SINCRONIZACIÓN DE DATOS DE USUARIO EN REDES DE PRÓXIMA GENERACIÓN

EIVAR EDER ARMERO LUNA

DIEGO FERNANDO RODRIGUEZ CHAMORRO

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Servicios Avanzados de Telecomunicaciones
Popayán, Septiembre de 2008

SINCRONIZACIÓN DE DATOS DE USUARIO EN REDES DE PRÓXIMA GENERACIÓN

EIVAR EDER ARMERO LUNA

DIEGO FERNANDO RODRIGUEZ CHAMORRO

Trabajo de Grado presentado como requisito para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Directora: I.E. Mary Cristina Carrascal Reyes

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Servicios Avanzados de Telecomunicaciones
Popayán, Septiembre de 2008

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

LISTA DE ACRÓNIMOS

3G (3rd Generation)
3GPP (Third Generation Partnership Project)
3GPP2 (Third Generation Partnership Project 2)
AAA (Authentication, Authorization, and Accounting)
ACID (Atomicity, Consistency, Isolation and Durability)
AGC (Aplicación Generadora de Coordenadas)
API (Application Programming Interface
AS (Application Server)
ATIS (Alliance for Telecommunications Industry Solutions),
AuC (Authentication Centre)
AVPs (Attribute Value Pairs)
AGI (Aplicación Generadora de Información)
BGCF (Breakout Gateway Control Function)
BS (Base Station)
BSC (Base Station Control)
CAMEL (Customized Applications for Mobile networks Enhanced Logic)
CDMA (Code Division Multiple Access)
CSCF (Call Session Control Function)
DBMS (Database Management System)
DDB (Distributed Database)
DDBMS (Distributed Database Management System)
DDBS (Distributed Database System)
DNS (Domain Name Server)
EIR (Equipment Identity Register)
ETSI (European Telecommunications Standards Institute)
FHoSS (FOKUS HSS)
FIFO (First In First Out)
GGSN (Gateway GPRS Support Node)
GPL (General Public License)
GPS (Global Positioning System)
GPRS (General Packet Radio Service)
GSM (Global System for Mobile)
gsmSCF (GSM Service Control Function)
HLR (Home Location Register)
HSS (Home Subscriber Server)
I-CSCF (Interrogating – CSCF)
IDE (Integrated Development Environment)
IETF (Internet Engineering Task Force)
IMT-2000 (International Mobile Telecommunications-2000)
IMS (IP Multimedia Subsystem),
IM-SSF (IP Multimedia Service Switching Function)
IP (Internet Protocol)
ISC (IMS Service Control)

ISDN (Integrated Services Digital Network)
ITU (International Telecommunication Union)
JDBC (Java Database Connectivity)
JRE (Java Runtime Environment)
JSLEE (Java Service Logic Execution Environment)
JVM (Java Virtual Machine)
LDAP (Lightweight Directory Access Protocol)
MG (Media Gateway)
MGC (Media Gateway Controller)
MGCF (Media Gateway Control Function)
MGW (Media Gateway)
MRF (Media Resource Function)
MRFC (MRF Controller)
MRFP (MRF Processor)
MS (Mobile Station)
MSC (Mobile Switching Center)
NGN (Next Generation Networks)
NGN-FG (NGN Focus Group)
NSS (Network Switching Subsystem)
OMA (Open Mobile Alliance)
OSA-SCS (Open Service Access-Service Capability Server)
PCM (Pulse Code Modulation)
P-CSCF (Proxy - CSCF)
PNA (Push-Notifications-Answer)
PNR (Push-Notification-Request)
PSTN (Public Switched Telephone Network)
PUA (Profile-Update-Answer)
PUR (Profile-Update-Request)
QoS (Quality of Service)
RAIDb (Redundant Array of Inexpensive Disks)
RDSI (Red Digital De Servicios Integrados)
RPC (Remote Procedures Call)
RTP (Real-time Transport Protocol)
RTPC (Red Telefónica Pública Conmutada)
SCP (Service Control Point)
S-CSCF (Serving – CSCF)
SDK (Software Development Kit)
SDP (Service Data Point)
SGW (Signaling Gateway)
SGSN (Serving GPRS Support Node)
SIP (Session Initiation Protocol)
SLF (Subscription Locator Function)
SNA (Subscribe-Notification-Answer)
SNR (Subscribe-Notifications-Request)
SS7 (Sistema de Señalización No. 7)

TISPAN (Telecoms & Internet converged Services & Protocols for Advanced Networks)
UDA (User-Data-Answer)
UDR (User-Data-Request)
UMTS (Universal Mobile Telecommunications System)
URL (Universal Resource Locator)
VDB (Virtual Database)
VLR (Visitor Location Register)
VSCAST (View Synchronous Broadcast)
WAN (Wide Area Network)
WLSS (Weblogic Sip Server)
WLSS GW (Gateway WLSS)

INTRODUCCIÓN

La tendencia mundial de desarrollo e innovación dirigida hacia la convergencia tecnológica, ha fomentado la demanda de servicios que presenten características de accesibilidad en cualquier lugar, en cualquier momento y desde cualquier dispositivo. Esto, ha llevado a las industrias, organismos de estandarización, gobiernos y universidades a enfocar sus investigaciones en posibilitar la prestación de este tipo de servicios desde el enfoque que a cada uno de estos sectores le concierne.

Así, los diferentes actores han tomado partido en el desarrollo de dispositivos, aplicaciones, infraestructura de redes, mercados y políticas para satisfacer las necesidades de los usuarios de servicios de telecomunicaciones que viven en un mundo globalizado y competitivo.

Entonces, se hacen presentes las NGN (*Next Generation Networks*) con un *core* de red soportado en el uso de IP (*Internet Protocol*) y por ende dentro del dominio de la conmutación de paquetes. De esta forma, y junto con otros trabajos en el sector, se empieza a hacer realidad la mencionada convergencia de redes y servicios.

Los servicios convergentes, es decir la integración de los servicios de Internet, multimedios, presencia, etc., en el servicio de transmisión de voz, además de poder ser accedidos en diferentes circunstancias, también presentan características relevantes a la hora de su implementación y despliegue, como la posibilidad de su personalización por parte del usuario. Esto, ha llevado a realizar estudios enfocados en el mejoramiento del manejo de la información relacionada con el usuario, la cual se genera y consume en varios nodos heterogéneos presentes en las NGN.

Siguiendo la línea de los avances de la comunidad científica nacional e internacional, en la Facultad de Ingeniería Electrónica y Telecomunicaciones (FIET) de la Universidad del Cauca, ha comenzado a abrirse camino en el área de la sincronización de información de usuarios en las NGN con estudios como el presentado en este documento.

Para abordar detalladamente el proyecto, en el Capítulo I se presenta una descripción del estado del arte de los mecanismos de sincronización de información de usuario en las redes de telecomunicaciones no convergentes, donde se incluyen las redes de telefonía, las redes celulares, redes móviles de paquetes y las redes de datos en general.

Posteriormente, en el Capítulo II se enfatiza en el IMS (*IP Multimedia Subsystem*), como el estándar de las NGN propuesto para hacer realidad la convergencia. Se describe su arquitectura y el mecanismo de sincronización de información usado en una de sus interfaces, con el objetivo de tener un punto de partida para el análisis del problema, el planteamiento de la solución y la validación de la misma.

En el Capítulo III se enfatiza en el problema y la propuesta de solución. El Capítulo IV se enfoca en la implementación de un prototipo para validar la solución planteada, en el Capítulo V se exponen los resultados obtenidos en las pruebas y un análisis de los mismos y finalmente, en el Capítulo VI se condensan las conclusiones del proyecto y se plantean algunos trabajos que pueden ser desarrollados como continuación de la investigación en esta área.

CAPÍTULO I
ESTADO DEL ARTE DE LA SINCRONIZACIÓN DE INFORMACIÓN EN REDES DE TELECOMUNICACIONES

1.1. DEFINICIÓN DEL CONCEPTO DE SINCRONIZACIÓN DE INFORMACIÓN

Según el diccionario de la Real Academia de la Lengua Española, sincronizar se define como "hacer que coincidan en el tiempo dos o más movimientos o fenómenos". Trasladando esa definición al contexto del manejo de la información en redes de las telecomunicaciones se puede decir que sincronizar información es hacer que ésta coincida en el tiempo en ciertos elementos de red.

El intervalo de tiempo que se da para que dicha información se sincronice es relativo al tipo de servicio y los requerimientos que puedan presentar. Por ejemplo, para la presentación de servicios de tiempo real, la sincronización debe hacerse en el menor lapso posible.

Así, en las redes de telecomunicaciones, la información que es generada o modificada en algún elemento de red, se debe replicar a los demás nodos que la requieren en una forma consistente y en un intervalo de tiempo dado, garantizando de esta forma la sincronización de la misma.

1.2. SINCRONIZACIÓN DE INFORMACIÓN DE USUARIO EN LA RED TELEFÓNICA PÚBLICA CONMUTADA (RTPC) / RED DIGITAL DE SERVICIOS INTEGRADOS (RDSI)

En una RTPC/RDSI la información de los suscriptores de línea se encuentra almacenada en una base de datos presente en la central telefónica que es donde se reúnen las funciones de conmutación y control. Esta información se comparte con otros módulos de la central telefónica para ofrecer recursos adicionales a un suscriptor dependiendo de los servicios a los que se haya inscrito y se puede compartir con redes de otros operadores para procesos de tarificación. En los dos casos se hace uso de los protocolos del SS7 (Sistema de Señalización No. 7) [1].

1.3. SINCRONIZACIÓN DE INFORMACIÓN DE USUARIO EN LA RED INTELIGENTE

La información de suscriptor en la red inteligente se encuentra centralizada en el SDP (*Service Data Point*), que es la base de datos que contiene información sobre el suscriptor y alguna información necesaria para establecer una llamada. Esta información se lee desde el SCP (*Service Control Point*) cuando se requiera en el proceso de llamada y se puede compartir con otros elementos de red haciendo uso de protocolos estandarizados y propietarios [2].

1.4. SINCRONIZACIÓN DE INFORMACIÓN DE USUARIO EN LOS SITEMAS CELULARES

1.4.1. Redes celulares de telefonía móvil [3]

Una red de telefonía móvil que implementa los estándares GSM (*Global System for Mobile*) ó CDMA (*Code Division Multiple Access*), tiene ciertos componentes básicos dentro de su arquitectura, los cuales se aprecian en la Figura 1.

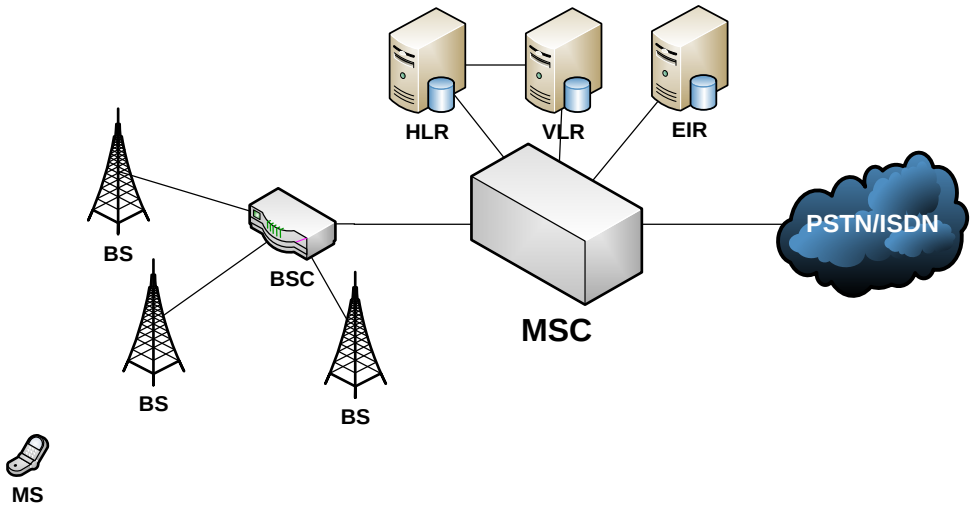

Figura 1. Arquitectura GSM.

- MS (*Mobile Station*): Dispositivo que el usuario usa para conectarse a la red.
- BS (*Base Station*): Estación base a la que se conecta una MS.
- BSC (*Base Station Control*): Elemento que controla una o varias BS.
- MSC (*Mobile Switching Center*): Central que se encarga de las funciones de conmutación para un grupo de MS dentro de su área de influencia.
- HLR (*Home Location Register*): Base de datos que almacena la información relacionada con los usuarios y se encarga de la gestión de los mismos.
- VLR (*Visitor Location Register*): Base de datos que se encarga de la gestión de los usuarios en proceso de *roaming*.
- EIR (*Equipment Identity Register*): Contiene las identidades de los equipos móviles.

Sin tener en cuenta las BS y la BSC, los demás equipos conforman el NSS (*Network Switching Subsystem*) que es la parte central de la red interconectando sus componentes a través de diferentes interfaces y usando en todos los casos SS7, el cual establece sus protocolos dentro del esquema de conmutación de circuitos. EL NSS se puede apreciar en más detalle en la Figura 2.

Para los diferentes procesos de registro de una MS a la red, establecimiento de llamada, terminación de llamada, *roaming*, tarificación, entre otros; se comparte información de usuario entre los nodos del NSS usando SS7 y las interfaces dispuestas.

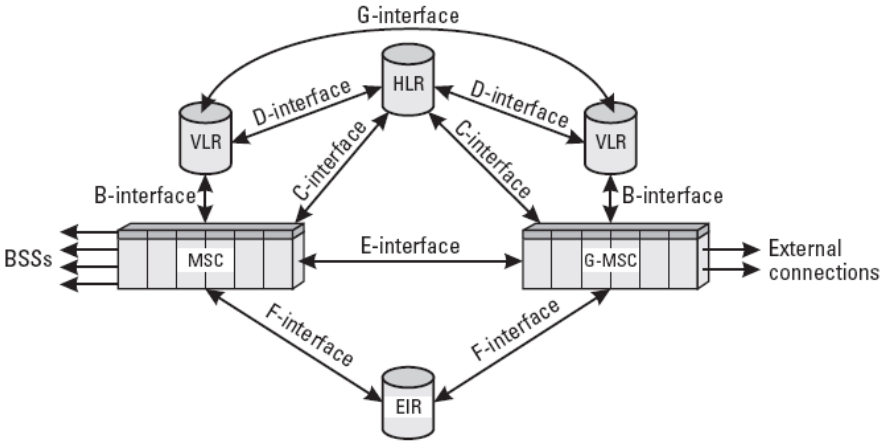

Figura 2. *Network Switching Subsystem* (NSS) [3]

1.4.2. General Packet Radio Service (GPRS) [4]

La red GPRS utiliza la infraestructura de la red GSM para transmitir paquetes de datos. Por lo tanto, es una extensión de conmutación de paquetes sobre GSM, con el objetivo de proporcionar un acceso más eficiente de las redes celulares a las redes de datos.

Para cumplir con este propósito, se introducen dos elementos a la red, los cuales se aprecian en la Figura 3.

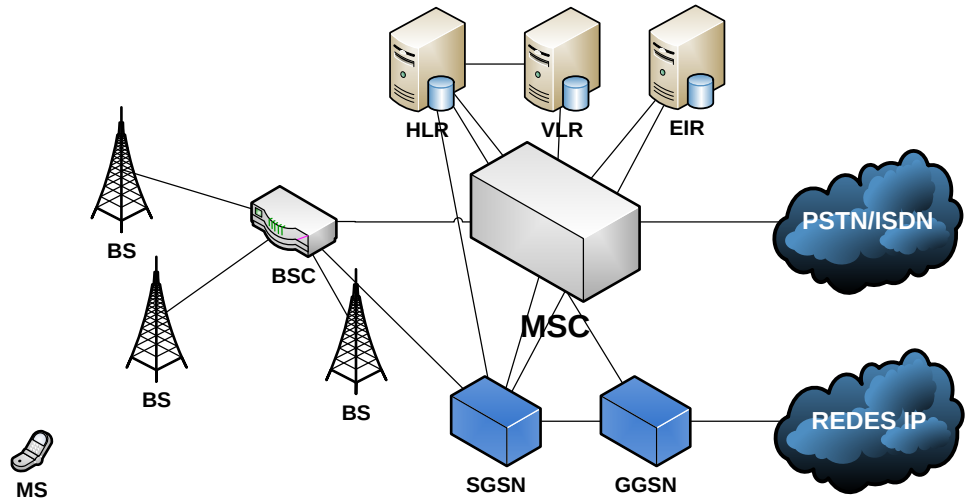

Figura 3. Arquitectura de la red GPRS/GSM.

- SGSN (*Serving* GPRS *Support Node*): se encarga de las funciones de conmutación, enrutamiento y transmisión de los paquetes de datos.
- GGSN (*Gateway* GPRS *Support Node*): es la pasarela GPRS a las redes externas de paquetes.

El SGSN además de conectarse al BSC, también se conecta con el HLR, el MSC, el VLR y el EIR, mientras que el GGSN se conecta con el HLR con el objetivo de compartir información de usuario en los diferentes procesos que se llevan cabo en la red.

Para la sincronización de esta información se han dispuesto varias interfaces entre los componentes y se hace uso del protocolo MAP (*Mobile Application Part*) que hace parte del SS7.

1.4.3. Redes de Tercera Generación 3G – UMTS (Universal Mobile Telecommunications System) Release 4 [4]

Después de adicionar la tecnología GPRS a las redes GSM, el siguiente paso es constituir un núcleo de red totalmente basado el esquema de conmutación de paquetes, por lo que se toma al *Internet Protocol* (IP), como la base sobre la cual se cimenta esa idea.

Así, nace el estándar UMTS (*Universal Mobile Telecommunications System*), pasando por varias versiones en las que se modifica la red desde el acceso hasta el núcleo.

Para el año 2001 se libera la versión 4 del estándar UMTS en la que se consigue un núcleo totalmente basado en IP en el que la voz se transporta sobre este protocolo.

Teniendo como referencia la arquitectura GSM/GPRS, los cambios que se introducen son principalmente en el MSC, donde se dividen las funciones de control y conectividad, encomendándolas al Servidor de Control y al MG (*Media Gateway*) respectivamente. Este último, hace uso del MGC (*Media Gateway Controller*) para proveer una correcta conexión a las redes de conmutación de circuitos.

Como se puede observar en la Figura 4, los protocolos de señalización que se introducen son SIP (*Session Initiation Protocol*) y MEGACO. Sin embargo para la conexión de estos nuevos elementos con el HLR, VLR y EIR para la sincronización de información de usuario se sigue usando SS7.

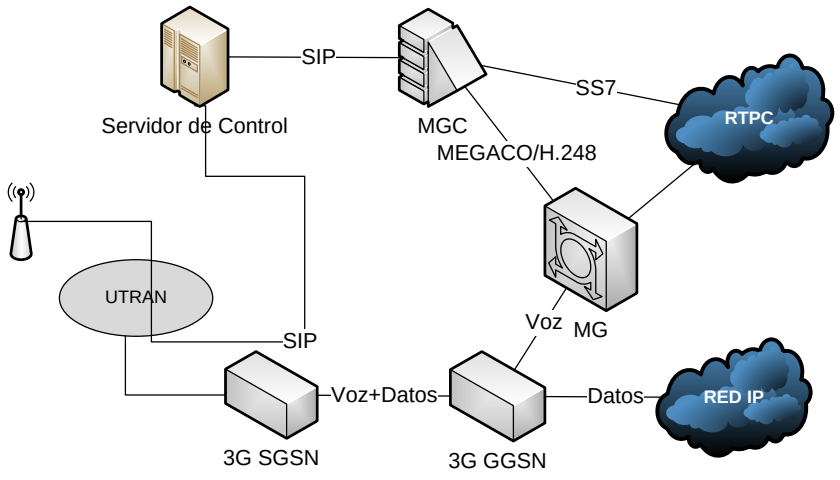

Figura 4. Arquitectura UMTS versión 4 [4].

1.5. SINCRONIZACIÓN DE INFORMACIÓN DE USUARIO EN REDES DE DATOS

Hablar de información de usuario dentro de las redes de datos, es hablar de un sin número de datos que pueden conformar dicha información, todo esto dependiendo del contexto en el que se encuentre. Las bases de datos son la principal forma de almacenamiento de información en este tipo de redes ya que son capaces de almacenar de una forma estructurada una gran cantidad de datos relacionados con uno o más entes, ya sean personas, objetos, sistemas, etc.

La replicación de bases de datos se está convirtiendo día tras día en un proceso crítico para poder ofrecer una alta disponibilidad, confiabilidad y desempeño a las aplicaciones que hacen uso de bases de datos. Sin embargo para poder ofrecer una sincronización de datos útil es necesario superar el problema que se presenta al tratar de mantener la consistencia de la información entre todas las réplicas [5].

La replicación trae consigo ventajas para todo el sistema como lo son la disminución del tráfico de red, a través de la WAN (*Wide Area Network*) en casos donde los clientes se encuentran geográficamente dispersos, lo cual permite optimizar el uso de este recurso, así como también proporciona una mayor tolerancia a fallos ya que si un nodo replicado deja de funcionar correctamente no ocasionara una caída total del sistema, puesto que este seguirá obteniendo los datos de la réplica [6].

Al momento de realizar los procesos de replicación, es necesario recordar que existen dos aspectos fundamentales a tener en cuenta, uno es la trasparencia al usuario final, lo que significa que este no debe ser consciente de todos los procesos que se están llevando a cabo, sino que debe sentir el comportamiento del sistema igual que si estuviese usando una base de datos sin replicar. El otro aspecto es la coherencia que debe existir entre las diferentes replicas, es decir que no debe existir diferencias entre las respuestas proporcionadas por una u otra replica [7].

1.5.1. Técnicas de replicación tradicionales

Las técnicas de replicación son un tema que ha cobrado mucho interés principalmente en dos áreas: los sistemas distribuidos y las bases de datos y aunque los protocolos y mecanismos usados en estas dos áreas son conceptualmente similares, al momento de llevarlos a la práctica terminan siendo muy diferentes debido a la gran cantidad de detalles involucrados [8].

1.5.1.1. Replicación en sistemas distribuidos

- Replicación Activa: También conocida como método de maquinas de estado, es una técnica de replicación no centralizada, su funcionamiento está basado en el hecho de que todas las replicas reciben y procesan la misma secuencia de peticiones de los clientes, es decir que se asume que cuando se provee una misma entrada en el mismo orden a todas las replicas, estas deben responder con la misma salida, lo cual conlleva a que los servidores procesen las solicitudes de una forma determinística.

  La principal ventaja de esta técnica es su simplicidad y transparencia en presencia de fallos, ya que el cliente nunca notará si una réplica falla, esto debido a que las demás están recibiendo las mismas instrucciones y por ende procesando y entregando una respuesta al cliente. Por el contrario que se tenga que trabajar de forma determinística es tal vez su

mayor inconveniente, así como también el hecho que todas las replicas procesen la misma solicitud, ya que esto implica un mayor gasto de recursos.

- Replicación Pasiva: también conocida como replicación de copia primaria, basa su funcionamiento en una réplica primaria a la cual todos los clientes envían sus peticiones, esta las procesa y posteriormente envía mensajes de actualización a las demás réplicas haciendo uso de VSCAST (*View Synchronous Broadcast*)[1], las cuales no deben realizar ningún proceso de la solicitud sino únicamente aplicar los cambios que se hayan dado lugar. Dado este tipo de funcionamiento, esta técnica no necesita de un punto de determinismo, solucionando de esta forma la principal desventaja de la replicación activa [9].

  Al ser necesario que todas las solicitudes las procese un servidor primario, este mecanismo trabaja de forma centralizada lo cual se convierte en su principal desventaja, ya que si bien esto consume menos recursos en comparación con otras técnicas, en el momento que la réplica primaria falle será necesario un procesamiento extra para la reconfiguración del sistema lo cual puede conllevar a problemas de consistencia.

- Replicación Semi-activa: esta es una solución intermedia entre las dos técnicas presentadas anteriormente, esto es que a pesar que todas las replicas reciben las peticiones del cliente, no deben procesarlas de una forma determinística. La decisión de las actualizaciones es tomada por una réplica líder quien posteriormente comunica esta decisión a las demás replicas haciendo uso de VSCAST.

1.5.1.2. Replicación en bases de datos

La replicación en bases de datos puede clasificarse de acuerdo a dos aspectos fundamentales, el primero es en qué momento se realiza la replicación y el segundo es quien es capaz de realizar las actualizaciones en las diversas replicas [8].

Dependiendo del momento en el que se realiza la replicación esta puede clasificarse en *Eager* (temprana) o *Lazy* (Perezosa) [10].

- Replicación *Eager:* también conocida como replicación síncrona, consiste en que las actualizaciones son propagadas dentro del tiempo de ejecución de la transacción solicitada por el cliente, es decir la transacción es recibida por un nodo en especial, y este la ejecuta localmente, luego envía las actualizaciones que hayan tenido lugar a las demás replicas y queda en espera de una respuesta de estas para verificar que no existan problemas de inconsistencia, una vez garantizada la consistencia del sistema, el nodo que recibió la solicitud entrega una respuesta al cliente, en caso de encontrarse problemas de consistencia la transacción es abortada por completo y ningún cambio surte efecto en ninguna replica [8][10][11].

  Esta técnica inherentemente garantiza la consistencia de la información lo cual es su

[1] VSCAST: Una primitiva de *group communication* la cual se define en función de un grupo de procesos y está basada en la noción de una secuencia de vistas del grupo, donde cada una de ellas define la composición del grupo en un tiempo dado.

principal ventaja frente a otro tipo de esquemas, además ofrece una gran tolerancia a fallos, todo esto a costa de un alto tiempo de respuesta debido a la espera que debe realizarse para garantizar dicha consistencia [12].

- Replicación *Lazy:* permite que las transacciones se realicen localmente en el servidor que recibió la petición e inmediatamente después de completadas una respuesta es enviada al cliente. La propagación de las actualizaciones hacia las demás replicas únicamente es enviada después de completada la transacción. Este comportamiento ofrece un bajísimo tiempo de respuesta lo cual sumado con la simpleza de la técnica se toman como el mayor fuerte de esta, sin embargo en contraste con las técnicas *Eager, al no existir* una fase de acuerdo entre todas las replicas la consistencia de la información es el factor débil de este tipo de replicación y por ende muchas veces es necesario hacer uso de técnicas de reconciliación, para corregir los errores presentados durante la replicación [8][10].

Ahora, de acuerdo a quien realiza la las actualizaciones, la replicación puede clasificarse en *Primary Copy* (Copia Primaria) o *Update-anywhere* (Actualización en cualquier punto).

- Replicación *Primary Copy:* en este esquema todas las actualizaciones deben ser realizadas por una réplica primaria y posteriormente son propagadas a las demás replicas, esto implica que únicamente la réplica primaria está en capacidad de actualizar la información, por lo tanto las demás son únicamente replicas de lectura. Esto simplifica enormemente el control de la replicación, ya que únicamente un nodo es responsable de las actualizaciones, sin embargo al estar el control centralizado se genera un cuello de botella para todo el sistema además de una pobre tolerancia a fallos [10].

- Replicación *Update-Anywhere:* al contrario del caso anterior, en estas técnicas las actualizaciones de información pueden ser llevadas a cabo por cualquier replica sin necesidad de pasar por ningún intermediario, esto hace que el sistema tenga un acceso mucho más rápido a la información pero de igual forma implica que los procesos de coordinación de la replicas sea mucho más complejo y la posibilidad de que se generen conflictos entre distintas transacciones es mucho mayor [8].

Teniendo en cuenta la anterior clasificación se pueden obtener una combinación de técnicas dependiendo del momento en que se realiza y de quien realiza la replicación como se puede apreciar en la Figura 5.

Figura 5. Técnicas de replicación en bases de datos.

1.5.2. Técnicas de replicación en evolución

1.5.2.1. *Middleware*

Middleware es un software diseñado para ayudar a manejar la complejidad y la heterogeneidad inherente de los sistemas distribuidos. Este se define como una capa software que se encuentra ubicada sobre el recurso que se desea manejar y por debajo de la aplicación que lo manejará, para lograr esto, un *middleware* provee herramientas para los programadores tales como librerías y APIs (*Application Programming Interface*), lo que reduce significativamente las labores de desarrollo y evita de alguna forma que se cometan errores [13].

Aunque el uso de plataformas *middleware* para replicación de bases de datos ha sido estudiado hace ya algún tiempo, es sólo en los últimos años cuando han cobrado fuerza los estudios y los trabajos de investigación en torno a este tipo de escenarios pensados en brindar soluciones que tengan un alto desempeño y disponibilidad en sitios geográficamente distribuidos .

Típicamente, existen dos alternativas para ofrecer replicación en bases de datos, la primera es realizando extensiones al código fuente del *core* del DBMS (*DataBase Management System*), y la otra es hacer uso de una capa *middleware*. La primera ofrece un alto desempeño debido al poco *overhead* generado, pero es una solución completamente dependiente del proveedor del motor de base de datos lo cual es una limitante en términos de interoperabilidad, portabilidad, migración y escalabilidad.

La segunda alternativa ofrece independencia del proveedor del DBMS, lo cual facilita su interoperabilidad con otros entornos y facilita la portabilidad de aplicaciones además de su migración a versiones más recientes de los motores de bases de datos usados; así pues se podría pensar que estas ventajas compensan el elevado *overhead* producido por este tipo de soluciones [14].

Además de las mencionadas anteriormente los sistemas *middleware* presentan una serie de ventajas que los convierte en una muy buena opción como sistema de replicación de bases de datos, entre las cuales se destacan las siguientes [13]:

- Compatibilidad para interactuar con sistemas heredados (*legacy*), esto dada la capacidad inherente de este tipo de sistemas para trabajar en entornos heterogéneos, lo cual permite realizar una fácil migración y adaptación de aplicaciones y servicios que no fueron concebidos para interoperar con otros, por este motivo estos sistemas algunas veces son conocidos como tecnología "pegante".

- Facilidad de programación, esto significa que los desarrolladores de aplicaciones no necesitan realizar cambios ni aprender nuevos lenguajes de programación para utilizar estos sistemas dado que la mayoría proporcionan librerías con las funcionalidades que proveen, las cuales están escritas en lenguajes conocidos como son Java o C++.

- Distribución en capas, esto significa que pueden existir múltiples capas *middleware* en un sistema dado, esto permite maximizar los beneficios que aportan estos sistemas, así por ejemplo puede existir una capa que proporcione sincronía a través de un servicio de entrega atómica (ver 1.5.2.2) y a su vez esta sea utilizada por otra capa encargada de proveer tolerancia a fallos o balance de carga.

- Gestión de la calidad de servicio (QoS - *Quality of Service*), al ser el dinamismo una característica propia de los sistemas de bases de datos distribuidas, la programación alrededor de estos se torna compleja, es por esta razón que se busca hacer una abstracción de los requerimientos de QoS de las aplicaciones y llevarlos a un bajo nivel, donde sistemas gestores de recursos garantizan el correcto funcionamiento del sistema completo. Las capas *middleware* se adaptan perfectamente a este diseño, permitiendo a los desarrolladores enfocarse en la lógica de las aplicaciones o servicios que están implementando y delegar la complejidad de la gestión de la QoS a las capas inferiores.

1.5.2.2. *Group Communication* [15][16]

Una de las técnicas más utilizadas en los sistemas distribuidos es la Llamada a Procedimientos Remotos (RPC – *Remote Procedures Call*), la cual oculta el envió de los mensajes a otro nodo de la red y hace parecer que el procedimiento se está ejecutando localmente. Esta solución es eficiente pero está diseñada para trabajar en entornos punto – punto, sin embargo existen muchas aplicaciones y servicios que necesitan establecer comunicaciones punto – multipunto lo cual implicaría un elevado número de mensajes si se quisiera utilizar RPC en estos casos. Por ejemplo, si necesita comunicarse con n nodos en la red, entonces serían necesarios n-1 mensajes con sus respectivos ACK, por lo que en total se estarían enviando 2(n-1) mensajes.

