Programa Oficial de Postgrado en Ciencias, Tecnología y Computación

Máster en Computación

Facultad de Ciencias – Universidad de Cantabria

TESIS DE MÁSTER

ANÁLISIS DE CALIDAD DE SERVICIO DE MIDDLEWARES ASOCIADOS A LA NORMA IEC61850

Unai Díaz de Cerio Urain
udiazcerio@ikerlan.es

Director:
Aitor Urbieta Arteche
IK4-Ikerlan
Área de Tecnologías del Software

Tutor:
Michael González Harbour
Departamento de Electrónica y Computadores
Grupo de Computadores y Tiempo Real

Curso 2012 / 2013

Santander, Noviembre de 2012

Agradecimientos

A IK4-Ikerlan, y en especial a Juanpe, por la oportunidad que me han dado de hacer este trabajo.

A mis compañeros de IK4-Ikerlan. En particular, a Rosa, por la ayuda que me ha dado durante este tiempo, sobre todo en los temas en los que peor me desenvuelvo; y a Aitor, director y compañero, por el apoyo que me ha dado.

A todo el grupo de CTR de la Universidad de Cantabria, por los conocimientos que me han transmitido y toda la ayuda que me han dado.

Por último, a mi familia, por ayudarme a llegar hasta aquí.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En la última década el uso de servicios web se ha extendido ampliamente, tanto en el entorno empresarial, como en el entorno del hogar, llegando a abarcar un gran número de campos de aplicación. La dificultad de portar esta tecnología a ciertos campos, como la automatización industrial, aeronáutica, control aéreo, domótica e industria energética, debido a la alta necesidad de determinismo en sus procesos, ha llevado a realizar muchos trabajos de investigación en torno a ello. La mayor parte de estos trabajos se han centrado en extender la arquitectura orientada a servicios (en inglés, Service Oriented Architecture – SOA) para dotarle de mayor determinismo, y de esta forma definir una arquitectura Real Time SOA [HEL05, TSA06, PAN09, MOU10]. También se han realizado trabajos para dotar de determinismo a las capas inferiores de red, en este caso, al protocolo de Ethernet [PED05, FEL05, HOA02].

A pesar de estos trabajos de investigación, debido a las dificultades que supone su aplicación en entornos empresariales, esta línea de trabajo no ha terminado de afincarse masivamente de forma práctica en el entorno empresarial. Se ha optado más por otra línea, en la que los servicios ofrecen una calidad de servicio asociada (en inglés, Quality-of-Service - QoS) y el cliente utiliza el servicio en función de dicha calidad. Con esta especie de contrato entre cliente y servicio se consigue un tiempo real laxo (Soft Real Time), adecuado para muchos campos de aplicación.

Conocer los tiempos de respuesta de los middleware de servicios web es muy importante para garantizar la QoS requerida por cada aplicación, sin embargo las comunicaciones, las redes de acceso o el procesamiento utilizado hacen que estos tiempos de respuesta sean poco deterministas.

A continuación, se describe el estado del arte sobre diversos aspectos de QoS (esto es, trabajos de investigación, estándares, herramientas). En la Sección 1.2, se describe el middleware conforme a la norma IEC61850 que se utilizará como caso de

estudio y en la Sección 1.3 se detallan características de los servicios web RESTful y SOAP. En esta sección, finalmente se exponen los objetivos de este trabajo.

1.1. Antecedentes

Este trabajo, centrado en la medición de QoS, no hubiese sido posible sin el trabajo realizado anteriormente en este campo. Por ello a continuación se describe brevemente el conocimiento desarrollado en los últimos años.

1.1.1. Medición de QoS en servicios web

La QoS está representada por un amplio número de características que permiten diferenciar entre dos servicios con la misma funcionalidad [MEN02]. Estas características están divididas en diferentes aspectos: de ejecución, de configuración, de seguridad, etc.; recogidos en el trabajo de Ran [RAN03].

Aunque existen muchos aspectos relativos a la QoS, la mayoría de los trabajos se centran en lo relacionado con el rendimiento del servicio [STR01, GOV04, MAC05, THI05]. A pesar de que todos estos trabajos se centran en aspectos de rendimiento y concretamente en el tiempo de respuesta del servidor, cada uno trabaja unos parámetros concretos. Por ejemplo, Strohmeier analiza diferentes aspectos: tiempo de envío (One-way Delay), variación en el tiempo de respuesta (IP Packet Delay Variation), número de paquetes perdidos (One-way Packet Loss), patrón de paquetes perdidos (One-way Packet Loss Patterns) y throughput [STR01]. Mientras que los otros trabajos se analiza únicamente el tiempo de respuesta (End To End Performance o RoundTrip Delay) [GOV04, MAC05, THI05]. Por otra parte, en el trabajo de Machado [MAC05] se desgrana el cálculo del tiempo de respuesta para determinar qué cantidad de ese tiempo se debe a las fases de serialización/deserialización del mensaje, envío/recepción del mensaje, etc.

En este trabajo se realizarán mediciones del tiempo de respuesta de peticiones del cliente al servidor desde que éstas se generan en el cliente hasta que la respuesta es tratada de nuevo en el cliente (RoundTrip Delay) (Figura 1). Además, similar al trabajo de [MAC05], se han realizado sub-medidas, en los casos que se ha considerado adecuado, para poder determinar en qué partes del código se encuentran las ineficiencias del middleware.

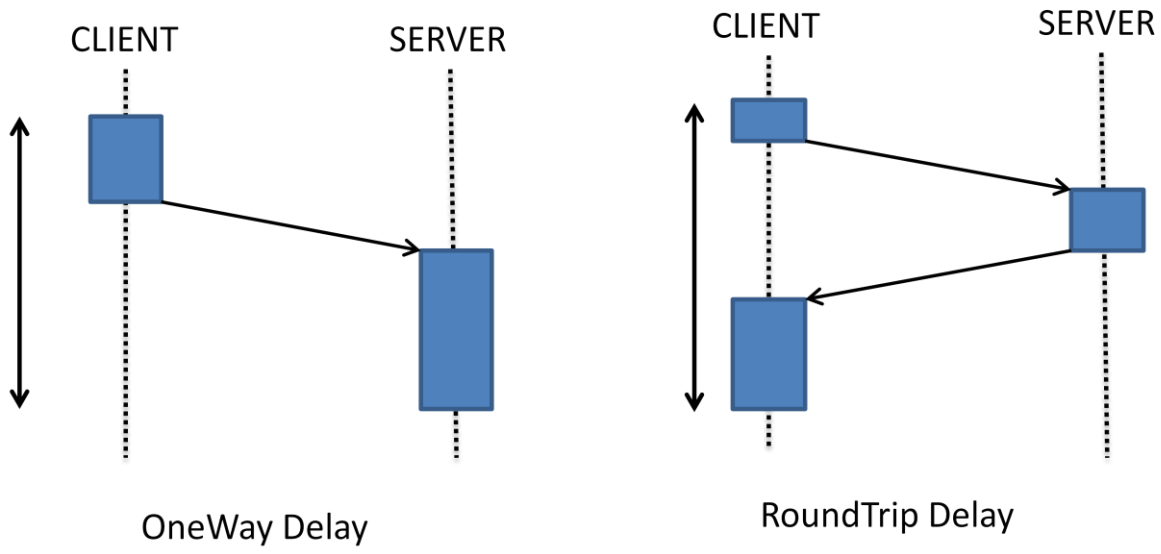


Figura 1. OneWay Delay y Round-Trip Delay

1.1.2. Instrumentación y medición de tiempos de respuesta

Para la medición de tiempos de respuesta mencionados en el punto anterior es necesario realizar una instrumentación del código de la aplicación que se desea analizar. Esta instrumentación dependerá del tipo de medida que se quiera realizar. Cabe destacar dos tipos de medición del tiempo de respuesta: One Way Delay y Round-Trip Delay (Figura 1). La diferencia entre las dos consiste en donde se recogen las dos medidas que se toman para el cálculo del tiempo de respuesta. En el caso del Round-Trip Delay, ambas medidas se recogen en el cliente, mientras que en el One Way Delay una medida se recoge en el cliente y la otra en el servidor. Debido a esto, en el cálculo del Round-Trip Delay no es necesaria ninguna sincronización entre los relojes del cliente y del servidor [PAK08]. Mientras que para calcular el One Way Delay es necesario que los relojes del cliente y servidor estén sincronizados [VIT08].

En lo que respecta a la sincronización de relojes se han realizado muchos trabajos en los últimos años. Existen diferentes métodos de sincronización: mediante hardware externo GPS (Global Positioning System) [VIT08] y mediante algoritmos de sincronización como NTP (Network Time Protocol) [VIT08], el estándar IEEE 1588 [VIT08] y otros algoritmos [PAS02, HAN06]. En este trabajo, no se harán uso de algoritmos de sincronización dado que se medirán round-trip delays.

1.1.3. Estándar Application Response Measurement (ARM)

ARM (Application Response Measurement) es un estándar abierto desarrollado por el Open Group [ARM12]. ARM describe un método común para la integración de aplicaciones empresariales como entidades gestionables. Para ello, permite a los usuarios medir la disponibilidad, rendimiento y uso de las aplicaciones; además de, tiempo de respuesta de las transacciones extremo a extremo.

El estándar define una API (Application Programming Interface) en C y Java para la medición de estos parámetros.

1.1.4. Herramientas de instrumentación y medición de tiempos de respuesta

Existen numerosas herramientas para instrumentar código y medir el tiempo de respuesta de un servicio. En este apartado se describirán las herramientas analizadas en este proyecto.

1.1.4.1. TAU

TAU (Tuning and Analysis Utilities) es un conjunto de herramientas desarrolladas por la Universidad de Oregon, el centro de investigación Juelich y el laboratorio nacional de Los Alamos [TAU12]. Este conjunto de herramientas estáticas y dinámicas proporcionan interacción gráfica con el usuario para formar un entorno integrado para el análisis de aplicaciones paralelas en Fortran, C++, C, Java y Python [SHE06].

TAU permite instrumentar el código mediante dos métodos: automáticamente o manualmente. El primer método permite calcular los tiempos de respuesta de todas las funciones del código de la aplicación, mientras que el segundo método, mediante una API proporcionada, permite centrarse únicamente en las medidas de interés del usuario.

1.1.4.2. DelayTool

DelayTool es una herramienta desarrollada por el centro de investigación VTT [PAA11]. Esta herramienta ofrece una API para instrumentar el código en varios lenguajes: C, C++, Java y Javascript.

La mayor aportación de esta herramienta es que complementa la instrumentación de código con la sincronización de relojes mediante GPS. Con ello, permite la medición de tiempos en sistemas distribuidos con una alta precisión.

