Universidad del Bío-Bío, Concepción, 2014.
Facultad de ciencias empresariales

UNIVERSIDAD DEL BÍO-BÍO

“Descarga en paralelo de archivos sensible a variaciones del ancho de banda”

Proyecto de título para optar al título profesional de

Ingeniería civil informática.

Autor : Cristopher Barrientos Matamala

Profesor guía : Patricio Galdames Sepúlveda

Resumen

Este proyecto se presenta para dar conformidad a los requisitos exigidos por la universidad del Bío-Bío en el proceso de titulación para la carrera de ingeniería civil informática. El proyecto titulado *"Descarga en paralelo de archivos, sensible a variaciones del ancho de banda"* propone y evalúa una aplicación que realiza la descarga concurrente de archivos de gran tamaño desde la Internet. La aplicación requiere que el archivo esté almacenado en varios servidores y descarga distintas partes desde cada servidor. En este trabajo proponemos una técnica que realiza una estimación de la tasa de transmisión lograda con cada servidor en uso y ajusta el tamaño de las descargas de acuerdo a esta métrica. La idea es descargar más Bytes desde aquellos servidores que proveen una mejor tasa de transmisión. Aunque la aplicación fue implementada sobre el protocolo de aplicación FTP [1], esta puede ser acondicionada sobre cualquier protocolo de aplicación que permita definir rangos de Bytes, como también lo es HTTP [2]. Además se escogió realizar la implementación con el lenguaje de programación Java, de modo que nuestra aplicación puede ser ejecutada sobre cualquier sistema operativo que tenga instalada la máquina virtual de Java. Comparamos el rendimiento de nuestra aplicación con respecto a una versión que no se ajusta a la variaciones del ancho de banda y también con respecto a la descarga del archivo desde un solo servidor. Nuestros resultados muestran que nuestra técnica logra reducir los tiempos de descarga y logra distribuir de mejor forma la descarga a través de diversos servidores.

Abstract.

This project was carried out to fulfill the degree requirements of the Universidad del Bio-Bio in order to obtain the professional title in computer engineering. The project entitled "Parallel File Download sensitive to variations in bandwidth " proposes and evaluates an application that performs concurrent downloading of large files from the Internet . The application requires the target file must be stored on multiple servers and this download different parts from each server. In this proyect we propose a technique that estimates transmission rate achieved with each server in use and it adjusts the size of the part to be download according to this metric . The idea is to download more bytes from servers that provide better transmission rate . Although the application was implemented on the FTP [1] protocol implementation , this can be fitted to any applicaton protocol that allows Byte range, as is HTTP [2]. Our application was chosen to be implemented with the Java programming language. Thus, our application can be run on any operating system that has the Java virtual machine installed. We compare the performance of our implementation with regard to a version that is blind to the bandwidth variation and also with regard to the downloading from a single server . Our results show that our technique is able to reduce download times and achieves a better way to distribute the download through various servers

Índice General

Índice de tablas

Índice de ilustraciones

1 INTRODUCCIÓN

Este proyecto tiene a lugar en la Universidad del Bío-Bío sede Concepción, ubicada en la Región del Bío-Bío, Chile. En el cual se presenta una estudio comparativo de diversas técnicas de descargas en paralelo de archivos grandes desde múltiples servidores, con el fin de disminuir sus tiempos de descarga.

El incremento de ancho de banda en la red que experimentan los usuarios ha estimulado el desarrollo de material Web con características multimediales, es decir, material que contiene audio y video cuyo tamaño puede ser de cientos de Mbs o GBs. En muchas ocasiones tal material, el cual llamamos simplemente archivo, está almacenado en diversos servidores ya sea por propósito de disponibilidad o bien para reducir su tiempo de descarga (rendimiento).

Dado que el material está replicado, el usuario debe de seleccionar el mejor servidor para ejecutar desde este la descarga. La mayoría de las veces un usuario desconoce cual es el mejor servidor para comenzar la descarga y aún si la conociese, dada la inestabilidad del ancho de banda de la red y de las variaciones de la carga de trabajo del servidor elegido, este pudiera no mantener su alto rendimiento (bajo tiempo de descarga) durante todo el tiempo de la descarga. En una situación extrema puede suceder que la descarga sea interrumpida, teniendo que ser reanudada desde otro servidor.

Para enfrentar estos problemas, diversos investigadores [2-8] han propuesto a la descarga en paralelo. Esta consiste en establecer varias sesiones TCP concurrentes de descarga de partes de un archivo desde diversos sitios que poseen el archivo requerido. Cuando todas las partes del archivo son finalmente descargadas, el archivo es reconstituido y verificada su correcta organización. Esta técnica de descarga ha sido extensivamente usada por servicios de distribución de software que operan con servidores espejo, por ejemplo empresas desarrolladoras de Linux, o bien aplicaciones de almacenamiento basadas en la nube o redes de archivos basadas en P2P, como es el caso de BitTorrent.

En este proyecto proponemos una nueva técnica de descarga en paralelo que es consciente a las variaciones del ancho de banda. Nuestra idea es periódicamente realizar estimaciones de ancho de banda a cada uno de los servidores escogidos y realizar ajustes del rango de Bytes descargados desde cada servidor. Mientras mayor sea la tasa de descarga obtenida con un servidor, mayor será el número de Bytes a descargar desde ese servidor. Dado que el chequeo es periódico nuestra técnica es capaz de adaptarse a fuertes cambios del ancho de banda. Realizamos extensivos experimentos de la técnica propuesta y comprobamos que se adapta adecuadamente a escenarios de ancho de banda variable.

Este trabajo de título se resume del siguiente modo. En el Capítulo uno se define cual es el propósito de este proyecto y en que consiste lo propuesto. Luego en el capítulo dos se da a conocer lo que la aplicación necesita

para poder ejecutarse y lo que debe realizar o no realizar. El capítulo tres se realiza un análisis de los procesos que tiene la aplicación. Antes de terminar en el capítulo cuatro se presentan los experimentos realizados con las técnicas propuestas, para luego terminar en el capítulo cinco con las conclusiones que se obtuvieron.

2 DEFINICIÓN PROYECTO

2.1 Definiciones, siglas y abreviaciones.

- **DPA:** Descarga Paralela Adaptativa.
- **DPNA:** Descarga Paralela No Adaptativa.
- **P2P:** Peer-to-peer, red punto a punto es una red de computadoras en la que todas actúan como nodos, sin clientes ni servidores, para compartir archivos.
- **FTP:** File transfer protocol, es un protocolo de red para la transferencia de archivos.
- **TCP:** Transmission control protocol, es uno de los protocolos fundamentales de Internet, que aporta transporte confiable y bidireccional de datos.
- **IDE:** Integrated development environment, es un programa informático que consiste en un conjunto de herramientas para facilitar la edición de código para uno o varios lenguajes de programación, como también la compilación y depuración del código.
- **HTML:** Hypertext markup language, es un lenguaje de etiquetas para la elaboración de páginas Web.
- **CDN:** Content delivery network, es una red de computadoras colocadas de forma que los usuarios puedan sacar el máximo provecho al ancho de banda. Es descentralizado para evitar embudos de que todos los usuarios accedan al mismo servidor central.

2.2 Objetivos del proyecto.

- **Objetivo general.**
  - Desarrollar una técnica sobre el protocolo FTP que permita descargar archivos grandes en forma paralela, adaptándose al desempeño de cada servidor.
- **Objetivos específicos.**
  - Conocer técnicas propuestas por otros autores dentro de la misma área de estudio a fin de plantear un técnica que difiera de las demás.
  - Conocer el funcionamiento del protocolo FTP con el fin de establecer los diversos aspectos en que funciona dicho protocolo.

- Obtener los tiempos del método de descarga en paralelo no adaptativa, al ejecutar dentro del mismo ambiente de trabajo que la técnica propuesta.
- Establecer un algoritmo que permita medir y distribuir la carga de los servidores.

2.3 Ambiente de ingeniería de software.

- **Metodología de desarrollo utilizada.**

  La metodología incremental es la que se utiliza en este proyecto debido al número limitado de desarrolladores y porque el proyecto es de tamaño pequeño. Esta metodología consiste en realizar iteraciones. Una característica muy útil en esta metodología es desarrollar versiones entregables del sistema, empezando con una implementación simple con respecto a los requerimientos del software, para luego ir incorporando nuevas funcionalidades y mejoras a la nueva versión que se entregará en la siguiente iteración, para luego retroalimentarse de la iteración anterior .

- **Estándares de documentación.**

  Para documentar el proyecto se hace uso de javadoc que dispone Java, el cual simplifica el proceso de documentar, ya que es una utilidad del lenguaje que crea un documento en HTML a partir del código fuente y que está presente en la mayoría de los IDEs de este lenguaje.

2.4 Descarga en Paralelo de Archivos.

La descarga en paralelo de archivo consiste en una técnica que permite la descarga concurrente de distintas partes de un archivo completamente almacenado en diferentes servidores. Al finalizar la descarga de las partes, la misma aplicación debe verificar que la unión de ellas permita recuperar exactamente el archivo original. Esta técnica ha sido estudiada para reducir el tiempo de descarga de archivos grandes y para mitigar los efectos de fallas del proveedor del contenido debido a que asume que el archivo está replicado en distintos sitios. A continuación presentaremos una descripción de distintos escenarios donde ha sido empleada y presentaremos los detalles de la técnica empleada en este trabajo.

2.4.1 Trabajos Relacionados.

La descarga en paralelo de archivos (DP), ha sido utilizada en diversos escenarios como en redes P2P, proveedores de distribución de contenido o CDN y proveedores de discos virtuales en la nube con el fin de reducir los tiempos de descarga de archivos de gran tamaño (sobre 1 GB). En estos escenarios ya sea una parte o todo el archivo completo es copiado y distribuido en distintos servidores, clusters, proxies o pares ubicados en

distintos puntos de la Internet con el fin de garantizar su disponibilidad.

DP fue inicialmente investigada por Byers [6] para mejorar la disponibilidad del contenido, quien propuso dividir en k partes un archivo y luego genera m partes redundantes con la k partes iniciales. La idea es que cualquier grupo de k partes puede ser usada para reconstruir el archivo original. Rodríguez [3] fue el primero en tomar en cuenta los recursos de red de cada servidor con el fin de determinar la carga de trabajo o el tamaño de la parte de un archivo a descargar. Este último trabajo propuso dos variantes de DP basado en HTTP. La primera opción consiste en mantener información histórica de la tasa de descarga obtenida por cada servidor y seleccionar el tamaño de la parte a descargar de manera proporcional a la tasa de descarga. La segunda opción denominada dinámica descarga pequeños bloques de 32KB desde cada servidor hasta completar la descarga del archivo original. Funasaka et al. [5] propuso una técnica DP que toma en cuenta la variabilidad del ancho de banda y propone un método adaptativo para determinar el tamaño de un bloque a descargar desde un servidor.

Otras implementaciones alternativas de DP están basadas en FTP. Similarmente a [5], Sohail et al [7] propuso un DP que dinámicamente cambia las partes de un archivo mediante el monitoreo del flujo FTP. Al-Jaroodi at al. [4] propuso hacer la descarga concurrente de un bloque de un archivo desde diferentes direcciones. Suponga que hay 2 servidores que contienen una copia del archivo. El cliente comienza una primera descarga con el primer servidor desde el inicio del archivo y simultáneamente comienza una segunda descarga del archivo hacia el segundo servidor pero comenzado desde final del archivo. La descarga se detiene cuando ambas descargas concurrentes se han encontrado. Asumiendo que el archivo está copiado en n servidores, esta idea fue generalizada a n servidores. El archivo es particionado en n/2 partes y luego cada parte se descarga concurrentemente desde dos servidores.

Grid-FTP [8] fue diseñado para escenarios de clusters al incorporar nuevas funcionalidades no incluida en el estándar FTP, por lo que su uso requiere de un servidor FTP especial, a diferencia de los trabajos previos que operan con el estándar básico de FTP.

2.4.2 Técnica propuesta: Descarga en Paralelo Adaptativa.

En esta sección detallaremos una nueva técnica de descarga en paralelo de archivos, que llamamos DPA-1. Mas tarde introduciremos dos variantes desarrolladas a partir de esta técnica. Los detalles de DPA-1 se explican continuación:

Paso 1: Inicialización

1. El cliente selecciona *n* servidores que contengan una copia completa del archivo a descargar.
2. El archivo es particionado en *k* bloques de igual tamaño, cada bloque es llamado etapa (ε) y el valor de cada etapa es en Bytes. El valor de *k* puede ser arbitrario pero se fija igual a *n* para simplificarle al

usuario el uso de la aplicación. El valor de *k* determina cuantas veces el algoritmo realizará mediciones de tasa de transferencia a cada uno de los servidores. El valor de *k* debe ser de al menos 2 con el fin de permitir al menos un ajuste del tamaño de un bloque de acuerdo a la variabilidad del ancho de banda.

3. Cada bloque es a su vez dividido en *n* partes. El rango de bytes de cada parte que debe ser descargado desde cada servidor dependerá de la medición del ancho de banda de cada servidor.

Paso 2: Medición.

1. A cada servidor se le solicita descargar una parte del archivo a la que denominaremos muestra, esta muestra tiene un tamaño fijo, en el capítulo de experimentos se justifica el tamaño de la muestra. En esta versión DPA-1 la muestra siempre se extrae desde el principio de archivo y no se almacena en el disco.
2. Cada muestra se descarga desde cada servidor de manera secuencial, es decir, primero se descarga la muestra desde un servidor solamente y cuando termine se descarga la muestra desde el siguiente servidor. Esto se realiza con el fin de lograr una mejor estimación de la velocidad de carga del servidor.
3. Una vez que todas las muestras se descarguen se obtienen los tiempos que a cada servidor le tomó descargar la muestra. Para luego obtener los Bytes por milisegundo (B/ms) de cada servidor, esto se obtiene de la siguiente manera; Sea r_{ij} la tasa de descarga desde el servidor i en la etapa j, l_i el tamaño de la muestra en Bytes descargada desde el servidor i y t_i el tiempo de descarga, entonces, la tasa de descarga se estimó del siguiente modo:
4. $r_{ij} = (1-\gamma)* l_i/t_i + \gamma*r_{ij-1}$ (1).

   donde γ es general 1/8 [11], pero por simplicidad hemos usado $\gamma = 0$.
5. El siguiente paso es conocer la cantidad de Bytes que descargará cada servidor. Para esto se debe realizar una ecuación de primer grado, donde la variable X es la constante que al multiplicarla por cada r_{ij} de cada servidor resulta la cantidad de Bytes a descargar por cada servidor, y la suma de estas cantidades debe ser igual al tamaño de la etapa (ε).

   $\varepsilon = \Sigma\, r_{ij} \cdot X.$ (2)

   $X=\varepsilon / \Sigma\, r_{ij}$ (3)

   Luego el tamaño del rango de bytes a descargar desde cada servidor es

   $S_i= \varepsilon \cdot r_{ij} / \Sigma\, r_{ij}$ (4)

   Quedando proporcional a la tasa de transferencia.

Paso 3: Descarga

1. Para cada servidor seleccionado se envía una petición de descarga de acuerdo al rango de bytes asignado y se comienza a recepcionar los bytes recibidos desde cada servidor.
2. Ir al *Paso 2*.

Paso 4: Verificación

Cuando todo el archivo es descargado su integridad puede ser verificada por el usuario mediante programas de suma de verificación, tales como sha256sum que detectan cambios en la secuencia de datos con respecto a los valores obtenidos con el archivo original.

Un ejemplo para visualizar el uso de la técnica propuesta es la siguiente. Suponga que un archivo de 999MB desea ser descargado en paralelo desde 3 servidores. De acuerdo al *Paso 1* de nuestra técnica de descarga, el archivo es particionado en k =3 bloques de 333MB que equivale a 349175808 Bytes, como ε es el valor de cada bloque que a su vez es una etapa, ε = 349175808. El valor de k define los momentos en los cuales se realizarán las mediciones de ancho de banda, que son antes de comenzar la descarga, luego de completar los 333MB y finalmente luego de completar los 666MB.

En el primer punto del *Paso 2* de nuestra técnica, el primer bloque de 333MB es dividido en $n=3$ partes. El tamaño de cada parte se calcula según la medición que es el *Paso 2*. En este paso supongamos que el tamaño de la muestra es 16MB (16777216 Bytes), el servidor 1 demora 6500 milisegundos, el servidor 2 demora 10000 milisegundos y el servidor 3 demora 12000 milisegundos en descargar la muestra. Según la fórmula (1) para el servidor 1 r_{ij} =2581,11, para el servidor 2 r_{ij}=1677,72 y para el servidor 3 r_{ij} =1398,10. Luego calculamos el valor de X en la fórmula (3), lo que resulta X=61725,31. Por último la cantidad de Bytes a descargar de cada servidor según (4), sin decimales ya que no se puede descargar una fracción de un Byte, es; S1=159319814 Bytes, S2=103557787 Bytes y S3=86298155 Bytes, y la suma de las 3 partes es 349175756 Bytes, lo que debe ser igual al total del tamaño de la etapa, por lo tanto lo que falta para completar el tamaño total se le suma lo faltante a cualquier servidor, en este caso 52 Bytes. Luego se realiza la descarga de los rangos de Byte de cada servidor, en forma paralela y al finalizar se vuelve a realizar una medición antes de comenzar la siguiente etapa.

Finalmente, se verifica la integridad del archivo descargado. Este paso consiste en verificar que el tamaño del archivo declarado por el servidor es igual al tamaño del archivo descargado. Si uno de los servidores provee también el calculo de hash (MD5 o SHA-1 [9]) del archivo descargado el usuario también puede realizar este chequeo.

Variaciones a la técnica propuesta

Estas variaciones se realizan con la suposición de que aportan mejoras a la técnica original y que luego son comparadas en los experimentos para confirmar si realmente realiza una mejora.

DPA-2: incluye muestra en el archivo en cada etapa de la DPA.

