Acceso a Servicios Web Basados en SOAP desde Redes de Baja Capacidad

Johana Quiñonez Collazos
Víctor Garzón Marín

TRABAJO DE GRADO
Presentado como requisito para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Director
Francisco Orlando Martínez Pabón
Magister en Ingeniería, área Ingeniería Telemática

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación de Servicios Avanzados de Telecomunicaciones
Popayán, Julio de 2012

Agradezco a Dios y la Virgencita por colmarme de bendiciones
y ser mis guías en cada momento de mi vida.
A mis padres las personas que más amo en el mundo, por quienes hago todo
para ser siempre un orgullo, gracias por su apoyo, amor y compresión.
A mis hermanos y sobrinos por ser la base fundamental de mi vida, gracias
por su apoyo incondicional y por comprender siempre mis locuras.
A mi novio, por convertir el amor en uno de los pilares de mi vida.
A Lonita mi "hijita" hermosa por su amor incondicional, y por
su compañía incansable en todas las trasnochadas.
A mis amigos por sus preciados consejos y gratos momentos, serán
por siempre el mejor recuerdo en mi paso por la universidad.
A mi amiguito Víctor, mil gracias por todo su apoyo
y compresión, te llevo en mi corazón.
A todos muchas gracias.

Johana Quiñonez C.

Agradezco a Dios por bendecir mi familia y
por todas las personas que ha puesto en mi camino.
A mis padres, pilares fundamentales en mi formación y
quienes hicieron de su familia el mejor sueño y el logro más preciado.
A mis hermanos y novia por su apoyo incondicional.
A mis compañeros de estudio, de trasnochas,
de triunfos y fracasos, y especialmente a Jowis,
por su comprensión y colaboración.

Víctor Garzón M.

A nuestro director, colaborador y amigo el Ing. Francisco Martínez, gracias por sus incontables consejos, guías y profundo apoyo en esta última etapa, nada hubiese sido lo mismo sin su presencia en toda la investigación.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

1. INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Dada la necesidad empresarial por responder de forma ágil y eficaz ante las oportunidades de negocio que puedan presentarse, surge la importancia de integrar los procesos y la información; pero en muchos casos la infraestructura de las TIC (tecnologías de la información y las comunicaciones) con las que cuentan las organizaciones, impiden mantener su nivel de competitividad y garantizar su crecimiento. Sin embargo, dicha infraestructura presta de manera eficiente servicios individuales en campos como gestión financiera, marketing, control de clientes, entre otros, el problema se presenta cuando se requiere interactuar entre aplicaciones o extraer funcionalidades de cada una de ellas con el objetivo de obtener una visión general de los procesos de la empresa para abarcar distintas áreas funcionales [1].

En este sentido, la arquitectura orientada a servicios SOA (Service Oriented Architecture) establece un modelo arquitectural cuyo objetivo es mejorar la agilidad, eficiencia y productividad de una empresa [2], mediante la integración de sus aplicaciones independientes y permitiendo el acceso a sus funcionalidades (las cuales se prestan como servicios) a través de la red; esto significa que SOA pone a disposición recursos o servicios que pueden ser descubiertos y consumidos por entidades que no necesitan conocer con anterioridad la existencia de los mismos [3]. Los servicios son funcionalidades concretas que pueden ser descubiertas en internet, y que ejecutan tareas que van desde la realización de una operación matemática simple, hasta otras más complejas, como por ejemplo, la consulta o modificación de información de clientes en una base de datos. Aunque no necesariamente se deben implantar servicios Web en el camino hacia la adopción de una solución de diseño basada en SOA, sin duda alguna, ésta es la implementación más habitual y madura de éste modelo arquitectónico [1, 4, 5].

Generalmente los servicios Web basados en SOAP (Simple Access Object Protocol) para el intercambio de datos, utilizan documentos XML (Extensible Markup Language) para representar la información. El lenguaje XML está escrito en texto plano (por esta razón requiere mayor información para representar un dato) y se basa en el uso de etiquetas generando una sobrecarga significativa en el tamaño de los documentos [6], estas características hacen necesario el uso de redes de alta capacidad en términos de ancho de banda para su transporte [7-9]. Sin embargo, en muchas zonas rurales colombianas y de otros países latinoamericanos el acceso a internet aún se realiza a través de redes de muy bajo ancho de banda [10]. Organizaciones como FITEL en Perú [11] y Compartel en Colombia [12] adelantan proyectos para la prestación de servicios de telecomunicaciones en estas zonas. El programa EHAS (Enlace Hispano-Americano de Salud) [13] también ha desarrollado diversos proyectos en busca de soluciones de conectividad en zonas rurales y

ha instalado hasta la fecha 28 sistemas de comunicación de voz y datos, inicialmente utilizando tecnología VHF/HF con costos de instalación muchos más bajos que los de la tecnología VSAT (Very Small Aperture Terminal) utilizada por la compañía Gilat [14] (que trabaja en compañía con FITEL), y luego con tecnologías WiFi para enlaces en los que sea posible asegurar línea de vista [10]. Adicionalmente, en los municipios de Silvia y Jambaló en el departamento del Cauca, el programa EHAS consolidó un laboratorio de comunicaciones de bajo costo en el que instaló un servidor que interconecta microredes VHF con redes WiFi de mayor capacidad. En otro contexto, las redes tácticas militares representan un entorno similar al ya mencionado; son utilizadas principalmente por unidades con un alto nivel de movilidad, como buques de guerra, aviones, teléfonos móviles o fuerzas especiales que suelen cargar un radio entre su equipo. Dicha movilidad, impide utilizar una infraestructura de comunicaciones fija y obliga a realizar sus comunicaciones a través de enlaces VHF/HF [15].

Las tecnologías de radio comunicaciones están caracterizadas por tener anchos de banda muy limitados, no más de 9600 bps (bits por segundo) en el caso de VHF y 2500 bps en HF [11]. Estos entornos no son los más adecuados para el acceso a los servicios Web previamente descritos, dada la gran cantidad de transferencia de información que sugiere el intercambio de mensajes XML, limitando en gran parte el acceso a los beneficios brindados por la arquitectura orientada a servicios, en ambientes civiles y militares.

Teniendo en cuenta el escenario descrito se plantea la siguiente pregunta de investigación: *"¿Cómo facilitar el acceso a servicios Web basados en SOAP desde redes de baja capacidad?".*

1.2 OBJETIVOS

1.2.1 Objetivo General

Definir un mecanismo para facilitar el acceso a servicios Web basados en SOAP desde redes de baja capacidad.

1.2.2 Objetivos Específicos

- Definir metadatos que permitan describir las capacidades de las conexiones que usan los clientes para acceder a los servicios.
- Adaptar un mecanismo de compresión para los mensajes XML intercambiados que se ajuste automáticamente, de acuerdo a la información entregada por los metadatos que describen la conexión.
- Construir un piloto que demuestre el funcionamiento del mecanismo de descripción y compresión de mensajes propuesto.

1.3 ALCANCE DEL PROYECTO

Los aportes alcanzados con la realización de este proyecto son:

- Definición de un conjunto de Metadatos para describir las características de las conexiones de red como ancho de banda, memoria y capacidad de procesamiento de los dispositivos cliente que acceden a los Servicios Web.
- Haciendo uso de la descripción de las capacidades de la red entregadas por los metadatos se determina cuál es el mejor mecanismo de compresión que se puede aplicar al mensaje SOAP de respuesta.
- Un piloto que integra las características funcionales de la descripción de conexiones de red y los mecanismos de compresión, para evaluar los resultados obtenidos con respecto a la precisión de los metadatos que describen las características de la red y a la eficiencia del mecanismo de compresión utilizado.

1.4 ESTRUCTURA DE LA MONOGRAFÍA

De acuerdo al contexto planteado, se han definido las siguientes secciones:

Capítulo 2.
Se construye una base inicial de conocimiento sobre las temáticas requeridas para el desarrollo del proyecto. Adicionalmente se realiza una breve descripción de los entornos de referencia y se efectúa una descripción de los aportes y brechas encontrados en algunos trabajos relacionados.

Capítulo 3.
Se presenta la arquitectura base del sistema mediador como solución al problema planteado, que integre el uso de mecanismos de compresión para disminuir el tamaño de la información trasmitida y el uso de metadatos que contengan información descriptiva del dispositivo que intenta acceder al servicio y de su conexión, para ayudar a seleccionar el compresor más optimo a implementar. Adicionalmente, con el fin de determinar la lógica del sistema mediador, se desarrolla un análisis teórico y un conjunto de pruebas que buscan determinar los parámetros que afectan el comportamiento de las herramientas de compresión y la información que debe ir contenida en el metadato.

Capítulo 4.
Se describe la implementación de la arquitectura del sistema mediador, donde se desglosa la funcionalidad de cada uno de sus componentes. Posteriormente, se presenta el algoritmo de selección basado en los modelos matemáticos desarrollados en el capítulo 3; y finalmente se presenta el desarrollo de un piloto de pruebas en un entorno de red

simulado con limitación en su ancho de banda y capacidad de procesamiento, que permite validar el sistema mediador.

Capítulo 5.
Se resumen los principales aportes del proyecto, se presentan las conclusiones más relevantes sobre el trabajo y se proponen algunos trabajos futuros.

Adicionalmente, en los anexos se presenta información complementaria sobre los temas tratados en los capítulos, así:

Anexo A.
Inicialmente se describen los servicios Web y operaciones implementadas para el desarrollo de las pruebas necesarias en el proyecto; posteriormente se presentan los resultados de las pruebas realizadas para determinar los parámetros que afectan a las herramientas de compresión.

Anexo B.
Se desarrollan los modelos matemáticos necesarios para determinar la lógica del sistema mediador.

Anexo C.
Se describe explícitamente el sistema mediador, las clases, paquetes y demás componentes internos.

Anexo D.
Se presentas los resultados obtenidos en los 10 escenarios propuestos, como pruebas de validación del sistema mediador.

Anexo E.
Manual de instalación y configuración de las herramientas empleadas en el desarrollo del piloto del sistema mediador.

Anexo F.
Artículos creados a partir de la investigación, desarrollo y producción de la solución planteada en el proyecto.

2. ESTADO ACTUAL DEL CONOCIMIENTO

2.1 MARCO TEÓRICO

El proceso de investigación está encaminado hacia definición, diseño e implementación de de un mecanismo que logra facilitar el acceso a los servicios Web disponibles en internet desde redes de baja capacidad; de esta forma, para contextualizar la investigación, se presentan los conceptos más relevantes en la creación del mecanismo desarrollado.

2.1.1 XML (Extensible Markup Language):

Define un lenguaje estándar que describe una clase de objetos de datos llamados *documentos XML (XML documents)* y el comportamiento del software que los procesa [6]. Respecto a su estructura, el documento conserva una jerarquía en árbol y está compuesta tanto de una parte lógica como física: físicamente, está conformado por unidades llamadas entidades que son generalmente representadas por archivos o porciones de archivo. Lógicamente, el documento se compone de declaraciones, elementos, comentarios, referencias a caracteres e instrucciones de procesamiento, todos ellos indicados por un marcado explícito [16]. Es importante mencionar que XML es el fundamento básico sobre el cual están construidos los servicios Web.

2.1.2 Servicios Web

Los servicios Web son componentes software que implementan funciones concretas y que permiten a las aplicaciones en desarrollo y a las ya existentes exponer sus funcionalidades para que puedan ser utilizadas en todos los puntos donde se necesiten. Existen dos formas de implementar Servicios Web, una de estas formas es un estilo de arquitectura conocido como REST (REpresentation State Transfer), la cual se ha popularizado en los últimos años y se centra en la interacción con recursos. Una segundo forma son los servicios Web basados en SOAP (Simple Object Access Protocol), los cuales aunque no son tan modernos siguen siendo muy populares y utilizados.

En un servicio Web basado en SOAP, la interacción con otros sistemas está prescrita en el documento WSDL y se realiza a través del intercambio de mensajes SOAP, típicamente usando protocolo HTTP y documentos XML serializados combinados con otros estándares de la Web definidos por el W3C [17]. Su arquitectura de se basa en la interacción entre tres componentes: proveedor (service provider), registro (service registry) y consumidor (service requestor). El proveedor alberga un módulo software (implementación de un servicio Web) accesible en la red, define la descripción para dicho servicio y la publica en el registro. El consumidor utiliza una operación de búsqueda para obtener la descripción del servicio desde el registro y usa dicha descripción para enlazarse con el proveedor e invocar o interactuar con la implementación del servicio Web. De esta forma estos tres componentes permiten la publicación, descubrimiento e invocación de servicios [18].

Aunque existen diferentes estilos para referirse a los servicios Web tales como REST y RPC [19], el enfoque del presente trabajo de grado se enmarca en los servicios Web basado en SOAP. En este sentido, este tipo de servicios se apoyan en algunas tecnologías para transportar y transformar la información, tales como WSDL, SOAP y UDDI [20].

2.1.2.1 WSDL (Web Service Description Language)

Es un lenguaje basado en XML que define las interfaces del servicio Web, los tipos de mensajes, los patrones de interacción, el mapeo de protocolos [21] y otros elementos que hacen posible agrupar definiciones de varios servicios Web en un sólo documento WSDL [22].

2.1.2.2 SOAP (Simple Object Access Protocol)

Es un protocolo basado en XML, que define el empaquetado para la comunicación de servicios Web y proporciona un formato de serialización para la transmisión de documentos XML a través de la red. SOAP consta de tres partes: un sobre (envelope) para describir lo que está en el mensaje y como procesarlo; un conjunto de reglas de codificación para expresar instancias de tipos de datos definidos por la aplicación; y una convención para representar llamadas a procedimientos remotos y sus respuestas [23, 24].

2.1.2.2.1. Características

a. Tipos de Datos

En [23] se indican los tipos de datos que pueden ser utilizados dentro de un mensaje SOAP, clasificados en dos grupos: simples y compuestos. Los tipos de datos simples, definidos como aquellos que no contienen partes, han sido tomados de las recomendaciones del W3C presentados en [25], y son una unión de los tipos de datos primitivos y derivados. Los tipos de datos primitivos son lo que no pueden ser definidos en términos de otros. Por otro lado, los derivados, son definidos en términos de primitivos. Algunos tipos de datos simples son: String, que representan cadenas de caracteres en XML, float y double, que corresponden al tipo de punto flotante de 32 y 64 bits de la IEEE, respectivamente; boolean, que es un tipo de dato que soporta el concepto de lógica matemática (verdadero o falso); e int, un tipo de dato para representar el concepto matemático de número entero, entre otros. Los tipos de datos compuestos son aquellos que se representan como relaciones entre otros valores, en ellos se pueden incluir arreglos y estructuras de datos. La principal diferencia entre una estructura y un arreglo, es que en la primera se utiliza el nombre de los subelementos para acceder a ellos, mientras que en el segundo se utiliza su posición dentro del arreglo. Algunos tipos de datos compuestos son: Object, para representar objetos con atributos; Arrays, que representan arreglos de objetos o tipos de datos iguales; Mixed, se pueden considerar como arreglos que contienen objetos y tipos de datos de diferente tipo (entre los que se pueden también incluir arreglos) [24, 26].

b. Estilo

"Este atributo permite determinar si la operación utilizada en el servicio Web está orientada a RPC, es decir, que el mensaje contiene parámetros y valores de retorno; o está orientada a documentos, indicando que el mensaje contiene documentos. Esta información puede ser utilizada para seleccionar un modelo de programación apropiado y afecta la manera como se construye el cuerpo del mensaje" [21]. De esta manera, existen dos tipos de estilo RPC y Document. Aunque, la definición presentada inicialmente fue tomada del documento oficial de la W3C para describir WSDL, se puede generar una gran discusión en torno a lo que realmente representa este atributo en un mensaje SOAP. En [27] se establece que éste no tiene nada que ver con el modelo de programación y que únicamente indica como traducir un documento WSDL en un mensaje SOAP.

Como se mencionó, el estilo afecta la manera como se construye el cuerpo del mensaje. En RPC la carga del mensaje se encuentra enmarcada en un elemento con el nombre de la operación del servicio Web que fue invocada [3, 28]. Document, por el contrario, ubica la carga directamente en el cuerpo del mensaje. Las figuras Figura 2-1 y
Figura 2-2 representan un mensaje utilizando estilo RPC y Document respectivamente.

Figura 2-1. Mensaje SOAP: Estilo RPC

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:getBooleanResponse xmlns:ns2="http://ws.asoap.unicauca.org/">
            <return>true</return>
        </ns2:getBooleanResponse>
    </S:Body>
</S:Envelope>
```

Figura 2-2. Mensaje SOAP: Estilo Document

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <return>true</return>
    </S:Body>
</S:Envelope>
```

2.1.2.3 UDDI (Universal Description, Discovery and Integration):

Es un mecanismo para registrar y descubrir servicios Web. Está construido sobre los estándares de Internet del W3C y de la IETF (Internet Engineering Task Force), como XML, HTTP, etc. UDDI representa un gran avance para el desarrollo de Internet como plataforma de negocios de tecnologías de la información; antes de su desarrollo no existía ningún estándar que permitiese localizar, ubicar o publicitar servicios de tratamiento de información [22, 29].

2.1.2.4 Plataformas de Desarrollo para Servicios Web.

Los servicios Web han sido una tecnología de amplia adopción dentro de SOA. En gran medida este éxito se debe a la aplicación de los diferentes estándares WS empresariales

en plataformas de desarrollo de aplicaciones y herramientas [30]. Por tanto, en la actualidad existen numerosas y variadas plataformas disponibles tanto en el mercado propietario como en la comunidad libre. De esta manera, se aclara que para el grupo objetivo de este proyecto, las plataformas libres se convierten en una elección natural, descartando las herramientas propietarias debido a sus altos costos. Dentro de las plataformas libres, existe una gran diversidad con variadas características y funciones; lo que hace necesario un estudio comparativo de las ya existentes [31]. Por tanto, con el fin de forjar una base teórica e identificar su idoneidad dentro del proyecto, se realiza una descripción de algunas plataformas libres líderes en servicios web, a partir de los criterios generados en la investigación [31]. Estos son:

- facilidad de uso e instalación.
- disponibilidad de información técnica, es decir disponibilidad de recursos e información detallada.
- desarrollo continuo.
- relevancia en el comercio (se refiere al alcance de uso por empresas o individuos).
- libre y de código abierto.

De esta manera, a continuación se describen a las plataformas: AXIS2, CXF y Metro como plataformas lideres en los servicios web.

a. AXIS2 [32]

- Es de código abierto.
- Su criterio de diseño es flexibilidad y extensibilidad.
- Sus implementaciones están disponibles en Java y C.
- Es una potente y flexible pila de servicios Web de tercera generación.
- Es la evolución de Apache SOAP Axis.
- Está construida con mayor rendimiento y flexibilidad para soportar variedad de estándares de servicios web, como también servicios asíncronos.
- Tiene soporte para REST.
- Posee una única plataforma de servidor, la cual proporciona funciones de servicio de monitoreo y permite fácil implementación y gestión de servicios.
- Es capaz de añadir interfaces de servicios Web a las aplicaciones Web.
- Puede funcionar como una aplicación de servidor independiente.

b. CXF [33]

- Es un framework de código abierto.
- Se construyen y desarrollan los servicios, a través de JAX-WS y JAX-RS.
- Entiende una variedad de protocolos como SOAP, XML/HTTP, REST HTTP, o CORBA
- Maneja una variedad de protocolos de trasporte como HTTP, JMS o JBI.
- Se basa en una arquitectura configurable, con el fin de apoyar una alta gama de transportes, enlaces de datos y tecnologías de extensión.

c. Metro [34]

- También utiliza JAXWS para el desarrollo de servicios, además evoluciona el código base de la implementación de referencia de la API Java para la especificaciones de los servicios Web XML JAX-WS.
- La base de código actual es compatible con: "JAX-WS 2.x Web Services Standars" y "JAX 2.x Data Binding".
- Utiliza componentes adicionales para proporcionar funcionalidades más allá de JAX-WS.
- El único mecanismo "XML binding" soportado es JAXB, aunque otros puedes adicionarse.
- No proporciona igual flexibilidad que Axis2 o CXF.

Teniendo claro los conceptos expuestos anteriormente, a continuación se presentan los escenarios de referencia que son la motivación del desarrollo de este proyecto y se identifican las necesidades encontradas.

2.2 ESCENARIOS DE REFERENCIA

Las características y beneficios de los servicios Web, los hacen útiles en diferentes escenarios, por ejemplo los escenarios empresariales, donde los recursos de red son extremadamente altos y casi ilimitados; y no empresariales donde no siempre se puede contar con las características de red o con los equipos necesarios para que los servicios Web puedan ser desplegados de una manera sencilla. Algunos de los escenarios considerados en el presente trabajo son los ambientes de tipo civil, y los escenarios militares.

2.2.1 Escenarios Civiles

Los escenarios civiles pueden ser representados fácilmente por las zonas rurales Colombianas o de otros países Latino-Americanos. En este tipo de zonas, su topografía, impide la utilización de equipos de bajo coste para la prestación de servicios de datos. Generalmente, no es posible en ellas asegurar una línea de vista, por lo que no siempre es viable utilizar tecnologías WiFi; los enlaces de tipo satelital por otro lado representan costos de mantenimiento e instalación demasiado altos para ser adoptados por comunidades aquí residentes. Finalmente la utilización de redes basadas en radios HF y VHF se convierte en casi la única opción para redes ubicadas en estos escenarios, limitando la velocidad de acceso a lo que la tecnología utilizada permite utilizar.

Organizaciones como FITEL en Perú [11] y Compartel en Colombia [12] adelantan proyectos para la prestación de servicios de telecomunicaciones en estas zonas. El programa EHAS (Enlace Hispano-Americano de Salud) [13] también ha desarrollado diversos proyectos en busca de soluciones de conectividad en zonas rurales y ha instalado

hasta la fecha 28 sistemas de comunicación de voz y datos, inicialmente utilizando tecnología VHF/HF con costos de instalación muchos más bajos que los de la tecnología VSAT (Very Small Aperture Terminal) utilizada por la compañía Gilat [14] (que trabaja en compañía con FITEL), y luego con tecnologías WiFi para enlaces en los que sea posible asegurar línea de vista [10]. Adicionalmente, en los municipios de Silvia y Jambaló en el departamento del Cauca, el programa EHAS consolidó un laboratorio de comunicaciones de bajo costo en el que instaló un servidor que interconecta microredes VHF con redes WiFi de mayor capacidad.

2.2.2 Escenarios Militares

Los escenarios militares, además de estar ubicados en zonas de difícil acceso geográfico como los mencionados en escenarios civiles, agregan un problema adicional puesto que mantienen un alto nivel de movilidad. Estas redes son utilizadas principalmente por buques de guerra, aviones, teléfonos móviles, radio teléfonos, entre otros [15]. Algunos proyectos se han enfocado en determinar los beneficios de los servicios Web en estas zonas, enfocándose en la prestación de información eficiente a las unidades que realmente lo necesitan y en las ventajas dentro del campo de batalla al tener un mejor y eficaz acceso a la información [5]. En este tipo de redes los altos niveles de movilidad ya mencionados, impiden utilizar tecnologías de comunicaciones fijas y en la mayoría de los casos obligan a la utilización de redes basadas en radio-enlaces HF/VHF.

Después de identificar los escenarios de referencia, se presentan los conceptos de disadvantages grids, compresión de datos y metadatos, necesarios para entender el proceso de desarrollo de este proyecto de investigación.

2.3 DISADVANTAGED GRIDS

Se refiere a redes de baja capacidad caracterizadas por tener bajo ancho de banda, altos retardos, frecuentes interrupciones de servicio, rendimiento variable, conectividad poco fiable y limitaciones impuestas por la energía de la red de comunicaciones inalámbrica que conecta a los diferentes nodos [35, 36]. Las redes de baja capacidad son determinadas así debido a la infraestructura utilizada y a la tecnología que se ven obligadas a implementar ya que se encuentran en lugares donde la topografía de la zona no permite asegurar línea de vista lo cual excluye el uso de tecnologías como WiFi, y la instalación de redes satelitales tiene un costo muy elevado debido a sus equipos y a todos los requerimientos necesarios para su montaje, obligando así a la utilización de enlaces de radio como tecnología a implementar.
En este sentido, su implementación a través de redes VHF y HF, las cuales, no permiten transmisiones de datos superiores a los 9,6 Kbps para VHF y a 2.5Kbps para HF lo cual influye directamente en el rendimiento de la red [13, 36].

2.4 COMPRESIÓN DE DATOS.

La compresión de datos es una forma eficaz de reducir el espacio de almacenamiento y ancho de banda requerido para su transmisión, dado que permite representar la misma información en un menor tamaño. En este contexto, existen dos tipos de compresión: compresión con pérdida y compresión sin pérdida. La primera, se utiliza para datos que puedan tolerar cierta pérdida de información como es el caso del audio o video. La segunda, por el contrario, se utiliza para datos que necesitan mantener la información intacta después de realizada la descompresión[37].

En el contexto de los documentos XML, generalmente se utiliza compresión sin pérdidas, pues es necesario garantizar la integridad de la información. Sin embargo, algunas herramientas como XMill [38] en ocasiones obvian etiquetas de cerrado y espacios entre líneas, lo que permite catalogarla como una herramienta de compresión con pérdidas. Al igual que XMill se han diversos proyectos que aprovechan características del documento XML como su estructura, marcado, entre otras, para realizar la compresión [5]. Este tipo de compresores varían su funcionamiento y efectividad de acuerdo al objetivo para el cual hayan sido diseñados.

Con el objetivo de comparar las diferentes herramientas de compresión a nivel de desempeño, se requiere analizar sus propiedades. A continuación se presentan algunas métricas que permiten medir el desempeño de las herramientas de compresión.

2.4.1 Métricas que Miden el Desempeño de las Herramientas de Compresión

Las siguientes métricas, descritas en [39] permiten realizar un paralelo entre eficacia, es decir, que tanto una herramienta de compresión logra reducir la longitud del documento; y eficiencia, que se refiere a la velocidad en que se realiza el proceso de compresión.

2.4.1.1 Radio Compresión

Permite medir la eficacia del mecanismo de compresión. Representa la relación entre la longitud del documento comprimido L_C y la longitud original L. Puede ser calculado a partir de la Ecuación 2-1.

Ecuación 2-1. Radio de compresión

$$R_C = \frac{L_C}{L}$$

2.4.1.2 Tiempo de Compresión

Permite medir la eficiencia de la herramienta al realizar el proceso de compresión. Representa el tiempo desde el inicio de una compresión hasta su finalización exitosa.

2.4.1.3 Tiempo de Descompresión

Al igual que el tiempo de compresión, esta métrica permite medir la eficiencia de la herramienta, pero esta vez representa el tiempo desde el inicio de una descompresión hasta la obtención exitosa del documento original.

2.4.2 Clasificación de Herramientas de Compresión

En [40] se presenta una clasificación de herramientas de compresión. Basándose en el contexto de los documentos XML las herramientas de compresión se pueden clasificar de acuerdo a cuatro características principales:

2.4.2.1 Consciencia de la Estructura del Documento

De acuerdo a esta característica, se pueden dividir en dos grupos:

a. Compresores de Texto para Uso General

El objetivo de este tipo de herramientas es representar el texto en un menor espacio mediante la sustitución de símbolos por otros equivalentes que utilizan una menor cantidad de bits o bytes. Su principal ventaja es que requiere menos espacio de almacenamiento y por lo tanto, leer el documento del disco o transmitirlo por un canal de datos resulta ser más rápido. Su desventaja radica en la necesidad de mayores capacidades de cómputo para el procesamiento de los documentos [41]. Dado que los documentos XML son almacenados utilizando un formato en texto plano, es posible procesarlos utilizando herramientas de compresión de texto.

Algunas herramientas de este tipo son:

- ***GZip (GNU Zip):*** Independiente del tipo de CPU, sistema operativo, sistema de archivos, y conjunto de caracteres que ha sido adoptado por el proyecto GNU [42, 43]. Está basado en el algoritmo de compresión sin pérdidas DEFLATE [44], el cual utiliza una combinación del algoritmo LZ77 [45] y de codificación *Huffman [46]*. LZ77 logra comprimir datos remplazando porciones de ellos con símbolos para que ocupen una menor cantidad de espacio. *Huffman* se utiliza para determinar cuáles de los símbolos se presentan con mayor frecuencia y los representa utilizando datos de pocos bits, mientras que los símbolos menos frecuentes los representa utilizando datos que ocupen mayor espacio.

- ***PPM (Prediction by Partial Matching):*** Utiliza un modelo basado en estadísticas que intentan calcular y predecir, a partir de una serie de símbolos, los siguientes en el documento. El resultado de dicho proceso es en realidad la diferencia entre los datos predichos y los datos reales. Para realizar el proceso de descompresión, el decodificador utiliza ésta diferencia y el mismo modelo utilizado en la compresión [47, 48].

- ***BZip2:*** utiliza una transformación Burrows-Wheeler [49] para convertir caracteres recurrentes en una cadena de letras idénticas. La transformación se aplica de tal forma que si una cadena contiene sub-cadenas que también han sido representadas en el modelo, la representación final se convierte en una extensión de las sub-cadenas. BZip2 puede lograr radios de compresión mucho más relevantes que GZip pero el tiempo de compresión requerido generalmente es mayor.

b. Compresores XML Conscientes

A diferencia de los compresores de texto, este tipo de herramientas de compresión se caracterizan porque aprovechan la estructura de los documentos XML para lograr mejores radios de compresión o mejores tiempos de compresión y/o descompresión. Algunas herramientas de este tipo son:

- ***XMill:*** Basa su funcionamiento en tres principios: el primero es separar la estructura de los datos, con la estructura se refiere a las etiquetas incluidas en el documento y los datos son las cadenas de texto que representan la información; el segundo principio es agrupar los datos relacionados en contenedores, por ejemplo, los datos de tipo <nombre> se almacenan en un contenedor y los de tipo <teléfono> en otro; finalmente, el tercer principio consiste en aplicar compresores semánticos, refiriéndose a la utilización de compresores que generen un mejor rendimiento para cada contenedor [38].

- ***SCMPPM (Structural Context Models PPM):*** utiliza un modelo de codificación específico independiente para codificar el texto contenido en un determinado elemento de estructura o contexto estructural. Por ejemplo, si se desea codificar el texto contenido en un correo electrónico, es de esperar que el campo *Date* contenga fechas, mientras que el campo *From* contenga nombres y direcciones de correo electrónico. El uso de modelos diferentes para codificar elementos estructurales diferentes mejora el radio de compresión [50].

Las herramientas de compresión conscientes de la estructura del documento XML pueden dividirse en tres grupos de acuerdo a las siguientes características

2.4.2.2 Soporte de Consultas

El soporte de consultas (*query)* permite recuperar información directamente del archivo comprimido. La principal ventaja de este tipo de compresores es que disminuyen considerablemente el tiempo de procesamiento pues no se requieren descompresión, sin embargo, no son muy eficientes en cuanto a radios de compresión. Pueden dividirse en dos tipos, homomórficos y no homomórficos. Un mecanismo homomórfico se caracteriza porque el documento comprimido conserva la misma estructura y puede ser parseado de la misma forma como si fuese el documento original. En las herramientas de tipo no homomórfico, el proceso de codificación altera de alguna forma la estructura del

documento XML original. Algunas herramientas de compresión capaces de soportar consultas son XSeq [51], el cual es no homomórfico y QXT [52] que es una extensión capaz de soportar consultas y del tipo homomórfico de XWRT [53, 54].

Los mecanismos de compresión que no soportan consultas son utilizados como mecanismos de almacenamiento, pues pueden lograr radios de compresión mucho mayores aunque su tiempo de procesamiento también es mayor.

2.4.2.3 Codificación de Datos Binario

Algunos compresores XML conscientes se han concentrado en el desarrollo de un formato más compacto para los documentos y han explotado el concepto de XML Binario, el cual, consiste en conservar la estructura del documento pero representar la información en formatos de datos binarios en lugar de usar texto plano. De esta manera, se reduce el espacio de almacenamiento requerido y el procesamiento del documento se realiza mucho más rápido. Algunos proyectos que utilizan XML binario son Xebu [55], FastInfoset [56] de SunMicrosystem, Efficient XML [57] de Agile Delta y WAP Binary XML [58] desarrollado por Open Mobile Alliance y luego adoptado por W3C como uno de sus estándares.

Los compresores de datos binario pueden incluirse también en el grupo de compresores homomórficos, puesto que mantienen la estructura del documento y en algunos casos permiten recuperar la información sin realizar todo el proceso de descompresión.