1.1.4.3. MyARM

MyARM es una implementación multiplataforma (Linux, Windows, Solaris) y multilenguaje (C, Java, C++, C#, Python) del estándar ARM [MYA12]. MyARM, además de ofrecer una API para instrumentar el código de acuerdo al estándar ARM, ofrece herramientas para recoger las mediciones realizadas y herramientas gráficas para el manejo y estudio de la información recogida. Estas herramientas gráficas permiten generar gráficos y estadísticas de las medidas registradas.

1.2. Middleware de comunicación conforme a la norma IEC61850

La norma IEC 61850 es parte del International Electrotechnical Commission's (IEC) Technical Committee 57 (TC57) architecture for electric power systems [IEC12, MAC06]. Esta norma define un estándar de comunicación entre equipos de protección, control y medida dentro de una subestación eléctrica. Además, esta norma se ha extendido a diversos campos, como el eólico (IEC 61400-25), los recursos de energía distribuida o DER (Distributed Energy Resources) (61850-7-420), y su extensión a otros campos de sistemas de control y monitorización es clara. En el caso concreto de IK4-Ikerlan, se ha aplicado al dominio del transporte ferroviario, transporte vertical (ascensores), DER y domótica.

Aunque este grupo de normas ofrece "mappings" a diferentes protocolos de comunicación, en IK4-Ikerlan se ha optado por "mapear" la norma a dos protocolos diferentes: SOAP (Service Oriented Application Protocol) Web Services y RESTful (Representational State Transfer) Web Services. De esta forma, se han desarrollado dos middlewares con las mismas funcionalidades pero con diferentes protocolos de comunicación. Ambas implementaciones han sido desarrolladas en el lenguaje C++.

1.2.1. Modelo de datos

El modelo de datos definido por la norma IEC61850 para modelar subestaciones eléctricas está compuesto de diversos elementos (Figura 2):

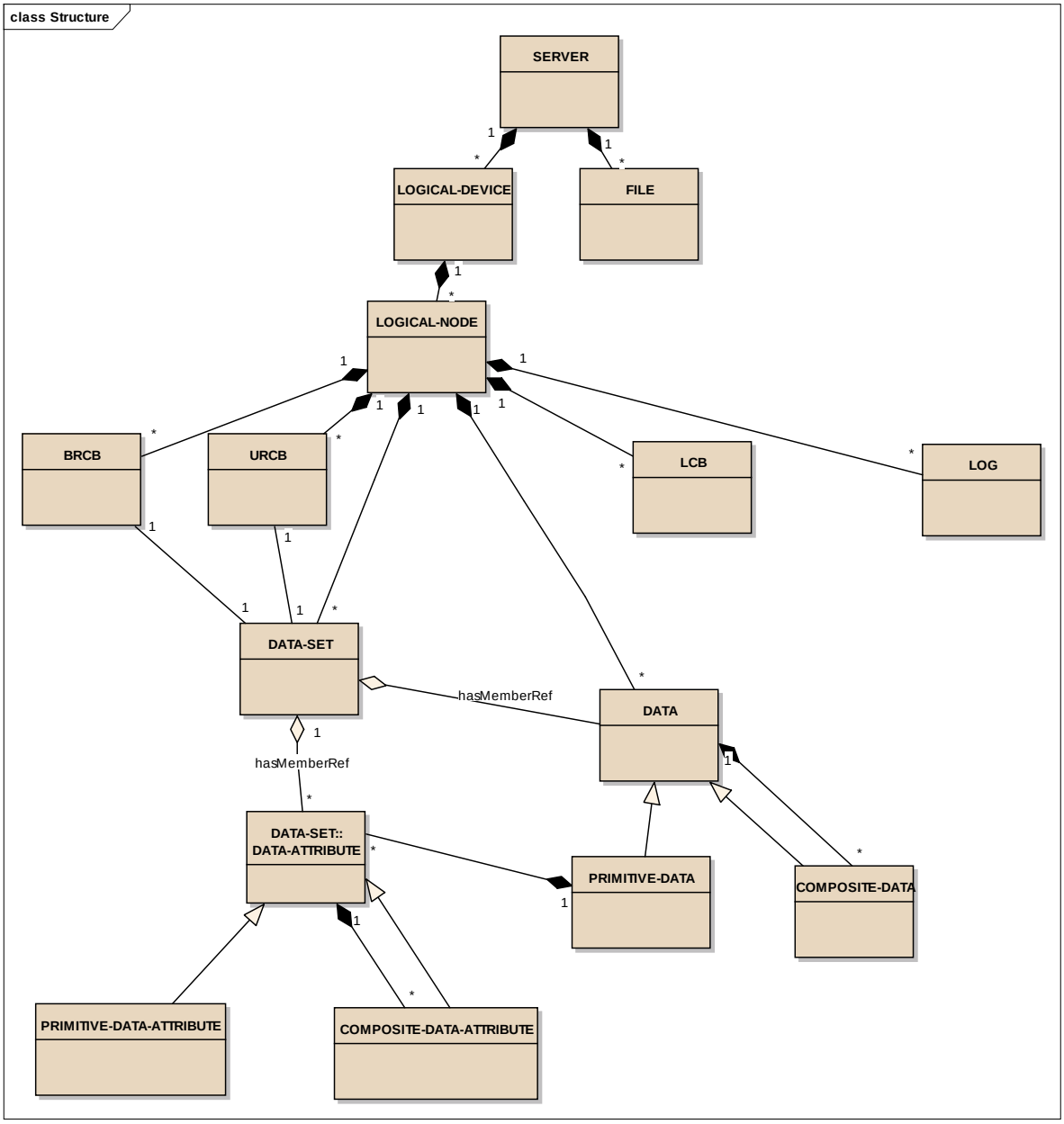


Figura 2. Modelo de datos de la norma IEC61850

- SERVER: La clase SERVER representa el comportamiento visible externamente de un dispositivo. Puede estar compuesto por LOGICAL-DEVICEs y FILEs.

- LOGICAL-DEVICE: Entidad que representa un conjunto de funciones típicas de una subestación. Se trata de una composición de LOGICAL-NODEs. Un LOGICAL-DEVICE puede ser utilizado simplemente como un conjunto de LOGICAL-NODES o como un dispositivo que funciona como un gateway o proxy.

- FILE: El SERVER puede contener FILEs. Los servicios de trasferencia de FILEs ASCII proporcionarán funcionalidad para la transferencia de FILES desde y hacia colecciones de FILES y su administración.

- LOGICAL-NODE: Entidad que representa una función típica de una subestación. Se trata de una composición de DATA, DATA-SET, BRCB, URCB, LCB y LOG.

- BRCB: Los Buffered-Report-Control-Block son eventos internos (causados por cambios en datos, cambios de calidades y actualización de datos) que emiten inmediatamente el envío de reportes o encolan los eventos (hasta cierto límite) para su transmisión, de manera que los valores de DATA no se pierdan debido a las restricciones de control del flujo de transporte o por la pérdida de conexión. BRCB proporciona la funcionalidad de la secuencia de los eventos (SOE). El Report-Control-Block controlará los procedimientos que se requieren para informar de los valores DATA de uno o más LOGICAL-NODES a un cliente. Las instancias del Report Control se configurarán en el servidor en tiempo de configuración. El servidor puede restringir el acceso a una instancia del Report Control a un cliente. El BRCB contiene un DATA-SET.

- URCB: Los Unbuffered-Report-Control-Block son eventos internos (causados por cambios en datos, cambios de calidades y actualización de datos) que emiten inmediatamente el envío de reportes en base a "best-effort". Si no existe una asociación o si el flujo de transporte de datos no es lo suficientemente rápido para soportarlo los eventos se pueden perder. El Report-Control-Block controlará los procedimientos que se requieren para informar de los valores DATA de uno o más LOGICAL-NODES a un cliente. Las instancias del Report Control se configurarán en el servidor en tiempo de configuración. El servidor puede restringir el acceso a una instancia del Report Control a un cliente. El URCB contiene un DATA-SET.

- LCB: El LCB controlará los procedimientos que se requieren para el almacenamiento de los valores de DATA-ATTRIBUTE en un LOG. Cada LCB permitido asociará un DATA-SET con un LOG. Los cambios en el valor de un miembro de un DATA-SET se almacenarán como entrada en el LOG. Múltiples LCBs permiten múltiples DATA-SETs para alimentar a un LOG.

- LOG: Desde el punto de vista de implementación, el LOG puede estar considerado como un buffer circular que sobrescribe los valores más viejos

en el LOG. Sin embargo, esto está oculto para el cliente. Desde el punto de vista del cliente el LOG es un buffer lineal donde las entradas en el LOG son identificadas por: un identificador único de una entrada del LOG (EntryID) y el tiempo en el que se ha añadido la entrada en el LOG (TimeOfEntry).

- DATA-SET: Un DATA-SET es un grupo ordenado de ObjectReferences de DATAs o DATA-ATTRIBUTEs, organizados como una colección simple para satisfacción del cliente. Como la composición y orden de los ObjectReferences en un DATA-SET es conocido por el cliente y por el servidor, sólo se necesita transmitir el nombre del DATA-SET y el valor de los DATA y DATA-ATTRIBUTES referenciados. Esta capacidad permite un uso más eficiente en las comunicaciones.

- DATA-ATTRIBUTE: Atributos que contienen información de la clase que los contiene. La clase DATA-ATTRIBUTE es una clase abstracta auxiliar para la composición de las clases COMPOSITE-DATA-ATTRIBUTE y PRIMITIVE-DATA-ATTRIBUTE. Un DATA-SET está compuesto por DATA-ATTRIBUTEs.

- PRIMITIVE-DATA-ATTRIBUTE: Atributo de un PRIMITIVE-DATA.

- COMPOSITE-DATA-ATTRIBUTE: Atributo de un COMPOSITE-DATA.

- DATA: Las clases DATA representan información significativa de aplicaciones ubicadas en dispositivos de automatización. Los valores de las instancias de DATA pueden, por ejemplo, ser asignados y leídos. Cualquier conjunto de DATA puede formar un DATA-SET. La clase DATA es una clase abstracta auxiliar para la construcción de las clases simples y compuestas de datos comunes: PRIMITIVE-DATA y COMPOSITE-DATA.

- PRIMITIVE-DATA: La clase PRIMITIVE-DATA está compuesta por DATA-ATTRIBUTEs

- COMPOSITE-DATA: La clase COMPOSITE-DATA está compuesta por una clase o más del tipo SIMPLE-DATA y/o DATA-ATTRIBUTEs.

1.2.2. Funcionalidades del middleware

El middleware cuenta con un número alto de funcionalidades que se dividen en 4 tipos: consulta de variables de estado, control de variables, envío de informes y gestión de ficheros (Tabla 1).