Luego de analizar la DPA-1 y ver los resultados de los experimentos se propuso mejorar los tiempos total de descarga. Esta versión DPA-2 realiza los mismos pasos que la versión DPA-1, pero la diferencia consta en lo siguiente:

1. en el *paso 2: Medición*, la muestra descargada no se descarta y se agrega al archivo. Para lograr esto por cada muestra descargada se debe restar al tamaño de la etapa el tamaño de la muestra. Por ende a la fórmula (2) se le realiza el siguiente cambio.

   $\varepsilon - (l_i \cdot n) = \Sigma\, r_{ij} \cdot X.$ (5)

   $S_i = (\varepsilon - (l_i \cdot n)) \cdot r_{ij} / \Sigma\, r_{ij}$

   Cada muestra corresponde a un rango de Bytes que comienza con la secuencia de Bytes siguiente a la descargada en la muestra anterior o en la etapa anterior, ya que en la DPA-1 cada muestra comienza su secuencia de Bytes desde el Byte cero, no se realiza reanudación como en el caso de esta variación.

DPA-3: La medición de los servidores se realiza con la parte que descarga cada servidor en la etapa anterior.

En esta variante el *Paso 2: Medición* sólo se realiza al principio en la primera etapa y únicamente en esa etapa. Para realizar las adaptaciones en las siguientes etapas se hace uso de lo descargado en la etapa anterior, es decir, sigue el mismo procedimiento que la DPA-2 pero solamente hasta que termina la etapa 1. Luego de terminar la etapa 1 en el *Paso 2: Medición* se ocupan las mismas fórmulas (1), (2) y (3), pero en vez de descargar una muestra se ocupan los datos de la etapa 1, es decir, en la fórmula (1) l_i ahora corresponde al tamaño descargado por el servidor i en la etapa 1 y t_i corresponde al tiempo que tardo en descargar el servidor i su rango de Bytes asignado en la etapa 1.

A continuación se describe el procedimiento realizado en la DPA-3:

Paso 1: Inicialización.

1. Este paso es el mismo que DPA-2.

Paso 2: Medición.

1. Este paso es el mismo que DPA-2.

Paso 3: Descarga.

1. Para cada servidor seleccionado se envía una petición de descarga de acuerdo al rango de bytes asignado y se comienza a recepcionar los Bytes recibidos desde cada servidor.
2. Una vez terminada la descarga de los rangos de Bytes asignados a cada servidor se guarda la cantidad de Bytes descargados en l_i y el tiempo que le tomó a ese servidor descargarlos en t_i.

Paso 4: Calcular rango de Bytes.

1. En este paso sólo se realiza los cálculos de las fórmulas (1), (2) y (3). Sin descargar muestras, ya que los datos necesarios para realizar estos cálculos se obtienen del *Paso 3: Descarga*.
2. Ir al *Paso 3*.

3 ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

3.1 Alcances.

La aplicación tiene como propósito implementar las técnicas descarga en paralelo de archivos descritas en el Capítulo dos. Para todas las variantes propuestas, el usuario debe indicar al menos dos servidores donde se encuentre el archivo para poder realizar la descarga paralela.

La implementación de cualquiera de las variantes requiere solamente de servidores FTP, pero pueden ser extendidas sin mayor complejidad a otro nuevo protocolo de transporte como HTTP. Sólo se requiere que el protocolo provea de algún mecanismo que permite explicitar al servidor el rango de bytes que se desea descargar.

El software por si solo no puede realizar la tarea de gestor de descargas, la que consiste en colocar en cola varios archivos a descargar.

Cuando una descarga es interrumpida desde al menos un servidor, el software no puede realizar la reanudación de la descarga con los demás servidores que siguen activos.

3.2 Objetivo del software.

- **Objetivo General.**

  Implementar en Java las 3 variantes descritas en el capitulo 2.

3.3 Requisitos mínimos del software.

La aplicación requiere de unas condiciones para su completo funcionamiento, ya sean requisitos de software o de hardware. En este apartado veremos los requisitos mínimos que requiere el software.

Los requisitos de Sistema operativo y de hardware que se mencionarán a continuación son los requisitos que Oracle, la empresa a cargo de Java [10], requiere para su funcionamiento. Cabe destacar que para este proyecto no se comprobó el funcionamiento de la aplicación en todos los sistemas que Oracle menciona.

3.3.1 Requisitos de sistema operativo.

Dado que nuestra aplicación requiere que la máquina virtual de Java se encuentre instalada sobre el sistema operativo, los siguientes son los sistemas operativos y la versión en cual se puede ejecutar según recomendación

de Oracle.

Nombre: Linux.

Desarrollador: Varios.

Versión:

- Oracle Linux 5.5+
- Oracle Linux 6.x (32 bits), 6.x (64 bits)
- Red Hat Enterprise Linux 5.5+, 6.x (32 bits), 6.x (64 bits)
- Ubuntu Linux 10.04 y superior
- SUSE Linux Enterprise Server 10 SP2, 11.x

Nombre : Windows.

Desarrollador: Microsoft.

Versión:

- Windows 8 (escritorio)
- Windows 7
- Windows Vista SP2
- Windows XP SP3 (32 bits); Windows XP SP2 (64 bits)
- Windows Server 2008
- Windows Server 2012 (64 bits)

Nombre: Mac OS X

Desarrollador: Apple Inc.

Versión:

- Mac basado en Intel que ejecuta Mac OS X 10.7.3 (Lion) o posterior.

3.3.2 Requisitos de software.

Nombre: Java Runtime Enviroment

Desarrollador: Oracle.

Versión: 7 update 45 o posterior.

3.3.3 Requisitos de hardware.

Los requisitos de hardware hacen referencia a la configuración mínima de los dispositivos básico que un computador necesita para poder ejecutar la aplicación. En este apartado no se menciona el requisito del procesador por el motivo de que si puede ejecutar el sistema operativo puede ejecutar la máquina virtual de Java.

- Windows.
  - Ram: 128 MB y 64 MB para windows XP (32bits).
  - Espacio en disco: 124 MB.
- Mac OS X.
  - Sin restricción.
- Linux.
  - Ram: 64MB.
  - Espacio en disco: 58 MB.

3.3.4 Interfaces de comunicación.

En las interfaces de comunicación se menciona los protocolos con los cuales la aplicación emplea para comunicarse con los servidores.

- **Nombre:** File Transfer Protocol

  Abreviación: FTP

  Es el protocolo específico que deben usar los servidores en donde están alojados los archivos para la comunicación entre el servidor y la aplicación.

- **Nombre:** Transmission Control Protocol

  Abreviación: TCP

  Es el protocolo que se usa para el nivel de transporte de los segmentos de una manera confiable libre de errores y sin pérdidas.

3.4 Requerimientos funcionales del sistema.

Id	Nombre	Descripción
1	URLs de servidores.	El sistema debe dar aviso que requiere al menos dos URLs al usuario cuando se intenta iniciar con una o menos URLs.
2	Integridad del archivo	El archivo descargado debe contener la misma suma de verificación que el archivo original para asegurar su integridad.
3	Directorio de la descarga.	El sistema debe permitir al usuario seleccionar un directorio y descargar el archivo en el directorio seleccionado.
4	Interrupción de la descarga	El sistema debe notificar al usuario cuando se interrumpe la descarga con un servidor.
5	Tiempos	El sistema debe notificar al usuario el tiempo efectuado en la descarga total del archivo y el tiempo del desbalance.
6	URLs invalidas	El sistema debe notificar al usuario cuando una URL no es válida o no logra conexión con el servidor.
7	Avance de la descarga	El sistema debe mostrar el avance de la descarga al usuario constantemente.

Tabla 1: Requerimientos funcionales del sistema.

4 ANÁLISIS

4.1 Diagrama del modelo de procesos de negocio.

En este apartado se muestra el diagrama de modelado de procesos de negocio, también conocido como BPMN por sus siglas en inglés, el cual muestra los procesos que se efectúan durante la ejecución de la aplicación y la interacción con las otras entidades como es el usuario y los servidores. Este diagrama es realizado en formato de flujo de datos para que sea fácilmente legible y entendible para la mayoría de los lectores.

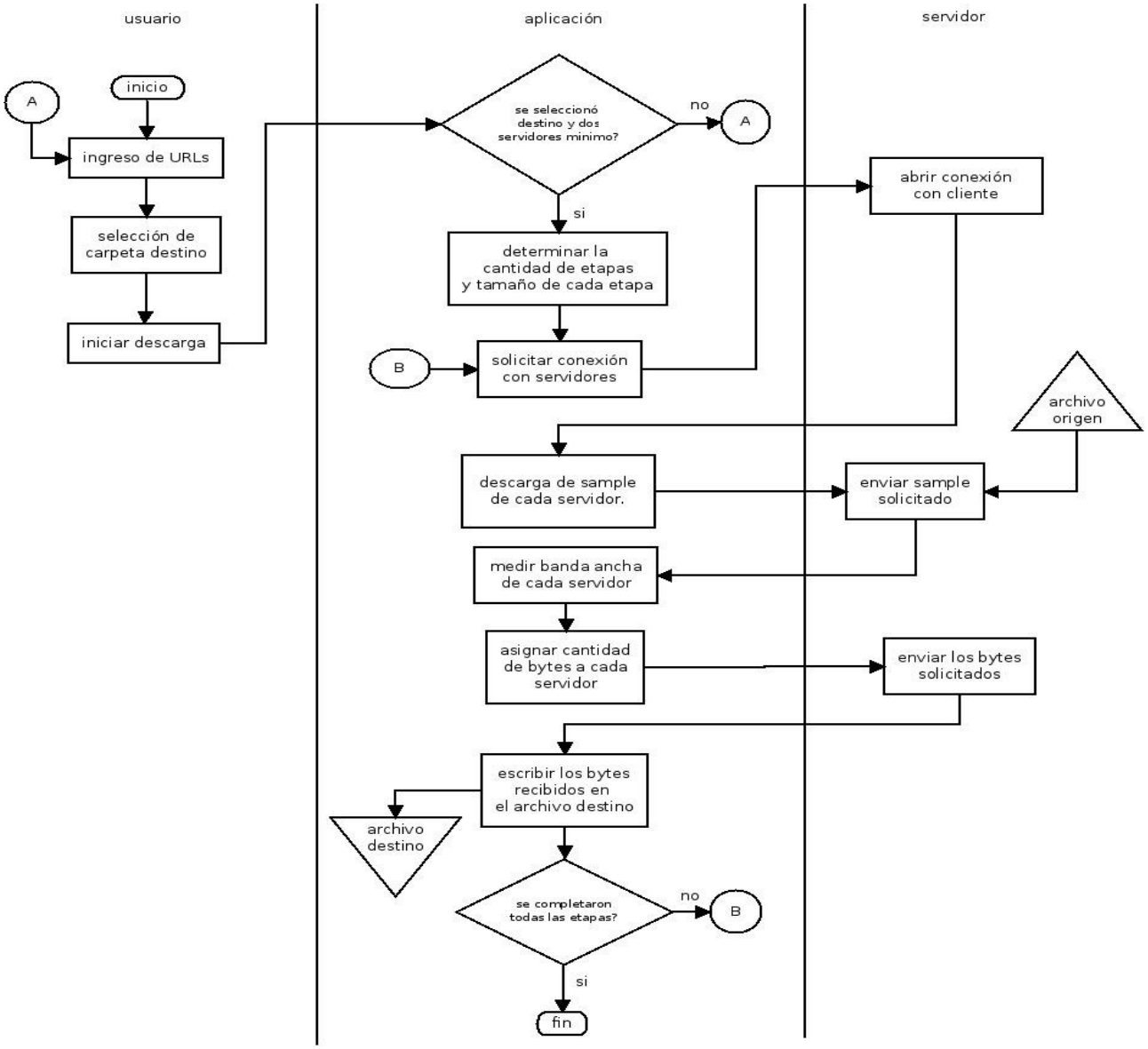

Ilustración 1: BPMN.

4.2 Diagrama de flujo de datos.

En este diagrama se presentan los flujos de datos que entran y salen de cada proceso del sistema y la interacción entre el sistema y las entidades externas, que en este caso es el usuario y los servidores.

Los diagramas de flujo de datos (DFD) consta de 3 niveles; el primer nivel es el de contexto, que es una visión mas general y donde se ve el sistema como un solo proceso, el segundo nivel es el superior, en este nivel se detalla los procesos generales que realiza el sistema. Finalmente esta el tercer nivel que es el de detalle, en este nivel se realiza un DFD de detalle por cada proceso descrito en el nivel superior, lo que genera subprocesos más específicos que realiza un proceso del nivel de detalle.

4.2.1 Diagrama de flujo de datos nivel contexto.

En este nivel existe sólo un proceso que representa al sistema en general, se enumera como el proceso cero. El objetivo de este nivel es introducir al lector a la interacción que se realiza entre el sistema y las entidades, mostrando los datos que entran y salen del sistema, sin mostrar los procesos que realiza el sistema.

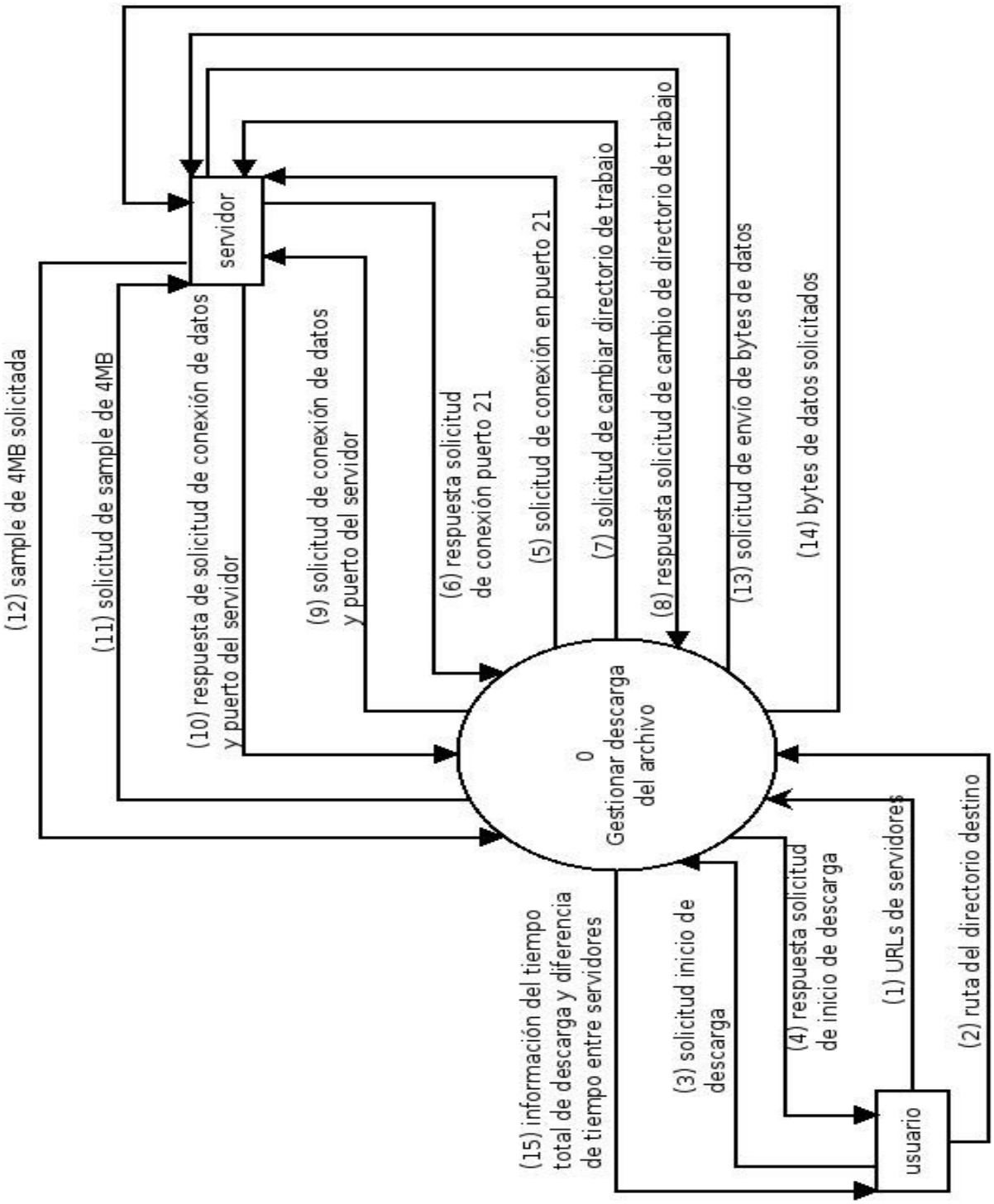

Ilustración 2: DFD - nivel contexto.

4.2.2 Diagrama de flujo de datos nivel superior.

En este nivel se especifica los principales procesos que realiza el sistema, estos procesos se enumeran desde el 1 en adelante. Nuevos flujos de datos surgen de la interacción entre los procesos del sistema.

- *1 Comprobar y gestionar datos ingresados:* Este proceso define el trabajo de la interfaz gráfica del usuario (GUI), es decir solicita los datos necesarios para realizar la descarga y los verifica.
- *2 Gestionar envío de comandos FTP*: Este proceso es el encargado de conectarse con el servidor abriendo un socket a través del puerto 21 para el envío de comandos tales como inicio de sesión, reanudación de descarga, recibir el puerto por el cual el servidor abre una conexión de datos.
- *3 Gestionar etapas y tiempos*: Este proceso es el encargado de coordinar todo el sistema, controla el avance de las etapas, realiza la distribución de los Bytes, define el tamaño y número de las etapas, y coordina los Thread que se ejecutan paralelamente.
- *4 Gestionar transferencia de datos*: Este proceso es el encargado de recibir los datos entrantes, controlar que se descargue solamente el rango de Bytes solicitado y almacenar los Bytes en el archivo.