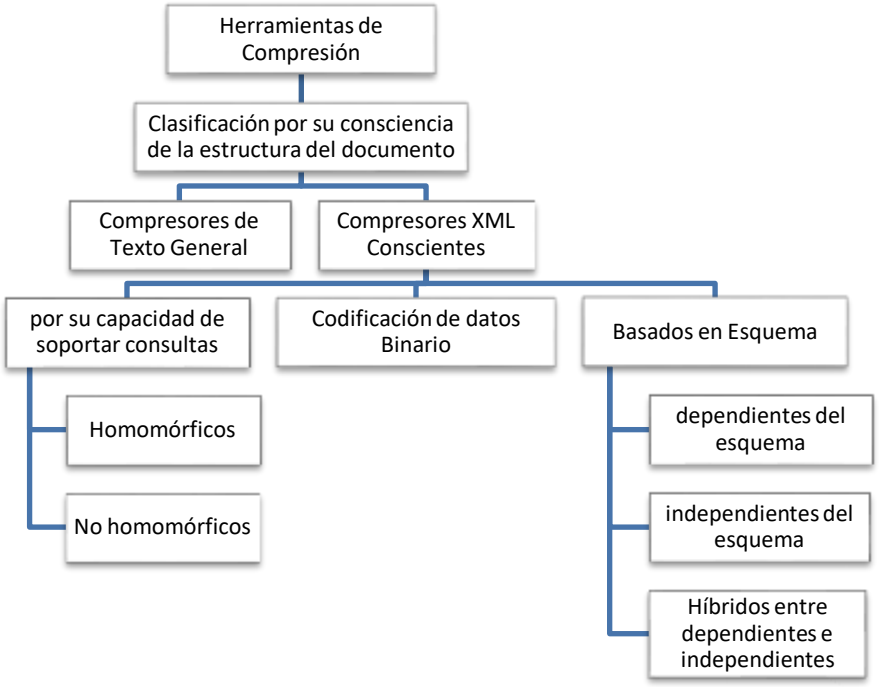
2.4.2.4 Compresores Basados en Esquema

Algunos documentos XML cuentan con un documento de esquema, el cual contiene información que describe los datos contenidos, sus etiquetas, la codificación utilizada, entre otros. Los compresores de éste tipo utilizan dicho documento de esquema, para disminuir la cantidad de información a partir de la utilización de la descripción en él incluida. Dado que cierta información es omitida en el proceso de compresión, estas herramientas requieren el esquema también para la descompresión. Entre los compresores basados en esquema se pueden encontrar: DTDPPM [59], Millau [60], EXIFicient [61, 62], entre otros. Su principal desventaja es que no siempre se puede asegurar la existencia de un documento de esquema, por lo que no son muy usados en la práctica [40].

De acuerdo al documento de esquema, los compresores se pueden dividir en tres grupos: los compresores dependientes del esquema, que requieren de la existencia de un documento de esquema; los compresores independientes del esquema, no tienen en cuenta el documento de esquema; y finalmente, los que se pueden considerar como una combinación de estos dos últimos, pues pueden funcionar sin un documento de esquema, pero ante su existencia aprovechan la información allí contenida [63].

La Figura 2-3 presenta la clasificación planteada de las herramientas de compresión.

Figura 2-3. Clasificación de las herramientas de compresión



2.4.3 Metadatos

Dado el aumento vertiginoso de la información en formato electrónico y gracias a la aparición de Bibliotecas Digitales, surge la necesidad de solucionar los inconvenientes presentados en la recuperación de información [64, 65]. Los diferentes servicios de búsqueda actuales, recuperan un alto número de documentos, que en su mayoría no satisfacen la búsqueda de información, debido a su baja pertinencia. Para ello numerosas comunidades desarrollan y promueven el uso de la "Web semántica", que puede ser entendida como una extensión de la ya existente, con la adición de metadatos que den significado a la información, permitiendo razonar automatizada mente sobre ella. De esta manera la Web semántica busca facilitar el procesamiento de la información por parte de las maquinas, a través de integración, reutilización y procesamiento de la información contenida en la Web [66].

2.4.3.1 Definición

En la actualidad son comúnmente definidos como "los datos sobre los datos". Aunque realmente en la literatura actual no existe un consenso sobre lo que son, se pueden entender como información descriptiva sobre el contexto, condición, calidad o características de un objeto, recurso y dato; que tienen como objetivo facilitar su recuperación, evaluación, autenticación, preservación o interoperabilidad [64].

2.4.3.2 Importancia

Los beneficios que aportan los metadatos y su importancia en el la actualidad dependen del área en que se utilicen [67], pero en términos generales se puede decir:

- ***Disminución de tráfico en la red:*** permiten indizar gran cantidad de datos de diferentes tipos sin generar sobrecarga en la red, es decir utilizan poco ancho de banda, debido a que se indiza la representación del objeto y no el objeto en sí [68].
- ***Incrementan la accesibilidad:*** la posibilidad de acceder a la información es mucho más factible con la existencia de conjuntos de metadatos que describan los objetos o recursos de información [69].
- ***Expandir conocimiento:*** los metadatos, agregan contenido, estructura y contexto a los objetos de información. De esta manera, colaboran con el proceso de descubrimiento y recuperación de conocimiento desde repositorios de objetos, ya que analizan el contenido del objeto a profundidad [66].
- ***Control de Información:*** a través de ellos es posible realizar un control de acceso a información restringida o a los recursos [68]. Es posible entonces establecer sistemas de filtrado, generar bases para una autenticación y mecanismos para definir grados de confianza sobre las fuentes de información [66].
- ***Intercambio de Información:*** "Los metadatos permiten el intercambio de la información sin la necesidad de involucrar el intercambio de los recursos mismos. Esta particularidad facilita entre otras cosas las búsquedas sobre colecciones distribuidas. Además los metadatos permiten una descripción precisa y discreta de los recursos permitiendo la creación de colecciones virtuales de descripciones donde agrupan los objetos de información para satisfacer requerimientos específicos" [66].
- ***Preservación del objeto original:*** La utilización de metadatos y el uso de lenguajes controlados, permite aumentar la precisión de la mayoría de las búsquedas en Internet [70].

2.4.3.3 Lenguajes de Metadatos

Surgen como una posibilidad de expandir las capacidades de los metadatos, facilitando el acceso al documento objeto y enriqueciendo su descripción. A través de ellos se define la sintaxis de las estructuras de los metadatos, y además proveen las especificaciones semánticas necesarias [64, 67] . A continuación son presentadas tres opciones de adopción de los metadatos, la primera es SGML (Standard Generalized Markup Language), la segunda es XML (eXtensible Markup Language) y finalmente HTML (HyperText Markup Language) como una adopción de adición al documento HTML descriptor del servicio web con la etiqueta META.

- ***SGML:*** definido por la norma ISO 8879 [71]. Permite la creación de diferentes lenguajes de etiquetado a partir de una DTD (Document Type Definition), los cuales pueden convertirse en estándares orientados según la comunidad de usuarios para quienes se

especifique. Proporciona una sintaxis para definir la estructura lógica de los documentos, los componentes y elementos [64]. Algunas de las características de SGML son:

- No existe limite en el tamaño del documento,
- Es un estándar Internacional
- Posee flexibilidad en el uso de texto
- Maneja un esquema jerárquico, que interrelaciona los niveles necesarios.

- ***XML:*** Es un lenguaje derivado de SGML, su descripción se presentó en la sección 2.1.1

- ***HTML:*** Es un lenguaje de marcas de hipertexto. Se basa en el metalenguaje SGML y es el formato de los documentos de la World Wide Web. No describe como tal un Metadato externo; en HTML, los elementos META o metatag se utilizan para especificar información sobre el documento y se insertan en la cabecera del documento [67, 72]. Lo cual, puede realizarse de dos formas diferentes:
  - desde dentro de un documento, por medio del elemento META. La estructura de las etiquetas <META> es la de una declaración *propiedad: valor*.
  - desde fuera de un documento, vinculando los metadatos por medio del elemento LINK.

  El standard HTML define la estructura de una etiqueta META, pero no sus valores; es decir, aunque es común insertar metadatos en un documento HTML por medio de la etiqueta <META>, no existe ninguna regla sobre la información que se puede o no incluir en los metatags [73].

2.4.3.4 Sistemas de Metadatos (Estructuras)

Actualmente es posible encontrar numerosos sistemas de metadatos, también conocidos en la literatura como estructuras. Seguidamente se describen algunos sistemas de metadatos seleccionados, partiendo del análisis de los criterios: descripción de contenido, localización y accesibilidad, identificación de documentos en un entorno distribuido y desarrollo y avance del proyecto.

De esta forma la descripción de cada sistema de metadatos se hizo basada en los parámetros seleccionados de la investigación realizada por Dempsey & Heery [74], Taylor [2004] y Heery [75], [76], también mostrada por Martín [77].

Inicialmente se presenta cada parámetro y a lo que hacen referencia:

- ***Origen:*** Inicio del proyecto
- ***Mantenimiento:*** Se refiere a la el nivel de actualización y mantenimiento que tiene el esquema de metadatos.
- ***Documentación existente:*** cantidad de información disponible: manuales, listas de distribución, tutoriales, foros, etc.
- ***Objetivo/misión:*** como su nombre lo indica esto se refiera a la finalidad del esquema.

- ***Disciplina/ámbito de aplicación:*** se refiere a al enfoque de la comunidad que lo aplica activamente. Abarca 2 posibilidades: Disciplina académica o Profesional.
- ***Simplicidad/complejidad:*** Se refiere a la complejidad en la creación de un metadato basado en ese esquema en particular. En este campo se ubican las personas encargadas de la creación de los metadatos y si ellos deben tener un conocimiento especializado para poder lograrlo.
- ***Interoperabilidad:*** se refiere a la posibilidad de comunicación e intercambio de información entre diferentes tipos de sistemas, sin la necesidad de manipulaciones técnicas adicionales.
- ***Extensibilidad:*** se refiere a la posibilidad de incluir mayor nivel de detalle en la descripción, a través de campos opcionales o calificadores.
- ***Flexibilidad:*** en el aspecto interno se refiere a la flexibilidad para describir diferentes tipos de documentos, y de manera general se refiere a la evolución que puede tener de acuerdo al avance tecnológico sobre el mismo contexto.
- ***Estructura:*** Define las características del esquema. Una estructura se puede clasificar dependiendo del manejo permitido o restringido que se le pueda dar a sus campos, como se describe a continuación.
    - ***Opcional:*** se refiere a si contiene o no, campos opcionales u obligatorios dentro de las opciones con las que cuenta el esquema.
    - ***De repetición:*** se refiere a si permite o no, repetir campos o subcampos para la formación del metadato.
    - ***Con subcampos:*** es decir si permite o no, la admisión de subcampos
    - ***Con control:*** hace referencia a si cuenta o no con campos donde se permite el control de valores.
- ***Sintaxis:*** Se evalúa si el esquema de metadatos aplica una sintaxis, que sustente la forma como deben ser codificados los elementos.
- ***Normalización internacional***: Se tiene apoyo de organismos expertos en estándares, y cuales con los estándares que los fundamentan.

Por tanto, a continuación se desglosan los parámetros nombrados para cada sistema de metadatos.

2.4.3.4.1. DC o DCMI (Dublin Core Metadata Initiative).

- ***Origen:*** 1995
- ***Organismo creador:*** Online Computer Library Center (OCLC) y National Center for Supercomputing Applications (NCSA).
- ***Encargado del Mantenimiento:*** The Dublin Core Metadata Initiative (DCMI)
- ***Documentación existente:*** manuales en línea.
- ***Objetivo/misión:*** creado para la catalogación de documentos electrónicos con el fin de poner a disposición de los autores elementos básicos para describir recursos Web.

Además tiene mayor énfasis en el acceso y recuperación de la información que en la descripción.

- ***Disciplina/ámbito de aplicación:*** es utilizado por múltiples comunidades, dada su facilidad de descripción de cualquier tipo de documentos.
- ***Simplicidad/complejidad:*** es simple e intuitivo, gracias a que ha sido diseñado para que los metadatos puedan ser incluidos en el momento de crear el documento. Para su creación no requiere personal experto, lo que genera mayor productividad.
- ***Interoperabilidad:*** puede ser exportado a distintos formatos bibliotecarios de catalogación y de metadatos.
- ***Extensibilidad:*** En principio la descripción es sencilla, pero permite tener mayor nivel de detalle a través de calificadores.
- ***Flexibilidad:*** alta flexibilidad, dado que nada es obligatorio, de esta forma el usuario es quien elije la profundidad en la descripción.
- ***Estructura:*** propone 15 elementos básicos, agrupados en tres grupos, así: contenido, propiedad intelectual e instalación/manipulación. Estos son: 1. Titulo, 2. Autor o creador, 3. Materias y palabras claves, 4. Descripción, 5. Editor, 6. Otras contribuciones, 7. Fecha, 8. Tipo de recurso, 9. Formato, 10. Identificador del recurso, 11. Fuente, 12. Idioma, 13. Relaciones, 14. Alcance, 15. Gestión de derechos.
  - ***Opcional:*** todos los elementos son opcionales
  - ***Repetición:*** todos los campos son repetibles
  - ***Subcampos:*** no tiene subcampos
  - ***Control:*** no se requiere control de valores
- ***Sintaxis:*** HTML, XML, RDF.
- ***Normalización internacional***: es una norma de derecho desde octubre de 2001 ANSI/NISO Standard Z39.85-2001. Y posee versiones actualizadas que datan de Febrero de 2003 y Mayo de 2007.

2.4.3.4.2. RDF (Resource Description Framework).

- ***Origen:*** 1997
- ***Organismo creador:*** World Wide Web Consortium (W3C) apoyado por las organizaciones de normalización NISO, ISO, y por las empresas de red como Netscape y Microsoft. Sus orígenes se dieron a través de los proyectos Netscape's Meta-Content Framework y Platform for Internet Content Selection-PICS.
- ***Encargado del Mantenimiento:*** World Wide Web Consortium (W3C)
- ***Documentación existente:*** recomendaciones, manuales y tutoriales en línea, software, grupos de discusión, wikis y presentaciones de uso educativo.
- ***Objetivo/misión:*** su principal objetivo es proporcionar interoperabilidad entre aplicaciones que intercambien información en la Web; a través de la definición de un modelo y sintaxis óptimos para el intercambio de descripciones de recursos Web.
- ***Disciplina/ámbito de aplicación:*** es utilizado por múltiples comunidades, dada su facilidad de descripción de cualquier tipo de documentos.

- ***Simplicidad/complejidad:*** es simple, gracias a que proporciona un sistema, que puede entender al ordenador, permitiendo especificar tipos de recursos y propiedades. Para su creación no requiere personal experto, lo que genera mayor productividad.
- ***Interoperabilidad:*** permite la interoperabilidad entre aplicaciones que intercambian información compresible por la red, con el fin de proporcionar una infraestructura que soporte actividades de metadatos.
- ***Extensibilidad:*** lograr la descripción es simple y basado en las 3 partes que lo componen es posible lograr mayor nivel de detalle, el usuario es quien elije la profundidad en la descripción
- ***Flexibilidad:*** alta flexibilidad, gracias a su facilidad de acople para describir cualquier tipo de documento y además, por el apoyo de organizaciones importantes que lo mantienen en continuo crecimiento.
- ***Estructura:*** se basa en la identificación de recursos web a través de URIs (Uniform Resource Identifiers) en términos de propiedades y valores simples. Una descripción RDF es un conjunto de proposiciones simples denominadas cada una como tripleta, dado que se compone de un sujeto, un predicado y un objeto.
    - ***Opcional:*** al no poseer campos establecidos, sino una estructura definida para las preposiciones contenidas en la descripción; toda la información que se desee puede ser descrita a través de RDF.
    - ***Repetición:*** por lo anterior es posible repetir la estructura de una proposición RDF cuantas veces se desee.
    - ***Subcampos:*** al estar basado en XML los subcampos se definen vasados en la sistaxis***.***
    - ***Control:*** no se requiere control de valores pero se debe cumplir con la tripleta: sujeto, predicado y objeto.
- ***Sintaxis:*** XML.
- ***Normalización internacional***: RDF está recogido en las recomendaciones del W3C: Primer [78], Concepts [79], Syntax [80], Semantics [81], Vocabulary (Schema) [82] y Test Cases [83].

2.4.3.4.3. TEI (Text Encoding Initiative).

- ***Origen:*** 1987
- ***Organismo creador:*** surge de la investigación en conjunto de 3 asociaciones: Association of Computers in the Humanities (ACH), Association for Computational Longuistics (ALC), y Association of Leterary and Linguistic Computins (ALLC).
- ***Encargado del Mantenimiento:*** TEI Consortium
- ***Documentación existente:*** manuales y tutoriales en línea, software, grupos de discusión, wiki y presentaciones de uso educativo.
- ***Objetivo/misión:*** creado para la representación de diversos textos literarios y lingüísticos, con el fin de su investigación, preservación y búsqueda en línea.

- ***Disciplina/ámbito de aplicación:*** se especializa en la comunidad de la literatura, las humanidades y las ciencias sociales.
- ***Simplicidad/complejidad:*** su complejidad depende de la cantidad de información que se desee incluir. Aunque requiere de conocimiento previo del modelo y de las reglas bibliotecarias para su creación.
- ***Interoperabilidad:*** puede ser concebido para intercambio.
- ***Extensibilidad:*** brinda un marcado semántico detallado.
- ***Flexibilidad:*** es dependiente del usuario y a sus necesidades.
- ***Estructura:*** propone 15 elementos básicos, agrupados en tres grupos, así: contenido, propiedad intelectual e instalación/manipulación.
    - ***Opcional:*** todos los elementos son opcionales
    - ***Repetición:*** todos los campos son repetibles
    - ***Subcampos:*** no tiene subcampos
    - ***Control:*** no se requiere control de valores
- ***Sintaxis:*** SGML y XML.
- ***Normalización internacional***: TEO hace referencia a los siguientes tres estándares: ISO 646: Information Technology – ISO7-bit coded carácter der for information interchange, ISO 10646: Information Technology – Universal Multiple-Octel Coded Character Set (UCS) y Unicode.

2.4.3.4.4. MARC DTD (Machine Readable Cataloging Document Type Definition).

- ***Origen:*** inicia en 1960. En 1992 incluye el campo 856 y en 2002 aparece la versión de MARC XML
- ***Organismo creador:*** Library of Congress
- ***Encargado del Mantenimiento:*** Network Development and MARC Standards Office, Library of Congress.
- ***Documentación existente:*** existen: Manuales en diferentes idiomas, listas de distribución y tutoriales.
- ***Objetivo/misión:*** inicia para catalogación de documentos impresos y posteriormente incorpora campos para otro tipo de recursos: publicaciones seriadas, mapas, música, archivos de computadora, registros sonoros, manuscritos, material audiovisual y recursos electrónicos.
- ***Disciplina/ámbito de aplicación:*** se enfoca en la comunidad bibliotecaria.
- ***Simplicidad/complejidad:*** es un formato complejo, se requiere de personal capacitado y especializado, con alto conocimiento de las reglas de catalogación. Lo que genera altos costos.
- ***Interoperabilidad:*** puede ser exportado a distintos formatos bibliotecarios de catalogación y de metadatos.
- ***Extensibilidad:*** si permite incluir mayor nivel de detalle a través de sus campos opcionales y de subcampos.
- ***Flexibilidad:*** tiene poca flexibilidad.

- ***Estructura:*** este formato posee más de 900 campos, con el fin de incluir características propias de los recursos electrónicos como: requerimientos del sistema, características del archivo, tipo de archivo y acceso y localización electrónica.
  - ***Opcional:*** contiene campos opcionales y obligatorios.
  - ***Repetición:*** algunos campos y subcampos pueden ser repetibles, otros pocos no.
  - ***Subcampos:*** contiene numerosos subcampos.
  - ***Control:*** se recomienda controlar valores en algunos campos.
- ***Sintaxis:*** HTML y XML
- ***Normalización internacional***: su implementación se apoya en las normativas nacionales e internacionales: *Information Interchange Format (ANSIZ39.2) y Format for Information Exchange (ISO 2709)* respectivamente.

Como un concepto adicional, a continuación se presenta la descripción de la técnica matemática denominada Regresión Matemática que será útil para el desarrollo de la solución propuesta en este proyecto.

2.4.4 Regresión Matemática

El término *regresión* fue introducido por Galton en [84] refiriéndose a la ley de la regresión universal, la cual es una técnica estadística empleada para estudiar la relación entre variables cuantitativas. Existen dos tipos de regresión: simple, es decir la relación de dos variables, y múltiple que es la relación de más de dos variables.

El análisis de regresión lineal, explora y cuantifica la relación entre una variable denominada dependiente (Y) y una o más variables denominadas independientes o predictivas (X_1, …, X_n)

De esta manera el objetivo es reconocer si hay relación entre las variables que se estén analizando, de qué tipo es la relación y si es posible predecir el valor de una de ellas en función de la otra. Para ello existen métricas que son vitales para determinar si la regresión es válida o no. A continuación se presenta la definición de dichas métricas:

2.4.4.1 Coeficiente de Correlación Lineal de Pearson (r):

Indica si los datos representados por puntos (x,y) en un diagrama, tienen tendencia a disponerse alineadamente. Esto excluye restas verticales y horizontales. El signo puede determinar si la relación es directa o inversa. Algunas propiedades son:

- Es adimencional
- Toma valores en un rango [-1,1]
- Las variables serán incorreladas si r=0, es decir no existe correlación entre los datos (x,y)
- Existe relación lineal perfecta entre dos variables si r=1 o r=-1
- Los mejores casos en la relación lineal, serán cuando r esté más cerca de +1 o -1.

- Aunque no se puede asegurar que exista una buena relación lineal si r<0,7 y la relación se puede considerar como anómala si r<0,4.
- Si el coeficiente de correlación es positivo significa que existe una relación directa o creciente entre X e Y, mientras que si es negativo quiere decir que existe una relación inversa o decreciente entre X e Y.

2.4.4.2 Coeficiente de determinación R^2 o Porcentaje de variabilidad:

Mide la proporción de la variabilidad total explicada por el modelo de regresión planteado, o la proporción del total que es debida a la regresión. Se espera (lo deseado) que esta proporción sea alta y cerca de 100% y solo una pequeña parte sea debido al error. Algunas de sus propiedades son:

- Es una cantidad adimencional
- Toma valores en un rango [0,1] o se puede expresar de 0 a 100%.
- Cuando un ajuste es bueno, R^2 será cercano a uno (100%)
- Cuando un ajuste es malo, R^2 será cercano a cero
- R^2 puede ser considerado en cualquier modelo de regresión, pero en el caso particular de regresión lineal simple, la expresión $R^2 = r^2$ se cumple.

Finalmente, se presenta el soporte de esta investigación, a través de trabajos relacionados encontrados a nivel nacional e internacional.

2.5 TRABAJOS RELACIONADOS

Durante el desarrollo de este proyecto se encontraron diferentes trabajos relacionados, de acuerdo a los campos abarcados en el proceso de investigación. El trabajo presentado en [85], es un ejemplo claro de los beneficios entregados por los servicios Web también en redes de baja capacidad, en él se presenta un sistema llamado DUNAJ que permite la interconexión de diferentes tipos de usuarios en ambientes militares y que ha sido desarrollado utilizando una arquitectura orientada a servicios. Igualmente en [86] se presentan los beneficios de los servicios Web en estos ambientes, además se agrega el problema asociado a la utilización de documentos XML, en éste trabajo aconsejan utilizar MMHS (un protocolo de transporte de uso militar) en vez de http como una forma de solucionar el problema.

Los siguientes trabajos se centran en el consumo de servicios Web en redes de baja capacidad y buscan dar consejos a tener en cuenta cuando se quieran realizar sistemas que faciliten este acceso. En el trabajo de investigación presentado en [4] se presenta la necesidad de adaptar el servicio a las posibilidades en cuanto a capacidades del equipo que tiene el usuario que intenta consumirlo, para ello indican que es necesario adoptar un mecanismo que permita de alguna manera describir el equipo con el que el usuario intenta consumir el servicio. Igualmente en [35] expone la necesidad de tener alguna

información sobre la red disponible para cualquier mecanismo que se desarrolle como solución al problema de acceso a un Servicio Web desde redes de baja capacidad. Por otro lado, en [87] se muestra que si bien es necesario utilizar algún mecanismo de compresión de datos para reducir el monto de información transmitida, debe tenerse en cuenta el tiempo de procesamiento utilizado por la herramienta seleccionada.

Otros trabajos están relacionados a la utilización de compresión de datos, [88] describe una comparación de herramientas de compresión que incrementan la eficiencia de SOAP en rede de baja capacidad, pero solo se centra en el estudio de tres herramientas: GZip, FastInfoset y EfficientXML. En la investigación presentada en [89] se presenta un estudio de diferentes herramientas de compresión para XML y se centran específicamente en XMill y WBXML.

De manera similar, se encuentran trabajos relacionados a la adopción de Metadatos como necesidad para describir información que pueda ser accedida en internet; algunos trabajos analizados en esta área son [49] que presenta la importancia y necesidad de los metadatos; y [90] y [69] que logran explicar los componentes internes de los metadatos, algunas características de lenguajes y estructura.

A continuación se presentan los trabajos que generaron un mayor aporte a la presente investigación.

2.5.1 SOA – Cross Domain and Disadvantaged Grids

R. Haakseth en [91] describe tres proyectos con los que la Defensa de Investigación Noruega (Norwegian Defence Research Establishment) participó en la "NATO Coalition Warrior Interoperability Demonstration (CWID)", los cuales describen una serie de experimentos llevados a cabo con el fin de cumplir con los objetivos de cada proyecto. De los tres proyectos expuestos se ha seleccionado uno de ellos como aporte importante para la base documental y conceptual de este anteproyeto. El proyecto se denomina "1086 – Secure and Pervasive SOA" y recoge 2 experimentos importantes:

- ***Cross Domain Web Service:*** Este experimento se desarrolló con el fin de lograr el intercambio automático de información entre dominios de seguridad[1], a través del uso de etiquetas y filtrado confiable; su principal objetivo fue determinar cómo una etiqueta XML puede ser utilizada para proporcionar descripción y seguridad sobre el contenido de los mensajes SOAP, el cual se probó evaluando la funcionalidad de 2 etiquetas con atributos diferentes en el intercambio de información entre dos dominios con políticas de seguridad distintas[2]. Los resultados obtenidos fueron satisfactorios

1"Conjunto de entidades a las que se aplica una política de seguridad igual ejecutada por una autoridad única".

2"Reglas aplicadas en las bordes de la red con el fin de protegerla de la fuga de información"

aplicando una combinación confiable de etiquetas descriptivas y políticas de seguridad implementadas en los bordes de la red con el fin de evitar fugas de información.
Este trabajo expone la necesidad de usar etiquetas descriptivas dentro del mensaje SOAP (metadato) que permitan manejar la información y los mecanismos necesarios para el tratamiento de los mensajes; sin embargo, no hace explícita la implementación y el desarrollo del algoritmo usado para la definición del metadato, ni la manera como se realiza la identificación de las características o las políticas de seguridad de la red.

- ***Disadvantaged Grids:*** Teniendo en cuenta que los servicios Web son la forma más adaptada para aplicar SOA, y que éstos necesitan para su acceso un alto ancho de banda, es un desafío para este trabajo lograr el acceso a los servicios Web desde redes de baja capacidad. El informe propone diferentes mecanismos para optimizar el acceso a servicios Web desde redes de baja capacidad, bajo las condiciones de ciertas medidas preestablecidas:
  - Es necesaria una representación eficaz de la información y el uso adecuado de modelos de datos.
  - Se debe utilizar un método de compresión de datos con el fin de reducir la sobrecarga de información y el tamaño de los mensajes SOAP. Proponen el uso de EFX Efficient XML de manera genérica.
  - Se requiere un filtrado de la información, con el fin de limitar el envío de datos innecesarios y así ahorrar ancho de banda.
  - Uso adecuado del protocolo de transporte para el envío de los mensajes SOAP. El documento propone la Pila de XOmail en reemplazo de HTTP sobre TCP/IP.

  Este experimento comprueba la necesidad de implementar diferentes medidas que logren el acceso a servicios Web, con el fin de mejorar la interoperabilidad de los mensajes SOAP y reducir los requerimientos de ancho de banda. Sin embargo, no realiza un análisis a fondo de las opciones en cuanto a los modelos de datos y se limita al uso exclusivo de Efficient XML como compresor de datos sin tener en cuenta ciertos casos en los cuales otros tipos de compresión podrían tener un mejor desempeño y rendimiento: Por otro lado, el stack de XOmail no es libre lo que limita su facilidad de acceso y aplicación.

2.5.2 Compressing SOAP Messages by Using Pushdown Automata

En ambientes con anchos de banda limitados, el alto monto de carga generado por los encabezados de los mensajes SOAP resulta desventajoso. Bajo esta premisa, el artículo en [8] presenta algunas investigaciones centradas en la búsqueda de una representación más compacta de documentos XML y específicamente en representaciones binarias de los mismos. Sin embargo, agrega que dadas las características propias de los mensajes SOAP, muchos de los resultados no son aplicables al campo de los servicios Web. En el artículo se exploran algunos esquemas de compresión conocidos, entre los que están: GZip, XMill, xml ppm y FastInfoset. Finalmente presenta técnicas para comprimir mensajes SOAP de una manera eficiente, basándose en un nuevo enfoque que explota el hecho de que los

mensajes de red en XML suelen ser descritos por una gramática conocida tanto por el emisor como por el receptor y que esta gramática se puede obtener a partir de los documentos WSDL de los servicios Web.

Este trabajo aporta una visión general de algunos esquemas de compresión conocidos, que pueden usarse en la búsqueda de una solución para reducir el tamaño de los mensajes SOAP, pero no tiene en cuenta el consumo de recursos de procesamiento y memoria que requerirán los esquemas de compresión presentados, ni considera el tipo de conexión que se utilizará para transmitir los mensajes.

2.5.3 Compressing XML with Multiplexed Hierarchical PPM Models

J. Cheney en [92] describe alternativas para la compresión de XML e ilustra paralelos entre efectividad y velocidad. Describe experimentos usando diferentes compresores de texto, entre los que incluye XMILL (un compresor que transforma los documentos al exponer redundancias y luego aplica una compresión estándar de texto) para comprimir diferentes documentos XML. Desde este punto de vista, se describen los dos resultados principales, un codificador binario online para XML llamado Encoded SAX (ESAX) que comprime mejor y más rápido que los métodos existentes; y un codificador online, adaptativo, basado en XML-conscious que comprime hasta un 35% mejor que los métodos existentes, pero requiere mayor tiempo de procesamiento.

Finalmente describe una nueva técnica llamada “multiplexed hierarchical model MHM”, la cual, combina la compresión de texto con el conocimiento de la estructura jerárquica del documento XML. MHM comparado con otros métodos como GZip y BZip2, mejora hasta en un 5% la compresión de documentos XML en formato de texto y entre 10% y 35% utilizando datos estructurados.

Este trabajo aporta una visión de la compresión de datos XML sin dejar de lado las capacidades de procesamiento que podrían requerirse. Sin embargo, no estudia la aplicación de éstos métodos en los servicios Web, ni en la transmisión de mensajes SOAP; por consiguiente, tampoco tiene en cuenta el tipo de conexión que utilizada.

2.5.4 XML compression techniques: A survey and comparison

Teniendo en cuenta la importancia que ha tenido el lenguaje XML en el intercambio de información en la red, pero a su vez considerando su principal desventaja respecto a la gran cantidad de información adicionada por el uso de etiquetas, S. Sakr en [40] presenta un estudio de algunos mecanismos de compresión, donde establece una completa clasificación de dichos mecanismos y una comparación de 11 de ellos. Dicho estudio y comparación concluye con algunas sugerencias a la hora de utilizar un mecanismo de compresión que mejor se adapte a los requerimientos del usuario. Sin embargo, en esta comparación no se tiene en cuenta ningún mecanismo basado en una representación binaria de datos como Efficient XML o FastInfoset. Cabe aclarar además, que este

documento solo se basa en el estudio de mecanismos de compresión de documentos XML y no en su aplicación a servicios Web basados en SOAP.

2.5.5 Aportes y Brechas

Tabla 2-1. Aportes y Brechas

T. RELACIONADO	APORTES	BRECHAS
"SOA – Cross Domain and Disadvantaged Grids" – Experiment "Cross Domain Web Service"	Este experimento expone las ventajas y la necesidad del uso de etiquetas descriptivas (metadatos) dentro del mensaje SOAP que permitan el manejo de la información y los mecanismos para el tratamiento de los mensajes.	No expresa la implementación y el desarrollo del algoritmo usado para la determinación del metadato, ni la manera como se realiza la identificación de las características o las políticas de seguridad de la red.
"SOA – Cross Domain and Disadvantaged Grids" – Experiment "Disadvantaged Grids"	Comprueba la necesidad de implementar diferentes medidas que logren el acceso a servicios Web desde redes de baja capacidad, con el fin de mejorar la interoperabilidad de los mensajes SOAP y reducir los requerimientos de ancho de banda. Entre sus aportes se destacan la representación eficaz de la información y los métodos de compresión.	No realiza un análisis a fondo de los modelos de datos y se limita al uso exclusivo de Efficient XML como compresor sin tener en cuenta que en algunos casos otros mecanismos de compresión podrían tener un mejor desempeño. Por otro lado, el stack XOmail utilizado no es libre lo que limita su facilidad de acceso y aplicación.
Compressing SOAP Messages by Using Pushdown Automata	Una visión general de algunos esquemas de compresión conocidos, que pueden usarse en la búsqueda de una solución para reducir el tamaño de los mensajes SOAP.	No tiene en cuenta el consumo de recursos de procesamiento, ni de memoria que requieren los esquemas de compresión presentados. Tampoco considera el tipo de conexión en la cual se transmiten los mensajes SOAP.
Compressing XML with Multiplexed Hierarchical PPM Models	Una visión de la compresión de datos XML sin dejar de lado las capacidades de procesamiento que podrían requerirse.	Sólo se centra en la compresión de documentos XML, mas no en la aplicación de estos a los servicios Web, ni en la transmisión de mensajes SOAP; tampoco tiene en cuenta el tipo de conexión que se utiliza.
XML compression techniques: A survey and comparison	Una clasificación de herramientas de compresión en el contexto de los documentos XML.	No enfoca el estudio en documentos SOAP sino en general para XML, tampoco se enfoca en la utilización de herramientas de compresión para el consumo de servicios Web.

