UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TESIS DOCTORAL

SOSLite: Soporte para Sistemas Ciber-Físicos y Computación en la Nube

Autor: Juan Vicente Pradilla Cerón

Director: Carlos E. Palau Salvador

Valencia, Noviembre 2016

Resumen

Los Sistemas Ciber-Físicos (CPS) se han convertido en uno de los temas de investigación con mayor proyección en la actualidad; debido a que plantean una nueva disciplina compleja, que aborda sistemas existentes y futuros de gran auge como: la Internet, la Internet de las Cosas, las redes de sensores y las redes eléctricas inteligentes. Como disciplina en gestación, existen muchas posibilidades para aportar al estado del arte, siendo la interoperabilidad uno de los más relevantes.

Así, esta tesis se ha creado en el marco de la interoperabilidad para los CPS, mediante la utilización del estándar SOS (Sensor Observation Service) perteneciente al marco de trabajo SWE (Sensor Web Enablement) del OGC (Open Geospatial Consortium). Se ha desarrollado para dar surgimiento a una nueva línea de investigación dentro del grupo SATRD (Sistemas y Aplicaciones de Tiempo Real Distribuidos) del Departamento de Comunicaciones de la UPV (Universitat Politècnica de València).

La aproximación con la cual se ha abordado la interoperabilidad en los CPS es de tipo sintética (pasar de las partes al todo), iniciando desde una solución, verificable y realizable, para la interoperabilidad en las redes de sensores, uno de los CPS más significativos debido a que se integra en muchos otros CPS, y pasando a adaptar y comprobar dicha solución en CPS de mayor complejidad, como la Internet de las Cosas.

De esta forma, se propone una solución de interoperabilidad en las redes de sensores fundamentada en el SOS, pero adaptada a unos requerimientos que hacen de este mecanismo una versión más ligera del estándar, con lo que se facilita el despliegue de futuras implementaciones debido a la posibilidad de emplear dispositivos limitados para tal fin. Dicha solución teórica, se lleva a una primera implementación, denominada SOSLite, la cual se prueba para determinar su comportamiento característico y verificar el cumplimiento de su propósito.

De forma análoga y partiendo de la misma solución teórica, se proyecta una segunda implementación, llamada SOSFul, la cual propone una actualización del estándar SOS de forma que sea más ligero, eficiente y fácil de emplear. El SOSFul, tiene una proyección más ambiciosa al abordar la Internet de las Cosas, un CPS más complejo que las redes de sensores. Como en el caso del SOSLite, se realizan pruebas y se valida mediante un caso de uso.

Así, tanto el SOSLite como el SOSFul se proyectan como soluciones de interoperabilidad en los CPS. Ambas implementaciones parten de la propuesta teórica de SOS ligero y se encuentran disponibles de forma gratuita y bajo código libre, para ser empleados por la comunidad investigativa para continuar su desarrollo y aumentar su uso.

Resum

Els sistemes ciberfísics (CPS, C*yber-Physical Systems*) s'han convertit en un dels temes de recerca amb major projecció en l'actualitat, a causa del fet que plantegen una nova disciplina complexa que aborda sistemes existents i futurs de gran auge, com ara: la Internet, la Internet de les Coses, les xarxes de sensors i les xarxes elèctriques intel·ligents. Com a disciplina en gestació, hi ha moltes possibilitats per a aportar a l'estat de la qüestió, sent la interoperabilitat una de les més rellevants.

Així, aquesta tesi s'ha creat en el marc de la interoperabilitat per als CPS, mitjançant la utilització de l'estàndard SOS (Sensor Observation Service) pertanyent al marc de treball SWE (Sensor Web Enablement) de l'OGC (Open Geospatial Consortium). S'ha desenvolupat per a iniciar una nova línia de recerca dins del Grup de SATRD (Sistemes i Aplicacions de Temps Real Distribuïts) del Departament de Comunicacions de la UPV (Universitat Politècnica de València).

L'aproximació amb la qual s'ha abordat la interoperabilitat en els CPS és de tipus sintètic (passar de les parts al tot), iniciant des d'una solució, verificable i realitzable, per a la interoperabilitat en les xarxes de sensors, un dels CPS més significatius pel fet que s'integra en molts altres CPS, i passant a adaptar i comprovar aquesta solució en CPS de major complexitat, com la Internet de les Coses.

D'aquesta forma, es proposa una solució d'interoperabilitat en les xarxes de sensors fonamentada en el SOS, però adaptada a uns requeriments que fan d'aquest mecanisme una versió més lleugera de l'estàndard, amb la qual cosa es facilita el desplegament de futures implementacions per la possibilitat d'emprar dispositius limitats a aquest fi. Aquesta solució teòrica es porta a una primera implementació, denominada SOSLite, que es prova per a determinar el seu comportament característic i verificar el compliment del seu propòsit.

De forma anàloga i partint de la mateixa solució teòrica, es projecta una segona implementació, anomenada SOSFul, que proposa una actualització de l'estàndard SOS de manera que siga més lleuger, eficient i fàcil d'emprar. El SOSFul té una projecció més ambiciosa quan aborda la Internet de les Coses, un CPS més complex que les xarxes de sensors. Com en el cas del SOSLite, es realitzen proves i es valida mitjançant un cas d'ús.

Així, tant el SOSLite com el SOSFul, es projecten com a solucions d'interoperabilitat en els CPS. Ambdues implementacions parteixen de la proposta teòrica de SOS lleuger, i es troben disponibles de forma gratuïta i en codi lliure per a ser emprades per la comunitat investigadora a fi de continuar el seu desenvolupament i augmentar-ne l'ús.

Abstract

Cyber-Physical Systems (CPS) have become one of the greatest research topics today; because they pose a new complex discipline, which addresses big existing and future systems as the Internet, the Internet of Things, sensors networks and smart grids. As a recent discipline, there are many possibilities to improve the state of the art, interoperability being one of the most relevant.

Thus, this thesis has been created within the framework of interoperability for CPS, by using the SOS (Sensor Observation Service) standard, which belongs to the SWE (Sensor Web Enablement) framework of the OGC (Open Geospatial Consortium). It has been developed to give rise to a new line of research within the Distributed Real-Time Systems and Applications group (SATRD for its acronym in Spanish) from the Communications Department of the Polytechnic University of Valencia (UPV for its acronym in Valencian).

The approach, with which the interoperability in the CPS has been addressed, is of synthetic type (from parts to whole), starting from a verifiable and workable solution for interoperability in sensor networks, one of the most significant CPSs because it is integrated in many other CPSs, next adapting and testing the solution in more complex CPS, such as the Internet of Things.

In this way, an interoperability solution in sensor networks is proposed based on the SOS, but adapted to some requirements that makes of this mechanism a lighter version of the standard, which facilitates the deployment of future implementations due to the possibility of using limited devices for this purpose. This theoretical solution is brought to a first implementation, called SOSLite, which is tested to determine its characteristic behavior and to verify the fulfillment of its purpose.

Analogously, and starting from the same theoretical solution, a second implementation is projected called SOSFul, which proposes an update to the SOS standard so that it is lighter, more efficient and easier to use. The SOSFul, has a more ambitious projection by addressing the Internet of Things, a more complex CPS than sensors networks. As in the case of the SOSLite, tests are performed and validation is made through a use case.

So, both the SOSLite and the SOSFul are projected as interoperability solutions in the CPS. Both implementations are based on the theoretical proposal of a light SOS and are available for free and under open source licensing so that it can be used by the research community to continue its development and increase its use.

Dedicatoria

A los que han coincidido en este tiempo y lugar, para hacer de él una extensión del amor de Dios. Mi esposa amorosa y consejera. Mis padres sabios y dispuestos. Mi familia incondicional y afectuosa.

Agradecimientos

A todos aquellos que con su sabiduría han instruido mi amor por el conocimiento y han aportado al crecimiento de las virtudes que Dios ha depositado en mi vida. En especial a Carlos E. Palau Salvador Ph.D. que con su consejo e instrucción ha guiado y apoyado mi desarrollo profesional y la realización de este trabajo.

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Introducción

1.1.1. Resumen

La tesis aborda el problema de interoperabilidad en los Sistemas Ciber-Físicos (*Cyber-Physical Systems* - CPS), partiendo desde una de sus expresiones más adoptadas hasta la fecha, las Redes de Sensores (*Sensor Networks* - SN), y tomando las mismas como base para el desarrollo de una solución más general; pasando por una validación sobre uno de los CPS de mayor investigación en la actualidad, la Internet de las Cosas (*Internet of Things* - IoT).

La articulación de la tesis como un proyecto de investigación aplicada, que involucra el desarrollo y la validación de prototipos, la gestión del proyecto, la divulgación y la escritura de la memoria misma; se corresponde con la respuesta a un problema que se ha hecho evidente con el paso del tiempo sobre los CPS, la interoperabilidad.

Es por tal motivo, que este capítulo de introducción aborda la definición del problema detectado, las motivaciones para buscar la solución al problema, los objetivos que rigen el desarrollo del proyecto investigativo, el alcance demarcado para dar una solución dentro del marco concreto, la metodología seguida y las aportaciones realizadas en el campo de investigación.

1.1.2. CPS: SN e IoT

En la actualidad, las maquinas se comunican entre sí en procesos automatizados y colaborativos, cuyos datos son almacenados y analizados por sistemas de informáticos; los cuales a su vez, generar información consolidada útil para humanos y maquinas, quienes toman decisiones o ejecutan acciones que repercuten en el comportamiento del sistema completo, creando así un ciclo cerrado de información. A estos sistemas información/comunicación de ciclo cerrado que influyen y son influidos por la realidad física, se les conoce como CPS.

Los CPS nacen de una intersección precisa entre computación, redes y procesos físicos [1]. Estos perciben el mundo mediante SN y lo afectan haciendo uso de redes de actuadores, robots y UAV (*Unmanned Aerial Vehicle*) [2]; mientras que las operaciones son monitorizadas, coordinadas, controladas e integradas por un núcleo de computación y redes [3]. De esta forma se crea un ciclo de retroalimentación donde los procesos físicos afectan los cálculos y viceversa.

La parte del núcleo compuesto por las redes de comunicación, la cuales permiten la interconexión entre los componentes de un CPS, son fundamentales y deben ajustarse al entorno en el cual se despliegan los sensores/actuadores y a los requerimientos del sistema para prestar servicios específicos, como pueden ser tiempo de respuesta, ancho de banda o resistencia a interferencias.

De igual forma, la parte del núcleo asociada a la computación, encargada de la coordinación, la monitorización y el control de las operaciones; debe desplegar sistemas informáticos que almacenen y procesen datos para generar información que puede ser empleada de forma automatizada o con la intervención humana. Esta parte del núcleo bien puede ser centralizada o distribuida, y estar desplegado en una amplia verdad de plataformas de hardware: desde un dispositivo limitado hasta un conjunto de grandes

centro de datos (*datacenters*). Además, se pueden integrar con los paradigmas basados en virtualización de *Fog Computing (FC)* y *Cloud Computing (CC)*.

Existen diversos ejemplos de CPS en la actualidad, como son: sistemas aeroespaciales, dispositivos médicos, vehículos de transporte y carreteras inteligentes, sistemas de defensa, robots, control de procesos, automatización de fábricas, control de construcciones, control de medio ambiente, espacios inteligentes y muchos otros [3]. Sin embargo, hay dos de ellos que destacan sobre los demás: las SN y la IoT [1].

Las SN son redes, con infraestructura o sin ella, compuestas por nodos sensores (Figura 1) que monitorizan condiciones físicas o ambientales, como: temperatura, sonido, vibración, presión, movimiento, etc. [4]. Cada nodo sensor está compuesto por los siguientes subsistemas: sensado, procesamiento, energía, memoria y comunicación [5].

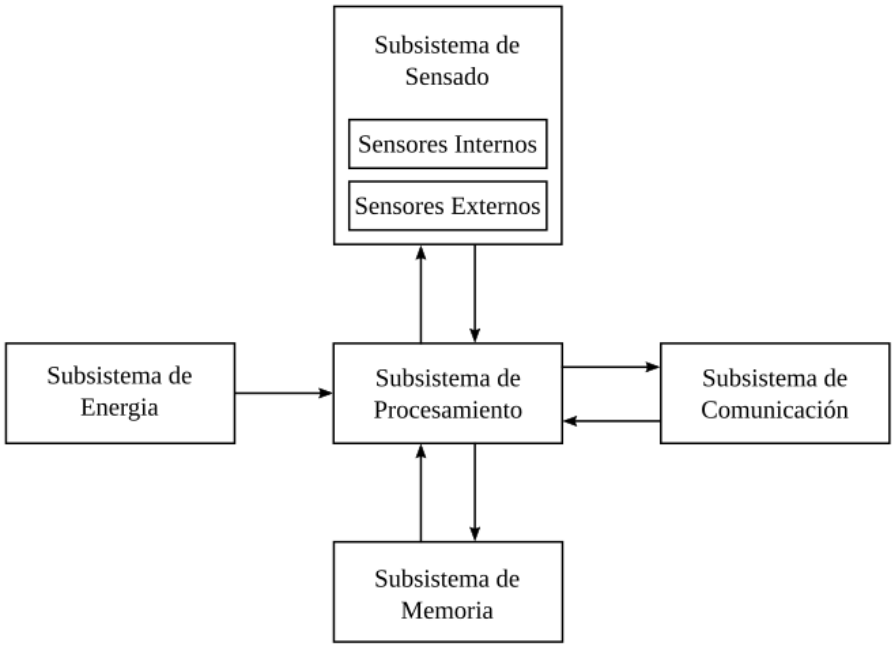


Figura 1 - Estructura de un nodo sensor

Por su parte, la IoT ha sido definida de muchas formas y se puede considerar un concepto aun en desarrollo que en muchas ocasiones traspasa los límites establecidos por una definición específica. Esta tesis se guiará por la definición del *European Research Cluster on IoT* (IERC), que es acorde a la visión de considerar a la IoT como un caso especial de los CPS.

Según el IERC la IoT es "una infraestructura dinámica de red global con la capacidad de auto-configuración basada en protocolos de comunicación estándar e interoperables, donde "cosas" físicas y virtuales tienen identidades, atributos físicos y personalidades virtuales, utilizan interfaces inteligentes, y están perfectamente integrados en la red de información" [6]. Esta definición contiene conceptos importantes, de los que se resaltan que la IoT es una red global y que la IoT integra mediante una red de información "cosas" físicas y virtuales.

En definitiva, en la actualidad se cuenta con la capacidad para crear, modificar y compartir una representación virtual de la realidad física y mediante esta, generar

acciones que modifiquen la realidad. Al conjunto de todas estas capacidades se lo conoce como CPS, comprendiendo: las SN, las redes de actuadores, los objetos inteligentes, las comunicaciones máquina a máquina y la IoT, entre muchas otras tecnologías.

1.1.3. Abordando la interoperabilidad en los CPS

Dada su naturaleza integradora, los CPS se enfrentan a un enorme reto al llevarse a la implementación. Este reto es la interoperabilidad, que es la habilidad de dos o más plataformas o componentes para intercambiar y utilizar información [7], y surge debido a la heterogeneidad de dispositivos, redes y protocolos que deben interactuar dentro de las diferentes tecnologías que se emplean en la construcción de los CPS.

Se pueden establecer cuatro niveles de interoperabilidad (Figura 2) [8] los cuales permiten a los sistemas heterogéneos interactuar con independencia de las tecnologías utilizadas y los fabricantes que proveen estas tecnologías. Los cuatro niveles son:

1. Interoperabilidad técnica [7][8]: se centra en los protocolos de comunicación y la infraestructura necesaria (componentes de hardware y software, sistemas y plataformas) para que estos funcionen de forma que provean comunicación entre dispositivos.

2. Interoperabilidad sintáctica [7][8]: se asocia con los formatos de datos de los mensajes que intercambian los dispositivos, ya que estos debe tener una sintaxis y codificación bien definida.

3. Interoperabilidad semántica [7][8][9]: implica el conjunto de técnicas necesarias para permitir que el significado de la información pueda ser compartida.

4. Interoperabilidad organizacional [7][8]: se refiera a la capacidad entre organizaciones de contar con una comunicación efectiva sin importar los componentes TIC que estos utilicen.

Figura 2 - Niveles de interoperabilidad

La respuesta para abordar la interoperabilidad viene de parte de los estándares, los protocolos y las convenciones [10], donde son muchas las propuestas que en la actualidad intentan ser la referencia dentro de los CPS, sin tener aún una opción que impere sobre las demás. Para esta tesis se ha optado por dar abordar el reto de interoperabilidad a una porción de los CPS, concretamente a las SN, con miras a que paulatinamente se integre con el resto de estos sistemas.

Así, esta memoria manifiesta el trabajo realizado para encontrar y probar una solución al problema de interoperabilidad en los CPS. La solución encontrada, está centrada en brindar un mecanismo de interoperabilidad semántica, a partir de los desarrollos de las SN y generalizándolo para abarcar un mayor rango de CPS.

1.2. Motivaciones

Los CPS se hacen cada día más omnipresentes a nivel mundial, hecho que conlleva que su impacto en las actividades sociales, culturales y económicas, también sean cada vez más importante con el transcurrir del tiempo. De forma que, investigar en este campo es relevante y con un potencial de impacto a gran escala.

Sin embargo, esta importancia de la cual están revestidos los CPS, también implica que los problemas y retos a los que se enfrentan estos sistemas, tienen una influencia significativa sobre las actividades humanas. En el caso concreto de la interoperabilidad, afecta el desarrollo tecnológico y económico que tiene sobre los CPS. Por ejemplo, en el caso de la IoT, se ha estimado que el 40% de los beneficios potenciales no se podrán obtener sin dar respuesta a este reto [11].

Los CPS son un tema demasiado extenso y variado para ofrecer una solución global y poder comprobar la veracidad y la confiabilidad de la misma en un tiempo reducido, con lo que, abordar un subconjunto de estos sistemas e ir escalando es la vía más sensata para realizar una investigación cabal sobre la interoperabilidad y su impacto en los CPS.

Así, esta tesis aborda el problema de la interoperabilidad en los CPS, tomando como punto de partida las SN. Comenzar con las SN es una consecuencia de la importancia de estas dentro de los CPS; importancia asociada a que estas redes constituyen por si mismas un CPS y además, forman parte de muchos otros CPS, como es el caso de la estructura de IoT [12].

Dado que las SN llevan varios años en desarrollo, se han presentado una gran multitud de opciones para proveerles de interoperabilidad. Dentro estas opciones, destacan:

- El marco de trabajo *Sensor Web Enablement* (SWE) [13], publicado por el OGC, y que abarca siete estándares, los cuales son ampliamente aceptados y utilizados en implementaciones existentes.

- *OneM2M Global Initiative* [14], el cual es un proyecto de cooperación internacional, establecido con el fin de producir especificaciones de la capa de servicio, relacionadas con las soluciones *machine-to-machine* (M2M), que sean de acceso independiente y aplicables a nivel mundial.

- Los dos estándares de *Open Data Format* (O-DF) [15] y *Open Messaging Interface* (O-MI) [16] publicados por *The Open Group*, centrados en la interoperabilidad semántica y recientemente actualizados para ajustarse a la interoperabilidad en IoT.

Dentro del SWE, el estándar *Sensor Observation Service* (SOS) [17] presenta un servicio para el almacenamiento de datos y meta-datos de sensores y observaciones, es una vía para la interoperabilidad semántica que resulta interesante, debido a su independencia de las tecnologías de red y de los fabricantes de dispositivos.

Considerar el SOS como un eslabón para alcanzar la interoperabilidad en los CPS, siendo este un servicio transversal y basado en estándares ya consolidados, no sólo

parece plausible, sino que además puede ser viable y deseable.

Para esto, es necesario considerar el funcionamiento de los CPS cuando son llevados a las implementaciones reales. En estos casos se presentan dos alternativas para la arquitectura de los dispositivos que las componen: centralizada y distribuida (Figura 3).

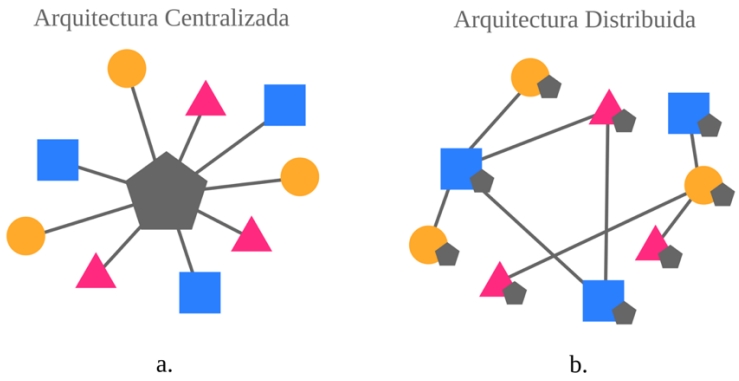


Figura 3 - Arquitectura centralizada y arquitectura distribuida

Cuando se tiene una arquitectura centralizada los recursos computacionales residen en un dispositivo o un conjunto de ellos (*Cluster*). Los datos puedan ser almacenados y procesados para obtener información a partir de ellos, la cual es accesible remotamente, desde cualquier punto de la red.

Por su parte, en la arquitectura distribuida, los recursos computacionales se encuentran en varias ubicaciones (*Sites*), los cuales son autosuficientes para almacenar datos y procesarlos, sin necesidad de llevarlo a un punto central. En esta aproximación cada *site* cuentan con sus respectivos dispositivos o bloques/grupos (*Cluster*) de dispositivos y pueden acudir a un nodo central para operaciones específicas.

Ante estas dos arquitecturas, existe la posibilidad de integrarlas con los paradigmas de FC y CC; permitiendo el uso de máquinas virtuales, que promueven el aislamiento de procesos, y la gestión de asignación de recursos de forma dinámica y bajo demanda, que facilitan la eficiencia en el aprovechamiento de los recursos computacionales.

El uso de los paradigmas CC/FC permiten una mayor versatilidad para afrontar la variabilidad de casos de uso a los que atienden los CPS. El FC acerca la asignación de recursos dinámicos a los usuarios finales, mientras que el CC brinda unos recursos virtualmente ilimitados que pueden ser accedidos bajo petición.

Si se profundiza en las implementaciones disponibles del estándar SOS y las posibilidades que estas brindan; se aprecia que solo existe una alternativa abierta que se encuentra en una versión estable y que brinda todas las posibilidades necesarias para un despliegue en un entorno real. Esta implementación es *52 North SOS* [18], la cual se amolda adecuadamente a las SN con arquitectura centralizada y basada en la nube.

Las características del *52 North SOS* hacen que se requiera un dispositivo con amplios recursos computacionales para funcionar de forma correcta. Este hecho, conlleva a que las aproximaciones a las SN distribuidas, que utilicen este software, requieran de grandes inversiones de capital y que las aplicaciones masivas sean inviables. Si esa

consecuencia se extrapola a grandes CPS, como la IoT, es un claro limitante para el uso del SOS como facilitador de la interoperabilidad a gran escala.

Esta primera limitación, vinculada a la parte de computación del núcleo de los CPS no es la única. Si se profundiza en los requerimientos de ancho de banda propios del SOS, se encuentra que las redes de comunicaciones de las SN que utilizan al SOS como mecanismo de interoperabilidad necesitan una asignación amplia de ancho de banda. Así, cuando se tienen despliegues con gran cobertura o una cantidad importante de sensores se debe descartar este estándar para la interoperabilidad semántica.

1.3. Objetivos

A partir de los precedentes y las motivaciones, la presente tesis atiende a la resolución del problema de la interoperabilidad, centrándose en el nivel semántico, contribuyendo de esta forma al futuro despliegue y utilización masiva de los CPS; incluyendo posibilidades de integración con los paradigmas de CC y FC.

1.3.1. Objetivo global del proyecto de tesis

El objetivo global de la tesis es brindar un mecanismo de interoperabilidad basado en los estándares SWE del OGC, que puedan ser aplicadas en arquitecturas distribuidas a gran escala y que se integren con los paradigmas de CC y FC; facilitando así la apropiación masiva de los CPS en los próximos años.

1.3.2. Objetivos específicos

El objetivo principal de la tesis se desglosa en los siguientes objetivos específicos que se relacionan con las motivaciones de la creación de la tesis son:

- Analizar diversas arquitecturas de interoperabilidad para las SN y el IoT, como casos representativos de los CPS; y determinar la idoneidad de un mecanismo de interoperabilidad basado en los estándares SWE.

- Diseñar un mecanismo de interoperabilidad para SN que sea extensible a otros CPS en los cuales se utilicen los sensores, como medio de generalización para la interoperabilidad en los CPS.

- Desarrollar un prototipo de SOS que se adapte a los requerimientos de las redes distribuidas de gran escala. El cual, debe seguir el estándar y las recomendaciones de la OGC, soportando las operaciones más utilizadas dentro de las SN actualmente desplegadas. Además, el prototipo debe ser modelando siguiendo una arquitectura de software que priorice el uso de recursos limitado y la velocidad en la respuesta a las operaciones que soporte.

- Realizar pruebas al prototipo del SOS, verificando su idoneidad como mecanismo de interoperabilidad en las SN distribuidas, mediante la caracterización de su desempeño en condiciones controladas. Estas pruebas deben enfatizar los despliegues sobre dispositivos limitados y como micro-instancia de CC/FC, para determinar si es una opción adecuada para la masificación de los CPS en el futuro próximo.

- A partir de la evaluación del desarrollo inicial, crear un prototipo de SOS mejorado que sirva de referencia para las nuevas especificaciones del estándar SOS, que tenga en cuenta la intenciones pronunciadas por la OGC y las necesidades de los CPS actuales, principalmente las de IoT.

- Definir, implementar y probar dos casos de uso para los desarrollos de SOS generados. De forma que se ejemplifique la potencialidad de su uso en los conceptos principales a los que atiende la tesis.

1.4. Alcance y Metodología

1.4.1. Alcance de la tesis

Este proyecto de tesis está orientado a generar conocimiento práctico que sea válido para extender el área de ingeniera dentro del campo de la telecomunicaciones; más específicamente, dando una respuesta al reto de la interoperabilidad dentro de los CPS distribuidos partiendo de una aproximación a las SN que pueda ser extensibles a otros sistemas.

Tal como lo detallan los objetivos, el proyecto analiza, diseña, desarrolla, valida y mejora implementaciones de SOS ligero, como mecanismos de interoperabilidad en las SN, considerando estas mismas como sistema primario dentro de los CPS distribuidos. Para esto, se mantienen las condiciones controladas y los niveles de detalle ajustable en el contexto de laboratorio tecnológico.

Finalmente, se plantean algunos casos de uso en los cuales se puede hacer uso de la solución planteada, en su versión de producción, de forma que se esclarezca la utilidad de esta, fuera de las condiciones de laboratorio.

1.4.2. Metodología

Para la realización de la tesis se empleará la metodología de marco lógico [19]. En esta metodología la ejecución exitosa de un proyecto es consecuencia de la realización de un conjunto de acontecimientos que guardan una relación causal. Con este fin, se describen cada uno de los niveles: actividades, resultados, objetivo del proyecto y objetivo global; mientras que, las incertidumbres del proyecto se explican mediante supuestos asociados con cada nivel.

En el caso del proyecto de tesis el objetivo se corresponde con el anteriormente mencionado y el mismo se verifica mediante seis indicadores objetivamente verificables:

1. Existencia de un SOS basado en estándares y adecuado a las necesidades de las SN distribuidas.

2. Resultados de las prueba del SOS ligero desplegado en un dispositivo limitado.

3. Resultados de las prueba del SOS ligero desplegado como micro instancia en la nube.

4. Existencia de un SOS mejorado basado en los aprendizajes del desarrollo inicial y que este adaptado a las exigencias de los CPS actuales y que sirva de referencia para nuevas especificaciones del estándar SOS.

5. Resultados de pruebas comparativas entre los desarrollos realizados y el software de referencia.

6. Propuestas del uso del SOS Ligero en casos de uso fuera del laboratorio.

Además se pueden identificar los tres resultados o entregables esperados de la tesis:

1. SOSLite para despliegue en dispositivos limitados y en como micro-instancia nube.

2. SOSFul para el uso dentro de la IoT, bien sea desplegado como micro-instancia en la nube o en un dispositivo limitado.

3. Propuestas de casos de uso detallados.

Los mismos se verificaran con el cumplimento de los siguientes indicadores (ver las principales aportaciones):

- Existencia del SOSLite publicado bajo licencia de código abierto. Se ha realizado el registro de software y publicado un artículo en congreso y un artículo en revista relacionado con el desarrollo.

- Existencia del SOSFul publicado bajo licencia de código abierto. Se ha realizado el registro de software de la implementación y de un proxy de interacción con CoAP. Además se publicado un artículo en congreso y se encuentra a la espera de la aceptación de un artículo en revista.

- Existencia de las propuestas de casos de uso de los desarrollos y sus pruebas. A partir de los casos de usos recogidos en la tesis se han asociado proyectos de investigación concedidos por la Unión Europea y el gobierno Español.

Del cumplimiento de los resultados se ha derivado el logro de los objetivos y estos a su vez han sido suficientes para alcanzar con éxito el objetivo global de la tesis. Para complementar la descripción del proyecto de tesis, a continuación, se presenta un listado de las actividades del proyecto agrupadas por áreas:

1. Documentación (Tabla I): engloba aquellas actividades de búsqueda, selección y recopilación bibliográfica relacionada con: SN, IoT y CPS; tanto como: SWE, y SOS. De igual forma, se incluye la generación y divulgación de conocimiento mediante congresos, capítulos de libro y artículos en revista.

2. Programación (Tabla II): consiste en el análisis, el diseño, la evaluación y selección de las tecnologías a utilizar y la implementación de los prototipos.

3. Pruebas (Tabla III): en esta área se configuran y desarrollan las pruebas para determinar el funcionamiento correcto de los prototipos en el entorno de laboratorio. Como también, la generación de las propuestas de casos de uso.

4. Análisis (Tabla IV): radica en analizar y evaluar los resultados obtenidos en las actividades de pruebas y determinar el valor del prototipo y su posible funcionalidad dentro de casos de uso reales. Mediante estas actividades se establece la viabilidad de la solución como respuesta al reto de la interoperabilidad dentro de las SN y los CPS.

Tabla I - Área 1 - Documentación

No.	Actividades
1	Búsqueda de información bibliográfica
2	Análisis de información
3	Selección de información bibliográfica
4	Redacción del estado del arte
5	Documentación del código
6	Elaboración de la memoria de tesis
7	Participaciones en congresos
8	Redacción del capítulo de libro
9	Participaciones en revistas

Tabla II - Área 2 - Programación

No.	Actividades
1	Análisis de requerimientos funcionales y no funcionales
2	Determinación del diseño de la aplicación
3	Definición de la arquitectura
4	Selección de herramientas de programación
5	Implementación del código
6	Gestión de versiones

Tabla III - Área 3 - Pruebas

No.	Actividades
1	Definición del diseño de las pruebas
2	Implementación de las pruebas
3	Ejecución de las pruebas
4	Formulación de los casos de usos

Tabla IV - Área 4 - Análisis

No.	Actividades
1	Recolección de los datos de las pruebas
2	Normalización de los datos de las pruebas
3	Análisis de los datos

En las actividades de programación, se ha utilizado una metodología de desarrollo iterativa e incremental (Figura 4). Por consiguiente, la aplicación se realiza cíclicamente, empezando por una planificación inicial y finalizando con una despliegue; teniendo en medio una serie de etapas: análisis y diseño, implementación, pruebas y evaluación; en las cuales las partes del prototipo, se construyen en diferentes momentos o a diferentes velocidades, e integran a medida que se completan [20].

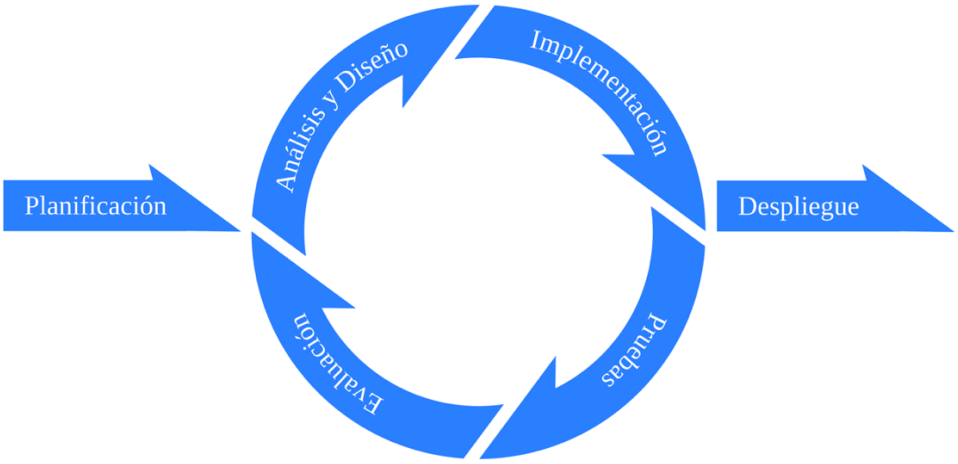


Figura 4 - Metodología de desarrollo iterativa e incremental

1.5. Principales aportaciones

1.5.1. Artículos en revista

- **Juan Pradilla**, Carlos Palau, Manuel Esteve; *"SOSLite: Lightweight Sensor Observation Service (SOS)"*; IEEE Latin America Transactions, Volume: 13, Issue: 12, pp: 3758 – 3764, December 2015.

1.5.2. Capítulo de libro

- **J.V. Pradilla**, C.E. Palau; *"Micro Virtual Machines (MicroVMs) for Cloud-Assisted Cyber-Physical Systems (CPS)"*; Internet of Things: Principles and Paradigms; pp. 125 – 142; June 2016.

