Optimización de la Navegación Web: un Sistema de *Prefetching* Basado en el Historial de Predicciones

José Martínez Sugastti, Felipe Stuardo, Vicente González
Departamento de Electrónica e Informática, Universidad Católica "Nuestra Señora de la Asunción"
Asunción, Paraguay

***Resumen*—Web navigation response times are affected by the heterogeneous nature of Internet: links with disparate bandwidth, servers and routers with diverse specifications and network segments with particular traffic shaping policies. Also, on a daily basis, several new information systems gets connected to the Internet consequently increasing traffic. Aiming to reduce response times, prefetching systems predicts which objects will be requested by users based on predictive models. This way, predicted objects are requested and stored in cache in advance. However, literature shows that current prefecthing systems does not contemplate how effective the predictions are. This paper presents a feedback based prefecthing system. The proposed approach considers previous predictions hits and misses to update the predictive model thus enhancing the precision of predictions. The experimental study shows an improvement on users response time obtained by the proposed system.**

1. Introducción

Una técnica tradicional para la reducción de los tiempos de respuesta en la navegación *web* es el almacenamiento en *caché* de las respuestas enviadas por los servidores, de modo que solicitudes posteriores sean servidas desde una locación cercana a los usuarios finales. Como complemento al almacenamiento en *caché*, los sistemas de prebúsqueda (*prefetching*) buscan determinar cuáles serán los recursos *web* solicitados por los clientes en el futuro, de modo a obtenerlos anticipadamente y almacenarlos. Los sistemas de *prefetching* abarcan dos áreas: la predicción y la obtención de los recursos predichos. Las predicciones sobre qué recursos serán solicitados por los usuarios se realizan a partir de un modelo predictivo. Los recursos predichos luego deben ser obtenidos y almacenados en *caché*.

En la literatura se encuentran trabajos recopilatorios que analizan los diferentes aspectos que abarcan los sistemas de *prefetching*, como los algoritmos utilizados, el consumo de recursos computacionales, la selección de métricas para evaluar las predicciones, la obtención de la información utilizada como fuente para los modelos predictivos, entre otros [1], [2], [3].

Un aspecto a considerar es que varios de los sistemas propuestos realizan predicciones y solicitan los recursos predichos sin considerar el resultado final de éstas acciones al momento de actualizar el modelo predictivo, es decir, no existe una retroalimentación respecto a si un recurso predicho fue almacenado en *caché* y llegó a ser solicitado por algún usuario.

En este trabajo se presenta un sistema de *prefetching* que considera los aciertos de las predicciones realizadas para la actualización del modelo predictivo. De esta manera, se busca obtener predicciones más precisas que resulten en una mejora de los tiempos de respuesta de las solicitudes *web* de los usuarios.

Este documento está organizado de la siguiente manera: la Sección 2 presenta los fundamentos de los sistemas de *prefetching* y el estado del arte; la Sección 3 define el problema y propone un sistema para obtener mejores predicciones; la Sección 4 presenta los resultados obtenidos y su evaluación; la Sección 5 contiene las conclusiones, aportes relevantes y trabajos futuros.

2. Marco Teórico y Trabajo Relacionado

Los sistemas de prefetching abarcan dos áreas: la predicción y la obtención de recursos.

2.1. Predicción

Los algoritmos utilizados para generar los modelos predictivos son el principal componente de un sistema de *prefetching*. El objetivo de estos modelos es que los recursos predichos como próximos a ser solicitados efectivamente lleguen a ser solicitados por los clientes. No obstante, al ser predictivos pueden presentar tanto aciertos como fallas. Un algoritmo que genere un modelo con un índice bajo de aciertos puede resultar contraproducente, ya que sugerirá gestionar recursos que no llegarán a ser solicitados por los usuarios.

Los algoritmos son clasificados en base a la información utilizada para generar el modelo predictivo. De acuerdo a [7], los algoritmos son clasificados en dos categorías: basados en el análisis de sesiones de navegación y basados en el análisis del contenido de las páginas *web*. Entre los algoritmos que analizan las sesiones de navegación, se encuentran los basados en modelos de Markov [8]. Éstos algoritmos son los que requieren de menor tiempo computacional para generar predicciones [7], por ello este trabajo se centra en los algoritmos de esta clase.

Los modelos basados en el análisis de las sesiones de navegación se fundamentan en los conceptos de localidad espacial de referencia [9] y localidad temporal de referencia [10]:

- La localidad espacial se refiere a la existencia de relaciones y referencias entre las solicitudes de recursos que conforman una sesión de navegación.
- La localidad temporal se refiere a la probabilidad que recursos solicitados recientemente vuelvan a ser solicitados en el futuro. En este concepto se basan las *cachés*.

Los algoritmos basados en modelos de Markov y que analizan las sesiones de navegación de los usuarios pueden ser agrupados en dos categorías: Grafo de Dependencias (*Dependency Graph* - DG) [4] y Predicción por Coincidencias Parciales (*Prediction by Partial Matching* - PPM) [11].

2.1.1. Algoritmo DG

El algoritmo DG tiene su origen en un algoritmo predictivo utilizado en sistemas de archivos [12]. El modelo utilizado para representar las solicitudes de una sesión de navegación está basado en un grafo dirigido. Cada nodo en el grafo representa un recurso solicitado en una sesión. El algoritmo define un arco A $\Rightarrow$ B entre dos nodos A y B si B fue solicitado dentro de las n solicitudes posteriores al nodo A. Para mantener un registro de las n solicitudes posteriores, el modelo utiliza una estructura llamada ventana de accesos. El valor de n indica el tamaño de la ventana. Las probabilidades de cada arco del grafo se calculan en base a la cantidad de ocurrencias de cada nodo y cada transición. Las predicciones para un nodo A son generadas seleccionando los arcos salientes de A con probabilidades más altas y/o cuya probabilidad sea mayor a cierto valor mínimo.

El modelo generado por DG es un modelo de Markov de orden 1, ya que la transición a un nodo siguiente sólo depende de las probabilidades de los arcos salientes del nodo actual. En [4] se implementó por primera vez el algoritmo DG a la predicción de solicitudes *web*. En [5] se utilizó una ventana basada en el tiempo transcurrido entre las solicitudes de los objetos, y no basado en la cantidad de accesos recientes. En [6] se diferenció a los componentes de una página *web* entre objetos contenedores y objetos contenidos.

2.1.2. Algoritmo PPM

El algoritmo PPM está basado en un algoritmo de compresión de texto [13]. El modelo utilizado para representar las secuencias de solicitudes de una sesión de navegación está basado en un árbol. El árbol tiene un nodo raíz y cada recurso solicitado en una sesión constituye la raíz de un subárbol. Cada subárbol contiene en sus ramas los recursos subsecuentemente solicitados. Las transiciones entre nodos representan las solicitudes de los recursos y cada rama del árbol representa una secuencia de solicitudes. El parámetro principal del modelo es el orden del árbol. Éste indica la cantidad de nodos a ser analizados desde las raíces de los subárboles para realizar las predicciones. Las probabilidades de cada transición se calculan de acuerdo a la cantidad de ocurrencias de cada nodo del árbol y de cada transición.