3. ANÁLISIS Y DISEÑO DE LA ARQUITECTURA DE REFERENCIA PARA EL SISTEMA MEDIADOR

A partir de la información presentada en capítulos anteriores, es claro que XML agrega carga significativa a la información de un servicio Web, obligando a disponer de redes con anchos de banda considerables para su consumo. Sin embargo, existen escenarios donde dicho ancho de banda no puede ser garantizado. De esta forma, el utilizar mecanismos de compresión que ayuden a disminuir el tamaño de la información transmitida, se convierte en una solución viable para facilitar el acceso a servicios Web en redes con capacidad limitada [5]. No obstante, y como ya se ha mencionado, existen diferentes tipos de mecanismos de compresión que varían su efectividad y eficiencia de acuerdo a los componentes hardware del equipo donde se ejecuten, haciendo necesario el uso de algunos parámetros de entrada a la hora de elegir el mejor mecanismo de compresión en cada caso; dichos parámetros representan información descriptiva sobre el servicio Web y sobre el cliente que intenta accederlo, los cuales pueden ser representados a través de metadatos.

En este capítulo se presenta la arquitectura de referencia para un sistema mediador que busca dar solución al problema planteado. El sistema en cuestión, realiza dos tareas importantes: la primera, es crear y consumir metadatos con información descriptiva de los clientes y la segunda, es determinar la herramienta de compresión encargada de procesar los mensajes SOAP a partir de los metadatos.

De esta manera, con el objetivo cumplir las tareas del mediador, se desarrollaron un conjunto de pruebas con el fin de determinar cuáles parámetros afectan el comportamiento de las herramientas de compresión seleccionadas. A partir de los resultados obtenidos, se diseñaron modelos matemáticos que describen el comportamiento de las herramientas y se determino la información que debe ir incluida en el metadato. Finalmente se realizo una selección de las mejores opciones de metalenguajes y estructuras implementados en el metadato.

3.1 DOMINIO DEL PROBLEMA

La arquitectura clásica de los servicios Web está compuesta por tres componentes básicos y su interacción es como se describe en la sección 2.1.1. La Figura 3-1 presenta dicha arquitectura y hace explícito el núcleo del problema ubicado en el consumo del servicio, es decir, en la interacción entre el consumidor y el proveedor del servicio, no en el descubrimiento o publicación del mismo. De esta manera, la solución propuesta y desarrollada en este documento, es diseñar e implementar un *sistema mediador* que

facilite dicha interacción haciendo uso de herramientas de compresión. A continuación, se analizará la arquitectura de referencia para el *sistema mediador*.

Figura 3-1. Ubicación del problema en una arquitectura clásica de servicios Web SOAP

3.2 ARQUITECTURA DE REFERENCIA PARA EL SISTEMA MEDIADOR

La Figura 3-2 presenta una arquitectura de referencia para el s*istema mediador*, dicha arquitectura sigue un esquema cliente-servidor. Del lado del cliente, se encuentra el modulo encargado de recibir las peticiones SOAP del consumidor del servicio y enviarlas a través de la red de baja capacidad al servidor, luego recibe las respuestas comprimidas y las descomprime transformándolas de nuevo a un formato SOAP para entregarlas al consumidor. Del lado del servidor, es donde se realizan las tareas más importantes, en él se han incluido dos subcomponentes principales: el primero se encarga de la recolección y análisis de metadatos, mientras que el segundo se encarga de realizar la compresión; la funcionalidad de este módulo, consiste en recibir las peticiones SOAP del cliente, enviarlas al proveedor del servicio para obtener las respuestas, comprimir dichas respuestas de acuerdo a la información contenida en los metadatos y enviarlas de vuelta al cliente a través de la red de baja capacidad. Los metadatos nombrados anteriormente contienen información descriptiva del cliente; estos se almacenan en una base de datos y el servidor accede a ellos para tomar la decisión de cuál es la herramienta de compresión más acorde según la lógica definida por los modelos matemáticos explicados en la sección 3.3.3.

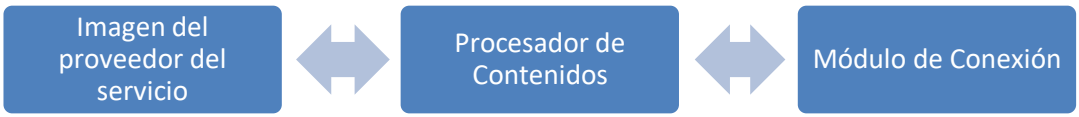
Figura 3-2. Arquitectura de referencia sistema mediador

A continuación se explica con más detalle cada módulo de esta arquitectura.

3.2.1 Mediador del Lado del Consumidor: Cliente

El Cliente se compone de tres módulos importantes:

Figura 3-3. Módulos básicos del Cliente



3.2.1.1 Imagen del Proveedor del Servicio

Para que el consumo del servicio se pueda realizar de forma efectiva y transparente, debe existir un mecanismo que mantenga la misma naturaleza del proveedor del servicio Web, por ello, es importante la adición de este módulo, debido a que su función es actuar como una imagen del proveedor pero ubicado en el mismo lado del consumidor. Sus principales funciones son: recibir las peticiones SOAP del consumidor y mantener la conexión hasta la entrega efectiva de las respuestas.

3.2.1.2 Procesador de Contenidos

Una vez la imagen del proveedor del servicio recibe una petición SOAP, la pasa al procesador de contenidos, quien se encarga de transformar la petición en un formato adecuado para que el módulo de conexión la pueda transferir al Servidor. Cuando el Servidor envía la respuesta, en este módulo se realiza el proceso de descompresión y adecuación del mensaje nuevamente en formato SOAP, para ser enviado al consumidor a través de la imagen del proveedor.

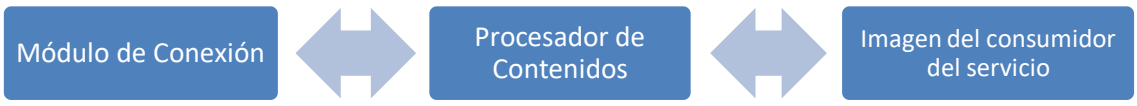
3.2.1.3 Módulo de Conexión

Como su nombre lo indica se encarga de la conexión entre el Cliente y el Servidor; cada vez que recibe una respuesta la debe pasar de nuevo al procesador de contenidos, quien después de realizar todo el proceso de transformación, la entrega a la imagen del proveedor.

En el capítulo 4 se explicará con más detalle cada uno de los módulos que componen al Cliente.

3.2.2 Mediador del Lado del Proveedor: Servidor

Figura 3-4. Componentes básicos del Servidor



3.2.2.1 Módulo de Conexión

Cumple las mismas funciones que el módulo de conexión del Cliente véase 3.2.1.3, es decir se encarga de la conexión entre el cliente y el servidor, enviando las peticiones/respuestas, desde y hacia, el procesador de contenidos y la imagen del consumidor del servicio.

3.2.2.2 Procesador de Contenidos

En este caso, el procesador de contenidos debe contar con dos componentes importantes: el primero es un Repositorio de Metadatos donde se encuentra alojada la información del equipo consumidor y la caracterización de su conexión; el segundo es el Motor de Compresión quien se encarga inicialmente, de hacer el análisis necesario para determinar entre las opciones disponibles cual es la herramienta de compresión que mejor se ajusta a la información contenida en los metadatos y a las características del documento SOAP de respuesta; posteriormente, dicha herramienta se utiliza para comprimir el mensaje. Estos componentes son de suma importancia y su descripción detallada se encuentra en las secciones 3.3 y 3.4.

3.2.2.3 Imagen del Consumidor del Servicio

De igual manera que en el cliente, para que el consumo del servicio se pueda realizar de forma efectiva y transparente al proveedor, este modulo actúa como imagen del consumidor y se encarga de realizar el consumo final del servicio.

En este punto, se observa la importancia de las herramientas de compresión y los metadatos dada su influencia dentro del mediador, por ello se ve la necesidad de describir detalladamente los componentes: Motor de Compresión y Repositorio de Metadatos, en las siguientes secciones.

3.3 MOTOR DE COMPRESIÓN

De acuerdo al planteamiento realizado en la sección 3.2.2.2, el Motor de Compresión es uno de los componentes del Módulo Procesador de Contenidos del lado del servidor. Este componente contiene todas las herramientas de compresión disponibles y la lógica necesaria para tomar la decisión de cuál es la herramienta más acorde para el consumidor. La mejor herramienta de compresión para el acceso a un servicio Web específico, puede ser vista como la que genera un menor tiempo de respuesta, siendo éste, el tiempo medido desde el envío de la petición hasta su consumo efectivo. Dicho tiempo de respuesta está condicionado por:

- **El tiempo que tarda el proveedor en entregar la respuesta:** Es el tiempo requerido por el proveedor del servicio para la recepción de la petición, su procesamiento interno y el envío de la respuesta al cliente.
- **El retardo propio de la conexión:** es una característica propia se las conexiones y se refiere al tiempo que le toma a una unidad de información atravesar la red.
- **El tiempo de procesamiento agregado por el mediador:** es el tiempo de procesamiento requerido por el mediador para el envío de información entre cada uno de sus módulos.
- **El tiempo generado por la herramienta de compresión seleccionada:** es el tiempo utilizado por la herramienta para comprimir y/o descomprimir el documento.

De los anteriores, el tiempo generado por la herramienta de compresión seleccionada se ve afectado adicionalmente por tres propiedades que han sido definidas como métricas que miden el comportamiento del mecanismo de compresión. Estas métricas son: radio de compresión, tiempo de compresión y el tiempo de descompresión [39]. El radio de compresión está ligado a la eficacia del mecanismo, es decir, que tanto puede reducir el tamaño de la información; es una característica propia del compresor y por tanto se asume que sin importar el equipo donde sea ejecutado, dicho radio no va a variar para un documento dado. Los tiempos de compresión y descompresión hacen referencia a la eficiencia, es decir, que tan rápido se hace el trabajo de compresión y descompresión, estas métricas dependen en gran medida de la capacidad de procesamiento del equipo donde se ejecute la herramienta.

De esta manera, la construcción de este componente se lleva a cabo en tres fases:

- **Fase I. Selección de las Herramientas de Compresión**: en esta fase se determina cuáles de las herramientas de compresión investigadas estarán incluidas dentro del motor de compresión, a partir de un análisis comparativo de sus características.
- **Fase II. Análisis de Comportamiento:** se determinan los parámetros que afectan el comportamiento de las herramientas de compresión seleccionadas de acuerdo a las métricas: radio de compresión, tiempo de compresión y descompresión, a través de pruebas experimentales.
- **Fase III. Creación de Modelos Matemáticos:** *Después de determinar* los parámetros *que influyen en el rendimiento de las herramientas de compresión, se diseña un modelo matemático que se convertirá en la lógica del servidor.*

A continuación se desarrolla cada una de las fases.

3.3.1 FASE I. Selección de las Herramientas de Compresión (De acuerdo a sus características)

Durante el proceso de investigación de este proyecto se recolectaron en total 21 herramientas de compresión; en la sección 2.1.4 se realizó una clasificación de ellas

basándose en la información presentada en [40]. Finalmente se reunieron sus principales características como se muestra en la Tabla 3-1 (la explicación de las casillas se encuentra en la sección 2.1.4).

Tabla 3-1. Comparación teórica de las características de los métodos de compresión

No.	Herramienta de Compresión	XML Binario	XML Consciente	Soporta Consultas	Código fuente y/o binario	Funciona	Depende del Esquema
1	Axechop [93]	NO	NO	NO	NO	--	NO
2	Bzip2 [94]	NO	SI	NO	SI	SI	NO
3	DTDPPM [59]	NO	NO	NO	SI	SI	SI - DTD
4	Exalt [95]	NO	NO	NO	SI	NO	NO
5	EXIFicient [61, 62]	SI	NO	NO	SI	SI	NO*
6	FastInfoset [96]	SI	NO	NO	SI	SI	NO
7	Gzip [97]	NO	SI	NO	SI	SI	NO
8	Millau [60]	NO	NO	NO	NO	--	SI
9	OpenEXI [98]	SI	NO	NO	SI	SI	NO*
10	PPM [92]	NO	SI	NO	SI	SI	NO
11	RnGzip [99]	NO	NO	SI	--	--	SI
12	SCMPPM [100]	NO	NO	NO	SI	SI	NO
13	TREECHOP [101]	NO	NO	SI	SI	--	NO
14	Xaust [102]	NO	NO	NO	SI	SI	SI - DTD
15	Xgrind [103]	NO	NO	SI	SI	--	NO
16	XMILL [38]	NO	NO	NO	SI	SI	NO
17	XMLPPM [104]	NO	NO	NO	SI	SI	NO
18	Xpath [105]	NO	NO	SI	--	--	NO
19	Xpress [106]	NO	NO	SI	--	--	NO
20	XQueC [107]	NO	NO	SI	--	--	NO
21	XWRT [53]	NO	NO	NO	SI	SI	NO

NO* = puede o no necesitar el esquema

De acuerdo a la información resumida en la tabla anterior, se realizó una selección de un subconjunto de herramientas de acuerdo a los siguientes criterios:

- **Criterio 1-** los compresores que pueden soportar consultas tienen una característica adicional y es que no logran radios de compresión relevantes [108]. Teniendo en cuenta que el principal problema que se quiere superar es la baja tasa de transferencia, se buscan mecanismos que permitan obtener radios muy pequeños de compresión, de esta manera se descartaron todos los mecanismos capaces de soportar consultas.

- **Criterio 2-** se requiere de la existencia del código fuente o al menos un binario para poder utilizar el compresor como una herramienta dentro del mediador. Por tanto, fueron descartados todos aquellos de los cuales no fue posible encontrar dicho código fuente o binario, o que por una u otra razón no fue posible lograr su funcionamiento.
- **Criterio 3-** al igual que en los documentos XML comunes, en los servicios Web no es posible garantizar en todos los casos un documento de esquema; sin embargo, sí existe tal documento es posible encontrarlo fácilmente haciendo uso de la información contenida en el WSDL. Por otro lado, una herramienta de compresión basada en esquema requiere dicho documento, tanto para la compresión como para la descompresión [40], obligando a que sea compartido entre el cliente y el servidor, lo que aumenta la cantidad de información transmitida. Por ende se descartan los mecanismos basados en esquema.

Siguiendo las condiciones presentadas, los compresores seleccionados teóricamente para pasar a la siguiente fase son los que se listan a la Tabla 3-2.

Tabla 3-2. Herramientas de compresión seleccionados

No.	Compresor
1	Bzip2
2	EXIFicient[1]
3	EXIFicient + GZip[2]
4	FastInfoset
5	Gzip
6	PPM
7	SCMPPM
8	XMILL
9	XMLPPM
10	XWRT

3.3.2 FASE II. Análisis de Comportamiento de las Herramientas de Compresión.

Como se explicó en la sección 3.3, en ésta fase se analiza el comportamiento de las 10 herramientas de compresión seleccionadas. Primero se realiza una serie de pruebas para seleccionar la API de referencia y posteriormente se determinan los parámetros que influyen en la selección de la herramienta a utilizar durante el consumo del servicio.

[1] Tanto EXIFicient como OpenEXI son implementaciones de Efficient XML, su única diferencia radica en la licencia aunque ambas son de código libre. Para este proyecto se escogió EXIFicient por encima de OpenEXI porque esta primera contaba con mayor documentación de respaldo.

[2] EXIFicient + GZip ha sido tenido en cuenta como una combinación de EXIFicient agregando una segunda fase de compresión utilizando GZip.

3.3.2.1 Descripción del Entorno de Pruebas

Las pruebas se realizaron en escenarios con algunas características que difieren unas de otras, a continuación se explicarán las configuraciones que fueron similares y las adicionales se explicaran en cada prueba.

a. Descripción del Equipo

El plan de pruebas se llevó a cabo utilizando la siguiente configuración del sistema:

Procesamiento	
Procesador:	Intel core i7
Núcleos de procesamiento:	8
Velocidad por núcleo:	2GHz
Velocidad total:	16GHz
Memoria RAM	4GB

Sistema Operativo	
Sistema:	Linux
Distribución:	Ubuntu 11.10
Arquitectura:	X86

La memoria RAM no se tuvo en cuenta en ninguno de los equipos o pruebas desarrolladas en este proyecto, ya que no es una característica que afecte directamente a las métricas que miden el desempeño de una herramienta de compresión de datos.

b. Descripción de las Muestras

Los documentos SOAP sobre los que se realizaron las pruebas fueron generados por diferentes servicios Web[1] desarrollados en APIs basadas en AXIS y METRO. Dependiendo de la prueba se tomó el servicio Web y el número de documentos necesarios para realizarla. En la descripción de cada una de las pruebas, se explicará en detalle cual es la muestra que se toma de base.

3.3.2.2 Selección de la API de Referencia.

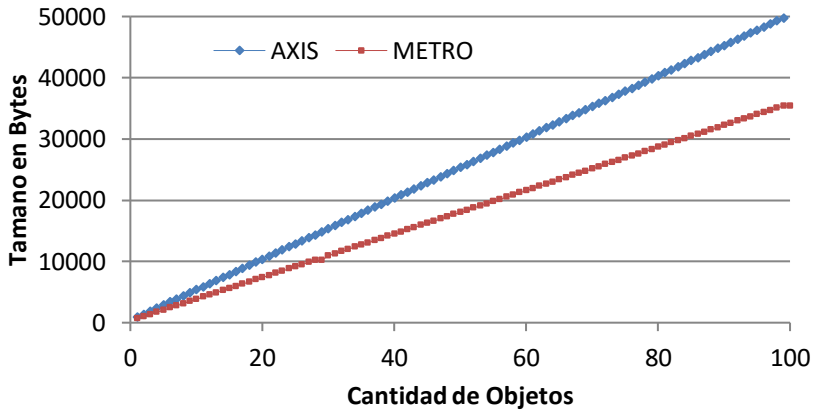
La naturaleza de SOAP no especifica una API que soporte de la mejor manera en todos los casos el desarrollo y construcción de servicios Web, por lo que su implementación está sujeta al lenguaje de programación que se emplee; siendo Java el lenguaje utilizado, se encuentra una gran cantidad de plataformas útiles para implementar servicios Web SOAP como se describe en la sección 2.1.2.4, por lo que es necesario determinar cuál de las tres propuestas es la más adecuada para un óptimo desarrollo del ambiente de pruebas. La selección de ésta se realizó en dos partes, una teórica y una práctica.

[1] En el Anexo A se encuentra la explicación de los servicios Web implementados.

Teóricamente se toma como referencia la investigación realizada en los trabajos [31] y [109]. En el primero se concluye que entre las plataformas evaluadas, AXIS, CXF y METRO son las que mejor se desempeñan en determinados casos; y en el segundo, que METRO es una mejor opción por encima de CXF. De esta forma, se infiere la preferencia teórica de AXIS y METRO cuando se realiza una implementación de servicios Web utilizando Java.

A partir de lo anterior, se procede a realizar pruebas experimentales con el fin de determinar la plataforma de desarrollo entre AXIS y METRO que genera un mejor comportamiento. Inicialmente se analiza la relación de tamaño; en la **¡Error! No se encuentra el origen de la referencia.**, se aprecia que los documentos AXIS aún con la misma cantidad de información, son de mayor tamaño que los documentos METRO. Este comportamiento se debe a que AXIS en comparación con METRO, utiliza al menos una etiqueta adicional, siendo catalogado como "más verboso".

Figura 3-5. Relación de tamaño entre documentos AXIS y METRO



Se hace necesario entonces, determinar para cada herramienta la implicación en el radio de compresión generada por el aumento en el tamaño del documento debido a la API utilizada; para ello se plantean las siguientes pruebas:

a. Prueba de Radio Compresión Basada en la Cantidad de Objetos Contenidos en el Documento

- ***Objetivo:*** Determinar cuál API presenta un mejor desempeño cuando el mensaje SOAP contiene la misma cantidad de información.

- **Descripción:** Las pruebas fueron realizadas sobre una muestra de 100 documentos SOAP que varían la cantidad de objetos contenidos de la misma forma para cada API, es decir, de 1 a 100 objetos con secuencia ascendente de 1. A esta muestra se le realiza el proceso de compresión con las 10 herramientas. Los documentos SOAP contienen igual cantidad de objetos tanto en AXIS como en METRO. Se espera que AXIS presente un radio de compresión menor por la naturaleza de su estructura, al

clasificarse como verboso; se podría pensar que los métodos de compresión actúan mejor al tener "mas información" que comprimir.

- **Resultados:** en las figuras Figura **3-6**, Figura **3-7** y Figura **3-8** se presentan los resultados arrojados al realizar la compresión con BZip2, EXificient y EXIficient+GZip respectivamente[1]. Las figuras presentan una comparación de la cantidad de objetos contenida en el documento SOAP, tanto para AXIS como para METRO, con respecto al radio de compresión hallado.

Figura 3-6. Radio Compresión para BZip2 según la cantidad de objetos

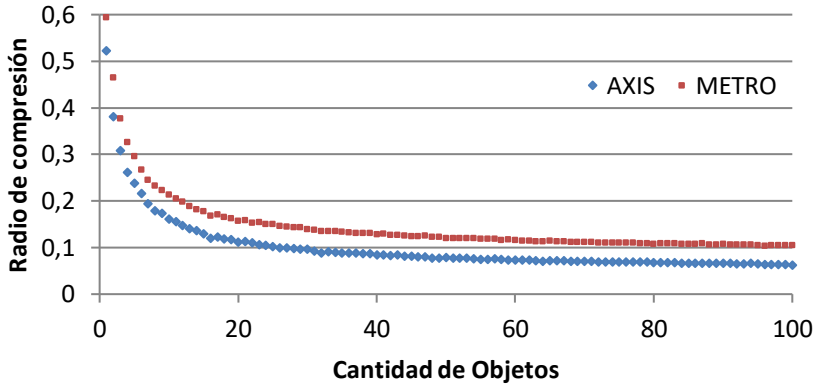
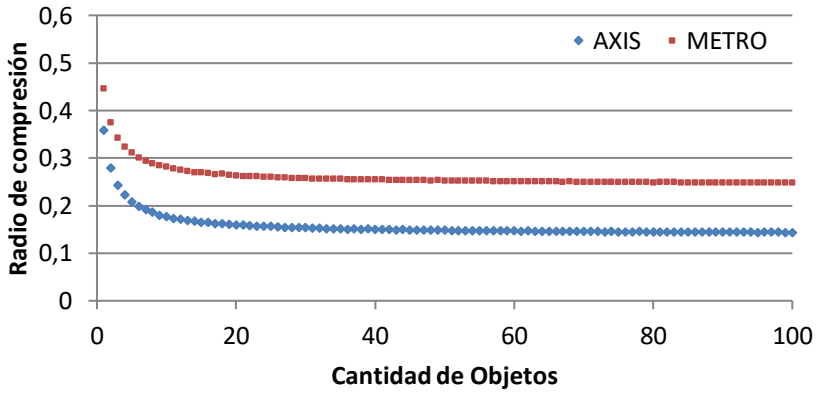
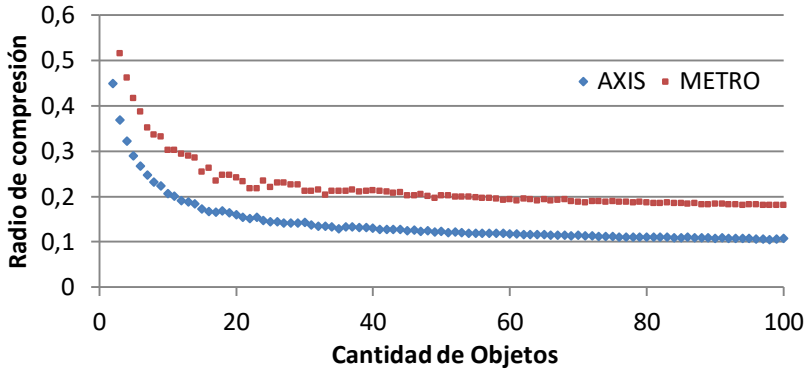


Figura 3-7. Radio Compresión para EXIficient según la cantidad del Objetos



[1] En el Anexo A se pueden encontrar las demás resultados para esta prueba.

Figura 3-8. Radio Compresión para EXIficient+GZip según la cantidad del Objetos



- **Conclusiones:** A partir de las anteriores figuras se puede concluir que AXIS presenta un mejor comportamiento en cuanto al radio de compresión cuando el mensaje contiene la misma cantidad de información.

  De acuerdo al planteamiento anterior, se podría inferir que AXIS logra un mejor desempeño; pero aun no hay claridad en su comportamiento en casos en los que el documento SOAP tenga igual tamaño en AXIS y METRO, pero que representen diferente cantidad de información. Es decir, cómo se observo en la ¡Error! No se encuentra el origen de la referencia., un documento SOAP de 1 objeto en AXIS tiene un tamaño de 840 bytes mientras que en METRO es de 623 bytes, con la anterior prueba se puede ver por ejemplo que aplicando BZip2 como método de compresión, el documento de un objeto en AXIS queda de 439 Bytes con un radio de 0,5226 y en METRO de 370 bytes con un radio de 0,5939. A pesar de que METRO entrega un documento más pequeño, el radio de compresión en Axis es más favorable. Por lo anterior, el siguiente experimento pone a prueba documentos con diferente información contenida, pero igual o aproximado tamaño en bytes, con respecto al radio de compresión.

b. Prueba de Radio Compresión Basada en el Tamaño en Bytes del Documento

- **Objetivo:** Descubrir cuál API presenta un mejor desempeño para este escenario.

- **Descripción:** Para la muestra de esta prueba se realizaron 30 documentos SOAP en AXIS y METRO, que presentan similitud en tamaño (en bytes), con diferente cantidad de objetos contenidos dentro del documento. Posteriormente, con el fin de analizar su comportamiento, la muestra es sometida a la compresión con las 10 herramientas seleccionadas.

- **Resultados:** En las figuras Figura **3-9**, Figura **3-10** y Figura **3-11** se presentan los resultados arrojados al realizar la compresión con BZip2, EXificient y

EXIficient+GZip respectivamente[1]. Las figuras presentan una comparación del tamaño de los documentos SOAP para AXIS y METRO con respecto al radio de compresión hallado.

Figura 3-9. Radio Compresión BZip2 vs tamaño

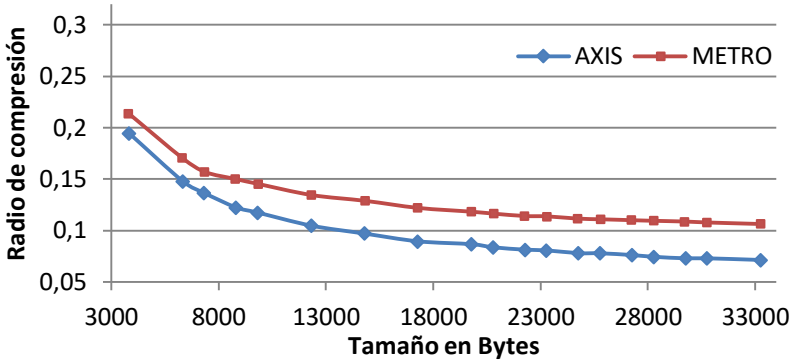


Figura 3-10. Radio Compresión EXIficient vs tamaño

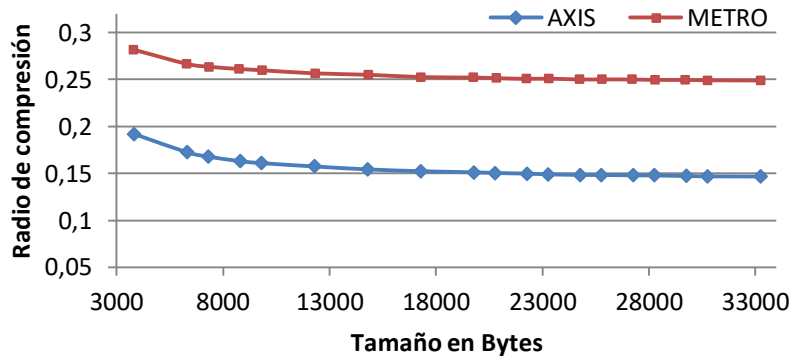
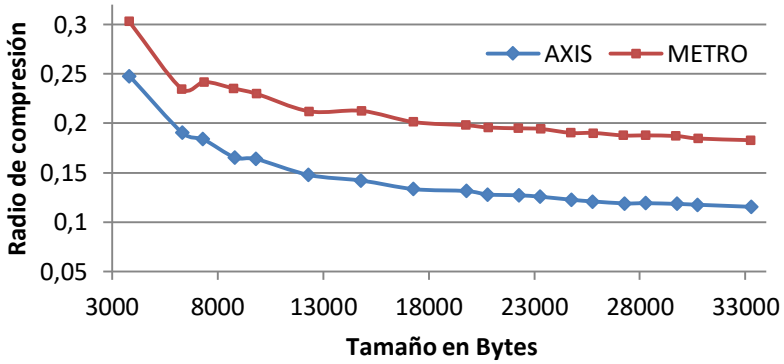


Figura 3-11. Radio Compresión EXIficient+GZip vs tamaño



[1] En el Anexo A se pueden encontrar las demás resultados para esta prueba.

- **Conclusiones:** Con esta segunda prueba, se logra corroborar que la API que presenta un mejor desempeño en cuanto al radio de compresión es AXIS y será utilizada en las pruebas subsiguientes realizadas en el proyecto.

Por lo anterior, de aquí en adelante, se adopta a la API basada en AXIS como referencia a utilizar dentro del proyecto para las siguientes pruebas.

3.3.2.3 Selección de Parámetros que Influyen en las Herramientas de Compresión

En las secciones anteriores, se realizó una comparación teórico/práctica de algunas APIs disponibles para procesamiento de servicios Web SOAP. A continuación, se busca determinar los parámetros que influyen en el proceso de compresión de datos a través de una experimentación dividida según las métricas de compresión; esto es, según el *radio de compresión*, y los *tiempos de compresión y descompresión*. En éste análisis se incluyeron características propias del documento XML SOAP como su longitud y características del equipo como su capacidad de procesamiento, cabe aclarar que si bien la memoria RAM debe ser tenida en cuenta en el análisis de las herramientas de compresión, ninguna de las herramientas estudiadas presentaba dentro de sus especificaciones un límite de memoria RAM a utilizar, por lo que su estudio no es incluido en este análisis.

3.3.2.3.1. Pruebas de Radio de Compresión

Los documentos SOAP cuentan con tres características relevantes (tipo de dato, estilo y longitud del documento), como se describe en la sección 2.1.2.2; a continuación se realiza una serie de pruebas según dichas características.

a. Tipo de Dato:

- **Objetivo de la prueba:** determinar si el tipo dato debe tenerse en cuenta como una característica relevante en el momento de seleccionar un mecanismo de compresión dado.

- **Descripción:** Se aplican las 10 herramientas de compresión a 3 grupos de respuestas SOAP de diferentes tamaños (de 1, 20 y 100 elementos contenidos en el documento). Cada grupo mantiene constante el estilo y el número de elementos dentro del documento, mientras se varía el tipo de dato que se retorna en la respuesta SOAP. Es decir, el grupo 1, consta de 4 documentos SOAP, que manejan igual estilo, igual número de elementos (1 solo en este caso), pero varia el tipo de dato retornado (Boolean, Double, Integer y String). Los grupos 2 y 3 son similares al 1 pero varían el número de elementos (20 y 100 respectivamente) contenidos en el documento SOAP. Una característica de los documentos XML es su almacenamiento como texto plano; aunque algunos compresores aprovechan la estructura de los documentos para lograr una mejor compresión, muy pocos tienen en cuenta el tipo de dato para codificar la información. Por lo tanto, se

espera que no exista variación alguna en los resultados obtenidos para documentos similares con tipo de dato distinto. Una excepción a la regla planteada, está en los compresores binarios FastInfoset y EXI, pues su funcionamiento está basado en codificar los datos de forma diferente teniendo en cuenta lo que representan, es decir, codificar un entero como entero y no como String, lo mismo para un flotante y un booleano, de esta forma se disminuye el tamaño del documento.

- **Resultados:** Las figuras Figura **3-12**,

- **Figura 3-13** y Figura **3-14** fueron realizadas utilizando los datos arrojados por la prueba de tipo de dato; en cada una, el eje horizontal representa la cantidad de objetos dentro del documento mientras que el eje vertical corresponde al radio de compresión. BZip2 se presenta como representante del grupo de compresores de uso general, XMill para el grupo de compresores conscientes XML y EXIFicient para los compresores binarios[1]; en las tres gráficas se puede apreciar que no existe una variación relevante en el radio de compresión obtenido, sin embargo, cabe destacar que en EXIFicient se encontró que para tipos de dato “double” el radio de compresión siempre es mayor, esto es, porque los tipos de dato con punto flotante siempre requieren un mayor tamaño que los enteros [110] al ser codificados.

- **Conclusiones:** considerando que el tipo de dato no genera una variación relevante en el radio de compresión, se concluye que no es necesario incluirlo como un parámetro que influya en la selección de la herramienta de compresión.

Figura 3-12. Comparación tipo de dato con BZip2

[1] En el Anexo A se pueden encontrar las demás resultados para esta prueba.