Tabla 1. Funcionalidades del middleware

Nombre de la funcionalidad	Descripción
RF-App Assoc. 01	Conexión desde el dispositivo al Servidor IEC61850
RF-App Assoc. 02	Desconexión del dispositivo del Servidor IEC61850
RF-DataSet 01	Consulta de valores de un Data Set
RF-DataSet 02	Consulta de los elementos existentes en un Data Set
RF-Control 01	Envío de consignas
RF-Reporting 01	Notificación fiable de eventos
RF-Reporting 02	Notificación no fiable de eventos
RF-File 01	Transferencia de ficheros a Servidor IEC61850
RF-File 02	Descarga de ficheros al dispositivo
RF-File 03	Borrado de ficheros del Servidor IEC61850
RF-File 04	Consulta parámetros de fichero
RF-Server 01	Consulta de elementos existentes en el Servidor IEC61850
RF-Log. Dev. 01	Consulta de Logical Nodes de un Logical Device
RF-Log. Node 01	Consulta de elementos existentes en un Logical Node
RF-Log. Node 02	Consulta de los valores de los atributos de un Logical Node
RF-URCB 01	Consulta de valores de un URCB en concreto
RF-URCB 02	Modificación de los valores de un URCB en concreto
RF-BRCB 01	Consulta de valores de un BRCB en concreto
RF-BRCB 02	Modificación de los valores de un BRCB en concreto
RF-Data 01	Consulta de los valores de un Data

RF-Data 02	Consulta de nombres de elementos existentes en un Data
RF-Data 03	Consulta de la definición de un Data

Consulta de variables de estado

Las funcionalidades de consulta de variables de estado permiten la consulta de los estados (valores) de cualquiera de las variables definidas en el modelo de datos.

Las consultas pueden ser globales o bien específicas, es decir, es posible consultar el valor de todas las variables de un elemento o bien consultar una propiedad específica del mismo.

Además de la consulta de datos es posible consultar la estructura de los elementos, pudiendo de esta manera establecer la jerarquía del modelo de datos de manera dinámica.

Control de variables

Permite el control de las variables de los elementos de la red. Es decir, permite el envío de consignas para la actualización o modificación de las variables de los elementos de la red.

Sólo podrán ser modificadas aquellas variables que han sido definidas para tal caso.

Envío de informes

El sistema puede enviar informes periódicos, bajo demanda o asociados a cambios en las variables. Existen dos tipos de informes:

- **BRCB:** Ofrece notificación fiable de informes. Estos informes son enviados al suscriptor. En caso de que no se reciba ACK el informe será almacenado para enviarlo posteriormente. De esta manera, se asegura que el informe llegue al destino.

- **URCB:** Ofrece notificación no fiable de informes. Estos informes son enviados al suscriptor. Tanto se reciba o no ACK el informe es descartado tras ser enviado.

Gestión de ficheros

Mecanismos para la gestión de ficheros, tanto binarios como ASCII. Este grupo de funcionalidades permiten tanto el envió de fichero como la descarga y actualización de ficheros.

1.2.3. Implementación en diferentes campos de aplicación

Como se ha mencionado antes esta norma se ha extendido a muchos campos. Por ejemplo, en la Figura 3 se puede observar un ejemplo de despliegue del middleware de distribución de un sistema de trenes. En este despliegue se pueden observar varios elementos, entre los que tenemos el PCC (Puesto de Control Central), los trenes y varios elementos de la infraestructura de la vía, como son, las subestaciones de carga, las señales de paso o los semáforos. El PCC puede consultar datos de los elementos de la vía como de los trenes. Además, el PCC puede controlar y modificar las variables de los trenes definidas para ello. Por último, tanto el tren, como los elementos de la vía, generan eventos que son recogidos por el PCC. Por lo tanto los elementos del modelo se pueden clasificar en los siguientes tipos de elementos:

- Control y consulta de datos (PCC)
- Servidores de datos (Tranvía, subestación de carga, semáforo)
- Generación de eventos (Tranvía, subestación de carga, señal de paso)
- Consumidor de eventos (PCC)

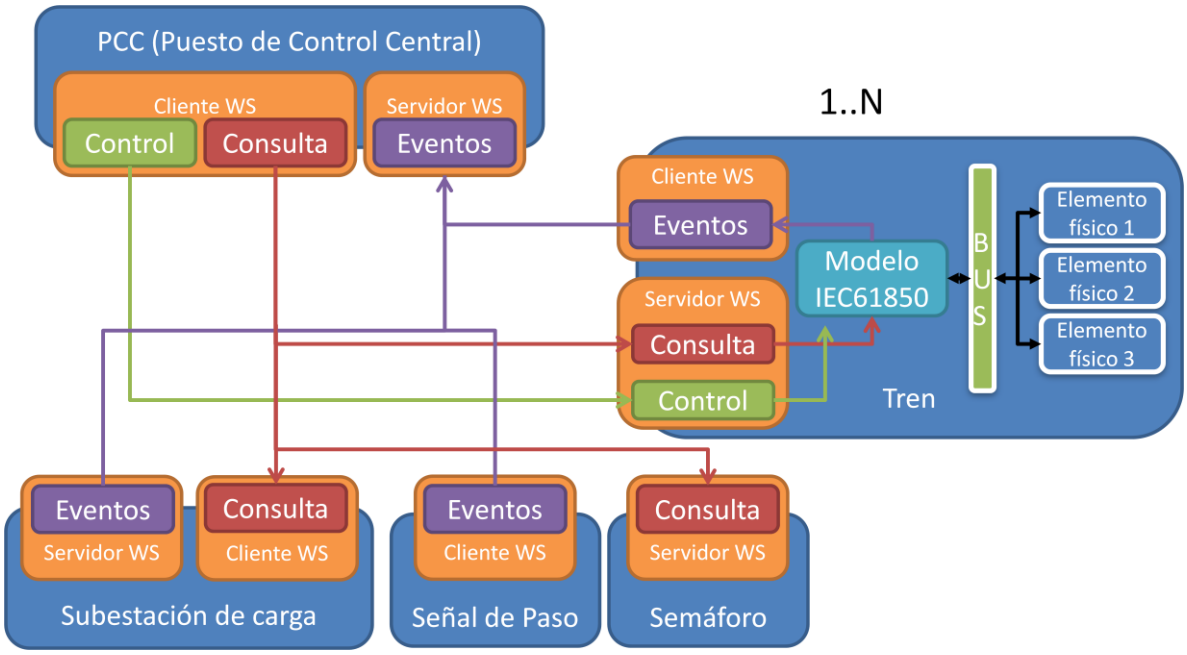


Figura 3. Despliegue del middleware de comunicaciones en un sistema de trenes

En la Figura 4 también se puede observar la aplicación de este middleware a otro campo de aplicación, en este caso, un sistema de ascensores. El modelo de este sistema es similar al del sistema de tranvías, contando con el mismo tipo de elementos.

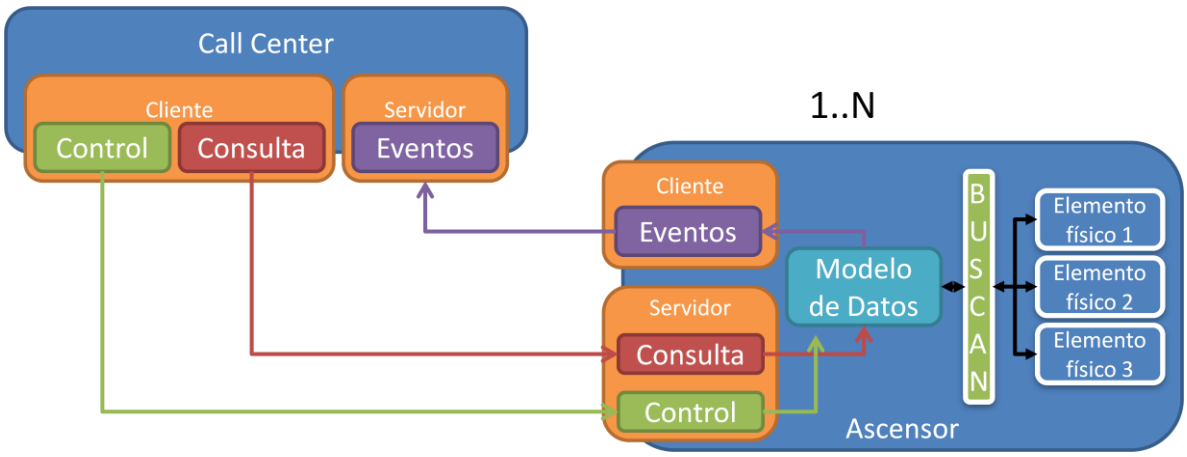


Figura 4. Despliegue del middleware de comunicaciones en ascensores

1.3. RESTful Web Services vs. SOAP Web Services

Como se ha mencionado anteriormente el middleware ha sido implementado en dos versiones diferentes con dos protocolos de comunicaciones diferentes: SOAP Web Services y RESTful Web Services.

La implementación SOAP Web Service del middleware utiliza la librería gSOAP (versión 2.8.3) [GSO12] para el desarrollo de los servicios. En la implementación REST del middleware se utiliza la librería libmicrohttpd (versión 0.9.19) [LMH12] como servidor http, la librería libcurl (versión 7.26.0) [LCU12] como cliente http y la librería JSONcpp (versión 0.5.0) [JSO12] como serializador JSON (JavaScript Object Notation).

Los servicios web proporcionan un medio estándar de interoperabilidad entre diferentes aplicaciones de software, que se ejecuta en una variedad de plataformas y/o frameworks. La arquitectura de servicios Web es una arquitectura de interoperabilidad: se identifican aquellos elementos globales de la red mundial de servicios de red que se requieren con el fin de garantizar la interoperabilidad entre los servicios Web. La organización W3C definió la arquitectura SOAP Web Service estableciendo una serie de estándares mediante los que se permite la interoperabilidad de las aplicaciones: SOAP, XML (Extensible Markup Language), WSDL (Web Service Description Language), UDDI (Universal Description, Discovery and Integration), etc.

El termino REST se acuño en el año 2000, cuando Fielding en su tesis lo definió como una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web [FIE00]. Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz web simple que utiliza XML y HTTP (Hypertext Transfer Protocol), sin las abstracciones adicionales del protocolo de servicios web SOAP.

Las reglas que establece REST son las siguientes:

- Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes.

- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE.

- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.

- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML (Hypertext Markup Language) o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

Una razón importante en la aplicación de REST a los servicios web se debe a la mejora en rendimiento que ofrece sobre los servicios web SOAP. Esto se fundamenta en la reducción en el tamaño de los mensajes respecto a los que se producen en el protocolo SOAP. En los últimos años, muchos estudios han realizado comparativas estudiando el rendimiento en ambos protocolos que aseveran esta diferencia de rendimiento[HAM10, KUM11, PAU08].