Además del *overhead* producido al utilizar únicamente comunicaciones punto – punto, el garantizar la consistencia en sistemas de bases de datos distribuidas es una tarea compleja, ya que para este fin es necesario que los mensajes tengan un orden y lo conserven al llegar a cada uno de los nodos.

Teniendo en cuenta lo anterior surgen nuevas posibilidades de comunicación haciendo uso de protocolos que implementen primitivas de *group communication*, el cual puede ser visto como una forma de proveer comunicación multipunto – multipunto organizando los procesos que se desean comunicar en grupos, por ejemplo un grupo puede consistir en un conjunto de usuarios que desean establecer una partida de un juego en línea. Cada grupo tiene asociado un nombre lógico, así

cuando un proceso desea comunicarse con el grupo lo hace enviando un mensaje al nombre que tenga asignado el grupo.

Además de reducir el *overhead*, este tipo de protocolos garantizan que los mensajes están totalmente ordenados, lo cual es uno de los requisitos básicos que se tienen al momento de realizar replicación en bases de datos.

Entre las principales primitivas que hacen de los protocolos que implementan *group communication* una alternativa eficiente para la replicación, se encuentran las siguientes:

- Confiabilidad, permite a esta clase de protocolos brindar una comunicación fiable. Es en este punto donde recae el peso de la recuperación de fallos en la comunicación tales como *overflow* en los *buffers* y el manejo de los paquetes erróneos o corruptos.

- Ordenamiento, esto hace referencia en la forma en la que los mensajes son enviados al grupo. Comúnmente existen cuatro formas de realizar este proceso: sin orden, ordenamiento FIFO (*First In, First Out*), ordenamiento causal y ordenamiento total. El ordenamiento FIFO garantiza que todos los mensajes enviados por un miembro del grupo son recibidos en el mismo orden en que este los envió. El ordenamiento causal garantiza que todos los mensajes que un miembro envíe y los que estén relacionados con éste se entregarán en el mismo orden. Mientras que en el ordenamiento total se garantiza que todos los miembros del grupo reciben todos los mensajes enviados por cualquier otro en el mismo orden para todos los casos. Esta última alternativa es la más conveniente al momento de pensar en replicación, sin embargo y como es de esperarse es la más compleja de implementar.

- Semántica de entrega, hace referencia a en qué momento un mensaje se considera que fue entregado satisfactoriamente al grupo, existen tres opciones para considerar esto: *k*-entregas, quórum y entrega atómica. En el primer caso se considera que el mensaje fue entregado cuando ha sido recibido por un numero *k* de miembros, en el segundo como su nombre lo indica se considera entregado cuando la mayoría de los miembros del grupo han recibido el mensaje y finalmente la entrega atómica se refiere a que únicamente se considera que el mensaje fue entregado cuando todos los miembros del grupo reciban el mensaje.

  Esta última opción es la ideal para ser usada en el caso de la replicación en bases de datos, ya que esto permite cumplir con las propiedades ACID (*Atomicity, Consistency, Isolation and Durability*) de las transacciones, que se verán en la sección 1.5.3.1.

1.5.3. Sistemas de bases de datos distribuidos

Una base de datos distribuida (DDB - *Distributed Database*) es un conjunto de bases de datos lógicamente interrelacionadas las cuales se hayan distribuidas a lo largo de una red de computadores, un sistema de gestión de bases de datos distribuidas (DDBMS - *Distributed Database Management System*) es el software capaz de manejar bases de datos distribuidas, el cual provee los mecanismos de acceso hacia estas y hace que la distribución sea transparente para el usuario. De esta forma se tiene que un sistema de bases de datos distribuidas (DDBS - *Distributed Database System*) es la unión de los dos componentes anteriormente mencionados [17].

Las características de un sistema de bases de datos distribuidas pueden ser visto como una serie de transparencias de distribución, de transacción, de fallas, de heterogeneidad y de desempeño, las cuales tienen un objetivo común: el hacer que la base de datos distribuida se comporte como un sistema centralizado, es decir que el usuario final no note que el sistema está distribuido, sino que lo perciba como una única base de datos la cual contiene toda la información que él esté consultando.

La transparencia de distribución permite gestionar la base de datos que ha sido distribuida físicamente y se encuentra en diferentes nodos de la red como una única base de datos centralizada. La transparencia de transacción, asegura que la ejecución de las transacciones mantenga la integridad de la DDB.

Así mismo, se tiene que la transparencia de fallas garantiza un correcto funcionamiento de la base de datos independientemente de los fallos que puedan presentarse en uno o más nodos de la red, La transparencia de desempeño permite que el rendimiento total del sistema no se vea fuertemente afectado en comparación con un sistema centralizado. Mientras que la transparencia de heterogeneidad permite que el sistema distribuido integre diversos tipos de bases de datos locales en un solo sistema global [18].

1.5.3.1. Arquitectura de un sistema de bases de datos distribuidas

En la Figura 6 se puede apreciar el esquema general que sigue un sistema de bases de datos distribuidas, donde la base de datos global, está conformada por la información contenida en cada una de las bases de datos locales y donde para los clientes de dicha base de datos la distribución de esta es transparente.

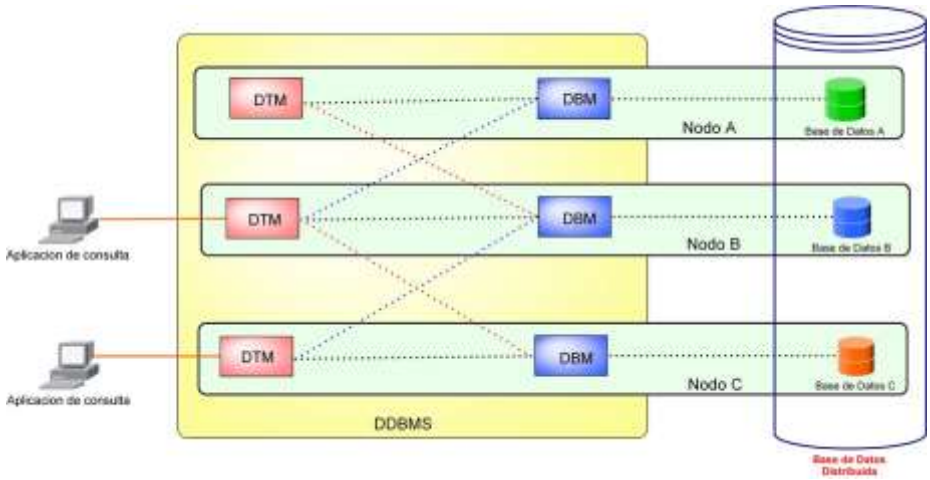

Figura 6. Arquitectura de un sistema de bases de datos distribuidas.

Un sistema gestor de bases de datos distribuidas (DDBMS), está formado por las transacciones y los administradores de base de datos distribuidos de cada uno de los nodos presentes en la red.

Un nodo es un computador capaz de ejecutar un administrador de transacciones distribuidas (DTM), un administrador de la base de datos (DBM) o ambos. Un nodo de transacción únicamente ejecuta un DTM, mientras que los de base de datos ejecutan un DBM y su respectiva base de datos.

El DTM es un software encargado de recibir las solicitudes de procesamiento de las aplicaciones de consulta y a su vez las traduce en acciones para los DBM. Una función importante del DTM es coordinar y controlar dichas acciones.

El DBM es el software que procesa cierta porción de la base de datos distribuida, como es el hecho de recuperar y actualizar datos de acuerdo con comandos de acción recibidos de los DTM.

Los DTM se comunican con los DBM por medio de acciones a ejecutarse en una base de datos específica, así por ejemplo si un nuevo registro debe almacenarse en la base de datos A y en la base de datos B, el DTM traducirá la solicitud de almacenamiento en dos acciones, una dirigida al DBMa y la segunda se dirigida al DBMb para de esta forma almacenar los nuevos datos [19].

Las transacciones pueden ser consideradas como unidades básicas de trabajo que consisten en una secuencia de operaciones que deben ser ejecutas en orden, estas son consideradas el eje fundamental de las labores diarias de muchas industrias como los son las bancarias, las manufactureras y el mercado de acciones [20].

Además de lo mencionado anteriormente, las transacciones deben cumplir con las propiedades ACID, necesarias para que un DBM pueda realizar un correcto manejo de ellas. Dichas propiedades se describen a continuación [20][21]:

- Atomicidad: una transacción es una unidad de trabajo atómica, es decir que no puede ser realizada parcialmente, lo cual implica que en su ejecución es completada en su totalidad o abortada en su totalidad si alguna de las operaciones que la componen falla.

- Consistencia: Una transacción debe llevar al sistema de un estado consistente a otro estado consistente lo cual únicamente ocurre cuando una transacción es considerada legal, esto es que la transacción cumpla con las condiciones de integridad de la información definidas por el usuario.

- Aislamiento: Los efectos de una transacción no deben ser visibles para las demás transacciones hasta que esta haya sido completada en su totalidad.

- Durabilidad: Una vez completada una transacción sus efectos deben ser almacenados de forma permanente.

1.5.3.2. Ventajas y desventajas de los DDBS

Entre las principales ventajas de utilizar una arquitectura de DDBS se puede mencionar las siguientes [19][22][23]:

- Segmentación: Permite reflejar la segmentación y distribución, ya sea física o lógica que puede tener una empresa en unidades organizacionales, por ejemplo departamentos, divisiones, áreas, etc.

- Escalabilidad: Al aumentar el número de nodos presentes en la red, no es necesario hacer una reconfiguración total de la base de datos y este fácilmente entra a formar parte de la DDB.

- Disponibilidad: Por el hecho de la distribución de la información, los DDBS presentan una alta tolerancia a fallos.

- Confiabilidad: Al tener replicas de la información a lo largo de los diferentes nodos de la red, es posible recuperar información que haya sido dañada a partir de una copia anterior.

- Economía: El hecho de compartir los recursos entre las diferentes unidades existentes en una organización, permite un uso más eficiente de los mismos, lo cual conlleva a una reducción de costos.

De igual forma que una arquitectura de este tipo presenta ventajas frente a sistemas de tipo centralizado, también presenta una serie de desventajas e inconvenientes que deben ser solventados para el correcto funcionamiento de sistema. Principalmente se tiene:

- Complejidad: Por el hecho de tener que ocultar la distribución al usuario final, es decir de lograr una transparencia total para el usuario, un sistema de este tipo es mucho más complejo que una centralizado.

- Consistencia: se deben implementar mecanismos de replicación que garanticen la consistencia y validez de los datos en todos los nodos de la red.

- Mayor tiempo de proceso: El hecho de tener que enviar mensajes de diferentes tipos a través de la red, hace que el tiempo de procesamiento en cada nodo tarde más que si se tratara de un sistema centralizado.

- Aumento de tráfico: Cuando se accede frecuentemente a datos almacenados en un nodo remoto, hay un incremento en el tráfico que está cursando por la red, lo cual dependiendo del volumen de información que se esté manejando, así como del tipo de red del que se disponga puede llegar a convertirse en un cuello de botella.

1.5.3.3. Técnicas de replicación en DDBS

Con lo expuesto anteriormente se puede apreciar que la replicación de datos es un campo adoptado tanto por los sistemas distribuidos como por las bases de datos, con la diferencia de los propósitos que principalmente persigue cada uno de estos sistemas. Mientras que los primeros buscan conseguir una mayor tolerancia a fallos, las bases de datos la usan para lograr un mejor desempeño.

Actualmente la convergencia de estas dos ramas ha promovido el surgimiento de nuevos protocolos y técnicas de replicación para bases de datos basadas en primitivas de comunicación de grupo, las cuales a través de sus propiedades tales como el ordenamiento y atomicidad de los mensajes ofrecen enormes ventajas a este nuevo tipo de protocolos logrando de esta forma un incremento en el desempeño [12].

A continuación se listan algunas de las técnicas más relevantes que hacen uso de primitivas de comunicación de grupo para obtener una replicación confiable y eficiente.

- *Active Replication*: esta técnica es una aplicación directa a bases de datos de la técnica de replicación activa de los sistemas distribuidos, la transacción completa a ejecutar es

colocada dentro de un mensaje el cual es posteriormente propagado a todos los servidores haciendo uso de comunicación de grupo, la diferencia se encuentra en que esta propagación no es realizada por el cliente, sino por un servidor delegado quien es el encargado de hacer el reenvío de los mensajes. Como todos los servidores reciben y procesan la transacción de una forma determinística, si uno de ellos aborta la transacción todos los demás deberán abortarla [12][24].

Dado que en esta técnica el punto de determinismo esta al inicio, se hace imposible realizar transacciones de forma interactiva, es decir todas las operaciones deben ser conocidas desde el principio y no hay posibilidad de modificar alguna operación una vez la transacción ha iniciado. Otra restricción presentada por esta técnica es que todas las operaciones tanto de lectura como de escritura son realizadas por todos los servidores, lo cual evita el uso de uno de los principales beneficios de la replicación que es el hacer un balance de carga en las operaciones de lectura [24].

- *Certification-Based Replication:* también conocida como *Database State Machine,* también hace uso de un servidor delegado quien interactúa directamente con el cliente, este ejecuta la transacción localmente pero aun no da ninguna respuesta al cliente, en lugar de esto reenvía el mensaje a las demás replicas haciendo uso de comunicación de grupo, cuando las replicas reciben el mensaje ejecutan una fase determinística de certificación, la cual va a decidir si la transacción es aceptada o debe ser abortada. Al contrario de la anterior técnica esta si puede hacer uso de transacciones interactivas [24].

- *Weak Voting Replication:* cuando el servidor delegado recibe la transacción, este la ejecuta localmente y luego usando comunicación de grupo reenvía a las demás replicas las operaciones de escritura para que sean ejecutadas, pero los cambios aun no son aplicados a la base de datos. Con dicha acción el servidor delegado es capaz de determinar si existen conflictos en la transacción y puede establecer si la transacción se terminará de forma correcta o va a ser abortada. Una vez realizado este proceso, envía su decisión a las demás replicas para que de esta forma terminen la ejecución de la transacción y los cambios surtan efecto [24].

CAPITULO II
FUNDAMENTOS DEL IP MULTIMEDIA SUBSYSTEM – IMS

Las tendencias del mercado, junto con las exigencias de los usuarios y los avances tecnológicos en el campo de las telecomunicaciones, han impulsado el desarrollo y el despliegue de un importante número de servicios, que tienen como características el uso de sistemas multimedia y la ubicuidad en su acceso. Así, el mercado apunta a ofrecer servicios de información y entretenimiento amigables para el usuario y a un costo razonable. Para que esto sea efectivo, los operadores y proveedores necesitan establecer nuevas formas de mercadeo, ofreciendo paquetes de servicios o cobrando por sesiones, etc. [25].

Las demandas de los usuarios se centran en servicios personalizados basados en sus necesidades y útiles para cumplir con sus actividades cotidianas [26]. Además, de que sean fáciles de usar, seguros y que puedan ser accedidos en cualquier lugar, en cualquier momento y de la forma que el usuario desee.

Para poder cumplir con estas nuevas exigencias, sin descuidar la prestación de los tradicionales servicios de conmutación de circuitos, se pretende integrar las redes celulares e Internet en las llamadas redes 3G (*3rd Generation*) [27]. Para lograr este propósito, se ha planteado adoptar un núcleo de red basado totalmente en el protocolo IP y el uso de pasarelas (*gateways*) de medios y de señalización para la integración con redes que no sean basadas en dicho protocolo [25][28].

Con la colaboración de la industria y las universidades, los organismos de estandarización dirigen sus esfuerzos a la consecución de las redes NGN (*Next Generation Networks*). Por una parte se encuentran, el 3GPP (*Third Generation Partnership Project*) y el 3GPP2 (*Third Generation Partnership Project 2*), los cuales son participes del ITU (*International Telecommunication Union*) IMT-2000 (*International Mobile Telecommunications-2000*), que es el estándar global para las redes 3G. El 3GPP y el 3GPP2 han propuesto el IMS (*IP Multimedia Subsystem*) como una arquitectura base para las NGN, haciendo uso de de protocolos y estándares abiertos propuestos por el IETF *(Internet Engineering Task Force*). También la OMA (*Open Mobile Alliance*), juega un papel muy importante en el desarrollo de servicios y habilitadores de servicios IMS [27].

Por otra parte, la regulación en cuanto a redes 3G fijas concierne, tuvo sus inicios en el 2004. El ETSI (*European Telecommunications Standards Institute*), creó el TISPAN (*Telecoms & Internet converged Services & Protocols for Advanced Networks*), con el objetivo de estandarizar una NGN para acceso desde redes fijas basadas en IMS. Además, la ITU-T creó el NGN-FG (*NGN Focus Group*), que trabaja en especificaciones para el acceso a NGNs mediante líneas fijas basadas en IMS. Igualmente en Estados Unidos, la ATIS (*Alliance for Telecommunications Industry Solutions*), creó un grupo de estudio para analizar la aplicabilidad de las NGN e IMS en las redes de acceso fijas norteamericanas [27].

2.1. GENERALIDADES DEL IP *MULTIMEDIA SUBSYSTEM* – IMS

IMS nace con el objetivo de hacer realidad la integración de las redes celulares e Internet, pero de una forma apropiada; es decir, reuniendo las características más sobresalientes del dominio de conmutación de circuitos y del dominio de conmutación de paquetes. Del primero, se extraen características como la alta disponibilidad, la ubicuidad, la facilidad de uso, la calidad de servicio,

entre otras; mientras que del segundo se puede rescatar los altos índices de escalabilidad, la eficiencia en el uso de los recursos, los altos anchos de banda y el uso del protocolo IP como piedra angular para las NGN [29].

En consecuencia, IMS no sólo provee a los usuarios la posibilidad de acceder a servicios de Internet a través de dispositivos móviles, lo cual ya se encuentra actualmente disponible para usuarios 3G, sino que también se preocupa por garantizar tres características adicionales: QoS, diferentes formas de cobro e integración de servicios [27].

La razón para querer garantizar QoS, se encuentra en que el dominio de paquetes de las redes 3G provee servicios multimedia de tiempo real con una característica *best-effort*, que repercute claramente en la experiencia que el usuario puede tener con los servicios en tiempo real, como una conversación de voz o una videoconferencia.

Igualmente, es necesario establecer apropiadamente la forma en que se puede realizar el cobro de las sesiones multimedia, para que el operador y el proveedor puedan seleccionar un modelo de negocio apropiado, que pueda generar competencia en el mercado, retribuyendo ganancias sustanciales a las empresas y generando diversas formas de pago que se acoplen a los diferentes usuarios.

Además, dentro de la razón de ser de IMS también se encuentra la integración de servicios, lo que hace referencia a que diferentes desarrolladores puedan vender sus servicios sin necesidad de montar toda la infraestructura y que los operadores puedan ofrecer una gran variedad de servicios haciendo re-uso de los ya existentes.

Para lograr sus cometidos, como se había dicho antes, IMS adopta el protocolo IP como base y el uso de diferentes interfaces y pasarelas para brindar independencia de la red de acceso que se utilice. Pero también hace uso del concepto de arquitectura horizontal, la cual se entiende como aquella en la que existen varios niveles que albergan un número de nodos con funciones del mismo tipo, lo que permite realizar una actualización o reemplazo independientes en cada nivel de acuerdo con los requerimientos o cambios tecnológicos, esto sin que afecte a los otros niveles [25].

La ventaja de tener una arquitectura horizontal se encuentra en la posibilidad de hacer rehúso de las capacidades y los habilitadores para la creación y prestación de nuevos servicios totalmente basados en IP.

2.2. ARQUITECTURA DEL IMS

IMS distribuye sus entidades funcionales dentro de tres niveles de una arquitectura horizontal, los cuales interactúan entre sí mediante unas interfaces estandarizadas [25]. En la Figura 7 se pueden apreciar dichos niveles, denominados de Conectividad, de Control y de Aplicaciones.

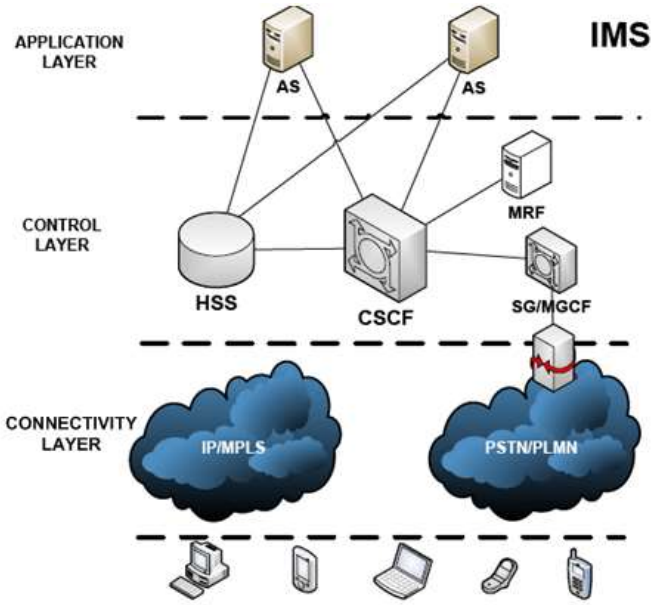

Figura 7. Arquitectura horizontal en IMS.

- Conectividad: El nivel de acceso de red abarca un número importante de diferentes tipos de redes las cuales puede elegir el usuario para conectarse y hacer uso de los servicios prestados por los proveedores.

- Control: sus componentes principales son: el CSCF(*Call Session Control Function),* que se encarga del manejo y asignación de servidores a los clientes que hayan iniciado una sesión y el HSS(*Home Subscriber Server*) que es la base de datos que guarda toda la información relacionada con los usuarios, las sesiones, los proveedores, etc.

- Aplicación: los servidores de aplicación cumplen el papel de ejecutar los servicios que los clientes estén demandando [25].

A continuación se describe más detalladamente las principales entidades funcionales de IMS:

2.2.1. HSS (*Home Subscriber Server*) [30]

El HSS es la base de datos principal para un usuario dado. Es la entidad que contiene información relacionada con la subscripción para dar soporte a las entidades de red que manejan las sesiones o llamadas.

Por cada usuario ésta base de datos contiene tanto, datos dinámicos (por ejemplo la localización del terminal de usuario), como datos estáticos (por ejemplo números y direcciones asociadas al usuario); así mismo, en ésta se guarda información relacionada con la autenticación, el cifrado y el perfil para cada usuario.

El HSS cumple con un subconjunto de tareas requeridas por los dominios de conmutación de circuitos y de conmutación de paquetes, las cuales son heredadas del HLR (*Home Location Register*) y del AuC (*Authentication Centre*) de las redes GSM. Así mismo, el HSS tiene la función de

proveer soporte a las funciones de control del IMS tales como el CSCF, lo que resulta necesario para habilitar al abonado el uso de los servicios IMS.

Las funciones lógicas del HSS se pueden resumir en los siguientes puntos:

- Gestión de la Movilidad
- Soporte para el establecimiento de la Sesión / Llamada
- Generación de la información de seguridad del usuario
- Soporte de la seguridad del usuario
- Manejo de la identificación del usuario
- Autorización de acceso
- Soporte para el servicio de autorización
- Soporte para el servicio de aprovisionamiento
- Almacenamiento de datos del perfil de usuario
- Soporte para servicios CAMEL (*Customised Applications for Mobile networks Enhanced Logic)*

2.2.2. SLF (*Subscription Locator Function*) [27]

Es posible que una red contenga varios HSS para manejar un número elevado de usuarios. En este caso, es necesario tener un SLF para conseguir la correspondencia del HSS asignado a cada usuario.

2.2.3. CSCF (*Call/Session Control Function*)

Es un grupo de servidores SIP o *proxies*, quienes procesan los mensajes de señalización en diferentes situaciones. Estos son el P-CSCF, el I-CSCF y el S-CSCF, los cuales se describen a continuación.

2.2.3.1. P-CSCF (*Proxy* - CSCF) [30]

El *Proxy* - CSCF es el primer punto de contacto entre el terminal de usuario y el núcleo IMS, ya sea que se encuentre en la red local o en una red visitada. Como lo dice su nombre es un servidor SIP que actúa como *proxy* para un terminal, aceptando sus peticiones, tratándolas internamente o reenviándolas.

El P-CSCF es asignado a un terminal durante su registro y éste no cambia mientras dure dicho registro. Además de re-dirigir los mensajes SIP entre el terminal de usuario y ciertos puntos del núcleo IMS, el P-CSCF también tiene como función mantener una asociación de seguridad entre él y cada terminal de usuario, ejecutar la compresión y de-compresión de los mensajes SIP para reducir la latencia y autorizar el uso de los recursos y el manejo de las políticas de calidad de servicio.

2.2.3.2. I-CSCF (*Interrogating* – CSCF) [30]

El I-CSCF es el punto de contacto para todas las conexiones destinadas a un terminal de usuario de una red o a un usuario itinerante (*roaming*) que se encuentre en dicha red, debido a que su dirección es publicada en un DNS (*Domain Name Server*) y puede ser encontrada por servidores remotos.

Durante el proceso de registro el I-CSCF realiza peticiones de localización de usuario al HSS usando el protocolo DIAMETER a través de la interfaz Cx y se encargan de asignarle al terminal de usuario un S-CSCF quien le proveerá el servicio dependiendo de ciertos parámetros como capacidad y localización.

Algunas de las funciones más importantes del I-CSCF son, el correcto enrutamiento de los paquetes SIP hacía el S-CSCF correspondiente y responder a las peticiones del mismo, pero también puede realizar funciones como ocultar la topología y configuración de la red de origen, funciones de tarificación y de *firewall*.

2.2.3.3. S-CSCF (*Serving* – CSCF) [30]

El S-CSCF ejecuta los servicios de control de sesión para el terminal de usuario dentro del núcleo IMS y mantiene el estado de dicha sesión para poder prestar un correcto soporte de los servicios.

Para realizar la provisión de los servicios el S-CSCF se basa en el perfil de usuario que obtiene del HSS usando el protocolo DIAMETER y decide para cual servidor de aplicaciones deben ir dirigidos los mensajes SIP.

Además, el S-CSCF realiza ciertas tareas en el proceso de registro y autenticación del usuario IMS y vela por el cumplimiento de las políticas del operador de red para evitar el uso no autorizado de los servicios.

2.2.4. AS (*Application Server*) [27]

El AS o Servidor de Aplicaciones es una entidad SIP que se encarga de mantener y ejecutar servicios.

Existen tres tipos de AS que se describen a continuación:

- SIP AS: es el servidor nativo que ejecuta Servicios Multimedia IP basados en SIP.
- OSA-SCS (*Open Service Access-Service Capability Server*): es un servidor de aplicaciones que provee una interfaz para el Servidor de Aplicaciones del *framework* OSA.
- IM-SSF (IP *Multimedia Service Switching Function*): este servidor habilita al gsmSCF (GSM *Service Control Function*), para que controle una sesión IMS y de esta forma reutilizar los servicios CAMEL que fueron desarrollados para GSM.

Los tres tipos de AS se comunican con el S-CSCF utilizando el protocolo SIP. Pero para la comunicación con el HSS, el SIP-AS y el OSA-SCS hacen uso de una interfaz basada en el

protocolo DIAMETER, mientras que el IM-SSF usa una interfaz basada en MAP (*Mobile Application Part*).

2.2.5. MRF (*Media Resource Function*) [27]

El MRF provee una fuente de medios para la red *home*. Así, le permite desplegar anuncios, mezclar flujos, tras-codificar entre diferentes *codecs* y obtener estadísticas. Se divide en MRFC (MRF *Controller*) y MRFP (MRF *Processor*), para el control y el procesamiento de los medios respectivamente.

2.2.6. BGCF (*Breakout Gateway Control Function*) [27]

Es un servidor SIP que a través de su funcionalidad de enrutamiento determina el próximo salto de un mensaje SIP en una sesión iniciada por un terminal IMS y que va dirigido a un terminal en el dominio de conmutación de circuitos.

2.2.7. SGW (*Signaling Gateway*) [27]

Provee una interfaz para el plano de señalización de una red de conmutación de circuitos. También, ejecuta la conversión del protocolo de bajo nivel.

2.2.8. MGCF (*Media Gateway Control Function*) [27]

Implementa las funciones que permiten la conversión y el mapeo de SIP en un protocolo de control de llamada adecuado para redes del dominio de circuitos. También controla los recursos de la MGW (*Media Gateway*).

2.2.9. MGW (*Media Gateway*) [27]

Es la interfaz para el plano de medios de una red del dominio de conmutación de circuitos. Recibe información en RTP (*Real-time Transport Protocol*) y la ubica en *time slots* PCM (*Pulse Code Modulation*) para conectarse a la red de conmutación de circuitos. Si es necesario, también realiza tras codificación cuando el terminal IMS no soporta el *codec* usado en el lado de conmutación de circuitos.

2.3. IDENTIFICACIÓN DE LA ARQUITECTURA DE SERVICIOS IMS NATIVOS

Para la prestación de servicios en IMS se puede hacer uso de los tres tipos de Servidores de Aplicación que existen; SIP AS, OSA-SCS, IM-SSF. Debido a que la mayoría de servicios nuevos que se van a ofrecer en IMS se basarán en SIP, el proyecto expuesto en este documento, se enfoca en la arquitectura de servicios para los Servicios IMS Nativos, prestados por los SIP AS.

Dichos Servidores de Aplicaciones SIP se pueden encontrar tanto en la red *home* como también en redes de terceros como se puede apreciar en la Figura 8. En la misma figura, también se pueden observar las interacciones directas que presentan los SIP AS al momento de ejecutar servicios. Las funciones con las que éstos interactúan son el S-CSCF de la red *home* y el HSS.

El S-CSCF se encarga de seleccionar el SIP AS adecuado para la prestación del servicio que el usuario esté demandando. Para cumplir con esta tarea el S-CSCF hace uso de la información contenida en el Perfil de Usuario que se encuentra guardado en el HSS y que es descargado al S-CSCF en el momento del registro de dicho usuario.

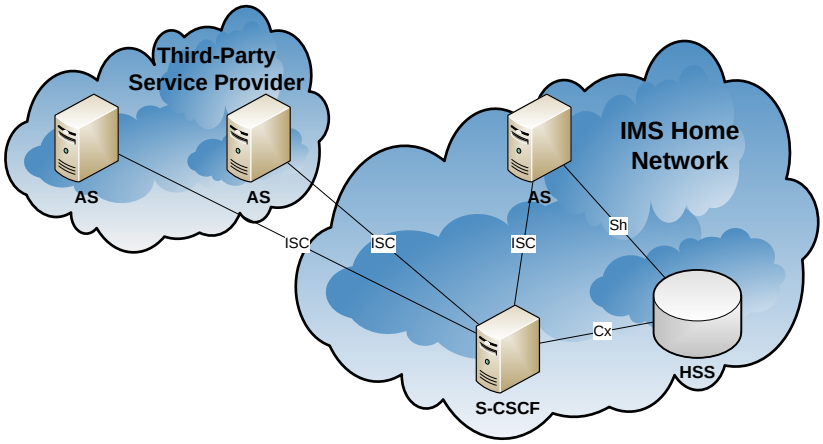

Figura 8. Arquitectura para la prestación de Servicios IMS Nativos.

El camino de señalización con el cliente se completa con la comunicación del S-CSCF con el P-CSCF asignado y de este con el equipo de usuario IMS.

Para realizar la comunicación entre las funciones que aparecen en la Figura 8, el 3GPP ha definido interfaces y protocolos que se describen a continuación junto con los procesos que se ejecutan a través de ellas.

2.3.1. Interfaz ISC

La interfaz ISC (*IMS Service Control*), es usada para la comunicación entre el S-CSCF y los SIP-AS mediante el uso del protocolo SIP.