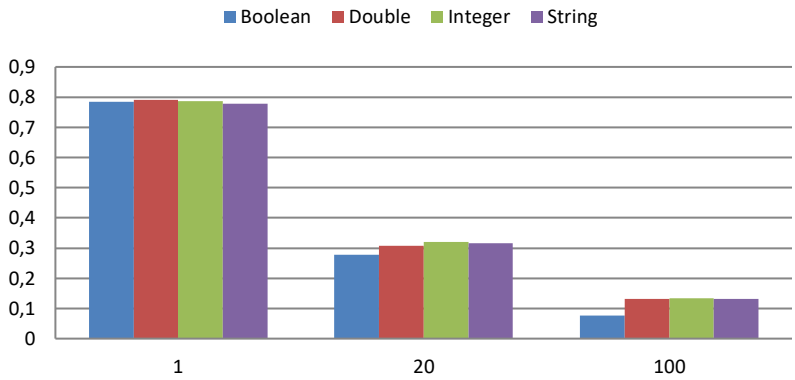


Figura 3-13. Comparación tipo de dato con XMill

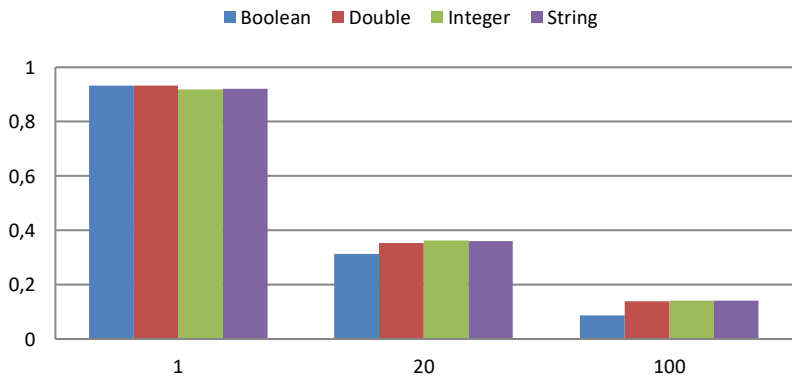
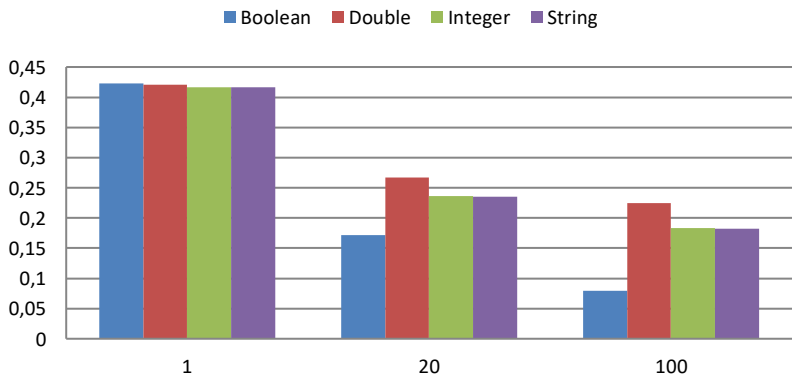


Figura 3-14. Comparación tipo de dato con EXIficient



b. **Estilo**: Existen dos estilos definidos en los mensajes SOAP denominados RPC y Document. La diferencia que a simple vista se puede encontrar entre ellos, es la variación en la profundidad del mensaje, RPC siempre tendrá al menos una etiqueta más que Document [27].

- **Objetivo de la prueba:** determinar si el estilo del documento debe tenerse en cuenta en el momento de seleccionar un mecanismo de compresión dado.

- **Descripción:** al igual que en la prueba anterior, ésta consta de una muestra dividida en 3 grupos como se explicó en el inciso a, pero ahora se analizan respuestas SOAP, variando únicamente el estilo. De esta forma, se aplican las 10 herramientas de compresión a la muestra y se comparan los radios de compresión. Como ya se mencionó la principal diferencia entre los dos estilos es la profundidad en un nivel muy leve, por lo que se espera que el estilo no afecte de forma relevante el radio de compresión.

- **Resultados:** Las figuras Figura **3-15**, Figura **3-16** y Figura **3-17** [1] fueron realizadas empleando respuestas SOAP idénticas pero utilizando diferentes estilos. En ellas se puede notar que no existe una variación relevante del radio de compresión cuando se utiliza estilo *Document* (Rojo) y RPC (Azul).

Figura 3-15. Comparación de estilo con BZip2

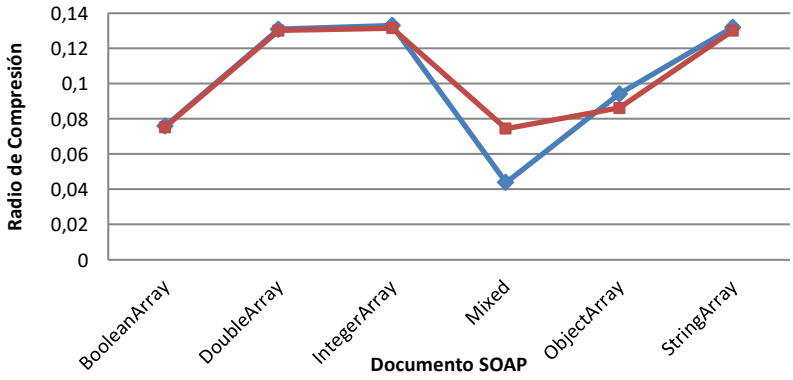


Figura 3-16. Comparación de estilo con EXIficient

[1] En el Anexo A se pueden encontrar las demás resultados para esta prueba.

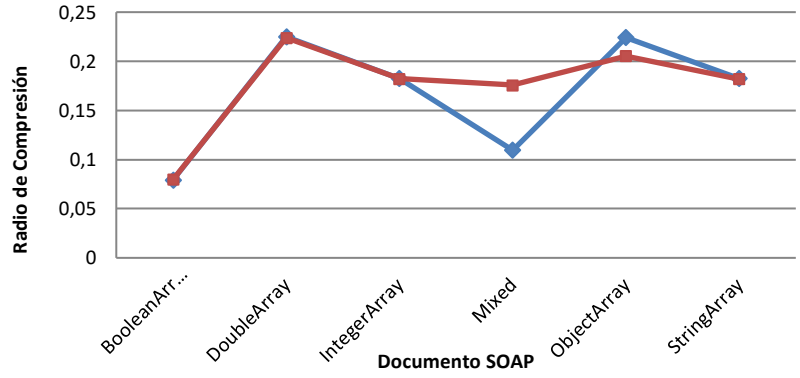
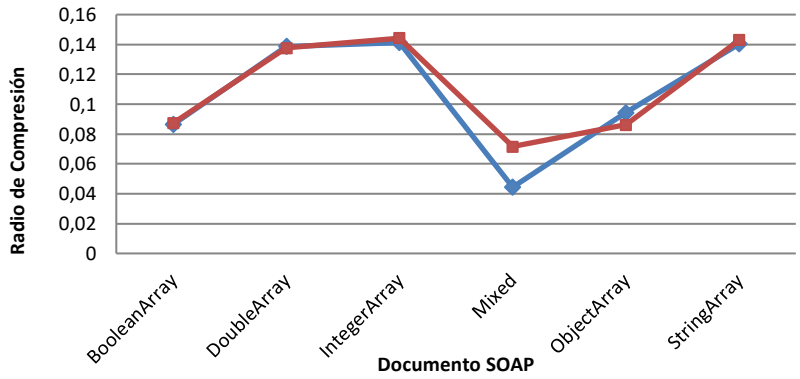


Figura 3-17. Comparación de estilo con XMill



- **Conclusiones:** el estilo no es una característica relevante en el momento de escoger una herramienta de compresión, y por tanto, no requiere tenerse en cuenta para la selección de la misma.

c. **Longitud del documento:** No existe una medida máxima para los documentos SOAP, estos pueden variar desde unos pocos bytes, hasta una cantidad apreciable de megabytes dependiendo la información que arroje el servicio Web. En los documentos XML comunes, el tamaño refiriéndose a la cantidad de bytes que contiene, puede afectar de manera significativa en el radio de compresión de los mecanismos [40], de esta forma se analizará si sucede lo mismo con los documentos SOAP.

- **Objetivo de la prueba:** considerar, si es necesaria, la longitud del documento como un parámetro que afecta la selección de la herramienta de compresión a utilizar y evaluar el comportamiento del radio de compresión cuando se varía la longitud del documento.

- **Descripción:** Esta prueba consiste en obtener diferentes documentos que varían la cantidad de objetos contenidos y por tanto el tamaño del mensaje; posteriormente

se aplican los diferentes mecanismos de compresión a cada documento y se obtiene el comportamiento del radio de compresión de acuerdo a la variación del tamaño del estos. Los mensajes SOAP son documentos XML, por tanto se espera que la variación del tamaño del mensaje afecte en el radio de compresión de todos los mecanismos.

- **Resultados:** Se realizaron pruebas con arreglos de objetos en tres muestras diferentes. Con el fin de analizar a fondo la selección de muestras y los resultados obtenidos, se toma como ejemplo al compresor GZip. El análisis de los demás compresores se encuentra en el Anexo A.

  La primera muestra contiene 44 documentos XML SOAP desde 480KB hasta 20.4MB, estos documentos se obtuvieron con arreglos de 1 hasta 43000 objetos, en secuencia creciente de 1000 objetos. Al realizar la compresión de esta muestra con GZip como se ve en la Figura 3-18, el comportamiento a partir de cierto punto posee una tendencia lineal, aproximadamente a partir del punto 499157 Bytes (puntos en rojo en la figura); además se puede observar que en los rangos pequeños la variación de la curva es significativa, por lo que se plantea una segunda muestra de 100 documentos entre 840 Bytes y 52KB, obtenidos con arreglos de objetos desde 1 hasta 100, en secuencia creciente de 1 objeto; los resultados de esta compresión se pueden ver en la Figura 3-19. Finalmente se realiza una tercera muestra de 200 documentos entre 54KB y 1.024MB, obtenidos con arreglos de objetos desde 110 hasta 2100 en secuencia creciente de 10 objetos, cuyos resultados se presentan en la Figura 3-20. Con las tres muestras se puede determinar claramente el comportamiento de los mecanismos de compresión.

Figura 3-18. Compresión Gzip de 1 - 43000 Objetos

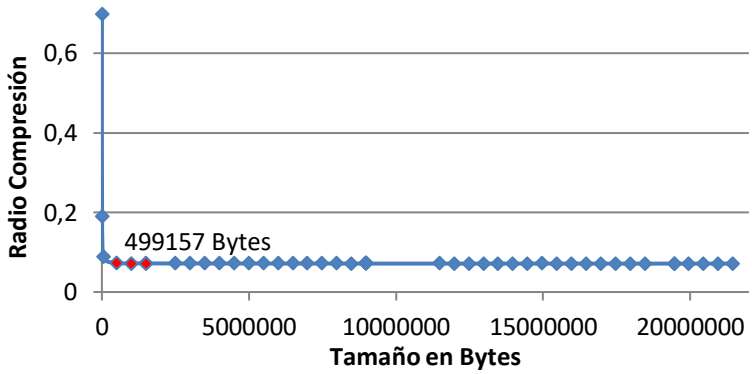


Figura 3-19. Compresión Gzip de 1 a 100 Objetos

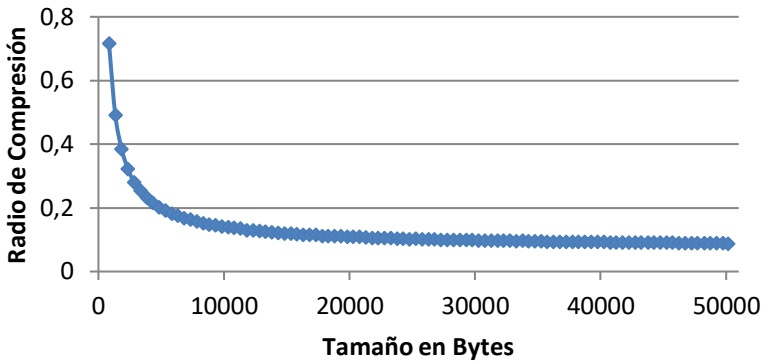
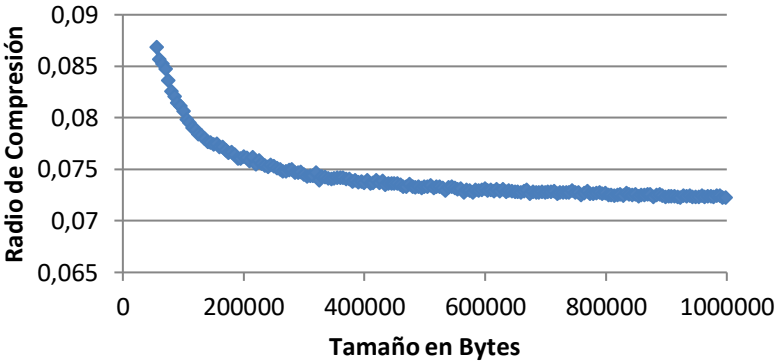
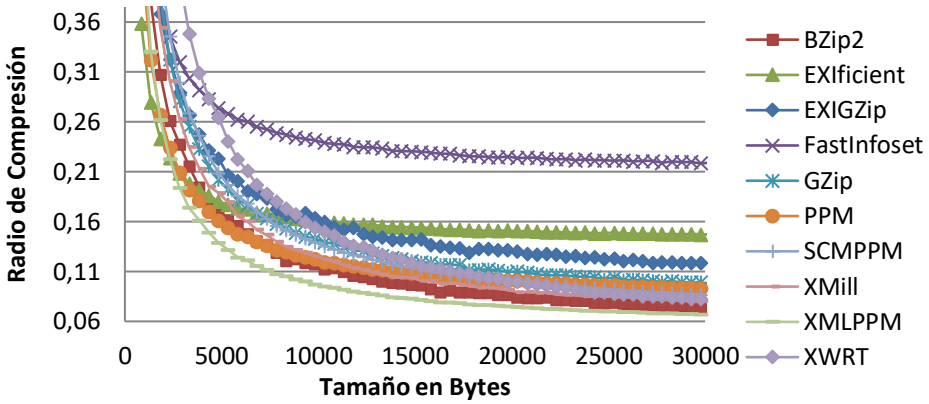


Figura 3-20. Compresión Gzip de 110 - 2100 Objetos



- **Conclusiones:** A partir de los resultados de las tres muestras, se observa el comportamiento variable del radio de compresión según el tamaño del documento; en la Figura 3-21, se aprecia el comportamiento general de todos los compresores para la muestra 2; de esta manera se puede deducir que el tamaño del documento es un factor que afecta el comportamiento de la herramienta de compresión. Por lo tanto, se concluye que el tamaño del documento es una característica relevante que influye en la selección del mecanismo de compresión a utilizar en el mediador.

Figura 3-21. Compresión AXIS



3.3.2.3.2. Tiempo de compresión y descompresión

El tiempo de compresión y descompresión ha sido definido como el intervalo de tiempo desde el inicio de una compresión o descompresión hasta su finalización exitosa. En [40] se muestra que la longitud en bytes del documento XML normal y la capacidad de procesamiento del equipo donde se efectúen los procesos, afectan el comportamiento del tiempo de compresión y descompresión. Es entonces interesante para este proyecto evaluar dicho comportamiento, dependiente de la longitud y de la capacidad de procesamiento, pero teniendo en cuenta documentos XML SOAP; por lo anterior, se desarrollaron dos pruebas, explicadas a continuación.

a. Tiempo de Compresión y Descompresión vs Longitud del Documento

- **Objetivo de la prueba:** determinar el comportamiento del tiempo de compresión y descompresión para cada mecanismo cuando se varía la longitud del documento SOAP XML.

- **Descripción:** Variar la longitud del documento; aplicar los diferentes mecanismos de compresión, manteniendo constante la capacidad de procesamiento del equipo y obtener los tiempos de compresión y descompresión. Se espera que el tiempo de compresión y descompresión aumente a medida que la longitud del documento también lo hace, se desconoce cuál puede ser patrón de crecimiento.

- **Resultados:** A partir del servicio Web AxisVectorsWS[1] y la operación getObjectArray se obtuvo un grupo de documentos SOAP XML cuyas longitudes varían entre 1 y 3500 objetos. Al conjunto de documentos generados les fueron aplicadas las diferentes herramientas de compresión y se tomaron los datos de tiempo de compresión, las figuras Figura **3-22** y Figura **3-23** presentan los resultados obtenidos para el compresor BZip2 utilizando un procesador de 128MHz.

[1] En el Anexo A se encuentran descritos los servicios Web y operaciones utilizadas dentro de los entornos de pruebas.

Figura 3-22. BZip2: Tiempo de Compresión vs Longitud del documento

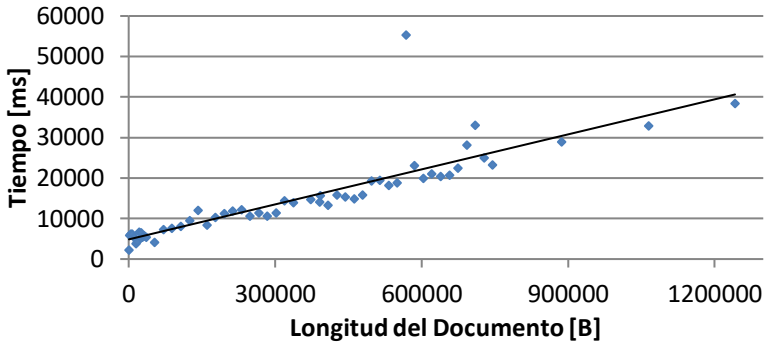
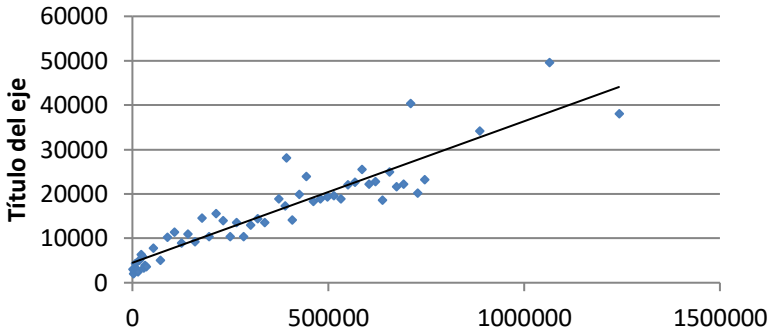


Figura 3-23. BZip2: Tiempo de Descompresión vs Longitud del documento



Las líneas negras ubicadas sobre las graficas representan la curva de tendencia de los puntos; se puede apreciar que ambas gráficas siguen una tendencia lineal creciente.

- **Conclusiones:** se puede asegurar que el tiempo de compresión y descompresión es directamente proporcional a la longitud del documento, puesto que la grafica de tiempo contra longitud del documento describe una tendencia lineal creciente.

b. Tiempo de Compresión y Descompresión vs Capacidad de Procesamiento

- **Objetivo de la prueba:** determinar el comportamiento de la curva de tiempo de compresión y descompresión en contra de la longitud del documento cuando se varía la capacidad de procesamiento del equipo.

- **Descripción:** Variar la capacidad de procesamiento, aplicar los diferentes mecanismos de compresión al tiempo que se varía la longitud de los documentos, obtener los tiempos de compresión y descompresión. Se espera encontrar alguna alteración en las curvas de tiempo de compresión y descompresión contra longitud del documento, un cambio posible es un desplazamiento sobre el eje del tiempo, puesto que al limitar la capacidad de procesamiento el principal efecto se observa sobre el tiempo necesario para realizar los procesos.

En la prueba anterior se determinó que los tiempos de compresión y descompresión describen una tendencia en línea recta cuando se varía la longitud del documento. Ahora el objetivo es definir cómo es el comportamiento de aquella curva cuando se varía la capacidad de procesamiento del equipo. Para realizar las variaciones de procesamiento se utilizó la herramienta CPULimit [111], esta herramienta para sistemas operativos Linux, permite limitar la capacidad de procesamiento que se asigna a un proceso en específico. Para su funcionamiento recibe como parámetro de entrada el porcentaje de procesamiento que se quiere asignar y el identificador del proceso, dicho porcentaje es tal que el 100% indica la velocidad total de uno de los núcleos del equipo, es decir, si el equipo cuenta con 2 núcleos de procesamiento y se quiere que el proceso los utilice ambos, el porcentaje debe ser 200%.

- **Resultados:** Considerando que el equipo donde se ejecuta el mediador requiere de un sistema operativo completo en el que además de las herramientas de compresión, se encuentran corriendo otros procesos como los encargados de la interfaz grafica del sistema, del monitoreo, funciones de red, entre otros; es lógico pensar que limitar simplemente el procesamiento de la herramienta de compresión puede dar como resultado una medida errónea del tiempo de compresión y descompresión pues se pasaría por alto el consumo de recursos por parte de los demás procesos. Para simular el entorno de una manera mucho más real, se utilizó el software virtualbox[1] [112] para crear una máquina virtual en la que se instaló el sistema operativo Ubuntu 11.04 con escritorio Xfce [113][2] (Xubuntu 11.04) y dentro de ella se instalaron las herramientas de compresión. La maquina virtual fue configurada para utilizar todas las capacidades de procesamiento del equipo anfitrión y se utilizó CPULimit para limitar su procesamiento. Los porcentajes asignados fueron 6.4%, 12.5%, 18.75% y 25%, dando como resultados procesadores de 0.128GHz, 0.25GHz, 0.375GHz y 0.5GHz respectivamente; los anteriores porcentajes fueron seleccionados con el ánimo de lograr capacidades de procesamiento bajas.

Figura 3-24. BZip2: Tiempo de compresión variando la capacidad de procesamiento

[1] Virtualbox es un software de virtualización que permite emular todos los componentes hardware de una máquina para ejecutar sistemas operativos completos.
[2] Xfce es un entorno de escritorio ligero para sistemas tipo UNIX. Su objetivo es ser rápido y de bajos recursos del sistema.

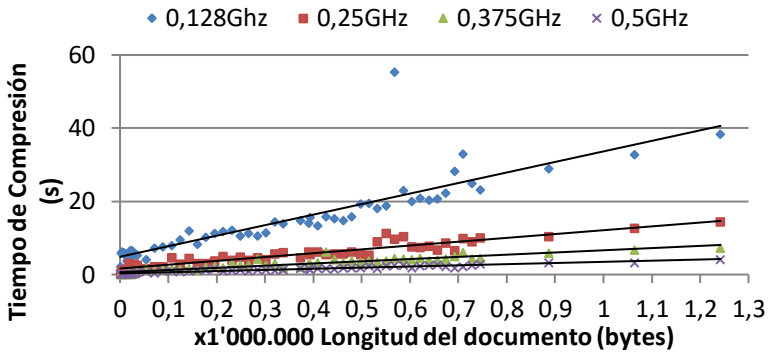
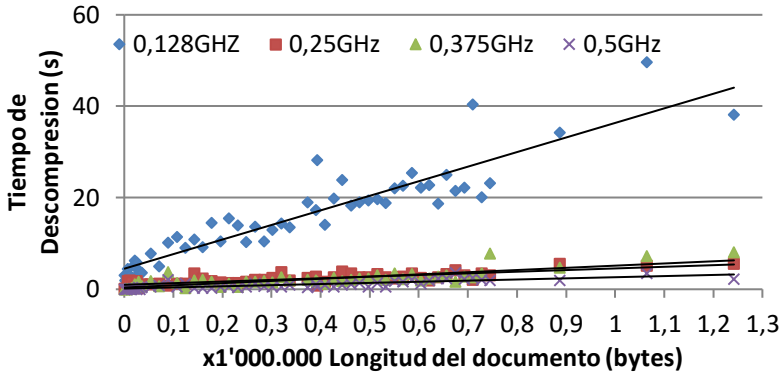


Figura 3-25. BZip2: Tiempo de descompresión variando la capacidad de procesamiento



- **Conclusiones:** En las figuras Figura **3-24** y
- **Figura 3-25** se puede apreciar que una variación en la capacidad de procesamiento del equipo resulta en una variación en la inclinación y en el corte con el eje vertical de la línea de tiempo contra longitud, a partir de estos resultados es posible concluir que la capacidad de procesamiento influye de manera relevante en el tiempo de compresión y descompresión de un documento dado.

Como conclusión final de las pruebas con las diferentes herramientas de compresión, se puede decir que: El tipo de dato y el estilo contenido dentro del mensaje SOAP, no son parámetros que influyen en la selección de las herramientas de compresión; mientras que la longitud del documento y la capacidad de procesamiento, si son parámetros que afectan el radio de compresión y el tiempo de compresión y descompresión, respectivamente.

En la siguiente fase se determina un modelo matemático que tomando como entrada los parámetros influyentes, pueda describir el comportamiento de las métricas para cada una de las herramientas de compresión. Este modelo matemático representa entonces una regla para seleccionar la mejor herramienta de compresión de acuerdo a los parámetros

que describen el entorno donde se consume el servicio Web. A continuación se describe el desarrollo de los modelos matemáticos necesarios.

3.3.3 FASE III. Generación de Modelos Matemáticos que Permitan Describir el Comportamiento de las Métricas de Evaluación de las Herramientas de Compresión

Al inicio de la sección 3.3 se explicó que un mecanismo de compresión puede ser descrito mediante tres métricas: radio de compresión, tiempo de compresión y tiempo de descompresión; además se determinó que la selección del mejor mecanismo de compresión se puede realizar a través del menor tiempo de respuesta, que además de algunas características propias de la red y del servicio Web, depende de los tiempos de transmisión, compresión y descompresión. Siguiendo ese orden de ideas el tiempo de respuesta puede ser expresado como:

Ecuación 3-1. Tiempo de Respuesta para el acceso a un servicio Web

$$T_R = t_{Tx} + t_C + t_D + C$$

Donde t_D hace referencia al tiempo de descompresión, t_C al tiempo de compresión, t_{Tx} representa el tiempo necesario para la transmisión de información entre el Cliente y el Servidor; C es un valor que incluye los demás tiempos ajenos a las herramientas de compresión y se consideran constantes para entornos iguales; es decir C será igual en escenarios donde se tenga el ancho de banda y la capacidad de procesamiento fijos, y solo se varíe la herramienta de compresión aplicada. Los dos primeros tiempos son las métricas de tiempo de descompresión y compresión respectivamente, el tiempo de transmisión por otro lado depende del radio de compresión y del ancho de banda disponible para la transmisión; de forma que,

Ecuación 3-2. Tiempo de transmisión de un documento comprimido entre el Cliente y el Servidor

$$t_{Tx} = \frac{r_c L}{BW}$$

Donde L es la longitud en bytes del documento y BW el ancho de banda en bytes/s. El objetivo del plan de pruebas propuesto era determinar qué características, tanto del documento XML SOAP como del equipo donde se ejecuta la herramienta de compresión, afectan de manera relevante estas tres métricas. De esta manera, se concluyo que el tipo de dato y el estilo utilizado no afectan significativamente en el radio de compresión, mientras que la longitud del documento y la capacidad de procesamiento del equipo donde se ejecuta la herramienta si influyen en el radio de compresión y en los tiempos de compresión y descompresión; por lo tanto se ve la posibilidad de analizar cada uno de dichos parámetros con el objetivo de obtener modelos matemáticos que permitan predecir las métricas de los compresores y finalmente obtener un valor teórico del tiempo de respuesta.

3.3.3.1 Modelo Matemático para la Función de Radio de Compresión

De la Figura 3-21 se puede apreciar que el radio de compresión respecto al tamaño del documento tiene un comportamiento no lineal, que se ve afectado por las variaciones de tamaño. De esta manera, matemáticamente es posible expresar estos valores como una función que depende de la longitud.

Ecuación 3-3. Radio de compresión en función de la longitud del documento

$$r_C = f(L)$$

Donde nuevamente L representa la longitud del documento. El comportamiento del radio de compresión difiere según el mecanismo de compresión, por ello es necesario realizar un análisis individual para cada uno. A continuación se describe la obtención de la función del radio de compresión mediante un análisis de regresión matemática, tomando como ejemplo al mecanismo GZip; la función de los demás compresores se explican en el *Anexo B.*

3.3.3.1.1. Regresión Matemática para la Función de Radio de Compresión del Compresor GZip

Dado el comportamiento de la curva de Radio de compresión vs. tamaño del documento, no es posible obtener una única regresión matemática que la describa. La herramienta StatGraphics [1] [114] fue utilizada para realizar las regresiones y en ella se consideraron condiciones como: coeficiente de correlación próximo a l1l, error estándar mínimo, alto porcentaje de relación con la representación matemática y alto porcentaje de confianza del modelo matemático entregado por la herramienta, con el fin de determinar los rangos adecuados donde se debe crear una función. Es decir, sin realizar análisis de coeficiente de correlación, ni error estándar o porcentajes de relación, la regresión que Statgraphics entrega es la que se muestra en la Figura 3-26, donde el coeficiente de correlación es bueno pero el porcentaje de aproximación y el error estándar no.

Coeficiente de Correlación = 0,897337
R-cuadrada = 80,5214 porciento
Error estándar del est. = 0,131948

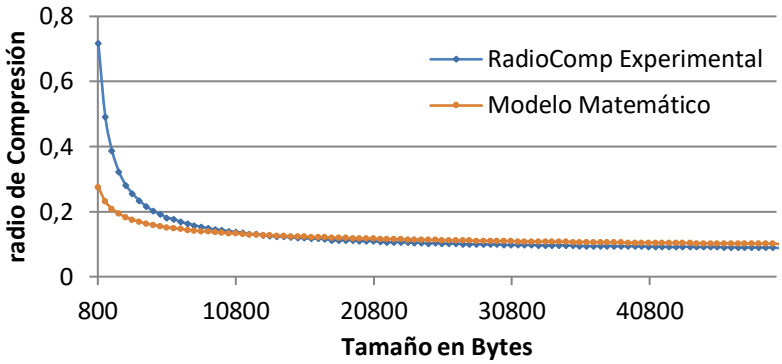
Ecuación del modelo ajustado:

$$r_c = e^{-3{,}94536 + \frac{17{,}8678}{\mathrm{Ln(L)}}}$$

Las expresiones matemáticas presentadas anteriormente y a continuación son resultado de la aplicación de la herramienta Statgraphics, la cual para obtenerlas, realiza una relación de los datos experimentales con los datos que se pueden hallar con la formula que más se acerque al valor experimental.

Figura 3-26. Aproximación general del radio de compresión para GZip

[1] StatGraphics es una herramienta de análisis de datos que combina una amplia gama de procedimientos analíticos. Aunque es una herramienta de pago, existe una versión demo completamente funcional por 30 días.



Por lo anterior es necesario realizar un modelo de regresión matemática acorde al comportamiento de la curva, siendo la base mejorar los factores de relación. Así, analizando el coeficiente de correlación se obtienen cuatro modelos matemáticos para GZip:

Modelo Matemático 1 GZip:
Este modelo se aplica para documentos entre [0 – 10321] bytes.

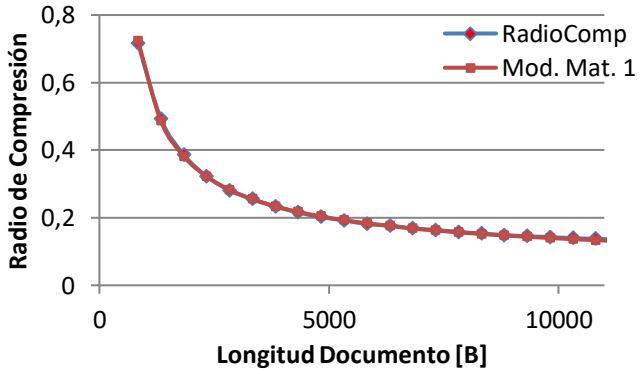
Factores de relación:

Coeficiente de Correlación = 0,999822
R-cuadrada = 99,9644 porciento
Error estándar del est. = 0,00867885
Error absoluto medio = 0,00708868
Nivel de Confianza = 95,00%

Ecuación del modelo ajustado:

$$r_c = e^{-6{,}47081 + \frac{41{,}4004}{\ln(L)}}$$

Figura 3-27. Modelo matemático 1 para GZip



Modelo Matemático 2 GZip:
Este modelo se aplica para documentos entre [10321 – 15808] bytes.

Factores de relación:

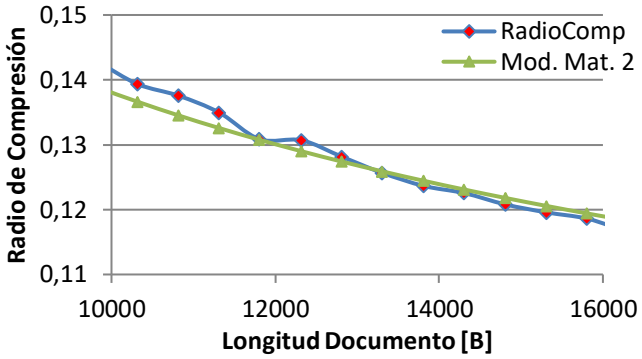
Figura 3-28. Modelo matemático 2 para GZip

Coeficiente de Correlación = 0,994615
R-cuadrada = 98,9258 porciento

Error estándar del est. = 0,0146264
Error absoluto medio = 0,0121229
Nivel de Confianza = 94,00%

Ecuación del modelo ajustado:

$$r_c = e^{-5{,}04454 + \frac{28{,}2243}{\ln(L)}}$$



Modelo Matemático 3 GZip:
Este modelo se aplica para documentos entre [15808 - 1047782] bytes.