El modelo generado por el algoritmo PPM es un modelo de Markov de orden entre 1 y m. En [14] se realizó una adaptación a la navegación *web* del algoritmo descrito en [13]. En [15], se definió que las ramas del árbol que contienen los Localizadores Uniformes de Recursos (*Uniform Resource Locators* - URLs) más populares alcancen mayores profundidades que las ramas con URLs menos populares. En [16] se estableció una clasificación para las URLs en base a criterios de popularidad, y se determina dinámicamente la longitud máxima de cada rama del árbol de acuerdo a dicha clasificación. En [17] se propuso una variante para la reducción de la memoria requerida por el modelo PPM.

2.2. Obtención de Recursos

El objetivo del motor de predicciones es la generación de un listado de recursos con altas probabilidades de ser solicitados. El proceso que gestiona los recursos predichos es el motor de *prefetching*. Los sistemas de *prefetching* han sido implementados en clientes, *proxies* y servidores [4], [5], [6]. No obstante, la ubicación de los motores de *prefetching* se da mayormente en los navegadores (*browsers*) a nivel del cliente o en los *proxies*.

Los motores de *prefetching* evalúan diversos factores para gestionar (o no) anticipadamente los recursos del listado de predicciones. Dependiendo de su ubicación, estos factores pueden ser: ancho de banda disponible, carga del procesador, restricciones de contenidos en listas de acceso, servidor al cual se debe solicitar el recurso y latencia.

2.3. Sesiones de Navegación *Web*

Para la creación del modelo predictivo se analizan las sesiones de navegación de los usuarios. Éstas son extraídas de bitácoras del sistema (archivos de *log*) de navegación *web*. Entre los formatos más difundidos se encuentran el Formato Común de Archivo de Log (*Common Log File Format* - CLF), el Formato Combinado de Archivo de *Log* (*Combined Log Format*) y el Formato Extendido de Archivo de *Log* (*Extended Log File Format* - ELF). Las principales diferencias radican en la cantidad y personalización de campos a ser utilizados para cada transacción *web*.

Debido a que los modelos predictivos de navegación se construyen en base al análisis de sesiones de navegación, la identificación y el registro del tráfico *web* es una tarea fundamental. La clasificación de flujos de red verificando los números de puerto del Protocolo de Control de Transmisión (*Transmission Control Protocol* - TCP) en las cabeceras de los paquetes constituye el tipo más simple de clasificación. No obstante, este enfoque presenta los siguientes problemas: existen servicios que no tienen asignado un puerto bien conocido, otros servicios utilizan puertos de manera aleatoria, otros servicios utilizan los puertos no asignados originalmente para dicho servicio (*tunneling*).

Como opciones para obtener mayor precisión en la identificación y clasificación del tráfico de red, existen enfoques como: basados en la inspección de la carga útil de los paquetes, basados en aprendizaje de máquina (*machine learning*) y basados en el comportamiento de los dispositivos de redes de datos (*hosts*). Éstos se diferencian en el acceso a la carga útil del paquete (*payload*), la identificación de contenidos cifrados y el costo computacional que implica aplicarlos.

3. Formulación del Problema

Los sistemas de *prefetching* analizados en la literatura realizan predicciones y solicitan los recursos predichos sin considerar el resultado final de éstas acciones al momento de actualizar modelo predictivo. No se considera una retroalimentación respecto a si un recurso predicho fue almacenado en *caché* y llegó a ser requerido luego por algún cliente.

Por ello, se propone un sistema de *prefetching* que considere los aciertos de las predicciones realizadas previamente para la actualización del modelo predictivo. De esta manera, se busca obtener una mayor cantidad de aciertos en las predicciones, que resulten en una mejora de los tiempos de respuesta de las solicitudes *web* de los clientes.

3.1. Arquitectura de la Solución

En el marco de la arquitectura de red cliente - *proxy* - servidor, el sistema de *prefetching* fue ubicado en el *proxy* por las siguientes razones:

- *Acceso a múltiples clientes:* Se tiene acceso a solicitudes enviadas y respuestas recibidas de múltiples clientes de una red local.
- *Acceso a múltiples servidores:* Se tiene acceso a solicitudes recibidas y respuestas enviadas de múltiples servidores.

- *Red local*: El presente trabajo está orientado a obtener beneficios del *prefetching* para los clientes de una red local, como la de los laboratorios de estudiantes de una universidad.
- *Retroalimentación:* Al llevarse a cabo tanto las predicciones como la obtención de recursos en el *proxy*, se busca que ambos componentes compartan información de modo a actualizar el modelo predictivo en base a dicha interacción.
- *Complementa a browsers:* Para el caso de *browsers* que realicen *prefetching*, las solicitudes de los *browsers* pueden coincidir con recursos ya obtenidos anticipadamente por el sistema de *prefetching* del *proxy*. Para *browsers* que no realicen *prefetching*, las respuestas podrán ser servidas igualmente desde el *proxy*.

La solución propuesta abarca tres componentes: módulo de reportes de navegación, motor de predicciones y motor de *prefetching* (Figura 1).

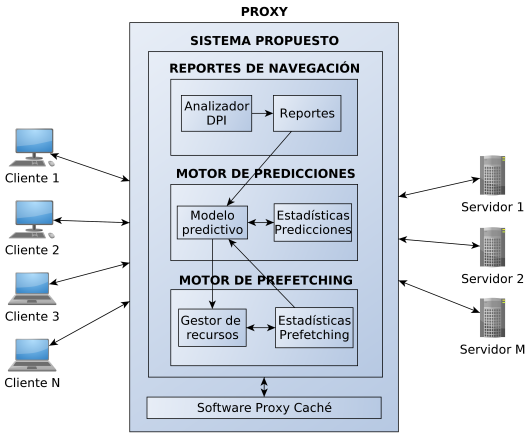


Figura 1. Arquitectura y procesos componentes de la solución propuesta.

3.1.1. Módulo de Reportes de Navegación

El módulo de reportes de navegación está compuesto por un analizador de paquetes de red y por los reportes que son generados. El analizador toma archivos de captura de paquetes. Éstos archivos contienen copias de los paquetes de red que pasaron por una interfaz de red en un intervalo de tiempo determinado.

Considerando los problemas con la clasificación del tráfico *web* verificando los números de puerto TCP (Sección 2.3), se eligió la técnica de Inspección Profunda de Paquetes (*Deep Packet Inspection* - DPI), basada en el análisis de la carga útil de los paquetes, por los siguientes motivos:

- *Independencia de puerto:* Al basar la clasificación de paquetes en el contenido de la carga útil, se podrá identificar a los paquetes con contenido de Protocolo de Transferencia de HiperTexto (*HyperText Transfer Protocol* - HTTP) independientemente del puerto utilizado por los clientes y servidores.
- *Acceso a la carga útil:* Mediante el acceso a la carga útil de los paquetes se podrá extraer datos de los campos HTTP.
- *Signature definido:* HTTP tiene un conjunto de palabras (*signature*) fijo basado en los Documentos de Solicitud de Comentarios (*Request for Comments* - RFC) que lo definen, a diferencia de otros protocolos que no están estandarizados o sus especificaciones son propietarias.
- *Comunicación no encriptada:* La comunicación HTTP no va encriptada, por lo cual se puede acceder a la carga útil.
- *Análisis fuera de línea (offline):* El análisis con DPI se realizará fuera de línea, no en tiempo real, lo cual resultaría costoso computacionalmente en una Computadora Personal (*Personal Computer* - PC) de uso genérico [18].

