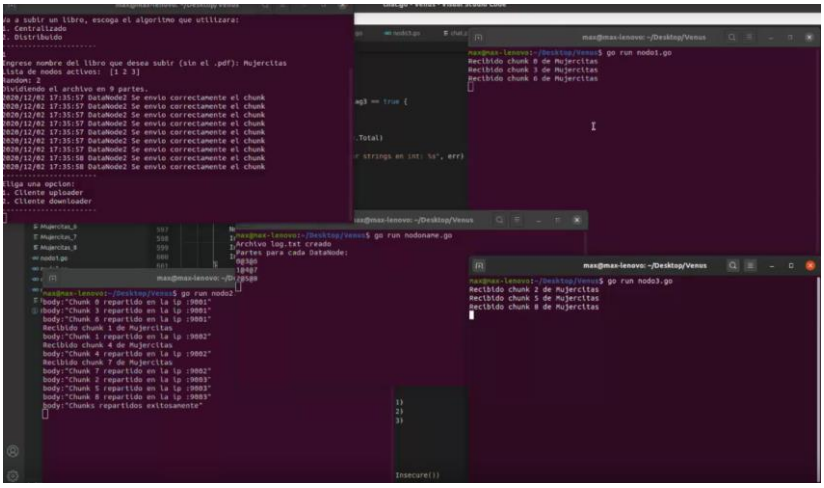


Figura 4 Algoritmo centralizado

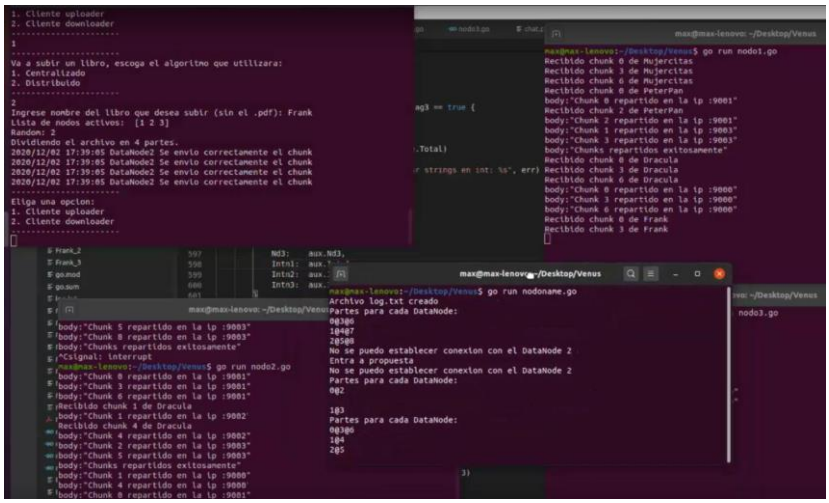


Figura 5 Algoritmo Distribuido

Gracias a los print podemos observar a simple vista que ambos algoritmos funcionan y realizan la función de separación de un archivo y su distribución de igual manera, al menos a simple vista ya que en términos de lo que ocurre por detrás es altamente distinto, mas que nada en el quien realiza que actividad y cuanto peso se le coloca al NameNode, en términos de cual de los 2 algoritmos es más rápido o realiza menos mensajes entre nodos el algoritmo centralizado solo tiene un O(n), por ejemplo si consideramos que hay n Nodos y n Chunks el algoritmo centralizado solo enviará una propuesta al NameNode el cual revisara los n Nodos para posteriormente guardar el log mientras que el Nodo que envió la propuesta reparta los N Chunks, todo esto da un total de 2n+2 en teoría por lo que su tiempo de ejecución es de O(n). En el caso

del algoritmo descentralizado o distribuido fue mencionado que la propuesta debe ser aceptada por todos los nodos, por lo que la propuesta se envía n veces para que estos n Nodos revisen la propuesta y comenten su respuesta para que la distribución de Chunks sea realizada por el Nodo que creó la propuesta. Sumando esos 2 tiempos mas el tiempo que el Nodo inicial pasa enviando el log al NameNode podemos observar que también llegamos un total de 2n+2 o 3n+2 en el tiempo de ejecución de O(n) el cual es en teoría el mismo que el del algoritmo centralizado.

Teniendo ambos tiempos en mente y considerando como funcionan ambos algoritmos podemos observar que una de las mayores diferencias entre estos algoritmos reside en el rol que se le da a los Nodos y al NameNode dependiendo del algoritmo. En el caso del algoritmo centralizado el NameNode se encarga de revisar cada propuesta, crear una nueva propuesta en el caso de que sea necesario. De igual manera en el algoritmo centralizado el NameNode solo se encarga de escribir el log de las propuestas que recibe al igual que enviar los logs necesarios cuando el cliente cree *requests* de descarga de libros mientas que en el caso del algoritmo distribuido todo el peso de la creación de propuestas y la aprobación de ellas se encuentra en los Nodos los cuales solo informan al NameNode de las propuestas aprobadas. Con esto podemos observar que, si bien los tiempos son similares, en el caso del algoritmo descentralizado, o distribuido, el tiempo de ejecución se reparte en los nodos por lo que es posible ver una disminución en los tiempos de ejecución final ya que no es un nodo el que realiza todo el trabaja si no que todos los nodos menos uno realizan el trabajo.

Teniendo todo lo previamente mencionado en mente sería correcto decir a primera vista que el algoritmo distribuido es “mejor” que el centralizado aun cuando es mas complicado de aplicar que el centralizado, pero la conclusión que se le dio a esta pregunta es que ninguno de los algoritmos es mejor que el otro si no que depende. No sería correcto decidir cual algoritmo es mejor si no se tiene en mente cual será el ambiente en el cual se utilizará el algoritmo, ni tampoco sin saber una estimación del flujo de clientes. Por ejemplo, si una aplicación será utilizada por una cantidad reducida de personas entonces se podría hacer el argumento que el algoritmo centralizado y pocos nodos es mucho mas conveniente y ahorrador, mientras que en el caso de una aplicación con un gran flujo de clientes será más conveniente colocar más nodos y utilizar el NameNode solo para la escritura de logs y para la respuesta a clientes.

De igual manera otro punto de vista es el tiempo de respuesta al cliente. Si bien una aplicación puede demorarse en realizar lo que el cliente le pide, no es aceptable que una aplicación se demore en preguntarle al cliente lo que este necesita. El cliente debe ser la prioridad de las aplicaciones como la desarrollada en este informe y la posibilidad de que el NameNode este siendo sobre utilizado o este trabajando con muchas requests provenientes de otros clientes las cuales resulten en una respuesta tardía del NameNode al cliente es mucho mas probable cuando se utiliza el algoritmo centralizado, teniendo esto en mente fue concluido que para este caso en especifico como lo sería una biblioteca online puede que lo mas correcto sea la utilización del algoritmo de exclusión mutua distribuida ya que al final del día el cliente es lo mas importante para una empresa pues sin ellos la empresa pierde su razón de ser.