Esta interfaz transporta información relacionada con el registro de los usuarios y las capacidades y características del Equipo de Usuario; información que el S-CSCF determinará si debe ser conocida por el AS para prestar un servicio de forma adecuada.

2.3.2. Interfaz Cx [31]

La interfaz Cx se encuentra especificada entre el S-CSCF y el HSS. Aunque también se ha estandarizado como conexión entre el HSS y el I-CSCF. Por otra parte, cuando existen varios HSS en una misma red se involucra al SLF el cual se comunica con el I-CSCF y con el S-CSCF a través de la interfaz Dx, la cual cumple con la misma funcionalidad que la Cx [27].

Las funciones que tiene la interfaz Cx son:

- Descargar desde el HSS al S-CSCF los vectores de autenticación de usuario.
- Grabar en el HSS la dirección del S-CSCF asignado al usuario.
- Descargar desde el HSS el Perfil del Usuario.
- Sobrescribir el Perfil de Usuario en el S-CSCF cuando éste haya cambiado en el HSS.
- Informar al HSS sobre el estado de registro de una identidad de usuario.

El protocolo que se maneja en esta interfaz es DIAMETER, y existe una aplicación propietaria que ha aceptado el 3GPP, la cual tiene definidos ciertos comandos y AVPs (*Attribute Value Pairs*) para ejecutar sus funciones.

2.3.3. Interfaz Sh [31]

La interfaz Sh se usa para transferir información entre un SIP AS o un OSA-SCS y un HSS y viceversa. Se caracteriza porque es una interfaz intra-operador; es decir, que los dos puntos que conecta se encuentran en la misma red.

Esta interfaz provee la funcionalidad de guardar y recuperar información desde el HSS, el cual se hace responsable de la información suministrada a cada AS. Además de descargar información desde el HSS, el AS puede suscribirse a un servicio de notificación, mediante el cual es alertado sobre cambios realizados en la información contenida en el HSS.

Con la interfaz Sh también se puede transferir información relacionada con el usuario o con el servicio y que sea completamente transparente (no entendible) para la interfaz y el protocolo.

Otro tipo de información que se puede transportar es información estandarizada, o aquella que puede ser compartida y comprendida por varios AS. Así mismo, un AS puede enviar información relacionada con la activación o desactivación de su propio *Initial Filter Criteria* o requerir información de localización del usuario.

El protocolo que se usa en esta interfaz es DIAMETER, con la adición de una Aplicación DIAMETER propietaria, que de la misma forma que para la interfaz Cx, ha definido unos comandos y AVPs dispuestos para soportar la funcionalidad de la interfaz.

2.4. DISTRIBUCIÓN DE LA INFORMACIÓN RELACIONADA CON EL USUARIO

Para la correcta prestación de servicios IMS nativos, es necesario el intercambio de cierta información de usuario entre las funciones de la arquitectura de servicios que se aprecia en la Figura 8 (HSS, S-CSCF y SIP AS). Este intercambio de información se realiza en el momento del registro del usuario y algunos datos pueden variar una vez el usuario se encuentre registrado en la red.

A continuación se listan los parámetros que se intercambian en la prestación de servicios según la interfaz que se usa para tal fin.

2.4.1. Datos Transferidos por la interfaz ISC [32]

Esta información se transmite para notificar al AS desde el S-CSCF sobre un registro implícito de *Public User Identities*, el estado del registro y las capacidades y características del Equipo de Usuario.

2.4.2. Datos Transferidos por la interfaz Cx [27][32]

Estos datos se transfieren entre el HSS y el S-CSCF.

- Identidades públicas y privadas de usuario
- Identidades públicas y privadas de servicio

- Indicación de prohibición de una Identidad Pública de Usuario para cualquier comunicación
- Lista de redes visitadas autorizadas
- Información relacionada con el estado de registro
- Información de autenticación y cifrado
- Nombre en *Display*
- Nombre y dirección de AS
- Datos relacionados con los servicios de la red *core*
- Nombre y dirección de S-CSCF
- Información de tarificación
- Perfil de Usuario

Entre la información anteriormente citada se encuentran unos datos que pueden ser modificados, tales como, el nombre del S-CSCF, el estado del registro, las direcciones y nombres del AS y S-CSCF, información de autenticación y cifrado, la dirección del HSS y cierta información propia del perfil de usuario.

2.4.3. Datos Transferidos por la interfaz Sh [27][31]

La interfaz Sh transfiere información entre el HSS y el AS, como la que se lista a continuación:

- Datos transparentes a la interfaz, la cual es información específica sobre el servicio que el AS se encuentra prestando y que se guarda en *Repository Data* del HSS
- Identidades públicas
- Estado del usuario
- Nombre del S-CSCF
- Información de localización
- Información de tarificación
- Perfil de usuario

Parte de esta información puede ser modificada, como los datos transparentes, el estado de usuario, el nombre del S-CSCF y la información contenida en el *Repository Data* [32]. Estos datos deben ser actualizados automáticamente en el HSS o en el AS, dependiendo de donde se haga la modificación.

2.5. MECANISMO DE SINCRONIZACIÓN DE INFORMACIÓN DE USUARIO A TRAVÉS DE LA INTERFAZ Sh

Como se describió en 2.3 y 2.4, la interfaz Sh se encuentra definida para comunicar un SIP *Application Server* (SIP AS) o un OSA *Service Capabilities Server* (OSA-SCS) con el HSS, con el objetivo de:

- Descargar y actualizar datos de usuario
- Pedir y enviar notificaciones en cambios en datos de usuario

Así, la sincronización de información de usuario entre el HSS y los AS se resume en los anteriores puntos. Para cumplir con este propósito se usa el protocolo DIAMETER en su forma base con una extensión que maneja los mensajes definidos para esta interfaz.

DIAMETER es un protocolo usado para el manejo de procesos de autorización, autenticación y contabilidad (AAA, por sus siglas en inglés) en IMS. Surge como una evolución de RADIUS y opera en la parte superior de la pila de protocolos de trasporte seguros como lo son TCP y SCTP. El RFC 3588 del IETF define el protocolo base el cual provee la capacidad de negociaciones entre los entes involucrados en la comunicación, la notificación de errores y la entrega de *Attribute-Value-Pair* (AVP) como las unidades de información dentro de un mensaje DIAMETER. También es posible realizar extensiones al protocolo, como la aplicación Sh, adhiriendo nuevos comandos y AVPs [33].

2.5.1. La Aplicación Sh

Para la implementación de la aplicación Sh, el 3GPP específica una serie de *Command Codes* que extienden el protocolo base DIAMETER, los cuales son los siguientes [34]:

- UDR *User-Data-Request*: se usa por el AS para requerir información de usuario al HSS.
- UDA *User-Data-Answer*: la respuesta al UDR con el *result code* y la información requerida si es el caso.
- PUR *Profile-Update-Request*: lo usa el AS para actualizar información relacionada con un servicio, que haya sido guardada en el HSS.
- PUA *Profile-Update-Answer*: respuesta al PUR por parte del HSS.
- SNR *Subscribe-Notifications-Request*: es el comando usado por un AS para subscribirse a las notificaciones de actualización automática de información.
- SNA *Subscribe-Notification-Answer*: respuesta por parte del HSS al SNR.
- PNR *Push-Notification-Request*: el HSS usa este comando para enviar la información que ha cambiado al AS que se haya suscrito.
- PNA *Push-Notifications-Answer*: es la confirmación de recepción por parte del AS al PNR.

2.5.2. Tabla de permisos para los AS

El HSS mantiene una lista de permisos para cada AS y la verifica cada vez que este último requiere información relacionada con el usuario. Entonces, para cada AS se establecen los derechos que tiene a leer, actualizar o suscribirse a notificaciones [35].

2.5.3. Procedimientos del mecanismo de sincronización de información usando la interfaz Sh

Los *Command Codes* definidos para la aplicación de la interfaz Sh se usan en los siguientes procedimientos que resumen el mecanismo de sincronización de información de usuario.

2.5.3.1. Lectura de Datos (Sh-Pull)

Este procedimiento es invocado por el AS para leer información de un usuario específico desde el HSS y es mapeado a los comandos UDR/UDA de la aplicación Sh.

El AS crea un UDR con el tipo de información que requiere y la identidad pública del usuario al que pertenece dicha información. El HSS recibe este comando y verifica los derechos de lectura en la tabla de permisos para el AS en cuestión. Si el AS se encuentra habilitado, el HSS envía un UDA con la información requerida o un *result code* con una excepción si es el caso.

2.5.3.2. Actualización de Datos (Sh -Update)

El procedimiento Sh-Update se usa entre el AS y el HSS para permitir al AS actualizar información transparente guardada en el *Repository Data* del HSS. Este procedimiento es mapeado a los comandos PUR/PUA de la aplicación Sh.

El AS envía un mensaje PUR al HSS con el tipo de los datos que desea actualizar, el valor de los mismos, el servicio al que pertenecen y la identidad pública del usuario al cual pertenecen. Aquí el HSS también verifica los permisos de escritura en la tabla de permisos y responde con un mensaje PUA, el que contiene el resultado de la operación.

2.5.3.3. Subscripción a Notificaciones (Sh-Subs-Notif)

Este procedimiento se invoca por el AS y se usa para subscribirse a notificaciones cuando la información de un usuario en particular haya sido actualizada desde el HSS. Este procedimiento es mapeado a los comandos SNR/SNA de la aplicación Sh.

El AS envía la petición de subscripción a notificaciones al HSS con un SNR. Una vez este compruebe sus permisos, le regresa el resultado en un SNA.

2.5.3.4. Notificaciones (Sh-Notif)

Este procedimiento es usado entre el AS y el HSS y lo invoca este último para informar al AS sobre cambios en la información a la cual éste se ha suscrito previamente para recibir notificaciones. Es mapeado a los comandos PNR/PNA de la aplicación Sh.

El HSS envía la información actualizada en el PNR y el AS regresa el resultado de la operación en un PNA.

CAPITULO III
MECANISMO DE SINCRONIZACIÓN DE BASES DE DATOS PARA LAS NGN

3.1. PLANTEAMIENTO DEL PROBLEMA

Uno de los objetivos de las NGN es proveer a los usuarios de servicios multimedia personalizados y en tiempo real, según las preferencias y condiciones que se presenten en un momento dado [36]. Para la prestación de estos servicios personalizados es de vital importancia conocer y gestionar adecuadamente el Perfil de Usuario, que es una colección de información relacionada con el usuario, la cual se genera a partir de una serie de parámetros activados por diferentes condiciones como su ubicación, las capacidades del dispositivo, las preferencias de QoS, la presencia, la adición o la sustracción de ciertos servicios en tiempo real, entre otros [36][37].

Además, para el proyecto descrito en este documento, el Perfil de Usuario se considera como el conjunto de datos relacionados con el usuario que ha tenido origen en diferentes redes, dominios y dispositivos. Esta información puede ser pública o privada para los diferentes operadores de red o proveedores de servicio y está directamente relacionada con las preferencias y servicios a los cuales el usuario se encuentra suscrito.

En ambientes previos al estándar IMS, es decir donde no se aplica el concepto de convergencia de redes, los diferentes proveedores de servicios y operadores, mantienen los datos de sus usuarios en bases de datos independientes en cada red y en cada dominio, por lo cual no se comparte información de este tipo entre ellas. Por ejemplo, la información de un usuario de la red inteligente no es la misma de la red GSM/GPRS o de una red UMTS, si no que en cada una se tienen datos distintos para el mismo usuario. Además, en estas redes los proveedores de servicio no tienen información en las bases de datos del operador y viceversa.

Lo anterior está en oposición con uno de los propósitos de las NGN y específicamente con los requerimientos de la arquitectura IMS, de mantener los datos distribuidos y redundantes en la red [38], mientras se comparta información que sea útil para la prestación de los servicios personalizados. En otras palabras, para poder prestar los diferentes servicios en este tipo de redes, es necesario tener una colección de datos relacionados con el usuario debidamente actualizada, y que pueda ser accedida muy rápidamente, en cualquier momento, desde cualquier lugar y sin importar la carga que pueda tener la red [36][38][39].

Así, el despliegue de servicios en las NGN implica un elevado número de fuentes heterogéneas y consumidores de datos ubicados en diferentes redes, dominios y dispositivos, que exige un manejo eficiente de la información. Entre estos datos puede encontrarse información específicamente relacionada con el usuario pero que ha tenido origen en diferentes entidades, la cual debe ser distribuida entre las mismas para facilitar la gestión de las preferencias y la personalización de los servicios entre otros aspectos, con el objetivo de brindar al usuario final, servicios de calidad y de forma transparente [38].

En este ambiente, donde se requiere que la información de usuario se transporte desde y hacia varios nodos de red en forma oportuna, también es necesario garantizar la consistencia de los datos mediante el uso de diversos mecanismos que permitan obtener una réplica coherente de la

información donde esta sea demandada y que además posibiliten el uso eficiente de los recursos de la red [36][39].

Pasando a un escenario más puntual, en el Capítulo II se describió como en IMS se ha definido un mecanismo para el trasporte y replicación de datos de usuario dentro de la Arquitectura de Servicios Nativos, usando las interfaces Sh y Cx con la implementación de aplicaciones DIAMETER que ejecutan los procesos de actualización y notificación entre los nodos de dicha arquitectura, es decir entre el HSS, el S-CSCF y los AS [34][35]. Sin embargo, los organismos encargados de la estandarización, no se han preocupado por definir un mecanismo de replicación que ayude a mejorar el desempeño de los nodos comprometidos mientras se garantiza la consistencia de la información.

Además, la habilitación de las interfaces Sh y Cx, junto con la implementación de las aplicaciones DIAMETER en los nodos, resultan ser dos procesos que demandan bastante tiempo y dinero ya que es necesario modificar el núcleo de estos nodos, adicionando código que ejecute los procesos definidos para la sincronización de la información.

Así, se observa la necesidad de definir un mecanismo de replicación de información que sea fácil de implementar y sea capaz de inter-operar con las distintas bases de datos que se encuentran en una red IMS y en general en las NGN.

3.2. REQUERIMIETOS DE LAS NGN PARA LA SINCRONIZACIÓN DE BASES DE DATOS

Para poder proponer un mecanismo que solucione el problema planteado anteriormente, se realizó un estudio de las exigencias de desempeño de las NGN en cuanto al tratamiento de información de señalización se refiere. Después de un análisis se encontró los siguientes requerimientos como los más sobresalientes.

- Sistema abierto de protocolos internacionalmente aceptado y basados en IP [37]: la solución debe limitarse al uso de protocolos que se encuentren estandarizados y que se sean de libre uso con el objetivo de lograr una interoperabilidad con todas las bases de datos de las NGN. Dichos protocolos deben estar de acuerdo con la piedra angular de la convergencia de redes, el protocolo IP.

- Red funcionalmente distribuida [37][38]: que las funciones de la red se distribuyan en varios nodos lógicos y físicos con el objetivo de mejorar el rendimiento de sus componentes.

- Uso eficiente de recursos [37]: tanto a nivel del procesamiento de los componentes de red, como en el uso del ancho de banda disponible.

- Alta escalabilidad [38]: se requieren soluciones capaces de soportar a largo plazo, el constante incremento del número de usuarios de los sistemas de telecomunicaciones a nivel mundial.

- Transporte oportuno de la información [38][26][38]: para poder prestar servicios multimedia y en tiempo real con altos índices de calidad y de forma transparente para el usuario.

- Consistencia de la información [38]: una coherencia semántica de la información replicada debe garantizarse como punto de partida para la adecuada prestación de servicios.

- Red de telecomunicaciones geográficamente dispersa [26]: la interconexión y operación de los sistemas de telecomunicaciones a nivel mundial, proporciona la oportunidad de integrar en la prestación de un servicio a actores que encuentran en diferentes partes del mundo. Por lo tanto, la solución debe ajustarse a esta situación sin presentar un mayor traumatismo en su desempeño.

Así, con los estudios descritos en los Capítulos I y II y los requerimientos claramente establecidos, fue posible delimitar el rango de posibilidades existentes para optar por una solución al problema planteado.

3.3. MECANISMO DE SINCRONIZACIÓN DE BASES DE DATOS EN EL CONTEXTO DE IMS

Reiterando lo expuesto en el Capítulo II, el estándar IMS se perfila como una solución efectiva y aceptada mundialmente para la consecución de la convergencia de redes y por ende de las NGN. Por este motivo, se enfatizó el análisis en la proposición de un mecanismo que se integre fácilmente en un entorno IMS.

Con el problema puntualizado a IMS y comprendidos los requerimientos de las NGN para la sincronización de información, se procedió a analizar de forma detallada los estudios consignados en el Capítulo I. Es decir, se examinó tanto el Estado del Arte de las técnicas de replicación en las redes de datos, como también la evolución de los mecanismos de replicación en los sistemas de telecomunicaciones fijos y móviles hasta llegar a IMS.

De esta forma, se lograron las siguientes deducciones:

- Por las características de su arquitectura, se puede determinar que IMS presenta una similitud con los DDBS, ya que las bases de datos de cada red y dominio existentes comparten información entre ellas. Por lo tanto, se considera viable la aplicación de técnicas de replicación de los DDBS a IMS.

- Dichas bases de datos por pertenecer físicamente a diferentes operadores de red y proveedores de servicio son heterogéneas en ciertas características, como el DDBMS que usan (Oracle, PostgreSQL, MySQL), la funcionalidad de sus servidores (HLR, HSS, VLR, etc.) y los protocolos e interfaces que tienen implementadas los mismos.

- En IMS existen varias bases de datos que se pueden considerar heredadas, esto debido a la interoperabilidad con otros tipos de redes (GPRS/GSM, UMTS, Red Inteligente, PSTN).

- El *core* IP en IMS es de gran importancia para una eventual interconexión de estas bases de datos.

- El HSS, como repositorio central de la información en IMS, tiene varias tareas asignadas, por lo que es bastante útil delegar la funcionalidad de replicación de información a otro módulo cuando esta labor requiera un alto consumo de recursos.

Así, se determinó que la forma de mejorar el desempeño en los procesos de replicación de bases de datos en IMS garantizando la consistencia en su información, era introducir las técnicas de replicación usadas en el contexto de las redes de datos, en especial en los sistemas de bases de datos distribuidos (DDBS). Esto, debido principalmente a la posibilidad de ver las bases de datos de IMS como un DDBS y a la opción de usar IP como protocolo base para las conexiones necesarias.

Posteriormente, se fueron examinando los requerimientos encontrando que las primitivas de *group communication* usadas en las técnicas de replicación de los DDBS, junto con las arquitecturas *middleware* ayudan a cumplirlos de la siguiente forma:

- Las técnicas de replicación que usan *group communication* ayudan a lograr un transporte oportuno de la información, ya que reducen el *overhead* en los procesos de replicación [11].

- *Middleware* ofrece independencia del proveedor del DBMS, lo cual facilita su interoperabilidad en entornos de bases de datos heterogéneas y heredadas. Además, facilita la portabilidad de aplicaciones, ayudando a que la solución sea poco intrusiva y fácil de implementar [14].

- El uso de *group communication* en un mecanismo de replicación garantiza la consistencia de la información gracias a las primitivas que usa [12].

- Una arquitectura *middleware* con *group communication* permite un uso eficiente de los recursos de red ya que reduce considerablemente el *overhead* [14].

- El manejo de un número elevado de usuarios no es un problema con las arquitecturas *middleware* ya que la escalabilidad es una propiedad inherente de ellas [40].

- Una arquitectura middleware está pensada para trabajar en entornos geográficamente distribuidos [14][41].

En la Figura 9 se puede apreciar el diseño inicial para la incorporación de la plataforma *middleware* con *group communication* en IMS. Una plataforma especializada se encarga del proceso de replicación usando las técnicas de los DDBS y las bondades ofrecidas por las arquitecturas middleware, interconectando las bases de datos que sean necesarias para compartir en tiempo real la información que necesiten.

Entonces, dependiendo de las exigencias, puede replicarse información entre bases de datos de la misma red IMS, con otras redes IMS, con redes de proveedores de servicios o con redes heredadas.

Dentro del estudio realizado sobre las de técnicas de replicación que usan *group communication*, se observó que la mayor parte de las investigaciones se han centrado en el campo de las bases de datos relacionales. Entonces, con el ánimo de basarse en la teoría ya estudiada, la solución de sincronización de datos de usuario presentada en este proyecto se plantea para ser usada entre repositorios que hagan uso de bases de datos relacionales.

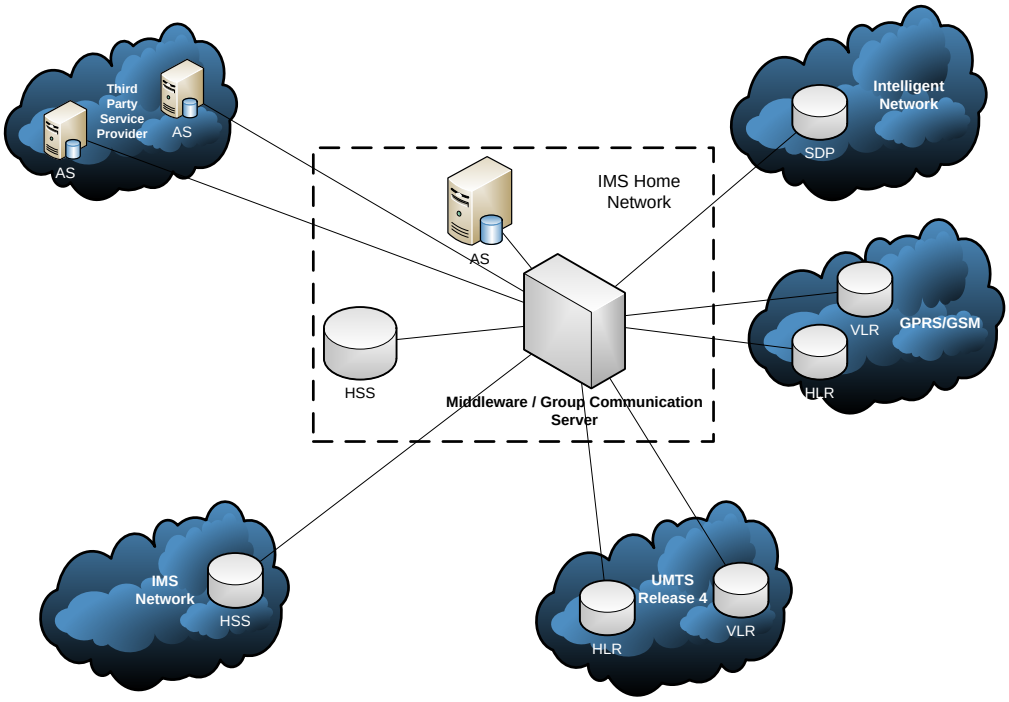

Figura 9. Incorporación de la plataforma middleware con *group communication* en IMS.

3.4. PROTOTIPO PARA LA VALIDACIÓN DEL MECANISMO PROPUESTO

Después de llegar a una idea clara y concisa de la solución, se procedió a realizar el diseño de un prototipo para la validación del mecanismo de sincronización de información propuesto.

Además se hizo el estudio de una serie de posibles herramientas a ser usadas en dicho prototipo, el análisis correspondiente y la selección de las más indicadas para la implementación.

3.4.1. Diseño

De acuerdo a lo analizado en el Capítulo II, entre los módulos de la Arquitectura de Servicios Nativos de IMS se comparte cierta información que puede ser estática o puede variar durante una sesión establecida. Por lo tanto, para realizar la implementación del prototipo dentro de esta arquitectura se debe seleccionar un tipo de información dinámica y que pueda ser modificada fácilmente.

Además, dado que el mecanismo se ha planteado para la sincronización de información entre bases de datos, debe seleccionarse dos módulos que tengan bases de datos en su estructura.

Entonces, se determinó probar el mecanismo propuesto, sincronizando los nodos que comparten la información del *Repository Data* para un servicio dado dentro del IMS. Estos nodos son el HSS y los AS que prestan el servicio tal como se especifica en [32].

El HSS y los AS tiene bases de datos que pueden ser conectadas a la plataforma *middleware* y por otra parte, la información del *Repository Data* es de gran importancia para la prestación de servicios personalizados, ya que los proveedores de servicio pueden ingresar en estos campos cualquier tipo de dato relacionado con los servicios que un usuario en particular consume [27].

De esta forma, se plantea la inclusión de una plataforma *middleware* entre los nodos mencionados. La arquitectura del prototipo para la validación se muestra en la Figura 10.

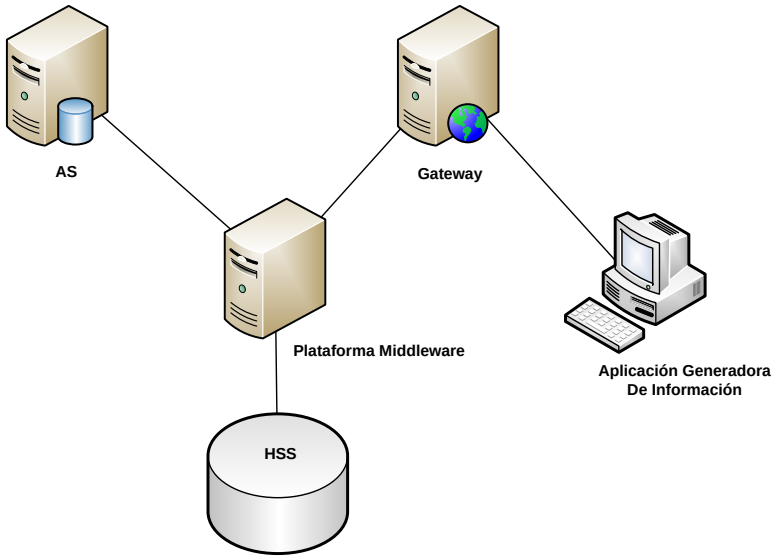

Figura 10. Arquitectura general del prototipo.

Sin el ánimo de plantear una modificación específica al estándar IMS, si no con la finalidad de realizar las pruebas del mecanismo propuesto, se inhabilita la interfaz Sh y por ende su propio mecanismo de sincronización.

Además de los elementos mencionados anteriormente, se decide realizar la inclusión de un AS que actúa como *gateway* entre la aplicación que genera la información a ser sincronizada y la plataforma de replicación. Las funciones que realiza cada uno de los elementos se describe en 3.4.2.

3.4.2. Descripción del funcionamiento

A continuación se presenta una descripción del papel que juega cada uno de los módulos presentes en la arquitectura de la Figura 10 y la forma en la cual cada uno de ellos se relaciona con los demás.

- AS: El servidor de aplicaciones es uno de los principales actores en este prototipo, ya que es a él a quien interesa la información que será sincronizada. En este modelo se asume que en la misma máquina donde está corriendo el AS se encuentra el motor de bases de datos al cual se pretende replicar la información del HSS.

  El AS se conecta a la plataforma *middleware* mediante las librerías o *drivers* que esta provea, de esta forma se aprovechan las ventajas de este tipo de sistemas y no se deben realizar cambios sustanciales en las aplicaciones y servicios que se encuentren desplegados en este.

- HSS: Este es el contenedor principal de toda la información en la red IMS y por lo tanto es uno de los puntos críticos de esta propuesta, ya que la información que va a ser replicada

debe ser escogida cuidadosamente para así evitar enviar información privada de los usuarios o del operador hacia los AS que no lo necesiten.

Al igual que en el caso del AS, el HSS se conecta a la plataforma *middleware* mediante los controladores que esta provea.

- Plataforma *Middleware*: Es el corazón de la propuesta y es quien va a proporcionar la implementación de los protocolos de replicación para las bases de datos de una forma transparente, permitiendo así un desarrollo más rápido de servicios y su posterior despliegue.

- Aplicación Generadora de Información (AGI): Esta es una aplicación desarrollada en lenguaje Java y como su nombre lo indica es la encargada de generar la información que se pretende sea sincronizada entre el AS y el HSS. Esta aplicación no se conecta directamente con la plataforma *middleware* sino que lo hace mediante una *Gateway*, por lo tanto no hace uso de los *drivers* del *middleware* sino que envía las peticiones a través de mensajes HTTP (*HyperText Transport Protocol*) hacia una aplicación *web* que se encuentra en la *Gateway*.

- *Gateway*: Como se explica en el Capítulo IV, para efecto de pruebas se realizó una comparación de desempeño entre la propuesta realizada en este trabajo y el mecanismo para sincronización de información propio de IMS mediante la interfaz Sh, por tal motivo se optó por adicionar una *Gateway* que permitiera hacer uso de librerías e implementaciones existentes de esta interfaz y de esta forma facilitar el desarrollo del proyecto e invertir tiempo innecesario.

  A pesar que este módulo se lo incluye principalmente para realizar las pruebas en el otro mecanismo de sincronización se decidió tenerlo en cuenta en la arquitectura del prototipo para la validación del mecanismo propuesto, aunque no es necesario, ya que la AGI puede ser conectada directamente a la plataforma middleware. Entonces, la adición de este módulo se hace con el fin garantizar que todas las pruebas que se realicen estén bajo las mismas condiciones y de esta forma no haya duda en los resultados obtenidos

3.4.3. Exploración y análisis de herramientas para la implementación

Una vez realizado el diseño del prototipo para la validación de la propuesta del mecanismo de sincronización presentada en la sección 3.3, se realizó una exploración y un análisis de las herramientas existentes para elegir las que mejor se adecúen a las necesidades del proyecto.

Para cada uno de los nodos de la Figura 10 se hizo la correspondiente exploración, buscando en primer lugar una herramienta que permitiera la simulación, emulación o implementación no sólo del HSS si no de todos los elementos del *core* IMS (CSCF y HSS) y a la cual se le pudiera realizar los cambios o adaptaciones pertinentes para la inclusión de la plataforma *middleware* y para la implementación del mecanismo de sincronización de IMS usando la interfaz Sh con el que se hicieron comparaciones de desempeño, tal como se explica en el Capítulo IV.

Seguidamente, se investigó sobre las herramientas que permitieran la implementación de la plataforma *middleware* con las técnicas de replicación de los DDBS, en especial las que usaran *group communication*.

Así mismo, se exploraron varios AS SIP que permitieran la interconexión con el HSS través de la plataforma *middleware* y de la interfaz Sh.

Por último se examinaron las herramientas de análisis de protocolos y generación de estadísticas para el correspondiente análisis de desempeño expuesto en el Capítulo V.

A continuación se describen las herramientas exploradas.

3.4.3.1. Herramientas para el core IMS

- IMS Network Emulator

Como su nombre lo indica, es un emulador de un *core* IMS desarrollado por Nokia-Siemens dentro del proyecto IMS *Developer Program*, el cual hace posible la prestación de servicios. Además permite que se prueben aplicaciones para este tipo de redes sin necesidad de contar con una infraestructura real, lo cual se convierte en una gran ventaja para los desarrolladores. Además del *core*, la solución de Nokia-Siemens provee una serie de herramientas tales como SDKs (*Software Development Kits*), APIs (*Application Programming Interfaces*) y servidores de aplicaciones

Entre las facilidades mostradas por esta alternativa luego de hacer una serie de cortas pruebas se pueden mencionar su facilidad de instalación, y una fácil configuración, mientras que su principal desventaja es el hecho de ser un software privativo, lo cual impide cualquier intento de hacer cambios en la forma en cómo este funciona o interactúa con los componentes de la red, además de esto otra de las limitantes presentadas es el hecho que es dependiente de la plataforma, ya que únicamente puede ser instalado en plataformas Windows con el *framework* de .NET.

Otro punto en contra que posee esta plataforma es el anuncio realizado en el mes de junio de 2008 del cierre del programa donde se venía desarrollando esta herramienta, lo cual implica un corte en el soporte que se puede tener al utilizar las versiones que anteriormente habían sido liberadas.