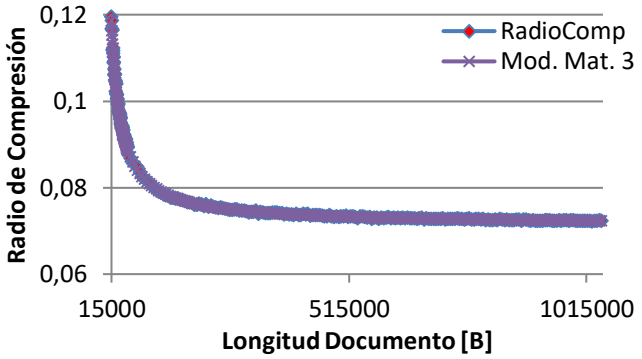
Factores de relación:

Coeficiente de Correlación = 0,998637
R-cuadrada = 99,7276 porciento
Error estándar del est. = 0,0000232671
Error absoluto medio = 0,0000166154
Nivel de Confianza = 95,00%

Ecuación del modelo ajustado:

$$r_c = \sqrt{0{,}00510421 + \frac{137{,}164}{L}}$$

Figura 3-29. Modelo matemático 3 para GZip



Modelo Matemático 4 GZip:
Este modelo se aplica para documentos superior a 1047782 bytes.

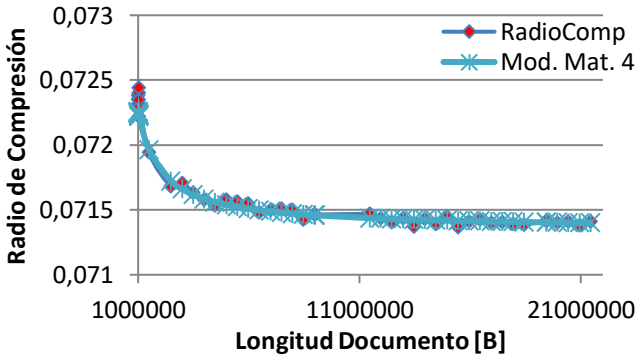
Factores de relación:

Coeficiente de Correlación = 0,974337
R-cuadrada = 94,9332 porciento
Error estándar del est. = 0,00000363894
Error absoluto medio = 0,00000288772
Nivel de Confianza = 91,00%

Ecuación del modelo ajustado:

$$r_c = \sqrt{0{,}00509182 + \frac{129{,}573}{L}}$$

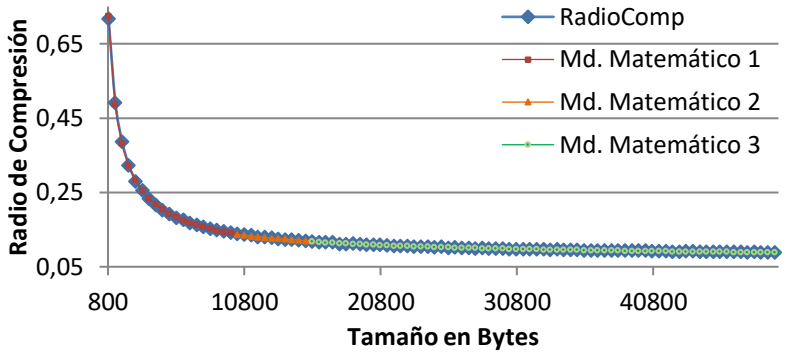
Figura 3-30. Modelo matemático 4 para GZip



De esta forma el modelo matemático ajustado para el mecanismo de compresión de GZip para documentos entre los 0 y 50Kbytes, como se muestra en la Figura 3-31 se aproxima en promedio en un 98.39% a los datos experimentales obtenidos en las pruebas (Este

porcentaje se calcula tomando el promedio entre los valores R-cuadrada determinados para los modelos 1-4).

Figura 3-31. Modelo matemático total para GZip



A continuación en la Tabla 3-3 se presenta un resumen de los modelos matemáticos obtenidos para cada mecanismo de compresión siguiendo la metodología anterior, aplicada en GZip.

Tabla 3-3. Modelos matemáticos para la función de radio de compresión de los mecanismos de compresión

No.	Mecanismo Compresión	Ecuación	Rango		Coeficiente Correlación
			Inicial	Final	
1.	BZIP2	$r_c = e^{-6,45237+\frac{39,5594}{\ln(L)}}$	0	31772	0,999125
		$r_c = e^{-5,17384+\frac{26,2972}{\ln(L)}}$	31772	60198	0,997063
		$r_c = \sqrt{0,00179997+\frac{121,72}{L}}$	60198	125061	0,997158
		$r_c = e^{-5,17384+\frac{26,2972}{\ln(L)}}$	125061	179925	0,997063
		$r_c = e^{-4,21636+\frac{14,6926}{\ln(L)}}$	179925	593956	0,996877
		$r_c = e^{-2,78204-(0,00184184)*(\ln(L))^2}$	593956	1032838	-0,92277
		$r_c = \sqrt{0,00187242+\frac{99,9059}{L}}$	1032838	Mayores	0,936835
2.	EXIficient	$r_c = 0,141201+\frac{185,825}{L}$	0	7821	0,99957
		$r_c = e^{-2,82724+\frac{9,20029}{\ln(L)}}$	7821	16801	0,996203
		$r_c = 0,141201+\frac{185,825}{L}$	16801	27280	0,99957
		$r_c = e^{-2,38074+\frac{4,79846}{\ln(L)}}$	27280	60198	0,985495
		$r_c = e^{-2,10806+\frac{1,79158}{\ln(L)}}$	60198	2494420	0,988689
		$r_c = e^{-1,65621-(0,0015374)*(ln(L))^2}$	2494420	Mayores	-0,991145
3.	EXIficient + GZip	$r_c = e^{-5,48253+\frac{33,6888}{\ln(L)}}$	0	14304	0,999076
		$r_c = e^{-4,40795+\frac{23,4136}{\ln(L)}}$	14304	55206	0,996553

		$r_c = \frac{1}{-5{,}58246 + 1{,}39211 * \ln(\mathrm{L})}$	55206	194880	0,997601
		$r_c = \sqrt{0{,}00580793 + \frac{379{,}141}{L}}$	194880	3492110	0,997024
		$r_c = \sqrt{0{,}00876322 - 0{,}000188324 * \ln(L)}$	3492110	Mayores	-0,994367
4.	**Fast Infoset**	$r_c = 0{,}208664 + \frac{313{,}115}{L}$	0	60198	0,999852
		$r_c = e^{-1{,}69207 + \frac{1{,}55607}{\ln(L)}}$	60198	2494420	0,941014
		$r_c = e^{-1{,}34135 - (0{,}0011381) * (\ln(L))^2}$	2494420	Mayores	-0,981827
5.	**GZip**	$r_c = e^{-6{,}47081 + \frac{41{,}4004}{\ln(L)}}$	0	10321	0,999822
		$r_c = e^{-5{,}04454 + \frac{28{,}2243}{\ln(L)}}$	10321	15808	0,994615
		$r_c = \sqrt{0{,}00510421 + \frac{137{,}164}{L}}$	15808	1047782	0,998637
		$r_c = \sqrt{0{,}00509182 + \frac{129{,}573}{L}}$	1047782	Mayores	0,974337
6.	**PPM**	$r_c = \sqrt{0{,}00463428 + \frac{135{,}552}{L}}$	0	1841	0,986597
		$r_c = \sqrt{0{,}00540506 + \frac{106{,}14}{L}}$	1841	4333	0,998088
		$r_c = \frac{1}{-13{,}6401 + 2{,}36067 * \ln(L)}$	4333	38737	0,996126
		$r_c = \sqrt{0{,}00463428 + \frac{135{,}552}{L}}$	38737	49727	0,986597
		$r_c = e^{-3{,}35706 + \frac{9{,}67345}{\ln(L)}}$	49727	184884	0,997117
		$r_c = \sqrt{0{,}00540506 + \frac{106{,}14}{L}}$	184884	294650	0,998088
		$r_c = \sqrt{0{,}00526645 + \frac{146{,}65}{L}}$	294650	2494420	0,997994
		$r_c = \frac{1}{12{,}2558 + 0{,}108406 * \ln(L)}$	49727	Mayores	0,996643
7.	**SCMPPM**	$r_c = e^{-6{,}47713 + \frac{41{,}6091}{\ln(L)}}$	0	13306	0,998622
		$r_c = \sqrt{0{,}00282591 + \frac{142{,}828}{L}}$	13306	1047782	0,998595
		$r_c = e^{-3{,}22108 + \frac{4{,}10331}{\ln(L)}}$	1047782	Mayores	0,994517
8..	**XMILL**	$r_c = e^{-6{,}40885 + \frac{39{,}9948}{\ln(L)}}$	0	29769	0,998519
		$r_c = \frac{1}{-25{,}4787 + 3{,}68593 * \ln(L)}$	29769	44723	0,99427
		$r_c = \sqrt{0{,}00160581 + \frac{156{,}184}{L}}$	44723	75158	0,998975

		$r_c = e^{-5,6098+\frac{31,5231}{\ln(L)}}$	75158	100091	0,995936
		$r_c = \sqrt{0,00160581 + \frac{156,184}{L}}$	100091	159950	0,998975
		$r_c = e^{-5,6098+\frac{31,5231}{\ln(L)}}$	159950	204858	0,995936
		$r_c = \sqrt{0,00160581 + \frac{156,184}{L}}$	204858	519120	0,998975
		$r_c = e^{-3,96103+\frac{10,9255}{\ln(L)}}$	519120	1027824	0,997066
		$r_c = \sqrt{0,00160581 + \frac{156,184}{L}}$	1027824	1496767	0,998975
		$r_c = -2,50724 - 0,18104 * \sqrt{\ln(L)}$	1496767	Mayores	-0,996331
9.	**XMLPPM**	$r_c = e^{-6,74484+\frac{40,5473}{\ln(L)}}$	0	10817	0,999799
		$r_c = \frac{1}{-26,0171 + 3,96487 * \ln(L)}$	10817	15314	0,997733
		$r_c = e^{-5,34537+\frac{27,2754}{\ln(L)}}$	15314	55206	0,996766
		$r_c = e^{-4,09154+\frac{13,6122}{\ln(L)}}$	55206	259755	0,995924
		$r_c = e^{-3,5422+\frac{6,75834}{\ln(L)}}$	259755	1047782	0,99495
		$r_c = \frac{1}{-21,9501 - \frac{768670}{L}}$	1047782	Mayores	-0,950685
10.	**XWRT**	$r_c = 0,051181 + \frac{967,91}{L}$	0	4330	0,999448
		$r_c = e^{-7,68775+\frac{53,5237}{\ln(L)}}$	4330	19295	0,999345
		$r_c = 0,051181 + \frac{967,91}{L}$	19295	26770	0,999448
		$r_c = e^{-7,68775+\frac{53,5237}{\ln(L)}}$	26770	45731	0,999345
		$r_c = \sqrt{0,00164257 + \frac{131,267}{L}}$	45731	55206	0,984861
		$r_c = 0,0409164 + \frac{1321,36}{L}$	55206	558974	0,999579
		$r_c = \sqrt{0,00164257 + \frac{131,267}{L}}$	558974	648844	0,984861
		$r_c = 0,0409164 + \frac{1321,36}{L}$	648844	863345	0,999579
		$r_c = \sqrt{0,00164257 + \frac{131,267}{L}}$	863345	Mayores	0,984861

3.3.3.2 Modelo Matemático para la Función de Tiempo de Compresión y Descompresión

Tanto el tiempo de compresión como el tiempo de descompresión tienen un comportamiento lineal creciente cuando se aumenta la longitud del documento; además dicho comportamiento se ve afectado al modificar la capacidad de procesamiento, de tal forma que, matemáticamente es posible expresar estos valores como funciones que dependen de la longitud y de la capacidad de procesamiento,

Ecuación 3-4. Tiempo de compresión en función de L y P

$$t_C = f(L, P)$$

Ecuación 3-5. Tiempo de descompresión en función de L y P

$$t_D = f(L, P)$$

L nuevamente representa la longitud del documento y P la capacidad de procesamiento. A continuación se realizara la obtención de la función para el tiempo de compresión mediante un análisis de regresión matemática, tomando como referencia la herramienta de compresión BZip2.

3.3.3.2.1. Tiempo de Compresión vs Longitud del documento para BZip2

a. Capacidad de Procesamiento Constante

Si la capacidad de procesamiento se mantiene constante el tiempo de compresión puede ser expresado mediante la ecuación de una línea recta que depende de la longitud, y la cual será llamada ecuación de regresión:

Ecuación 3-6. Ecuación de regresión para el tiempo de compresión

$$t_C = mL + b$$

Donde t_C es la variable dependiente y L es la variable independiente. A partir del software estadístico StatGraphics, utilizando la opción de regresión lineal simple se obtienen los siguientes datos:

Procesamiento = 0.128GHz	
Coeficiente de Correlación =	0.85875
R^2 =	0.73745
R^2(ajustado) =	0.73250
Error Estándar =	5121.4547
Ordenada (b) =	4852.08935
Pendiente (m) =	0.0288
P-Valor =	0.0000

La ecuación que describe el comportamiento de las variables es, por tanto,

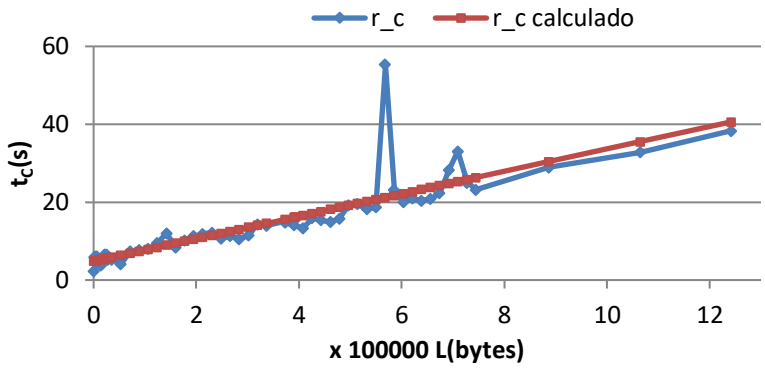
Ecuación 3-7. Tiempo de compresión para BZip2

$$t_C = 0{,}0288L + 4852{,}08935$$

El coeficiente de correlación es muy cercano a 1 indicando que existe una gran dependencia entre t_C y L, y puesto que es mayor que 0, sabemos que existe una relación directamente proporcional. El valor R^2 indica que el modelo explica un 73.6452% de la variabilidad de t_C. En cuanto al error estándar, éste muestra que existe una desviación típica de 5121.45ms. Finalmente dado que el P-Valor es menor que 0.01, existe una

relación estadísticamente significativa entre t_C y L para un nivel de confianza de 99% en el modelo. La Figura 3-32 presenta una comparación entre los valores medidos y los calculados a partir del modelo obtenido.

Figura 3-32. Valores medidos y valores calculados del radio de compresión de BZip2



b. Capacidad de Procesamiento Variable

Una variación en la capacidad de procesamiento resulta en una variación en la pendiente y ordenada de la curva de t_C respecto a L, por tanto la ecuación de t_C puede ser expresada como:

Ecuación 3-8. Tiempo de compresión con capacidad de procesamiento variable

$$t_C = M(P)L + B(P)$$

Donde M(P) y B(P) son funciones que dependen del valor de P. La Tabla 3-4 presenta una tabulación de valores para el compresor BZip2 cuando se varía la capacidad de procesamiento.

Tabla 3-4. Pendiente y Ordenada de acuerdo al procesamiento

Procesamiento (GHz)	ordenada	pendiente
0.12800	4852.08935	0.02880
0.25000	1642.44575	0.01047
0.37500	627.52745	0.00600
0.50000	307.92364	0.00316

Las figuras Figura 3-33 y Figura 3-34 permiten apreciar que las curvas de M(P) y B(P) describen un comportamiento exponencial de la forma,

Ecuación 3-9. Comportamiento exponencial de M(P) y B(P)

$$Y = aX^d$$

Por tanto, este par de funciones pueden expresarse como:

Ecuación 3-10. Ecuación de regresión de M(P)

$$M(P) = a_m P^{d_m}$$

Ecuación 3-11. Ecuación de regresión de B(P)

$$B(P) = a_b P^{d_b}$$

Figura 3-33. Variación de la pendiente respecto al procesamiento

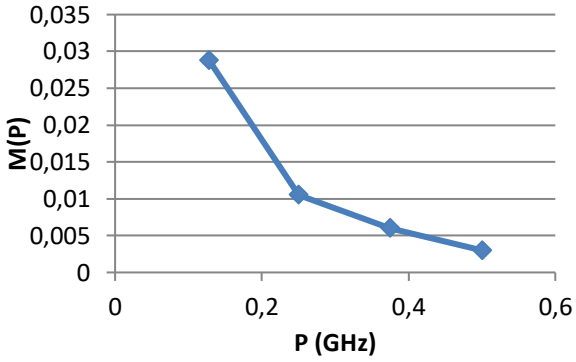
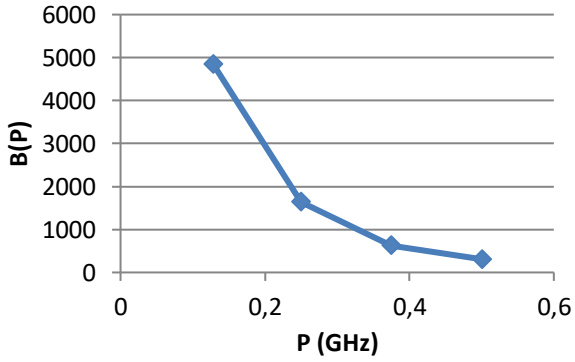


Figura 3-34. Variación de la ordenada respecto al procesamiento



La obtención de un modelo matemático que describa las funciones se resume en encontrar los valores para *a* y *d*. Si en la

Ecuación 3-9 aplicamos logaritmos naturales en ambos lados, tenemos los siguientes resultados

$$\ln Y = \ln(aX^d)$$

Ecuación 3-12. Transformación de un comportamiento exponencial a uno lineal

$$\ln Y = \ln a + d \ln X$$

La cual es una ecuación de una línea recta donde la pendiente es *d* y el valor de la ordenada es *ln(a)*, ahora es posible hallar los valores de la misma forma como se encontraron para la curva de t_C respecto a L cuando se deja constante la capacidad de procesamiento. La Tabla 3-5 resume los datos obtenidos al aplicar la regresión lineal simple.

Tabla 3-5. Resumen de datos de regresión matemática para M(P) y B(P) para el compresor BZip2

función	Ln(a)	a	d	Correlación	P-Valor
M(P)	-6,81511	0,0011	-1,60898	-0,9946	0,0053
B(P)	4,43919	84,7063	-2,01446	-0,9941	0,0059

En ambas funciones el valor absoluto del coeficiente de correlación está por encima de 0,99 sugiriendo una dependencia importante entre M y B respecto a P. Los P-Valores de nuevo son menores que 0,01 dotando al modelo de un gran nivel de confianza

(aproximadamente el 99%, información entregada por StatGraphics). Finalmente las ecuaciones para M(P) y B(P) son:

Ecuación 3-13. Modelo matemático para M(P) en BZip2

$$M(P) = 0{,}0011P^{-1{,}60898}$$

Ecuación 3-14. Modelo matemático para B(P) en BZip2

$$B(P) = 84{,}7063P^{-2{,}01446}$$

Las figuras Figura 3-35 y Figura 3-36 muestran en paralelo los datos obtenidos de forma práctica y los datos utilizando el modelo matemático.

Figura 3-35. Comparación de valores prácticos y valores calculados para M(P) en BZip2

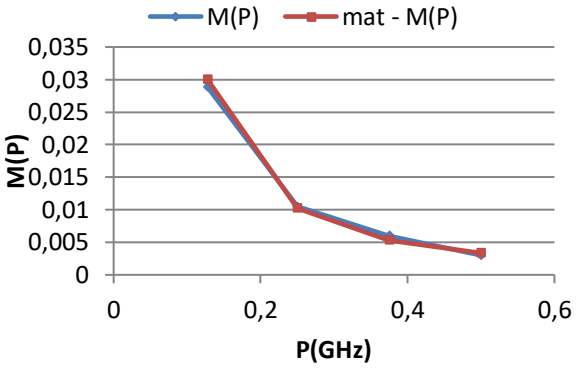
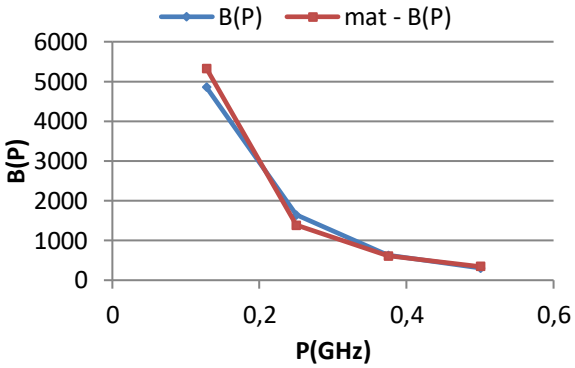


Figura 3-36. Comparación de valores prácticos y valores calculados para B(P) en BZip2



La Tabla 3-6 resume los datos obtenidos para las herramientas de compresión restantes[1].

Tabla 3-6. Resumen de datos de regresión matemática para el tiempo de compresión y descompresión

			Tiempo de Compresión			Tiempo de Descompresión		
No	**Compresor**	**Ecuación**	**d**	**a**	**Correlación**	**d**	**a**	**Correlación**
1	BZip2	M(P)	-1.6090	0.0011	-0.9947	-1.7407	0.0007	-0.9056
		B(P)	-2.0145	84.7063	-0.9941	-2.4264	32.2107	-0.9978
2	Exificient	M(P)	-1.7601	0.0008	-0.9954	-1.2157	0.0010	-0.9775
		B(P)	-2.3150	32.1637	-0.9416	-4.2612	1.9588	-0.9968
3	Exificient+GZ	M(P)	-1.5062	0.0018	-0.9927	-2.2557	0.0004	-0.9729
		B(P)	-1.1986	347.4428	-0.9889	-2.7364	36.1863	-0.9885
4	FastInfoset	M(P)	-1.4641	0.0007	-0.9852	-1.8412	0.0004	-0.9847
		B(P)	-3.3426	6.7363	-0.9919	-2.8582	8.1301	-0.9973
5	Gzip	M(P)	-1.4390	0.0007	-0.9932	-2.1310	0.0002	-0.9973
		B(P)	-1.8153	29.5945	-0.9560	-2.4153	2.0030	-0.9979
6	PPM	M(P)	-1.7117	0.0086	-0.9942	-1.4596	0.0018	-0.9952
		B(P)	-1.6619	156.1395	-0.9781	-1.3047	288.3659	-0.9163
7	SCMPPM	M(P)	-1.7099	0.0226	-0.9992	-2.2888	0.0009	-0.9995

[1] En el Anexo B se pueden encontrar las demás gráficas y modelos matemáticos obtenidos.

		B(P)	-0.3142	712.0157	-0.3552	-1.6040	237.3723	-0.9835
8	Xmill	M(P)	-1.5464	0.0006	-0.9967	-1.6074	0.0006	-0.9730
		B(P)	-1.5807	158.2348	-0.9797	-2.1088	89.1580	-0.9974
9	XMLPPM	M(P)	-1.5648	0.0012	-0.9953	-2.1213	0.0012	-0.9938
		B(P)	-1.8737	97.1309	-0.9883	-1.7591	266.3630	-0.9896
10	XWRT	M(P)	-1.7350	0.0003	-0.9950	-2.9755	0.0000	-0.9669
		B(P)	-1.1826	1887.0122	-0.9897	-1.3085	2191.6282	-0.9749

De esta manera se termina la FASE III y por tanto, la obtención del modelo matemático que permite realizar una selección automática de la herramienta de compresión que mejor se ajuste a las características del equipo que utiliza el consumidor y del servicio Web que intenta consumir.

La siguiente sección se centra en el desarrollo del componente *Repositorio de Metadatos.*

3.4 REPOSITORIO DE METADATOS

Como se indicó en la sección 2.1.5, los metadatos son información descriptiva que permite identificar, analizar y localizar recursos en internet. Desde sus inicios se han convertido en tema de estudio en la búsqueda por lograr una Web semántica que permita el uso masivo de los diferentes recursos disponibles en internet.

En la sección 3.2.2.2 se expresó que el *Repositorio de Metadatos* es uno de los dos componentes del *Módulo Procesador de Contenidos* del servidor. El repositorio consta de una base de datos que permite el almacenamiento de metadatos con la descripción de cada dispositivo cliente; su funcionalidad, como se ha planteado anteriormente, es brindarle al motor de compresión la información necesaria, para que éste decida cuál es la mejor herramienta de compresión a utilizar. En la Figura 3-37 se presenta gráficamente el documento XML que contiene la información que describe al cliente y su almacenamiento en la base de datos.

A continuación se analizará a partir del componente *Motor de Compresión,* la información que debe contener cada metadato y se procede a seleccionar la forma y el lenguaje de representación de dicha información.

Figura 3-37. Extracción de las características del dispositivo cliente

3.4.1 Información Contenida en el Metadato

En este punto gracias a las diferentes pruebas realizadas para el *Motor de Compresión,* ya se tiene claro que las características del dispositivo cliente que afectan las métricas de las herramientas de compresión son: el ancho de banda y la capacidad de procesamiento.

De esta manera, se concluye que el metadato contendrá únicamente:

- La dirección IP que identifique al dispositivo cliente,
- El ancho de banda al cual el consumidor tiene acceso y
- La capacidad de procesamiento con la que cuenta el dispositivo cliente.

Ahora, teniendo clara la función de los metadatos y la información que contienen, es necesario determinar el lenguaje y estructura[90], con el fin de lograr su definición teórica. En el capítulo 4 se explica: el modelo desarrollado para la creación manual del metadato, aplicando la definición teórica descrita a continuación; y su implementación dentro del sistema mediador.

3.4.2 Metalenguajes y Estructura del Metadato

En la sección 2.1.5.3 y la sección 2.1.5.4 se presentó una descripción detallada de cada uno de los metalenguajes y estructuras existentes; a continuación en la Tabla 3-7 se muestra una comparación general, que permite determinar cuál lenguaje es el más acorde y cuál es la estructura que mejor se ajusta a las necesidades del proyecto.

a) Comparación de Metalenguajes:

Tabla 3-7. Comparación de metalenguajes

Característica	SGML	XML	HTML
Estructura de Etiquetas	SI	SI	SI
Complejo	SI	NO	NO
Sintaxis precisa	SI	SI	SI
Flexibilidad en uso de texto	SI	SI	SI
Lenguaje Gráfico	NO	NO	SI

De la tabla anterior se puede decir que en términos generales, los 3 metalenguajes poseen características que aportan a la definición de metadatos; no obstante, la elección considerada para este trabajo de grado obedece a los siguientes criterios:

- La simplicidad hace descartar a SGML como un lenguaje para los metadatos, comparativamente con XML y HTML.
- Tradicionalmente, HTML se ha especializado en la presentación de información mientras que XML conserva una naturaleza semántica. Dada la necesidad de representar metadatos de manera acorde a lo que significan, se elije a XML como el Metalenguaje de referencia para los Metadatos del proyecto [72].

b) Comparación de Esquemas de Metadatos:

Cada una de las estructuras y lenguajes esperan mejorar el desempeño y eficiencia en la búsqueda, la recuperación de la información y la descripción de los recursos de la red a través de sus diferentes características como sintaxis, complejidad, documentación existente, interoperabilidad, estructura, contenido incluido en la descripción, entre otros. Es por ello que los diferentes estudios y comparaciones realizados se centran en las mejores opciones en cuanto a autonomía, manejo de contenido y facilidad de acceso. Sin embargo, para el propósito del presente proyecto no se requieren muchas de las funcionalidades que presentan estos esquemas, por ello en la Tabla 3-8 se presenta una comparación de los aspectos más importantes y afines a la investigación.

Tabla 3-8. Resumen comparativo de esquemas de metadatos

Característica	MARC	DUBLIN CORE	RDF	TEI
Simplicidad	NO	SI	SI	NO
Intuitivo y fácil	NO	SI	SI	NO
Prioridad en la descripción de información	SI	NO	NO	SI
Prioridad en el descubrimiento de los datos	NO	SI	SI	NO
En su Estructura posee:				
• **Campos Obligatorios**	SI	NO	NO	SI
• **Campos opcionales**	SI	SI	SI	SI
• **Repetición de campos**	SI	SI	SI	SI
• **Subcampos**	SI	NO	SI	NO
• **Control de valores**	SI	NO	NO	SI
Proceso de creación ágil	NO	NO	SI	NO
Interoperabilidad en bases de datos	NO	SI	SI	NO
Sintaxis	HTML/XML	HTML, XML	XML	SGML, XML

Así a partir de la sección 2.1.5.4 y basados en la tabla anterior se concluye que:

- MARC y TEI además de tener un propósito diferente al buscado en la investigación, poseen formatos complejos por el énfasis en la descripción de los recursos; requieren

de un alto manejo de normas y estándares establecidos, y poseen dentro de la estructura del metadato campos obligatorios. Por lo anterior se descarta a MARC y TEI como posibles esquemas de los metadatos a implementar.

- Dublin Core y RDF son sin duda los esquemas con mayor sencillez en su descripción, por ello, se indaga en los dos esquemas, con el fin de compararlos y llegar a una única opción de metadato.

A continuación en las figuras Figura 3-38 y Figura 3-39 se presentan como ejemplo, la estructura de un metadato definido según Dublin Core y RDF.

Figura 3-38. Metadato: DUBLIN CORE

```
<?xml version="1.0" encoding="UTF-8"?>
<head profile="http://dublincore.org/documents/dcq-html/">
<meta name”DC.Identifier” content=”168.162.1.11 (IP)“>
<meta name”DC.Subject” content=”CP=128MH”>
<meta name”DC.Subject” content=”BW=256Kbps”>
<head>
```

Figura 3-39. Metadato: RDF

```
<?xml version="1.0" encoding="UTF-8"?>
      <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      <ndl:Configuration rdf:about="#host_1">
            <ndl:IP>198.168.0.1</ndl:IP>
            <ndl:CP>128MH</ndl:CP>
            <ndl:BW>256Kbps</ndl:BW>
      </ndl:Configuration >
</rdf:RDF>
```

Si hacemos una relación de tamaño encontramos que un documento RDF es un poco más liviano que uno de Dublin Core si se representa la misma información. Así se infiere que Dublín Core es un esquema con alto nivel de simplicidad, pero se complica al momento de utilizar calificadore*s* en su estructura, lo que lo hace un poco más grande que RDF.

Adicionalmente, de acuerdo con la investigación presentada en [68], la originalidad y la mayor capacidad de RDF frente al Dublin Core residen en tres factores relevantes:

- RDF posee una sintaxis para múltiples formatos de metadatos y además permite especificaciones semánticas;
- RDF es definido en XML, y éste a su vez cuenta con el apoyo de reconocidas empresas de red como Netscape y Microsoft.
- Existen mayores posibilidades de uso de RDF al integrarse en la estructura XML (Extensible Markup Language);

De esta forma, dada la relación de tamaño y las anteriores características, RDF se convierte en el estándar más prometedor para la búsqueda y recuperación de informaci6n tanto en Internet como en las bibliotecas digitales. Por tanto se concluye que RDF es la estructura que cumple, de la mejor manera, con el objetivo de los metadatos en este proyecto: es simple y no genera sobrecarga. Así el lenguaje seleccionado es XML y la estructura RDF.

4. IMPLEMENTACIÓN DEL SISTEMA MEDIADOR

En el capítulo 3 se planteó la arquitectura base para la implementación de un sistema mediador que facilite el acceso a los servicios Web basados en SOAP desde redes de baja capacidad. La principal función del sistema tiene que ver con la compresión de la respuesta SOAP, condicionada por la selección del mejor mecanismo de compresión; dicha selección depende de la información contenida en los metadatos que describen al cliente del servicio, y la información propia del documento que se intenta comprimir.

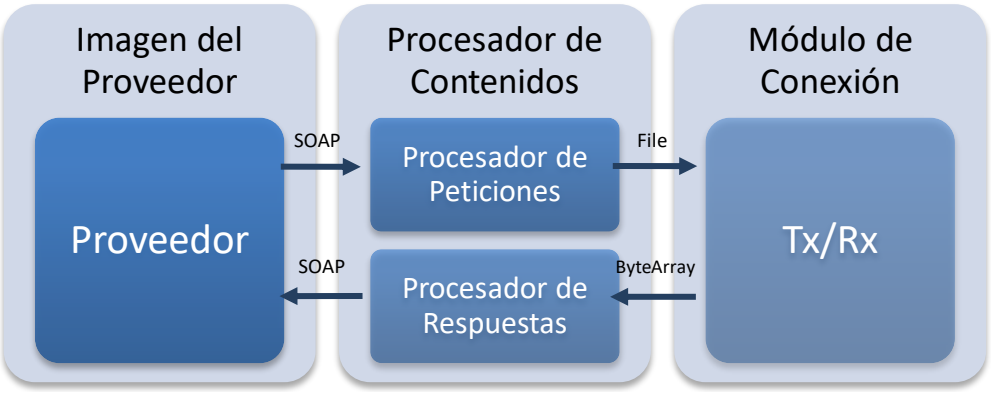
En este capítulo se describen los detalles de implementación del mediador desarrollado que utiliza un algoritmo de selección basado en los modelos matemáticos obtenidos en el capítulo 3; adicionalmente se muestran los resultados obtenidos en la implementación de un piloto de evaluación y se realiza un conjunto de pruebas con el fin de validar el algoritmo de selección de las herramientas de compresión y el funcionamiento del mediador.