A los paquetes identificados con contenido HTTP se extraen los siguientes datos: direcciones de Protocolo de Internet (*Internet Protocol* - IP) de origen y destino, puertos de origen y destino, URL del recurso solicitado, método de la solicitud (ej. GET) y nombre del servidor remoto (campo *host*).

Una vez analizadas por completo las capturas de paquetes de red, se generan reportes donde se individualizan las sesiones de navegación de cada cliente. La finalidad de los reportes es estructurar la información sobre las sesiones de navegación de los usuarios de una red, de modo que puedan ser utilizados por algoritmos que generan modelos predictivos.

Los reportes de las sesiones de navegación se estructuran utilizando el Lenguaje de Marcado Extensible (*eXtensible Markup Language* - XML) [19]. XML es de los formatos más difundidos para compartir información entre programas, redes y personas. Existen analizadores (*parsers*) en diversos lenguajes de programación y para diversas plataformas, lo que permite que el reporte pueda ser utilizado en diversos entornos. Así también, XML puede ser transformado a otros formatos.

Cabe señalar que la utilización de estos reportes no está restringida a este trabajo. Pueden ser utilizados en cualquier ámbito que se analicen sesiones de navegación *web*.

3.1.2. Motor de Predicciones

El motor de predicciones procesa los reportes generados por el módulo de reportes. Éstos reportes son utilizados para generar el modelo predictivo. Además del modelo predictivo, el motor de predicciones está compuesto por una base de datos que almacena estadísticas sobre su funcionamiento.

En este trabajo, se propone una variante (Sección 3.2) del algoritmo DG basada en la retroalimentación de datos entre los motores de predicción y de *prefetching* (Figura 3). De este modo, el modelo predictivo se actualizará no sólo en base a las sesiones de navegación, sino también en base a los aciertos de predicciones anteriores. Se ha elegido el algoritmo DG por los siguientes motivos:

- *Correlaciona solicitudes no consecutivas*: El algoritmo DG correlaciona recursos que fueron solicitados tanto consecutiva como no consecutivamente, a diferencia del algoritmo PPM.
- *Modelo de Markov de orden 1*: El modelo generado es un modelo de Markov de orden 1, con lo cual el costo computacional para la generación de predicciones es menor que el de un modelo de mayor orden. Esto es importante debido a que las consultas al modelo y la generación de predicciones deben ser veloces, ya que serán realizadas a medida que el *proxy* vaya recibiendo las solicitudes de los usuarios.

3.1.3. Motor de Prefetching

El motor de *prefetching* está compuesto por un gestor de recursos y una base de datos que almacena estadísticas sobre el desempeño del gestor. Los motores de *prefetching* utilizan los listados de predicciones a modo de sugerencias sobre recursos a ser obtenidos anticipadamente, antes que los usuarios lleguen a solicitarlos. El gestor de recursos solicita los recursos predichos y

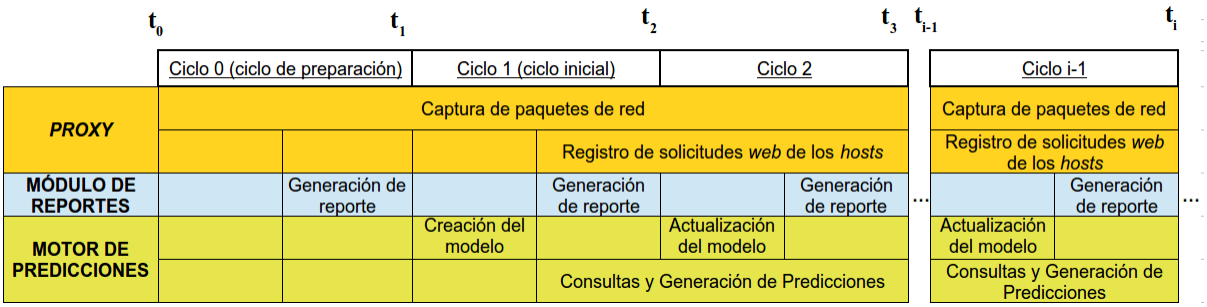


Figura 2. Secuencia de ciclos del sistema propuesto.

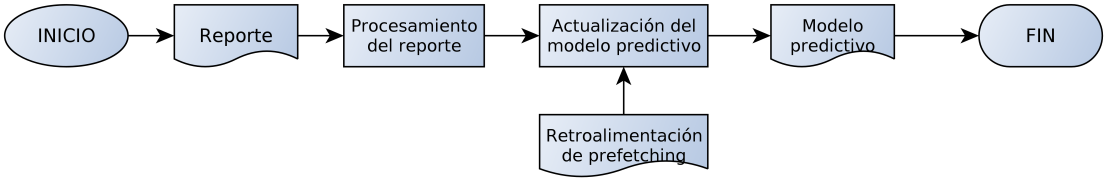


Figura 3. Motor de predicciones. Generación del modelo predictivo.

éstas solicitudes son procesadas por un *software* de *proxy caché* el cual obtendrá y almacenará (o no) los objetos solicitados en base a su configuración.

Las estadísticas almacenadas en la base de datos son utilizadas para evaluar las predicciones y el *prefetching*. Ésos datos serán luego utilizados para la actualización del modelo predictivo.

3.2. Algoritmo Propuesto

Este trabajo propone una variante del algoritmo DG (Sección 2.1.1), la cual utiliza datos de retroalimentación entre los motores de predicción y de *prefetching* de modo que el modelo predictivo se actualice no sólo en base a las sesiones de navegación, sino también en base a los aciertos de las predicciones anteriores. Para ello, se deben verificar los recursos que han sido predichos por el sistema, los recursos almacenados en *caché* como consecuencia de las predicciones y las solicitudes de los usuarios durante un ciclo.

Un ciclo consiste en un intervalo de tiempo en el que:

- se capturan paquetes de red para generar reportes de navegación, y/o;
- se registran las solicitudes de los usuarios, y/o;
- se crea o actualiza el modelo predictivo, y/o;
- se realizan consultas al modelo para generar predicciones.

La secuencia de ciclos del sistema propuesto se observa en la Figura 2. El *proxy* captura permanentemente paquetes de red, a partir de los cuales son generados los reportes de navegación por el módulo de reportes al final de cada ciclo. Cada ciclo finaliza con la generación de un reporte de navegación.

El ciclo 0 (ciclo de preparación) consiste en la generación de uno o varios reportes, los cuales son utilizados para la creación de la primera versión del modelo predictivo.