- Open IMS Core

Open IMS Core, una implementación de un *core* IMS (CSCF y HSS) realizada en el instituto Fraunhofer de Alemania, la principal característica de esta iniciativa, como su nombre lo indica, es que está basada en software libre tal como OpenSER y MySQL y es usable bajo licencia GPLv2 (*General Public License*), la cual permite su libre modificación y distribución.

Esta plataforma ha alcanzado suficiente madurez y es usada como cama de pruebas dentro del Open IMS Payground, un proyecto liderado por el grupo Fokus del instituto Fraunhofer, que ofrece servicios como implementación de soluciones IMS, consultoría, *benchmarking* entre otros.

A pesar de que los creadores de esta plataforma ofrecen un soporte pago, esto no implica que no exista soporte y ayuda para las versiones que no han adquirido este paquete, ya que existe

una gran comunidad de usuarios suscritos a diferentes listas de correo, además de un extenso historial con las preguntas, dudas y aportes hechos en estas listas desde noviembre de 2006.

La implementación del Open IMS Core puede ser vista como la unión de dos desarrollos, por una parte se tiene la adaptación del servidor de aplicaciones SIP OpenSER para cumplir con el papel de los servidores de aplicaciones que componen el CSCF, es decir el P-CSCF, el I-CSCF y el S-CSCF. Esta parte del *core* está escrita totalmente en lenguaje C para Unix/Linux, lo cual implica que su instalación debe hacerse en este tipo de sistemas limitando de esta forma el rango de posibilidades donde esta puede ser implantada.

Por otra parte se encuentra la implementación del repositorio central del *core* IMS, el HSS la cual ha sido realizada en lenguaje Java, permitiendo de esta forma su despliegue en cualquier plataforma que cuente con una JVM (*Java Virtual Machine*).

- Ericsson Service Development Studio

Consiste en una herramienta para el desarrollo de aplicaciones convergentes IMS, tanto en el lado del servidor como en la parte del cliente, esta *suite* contiene un simulador de la red IMS basada en los estándares especificados para la misma, así como emuladores de servicios.

Esta herramienta está basada en el conocido IDE (*Integrated Development Environment*) Eclipse y provee una serie de APIs de alto nivel las cuales facilitan el desarrollo y despliegue de aplicaciones ocultando la complejidad de la red y de los dispositivos a los diseñadores. Además de esto tiene soporte para el servidor de aplicaciones SIP Sailfin/Glassfish e incluye mejoras para el soporte de comunicaciones P2P, VoIP y para el acceso a través de WLAN.

Al igual que la solución de Nokia-Siemens esta alternativa presenta el inconveniente que si bien es de libre uso, no se tiene acceso al código fuente o a configuraciones que permitan realizar modificaciones sustanciales en el *core* de la red a las que haya lugar al momento de introducir la plataforma middleware para realizar las pruebas de desempeño que permitan validar el funcionamiento de la propuesta planteada.

Por otra parte presenta una gran facilidad para desarrollar aplicaciones y servicios que no requieran de una manipulación de los elementos del *core*, simplificando de esta forma los procesos y agilizando la salida al mercado de soluciones específicas.

3.4.3.2. Herramientas para la plataforma *middleware*

- SEQUOIA

SEQUOIA, un proyecto *open source* de la empresa Continuent. Es una solución *middleware* muy completa capaz de manejar *clusters*, además de ofrecer mecanismos de balance de carga y de recuperación de fallos. Su implementación ha sido realizada 100% en lenguaje Java, lo cual garantiza su portabilidad a cualquier sistema que posea un JRE (*Java Runtime Environment*) superior al 1.4 así como la interoperabilidad con cualquier base de datos que provea un controlador JDBC (*Java Database Connectivity*). Entre sus principales ventajas se encuentra que para su implantación no es necesario hacer cambios sustanciales en las aplicaciones

existentes, ya que únicamente basta con cambiar la URL (*Universal Resource Locator*) de conexión hacia la base de datos.

El *core* del sistema está compuesto por controladores que implementan la tecnología RAIDb (*Redundant Array of Inexpensive Disks*), los cuales se encuentran replicados para de esta forma garantizar una alta disponibilidad y una mejor escalabilidad, estos mantienen la sincronización del *cluster* haciendo uso de primitivas de *group communication*, a través de HEDERA, un *wrapper* que permite ser configurado para trabajar con diferentes implementaciones de *group communication* tales como JGroups, Appia y Spread.

En cada uno de los controladores se ubica una base de datos virtual (VDB – *Virtual Database*), a la cual los clientes se conectan. A su vez esta base de datos virtual ejecuta todas las transacciones recibidas por parte de los clientes en los servidores de bases de datos, llamados *backends*, esta conexión se realiza a través del controlador JDBC adecuado para cada tipo de motor. En la configuración de la VDB se especifican el grado de replicación RAIDb a utilizar, así como el mecanismo de balance de carga y la implementación de *group comunication*.

- ParGRES

Es un *middleware* para manejo de *clusters* de bases de datos cuyo principal objetivo es hacer un proceso eficiente de consultas complejas haciendo uso de técnicas de paralelismo, mientras se hace la replicación.

Si bien este proyecto posee algunas características deseadas en la propuesta realizada, un aspecto desfavorable es el hecho de que no ha seguido una línea de desarrollo constante, ya que según la web oficial del proyecto no se han hecho actualizaciones ni modificaciones a la última versión desde el 2005.

3.4.3.3. Servidores de aplicaciones SIP

- Mobicents

Es un servidor de aplicaciones basado en JBoss, certificado para cumplir con las especificaciones del estándar de JSLEE (*Java Service Logic Execution Environment*), el cual está diseñado para el desarrollo de aplicaciones en el entorno de las telecomunicaciones, esto debido a sus características de baja latencia y alto *throughput*. Aunque ésta implementación es *open source* su licencia no es GPL, lo cual es un limitante si se necesitan realizar modificaciones sustanciales para la implantación de un determinado servicio o aplicación.

Una de sus principales ventajas es poseer un *plugin* para el IDE Eclipse, lo cual facilita el desarrollo de aplicaciones usando esta tecnología; sin embargo el uso de JSLEE comparado con otros estándares para el manejo de aplicaciones SIP como SIP Servlets, es mucho más complejo dada la robustez que este presenta.

- SailFin

El proyecto SailFin que cuenta con el apoyo de importantes empresas del sector de las telecomunicaciones como es el caso de Ericsson, se basa en la robusta y escalable tecnología

de Sip Servlets, haciendo uso del servidor de aplicaciones Glassfish v2. El objetivo de este proyecto es implementar la especificación SIP Servlet 1.1 (JSR 289) en su primera entrega, prevista para mediados del 2008.

La implementación de la tecnología Servlet SIP requiere un *stack* SIP para controlar las operaciones de señalización. Este *stack* es suministrado por Ericsson Inc, y actualmente lo utiliza en sus productos IMS.

El uso de Sailfin es completamente libre y su implementación es *open source*, además cuenta con un *plugin* para el desarrollo de aplicaciones mediante NetBeans.

- BEA Weblogic SIP Server

El Servidor SIP de BEA Weblogic es un servidor convergente para aplicaciones Java EE (*Enterprise Edition*), IMS-SIP y SOA, lo que quiere decir que provee un contenedor con soporte integrado para los estándares Java, de Servicios Web e IMS.

El servidor SIP es el líder del mercado en servidores convergentes para el nivel de servicios de IMS.

Para la correcta interoperabilidad con un *core* IMS, el Servidor SIP Weblogic ó WLSS (Weblogic Sip Server), provee varias interfaces con diferentes componentes del nivel de control del estándar IMS. Dichas interfaces son la Sh, ISC, Ro y Rf.

Weblogic también provee una serie de APIs para el desarrollo de aplicaciones SIP o DIAMETER con el objetivo de facilitar el despliegue de servicios IMS nativos en el WLSS. Así mismo, una interfaz grafica para la gestión de los servidores y de las y aplicaciones servicios desplegados, ayudan a configurar fácilmente el entorno y a controlar los procesos en curso.

3.4.3.4. Herramientas de medición y análisis de tráfico

- WireShark

Es el analizador de protocolos de uso más extendido alrededor del mundo, el cual ofrece todas las herramientas para la medición y generación de estadísticas necesarias en el análisis de los resultados.

En las pruebas del prototipo, WireShark presenta la posibilidad de observar los mensajes entre los nodos involucrados en la sincronización de la información.

3.4.4. Selección de herramientas

Una vez presentadas las herramientas disponibles para el uso en los diferentes módulos del prototipo, se seleccionaron las siguientes:

Open IMS Core: Se escogió está opción dadas sus facilidades en cuanto a la posibilidad de realizar cambios tanto en el código como en la configuración de sus diferentes componentes y aplicaciones, lo que permite una fácil adaptación de la plataforma *middleware*, además esta implementación ha

sido probada con diferentes AS y clientes SIP e IMS comprobando de esta forma su correcta interoperabilidad.

La modularización del *core* es un punto a favor muy importante, ya que para la implementación del prototipo únicamente es necesario interactuar con algunos de estos módulos. Por otra parte el hecho que tenga implementadas varias interfaces y protocolos definidos en los estándares facilita el desarrollo del proyecto y permite una implementación más realista del prototipo.

El uso de esta cama de pruebas en diferentes proyectos alrededor del mundo avala su utilidad y confiabilidad en el desarrollo del IMS.

SEQUOIA**:** entre las opciones presentadas ésta es la única que cumple con todas las características que se plantean en 3.4, además al ser un proyecto en constante evolución y desarrollo, se garantiza que la propuesta planteada en este proyecto no quede como una mera aproximación académica sino que pueda ser aplicada en un escenario empresarial.

BEA Weblogic SIP Server - WLSS: La posibilidad de trabajar con una licencia académica con uno de los mejores y más populares Servidores SIP fue una de las razones para seleccionar el WLSS. Por otra parte este servidor tiene implementadas varias interfaces y provee un buen número de APIs para el desarrollo de aplicaciones, característica que ayuda a reducir el tiempo de desarrollo de cualquier proyecto.

CAPITULO IV
IMPLEMENTACIÓN DEL PROTOTIPO PARA EL MECANISMO PROPUESTO

La implementación del prototipo fue dividida en dos etapas claramente distinguibles, primero se realizó la instalación, configuración y puesta a punto de las herramientas seleccionadas para cumplir con cada tarea, una vez completado lo anterior se procedió al desarrollo de aplicaciones que posteriormente permitieron realizar las pruebas planeadas.

Los procesos de instalación de cada una de las herramientas son descritos con detalle en los anexos A, B y C, de esta forma en las secciones siguientes sólo se realiza una descripción de los pasos de configuración necesarios para el correcto funcionamiento de cada módulo de la arquitectura presentada en 3.4.

4.1. CONFIGURACIÓN

4.1.1. Prerrequisitos

Para realizar la configuración y despliegue de las aplicaciones presentadas en este capítulo se debieron tener instaladas las siguientes herramientas:

- JRE 1.5 o superior
- Open IMS Core Rev. 570 o superior (Ver Anexo B)
- SEQUOIA V2.9 o Superior (Ver Anexo A)
- BEA Web Logic SIP Server V3.1 o superior (Ver Anexo C)

4.1.2. SEQUOIA

La configuración de SEQUIOA se realiza a través de diferentes archivos para cada uno de sus elementos, los cuales son descritos con detalle en el anexo A.

4.1.2.1. Configuración del Controlador

La configuración del controlador se realiza en el archivo controller.xml y esta afectará a todas las bases de datos virtuales que estén albergadas por este controlador.

Para iniciar la configuración se debe crear un archivo llamado controler.xml en la ruta config/controller/ del directorio de instalación de sequoia, el cual debe tener al menos la siguiente estructura:

- Identificador público al inicio del archivo, este verifica la consistencia de la configuración escrita con la plantilla original escrita por los creadores de sequoia.

  Para el controlador configurado se tienen las siguientes líneas:

  <?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE SEQUOIA-CONTROLLER PUBLIC "-//Continuent//DTD SEQUOIA-CONTROLLER 2.10.9//EN" "http://sequoia.continuent.org/dtds/sequoia-controller-2.10.9.dtd">

- Elemento Controller, especificando en el atributo ipAddress la dirección IP de la interfaz donde el controlador escuchará las peticiones. Así mismo, en el atributo port se debe detallar el puerto donde el controlador aceptará las conexiones que los clientes realicen a través del Sequoia-Driver. El puerto por defecto es el 25322.

  <SEQUOIA-CONTROLLER>
  <Controller ipAddress="10.200.2.190" port="25322">

- Elemento JmxSettings y dentro de este un elemento RmiJmxAdapter, especificando el puerto al que se realizan las conexiones. La definición de este último elemento permitirá administrar el controlador a través de la línea de comandos de SEQUIOA.

  <JmxSettings>
  <RmiJmxAdaptor port="1090"/>
  </JmxSettings>

Adicionalmente se pueden incluir líneas de configuración para que las bases de datos virtuales que alojará el controlador se carguen automáticamente, para ello se define dentro del elemento Controller el elemento VirtualDataBase, especificándole los atributos de archivo de configuración, el nombre y parámetros opcionales como por ejemplo, si que quiere que los *backends* se habiliten automáticamente al cargar el controlador.

```
<VirtualDatabase configFile="mysql.xml"
    virtualDatabaseName="repository" autoEnableBackends="false"/>
```

Así pues, el archivo de configuración completo para el controlador usado en las pruebas es como el siguiente:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE SEQUOIA-CONTROLLER PUBLIC "-//Continuent//DTD SEQUOIA-CONTROLLER 2.10.9//EN" "http://sequoia.continuent.org/dtds/sequoia-controller-2.10.9.dtd">
<SEQUOIA-CONTROLLER>
 <Controller ipAddress="10.200.2.190" port="25322">
  <JmxSettings>
   <RmiJmxAdaptor port="1090"/>
  </JmxSettings>
  <VirtualDatabase configFile="mysql.xml"
   virtualDatabaseName="repository" autoEnableBackends="false"/>
 </Controller>
</SEQUOIA-CONTROLLER>
```

4.1.2.2. Configuración de la base de datos virtual

Para la configuración de la base de datos virtual no hay una restricción en cuanto al nombre del archivo que la contiene, únicamente se debe tener en cuenta que este dentro la carpeta /config/virtualdabase de la instalación de sequoia.

El archivo de configuración debe poseer un elemento principal llamado SEQUOIA y dentro de este se definirán el resto de elementos y subelementos.

- Elemento VirtualDatabase: donde se debe definir por lo menos el nombre de la base de datos virtual, el cual será el que usen los clientes para realizar sus transacciones. De igual forma se pueden definir atributos adicionales como el número máximo de conexiones permitidas.

  <VirtualDatabase name="repository" maxNbOfConnections="0">

- Distribution: este elemento se usa para habilitar el uso de las primitivas de *group communication* y de esta forma garantizar que el controlador no va a ser un punto de fallo que formará cuellos de botella. Además se define la forma en la que el controlador va a manejar los mensajes de *group communication*.

  ```
  <Distribution>
    <MessageTimeouts/>
  </Distribution>
  ```

- Backup: se debe definir cuál será el manejador de los *backups* de la base de datos virtual, en este caso se escoge el de mysql dado que todos los *backends* utilizan este motor.

  ```
  <Backup>
  <BackuperbackuperName="mysql" ClassName="org.continuent.sequoia
  .controller.backup.backupers.MySQLBackuper" />
   </Backup>
  ```

- Athentication Manager: aquí se deben definir los nombres de usuario y *passwords* del administrador de la base de datos virtual y de los usuarios que tendrán derecho a utilizar esta misma.

  ```
  <AuthenticationManager transparentLogin="on">
          <Admin>
                  <User username="admin" password="admin" />
           </Admin>
          <VirtualUsers>
                  <VirtualLogin vLogin="user" vPassword="tesis" />
           </VirtualUsers>
   </AuthenticationManager>
  ```

- DataBaseBackend: aquí se definen todos los *backends* que van a formar parte de la base de datos virtual, especificando el controlador JDBC que cada uno de estos usa. En este punto se deben tener en cuenta ciertos elementos propios de cada DBMS tal como el connection

statement. Cabe anotar que para el correcto funcionamiento de SEQUIOA se debe habilitar el acceso remoto al motor de bases de datos.

```
<DatabaseBackend name="as" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://thor.unicauca.edu.co:3306/user_data"
driverPath="/home/earmero/sequoia2/drivers/mysql-connector-java-3.0.15-ga-
bin.jar" connectionTestStatement="select 1" nbOfBackendWorkerThreads="5">
<ConnectionManager vLogin="user" rLogin="userd" rPassword="data">
 <VariablePoolConnectionManager initPoolSize="40" maxPoolSize="0"
idleTimeout="0" waitTimeout="0" />
 </ConnectionManager>
 </DatabaseBackend>
```

- RequestManager: en este elemento se deben definir el mecanismo de balance de carga y el grado de replicación deseado, así como la definición de la base de datos del recovery log, la cual depende del DBMS que se use para esta tarea. En este caso se definió utilizar un grado de replicación RAIDb-2, ya que esta permite manejar una actualización a nivel de tablas y se adapta perfectamente a la arquitectura planteada en 3.4.

```
<RequestManager caseSensitiveParsing="false" beginTimeout="60"
commitTimeout="60" rollbackTimeout="60">
        <RequestScheduler>
                <RAIDb-2Scheduler level="passThrough" />
         </RequestScheduler>
        <RequestCache>
        <MetadataCache maxNbOfMetadata="10000" maxNbOfField="0" />
        <ParsingCache backgroundParsing="false" maxNbOfEntries="5000" />
        </RequestCache>
        <LoadBalancer transactionIsolation="databaseDefault">
                <RAIDb-2>
                <WaitForCompletion policy="first" enforceTableLocking="false"
        deadlockTimeoutInMs="30000" />
                  <RAIDb-2-LeastPendingRequestsFirst />
                </RAIDb-2>
         </LoadBalancer>
```

La configuración de la base de datos del recovery log que se utilizó para la validación del mecanismo propuesto corresponde a la de una dase de datos MySQL y se muestra a continuación.

```
<RecoveryLog driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://thor.unicauca.edu.co:3306/log" login="log" password="loger"
requestTimeout="60" recoveryBatchSize="10" idleConnectionTimeout="0">
        <RecoveryLogTable tableName="RECOVERY" logIdColumnType="BIGINT
        NOT NULL" vloginColumnType="TEXT NOT NULL" sqlColumnName="sql2"
```

```
        sqlColumnType="TEXT NOT NULL" sqlParamColumnType="TEXT NOT NULL"
        transactionIdColumnType="BIGINT NOT NULL"
        extraStatementDefinition=",PRIMARY KEY (log_id)" createTable="CREATE
        TABLE" autoConnTranColumnType="CHAR(1) NOT NULL"
        requestIdColumnType="BIGINT" execTimeColumnType="BIGINT"
        updateCountColumnType="INT" />

        <CheckpointTable tableName="CHECKPOINT"
        checkpointNameColumnType="VARCHAR(200) NOT NULL"
        createTable="CREATE TABLE" logIdColumnType="BIGINT"
        extraStatementDefinition=",PRIMARY KEY (name)" />

        <BackendTable tableName="BACKEND" databaseNameColumnType="TEXT
        NOT NULL" backendNameColumnType="TEXT NOT NULL"
        checkpointNameColumnType="TEXT NOT NULL" createTable="CREATE
        TABLE" backendStateColumnType="INTEGER" extraStatementDefinition=""
        />

        <DumpTable tableName="DUMP" dumpNameColumnType="TEXT NOT
        NULL" dumpDateColumnType="TIMESTAMP" dumpPathColumnType="TEXT
        NOT NULL" dumpFormatColumnType="TEXT NOT NULL"
        checkpointNameColumnType="TEXT NOT NULL"
        backendNameColumnType="TEXT NOT NULL" tablesColumnType="TEXT
        NOT NULL" createTable="CREATE TABLE" tablesColumnName="tables"
        extraStatementDefinition="" />
    </RecoveryLog>
```

Así, el archivo de configuración completo llamado mysql.xml es de la siguiente forma:

```
<VirtualDatabase name="repository" maxNbOfConnections="0">
<Distribution>
   <MessageTimeouts/>
</Distribution>
<Backup>
        <BackuperbackuperName="mysql" ClassName="org.continuent.sequoia
        .controller.backup.backupers.MySQLBackuper" />
 </Backup>

<AuthenticationManager transparentLogin="on">
<Admin>
        <User username="admin" password="admin" />
 </Admin>
<VirtualUsers>
        <VirtualLogin vLogin="user" vPassword="tesis" />
```

```
 </VirtualUsers>
 </AuthenticationManager>

<DatabaseBackend name="as" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://thor.unicauca.edu.co:3306/user_data"
driverPath="/home/earmero/sequoia2/drivers/mysql-connector-java-3.0.15-ga-bin.jar"
connectionTestStatement="select 1" nbOfBackendWorkerThreads="5">
<ConnectionManager vLogin="user" rLogin="userd" rPassword="data">
<VariablePoolConnectionManager initPoolSize="40" maxPoolSize="0" idleTimeout="0"
waitTimeout="0" />
 </ConnectionManager>
 </DatabaseBackend>

<DatabaseBackend name="hss" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://anubis.unicauca.edu.co:3306/hss_bd"
driverPath="/home/earmero/sequoia2/drivers/mysql-connector-java-3.0.15-ga-bin.jar"
connectionTestStatement="select 1" nbOfBackendWorkerThreads="5">
<ConnectionManager vLogin="user" rLogin="hss" rPassword="hss">
 <VariablePoolConnectionManager initPoolSize="40" maxPoolSize="0" idleTimeout="0"
waitTimeout="0" />
 </ConnectionManager>
 </DatabaseBackend>