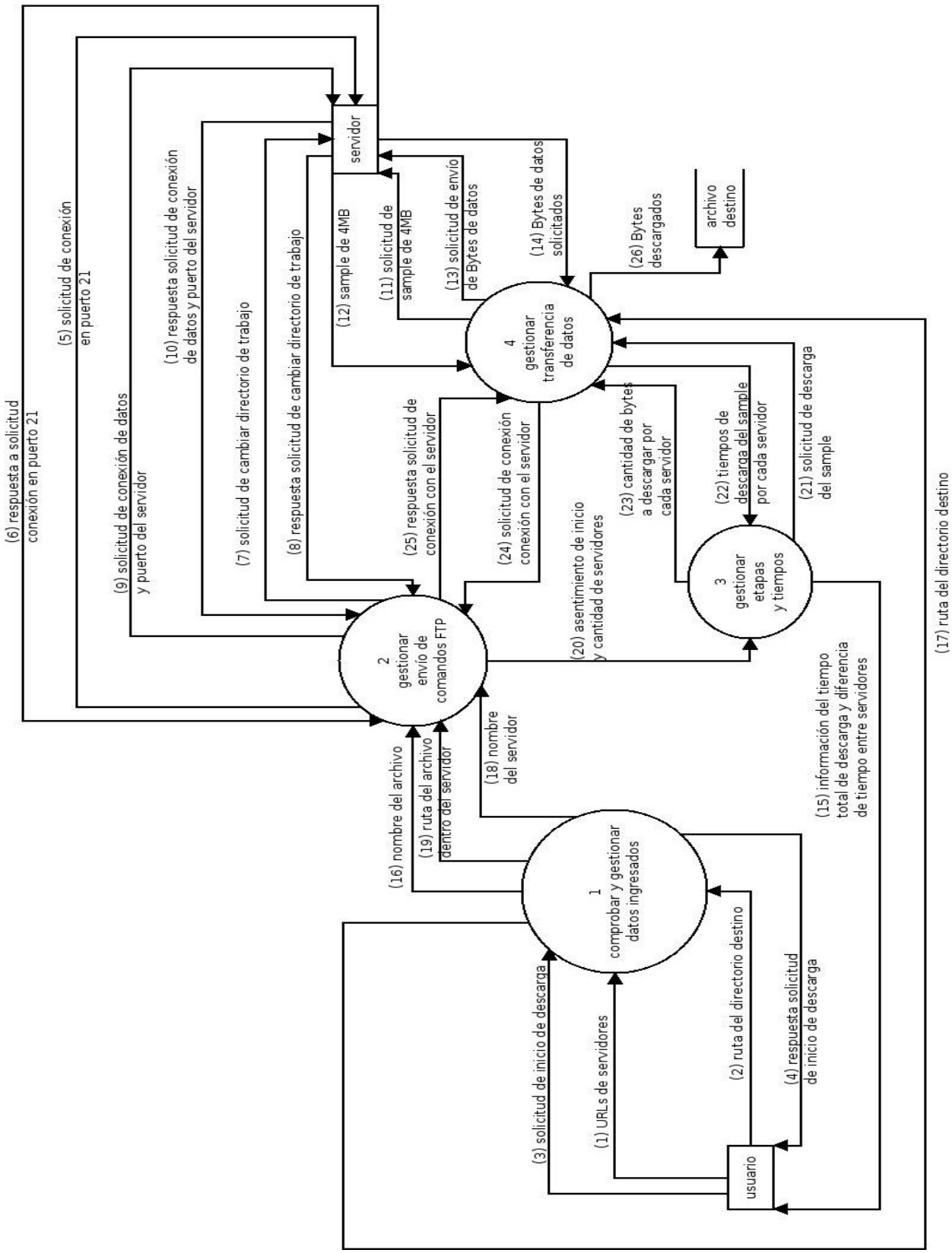

Ilustración 3: DFD - nivel superior.

4.2.3 Diagrama de flujo de datos nivel detalle.

En el nivel de detalle cada proceso del nivel superior se describe en un DFD del nivel de detalle, estos subprocesos son enumerados primero con el nivel a cual corresponden en el nivel superior y luego su enumeración. Las reglas son las mismas que el nivel superior, no deben crearse nuevos flujos que salen o ingresan a las entidades o procesos distintos al que se está describiendo.

- **Proceso 1**
  - 1.1 *Comprobar el ingreso de los datos requeridos*: En este proceso se validan las URLs, como también se verifica que le usuario haya ingresado un directorio de destino para el archivo.
  - 1.2 *Extraer datos desde las URLs y enviarlos*: En este proceso se identifican las partes de la URL que es el nombre del servidor, el puerto, el directorio y el archivo.

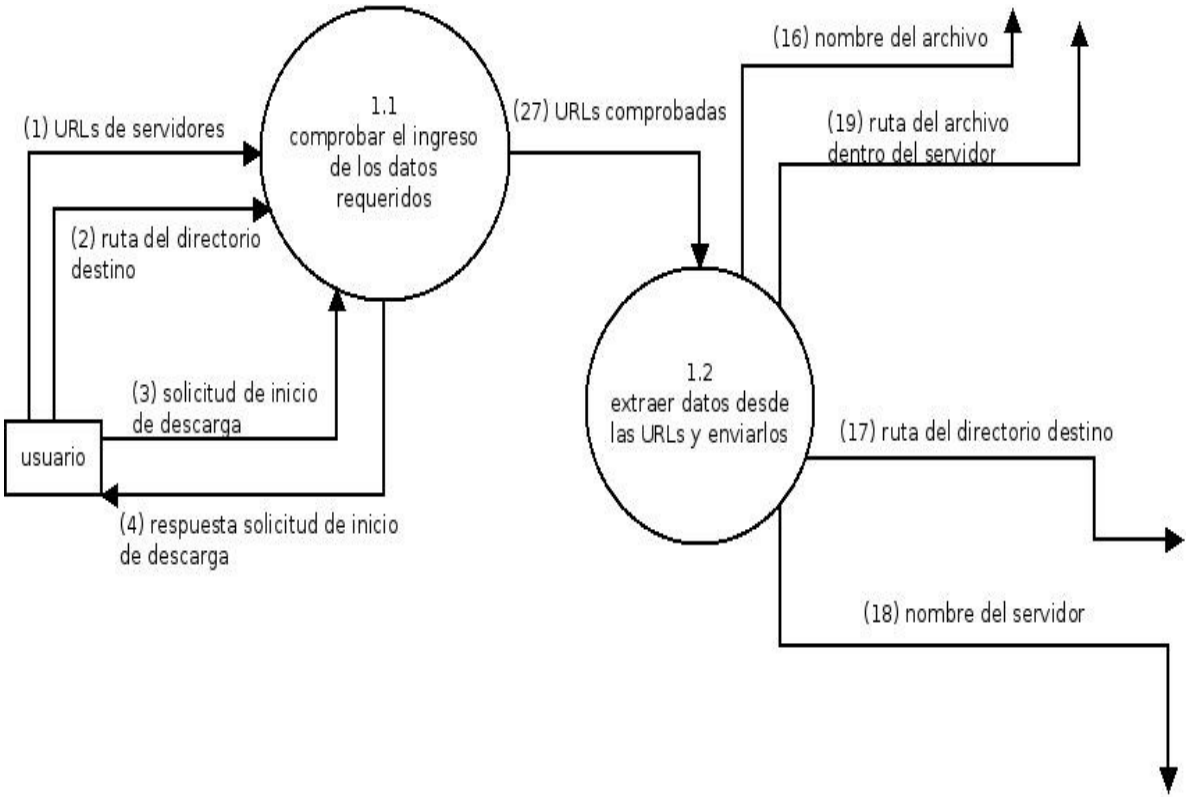

Ilustración 4: DFD - nivel detalle del proceso1.

- **Proceso 2**
  - 2.1 *Gestionar conexión de control:* Este proceso se encarga exclusivamente de abrir y cerrar los socket de java para el canal de control a través del puerto 21.
  - 2.2 *Gestionar conexión de datos:* Este proceso se encarga exclusivamente de abrir y cerrar los socket de java para el canal de datos según el puerto que le asigne el servidor.
  - 2.3 *Gestionar login y navegación en el servidor:* Este proceso se encarga de iniciar sesión en el servidor mediante un nombre de usuario y clave que son genéricos. También se encarga de realizar los cambios de directorio hacia la ruta donde se encuentra el archivo a descargar dentro del servidor.

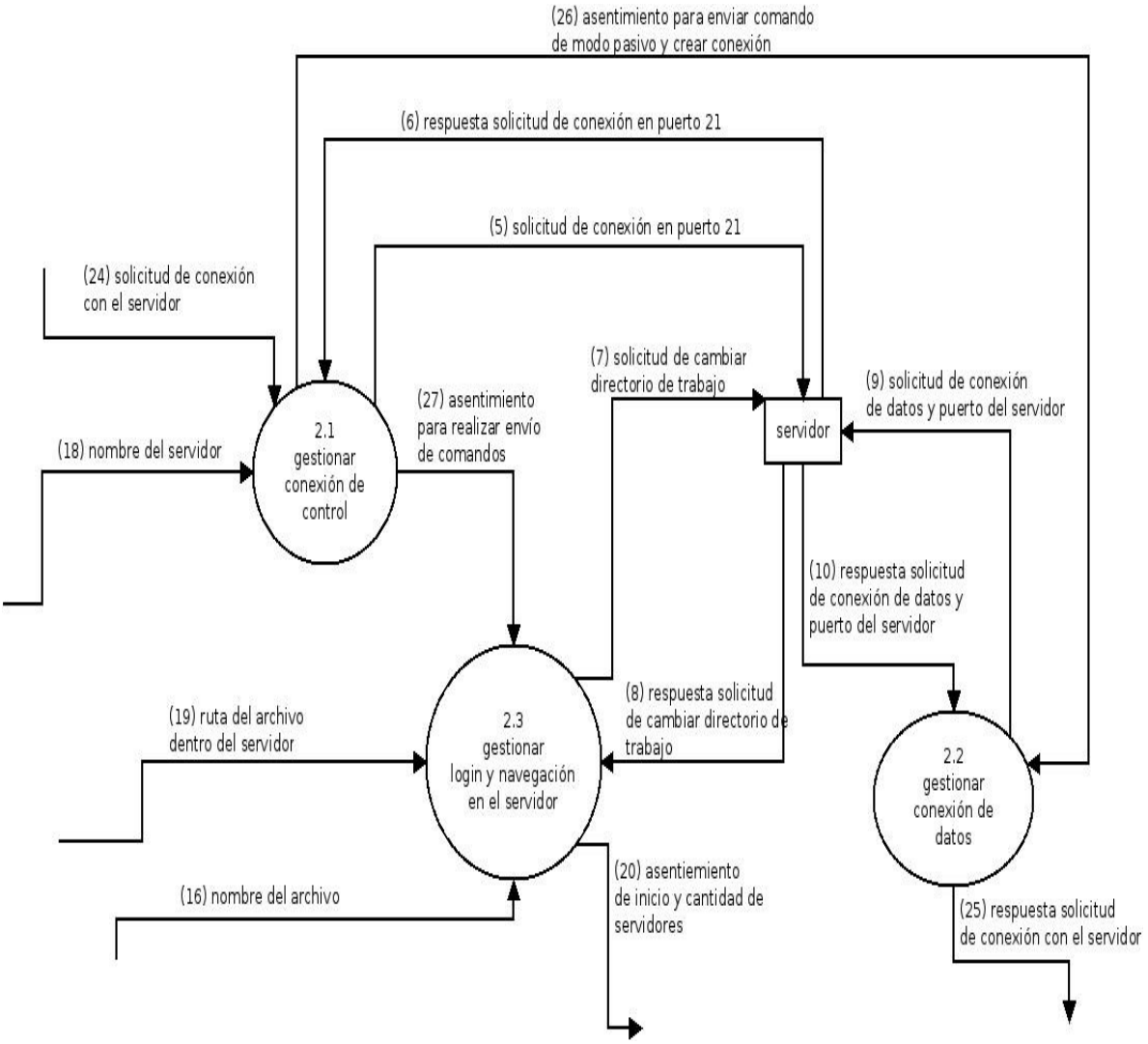

Ilustración 5: DFD - nivel detalle del proceso 2.

- Proceso 3
  - 3.2 *Calcular la cantidad de Bytes de cada servidor en la etapa:* Este proceso recibe los tiempos de descarga de las muestras para realizar una asignación proporcional de Bytes según la tasa de transferencia de cada servidor.
  - 3.3 *calcular n° de etapas y su tamaño:* Este proceso divide el tamaño total del archivo según el número de etapas elegidas.
  - 3.4 *Cronometrar tiempo total de ejecución y tiempo de descarga en cada servidor:* Calcula el tiempo que le toma a cada servidor descargar una muestra o su parte en la etapa, como también el tiempo en descargar el archivo completo.

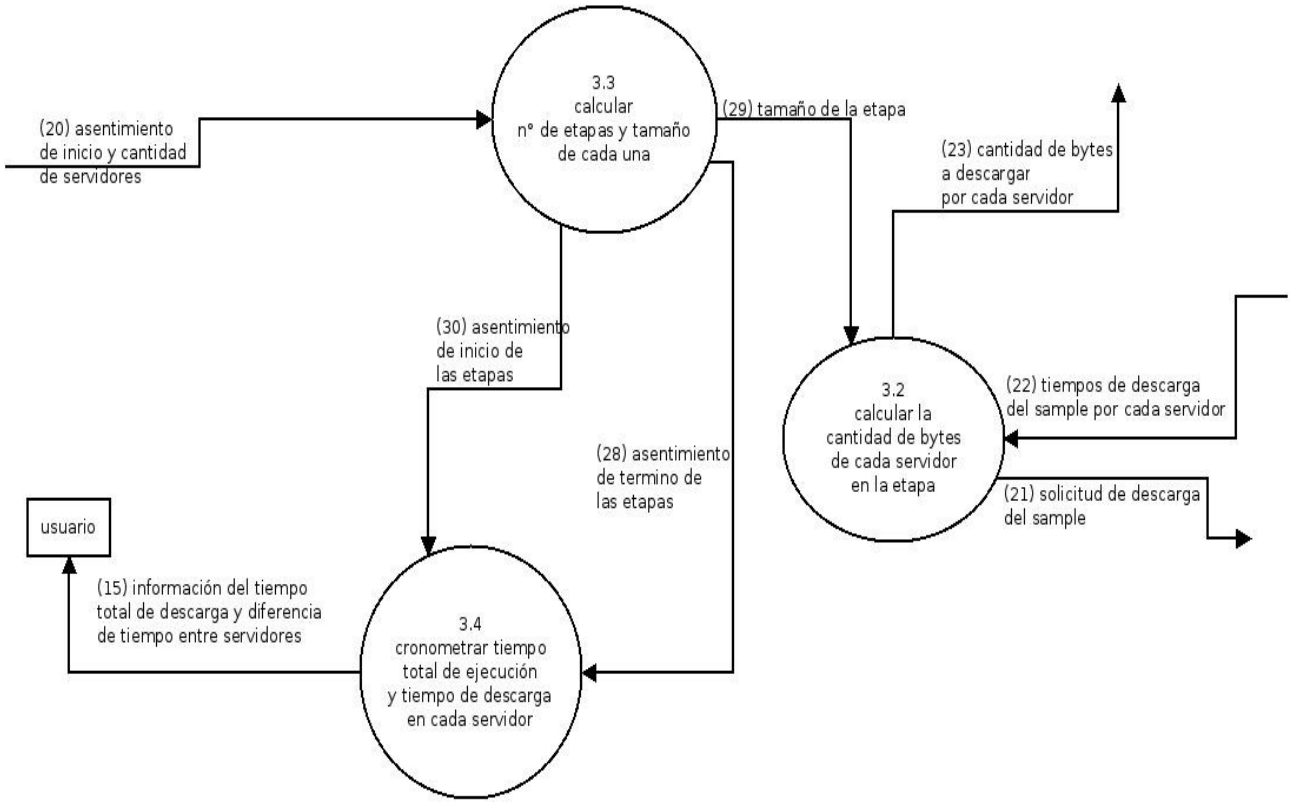

Ilustración 6: DFD - nivel detalle del proceso 3.

- Proceso 4
  - 4.1 *Gestionar medición de ancho de banda de cada servidor:* Este proceso realiza la descarga de la muestra para cada servidor, tomando el tiempo que se demora cada servidor.
  - 4.2 *Gestionar recepción de datos:* Este proceso se encarga de recibir los datos entrantes a través de la conexión de datos y los almacena en el archivo destino.
  - 4.3 *Solicitar conexión con el servidor:* Este proceso se encarga de solicitar una conexión de datos o de control con el servidor seleccionado.
  - 4.4 *Solicitar una cantidad de Bytes a cada servidor:* Este proceso se encarga del buffer de datos, el cual espera a que se llene el buffer con la cantidad de Bytes desde el servidor y luego vuelca esos Bytes al disco para vaciar el buffer y recepcionar más Bytes.

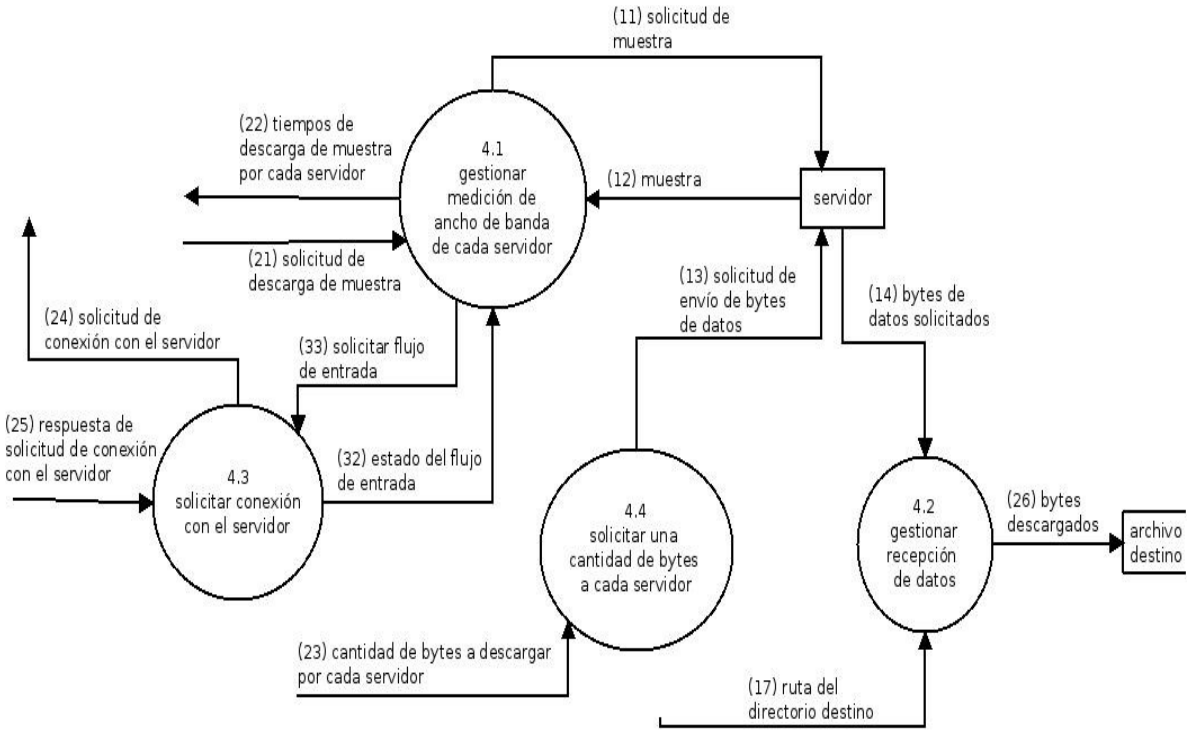

Ilustración 7: DFD - nivel detalle del proceso 4.