1.5.3. Artículos en congreso

- **Juan Pradilla**, Carlos Palau, Manuel Esteve; *"SOSLite: Lightweight Sensor Observation Service (SOS) for the Internet of Things (IOT)";* ITU Kaleidoscope: Trust in the Information Society (K-2015), 9-11 December, Barcelona (Spain), 2015.

- **Juan Pradilla**, Regel González, Manuel Esteve, Carlos Palau; *"Sensor Observation Service (SOS) / Constrained Application Protocol (CoAP) Proxy Design"*; Melecon 2016; 18-20 April, Limassol (Cyprus), 2016.

1.5.4. Registros de derechos de autor

- **Juan Pradilla**; *"SOSLite"*; 2016; Registro: 13-52-22, Colombia, plataforma: múltiples, ambiente: servidor.

- **Juan Pradilla**; *"SOSFul"*; 2016; Registro: 13-52-163, Colombia, plataforma: múltiples, ambiente: servidor.

- **Juan Pradilla**; *"CoAP-SOSFul Proxy"*; 2016; Registro: 13-52-164, Colombia, plataforma: múltiples, ambiente: servidor.

1.5.5. Proyectos de investigación

Se han realizado colaboraciones en los siguientes proyectos de investigación, y algunos de los artículos publicados han sido financiados por los mismos:

- H2020 ICT-30-2015, "Interoperability of Heterogeneous IoT Platforms" (INTER-IoT), Enero 2016 -Diciembre 2018.

- Retos de la Sociedad (MINECO), "Sistema de Monitorización y Control Seguro de Residencias de la Tercera Edad" (SAFE-ECH), Septiembre 2015 - Agosto 2017.

- INNPACTO (MINECO), "Sistema de Transporte Logístico Inteligente Multimodal" (STIMULO), Enero 2013 - Diciembre 2015.

- INNPACTO (MINECO), "Cockpit para la monitorización y auditoria continua de la seguridad de las operaciones de infraestructuras de comunicación abiertas basadas en Perfiles de Aseguramiento de la Seguridad (SAC) TR 187 023" (UniverSEC), Septiembre 2012 - Agosto 2015.

1.6. Organización de la memoria

En este primer capítulo se han introducido los principales factores que motivan la realización de esta tesis, así como los objetivos que guiaron el desarrollo de las mismas y la metodología empleada para su realización exitosa. El resto de la memoria se desarrolla por medio de capítulos tal y como se indica a continuación.

En el capítulo 2 se presenta un estado del arte pormenorizado sobre el desarrollo de Internet y las perspectivas para el año 2020, el SWE, el SOS, la recomendaciones para SOS ligero del OGC, los modelos de datos *Sensor Model Language* (SensorML) y *Observations and Measurements* (O&M). De igual forma, se presenta una introducción a las SN, la IoT, los CPS, el FC y el CC.

Por su parte, en el capítulo 3 se detalla el desarrollo del SOSLite. Se presentan las principales decisiones de diseño, las herramientas utilizadas para la implementación, las adaptaciones del estándar SOS y de los modelos de datos SensorML y O&M. El capítulo finaliza con las pruebas del SOSLite desplegado como micro-instancia en la nube y dentro de un dispositivo de capacidades limitadas.

Después, en el capítulo 4 se presenta como se puede aprovechar el SOSLite dentro de las SN. Siendo este el punto de partida para abordar posteriormente otros CPS, presenta además la primera propuesta de caso de uso, de forma que se identifique la potencialidad del uso del SOSLite.

De igual forma, en el capítulo 5 se presenta el proceso de desarrollo del SOSFul, las decisiones de diseño que le dieron forma, las herramientas tecnológicas escogidas para su implementación, las adaptaciones del estándar SOS y los modelos de datos SensorML y O&M, realizadas con miras a disminuir el consumo de ancho de banda y facilitar la aceptación dentro de los desarrollos de la IoT.

Luego se da paso, en el capítulo 6, al uso del SOSFul en IoT. Profundizando en cómo este SOS ligero y actualizado puede ser aprovechado para como solución de interoperabilidad semántica en este CPS. Finalmente, se aborda un caso de uso centrado en las ciudades inteligentes, que muestra cómo se puede integrar el SOS dentro de la arquitectura de IoT y en las tecnologías ya existentes.

Finalmente, el capítulo 7 comprende las conclusiones principales extraídas en la elaboración de la tesis, así como unas líneas futuras de posible investigación.

2. Estado del Arte

2.1. Resumen

Son muchas las tecnologías que han emergido o se han afianzado en los últimos años, con miras a transformar la interacción humana; bien sea, entre personas, entre personas y su entorno o entre personas y las representaciones virtuales de datos e información, que se manipulan mediante las tecnologías de la información y la comunicación (TIC). Y es común encontrar que en este amplio conjunto de tecnologías, las mismas comiencen a confluir dando lugar a tendencias y evoluciones que integran conceptos anteriores, posibilidades actuales y perspectivas de futuro. Así, nacen y se consolidan los tres temas centrales para esta tesis: los CPS, el CC y el FC. Además, se comienzan a presentar los problemas de interoperabilidad; la motivación principal para iniciar este trabajo de tesis.

Sobre estos cuatro ejes: CPS, CC, FC e interoperabilidad se ha construido este capítulo con el objetivo de establecer los conceptos abordados en los capítulos posteriores. Exponiendo en primer lugar, el panorama tecnológico actual y su influencia en el desarrollo de los CPS; acto seguido, abordando los CPS y profundizando en las SN y en la IoT como dos de los casos de mayor relevancia en la actualidad; a continuación, esclareciendo los paradigmas de CC y FC y su integración con los CPS; posteriormente, presentando la definición de interoperabilidad que se empleará en esta tesis; seguidamente, profundizando en el marco de trabajo SWE y en el SOS; para dar paso, a las herramientas tecnologías empleadas.

Así, este capítulo introduce los conceptos básicos y profundiza en los aspectos más relevantes, para obtener un entendimiento cabal del marco conceptual sobre el cual se construye el presente trabajo de tesis; además, sirve de referencia para varios de los temas que se presentan en capítulos posteriores y que son de interés para el lector.

2.2. Panorama tecnológico

El panorama tecnológico, que abarca el desarrollo pasado y las perspectivas de futuro, es fundamental para comprender la importancia actual de los CPS, el CC y el FC, y brinda un marco temporal para apoyar la justificación del desarrollo del proyecto de tesis.

2.2.1. Desarrollo de las TIC

Para entender como se ha alcanzado el estado actual de los CPS y del CC/FC, es necesario dar una mirada al pasado y observar dos aspectos primordiales en el desarrollo de las TIC actuales: las tendencias seguidas por los insumos tecnológicos y la penetración de las redes de telecomunicaciones de banda ancha con el advenimiento de Internet y la World Wide Web (WWW).

En cuanto a los insumos tecnológicos, en los últimos años se han vivido condiciones de crecimiento de capacidad [21][22], en las áreas de: procesamiento (Figura 5.a) [23][24], donde la cantidad de transistores por mm^2 ha crecido de forma exponencial; almacenamiento (Figura 5.b) [25], donde la capacidad de los discos individuales ha llegado a alcanzar las cifras de terabytes; y ancho de banda (Figura 5.c) [26], el cual muestra una tendencia creciente imparable desde el nacimiento de Internet.

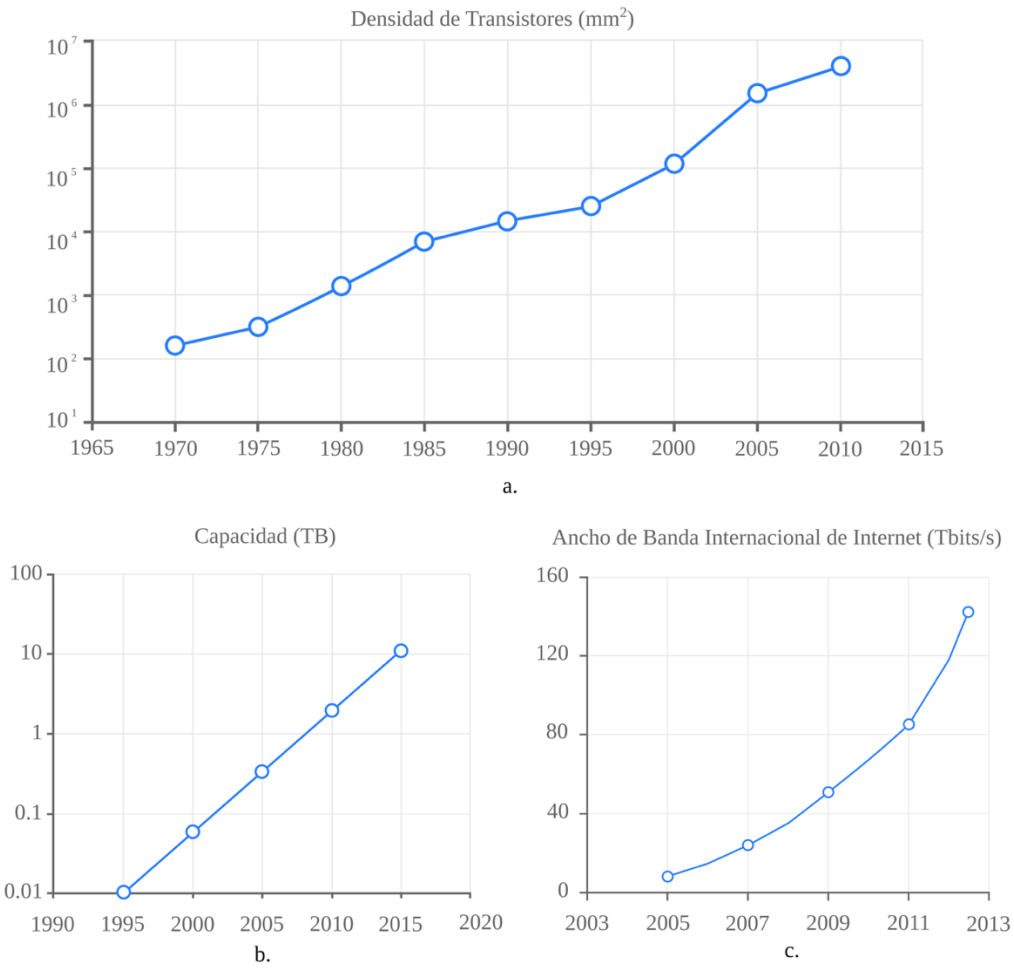
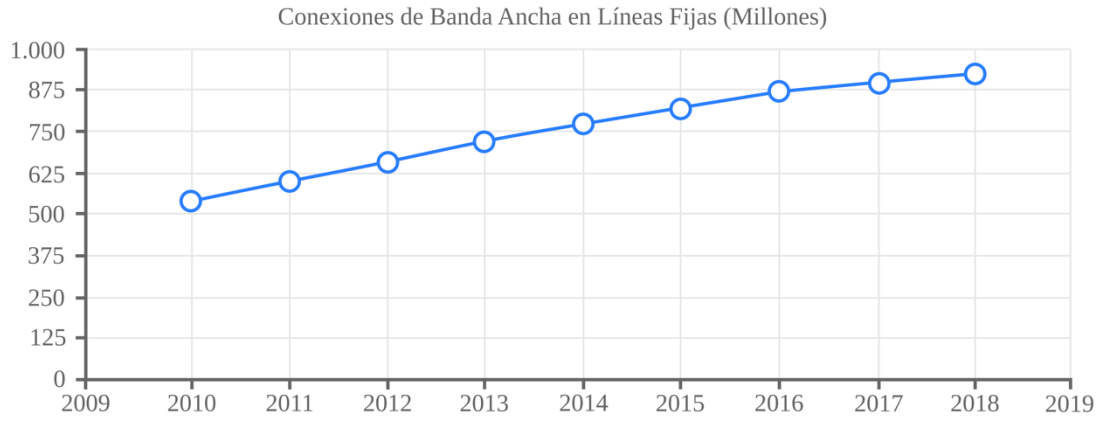
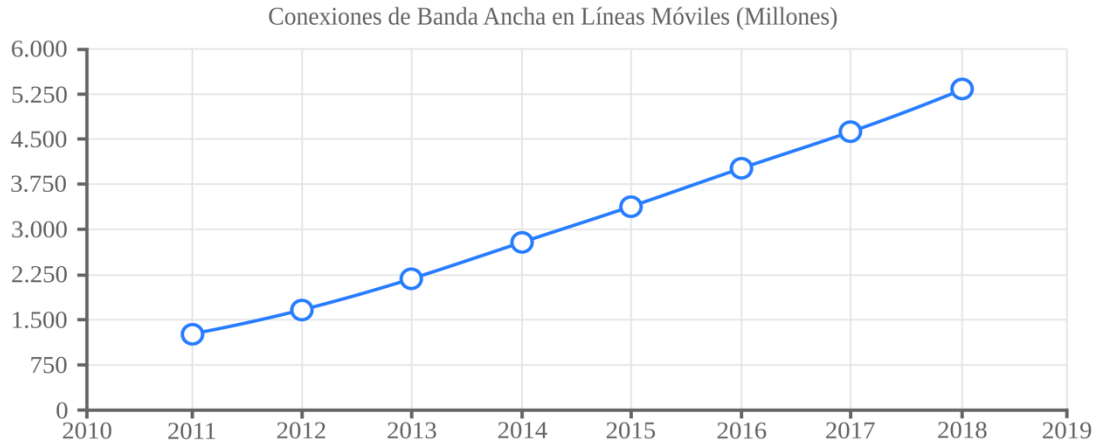


Figura 5 – Aspectos primordiales en el desarrollo de las TIC actuales
Insumos: (a) procesamiento, (b) almacenamiento y (c) ancho de banda

Por su parte, la penetración de las redes de telecomunicaciones de banda ancha mantiene una tendencia creciente en el número de líneas fijas (Figura 6.a) [27] y un ascenso acelerado en la líneas móviles (Figura 6.b) [27], las cuales se están convirtiendo en el método predilecto de acceso a Internet de muchas personas a nivel global.



a.



b.

Figura 6 - Aspectos primordiales en el desarrollo de las TIC actuales - Penetración de la banda ancha

2.2.2. Previsiones para las TIC en año 2020

Si se continua con las tendencias revisadas, se espera que para el 2020 existan entre 50.000 y 100.000 millones de dispositivos conectados [28], incluyendo: objetos inteligentes, sensores y actuadores. Así mismo, se espera que para ese mismo año entre el 53% y 60% de la población mundial, cerca de 4.500 millones, se conecten a Internet [29]; siendo los países asiáticos, africanos y sudamericanos los que más personas aporten a este incremento. De igual forma, se considera que los insumos tecnológicos, de los cuales se ha hablado, seguirán con las tendencias actuales, mejorando de forma considerable las capacidades de procesamiento, almacenamiento y ancho de banda en los siguientes cuatro años.

Como consecuencia de estos avances se prevé una interacción más transparente entre las

personas y los dispositivos, hasta llegar a una computación ubicua, al servicio de las necesidades personales y sociales de la humanidad. En esta perspectiva de futuro se vislumbran cambios en las formas de producción, transporte y venta de productos, tanto como en el análisis de información.

En cuanto a la producción, se puede destacar las revoluciones de la industria inteligente (*Smart Factory* o *Industry* 4.0) [30][31] y la agricultura inteligente (*Smart Agriculture* o *Precision Agriculture*) [32][33]; basadas en los CPS. En el área de transporte, la integración de los vehículos no tripulados (*Unmanned Aerial Vehicle* – UAV, *Unmanned Maritime System* - UMS y *Unmanned Ground Vehicle* - UGV) [34][35] y las carreteras inteligentes (*Smart Highway*)[36] basadas en las comunicaciones vehículo a vehículo (*Vehicle-to-vehicle* - V2V)[37] y vehículo a infraestructura (*Vehicle-to-infrastructure* - V2I)[37] son las tecnologías más prominentes.

Por su parte, en las ventas de productos, se esperan innovaciones de la mano de los sistemas de posicionamiento en interiores (*Indoor Positioning System* – IPS)[38][39] y los "beacons"; basados en comunicaciones de corto alcance y bajo consumo.

De forma análoga, en el análisis de información, los avances esperados más relevantes vienen de tres vertientes: el procesamiento en tiempo real (*Stream Processing*) [40] [41], el procesamiento de grandes cantidades de datos (*Big Data*) y la inteligencia artificial (*Artificial Intelligence* - AI) aplicada a los grandes datos, en especial las técnicas provenientes del aprendizaje profundo (*Deep Learning*) [42][43], las redes neuronales (Artificial Neural Networks) y la lógica difusa (*Fuzzy Logic*).

2.3. Sistemas Ciber-Físicos – Cyber-Physical Systems (CPS)

2.3.1. Definición de los CPS

Como se definió en la introducción (sección 1.1.2) los CPS son una mezcla de: computación, redes y procesos físicos; los cuales están compuestos por: un núcleo de computación/redes y una serie de sensores, actuadores y objetos inteligentes con los cuales interactúa con la realidad. Así, los CPS pueden generar una representación virtual de la realidad, la cual, es la base para realizar acciones sobre el entorno físico.

Los CPS se manifiestan desde la escala milimétrica (microrobots) hasta la kilométrica (smartgrid) y de forma independiente (marcapasos) o como parte de un CPS mayor (brazo robótico en una línea de montaje). Esto, conlleva a que existan una gran cantidad de CPS desplegados en la actualidad. Dentro de esta tesis se hará hincapié en dos de los CPS de mayor interés: SN y IoT.

2.3.2. Redes de sensores – Sensor Network (SN)

Las SN se componen de nodos sensores autónomos de bajo coste que se intercomunican a través de redes de comunicación, generalmente inalámbricas, para realizar tareas cooperativas orientadas a determinar el estado de un área de interés. Este estado, es una representación virtual de variables físicas que se han medido por medio de los sensores desplegados [44][45].

Un nodo sensor autónomo de bajo coste, en adelante sensor (Figura 1), dispone de recursos limitados de: procesamiento, almacenamiento, potencia de transmisión y potencia energética. Se caracterizan por un tamaño reducido y una composición de hardware y software orientada a las comunicaciones y a la eficiencia energética (normalmente las plataformas se gestionan según un paradigma orientado a eventos que supone que el dispositivo está en un estado de bajo consumo durante gran parte de su ciclo de vida).

Además de los sensores, las SN contienen una (o más) estación base, también denominadas sumideros (*sink*). Una estación base, suele ser un dispositivo que dispone de mayores recursos en comparación a los sensores y que cumple con la funcionalidad de ser el punto de destino de toda la información generada y retransmitida por los sensores. Una estación base además, proporciona funciones de pasarela para la información hacia otros dispositivos fuera de la red de sensores, como puede ser, un servidor centralizado; de igual forma, cuando facilita la interconexión de dos redes de comunicación diferentes, se le da el nombre de *Gateway*.

Las SN tiene un gran campo de acción, que incluye la adquisición de datos, monitorización y control de: aplicaciones médicas y cuidado de salud; aplicaciones para seguridad; vigilancia y seguimiento; aplicaciones para agricultura y ganadería; automoción; control de tráfico; monitorización ambiental; de estructuras y de fenómenos naturales.

2.3.3. Internet de las Cosas – Internet of Things (IoT)

La IoT es sin duda el CPS de mayor relevancia en la actualidad, debido a que está pasando de la etapa investigativa a la etapa de desarrollo comercial. Despertando el

interés económico y social, que crea sinergia entre la industria, el gobierno y la ciudadanía. Aprovechando este auge, en esta tesis se plantea profundizar en este sistema, y por tal motivo, se detalla en esta sección los aspectos fundamentales para introducirse en esta temática.

2.3.3.a. Alcance de la IoT

Tal como se mencionó en la sección 1.1.2, el concepto de IoT aún se encuentra en desarrollo, siendo la definición del IERC la empleada en la realización de esta tesis. Esta definición se recalca nuevamente en esta sección aunque ha sido presentada con anterioridad.

Según el IERC la IoT es "una red global de infraestructura dinámica con capacidades de autoconfiguración, basada en estándares y protocolos de comunicación interoperables, donde las 'cosas' físicas y virtuales tienen identidad, atributos físicos y personalidad virtual; y que usa interfaces inteligentes y perfectamente integradas en la red de información" [46].

Otra definición interesante de IoT, para esta tesis, ha sido descrita por *Gartner*, la cual, define a la IoT como: "una red de objetos físicos que contienen tecnologías para comunicarse y sensar/interactuar con los estados internos o externos del entorno" [47].

Además, para obtener una comprensión más específica sobre el concepto de IoT se puede referir al artículo original de Kevin Ashton, creador del concepto de IoT, donde la describió como la posibilidad de "otorgarle a los dispositivos la capacidad de recopilar información con sus propios medios, para que puedan ver, oír y oler el mundo por sí mismos, en todo su azaroso esplendor. Las tecnologías de RFID y sensores permiten a los dispositivos observar, identificar y comprender el mundo, sin las limitaciones de los datos introducidos por los humanos" [48].

Así, la IoT ofrece una gran oportunidad de mercado para los desarrolladores, los proveedores de servicios en Internet y los creadores de hardware. Se espera que cerca de 212 mil millones de dispositivos se desplieguen por todo el mundo para el 2020 [49] y que el 45% del tráfico en Internet corresponda a los flujos de IoT para principios de la próxima década [50][49][51]. Este crecimiento tendrá un impacto económico esperado de entre 2,7 a 6,2 billones de dólares para el 2025 [52].

Por su parte, el alcance de la IoT como uno de los CPS más complejos, se puede apreciar al revisar el mapa de sectores de IoT de Beecham Research [53] (Figura 7). En este mapa se observa claramente como la IoT hace parte de los sectores de: transporte, construcción y energía, entre otros.

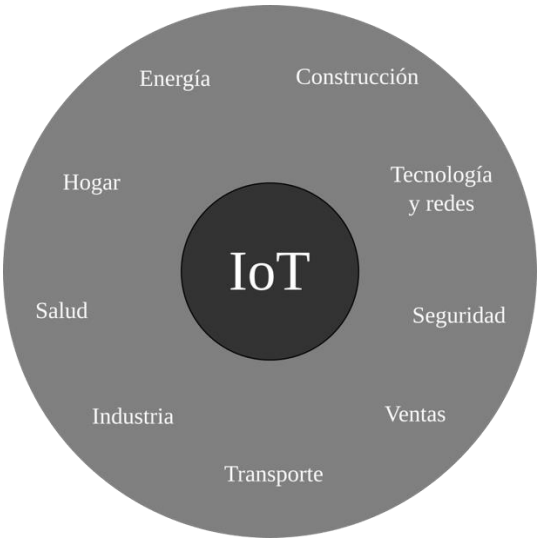


Figura 7 - Mapa de sectores de IoT

2.3.3.b. Arquitecturas de IoT

Para dar cabida a la gran cantidad de dispositivos esperados y alcanzar el potencial económico, la IoT debe estructurase siguiendo una arquitectura flexible y escalable. Son muchas las propuesta que han surgido en torno a este reto a lo largo de los últimos años; consolidándose las basadas en una estructura de múltiples capas. Como ejemplo, se pueden observar las cuatro arquitecturas presentadas (Figura 8), siendo la más básica la arquitectura de tres capas: percepción, red y aplicación [54] (Figura 8.a); las otras arquitecturas están compuestas por cinco capas: basada en *middleware*[55] (Figura 8.b), basada en servicios [56] (Figura 8.c) y basada en el negocio [57][58] (Figura 8.d).

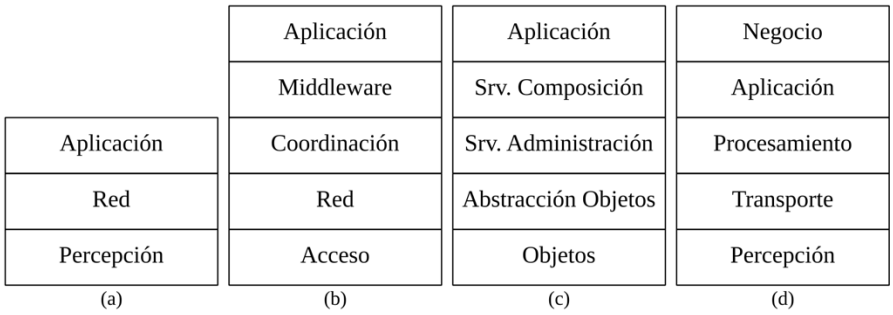


Figura 8 - Arquitecturas de IoT

2.3.3.c. Componentes de IoT

Existen seis componentes principales de IoT: identificación, percepción/actuación, comunicación, computación, servicios y semántica (Tabla V). Estos componentes son fundamentales para la construcción de la IoT y se encuentran en continuo desarrollo; están constituidos por una integración de hardware, software y comunicaciones, que se expresan mediante estándares y dispositivos.

Tabla V - Componentes de la IoT

Componente	Descripción
Identificación	Cada "cosa" debe poder ser identificada dentro de la IoT, permitiendo su distinción de otras "cosas" y asociando con esta los servicios a los que tiene acceso
Percepción/Actuación	Permite la interacción con la realidad física, bien sea al realizar mediciones sobre esta (mediante sensores) o al producir una acción física (mediante actuadores)
Comunicación	Parte central de la IoT es lograr que las "cosas" se comuniquen, para esto existen diversas tecnologías como RFID, Zigbee, WiFi, etc. que se encargan de transmitir información empleando la menor cantidad de energía
Computación	Constituido por las unidades de procesamiento (microprocesadores, FPGA, microcontroladores, etc.) y por el software necesario para operarlos; son el eje principal del procesado de información dentro de la IoT
Servicios	Los servicios de IoT, se puedo clasificar en cuatro categorías: 1. Identificación: permite diferenciar entre las "cosas" vinculadas a la IoT 2. Agregación de información: recolecta y resumen los recolectados por los sensores 3. Colaboración: obtiene datos para tomar decisiones y reaccionar acorde con estas 4. Ubicuos: hace que los servicios estén disponibles en cualquier momento, en cualquier lugar y para cualquiera
Semántica	Es la habilidad de extraer conocimiento a partir de los datos y proveer servicios en concordancia.

La siguiente tabla (Tabla VI) expone algunos de los ejemplos más emblemáticos de cada uno de los componentes de la IoT.

Tabla VI - Ejemplos de componentes de IoT

Componente		Ejemplo
Identificación	Nombramiento	EPC, uCode
	Direccionamiento	IPV4, IPv6
Percepción/Actuación		Sensores, actuadores, Tags RFID
Comunicación		RFID, NFC, Bluetooth, Zigbee, WiFi, LTE-A
Computación	Hardware	Arduino, Raspberry Pi, SmartPhones
	Software	Sistemas operativos (COntiki, TinyOS, Android); Cloud (Hadoop, Nimbits)
Servicios		Identificación, Agregación de información, Colaboración y Ubicuos
Semántica		RDF, OWL, EXI

2.3.3.d. Protocolos de IoT

En la actualidad existen muchos protocolos estándares que buscan facilitar la integración de la IoT dentro de los proveedores de servicios y los desarrolladores de software. Varias de las agrupaciones tecnológicas más importantes, incluyendo: IEEE (Institute of Electrical and Electronics Engineers), W3C (World Wide Web Consortium) e IETF (Internet Engineering Task Force); han realizado sus contribuciones. La siguiente tabla (Tabla VII) agrupa los protocolos más relevantes [59].

Tabla VII - Protocolos relevantes de IoT

Nivel		Protocolo					
Aplicación		DDS	CoAP	AMQP	MQTT	XMPP	HTTP/REST
Descubrimiento		mDNS			DNS-SD		
Infraestructura	Enrutamiento	RPL					
	Red	6LoWPAN			IPv4/IPv6		
	Enlace	IEEE 802.15.4					
	Físico	LTE-A		IEEE 802.15.4		Z-Wave	
Influyentes		IEEE 1888.3, IPSec			IEEE 1905.1		

2.4. Cloud & Fog Computing

2.4.1. Computación en la nube - Cloud Computing (CC)

CC es un paradigma para permitir el acceso compartido a un conjunto configurable de recursos computacionales y de red, el cual cuenta con cinco características que son esenciales para su funcionamiento [60]:

1. Auto-servicio bajo demanda: un consumidor puede obtener mayores capacidades computacionales, como: mayor procesamiento o almacenamiento, sin necesidad de la intervención humana.

2. Amplio acceso desde la red: los servicios y capacidades del CC están disponibles para el acceso desde dispositivos heterogéneos (teléfonos inteligentes, ordenadores, etc.).

3. Recursos compartidos: los recursos computacionales e informáticos son puestos a disposición de los clientes mediante un modelo de múltiples usuarios, con recursos físicos y virtuales asignados dinámicamente y reasignados según la demanda.

4. Rápida adaptabilidad: los recursos pueden ser provistos y liberados de forma rápida, y en muchas ocasiones de automáticamente, para escalar horizontal o verticalmente según la demanda de los clientes.

5. Servicio controlado: en CC los recursos son monitorizados, controlados, reportados y optimizados, gracias a la capacidad de medición con que cuenta el sistema.

Dentro del CC se encuentran de forma asidua tres modelos de servicio (Figura 9) [61]:

1. *Software as a Service* (SaaS): provee al consumidor aplicaciones que se ejecutan sobre la infraestructura del *Cloud*. Las aplicaciones pueden ser accedidas por varios dispositivos usando interfaces de cliente ligeras o una interfaz de programación. Los consumidores no administran o controlan la infraestructura de la nube y se limitan a la configuración de las capacidades necesarias para el buen funcionamiento de la aplicación.

2. *Platform as a Service* (PaaS): proporciona la capacidad de desplegar, sobre la infraestructura del proveedor, aplicaciones creadas o adquiridas, que hacen uso de lenguajes de programación, liberarías, servicios y herramientas soportadas en el *Cloud*. Los consumidores no pueden administrar o controlar la infraestructura pero si se encargan del despliegue y la gestión de las aplicaciones y sus configuraciones.

3. *Infrastructure as a Service* (IaaS): habilita el uso de recursos computacionales como: almacenamiento, procesamiento y red; donde el consumidor puede desplegar cualquier software que desee. Aunque, el consumidor no tiene el control o la administración de la infraestructura si puede determinar el sistema operativo, las aplicaciones y de igual forma realizar alguna configuración limitada a los dispositivos de red.

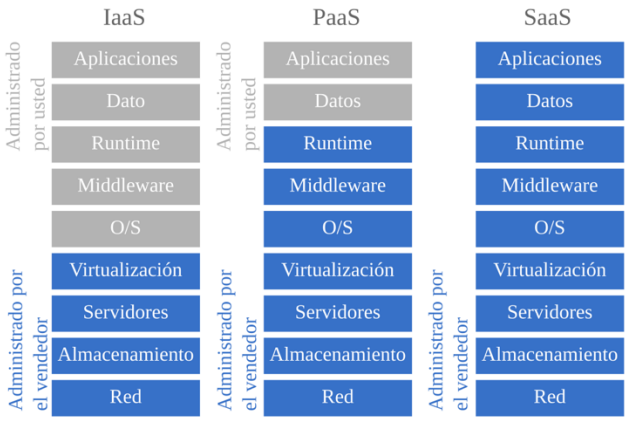


Figura 9 - Modelos de Cloud Computing

2.4.2. Computación en la niebla - Fog Computing (FC)

Cuando los recursos de computacionales y de red del CC se acerca a los productores/consumidores de información, sean seres humanos o sensores/actuadores se comienza a hablar de FG (Figura 10). Este acercamiento permite que el sistema ofrezca: latencia ultra baja, alto ancho de banda, acceso en tiempo real y conocimiento del contexto [62].

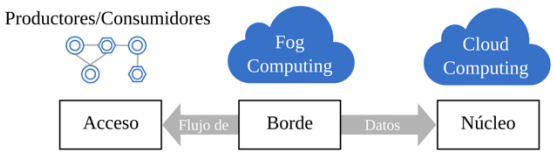


Figura 10 - Fog Computing

2.4.3. Máquinas virtuales – Virtual Machine (VM)

Las máquinas virtuales (VM) son una importante herramienta que se ha venido utilizando junto con el CC y el FC; y que ha permitido una gran adaptabilidad y seguridad a los sistemas que en estas se despliegan.

Una VM simula los recursos de hardware de un dispositivo, delimitando de esta forma, los recursos de hardware real a los cuales se tendrá acceso desde la VM. Esto es importante porque permite que varias VM compartan un mismo dispositivo real y además posibilita que esa limitación auto-impuesta pueda ser modificada dinámicamente, siempre que no sobrepase las capacidades reales del hardware sobre el que se despliega [63].

Así, al simular los recursos de hardware, una VM puede ser tratada como una maquina real cuyo hardware puede diferir del que realmente cuenta en el lugar donde se despliega. Esto trae como ventaja que la VM puede ser transportada y desplegada en

diferentes dispositivos, manteniendo su integridad y su funcionalidad.

De igual forma, una VM delimita un dominio de seguridad, el cual está aislado del resto de VMs que se ejecutan en un mismo dispositivo, como si se tratasen de equipos totalmente independientes. Este hecho unido a la independencia de hardware hace que las VMs sean ideales para los casos que requieren copias de seguridad, movilidad, rápido despliegue de infraestructura y alta tolerancia a fallos.

Dentro de las VM se instala un sistema operativo que interactúa con el hardware simulado permitiendo su uso por parte de servicios y aplicaciones. Es necesario especificar que las VMs; se despliegan sobre un ecosistema que incluye: el VM manager o hypervisor, el hardware real y el sistema operativo del dispositivo real (Figura 11) [64].