<RequestManager caseSensitiveParsing="false" beginTimeout="60" commitTimeout="60"
rollbackTimeout="60">
        <RequestScheduler>
           <RAIDb-2Scheduler level="passThrough" />
         </RequestScheduler>
        <RequestCache>
          <MetadataCache maxNbOfMetadata="10000" maxNbOfField="0" />
         <ParsingCache backgroundParsing="false" maxNbOfEntries="5000" />
        </RequestCache>
           <LoadBalancer transactionIsolation="databaseDefault">
        <RAIDb-2>
        <WaitForCompletion policy="first" enforceTableLocking="false"
        deadlockTimeoutInMs="30000" />
            <RAIDb-2-LeastPendingRequestsFirst />
        </RAIDb-2>
          </LoadBalancer>
<RecoveryLog driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://thor.unicauca.edu.co:3306/log" login="log" password="loger"
requestTimeout="60" recoveryBatchSize="10" idleConnectionTimeout="0">
        <RecoveryLogTable tableName="RECOVERY" logIdColumnType="BIGINT NOT
        NULL" vloginColumnType="TEXT NOT NULL" sqlColumnName="sql2"
```

sqlColumnType="TEXT NOT NULL" sqlParamColumnType="TEXT NOT NULL" transactionIdColumnType="BIGINT NOT NULL" extraStatementDefinition=",PRIMARY KEY (log_id)" createTable="CREATE TABLE" autoConnTranColumnType="CHAR(1) NOT NULL" requestIdColumnType="BIGINT" execTimeColumnType="BIGINT" updateCountColumnType="INT" />

<CheckpointTable tableName="CHECKPOINT" checkpointNameColumnType="VARCHAR(200) NOT NULL" createTable="CREATE TABLE" logIdColumnType="BIGINT" extraStatementDefinition=",PRIMARY KEY (name)" />

<BackendTable tableName="BACKEND" databaseNameColumnType="TEXT NOT NULL" backendNameColumnType="TEXT NOT NULL" checkpointNameColumnType="TEXT NOT NULL" createTable="CREATE TABLE" backendStateColumnType="INTEGER" extraStatementDefinition="" />

<DumpTable tableName="DUMP" dumpNameColumnType="TEXT NOT NULL" dumpDateColumnType="TIMESTAMP" dumpPathColumnType="TEXT NOT NULL" dumpFormatColumnType="TEXT NOT NULL" checkpointNameColumnType="TEXT NOT NULL" backendNameColumnType="TEXT NOT NULL" tablesColumnType="TEXT NOT NULL" createTable="CREATE TABLE" tablesColumnName="tables" extraStatementDefinition="" />

</RecoveryLog>
</RequestManager>
</VirtualDatabase>
</SEQUOIA>

4.1.3. Open IMS Core

4.1.3.1. Configuración del HSS

Para realizar la integración entre el SEQUIOA y Open IMS Core se realizaron algunos cambios en la configuración de este último.

Primero que todo se cambió la forma en que el HSS consulta su base de datos. Teniendo en cuenta que el desarrollo del FHoSS (FOKUS HSS) utiliza el patrón de acceso a datos de Hibernate, esta tarea se simplifica enormemente y únicamente bastó con cambiar algunas líneas en el archivo /opt/openimscore/fhoss/deploy/hibernate.properties.

Ahora una vez realizado esto, se debió proporcionar al Open IMS Core el controlador de SEQUOIA para que a través de este realice las conexiones a su base de datos. Para realizar esta tarea bastó con copiar el controlador de Sequoia en la carpeta /opt/openimscore/fhoss/lib de la instalación del OpenIMS Core, ya que aquí es donde por defecto el va a buscar cualquier librería externa.

Por último se debió permitir el acceso externo a la base de datos, esto con el fin de garantizarle al controlador SEQUIOA un adecuado manejo del *backend* denominado HSS.

Para realizar esto fue necesario cambiar la configuración que por defecto utiliza MySQL la cual se encuentra en el archivo /etc/mysql/my.cfn y cambiar la línea:

BindAddrees 127.0.0.1

Por

BindAddress 10.200.2.190

Esta última es la dirección IP correspondiente a la interfaz a la cual se conecta el controlador. Una vez realizado esto bastó con reiniciar el servicio de MySQL para que todo quede preparado para las pruebas a las que se dio lugar.

4.2. DESARROLLO DE APLICACIONES

4.2.1. Aplicación Generadora de Información - AGI

La Aplicación Generadora de Información, (AGI) consiste en una aplicación java, la cual pretende simular un servidor que genera información de los usuarios IMS. Esta aplicación puede ser configurada mediante una interfaz, siendo posible establecer parámetros como el número de usuarios para los que se está actualizando su información y la frecuencia con que estas actualizaciones ocurren.

4.2.1.1. Diagrama de casos de uso

En la Figura 11 se muestra el diagrama de casos de uso para la AGI.

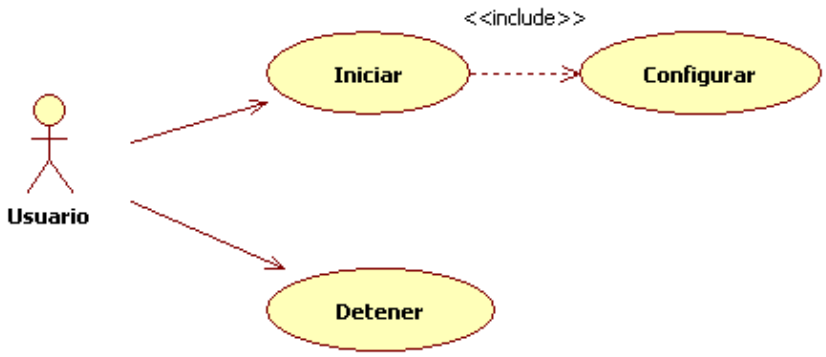

Figura 11. Diagrama de casos de uso para la AGI.

- Actor

  Usuario: este es el actor que interactúa directamente con la aplicación y es el encargado de realizar las tareas de configuración para la simulación que se desee realizar.

- Casos de uso

Caso de uso	Configurar
Actor	Usuario
Descripción	En este caso de uso se toman los parámetros de configuración proporcionados, ya sea por el usuario o los predeterminados de la aplicación, para iniciar la simulación, los parámetros que pueden ser variados son el número de usuarios que están actualizando la información, la frecuencia con la que se realizan las actualizaciones y el método a través del cual se va a realizar la actualización.

Caso de uso	Iniciar
Actor	Usuario
Descripción	Inicia la simulación del servidor de localización

Caso de uso	Detener
Actor	Usuario
Descripción	Detiene una simulación previamente iniciada

4.2.1.2. Descripción del los casos de uso esenciales

Caso de uso	Iniciar
Propósito	Iniciar la simulación del servidor de Localización
Actor	Usuario
Precondiciones	• La aplicación web Gateway debe estar corriendo en un servidor de aplicaciones • Se debe realizar el proceso de configuración de los parámetros de simulación, ya sea manualmente o con valores por defecto.
Resumen	Es activado cuando se desea dar inicio a la simulación. Si el usuario no ha realizado una configuración previa de los parámetros se tomarán valores por defecto.
Excepciones	• No puede establecerse conexión con la Gateway. • El usuario no está registrado en la red • Los valores de de configuración introducidos son inválidos

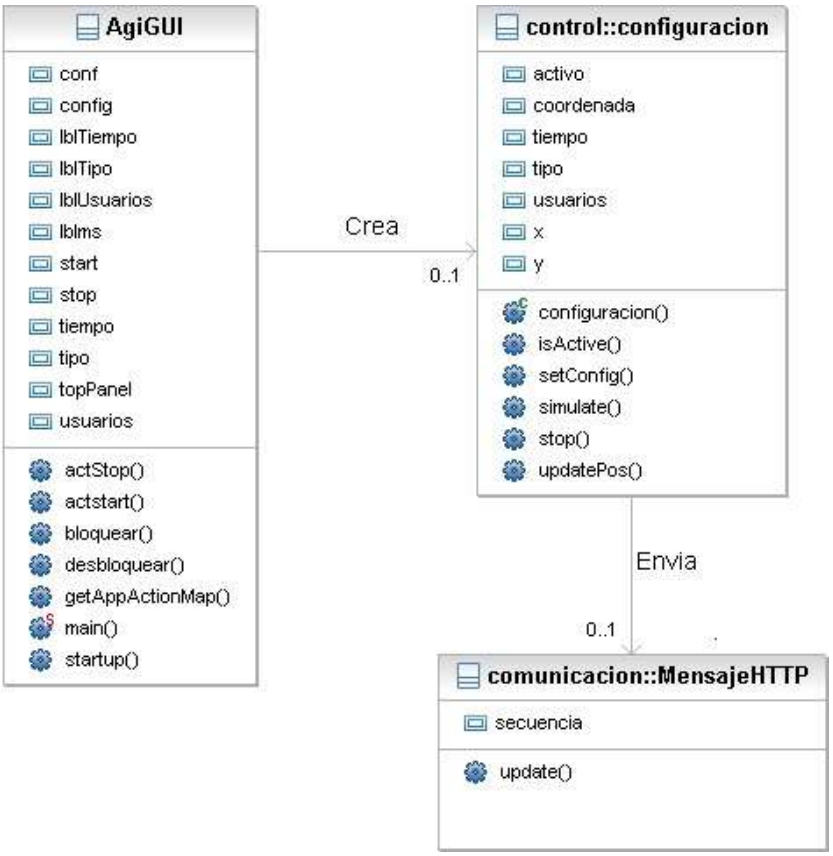

Figura 12. Diagrama de clases del caso de uso Iniciar.

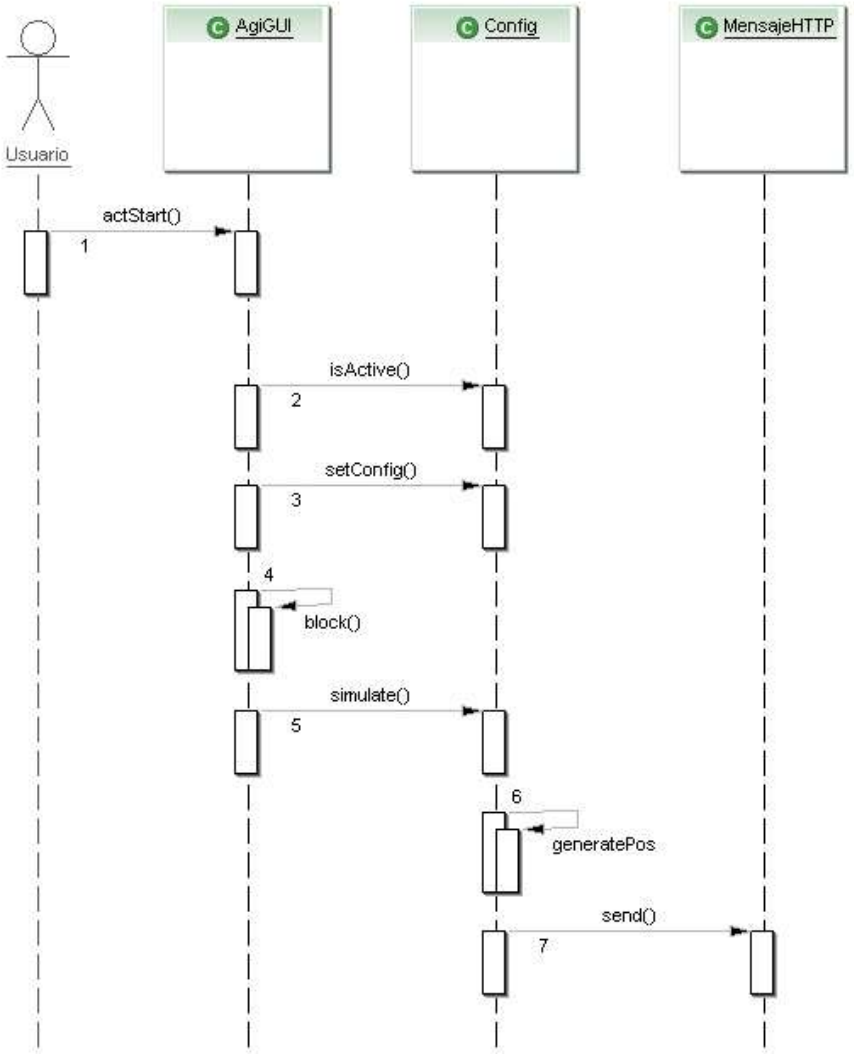

Figura 13. Diagrama de secuencia del caso de uso Iniciar.

Caso de uso	Detener
Propósito	Detener la simulación del servidor de localización
Actor	Usuario
Precondiciones	• Exista una simulación actualmente corriendo
Resumen	Este caso de uso es activado cuando el usuario desea terminar o interrumpir una simulación.
Excepciones	• No existe ninguna simulación corriendo

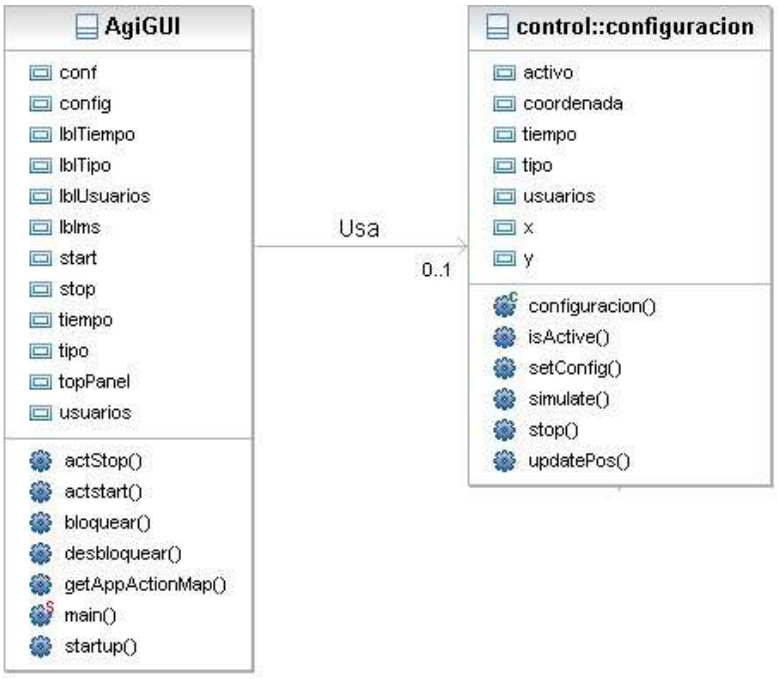

Figura 14. Diagrama de clases del caso de uso Detener.

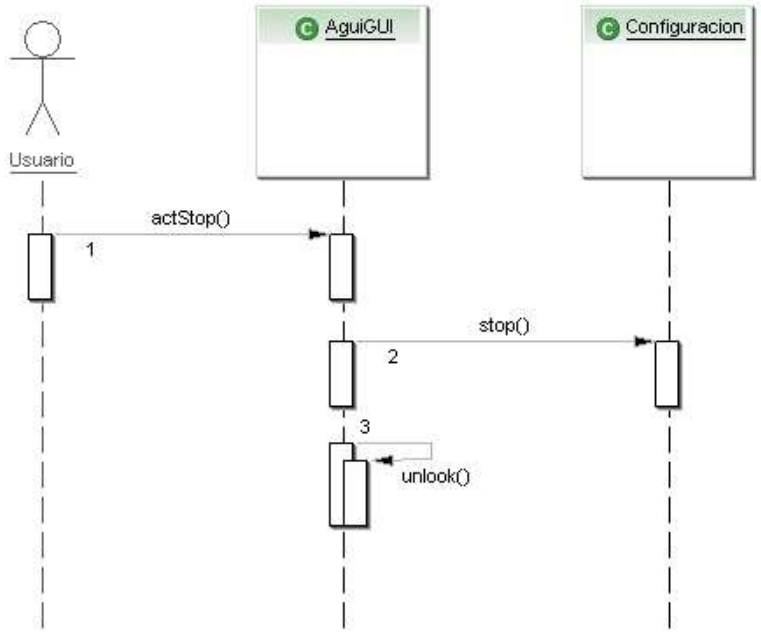

Figura 15. Diagrama de secuencia del caso de uso Detener.

4.2.2. Aplicación Web Gateway

Es una aplicación Java cuya función es realizar la traducción de los mensajes HTTP enviados por la AGI en los correspondientes mensajes que son enviados al controlador SEQUOIA para realizar el proceso de sincronización de información. Es de resaltar que esta aplicación se realizó únicamente

para realizar pruebas comparativas con dos mecanismos en escenarios semejantes (ver Capítulo V) y de esta forma obtener resultados más cercanos a la realidad, ya que hubiese sido posible realizar la conexión de la AGI directamente al controlador SEQUOIA.

4.2.2.1. Diagrama de casos de uso

En la Figura 16 se muestra el diagrama de casos de uso para la aplicación web Gateway.

Figura 16. Diagrama de casos de uso para la Gateway.

- Actor

El actor hace uso de la funcionalidad provista por esta aplicación es la aplicación generadora de información presentada anteriormente.

- Casos de Uso

Caso de uso	Traducir
Actor	AGI
Descripción	Realiza la conversión de formato de los mensajes enviados por la AGI

Caso de uso	Enviar
Actor	AGI
Descripción	Una vez que se ha realizado la traducción de los mensajes se los envía hacia el controlador SEQUOIA

4.2.2.2. Descripción del los casos de uso esenciales

Caso de uso	Traducir
Propósito	Convertir los mensajes HTTP enviados por la AGI en mensajes SEQUIOA
Actor	AGI
Precondiciones	• Debe existir un controlador SEQUIOA al cual enviar los mensajes.
Resumen	Este Caso de uso es activado cuando la AGI accede a la URL donde esta corriendo la Gateway ya sea a través de peticiones GET ó POST
Excepciones	• El usuario no está registrado en la red

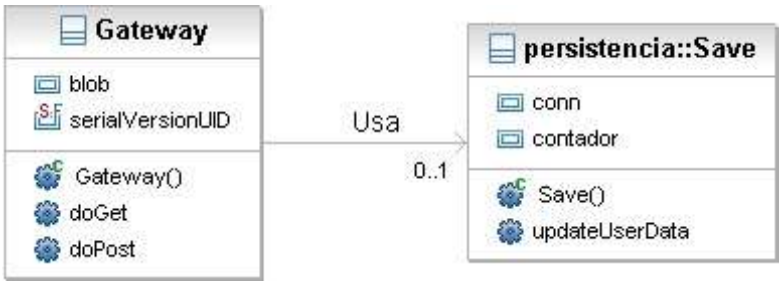

Figura 17. Diagrama de clases del caso de uso Traducir.

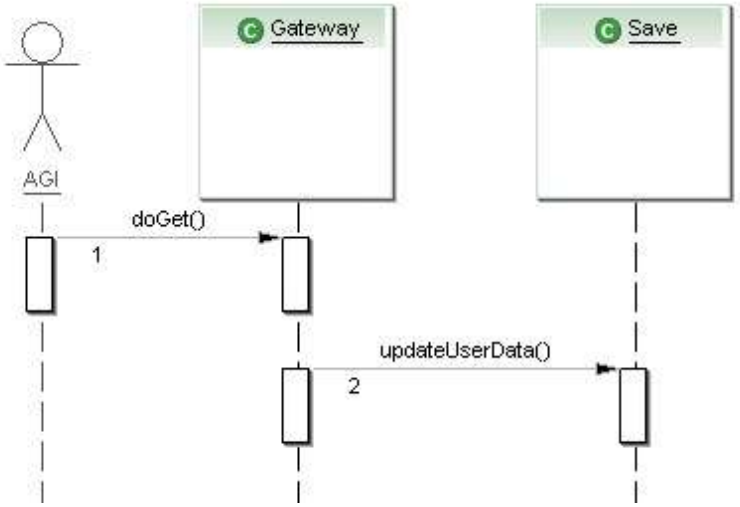

Figura 18. Diagrama de secuencia del caso de uso Traducir.

CAPITULO V
VALIDACIÓN DEL MECANISMO DE SINCRONIZACIÓN DE INFORMACIÓN PROPUESTO

En el Capítulo III se describió la propuesta en una forma general y en el Capítulo IV se expuso la forma de implementar el prototipo para la validación de la misma. Continuando con el proceso, en este apartado se describe el escenario de pruebas planteado para validar el funcionamiento del mecanismo propuesto y probar el mecanismo de sincronización de información en IMS a través de la interfaz Sh, el cual se tomó como punto de referencia para realizar el análisis de desempeño de la propuesta.

Por último, con los resultados obtenidos en las pruebas de los dos mecanismos, se realizó un análisis comparativo con el propósito de determinar las ventajas y desventajas de la propuesta.

5.1. DESCRIPCIÓN DEL ESCENARIO GENERAL DE PRUEBAS

Para la validación de la propuesta, se plantea la simulación de un servicio en la Arquitectura de Servicios Nativos IMS, el cual genera y requiere información del *Repository Data*.

Este servicio de pruebas se establece dentro de los servicios en tiempo real, siendo estos los más críticos, en cuanto a la exigencia de rapidez y consistencia a la hora de requerir información de diferentes entidades.

El servicio simulado es el de localización de un usuario móvil y de lugares de interés en mapas que se descargan desde el AS al equipo del usuario. Por ejemplo, es de gran utilidad para un ejecutivo que se encuentra en una ciudad que no conoce, trasladándose en un taxi desde el aeropuerto y necesita saber si va en la dirección correcta hacia el hotel donde tiene reservaciones. Entonces él usa su teléfono móvil para conocer su ubicación actual y al mismo tiempo va ubicando en el mapa lugares como restaurantes, bancos, casas de cambio y otros que le pueden ser útiles durante su estadía.

EL AS se encarga de actualizar la ubicación del usuario en el mapa de acuerdo a la información suministrada por una Aplicación Generadora de Información (AGI) descrita en la sección 4.2.1 que simula un servidor de localización. Por este motivo, en este capítulo la AGI será llamada Aplicación Generadora de Coordenadas (AGC).

La AGC genera las coordenadas para cada usuario, las cuales se guardan en la base de datos del HSS (específicamente en el *Repository Data*) y en la base de datos del AS y son actualizadas cada vez que se genere otra actualización usando un mecanismos de sincronización de información.

Se plantearon dos escenarios, en el primero se usó el mecanismo de sincronización de la interfaz Sh de IMS y para el segundo se usó el mecanismo propuesto empleando una plataforma *middleware* que implemente las técnicas basadas en *group communication*. Como ya se había dicho anteriormente, se realizaron pruebas con estos dos mecanismos con el objetivo de tener un punto de referencia que permita evaluar los resultados obtenidos.

Con cada mecanismo se realizaron tres capturas de los mensajes que se transfirieron entre los nodos que intervienen en el proceso. En cada captura se mantuvo fijo el número de actualizaciones

en 50, pero se variaron el número de usuarios para el cual la AGC está actualizando la información de localización y el retardo entre cada actualización después de recibir un acuse de recibo en la AGC. Esto se hizo con el ánimo de variar la carga de la red en cuanto a procesamiento y número de bytes por segundo que se transporta por la misma.

El número de usuarios se varió entre 10, 50 y 100, mientras que el retardo para cada captura fué 500 ms, 250 ms y 125 ms. Así, la captura con menos carga para la red fue con 10 usuarios y con un retardo de 500 ms entre cada actualización y la captura con más carga para la red fue con 100 usuarios y un retardo de 125 ms.

Para cada uno de los casos que se presentan, se tomaron estadísticas del tiempo que se tardó cada mecanismo en completar todas las actualizaciones y el número de Bytes que se generaron y se trasportaron por en la red.

El analizador de protocolos Wireshark, además suministró estadísticas de otros parámetros, los cuales se describen a continuación.

- Tiempo entre el primer y el último paquete (segs): es el tiempo total que se tarda el mecanismo en procesar todas las peticiones y enviar los mensajes a los nodos que correspondan.