4.1 MODELO DEL SISTEMA

4.1.1 Arquitectura del Sistema

Las figuras Figura 4-1 y **¡Error! No se encuentra el origen de la referencia.** presentan la arquitectura funcional *Cliente* y *Servidor* respectivamente; como se puede apreciar, cada una cuenta con tres módulos básicos presentados en la arquitectura base, los cuales se describen a continuación con más detalle.

Figura 4-1. Arquitectura Cliente



4.1.1.1 Cliente

4.1.1.1.1. Proveedor

El proveedor es una implementación de la clase *Provider* de Java y se encarga de iniciar un hilo de procesamiento para atender cada solicitud SOAP recibida. La conexión entre el *cliente* y el *servidor* se realiza a través de protocolo TCP/IP, el cual puede ser utilizado tanto de forma síncrona como asíncrona [115] y para este proyecto en particular se ha utilizado su configuración síncrona. Sin embargo, para en el contexto de los servicios Web la comunicación de esta forma es considerada asíncrona, debido a que la respuesta SOAP no es enviada de vuelta de forma inmediata, sino una vez ha pasado por toda una serie de procesos entre los que se incluyen compresión, envío y descompresión; es por esto que el proveedor debe agregar información a la solicitud que permita identificar el hilo que la atiende y luego entregarla al *procesador de peticiones*; una vez hecho esto, detiene su ejecución hasta que la respuesta se encuentre disponible para ser entregada al consumidor.

4.1.1.1.2. Procesador de Peticiones

Este módulo toma las peticiones en formato SOAP provenientes del *Proveedor* y se encarga inicialmente de agregarles un encabezado con la dirección IP del equipo donde se está ejecutando el cliente, de tal manera que el servidor pueda determinar a donde enviar de vuelta la respuesta; posteriormente, genera un archivo externo con la información de la petición y entrega la ruta de acceso a dicho archivo al módulo de conexión.

4.1.1.1.3. Módulo de Conexión TX/RX

Como su nombre lo indica, se encarga de la transmisión de información entre el cliente y el servidor. Para la transmisión, toma la ruta del archivo externo que convierte en un arreglo de bytes para enviarlos a través del canal de datos; y para la recepción, cuenta con dos componentes importantes: un *listener* o componente de escucha y un *controlador* para la conexión. El primero se encarga de atender la recepción y cada vez que el servidor inicia una transferencia, notifica al *controlador*, el cual inicia un hilo de comunicación y mantiene la conexión abierta hasta que se ha recibido completamente la información. Cuando el procedimiento se completa, el módulo entrega el arreglo de bytes recibido al procesador de respuestas y cierra la conexión indicando al servidor que la transferencia fue exitosa.

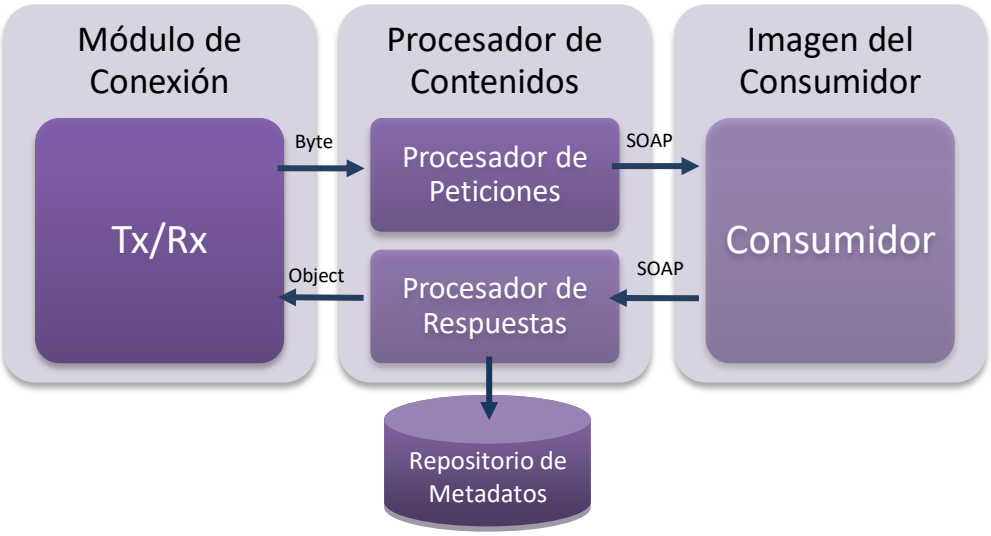
4.1.1.1.4. Procesador de Respuestas

El arreglo de bytes proveniente del *módulo de conexión* contiene una respuesta comprimida a una petición específica y requiere un proceso de descompresión para obtenerla en formato SOAP. La primera función de éste módulo es convertir el arreglo de bytes en un objeto Java que contenga la información sobre el mecanismo de compresión utilizado y la respuesta comprimida; con esta información se procede a realizar la descompresión, obteniendo como resultado un archivo externo. Posteriormente toma la información de dicho archivo para generar un mensaje SOAP, accede a los encabezados

del mensaje para obtener el identificador del hilo encargado del consumo, fija la respuesta al hilo identificado y le notifica para que continúe su ejecución realizando la entrega al consumidor del servicio.

4.1.1.2 Módulo Servidor

Figura 4-2. Arquitectura Mediador: Módulo Servidor



4.1.1.2.1. Módulo de Conexión Tx/Rx

Su funcionamiento es igual al modulo de conexión incluido en el cliente. Véase el numeral 4.1.1.1.3.

4.1.1.2.2. Procesador de Peticiones

El módulo de conexión entrega un arreglo de bytes con la información de las peticiones y el procesador de peticiones se encarga de realizar la transformación a formato SOAP para entregarla a la imagen del consumidor.

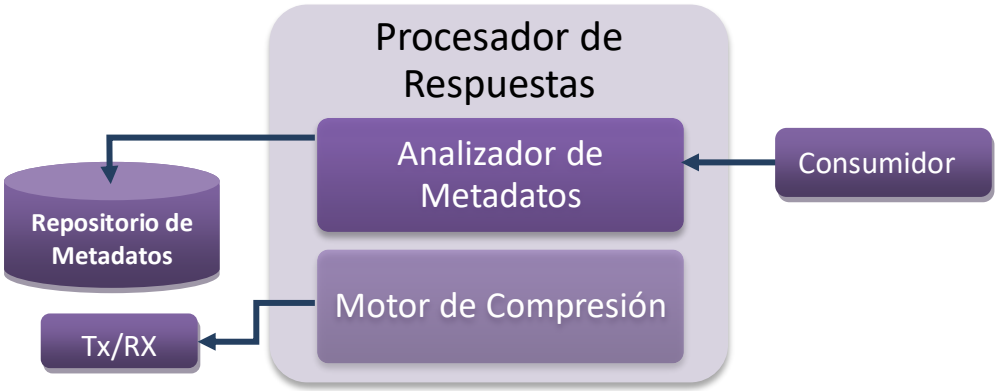
4.1.1.2.3. Consumidor

Esta imagen del consumidor es la encargada de realizar el consumo final del servicio. Para ello, examina los encabezados de la petición obteniendo: la dirección del proveedor del servicio, la dirección IP del modulo cliente que realizó la solicitud y el identificador del hilo de procesamiento encargado de atender dicha petición. Posteriormente, cuando recibe la respuesta, agrega a sus encabezados el id del hilo de procesamiento, para luego entregarla al procesador de respuestas junto con la IP del *cliente*.

4.1.1.2.4. Procesador de Respuestas

Este es uno de los módulos más importantes del servidor. En él se encuentran el *Repositorio de Metadatos* y el *Motor de Compresión*, como se muestra en la Figura 4-3; este último, después de realizar la compresión, empaqueta el resultado en un objeto Java al que además agrega un identificador que le permitirá al cliente determinar cual herramienta de compresión fue utilizada. Finalmente entrega el paquete al módulo de conexión para que realice el envío de vuelta al cliente.

Figura 4-3. Procesador de Respuestas



4.1.2 Diagrama de Despliegue

Las figuras Figura 4-4 y Figura 4-5 presentan el diagrama de despliegue del *cliente* y el *servidor* respectivamente. El *cliente* ha sido desplegado sobre un equipo con sistema operativo Ubuntu 11.10 y con el objetivo de poder ejecutar la aplicación requiere de la maquina virtual de Java versión 1.6. El componente *CompressionMotor.jar* fue implementado como un conjunto de librerías que permite la compresión y descompresión de información mediante la utilización de las herramientas seleccionadas en el capítulo 3. En cuanto a las interfaces, expone una implementación de la clase *Provider* que utiliza un empaquetado SOAP y se convierte en un punto final o *EndPoint* que permite recibir peticiones desde clientes de servicios Web a través de protocolo HTTP [15, 116]. Dicho *Provider* está configurado para acceder al mensaje SOAP completo, es decir, tanto encabezados como carga útil. También se exponen los puertos 4444 y 5555, el primero para la recepción de información a través de TCP/IP y el segundo para el envío de la misma.

El consumidor del servicio Web debe utilizar un *MessageHandler*, el cuál es un componente encargado de interceptar los mensajes SOAP salientes; al interceptarlos, utiliza la información del mensaje para determinar la dirección del proveedor y la agrega a los encabezados para enviar el mensaje al modulo cliente.

El *servidor*, al igual que el *cliente*, se ha desplegado sobre un sistema operativo Ubuntu 11.10 y utiliza la máquina virtual de Java versión 1.6; puesto que en él se realizan todos los procesos de compresión, requiere también del componente *CompressionMotor.jar*. Para la recuperación de metadatos que describen el equipo del consumidor utiliza la librería *Jena2.6.4*. Finalmente el Servidor expone los puertos 4444 para el envío de información y 5555 para su recepción a través de TCP/IP.

Figura 4-4. Diagrama de Despliegue: Cliente

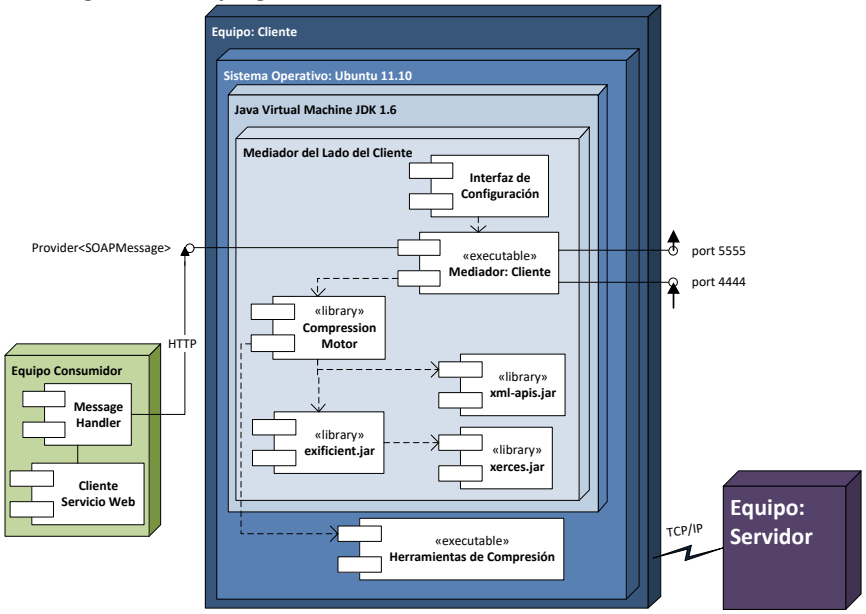
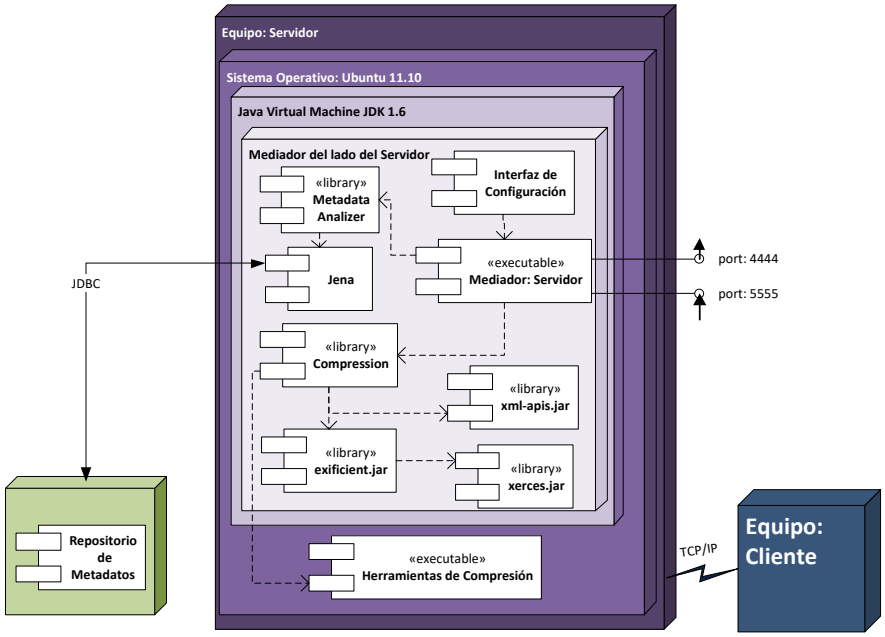


Figura 4-5. Diagrama de Despliegue: Servidor

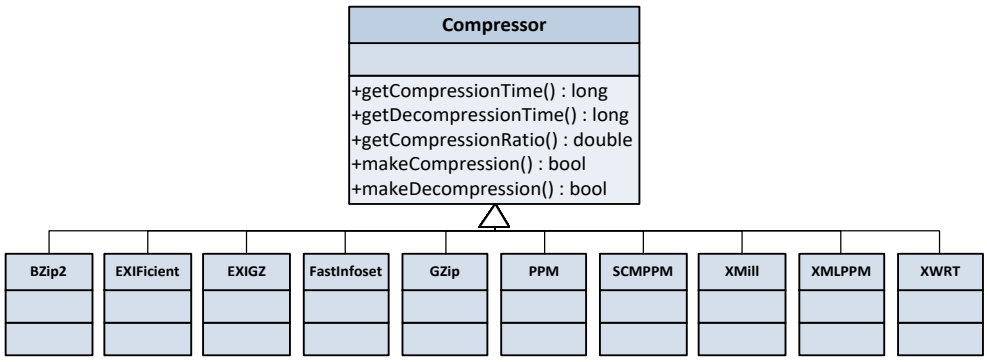


A continuación se describen con más detalle los componentes utilizados por el Cliente y el Servidor.

4.1.2.1 CompressionMotor.jar:

Contiene las clases necesarias para la compresión y descompresión de datos e implementa las funciones para calcular el tiempo aproximado requerido por una herramienta para realizar los procesos de compresión y descompresión; igualmente, determina el radio de compresión aproximado de cada una de las herramientas de acuerdo a los modelos matemáticos obtenidos en el capítulo 3. La Figura 4-6 presenta un bosquejo del diagrama de clases utilizado por el componente; como se puede observar existe una clase padre llamada Compressor y de ella heredan las demás que representan cada una de las herramientas de compresión. En su mayoría las herramientas utilizadas fueron desarrolladas utilizando lenguaje C y por tanto se manejan como externas a la aplicación, Exificient por el contrario ha sido desarrollada en Java y para su funcionamiento utiliza los componentes xml-apis.jar y exificient.jar, éste último a su vez requiere del componente xerces.jar.

Figura 4-6. Diagrama de clases resumen CompressionMotor.jar



4.1.2.2 Xerces.jar

Apache Xerces es un proyecto de desarrollo de software colaborativo dedicado a proporcionar analizadores XML robustos, completos, con calidad comercial pero de libre distribución, además de tecnologías estrechamente relacionadas, desarrolladas en una gran variedad de plataformas para soportar diferentes lenguajes de programación [117]. Xerces.jar es una implementación desarrollada en Java del proyecto Xerces.

4.1.2.3 Xml-apis.jar

JAXP[1] permite a las aplicaciones analizar, transformar, validar y realizar consultas sobre documentos XML utilizando una API que es independiente de la implementación del procesador XML utilizado. La API proporciona una capa que permite a los proveedores proporcionar sus implementaciones sin la necesidad de agregar nuevas dependencias [118]. Xml-apis.jar es una librería de implementación del API JAXP [119].

[1] JAXP: *Java API for XML Processing.*

4.1.2.4 Exificient.jar

Es una librería de código abierto desarrollada en Java por el proyecto EXIficient, el cual es una implementación libre del formato EXI[1] [61].

4.1.2.5 MetadataAnalizer.jar

En este paquete se han incluido las clases que permiten la recuperación de recursos que representan los equipos de los consumidores. Para realizar dicha recuperación utiliza la librería Jena2.6.4.

4.1.2.6 Jena2.6.4

Este framework que hace parte del proyecto Apache, proporciona una amplia biblioteca Java para ayudar a los desarrolladores a construir código que requiere el procesamiento de RDF, RDFS, RDFa, OWL y SPARQL utilizando recomendaciones del W3C. Incluye un motor de inferencia para llevar a cabo un razonamiento basado en ontologías OWL y RDFS, y una variedad de estrategias de almacenamiento entre las que se incluyen bases de datos relacionales [120].

4.1.3 Diagramas de Secuencia

Los siguientes diagramas de secuencia pretenden describir: el funcionamiento del sistema mediador, la información intercambiada entre cada módulo y el comportamiento del mensaje SOAP durante todo el proceso de consumo del servicio Web. Con este objeto, se presenta el siguiente escenario:

Supóngase que un usuario llamado Alice desea consumir un servicio Web llamado *Calculator* y específicamente la operación *add*; dicha operación recibe como parámetros de entrada dos valores enteros *x* e *y* para retornar su adición (*x + y*), la configuración de servicios y equipos es la siguiente:

- Dirección IP del equipo de Alice: 192.168.0.2
- Dirección IP donde está ubicado el mediador del lado del consumidor, a partir de ahora llamado Cliente: 192.168.0.3
- Dirección IP donde está ubicado el mediador del lado del proveedor, a partir de ahora llamado Servidor: 192.168.1.2
- Dirección del proveedor del servicio Web: 192.168.1.3 :8080/WebCalculator/Calculator
- Operación del servicio a invocar: add

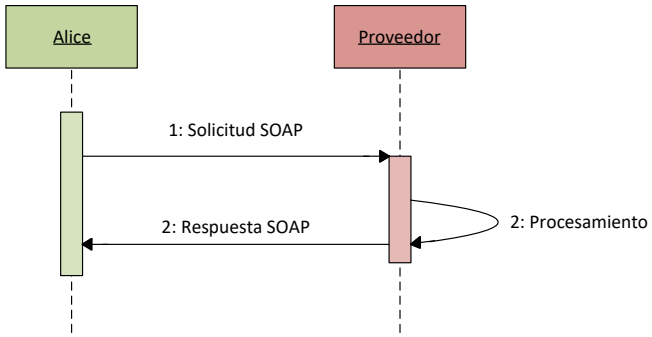
Cabe aclarar que entre el Cliente y el Servidor hay una red de baja capacidad.

[1] EXI: *W3C Efficient XML Interchange.*

4.1.3.1 Interacción Alice-Proveedor Sin Incluir el Sistema Mediador

La Figura 4-7 presenta la interacción básica entre Alice y el Proveedor del servicio sin utilizar el sistema mediador; dicha interacción se puede resumir en el simple intercambio de mensajes SOAP. Alice envía una petición al servicio Web y el proveedor después de hacer el procesamiento necesario envía una respuesta de vuelta a Alice.

Figura 4-7 Diagrama de Secuencia: Interacción Alice-Proveedor



Las figuras Figura 4-8 y Figura 4-10 son los mensajes de solicitud y respuesta que se obtienen al consumir el servicio. Ambos cuentan con el modelo propio de los mensajes SOAP compuesto por un elemento *envelope*, un encabezado o *header* y un cuerpo del mensaje o *body* como lo muestra la Figura 4-9. Dado que el encabezado es un elemento opcional, en este caso no se encuentra.

Figura 4-8. Petición SOAP: Webcalculator - add

```
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:add xmlns:ns2="http://ws.webcalculator.org/">
            <x>23</x>
            <y>58</y>
        </ns2:add>
    </S:Body>
</S:Envelope>
```

Figura 4-9. Estructura básica de un mensaje SOAP

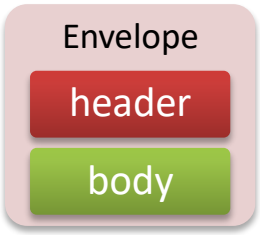


Figura 4-10. Respuesta SOAP:Webcalculator – add

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:addResponse xmlns:ns2="http://ws.webcalculator.org/">
            <return>81</return>
        </ns2:addResponse>
    </S:Body>
</S:Envelope>
```

4.1.3.2 Interacción Alice-Proveedor Incluyendo el Sistema Mediador

El diagrama de la Figura 4-12 representa de nuevo la interacción entre Alice y el proveedor del servicio Web, pero esta vez se han incluido los componentes del sistema mediador. Cuando el mensaje pasa a través del componente *MessageHandler* se agrega información en los encabezados del mensaje. Para casos en los que no exista la sección de encabezados dentro de la solicitud, el componente se encarga de crear dicha sección; la Figura 4-11 es un ejemplo del mensaje de solicitud al agregar el encabezado con la dirección del proveedor.

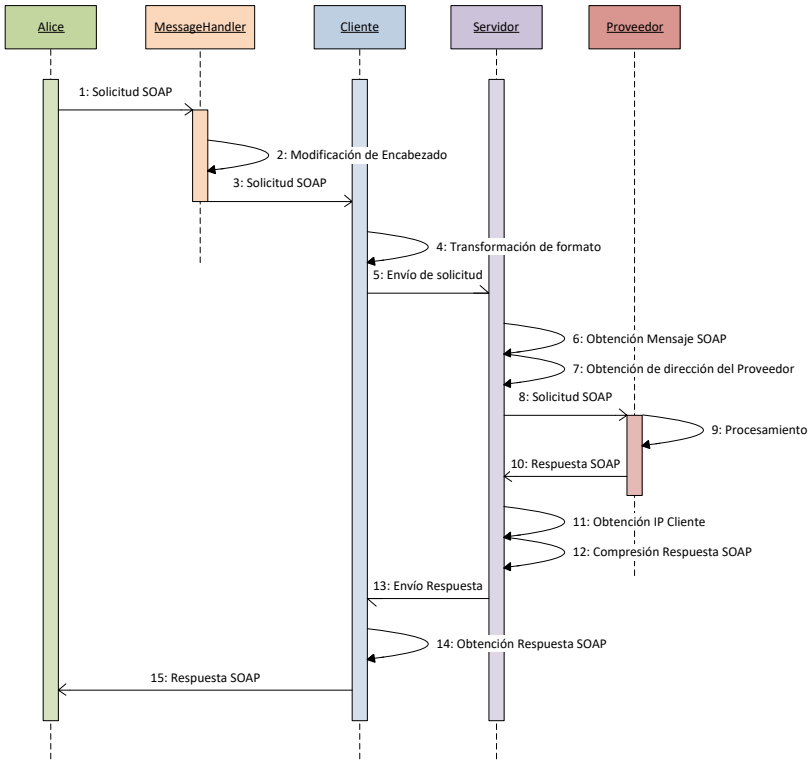
Figura 4-11. Solicitud SOAP con dirección del proveedor en los encabezados

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header>
        <spEndPoint xmlns="asoaplc">
           http://192.168.1.3 :8080/WebCalculator/Calculator
        </spEndPoint>
    </S:Header>
    <S:Body>...</S:Body>
</S:Envelope>
```

Se ha designado la etiqueta *spEndPoint*[1] para representar al proveedor del servicio Web; a partir de ese nombre el Servidor iniciará la búsqueda para determinar cuál es la dirección del proveedor. Una vez el *Cliente* recibe la solicitud realiza todo el proceso de transformación de la misma para enviarla al *Servidor*. Como ya se mencionó el envío se ha considerado como asíncrono en el contexto de los servicios Web, de esta forma, los puertos 5555 del *Cliente* y el *Servidor* que se enlazan para hacer el intercambio de la petición, cierran la conexión en el momento en que la petición ha sido enviada al *Servidor*; posteriormente la respuesta es retornada a través de los puertos 4444. Cuando el Servidor recibe el paquete con la petición, inicialmente realiza el proceso de desempaquetado para obtener el mensaje SOAP allí contenido y luego obtiene la dirección del proveedor para realizar el consumo del servicio Web. Como resultado, se obtiene una respuesta SOAP que es comprimida y enviada de vuelta al *Cliente*, quien a su vez se encarga de realizar la descompresión y obtener la respuesta para enviarla a Alice.

[1] spEndPoint: Service Provider End Point – Punto final del proveedor del servicio.

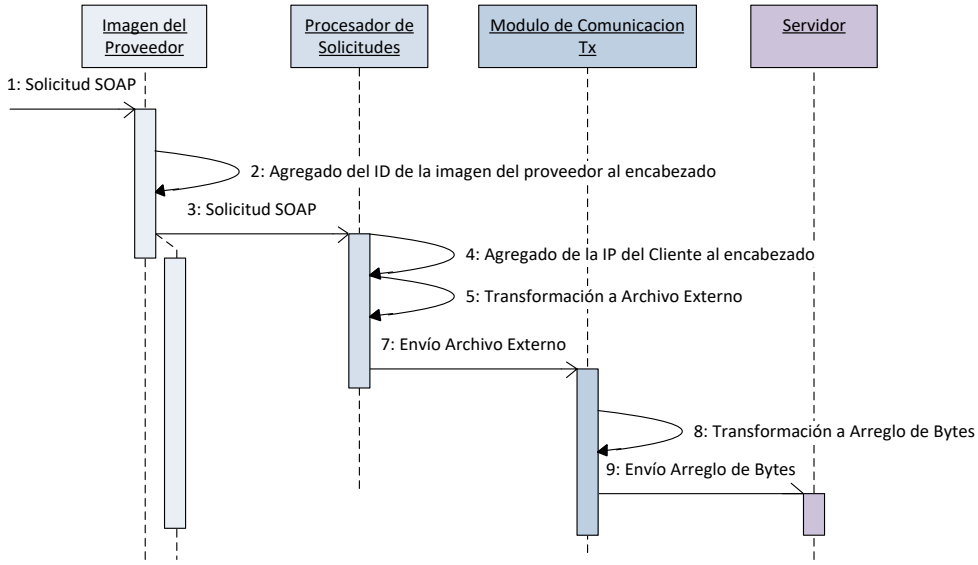
Figura 4-12. Diagrama de Secuencia: Interacción Alice-Mediador-Proveedor



4.1.3.3 Interacción Entre los Componentes del Cliente

Como muestra en la Figura 4-13 cuando la imagen del proveedor recibe una solicitud SOAP, su primera función es agregar el identificador de dicha imagen al encabezado; después de entregar la solicitud al procesador de solicitudes, detiene su ejecución. El procesador de solicitudes agrega la IP del Cliente al encabezado y luego transforma la solicitud a un archivo externo, entrega una ruta de dicho archivo al modulo de comunicaciones para que éste realice todo el proceso de envío al Servidor. La Figura 4-14 presenta un ejemplo de la solicitud SOAP después de agregar todos los encabezados.

Figura 4-13. Diagrama de Secuencia: Interacción de componentes del Cliente para el envío de una solicitud



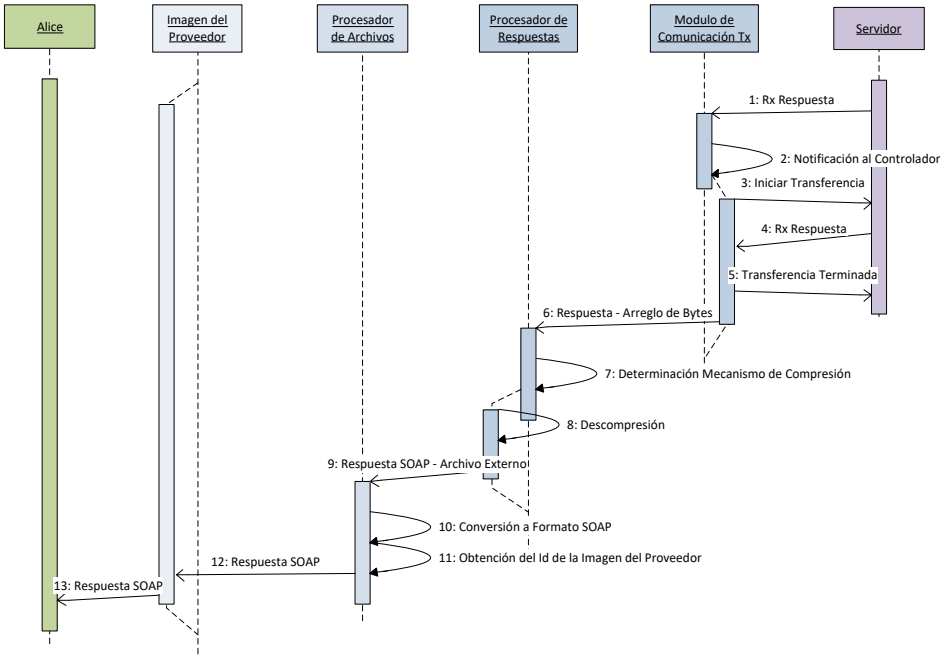
Se han reservado las etiquetas *mcEndPoint (Mediator Client End Point* – Punto Final del Cliente Mediador*)* e *idProvider* (*Provider Identificator* – Identificación del Proveedor) para denotar la dirección IP del *Cliente* y el identificador de la imagen del proveedor respectivamente.