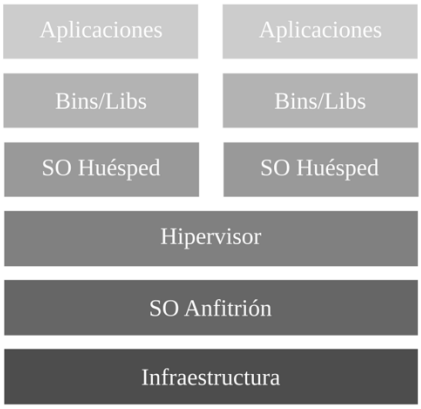


Figura 11 - Máquinas Virtuales

2.4.4. Contenedores Linux y Docker

Los contenedores Linux (LXC) son una aproximación diferente a la virtualización con VM, mediante la creación de un entorno virtual con su propio espacio de recursos computacionales asignado. Los LXC proveen las mismas ventajas que las VM pero son mucho más livianos al no requerir de todo un sistema operativo invitado para su funcionamiento (Figura 12) [65][66].

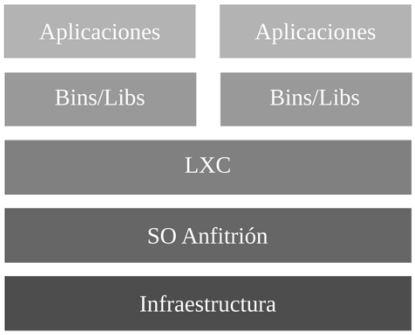


Figura 12 - Contenedores Linux

Docker es un proyecto, que haciendo uso de los LXC, automatiza el despliegue de aplicaciones; brindando a estas, la posibilidad de tener una visión privada del sistema

operativo, al tiempo que brinda una distribución liviana.

2.5. Marco de trabajo Sensor Web Enablement (SWE)

2.5.1. Marco de trabajo para SN

SWE es un marco de trabajo compuesto por modelos de información y servicios estandarizados, que permite integrar conjuntos de sensores heterogéneos, y la información que estos generan, a través de interfaces basados en estándares Web como SOAP y XML. Tanto los servicios como los modelos de información facilitan la gestión de sensores, permitiendo la fusión de múltiples modelos y formatos de datos, bajo un modelo común de datos y unas interfaces de comunicación estandarizadas.

La arquitectura SWE de OGC consta de dos bloques principales (Figura 13): el modelo de información y el modelo de servicio. El primero, agrupa los modelos conceptuales y las codificaciones; y el segundo, contiene la especificación de los servicios.

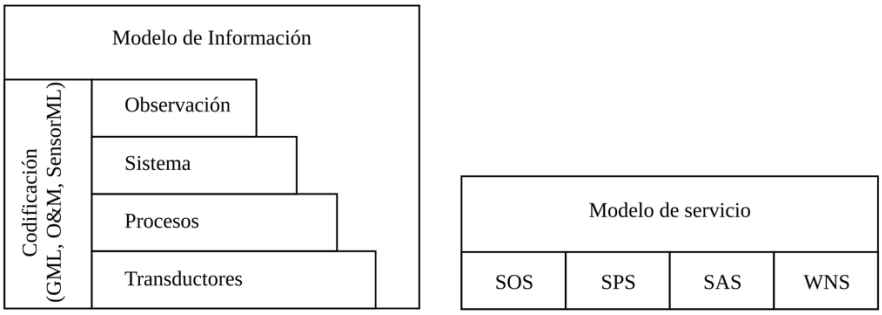


Figura 13 - Bloques del SWE

2.5.2. Modelo de información

Dentro del modelo de información del SWE existe una agregación de información por capas, aumentando el grado de abstracción e interoperabilidad con cada una de ellas; partiendo desde los transductores, en la capa inferior, hasta alcanzar las observaciones, en la capa superior. Las aplicaciones que hacen uso del SWE tienen acceso a toda la información de las capas, así que, para lograr el mayor grado de interoperabilidad, flexibilidad y profundidad de la información, es aconsejable el uso de las capas superiores. La siguiente tabla agrega los diferentes componentes del modelo de información.

Tabla VIII - Componentes del modelo de información

Componente	Descripción
Transductores	Los transductores encargados de convertir fenómenos físicos en datos digitales se denominan comúnmente sensores, mientras que los transductores que convierten datos digitales en fenómenos físicos, tradicionalmente son llamados actuadores. Tanto los sensores como los actuadores representan la interfaz entre el mundo real y el mundo digital, por lo que forman los elementos básicos de una red de sensores.
Procesos	Dentro del marco SWE se permite el almacenamiento de la información y el consumo/procesado de la misma. Así, un proceso es algoritmo que genera una o más salidas a partir de una o más entradas.
Sistema	Es una agrupación de transductores y procesos, para los cuales se ha definido una posición relativa a un sistema de coordenadas interno definido. Al relacionar el sistema de coordenadas interno con sistema geográfico de referencia, se puede georeferenciadas el sistema, sus componentes y sus mediciones producidas.
Observación	Es cada una de las mediciones realizadas por los sensores sobre los fenómenos físicos. Cada observación registra: el valor de la medición, el tiempo en el cual se tomó la medición y el fenómeno físico observado, entre otros datos.

Los elementos comunes sobre los cuales se fundamenta el modelo de información del SWE y cada uno de sus componentes, se expresan como sub-lenguajes de XML, siendo los más relevantes:

- *Transducer Markup Language* (TML) [67]: aunque este estándar ha entrado en desuso, durante muchos años fue uno de los pilares sobre los cuales se construyó el SWE. TML proporciona información de cuando y donde se produce una acción sobre un transductor o un sistema; detallando los tipos específicos de componentes involucrados, las relaciones virtuales y físicas entre estos, además de los datos producidos y consumidos por los mismos. El estándar define un modelo a partir del cual se puede enviar, archivar, agregar y analizar de manera eficiente los datos de los transductores y los sistemas siguiendo un formato común. Siendo su uso principal, el de enviar información en vivo, mediante un flujo de datos, desde los sensores hasta los clientes.

- *Sensor Model Language* (SensorML)[68]: este lenguaje describe un modelo de información común para la descripción de sensores y sistemas de sensores lo que facilita el descubrimiento de sensores, así como el análisis y procesado de datos de sensores. Tras el SensorML se encuentra la idea de que los sensores pueden ser modelados como procesos; los cuales definen entradas y salidas, el sistema de referencia empleado, parámetros disponibles y el método de un proceso. Adicionalmente, se pueden introducir metadatos en un proceso. Estos metadatos contienen, información general empleada en la identificación y clasificación de un proceso, como también, información acerca de las propiedades y capacidades del mismo.

- *Observation & Measurements* (O&M) [69]: es el modelo de información para representar las mediciones realizadas por los sensores, también llamadas observaciones. Una observación es un evento que tiene lugar en un momento determinado y que está asociada a una medición de un valor sobre un fenómeno físico específico. Además del momento temporal y el valor de la medida, O&M es capaz de describir otras propiedades de la medición, por ejemplo, el proceso empleado en generar la medida así como la ubicación y la calidad de la medida. O&M considera que el valor medido es una aproximación de un atributo de una propiedad de interés (FoI, *Feature of Interest*) observada. Se puede incluir información adicional como metadatos para un mayor análisis e interpretación de los datos.

2.5.3. Modelo de servicio

El modelo de servicio describe los servicios del marco de trabajo SWE; estos servicios permiten, a las aplicaciones, acceder de forma interoperable a la información generada y almacenada, así como realizar operaciones sobre dicha información. Los servicios incluidos en la propuesta del OGC son:

- *Sensor Observation Service* (SOS) [17]: proporciona operaciones de almacenamiento estándar, sobre los datos y los meta-datos de sensores y observaciones. El SOS utiliza la especificación O&M para modelar observaciones de sensores, así como con la especificación SensorML para

modelar sensores y sistemas sensores (metadatos de sensores). El SOS permite organizar un cierto número de observaciones en lo que se denomina "*Observation Offering*", que consiste en un grupo de observaciones relacionadas que no se solapan ni espacial, ni temporalmente. Las implementaciones de SOS son independientes del dominio de usuario y de aplicaciones específicas; facilitando su uso desde cualquier cliente SWE compatible, sin necesidad de disponer de un conocimiento previo. La especificación dispone las diferentes operaciones SOS en varios perfiles (Tabla IX): perfil básico (*core profile*); perfil transaccional (*transactional profile*); perfil mejorado (*enhanced profile*) y perfil para la gestión de resultados (*result handling profile*). El soporte del perfil básico es obligatorio para cualquier implementación SOS, mientras que el resto de perfiles son opcionales.

Tabla IX - Operaciones SOS

Extensión	Operación	Descripción
Core	GetCapabilities	Pide la auto-descripción del servicio
	GetObservation	Obtiene los datos medidos por un sensor en formato O&M
	DescribeSensor	Adquiere los meta-datos de un sensor en formato SensorML
Enhanced	GetFeatureOfInterest	Obtiene un *feature* en formato GML
	GetObservaitonById	Pide los datos medidos por un sensor en formato O&M por un identificador especifico
Transactional	InsertSensor	Agrega un nuevo sensor
	UpdateSensorDescription	Actualiza la descripción de un sensor
	DeleteSensor	Elimina un sensor
	InsertObservation	Agrega los datos medidos por un sensor
Result Handling	InsertResultTemplate	Inserta una plantilla que describe la estructura y la codificación que se utilizara para retornar los datos en la operación GetResult
	InsertResult	Carga los datos en bruto según una estructura y una codificación definido en la operación InsertResultTemplate
	GetResultTemplate	Adquiere la estructura para los resultados y la codificación para un conjunto de parámetros
	GetResult	Obtiene los datos en bruto para un conjunto de parámetros específicos

- *Sensor Alert Service* (SAS) [70]: es un sistema de notificación por eventos basado el patrón de publicador/subscriptor; en el cual, un productor puede anunciar nuevos tipos de eventos al servicio; a los cuales el consumidor se puede suscribir; de esta forma, el consumidor será notificado automáticamente cada vez que tenga lugar un evento al que previamente se suscribió. Es posible identificar varios tipos de eventos: la simple generación de una medida puede ser considerado como un evento, así como la superación de un valor umbral definido por el usuario o un mensaje de estado procedente de un sensor (por ejemplo, el estado de la batería).

- *Sensor Planning Service* (SPS) [71]: este servicio ofrece una interfaz estándar para el control de los sensores. Por ello la interfaz contiene operaciones como:

acceso a metadatos del servicio, obtener los parámetros de tareas de un sensor, inspeccionar la viabilidad de una tarea, enviar una tarea, modificar una tarea, cancelar una tarea y obtener el estado actual de una tarea. Dado que el tiempo requerido en completar una tarea se desconoce a priori, el SPS puede usar el servicio WNS para comunicarse con el cliente de una forma asíncrona. El almacenamiento de los datos recogidos queda fuera del ámbito del SPS.

- *Web Notification Service* (WNS) [72]: este servicio proporciona una interfaz para una comunicación asíncrona con un usuario o un servicio web. En general, los servicios web soportan una comunicación síncrona con el cliente. Si asumimos que enviamos una tarea al SPS que requiere un cierto tiempo para su ejecución (por ejemplo, obteniendo imágenes de satélite), entonces debemos disponer de una forma de saber cuándo se completó dicha tarea. En este caso el servicio WNS se puede usar. WNS soporta la transmisión de notificaciones a través de varios protocolos de transporte. Los mensajes se pueden enviar a través de HTTP, mensajería instantánea (XMPP), e-mail, SMS, fax e incluso teléfono. La especificación distingue dos patrones de comunicación: notificación unidireccional y bidireccional.

- *Catalogue Service Web* (CSW) [73]: es estándar define un modelo abstracto de servicio para la gestión y búsqueda de metadatos, basado en: *OGC Common Catalogue Query Language, General Catalogue Interface Model*. El primero define un lenguaje mínimo y abstracto de consulta de metadatos; mientras que el segundo determina las interfaces para la gestión de los catálogos, la búsqueda de metadatos, la gestión de sesiones y la mediación con metadatos que no pueden ser accedidos de forma directa. CSW permite una búsqueda distribuida sobre varias instancias. Los catálogos son un componente esencial de la infraestructura orientada a servicio de OGC permitiendo la búsqueda espacial y temporal de sensores y observaciones.

2.5.4. Implementaciones del SOS

Entre las implementaciones del SOS, la más utilizada y completa en *52 North SOS* [74] creada por *52 North*. Este organismo fue fundado por el Instituto de Geoinformática de la Universidad de Münster, y tiene una amplia comunidad que mantiene y actualiza las distintas versiones del SOS. Esta implementación brinda todos los perfiles del estándar y está creada sobre el lenguaje de programación JAVA.

Existen otras implementaciones como son PySOS, desarrollado por la comunidad de investigación oceánica, y la única implementado en Python; MapServer SOS [75], de la popular web de aplicaciones espaciales abiertas; Deegree SOS [76], que forma parte de la versión de desarrollo del marco de servicios web OGC deegree conocido como deegree3; etc. Pero todas estas implementaciones ofrecen funcionalidades muy limitadas y hace tiempo que dejaron de ofrecer soporte.

2.5.5. Recomendaciones para perfil de SOS Ligero

Motivados por esta situación, el OGC ha decidido dar un paso adelante y realizar una recomendación para SOS Ligero, recogida en el documento "*OGC® Best Practice for Sensor Web Enablement Lightweight SOS Profile for Stationary In-Situ Sensors*" [77]. Esta recomendación simplifica las operaciones *core* del SOS de forma que se adapte a

escenarios de sensores estacionarios; al tiempo que, acorta los lenguajes para descripción de sensores (SensorML) y observaciones (O&M) haciéndolos más sencillos de procesar; encontrando un equilibrio entre eficiencia, facilidad de implementación y cumplimiento del estándar.

La recomendación de SOS ligero se enfoca en tres aspectos: reducir el número de operaciones, centrándose solo en las operaciones *core*; reducir la complejidad de las operaciones, por ejemplo limitando el tipo de filtros; y enfocándose en los sensores estacionarios, los cuales se ha comprobado son los que mayormente se utilizan en las implementaciones reales de SOS

2.6. Tecnologías aplicadas

2.6.1. Uniform Resource Identifier (URI)

Una URI es una cadena de caracteres que identifica un recurso de forma unívoca en toda la red; las mismas están compuestas por:

- Esquema: hace referencia a una especificación para asignar los identificadores y comúnmente identifica el protocolo de acceso al recursos, ejemplo: http:, coap:, ftp:, etc.

- Autoridad (opcional), contiene los siguiente:

    - Dos barras (//).
    - Una sección opcional con el nombre de usuario y la contraseña.
    - El nombre del host como el nombre de dominio o la dirección IP.
    - Número de puerto (opcional).

- Ruta: información sobre la localización del recurso expresado, generalmente, como una jerarquía.

- Consulta (opcional): expresa, generalmente en clave-valor, información de búsqueda asociada con el recurso.

- Fragmento (opcional): indica una parte del recurso o una representación especifica del mismo.

Así, la estructura de una URI es:

esquema:[//[usuario:contraseñ@]dominio[:puerto]][/]ruta[?consulta][#fragmento]

Y ejemplos de ella pueden ser:

- http://soslite.ue/test/capabilities
- coap://soslite.ue/test/observations?procedure=85

2.6.2. Representational State Transfer (REST)

REST es un estilo arquitectura de software para sistemas hipermedia distribuidos, el cual fue creado por uno de los principales autores de la especificación HTTP, Roy Thomas Fielding, en el año 2000 como parte de su tesis doctoral en *Information and Computer Science* de la *University of California, Irvine* [78].

Aunque REST es independiente del protocolo y por lo tanto no está acoplado a HTTP, muchos servicios que siguen la arquitectura REST se encuentran desplegados sobre WWW y por ende utilizan HTTP como protocolo de transporte, esto no es una coincidencia dado que algunos verbos HTTP: GET, POST, PUT y DELETE; se corresponden directamente con las operaciones REST.

2.6.3. JavaScript Object Notation (JSON)

JSON es un formato de intercambio de datos, nacido como un subconjunto de la notación de objetos del lenguaje de programación JavaScript y que representa información estructurada mediante texto ligero.

En la actualidad es un formato empleado extensivamente en las tecnologías asociadas a la Web y se ha hecho a un lugar como opción al uso de XML, debido a su mayor simplicidad la cual conlleva a un menor uso de ancho de banda sin realizar grandes sacrificios a nivel sintáctico.

3. SOSLite

3.1. Resumen

El objetivo de la tesis es definir, diseñar, implementar y probar un mecanismo de interoperabilidad. Como se estableció en el capítulo de introducción, el desarrollo del mecanismo de interoperabilidad propuesto en esta tesis, parte de una solución implementada y probada en las SN, el SOS, la cual se modifica y actualiza, de forma que, se adapte a los requerimientos de los CPS actuales.

Partir de una solución probada y aceptada como el SOS, tiene como ventaja el lograr una rápida transición entre la propuesta teórica y la implementación real. Además, se puede contrastar los resultados obtenidos con los existentes. Sin embargo, también presenta el desafío de ajustar una tecnología conocida, sin perder sus bondades en el camino.

Teniendo en cuenta estos factores, se han analizado los estándares presentados por las OGC, centrándose en los estándares: SOS [17], SensorML [68], O&M [69]; los cuales describen el servicio de gestión de datos y metadatos de los sensores y sus mediciones (conocidas como observaciones), y los modelos de datos para sensores y las observaciones.

Además, se revisó como la OGC, después de años de implementaciones exitosas, observó que en muchos casos de uso estos estándares sobrepasan las necesidades reales y por ende definió unas buenas prácticas para el desarrollo de un perfil de SOS ligero [79]. Estas buenas practicas, simplifican los estándares SOS, SensorML y O&M para que se adapten a entornos con menor disposición de recursos computacionales.

Así, a partir de los tres estándares y la recomendación, todos emitidos por la OGC, se concretó una definición de SOS ligero, como un mecanismo idóneo para la interoperabilidad en los CPS (Figura 14). Partiendo de su adaptación a las SN y observado si extensión a la IoT.

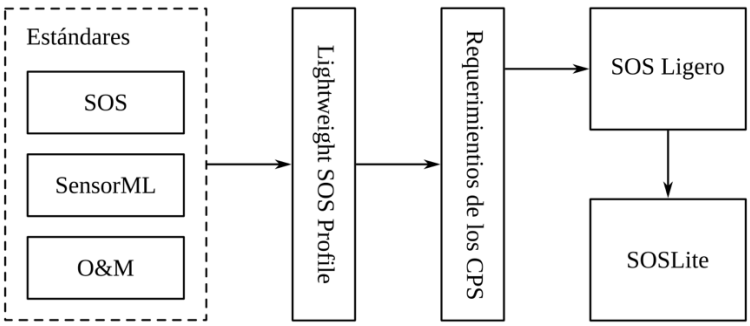


Figura 14 - Proceso de definición del SOS ligero

Como definición teórica, presenta la ventaja de ser compatible con los estándares, estar basada en recomendaciones respaldados por la industria y de poderse desplegar sobre dispositivos con capacidades computacionales reducidas. Esto hace que la propuesta de SOS ligero, atienda de forma eficiente a los casos de uso basados en computación distribuida y brinda la posibilidad de integrarlo con CC/FC.

Sin embargo, se deben comprobar estas consecuencias lógicas esperadas y para esto es necesario la realización de una implementación que se ajuste a la definición de SOS

ligero; es decir, que cumpla con el estándar SOS y que a su vez pueda ser desplegado sobre un dispositivo de capacidades limitadas o como una micro-instancia en la nube.

A ese programa se lo ha denominado SOSLite y en este capítulo se describirá su desarrollo, siguiendo una metodología de desarrollo iterativa e incremental (Figura 4). Este capítulo entonces, presenta el análisis, el diseño, la implementación, las pruebas y la evaluación del SOSLite; de forma que se comprendan como las decisiones tomadas en el diseño a partir del análisis realizado, impactan en el rendimiento del servicio y lo hacen más adecuado para su uso en los CPS.

Pero antes de entrar en materia, es necesario clarificar las características de un dispositivo de capacidades limitadas y de una micro-instancia en la nube. Es razonable la inexistencia de un consenso sobre la definición de dispositivo limitado, si se tiene claro que, para cada caso específico la misma pueda variar; por ejemplo: un dispositivo limitado cuya funcionalidad sea la de actuar como un servidor para pequeñas empresas tendrá 8 GB de memoria RAM, 500 GB de disco duro y un procesador de 4 núcleos; mientras que, un dispositivo limitado para un nodo sensor tendrá 10 kB de memoria RAM, 100 kB de memoria Flash y utilizara un microcontrolador (nodo limitado clase 1) [80].

Dado que el SOSLite ha sido creado para ser parte de las redes distribuidas y en especial para dar soporte a CPS y como parte representativos de estos sobre SN e IoT; se espera que el dispositivo donde se ejecute, tenga mayores recursos que los nodos sensores/actuadores, y que además, tenga alimentación y conectividad permanente. Un buen ejemplo de un dispositivo limitado que cumple con estas características es Raspberry Pi 1 Modelo B+, que cuenta con una memoria RAM de 512 MB, almacenamiento mediante memoria MicroSD (4 GB es una capacidad que se suele utilizar), un procesador ARM a 700 MHz y un consumo energético de 3.0 W.

De forma análoga, se puede pensar en las micro-instancias en la nube, en este caso se tomara como referencia el tipo de instancia t2.micro de Amazon EC2, la cual cuenta con: 1 GB de memoria RAM, 30 GB de disco duro y 1 vCPU (equivalente al 20% de uso de un procesador de 8 núcleos a 2 GHz). Como se puede observar, las características de estas micro-instancias superan a las del dispositivo limitado de referencia, con lo que se tomara como restricciones las capacidades de la Raspberry Pi 1 Modelo B+.

Por otra parte, las SN pueden tener diferentes arquitecturas donde sus componentes: nodos sensores, el dispositivo de puerta de enlace (estaciones base o sumideros) y las aplicaciones externas; pueden publicar, consultar, modificar o eliminar información. Esta amalgama de posibilidades hace que el análisis que se presenta en este capítulo pueda resultar excesivamente complejo, por lo que se ha optado por agrupar funcionalidades específicas en dos actores diferentes, así: un productor de datos será el encargado de publicar, modificar y eliminar la información; mientras que, el consumidor de datos estará encargado únicamente de consultar la información.

3.2. Análisis

El análisis del SOSLite parte de la definición teórica de SOS ligero, que es producto de pasar los estándares SOS, SensorML y O&M; bajo el filtro de la recomendación de SOS ligero de la OGC y de los requerimientos de los CPS actuales (Figura 14). Con esto se obtiene un SOS ligero que toma las operaciones presentadas en la recomendación pero que aúna las operaciones del perfil *core* y del *transactional*; haciendo del SOSLite una implementación de SOS ligero, que puede ser desplegada en dispositivos limitados y que permite agregar y consultar la información de sensores y observaciones; hecho que lo convierte en un mecanismo idóneo para brindar interoperabilidad a nivel semántico entre equipos, redes y tecnologías heterogéneas dentro de las SN.

Desde este punto de partida, se formaliza el análisis mediante el uso de la ingeniería de software, concretamente mediante el uso de la definición de requerimientos funcionales (RF) y no funcionales (RNF), la definición de la arquitectura que se empleará y la determinación del modelo de conceptos a utilizar.

3.2.1. Requerimientos funcionales (RF) y no funcionales (RNF)

Para determinar las funcionalidades y características a las que debe responder el SOSLite se ha optado por utilizar un proceso de gestión de requerimientos de software. Para ilustrar el proceso, a continuación, se detallan las suposiciones (Tabla X), los actores (Tabla XI) y los RF (descritos mediante casos de uso en fase enfocados [81]) y RNF [82] que debe cumplir el programa, siguiendo algunos apartes del estándar IEEE 830 [83].

En primer lugar, las suposiciones (Tabla X) indican aquellos atributos vinculados con el despliegue del SOSLite y que estarán disponibles de forma asegurada al momento de llevar el sistema a un caso de uso real. Estas suposiciones son relevantes debido a que las pruebas se basan en un entorno mínimo, que es acorde con los despliegues reales que se esperan en el futuro. Se ha de recalcar que se ha optado por estas características debido a su representatividad, pero esto no deja de lado la posibilidad de desplegarse en otros dispositivos.

Tabla X - Suposiciones de desarrollo del SOSLite

Nombre	Descripción
Dispositivo	El SOSLite se ejecutara en dispositivo o instancia virtual que cuenta con al menos las características de una Raspberry Pi 1 Modelo B+: 512 MB RAM, almacenamiento 4 GB en, procesador ARM a 700 MHz
Conectividad	El SOSLite cuenta con conectividad con un ancho de banda mínimo de 10 Mbps. El mismo puede ser accedido de forma continua desde los nodos sensores y los programas de procesamiento de datos
Alimentación	El dispositivo sobre el cual se despliegue el programa cuenta con alimentación eléctrica continua y estable
Sincronización	El sistema cuenta con un mecanismo de sincronización que permite la programación de los envíos de información desde los sensores

En segundo lugar, se detalla el listado de actores (Tabla XI) que interactúa con el sistema. Poniendo en manifiesto, desde una primera instancia, la nomenclatura utilizada en todo el trabajo de tesis y de forma inmediata en el desarrollo de los requerimientos funcionales. Estos actores constituyen modelos conceptuales que quedaran definidos dentro de la implementación (como la base de datos) o que adquirirán un rol específico

en el caso de uso que se desplieguen (un proveedor de datos puede ser una estación meteorológica o una habitación, por ejemplo).

Tabla XI - Listado de actores del SOSLite

ID	Nombre	Información
A01	Proveedor de datos	Cada uno de los dispositivos o aplicaciones que pueden registran información en el sistema. Ejemplo: nodo sensor, simuladores de SN, etc.
A02	Consumidor de datos	Cada una de los dispositivos o aplicaciones que consultan los datos del sistema para procesarlos. Ejemplo: procesador de eventos complejos, procesador de grandes datos, etc.
A03	Base de datos	Base de datos para almacenar la información y meta-información sobre sensores y observaciones
A04	Sistema de ficheros	Sistema de ficheros para el almacenamiento de los datos de los sensores y las observaciones en sus modelos de datos correspondientes

En tercer lugar, se presenta el diagrama de casos de uso (Figura 15). Este diagrama presenta una visión estática del SOSLite y permite conocer los requerimientos funcionales y como estos se relacionan con los actores que se han descrito con anterioridad. Este diagrama sigue las directrices del *Unified Modeling Language* (UML).

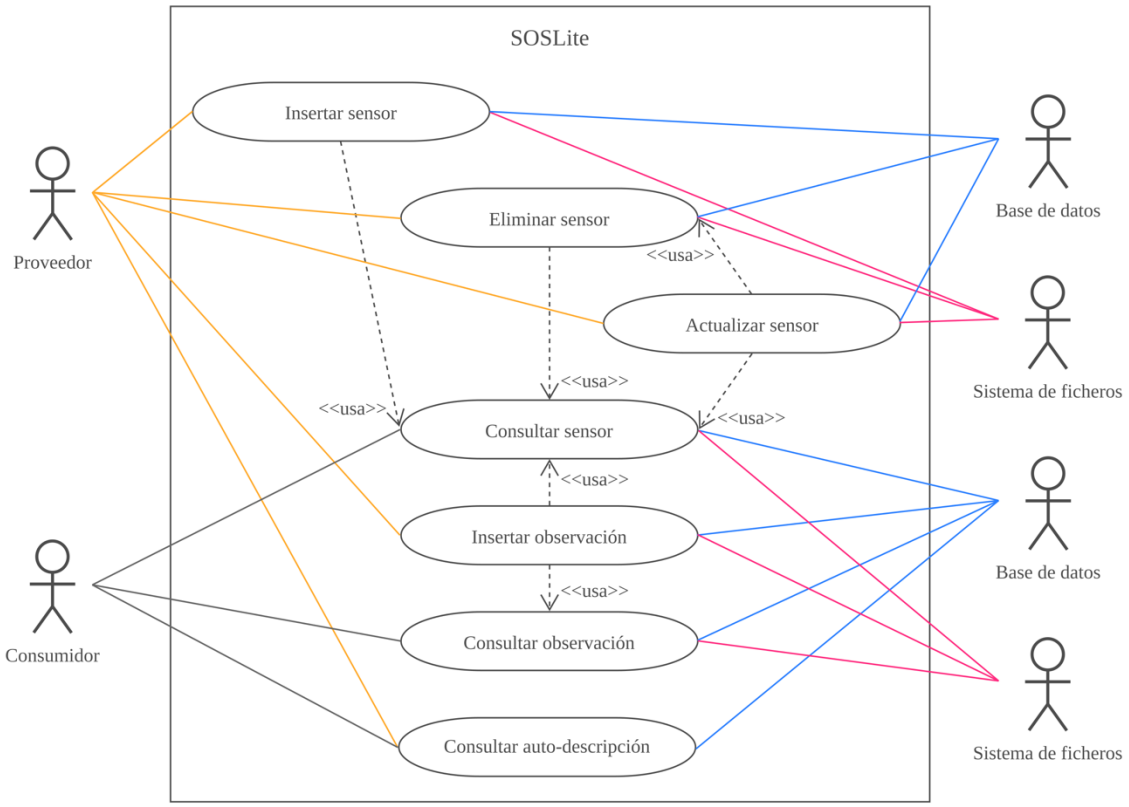


Figura 15 - Diagrama de casos de uso del SOSLite

En cuarto lugar, se listan los requerimientos funcionales (Tabla X) para tener una referencia directa sobre las prestaciones esperadas dentro del SOSLite y poder determinar cuáles son los más relevantes al momento de realizar las pruebas.

Tabla XII - Resumen de los requerimientos funcionales

Identificación	Nombre	Descripción
RF01	Consultar la auto-descripción del sistema	Consulta la auto-descripción del SOS desplegado
RF02	Insertar sensor	Agrega un sensor al sistema
RF03	Consultar sensor	Consulta la información de un sensor
RF04	Actualizar un sensor	Actualiza los metadatos y los datos de un sensor
RF05	Eliminar un sensor	Elimina los metadatos y los datos del sensor
RF06	Insertar observación	Agrega una observación al sistema
RF07	Consultar observación	Consulta la información de los sensores que cumplen con las condiciones y los filtros pasados

La obtención de los RF nace de una revisión documental del estándar SOS y la recomendación para el perfil de SOS ligero de la OGC; buscando obtener las operaciones necesarias para manipular los sensores y sus observaciones. Así, se analizaron las operaciones del perfil de SOS ligero y se extendieron a las operaciones del perfil *transactional*, de forma que, además de consultar se pudiera insertar y eliminar.

La descripción de los RF, detallados como casos de uso en fase enfocados se lista a continuación. Mediante estas descripciones se conoce de primera mano las funcionalidades que brindan el SOSLite y los actores que están vinculados con cada una de las estas.

Tabla XIII – RF del SOSLite: Consultar la auto-descripción del sistema

Identificación	**RF01**	
Nombre	Consultar la auto-descripción del sistema	
Fase	Enfocados	
Descripción	Consulta la auto-descripción del SOS desplegado, incluyendo: la identificación del servicio, los filtros soportados y los metadatos de la información almacenada. Soporta filtros de sección: *ServiceIdentification, ServiceProvider, FilterCapabilities, InsertionCapabilities, Contents*.	
Actores	Principales: (A01) Proveedor de datos, (A02) Consumidor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita la auto auto-descripción del sistema	
		2. Procesa los datos de la petición y determina la información que se enviara como respuesta según los filtros pasados en la petición. Si no se encuentra ningún filtro se envía toda la descripción
		3. Consulta la base de datos
		4. Construye la respuesta
		5. Envía la respuesta
Excepciones	N/A	
Casos relacionados	N/A	

Tabla XIV - RF del SOSLite: Insertar sensor

Identificación	**RF02**
Nombre	Insertar sensor
Fase	Enfocados
Descripción	Agrega un sensor al sistema, almacenando los metadatos en la base de datos y

	los datos en formato SensorML dentro del sistema de ficheros	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos, (A04) Sistema de ficheros	
Guion	**Actor**	**Sistema**
	1. Envía los datos del sensor en formato SensorML	
		2. Procesa los datos de la petición
		3. Consulta si el sensor existe (ver caso RF03)
		4. Almacena los metadatos del sensor en la base de datos
		5. Almacena los datos del sensor en formato SensroML dentro del sistema de ficheros
		6. Construye la respuesta
		7. Envía la respuesta
Excepciones	*El sensor ya existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	RF03	

Tabla XV - RF del SOSLite: Consultar sensor

Identificación	**RF03**	
Nombre	Consultar sensor	
Fase	Enfocados	
Descripción	Consulta la información del sensor cuyo identificador es pasado dentro de la petición	
Actores	Principales: (A02) Procesador de datos Secundarios: (A03) Base de datos, (A04) Sistema de ficheros	
Guion	**Actor**	**Sistema**
	1. Solicita los datos de un sensor cuyo identificador es pasado dentro de la petición	
		2. Procesa los datos de la petición y obtiene el identificador del sensor
		3. Consulta los metadatos del sensor en la base de datos
		4. Obtiene los datos del sensor en formado SensorML desde el sistema de ficheros
		5. Construye la respuesta
		6. Envía la respuesta
Excepciones	*El sensor no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	N/A	

Tabla XVI - RF del SOSLite: Actualizar un sensor

Identificación	**RF04**	
Nombre	Actualizar un sensor	
Fase	Enfocados	
Descripción	Actualiza los metadatos y los datos del sensor pasado en la petición	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos, (A04) Sistema de ficheros	
Guion	**Actor**	**Sistema**

	1. Envía los datos del sensor en formato SensorML	
		2. Procesa los datos de la petición
		3. Consulta si el sensor existe (ver caso RF03)
		4. Elimina los datos del sensor (ver caso RF05)
		5. Inserta los datos del sensor (ver caso RF02)
		6. Construye la respuesta
		7. Envía la respuesta
Excepciones	*El sensor no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	RF02, RF03, RF05	

Tabla XVII - RF del SOSLite: Eliminar un sensor

Identificación	**RF05**	
Nombre	Eliminar un sensor	
Fase	Enfocados	
Descripción	Elimina los metadatos y los datos del sensor con el identificador pasado en la petición	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos, (A04) Sistema de ficheros	
Guion	**Actor**	**Sistema**
	1. Solicita la eliminación de los datos de un sensor cuyo identificador es pasado dentro de la petición	
		2. Procesa los datos de la petición y obtiene el identificador del sensor
		3. Consulta si el sensor existe (ver caso RF03)
		4. Elimina los datos del sensor del sistema de ficheros
		5. Elimina los metadatos del sensor del sistema de ficheros
		6. Construye la respuesta
		7. Envía la respuesta
Excepciones	*El sensor no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	RF03	

Tabla XVIII - RF del SOSLite: Insertar observación

Identificación	**RF06**	
Nombre	Insertar observación	
Fase	Enfocados	
Descripción	Agrega una observación al sistema, almacenando los metadatos en la base de datos y los datos en formato O&M dentro del sistema de ficheros	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos, (A04) Sistema de ficheros	
Guion	**Actor**	**Sistema**
	1. Envía los datos de la observación en formato O&M	
		2. Procesa los datos de la petición

		3. Consulta si la observación existe (ver caso RF07)
		4. Almacena los metadatos de la observación en la base de datos
		5. Almacena los datos de la observación en formato O&M dentro del sistema de ficheros
		6. Construye la respuesta
		7. Envía la respuesta
Excepciones	*La observación ya existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	RF07	

Tabla XIX - RF del SOSLite: Consultar observación

Identificación	**RF07**	
Nombre	Consultar observación	
Fase	Enfocados	
Descripción	Consulta la información de los sensores que cumplen con las condiciones y los filtros pasados en la petición. Condiciones soportadas: *offering, procedure, observedProperty y featureOfInterest*. Filtros soportados: *temporalFilter* (de tipo *Equal y During*) y *spatialFilter* (de tipo *Bounding box*)	
Actores	Principales: (A02) Consumidor de datos Secundarios: (A03) Base de datos, (A04) Sistema de ficheros	
Guion	**Actor**	**Sistema**
	1. Solicita los datos de un sensor cuyo identificador es pasado dentro de la petición	
		2. Procesa los datos de la petición y determina la información que se enviara como respuesta según las condiciones y los filtros pasados en la petición.
		3. Consulta los metadatos de las observaciones en la base de datos
		4. Obtiene los datos de las observaciones en formado O&M desde el sistema de ficheros
		5. Construye la respuesta
		6. Envía la respuesta
Excepciones	*El sensor no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	N/A	

Finalmente, se presentan los RNF siguiendo las indicaciones de especificación de requisitos de software consignadas en el estándar IEEE 830 [83]. Estos requerimientos son direccionadores arquitecturales, es decir, marcan pautas importantes a tener en cuenta al momento de decantarse por una arquitectura de software específica; de igual forma, las tácticas arquitecturales empleadas buscan satisfacer estos RNF.