- Paquetes: es el número total de paquetes generados por todos los nodos que intervienen en el mecanismo.
- Promedio Paquetes/seg: es el promedio de paquetes generados que se transportan por la red en un segundo.
- Promedio tamaño de paquetes (Bytes): es el promedio del tamaño de todos los paquetes generados.
- Bytes: es el número de Bytes total generados por todos los nodos que intervienen en el mecanismo.
- Promedio Bytes/seg: es el promedio de Bytes generados que se transportan por la red en un segundo.
- Promedio Mbit/seg: es el promedio de Mega bits de datos que se transportan por la red en un segundo, lo cual indica la carga promedio de la red en un segundo.

5.2. ESCENARIO A

5.2.1. Descripción

En este escenario se probó el mecanismo de sincronización de IMS, el cual hace uso de la interfaz Sh para manejar los mensajes DIAMETER que contienen la información que se está actualizando.

Para realizar las pruebas se hizo uso de dos SIP AS de BEA Weblogic (WLSS) y del HSS del Open IMS Core (FHoSS), estos nodos con la característica de tener la interfaz Sh habilitada.

Uno de los WLSS funciona como una Gateway entre la AGC y el FHoSS, convirtiendo las peticiones HTTP que envía la primera en mensajes DIAMETER de petición de actualización de perfil (*Profile Update Request* - PUR).

El otro WLSS es el AS prestador del servicio de actualización. Este AS se encuentra suscrito a las notificaciones de cambios en el *Repository Data* y por ende recibe las actualizaciones de dicha información mediante mensajes DIAMETER PNR (*Push-Notification-Request*).

La implementación de este mecanismo se describe detalladamente en el Anexo D.

En la Figura 19 se puede apreciar la arquitectura de pruebas para el mecanismo de sincronización de información usando la interfaz Sh descrita en esta sección.

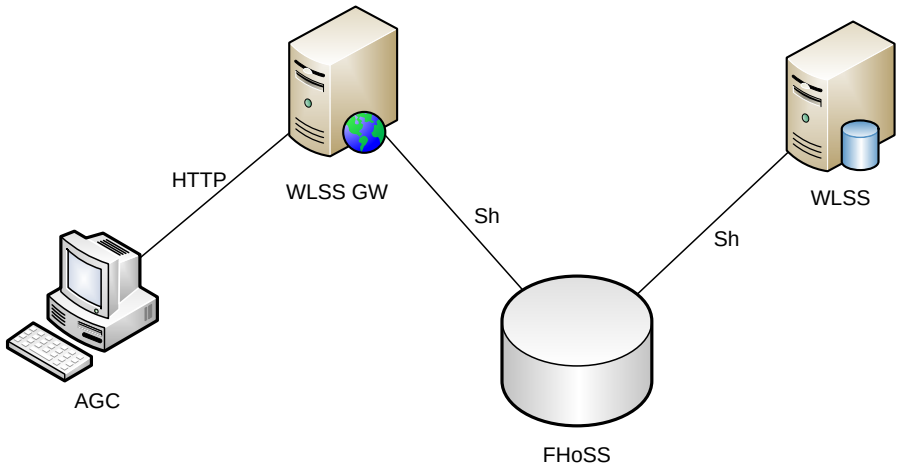

Figura 19. Arquitectura para el Escenario A.

Para esta prueba se capturaron los mensajes DIAMETER de los procesos especificados para la interfaz Sh y los mensajes MySQL de consulta a la tabla del *Repository Data* en la base de datos del FHoSS. Estos son los mensajes que llevan la información y que se generan en el proceso de sincronización de información usando este mecanismo.

El flujo de información durante una actualización se puede observar en la Figura 20.

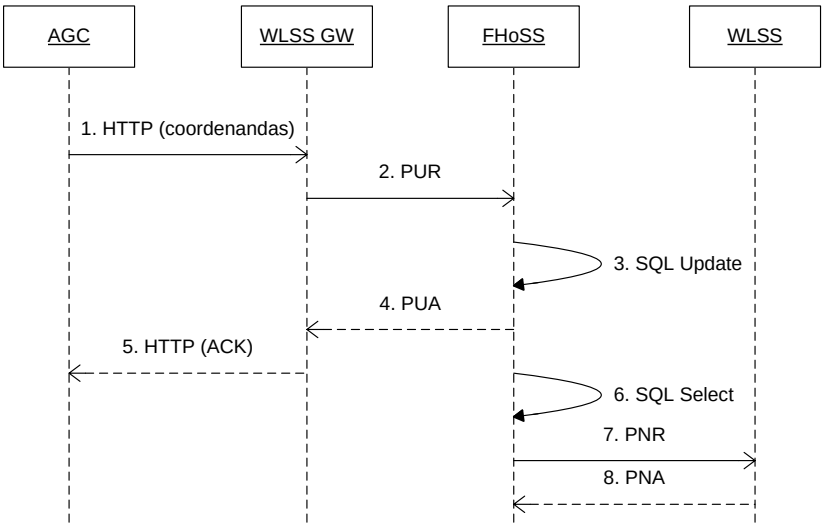

Figura 20. Flujo de Información en el Escenario A

1. La AGC envía un mensaje HTTP con la información generada hacia el Gateway WLSS (WLSS GW).

2. El WLSS GW envía un mensaje DIAMETER PUR hacia el FHoSS con la información que se va a actualizar en el *Repository Data*.

3. Con la llegada del PUR, el FHoSS ejecuta el comando *SQL update* para actualizar la información en su base de datos.

4. El FHoSS responde con un mensaje PUA de confirmación de actualización.

5. Con la llegada del PUA, el WLSS GW envía un ACK HTTP a la AGC para que ésta envíe otra petición después del retardo establecido.

6. El FHoSS prepara la notificación de cambios en la información de su base de datos consultando en su base de datos la información que se va a enviar al WLSS mediante el comando SQL *locate* .

7. El FHoSS envía la información actualizada en el *Repository Data* al WLSS a través de un mensaje PNR.

8. El WLSS responde con un PNA al FHoSS.

5.2.2. Resultados obtenidos

- Captura 1

  - Número de Usuarios: 10.
  - Retardo de Actualización: 500 ms.

Tiempo entre el primer y el último paquete (segs)	61,838
Paquetes	2838
Promedio Paquetes/seg	45,894
Promedio tamaño de paquetes (Bytes)	288,001
Bytes	817346
Promedio Bytes/seg	13217,531
Promedio Mbit/seg	0,106

Tabla 1. Resultados Escenario A Captura 1

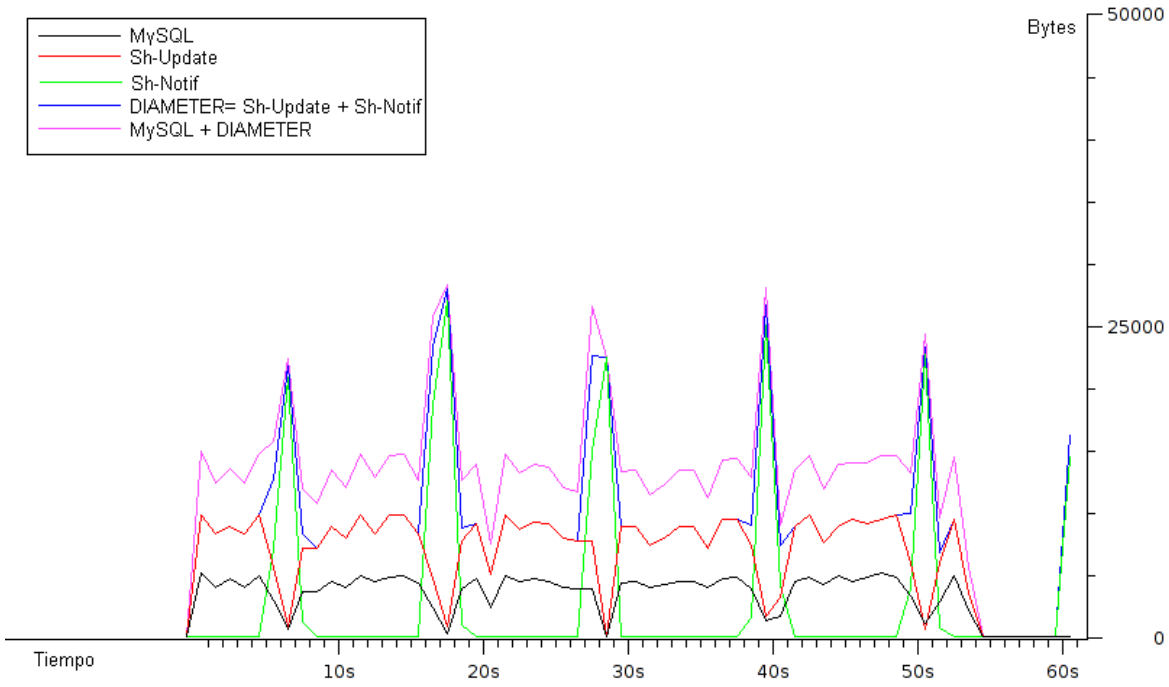

Figura 21. Escenario A Captura 1

- Captura 2
  - Número de Usuarios: 50
  - Retardo de Actualización: 500 ms.

Tiempo entre el primer y el último paquete (segs)	170,655
Paquetes	15690
Promedio Paquetes/seg	91,940
Promedio tamaño de paquetes (Bytes)	295,919
Bytes	4642968
Promedio Bytes/seg	27206,752
Promedio Mbit/seg	0,218

Tabla 2. Resultados Escenario A Captura 2

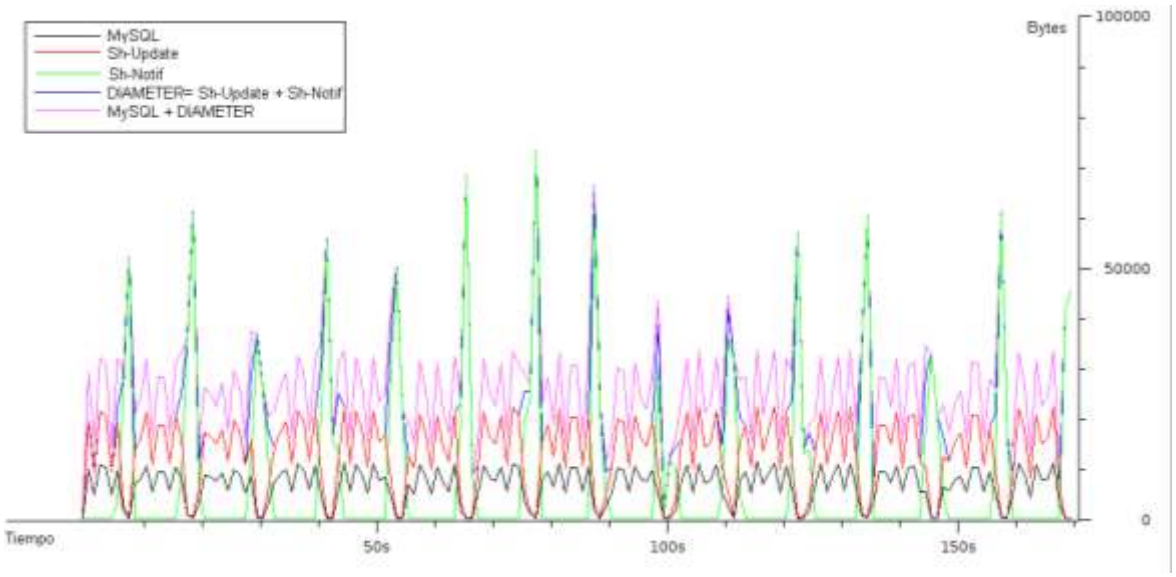

Figura 22. Escenario A Captura 2

- Captura 3
  - Número de Usuarios: 100
  - Retardo de Actualización: 500 ms.

Tiempo entre el primer y el último paquete (segs)	282,132
Paquetes	32030
Promedio Paquetes/seg	113,528
Promedio tamaño de paquetes (Bytes)	299,912
Bytes	9606182
Promedio Bytes/seg	34048,488
Promedio Mbit/seg	0,272

Tabla 3. Resultados Escenario A Captura 3

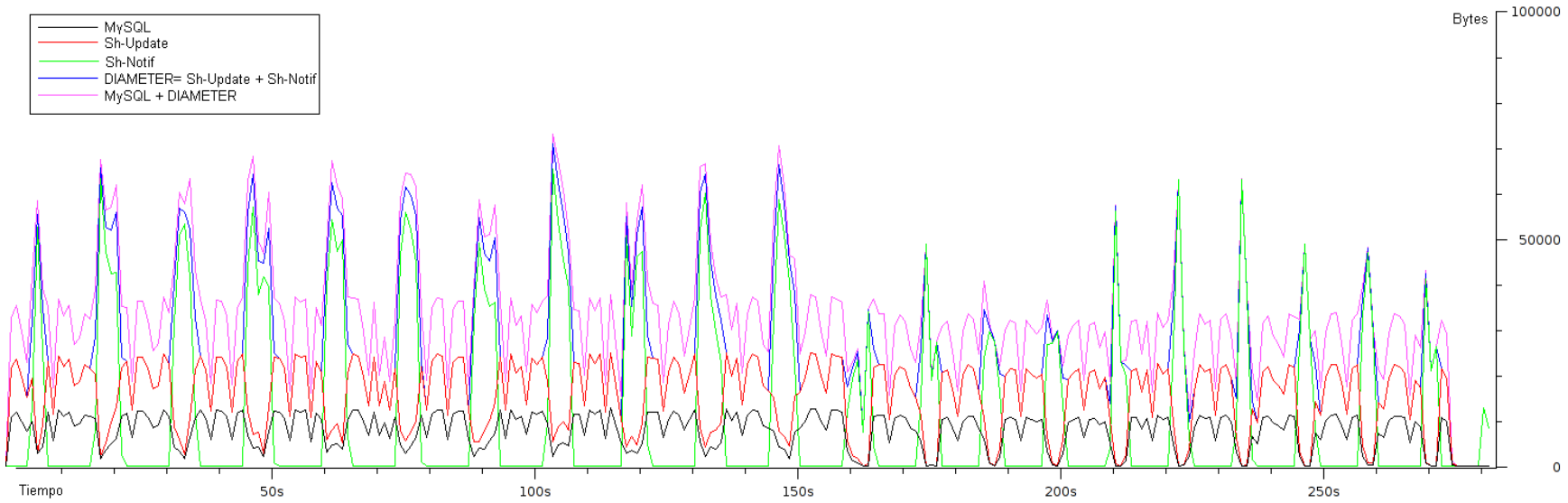

Figura 23. Escenario A Captura 3

- Captura 4
  - Número de Usuarios: 10
  - Retardo de Actualización: 250 ms.

Tiempo entre el primer y el último paquete (segs)	40,923
Paquetes	2953
Promedio Paquetes/seg	72,160
Promedio tamaño de paquetes (Bytes)	291,264
Bytes	860102
Promedio Bytes/seg	21017,567
Promedio Mbit/seg	0,168

Tabla 4. Resultados Escenario A Captura 4

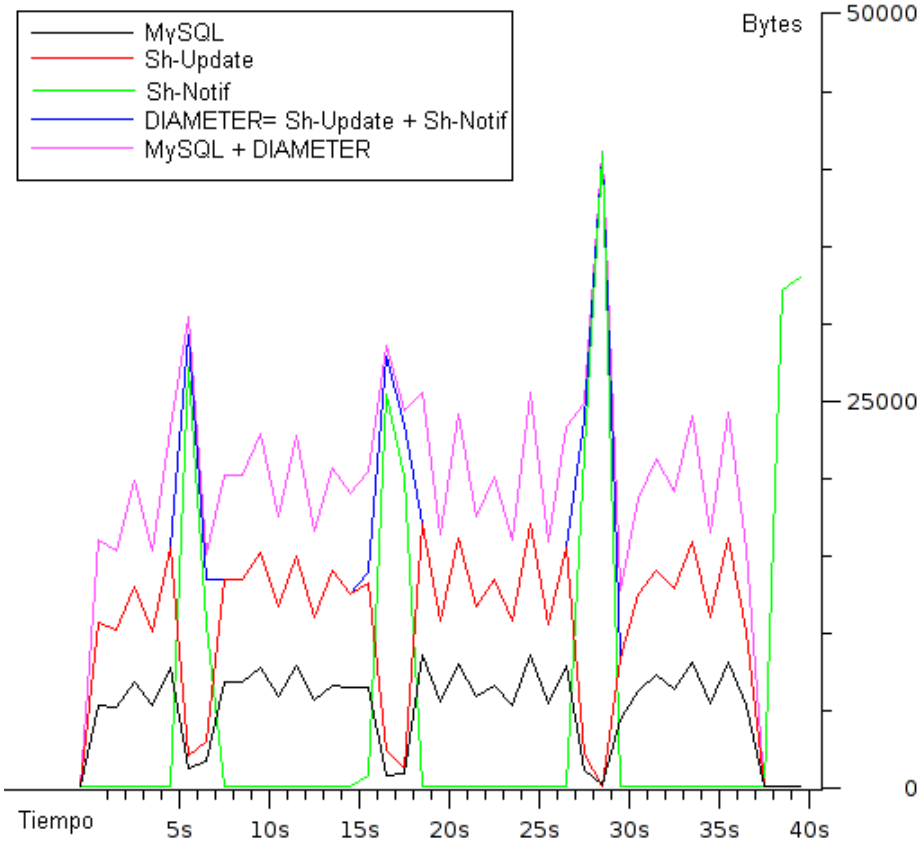

Figura 24. Escenario A Captura 4

- Captura 5
  - Número de Usuarios: 50
  - Retardo de Actualización: 250 ms.

Tiempo entre el primer y el último paquete (segs)	138,124
Paquetes	16008
Promedio Paquetes/seg	115,896
Promedio tamaño de paquetes (Bytes)	300,714
Bytes	4813824
Promedio Bytes/seg	34851,367
Promedio Mbit/seg	0,279

Tabla 5. Resultados Escenario A Captura 5

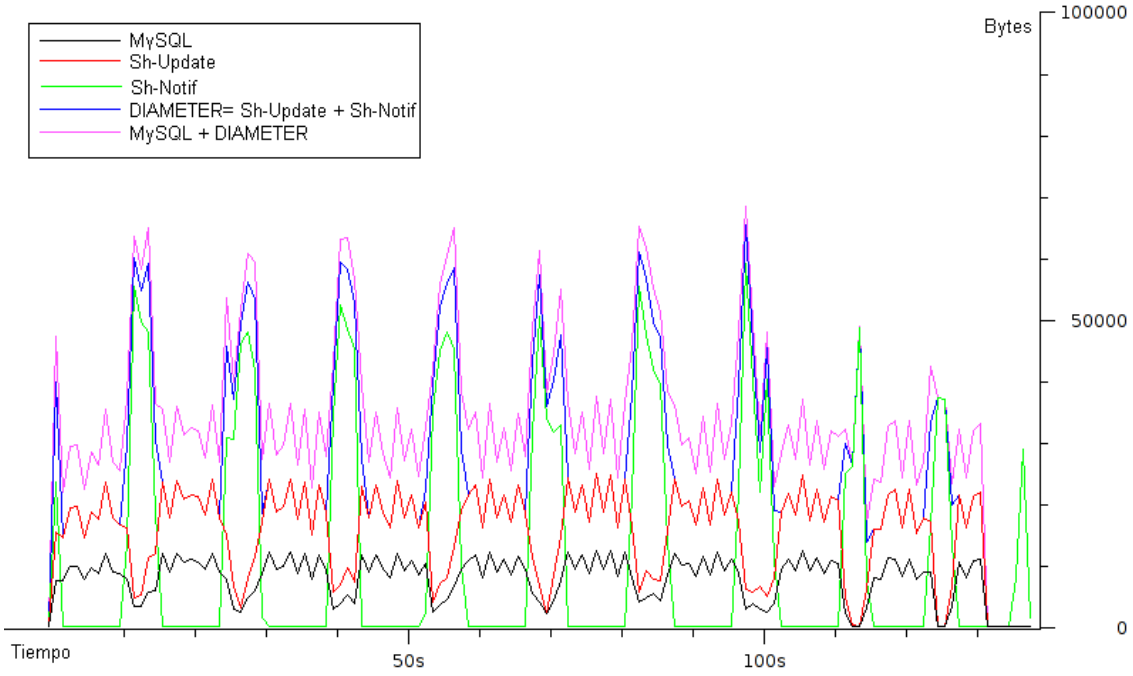

Figura 25. Escenario A Captura 5

- Captura 6
  - Número de Usuarios: 100
  - Retardo de Actualización: 250 ms.

Tiempo entre el primer y el último paquete (segs)	296,771
Paquetes	31861
Promedio Paquetes/seg	107,359
Promedio tamaño de paquetes (Bytes)	298,001
Bytes	9494601
Promedio Bytes/seg	31993,041
Promedio Mbit/seg	0,256

Tabla 6. Resultados Escenario A Captura 6

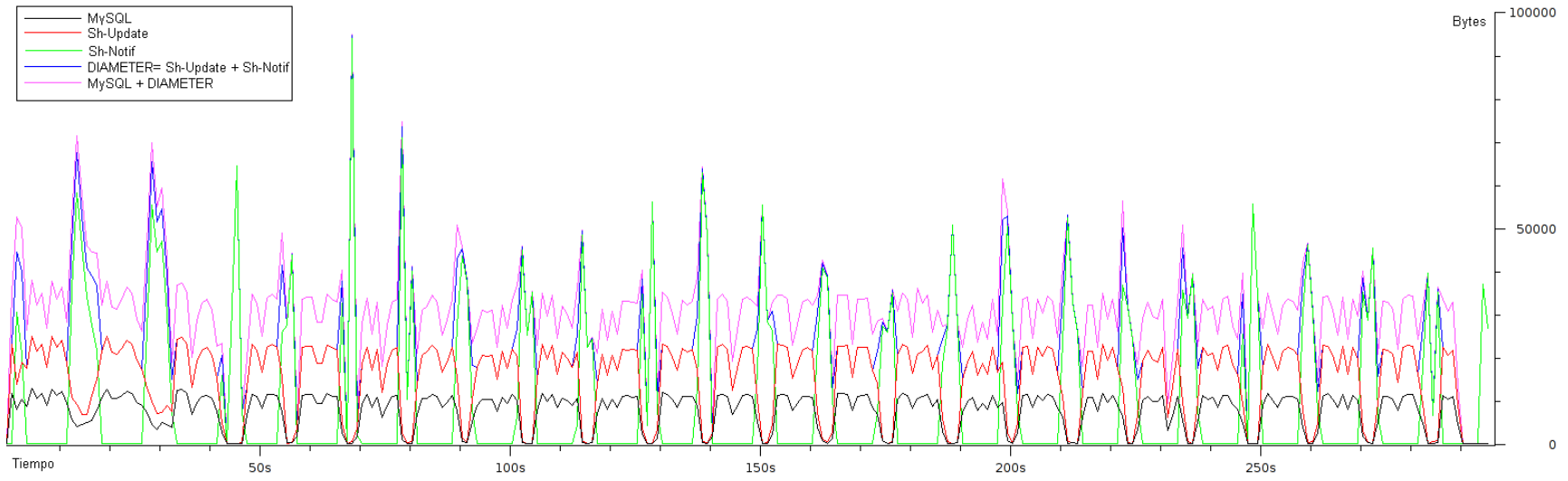

Figura 26. Escenario A Captura 6

- Captura 7
  - Número de Usuarios: 10
  - Retardo de Actualización: 125 ms.

Tiempo entre el primer y el último paquete (segs)	38,180
Paquetes	3150
Promedio Paquetes/seg	82,503
Promedio tamaño de paquetes (Bytes)	298,909
Bytes	941564
Promedio Bytes/seg	24660,920
Promedio Mbit/seg	0,197

Tabla 7. Resultados Escenario A Captura 7

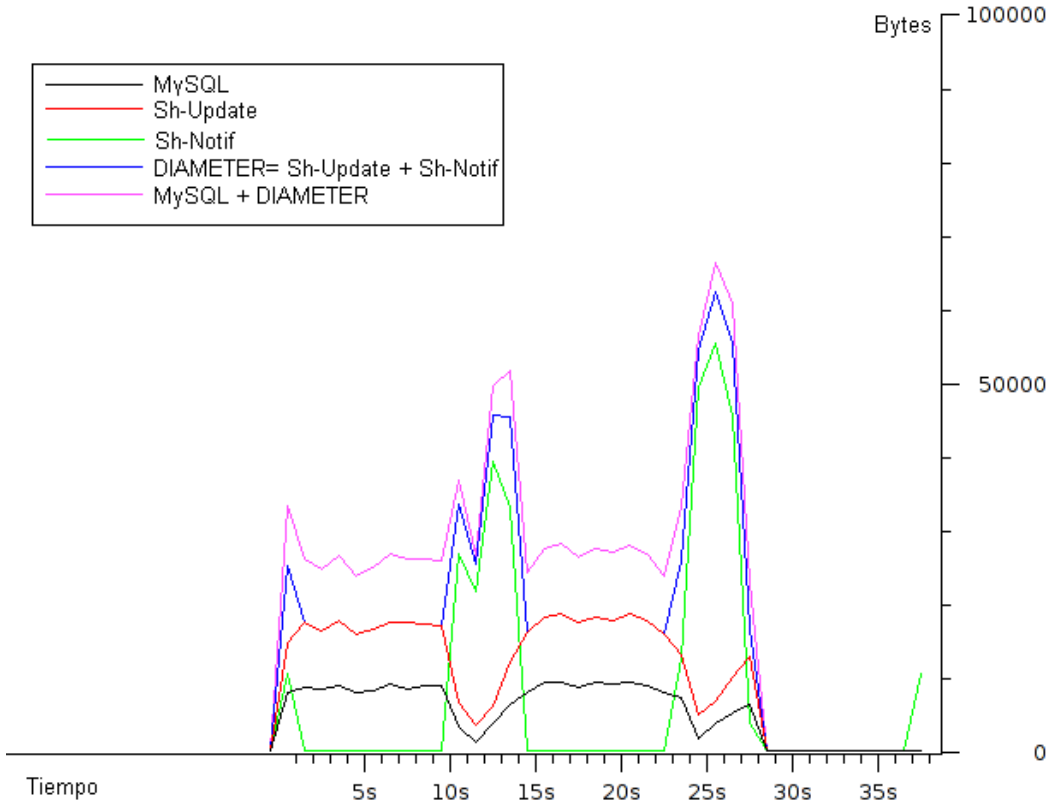

Figura 27. Escenario A Captura 7

- Captura 8
  - Número de Usuarios: 50
  - Retardo de Actualización: 125 ms.

Tiempo entre el primer y el último paquete (segs)	126,324
Paquetes	16274
Promedio Paquetes/seg	128,827
Promedio tamaño de paquetes (Bytes)	303,497
Bytes	4939109
Promedio Bytes/seg	39098,602
Promedio Mbit/seg	0,313

Tabla 8. Resultados Escenario A Captura 8

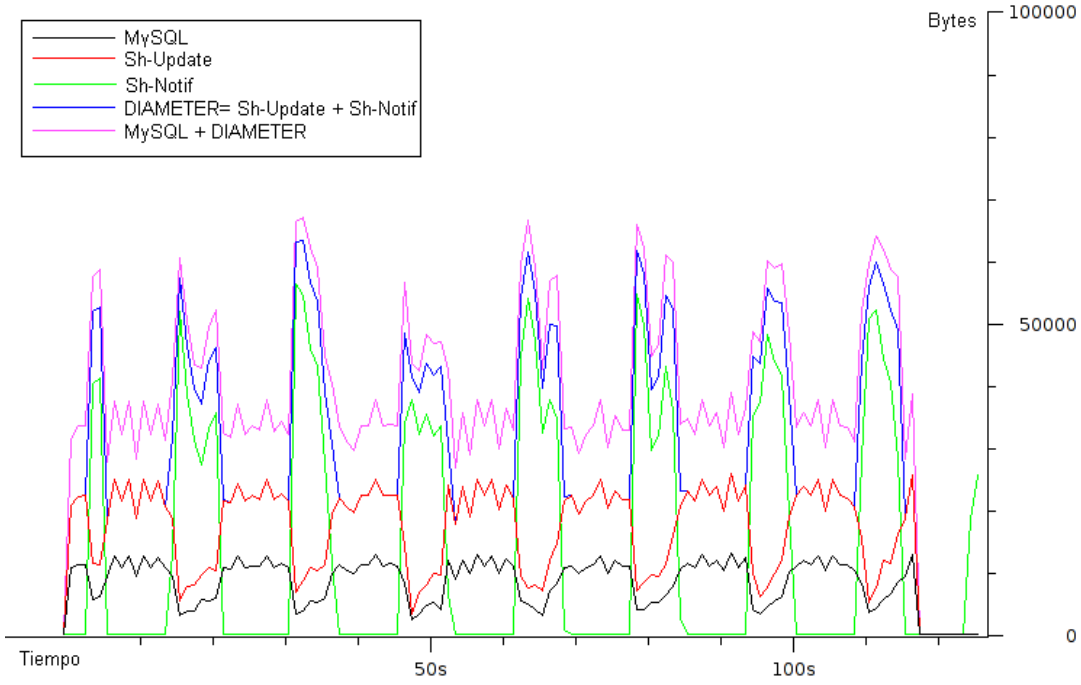

Figura 28. Escenario A Captura 8

- Captura 9
  - Número de Usuarios: 100
  - Retardo de Actualización: 125 ms.

Tiempo entre el primer y el último paquete (segs)	309,406
Paquetes	32079
Promedio Paquetes/seg	103,679
Promedio tamaño de paquetes (Bytes)	298,100
Bytes	9562750
Promedio Bytes/seg	30906,784
Promedio Mbit/seg	0,247

Tabla 9. Resultados Escenario A Captura 9

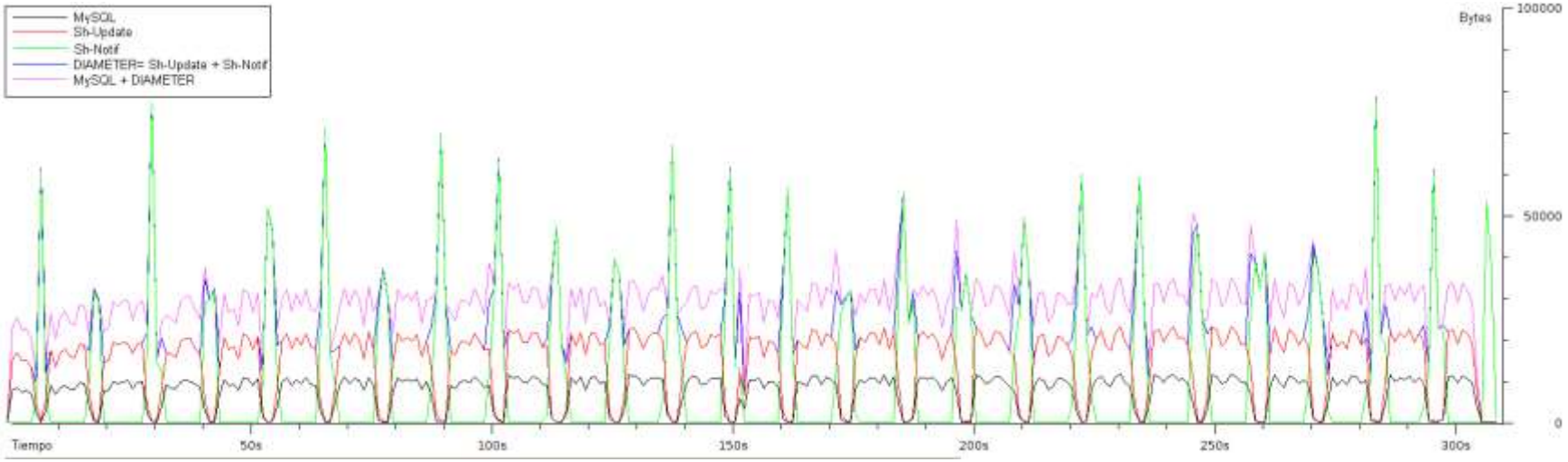

Figura 29. Escenario A Captura 9

5.2.3. Análisis de los resultados obtenidos

El análisis de los resultados obtenidos se enfocó en la observación del comportamiento de los dos mecanismos en cuanto al tiempo de respuesta y a la carga que se genera en la red, ya que como se expone en el Capítulo III, con la propuesta se pretende mejorar el desempeño de la red en la sincronización de información garantizado al mismo tiempo consistencia en la misma.

Al verificar los resultados obtenidos en las pruebas con el mecanismo de sincronización de información en IMS, se puede observar que el incremento en el número de Bytes no es totalmente lineal conforme el número de usuarios aumenta, lo cual indicaría que existe una pérdida de paquetes, hecho que contribuye a aumentar el overhead presente en la red.

Así mismo, al reducir el retardo de actualización, es decir al aumentar el número de transacciones por segundo, también se observar un incremento en el número de Bytes, por lo tanto también existe una pérdida de paquetes ocasionada por la carga sobre los componentes de red, en este caso sobre el FHoSS.

Por otra parte, para los casos en que el número de usuarios es de 10 y 50, el tiempo de respuesta se reduce a medida que se disminuye el retardo de actualización de 500 ms a 250 ms y posteriormente a 125 ms, ya que se intenta realizar las 50 actualizaciones en un menor espacio de tiempo y el FHoSS es capaz de procesar el número de peticiones que le están llegando. Sin embargo, cuando el número de usuarios es de 100, el tiempo de respuesta aumenta mientras el retardo de actualización disminuye indicando que el FHoSS no puede procesar rápidamente las peticiones que le llegan con este número de usuarios. Estos resultados se pueden apreciar en la Tabla 10.

		Retardo de Actualización		
		500 ms	**250 ms**	**125 ms**
No de Usuarios	**10**	61,838	40,923	38,18
	50	170,655	138,124	126,324
	100	282,132	296,771	309,406

Tabla 10. Comparación Tiempo de Respuesta en el Escenario A (resultados en segundos)

Se puede determinar un promedio del tiempo que tardan los diferentes componentes del mecanismo de sincronización en procesar un grupo de peticiones de actualización teniendo en cuenta el siguiente análisis.

El número de usuarios también indica el número de peticiones que se envían en un grupo, es decir 10, 50 o 100 peticiones en un grupo y cada grupo se repite 50 veces. Lo que también se varía es el retardo entre cada grupo de peticiones. Por ejemplo para el caso de un retardo de 250 ms y 50 usuarios, se estaría enviando 4 grupos de 50 peticiones por segundo.

Así, si se toma el tiempo total de respuesta del mecanismo y se divide entre 50, se tendría el tiempo promedio de respuesta para un grupo de peticiones incluido el retardo de actualización. Luego,

restando este retardo se tendría un promedio del tiempo del procesamiento de un grupo de peticiones de actualización. La ecuación es la siguiente:

$$PTP = \left(\frac{TTR}{50} \right) - RA$$

PTP= Promedio del Tiempo de Procesamiento
TTR= Tiempo Total de Respuesta
RA= Retardo de Actualización

Como ejemplo se selecciona la prueba con 50 usuarios y un retardo de actualización de 125 ms.

$$PTP = \left(\frac{126{,}324\text{s}}{50} \right) - 0{,}125s$$

$$PTP = 2{,}40148s$$

En este caso, el tiempo promedio que se toman los componentes del mecanismo de sincronización para atender un grupo de 50 peticiones cada 250 ms es de 2,40148 segundos, lo que indica claramente que las peticiones se están represando ocasionando un retardo considerable en la actualización de la información en las bases de datos que lo requieren.

El tiempo promedio de procesamiento para todas las capturas en el escenario A se condensan en la Tabla 11.

		Retardo de Actualización		
		500 ms	**250 ms**	**125 ms**
No de Usuarios	**10**	0,73676	0,56846	0,6386
	50	2,9131	2,51248	2,40148
	100	5,14264	5,68542	6,06312

Tabla 11. Promedio del Tiempo de Procesamiento de las Peticiones de Actualización en el Escenario A (resultados en segundos)

Observando los valores de la Tabla 11, se puede inferir que en ninguno de los casos analizados se pueden completar las operaciones antes de que llegue otro grupo de peticiones de actualización, ocasionando retardos y posibles inconsistencias en la información.

En las gráficas del tráfico capturado en el Escenario A se aprecia que el proceso Sh-Update se hace casi en forma constante, lo que quiere decir que las peticiones de actualización de información desde el WLSS GW hasta el FHoSS se llevan a cabo una detrás de otra.

Sin embargo, el proceso Sh-Notif que actualiza la información desde el FHoSS al WLSS se lleva a cabo aproximadamente cada 11 segundos aunque la información ya esté lista para replicarse. Esto genera un aumento del tráfico en la red en forma de picos que no superan los tres segundos y en donde se envía toda la información actualizada al WLSS.

Por lo tanto, aunque un grupo de peticiones de actualización tarde en procesarse 2.5 segundos, la información en el WLSS no va a estar disponible si no después de que se cumpla el período de 11 segundos que los desarrolladores del FHoSS han determinado para repetir el proceso Sh-Notif. Aunque este período se lo puede configurar, los mismos desarrolladores recomiendan no disminuirlo con el objetivo de evitar sobrecargar aún más el FHoSS.

Pasando a un análisis más puntual, para el caso del servicio simulado de localización que se ha seleccionado, se puede plantear un ejercicio imaginando un usuario en un automóvil que se mueve a 80 Km/h, es decir que este usuario se estaría moviendo 11,11 metros en 500 ms. Así, si las actualizaciones de las coordenadas de localización se realizaran cada 500 ms, el usuario se encontraría desplazado 11,11 metros de su ubicación anterior.