En el ciclo 1, con el modelo inicial ya construido, el *proxy* inicia el registro de las solicitudes de los clientes. A partir de éstas solicitudes, el motor de predicciones realiza consultas al modelo predictivo y se generan las predicciones para cada recurso solicitado. Los ciclos posteriores al ciclo 1, consisten en la actualización del modelo predictivo y la generación del reporte correspondiente

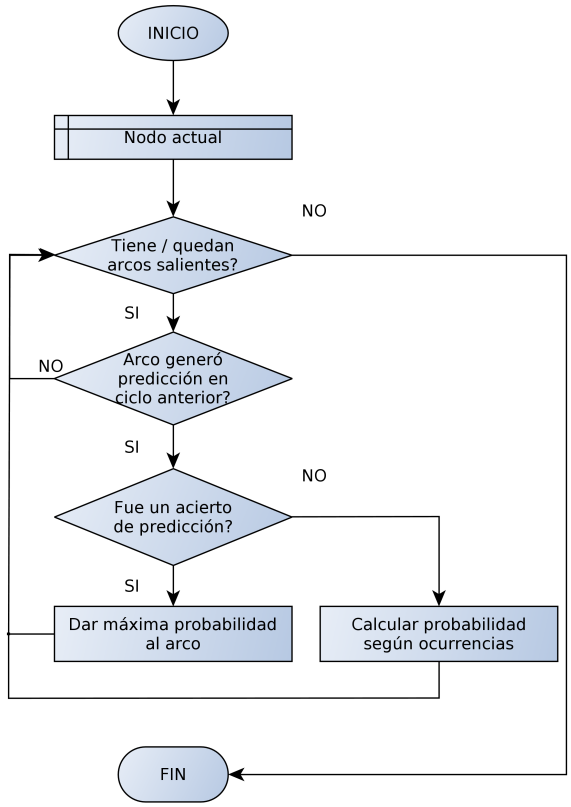
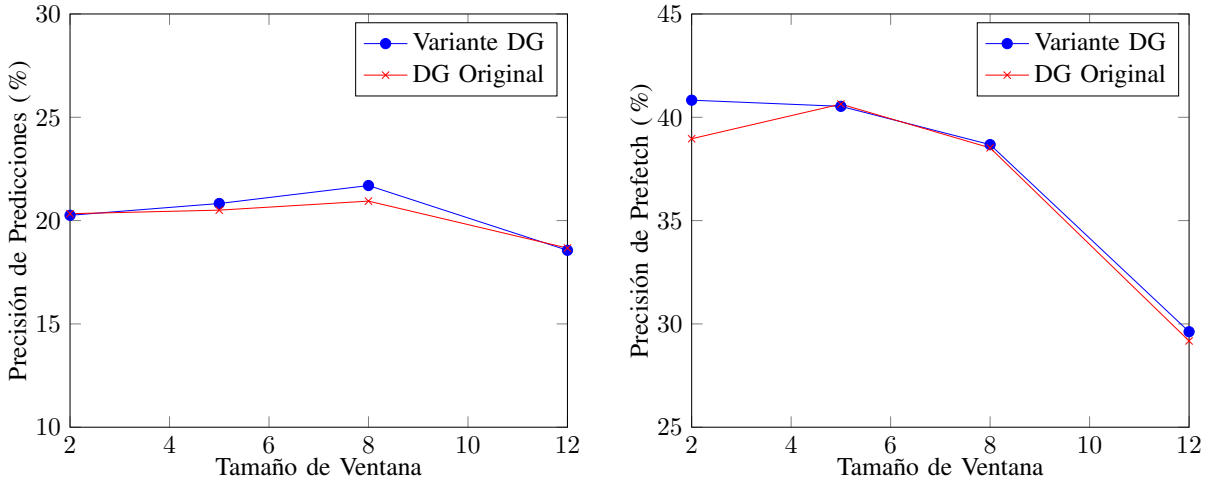


Figura 4. Cálculo de probabilidades con la variante del algoritmo DG considerando los aciertos de predicciones del ciclo anterior.



(a) Precisión de predicciones.

(b) Precisión de *prefetch*.

Figura 5. Precisión de predicciones y de *prefetch* con tamaño de ventana variable y confianza mínima de 40 %.

al ciclo. El tiempo que dura cada ciclo es determinado por las personas que administran el sistema de *prefetching*, acorde a los criterios que éstas definan. De igual manera, la duración puede ser definida de manera automatizada por programas.

La variante propuesta del algoritmo DG considera para el cálculo de probabilidades los aciertos de predicciones del sistema de *prefetching*. Para ello, en las actualizaciones del modelo se otorgará la máxima probabilidad a las transiciones cuyos recursos predichos hayan sido solicitados por los clientes en el ciclo anterior. La probabilidad de las transiciones que no hayan resultado en aciertos de predicciones serán calculadas conforme al algoritmo DG original, basado en la cantidad de ocurrencias de los nodos y arcos (Figura 4).

4. Resultados

4.1. Validación de la Solución Propuesta

Se debe considerar lo señalado en [7], donde se presenta la problemática en cuanto a las comparaciones entre investigaciones realizadas en el área de los sistemas de *prefetching* en la navegación *web*, debido a las diferentes metodologías utilizadas. Estas diferencias impiden una evaluación comparativa apropiada entre trabajos. Así también, no existe una utilización uniforme de las métricas para evaluar el desempeño de las soluciones presentadas en los trabajos. Por estas razones, para la evaluación de la solución propuesta se utilizaron las definiciones de métricas establecidas en [7]. La evaluación se realizó desde las siguientes perspectivas: sistema de *prefetching*, consumo de recursos computacionales y de red, *caché* y usuarios.

En cuanto al sistema de *prefetching*, con las siguientes métricas se midió el desempeño de la variante del algoritmo DG propuesta en este trabajo comparándola con la versión original del algoritmo DG:

- *Precisión de predicciones:* Razón entre los aciertos de predicciones y el total de predicciones realizadas.
- *Precisión de prefetching:* Razón entre los aciertos de *caché* a causa de predicciones y el total de recursos almacenados en *caché* por solicitudes del sistema de *prefetching*.

Para evaluar el impacto respecto al consumo de recursos computacionales y de red, se compararon los siguientes aspectos con y sin el sistema de *prefetching* en funcionamiento: utilización de memoria y del ancho de banda de red.

A nivel de *proxy*, se compararon la cantidad de recursos almacenados en *caché* con y sin el sistema de *prefetching* funcionando.

Desde el punto de vista de los usuarios, se compararon los tiempos de descarga de recursos *web* con y sin el sistema de *prefetching* en funcionamiento.

4.2. Caso de Estudio

La solución propuesta ha sido aplicada en los segmentos de red de cuatro laboratorios de la Facultad de Ciencias y Tecnología del Campus de Santa Librada de la Universidad Católica “Nuestra Señora de la Asunción”. Las redes son de tipo *Ethernet* de 100 Mbps. Cada laboratorio cuenta con una cantidad de entre 35 a 50 *hosts*. El sistema fue implementado en el enrutador denominado Policarpo. Este enrutador es una PC con sistema operativo CentOS versión 6.3, procesador doble núcleo Intel Pentium 4 de 3 GHz y 768 MB de Memoria de Acceso Aleatorio (*Random Access Memory* - RAM).

Para realizar las pruebas del sistema, se han realizado capturas del tráfico de red de los laboratorios durante una semana. Se han utilizado las capturas de los primeros seis días para la generación del modelo inicial y se han reproducido las sesiones de navegación del séptimo día. Para esto, se ha implementado un programa que reproduce las solicitudes de los clientes conforme al ritmo en que fueron realizadas originalmente.

En cuanto a la duración de los ciclos, el ciclo de preparación duró el tiempo requerido para la construcción del modelo inicial. El ciclo inicial y los ciclos posteriores tuvieron una duración de 10 minutos cada uno. Éste ha resultado un valor apropiado en las pruebas realizadas.