Tabla XX - RNF del SOSLite: Dispositivo

Identificación	**RNF01**
Nombre	Dispositivo
Descripción	Avalar el correcto funcionamiento al menos en un dispositivo con características equivalentes a una Raspberry Pi 1 Modelo B+: 512 MB RAM, almacenamiento 4 GB en, procesador ARM a 700 MHz y consumo energético de 3.0 W

Tabla XXI - RNF del SOSLite: Rendimiento

Identificación	**RNF02**
Nombre	Rendimiento
Descripción	Garantizar un mínimo de 2 operaciones por segundo

Tabla XXII - RNF del SOSLite: Portabilidad

Identificación	**RNF03**
Nombre	Portabilidad
Descripción	Asegurar el correcto funcionamiento en cualquier distribución GNU/Linux

Tabla XXIII - RNF del SOSLite: Cumplimiento de estándares/recomendaciones

Identificación	**RNF04**
Nombre	Cumplimiento de estándares/recomendaciones
Descripción	Establecer el cumplimiento del estándar SOS 2.0 y la recomendación para SOS ligero y los modelos de datos SensorML y O&M de la OGC

Tabla XXIV - RNF del SOSLite: Modificabilidad

Identificación	**RNF05**
Nombre	Modificabilidad
Descripción	Garantizar que el sistema puede ser modificado fácilmente para ampliar sus características o adaptarse a nuevas versiones de los estándares relacionados

3.2.2. Arquitectura

La arquitectura del sistema que conforman los nodos sensores, las aplicaciones externas y el SOSLite viene determinada por el estándar SOS 2.0 (RNF04) y el marco de trabajo SWE; siendo la misma, una arquitectura orientada a servicios (*Service Oriented Architecture* - SOA) [84]. En esta, el SOSLite es un servicio que ofrece flujos de información simétricos a los nodos sensores y las aplicaciones externas (Figura 16).

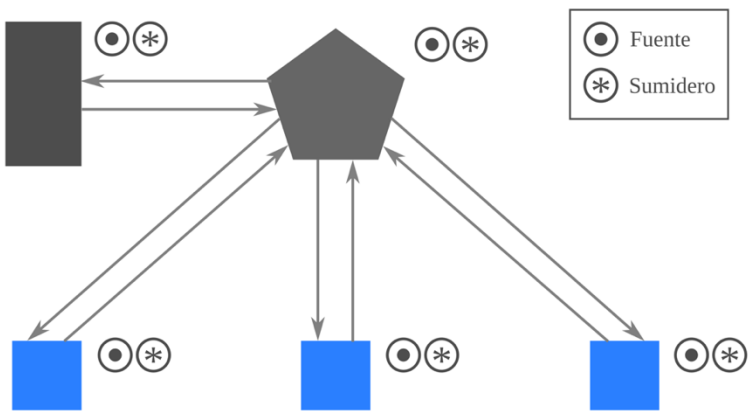


Figura 16 - Diagrama de flujos de información para el SOSLite

Por su parte, para la realización de la implementación se ha seguido un patrón arquitectural de capas favoreciendo así los requisitos de portabilidad (RNF03) y modificabilidad (RNF05), en detrimento del requisito de rendimiento (RNF02); esto, debido a que las capas permiten una alta cohesión y un bajo acoplamiento, pero aumentan la cantidad de código en ejecución de forma simultanea lo que conlleva a un mayor uso del procesador y de la memoria.

Así, la arquitectura del SOSLite cuenta con tres capas (Figura 17): (i) abstracción de datos, (ii) servicios de datos y (iii) servicios. La capa de abstracción de datos facilita el acceso transparente a la base de datos y al sistema de ficheros; por su parte, los servicios de datos se encargan de adaptar los datos transmitidos desde y hacia el sistema; finalmente, la capa de servicios ofrece el punto de entrada del SOSLite, incluyendo la descripción del servicio y las operaciones del estándar.

Figura 17 - Arquitectura de software para el SOSLite

Para dar cumplimiento a todos los requisitos no funcionales, la arquitectura será refinada mediante tácticas arquitecturales [85], específicamente se utilizara la reducción del *overhead* computacional y la introducción de concurrencia en el procesamiento de las operaciones. Para reducir el *overhead* computacional se hace uso del perfil de SOS ligero [79], ampliándolo para abarcar las operaciones de la extensión *transactional*, disminuyendo así la complejidad de procesamiento (RNF01, RNF02). Por su parte, para introducir concurrencia en el procesamiento de las operaciones se ha utilizado un servidor orientado a eventos para el despliegue del servicio, con lo que se ha asegurado unas altas prestaciones con un bajo consumo de recursos (RNF01, RNF02) [86].

3.2.3. Diagrama de conceptos

El SOSLite se centra en la gestión de datos y metadatos de sensores y observaciones (que son las mediciones que realiza un sensor en un determinado momento); sin embargo, existen otros conceptos asociados que la implementación debe gestionar, a fin de cumplir con el estándar; de forma que, los conceptos sobre los cuales se construye este SOS ligero son (Figura 18): *Procedure*, *PhysicalSystem*, *Observation*, *ObservationOffering*, *ObservedProperty* y *FeatureOfInterest*.

Tabla XXV - Conceptos del SOSLite

Conceptos	Descripción
Procedure	En las especificaciones del SOS resalta el hecho de que no existe un concepto *Sensor*, en su lugar, y por compatibilidad con el resto de los estándares del SWE, se utiliza la definición de *Procedure*. Dentro del SWE, un *Procedure* puede ser: un método, un algoritmo, un instrumento, un sensor o un sistema, el

	cual tiene la capacidad de realizar observaciones
PhysicalSystem	La recomendación de SOS ligero no utiliza el concepto genérico de *Procedure* para modelar los sensores, en su lugar utiliza una especialización de este, denominada *PhysicalSystem*. Un *PhysicalSystem* es un sistema de sensores estacionarios. Utilizando este concepto, un SOS ligero limita el concepto de *Procedure* logrando así una reducción de la complejidad en el procesamiento de los mismos. Se debe destacar que un *PhysicalSystem* modela un conjunto de sensores; determinando así, que no hay una equivalencia directa entre el concepto *Sensor* y el *PhysicalSystem*. Un ejemplo esclarecedor de la distinción de estos dos conceptos es una estación meteorológica, que se corresponde con un *PhysicalSystem*, y cada uno de sus sensores (anemómetro, pluviómetro, termómetro, etc.), que caben dentro del concepto *Sensor*
Observation	Las observaciones encuentran una representación directa en el concepto *Observation*. Dentro del SOSLite una observación siempre debe tener asociados: el *PhysicalSystem* que la genero, la *ObservedProperty* que está midiendo y el *FeatureOfInterest* sobre el que se hacen las mediciones
ObservationOffering	Muchas veces denominado *offering*, el cual es un conjunto de observaciones que provienen de un mismo *PhysicalSystem*
ObservedProperty	Identifica el fenómeno que está siendo observado
FeatureOfInterest	Referencia, en el SOS 2.0, el objeto del mundo real al que pertenece la observación. Sin embargo, debido a las restricciones del perfil de SOS ligero, el *FeatureOfInterest* únicamente pueden ser de tipo *SamplingPoint*, el cual suele indicar el emplazamiento del *PhysicalSystem*

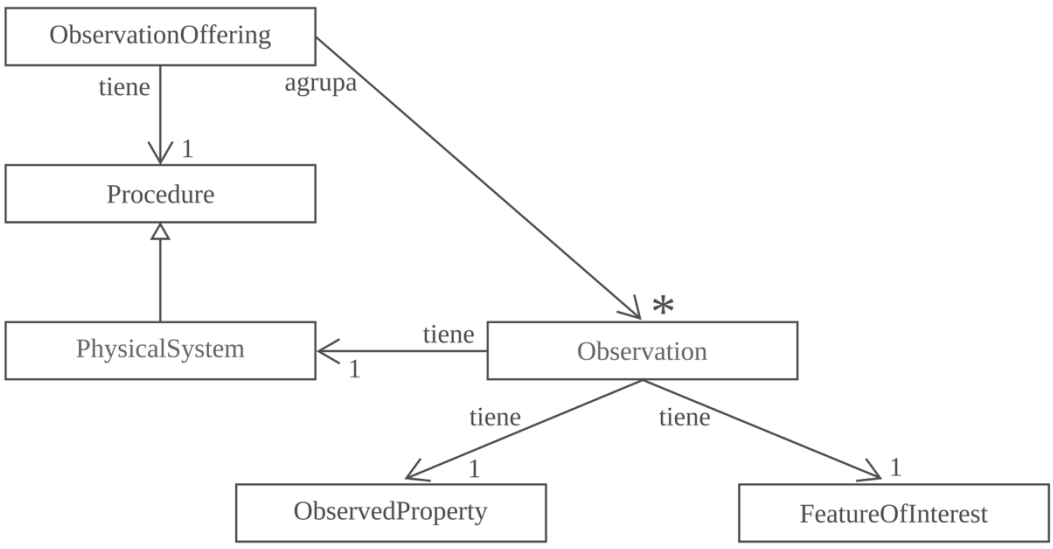


Figura 18 – Diagrama de conceptos del SOSLite

3.3. Diseño

Una vez se ha realizado el análisis que determina el marco en el cual el mecanismo de interoperabilidad basado en el estándar SOS será desarrollado, se pasa a diseñar la forma en la cual se llevara a cabo la implementación. De forma que, se obtiene un mayor detalle de las partes que componen la implementación y la interacción de las mismas. Para esto se proyecta una visión estática del sistema, donde se definen las partes, y una visión dinámica, donde se definen las interacciones entre las partes. Dando lugar a así al diagrama de clases, el modelo de datos y los diagramas de secuencia.

3.3.1. Clases del sistema

El diseño del SOSLite parte de la escogencia del paradigma de programación orientada a objetos [87] como paradigma de desarrollo; debido a la facilidad de integración con la arquitectura orientada a servicios y la arquitectura de capas, que según el análisis, son los más idóneos para la desarrollo de la aplicación; además, de las ventajas de alta cohesión y bajo acoplamiento que proporciona. De igual forma, para dar respuesta a las necesidades específicas de la implementación se hace uso de varios patrones de diseño de software [88], como son: *Facade*, *Data transfer object - DTO* y *Data Access Object – DAO*.

En el siguiente diagrama de clases (Figura 19) se detalla las directrices de programación que sigue el SOSLite. Para facilitar la identificación de los patrones de diseño utilizados, se asigna un sufijo a los nombres de las clases que los implementan. Además, el diagrama se complementa con la identificación de la capa a la que pertenecen las clases.

Como se puede observar, el punto de entrada al servicio es la clase *SOSLiteServer* la cual mapea el servicio SOS y utiliza las clases *Facade* para ofrecer las diferentes operaciones. Cada una de las tres clases *Facade*, se encargan de reducir la complejidad y minimizar las dependencias, mediante la simplificación y abstracción de la las clases y relaciones que intervienen en cada operación; de forma que, los objetos de la clase *SOSLiteServer* solo necesitan llamar a estas interfaces, sin la necesidad de conocer el funcionamiento interno de cada operación.

De igual forma, en el diagrama se puede apreciar la existencia de las clases *DTO*. Estas clases se utilizan para modelar los datos que circulan por el sistema y que serán almacenados en la base de datos. Por si mismos, no son una representación exacta de los atributos y la estructura que se encuentra en los modelos SensorML y O&M, para los sensores y observaciones respectivamente; si no, un compendio de atributos de estos modelos, sumados a algunos de las operaciones del SOS; los cuales son útiles al momento de realizar el almacenamiento y recuperación de los datos.

Por su parte, las peticiones llegan al sistema encapsuladas dentro de objetos de la clase *Message*, los cuales contienen la representación XML de las operaciones SOS y los modelos de datos. Estos mensajes deben ser transformados en dos clases distintas: un *File* y la respectiva clase *DTO* según sea una observación o un sensor. Con este fin, se utilizan las clases *Adapter* las cuales, toman un objeto *Message* y lo separan en un *DTO* con la información anteriormente descrita y en un *File* que contienen el modelo O&M o

SensorML que será almacenado en el sistema de ficheros. El respectivo objeto de la clase *DTO* mantiene una referencia al objeto *File* que se ha generado, permitiendo que este último sea accesible y utilizable mediante el objeto *DTO*.

Finalmente, se distinguen las clases *DAO*, encargadas de abstraer las operaciones de inserción, actualización, consulta y eliminación, en la base de datos y el sistema de ficheros; de forma que, el resto del sistema no dependa de la tecnología de motor de base de datos o del sistema de ficheros empleados.

3.3.2. Modelo de datos

En el caso de la gestión de los datos, se le ha dado prioridad a las operaciones de inserción, consulta y eliminación sobre las de actualización. Esto es acorde al funcionamiento esperado de un SOS ligero, en el cual, en una primera instancia se insertan los sensores, los cuales variaran muy poco en el tiempo, para posteriormente realizar grandes cantidades de inserciones de observaciones y consultas de las mismas, de forma sostenida a lo largo del funcionamiento de la aplicación.

Dado que los sensores se insertan y consultan en el modelo de datos que emplea SensorML, y a su vez, las observaciones hacen lo propio mediante O&M; se considera que cada uno de los registros que se realizan son inmutables; es decir, que cuando se agrega un sensor o una observación estos permanecen inalterables y, en el caso de los sensores, cualquier modificación implica la eliminación del registro anterior y la creación de uno nuevo; mientras que en el caso de las observaciones solo se pueden insertar y consultar.

Estas consideraciones aportan una mejora importante en rendimiento al momento de insertar y consultar datos y metadatos en el SOS; dado que las operaciones sobre la memoria se limitan a la lectura y escritura; que son mucho más rápidas que las actualizaciones. Así, al momento de realizar una operación de *InsertSensor* o *InsertObservation*, en la base de datos se almacena la información relevante para realizar las operaciones de consulta; mientras que los modelos se almacenan completos en el sistema de ficheros; de esta forma, cuando se realizan las operaciones de *DescribeSensor* o *GetObservation*, sobre la base de datos se aplican los filtros, pero la respuesta se construye tomando los modelos directamente desde el sistema de ficheros.

Por su parte, al analizar el estándar SOS 2.0, las peticiones/respuestas para cada una de sus operaciones y las necesidades del SOSLite, se puede llegar a obtener un modelo desnormalizado; es decir, que no se ha llevado a una forma normal, y poco relacional que se ajuste a la información que se almacenarán en la base de datos. Este modelo no es una representación puntual del diagrama de conceptos (Figura 18), en cambio, es una adaptación del mismo que responde a los requerimientos de dispositivo (RNF01) y de rendimiento (RNF02).

Para el SOSLite, el modelo de datos (Figura 20) consta de dos colecciones de documentos, la primera destinada a almacenar los datos de los sensores y la segunda para los datos de las observaciones. La estructura de los dos modelos guarda una referencia al fichero en la cual se ha guardado la representación O&M o SensorML.

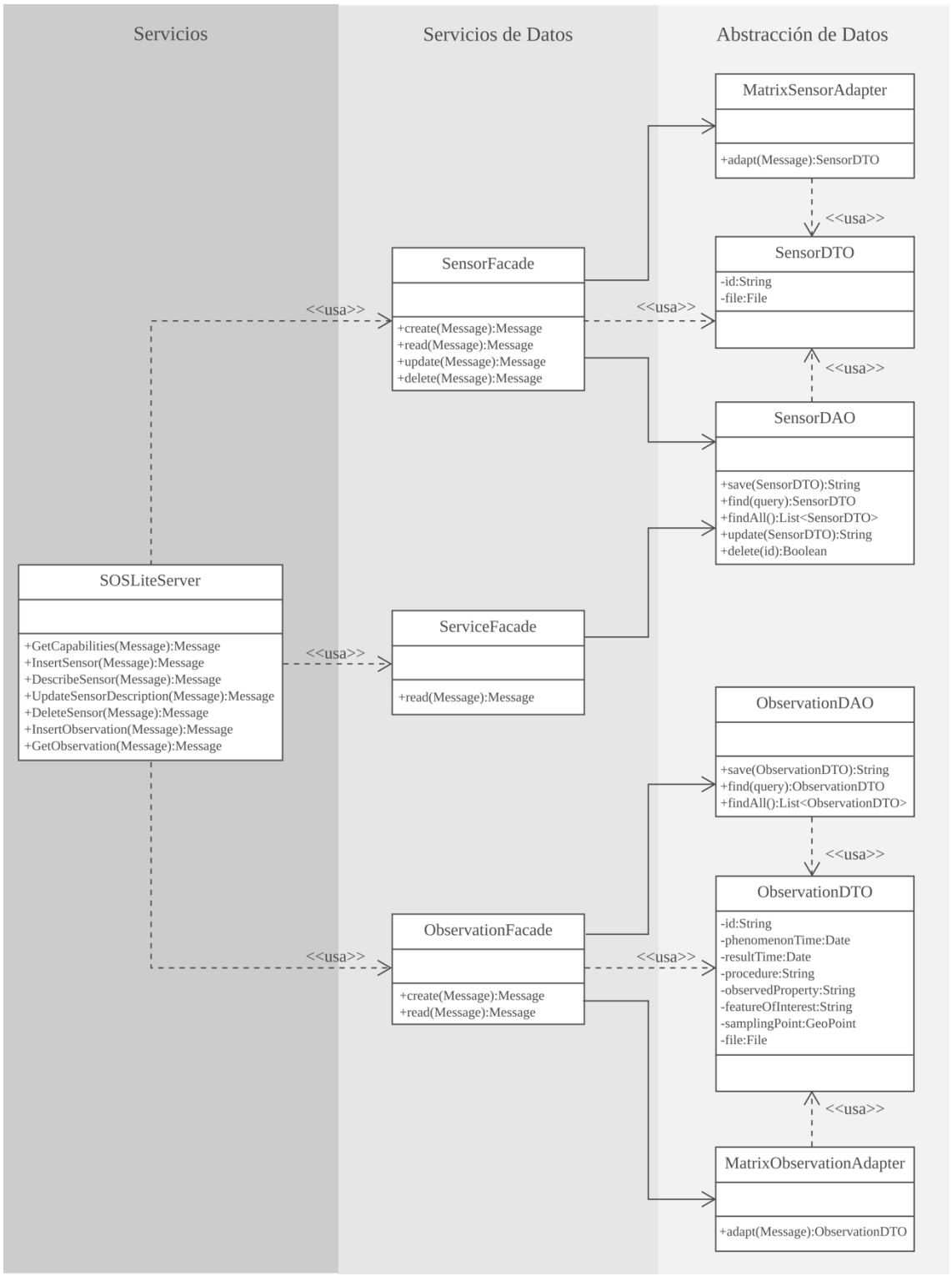


Figura 19 - Diagrama de clases del SOSLite

A partir del análisis anterior, se ha optado por el uso de una base de datos orientada a documentos [89], es decir, una de las opciones de base de datos NoSQL [90], en la cual los datos son almacenados en documentos con una estructura flexible y poco relacional, que cuentan con un gran rendimiento en operaciones de inserción y consultas que no requieran agregación de documentos.

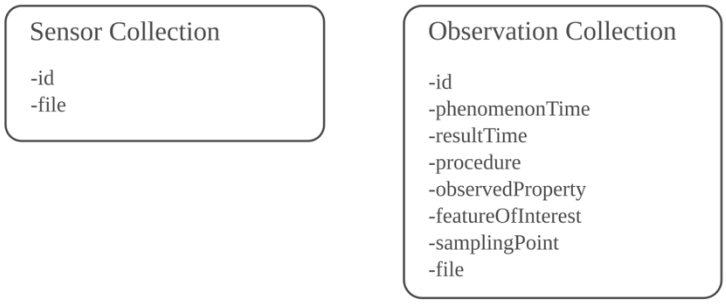


Figura 20 - Modelo de datos del SOSLite

3.3.3. Diagramas de secuencia (DS)

La descripción del funcionamiento del SOSLite contempla una visión estática, mediante el diagrama de casos de uso, el modelo de conceptos, el diagrama de clases y el modelo de datos; y una visión dinámica, utilizando para eso los diagramas de secuencia (DS). Cada uno de los DS se encuentra asociado con uno de los RF del sistema; mientras que, los RNF son indicadores de la dirección que debe seguir la arquitectura del programa y quedan integrados dentro de la visión estática del sistema y la configuración de despliegue final.

Así, se incluyen siete diagramas de secuencia (Tabla XXVI); que representan todas las interacciones necesarias para dar respuesta a los requisitos funcionales definidos en el análisis. Estos diagramas de secuencia son específicos y detallados para el SOSLite, como una implementación de SOS ligero enfocado en la interoperabilidad en las SN.

Tabla XXVI - Diagramas de secuencia

Nº	Diagrama de secuencia
1	Consultar la auto-descripción del sistema
2	Insertar sensor
3	Consultar sensor
4	Actualizar un sensor
5	Eliminar un sensor
6	Insertar observación
7	Consultar observación

Los diagramas de secuencia que se presentan a continuación, utilizan los objetos creados con base en el diagrama de clases (Figura 19) que se ha introducido anteriormente y las operaciones del estándar SOS presentadas en el marco teórico y cuya implementación es detallada en la siguiente sección. Esta aproximación permite una vinculación directa entre el diseño que se ha realizado y las especificaciones a las cuales responde.

3.3.3.a. Consultar la auto-descripción del sistema

La auto-descripción del estándar SOS permite conocer las operaciones que soporta la implementación del SOS utilizado, la identificación del servicio, los filtros soportados y los metadatos de la información almacenada en dicho SOS.

Con el fin de obtener la auto-descripción del SOS desplegado (RF01) se emplea la operación *GetCapabilitites* del estándar SOS. La petición puede ser iniciada tanto por el proveedor de datos como por el consumidor y es recibida por el punto de entrada al sistema, un objeto de la clase *SOSLiteServer*, quien a su vez llama al único método de la clase *ServiceFacade*; es en este método donde se recopilan los datos, se aplican los filtros pasados en la petición, y finalmente se prepara y envía la respuesta a quien ha originado la petición.

La siguiente figura (Figura 21) detalla el proceso completo que lleva a cabo el sistema en el caso en que la petición es iniciada por un proveedor de datos; el caso del consumidor de datos es similar por lo que se ha obviado.

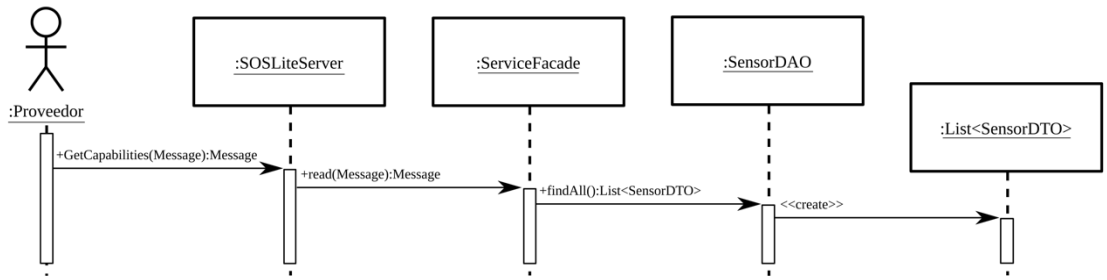


Figura 21 – DS del SOSLite: GetCapabilities

3.3.3.b. Insertar sensor

En el caso en el que se pretende insertar datos y metadatos de un sensor por primera vez al sistema (Figura 22) (RF02), se utiliza la operación *InsertSensor* del perfil *transactional* del SOS. La misma, tiene una representación directa en el método créate de la clase *SensorFacade*; este se encarga de: obtener la información desde el mensaje pasado, validar que no exista el sensor dentro del sistema, almacenarlo y finalmente preparar la respuesta para el proveedor, quien es el que inicia la secuencia.

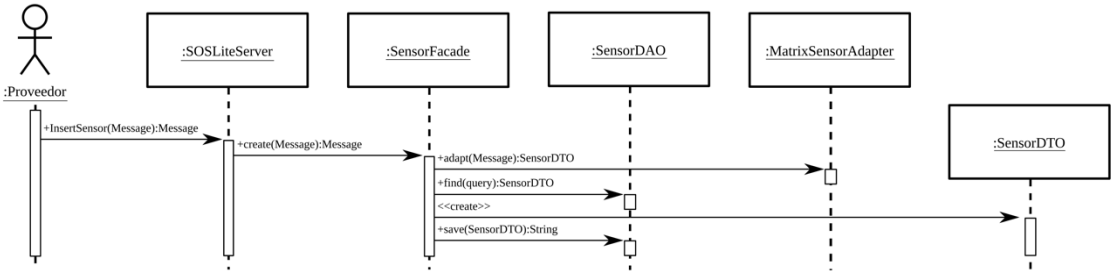


Figura 22 - DS del SOSLite: InsertSensor

3.3.3.c. Consultar sensor

Cuando se desean conocer los datos que el sistema tiene sobre un sensor (RF03), se utiliza la operación *DescribeSensor* (Figura 23). Esta operación es atendida por el SOSLite y mediante el método *read* de la clase *SensorFacade*, realiza una consulta para obtener los datos solicitados por el consumidor. La respuesta que el sistema entrega incluye los datos provenientes de la base de datos y del sistema de ficheros.

Es importante resaltar como el uso de la clase *SensorDAO* disminuye el acoplamiento en el sistema, debido a que presenta un acceso a los datos del sistema de ficheros y de la base de datos que es transparente y transversal a toda la aplicación, eliminado la dependencia de las herramientas utilizadas; además, permite una abstracción completa para los objetos de la clase *SensorFacade* haciendo que estos solo necesiten conocer cómo construir la *query* empleada a nivel de SOSLite.

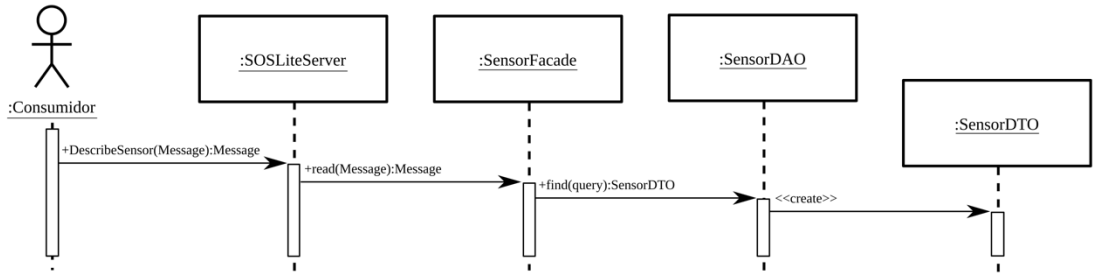


Figura 23 - DS del SOSLite: DescribeSensor

3.3.3.d. Actualizar un sensor

Similar al proceso para agregar un sensor nuevo es el de actualizar uno existente (RF04), con este fin, el proveedor utiliza la operación *UpdateSensorDescription* del estándar (Figura 24). Como en casos anteriores, el objeto *Facade* se encarga de manejar la complejidad de la interacción con las otras partes del sistema, validando que el sensor exista con anterioridad en el sistema, eliminado los datos y metadatos existentes, para finalmente, almacenar los nuevos datos enviados por el proveedor.

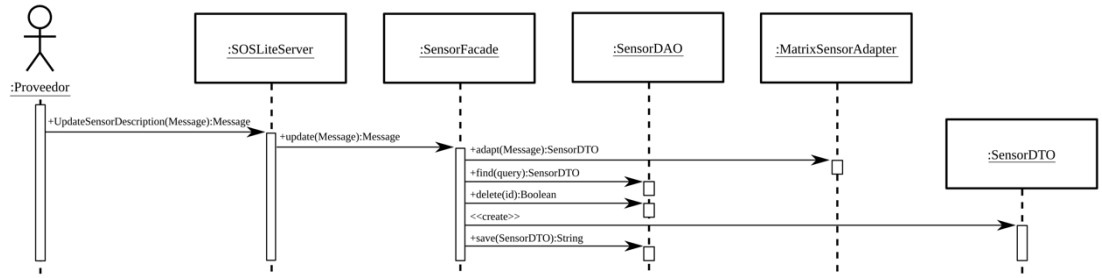


Figura 24 - DS del SOSLite: UpdateSensorDescription

3.3.3.e. Eliminar un sensor

Si la solicitud está orientada a eliminar un sensor existente en el sistema (RF05) se emplea la operación *DeleteSensor* del perfil *transactional*. Esta operación se encuentra encapsulada dentro del método *delete* de la clase *SensorFacade*, el cual se encarga de comprobar la existencia del sensor y de llamar al método de *SensorDAO* para eliminar los datos y metadatos del sensor cuyo identificador a enviado el proveedor (Figura 25).

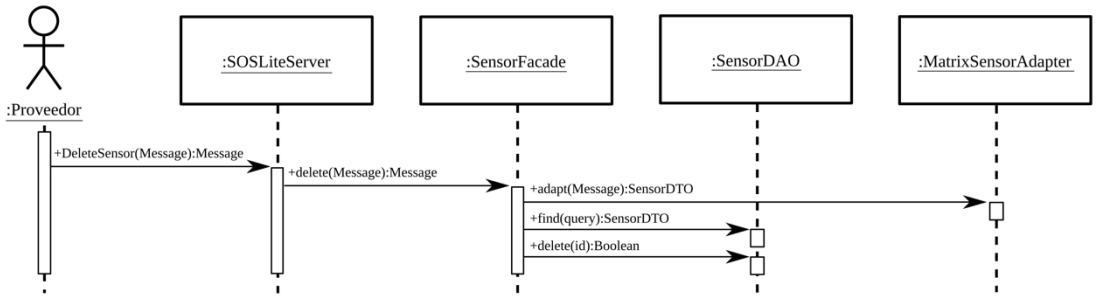


Figura 25 - DS del SOSLite: DeleteSensor

3.3.3.f. Insertar observación

Insertar observaciones al sistema (RF06) es una de las acciones que se realiza con mayor asiduidad dentro de las implementaciones SOS en casos reales desplegados en la actualidad; para llevarla a cabo el estándar provee la operación *InsertObservation* (Figura 26). En el caso del SOSLite, la complejidad de esta operación es abstraída dentro de la clase *ObservationFacade*, la cual interactúa con las otras partes del sistema para lograr que la observación quede registrada dentro del sistema y pueda ser accesible posteriormente.