1.4. Objetivos

El objetivo general del presente proyecto consiste en analizar la calidad de servicio que puede ofrecer el middleware anteriormente mencionado sobre diferentes redes. Aunque la QoS incluye un gran número de aspectos, en este trabajo solo se contempla analizar el tiempo de respuesta del servicio (round-trip delay). Para llevar a cabo este objetivo principal, será necesario completar los siguientes objetivos específicos:

- Identificar transacciones significativas dentro del middleware.

- Instrumentar código fuente del middleware para la medición de tiempos de respuesta. Serán instrumentados los dos mapeos del middleware, SOAP Web Services y RESTful Web Services.

- Establecer entornos de pruebas sobre diferentes redes (en un único PC, red local, red WAN (Wireless Area Network) con acceso ADSL (Asymmetric Digital Subscriber Line) y red WAN con acceso 3G) y realizar las mediciones de tiempo de respuesta sobre estos entornos de pruebas.

- Realizar un estudio comparativo entre los dos mapeos del middleware.

- Establecer la QoS, concretamente el tiempo de respuesta o round-trip delay, que el middleware puede ofrecer sobre las diferentes redes.

2. Estudio de la herramienta de cálculo de tiempos de respuesta MyARM

Para realizar la instrumentación se ha decidido usar el software MyARM por ser el que mejor se adaptaba a las necesidades del proyecto, ya que ofrecía una API simple, además de herramientas gráficas para el análisis de los tiempos de respuesta registrados. MyARM es una herramienta para el cálculo de tiempos de respuesta compatible con el estándar ARM 4.0 de OpenGroup. MyARM ofrece APIs para diferentes lenguajes de programación: C, Java, C#, C++ y Phyton.

El vendedor ofrece diferentes tipos de licencia en función de las características de la herramienta que se quieran utilizar. En este caso, se ha utilizado la versión gratuita "Community edition", que es la edición con funcionalidades más reducidas, pero satisface las necesidades de este proyecto. La versión utilizada para el desarrollo del proyecto ha sido la 1.4, versión vigente al inicio de este proyecto. A fecha de escritura de este documento, la versión utilizada no se encuentra disponible en la web del vendedor, habiendo sido sustituida por una nueva versión 2.0 [MYA12]. No se observa ninguna incompatibilidad entre ambas versiones.

2.1. API de instrumentación

MyARM ofrece una API compatible con el estándar ARM 4.0 para la medición de tiempos de respuesta en transacciones. Aunque esta API ofrece diferentes funcionalidades las más importantes son las siguientes:

- Registrar, iniciar y parar una aplicación: Esta funcionalidad permite crear un registro en el que se van a englobar todas las transacciones de la aplicación que se está midiendo.
  - arm_register_application
  - arm_start_application
  - arm_stop_application
- Registrar, iniciar y parar una transacción: Mediante esta utilidad se crean las transacciones y se recogen tiempos de inicio y fin de transacción.
  - arm_register_transaction
  - arm_start_transaction
  - arm_stop_transaction

Los tiempos registrados mediante esta API son registrados en una base de datos (En la "Community Edition" solo se puede usar una base de datos SQLite3).

2.2. Herramientas de análisis

MyARM ofrece 3 tipos de herramientas para analizar los resultados obtenidos mediante la instrumentación:

- Herramientas de comando para almacenar y gestionar los datos. Estas herramientas solo están disponibles en las versiones de pago.
- Herramienta web para análisis de datos de forma remota y con múltiples accesos. Esta herramienta solo están disponibles en las versiones de pago.
- Herramienta myarmmanager: Interfaz gráfica de usuario que permite el análisis y gestión de los datos recogidos.

La herramienta myarmmanager ofrece diferentes tipos de gráficas y cálculos estadísticos realizados sobre los datos recogidos. En la Figura 5 se puede observar un análisis mediante la aplicación MyARM de una aplicación con diferentes transacciones. Tres de estas transacciones tienen los tiempos de respuesta de las mediciones representados en la grafica mediante puntos de diferentes colores (Azul, rojo y verde).

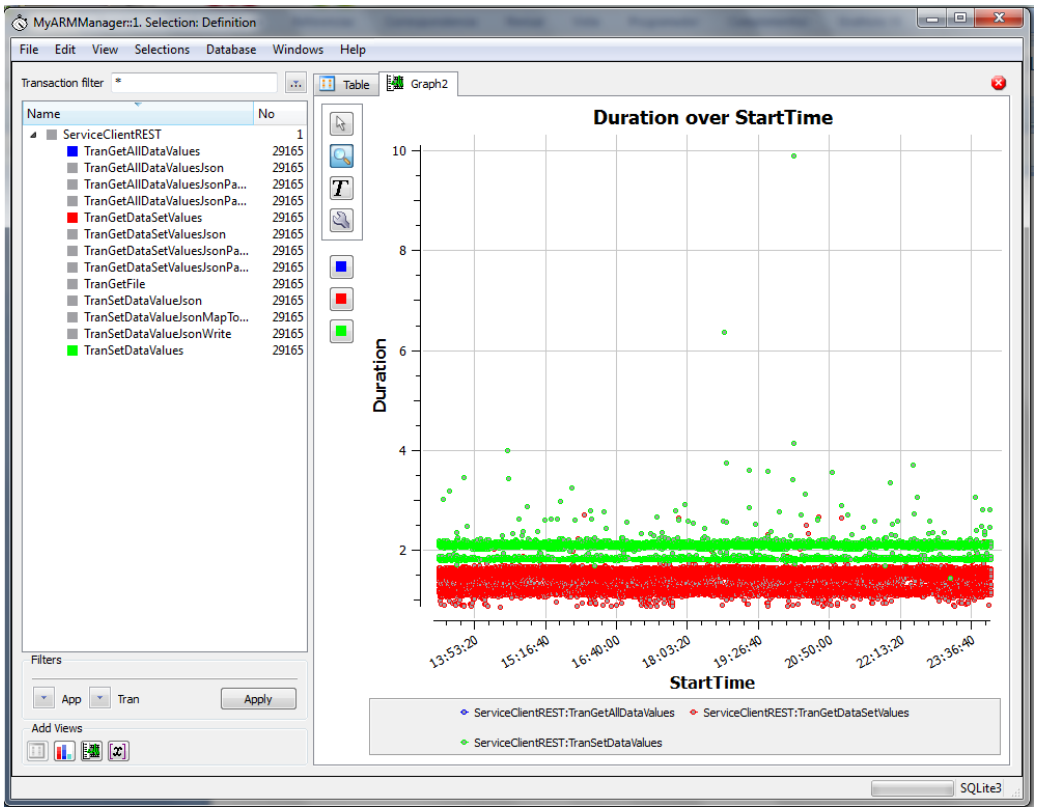
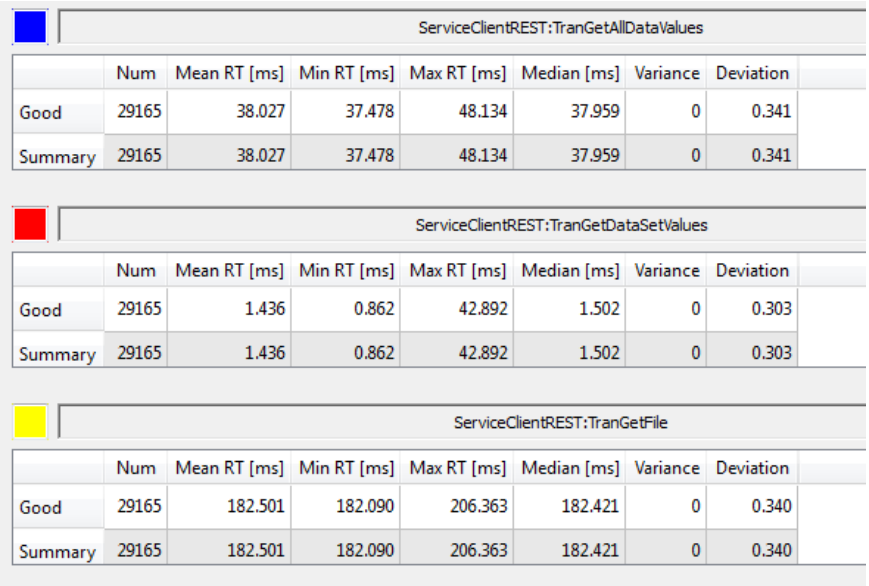


Figura 5. Gráfica generada con myarmmanager

En la Figura 6 se pueden observar los datos estadísticos de tres transacciones diferentes generados mediante la herramienta. Podemos observar datos como tiempo de respuesta medio, mínimo, máximo, mediana, desviación y varianza.



ServiceClientREST:TranGetAllDataValues

	Num	Mean RT [ms]	Min RT [ms]	Max RT [ms]	Median [ms]	Variance	Deviation
Good	29165	38.027	37.478	48.134	37.959	0	0.341
Summary	29165	38.027	37.478	48.134	37.959	0	0.341

ServiceClientREST:TranGetDataSetValues

	Num	Mean RT [ms]	Min RT [ms]	Max RT [ms]	Median [ms]	Variance	Deviation
Good	29165	1.436	0.862	42.892	1.502	0	0.303
Summary	29165	1.436	0.862	42.892	1.502	0	0.303

ServiceClientREST:TranGetFile

	Num	Mean RT [ms]	Min RT [ms]	Max RT [ms]	Median [ms]	Variance	Deviation
Good	29165	182.501	182.090	206.363	182.421	0	0.340
Summary	29165	182.501	182.090	206.363	182.421	0	0.340

Figura 6. Estadísticas generadas con myarmmanager

3. Escenarios de prueba

En este apartado se van a describir los escenarios de prueba asociados al middleware conforme a la norma IEC61850 que se han cubierto en este proyecto y el entorno de pruebas sobre el que se han realizado las mediciones.

3.1. Selección de transacciones a medir

Para seleccionar las transacciones en las que se iban a tomar medidas se partió de las funcionalidades que ofrecía el middleware (Tabla 1). Dado que el número de funcionalidades era muy amplio se decidió seleccionar un grupo significativo de estas que representasen a todas las demás. Mediante esta selección se ha buscado recoger todos los tipos de funcionalidad, así como, funcionalidades que intercambiasen mensajes de diferentes tamaños entre cliente y servidor. Las transacciones seleccionadas son las siguientes:

- GetDataSetValues (RF-DataSet 01): Se trata de una petición por parte del PCC al tren de todos los valores de un DataSet. Se trata de un tamaño de mensaje pequeño. En el DataSet seleccionado para las pruebas el tamaño aproximado de los mensajes es de 400 bytes en REST y 2200 bytes en SOAP.

- GetAllDataValues (RF-Log. Node 02): Se trata de una petición por parte del PCC al tren de todos los valores de un Logical Node. Se trata de un tamaño de mensaje grande. En el Logical Node seleccionado para las pruebas el tamaño aproximado de los mensajes es de 67 KB en REST y 255 KB en SOAP.