El tiempo de validez de las predicciones fue de 1 ciclo, de manera que las predicciones hayan sido generadas en base a la versión más reciente del modelo predictivo. Conforme al concepto de localidad temporal de referencia, se buscó evitar casos como, por ejemplo, si una URL fue predicha a las 8 A.M. y la misma fue

solicitada recién a las 2 P.M. ésta situación haya sido considerada una predicción exitosa.

4.3. Análisis de los Resultados

4.3.1. Sistema de Prefetching

Las primeras pruebas realizadas buscaron determinar la influencia del tamaño de la ventana de accesos. Para éstas pruebas se ha utilizado una confianza mínima del 40 %.

En la Figura 5a se compara la precisión de predicciones obtenida con la variante del algoritmo DG y el algoritmo DG original. Ambos algoritmos han sido implementados por los autores. Se puede observar que la variante ha tendido a obtener mayores niveles de precisión que el algoritmo original.

De las predicciones realizadas, las que no resultaron en aciertos se debieron a las siguientes razones:

- *Recursos predichos tardíamente:* Predicciones de recursos ya solicitados por los clientes. Esto se dio mayormente cuando un mismo recurso fue solicitado por un cliente y por el sistema de *prefetching*, y resultó procesada la solicitud del cliente en primer lugar.
- *Recursos predichos no solicitados:* Recursos predichos que no llegaron a ser solicitados por los clientes.

A medida que el tamaño de ventana aumentó hasta llegar al tamaño 8, el nivel de precisión de predicciones ha crecido. Con tamaños de ventana mayores a 8, la precisión mostró una tendencia decreciente. Esto se debió a que de la mayor cantidad de predicciones generadas para cada consulta, fueron menos las que resultaron en aciertos por las razones citadas en el párrafo anterior.

En la Figura 5b se compara la precisión de *prefetch* obtenida con la variante del algoritmo DG y el algoritmo DG original. Se puede observar que con tamaños de ventana 2, 8 y 12 la variante del algoritmo DG obtuvo mayor precisión de *prefetch*. Con ventana de tamaño 5 el algoritmo DG original obtuvo mayor precisión.

Las razones de la variación en la precisión de *prefetch* son las siguientes:

- *Solicitudes de prefetch tardías:* Recursos predichos que fueron solicitados por el motor de *prefetching* inmediatamente después o en simultáneo a clientes solicitando los mismos recursos.
- *Recursos no almacenables:* Recursos predichos solicitados por el motor de *prefetching* que no resultaron almacenables en *caché*. La determinación de si un recurso puede ser o no almacenado en *caché* se basa en las cabeceras HTTP de las respuestas enviadas por los servidores.

Respecto a las solicitudes de *prefetch* tardías, cuando un recurso es solicitado en simultáneo por el motor de *prefetching* y por un cliente, cualquiera de las dos solicitudes puede ser completada en primer lugar. Ésto se debe a factores como el tiempo de procesamiento de cada solicitud, el tiempo de generación de las predicciones y las solicitudes de los recursos predichos (factores relacionados al sistema de *prefetching*). No obstante, existen factores externos al sistema de *prefetching* como el consumo de recursos de la PC donde se ejecuta el sistema, la carga y velocidad de los enlaces de red y el tiempo de respuesta de los servidores, entre otros factores, los cuales configuran una variedad de escenarios que repercuten en cuál de las solicitudes resulta completada en primer lugar.

Considerando los resultados obtenidos, en las siguientes pruebas se ha utilizado un tamaño de ventana de 2, ya que con éste valor se obtuvieron los mayores niveles de precisión de *prefetch*. Así también, utilizando un tamaño de ventana reducido, el modelo generado será más pequeño al relacionar cada nodo del grafo con una cantidad menor de nodos.

Habiendo determinado el tamaño de ventana a ser utilizado en las siguientes pruebas, se comparó la variante del algoritmo DG y el algoritmo DG original modificando la confianza mínima requerida para la generación de predicciones (Figura 6a). Se observa nuevamente que la variante ha tendido a obtener mayores niveles de precisión de predicciones que el algoritmo original. Al aumentar la confianza mínima requerida, las predicciones generadas representan una correlatividad más fuerte entre un recurso solicitado por un cliente y los recursos predichos a partir de éste. La variante DG alcanzó una mayor precisión de predicciones con una confianza mínima de 80 %

En la Figura 6b se compara la precisión de *prefetch* para el algoritmo DG original y la variante. Se observa que la variante obtuvo mayores niveles de precisión de *prefetch* que el algoritmo DG original. Los valores han variado en las distintas pruebas debido a la influencia de los factores externos al sistema de *prefetching*.

4.3.2. Consumo de Recursos Computacionales y de Red

En cuanto al consumo de recursos computacionales, se analizó la cantidad de memoria utilizada por el sistema de *prefetching* durante su ejecución (implementado como un único proceso con varios hilos de ejecución (*threads*)). En la Figura 7a se observa la variación del Tamaño de Conjunto Residente (*Resident Set Size* - RSS) en memoria del algoritmo DG original y de la variante. RSS indica la cantidad de RAM ocupada por un programa en un momento determinado. Utilizando como referencia la prueba con tamaño de ventana 2 y confiabilidad mínima de 40 %, puede observarse que los valores de RSS fueron similares para ambos algoritmos. Con una diferencia final de 60 *kilobytes* más, la variante del algoritmo DG obtuvo una mayor precisión de *prefetch* (40,83 % contra 38,96 % del algoritmo DG original).

Un aspecto a considerar es que en los sistemas Linux el RSS de un programa puede no representar el total de espacio de memoria utilizado por dicho programa. Los sistemas Linux utilizan la técnica de la memoria virtual mediante la cual páginas de memoria virtual que no ocupan espacio en la RAM ocupan espacio en el disco duro. En la Figura 7b se observa que el algoritmo DG original como la variante utilizaron una cantidad similar de Tamaño de Memoria Virtual (*Virtual SiZe* - VSZ) a lo largo de su ejecución. Con una diferencia de 276 *kilobytes* más de memoria virtual, la variante del algoritmo DG obtuvo mayor precisión de *prefetch* (40,83 % contra 38,96 % del algoritmo DG original).

Desde la perspectiva de los recursos de red, se analizó el ancho de banda consumido en pruebas con y sin el sistema de *prefetching* en funcionamiento. En las Figura 8 se observa el ancho de banda saliente durante la prueba. Cada prueba tuvo una duración aproximada de 9 horas.

En las primeras 3 horas, se observó una mayor cantidad de picos de ancho de banda cuando se utiliza el sistema (Figura 8b). Esto se debió a que al inicio de la prueba la *caché* está vacía y varias de las solicitudes del motor de *prefetching* fueron respondidas desde los servidores en Internet. El *software* del *proxy* guardó en *caché* las respuestas almacenables.