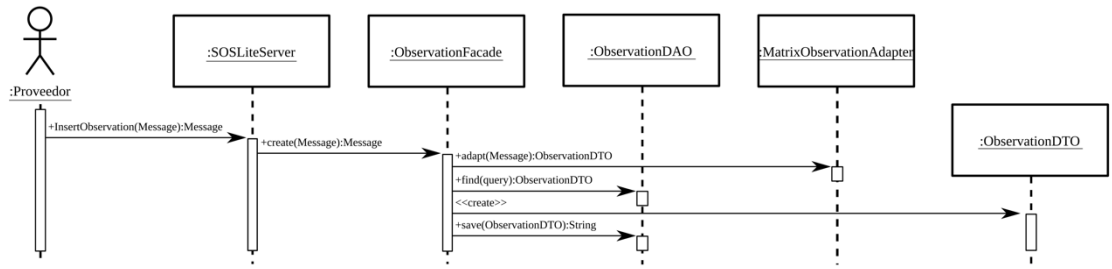


Figura 26 - DS del SOSLite: InsertObservation

3.3.3.g. Consultar observación

Junto con la acción anterior, consultar observaciones (RF08), es una de las utilidades del SOS más usadas. Dentro del SOSLite la operación *GetObservation* del perfil *core* provee esta funcionalidad y el la clase *ObservationFacade* la que brinda el método *read* la cual abstrae la complejidad de los filtros que soporta esta operación, transformado los mismos en una consulta para el *ObservationDAO* (Figura 27).

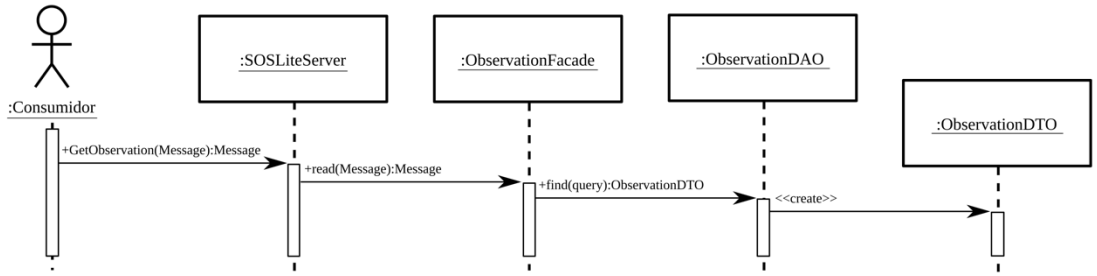


Figura 27 - DS del SOSLite: GetObservation

3.4. Implementación

3.4.1. Sensor Observation Service (SOS) Ligero

El SOSLite es una implementación del perfil de SOS ligero, pero que extiende sus funcionalidades al perfil *transactional* del estándar SOS; así que, permite operaciones de inserción, actualización y eliminación sobre sensores y de inserción sobre las observaciones; que se suman a las operaciones de consulta de sensores, observaciones y capacidades del sistema.

Así, el SOSLite realiza una adaptación del perfil *transactional* para hacerlo coherente con el perfil ligero, configurando las operaciones de *InsertSensor*, *UpdateSensorDescription*, *DeleteSensor* y *InsertObservation* mediante el ajuste de sus atributos asociados a las peticiones y respuestas. De igual forma, realiza algunas pequeñas modificaciones sobre el perfil de SOS ligero, de forma que los identificadores son asignados de forma automática por el SOSLite.

A continuación, se detallan cada una de las operaciones soportadas por el SOSLite; como se ha explicado anteriormente, las referencias al concepto *Sensor*, dentro de las siguientes descripciones de operación, se refiere a un *PhysicalSystem*, que es la unidad determinada en el perfil ligero y utilizada en la implementación del SOSLite.

3.4.1.a. GetCapabilities

La operación *GetCapabilities* hace parte del perfil *core* del SOS y provee los metadatos necesarios para que los clientes puedan realizar peticiones validas al SOSLite. En la siguiente tabla (Tabla XXVII) se muestra los atributos de las peticiones y las respuestas.

Tabla XXVII - SOSLite GetCapabilities atributos petición/respuesta

Mensaje	Atributo	Descripción
Petición	AcceptVersions	Identifica la versión del SOS que soporta. El SOSLite solo da soporte a la versión 2.0.0
	Sections	Indica que secciones quieren recibirse en la respuesta. Si no se indica ningún valor la respuesta incluye todas las secciones. Soporta los valores: *ServiceIdentification, ServiceProvider, FilterCapabilities, InsertionCapabilities, Contents*
Respuesta	ServiceIdentification	Esta sección incluye el título, la descripción y la versión de SOS del servicio; además, de los perfiles que soporta
	ServiceProvider	Indica los datos de la organización que despliega el servicio, incluyendo: el nombre de la organización, la URL de la página web y la información de contacto
	FilterCapabilities	Contienen la información sobre los filtros soportados sean temporales o espaciales
	InsertionCapabilities	Incluye el conjunto de tipos soportados para las operaciones de inserción en el SOSLite para los sensores, observaciones y featureOfInterest
	Contents	Agrupa la información registrada en el SOS incluyéndola relacionada con: *observableProperty, offering, procedure y observationType*

El siguiente ejemplo muestra una petición *GetCapabilities* y su respuesta. Para simplificar el ejemplo, se ha omitido la importación de los esquemas y el envoltorio SOAP; además, se han reducido algunos valores dejándolo indicado mediante puntos suspensivos.

```
<sos:GetCapabilities>
  <ows:AcceptVersions>
    <ows:Version>2.0.0</ows:Version>
  </ows:AcceptVersions>
  <ows:Sections>
    <ows:Section>ServiceIdentification</ows:Section>
    <ows:Section>ServiceProvider</ows:Section>
    <ows:Section>FilterCapabilities</ows:Section>
    <ows:Section>InsertionCapabilities</ows:Section>
    <ows:Section>Contents</ows:Section>
  </ows:Sections>
</sos:GetCapabilities>
```

```
<sos:Capabilities version="2.0.0">
  <!--ServiceIdentification-->
  <ows:ServiceIdentification>
    <ows:Title>SOSLite</ows:Title>
    <ows:Abstract>SOSLite Sensor Observation Service </ows:Abstract>
    <ows:ServiceType codeSpace="http://opengeospatial.net">OGC:SOS</ows:ServiceType>
    <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
    <ows:Profile>http://www.opengis.net/spec/SOS/2.0/conf/gfoi</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SOS/2.0/conf/obsByIdRetrieval</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SOS/2.0/conf/sensorInsertion</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SOS/2.0/conf/sensorDeletion</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SOS/2.0/conf/obsInsertion</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SOS/2.0/conf/resultInsertion</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SOS/2.0/conf/resultRetrieval</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SOS/2.0/conf/spatialFilteringProfile</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SOS/2.0/conf/soap</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SWE/2.0/conf/uml-block-components</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SWE/2.0/conf/uml-record-components</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SWE/2.0/conf/xsd-record-components</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/SWE/2.0/conf/xsd-block-components</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/OMXML/2.0/conf/samplingPoint</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/OMXML/2.0/conf/observation</ows:Profile>
    <ows:Fees>NONE</ows:Fees>
    <ows:AccessConstraints>NONE</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <!--ServiceProvider-->
  <ows:ServiceProvider>
    <ows:ProviderName>SOSLite</ows:ProviderName>
    <ows:ProviderSite xlink:href="http://soslite.com/"/>
    <ows:ServiceContact>
      <ows:IndividualName>Juan Vicente Pradilla</ows:IndividualName>
      <ows:PositionName>Investigador Asociado</ows:PositionName>
      <ows:ContactInfo>
        <ows:Address>
          <ows:DeliveryPoint>Camino de Vera, s/n</ows:DeliveryPoint>
          <ows:City>Valencia</ows:City>
          <ows:PostalCode>46022</ows:PostalCode>
          <ows:Country>España</ows:Country>
          <ows:ElectronicMailAddress>juaprace@teleco.upv.es</ows:ElectronicMailAddress>
        </ows:Address>
      </ows:ContactInfo>
    </ows:ServiceContact>
  </ows:ServiceProvider>
  <!--FilterCapabilities-->
  <sos:filterCapabilities>
    <fes:Filter_Capabilities>
      <fes:Conformance>
        <fes:Constraint name="ImplementsQuery">
          <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
        </fes:Constraint>
        <fes:Constraint name="ImplementsAdHocQuery">
          <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
        </fes:Constraint>
```

```
54.             <fes:Constraint name="ImplementsFunctions">
55.                 <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
56.             </fes:Constraint>
57.             <fes:Constraint name="ImplementsResourceId">
58.                 <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
59.             </fes:Constraint>
60.             <fes:Constraint name="ImplementsMinStandardFilter">
61.                 <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
62.             </fes:Constraint>
63.             <fes:Constraint name="ImplementsStandardFilter">
64.                 <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
65.             </fes:Constraint>
66.             <fes:Constraint name="ImplementsMinSpatialFilter">
67.                 <ows:NoValues/><ows:DefaultValue>true</ows:DefaultValue>
68.             </fes:Constraint>
69.             <fes:Constraint name="ImplementsSpatialFilter">
70.                 <ows:NoValues/><ows:DefaultValue>true</ows:DefaultValue>
71.             </fes:Constraint>
72.             <fes:Constraint name="ImplementsMinTemporalFilter">
73.                 <ows:NoValues/><ows:DefaultValue>true</ows:DefaultValue>
74.             </fes:Constraint>
75.             <fes:Constraint name="ImplementsTemporalFilter">
76.                 <ows:NoValues/><ows:DefaultValue>true</ows:DefaultValue>
77.             </fes:Constraint>
78.             <fes:Constraint name="ImplementsVersionNav">
79.                 <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
80.             </fes:Constraint>
81.             <fes:Constraint name="ImplementsSorting">
82.                 <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
83.             </fes:Constraint>
84.             <fes:Constraint name="ImplementsExtendedOperators">
85.                 <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
86.             </fes:Constraint>
87.             <fes:Constraint name="ImplementsMinimumXPath">
88.                 <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
89.             </fes:Constraint>
90.             <fes:Constraint name="ImplementsSchemaElementFunc">
91.                 <ows:NoValues/><ows:DefaultValue>false</ows:DefaultValue>
92.             </fes:Constraint>
93.         </fes:Conformance>
94.         <fes:Spatial_Capabilities>
95.             <fes:GeometryOperands>
96.                 <fes:GeometryOperand name="gml:Point"/>
97.                 <fes:GeometryOperand name="gml:Polygon"/>
98.             </fes:GeometryOperands>
99.             <fes:SpatialOperators>
100.                <fes:SpatialOperator name="BBOX"/>
101.            </fes:SpatialOperators>
102.        </fes:Spatial_Capabilities>
103.        <fes:Temporal_Capabilities>
104.            <fes:TemporalOperands>
105.                <fes:TemporalOperand name="gml:TimePeriod"/>
106.                <fes:TemporalOperand name="gml:TimeInstant"/>
107.            </fes:TemporalOperands>
108.            <fes:TemporalOperators>
109.                <fes:TemporalOperator name="During"/>
110.                <fes:TemporalOperator name="TEquals"/>
111.            </fes:TemporalOperators>
112.        </fes:Temporal_Capabilities>
113.    </fes:Filter_Capabilities>
114. </sos:filterCapabilities>
115. <!--InsertionCapabilities-->
116. <sos:extension>
117.     <sos:InsertionCapabilities>
118.         <sos:procedureDescriptionFormat>…/sensorML/1.0.1</sos:procedureDescriptionFormat>
119.         <sos:featureOfInterestType>…/SF_SamplingPoint</sos:featureOfInterestType>
```

```
120.          <sos:observationType>http://www.opengis.net/…/OM_Measurement</sos:observationType>
121.          <sos:observationType>http://www.opengis.net/…/OM_CountObservation</sos:observationType>
122.          <sos:observationType>http://www.opengis.net/…/OM_TruthObservation</sos:observationType>
123.          <sos:observationType>http://www.opengi…/OM_CategoryObservation</sos:observationType>
124.          <sos:observationType>http://www.opengis.net/…/OM_TextObservation</sos:observationType>
125.          <sos:supportedEncoding>http://www.opengis…/TextEncoding</sos:supportedEncoding>
126.       </sos:InsertionCapabilities>
127.    </sos:extension>
128.    <!--Contents-->
129.    <sos:contents>
130.       <sos:Contents>
131.          <swes:procedureDescriptionFormat>.../sensorML/2.0</swes:procedureDescriptionFormat>
132.          <sos:responseFormat>http://www.opengis.net/om/2.0</sos:responseFormat>
133.          <sos:featureOfInterestType> …/SF_SamplingPoint</sos:featureOfInterestType>
134.          <sos:observationType>…/OM_Measurement</sos:observationType>
135.          <sos:observationType>…/OM_CountObservation</sos:observationType>
136.          <sos:observationType>…/OM_TruthObservation</sos:observationType>
137.          <sos:observationType>…/OM_CategoryObservation</sos:observationType>
138.          <sos:observationType>…/OM_TextObservation</sos:observationType>
139.          <swes:observableProperty>…SOSLite.com/observableProperty/1</swes:observableProperty>
140.          <swes:offering>
141.             <sos:ObservationOffering>
142.                <swes:identifier>http://www.SOSLite.com/offering/1</swes:identifier>
143.                <swes:name>Offering for sensor 1</swes:name>
144.                <swes:procedure>http://www.SOSLite.com/procedure/1</swes:procedure>
145.                <swes:procedureDescriptionFormat>…/sensorML/1.0</swes:procedureDescriptionFormat>
146.                <sos:observedArea>
147.                   <gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
148.                      <gml:lowerCorner>39.482369, -0.343578</gml:lowerCorner>
149.                      <gml:upperCorner>39.482369, -0.343578</gml:upperCorner>
150.                   </gml:Envelope>
151.                </sos:observedArea>
152.                <sos:phenomenonTime>
153.                   <gml:TimePeriod gml:id="phenomenonTime">2016-05-02T17:47…</gml:TimePeriod>
154.                </sos:phenomenonTime>
155.             </sos:ObservationOffering>
156.          </swes:offering>
157.       </sos:Contents>
158.    </sos:contents>
159. </sos:Capabilities>
```

3.4.1.b. InsertSensor

Permite a los proveedores de datos registrar un nuevo sensor dentro del SOSLite. Esta operación forma parte del perfil *transactional*. Para registrar cualquier observación dentro del SOS, es necesario que primero se haya insertado un sensor. Sus atributos de peticiones y respuesta se muestran en la siguiente tabla (Tabla XXVIII):

Tabla XXVIII – SOSLite InsertSensor atributos petición/respuesta

Mensaje	Atributo	Descripción
Petición	procedureDescription	Contiene el modelo del sensor en formato SensorML
	procedureDescriptionFormat	Indica el formato del sensor. SensorML es el único formato soportado por el SOSLite
	observableProperty	Conjunto de identificadores de las *observableProperty* sobre las que el sensor generara observaciones
	Metadata	Incluye los identificadores de los tipos asociados las observaciones y los featureOfInterest
Respuesta	assignedProcedure	Es el identificador del *procedure* asignado en el sistema. En el SOSLite estos identificadores son proveídos por el sistema, con un formato numérico y una generación incremental

	assignedOffering	Es el identificador del *offering* asignado en el sistema. En el SOSLite estos identificadores son proveídos por el sistema y se corresponde al mismo número utilizado para identificar el *procedure*

A continuación, se presenta un ejemplo de la operación *InsertSensor*, tanto petición como respuesta. Las mismas se han simplificados omitiendo la importación de los esquemas y el envoltorio SOAP; además, se han reducido algunos valores dejándolo indicado mediante puntos suspensivos.

```
1.  <swes:InsertSensor>
2.    <!--procedureDescription-->
3.    <swes:procedureDescription>
4.      <PhysicalSystem>
5.      …
6.      </PhysicalSystem>
7.    </swes:procedureDescription>
8.     <!--procedureDescriptionFormat-->
9.    <swes:procedureDescriptionFormat>…/sensorML/1.0.1</swes:procedureDescriptionFormat>
10.   <!--ObservableProperty-->
11.   <swes:observableProperty>…SOSLite.com/observableProperty/1</swes:observableProperty>
12.   <!--Metadata-->
13.   <swes:metadata>
14.     <sos:SosInsertionMetadata>
15.       <sos:observationType>…/OM_Measurement</sos:observationType>
16.       <sos:observationType>…/OM_CategoryObservation</sos:observationType>
17.       <sos:observationType>…/OM_CountObservation</sos:observationType>
18.       <sos:observationType>…/OM_TextObservation</sos:observationType>
19.       <sos:observationType>…/OM_TruthObservation</sos:observationType>
20.       <sos:observationType>…/OM_GeometryObservation</sos:observationType>
21.       <sos:observationType>…/OM_SWEArrayObservation</sos:observationType>
22.       <sos:featureOfInterestType>…/SF_SamplingPoint</sos:featureOfInterestType>
23.     </sos:SosInsertionMetadata>
24.   </swes:metadata>
25. </swes:InsertSensor>
```

```
1.  <swes:InsertSensorResponse>
2.    <swes:assignedProcedure>http://www.SOSLite.com/procedure/1</swes:assignedProcedure>
3.    <swes:assignedOffering>http://www.SOSLite.com/offering/1</swes:assignedOffering>
4.  </swes:InsertSensorResponse>
```

3.4.1.c. DescribeSensor

Esta operación permite obtener los metadatos asociados con un sensor que se han registrado dentro del SOS. *DescribeSensor* hace parte del perfil *core* y es obligatoria en todos los SOS. Las peticiones y respuestas son muy sencillas y se pueden ver en la siguiente tabla (Tabla XXIX):

Tabla XXIX – SOSLite DescribeSensor atributos petición/respuesta

Mensaje	Atributo	Descripción
Petición	procedure	Es el identificador del sensor del cual se quieren recuperar los metadatos
Respuesta	description	Contiene el modelo del sensor en formato SensorML
	procedureDescriptionFormat	Indica el formato del sensor. SensorML es el único formato soportado por el SOSLite

El siguiente ejemplo simplificado (sin importación de esquemas) muestra una petición y

una respuesta para esta operación.

```
<swes:DescribeSensor
  <swes:procedure>http://www.SOSLite.com/procedure/1</swes:procedure>
</swes:DescribeSensor>
```

```
<swes:DescribeSensorResponse>
  <swes:procedureDescriptionFormat>…/sensorML/1.0.1</swes:procedureDescriptionFormat>
  <swes:description>
    <PhysicalSystem>
          …
    </PhysicalSystem>
  </swes:description>
</swes:DescribeSensorResponse>
```

3.4.1.d. UpdateSensorDescription

La operación *UpdateSensorDescription* permite el envío de un nuevo modelo en formato SensorML que remplaza los datos y metadatos de un sensor que existen con anterioridad dentro del SOSLite, preservando el identificador y las observaciones asociadas. Los atributos de sus mensajes (Tabla XXX) son idénticos a los de la operación *InsertSensor*.

Tabla XXX – SOSLite UpdateSensorDescription atributos petición/respuesta

Mensaje	Atributo	Descripción
Petición	procedureDescription	Contiene el modelo del sensor en formato SensorML
	procedureDescriptionFormat	Indica el formato del sensor. SensorML es el único formato soportado por el SOSLite
	observableProperty	Conjunto de identificadores de las *observableProperty* sobre las que el sensor generara observaciones
	metadata	Incluye los identificadores de los tipos asociados las observaciones y los featureOfInterest
Respuesta	updatedProcedure	Es el identificador del *procedure* que se ha actualizado

A continuación, se puede observar un ejemplo de actualización para un sensor existente en el sistema:

```
<swes:UpdateSensorDescription>
  <!--procedureDescription-->
  <swes:procedureDescription>
    <PhysicalSystem>
          …
    </PhysicalSystem>
  </swes:procedureDescription>
   <!--procedureDescriptionFormat-->
  <swes:procedureDescriptionFormat>…/sensorML/1.0.1</swes:procedureDescriptionFormat>
  <!--ObservableProperty-->
  <swes:observableProperty>...SOSLite.com/observableProperty/1</swes:observableProperty>
  <!--Metadata-->
  <swes:metadata>
    <sos:SosInsertionMetadata>
      <sos:observationType>…/OM_Measurement</sos:observationType>
      <sos:observationType>…/OM_CategoryObservation</sos:observationType>
      <sos:observationType>…/OM_CountObservation</sos:observationType>
      <sos:observationType>…/OM_TextObservation</sos:observationType>
      <sos:observationType>…/OM_TruthObservation</sos:observationType>
      <sos:observationType>…/OM_GeometryObservation</sos:observationType>
      <sos:observationType>…/OM_SWEArrayObservation</sos:observationType>
```

```
            <sos:featureOfInterestType>…/SF_SamplingPoint</sos:featureOfInterestType>
        </sos:SosInsertionMetadata>
    </swes:metadata>
</swes:UpdateSensorDescription>
```

```
<swes:UpdateSensorDescriptionResponse>
    <swes:updatedProcedure>http://www.SOSLite.com/procedure/1</swes:updatedProcedure>
</swes:UpdateSensorDescriptionResponse>
```

3.4.1.e. DeleteSensor

Permite a un proveedor, eliminar los datos de un sensor que ha añadido con anterioridad y todas las observaciones asociadas a este. Es una operación con atributos de petición y respuesta muy simple (Tabla XXXI), que forma parte del perfil *transactional*.

Tabla XXXI – SOSLite DeleteSensor atributos petición/respuesta

Mensaje	Atributo	Descripción
Petición	procedure	Es el identificador del sensor del cual se quieren eliminar
Respuesta	deletedProcedure	Es el identificador del sensor que se ha eliminado

A continuación, se observa un ejemplo de petición y respuesta para la eliminación de un sensor:

```
<swes:DeleteSensor
    <swes:procedure>http://www.SOSLite.com/procedure/1</swes:procedure>
</swes:DeleteSensor>
```

```
<swes:DeleteSensorResponse>
    <swes:deletedProcedure>http://www.SOSLite.com/procedure/1</swes:deletedProcedure>
</swes:DeleteSensorResponse>
```

3.4.1.f. InsertObservation

Esta operación habilita la posibilidad de agregar observaciones dentro del SOSLite. Permite que se envíen varias observaciones de forma simultánea, cada una de ellas en formato O&M. A continuación, se puede observar (Tabla XXXII) los atributos necesarios para realizar dicha operación y la respuesta esperada.

Tabla XXXII – SOSLite InsertObservation atributos petición/respuesta

Mensaje	Atributo	Descripción
Petición	observation	Contiene el modelo de la observación en formato O&M. SOSLite soporta los tipos de observación: *Measurement, CountObservation, TruthObservation, CategoryObservation y TextObservation*
	offering	Es el identificador del *offering* al que se va a agregar la observación
Respuesta	observation	Es el identificador de la observación asignado en el sistema. En el SOSLite estos identificadores son proveídos por el sistema, con un formato numérico y una generación incremental

Un ejemplo de petición y respuesta para esta operación se presenta a continuación:

```
<sos:InsertObservation>
    <sos:observation>
        <om:OM_Observation>
                …
        </om:OM_Observation>
```

```
    </sos:observation>
    <sos:offering>http://www.SOSLite.com/offering/1/</sos:offering>
</sos:InsertObservation>
```

```
<sos:InsertObservationResponse>
    <sos:observation>http://www.SOSLite.com/observation/1</sos:observation>
</sos:InsertObservationResponse>
```

3.4.1.g. GetObservation

La operación *GetObservation* es parte esencial de la especificación del SOS y se encuentra dentro del perfil *core*. La misma, permite la obtención de las observaciones previamente registradas en el SOSLite. Los atributos de los mensajes de petición y respuesta (Tabla XXXIII), se encuentran limitados como detalla la recomendación para el perfil de SOS ligero.

Tabla XXXIII – SOSLite GetObservation atributos petición/respuesta

Mensaje	Atributo	Descripción
Petición	procedure	Es el conjunto de identificadores de todos los *procedures* para los que se quieren obtener las observaciones
	offering	Incluye los identificadores de los *offerings* para los cuales se quieren consultar las observaciones
	observedProperty	Conjunto de identificadores de las *observableProperty* que tienen en común las observaciones que se desean
	featureOfInterest	Identificadores de las áreas de interés que contienen las observaciones que se desean consultar
	temporalFilter	Filtro temporal que deben cumplir las observaciones que el sistema va a devolver. Pueden ser de tipo: *During*, para un rango acotado de tiempo; o *TEquals*, para un momento de tiempo especifico
	spatialFilter	Filtro espacial que deben cumplir las observaciones que se están consultando. Puede ser de tipo BBOX, que indica un rectángulo sobre el que se hicieron las observaciones
	responseFormat	Indica el formato de las observaciones. O&M es el único formato soportado por el SOSLite
Respuesta	observationData	Cada una de las observaciones del conjunto de ellas que cumplen con los atributos de la petición, en formato O&M

El siguiente ejemplo muestra cómo aplicar todos los filtros en una petición y el formato de la respuesta esperada (los dos mensajes han sido simplificados).

```
<sos:GetObservation>
    <!--Procedure-->
    <sos:procedure>http://www.SOSLite.com/procedure/1</sos:procedure>
    <!--Offering-->
    <sos:offering>http://www.SOSLite.com/offering/1</sos:offering>
    <!--ObservedProperty-->
    <sos:observedProperty>http://www.SOSLite.com/observableProperty/1</sos:observedProperty>
    <!--FeatureOfInterest-->
    <sos:featureOfInterest>http://www.SOSLite.com/featureOfInterest/1</sos:featureOfInterest>
    <!--TemporalFilter-->
    <sos:temporalFilter>
        <fes:TEquals>
            <fes:ValueReference>phenomenonTime</fes:ValueReference>
            <gml:TimeInstant gml:id="ti_1">
                <gml:timePosition>2016-05-02T17:47:15.000+00:00</gml:timePosition>
            </gml:TimeInstant>
        </fes:TEquals>
```

```
18.      </sos:temporalFilter>
19.     <!--SpatialFilter-->
20.     <sos:spatialFilter>
21.       <fes:BBOX>
22.         <gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
23.           <gml:lowerCorner>38.482369 -0.443578</gml:lowerCorner>
24.           <gml:upperCorner>40.482369 -0.243578</gml:upperCorner>
25.         </gml:Envelope>
26.       </fes:BBOX>
27.     </sos:spatialFilter>
28.     <sos:responseFormat>http://www.opengis.net/om/2.0</sos:responseFormat>
29.  </sos:GetObservation>
```

```
1.   <sos:GetObservationResponse>
2.     <sos:observationData>
3.       <om:OM_Observation>
4.                 …
5.       </om:OM_Observation>
6.     </sos:observationData>
7.   </sos:GetObservationResponse>
```

3.4.2. Sensor Model Language (SensorML)

El SensorML es utilizado en el SOSLite para describir los *PhysicalSystem*, que representa una agrupación de sensores, en el intercambio de datos y metadatos de las operaciones *InsertSensor*, *DescribeSensor* y *UpdateSensorDescription* del SOS. El SensorML empleado en el SOSLite es una modificación sobre el estándar, haciéndolo menos versátil pero más ligero.

Dentro del SOSLite se sigue la recomendación del perfil ligero, la cual definen un conjunto mínimo de atributos que deben estar presentes de forma obligatoria dentro de la descripción del sensor. Sin embargo la implementación ignora el atributo *identifier* cuando se incluye en la descripción del sensor, remplazándolo por uno asignado por el sistema. El modelo de datos SensorML utilizado tiene los siguientes atributos (Tabla XXXIV):

Tabla XXXIV – SensorML atributos del modelo

Atributo	Descripción
description	Es un texto corto que describe el *PhysicalSystem*
keywords	Listado de palabras clave que describen el *PhysicalSystem*
identification	Conjunto de términos con los cuales se puede identificar el *PhysicalSystem*. Por ejemplo puede contener un nombre corto y un nombre largo que faciliten la identificación por parte de las personas
classification	Listado de términos que pueden facilitar la clasificación de un *PhysicalSystem*. Por ejemplo, el tipo de sistema de sensores que representa
contacts	Contiene la información de la organización que está operando el *PhysicalSystem*; incluyendo la información de contacto y el nombre de la organización
featuresOfInterest	Listado de las *featuresOfInterest* a las cuales el *PhysicalSystem* se encuentra asociado. Solo pueden ser de tipo *SamplingPoint*
outputs	Es un listado de todas las posibles salidas del *PhysicalSystem*, donde cada una de ellas especifica un nombre, la *observableProperty* a la que está asociada y el código UCUM que especifica la unidad de medición

A continuación, se presenta un ejemplo de un *PhysicalSystem* en formato SensorML (se ha simplificado eliminando la importación de esquemas y cambiando algunos valores):

```
<?xml version="1.0" encoding="UTF-8"?>
<PhysicalSystem>
  <!--Description-->
  <gml:description>
  Estación meteorológica de la UPV con sensores de: temperatura y humedad
  </gml:description>
  <!--Keywords-->
  <keywords>
    <KeywordList>
      <keyword>estación meteorológica</keyword>
      <keyword>temperatura</keyword>
      <keyword>humedad</keyword>
    </KeywordList>
  </keywords>
  <!--Identification-->
  <identification>
    <IdentifierList>
      <identifier name="shortName">
        <Term definition="urn:ogc:def:identifier:OGC:1.0:shortName">
          <label>short name</label>
          <value>UPV estación</value>
        </Term>
      </identifier>
      <identifier name="longName">
        <Term definition="urn:ogc:def:identifier:OGC:1.0:longName">
          <label>long name</label>
          <value>Estación meteorológica de la UPV</value>
        </Term>
      </identifier>
    </IdentifierList>
  </identification>
  <!--Classification-->
  <classification>
    <ClassifierList>
      <classifier>
        <Term definition="http://www.opengis.net/def/property/OGC/0/SensorType">
          <label>sensorType</label>
          <value>estación meteorológica</value>
        </Term>
      </classifier>
    </ClassifierList>
  </classification>
  <!--Contact-->
  <contact>
    <ContactList>
      <member>
        <gmd:ResponsibleParty>
          <gmd:individualName>Juan Vicente Pradilla</gmd:individualName>
          <gmd:organizationName>Universitat Politècnica de València</gmd:organizationName>
          <gmd:positionName>Investigador Asociado</gmd:positionName>
          <gmd:contactInfo>
            <gmd:address>
              <gmd:deliveryPoint>Camino de Vera, s/n</gmd:deliveryPoint>
              <gmd:city>Valencia</gmd:city>
              <gmd:postalCode>46022</gmd:postalCode>
              <gmd:country>España</gmd:country>
              <gmd:electronicMailAddress>juaprace@teleco.upv.es</gmd:electronicMailAddress>
            </sml:address>
          </gmd:contactInfo>
        </gmd:ResponsibleParty>
      </member>
    </ContactList>
  </contact>
  <!--FeaturesOfInterest-->
  <featuresOfInterest>
    <FeatureList>
```

```
67.          <member>
68.             <sams:SF_SpatialSamplingFeature>
69.                <sf:type xlink:href="…/SF_SamplingPoint"/>
70.                <sf:sampledFeature xsi:nil="true"/>
71.                <sams:shape>
72.                   <gml:Point gml:id="UOMlocation">
73.                      <gml:pos srsName="…/EPSG/0/4326">39.482369, -0.343578</gml:pos>
74.                   </gml:Point>
75.                </sams:shape>
76.             </sams:SF_SpatialSamplingFeature>
77.          </member>
78.       </FeatureList>
79.    </featuresOfInterest>
80.    <!--Outputs-->
81.    <outputs>
82.       <OutputList>
83.          <sml:output name="temperatura">
84.             <swe:Quantity>
85.                <swe:uom code="Cel"/>
86.             </swe:Quantity>
87.          </sml:output>
88.          <output name="humedad">
89.             <swe:Quantity>
90.                <swe:uom code="%"/>
91.             </swe:Quantity>
92.          </output>
93.       </OutputList>
94.    </outputs>
95. </PhysicalSystem>
```

3.4.3. Observations & Measurements (O&M)

El modelo O&M es utilizado por el SOSLite para codificar los datos medidos por cada uno de los sensores dentro del sistema. El estándar O&M es uno de los pilares del SWE pero sus posibilidades, mayoritariamente superan el uso que se le da; por esta razón, en el perfil ligero se simplifica al máximo en cuando a atributos y tipos. La implementación SOSLite se adhiere a las recomendaciones del perfil ligero, a excepción del atributo *identifier* que es asignado de forma automática por el sistema. Así, los atributos que comparten todos los tipos de medida son (Tabla XXXV):

Tabla XXXV - O&M atributos del modelo

Atributo	Descripción
phenomenonTime	Describe el instante (*TimeInstant*) o periodo de tiempo (*TimePeriod*) de los datos medidos que contiene la observación
resultTime	Determina el tiempo en el cual los datos estuvieron disponibles. Mayoritariamente es idéntico al *phenomenonTime*
procedure	Contiene el identificador del *PhysicalSystem* que genero la observación
observedProperty	Es el identificador del observedProperty que está monitorizando
featureOfInterest	Representa el identificador del featureOfInterest al cual está vinculada la observación
result	Es el valor medido, el cual, cambian según el tipo de observación que se esté realizando (Tabla XXXVI)

Por su parte, los tipos de observación soportados dentro del SOSLite son (Tabla XXXVI):

Tabla XXXVI - Tipos de observaciones soportados por el SOSLite

Tipo	Descripción	Ejemplo
Measurement	Numérico	<om:result uom="%">75 </om:result>
CountObservation	Contador	<om:result>5</om:result>
TruthObservation	Booleano	<om:result>true </om:result>
TextObservation	Cadena de texto	<om:result>'id':'ax500'</om:result>
CategoryObservation	Selección dentro de un listado acotado	<om:result codeSpace="http://soslite.com/types.xml"> x1</om:result>

El siguiente modelo en O&M ejemplifica una observación de tipo *Measurement* (se han eliminado la importación de esquemas y simplificado algunos valores):