Figura 4-14. Solicitud SOAP con ID de la imagen del proveedor e IP del Cliente

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header>
        <spEndPoint xmlns="asoaplc">
            http://192.168.1.3 :8080/WebCalculator/Calculator
        </spEndPoint>
        <idProvider xmlns="asoaplc">
            provider19
        </idProvider>
        <mcEndPoint xmlns="asoaplc">
            192.168.0.3
        </mcEndPoint>
    </S:Header>
    <S:Body>...</S:Body>
</S:Envelope>
```

Cuando el módulo de comunicaciones recibe una respuesta por parte del Servidor, notifica al controlador de la conexión; éste a su vez acepta los datos dando inicio a la transferencia. Al finalizar dicha transferencia notifica al servidor y cierra la conexión. Aunque el procesador de archivos no fue incluido en la arquitectura del cliente, éste hace parte del procesador de contenidos; su funcionalidad es la de tomar archivos externos y convertirlos en formato SOAP para que puedan ser entregado al consumidor. Esta interacción se puede apreciar en la Figura 4-15.

Figura 4-15. Diagrama de Secuencia: Interacción de componentes del Cliente para la recepción de una respuesta



4.1.3.4 Interacción Entre los Componentes del Servidor

Como se puede inferir, existe una gran concordancia entre el funcionamiento del *cliente* y el del *servidor*, de hecho fueron diseñados para contener los mismos módulos y por lo tanto, sus interacciones internas son muy similares en ambos casos. Las figuras Figura 4-16 y Figura 4-17 presentan la interacción entre los componentes del *servidor* cuando se realiza el consumo de un servicio. En este caso, la recepción de los datos sigue el mismo protocolo que en el *cliente*.

Cuando se recibe una respuesta por parte del proveedor del servicio, este mensaje de respuesta también debe ser modificado con la intención de agregar el identificador de la imagen del proveedor. Igualmente, se utiliza el atributo *contentDescription*[1] [121] de la clase *SOAPMessage,* para fijar en él la dirección IP del cliente. La Figura 4-18 representa el mensaje de respuesta como finalmente se entrega al procesador de respuestas.

[1] contentDescription: Atributo de la clase SOAPMessage que se refiere a una descripción del contenido del mensaje.

Figura 4-16. Diagrama de Secuencia: Interacción de componentes del Servidor para la recepción de una solicitud

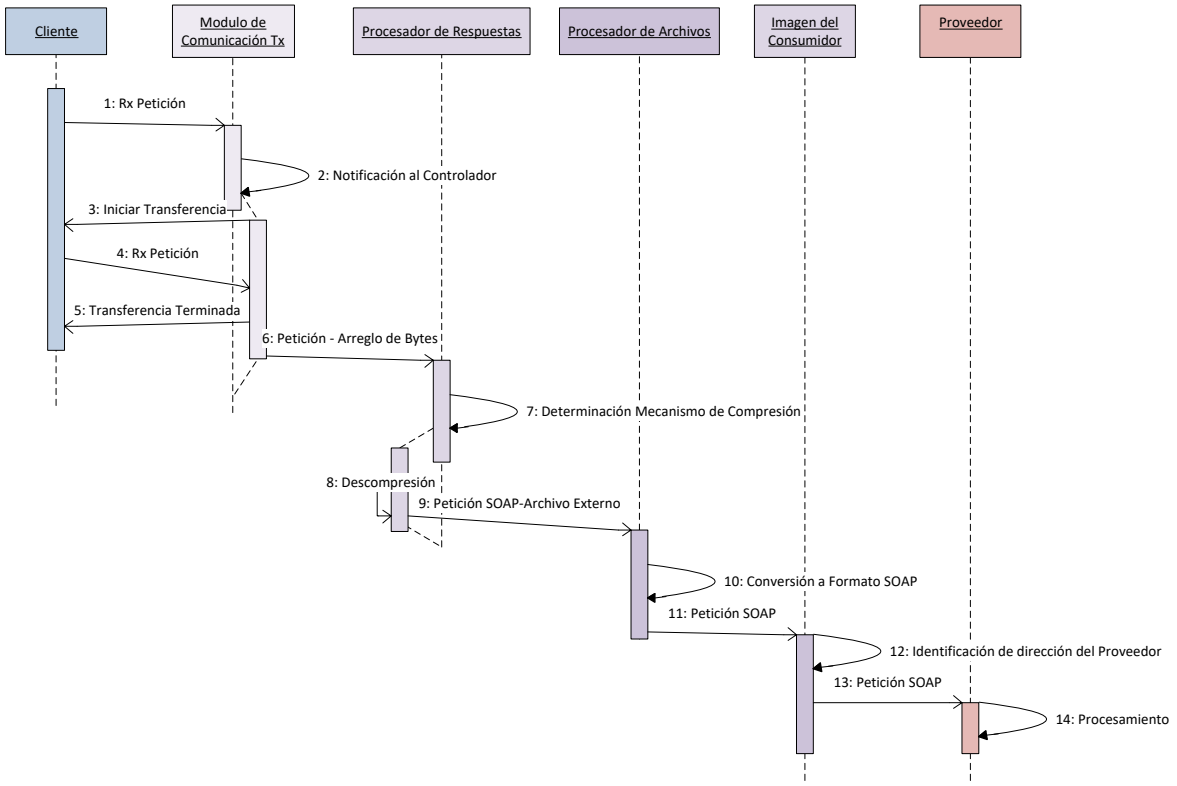


Figura 4-17. Diagrama de Secuencia: Interacción de componentes del Servidor para el envío de una respuesta

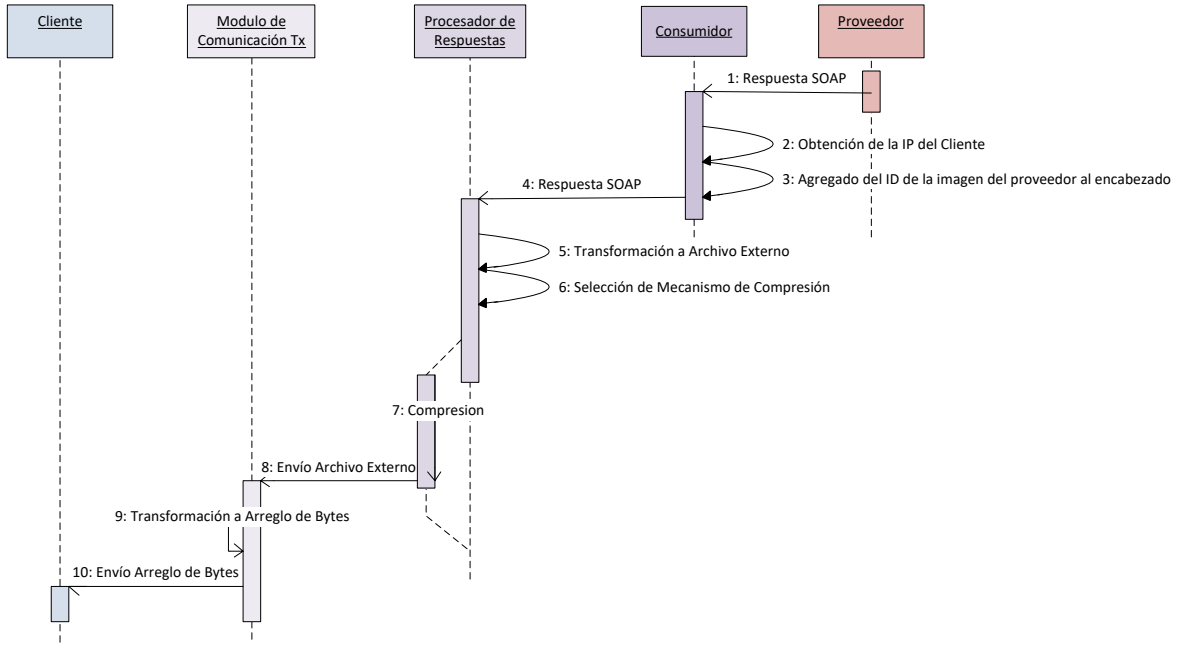
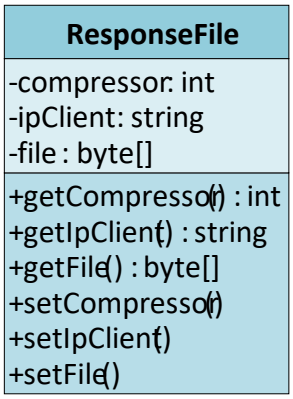


Figura 4-18. Respuesta SOAP: webcalculator - add con idProvider

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header>
        <idProvider xmlns="asoaplc">
           provider19
        </idProvider>
    </S:Header>
    <S:Body>
        <ns2:addResponse xmlns:ns2="http://ws.webcalculator.org/">
            <return>81</return>
        </ns2:addResponse>
    </S:Body>
</S:Envelope>
```

El procesador de respuestas toma el mensaje SOAP y a partir del analizador de metadatos determina el mejor mecanismo de compresión. Posteriormente, el *motor de compresión* se encarga de realizar dicho proceso como se presenta en el algoritmo de selección, véase numeral 4.2.3. La clase *ResponseFile* fue diseñada para contener la información necesaria para que el modulo de comunicaciones pueda enviar la respuesta al *Cliente*, la cual contiene los siguientes atributos: *ipClient* que hace referencia a la dirección IP del *Cliente*, *compressor* para identificar el tipo de compresor utilizado, y *file* el cual se refiere al archivo que será transmitido. En este caso, dicho atributo es directamente un arreglo de bytes con la información del archivo, y el módulo de comunicaciones se encarga de la serialización[1] [122] del objeto instancia de *ResponseFile.* La Figura 4-19 representa la clase *ResponseFile.*

Figura 4-19. Clase: ResponseFile



4.2 IMPLEMENTACIÓN DEL PILOTO

[1] La serialización de objetos es el proceso de almacenar el estado de un objeto en una secuencia de bytes, de esta forma no solo se facilita su reconstrucción posterior sino también su envío a través de interfaces que requieren que el objeto deje de existir en la máquina virtual, por ejemplo, el envío a través de una interfaz de red.

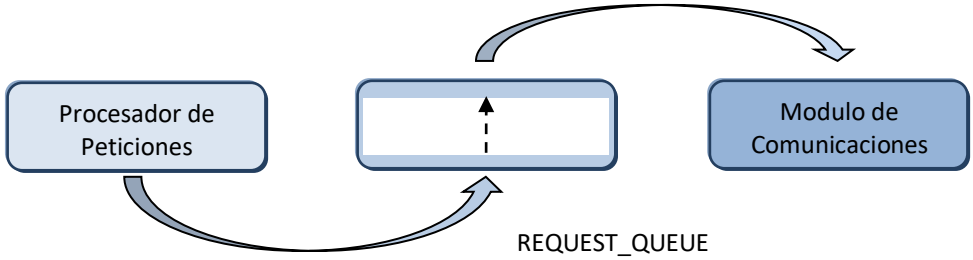
4.2.1 Procesamiento Basado en Colas

Muchos servidores Web, de bases de datos, de archivos, o servidores de correo electrónico, orientan su procesamiento en torno a un gran número de tareas simples y cortas que se realizan en fuentes remotas. Una petición llega al servidor en algún formato y debe pasar a través del protocolo de la red ya sea HTTP, FTP, POP, entre otros [123]. La arquitectura presentada para cada modulo principal del sistema, es decir, el Cliente y el Servidor ha sido diseñada para que un componente diferente se encargue de realizar cada tarea y por tanto no sea necesario que todo el proceso de consumo de un servicio Web, incluyendo modificación de encabezados, cambios de formato del mensaje, compresión, transmisión y descompresión, se ejecute en un solo componente generando tiempos de procesamiento muy altos y por consiguiente cuellos de botella en la transmisión.

De acuerdo a este contexto, cada componente se encarga de realizar alguna de las cortas tareas necesarias para permitir que Alice pueda consumir el servicio Web *calculator* de forma satisfactoria. Sin embargo, el dividir todo el proceso en pequeñas funciones resulta en un gran número de tareas realizadas de forma paralela y varios componentes funcionando al mismo tiempo. Haciendo posible que un componente termine de hacer su proceso y al entregarlo al siguiente, éste se encuentre ocupado y no pueda atender la solicitud generando un error en la ejecución del sistema.

Es en este punto donde utilizar un procesamiento basado en colas cobra un gran valor. De esta manera, cada componente al terminar su proceso no entrega directamente el resultado al componente siguiente, en vez de eso, pone su resultado en una cola FIFO[1] y el componente siguiente se encarga de atender los objetos contenidos en dicha cola, de forma que al terminar de procesar uno, continúa con el siguiente de la cola y se logra que todas las peticiones sean procesadas. La Figura 4-20 representa un ejemplo de lo mencionado, en ella, el Procesador de Peticiones al terminar un proceso y obtener un archivo externo lo agrega a la cola *REQUEST_QUEUE* de donde el módulo de comunicaciones lo recibe para enviarlo al Servidor.

Figura 4-20. Inserción de cola entre el Procesador de Peticiones y el Módulo de Comunicaciones



[1] FIFO: First In First Out, se refiere al orden en que se procesan los objetos contenidos en la cola, el primer componente en entrar será el primero en ser procesado, y se continuará atendiendo en el orden de llegada.

Las tablas Tabla 4-1 y Tabla 4-2 listan las colas utilizadas en el mediador, el formato de los objetos que almacena, el componente encargado de agregarle objetos y el componente encargado de procesarlas.

Tabla 4-1. Colas en el Cliente

Cola	Formato	Componente: almacena	Componente: procesa
REQUEST_Q	SOAP	Imagen del Proveedor	Procesador de Peticiones.
REQUEST_READY_Q	Archivo Externo	Procesador de Peticiones	Comunicaciones Tx
RESPONSE_COMPRESSED_Q	Arreglo de bytes	Comunicaciones Rx	Procesador de Respuestas
RESPONSE_Q	Archivo Externo	Procesador de Respuestas	Procesador de Archivos
RESPONSE_READY_Q	SOAP	Procesador de Archivos	Imagen del Proveedor

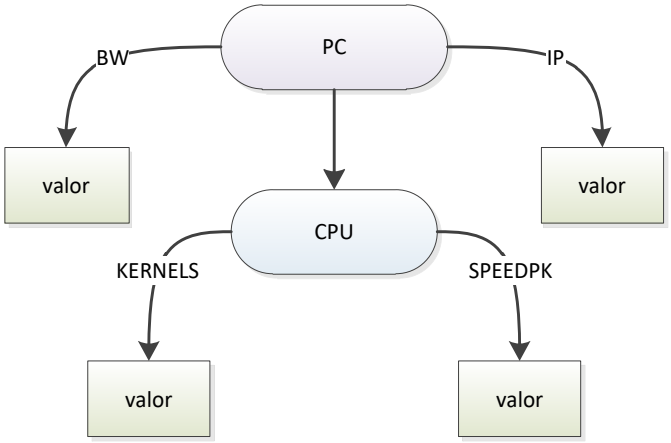
Tabla 4-2. Colas en el Servidor

Cola	Formato	Componente: almacena	Componente: procesa
REQUEST_COMPRESSED_Q	Arreglo de Bytes	Comunicaciones Rx	Procesador de Peticiones
REQUEST_Q	Archivo Externo	Procesador de Peticiones	Procesador de Archivos
REQUEST_READY_Q	SOAP	Procesador de Archivos	Imagen del Consumidor
RESPONSE_Q	SOAP	Imagen del Consumidor	Procesador de Respuestas
RESPONSE_FILE_Q	ResponseFile	Procesador de Respuestas	Comunicaciones Tx

4.2.2 Creación del Repositorio de Metadatos

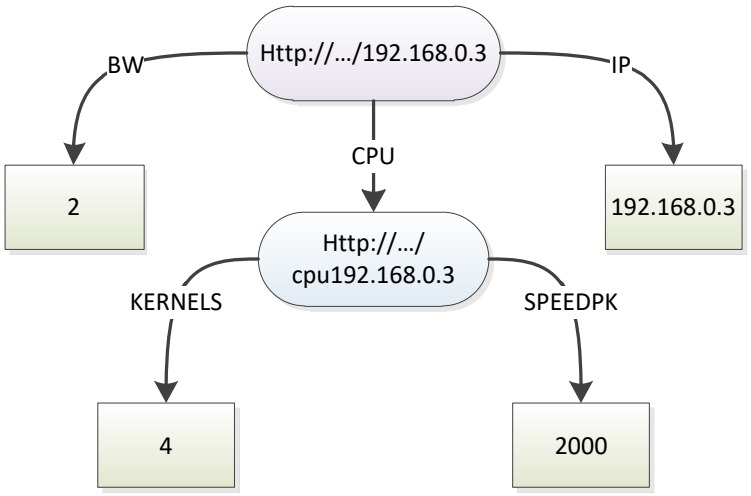
Como se mencionó en el capítulo 3, el lenguaje seleccionado para representar los metadatos fue RDF por su simplicidad y porque no contiene campos obligatorios que deban ser ingresados. A su vez, se utilizó Jena 2.6.4 como framework para la utilización de RDF dentro de Java. En este framework, toda la información proporcionada por una colección de RDF es contenida en una estructura de datos llamada *Model* la cual representa los nodos y sus propiedades. Cabe aclarar que un nodo es considerado como un recurso y una propiedad denota un valor contenido en el recurso o la conexión entre dos recursos. Para este caso específico fue necesario diseñar un modelo que permitiera la descripción de un equipo mediante las propiedades planteadas en el capítulo 3, la Figura 4-21 representa el modelo diseñado.

Figura 4-21. Modelo RDF para la descripción de un equipo



Las propiedades incluidas dentro del modelo son: BW o ancho de banda en kbps, IP representa la dirección ip, CPU es considerado como un recurso incluido en el pc y está descrito por: KERNELS que representa la cantidad de sus núcleos de procesamiento activos y SPEEDPK la velocidad en Mhz de cada uno de ellos. Del mismo modo, el nombre utilizado para el modelo fue *pc_dcard*; este nombre permite su recuperación dentro del mediador. Retomando el ejemplo de Alice y suponiendo que el equipo cliente cuenta con un tasa de transmisión de 2kbps, una CPU con 4 núcleos de procesamiento y cada núcleo con una velocidad de 2Ghz, el modelo sería el presentado en la Figura 4-22.

Figura 4-22. Modelo RDF para el equipo cliente utilizado por Alice



4.2.3 Implementación del Algoritmo de Selección

Siempre que se obtenga una respuesta SOAP, el algoritmo de selección del mejor mecanismo de compresión recibe como parámetro el recurso descriptor del equipo que realizó la petición SOAP. A partir de las características allí incluidas y mediante los modelos matemáticos se selecciona cada uno de los mecanismos de compresión y se calculan sus tiempos de respuesta; una vez se ha pasado por todos los compresores, se organizan los tiempos de respuesta de menor a mayor y, finalmente se selecciona el compresor que generó un menor tiempo de respuesta. La Figura 4-23 presenta el diagrama de flujo y la Figura 4-24 presenta la explicación en pseudocódigo de dicho algoritmo.

Para realizar el cálculo del tiempo de respuesta se requiere inicialmente estimar los tiempos de compresión y descompresión mediante los modelos matemáticos obtenidos; luego utilizar la longitud del documento para seleccionar el modelo matemático que mejor describa el radio de compresión y a partir de este determinar el tiempo de transmisión. Finalmente el tiempo de respuesta es calculado mediante la sumatoria de tiempos de compresión, transmisión y descompresión.

Figura 4-23. Diagrama de flujo del algoritmo de selección de mecanismos de compresión

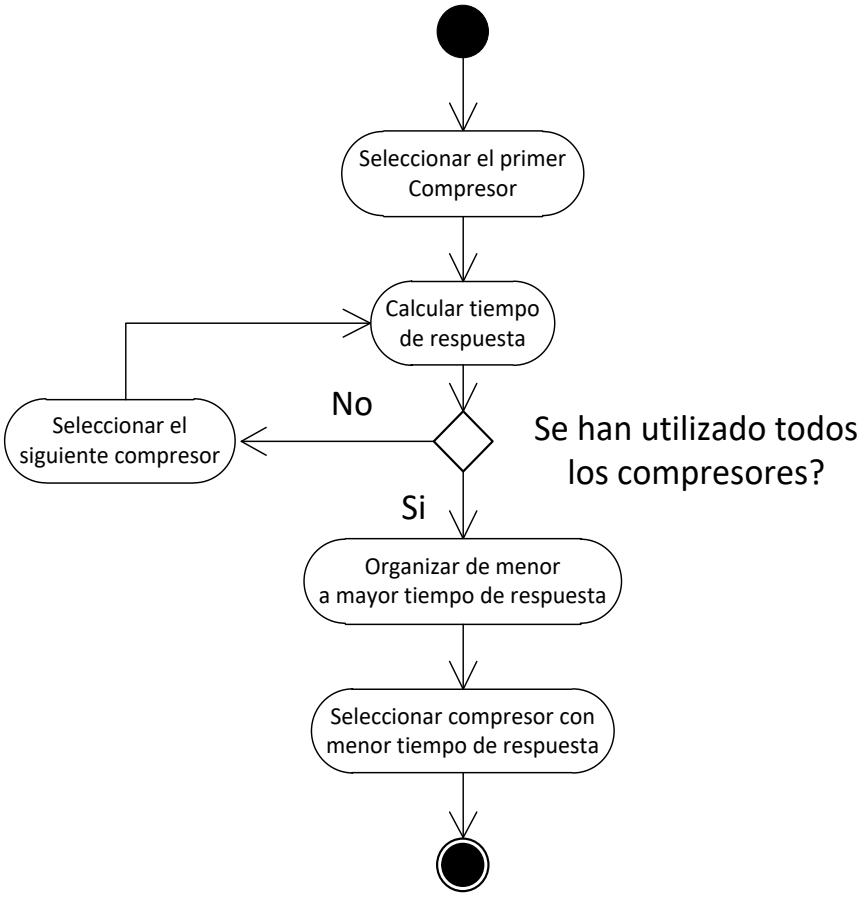


Figura 4-24 Pseudocódigo del algoritmo de selección de mecanismos de compresión

```
Inicio
  Variables   mapa   =   HashMap<Cadena,   Objeto>,   tiempoRespuesta=entero,
  compresorActual = Compresor

  compresorActual = Compresor1

  Hasta que (compresorActual = Compresor10)
    Hacer
      tiempoRespuesta = CalcularTiempoRespuesta para compresorActual
      agregar en mapa (compresorActual, tiempoRespuesta)
      Aumentar en 1 compresorActual

  Ordenar mapa de menor a mayor tiempoRespuesta
  CompresorSeleccionado = Compresor en posición 0 del mapa
```

4.2.4 Interfaz Gráfica de la Aplicación

4.2.4.1 Servidor

La Figura 4-25 presenta la interfaz gráfica principal del Servidor. En la cual es posible configurar los puertos de envío y recepción de información, seleccionar de forma estática el mecanismo de compresión a utilizar o dejar que sea el mediador quien lo decida aplicando el algoritmo de selección (está opción fue incluida para facilitar el desarrollo de las pruebas con el sistema). Los directorios en los que se almacenan las peticiones y respuestas también pueden ser modificados[1].

Figura 4-25. Interfaz gráfica Servidor

4.2.4.2 Cliente

El cliente permite modificar la dirección IP donde se encuentra alojado el servidor y el puerto del mismo, igualmente el puerto donde se encuentra expuesto el proveedor para el consumidor del servicio y los directorios donde se almacenan las peticiones y respuestas, la Figura 4-26 presenta la interfaz gráfica del Cliente.

[1] En el Anexo D y E se presentan la descripción interna del mediador y todo el proceso de configuración del sistema respectivamente.

Figura 4-26. Interfaz gráfica Cliente

4.3 VALIDACIÓN DEL PILOTO

En esta sección se describe todo el procedimiento llevado a cabo para obtener datos experimentales que permitieron validar el funcionamiento del sistema mediador, verificando que realmente facilita el consumo de un servicio Web en una red de baja capacidad, mediante la disminución de su tiempo de respuesta. El montaje realizado fue el descrito en la sección **¡Error! No se encuentra el origen de la referencia.**; se plantearon ocho escenarios diferentes en los que se varían las capacidades de procesamiento del equipo donde se ejecuta el Cliente y el ancho de banda de la red que lo conecta con el servidor. El servicio web a consumir fue el denominado *AXISVectorWS*[1] y específicamente la operación *getObjectArray.*

La variación de la capacidad de procesamiento se realizó apoyándose en la herramienta CPULimit (utilizada también en el capítulo 3 para realizar las pruebas de tiempo de compresión y descompresión) y para la variación del ancho de banda se utilizó una herramienta llamada WANem[2].

[1] Se describe en el Anexo A
[2] Wide Area Network emulator

4.3.1 Objetivo de las Pruebas.

El objetivo de estas pruebas es determinar la eficiencia del mediador en funcionamiento mediante la decisión que éste toma respecto al mecanismo de compresión que genere un menor tiempo de respuesta, basándose en los modelos matemáticos y en el algoritmo de selección descrito en 4.2.3.

4.3.2 Descripción del Procedimiento

En los ocho escenarios planteados se ejecutaron peticiones al servicio Web variando la cantidad de objetos solicitados. Inicialmente se dejó al mediador elegir el mecanismo de compresión a utilizar y luego se configuró para que la compresión se realizara con los demás compresores, adicionalmente se realizó el consumo del servicio Web sin utilizar el mediador. Los criterios seleccionados durante las pruebas realizadas fueron: tiempo de respuesta total, tiempo de compresión, tiempo de descompresión, tiempo de transmisión, radio de compresión, longitud en bytes de la respuesta original, longitud en bytes de la respuesta comprimida y el compresor seleccionado por el mediador. Se compararon los tiempos de respuesta total y se determinó si la decisión tomada por el mediador fue o no la correcta.

A continuación se describen los equipos utilizados en la simulación y la configuración de las dos herramientas utilizadas para el montaje de la misma.

4.3.2.1 Equipos.

Cliente	**Servidor**
Capacidad de Procesamiento	*Capacidad de Procesamiento*
Núcleos: 1	**Núcleos:** 8
Velocidad por Núcleo: 2087MHz	**Velocidad por Núcleo:** 2000MHz
Velocidad total: 2087MHz	**Velocidad total:** 16000MHz
Configuración de red	*Configuración de red*
IP: 192.168.190.32	**IP:** 192.168.19015

Para realizar las pruebas se utilizaron dos herramientas con el fin de facilitar la correcta implementación de los escenarios de pruebas; a continuación se explica la aplicación de dichas herramientas para variar el ancho de banda y limitar la capacidad de procesamiento; posteriormente, teniendo claro que las redes de baja capacidad y específicamente las redes VHF, se caracterizan por manejar anchos de banda entre los 2,5Kbps y 9,6Kbps, se determinó que las pruebas se realizarían en escenarios que varíen entre los 2 y los 10 Kbps. Con respecto a la capacidad de procesamiento, se tomaron cuatro casos particulares de variación desde los 128MHz hasta los 500MHz.

4.3.2.2 Limitación de la Capacidad de Procesamiento.

Al igual que en las pruebas de tiempo de compresión y descompresión realizadas en el capítulo 3, el equipo cliente es en realidad una máquina virtual. Las capacidades de procesamiento consideradas y los porcentajes utilizados en CPULimit son los que se muestran en la Tabla 4-3.

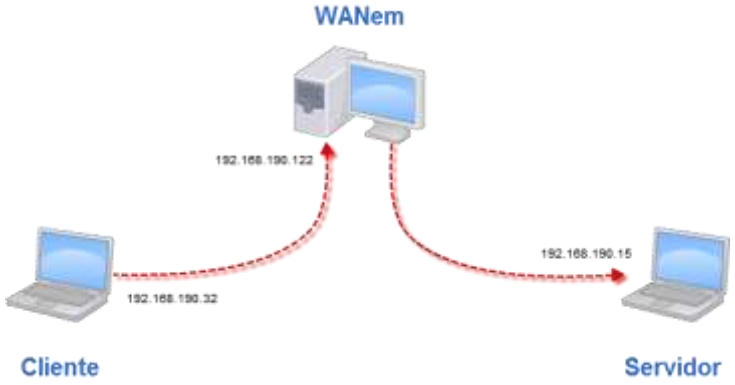
Tabla 4-3. Capacidades de procesamiento para el entorno de pruebas

Capacidad de procesamiento (MHz)	Porcentaje del procesador total
128	6.1315%
250	11.9789%
375	17.9683%
500	23.9578%

4.3.2.3 Variación del Ancho de Banda

WANem es un emulador de redes WAN que busca proporcionar una experiencia real a los desarrolladores de aplicaciones que requieren modificar los entornos de la red en los cuales se desempeñan las aplicaciones. Entre las características que permite configurar se encuentran: ancho de banda, retardo, jitter y desconexiones. Dichas pruebas pueden ser realizadas en una red LAN configurando a WANem como Gateway de conexión entre los equipos [124]. Para estas pruebas solo fue necesario modificar el ancho de banda de la conexión. Considerando que las redes de baja capacidad y específicamente las redes VHF, se caracterizan por manejar anchos de banda entre los 2,5Kbps y 9,6Kbps, se determinó que las pruebas se realizarían en escenarios que varíen entre los 2 y los 10 Kbps. La Figura 4-27 presenta las conexiones de red utilizando WANem.

Figura 4-27. Configuración de las conexiones utilizando WANem



4.3.2.4 Descripción de los Escenarios.

En la Tabla 4-4 se presentan los escenarios utilizados para la prueba. En ella se presenta la capacidad de procesamiento utilizada en el Cliente, el ancho de banda de la red, la longitud aproximada del mensaje de respuesta esperado y el mecanismo de compresión

que se espera el mediador elija (este Mecanismo Esperado responde a los resultados obtenidos en el capitulo anterior para pruebas en escenarios similares).

Tabla 4-4. Escenarios de prueba

Escenario	Capacidad Procesamiento	Ancho de Banda	Mensaje de Referencia Aprox.	Mecanismo Esperado
1	128MHz	2 Kbps	177,100 KB	BZip2
2	128MHz	10 Kbps	177,100 KB	GZip
3	250MHz	2 Kbps	770 B	EXIficient
4	250MHz	6 Kbps	770 B	GZip
5	375MHz	6 Kbps	18,000 KB	BZip2
6	375MHz	10 Kbps	18,000 KB	GZip
7	500MHz	2 Kbps	18,000 KB	XMLPPM
8	500MHz	6 Kbps	18,000 KB	BZip2

Para la selección de los escenarios se tuvo en cuenta el Mecanismo Esperado, presentado en la tabla anterior, con el fin de encontrar diferentes resultados en la elección del mecanismo de compresión aplicado y así poder demostrar la eficiencia del mediador en cada caso.

4.3.3 Resultados

A continuación, se presentan los resultados obtenidos para el escenario No. 1. Los resultados de los escenarios restantes pueden ser consultados en el *Anexo D*.

Para el escenario 1 se envió una solicitud de 765 objetos en la operación getObjetArray, dando como resultado una respuesta aproximada de 177,111KB. El mecanismo de compresión seleccionado por el mediador, utilizando el algoritmo de selección fue BZip2 y el tiempo de respuesta con este compresor fue de 140293ms. En la Tabla 4-5 se observan los resultados generados utilizando el sistema mediador con cada método de compresión, y sin utilizarlo (sin aplicar compresión).

De esta tabla se puede inferir que el menor tiempo de respuesta es el obtenido por el mecanismo de compresión BZip2, lo cual concuerda con la selección realizada por el mediador como el mejor caso posible. El tiempo de respuesta para el consumo del servicio sin utilizar el sistema mediador fue de 696754ms, aproximadamente cinco veces más que el tiempo requerido al utilizar el mediador. Lo anterior refleja claramente los beneficios de la solución propuesta; incluso si se aplica el mecanismo de compresión menos favorable, es decir, FastInfoset con un tiempo de respuesta total de 401733 existe una mejora en el consumo del servicio.

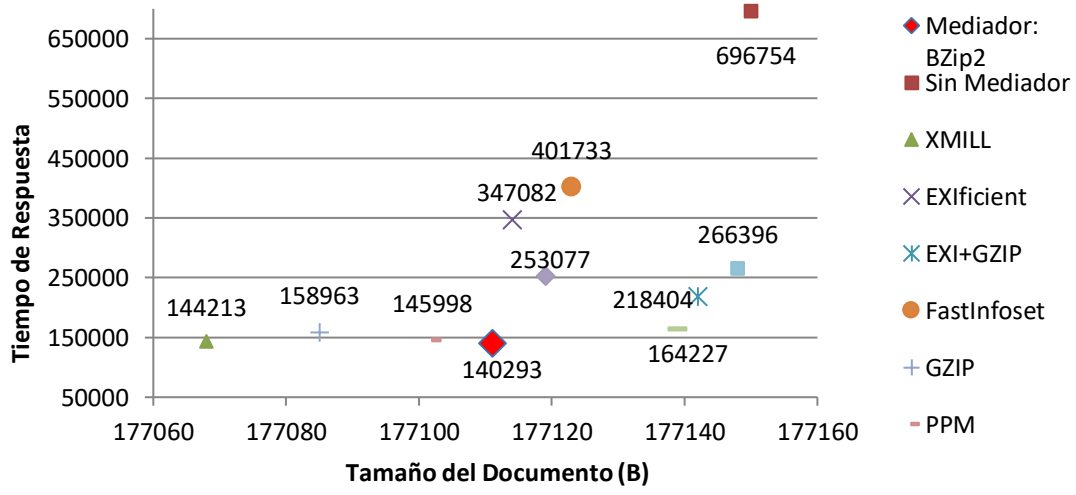
Tabla 4-5. Resultados obtenidos para la prueba en el escenario 1

UTILIZANDO EL SISTEMA MEDIADOR									
MECANISMO		Longitud Original (B)	Longitud Comprimida (B)	Radio Compresión	Tiempo de Compresión (ms)	Tiempo de Transmisión (ms)	Tiempo de Descompresión (ms)	Tiempo de Procesamiento (ms)	Tiempo Respuesta (ms)
1	Mediador: BZip2	177111	21994	0,12418201	60	86691	16262	37280	**140293**
2	EXIficient	177114	66436	0,37510304	563	293279	32484	20756	347082
3	EXI+GZIP	177142	39575	0,22340834	611	175715	3715	38363	218404
4	FastInfoset	177123	79739	0,45018998	379	364616	15298	21440	401733
5	GZIP	177085	31953	0,18043877	26	135774	497	22666	158963
6	PPM	177102	23816	0,13447618	47	92918	16170	36863	145998
7	SCMPPM	177139	19900	0,11234116	124	81674	32946	49483	164227
8	XMILL	177068	20686	0,11682517	114	81589	17494	45016	144213
9	XMLPPM	177119	19560	0,11043423	65	80999	120081	51932	253077
10	XWRT	177148	19864	0,11213223	401	174065	49536	42394	266396
SIN UTILIZAR EL SISTEMA MEDIADOR									
11	Sin Compresión	177150	---	---	0	696754	0	0	696754

El tiempo de procesamiento incluido en la tabla hace referencia a la diferencia entre el tiempo de respuesta total y la sumatoria de tiempos de compresión, transmisión y descompresión; éste tiempo de procesamiento es el utilizado por el mediador para realizar los procesos de agregación de encabezados, conversión de formatos, entre otros. Igualmente, se encuentra incluido el tiempo que tarda el proveedor del servicio en dar una respuesta y el tiempo de envío de la petición al servidor.

En la Figura 4-28 se puede apreciar gráficamente la comparación de los tiempos de respuesta arrojados por cada mecanismo de compresión.

Figura 4-28. Tiempos de respuesta generados en el mediador



Adicionalmente, es claro que algunos mecanismos de compresión presentan mejores radios de compresión pero sus tiempos de compresión y/o descompresión son desfavorables lo que hace que el mejor criterio de selección sea el tiempo de respuesta, pues este incluye una combinación de las métricas para la descripción de los mecanismos de compresión.

La Tabla 4-6 resume los 8 escenarios de pruebas realizados.

Tabla 4-6. Resumen de resultados del plan de pruebas

Escenario	Capacidad Procesamiento	Ancho de Banda	Mensaje de Referencia Aprox.	Mecanismo Esperado	Mecanismo Seleccionado
1	128MB	2 Kbps	177 KB	BZip2	BZip2
2	128MB	10 Kbps	177 KB	GZip	GZip
3	250MB	2 Kbps	770 B	EXIficient	EXIficient
4	250MB	6 Kbps	770 B	GZip	GZip
5	375MB	6 Kbps	18 KB	BZip2	BZip2
6	375MB	10 Kbps	18 KB	GZip	GZip
7	500MB	2 Kbps	18 KB	XMLPPM	XMLPPM
8	500MB	6 Kbps	18 KB	BZip2	BZip2

4.3.4 Conclusiones

Con las pruebas realizadas en los diferentes escenarios se demuestra que el mediador en todos los casos toma la mejor decisión en cuanto al tiempo de respuesta, basándose en el modelo matemático descrito en la sección 3.5 del capítulo anterior.