5 EXPERIMENTOS.

En este capítulo se compara las técnicas propuestas de Descarga en Paralelo Adaptativa contra una técnica adversaria. La técnica adversaria es la Descarga en Paralelo No Adaptativa (DPNA) que es aquella técnica que no realiza mediciones de ancho banda y considera a todos los servidores como iguales.

Las métricas que se evalúan para comparar estas técnicas son las siguientes:

- *Tiempo Total:* Es el tiempo total de descarga obtenido desde el instante en que se inicia la sesión de descarga hasta que termina la descarga del archivo completo.
- *Desbalance de Carga:* Es la Diferencia de Tiempo de Descarga Total obtenida por el servidor más rápido con respecto al servidor más lento. El desbalance es menor mientras mas cercano a cero es este valor.

La red donde se realizaron estos experimentos es una red de 100Mb/s. Respecto al software del servidor FTP, se optó por usar el programa FileZilla server versión 0.9.43 [12], ya que permite cambiar en tiempo real el ancho de banda disponible por conexión TCP.

El total de descargas para realizar estos experimentos fue de 82 de los cuales 18 fueron para el tamaño de pruebas, 32 para el escenario estático y 32 para el escenario dinámico.

5.1 Tamaño de las Muestras.

Al inicio de una descarga, el usuario desconoce la tasa de descarga que pueda obtener desde cada servidor seleccionado. Para realizar la estimación de tasa, la aplicación descarga una muestra desde cada servidor y con ella determina esta tasa. Si se elige una muestra de corto tamaño, si bien podemos tener una respuesta con prontitud, es posible que obtengamos una mala estimación de la tasa de descarga. Por el contrario si esta es muy grande, puede darnos una buena estimación del ancho de banda pero incurriendo en un mayor tiempo de medición.

A continuación realizamos una descarga con cada una de las variaciones de la técnica desde dos servidores, usando distintos tamaños de muestras. El objetivo de este experimento es obtener el tamaño de muestra con la mejor relación entre tiempo total de descarga y el desbalance de carga.

Ambos servidores se configuraron para ofrecer una tasa de transferencia constante de 5MB/s cada uno durante toda la descarga. El archivo a descargar tiene un tamaño de 1.1GB.

5.1.1 Tamaño vs Tiempo DPA-1.

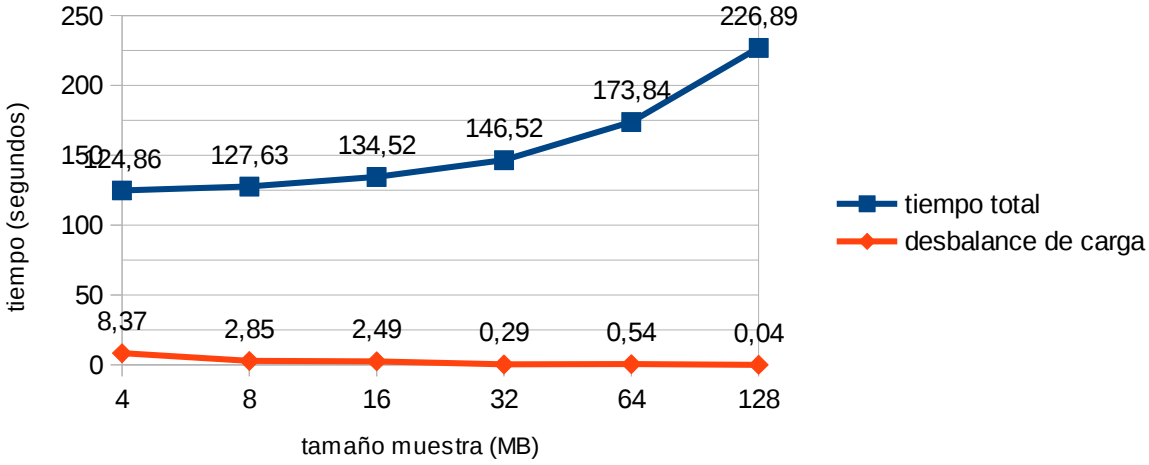

Ilustración 8: tamaño vs tiempo DPA-1

En este gráfico se puede observar que a los 8MB el desbalance de carga disminuye en gran cantidad y se mantiene ligeramente mas abajo a las 16MB, pero respecto al tiempo total el aumento es mayor.

En conclusión para este gráfico la mejor opción es la muestra de 8MB porque en 16MB el desbalance mejora 0,36 segundos y el tiempo total aumenta 6,89 segundos en relación a los 8MB lo cual la mejora de desbalance no compensa el aumento del tiempo total.

5.1.2 Tamaño vs Tiempo DPA -2.

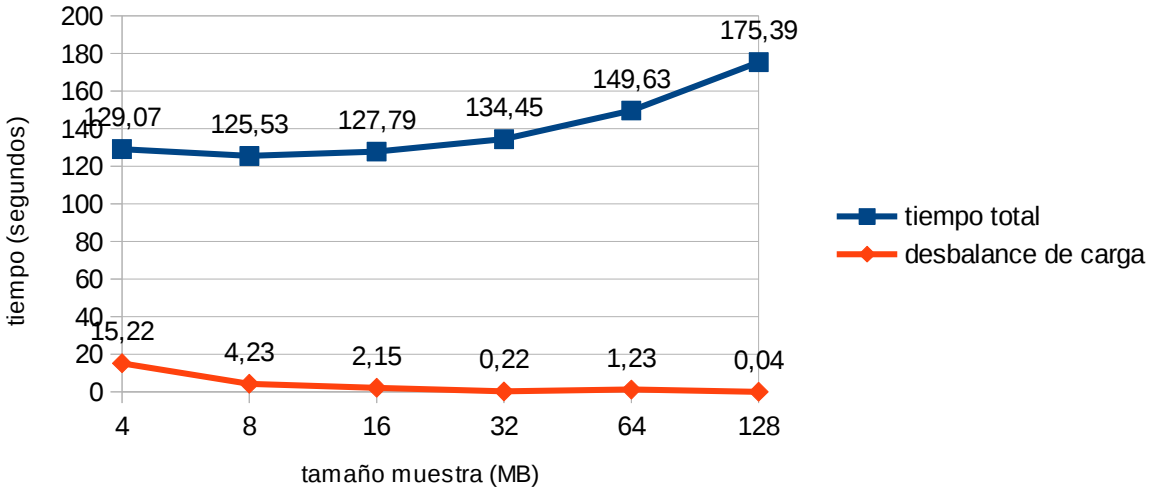

Ilustración 9: tamaño vs tiempo DPA-2

En este gráfico también se aprecia una gran disminución a los 8MB y a los 16MB también una disminución casi del 50% con respecto a la de 8MB, en cuanto a los tiempo totales se observa una inflexión a los 8MB, pero a los 16MB sigue manteniendo un tiempo menor al de 4MB.

En conclusión para este gráfico una buena opción es la muestra de 16MB.

5.1.3 Tamaño vs Tiempo DPA -3.

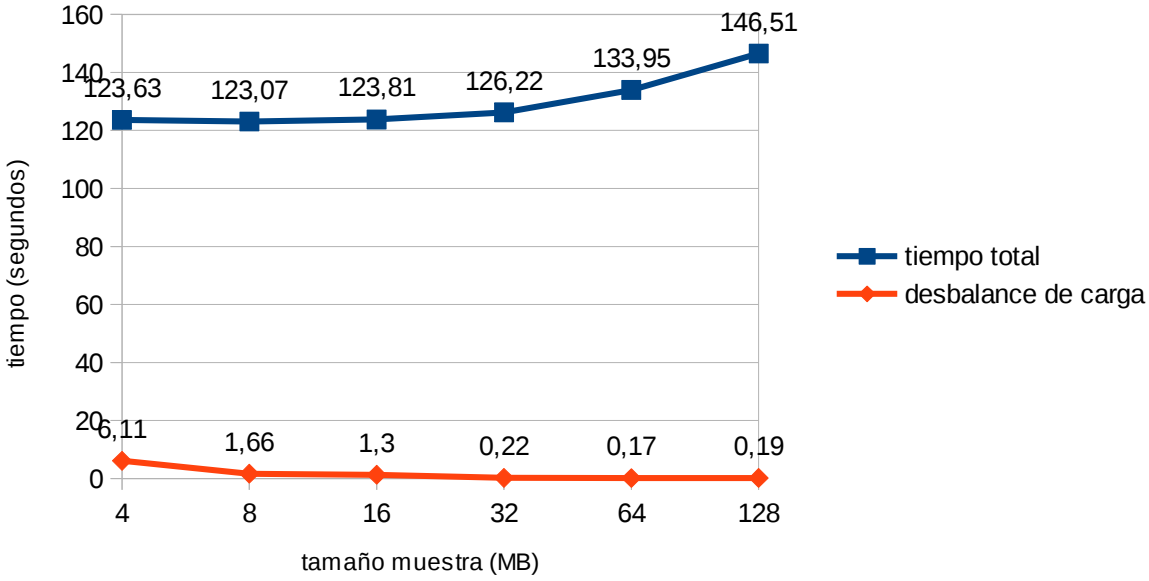

Ilustración 10: tamaño vs tiempo DPA-3

Esta variante muestra una muy buena curva en ambas métricas, respecto al tiempo total se mantiene ligeramente hasta los 16MB en donde comienza a crecer la curva. Por otro lado en el desbalance de carga disminuye notablemente a los 8MB peor disminuye ligeramente a los 16MB.

En conclusión para esta gráfica 16MB es buena opción ya que se sacrifica muy poco en el tiempo total con respecto al desbalance de carga.

5.1.4 Conclusión del tamaño de muestra.

En los 3 gráficos los 8MB y 16MB muestran un buen resultado, pero para este proyecto se requiere un menor desbalance de carga, por lo que se ha seleccionado los 16MB como tamaño de muestra para los experimentos de desempeño de la técnica.

5.2 Escenario estático.

En este escenario se dispone de 2 servidores. Cada servidor es configurado para ofrecer una tasa de transferencia constante durante toda la descarga, pero a cada servidor se le asigna una tasa distinta.

Para este escenario se realizan experimentos aumentando el número de etapas, con el fin de conocer si a mayor número de etapas aumenta o disminuye el desbalance de carga y el tiempo total de descarga.

5.2.1 Gráficos de pruebas estáticas.

Experimento N°1

El primer servidor es configurado con una tasa de 1.5MB/s y el segundo servidor es configurado con una tasa de 2.5MB/s.

- **Tiempos para la Descarga Paralela No Adaptativa.**

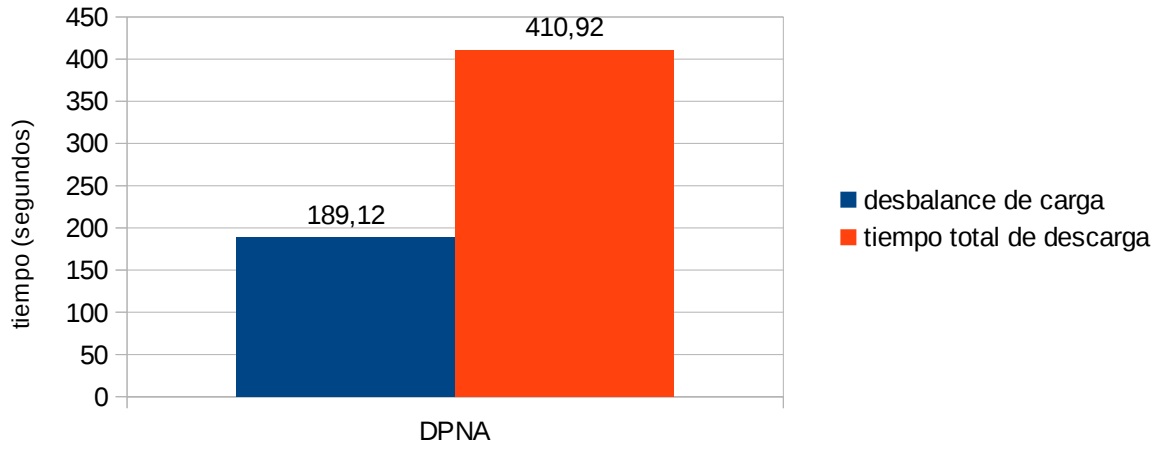

Ilustración 11: Escenario estático n°1 - tiempos DPNA

- **Gráfico tiempos totales de descarga de todas las variantes.**

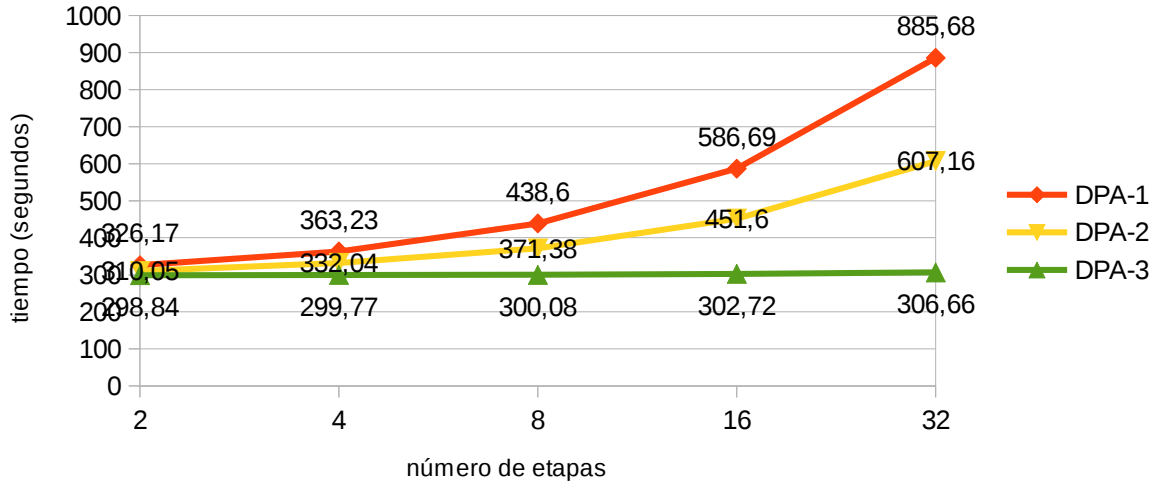

Ilustración 12: Escenario estático n°1 - tiempos total técnicas propuestas

Los resultados mostrados en este gráfico se denota la superioridad de la variante DPA-3 en donde se observa casi constante al aumento de etapas. Las demás variantes se muestran crecientes por el factor de las muestras. Estas muestras como se realizan en forma secuencial y no paralela, aumentan el tiempo total y entre mas etapas mas muestras y menos descarga paralela.

- **Gráfico desbalance de carga de todas las variantes.**

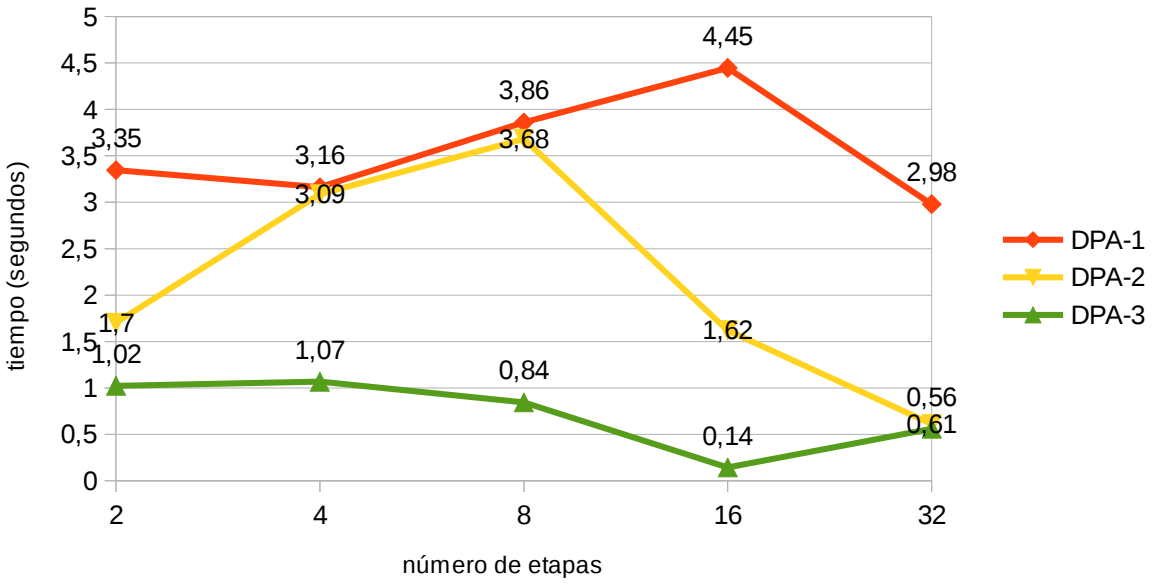

Ilustración 13: Escenario estático n°1 – desbalance de carga técnicas propuestas

En este gráfico se observa que en el desbalance se ven variaciones abruptas en algunos números de etapas. Estas variaciones tienen un rango de 0 a 4 segundos, pero si se observa el gráfico en un mayor rango se puede ver que tiende a ser constante y sólo tiene pequeñas variaciones. La variante DPA-3 tiende a ser constante pero adquiere una inflexión cuando el número de etapas es 16. Por otro lado la variante DPA-1 tiene su punto más alto cuando el número de etapas es 16.