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <om:OM_Observation>
3.     <!--PhenomenonTime-->
4.     <om:phenomenonTime>
5.        <gml:TimeInstant gml:id="phenomenonTime">
6.           <gml:timePosition>2016-05-02T17:47:15.000+00:00</gml:timePosition>
7.        </gml:TimeInstant>
8.     </om:phenomenonTime>
9.     <!--ResultTime-->
10.    <om:resultTime xlink:href="#phenomenonTime"/>
11.    <!--Procedure-->
12.    <om:procedure xlink:href="http://www.SOSLite.com/procedure/1"/>
13.    <!--ObservedProperty-->
14.    <om:observedProperty xlink:href="http://www.SOSLite.com/observableProperty/1"/>
15.    <!--FeaturesOfInterest-->
16.    <featuresOfInterest>
17.       <FeatureList>
18.          <member>
19.             <sams:SF_SpatialSamplingFeature>
20.                <sf:type xlink:href="…/SF_SamplingPoint"/>
21.                <sf:sampledFeature xsi:nil="true"/>
22.                <sams:shape>
23.                   <gml:Point gml:id="UOMlocation">
24.                      <gml:pos srsName="…/EPSG/0/4326">39.482369, -0.343578</gml:pos>
25.                   </gml:Point>
26.                </sams:shape>
27.             </sams:SF_SpatialSamplingFeature>
28.          </member>
29.       </FeatureList>
30.    </featuresOfInterest>
31.    <!--Result-->
32.    <om:result uom="Cel">25</om:result>
33. </om:OM_Observation>
```

3.4.4. Herramientas

Para la implementación se ha realizado una revisión de varias tecnologías, incluyendo: el lenguaje de programación, el sistema de base de datos y el servidor. Siguiendo la arquitectura propuesta, la cual, da respuesta a los requerimientos no funcionales y realizando una intercepción con las capacidades de desarrollo con las que se cuenta se ha optado por utilizar PHP, MongoDB y NGINX para realizar y probar la primera versión del SOSLite. La integración de estas tres tecnologías se puede apreciar en la siguiente figura (Figura 28):

Figura 28 - Tecnologías usadas en el SOSLite

3.4.4.a. PHP

Hypertext Preprocessor – PHP es un lenguaje de programación de código abierto que sigue un paradigma de programación orientada a objetos y que ha sido especialmente desarrollado para dar respuesta a las necesidades del entorno Web; brindando funcionalidades para la creación de servicios web basados en SOAP, HTTP y XML. Además, es multiplataforma y cuenta de una de las comunidades más extensas en la red. Estas características, lo hacen adecuado para dar implementación al diseño del SOSLite y responder a todos los requerimientos funcionales necesarios.

Se debe destacar que PHP es ampliamente utilizado en los ambientes web incluyendo varios de los sitios de mayor auge en el mundo, como: Wordpress.com, Facebook.com y Wikipedia.org. Este hecho, unido a su comunidad lo referencian como un lenguaje de programación maduro y versátil.

3.4.4.b. MongoDB

MongoDB es un sistema de base de datos de código abierto que está orientado a documentos y ha sido desarrollado con el rendimiento y el almacenamiento distribuido como principales estandartes. Responde de forma excepcional en operaciones de inserción y consultas, siempre y cuando, los datos se encuentren desnormalizados. Brinda soporte multiplataforma y, a pesar de su corto tiempo de existencia, cuenta con una de las comunidades más activas de la actualidad. Se adapta a la propuesta arquitectónica y al diseño propuestos para el SOSLite, aportando rendimiento y versatilidad; siendo ideal para el manejo de datos y metadatos de sensores y observaciones.

Esta base de datos se ha hecho muy popular en los últimos años y es utilizada por grandes empresas como: IBM, CERN, Forbes, Foursquare y LinkedIn. Como solución distribuida de gestión de datos es una excelente alternativa, siempre y cuando se reconozcan los casos en los cuales los datos se adaptan a su funcionamiento; siendo las SN uno de los que mayor provecho le puede sacar.

3.4.4.b. NGINX

NGINX es un servidor web y proxy inverso de código abierto que está desarrollado bajo el principio de *Event Driven*, siendo ligero y de alto rendimiento. Además, es multiplataforma y, en la actualidad, usado de forma extensa por grandes compañías alrededor del mundo, tales como: Facebook, Netflix, Wordpress y GitHub. Se puede integrar de forma eficiente con los módulos de PHP y dada su naturaleza de orientación

a eventos, mantiene un bajo uso de recursos computacionales; adaptándose las necesidades del SOSLite.

3.5. Pruebas y Evaluación

3.5.1. Escenarios de prueba

El rendimiento en dispositivos limitados y micro-instancias en la nube es clave para que el SOSLite se convierta en una alternativa viable para las interoperabilidad semántica en los CPS. Para realizar la evaluación del desempeño, se plantean seis potenciales escenarios (Tabla XXXVII); estos modelan entornos posibles en SN pequeñas que pueden ser atendidas por el SOSLite. Los escenarios se enfocan en la consulta (*GetObservation* - 3 escenarios) e inserción (*InsertObservation* - 3 escenarios) de observaciones, que son las operaciones más frecuentemente utilizadas en los casos de despliegues reales.

Tabla XXXVII - Escenarios de pruebas

Operación	Escenario	# Sensores	# Observaciones
GetObservation	1	2	2500
	2	10	500
	3	50	100
InsertObservation	1	2	2500
	2	10	500
	3	50	100

Los escenarios para cada una de las operaciones siguen una progresión similar, siendo los primeros intensivos en observaciones y los últimos casos de uso intensivos sensores (Figura 29). De esta forma se abarcan escenarios reales, que tienen una gran precisión en las mediciones (útiles en ambientes fuertemente variables) o una amplia disponibilidad de sensores (ideal para ambientes complejos).

Figura 29 - Casos de Uso

En cada uno de los escenarios se mide: el uso de procesador (%), el uso de memoria RAM (%), el *throughput* (peticiones/segundo), el promedio del retardo (s), la mediana del retardo (s) y la desviación estándar del retardo (s). Con estas medidas se caracteriza el rendimiento del SOSLite en un ambiente realista de pruebas, con datos generados de forma automática y que siguen la estructura y complejidad propia de los datos que circularían en una red de sensores real.

Por su parte, la generación de las pruebas de carga se ha realizado mediante Apache JMeter (versión 2.13) y los conjuntos de extensiones *standard* y *extras* en su versión 1.3.1. Los cuales son proyectos código abierto, escritos en JAVA. Estos programas se encuentran especializados en la generación de carga para el análisis de desempeño de

los servicios basados en estándares web (como SOAP/XML) y se han instalado en un ordenador personal con sistema operativo GNU/Linux distribución Ubuntu 16.04 LTS y OpenJDK 8.

3.5.2. Pruebas en dispositivos limitados

Para las pruebas en dispositivos limitados se ha realizado un montaje básico que cuenta con: un ordenador donde se ejecuta el programa para pruebas de carga, un equipo de intercomunicación y un dispositivo de recursos limitados (Figura 30). El dispositivo limitado es una Raspberry Pi modelo B, cuyas características son: procesador ARM a 700 MHz, 512 MB de memoria SDRAM, 2 GB de memoria de almacenamiento SD y un consumo energético de 700 mA, (3.5 W).

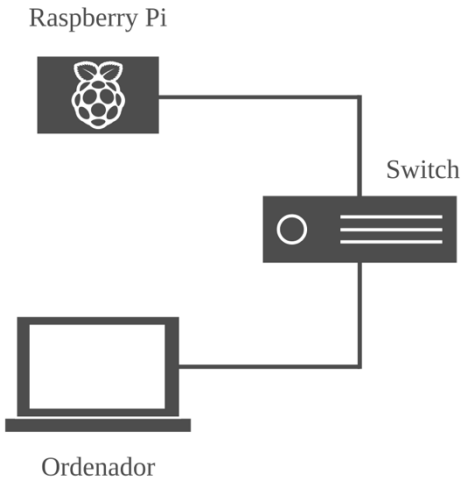


Figura 30 - Montaje de pruebas en dispositivo limitado

Una vez realizadas las pruebas se han graficado los resultados para las operaciones *GetObservation* e *InsertObservatión*. Para mejorar la visualización de los resultados se han eliminado las primeras 100 medidas que constituyen el periodo transitorio y que al encontrarse muy dispersas con respecto al comportamiento normal hacen que las gráficas pierdan detalle.

3.5.2.a. Uso de procesador

El uso del procesador del dispositivo limitado ronda el 50% en la operación *GetObservation* (Figura 31) y el 80% en la operación *InsertObservation* (Figura 32). La diferencia entre las dos operaciones se puede explicar al analizar los diagramas de secuencia para las dos operaciones (Figura 27 - Figura 26), en donde se evidencia que la operación de inserción conlleva un mayor número de interacciones dentro del sistema. Además, la poca diferencia entre el uso de CPU en los diferentes escenarios, cerca de un 10%, es un claro indicativo de la versatilidad del SOSLite para afrontar los casos reales donde se puede desplegar. De igual forma, se puede deducir que el aumento de dispositivos que reportan observaciones en el SOSLite no tiene un impacto lineal sobre el consumo de CPU.

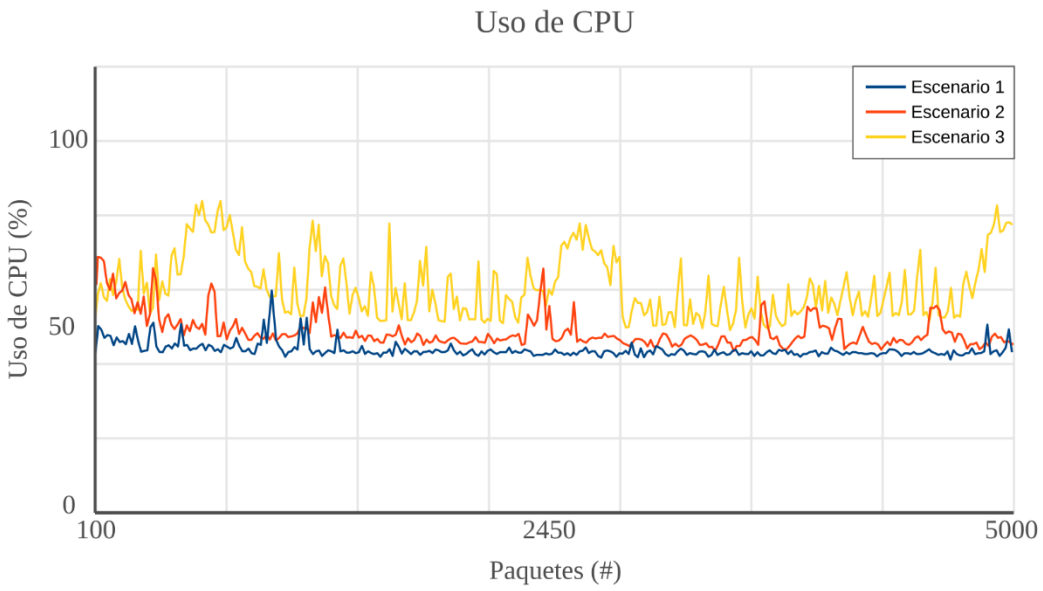


Figura 31 - Uso de procesador: GetObservation

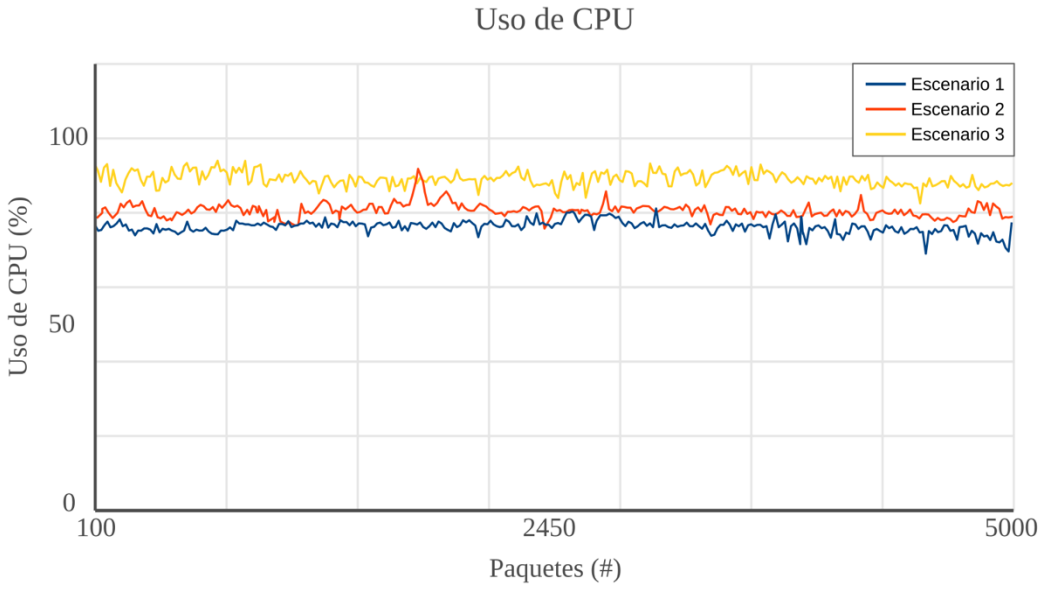


Figura 32 - Uso de procesador: InsertObservation

3.5.2.b. Uso de memoria RAM

A continuación, se puede observar el consumo porcentual de memoria RAM para la operación *GetObservation* (Figura 33) e *InsertObservation* (Figura 34). Al igual que para el uso de la CPU, existe una divergencia palpable entre las dos operaciones que puede ser atribuida a la diferencia en complejidad, medida en elementos que interaccionan, para dar respuesta a la solicitud. De esta forma, el uso de memoria RAM no presenta una diferencia determinante entre los casos intensivos en medidas y los intensivos en dispositivos; afirmando nuevamente el rango de casos a los que puede atender la implementación.

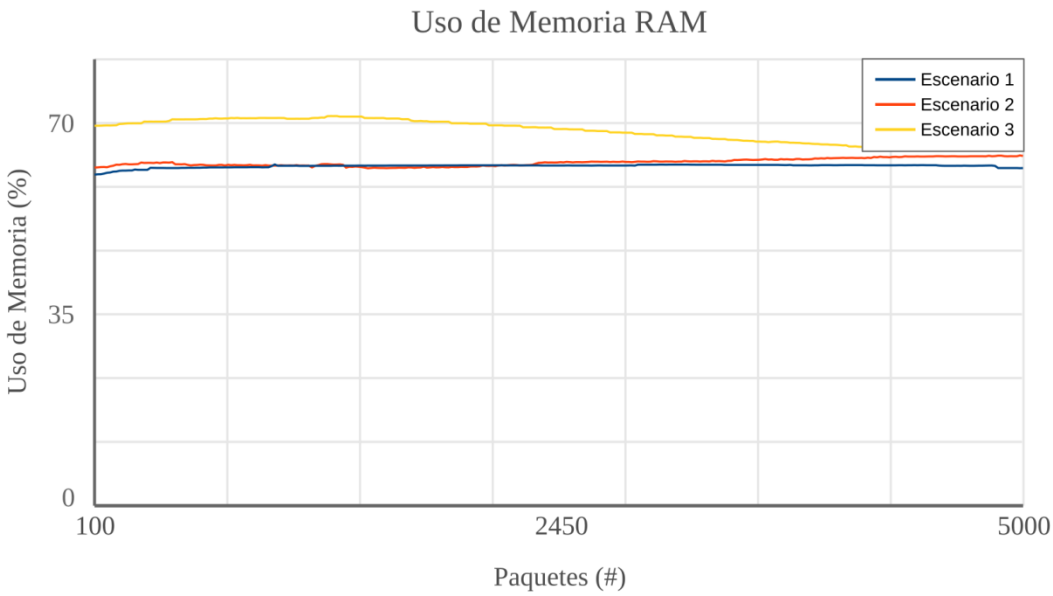


Figura 33 - Uso de memoria: GetObservation

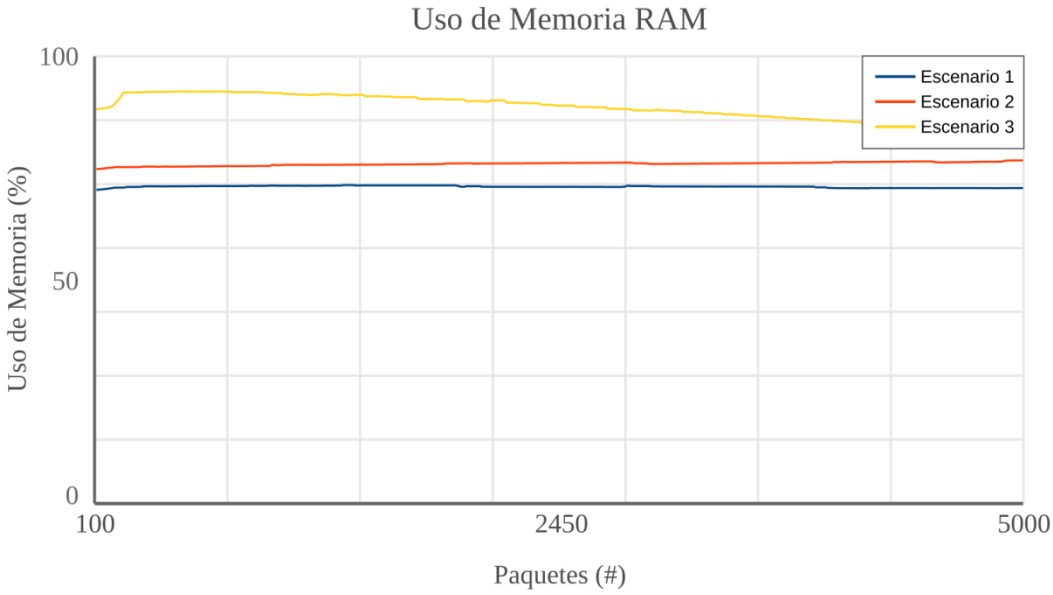


Figura 34 - Uso de memoria: InsertObservation

3.5.2.c. Throughput

En el entorno de pruebas se ha verificado como las dos operaciones estudiadas se comportan dentro de un ambiente estable y controlado. Parte importante de esta verificación, es la medida del *throughput*, que indica la tasa a la que el SOSLite procesa las peticiones que le llegan, y cuyos resultados (Figura 35 - Figura 36) muestran un promedio de 6 peticiones procesadas por segundo. Este dato indica que el SOSLite, en un ambiente local, puede llegar a procesar una de estas operaciones en menos de 200 ms, siendo un tiempo más que aceptable para la gran mayoría de casos de uso reales para los que está ideado.

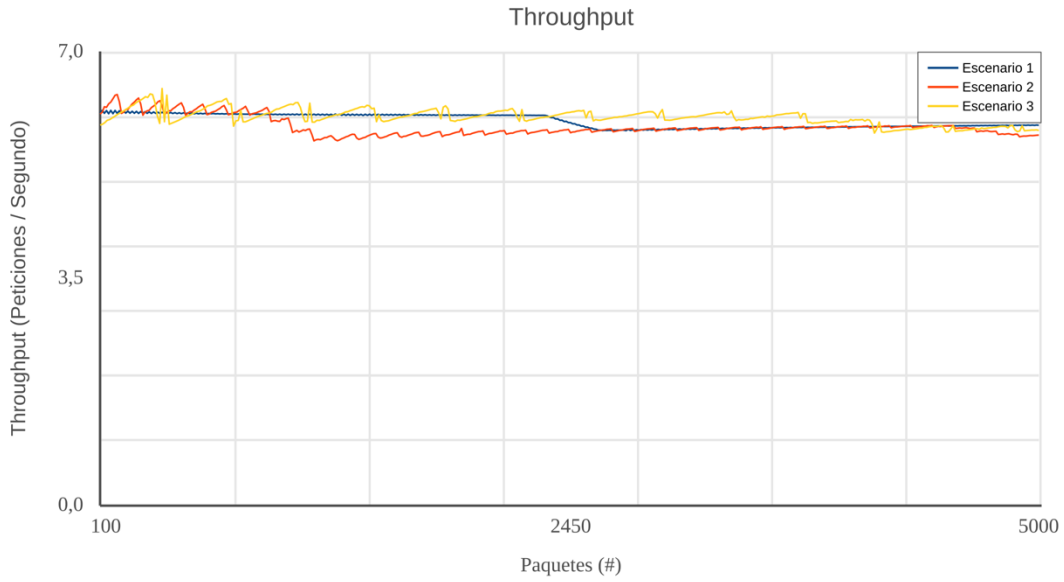


Figura 35 – Throughput: GetObservation

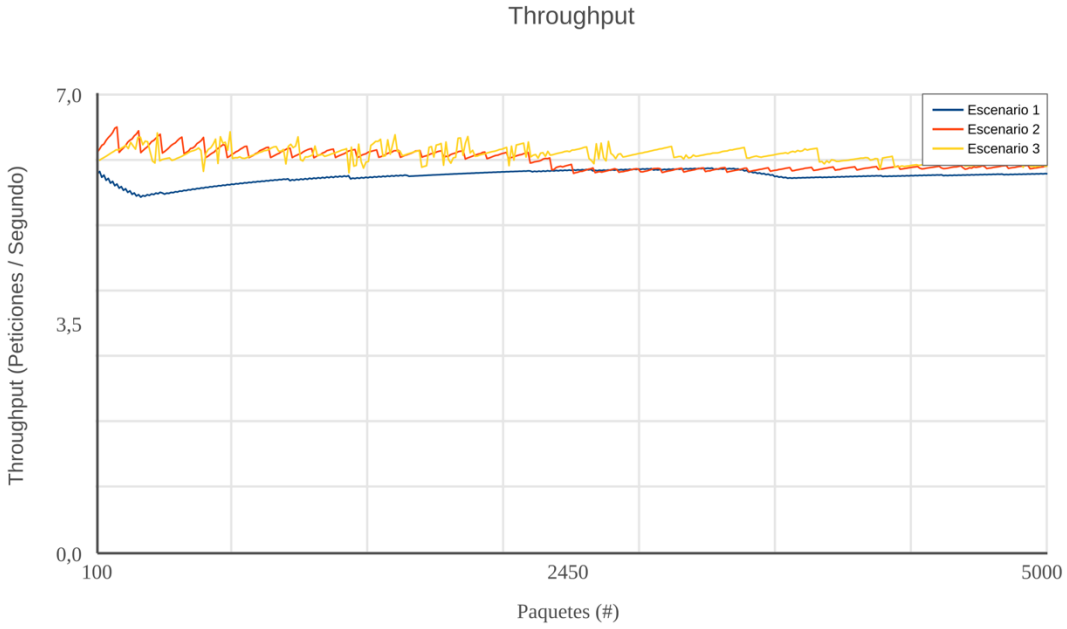


Figura 36 – Throughput: InsertObservation

3.5.2.d. Promedio del retardo

Otras importantes mediciones sobre el comportamiento del SOSLite en el entorno de pruebas, son el promedio, la mediana y la desviación estándar del retardo. En el caso del promedio del retardo, medido como el promedio del intervalo entre el momento en que

se genera la petición y el momento en el que se recibe la respuesta, se observa un comportamiento similar para las dos operaciones (Figura 37 - Figura 38), siendo cercana a cero segundos cuando se tiene pocos sensores y muchas mediciones (en azul) y próximo a los 10 segundos cuando son muchos sensores con pocas mediciones. En este punto es necesario enfatizar que, el escenario 3 tiene sus limitaciones en cuanto a la exactitud de la simulación de operaciones paralelas, debido a que todas las peticiones generadas pasan por la misma interfaz de red y por ende, el efecto del paralelismo solo se puede evidenciar en los nodos generador y SOSLite. Aun así, los datos presentados presentan un comportamiento aceptable cuando existen gran cantidad de sensores; con lo cual, en los casos en los que se realizan muchas mediciones con pocos sensores se obtiene un rendimiento superior.

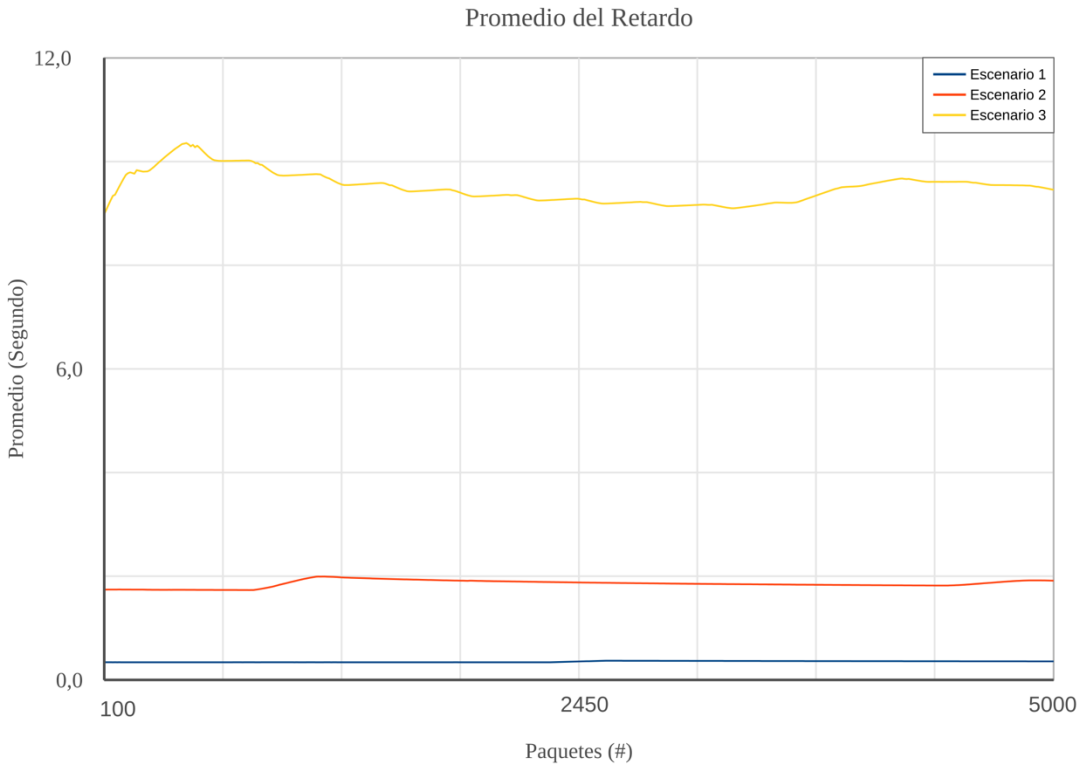


Figura 37 - Promedio del retardo: GetObservation

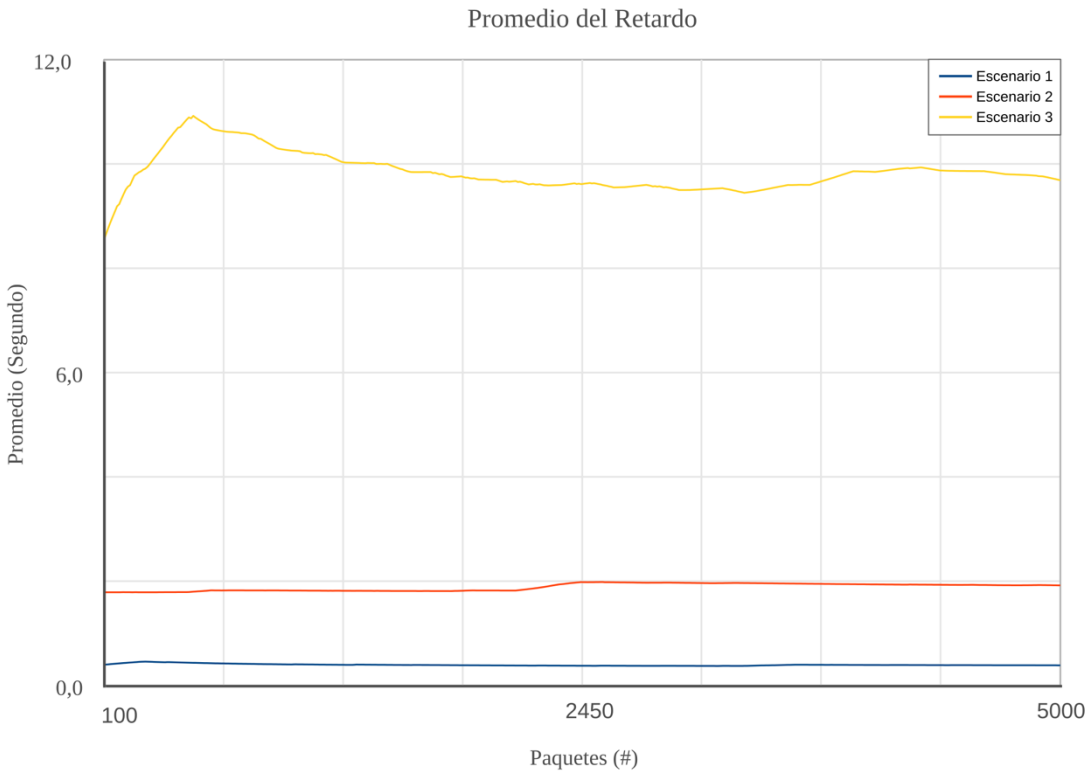


Figura 38 - Promedio del retardo: InsertObservation

3.5.2.e. Mediana del retardo

La mediana del retardo indica el valor central del retardo, medido cuando el valor del retardo se organiza de mayor a menor. Según los gráficos, la mediana del retardo confirma el comportamiento anteriormente descrito, teniendo una gran similitud entre las dos operaciones (Figura 39 - Figura 40) y con valores entre 0 y 10 segundos para los escenarios. Siendo destacable que la implementación presenta una gran estabilidad en el rango de casos de uso evaluados.

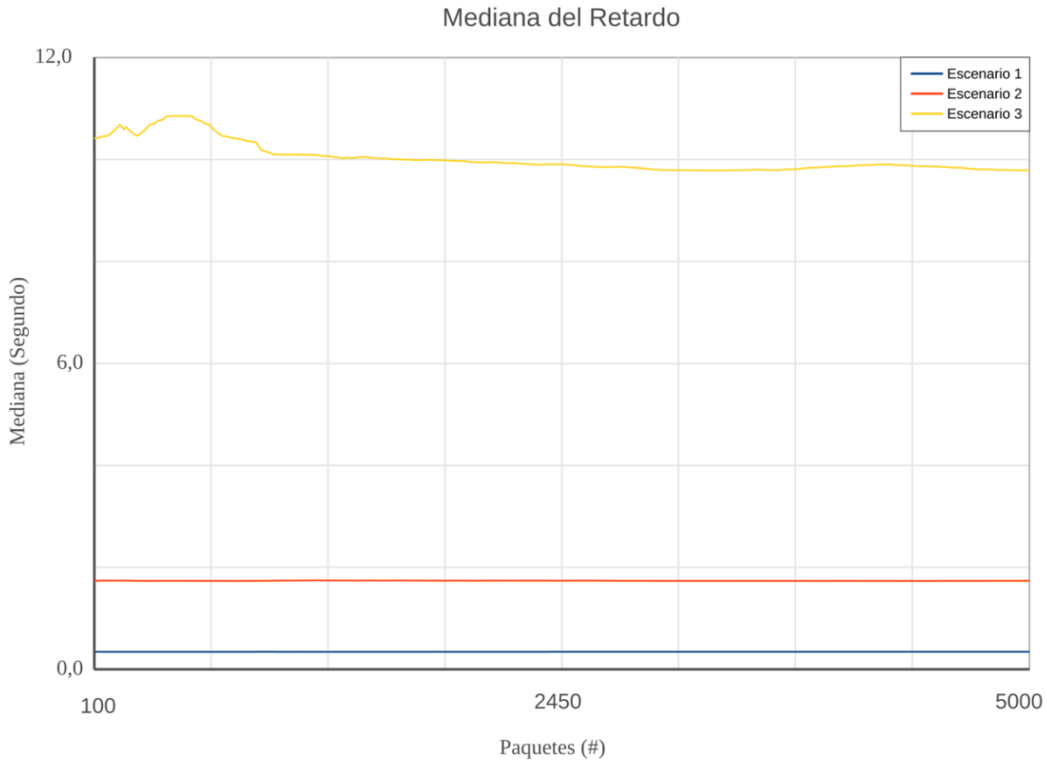


Figura 39 - Mediana del retardo: GetObservation

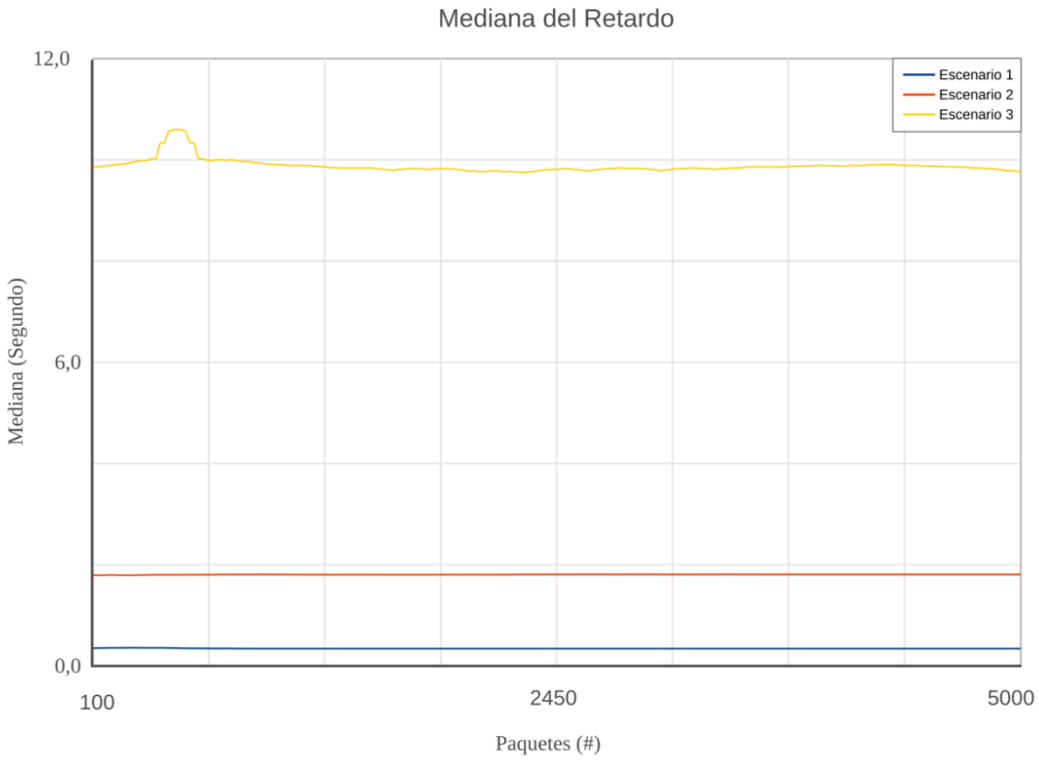


Figura 40 - Mediana del retardo: InsertObservation

3.5.2.f. Desviación estándar del retardo

Junto con las dos mediciones anteriores, la desviación estándar del retardo, permite caracterizar el comportamiento del SOSLite en un ambiente conectado; mostrando (Figura 41 - Figura 42) un gran estabilidad y un tiempo por debajo de los 500 ms cuando se presentan pocos sensores con muchas observaciones; y un comportamiento más errático cuando se cuenta con muchos sensores monitorizando el entorno.

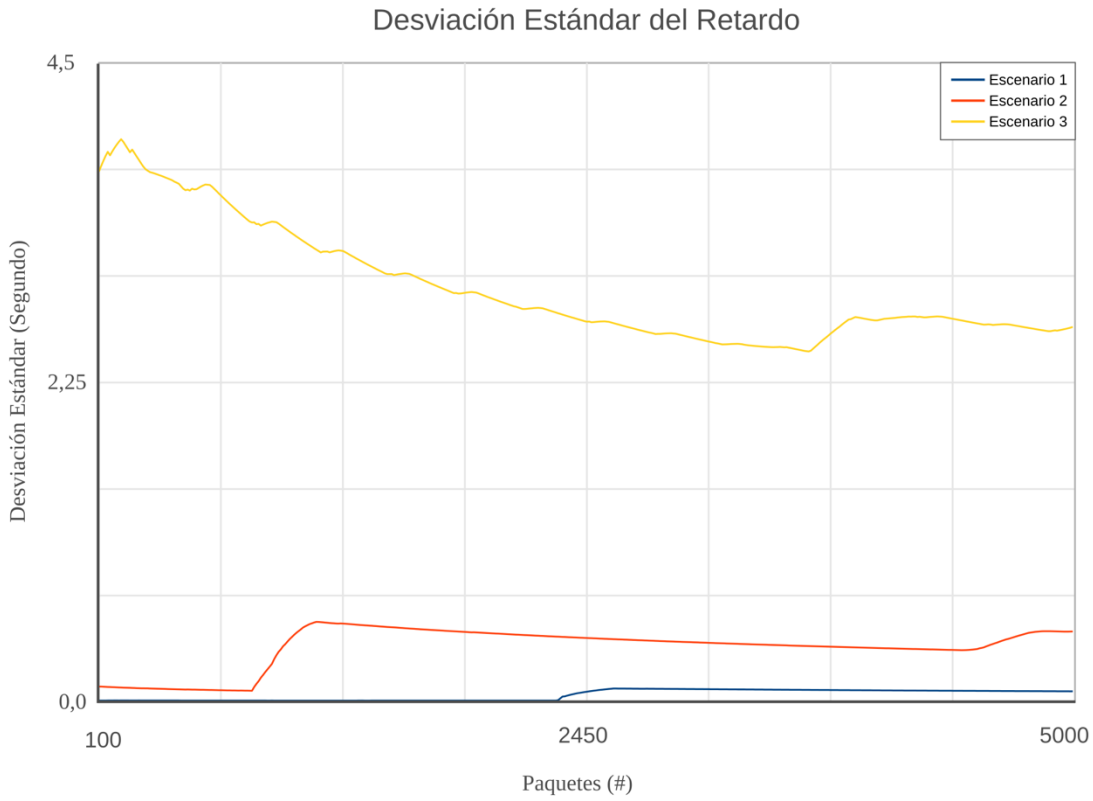


Figura 41 - Desviación estándar del retardo: GetObservation

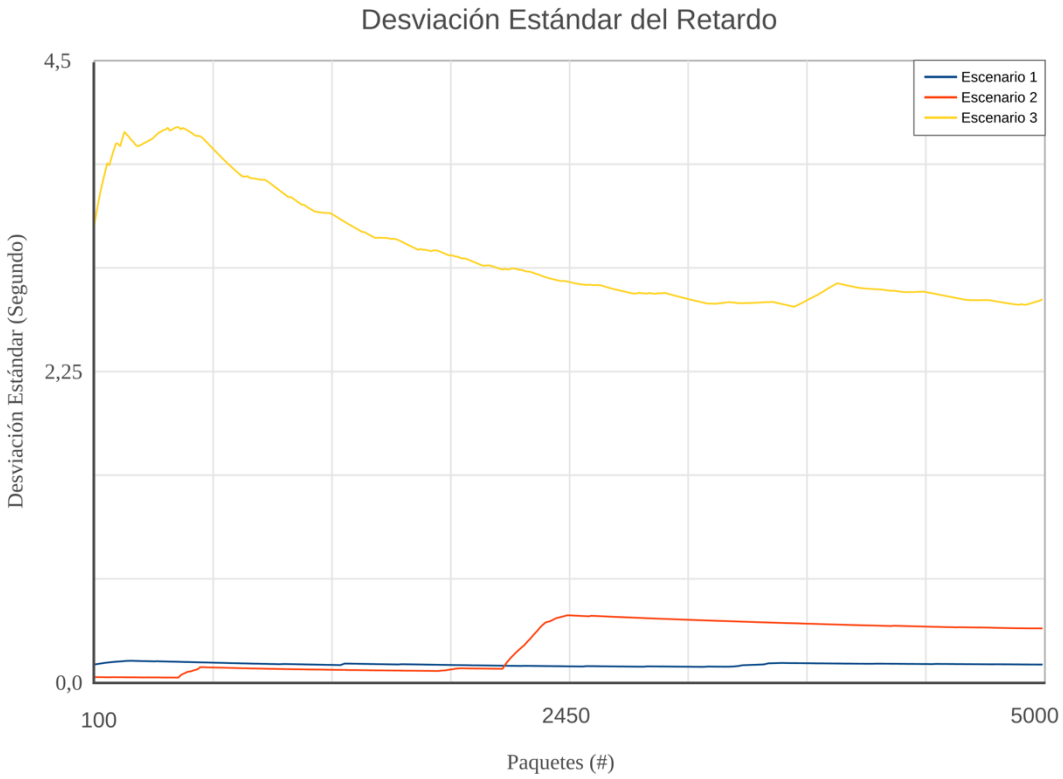


Figura 42 - Desviación estándar del retardo: InsertObservation

3.5.3. Pruebas en micro-instancias en la nube

Por su parte, para las pruebas en micro-instancias en la nube se ha realizado un montaje con: un ordenador para el programa de pruebas de carga, Internet como medio de comunicación y una micro-instancia en la nube (Figura 43). Para esta micro-instancia se ha utilizado el servicio Amazon EC2, el cual es uno de los servicios de CC más empleados en el mundo y que a su vez, brinda un servicio de micro-instancias que se ajusta a los requerimientos de la implementación, donde el SOSLite ha sido desplegado en un instancia t2.micro, la cual ofrece: 1 GB de memoria RAM, 30 GB de disco duro y 1 vCPU (equivalente al 20% de uso de un procesador de 8 núcleos a 2 GHz).

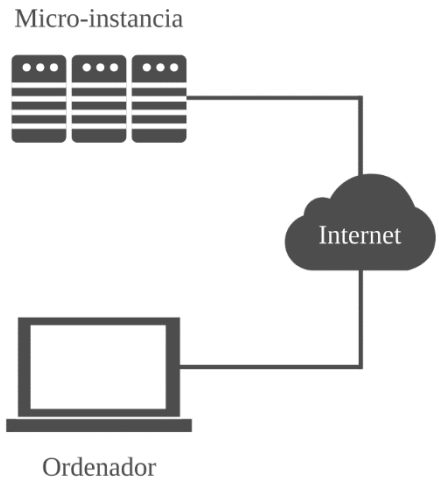


Figura 43 - Montaje de pruebas en micro-instancias en la nube

Tal como en las pruebas en dispositivos limitados, se han graficado los resultados para las operaciones *GetObservation* e *InsertObservatión*, eliminando las primeras 100 medidas que constituyen el periodo transitorio y hacen que las gráficas pierdan detalle.

3.5.3.a. Uso de procesador

El usos porcentual del vCPU en las micro-instancias, es inferior al 50% para las dos operaciones (Figura 44 - Figura 45). Siendo mejor cuando se realizan muchas mediciones con pocos sensores y peor cuando se tienen pocas mediciones y muchos sensores. El hecho de que el consumo sea inferior al 50% se traduce en un gran potencial para emplear las plataformas de CC para centralizar la gestión de datos y metadatos de los CPS.

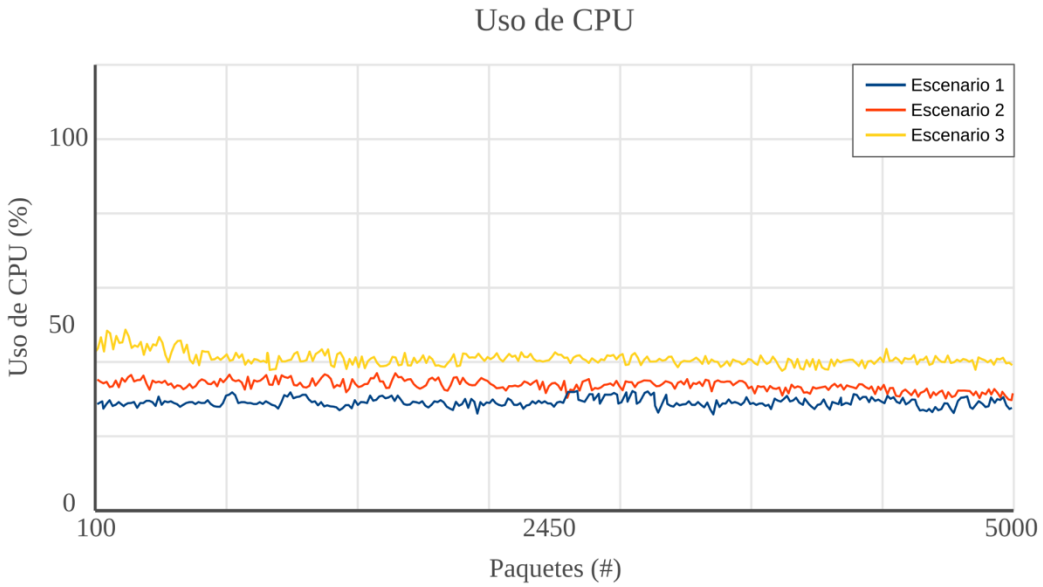


Figura 44 - Uso de procesador: GetObservation

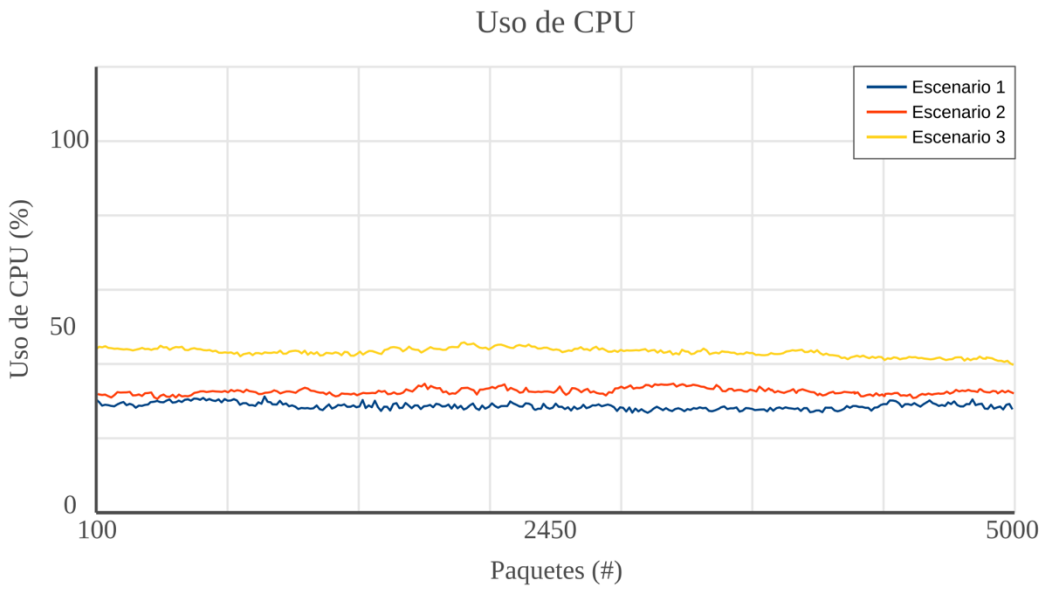


Figura 45 - Uso de procesador: InsertObservation

3.5.3.b. Uso de memoria RAM

En el caso del consumo de memoria RAM, las operaciones *GetObservation* (Figura 46) e *InsertObservation* (Figura 47) presentan un uso ajustado de la misma, estando entre un 20% y un 35% para la primera operación, y entre 30% y un 40% para la segunda operación. Donde se encuentra que con un uso comedido de recursos se puede logran afrontar un rango amplio de situaciones reales usando CC.

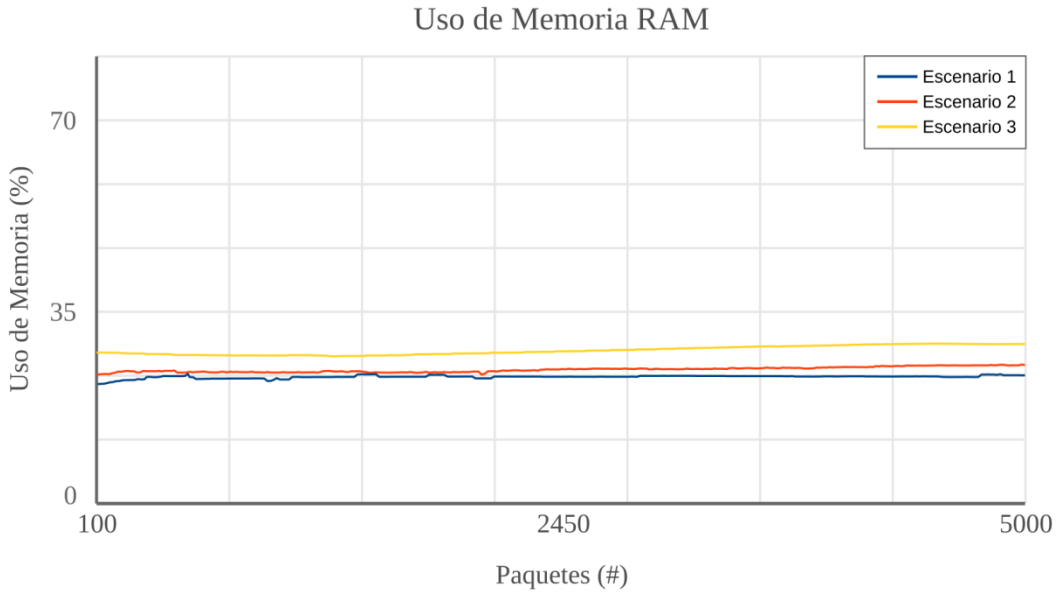


Figura 46 - Uso de memoria: GetObservation

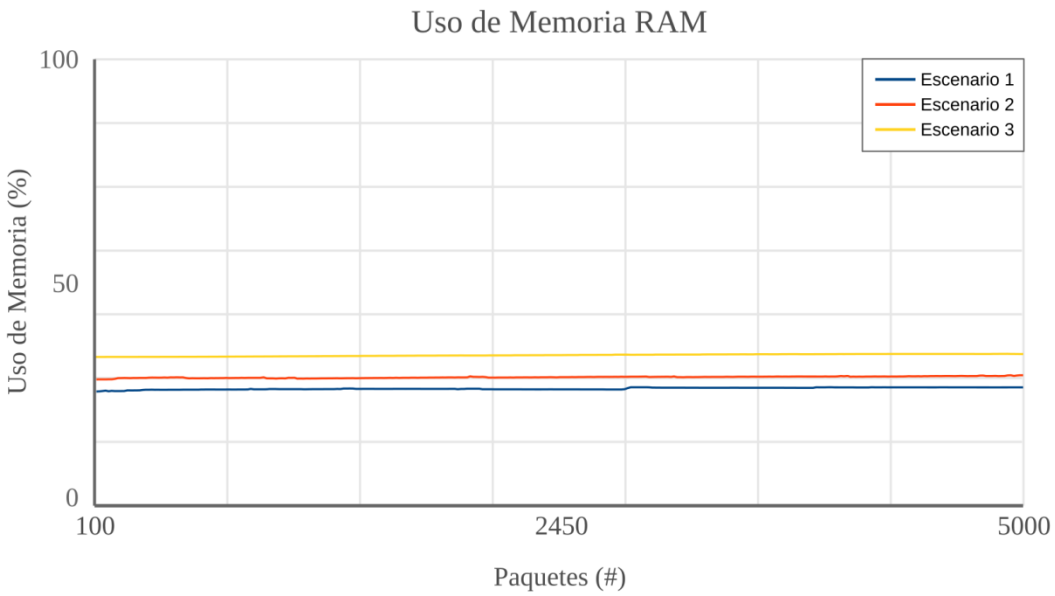


Figura 47 - Uso de memoria: InsertObservation

3.5.3.c. Throughput

Las peticiones por segundo para la operación *GetObservation* (Figura 48) tiene a ser de 2; al igual que para la operación *InsertObservation* (Figura 49). En los dos casos se mantienen bastante constantes durante todo el tiempo de las pruebas; este hecho evidencia que el funcionamiento del SOSLite se mantiene estable en el tiempo; además, indica que se adapta de forma correcta a los casos modelados. De igual forma, permite reconocer que en instancias en la nube se pueden realizar peticiones cada 500 ms manteniendo un rendimiento óptimo.

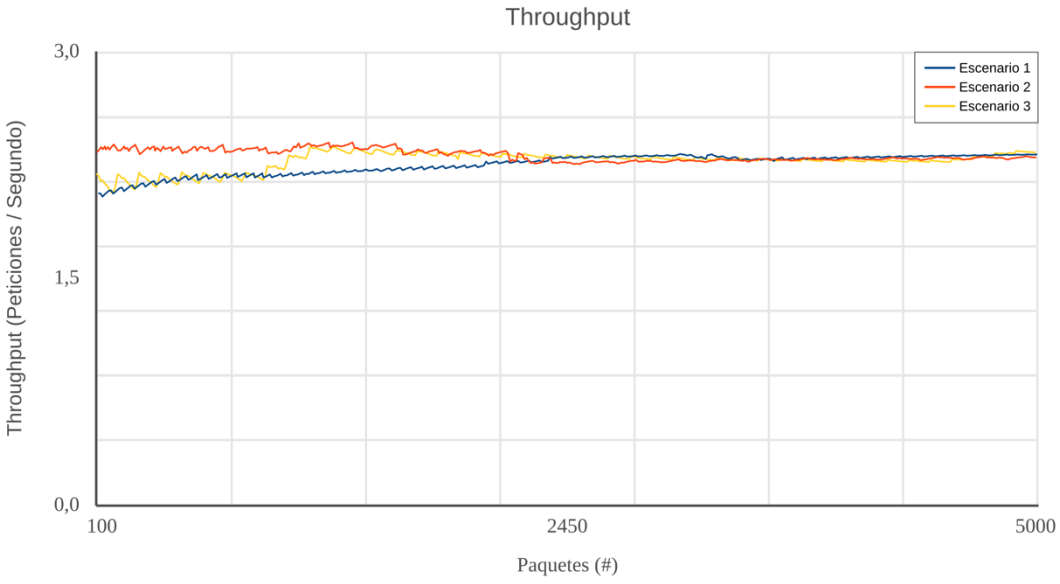


Figura 48 – Throughput: GetObservation

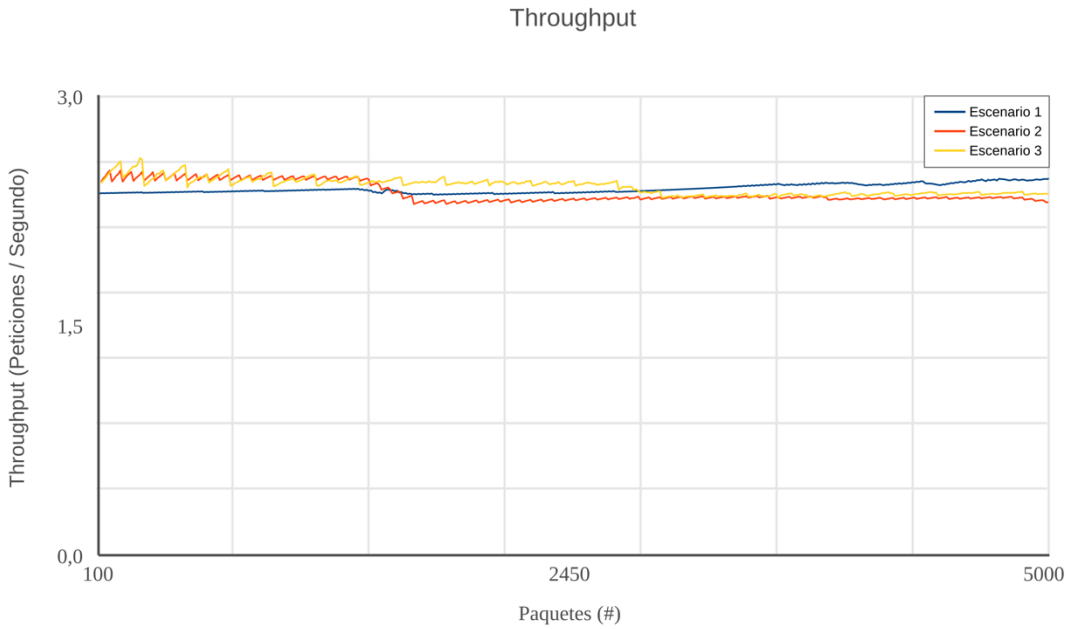


Figura 49 – Throughput: InsertObservation

3.5.3.d. Promedio del retardo