- GetFile (RF-File 02): Se trata de una petición de un fichero por parte del PCC al tren. El fichero seleccionado para hacer las mediciones tiene un tamaño aproximado de 2MB.

- SetData (RF-Control 01): Esta transacción consiste en el envío del valor de un Data desde el PCC para que este sea modificado en el tren. Se trata de un tamaño de mensaje pequeño. En el Data seleccionado para las pruebas el tamaño aproximado de los mensajes es de 400 bytes en REST y 2000 bytes en SOAP.

- SendReport (RF-Reporting 01): Esta transacción consiste en enviar un informe del tren al PCC. Se trata de un tamaño de mensaje pequeño. En el informe seleccionado para las pruebas el tamaño aproximado de los mensajes es de 1 KB en REST y 3 KB en SOAP.

3.2. Entorno de pruebas

Para las pruebas se ha decidido usar un modelo del tren más limitado que el descrito en la Figura 3. En este caso, únicamente habrá dos elementos que interactuaran en la red: el PCC y el tren (Figura 7).

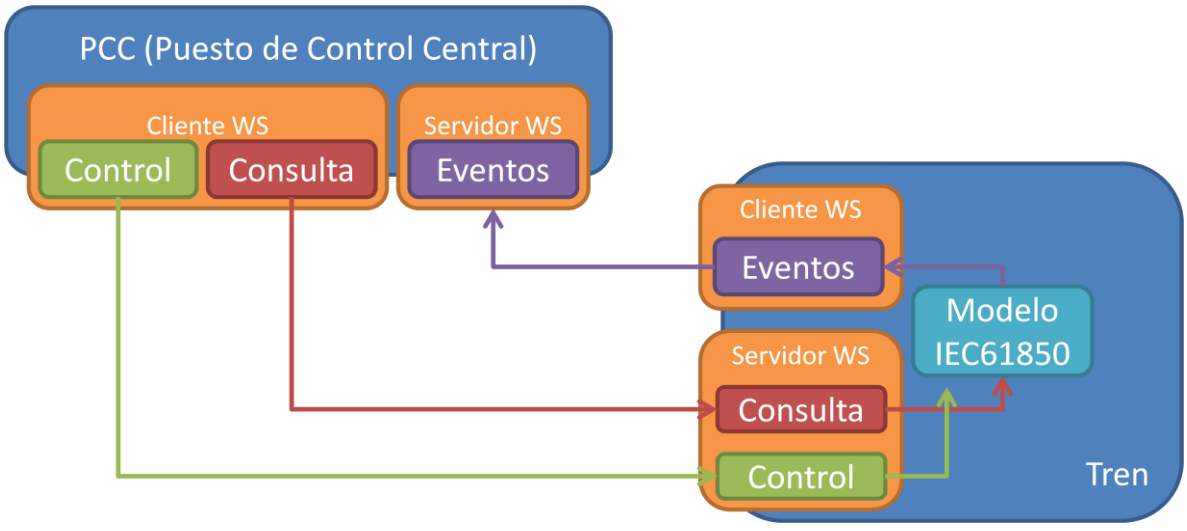


Figura 7. Despliegue del middleware de comunicaciones para el entorno de pruebas

Como se mencionaba anteriormente, uno de los objetivos del proyecto es analizar el middleware sobre diferentes redes. Por lo que se han definido 4 escenarios diferentes sobre los que hacer mediciones del tiempo de respuesta del middleware. La diferencia

principal entre estos 4 escenarios reside en la red mediante la que se conectan el PCC y el tren. Los 4 escenarios son los siguientes:

- Un PC: En este primer escenario, ambos elementos, PCC y tren se despliegan en el mismo PC. De esta forma, la red no tiene incidencia sobre los resultados, ya que no se hace uso de ella.

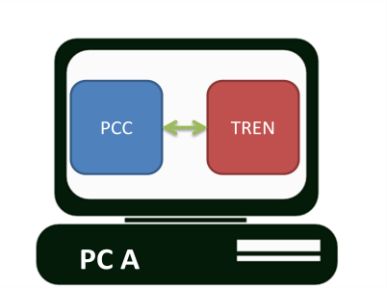


Figura 8. Despliegue sobre un único PC

- Dos PCs sobre red LAN: En este segundo escenario, el PCC y el tren se encuentran separados en dos PCs diferentes interconectados por una red local dedicada. La conexión se realiza mediante un switch de red Netgear Fast Ethernet Switch FS108 de 100 Mbps de velocidad.

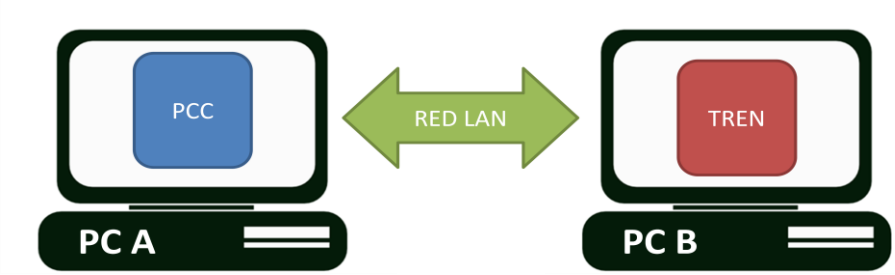


Figura 9. Despliegue sobre una red local

- Dos PCs sobre internet con acceso ADSL: En este caso, también se dispone de dos PCs, pero la interconexión se realiza a través de internet. La conexión a internet del PC A se realiza mediante el router Amper Xavi 7028r y la conexión del PC B se realiza mediante el router Comtrend CT-5361, ambos suministrados por Telefónica. La conexión de internet del PC A es una ADSL con 3Mb/s de velocidad de bajada y 300Kb/s de subida. La conexión del PC B es una ADSL con 2Mb/s de velocidad de bajada y 300Kb/s de subida.

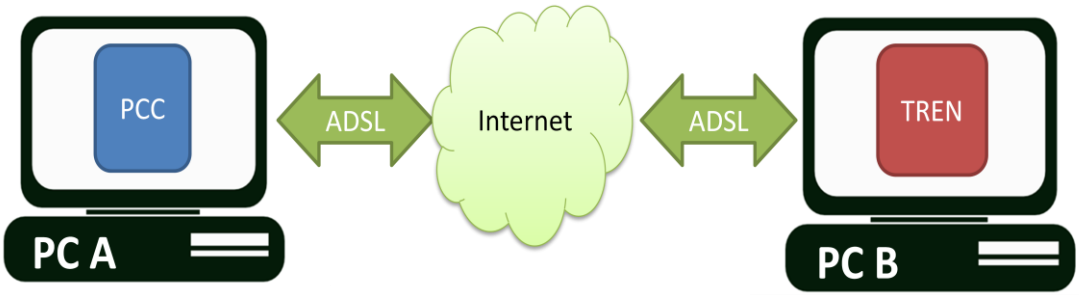


Figura 10. Despliegue sobre internet con acceso ADSL

- Dos PCs sobre internet con acceso inalámbrico 3G: En este último caso, la conexión entre los dos PCs también se realiza a través de internet. Pero, en este caso, la conexión de ambos PCs a internet se realiza mediante dispositivos USB de internet móvil Huawei E1752 suministrados por Movistar. La tarifa contratada en ambos casos es la Tarifa Internet Multidispositivo 25 que ofrece 2GB a máxima velocidad (7,2 Mbps de bajada y 1,4 Mbps de subida) y reduciendo la velocidad a partir de entonces (128 Kbps de bajada y 64 Kbps de subida).

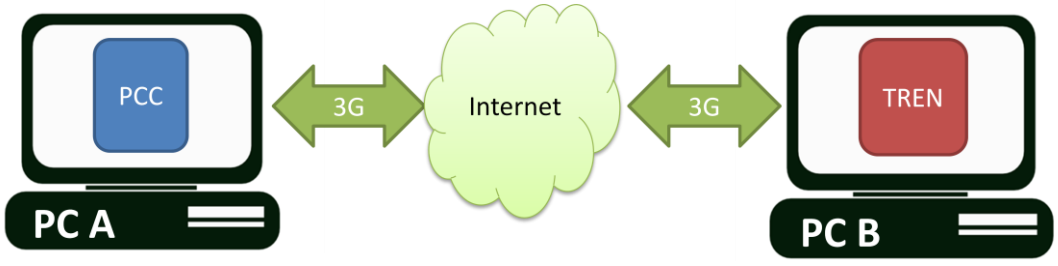


Figura 11. Despliegue sobre internet con acceso 3G

En todos los escenarios de prueba, tanto el PC A, como el PC B se tratan del mismo modelo de PC. El modelo concreto es un Dell Optiplex 755 con 2GB de memoria RAM y una CPU Intel Core 2 DuoE6550 2,33 GHz x2. La tarjeta de red es una Intel 82566DM-2 Gigabit Network Connection.

3.3. Proceso de medición

Para realizar un estudio comparativo entre las dos implementaciones del middleware era necesario realizar las mediciones en unas condiciones similares para ambas implementaciones. Esto no era sencillo, ya que el hardware disponible era limitado y no se podían realizar las pruebas en paralelo.

En el caso en el que, tanto el PCC, como el tren, se encuentran en el mismo PC y en el caso en el que se despliegan en una red local, reproducir el mismo escenario no resulta complicado. Pero, en los otros dos casos, en los que la aplicación se despliega sobre internet, reproducir el mismo escenario de carga de la red resulta imposible, ya que esto se encuentra bajo el control del operador telefónico. Siendo conscientes de que los resultados obtenidos en estos casos van a ser muy dependientes del estado de la red en ese instante, y para que los escenarios de las pruebas sean lo más similares posible, estas se han realizado durante las mismas horas del día y en días similares de la semana. Consiguiendo así un entorno lo más similar posible en todas las pruebas.

4. Resultados obtenidos

En este apartado se van a describir los resultados y conclusiones que se obtuvieron en las primeras pruebas realizadas. Por último se plantearán unas posibles soluciones a los problemas identificados en el primer análisis y se analizarán los resultados de dichas soluciones.