Con este gráfico podemos concluir que las descargas paralelas adaptativas cuando el escenario es estático no se ve disminución en el desbalance entre más etapas, sólo pequeñas variaciones de segundos.

Experimento N°2

El primer servidor es configurado con una tasa de 2MB/s y el segundo servidor es configurado con una tasa de 5MB/s.

- **Tiempos para la Descarga Paralela No Adaptativa.**

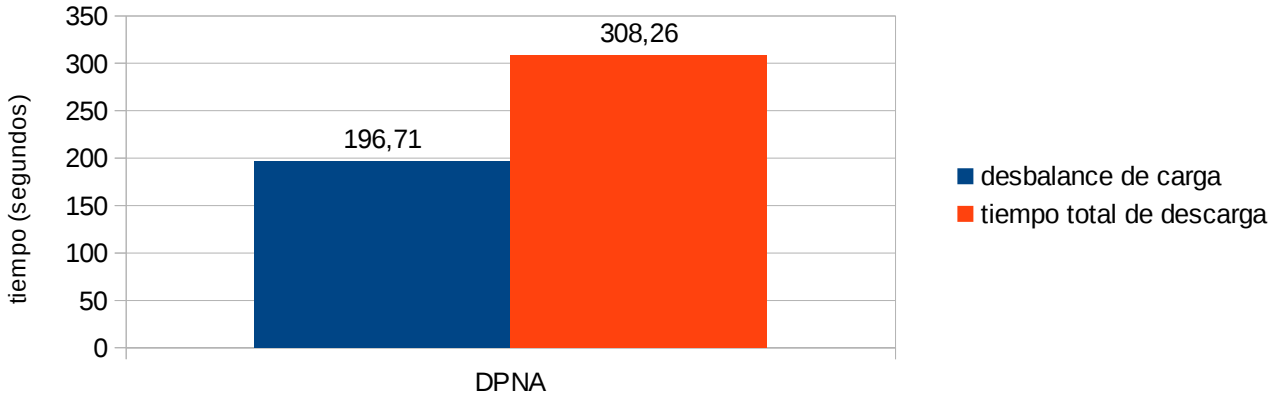

Ilustración 14: Escenario estático n°2 - tiempos DPNA

- **Gráfico tiempos totales de descarga de todas las variantes.**

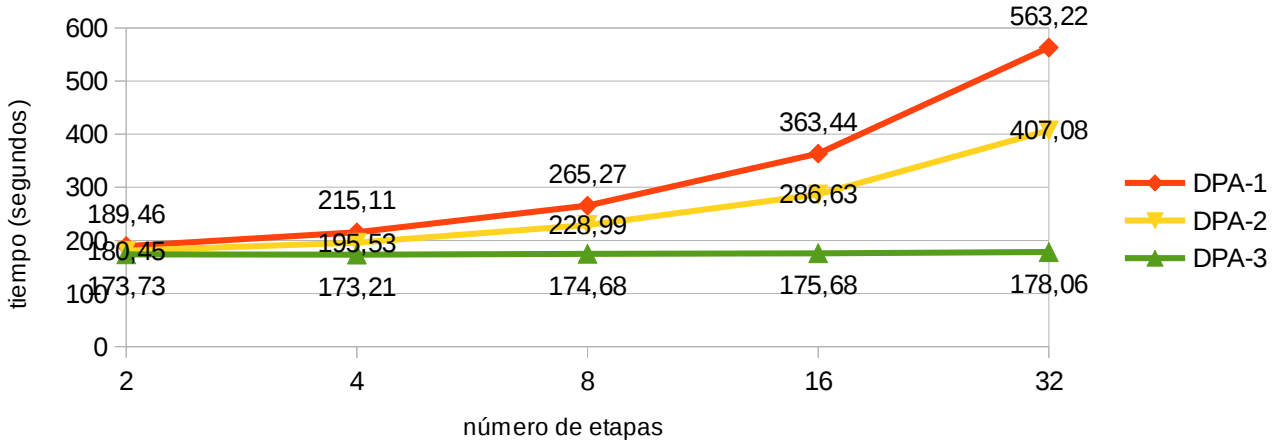

Ilustración 15: Escenario estático N°2 - tiempos total técnicas propuestas

Este gráfico de tiempos total se aprecia el mismo comportamiento que en el Experimento N°1, la única diferencia es que los tiempos totales son menores debido a la configuración inicial de la tasa de transferencia de los servidores.

- **Gráfico desbalance de carga de todas las variantes.**

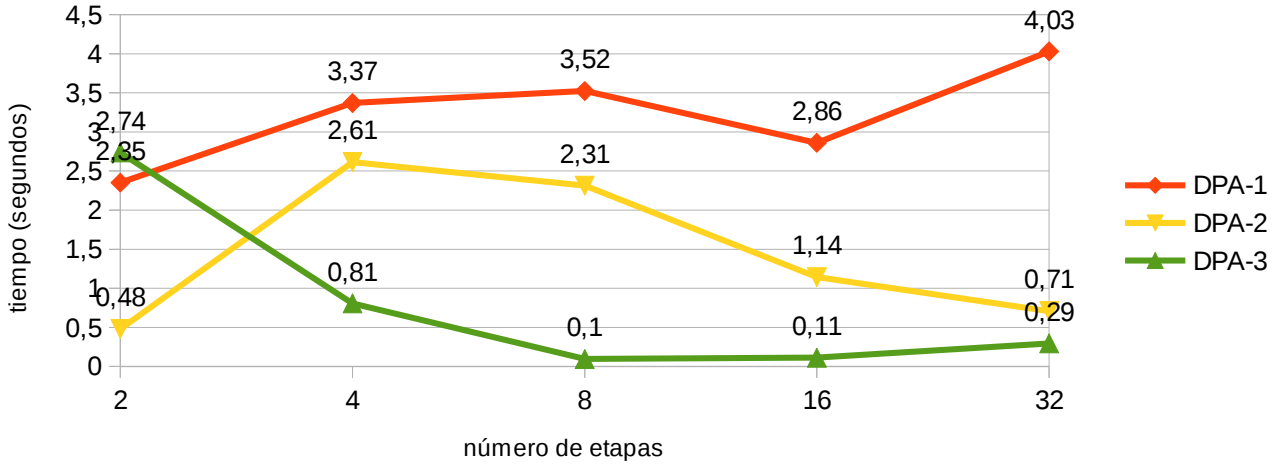

Ilustración 16: Escenario estático n°2 - desbalance de carga técnicas propuestas

En el gráfico del desbalance de carga los desbalances varían en un rango de 0 a 4 segundos igual que en el experimento N°1, con respecto a estos dos experimentos podemos concluir que en el escenario estático las DPA en todas sus variantes su desbalance no va mejorando conforme se aumenta el número de etapas.

5.3 Escenario dinámico.

En este escenario los servidores tienen una tasa de transferencia variable, es decir, se cambia la tasa de transferencia de un solo servidor en un punto del transcurso de la descarga. El objetivo de este escenario es ver cómo la descarga paralela adaptativa va balanceando la cantidad de Bytes a descargar por cada servidor en cada etapa. Se dispone de 2 servidores. Cada servidor es configurado para ofrecer una tasa de transferencia constante durante toda la descarga, pero a cada servidor se le asigna una tasa distinta. Ambos servidores son configurados a una tasa de 5MB/s y al transcurrir aproximadamente el 50% de la descarga sólo a un servidor se le disminuye su tasa a 1MB/s.

5.3.1 Gráficos de pruebas dinámicas.

Experimento N°3.

Ambos servidores inicialmente están configurados con una tasa de 5MB/s, luego la tasa de transferencia es disminuida en un servidor a 1MB/s.

- **Tiempos para la Descarga Paralela No Adaptativa.**

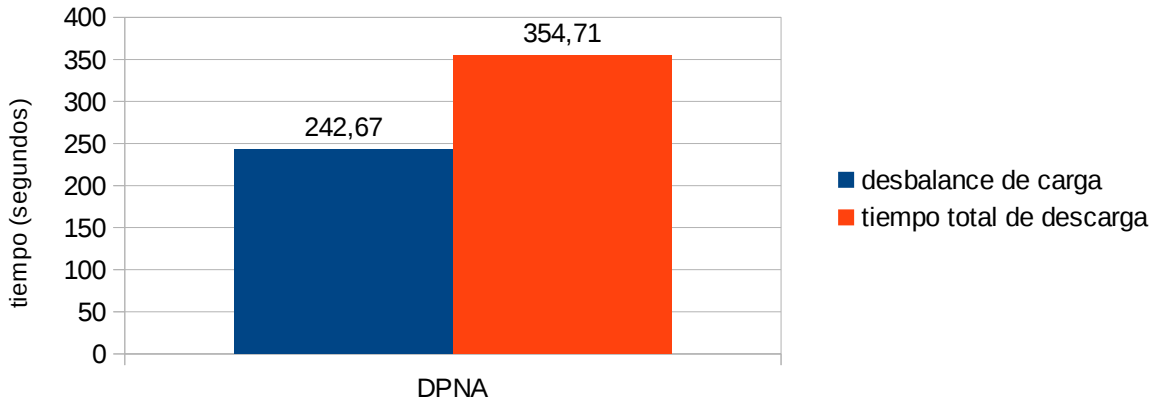

Ilustración 17: Escenario dinámico n°1 - tiempos técnica DPNA

- **Tiempos totales de descarga de todas las variantes.**

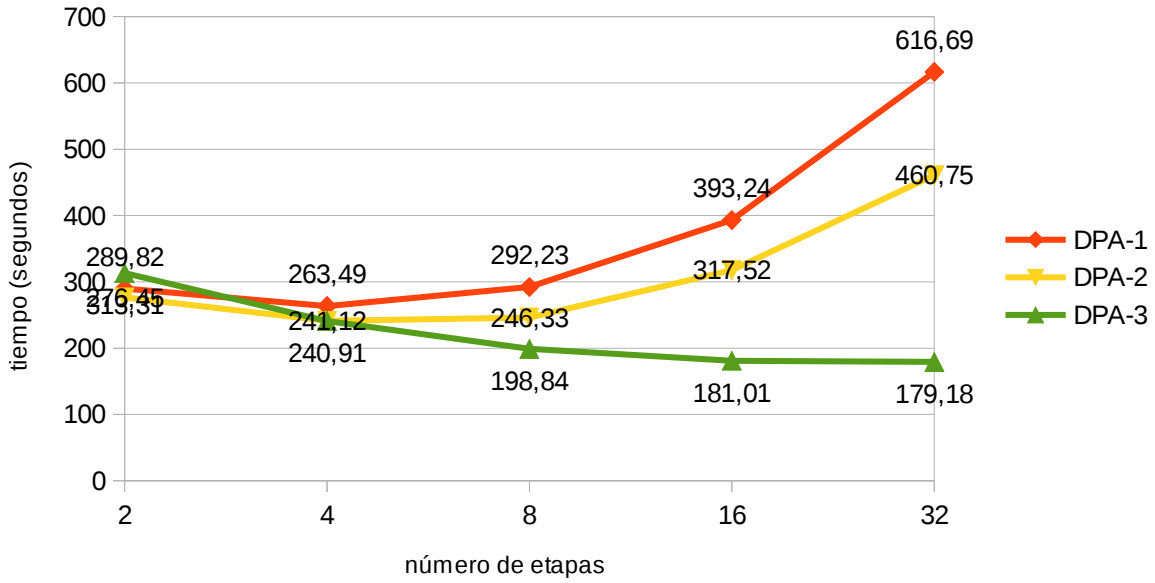

Ilustración 18: Escenario dinámico n°1 - tiempos total técnicas propuestas

Los resultados que muestra este gráfico una vez mas muestra la superioridad de la variante DPA-3, donde reacciona mucho mejor ante las variantes, es mas su tiempo total va disminuyendo entre mas etapas. Gran parte

de esta disminución se debe a que la variante DPA-3 no realiza descargas de muestras entre cada etapa lo que significa que sólo sucede una descarga secuencial al comienzo de la descarga.

- **Gráfico desbalance de carga.**

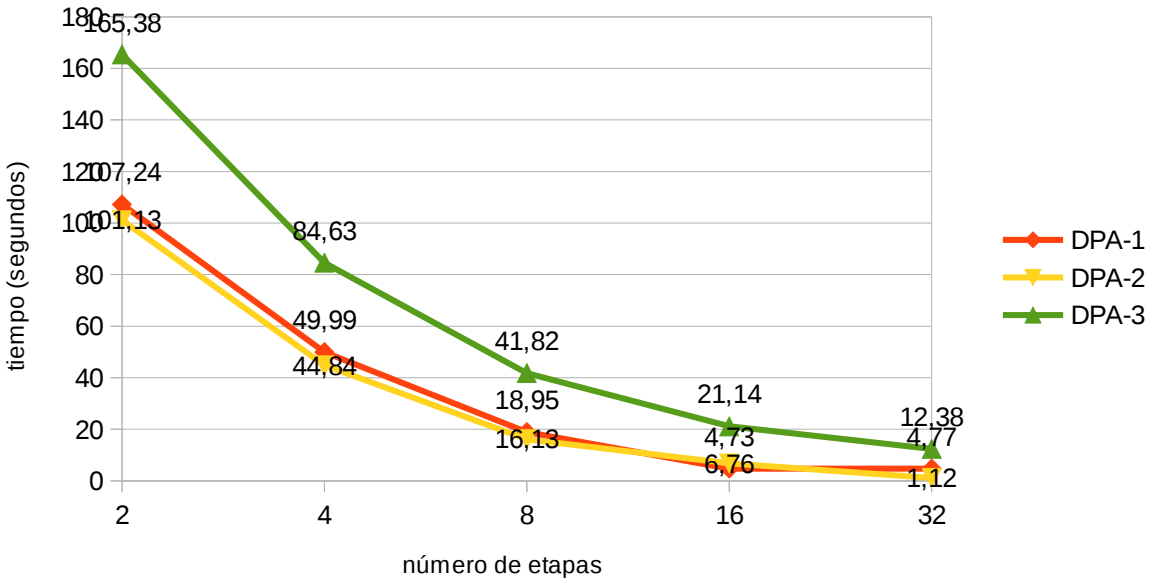

Ilustración 19: Escenario dinámico n°1 - Desbalance de carga técnicas propuestas

En este gráfico vemos que el desbalance en la variante DPA-3 es el más alto de las tres variantes, la explicación a este es el siguiente; si el cambio de tasa se produce en la mitad de la etapa, las variantes DPA-1 y DPA-2 lo corrigen en la medición de la siguiente etapa, pero la variante DPA-3 en la siguiente etapa mejora el desbalance pero no completamente, después en la subsiguiente el desbalance se disminuye completamente. Es por esta razón que entre mas etapas es menor el desbalance en la variante DPA-3 por el hecho que entre más etapas, menos es el tiempo para pasar a la siguiente y por ende menos tiempo en esa etapa menos desbalanceada.

Experimento N°4

Ambos servidores inicialmente están configurados con una tasa de 2.5MB/s y 5MB/s respectivamente, luego la tasa de transferencia es aumentada en el servidor 1 a 3.5MB/s.

- **Tiempos para la Descarga Paralela No Adaptativa.**

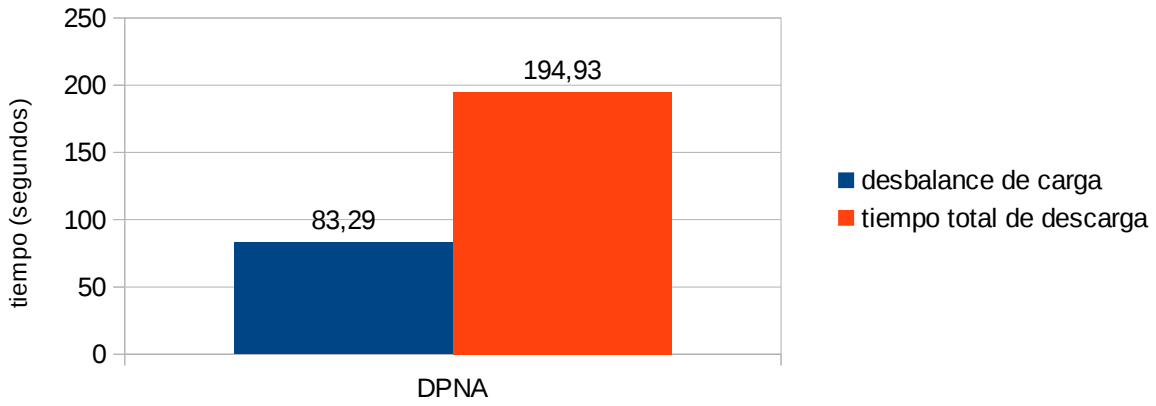

Ilustración 20: Escenario dinámico n°2 - tiempos técnica DPNA

- **Tiempos totales de descarga de todas las variantes.**

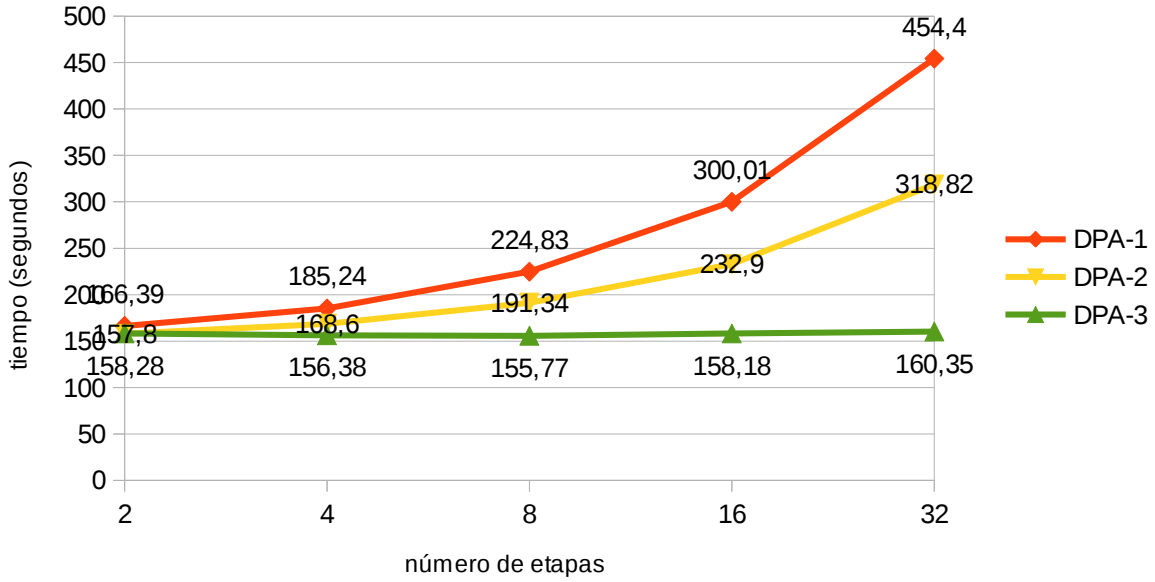

Ilustración 21: Escenario dinámico n°2 - tiempos total técnicas propuestas

Los resultados que muestra este gráfico una vez más muestra la superioridad de la variante DPA-3, Pero esta vez tiene una pequeña variación a medida que aumentan las etapas. En este experimento en vez de disminuir la tasa de un servidor se aumentó, y no genera una curva hacia abajo, se mantiene constante pero tiende a crecer la

curva. Además mencionar que el el aumento de la tasa de transferencia no fue tan alta como en el experimento N°3 lo que nos da a entender que entre más grande es la variación, más disminuye la curva en la variante DPA-3.