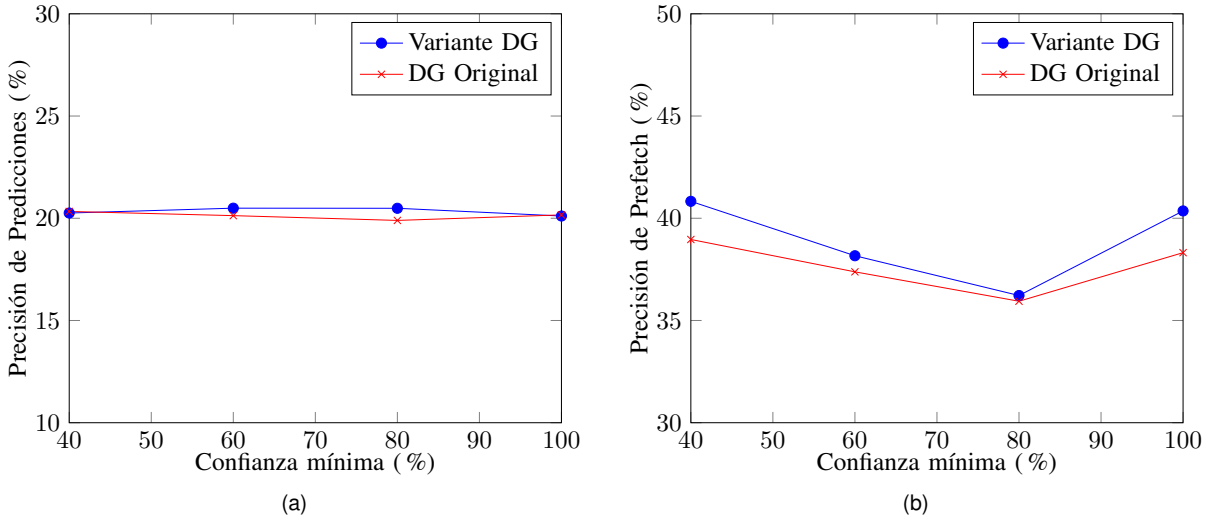


Figura 6. Precisión de predicciones(a) y de *prefetch* (b) con confianza mínima variable y tamaño de ventana 2.

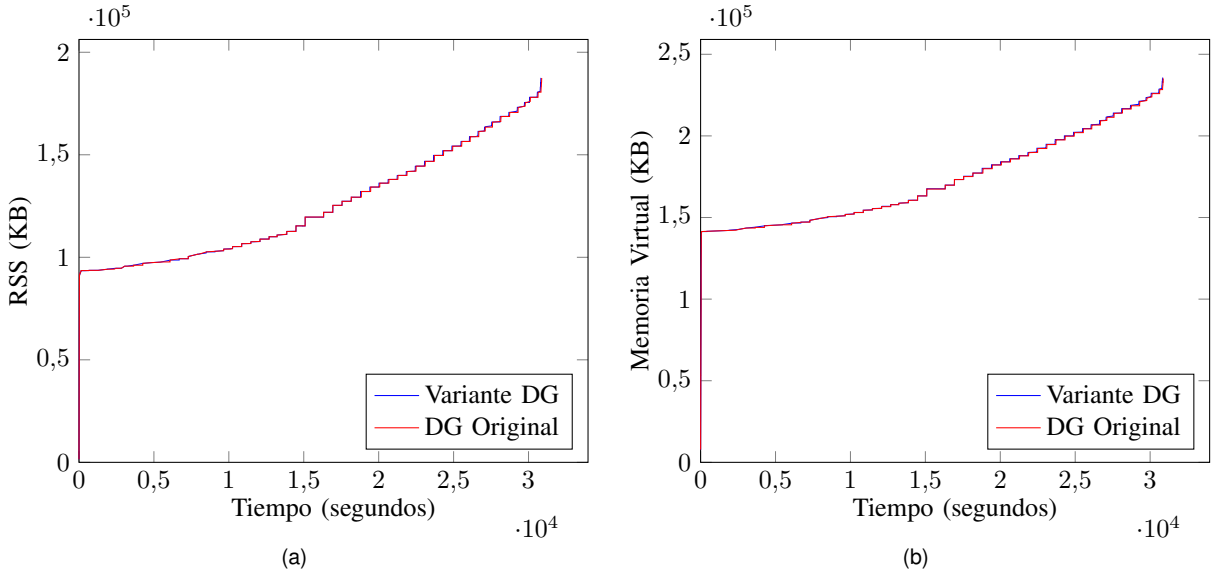


Figura 7. Variación de RSS (a) y VSZ (b) en prueba con tamaño de ventana 2 y confiabilidad mínima 40 %.

Entre las horas 4, 5 y 6 se concentró una mayor cantidad de clientes realizando solicitudes. Se observa que sin el sistema, el ancho de banda utilizado alcanzó picos entre los 14.000 y los 20.000 segundos desde el inicio de la prueba (Figura 8c). Con el sistema en funcionamiento los picos fueron menores, el ancho de banda consumido fue menor y se distribuyó de manera más uniforme en el intervalo (Figura 8d). Ésto se debió a que varias solicitudes de los clientes fueron servidas desde la *caché* y no desde Internet.

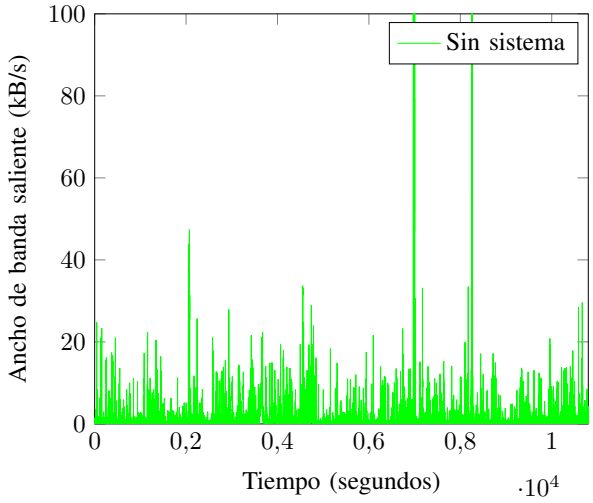
En las últimas 3 horas puede observarse picos de ancho de banda consumido con el sistema de *prefetching* (Figura 8f). No obstante, el ancho de banda saliente sin el sistema ha tendido a ser mayor (Figura 8e).

4.3.3. Caché

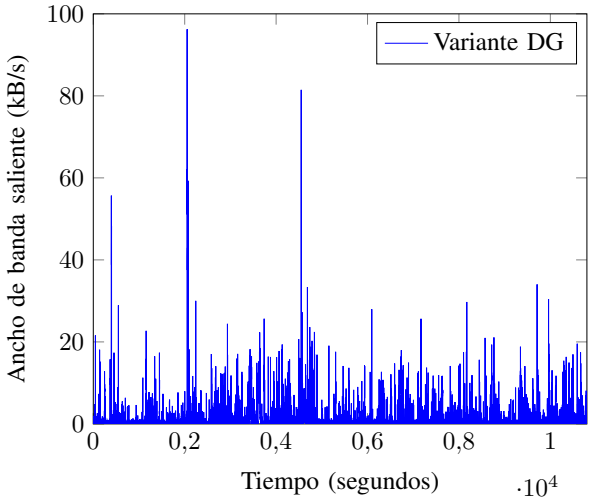
Para analizar la incidencia del sistema de *prefetching* en la *caché* del *proxy*, se verificaron la cantidad de recursos almacenados en *caché* durante las pruebas. Las cantidades consideradas en esta sección indican los recursos que en algún momento estuvieron en *caché*. Éstos recursos pudieron haber sido liberados de la *caché* y almacenados de vuelta en múltiples ocasiones durante las pruebas.