El rango de exactitud de los sistemas *Global Positioning System* (GPS) es de entre 10 y 15 metros, por lo tanto si se actualiza la información cada 500 ms para un usuario que se mueve a 80 Km/h, se estaría dentro de ese margen.

Entonces, observando los resultados obtenidos se aprecia que los retardos introducidos por el mecanismo en cuestión es mayor a los 500 ms, lo cual aumenta el error en el rango de exactitud de GPS, ya que en un momento dado en las bases de datos se dispondría de las coordenadas que el usuario tuvo hace algunos segundos o fracciones de segundo dependiendo de la carga de la red.

5.3. ESCENARIO B

5.3.1. Descripción

En el Escenario B, se realizaron las pruebas con el mecanismo de sincronización propuesto por este proyecto usando una plataforma middleware. Para tal motivo, se usó dos WLSS y el FHoSS, pero con la interfaz Sh deshabilitada. Así mismo, se usó la AGC y el Controlador Sequoia como plataforma *middleware*.

Al igual que en el Escenario A, uno de los WLSS es el prestador del servicio, el cual hace uso de la información generada por la AGC, la cual debe escribirse tanto en su propia base de datos, como también en la base de datos del FHoSS.

El otro WLSS se lo usó como una Gateway entre la AGC y el Controlador Sequoia, convirtiendo las peticiones HTTP de la primera en mensajes entendibles por el segundo haciendo uso del *driver* Sequoia.

En la Figura 30 se puede apreciar la arquitectura para este escenario de pruebas.

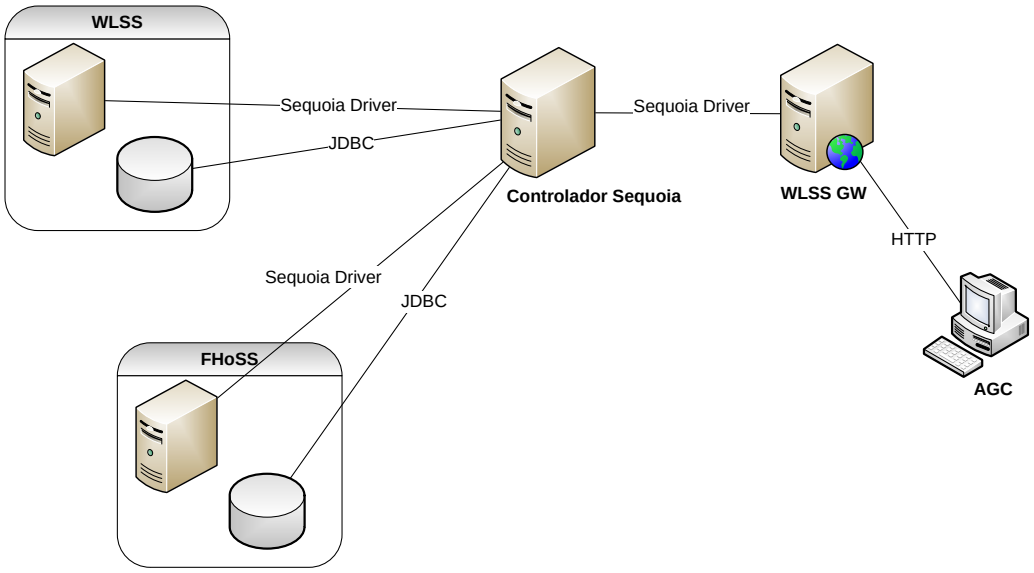

Figura 30. Arquitectura para el Escenario B

La mayoría de los mensajes que se capturaron en esta prueba fueron MySQL, correspondientes al tráfico que se genera entre el Recovery Log y el Controlador Sequoia y al tráfico entre este último y los *Backends*. También se capturan los mensajes entre el WLSS GW y el Controlador Sequoia.

El flujo de información para una actualización se puede apreciar en la Figura 31.

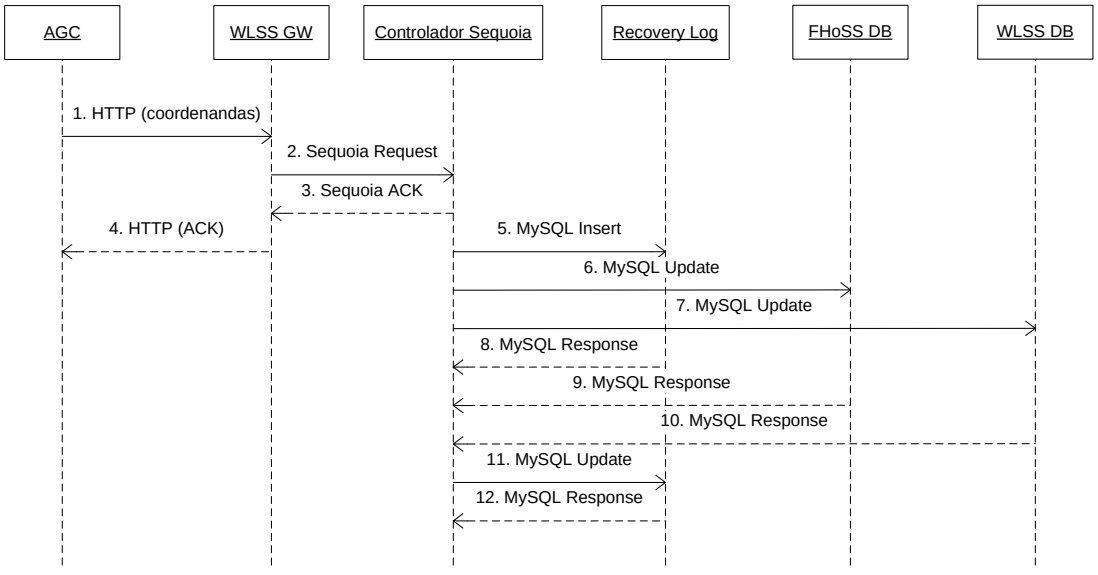

Figura 31. Flujo de Información en el Escenario B

1. La AGC envía un mensaje HTTP con la información generada hacia el Gateway WLSS (WLSS GW).
2. El WLSS GW envía a través del *driver* Sequoia al Controlador Sequoia la petición.
3. El Controlador Sequoia envía un ACK al WLSS GW.
4. El WLSS GW envía un ACK HTTP a la AGC para que ésta envíe otra petición después del retardo establecido.
5. El Controlador Sequoia envía un mensaje MySQL *Insert* al Recovery Log con la información que se va a actualizar para que sea un respaldo de los Backends.
6. El Controlador Sequoia envía un MySQL *Update* a la base de datos del FHoSS para que actualice la información.
7. El Controlador Sequoia envía un MySQL *Update* a la base de datos del WLSS para que actualice la información.
8. El Recovery Log responde al comando MySQL *Insert*.
9. La base de datos del FHoSS responde al comando MySQL *Update*.
10. La base de datos del WLSS responde al comando MySQL *Update*.
11. Una vez llega la respuesta del MySQL *Update* de alguno de los *Backends*, el Controlador actualiza permanentemente la información del Recovery Log con un MySQL Update.
12. El Recovery Log responde al comando MySQL Update.

5.3.2. Resultados obtenidos

- Captura 1

  - Número de Usuarios: 10
  - Retardo de Actualización: 500 ms.

Tiempo entre el primer y el último paquete (segs)	28,201
Paquetes	6637
Promedio Paquetes/seg	235,344
Promedio tamaño de paquetes (Bytes)	122,318
Bytes	811826
Promedio Bytes/seg	28786,891
Promedio Mbit/seg	0,230

Tabla 12. Resultados Escenario B Captura 1

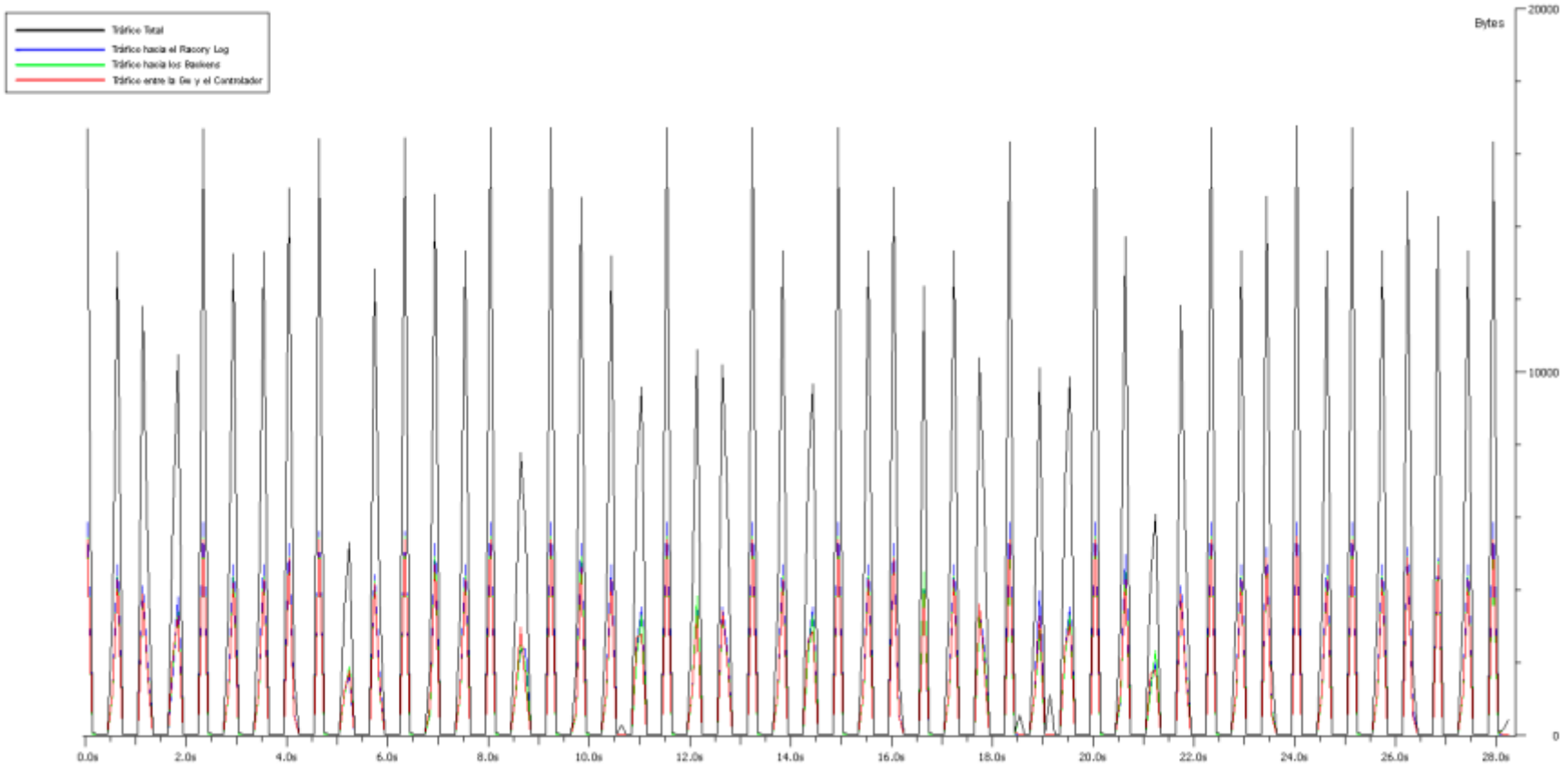

Figura 32. Escenario B Captura 1

- Captura 2
  - Número de Usuarios: 50
  - Retardo de Actualización: 500 ms.

Tiempo entre el primer y el último paquete (segs)	34,100
Paquetes	31230
Promedio Paquetes/seg	915,842
Promedio tamaño de paquetes (Bytes)	125,139
Bytes	3908105
Promedio Bytes/seg	114607,965
Promedio Mbit/seg	0,917

Tabla 13. Resultados Escenario B Captura 2

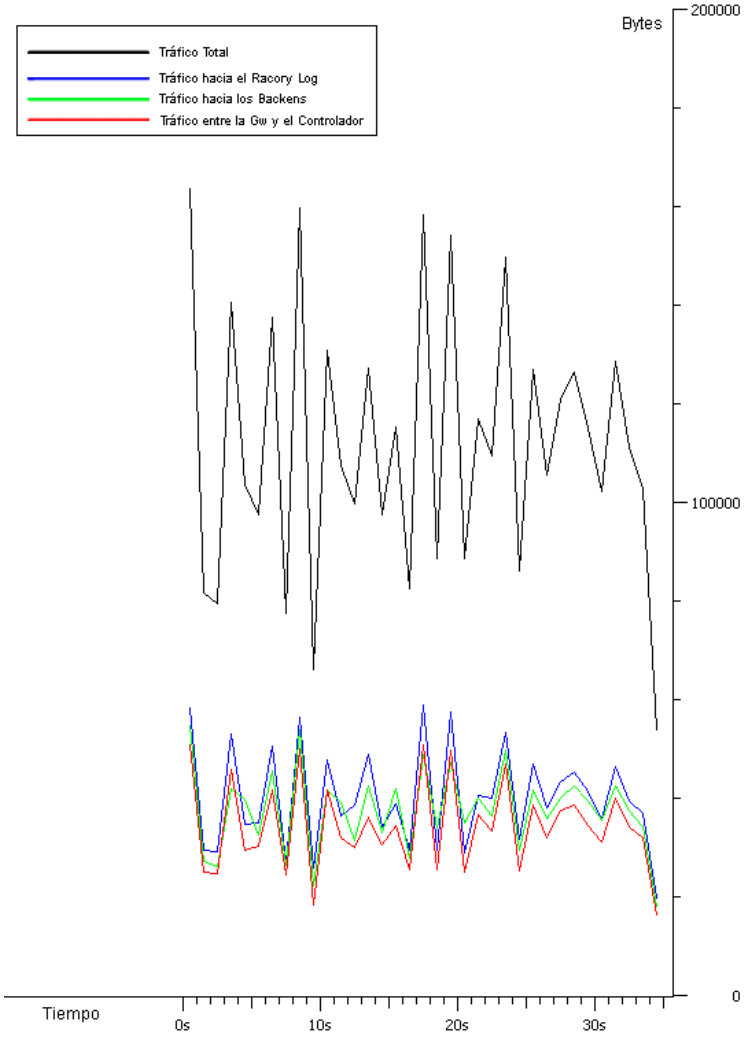

Figura 33. Escenario B Captura 2

- Captura 3
  - Número de Usuarios: 100
  - Retardo de Actualización: 500 ms.

Tiempo entre el primer y el último paquete (segs)	45,432
Paquetes	62493
Promedio Paquetes/seg	1375,520
Promedio tamaño de paquetes (Bytes)	125,273
Bytes	7828664
Promedio Bytes/seg	172315,061
Promedio Mbit/seg	1,379

Tabla 14. Resultados Escenario B Captura 3

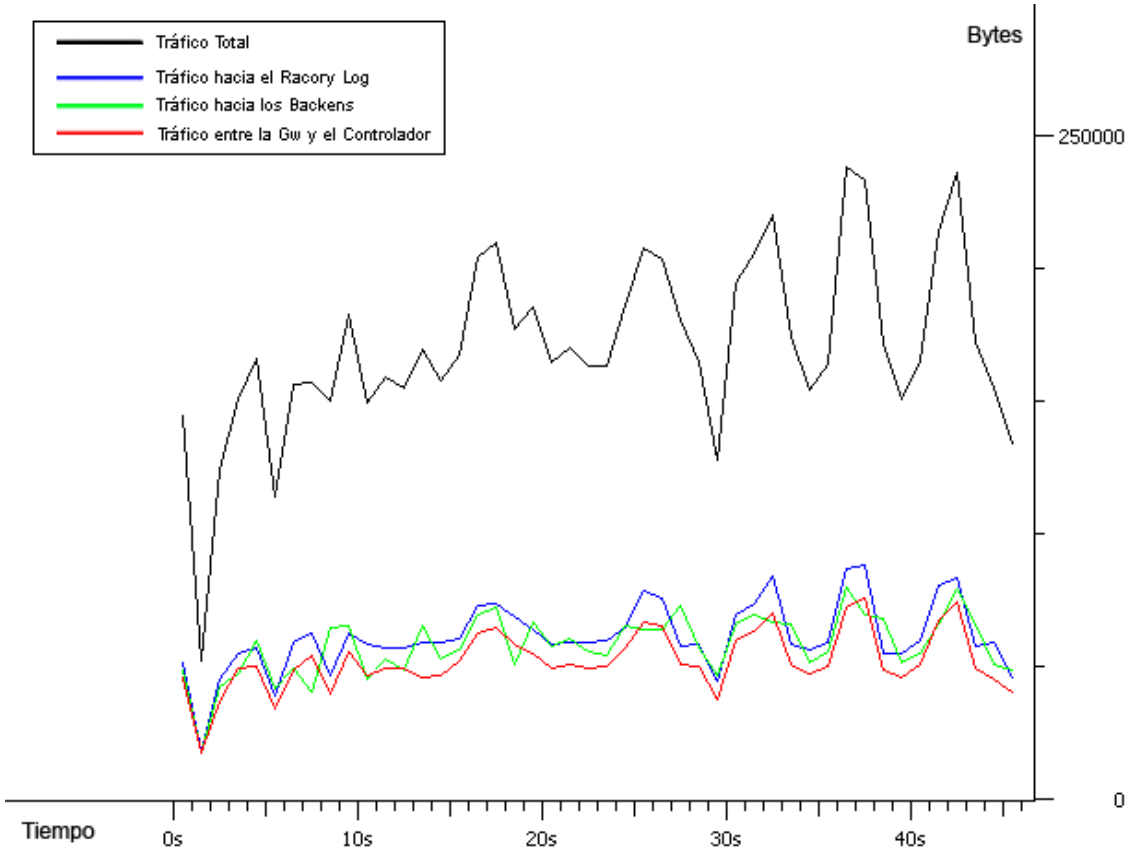

Figura 34. Escenario B Captura 3

- Captura 4
  - Número de Usuarios: 10
  - Retardo de Actualización: 250 ms.

Tiempo entre el primer y el último paquete (segs)	15,652
Paquetes	6564
Promedio Paquetes/seg	419,373
Promedio tamaño de paquetes (Bytes)	122,601
Bytes	804751
Promedio Bytes/seg	51415,446
Promedio Mbit/seg	0,411

Tabla 15. Resultados Escenario B Captura 4

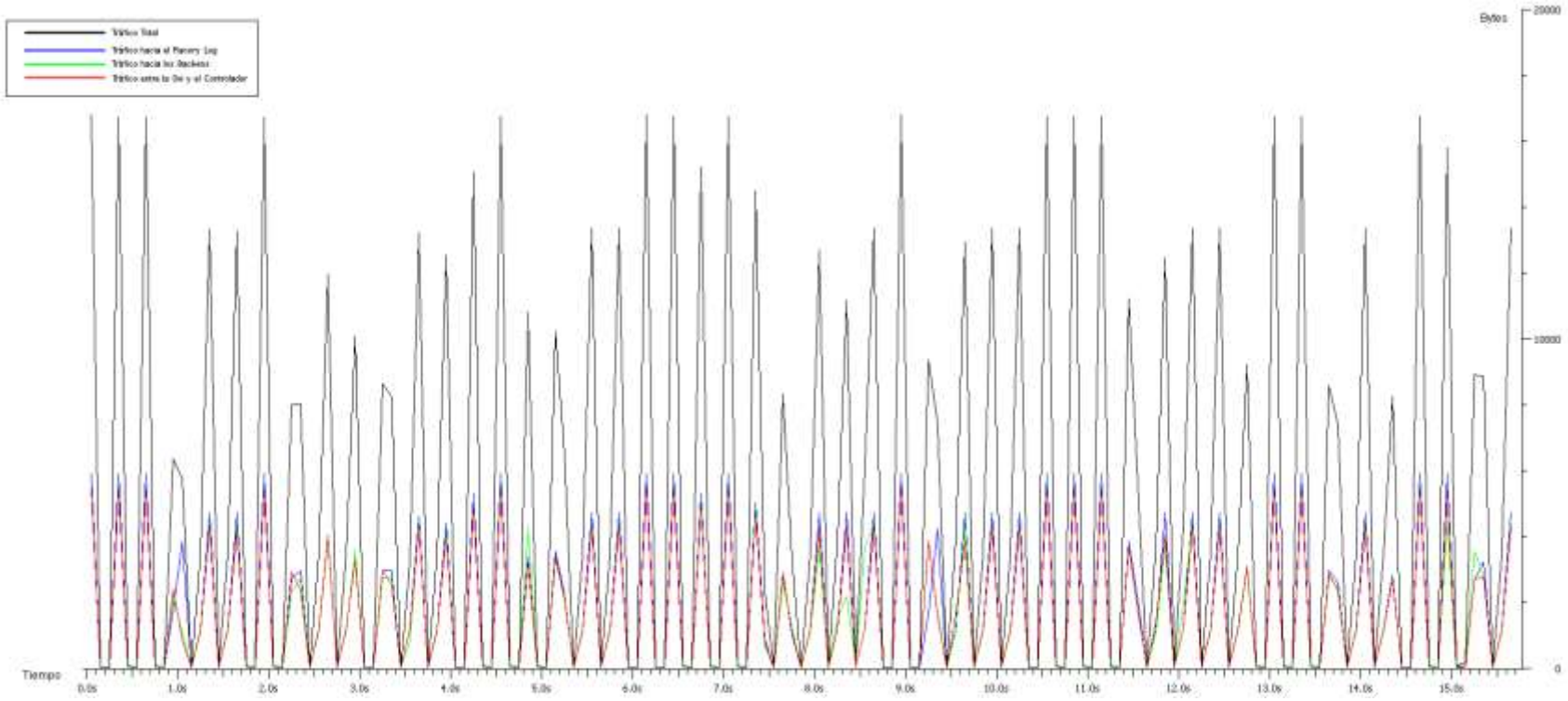

Figura 35. Escenario B Captura 4

- Captura 5
  - Número de Usuarios: 50
  - Retardo de Actualización: 250 ms.

Tiempo entre el primer y el último paquete (segs)	20,660
Paquetes	31607
Promedio Paquetes/seg	1529,836
Promedio tamaño de paquetes (Bytes)	125,173
Bytes	3956357
Promedio Bytes/seg	191494,899
Promedio Mbit/seg	1,532

Tabla 16. Resultados Escenario B Captura 5

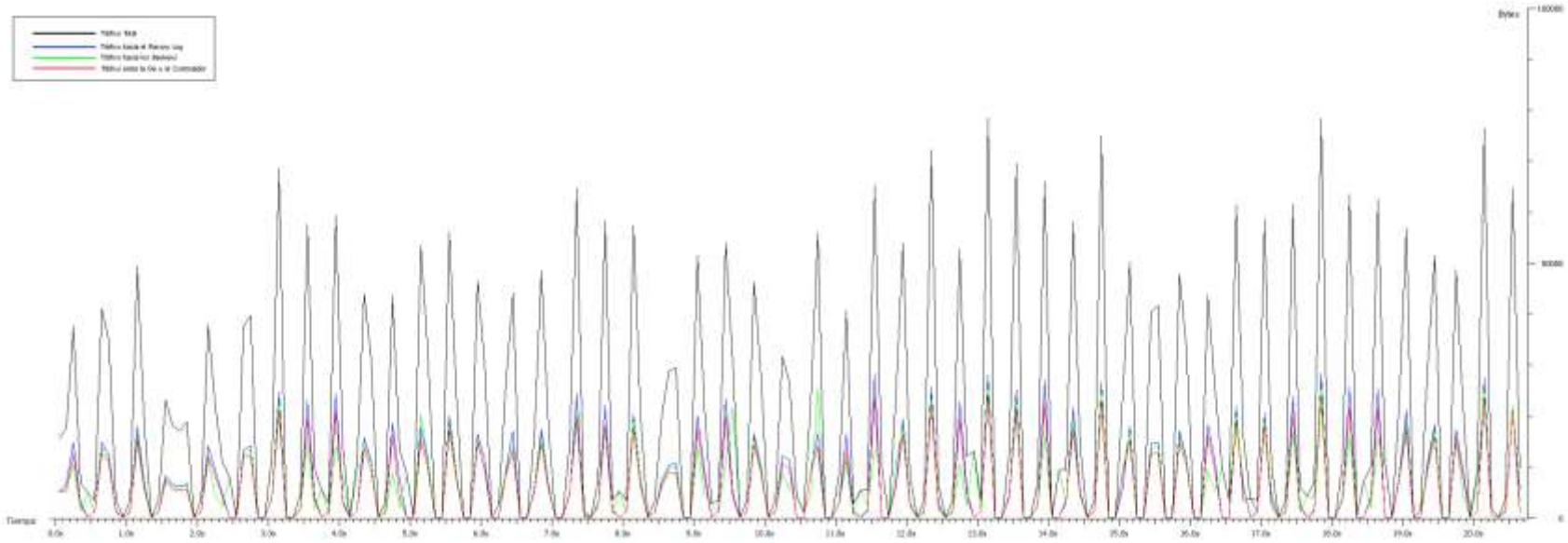

Figura 36. Escenario B Captura 5

- Captura 6
  - Número de Usuarios: 100
  - Retardo de Actualización: 250 ms.

Tiempo entre el primer y el último paquete (segs)	27,626
Paquetes	60962
Promedio Paquetes/seg	2206,690
Promedio tamaño de paquetes (Bytes)	125,579
Bytes	7655523
Promedio Bytes/seg	277113,103
Promedio Mbit/seg	2,217

Tabla 17. Resultados Escenario B Captura 6

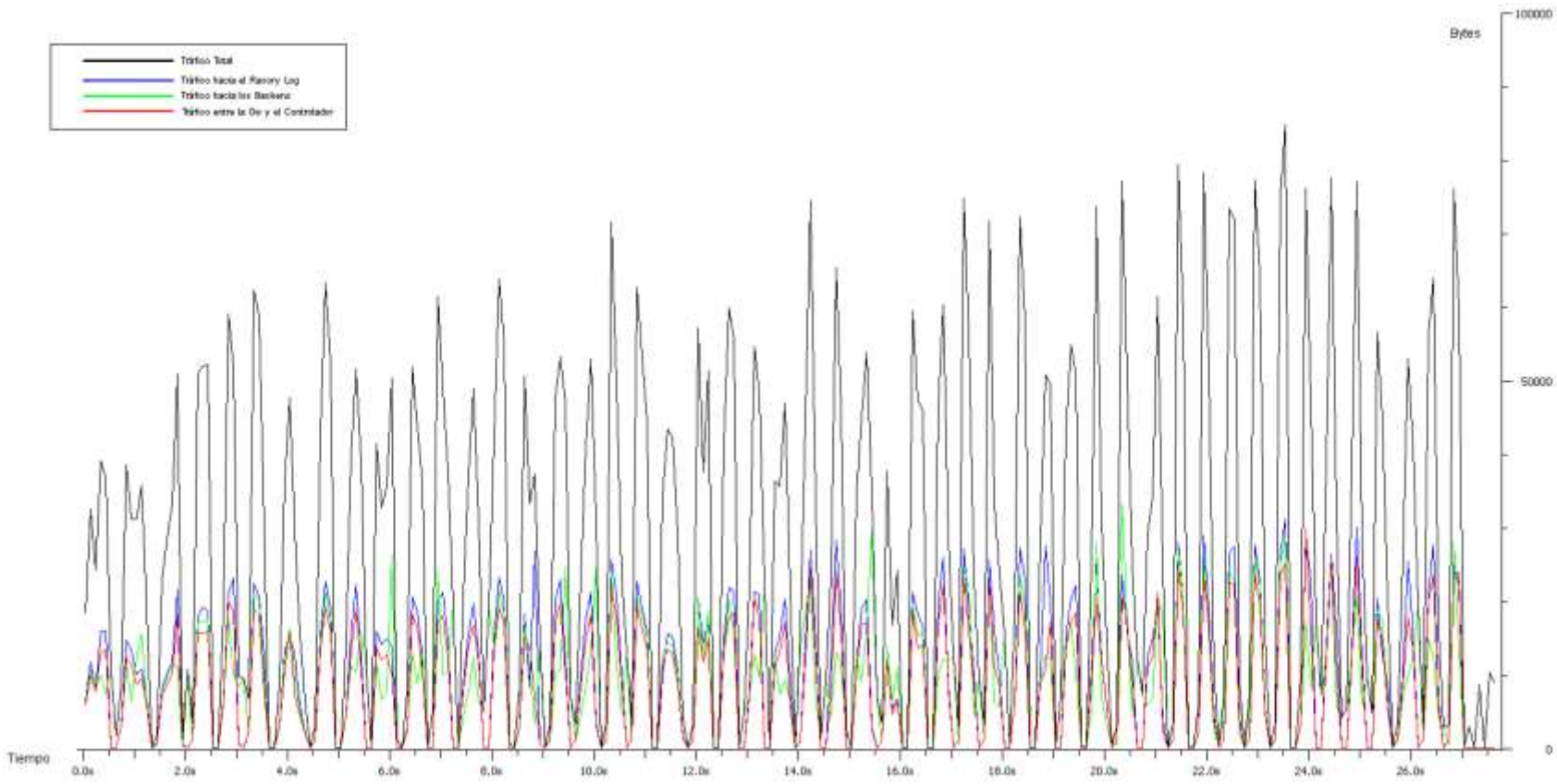

Figura 37. Escenario B Captura 6

- Captura 7
  - Número de Usuarios: 10
  - Retardo de Actualización: 125 ms.

Tiempo entre el primer y el último paquete (segs)	7,722
Paquetes	6397
Promedio Paquetes/seg	828,431
Promedio tamaño de paquetes (Bytes)	124,218
Bytes	794622
Promedio Bytes/seg	102905,957
Promedio Mbit/seg	0,823

Tabla 18. Resultados Escenario B Captura 7

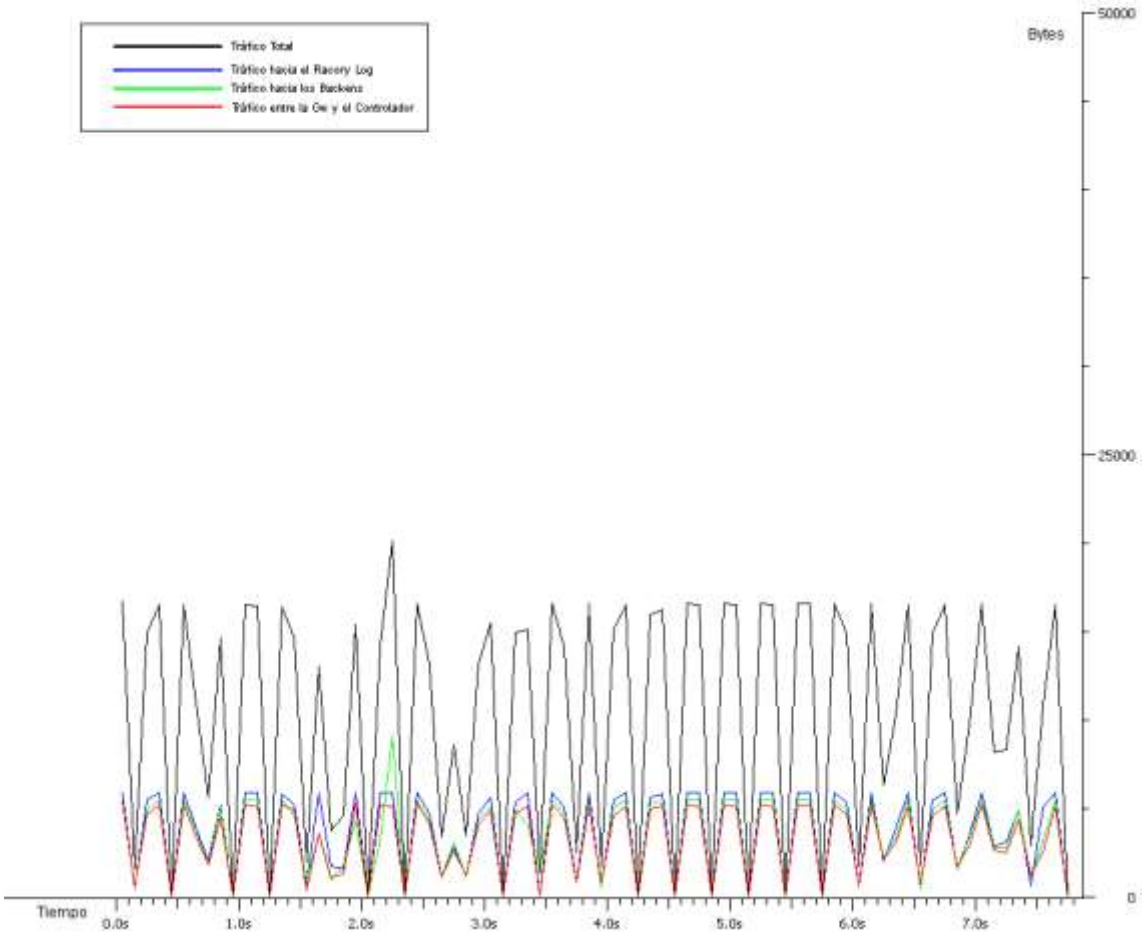

Figura 38. Escenario B Captura 7

- Captura 8
  - Número de Usuarios: 50
  - Retardo de Actualización: 125 ms.

Tiempo entre el primer y el último paquete (segs)	12,367
Paquetes	30996
Promedio Paquetes/seg	2506,264
Promedio tamaño de paquetes (Bytes)	125,579
Bytes	3892462
Promedio Bytes/seg	314735,320
Promedio Mbit/seg	2,518

Tabla 19. Resultados Escenario B Captura 8

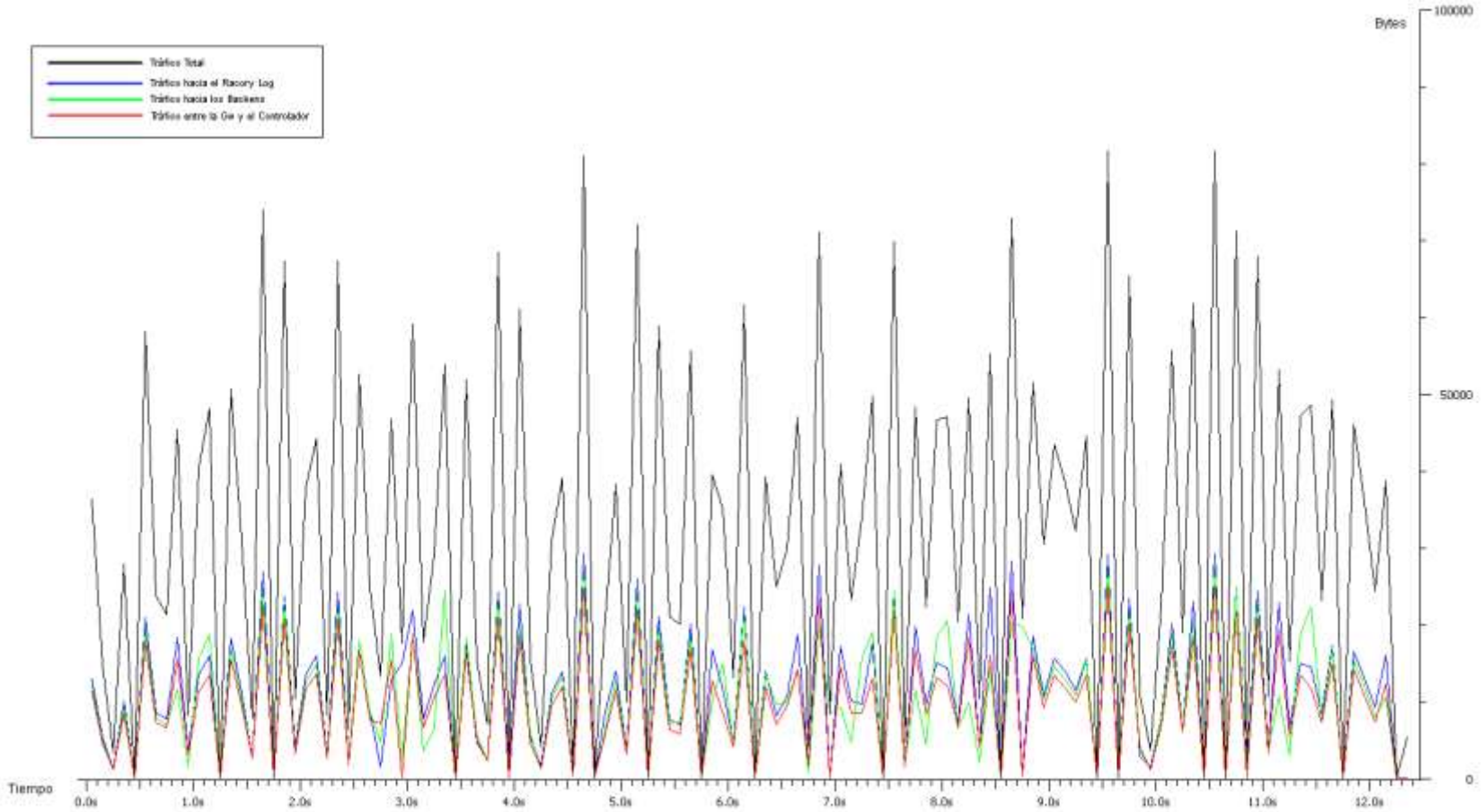

Figura 39. Escenario B Captura 8

- Captura 9
  - Número de Usuarios: 100
  - Retardo de Actualización: 125 ms.

Tiempo entre el primer y el último paquete (segs)	18,501
Paquetes	60775
Promedio Paquetes/seg	3284,975
Promedio tamaño de paquetes (Bytes)	125,747
Bytes	7642291
Promedio Bytes/seg	413076,715
Promedio Mbit/seg	3,305

Tabla 20. Resultados Escenario B Captura 9

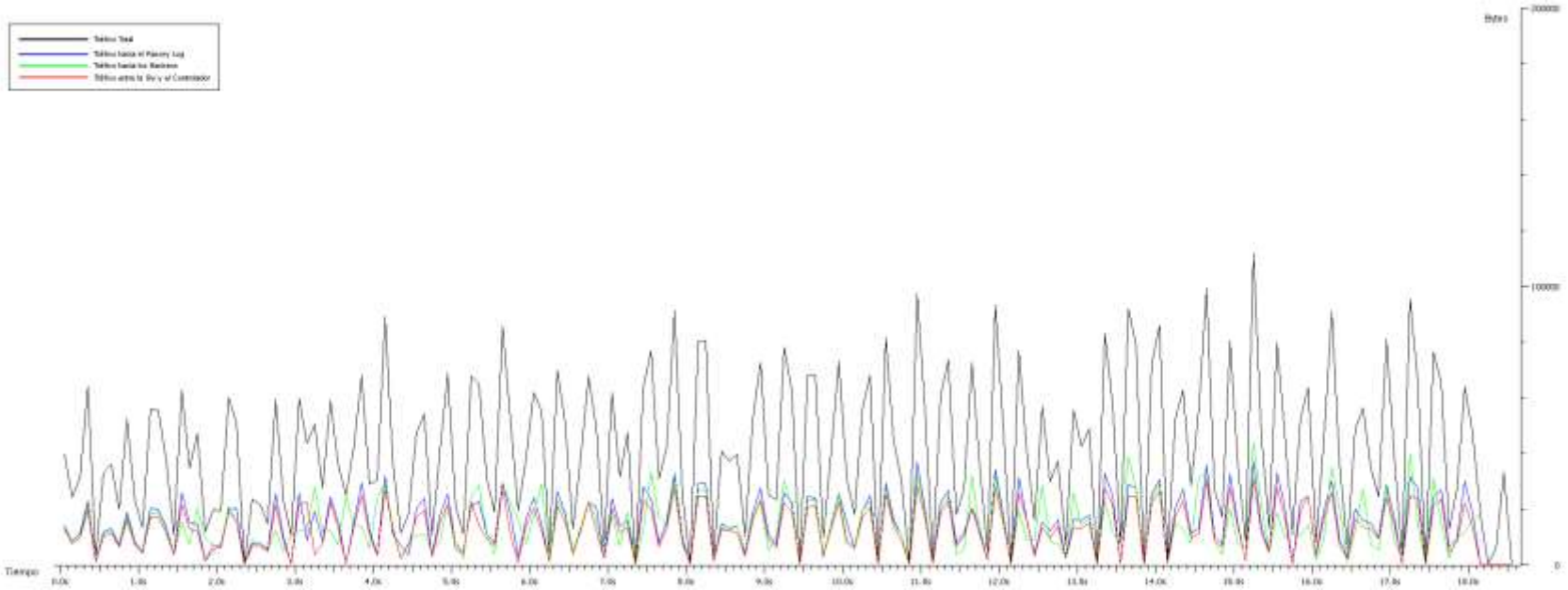

Figura 40. Escenario B Captura 9

5.3.3. Análisis de los resultados obtenidos

Con el mecanismo de sincronización de información propuesto se observa que el número de Bytes se mantiene casi constante, mientras el retardo de actualización varía, lo que indica que los componentes de red procesan de forma adecuada las peticiones que se están generando. Estos resultados se los puede apreciar en la Figura 41.

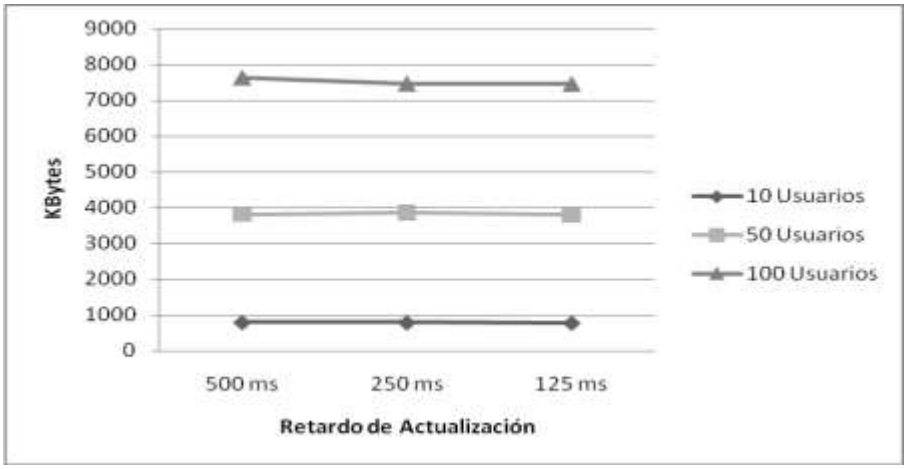

Figura 41. Comparación del Número de KBytes variando el Retardo de Actualización y con el Número de Usuarios Constante en el Escenario B.

También, el crecimiento en el número de Bytes conforme el número de usuarios aumenta tiene una tendencia casi lineal, mostrando un leve aumento en *overhead* por la regeneración de paquetes perdidos.

A medida que el número de peticiones por segundo aumenta, es decir que el retardo de actualización disminuye y el número de usuarios crece, se aprecia una disminución en el tiempo total de respuesta, lo que quiere decir que el mecanismo atiende eficientemente todas las peticiones de actualización hasta en el caso más crítico que se ha probado. La comparación de los resultados se encuentra en la Tabla 21.

		Retardo de Actualización		
		500 ms	**250 ms**	**125 ms**
No de Usuarios	**10**	28,201	15,652	7,722
	50	34,1	20,66	12,367
	100	45,432	27,626	18,501

Tabla 21. Comparación Tiempo de Respuesta en el Escenario B (resultados en segundos)

En la Tabla 22 se compilan los valores del tiempo promedio de procesamiento de un grupo de peticiones de actualización en el Escenario B. Se observa que los valores se encuentran en casi todos los casos por debajo del valor de retardo de actualización, lo que quiere decir que los

componentes de mecanismo alcanzan a procesar el grupo de peticiones de actualización antes que se genere otro, con lo se garantiza la sincronización oportuna y consistente de la información.

		Retardo de Actualización		
		500 ms	**250 ms**	**125 ms**
No de Usuarios	**10**	0,06402	0,06304	0,02944
	50	0,182	0,1632	0,12234
	100	0,40864	0,30252	0,24502

Tabla 22. Promedio del Tiempo de Procesamiento de las Peticiones de Actualización en el Escenario B (resultados en segundos)

La excepción se presenta en los casos de 100 usuarios con 250 ms y 125 ms de retardo de actualización. Aquí, el tiempo de procesamiento del grupo de peticiones es mayor al valor del retardo, por lo que la información presente en las bases de datos va a corresponder a una versión inmediatamente anterior a la que debería estar presente.

En algunas gráficas de las capturas es posible apreciar el comportamiento para cada grupo de peticiones de actualización. Aquellos picos más pronunciados corresponden a las actualizaciones que tienen la mejor respuesta en el tiempo, aunque generan mayor tráfico en la red en ese momento.

También se puede ver que el tráfico entre el WLSS GW y el Controlador, el tráfico hacia el Recovery Log y el tráfico hacia los *Backends* tienen un comportamiento idéntico cuando se presentan las mejores respuestas en el tiempo. Sin embargo, cuando por alguna razón alguno de estos tráficos cambia de comportamiento se deteriora la respuesta en general.

Así, en las capturas con 100 usuarios y período de actualización de 250 ms y 125 ms se puede ver que en ciertos momentos el procesamiento de los mensajes no es el mejor, generando retrasos en las siguientes actualizaciones.

De todas formas para el caso del servicio simulado de localización, con un usuario que se mueve en un automóvil a 80 Km/h se puede asegurar que las coordenadas que se tienen en las bases de datos del WLSS y del FHoSS corresponden a la ubicación actual ya que el mayor retardo que se introduce es de 0,40864 segundo correspondiente a 9,08 metros, que es una distancia menor a la resolución del GPS. Por lo tanto aunque el retardo esté presente, éste no va a ser relevante dado que una variación en la posición menor a 10 metros no es detectable por el GPS.

5.4. ANÁLISIS COMPARATIVO ENTRE LOS DOS ESCENARIOS

Para complementar la validación del prototipo del mecanismo de sincronización propuesto, se consideró pertinente realizar una comparación con el mecanismo de sincronización de IMS usando la interfaz Sh, con el propósito de verificar el comportamiento de la propuesta frente a una implementación que ya se encuentra definida para una arquitectura de las NGN.

Para cada uno de los retardos de actualización se muestran los resultados obtenidos del número de Bytes total y tiempo de respuesta con su respectiva comparación.

No de Usuarios	Open IMS Core Sh		Sequoia		Bytes Sh/Seq (veces)	Tiempo Sh/Seq (veces)
	KBytes	Tiempo (seg)	Kbytes	Tiempo (seg)		
10	798,189	61,838	792,799	28,201	1,0067	2,192
50	4534,148	170,655	3816,509	34,1	1,188	5,004
100	9381,037	282,132	7645,179	45,432	1,227	6,209

Tabla 23. Comparación de desempeño entre los mecanismos con un período de actualización de 500 ms

No de Usuarios	Open IMS Core Sh		Sequoia		Bytes Sh/Seq (veces)	Tiempo Sh/Seq (veces)
	Kbytes	Tiempo (seg)	Kbytes	Tiempo (seg)		
10	839,943	40,923	785,889	15,652	1,0689	2,614
50	4701	138,124	3863,629	20,66	1,217	6,686
100	9272,071	296,771	7476,096	27,626	1,240	10,742

Tabla 24. Comparación de desempeño entre los mecanismos con un período de actualización de 250 ms

No de Usuarios	Open IMS Core Sh		Sequoia		Bytes Sh/Seq (veces)	Tiempo Sh/Seq (veces)
	Kbytes	Tiempo (seg)	Kbytes	Tiempo (seg)		
10	919,496	38,18	775,998	7,722	1,184	4,944
50	4823,348	126,324	3801,232	12,367	1,269	10,214
100	9338,623	309,406	7463,174	18,501	1,251	16,723