El promedio del retardo muestra un comportamiento estable en las dos operaciones (Figura 50 - Figura 51), sin embargo, cuando se emplean una gran cantidad de sensores, el desempeño se ve mermado (13 segundos) en comparación al uso de pocos sensores. Estos valores pueden ser grandes para algunas aplicaciones con lo que se debe estudiar los casos en los cuales se usara el SOSLite, desplegado como micro-instancia en la nube, para conocer si es oportuno su uso; además, es necesario reconocer que las condiciones de la red pueden ser variables en el tiempo, de forma que, en pruebas de una duración mayor se pueden encontrar variaciones sobre los valores mostrados.

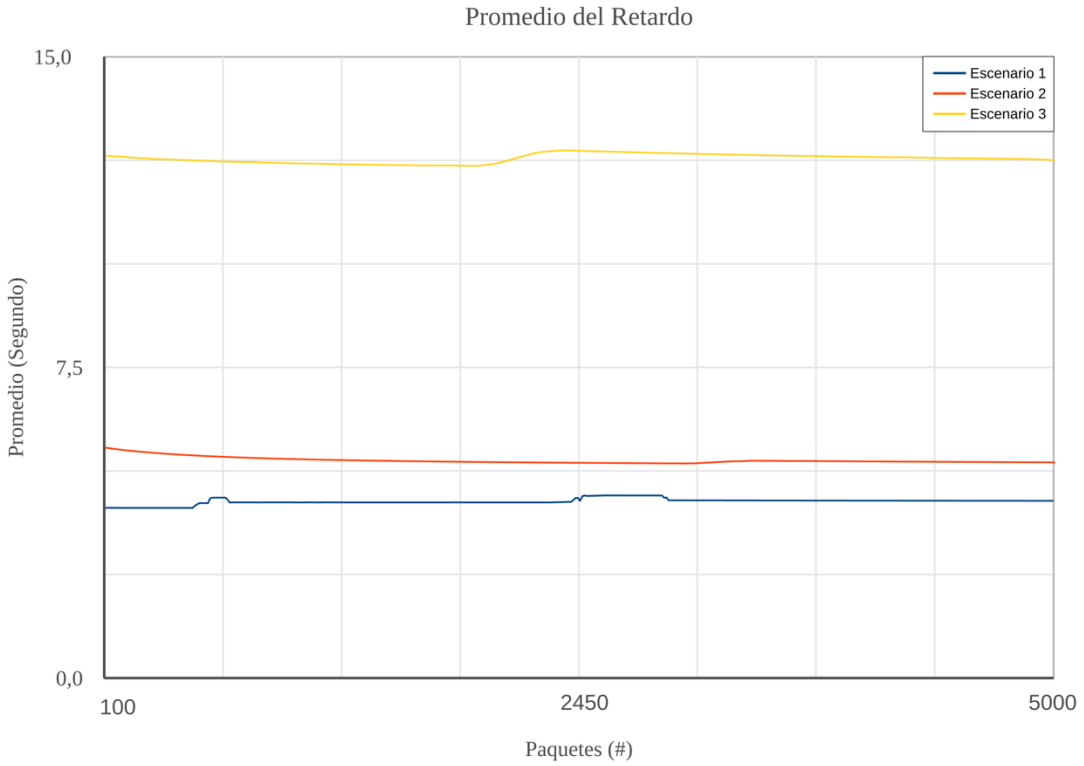


Figura 50 - Promedio del retardo: GetObservation

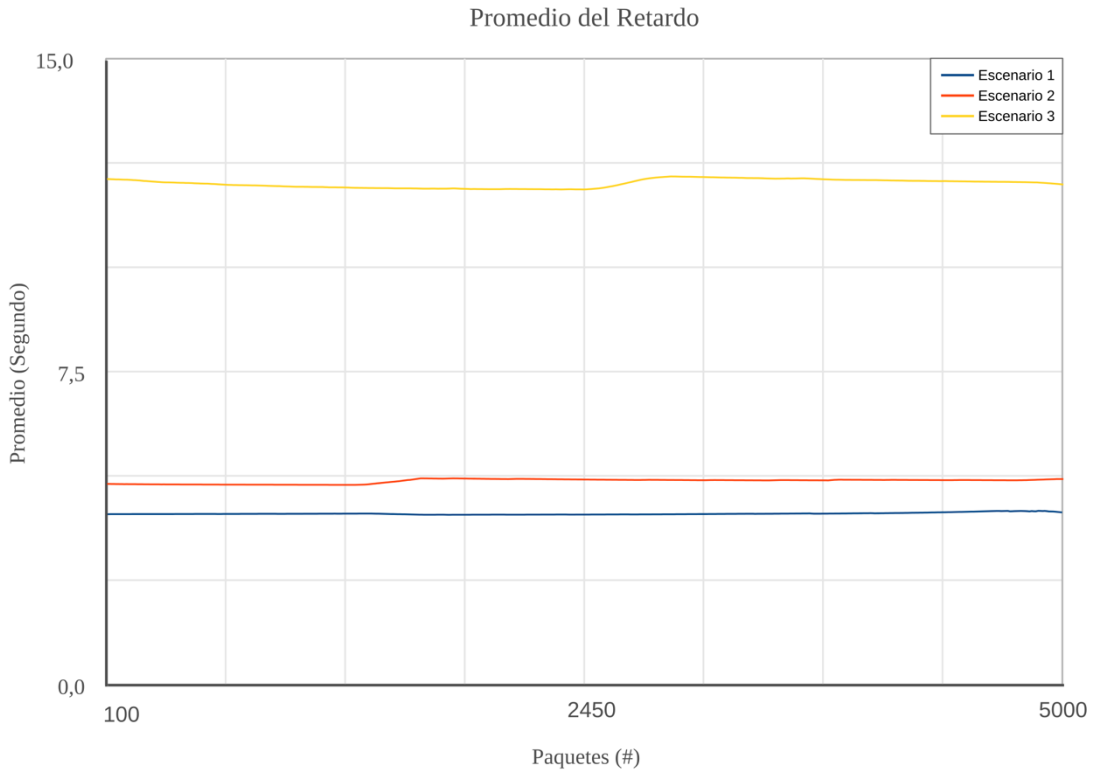


Figura 51 - - Promedio del retardo: InsertObservation

3.5.3.e. Mediana del retardo

La mediana del retardo, medido desde el momento que se genera la petición y hasta que se recibe una respuesta, confirma los datos aportados por el promedio, donde se observa un rendimiento muy superior cuando se tienen pocos sensores realizando muchas observaciones. Además, corrobora el funcionamiento estable para las dos operaciones (Figura 52 - Figura 53).

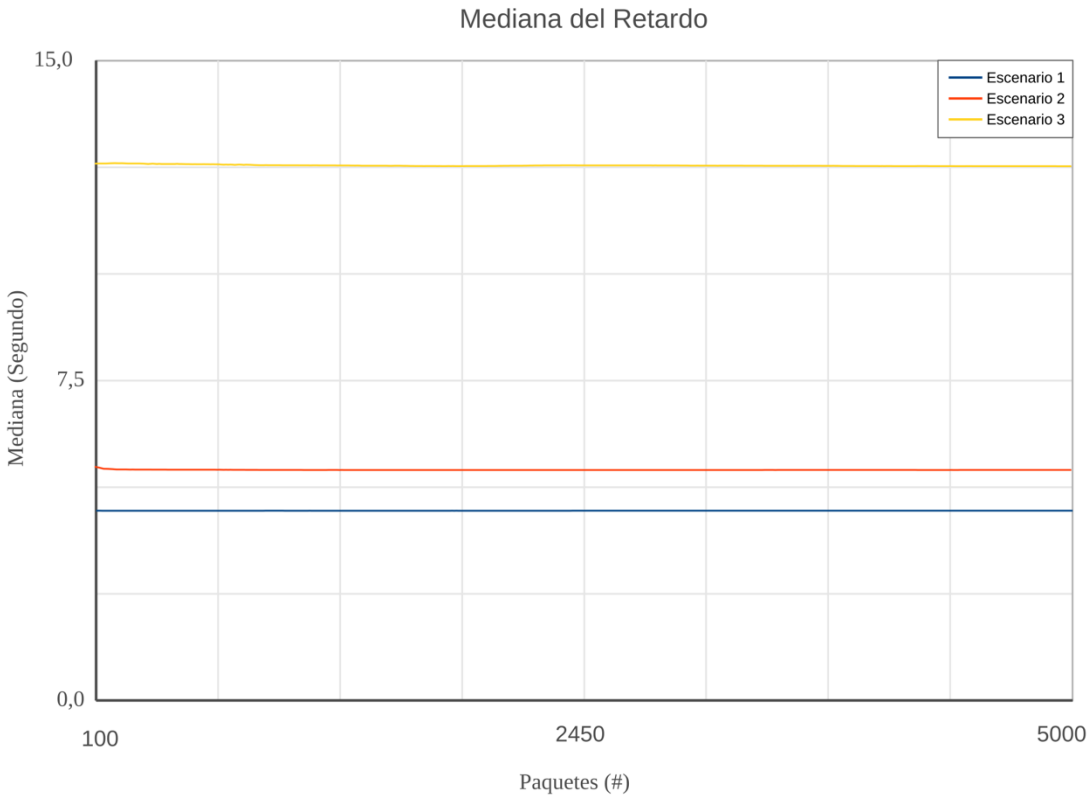


Figura 52 - Mediana del retardo: GetObservation

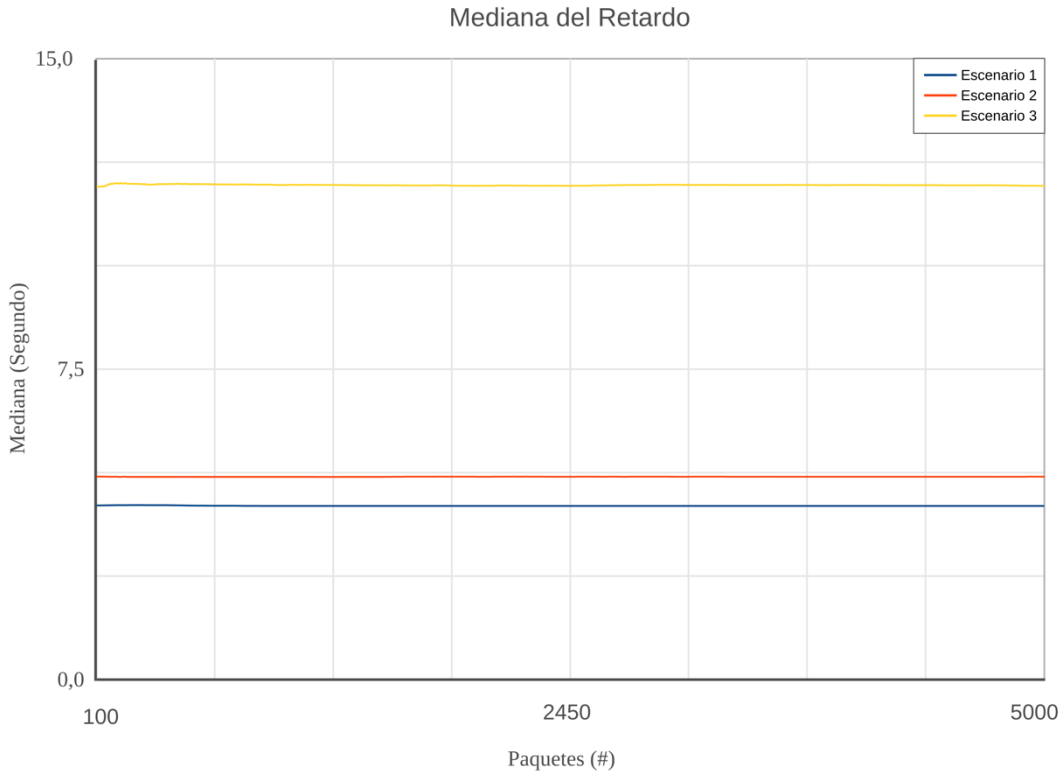


Figura 53 - Mediana del retardo: InsertObservation

3.5.3.f. Desviación estándar del retardo

Finalmente, la desviación estándar del retardo, presenta unos valores excelentes, siendo la variación cercada al segundo para todos los escenarios en las dos operaciones (Figura 54 - Figura 55). Con esto en mente se puede modelar un comportamiento del retardo muy estable entorno a los valores medios, brindando un servicio confiable para la gestión de datos y metadatos en los CPS.

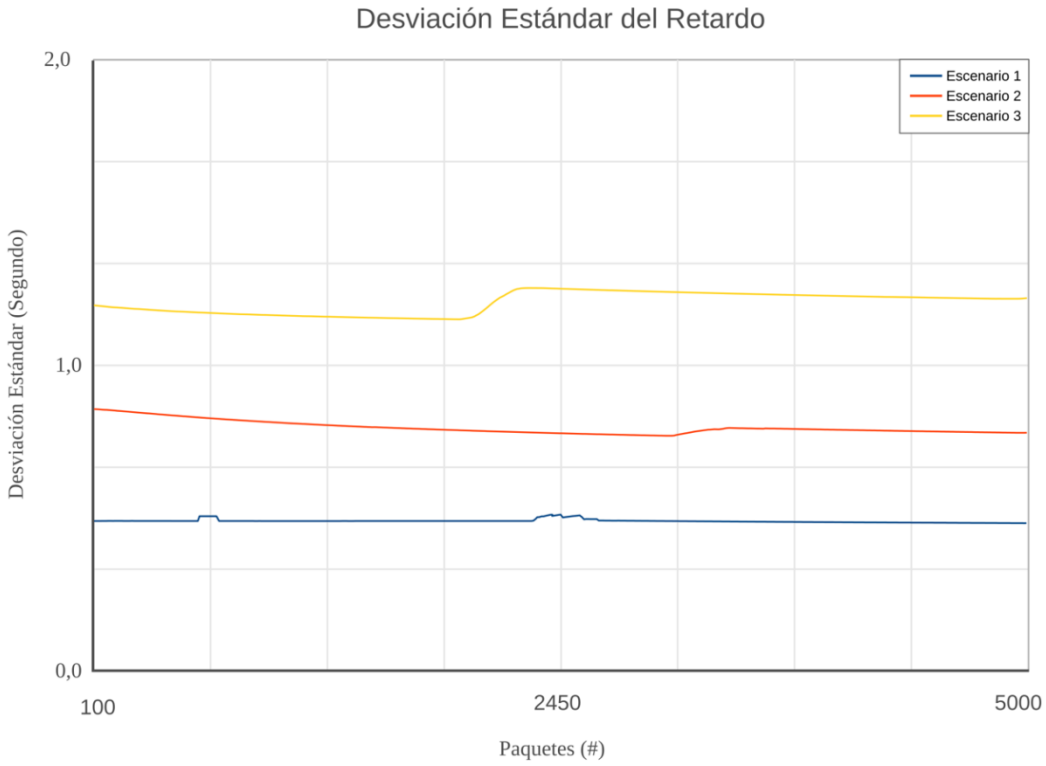


Figura 54 - Desviación estándar del retardo: GetObservation

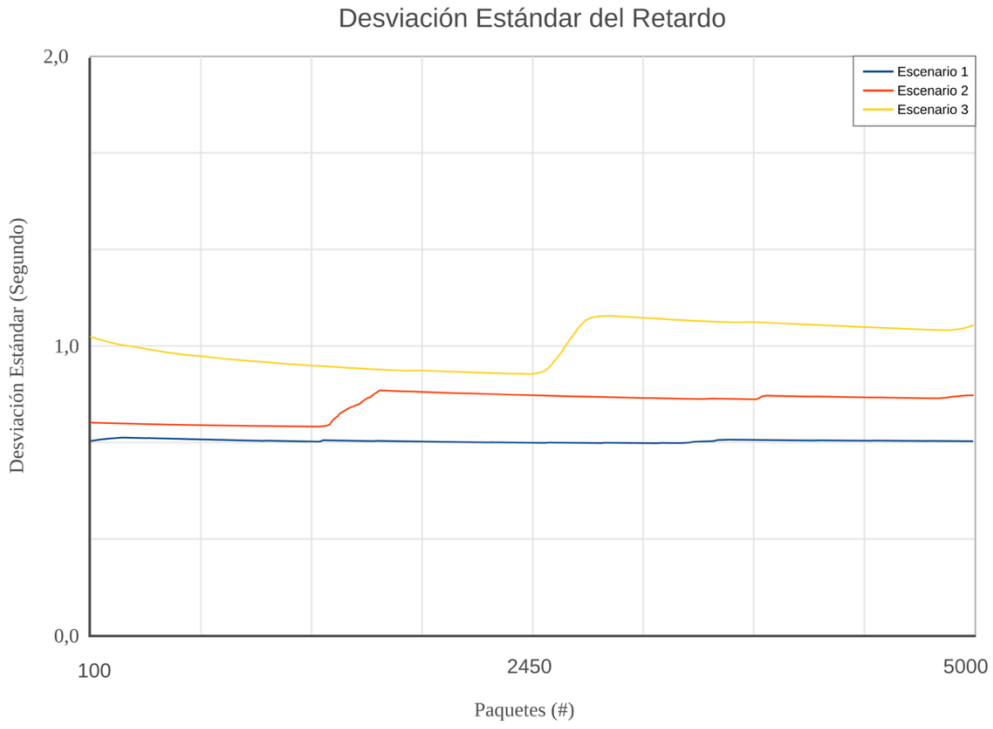


Figura 55 - Desviación estándar del retardo: InsertObservation

3.6. Conclusiones

El SOSLite es una implementación de SOS ligero que permite la gestión de datos y metadatos de sensores y observaciones; enfocada en dar respuesta las necesidades propias de las redes distribuidas y con el objetivo de aprovechar al máximo los recursos computacionales limitados, bien sea al desplegarlo sobre un dispositivo limitado o como una micro-instancia en la nube. Estas características guían el desarrollo propio del sistema y hace que las decisiones de diseño y desarrollo difieran de otras implementaciones de SOS existentes.

Así que con el fin de realizar un desarrollo formal, se ha seguido una metodología de desarrollo iterativa e incremental, tal como se mencionó en la introducción, apoyada en las buenas prácticas de la ingeniera de software. Para esto, se han utilizado herramientas como: el análisis de requerimientos, el diseño basado en UML y el paradigma de programación orientada a objetos.

Con cada iteración se han realizado pruebas para ir comprobando el funcionamiento correcto del prototipo y una vez el prototipo ha sido terminado, se ha realizado un despliegue sobre una Raspberry Pi y como micro-instancia en el AWS C2. Las pruebas incluidas en este capítulo se han realizado sobre los despliegues realizados, utilizando para ello el software de generación de trafico Apache JMeter y midiendo: el uso de procesador, el uso de memoria RAM, el throughput, el promedio del retardo, la mediana del retardo y la desviación estándar del retardo.

Una vez analizados los datos de las pruebas se puede concluir que el SOSLite cumple el propósito con el cual fue creado, manteniendo un uso controlado de recursos computacionales; al tiempo que, mantiene unos valores de retardo y throughput que lo hacen aprovechable en una gran cantidad de casos de uso reales. De igual forma, ha quedado evidente en los análisis de las pruebas que, el SOSLite provee grandes ventajas en un entorno local, haciéndolo ideal para los entornos con procesamiento de SN distribuidos.

Finalmente, con el SOSLite creado y validado, es necesario integrarlo en las SN, la IoT y los CPS. Los siguientes capítulos abordan el papel del SOSLite en estos tres ámbitos, las ventajas que una implementación de estas características ofrece y como ayuda a afrontar el reto de la interoperabilidad.

4. SOSLite en Redes de Sensores (SN)

4.1. Introducción

4.1.1. Presentación del capítulo

SOSLite es una herramienta desarrollada con el fin de facilitar la integración de dispositivos heterogéneos, al tiempo que hace uso de unos recursos de computación limitados. Al ser una implementación de SOS, nace con el fin de ser un repositorio estándar de datos y metadatos sobre sensores y observaciones, la cual es utilizada continuamente dentro de la SN.

Este capítulo profundiza en el uso fundamental de SOSLite dentro de su entorno nativo, las SN, y como se puede aprovechar sus cualidades al desplegarlo sobre dispositivos limitados, FC y CC. Además, se detalla cómo se integra el SOSLite dentro de estas redes, sus usos, sus ventajas frente a una implementación completa y finalmente, se presenta una propuesta de caso de uso, donde se muestra la potencialidad del SOSLite como parte de las SN.

Los resultados permitirán analizar como las SN integradas dentro de la IoT y de los CPS, pueden sacar provecho de las ventajas del SOSLite para lograr mejorar la interoperabilidad posibilitando así, la expansión de la IoT y los CPS en el futuro. De momento, se aborda, el reto de la interoperabilidad dentro de las SN.

4.1.2. Interoperabilidad en las SN y el SOS

Debido a que las SN suelen abarcar una gran cantidad de dispositivos, sean nodos sensores/actuadores o gateways, suele encontrase en ellas una amplia variedad de divergencias tecnológicas, en cuanto a: protocolos de red; protocolos de intercambio de mensajes; y capacidades de procesamiento, almacenamiento, conectividad y fuente de alimentación energética; las cuales deben coexistir y funcionar correctamente a fin de proveer servicios de calidad.

La interoperabilidad, entendida como el medio para alcanzar el fin citado, es un área de investigación con una gran relevancia, y es dentro de esta área, en la cual el SOSLite aporta soluciones para las SN. Siendo esta implementación un vínculo para interconexión de sistemas heterogéneos a nivel semántico, habilitando un lenguaje común, mediante los modelos de datos, y unas normas de comunicación definidas, mediante las interfaces del estándar SOS. Posibilitando, además, el uso de recursos computacionales limitados lo que la hace ideal para entornos distribuidos y despliegues masivos, debido a su bajo coste.

4.2. SOSLite y las SN

4.2.1. Arquitecturas de SN

SOSLite proporciona soporte a la interoperabilidad en el ámbito de las SN y para describir la misma hay que enmarcarlo dentro de una arquitectura de referencia. Una arquitectura de referencia, es un modelo mental, que agrupa características propias de un dominio específico, las SN en este caso, y que ayuda a poner en común una visión sobre el dominio abordado. Pueden existir una gran numero de arquitecturas de referencia y que las mismas, pueden acercarse en mayor o menor medida a ciertos despliegues reales de las SN.

Para el análisis de las SN en esta tesis, se empleará una arquitectura compuesta por cuatro capas (Figura 56): percepción, comunicación, plataforma y aplicación. Esta arquitectura, será un marco de trabajo común para las aproximaciones a IoT y a los CPS, refinándola para adaptarse a estos casos.

Figura 56 - Arquitecturas de referencia para SN

- Capa de percepción: permite medir diferentes eventos que ocurren en el entorno; la componen una gran cantidad de nodos sensores, los cuales son heterogéneos en cuanto a su funcionalidad, temporalidad y capacidades,

- Capa de comunicación: proporciona la infraestructura de comunicación necesaria para la interconexión de dispositivos con el objetivo de compartir información entre sí; asegurando una comunicación fiable y eficiente.

- Capa de plataforma: brinda servicios para la transmisión, almacenamiento y procesamiento de los datos; obteniendo así una independencia de la fuente de datos, al tiempo que permite que las aplicaciones puedan ser construidas de forma rápida e independiente. Esta capa es transversal a los dominios de aplicación, de forma que, puede ser compartida por varios de estos.

- Capa de aplicación: permite abordar diversos dominios de aplicación debido a su flexibilidad. Dado que se encuentra sobre la capa de plataforma puede hacer uso de todos los servicios de esta, interactuando con la información mediante que reside en ella. Dentro de esta capa puede coexistir aplicaciones para ciudades inteligentes, agricultura de precisión, salud, logística, etc.

Esta arquitectura de referencia, integra una visión funcional que facilita entender como el SOSLite afronta el reto de la interoperabilidad. Sin embargo, hace falta esclarecer como esta arquitectura se ve reflejada en los dispositivos que componen las SN y en

especial, revisar las soluciones disponibles para desplegar las capacidades de procesamiento y almacenamiento, sea en aquellas donde se hace uso de un dispositivo de recursos limitados, instancias de FC o instancias de CC. Para tal fin, la siguientes figuras presenta la arquitectura y la segregación de los dispositivos de las SN dentro de esta, considerando los casos en los cuales se utilizan dispositivos limitados (Figura 57), instancias de FC (Figura 58) o instancias de CC (Figura 59).

Figura 57 - Arquitectura de SN con dispositivo limitado

Figura 58 - Arquitectura de SN con FC

Figura 59 - Arquitectura de SN con CC

Como se observa, en la capa de percepción se encuentran los sensores, encargados de monitorizar el entorno. Así mismo, la capa de comunicación incluye los componentes de comunicación que utilizan los sensores para enviar información, al igual que los componentes de comunicación de los dispositivos limitados, los componentes de FC y CC; además en este nivel, se incluyen los dispositivos de conectividad (router, switch, access point, cables, etc.) de la red donde se encuentran los sensores y de la red externa a esta. Por su parte, las capas de plataforma (middleware) y aplicación, comparten los equipos, abarcando los dispositivos limitados, el FC y el CC; en estos, residirá el software que brinda servicios horizontales a los dominios de aplicación, y las

aplicaciones vinculadas a estos.

4.2.2. El SOSLite dentro de una arquitectura de sensores de referencia

Dentro de la arquitectura de referencia para las SN que se ha presentado, el SOSLite forma parte de la capa de plataforma. Siendo un servicio independiente del dominio de aplicación, y por lo tanto, puede ser empleado de forma independiente diferentes casos de uso.

El SOS utilizado en esta capa, se convierte en una solución de interoperabilidad a nivel semántico; permitiendo que exista independencia de los dispositivos de la capa de percepción y comunicación, de los protocolos de comunicación, y de los formatos de mensajes que se utilicen en las capas inferiores. Lo cual conlleva que productores y consumidores, de la información de sensores y observaciones, puedan interactuar a pesar de su heterogeneidad. De igual forma, brinda a las aplicaciones un mecanismo sencillo de consultar y generar datos y metadatos mediante unas interfaces y mensajes estándares.

4.2.3. El SOSLite en las SN sobre dispositivos limitados

Como se ha mencionado con anterioridad las SN pueden usar una variedad de soluciones para desplegar las capacidades de procesamiento y almacenamiento de la información de sensores y observaciones. Para profundizar en los usos y ventajas, que brinda cada solución para el despliegue del SOSLite, esta sección aborda los dispositivos limitados, las instancias de FC, las instancias de CC y un hibrido de instancias *Fog/Cloud*.

El SOSLite desplegado dentro de un dispositivo limitado como parte de una SN cumplirá la función básica de almacenar todos los datos que los nodos sensores generen; además, permitirá la consulta de información por parte de algún consumidor sea este perteneciente a la misma red donde se encuentran los sensores o una red externa a esta, siempre que cuente con las credenciales necesarias.

Lo más usual será que el mismo dispositivo que ejecute el SOSLite también realice tareas de pasarela/gateway, posibilitando el acceso externo a la red de sensores; sin embargo, en casos con mayor demanda de recursos computacionales pueden ser dispositivos separados que se comuniquen mediante una conexión de banda ancha.

Por su parte, los demás servicios de la capa de plataforma que se quieran integrar al sistema, pueden desplegarse en otros dispositivos limitados, estén dentro o fuera de la red donde se genera la información. Siendo el caso más común, que estos se encuentren en la misma red. De forma similar, las aplicaciones de la capa superior de la arquitectura residirán en otros dispositivos con acceso al SOSLite.

Este despliegue de SN con el SOSLite ejecutándose en un dispositivo limitado, es utilizado en casos donde se conoce de antemano las necesidades de recursos computacionales. Es decir, donde se saben la cantidad de sensores que reportaran al SOS, la frecuencia a la cual estos realizaran las mediciones y el tamaño de las mismas. En general, se excluirán los casos de uso de misión crítica, los de tiempo real y los de frecuencia de muestreo variable; con esto, no se limita el hecho de poder agregar nuevos

sensores a la red o condiciones con pequeñas variaciones de la frecuencia de muestreo, pero siempre se ha de considerar que las limitaciones del dispositivo en procesamiento y almacenamiento ponen un máximo a las capacidades mismas de la red.

Un ejemplo real donde esta implementación es válida, son los sistemas automatizados de asistencia ambiental; sea en construcciones, vehículos o algunos invernaderos. En estos sistemas, se permite el ajuste de la temperatura, la humedad y la luminosidad según las condiciones actuales del entorno. Además, las observaciones son menos significativas con el paso del tiempo, hecho que lleva a poderlas descartar, liberando espacio de almacenamiento. De forma adicional, los tiempos de muestreo de un minuto pueden ser más que suficientes para brindar un buen nivel de servicio, mientras que, los tiempos de respuesta en las aplicaciones permiten una holgura de un par minutos o más; debido a esto, la capacidad de procesamiento necesaria es poca. De igual forma, estos sistemas pueden permitir variaciones de la frecuencia de medición, por ejemplo si se detecta que no hay nadie en una habitación pasar de medir la temperatura cada minuto a cada 10 minutos. Finalmente, estos sistemas de confort, suelen estar compuestos por pocos nodos sensores, por ejemplo: un invernadero de 100 m^2 tiene 4 nodos que miden la temperatura, la humedad y la luminosidad en el ambiente, y tiene 10 nodos que miden la humedad en la tierra; este conjunto restringido de sensores ayuda a que las necesidades de ancho de banda, consumo eléctrico y procesamiento sean modestas y por lo tanto ideales para la implementación escogida.

Otro ejemplo en el cual resulta muy útil esta implementación, es el de los sistemas de registro de consumo energético, en los cuales, a diferencia del caso anterior, las observaciones deben mantenerse por varios años; sin embargo, las mediciones se realizan en un solo punto de la instalación eléctrica y una granularidad temporal de los datos de consumo acumulado de 30 minutos suele sobrepasar las necesidades de casos residenciales y empresariales; así un nodo sensor puede sumar el consumo producido en los 30 minutos pasados y reportar este valor al SOSLite para ser almacenado.

Un último ejemplo, son los sistemas de agricultura de precisión sobre aperos, los cuales se encargan de medir condiciones del suelo al momento de realizar la labranza. En estos casos, el SOSLite montado sobre el propio apero, que arrastra el tractor, puede medir el PH del suelo y la posición geográfica de la medición, durante toda la actividad y una vez terminada la jornada, descargar los datos e integrarlos junto con los de los otros aperos para que otros sistemas los puedan utilizar.