4.1. Primeros resultados

Una vez instrumentado el código y realizadas las primeras pruebas, se recogieron las mediciones de los tiempos de respuesta y se realizó un análisis estadístico. Las medidas se realizaron en todos los escenarios durante 16 horas, dando lugar a diferente número de datos recogidos para cada escenario, dependiendo de la velocidad de la red. Siendo aproximadamente 1000 mediciones en las redes más lentas y de 15000 aproximadamente en las redes más rápidas. Las estadísticas calculadas a partir de las mediciones hechas durante las pruebas se pueden observar en la Tabla 2, Tabla 3, Tabla 4 y Tabla 5. La Tabla 2 recoge los resultados obtenidos en la prueba realizada sobre el primer escenario, donde, tanto el PCC como el tren, se ejecutan en un solo PC. La Tabla 3 refleja los resultados obtenidos en el segundo escenario, en el que el PCC y el tren se encuentran en dos PCs interconectados mediante una red local dedicada. La Tabla 4 describe los tiempos de respuesta recogidos en el tercer escenario, en el que los dos PCs se encuentran conectados a internet mediante ADSL. Por último, la Tabla 5 recoge los datos registrados en el escenario en el que, igual que en el caso anterior, los PCs se conectan a través de internet, pero esta vez mediante una conexión 3G.

Tabla 2. Primeros resultados: Prueba realizada sobre un único PC

	Average Time (ms)	Median Time (ms)	Deviation (ms)	Max Response Time (ms)	Invoked msg size (bytes)	Received msg size(bytes)
RESTDataSet	0.480	0.482	0.043	2.950	187	211
SOAPDataSet	0.775	0.815	0.144	5.037	814	1416
RESTAllData	17.708	17.678	0.165	23.739	185	67587
SOAPAllData	14.895	14.818	0.395	20.329	847	255772
RESTGetFile	125.332	125.165	0.720	148.626	160	2196687
SOAPGetFile	99.228	98.717	1.454	109.781	777	2197169
RESTSetData	0.399	0.394	0.028	1.159	324	141
SOAPSetData	0.561	0.563	0.129	2.069	1200	1119
RESTReporting	39.009	39.031	1.162	44.638	1081	141
SOAPReporting	0.799	0.792	0.182	6.230	2767	626

Tabla 3. Primeros resultados: Prueba realizada sobre dos PCs en una red local dedicada

	Average Time (ms)	Median Time (ms)	Deviation (ms)	Max Response Time (ms)	Invoked msg size (bytes)	Received msg size(bytes)
RESTDataSet	1.175	1.185	0.113	3.454	187	211
SOAPDataSet	1.756	1.785	0.142	4.729	814	1416
RESTAllData	24.074	24.043	0.102	26.067	185	67587
SOAPAllData	31.205	31.132	0.429	37.032	847	255772
RESTGetFile	308.487	308.425	0.378	314.009	160	2196687
SOAPGetFile	243.491	243.412	0.530	257.199	777	2197169
RESTSetData	1.719	1.670	0.247	4.623	324	141
SOAPSetData	1.622	1.628	0.193	6.486	1200	1119
RESTReporting	42.310	40.831	6.176	77.811	1081	141
SOAPReporting	1.958	1.666	1.599	11.937	2767	626

Tabla 4. Primeros resultados: Prueba realizada sobre dos PCs en internet conectados mediante ADSL

	Average Time (ms)	Median Time (ms)	Deviation (ms)	Max Response Time (ms)	Invoked msg size (bytes)	Received msg size(bytes)
RESTDataSet	575.567	595.115	226.474	5453.328	187	211
SOAPDataSet	399.924	397.588	264.958	5337.168	814	1416
RESTAllData	1474.444	1455.686	342.240	11512.646	185	67587
SOAPAllData	4109.057	4096.141	306.236	14397.431	847	255772
RESTGetFile	37139.313	36725.153	6276.772	218991.314	160	2196687
SOAPGetFile	32918.491	32788.677	1288.132	69365.330	777	2197169
RESTSetData	550.060	548.882	6.212	589.606	324	141
SOAPSetData	336.974	334.537	7.530	376.358	1200	1119
RESTReporting	1332.807	621.630	2282.624	29492.785	1081	141
SOAPReporting	1922.351	2254.254	994.845	12705.776	2767	626

Tabla 5. Primeros resultados: Prueba realizada sobre dos PCs en internet conectados mediante 3G

	Average Time (ms)	Median Time (ms)	Deviation (ms)	Max Response Time (ms)	Invoked msg size (bytes)	Received msg size(bytes)
RESTDataSet	877.303	839.131	268.659	5598.984	187	211
SOAPDataSet	747.226	730.997	175.943	2918.483	814	1416
RESTAllData	1402.218	1352.666	266.745	5966.061	185	67587
SOAPAllData	2531.875	2486.270	244.778	5629.145	847	255772
RESTGetFile	15613.601	15561.211	234.744	17299.652	160	2196687
SOAPGetFile	15622.347	15504.324	490.811	19740.984	777	2197169
RESTSetData	841.025	819.485	86.279	1538.239	324	141
SOAPSetData	561.440	526.300	119.878	1490.201	1200	1119
RESTReporting	2244.469	2082.764	1029.087	9216.949	1081	141
SOAPReporting	1009.091	1019.468	271.840	5137.346	2767	626

4.2. Conclusiones sobre los primeros resultados

Repasando los resultados obtenidos en los dos primeros casos de uso (Tabla 2 y Tabla 3), se pueden observar algunos casos en los que la implementación SOAP es más rápida que la de REST (transacciones GetFile y Reporting, resaltadas en amarillo). Este

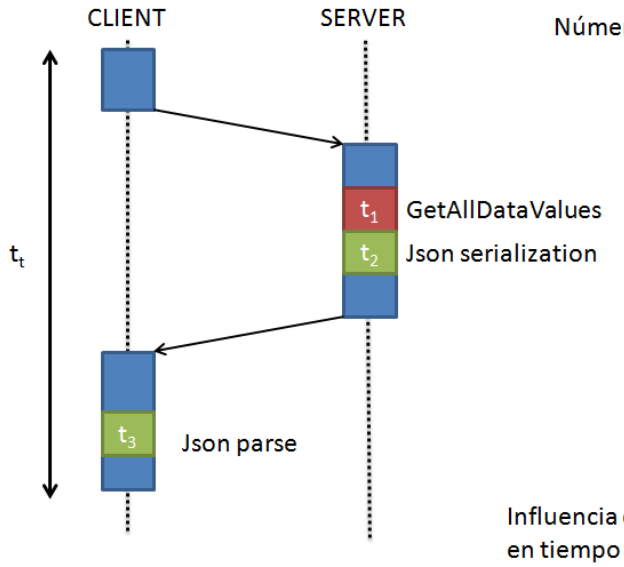
dato resulta sorprendente, ya que los estudios citados en el Apartado 1.3 afirman que REST es más eficiente, debido a su mayor simplicidad, en comparación con SOAP. Por otra parte, se puede observar que la transacción GetAllDataValues (resaltada en rojo), tiene un tiempo de respuesta superior en REST que en SOAP cuando las pruebas se realizan en un PC. En cambio, si las pruebas se realizan sobre una red local, el resultado es el contrario, la implementación de REST resulta algo más rápida que la de SOAP. Esto es debido a que el gran tamaño de la información transmitida hace que la diferencia en los mensajes de REST y SOAP sea grande y la implementación de REST saque provecho de ello (67 KB en REST y 255 KB en SOAP). También se puede observar las transacciones SetDataValue y GetDataValue (resaltado en verde), en las que los tiempos entre las implementaciones REST y SOAP son muy similares.

Debido a estos sorprendentes resultados, se decidió hacer un estudio más detallado de dichas transacciones. Para ello, además de medir el tiempo round-trip delay, se midieron también tiempos intermedios, tanto en el servidor, como en el cliente. Tras este análisis, se dedujeron las siguientes conclusiones:

- En la transacción Reporting, en la que los tiempos de respuesta en REST son hasta 40 veces más grandes que en SOAP, existe un problema con la librería libmicrohttpd. HTTP define un tipo de método de transferencia, llamado “chunked”, para enviar los mensajes HTTP. La librería libmicrohttp no responde eficientemente a este tipo de mensajes. Este tipo de mensajes solo se utilizaba en esta transacción.

- En la transacción GetFile la implementación REST es considerablemente más lenta que la de SOAP. Esto se debía a que en la implementación REST se hacia una serialización/deserialización del fichero totalmente innecesaria.

- En los demás casos, se observa una ineficiencia en el código de serialización/deserialización que utiliza la librería JSONcpp. Para comprobar esta ineficiencia se ha tomado una muestra de 200 medidas, en las que se ha medido el tiempo de serialización/deserialización en la transacción GetAllDataValues. Los resultados de la Figura 12 muestran que el tiempo de serialización/deserialización (t_2+t_3) supone más de un 78% del tiempo total (t_t).



Número de ejecuciones medidas = 200

Invoked msg size (bytes)	Received msg size(bytes)
185	67587

	JsonCpp
$t_{1\,(ms)}$	5.970
$t_{2\,(ms)}$	11.205
$t_{3\,(ms)}$	13.720
$t_{t\,(ms)}$	31.762

Influencia de serialización/deserialización en tiempo total $(((t_2+t_3)/t_t*100)$ = **78%**

Figura 12. Tiempos de respuesta en la transacción GetAllDataValues en la implementación REST con JSONcpp

Otra conclusión que se deduce de las pruebas realizadas es que las redes WAN, tanto ADSL, como 3G, tienen una gran impredecibilidad (Tabla 4 y Tabla 5). Los resultados de dos pruebas de una misma transacción resultan completamente diferentes, por lo que resulta imposible establecer una calidad de servicio entorno a estas redes.

4.3. Decisiones tomadas

Después de analizar los resultados y obtener las conclusiones descritas en el punto anterior, se decidió buscar solución a los problemas de ineficiencia en la implementación REST del middleware. Para ello, se realizaron los siguientes cambios:

- Para solucionar el problema de la transacción Reporting, en la que el uso del tipo de mensajes “chunked” provocaba una respuesta ineficiente de la librería libmicrohttpd del servidor, se decidió no usar este tipo de mensajes.

- La solución al segundo problema, en el que en la transacción GetFile de la implementación REST se realizaba una serialización/deserialización innecesaria, consistió en eliminar dicha serialización/deserialización innecesaria.

- Por último, para solucionar el problema de la ineficiencia de la librería JSONcpp, se hizo una búsqueda para encontrar una librería más rápida que

JSONcpp. La librería seleccionada fue RapidJSON, creada para buscar la máxima velocidad de serialización/deserialización [RAP12]. Según el estudio realizado, el parser RapidJSON supera en velocidad al JSONcpp, con una mejora hasta 20 veces superior [PER12].

4.4. Resultados finales

Una vez realizados los cambios mencionados en el apartado anterior, se procedió a realizar otra vez las pruebas.

Tras realizar un análisis comparativo entre las dos implementaciones con REST, una con la librería JSONcpp y la otra con la librería RapidJSON, se observa una mejora considerable de los tiempos de respuesta (Figura 13, Figura 14 y Figura 15). Por ejemplo, en la Figura 13 se refleja una mejora en el tiempo de serialización/deserialización en la transacción GetAllDataValues de un 35%, afectando al tiempo total en un 28%. En las otras dos transacciones, GetDataSetValues y SetDataValues (Figura 14 y Figura 15 respectivamente), aunque la mejora en el tiempo de serialización/deserialización es también grande, 31% y 18%, la influencia en el tiempo total es menor, 5% y 10%. Esto es debido a la menor influencia en estas transacciones del tiempo de serialización/deserialización respecto al tiempo total de la transacción.