Tabla 25. Comparación de desempeño entre los mecanismos con un período de actualización de 125 ms

Con los valores presentados en las tres tablas anteriores se puede confirmar un mejor comportamiento del mecanismo propuesto respecto al mecanismo propio de IMS, mejorando su respuesta en el tiempo en hasta 16 veces para el caso de mayor exigencia y siempre estando por debajo en número de Bytes generados.

Así, entre mayor es la carga para los componentes, mayor es la diferencia del desempeño de la propuesta frente al mecanismo de IMS corroborando las ventajas de las arquitecturas *middleware* en entornos donde se requiere de una alta escalabilidad.

A pesar que estas arquitecturas *middleware* introducen *overhead*, se observa que la propuesta presentada genera aproximadamente un 20% menos de Bytes que el mecanismo de IMS, mientras se ofrece una considerable mejora en la respuesta en el tiempo y se garantiza la consistencia de la información.

En las siguientes gráficas comparativas se muestra la magnitud de la diferencia de rendimiento entre los mecanismo probados.

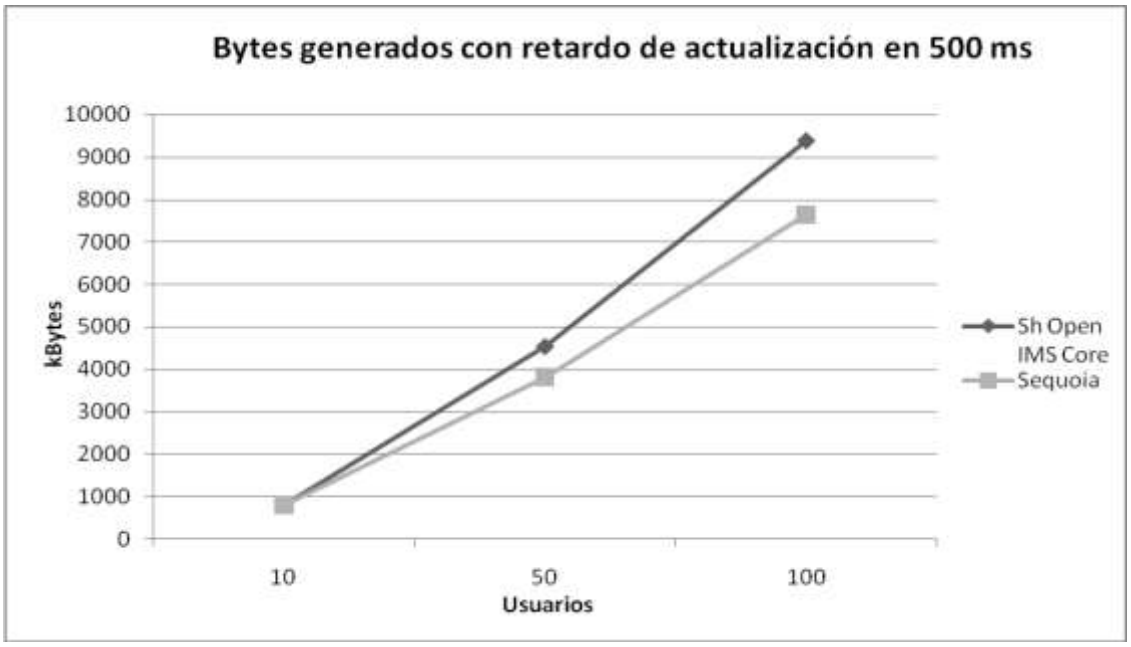

Figura 42. Comparación del número de KBytes generados con un retardo de actualización de 500 ms

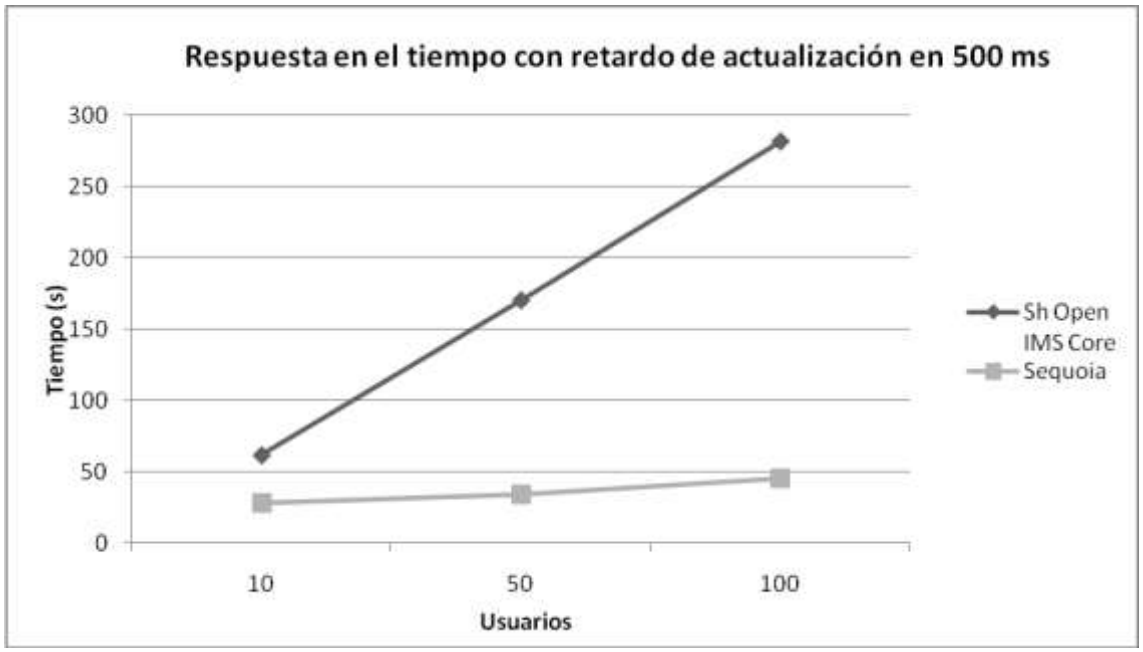

Figura 43. Comparación de la respuesta en el tiempo con un retardo de actualización de 500 ms

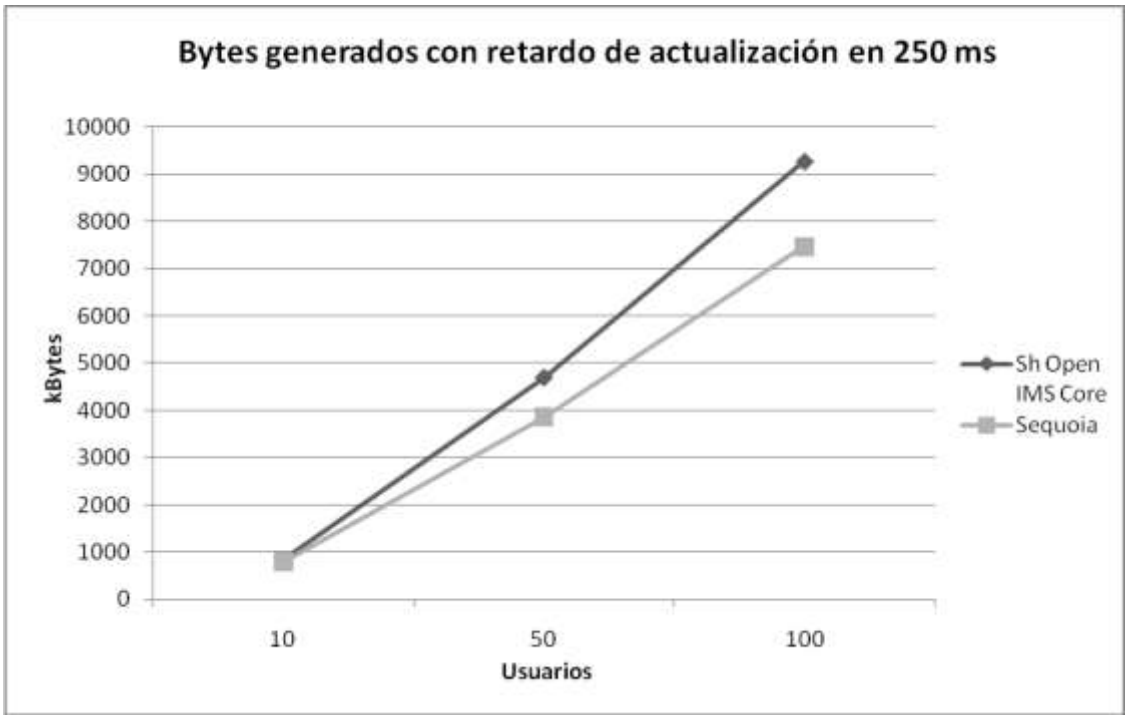

Figura 44. Comparación del número de KBytes generados con un retardo de actualización de 250 ms

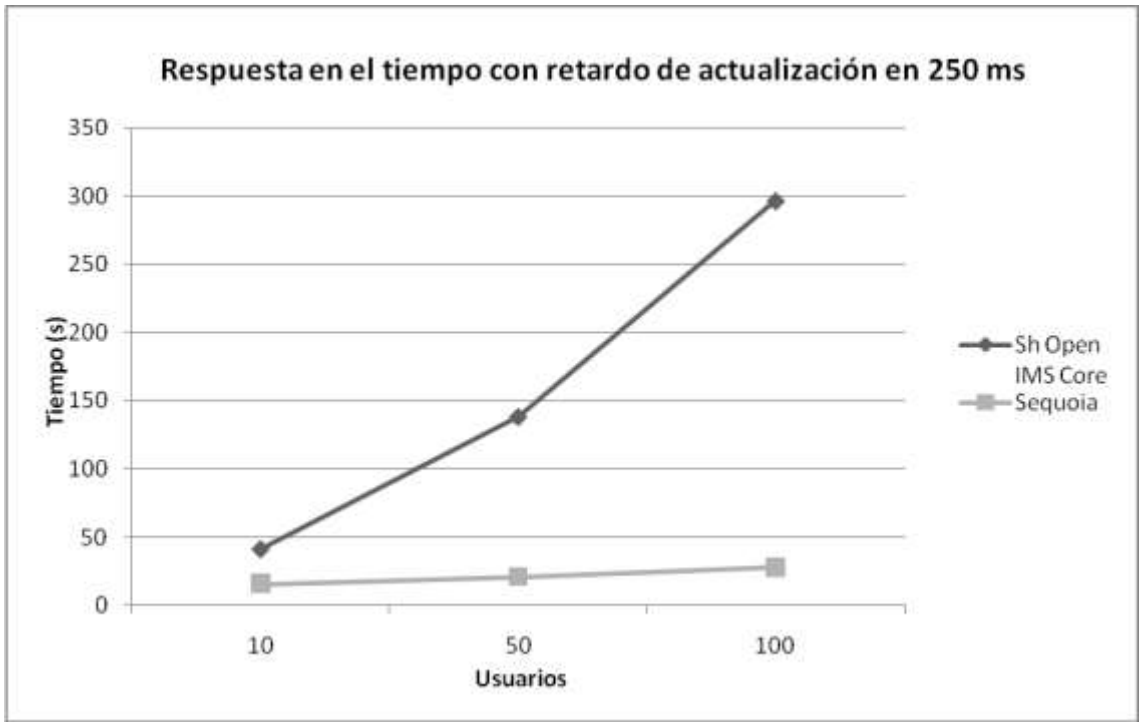

Figura 45. Comparación de la respuesta en el tiempo con un retardo de actualización de 250 ms

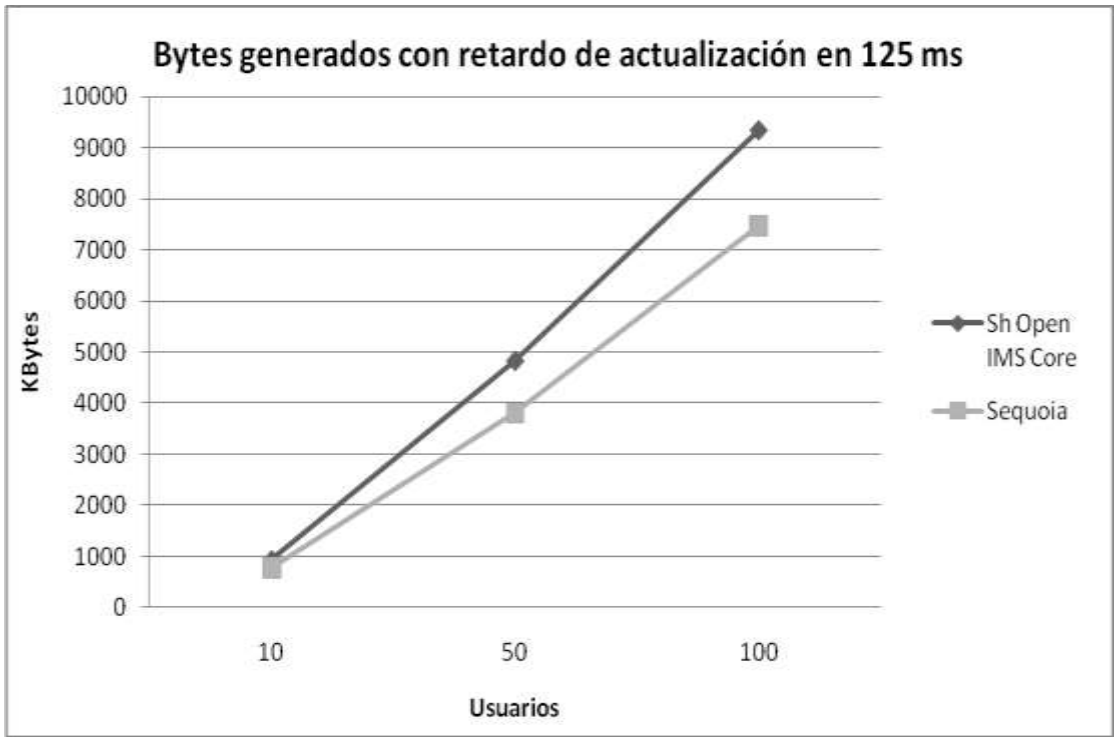

Figura 46. Comparación del número de KBytes generados con un retardo de actualización de 125 ms

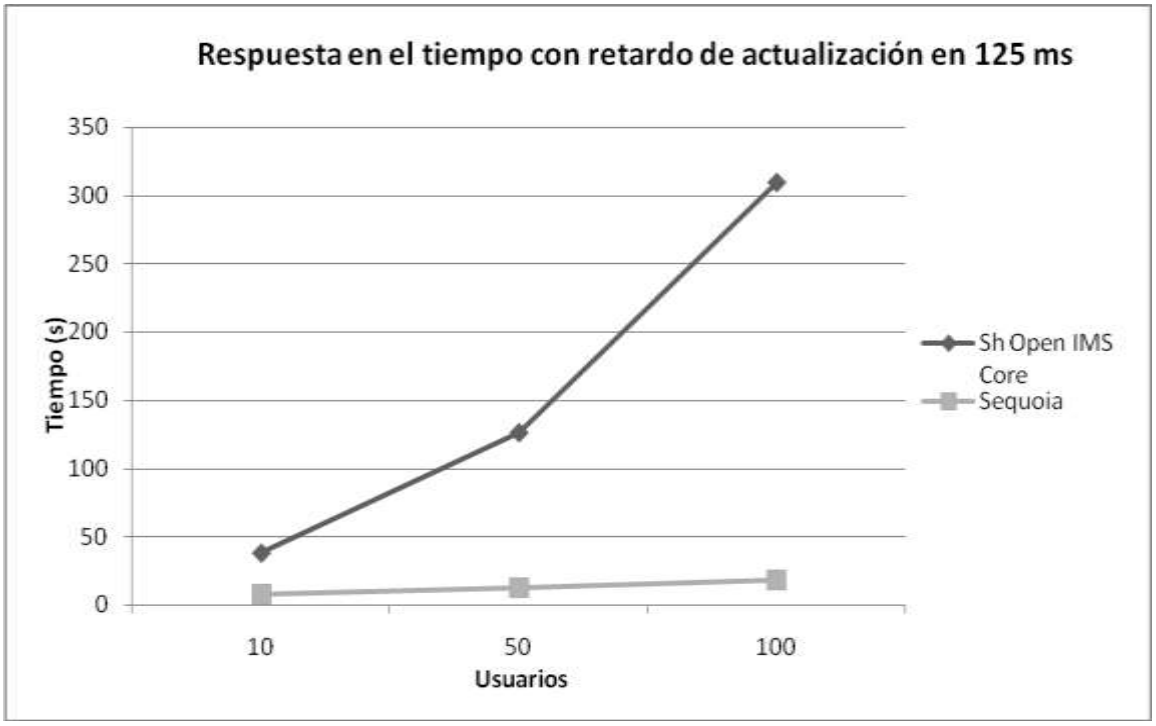

Figura 47. Comparación de la respuesta en el tiempo con un retardo de actualización de 125 ms

CAPITULO VI
CONCLUSIONES Y TRABAJOS FUTUROS

6.1. CONCLUSIONES

Después de validar el prototipo con la implementación del mecanismo propuesto y con el análisis de los resultados obtenidos en las pruebas de desempeño, se logró llegar a una serie de conclusiones que se desglosan a continuación.

- La sincronización de la información de usuario en el entorno de las NGN es factible de mejoras en cuanto al desempeño de la red y consistencia de los datos al hacer uso de las propuestas provenientes del campo de los Sistemas de Bases de Datos Distribuidos (DDBS - *Distributed DataBase Systems)*.

- La incorporación a las NGN de una plataforma *middleware* que implemente *group communication* y especializada en la sincronización de datos, permite un uso más eficiente de los recursos de red, y al mismo tiempo provee la facilidad de integración con los repositorios de datos de redes heredadas.

- Para el mecanismo de sincronización propuesto, el *overhead* introducido de forma inherente por hacer uso de una plataforma *middleware* es aceptable si se realiza una comparación del tráfico que se produce al hacer uso del mecanismo propio de IMS.

- Para la prestación de servicios *real time,* los cuales presentan un alto dinamismo de información y requieren una fuerte consistencia en la misma, es viable el uso de alternativas como las soluciones *middleware* con *group communication* para la actualización de datos de usuario, ya que se pueden mejorar considerablemente los tiempos de respuesta, traduciéndose esto en un servicio más eficiente y agradable para el usuario final.

- La independencia de proveedores de motores de bases de datos que admiten las plataformas *middleware*, permite integrar de forma rápida y eficiente los diferentes repositorios de información existentes en las NGN, sin presentar una alta exigencia en la compatibilidad de las plataformas que estos usen.

- La introducción de una plataforma *middleware* como mecanismo sincronización en el entorno de las NGN no implica la realización de cambios drásticos en la configuración de los componentes red por lo que es viable llevar su implementación a cabo en un entorno real.

- El uso del mecanismo propuesto permite una rápida salida al mercado de nuevos servicios que vayan a ser implementados por proveedores de servicios externos, ya que evita el uso de interfaces como la Sh y por ende de todo el desarrollo e implementación de aplicaciones que implica su despliegue.

- El uso de herramientas *open source* facilita el desarrollo de este tipo de proyectos, ya que estas permiten realizar cambios sustanciales a sus componentes sin necesidad de tener que pagar por licencias de ningún tipo. Además el apoyo encontrado en las comunidades, tanto de desarrolladores como de usuarios de estas herramientas, es muy grande y permite dar a conocer el trabajo que se viene desarrollando así como compartir experiencias con gente de todo el mundo.

- *Open IMS Core* y SEQUOIA se consolidan como herramientas útiles en las investigaciones realizadas en la FIET (Facultad de Ingeniería Electrónica y Telecomunicaciones) de la Universidad del Cauca, dada su naturaleza *open source* que permiten realizar modificaciones necesarias, contribuyendo al rápido avance en el desarrollo del conocimiento de las NGN dentro de la Universidad.

6.2. TRABAJOS FUTUROS

A continuación se presentan algunos trabajos que se pueden consideran para continuar y aportar a la propuesta aquí presentada.

- Agregar elementos de seguridad a las conexiones que se establecen entre los repositorios de información que usan el mecanismo propuesto en este trabajo. Por ejemplo implementando túneles VPN ó esquemas de cifrado y realizar conjuntamente un análisis del desempeño que se obtiene.

- Realizar una adaptación a bajo nivel de las primitivas de *group communication* sobre protocolos estandarizados para procesos AAA (*Authentication*, *Authorization*, *and Accounting*), como es el caso de los protocolos DIAMETER y LDAP (*Lightweight Directory Access Protocol*), ampliamente conocidos y utilizados en arquitecturas NGN.

REFERENCIAS

[1]. **G. Kessler, P. Southwick,** *RDSI Conceptos, funcionalidad y servicios*, McGraw – Hill, pág. 204-206, 2001

[2]. **J. Garrahan, P. Russo, K. Kitami y R. Kung,** *Intelligent Network Overview,* IEEE Communications Magazine, 1993.

[3]. **G. Hein,** *GSM Networks: Protocols, Terminology, and Implementation*. Artech House, Inc., 1999.

[4]. **A. Pachón,** *Evolución de los sistemas móviles celulares GSM.* Sistemas y Telemática, Universidad ICESI, 2004.

[5]. **Y. Amir y C. Tutu,** *From Total Order to Database Replication*, 22 international Conference on Distributed Computing Systems, Julio 2002.

[6]. **E. Shimokawa,** *Replicación de datos*, Universidad de Nacional de Trujillo. 2003 [En Línea] Disponible: http://inf.unitru.edu.pe/~edsh/documentos/bd_replicacion.pdf [Consulta: Abril de 2008].

[7]. **J. Seoane, A. Alonso,** *Replicación de Datos en Sistemas Distribuidos*, 2007, [En Línea]. Disponible: http://polaris.dit.upm.es/~aalonso/sodt/replicacion_07.pdf [Consulta: Enero 7 2008]

[8]. **M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso,** *Understanding Replication in Databases and Distributed Systems*, 2000.

[9]. **J. Acosta,** *Algoritmos de consistencia rápida orientados por la demanda en sistemas distribuidos de gran escala*, Tesis Doctoral, 2003.

[10]. **J.Gray, P. Helland, P. O'Neil, D. Shasha,** *The Dangers of Replication and a Solution*, 1996.

[11]. **M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso,** *Database Replication Techniques: a Three Parameter Classification*, 2000.

[12]. **GORDA Project,** *State of the Art in Database Replication*, 2006, [En línea]. Disponible: http://gorda.di.uminho.pt/library/wp1/GORDA-D1.1-V1.2-p.pdf [Consulta: Enero 2008].

[13]. **D. Bakken,** *Middleware*, Encyclopedia of Distributed Computing, Kluwer Academic Press, 2003. [En línea]. Disponible: http://www.eecs.wsu.edu/~bakken/middleware-article-bakken.pdf [Consulta: Junio de 2008].

[14]. **J.E. Armendariz, H. Decker, F.D. Muñoz, y J.R.G. Mendívil**, *A Closer Look at Database Replication Middleware Architectures for Enterprise Applications*, Trends in Enterprise Application Architecture, editorial Springer Berlin / Heidelberg pp. 69-83, 2007.

[15]. **M.F. Kaashoek,** *Group Communication In Distributed Computer Systems*, Tesis Doctoral, 1992.

[16]. **G. V. Chockler, I. Keidar y R. Vitenberg**, *Group Communication Specifications: A Comprehensive Study*, ACM Comput. Surv. 33, 4, 2001.

[17]. **M. Tamer, P. Valduriez**, *Principles of Distributed Database Systems*. Segunda Edición, Prentice Hall, 1998.

[18]. **P. Rob, C. Coronel,** *Sistemas de bases de datos: Diseño, implementación y administración*, Thomson Learning Ibero, 2004.

[19]. **V. Padilla,** *Exposición de bases de datos distribuidas*. Universidad de Colima. [En Línea]. Disponible: http://docente.ucol.mx/vpc1052/public_html/Expo%20SBDD.doc [Consulta: Diciembre de 2007].

[20]. **T.M. Thomas,** Java *Data Acces, JDBC, JNDI and JAXP*, M&T Books, 2002. pp. 235 – 236.

[21]. **J. O'Donahue**, *Java Database Programming Bible*, John Wiley & Sons, 2002. pp. 31 – 32.

[22]. **M. De la Rosa,** *Bases de Datos Distribuidas*, Editorial Universitaria. 2005. http://revistas.mes.edu.cu/eduniv/02-Libros-por-ISBN/959-16-0400/0336-Bases_de_Datos_Distribuidas.pdf

[23]. **Departamento de OEI,** *Diseño y optimización de bases de datos: Bases de datos distribuidas,* Universidad Politécnica de Madrid. [En Línea] Disponible: http://cmapspublic.ihmc.us/servlet/SBReadResourceServlet?rid=1161027337062_2033648941_456 [Consulta: Diciembre de 2007].

[24]. **M. Wiesmann, A. Schiper,** *Comparison of Database Replication Techniques Based on Total Order Broadcast*, 2005.

[25]. **Ericsson.** *IMS – IP Multimedia Subsystem. The value of using the IMS architecture*, 2004 [En Línea]. Disponible: http://www.citmo.net/library/Ericsson%20IMS.pdf [Consulta: Junio de 2007].

[26]. **W. Kellerer, M. Wagner, W. Balke, Wolf-Tilo.** Preference-based Session Management for Personalized Services. MoMuC (Mobile Multimedia Communications), 2003 [En Línea]. Disponible: http://www.l3s.de/~balke/paper/momuc03.pdf [Consulta: Julio de 2007].

[27]. **G. Camarillo, M.A Garcia-Martín,** *The 3G IP Multimedia Subsystem (IMS) Mergin the internet and cellular worlds,* Second Edition , John Wiley & Sons, 2006.

[28]. **A. Cuevas, C. García.** *Los pilares de las redes 4G: QoS, AAA y Movilidad*. [En Línea]. Disponible: http://www.ist-mobydick.org/publications/telecom02.pdf [Consulta: Julio de 2007].

[29]. **G. Jiménez**. *IMS, la visión de Lucent Technologies*, 2006 [En Línea]. Disponible: http://www.cintel.org.co/rctonline/noticia.php3?nt=5179&edicion=17 [Consulta: Julio de 2007].

[30]. **3GPP**. *Network architecture (Release 7)*. Technical Specification 23.002 Versión 7.1.0, 2006.

[31]. **3GPP**. *IP Multimedia Subsystem (IMS); Stage 2 (Release 8)*. Technical Specification 23.228 Versión 8.0.0, 2007.

[32]. **3GPP**. *Organization of Subscriber Data.* Technical Specification 23.008 Version 7.6.0, 2007.

[33]. **S. GIOIA.** *Understanding the Home Subscriber Server (HSS) Sh interface*. BEA Weblogic, sep 2006. [En Línea] Disponible: http://dev2dev.bea.com/pub/a/2006/10/home-subscriber-server.html?page=1 [Consulta: Febrero de 2008].

[34]. **3GPP**. *Sh Interface based on the Diameter protocol; Protocol Details (Release 8)*. Technical Specification 29.329 Versión 8.1.0, 2008.

[35]. **3GPP**. *IP Multimedia (IM) Subsystem Sh interface; Signalling flows and message contents (Release 8).* Technical Specification 29.328 Versión 8.1.0, 2008.

[36]. **3GPP**. *Service Aspects; Service Principles.* Technical Specification 22.101 Versión 7.9.0, 2007.

[37]. **A.M. Mejía. "***Redes Convergentes"*. Ciencia e Ingeniería Neogranadina No. 14. pp. 64 –7 4. 2004. [En Línea]. Disponible: http://www.umng.edu.co/www/resources/rev14_7.pdf [Consulta: Julio 2007].

[38]. **Solid Information Technology**. *Network Operators Must Address New Data Management Challenges of IMS.* 2007. [En Línea]. Disponible: http://www.solidtech.com/pdfs/IMSSolutionSheet.pdf [Consulta: Julio 2007].

[39]. **3GPP.** *Service Requirement for the 3GPP Generic User Profile (GUP); Stage 1.* Technical Specification 22.240 Versión 6.5.0, 2005.

[40]. **Sequoia Project,** *Continuent.org Sequoia 3.0 Basic Concepts,* 2007 [En Línea]. Disponible: http://sequoia.continuent.org/doc/latest/sequoia-basic-concepts.pdf , [Consulta: Agosto 2007].

[41]. **L. Gao, M. Dahlin, A. Nayate, J. Zheng, A. Iyengar**. *Improving availability and performance with application-specific data replication.* IEEE Transactions on Knowledge and Data Engineering, 2005.