En un escenario sin sistema de *prefetching*, todos los recursos almacenados en *caché* son a causa de solicitudes realizadas por los clientes. Con un sistema de *prefetching* funcionando, los recursos además pueden ser almacenados por las solicitudes realizadas por el sistema basado en las predicciones.

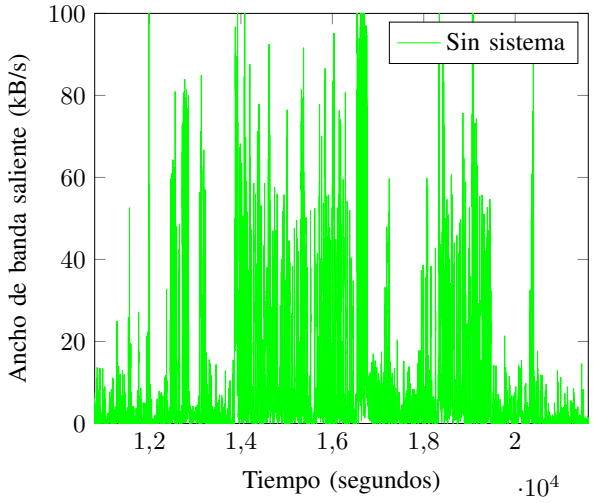
En la Figura 9 puede observarse que utilizando un sistema de *prefetching*, sea con el algoritmo DG original o la variante, la cantidad de recursos guardados en *caché* a causa de las solicitudes



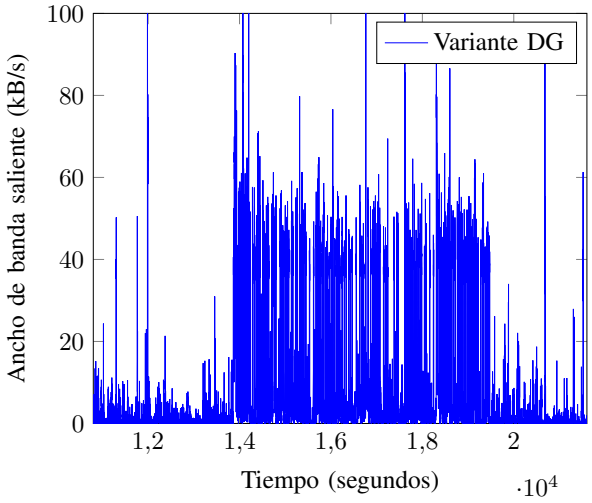
(a) Prueba sin sistema. Horas 1 a 3.



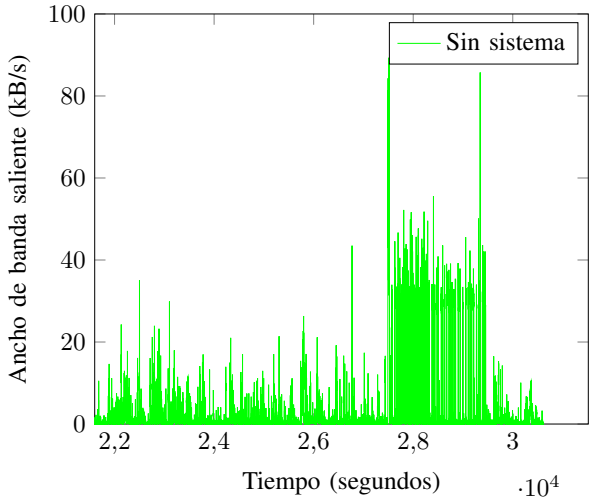
(b) Prueba con sistema. Horas 1 a 3.



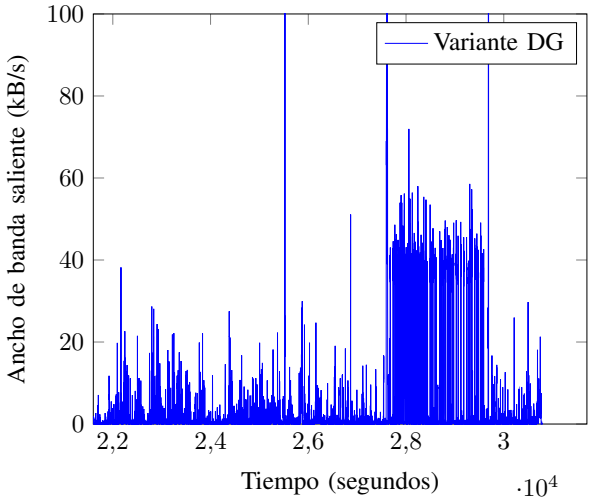
(c) Prueba sin sistema. Horas 4 a 6.



(d) Prueba con sistema. Horas 4 a 6



(e) Prueba sin sistema. Horas 7 a 9.



(f) Prueba con sistema. Horas 7 a 9.

Figura 8. Ancho de banda saliente en la prueba con la variante de DG (tamaño de ventana 2 y confiabilidad mínima 40 %) y sin sistema de *prefetching*.

de los clientes decreció. Esto se debió a que parte de los recursos son almacenados a causa de las solicitudes de *prefetch* del sistema.

La cantidad de recursos almacenados en *caché* por la variante y el algoritmo DG fueron similares. No obstante, la variante almacenó una menor cantidad de recursos en relación al total de recursos almacenados.

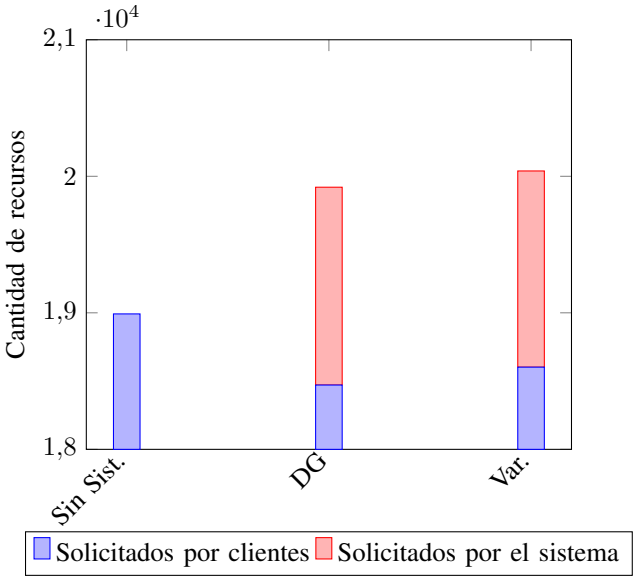


Figura 9. Recursos almacenados en caché sin y con el sistema de *prefetching* con ventana de tamaño 2 y confiabilidad mínima 40 %.

4.3.4. Usuarios

Con respecto a los tiempos de respuesta de las solicitudes de los usuarios, se verificaron los tiempos de descarga de recursos *web* con y sin el sistema de *prefetching*.