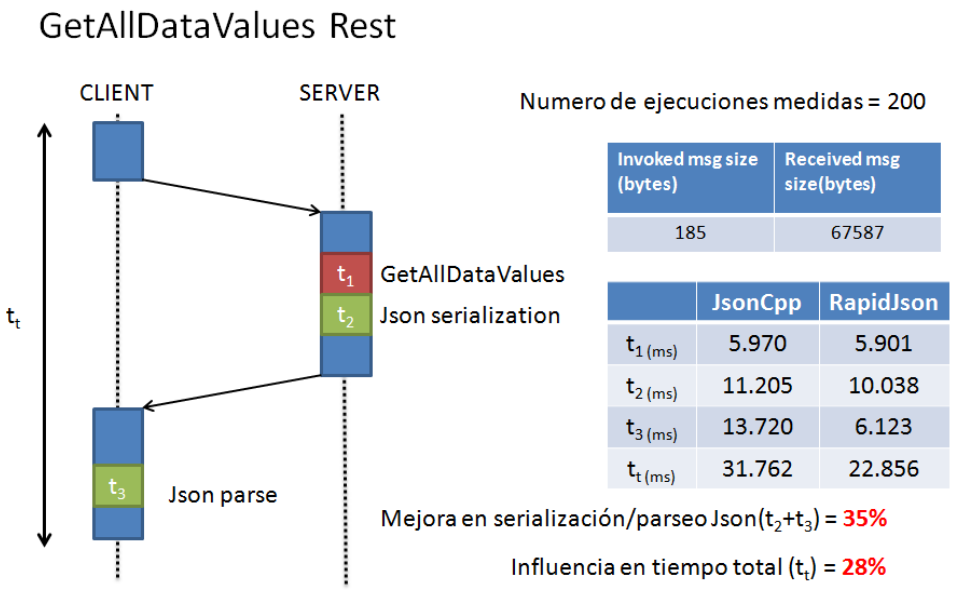
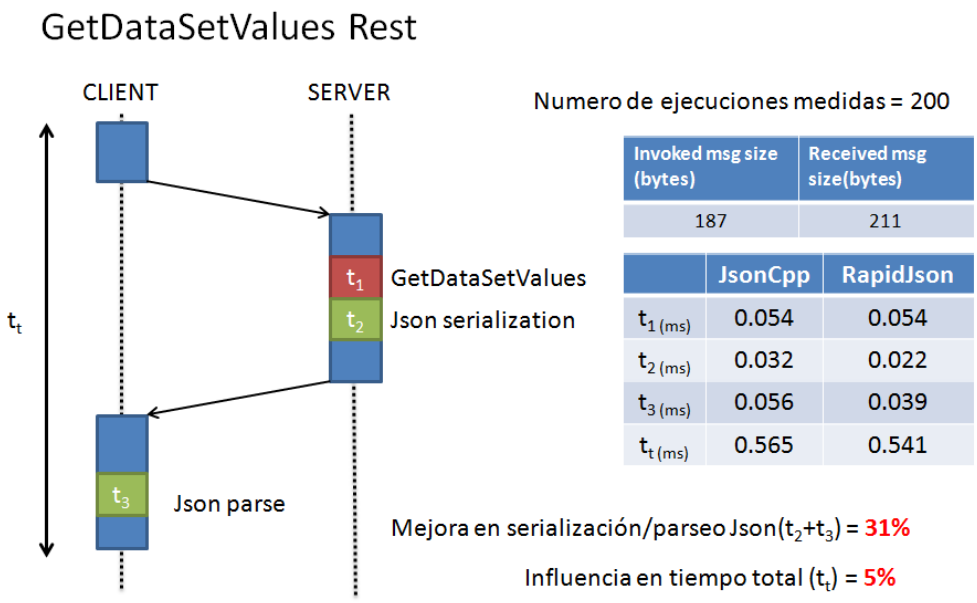


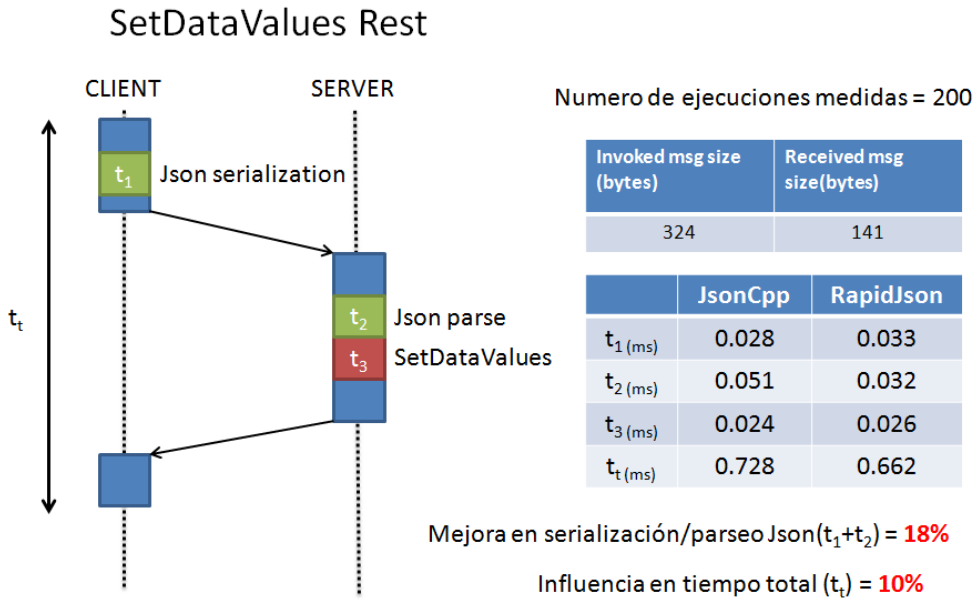
Figura 13. Comparación de tiempos de respuesta en la transacción GetAllDataValues con JSONcpp y con RapidJSON



Invoked msg size (bytes)	Received msg size(bytes)
187	211

	JsonCpp	RapidJson
$t_{1\,(ms)}$	0.054	0.054
$t_{2\,(ms)}$	0.032	0.022
$t_{3\,(ms)}$	0.056	0.039
$t_{t\,(ms)}$	0.565	0.541

Figura 14. Comparación de tiempos de respuesta en la transacción GetDataSetValues con JSONcpp y con RapidJSON



Invoked msg size (bytes)	Received msg size(bytes)
324	141

	JsonCpp	RapidJson
$t_{1\,(ms)}$	0.028	0.033
$t_{2\,(ms)}$	0.051	0.032
$t_{3\,(ms)}$	0.024	0.026
$t_{t\,(ms)}$	0.728	0.662

Figura 15. Comparación de tiempos de respuesta en la transacción SetDataValues con JSONcpp y con RapidJSON

En la Tabla 6 y Tabla 7 se recogen los resultados finales para todas las transacciones y para las 3 implementaciones del middleware, la de SOAP y las dos de REST (con las librerías JSONcpp y RapidJSON). Las mediciones, al igual que en el estudio anterior se han realizado durante 16 horas en cada escenario. Haciendo hincapié en la Tabla 6, que recoge las mediciones realizadas en una red local dedicada, se puede observar que;

aunque los tiempos de respuesta de la implementación REST-RapidJSON han mejorado respecto a los de REST-JSONcpp; estos se sitúan muy parejos a los de la implementación SOAP. Por lo tanto, se siguen sin cumplir las expectativas que marcaban que la implementación de REST debía ser más eficiente en tiempo de respuesta que la de SOAP.

Tabla 6. Resultados finales: Prueba realizada sobre dos PCs en una red local dedicada

		Average Time (ms)	Median Time (ms)	Deviation (ms)	Max Response Time (ms)	Invoked msg size (bytes)	Received msg size(bytes)
GetDataSet	REST-JSONCPP	1.436	1.502	0.303	42.892	187	211
	REST-RAPIDJSON	1.409	1.440	0.153	4.039	187	211
	SOAP	1.749	1.788	0.142	3.949	814	1416
GetAllData	REST-JSONCPP	38.027	37.959	0.341	48.134	185	67587
	REST-RAPIDJSON	32.947	33.913	2.453	45.960	185	67587
	SOAP	31.112	31.074	0.175	36.622	847	255772
GetFile	REST-JSONCPP	182.501	182.421	0.340	206.363	160	2196687
	REST-RAPIDJSON	180.405	180.329	0.309	188.062	160	2196687
	SOAP	240.141	240.081	0.369	261.933	777	2197169
SetData	REST-JSONCPP	2.067	2.089	0.119	9.895	324	141
	REST-RAPIDJSON	1.922	1.930	0.098	5.340	324	141
	SOAP	1.612	1.612	0.140	261.933	1200	1119
Reporting	REST-JSONCPP	2.996	2.261	3.297	31.398	1081	141
	REST-RAPIDJSON	2.910	2.131	3.370	27.628	1081	141
	SOAP	2.299	1.959	1.776	12.500	2767	626

Otro de los problemas en las primeras pruebas era la serialización/deserialización innecesaria que se realizaba en la transacción GetFile de la implementación REST. En la Tabla 2 se puede observar como el tiempo de respuesta en REST eran un 25% superior al de SOAP. En cambio, después de la eliminación del proceso de serialización/deserialización innecesario, se puede observar como el tiempo de respuesta en la implementación REST pasa a ser un 50% menor a la de SOAP (Tabla 7).

El último problema detectado era el de la transacción Reporting, en el que el tiempo de respuesta de la implementación REST era hasta 40 veces mayor que el de la implementación SOAP (Tabla 2). Tras solucionar el problema, eliminando el uso del tipo de mensaje HTTP “chunked”, se puede observar en la Tabla 7 como el tiempo de respuesta de las implementaciones REST es muy similar al de la implementación SOAP.