Entre más grande es el cambio de la tasa de transferencia mayor es el tiempo total, pero la variante DPA-3 responde muy bien ante los grandes cambios del ancho de banda de los servidores.

- **Gráfico desbalance de carga.**

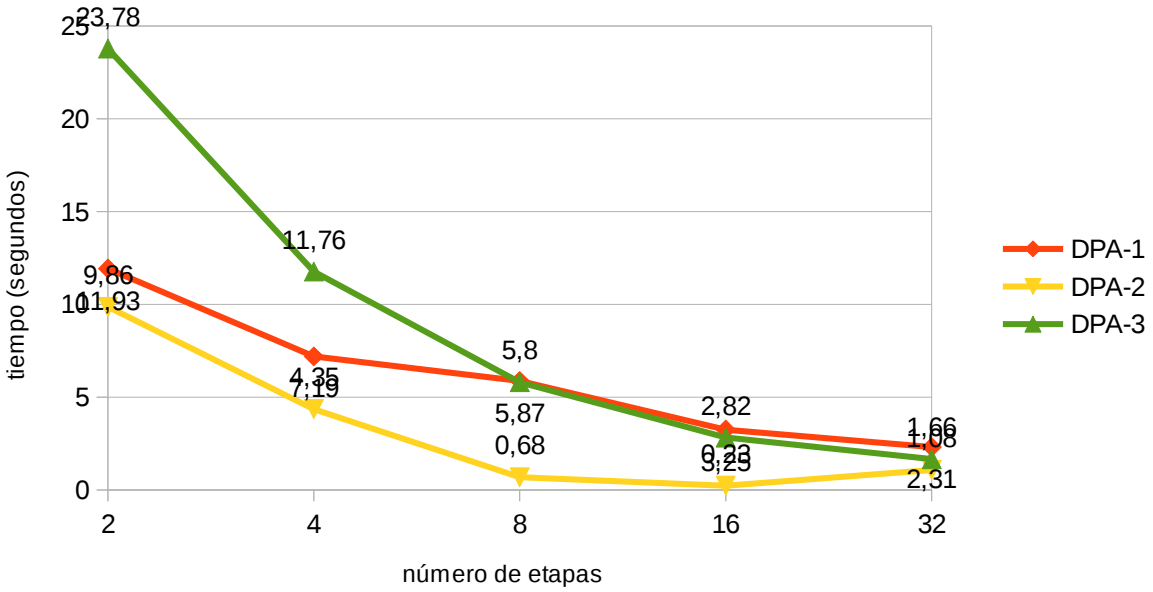

Ilustración 22: Escenario dinámico n°2 - Desbalance de carga técnicas propuestas

En este gráfico se obtiene que el desbalance de carga también tiende a bajar entre más etapas tenga. Pero se observan unas pequeñas inflexiones cuando el número de etapas es 8. Otro punto a observar es que los tiempos del desbalance de carga son mucho menores que en el experimento N°3, esto se debe a que el cambio de la tasa de transmisión es menor, lo que se ve reflejado en el gráfico. Pero sin importar qué grande sea el cambio de la tasa de transferencia entre más etapas el desbalance disminuye notablemente.

5.3.2 Conclusiones de experimentos.

En estos experimentos se demostró que el mejor tamaño para las pruebas es de 16MB que compensa bien el tiempo total con el desbalance de carga.

La pregunta ¿que variante tiene mejor desempeño? queda respondida al ver que tanto en el escenario estático como en el escenario dinámico la variante DPA-3 es la mejor, ya que esta variante sólo realiza mediciones al inicio. Gracias a estos experimentos se demostró que la estimación del ancho de banda con muestras disminuye

el desbalance de carga, pero a cambio de un aumento en el tiempo total de descarga de manera inversamente proporcional.

El objetivo de crear una técnica que mejore las Descargas Paralelas No Adaptativas, se ve cumplido ya que en ambos escenarios la variante DPA-3 supera por mucho a los resultados obtenidos por esta.

6 CONCLUSIONES.

Los objetivos del proyecto fueron alcanzados completamente al finalizar este proyecto. Las descargas paralelas adaptativas resultaron ser mucho mas rápidas que las descargas en paralelo no adaptativas. Por esa razón la aplicación resulta ser muy beneficiosa al ir adaptándose a las variaciones del ancho de banda.

La utilización de la programación orientada a objeto resultó ser de muy gran ayuda para poder separar el sistema en objetos y en especial por el uso de hilos, que es el punto fuerte en este proyecto, por el hecho de efectuar descargas paralelas.

El lenguaje de programación Java sin duda tiene un gran potencial, no solo por la gran cantidad de documentación que existe, también por una sintaxis relativamente simple que permite una mayor fluidez en la codificación. Pero una de las grandes ventajas de Java es el ser una herramienta multiplataforma que permite ejecutar la aplicación en muchas plataformas sin modificar el código, permitiendo una gran acogida por los usuarios.

La planificación inicial del proyecto estuvo bien organizada ya que los plazos entre cada fase fueron muy parecidos a los que se realizaron, permitiendo entregar versiones funcionales en cada etapa del proyecto. Aunque la estimación del tamaño del proyecto no resultó ser similar al obtenido realmente al finalizar el proyecto.

En lo que respecta al área académica, el proyecto fue un gran hito que me permitió conocer sobre la transmisión de datos a través de sockets de Internet, como también conocer como funciona el protocolo FTP para compartir datos en Internet. Pero lo mas importante es poder realizar una aplicación que permita mejorar los tiempos de descargas, una función muy importante en lo que respecta compartir archivos hoy en día en Internet, y dar camino al a posibilidad de mejorar la aplicación para otros protocolos.

Personalmente el proyecto me entrego nuevas habilidades para enfrentar desafíos mas grandes y motivación para crear proyectos independientes enfocados en el área de estudio de este producto.

7 BIBLIOGRAFIA.

[1] RFC 959, “Protocolo de Transferencia de Archivos (FTP)”.

[2] RFC 21616 “Protocolo de Transferencia de Hipertexto (HTTP/1.1)”

[3] Rodriguez P. y Biersack W. “Dynamic Parallel Access to Replicated Content in the Internet”. IEEE/ACM Transaction on Networking, 10(4), pp 455-465, Agosto 2002.

[4] Al-Jaroodi J. Y Mohamed N. “DDFTP: Dual-Direction FTP”, In the Proc. Of the 11th IEEE/ACM Symposium on Cluster, Cloud and Grid Computing (CCGrid'11), pp504-513, May 2011.

[5] Funasaka J., Nakawaki N., Ishida K. y Amano K. “A Parallel Downloading Method of Coping with Variable Bandwidth”. ICDCS Workshops 2003, pp 14-19, Mayo 2003

[6] J. Byers, M. Luby, and M. Mitzenmacher, “Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads”, In INFOCOM 99, April 1999.

[7] S. Sohail, S.-K. Jha, S. Kanhere, C-T. Chou, "QoS Driven Parallelization of Resources to Reduce File Download Delay," IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 10, pp. 1204-1215, October, 2006

[8] Allcock W., Bresnahan J. Kettimuthu R., Link M., Dumitrescu C., Raicu I., y Foster I.. "The Globus striped GridFTP framework and server." In Proceedings of the 2005 ACM/IEEE conference on Supercomputing, p. 54. IEEE Computer Society, 2005.

[9] Secure Hash Algoritm, <http://es.wikipedia.org/wiki/Secure_Hash_Algorithm>

[10] Oracle, “Maquina Virtual de Java” Versión 7 update 45, **Fuente:** http://www.java.com/es/download/, Fecha de versión: 15 de octubre de 2013.

[11] Kurose J. Et al. “Redes de Computadoras. Un Enfoque Descendente”, 5ta Ed. Editorial Pearson, 2010.

[12] Tim Kosse, “FileZilla” versión 0.9.43, Fuente: https://filezilla-project.org/download.php?type=server Fecha de versión: 02 enero 2014

8 ANEXO: DISEÑO

8.1 Árbol de descomposición funcional.

Este diagrama esta relacionado con el DFD que se presentó en el capitulo anterior, en este diagrama se especifica los módulos que posee el sistema asociados a los procesos del nivel superior del DFD.

Cada módulo realiza una función dentro del sistema y muchos procesos pueden ocupar un mismo módulo a modo de reutilización de código. Cada módulo es un método dentro del código de la aplicación.

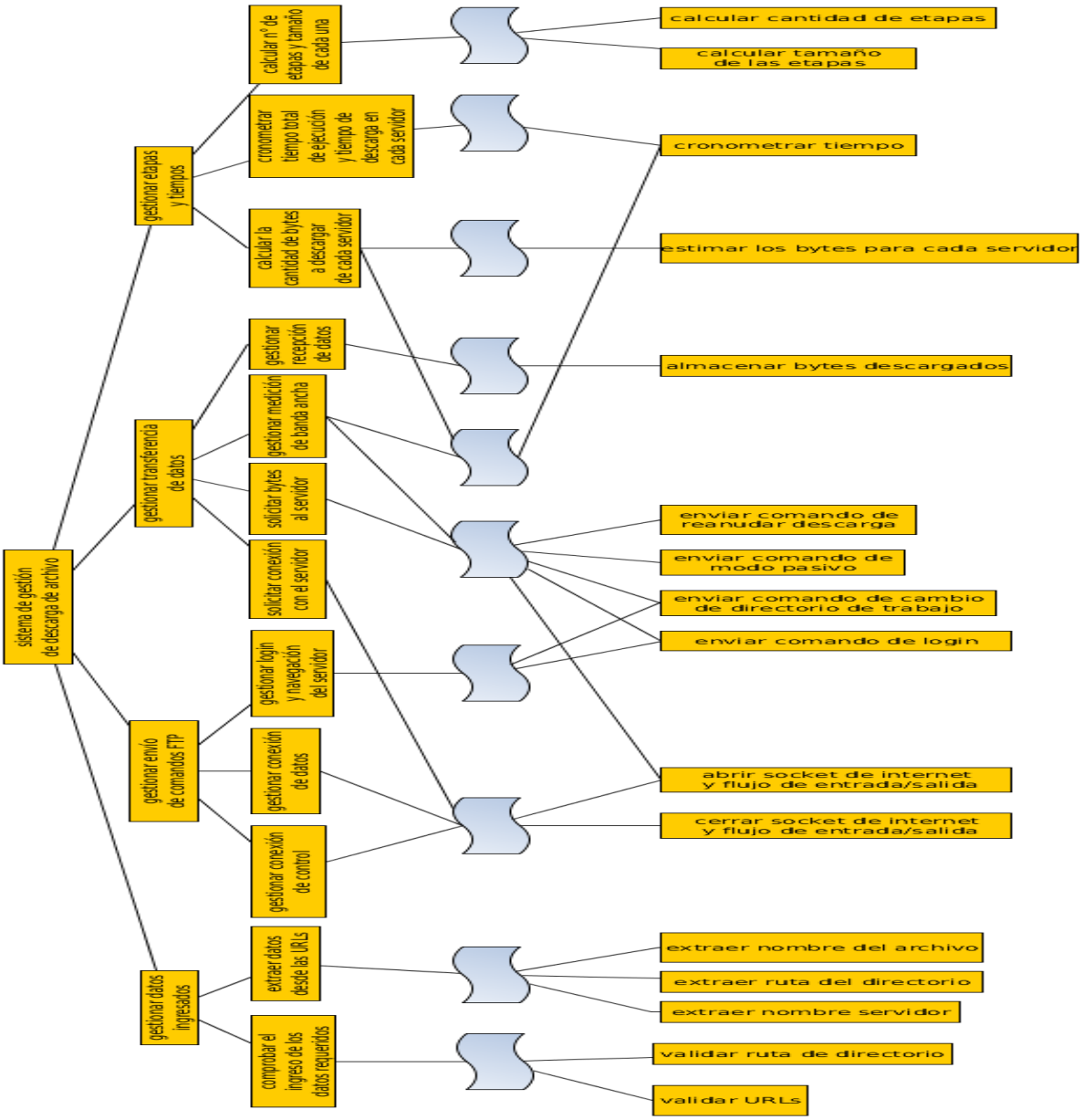

Ilustración 23: Árbol de descomposición funcional

8.2 Diseño de interfaz y navegación.

8.2.1 Diseño de la Interfaz gráfica del usuario.

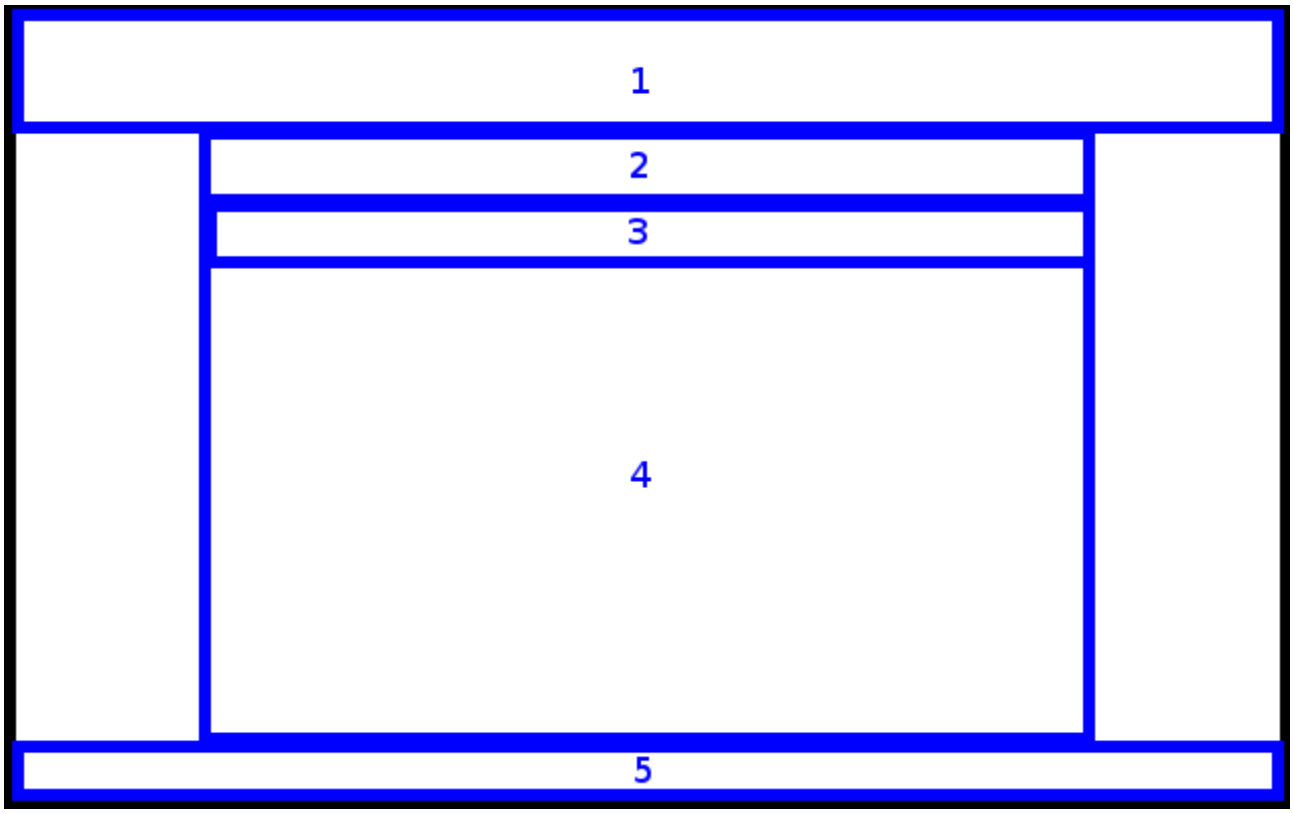

Ilustración 24: Diseño de interfaz gráfica.

- Área 1: Banner gráfico con el nombre de la aplicación.
- Área 2: información desplegada que indica al usuario el proceso que se está realizando.
- Área 3: panel de botones y opciones. Este panel sirve para agregar URLs, borrar URLs, seleccionar carpeta de destino, cancelar descarga, iniciar descarga y opciones varias.
- Área 4: área de trabajo donde al comienzo se muestra los campos de texto para ingresar las URLs y luego se muestra los avances de la descarga.
- Área 5: pie de página que indica el sistema.

8.2.2 Jerarquía del menú.

La jerarquía del menú muestra los niveles de profundidad que tienen las opciones del menú.