En el Cuadro 1 se observa que, para recursos predichos que fueron servidos desde las instalaciones del Campus de Santa Librada de la Universidad Católica "Nuestra Señora de la Asunción", se lograron reducciones de hasta un 88 % en los tiempos de descarga. Respecto a recursos predichos que fueron servidos desde redes externas al campus, en el Cuadro 2 se obtuvieron reducciones que van desde un 13 % hasta un 99 % en los tiempos de descarga.

Cuadro 1. TIEMPOS (EN MILISEGUNDOS) DE DESCARGA CON Y SIN EL SISTEMA PROPUESTO. RECURSOS SERVIDOS DESDE EL CAMPUS DE SANTA LIBRADA DE LA UNIVERSIDAD CATÓLICA "NUESTRA SEÑORA DE LA ASUNCIÓN".

Recurso	Sin sistema	Con sistema	Reducción
Archivo *css*	9	4	55,56 %
Archivo *css*	26	3	88,46 %
Imagen *png*	8	3	62,50 %
Imagen *jpg*	7	3	57,14 %
Imagen *jpg*	21	3	85,71 %
Archivo *css*	24	3	87,50 %
Archivo *pdf*	44	6	86,36 %
Archivo *pdf*	22	7	68,18 %

Cuadro 2. TIEMPOS (EN MILISEGUNDOS) DE DESCARGA CON Y SIN EL SISTEMA PROPUESTO. RECURSOS EXTERNOS AL CAMPUS DE SANTA LIBRADA DE LA UNIVERSIDAD CATÓLICA "NUESTRA SEÑORA DE LA ASUNCIÓN".

Recurso	Sin sistema	Con sistema	Reducción
Código *javascript*	203	175	13,79 %
Archivo *css*	36	4	88,89 %
Código *javascript*	2157	511	76,31 %
Imagen *jpg*	1501	275	81,68 %
Imagen *gif*	867	3	99,65 %
Imagen *jpg*	244	3	98,77 %

5. Conclusiones, Aportes y Trabajos Futuros

5.1. Conclusiones

Debido a que varias de las investigaciones en el área del *prefetching* no consideran los aciertos y fallos de las predicciones realizadas al momento de actualizar el modelo predictivo, se propuso un sistema de *prefetching* que considera el historial de aciertos de las predicciones. Los resultados muestran que el enfoque propuesto con la variante del algoritmo DG obtuvo mayores niveles de precisión de predicciones y *prefetch* que con el algoritmo DG original sin incurrir en un incremento importante de la memoria utilizada. Así también, la utilización del ancho de banda se atenuó una vez que la *caché* fue poblándose. Para los recursos predichos y almacenados anticipadamente en *caché*, se alcanzaron reducciones de incluso más del 90 % en los tiempos de descarga de los recursos que fueron solicitados por los usuarios.

5.2. Aportes

Los principales aportes de este trabajo son:

- Esquema para la implementación de un sistema de *prefetching* en *proxies*.
- Definición de una variante del algoritmo DG que obtuvo una mayor precisión de predicciones y *prefetch* sin incurrir en un incremento importante de la memoria utilizada.
- Definición de un formato de reporte de navegación *web*, basado en la técnica DPI, orientado a la creación de modelos predictivos de navegación.
- Desarrollo de pruebas de un sistema de *prefetching* en un entorno real, expuesto a los factores propios de una infraestructura de red.

5.3. Trabajos Futuros

- Implementación de un mecanismo de reducción del tamaño del modelo predictivo.
- Implementación de inteligencia para determinar la duración de los ciclos y validez de las predicciones
- Implementación de la solución propuesta en clientes y servidores

Referencias

[1] N. Nandini, H. K. Yogish, and G. T. Raju, "Pre-fetching techniques for effective web latency reduction; a survey," in *2013 Africon*, Sept 2013, pp. 1–6.

[2] J. Parmar and J. Verma, "State-of-artsurvey of various web prefetching techniques," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3, Aug 2016, pp. 1–7.

[3] S. Setia, Jyoti, and N. Duhan, "Survey of recent web prefetching techniques," in *2013 International Journal of Research in Computer and Communication Technology*, vol. 2, no. 12, Dec. 2013. [Online]. Available: http://ijrcct.org/index.php/ojs/article/view/499

[4] V. N. Padmanabhan and J. C. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *COMPUTER COMMUNICATION REVIEW*, vol. 26, pp. 22–36, 1996.

[5] A. Bestavros, "Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time for Distributed Information Systems," in *Proceedings of ICDE'96: The 1996 International Conference on Data Engineering*, New Orleans, Louisiana, Marzo 1996.

[6] J. Domenech, J. A. Gil, J. Sahuquillo, and A. Pont, "Using current web page structure to improve prefetching performance," *Comput. Netw.*, vol. 54, no. 9, pp. 1404–1417, Junio 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.11.016

[7] J. Domènech, "Evaluation, Analysis and Adaptation of Web Prefetching Techniques in Current Web," Master's thesis, Universidad Politécnica de Valencia, Marzo 2007.

[8] A. A. Markov, "The theory of algorithms," USSR Academy of Sciences, 1954.

[9] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing Reference Locality in the WWW," in *Proceedings of 1996 International Conference on Parallel and Distributed Information Systems (PDIS '96)*, Diciembre 1996, pp. 92–103. [Online]. Available: http://www.cs.bu.edu/faculty/crovella/paper-archive/pdis96.ps

[10] A. Bestavros and S. Jin, "Sources and Characteristics of Web Temporal Locality," *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, vol. 0, p. 28, 2000.

[11] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, "A Data Mining Algorithm for Generalized Web Prefetching," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 15, no. 5, pp. 1155–1169, 2003.

[12] J. Griffioen and R. Appleton, "Reducing file system latency using a predictive approach," 1994, pp. 197–207.

[13] K. M. Curewitz, P. Krishnan, and J. S. Vitter, "Practical prefetching via data compression," *SIGMOD Rec.*, vol. 22, no. 2, pp. 257–266, Junio 1993. [Online]. Available: http://doi.acm.org/10.1145/170036.170077

[14] L. Fan, P. Cao, W. Lin, and Q. Jacobson, "Web prefetching between low-bandwidth clients and proxies: potential and performance," in *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '99. New York, NY, USA: ACM, 1999, pp. 178–187. [Online]. Available: http://doi.acm.org/10.1145/301453.301557

[15] J. Pitkow and P. Pirolli, "Mining longest repeating subsequences to predict world wide web surfing," in *Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems - Volume 2*, ser. USITS'99. Berkeley, CA, USA: USENIX Association, 1999, pp. 13–13. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251480.1251493

[16] X. Chen and X. Zhang, "A Popularity-Based Prediction Model for Web Prefetching," *Computer*, vol. 36, no. 3, pp. 63–70, Marzo 2003. [Online]. Available: http://dx.doi.org/10.1109/MC.2003.1185219

[17] C. D. Gracia and S. Sudha, "Meppm- memory efficient prediction by partial match model for web prefetching," in *2013 3rd IEEE International Advance Computing Conference (IACC)*, Feb 2013, pp. 736–740.

[18] N. Cascarano, "Application Layer Traffic Classification," Master's thesis, Politecnico di Torino, Noviembre 2007.

[19] "Extensible Markup Language (XML)," Sitio Web. http://www.w3.org/XML/. [Online]. Available: http://www.w3.org/XML/