Tabla 7. Resultados finales: Prueba realizada sobre un único PC

		Average Time (ms)	Median Time (ms)	Deviation (ms)	Max Response Time (ms)	Invoked msg size (bytes)	Received msg size(bytes)
GetDataSet	REST-JSONCPP	0.563	0.565	0.052	5.976	187	211
	REST-RAPIDJSON	0.527	0.529	0.052	5.774	187	211
	SOAP	0.735	0.734	0.035	3.503	814	1416
GetAllData	REST-JSONCPP	32.159	31.953	0.617	40.594	185	67587
	REST-RAPIDJSON	23.331	23.115	0.667	36.853	185	67587
	SOAP	14.918	14.807	0.460	20.848	847	255772
GetFile	REST-JSONCPP	43.986	43.946	0.482	55.274	160	2196687
	REST-RAPIDJSON	44.999	44.943	0.499	56.764	160	2196687
	SOAP	98.025	97.785	1.430	125.199	777	2197169
SetData	REST-JSONCPP	0.782	0.724	0.131	6.313	324	141
	REST-RAPIDJSON	0.583	0.555	0.139	5.178	324	141
	SOAP	0.605	0.580	0.181	5.547	1200	1119
Reporting	REST-JSONCPP	0.953	0.887	1.411	128.941	1081	141
	REST-RAPIDJSON	0.803	0.769	0.324	36.326	1081	141
	SOAP	0.807	0.801	0.232	19.183	2767	626

5. Conclusiones y líneas futuras

Una vez finalizado el proyecto se ha llegado a una serie de conclusiones y se han analizado unas posibles líneas de trabajo a seguir en el futuro.

5.1. Conclusiones

Como se ha mencionado anteriormente, el objetivo principal del proyecto consistía en analizar la calidad de servicio que pueden ofrecer diferentes implementaciones del middleware conforme a la norma IEC61850 sobre diferentes tipos de redes, llevando a cabo para ello una instrumentación de código del propio middleware. Este objetivo se ha llevado a cabo satisfactoriamente obteniendo unos resultados muy interesantes.

La primera conclusión que se saca de este estudio es en relación a la comparativa entre la implementación con servicios web SOAP y la implementación con servicios web REST. En un principio, se creía que la implementación REST obtendría mejores tiempos de respuesta. Esto venía fundamentado por los diferentes estudios comparativos expuestos en el apartado 1.3. A pesar de esto, se ha comprobado que no es así para las dos implementaciones del middleware de este estudio. Además, como se ha demostrado cambiando la librería de serialización JSON, los tiempos de respuesta obtenidos, no solo varían dependiendo de la tecnología seleccionada, si no evidentemente, de las implementaciones concretas que se utilicen. Pudiendo resultar que una implementación de servicios web SOAP parece más optimizada y eficiente que una implementación de servicios web REST.

Por otra parte, del estudio de diferentes tipos de redes, se puede concluir que las redes WAN (tanto ADSL, como 3G) soportadas por los operadores telefónicos sufren de una gran impredecibilidad. Este hecho supone una gran dificultad para su uso en entornos deterministas o en los que sea necesario ofrecer una calidad de servicio mínima.

Por último, la herramienta MyARM ha resultado muy útil para la realización de la instrumentación del código y medición de tiempos de respuesta. Aunque, al no ser una herramienta de código abierto, es difícil determinar cómo afecta el uso mismo de la herramienta en el tiempo de respuesta.

5.2. Líneas futuras

Como continuación a este trabajo se han detectado ciertas líneas en las que seguir investigando.

La primera línea de trabajo que se desprende de este estudio es la de buscar mayor eficiencia en las implementaciones del middleware. Como se ha mencionado, los tiempos de respuesta del middleware parecen dependientes de las librerías que se utilicen para la distribución de los nodos. En el caso de SOAP, la librería gSOAP, y en el caso de REST, las librerías libmicrohttpd, libcurl, JSONcpp o RapidJSON. Resulta interesante la búsqueda de otras librerías que puedan mejorar el rendimiento de éstas.

Este trabajo complementa los trabajos sobre la monitorización y detección de incumplimientos de las calidades de servicios de acuerdo a los contratos SLA (Service Level Agreements) y sería interesante su aplicación en este campo. Actualmente las técnicas utilizadas se basan en la monitorización de la red, pero no incorporan la medición de QoS asociadas al middleware de distribución. Esto en conjunto permitiría medir aspectos de QoS entre los usuarios y los servicios sobre una red no dedicada como Internet.

Por último, resulta interesante desarrollar una herramienta de instrumentación y medición de tiempos propia, que permita adaptarse a las necesidades propias de los proyectos, pero sobre todo, controlar la influencia de la herramienta en los tiempos de respuesta medidos en el middleware o la aplicación.

6. Bibliografía

[HEL05] J. Helander and S. Sigurdsson, "Self-Tuning Planned Actions Time to Make Real-Time SOAP Real," presented at the Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2005.

[TSA06] W. T. Tsai, Y.-H. Lee, Z. Cao, Y. Chen, and B. Xiao, "RTSOA: Real-Time Service-Oriented Architecture," presented at the Proceedings of the Second IEEE International Symposium on Service-Oriented System Engineering, 2006.

[PAN09] M. Panahi, W. Nie, and K.-J. Lin, "A Framework for Real-Time Service-Oriented Architecture," presented at the Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing, 2009.

[MOU10] H. Moussa, T. Gao, I. L. Yen, F. Bastani, and J.-J. Jeng, "Toward effective service composition for real-time SOA-based systems," *Service Oriented Computing and Applications,* vol. 4, pp. 17-31, 2010.

[PED05] P. Pedreiras and L. Almeida, "Approaches to enforce real-time behavior in Ethernet," *The Industrial communication technology handbook,* 2005.

[FEL05] M. Felser. (2005, 6). *Real-time ethernet : Industry prospective*

[HOA02] H. Hoang, M. Jonsson, U. Hagström, and A. Kallerdahl, "Switched Real-Time Ethernet and Earliest Deadline First Scheduling - Protocols and Traffic Handling," presented at the Proceedings of the 16th International Parallel and Distributed Processing Symposium, 2002.

[MEN02] D. A. Menasce, "QoS Issues in Web Services," *IEEE Internet Computing,* vol. 6, pp. 72-75, 2002.

[RAN03] S. Ran, "A model for web services discovery with QoS," *SIGecom Exch.,* vol. 4, pp. 1-10, 2003.

[STR01] F. Strohmeier, H. Dörken, and B. Hechenleitner, *AQUILA distributed QoS Measurement*, 2001.

[GOV04] M. Govindaraju, A. Slominski, K. Chiu, P. Liu, R. v. Engelen, and M. J. Lewis, "Toward Characterizing the Performance of SOAP Toolkits," presented at the Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, 2004.

[MAC05] A. C. C. Machado and C. A. G. Ferraz, "Guidelines for performance evaluation of web services," presented at the Proceedings of the 11th Brazilian Symposium on Multimedia and the web, Pocos de Caldas - Minas Gerais, Brazil, 2005.

[THI05] N. Thio and S. Karunasekera, "Automatic Measurement of a QoS Metric for Web Service Recommendation," presented at the Proceedings of the 2005 Australian conference on Software Engineering, 2005.

[PAK08] P. Pääkkönen and D. Pakkala, "Benchmark of middleware protocols for application and service interaction," presented at the Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia, Umea, Sweden, 2008.

[VIT08] L. De Vito, S. Rapuano, and L. Tomaciello, "One-Way Delay Measurement State of the Art," *IEEE Transactions on Instrumentation and Measurement,* 2008.

[PAS02] A. Pásztor and D. Veitch, "PC based precision timing without GPS," Marina Del Rey, CA, 2002, pp. 1-10.

[HAN06] A. Hanzlik and A. Ademaj, "A Composable Algorithm for Clock Synchronization in Multi-Cluster Real-Time Systems," *4th Workshop on Intelligent Solutions in Embedded Systems - (WISES06), Proceedings of the,* 2006.

[ARM12] OpenGroup. *ARM Website*, https://collaboration.opengroup.org/tech/management/arm/

[TAU12] *TAU website*, http://www.cs.uoregon.edu/Research/tau/home.php

[SHE06] S. S. Sameer and D. M. Allen, "The Tau Parallel Performance System," ed, 2006.

[PAA11] P. Pääkkönen, J. Prokkola, and A. Lattunen, "Instrumentation-based tool for latency measurements," Karlsruhe, 2011, pp. 403-412.

[MYA12] *MyARM Website*, http://www.myarm.com/

[IEC12] IEC, IEC 61850 - Communication Networks and Systems in Substations, http://webstore.iec.ch/webstore/webstore.nsf/mysearchajax?Openform&key=IEC%2061850&sorting=&start=1&onglet=1.

[MAC06] R. E. Mackiewicz, "Overview of IEC 61850 and Benefits," presented at the Power Systems Conference and Exposition, 2006. PSCE '06. 2006 IEEE PES, Atlanta, GA, 2006.

[GSO12] *gSOAP Web Site*, http://www.cs.fsu.edu/~engelen/soap.html

[LMH12] *libmicrohttpd Web Site*, http://www.gnu.org/software/libmicrohttpd/

[LCU12] *libcurl Web Site*, http://curl.haxx.se/libcurl/

[JSO12] *JSONcpp Web Site*, http://jsoncpp.sourceforge.net/

[FIE00] R. T. Fielding, "Architectural styles and the design of network-based software architectures," University of California, Irvine, 2000.

[HAM10] H. Hamad, M. Saad, and R. Abed, "Performance Evaluation of RESTful Web Services for Mobile Devices," *International Arab Journal of e-Technology,* vol. 1, pp. 72-78, 2010.

[KUM11] P. K. Potti, "On the Design of Web Services SOAP vs. REST," Computing, University of North Florida, 2011.

[PAU08] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big"' web services: making the right architectural decision," presented at the Proceedings of the 17th international conference on World Wide Web, Beijing, China, 2008.

[RAP12] *RapidJSON Web Site*, https://code.google.com/p/rapidjson/

[PER12] *Performance comparison between RapidJSON and other parsers*, https://code.google.com/p/rapidjson/wiki/Performance

7. Acrónimos

QoS Quality of Service (Calidad de Servicio)

SOA Service Oriented Arquitecture (Arquitectura Orientada a Servicio)

GPS Global Positioning System (Sistema de Posicionamiento Global)

NTP Network Time Protocol (Protocolo de Tiempo de Red)

TAU Tuning and Analysis Utilities (Herramientas de Optimización y Análisis)

ARM Application Response Measurement (Medición de Respuesta de Aplicación)

API Application Programming Interface (Interfaz de Programación de Aplicación)

IEC International Electronical Commission (Comisión Electrónica Internacional)

DER Distributed Energy Resources (Recursos Energéticos Distribuidos)

SOAP Service Oriented Arquitecture Protocol (Protocolo de Arquitectura Orientada a Servicio)

REST Representational State Transfer (Transferencia de Estado Representacional)

XML Extensible Markup Language (Lenguaje de Marcas Extensible)

HTTP Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto)

HTML Hypertext Markup Language (Lenguaje de Marcas de Hipertexto)

WAN Wireless Area Network (Red de Área Extensa)

ADSL Asymmetric Digital Subscriber Line (Línea de Abonado Digital Asimétrica)

PCC Puesto de Control Central

JSON JavaScript Object Notation (Notación de Objetos JavaScript)