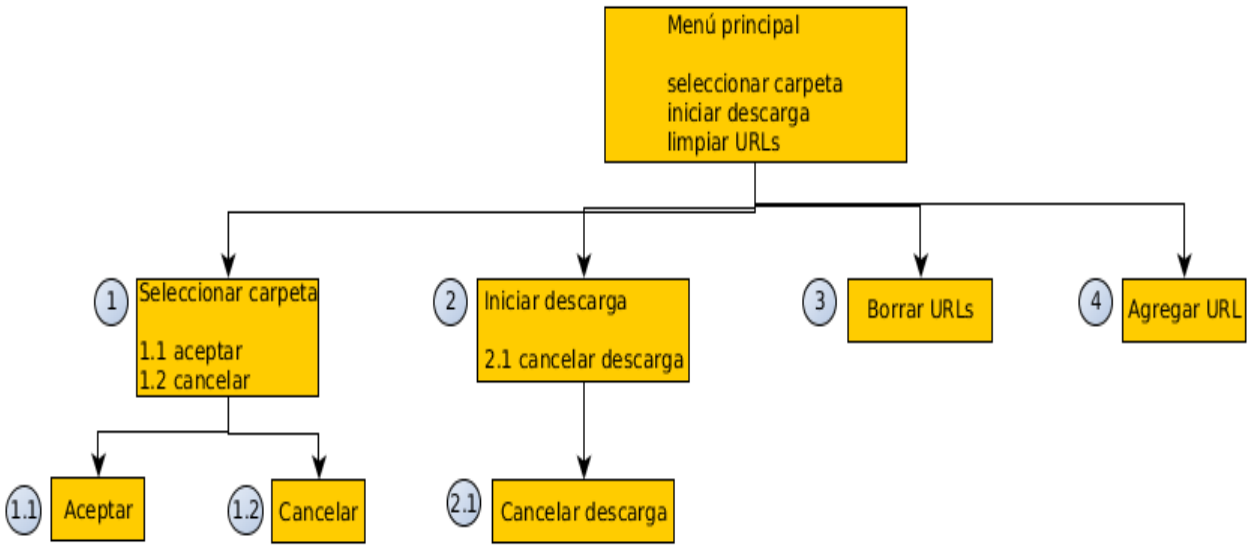

Ilustración 25: Jerarquía del menú

8.2.3 Navegación del menú.

En este diagrama se puede observar los caminos por los cuales el usuario puede navegar al seleccionar una opción del menú.

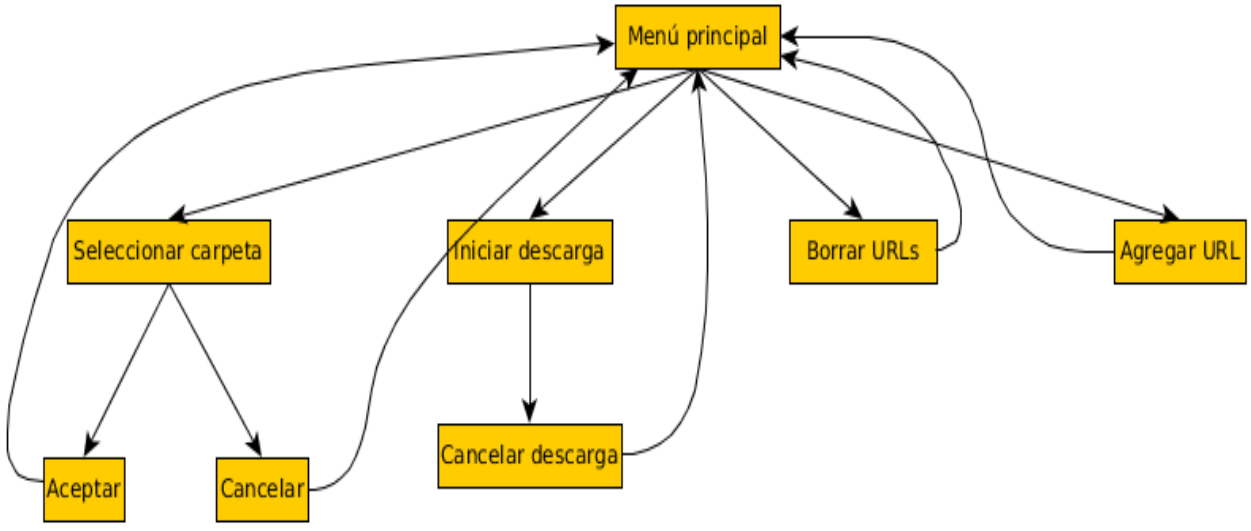

Ilustración 26: Navegación del menú

8.3 Especificación de módulos.

Estos módulos son partes del programa que trabajan de forma independiente y realizan tareas especificas, estos módulos se crearon para la reutilización de tareas que con constantemente ejecutadas simplificando el código del programa. En estos módulos se reciben los datos o parámetros de entrada y estos generan salidas de dicho proceso.

N° modulo: 1		Nombre modulo: validar URL	
Parámetros de entrada		Parámetros de salida	
Nombre: URL de servidor	Tipo de dato: String	Nombre: resultado de validación	Tipo de dato: boolean

N° modulo: 2		Nombre modulo: extraer nombre del servidor	
Parámetros de entrada		Parámetros de salida	
Nombre: URL de servidor	Tipo de dato: string	Nombre: nombre del servidor	Tipo de dato: string

N° modulo: 3		Nombre modulo: extraer ruta del servidor	
Parámetros de entrada		Parámetros de salida	
Nombre: URL de servidor	Tipo de dato: string	Nombre: ruta del servidor	Tipo de dato: string

N° modulo: 4		Nombre modulo: extraer nombre del archivo	
Parámetros de entrada		Parámetros de salida	
Nombre: URL de servidor	Tipo de dato: string	Nombre: nombre del archivo	Tipo de dato: string

N° modulo: 5		Nombre modulo: enviar comando de login	
Parámetros de entrada		Parámetros de salida	
Nombre: usuario y password	Tipo de dato: string	Nombre: respuesta	Tipo de dato: string

N° modulo: 6		Nombre modulo: enviar comando de cambio de directorio de trabajo	
Parámetros de entrada		Parámetros de salida	
Nombre: ruta del directorio	Tipo de dato: string	Nombre: respuesta	Tipo de dato: string

N° modulo: 7		Nombre modulo: enviar comando de modo pasivo	
Parámetros de entrada		Parámetros de salida	
Nombre: modo pasivo	Tipo de dato: string	Nombre: puerto al cual conectarse	Tipo de dato: string

N° modulo: 8		Nombre modulo: enviar comando de reanudar descarga	
Parámetros de entrada		Parámetros de salida	
Nombre: byte desde el cual reanudar	Tipo de dato: string	Nombre: respuesta	Tipo de dato: string

N° modulo: 9		Nombre modulo: almacenar bytes descargados	
Parámetros de entrada		Parámetros de salida	
Nombre: buffer de bytes	Tipo de dato: array de bytes	Nombre: archivo	Tipo de dato: binario

N° modulo: 10		Nombre modulo: estimar los bytes de cada servidor	
Parámetros de entrada		Parámetros de salida	
Nombre: tiempos de medición y tamaño de etapa	Tipo de dato: long	Nombre: bytes por cada servidor	Tipo de dato: long

N° modulo: 11		Nombre modulo: cronometrar tiempo	
Parámetros de entrada		Parámetros de salida	
Nombre: tiempo presente del SO	Tipo de dato: long	Nombre: tiempo transcurrido	Tipo de dato: long

N° modulo: 12		Nombre modulo: calcular tamaño de las etapas	
Parámetros de entrada		Parámetros de salida	
Nombre: tamaño total del archivo y número de etapas	Tipo de dato: long e integer	Nombre: cantidad de bytes de cada etapa	Tipo de dato: long

N° modulo: 13		Nombre modulo: calcular cantidad de etapas	
Parámetros de entrada		Parámetros de salida	
Nombre: número de servidores y tamaño del archivo	Tipo de dato: integer y long	Nombre: número de etapas	Tipo de dato: integer

9 ANEXO: CÓDIGO FUENTE.

En este capitulo se presenta el código de la clase más importante, en donde se realiza las mediciones y la distribución de los bytes. También esta clase coordina las etapas y los threads que son lanzados paralelamente para descargar desde cada servidor la parte que le corresponde.

Las otras clases son la clase “DownloadFTP.java”, las cual son los threads que realizan la descarga de cada parte, la clase “ConnectServersFTP.java” en donde se realiza el envío de comandos de control y se crean los socket de conexión de control y de datos con sus respectivos puertos. Estas clases son triviales lo cual no justifica el adjuntarlas en este documento, no tienen una mayor complejidad.

9.1 Clase “coordinate.java”

```
import java.io.*;
import java.util.ArrayList;
import java.util.Collections;
import javax.swing.*;
/**
 *
 * @author cristopher
 */
public class Coordinate extends Thread{
    private int totalConexiones = 0;
    private long sizeSample = 16*1024*1024;
    private GUI gui;
    private DownloadFTP download;
    private long actual=0, totalDescargado=0, finalSize=0, tiempoInicial=0,
tiempoFinal=0,tiempoProcesar=0,tiempoaux=0,tiempoMuestras = 0;
    private File destino;
    private String absolutPath;
    private RandomAccessFile randomFile;
    private ConnectServersFTP conectar;
    private ArrayList<JTextField> urls;
    private ArrayList<Long> times=new ArrayList<Long>();
    private ArrayList<Long> serverCronometro=new ArrayList<Long>();
    private ArrayList<Long> sizes=new ArrayList<Long>();
```

```
private ArrayList<ConnectServersFTP> conexiones=new ArrayList<ConnectServersFTP>();
private File log = null;
private FileWriter fw = null;
private BufferedWriter bw = null;
private PrintWriter pw = null;

public void run(){
   this.log = new File(this.absolutPath+File.separator+"log.txt");
   String url;
   for(JTextField z:urls){
      url=z.getText();
      this.conectionsList(url);
   }
   if(!checkFiles(conexiones)){
      JOptionPane.showMessageDialog(null, "los archivos no son iguales, asegurese de que sean identicos");
      System.err.println("los archivos no son iguales, asegurese de que sean identicos");
      System.exit(1);
   }
   this.totalConexiones = this.conexiones.size();
   destino = new File(this.absolutPath+File.separator+conexiones.get(0).getFile());
  try{
     this.fw = new FileWriter(this.log);
     this.bw = new BufferedWriter(fw);
     this.pw = new PrintWriter(bw);

      randomFile=new RandomAccessFile(destino,"rw");
      System.out.println("comenzando la transaccion...");
      this.startTransfer();
      System.out.println("el tamaño del archivo creado es: "+randomFile.length());
      randomFile.close();
      this.pw.close();
      this.bw.close();
      this.fw.close();
      for(ConnectServersFTP z:conexiones){
         z.closeInputOutput();
```

```
            }
        }
        catch(FileNotFoundException e){
            JOptionPane.showMessageDialog(null, "Archivo no encontrado");
            System.err.println("Archivo no encontrado");
            System.exit(1);
        }
        catch(IOException e){
            JOptionPane.showMessageDialog(null, "Error de lecturo/escritura en archivo");
            System.err.println("Error de lecturo/escritura en archivo");
            System.exit(1);
        }
        System.out.println("los bytes descargados fueron: "+this.totalDescargado);
        JOptionPane.showMessageDialog(null, "Finalizado!");
        System.out.println("finalizado!");
        System.exit(0);
    }

    public void recibirURLS(ArrayList<JTextField> urls,GUI gui, String absolutPath){
        this.absolutPath=absolutPath;
        this.urls=urls;
        this.gui=gui;
    }

    public void conectionsList(String url){
        conectar=new ConnectServersFTP();
        conectar.initial(splitURL(url));
        conexiones.add(conectar);
    }

    private boolean checkFiles(ArrayList<ConnectServersFTP> conexiones){
        int i;
        for(i=1;i<totalConexiones;i++){
            if(!conexiones.get(i).getSize().equals((conexiones.get(i-1).getSize()))){
                return false;
```

```
            }
        }
        return true;
    }

    private String[] splitURL(String url){
        String urlSplit[]=new String[3];
        int inicio,fin;
        inicio=0;
        fin=url.indexOf("/");
        if(fin==-1){
            JOptionPane.showMessageDialog(null, "ERROR! la URL: \\\"\"+url+\"\\\" no es válida o no indicó el
directorio del archivo en el URL");
            System.err.println("ERROR! la URL: \""+url+"\" no es válida o no indicó el directorio del archivo en el
URL");
            System.exit(1);
        }
        urlSplit[0]=this.removeSpace(url.substring(inicio, fin));
        inicio=fin+1;
        fin=url.lastIndexOf("/");
        if(fin==inicio-1){
            System.out.println("el archivo esta ubicado en la raiz");
            urlSplit[1]="";
            fin--;
        }
        else{
            urlSplit[1]=this.removeSpace(url.substring(inicio, fin));
        }
            urlSplit[2]=this.removeSpace(url.substring(fin+1));
        return urlSplit;
    }

    private String removeSpace(String palabra){
        String palabraSinEspacios="";
        String[] dividido=palabra.split(" ");
```

```
    for(String z:dividido){
        palabraSinEspacios+=z;
    }
    return palabraSinEspacios;
}

private void startTransfer(){

    ArrayList<DownloadFTP> threads=new ArrayList<DownloadFTP>();
    finalSize=Long.parseLong(conexiones.get(0).getSize());
    int i,j,etapas=32;
    long sizeEtapa=finalSize/etapas;

    tiempoInicial = System.currentTimeMillis();
    System.out.println("tamaño completo: "+finalSize);
    this.pw.write("tamaño total archivo: "+finalSize+" bytes"+"\n");

    System.out.println("tamaño etapa: "+sizeEtapa);

    for(j=0;j<totalConexiones;j++){
        this.serverCronometro.add((long)0);
    }

    for(j=0;j<etapas;j++){
        gui.setEtapa("Etapa "+(j+1));
        System.out.println("comenzando la etapa: "+(j+1));
        this.pw.append("comenzando etapa: "+(j+1)+"\n");
        threads.clear();
        if(j==etapas-1){
            sizeEtapa=finalSize-(sizeEtapa*(etapas-1));
            System.out.println("tamaño etapa final: "+sizeEtapa);
            this.pw.append("tamaño etapa: "+sizeEtapa+"\n");
        }
        else{
            this.pw.append("tamaño etapa: "+sizeEtapa+"\n");
```

```
        }
        if(j == 0){
           /**la primera etapa se divide en partes iguales para cada servidor
            * el primer servidor se lleva adicional los restos de la división
            */
           this.sizes.clear();
           for(int k=0;k<totalConexiones;k++){
              this.sizes.add(this.sizeSample);
           }
           sizeEtapa = sizeEtapa - (this.sizeSample*totalConexiones);
           this.defineTimes(conexiones);
           this.defineSizes(sizeEtapa);
           sizeEtapa = sizeEtapa + (this.sizeSample*totalConexiones);
        }
        else{
           this.defineSizes(sizeEtapa);
        }

        for(i=0;i<totalConexiones;i++){
           this.tiempoaux = System.currentTimeMillis();
           download=new DownloadFTP();
           System.out.println("tamaño de la parte "+i+": "+this.sizes.get(i));
           this.pw.append("tamaño asignado servidor "+this.conexiones.get(i).getHost()+": "+this.sizes.get(i)+"
bytes"+"\n");
           System.out.println("la parte "+i+" deberia escribir desde "+actual+" hasta "+(actual+this.sizes.get(i)));
           this.conexiones.get(i).closeInputOutput();
           this.conexiones.get(i).setSocketCommand();
           this.conexiones.get(i).setInOutCommand();
           this.conexiones.get(i).navigate();
           download.setting(this.conexiones.get(i).getIn(actual),this.sizes.get(i),conexiones.get(i).getHost());
           download.setRandomFile(this.destino,this.actual);
           threads.add(download);
           this.totalDescargado+=this.sizes.get(i);
           actual=actual+(this.sizes.get(i));
           this.tiempoProcesar = this.tiempoProcesar + (System.currentTimeMillis() - this.tiempoaux);
```

```
    }
    for(DownloadFTP z:threads){
       z.start();
    }
    boolean continuar=true;
    while(continuar){
       continuar=false;
       for(int z=0;z<threads.size();z++){
         if(!threads.get(z).getFinish()){
            continuar = true;
            this.sendPorcent(threads.get(z).getCount(),threads.get(z).getLimit(),z);
         }
         else this.gui.setProgressvalue(100,z);
       }
    }
    this.times.clear();
    for(int z=0;z<threads.size();z++){
       this.times.add(threads.get(z).getTotalTime());
       this.serverCronometro.set(z,this.serverCronometro.get(z)+threads.get(z).getTotalTime());
    }

    System.out.println(this.serverCronometro);
    for(ConnectServersFTP z:conexiones){
       z.closeInputOutput();
       z.setSocketCommand();
       z.setInOutCommand();
       z.navigate();
    }
  }
  tiempoFinal = System.currentTimeMillis() - tiempoInicial;
  System.out.println("Tiempo total: "+tiempoFinal);
  this.pw.append("Tiempo total: "+tiempoFinal+" milisegundos"+"\n");
  this.pw.append("Tiempo total procesamiento: "+this.tiempoProcesar+" milisegundos"+"\n");
  int horas = (int)(tiempoFinal/3600000);
  int restoHoras = (int)(tiempoFinal%3600000);
```

```
        int minutos = (int)(restoHoras/60000);
        int restoMinutos = (int)(restoHoras%60000);
        int segundos = (int)(restoMinutos/1000);
        int restoSegundos = (int)(restoMinutos%1000);

        for(i=0;i<totalConexiones;i++){
            this.pw.append("tiempo de descarga server "+this.conexiones.get(i).getHost()+":
"+this.serverCronometro.get(i)+" milisegundos"+"\n");
        }

        long delta = 0;
        Collections.sort(this.serverCronometro);
        Collections.reverse(this.serverCronometro);
        delta = this.serverCronometro.get(0) - this.serverCronometro.get(this.serverCronometro.size()-1);
        this.pw.append("diferencia entre mejor y peor server: "+delta+" milisegundos"+"\n");
        int deltaHoras = (int)(delta/3600000);
        int restoDeltaHoras = (int)(delta%3600000);
        int deltaMinutos = (int)(restoDeltaHoras/60000);
        int restoDeltaMinutos = (int)(restoDeltaHoras%60000);
        int deltaSegundos = (int)(restoDeltaMinutos/1000);
        int restoDeltaSegundos = (int)(restoDeltaMinutos%1000);
        JOptionPane.showMessageDialog(null, "\tDETALLES"
                +"\nTiempo total: "+horas+":"+minutos+":"+segundos+"."+restoSegundos
                +"\nLa diferencia entre servidores
fue :"+deltaHoras+":"+deltaMinutos+":"+deltaSegundos+"."+restoDeltaSegundos);
        this.pw.append("tiempo total de muestras: "+this.tiempoMuestras);
    }

    private void sendPorcent(long count,long limit,int index){
        try{
        long porcent = count*100/limit;
        this.gui.setProgressvalue((int)porcent,index);
        }catch(ArithmeticException e){
            JOptionPane.showMessageDialog(null, "ERROR INPREVISTO: división por cero");
            System.exit(1);
```

```
    }
  }

  private void defineSizes(long sizeEtapa){
    int i,index = 0;
    ArrayList<Double> bytespersecond = new ArrayList<Double>();
    long  total = 0, x = 0, sum = 0,rest = 0;
    double bps = 0, mayor = 0;
    try{
    for(i=0;i<totalConexiones;i++){
      bps = (double)this.sizes.get(i)/(double)this.times.get(i);
      System.out.println("bps: "+bps);
      System.out.println("time: "+this.times.get(i));
      System.out.println("size: "+this.sizes.get(i));
      if(bps>mayor){
        mayor = bps;
        index = i;
      }
      bytespersecond.add(bps);
      total += bps;
    }
    x = sizeEtapa/total;
    for(i=0;i<this.conexiones.size();i++){
      sum += (x*bytespersecond.get(i));
      this.sizes.set(i, (long)(x*bytespersecond.get(i)));
    }
    rest = sizeEtapa-sum;
    this.sizes.set(index, rest+this.sizes.get(index));
    }catch(ArithmeticException e){
      JOptionPane.showMessageDialog(null, "ERROR INPREVISTO: división por cero");
      System.exit(1);
    }
  }

  private void defineTimes(ArrayList<ConnectServersFTP> conexiones){
```

```
        gui.mostrarEspera(true);
        this.times.clear();
        boolean continuar;
        for(ConnectServersFTP z:conexiones){
            download=new DownloadFTP();
            z.closeInputOutput();
            z.setSocketCommand();
            z.setInOutCommand();
            z.navigate();
            download.setting(z.getIn(actual),this.sizeSample,z.getHost());
            download.setRandomFile(this.destino,this.actual);
            download.start();
            this.actual += this.sizeSample;
            continuar = true;
            while(continuar){
                continuar = false;
                if(!download.getFinish()){
                    continuar = true;
                }
                System.out.flush();
            }
            this.tiempoMuestras += download.getTotalTime();
            this.times.add(download.getTotalTime());
            download = null;
        }
        this.gui.mostrarEspera(false);
    }
}
```