Aunque pueden existir múltiples comportamientos del mediador, debido a la diversidad de descripciones del ancho de banda de la red del cliente o de su capacidad de procesamiento, se puede ver que los mecanismos de compresión más usuales son BZip2, EXIficient, GZip, XMill o XMLPPM.

El tiempo total en consumir el servicio Web sin aplicar ninguna compresión, supera aproximadamente 5 veces al tiempo de respuesta entregado por la herramienta de compresión seleccionada por el mediador, confirmando que el sistema implementado genera una mejora en el acceso al servicio Web desde una red de baja capacidad.

5. APORTES, CONCLUSIONES Y TRABAJO FUTURO

5.1 RESUMEN

Los beneficios asociados a los servicios Web, al permitir a las aplicaciones en desarrollo y a las ya existentes exponer sus funcionalidades para que puedan ser aprovechadas en puntos donde realmente se necesitan, se extienden no sólo a entornos empresariales sino a entornos rurales, civiles y militares, mejorando la interacción entre aplicaciones y agilizando el acceso a la información. En entornos civiles la prestación de servicios de salud y educación, entre otros, puede ser ampliamente mejorada mediante la inclusión de servicios Web; mientras que en entornos militares el ágil acceso a la información puede generar una gran ventaja dentro del campo de batalla, al facilitar la comunicación entre unidades aliadas. No obstante, el acceso a Servicios Web se dificulta en entornos donde las características topográficas obligan a la utilización de infraestructuras de red con limitadas tasas de transferencia y por tanto, bajas velocidades. La principal razón de este problema, se debe al lenguaje de representación de datos empleado por los servicios Web basados en SOAP. Si bien, XML se caracteriza por ser un lenguaje fácilmente interpretado, la utilización de etiquetas y su representación en texto plano son factores que aumentan considerablemente la carga de información que debe ser transmitida.

De esta forma, el diseño e implementación de un sistema mediador que integre mecanismos de compresión que ayuden a disminuir el tamaño de la información transmitida, se convierte en una solución viable en la búsqueda por facilitar el acceso a dichos servicios desde redes con capacidad limitada; no obstante, existen diferentes tipos de mecanismos de compresión que varían su efectividad y eficiencia de acuerdo a los componentes hardware del equipo donde se ejecuten; haciendo necesario el uso de algunos parámetros de entrada a la hora de elegir el mejor mecanismo de compresión en cada caso. Dichos parámetros representan información descriptiva sobre el servicio Web y sobre el cliente que intenta accederlo, los cuales pueden ser representados a través de metadatos.

Partiendo de este contexto, el presente trabajo de grado presenta la arquitectura para un sistema mediador que busca dar solución al problema planteado. El sistema en cuestión, ejecuta dos tareas importantes: la primera, es crear y consumir metadatos con información descriptiva de los clientes y la segunda, es a partir de los metadatos determinar la herramienta de compresión encargada de procesar los mensajes SOAP. De esta manera, se han desarrollado un conjunto de pruebas que buscan determinar los parámetros que afectan el comportamiento de las herramientas de compresión seleccionadas. A partir de los resultados obtenidos, se diseñan modelos matemáticos que describen el comportamiento de las herramientas y se determina la información que debe ir incluida en el metadato. Estos modelos y la información contenida en los metadatos

permiten diseñar la lógica para seleccionar la herramienta de compresión que mejor se ajuste a las capacidades del cliente.

Finalmente, se desarrolla un piloto de pruebas que implementa la lógica de selección planteada y valida las capacidades del mediador, demostrando que realmente facilita el acceso a servicios Web basados en SOAP desde redes de baja capacidad, disminuyendo considerablemente el tiempo necesario para su consumo.

5.2 APORTES

1. *Base de conocimiento en torno a la clasificación de herramientas de compresión y a su desempeño frente a documentos XML SOAP.* La selección de herramientas de compresión utilizadas dentro del mediador están soportadas, no solo sobre resultados experimentales, sino también sobre una base teórica que permitió, inicialmente realizar una clasificación de éstas de una forma un poco mas completa que las ya existentes; Sobre este proceso, se puede concluir:

   - Las herramientas de compresión pueden ser clasificadas de acuerdo a cuatro características: su conocimiento de la estructura de los documentos XML, su capacidad para soportar consultas, el tipo de codificación utilizada y la utilización de documentos de esquema.

   - El desempeño de una herramienta de compresión puede ser descrito a partir de tres métricas. El radio de compresión, que se refiere a la efectividad de la herramienta, es decir, qué porcentaje logra reducir el tamaño del documento; el tiempo de compresión y el tiempo de descompresión, que indican la eficiencia de la herramienta y permiten determinar cuánto tarda en realizar el proceso de compresión y descompresión, respectivamente.

   - Las herramientas de compresión capaces de soportar consultas han sido diseñadas para aumentar el tiempo de procesamiento de los documentos comprimidos, al permitir obtener la información del documento sin realizar todo el proceso de descompresión. Su principal desventaja es que no logran radios de compresión relevantes.

   - Las herramientas de compresión basadas en el tipo de codificación (Binary XML) basan su funcionamiento en el almacenamiento de datos de forma binaria y no en texto plano, de esta forma logran reducir el tamaño del documento, puesto que los datos binarios siempre requieren de un menor tamaño que la representación en texto.

- Por el conocimiento de la estructura del documento XML, las herramientas de compresión pueden dividirse en herramientas de texto de uso general y herramientas conscientes XML, estas últimas utilizan un proceso de restructuración de la información y luego aplican herramientas de uso general a subcomponentes del documento restructurado, de esta forma logran mejores radios de compresión.

2. *Determinación de las características que afectan las métricas de desempeño de las herramientas de compresión.* Basado en las métricas: radio de compresión, tiempo de compresión y tiempo de descompresión, se realizaron un conjunto de pruebas que permitieron determinar cuáles de las características del equipo donde se ejecuta la herramienta de compresión y del documento SOAP que se intenta comprimir afectan de alguna manera el proceso de compresión. Al respecto, se puede concluir:

   - En cuanto a la API utilizada para implementar los servicios Web en Java, se puede concluir que AXIS presenta un mejor comportamiento por encima de METRO, cuando es expuesta a las herramientas de compresión. Una razón posible para que esto suceda, es que AXIS agrega mayor carga a los datos, considerándose más verboso.

   - El tipo de dato y el estilo contenido dentro del mensaje SOAP, no son parámetros que influyen en la selección de las herramientas de compresión; en contraste, la longitud del documento y la capacidad de procesamiento, si son parámetros que afectan el radio de compresión y el tiempo de compresión y descompresión, respectivamente.

3. *Definición de metadatos que permitan describir las características del dispositivo cliente para acceder a los servicios.* A partir de una base de conocimiento en torno a la selección del metalenguaje y sistema de metadatos más acorde a las necesidades del sistema mediador, se realiza una comparación teórica entre los diversos lenguajes y estructuras de metadatos existentes, que se adapten de la mejor manera al contexto planteado en el proyecto. Adicionalmente, basándose en los resultados arrojados en el aporte 2, se determina el tipo de información que debe ir contenida dentro del metadato, logrando su completa definición, útil para la selección del mecanismo de compresión.

   - A partir de las pruebas realizadas por el componente Motor de Compresión, se concluye que las características que deben ir contenidas dentro del metadato son: la dirección IP que identifique al cliente, el ancho de banda al cual el consumidor tiene acceso y la capacidad de procesamiento con la que cuenta el dispositivo cliente.

   - Dado el énfasis en la descripción de recursos y la facilidad de adopción del lenguaje, se elije a XML como el metalenguaje de adopción para el proyecto.

- Aunque existe diversidad de estructuras de metadatos, RDF presenta mayor simplicidad y flexibilidad en la descripción de recursos, comparado con MARC y TEI; adicionalmente, posee menos calificadores dentro de su estructura en comparación con Dublin Core. De esta forma, RDF es adoptado como el sistema de metadato para el proyecto, por las anteriores comparaciones y por su énfasis en el descubrimiento de los metadatos, sin descuidar la importancia que tiene la descripción de recursos.

4. *Generación de modelos matemáticos que permiten describir el comportamiento de las herramientas de compresión.* Al realizar las pruebas que permitieron determinar el desempeño de las herramientas de compresión, se estableció que su comportamiento, en el contexto de los documentos SOAP XML puede ser descrito a partir de funciones matemáticas. Con base en esta información, se concluye que:

   - El radio de compresión de las herramientas, cuando se varía la longitud del documento, no puede ser descrito de forma precisa utilizando un único modelo matemático. Es necesaria la combinación de diferentes modelos con el fin de aproximar a |1|, el nivel de correlación cuando se aplica regresión simple lineal, y a 1 (o 100%) el coeficiente de determinación R^2 para los modelos de regresión simple no lineales.

   - El tiempo de compresión y descompresión, al variar la longitud del documento, tiene un comportamiento lineal simple, de forma que puede ser modelado a partir de una ecuación de línea recta; sin embargo al variar la capacidad de procesamiento se presenta una variación en la pendiente y en la ordenada de dicha línea. De esta manera, el procedimiento para obtener un modelo matemático que describa estas métricas, se resume entonces, en encontrar funciones que permitan obtener los valores de pendiente y ordenada a partir de la capacidad de procesamiento.

5. *Lógica para la selección automática de herramientas de compresión que mejor se ajusten a las capacidades del cliente, de acuerdo a la información contenida en los metadatos.* La información contenida en los metadatos utilizada como un parámetro de entrada dentro de los modelos matemáticos, permitió la generación de una lógica de selección de herramientas de compresión, con un alto nivel de precisión en el contexto de los documentos XML SOAP.

   - La herramienta de compresión que genere un mejor desempeño en un contexto específico, puede ser considerada como la que origine un menor tiempo de respuesta a la hora de consumir el servicio en dicho contexto. El tiempo de respuesta está condicionado por cuatro valores: el tiempo de compresión y descompresión, el tiempo de transmisión (que puede ser hallado a partir del radio de compresión y el ancho de banda) y un tiempo adicional que incluye: tiempo de

procesamiento por parte del proveedor, y tiempo de procesamiento para realizar las tareas internas dentro del sistema mediador.

- Gracias a la información contenida dentro de los metadatos, se logra determinar el tiempo de transmisión y los tiempos de compresión y descompresión. De esta manera se concluye que a partir de la descripción contenida en los metadatos se logra seleccionar el mejor mecanismo de compresión.

6. *Diseño de una arquitectura base para la prestación de servicios Web desde redes baja capacidad, que incluye: herramientas de compresión, metadatos con información descriptiva y demás módulos necesarios para garantizar el consumo del servicio.* Aunque la lógica de selección de las herramientas de compresión, se puede considerar como el núcleo de funcionamiento del sistema mediador, se deben generar otros módulos con el fin de realizar un consumo satisfactorio del servicio.

   - Para que el consumo del servicio se pueda realizar de forma efectiva y transparente, se deben implementar dos módulos en el Cliente y en el Servidor, denominados Imagen del proveedor e Imagen del Consumidor. El primero hace las veces de proveedor del servicio pero ubicado en el lado del consumidor y se encarga de mantener la conexión hasta la entrega efectiva de las respuestas. El segundo, ubicado del lado del Servidor, permite conectarse con el proveedor final del servicio para realizar las peticiones.

   - Es necesaria la inclusión de un módulo denominado Procesador de Contenidos, tanto en el Cliente como en el Servidor. Éste módulo se encarga de transformar las peticiones y respuestas en un formato adecuado para que el modulo siguiente las pueda utilizar. Esta transformación de mensajes se refiere ya sea a la compresión, descompresión, u obtención de arreglos de bits para ser enviados a través del canal de datos.

   - Para la comunicación entre el cliente y el servidor, se debe desarrollar un módulo de conexión encargado de enviar las peticiones/respuestas, desde y hacia, el procesador de contenidos y la imagen del consumidor del servicio.

7. *Construcción de un piloto que demuestra el funcionamiento de la lógica de selección de las herramientas de compresión planteada y de la arquitectura base propuesta.* Se desarrollaron las aplicaciones del cliente y servidor planteadas en la arquitectura base.

   - Con el fin de generar tiempos de procesamiento bajos y evitar cuellos de botella en la transmisión, se ha construido el piloto distribuido en componentes encargados de realizar tareas simples, permitiendo que todo el proceso de consumo de un servicio no se realice en un solo componente.

- Utilizar un procesamiento basado en colas permite que cada componente al terminar su proceso no entregue directamente el resultado al componente siguiente, en su lugar, ubica su resultado en una cola FIFO y el componente siguiente se encarga de atender los objetos contenidos en dicha cola, de forma que al terminar de procesar uno, continúa con el siguiente de la cola y se logra que todas las peticiones sean procesadas.

8. *Validación del sistema mediador a partir de la implementación de un entorno de pruebas basado en el contexto del proyecto.* A partir de las aplicaciones generadas en la aporte 7, se establece un entorno de pruebas en el que se limitan las capacidades de procesamiento del dispositivo cliente y el ancho de banda de la conexión.

   - El tiempo total en consumir el servicio Web sin aplicar ninguna compresión, supera aproximadamente cinco veces al tiempo de respuesta entregado por la herramienta de compresión seleccionada por el mediador, confirmando que el sistema implementado genera una mejora sustancial en el acceso al servicio Web desde una red de baja capacidad.

   - Con las pruebas realizadas en los diferentes escenarios simulados se demuestra que el sistema mediador, utilizando una lógica de selección basada en los modelos matemáticos definidos en el proyecto, toma, en todos los casos, la mejor decisión en cuanto al tiempo de respuesta generado al consumir el servicio.

5.3 TRABAJO FUTURO

- Como trabajo futuro se propone crear los metadatos de manera automática. Actualmente el proyecto presenta la definición del metadato que cumple con la funcionalidad de describir las características necesarias para la selección del mecanismo de la herramienta de compresión, pero su creación es manual.

- En entornos militares, se propone aplicar procesos de filtrado, codificación y encriptación de información, acorde a las necesidades de este contexto.

- Las pruebas realizadas hasta el momento se han hecho en entornos simulados. Como trabajo futuro se propone realizar pruebas en entornos reales.

REFERENCIAS

[1] Microsoft, "La arquitectura orientada a servicios (SOA) de Microsoft aplicada al mundo real," *Microsoft Corporation,* 2006.

[2] T. Erl, *SOA: Principles of service design*: Prentice Hall, 2007.

[3] S. St Laurent, J. Johnston and E. Dumbill, *Programming Web services with XML-RPC*, 1st ed. Beijing ; Sebastopol, Calif.: O'Reilly, 2001.

[4] J. Sliwa and M. Amanowicz, "A mediation service for Web Services provision in tactical disadvatnaged environment," *IEEE Xplore,* pp. 1-7, 2008.

[5] K. Lund*, et al.*, "Using web services to realize service oriented architecture in military communication networks," *Communications Magazine, IEEE,* vol. 45, pp. 47-53, 2007.

[6] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler and F. Yergeau, "Extensible markup language (XML) 1.0," *W3C recommendation,* vol. 6, 2000.

[7] M. Tian, T. Voigt, T. Naumowicz, H. Ritter and J. Schiller, "Performance considerations for mobile web services," *Computer communications,* vol. 27, pp. 1097-1105, 2004.

[8] C. Werner, C. Buschmann, Y. Brandt and S. Fischer, "Compressing SOAP messages by using pushdown automata," *IEEE Xplore,* pp. 19-28, 2006.

[9] W. Hanslo and K. MacGregor, "The efficiency of XML as an intermediate data representation for wireless middleware communication," presented at the Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries, Stellenbosch, Western Cape, South Africa, 2004.

[10] DF. Manquillo and A. Rendón, "Access to Digital Libraries from Low Speed Disconnected Environments," *IEEE Latin America Transactions,* vol. 8, 2010.

[11] FITEL. Proyecto Piloto en Telecomunicaciones, Date accessed: July 13, 2011 [Online]. Available: http://www.fitel.gob.pe/contenido.php?ID=49

[12] Ministerio de Comunicaciones de Colombia. Programa Compartel, Date accessed: August 1st, 2012 [Online]. Available: http://www.e-colombia.com.co/web/spip.php?rubrique53

[13] EHAS. Enlace Hispano Americano de Salud, Date accessed: August 1st, 2012 [Online]. Available: http://www.ehas.org

[14] Gilat Perú. Portal Rural, Date accessed: August 1st, 2012 [Online]. Available: http://www.portalrural.org.pe/

[15] V. Jodalen, A. Eggen, B. Solberg and O. Gronnerud, "Military messaging in IP networks using HF links," *Communications Magazine, IEEE,* vol. 42, pp. 98-104, 2004.

[16] J. F. Goetzmann, "XML-Extensible Markup Language," *Unpublished,* 2006.

[17] D. Booth*, et al.*, "Web services architecture," *W3C Working Group Note,* vol. 11, 2004.

[18] H. Kreger, "Web services conceptual architecture," *IBMTechnical Report WCSA,* vol. 1, 2001.
[19] R. N. Marset, "REST vs Web Services," *Unpublished,* 2007.
[20] E. Newcomer, *Understanding Web Services: XML, Wsdl, Soap, and UDDI*: Addison-Wesley Professional, 2002.
[21] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana, "Web services description language (WSDL) 1.1," in *W3C Note*, ed: Citeseer, 2001.
[22] C. Mateu, *Desarrollo de aplicaciones web*: Universitat Oberta de Catalunya, 2004.
[23] D. Box*, et al.*, "Simple Object Access Protocol (SOAP) 1.1," in *W3C Note*, ed, 2000.
[24] R. Englander, *Java and SOAP*: O'Reilly & Associates, Inc., 2002.
[25] W. W. W. Consortium, "XML Schema Part 2: Datatypes," *W3C XML Schema, October,* 2000.
[26] APIS Networks. SOAP Data Types, Date accessed: August 1st, 2012 [Online]. Available: http://old.apisnetworks.com/soap-data-types.php
[27] R. Butek. Which style of WSDL should I use, Date accessed: August 1st, 2012 [Online]. Available: https://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/
[28] D. Winer. Using XML-RPC for Web services: Getting started with XML-RPC in Perl, Date accessed: August 1, 2012 [Online]. Available: http://www.ibm.com/developerworks/library/ws-xpc1/?dwzone=ws
[29] T. Bellwood*, et al.*, "UDDI Version 3.0," *Published specification, Oasis,* vol. 5, 2002.
[30] R. Hodgman. SOA Case Study: Leveraging SOA in Business Processes, Date accessed: August 1st, 2012 [Online]. Available: http://soa.sys-con.com/node/453121
[31] I. N. Mba and M. O. Adigun, "Comparative Study of Web Services Platforms in a GUISET Environment," 2011.
[32] D. Jayasinghe and A. Azeez, *Apache Axis2 Web Services*: Packt Pub Limited, 2011.
[33] R. H. N. Balani and R. Hathi, *Apache CXF Web Service Development*: Packt Pub., 2009.
[34] D. Sosnoski. Java Web services: Introducing Metro, Date accessed: August 1st, 2012 [Online]. Available: http://www.ibm.com/developerworks/java/library/j-jws9/index.html
[35] M. S. A. Gibb, J. Michalak and J. Wieselthier, "Information Management over Disadvantaged Grids," *Final rep., RTO Info. Sys. Technology Panel, Task Group IST-030/RTG-012, RTO-TR-IST-030,* 2007.
[36] A. R. C. Lechtaler and R. J. Fusario, *Teleinformatica para Ingenieros en Sistemas de Informacion 2*: Reverté, 1999.
[37] F. T. Johnsen and T. Hafsøe, "Using NFFI Web services on the tactical level: An evaluation of compression techniques," presented at the 13th ICCRTS: C, 2008.
[38] H. Liefke and D. Suciu, "XMill: an efficient compressor for XML data," *ACM SIGMOD Record,* vol. 29, pp. 153-164, 2000.

[39] S. Sakr, "An Experimental Investigation of XML Compression Tools," *Arxiv preprint arXiv:0806.0075,* 2008.
[40] S. Sakr, "XML compression techniques: A survey and comparison," *Journal of Computer and System Sciences,* vol. 75, pp. 303-322, 2009.
[41] N. Ziviani, E. S. de Moura, G. Navarro and R. Baeza-Yates, "Compression: A key for next-generation text retrieval systems," *Computer,* vol. 33, pp. 37-44, 2000.
[42] R. Stallman, "The GNU project, Date accessed: August 1st, 2012," ed, 1998.
[43] P. Deutsch, "RFC1952: GZIP file format specification version 4.3," in *Internet RFCs*, ed, 1996.
[44] L. P. Deutsch, "DEFLATE compressed data format specification version 1.3," ed: IETF, 1996.
[45] D. Dubé and V. Beaudoin, "Improving LZ77 data compression using bit recycling," presented at the International Symposium on Information Theory and its Applications, Seoul, Korea, 2006.
[46] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE,* vol. 40, pp. 1098-1101, 1952.
[47] A. Moffat, "Implementing the PPM data compression scheme," *Communications, IEEE Transactions on,* vol. 38, pp. 1917-1921, 2002.
[48] H. Friedrich, "Prediction per Partial Matching," *Unpublished,* 2009.
[49] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm, Date accessed: August 1st, 2012 [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.141.5254
[50] J. A. Rodríguez, "La estructura de los documentos en el ámbito de recuperación de información: Propuesta para su comprensión, indexación y recuperación," Phd thesis, Universidad de Valladolid, 2004.
[51] X. Meng, Y. Jiang, Y. Chen and H. Wang, "XSeq: an indexing infrastructure for tree pattern queries," presented at the international conference on Management of data, 2004.
[52] P. Skibiński and J. Swacha, "Combining efficient XML compression with query processing," *Lecture Notes in Computer Science,* vol. 4690, pp. 330-342, 2007.
[53] P. Skibinski. XWRT: Word Replacing Transform for eXtended Markup Language, Date accessed: August 1st, 2012 [Online]. Available: http://sourceforge.net/projects/xwrt
[54] G. Graefe and L. D. Shapiro, "Data compression and database performance," *IEEE Xplore,* vol. Symposium on applied computing, pp. 22-27, 1991.
[55] S. T. J. Kangasharju and T. Lindholm, "Xebu: A binary format with schema-based optimizations for XML data," *Web Information Systems Engineering–WISE 2005,* pp. 528-535, 2005.
[56] P. Sandoz and S. Pericas-Geertsen. Fast Infoset, Date accessed: August 1st, 2012 [Online]. Available: http://java.sun.com/developer/technicalArticles/xml/fastinfoset/

[57] Agile Delta. Efficient XML, Date accessed: [Online]. Available: http://www.agiledelta.com/product_efx.html

[58] B. Martin and B. Jano, "Wap binary xml content format," *World Wide Web Consortium. W3C NOTE,* 1999.

[59] J. Cheney. DTDPPM: DTD-Conscious Compression, Date accessed: August 1st, 2012 [Online]. Available: http://xmlppm.sourceforge.net/dtdppm/index.html.

[60] M. Girardot and N. Sundaresan, "Millau: an encoding format for efficient representation and exchange of XML over the Web," *Computer Networks,* vol. 33, pp. 747-765, 2000.

[61] D. Peintner. EXIficient - open source implementation of the W3C Efficient XML Interchange (EXI) format, Date accessed: August 1st, 2012 [Online]. Available: http://exificient.sourceforge.net/

[62] D. Peintner, H. Kosch and J. Heuer, "Efficient XML interchange for rich internet applications," *IEEE Xplore,* pp. 149-152, 2009.

[63] M. Bordese, "Análisis y alternativas para la compresión de XML," Master's thesis, FaMAF, Universidad Nacional de Córdoba, Argentina, 2009.

[64] J. A. Senso and A. R. Piñero, "El concepto de metadato. Algo más que descripción de recursos electrónicos," *Ciência da informação,* vol. 32, pp. 95-106, 2003.

[65] F. F. Martínez. Metadatos y Organización de Recursos Electrónicos, Date accessed: August 1st, 2012 [Online]. Available: http://mail.udgvirtual.udg.mx/biblioteca/handle/20050101/848

[66] C. V. Paulus. Metadatos: Introducción e Historia, Date accessed: July 7, 2011 [Online]. Available: http://users.dcc.uchile.cl/~cvasquez/introehistoria.pdf

[67] M. Rosetto, "Introdución a Metadatos y Formatos de Metadatos," presented at the Seminario sobre los Manifiestos IFLA sobre Bibliotecas Públicas,
Escolares e Internet, 2007.

[68] V. Ortiz-Repiso Jiménez, "Nuevas perspectivas para la catalogación: metadatos versus MARC," *Revista española de documentación científica,* vol. 22, pp. 198-219, 1999.

[69] T. Gill, A. J Gilliland and M. Woodley, "Introduction to Metadata, Pathways to Digital Information," *Getty Information Institute,* 2009.

[70] J. Milstead and S. Feldman, "Metadata: Cataloging by Any Other Name," *ONLINE-WESTON THEN WILTON-,* vol. 23, pp. 24-31, 1999.

[71] I. I. O. f. Standardization), "ISO 8879:1986 - Standard Generalized Markup Language (SGML)," ed, 1986.

[72] M. J. Lamarca, "Hipertexto: El nuevo concepto de documento en la cultura de la imagen.," Phd thesis, Facultad de Ciencias de la Información. Dpto. de Biblioteconomía y Documentación, 2006.

[73] I. Daudinot Founier, "Organización y recuperación de información en Internet: teoría de los metadatos," *Acimed,* vol. 14, pp. 0-0, 2006.

[74] L. Dempsey and R. Heery, "A review of metadata: a survey of current resource description formats," *The UK Office for Library and Information Networking,* 1997.

[75] R. Heery, "Review of metadata formats," *Program: electronic library and information systems,* vol. 30, pp. 345-373, 1996.
[76] R. M. Heery, *What Is... RDF?*: Ariadne, 1998.
[77] S. G. Martín and S. M. Angelozzi, "Análisis y comparación de metadatos para la descripción de recursos electrónicos en línea," presented at the III Encuentro Internacional de Catalogadores, Buenos Aires, 2011.
[78] F. Manola and E. Miller, "RDF Primer," in *W3C Recommendation*, ed, 2007.
[79] G. Klyne and J. J. Carroll, "RDF Concepts and Abstract Syntax," in *W3C Recommendation*, ed, 2004.
[80] D. Beckett and B. McBride, "RDF/XML Syntax Specification (revised)," in *W3C recommendation* vol. 10, ed, 2004.
[81] P. Hayes and B. McBride, "RDF Semantics," in *W3C recommendation*, ed, 2004.
[82] D. Brickley and R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," in *W3C recommendation* vol. 10, ed, 2004.
[83] J. Grant and D. Beckett, "RDF Test Cases," in *W3C recommendation, February*, ed, 2004.
[84] F. Galton, "NATURAL INHERITANCE. 1889," *SERIES E: PHYSIOLOGICAL PSYCHOLOGY,* 1973.
[85] A. Kateki, "Polish Automated System for Air Forces-Polish Way to Introduce Network Enabled Capability," presented at the International Radar Symposium, 2006.
[86] F. Johnsen, A. Eggen, T. Hafsøe, and K. Lund, "Utilizing military message handling systems as a transport mechanism for SOA in military tactical networks," *Norwegian Defence Research Establishment (FFI),* 2008.
[87] M. Tian*, et al.*, "Performance considerations for mobile web services," *Computer communications,* vol. 27, pp. 1097-1105, 2004.
[88] T. Podlasek, J. Sliwa, J. and M. Amanowicz, "Efficiency of compression techniques in SOAP1," *Military Comunications Institute,* 2010.
[89] C. Augeri, D. Bulutoglu, B. Mullins, R. Baldwin and L. Baird, "An analysis of XML compression efficiency," *Experimental computer science,* pp. 1-7, 2007.
[90] J. Usero, "El uso de metadatos para mejorar la interoperabilidad del conocimiento en los servicios de administración electrónica," *El profesional de la información,* vol. 15, pp. 114-126, 2006.
[91] R. Haakseth*, et al.*, *Experiment report:" SOA-Cross Domain and Disadvantaged Grids": NATO CWID 2007*: FFI, 2007.
[92] J. Cheney, "Compressing XML with multiplexed hierarchical PPM models," *IEEE Xplore,* vol. Data compression conference, pp. 163-172, 2002.
[93] J. D. G. Leighton and T. Muldner, "AXECHOP: a grammar-based compressor for XML," *IEEE Xplore,* pp. 29-31, 2005.
[94] J. Seward. BZip2 Compressor, Date accessed: August 1st, 2012 [Online]. Available: http://www.bzip.org

[95] V. Toman, "Compression of XML data," *Master's thesis, Charles University, Prague,* 2004.

[96] P. Sandoz, A.Triglia and S. Pericas-Geertsen. Fast Infoset. Sun Developer Network Technical Article, Date accessed: August 1st, 2012 [Online]. Available: http://java.sun.com/developer/technicalArticles/xml/fastinfoset/

[97] J.-l. Gailly and M. Adler. GZip Compressor, Date accessed: August 1st, 2012 [Online]. Available: http://www.gzip.org/

[98] T. Kamiya. OpenEXI Project, Date accessed: August 1st, 2012 [Online]. Available: http://openexi.sourceforge.net/

[99] C. League. rngzip Compressor, Date accessed: August 1st, 2012 [Online]. Available: http://contrapunctus.net/league/haques/rngzip/.

[100] J. Adiego. SCMPPM Compressor, Date accessed: August 1st, 2012 [Online]. Available: http://www.infor.uva.es/~jadiego/files/scmppm-0.93.3.zip

[101] G. Leighton, J. Diamond and T. Müldner, "TREECHOP: A Tree-based Query-able Compressor for XML," *Jodrey School of Computer Science Technical Report,* vol. 5, 2005.

[102] P. Shankar. XAUST Compressor, Date accessed: August 1st, 2012 [Online]. Available: http://drona.csa.iisc.ernet.in/~priti/xaust.tar.gz

[103] P. M. Tolani and J. R. Haritsa, "XGRIND: A query-friendly XML compressor," *IEEE Xplore,* vol. Data engineering, pp. 225-234, 2002.

[104] J. Cheney. XMLPPM: XML-Conscious PPM, Date accessed: August 1st, 2012 [Online]. Available: http://xmlppm.sourceforge.net/

[105] H. Wang, J. Li, J. Luo and Z. He, "XCpaqs: compression of XML document with XPath query support," *IEEE Xplore,* vol. 1, pp. 354-358, 2004.

[106] J. Min, M. Park and C. Chung, "XPRESS: a queriable compression for XML data," in *International conference on Management of data*, 2003, pp. 122-133.

[107] A. Arion, A. Bonifati, G. Costa, S. D'Aguanno, I. Manolescu and A. Pugliese, "XQueC: Pushing queries to compressed XML data," in *international conference on Very large data bases*, 2003, pp. 1065-1068.

[108] W. Ng, W. Lam and J. Cheng, "Comparative analysis of XML compression technologies," *World Wide Web,* vol. 9, pp. 5-33, 2006.

[109] M. Barbos and E. Pop, "Web Services for Ubiquitous Mobile Device Applications," presented at the International Conferences on Advanced Service Computing, November 21, 2010, 2010.

[110] Sun Microsystems inc. Primitive Data Types, Date accessed: August 1st, 2012 [Online]. Available: http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

[111] A. Marletta. CPU limit, Date accessed: August 1st, 2012 [Online]. Available: http://cpulimit.sourceforge.net/

[112] Sun Microsystems inc. VirtualBox, Date accessed: August 1st, 2012 [Online]. Available: https://www.virtualbox.org/

[113] Equipo de desarrollo de Xfce. Entorno de Escritorio Xfce, Date accessed: August 1st, 2012 [Online]. Available: http://www.xfce.org

[114] OpenLand.es. Statgraphics | Software de análisis de datos estadístico y gráfico, Date accessed: August 1st, 2012 [Online]. Available: http://www.statgraphics.net/

[115] W. R. Stevens, *TCP/IP Illustrated: the protocols* vol. 1: Addison-Wesley Professional, 1994.

[116] E. Hewitt, *Java SOA cookbook*: O'Reilly Media, 2009.

[117] The Apache Software Foundation. The Apache Xerces Project - Charter, Date accessed: August 1st, 2012 [Online]. Available: http://xerces.apache.org/charter.html

[118] Sun Microsystems inc. JAXP Reference Implementation, Date accessed: August 1st, 2012 [Online]. Available: http://jaxp.java.net/

[119] E. R. Harold, *Processing XML with Java: a guide to SAX, DOM, JDOM, JAXP, and TrAX*: Addison-Wesley Professional, 2003.

[120] Apache Software Foundation. Apache Jena, Date accessed: August 1st, 2012 [Online]. Available: http://jena.apache.org/

[121] Sun Microsystems Inc. SOAPMessage (Java EE SDK), Date accessed: August 1st, 2012 [Online]. Available: http://glassfish.java.net/nonav/javaee5/api/javax/xml/soap/SOAPMessage.html#setProperty%28java.lang.String,%20java.lang.Object%29

[122] T. Greanier, "Discover the secrets of the Java Serialization API," *Sun Developer Network.,* 2000.

[123] B. Goetz. Java theory and practice: Thread pools and work queues, Date accessed: August 1st, 2012 [Online]. Available: http://www-106.ibm.com/developerworks/library/j-jtp0730.html

[124] T. C. Services. WANem 1.1 Wide Area Network Emulator - Setup Guide, Date accessed: August 1st, 2012 [Online]. Available: http://sourceforge.net/projects/wanem/files/Documents/WANemv11-Setup-Guide.pdf/download?use_mirror=hivelocity