10 ANEXO: PLANIFICACIÓN INICIAL DEL PROYECTO.

	Nombre	Duraci...	Inicio	Terminac
1	**análisis**	15 da...	**27-08-13 08:00 ...**	**16-09-13 05:**
2	Reunión con profesor guía	1 day?	28-08-13 08:00 AM	28-08-13 05:00
3	Lectura de papers relacionados	6 days?	28-08-13 08:00 AM	04-09-13 05:00
4	Reunión con profesor guía	1 day?	27-08-13 08:00 AM	27-08-13 05:00
5	Lectura del RFC del protocolo FTP y HTTP	4 days?	04-09-13 08:00 AM	09-09-13 05:00
6	buscar y analizar código fuente de projectos similares	6 days?	09-09-13 07:00 AM	16-09-13 05:00
7	**diseño**	16 da...	**16-09-13 08:00 ...**	**07-10-13 05:**
8	definir avances de informe con profesor guía	1 day?	16-09-13 08:00 AM	16-09-13 05:00
9	Desarrollar un modelo	3 days?	23-09-13 08:00 AM	25-09-13 05:00
10	definir los componentes del sistema	5 days?	26-09-13 08:00 AM	02-10-13 05:00
11	Reunión con profesor guía	1 day?	03-10-13 08:00 AM	03-10-13 05:00
12	avance con el informe final	3 days?	16-09-13 08:00 AM	18-09-13 05:00
13	definir las validaciones que el sitema debe tener.	3 days?	03-10-13 08:00 AM	07-10-13 05:00
14	**desarrollo**	59 da...	**08-10-13 08:00 ...**	**27-12-13 05:**
15	implementación de la descarga de los fragmento a cada servidor	14 days	08-10-13 08:00 AM	25-10-13 05:00
16	Reunión con profesor guía	1 day?	28-10-13 08:00 AM	28-10-13 05:00
17	avance con el informe final	3 days?	08-10-13 08:00 AM	10-10-13 05:00
18	mejora de la implementación anterior alternando el tamaño de la descar	7 days	28-10-13 08:00 AM	05-11-13 05:00
19	reunion con profesor guía para ver los avances y hacer cambios	1 day?	06-11-13 08:00 AM	06-11-13 05:00
20	avance con el informe final	3 days?	08-10-13 08:00 AM	10-10-13 05:00
21	Primera Entrega funcional del software	1 day?	06-11-13 08:00 AM	06-11-13 05:00
22	lectura del RFC de HTTP para implementar la funcion anterior en HTTP	4 days?	06-11-13 08:00 AM	11-11-13 05:00
23	implementar la primera entrega con el protocolo HTTP	7 days	12-11-13 08:00 AM	20-11-13 05:00
24	reunión con profesor guía para ver las técnicas de predicción de ancho	1 day?	21-11-13 08:00 AM	21-11-13 05:00
25	avance con el informe final	5 days?	21-11-13 08:00 AM	27-11-13 05:00
26	implementar tecnicas de predicción de ancho de banda	7 days?	21-11-13 08:00 AM	29-11-13 05:00
27	integrar las técnicas de predicción a la primera entrega para mejorar los	3 days?	30-11-13 08:00 AM	04-12-13 05:00
28	segunda entrega funcional del software	1 day?	05-12-13 08:00 AM	05-12-13 05:00
29	implementar las técnicas de predicción con el protocolo HTTP	5 days?	05-12-13 08:00 AM	11-12-13 05:00
30	implementar el diseño de la interfaz de usuario	5 days?	12-12-13 08:00 AM	18-12-13 05:00
31	reunion con profesor guía	1 day?	19-12-13 08:00 AM	19-12-13 05:00
32	avance con el informe final	6 days?	20-12-13 08:00 AM	27-12-13 05:00
33	Entrega final del software	1 day?	27-12-13 08:00 AM	27-12-13 05:00

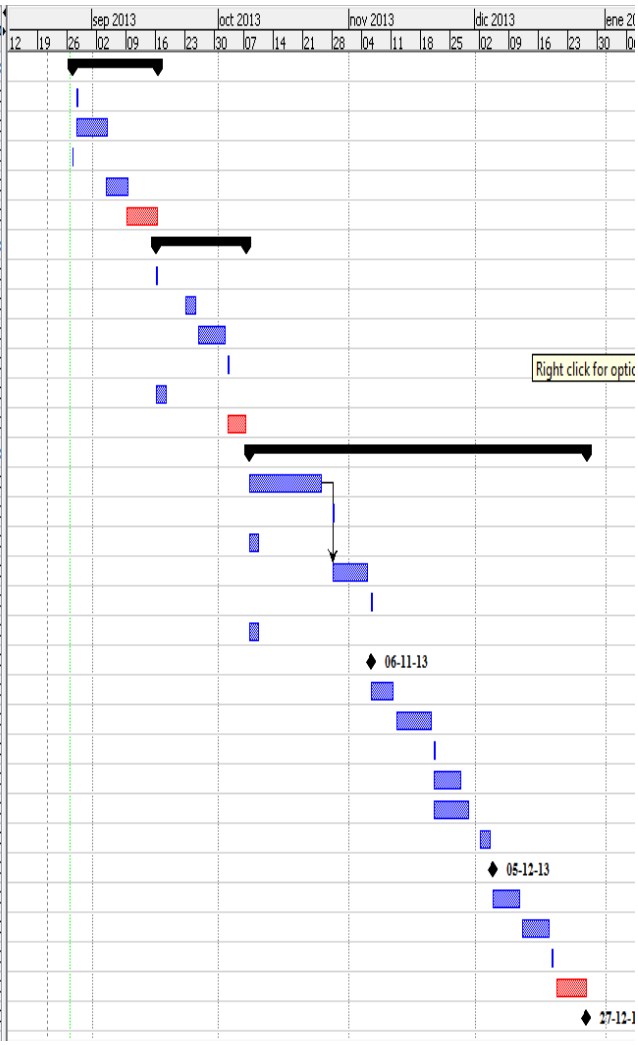

Ilustración 27: planificación inicial

10.1 Estimación inicial de tamaño.

- Entradas externas
  - El proceso de “comprobar y gestionar datos ingresados” contiene 2 entradas simples.
  - El proceso de “gestionar envío de comandos FTP” contiene 1 entrada promedio.
  - El proceso de “gestionar transferencia de datos” contiene 1 entrada compleja.
- Salidas externas
  - El proceso de “gestionar envío de comandos FTP” contiene 4 salidas simples. El proceso de “comprobar y gestionar datos ingresados” contiene 2 salidas simples.
  - El proceso “gestionar etapas y tiempos” contiene 2 salidas promedio.
  - El proceso “gestionar envío de comandos FTP” contiene 2 salidas complejas.
- Archivos lógicos internos.
  - El proceso de “gestionar transferencia de datos” contiene 1 entrada compleja.

Componente	**simple**	**Promedio**	**compleja**	**total**
EI	2*3	1*4	1*6	16
EO	6*4	2*5	2*7	48
EQ	1*3	4*4	1*6	35
ILF	0*7	0*10	1*15	15
EIF	0*3	0*7	0*10	0
				114

Tabla 2: Punto función sin ajustar

Comunicación de datos:	2
Proceso distribuido:	4
Objetivos de rendimiento:	4
Configuración de explotación compartida:	0
Tasa de transacciones:	0
Entrada de datos en línea:	0

Eficiencia con el usuario final:	0
Actualizaciones en línea:	0
Lógica de proceso interno compleja:	3
Reusabilidad del código:	4
Conversión e instalación contempladas:	0
Facilidad de operación:	5
Instalaciones múltiples:	0
Factibilidad de cambio:	0

FP = 114*[0.65+[0.01*22]]

FP = 114*0.87

FP = 99.18

Entorno y lenguaje	Líneas de código por PF	Horas por PF
Lenguajes 2GL	300	20 a 30
Lenguajes 3GL	100	10 a 20
Lenguajes 4GL	20	5 a 10

Tabla 3: esfuerzo vs entorno y lenguaje

Según la tabla indicada nuestro proyecto utiliza lenguajes de 3° generación por lo que nos situaremos en 10 horas por PF. De este modo el esfuerzo de desarrollo es:

- horas/ persona = FP/[1/10 personas/horas] = **991.8 horas/personas.**

Las líneas de código estimadas con los PF es:

- Líneas de código = FP*(líneas de código por PF) = **9918 líneas de código.**

10.2 Contabilización final del tamaño del software.

- La cantidad de líneas de código que tiene el software en total son 802. En este total no se tomo en cuenta los comentarios, pero si las líneas en blanco.. Todas los métodos de las clases son tomados en cuenta por el hecho de que se hacen llamadas a estos métodos reutilizables.
- La aplicación según el número estimado con los PF nos da un número mucho más alto, lo cual no es del todo malo ya que el proyecto se terminó en menos tiempo permitiendo mejorar aún más la aplicación.

- Según la carta Gantt creada en la planificación inicial del proyecto, los días en total de todas las fases son 90, de los cuales se trabajaron 4 horas por día. Esto nos da un total de 360 horas de trabajo en total dedicadas al proyecto.

11 ANEXO: RESULTADOS DE ITERACIONES EN EL DESARROLLO.

En este proyecto se seleccionó la metodología incremental, en la cual realizamos tres iteraciones donde en cada una de estas se entrega un producto funcional del sistema, para poder ser probado y reatroalimentar al equipo con las observaciones de los usuarios.

En la primera iteración el producto a entregar fue realizar la descarga desde un sólo servidor, en esta entrega no se dispone de interfaz, sólo a través de la consola de comandos se ingresa el URL del servidor FTP para luego entregar el tiempo total de la descarga. El resultado de esta entrega resultó exitoso, sin problemas.

En la segunda iteración se entrego una versión de descarga paralela, con interfaz simple, en el cual se especifican las URLs de los servidores y se realiza la descarga. Esta entrega tuvo un error en la verificación del archivo, ya que el archivo descargado no era idéntico al archivo alojado en el servidor debido a un problema con los bytes descargados no fueron colocados en el archivo en donde corresponden. Esta versión se mejoró los errores y se concluyó acá para tener la versión de descarga paralela no adaptativa.

En la tercera iteración se realizó una versión aparte que es la versión de descarga paralela adaptativa, esta versión es la que se desea que sea mejor que los otras versiones creadas anteriormente, y el objetivo de este proyecto. Esta versión tuvo algunos problemas con la medición de banda ancha que luego se solucionaron. Al terminar la iteración se llego a la fase de pruebas para ver los resultados entre la descarga paralela no adaptativa y la descarga paralela adaptativa.

Durante todas las iteraciones el usuario, quien es el profesor guía Patricio Galdames, estuvo retroalimentando el proyecto, dando correcciones sobre algunos errores, sobre todo en la parte de medición de banda ancha en la cual se requirió de una técnica para poder medir el desempeño de los servidores.

Las versiones disponibles que se ocuparon para las pruebas, producto de las iteraciones, fue una versión adaptativa y una versión no adaptativa de la descarga paralela, en la cual en la versión no adaptativa se puede ingresar un sólo servidor para medir el tiempo de la descarga desde un servidor único.

12 ANEXO: PRUEBAS.

En este capítulo se realizan pruebas para verificar el correcto funcionamiento del software de descarga en paralelo adaptativo. El desarrollo de estas pruebas se efectuará con 2 servidores en una red local.

12.1 Elementos de prueba.

- **Validación de los datos ingresados.**

  Este elemento debe validar que los datos ingresados por el usuario estén correctos.

- **Completar la descarga del archivo.**

  En este elemento de prueba se espera que la aplicación realice una descarga que tenga el mismo tamaño que el archivo original.

- **Validación la integridad del archivo.**

  Este elemento se aplica una vez descargado el archivo, al cual se le realiza una suma de verificación de la secuencia de Bytes con el fin de asegurar que el archivo descargado es una copia exacta del archivo original.

12.2 Especificación de las pruebas.

ID	Características a probar	Objetivo de la prueba	Enfoque para la definición de casos de prueba	Actividades de prueba	Criterios de cumplimiento
1	Integridad.	Detectar si la secuencia de Bytes esta correctamente ordenada. Si la suma de verificación del archivo es distinta a la del archivo original, quiere decir que algunos bytes no están bien ubicados.	Caja negra.	• Una vez descargado el archivo se procede a iniciar el programa sha256sum el cual verifica la integridad del archivo usando SHA-256 hashes una función que puede comprobar autenticidad e integridad del archivo. • Se compara la suma de verificación con la del archivo original.	EL archivo debe ser idéntico al original.
2	funcionalidad	El progreso de la descarga se debe completar al 100%.	Caja blanca	• Se ingresan las URLs y se selecciona un directorio. • Se inicia la descarga y se espera a que se complete en un 100% el progreso.	La aplicación debe notificar cuando se ha completado la descarga en un 100%.
3	validación	Se debe validar que se han ingresado al menos dos URLs.	Caja negra	• En la interfaz gráfica se ingresa solamente una URL.	• Se debe mostrar un aviso que se necesita al menos dos URLs.

ID	Características a probar	Objetivo de la prueba	Enfoque para la definición de casos de prueba	Actividades de prueba	Criterios de cumplimiento
4	validación	Se debe validar que las URLs estén correctas	Caja negra	• En la interfaz gráfica se ingresan dos URLs incompletas.	La aplicación debe notificar que la URL ingresada no está correcta o no se puede lograr la conexión con el servidor.
5	Validación	Se debe validar que se ha seleccionado un directorio donde almacenar la descarga.	Caja negra	• No se selecciona ningún directorio de destino.	La aplicación no debe permitir iniciar la descarga y debe notificar si aún no ha seleccionado un directorio destino.
6	funcionalidad	La aplicación debe notificar cuando se pierde la conexión con algún servidor.	Caja negra	• Cuando la descarga esté en progreso se debe desconectar un servidor.	La aplicación debe notificar que existe un error de entrada/salida con el servidor X.

Tabla 4: Especificación de pruebas

12.2.1 Resultados de pruebas.

ID	Datos de entrada	Salida esperada	Salida Obtenida	Éxito/fracaso
1	Suma de verificación del archivo original	d68a06f80b95a986dd8bb5b8ee107840a9836774d3398d0f578e769e4712689f	d68a06f80b95a986dd8bb5b8ee107840a9836774d3398d0f578e769e4712689f	éxito
2	dos URLs y directorio destino	Comprobación del termino de la descara	Notificación de la descarga completada	éxito
3	Una URL y directorio destino	Notificación de que se requiere al menos dos URLs	“debe ingresar al menos dos URLs”	éxito
4	Dos URLs incompletas y directorio destino	Notificación que indique que alguna URL no está correcta.	“error en host X”	éxito
5	Dos URLs	Notificación que indique que seleccione un directorio destino.	“debe elegir el destino del archivo”	éxito
6	Dos URLS y directorio destino	Notificación que indique que se perdió la conexión.	“no es posible realizar operaciones de Entrada y salida con el host X”	éxito

Tabla 5: Resultados de pruebas