De estos tres ejemplos se puede deducir algunas ventajas de la implementación del SOSLite sobre un dispositivo limitado frente a la implementación del estándar en dispositivos tradicionales con mayor capacidad:

- Movilidad: la red de sensores y el SOSLite pueden ser transportadas y mantener su funcionamiento por periodos prolongados de tiempo, sin necesidad de mantener conexión con redes externas.

- Integración: el SOSLite puede ser un punto intermedio que almacene los datos de forma temporal, los cuales pueden ser descargados de forma posterior o consolidados para reportar de forma más compacta.

- Adaptabilidad: en el uso del SOSLite se puede abarcar casos diversos, siempre y

cuando se tenga en cuenta sus limitaciones; de forma que se pueden aprovechar las capacidades de almacenamiento si las frecuencia de muestreo y la cantidad de sensores son pocas, o se pueden aprovechar las capacidades de procesamiento si los datos son efímeros.

- Coste: una de las características más interesantes ya que un dispositivo limitado se puede adquirir por menos de 50€, con lo que fácilmente se pueden realizar despliegues reales con un bajo costo.

- Proximidad: debido a que los equipos limitados son muy económicos es normal encontrar que en esta implementación los sensores y el SOSLite se encuentran a menos de dos saltos en la red de comunicación.

Por su puesto, esta implementación dista de ser perfecta, con lo que tiene algunas limitaciones:

- Itinerancia: debido a que el SOSLite se despliega directamente en el sistema operativo del dispositivo limitado, pasar los datos de un SOS a otro requiere el uso de las operaciones del estándar o de un proceso de copia del contenido de la base de datos y no es tan inmediato como en las otras dos implementaciones.

- Flexibilidad: los recursos de los que se disponen son limitados y en el caso del procesamiento difícilmente actualizables; lo que conlleva a que si el caso de uso donde se está utilizado requiere de mayores recursos, se deba cambiar el dispositivo limitado.

- Rendimiento: aunque los tiempos de respuesta son los más óptimos, las capacidades limitadas llevan a que en algunos entornos/escenarios el rendimiento pueda ser deficiente en cuanto a las necesidades que se tienen.

4.2.4. El SOSLite en las SN distribuidas con FC sobre dispositivo limitado

El SOSLite desplegado como instancia de FC, permite ampliar la utilidad del SOS sobre dispositivos limitados, al reducir las restricciones asociadas a los recursos computacionales vinculados a la implementación realizada. Cuando el SOSLite está sobre *Fog*, puede contar con la capacidad de extender de forma dinámica los recursos que tiene a su disposición y adaptarse así, a entornos/escenarios con un mayor requerimiento de recursos; teniendo siempre en cuenta que aunque su capacidad es mayor al de la implementación anterior no llega a ser ilimitada como en el caso de un despliegue sobre una instancia *Cloud*.

En un despliegue de estas características todos los servicios y aplicaciones pueden residir en el mismo *Fog site*; aunque no hay restricciones para desagregarlos de un modo más distribuido. Así, todos los productores de información reportan al SOSLite, el cual se adaptara a las necesidades del caso específico; mientras que los consumidores, en su mayoría aplicaciones residentes en el mismo *site*, acceden de forma transparente al repositorio de información. Otra posibilidad que se abre con las instancias de FC es que en un mismo *site* coexistan varios servicios de SOS, manteniendo una independencia entre ellos, al tiempo que facilita la interacción eficiente si se llegara a

necesitar.

Con las instancias de FC o de CC se obtienen ventajas adicionales, debido a que estas funcionan sobre VMs. Estas ventajas incluyen: una gran facilidad para hacer copias de seguridad, actualizaciones, migraciones e incluso tener instancias nómadas, que migren de un *site* a otro de forma transparente.

Un primer ejemplo donde la implementación del SOSLite como instancia de FC es ideal, son los sistemas de seguridad. En estos, ante un evento que indica una posible intrusión, las frecuencias de muestreo se aumentan y se activan de forma dinámica sensores. En el caso puntual de la monitorización de una sala de exposición en un museo, cuando un sensor de presencia alerta sobre el ingreso de una persona en la instancia, las cámaras de seguridad empiezan a registrar imágenes y aumenta la frecuencia de muestreo en los sensores de las obras expuestas; ante estos incrementos de las necesidades de recursos computacionales, la instancia puede ir aumentando su capacidades disponibles de almacenamiento y de procesamiento de ser necesario.

Otro ejemplo donde esta implementación se adapta correctamente son los sistemas de producción automática industrial, en los cuales el análisis en tiempo real es de suma importancia. Dada la cercanía entre el lugar donde se generan las mediciones, el lugar donde se almacena y el lugar donde se procesan, los tiempos de retardo en las comunicaciones son muy cortos (menos de 100 ms), si a esto se le une el hecho de que las capacidades de procesamiento son mayores en un *Fog site*, comparado con la implementación sobre dispositivos limitados, se puede utilizar esta implementación para la verificación de la calidad de las piezas producidas, el monitoreo de los niveles de producción o la clasificación automática que sea parte de una línea de producción.

De igual forma, se puede aprovechar esta implementación en un sistema domótico nómada, en el cual cuando una persona cambia su lugar de residencia, la red de sensores permanece anclada a cada una de las casas, pero las mediciones realizadas y las preferencias personales viajan con la persona que hace el cambio. Así, de forma transparente, se realizan las acciones de desvinculación de los sensores anteriores y la inclusión de los nuevos; sin perder el registro de las observaciones pasadas y, si se cuenta con la aplicación, se puede aprovechar este registro histórico para adaptar las condiciones del nuevo lugar a sus preferencias anteriores.

Así, se pueden observar varias de las ventajas que ofrece esta implementación frente a otras de diferentes características:

- Integración: en este caso se realiza a un nivel adicional que el tratado con la implementación en dispositivos limitados; el SOSLite como instancia de FC, permite además, la interacción eficiente con otros SOS, incluyendo los que residen en su mismo site o dentro de un dispositivo limitado.

- Adaptabilidad: extiende esta ventaja de la implementación anterior, permitiéndole abarcar un rango más amplio de casos de uso, al tener la posibilidad de aumentar sus capacidades de almacenamiento y procesamiento.

- Itinerancia: una instancia de *Fog* puede migrar desde un site a otro de forma transparente, también puede viajar hacia un data center para ser almacenada de

forma temporal (en el caso de las migraciones) o definitiva (en las copias de seguridad).

- Flexibilidad: es sin duda alguna una de las mayores ventajas de esta implementación y es que la posibilidad de utilizar recursos bajo demanda posibilita no solo la adaptabilidad (aplicada a los casos de uso) si no la coexistencia en un mismo site de instancias de SOSLite con requerimientos diferentes, permitiendo el aislamiento, al tiempo que, facilita la integración.

- Rendimiento: es otra ventaja sobre la que sobresale esta implementación, debido a que tiene una mayor disponibilidad de recursos computacionales, a los que puede acceder de forma flexible, frente a la implementación anterior, brinda un mayor rendimiento en el procesamiento de los datos y metadatos.

- Proximidad: el tener cercanía al sitio donde se genera y consume la información, brinda un mayor desempeño en términos de retardo que el brindado por una implementación basada en *Cloud*.

Se pueden considerar algunos puntos donde nuestra implementación en comparación con las otras analizadas en esta sección:

- Movilidad: aunque se puede lograr que un *Fog site* sea fácil de transportar, lo más habitual es que resida en un mini-datacenter que requiere de conexión permanente, lo que limita la movilidad.

- Coste: es la mayor deficiencia frente a la implementación en dispositivos limitados, dado que en comparación su costo es superior.

- Rendimiento: a diferencia de las instancias *Cloud*, los recursos computacionales tienen un límite que ninguna instancia del SOS puede superar.

- Proximidad: en este caso los tiempos de respuesta pueden ser ligeramente mayores que en la implementación anterior.

4.2.5. El SOSLite en las SN centralizadas con CC

La tercera implementación del SOSLite en las SN, es cuando se despliega como una instancia de CC. Se puede considerar que esta solución dista del propósito con el que fue inicialmente concebido la implementación, debido a que una aproximación común es tener una sola instancia corriendo en el *Cloud site* de forma que atienda a una gran demanda de solicitudes al SOS; sin embargo, la aproximación que se considera cuándo se despliega el SOSLite como instancia *Cloud* es que la misma esté acompañada por muchas otras instancias del SOS ligero, permitiendo un aislamiento completo entre estas, al tiempo que facilita una interconexión en los casos en los que resulta útil.

Además, mediante la utilización de plataformas de CC se asegura una disponibilidad de recursos virtualmente ilimitada, ideal para aplicaciones de misión crítica y en los casos en los cuales se gestionan grandes cantidades de SN o SN de gran tamaño. En esta implementación los servicios de la capa de plataforma y las aplicaciones de la capa superior de la arquitectura, suelen residir en el mismo proveedor de servicios *Cloud*.

Los sistemas de monitorización de volcanes son un excelente ejemplo en el cual emplear esta implementación, en estos sistemas se suelen tener grandes cantidades de sensores sísmicos ubicados a diferentes profundidades abarcando un área de medición de varios kilómetros; además, se cuenta con sensores de ceniza, GPS, dilatómetros, gravímetros, magnetómetros, inclinómetros, sensores de gases y un conjunto de sensores ópticos que incluyen: cámaras térmicas, cámaras en el infrarrojo cercano y cámaras de control visual. En estos casos, se suele desplegar un SOS para el registro de las observaciones de una determinada área geográfica; de forma que los datos y metadatos se puedan analizar paralelamente entre las diferentes áreas y obtener reportes consolidados.

Otro ejemplo del uso de esta implementación viene de los túneles de viento donde se prueban los coches de fórmula 1; en ellos existe una gran cantidad de sensores dentro y fuera del coche, los cuales generan una cantidad ingente de observaciones, que deben analizarse rápidamente. En estos casos, las capacidades de almacenamiento y procesamiento del CC son aprovechadas para determinar las interacciones físicas entre el coche y el viento. En estos casos se puede destinar una instancia del SOSLite a almacenar los datos de un tipo de sensor determinado; por ejemplo, los de temperatura de cada una de las partes del coche se registrarían en una instancia de SOSLite, mientras que, los de velocidad del viento se almacenarían en otra; esto permite el análisis de las variables independientemente o cruzar varias de estas para deducir comportamientos complejos.

Por su parte, las ciudades inteligentes son otro ejemplo en el cual esta implementación es idónea, dado que en estas ciudades existe una cantidad enorme de sensores y aplicaciones funcionando de forma simultanea; las mismas pueden estar desacopladas o integrarse según la necesidad. Así, una red de sensores de ocupación de las vías de transporte puede reportar los datos de a una instancia *Cloud* donde está desplegado un SOSLite y esta, ser utilizada por una aplicación que muestre el estado de las vías en los paneles, que hace parte del mobiliario de la ciudad; al tiempo, una red de sensores del nivel de llenado de los contenedores de basura, puede ingresar los datos en una instancia de SOS diferente; de forma que una aplicación puede tomar los datos de estas dos instancias del SOSLite y determinar la ruta más eficiente, en términos de consumo energético, para la recogida de basuras por parte de los camiones.

Como se puede determinar existen grandes ventajas en esta implementación, siendo las más relevantes:

- Integración: similar a la implementación con instancias de FC, permite que diversos despliegues de SOS puedan interactuar entre sí.

- Adaptabilidad: es una solución muy versátil que abarca un número de casos de uso enorme y permite que con una agregación de instancias del SOSLite se puedan resolver problemas complejos.

- Itinerancia: como en la implementación anterior, las VMs pueden ser utilizadas para migraciones o respaldos.

- Flexibilidad: extiende esta ventaja de la implementación con instancias *Fog*, al

permitir el uso de recursos sin un límite de crecimiento.

- Rendimiento: es la ventaja más relevante, dado que una instancia de SOSLite desplegada en un *Cloud site* puede tener capacidades ilimitadas de procesamiento y almacenamiento.

Sin embargo, es necesario puntualizar algunos puntos en los cuales esta implementación es menos favorable que las anteriores:

- Movilidad: las instancias *Cloud* están vinculadas a centros de datos (*datacenters*) que se encuentran anclados en algún emplazamiento, los mismos no suelen ser portátiles, aunque existen versiones en contenedores o que funcionan en barcazas.

- Coste: es sin duda la implementación de mayor costo, de las vistas hasta el momento, en especial si se la compara con los dispositivos limitados donde las inversiones se amortizan por más de una año.

- Proximidad: en comparación con las otras dos implementaciones, el SOSLite como instancia de *Cloud* suele encontrarse más alejada, influyendo esto en los tiempos de retardo.

4.2.6. El SOSLite en las SN hibridas CC/FC

La última implementación del SOSLite sobre SN es una mezcla de las dos anteriores, es decir soportadas por entornos *Fog* y *Cloud*, creando una jerarquía que responda a una mayor cantidad de casos de uso y permita una gran versatilidad en el consumo de recursos computacionales. Dentro de esta implementación las capas de plataforma y aplicación se pueden adaptar a las necesidades, desplegándose de forma distribuida o especializando instancias para responder a una de las capas.

Por su parte, la forma más habitual de desplegar el SOSLite en esta implementación, es mantener pocas instancias en el *Fog site* las cuales atienden a soluciones específicas, como el control de condiciones ambientales o la monitorización del consumo energético, y utilizar un gran número de instancias en el *Cloud site*, las cuales incorporen los registros de varias instancias *Fog*, atendiendo a un criterio de proximidad geográfica, como todas las mediciones en una ciudad, o a un criterio de solución, como la calidad de aire en los hogares monitorizados.

Un primer ejemplo del uso de esta implementación, se encuentra en las soluciones de agricultura de precisión para cultivos de caña de azúcar; en estas, se monitorizan grandes extensiones de cultivos con un nivel de precisión alta; para esto se utilizan estaciones meteorológicas diseminadas por el terreno, drones que sobrevuelan el terreno, equipos de medición sobre aperos, equipos de medición en las máquinas de corte, equipos de medición en las máquinas de abonado y control de plagas, entre otros. Los cultivos se dividen en parcelas que pueden ser atendida por una cuadrilla de operarios; cada parcela tiene asignados un centro de operaciones y un conjunto de maquinaria. En los centro de operación se cuenta con al menos un servidor, de características modestas, encargados de almacenar y procesar la información recolectada por la maquinaria y las estaciones meteorológicas; además, todos los centros

de operación se enlaza a la fábrica de procesamiento que incluye también las oficinas administrativas. El uso de esta implementación, permite que en cada centro de operación opere varias instancias del SOSLite (condiciones ambientales, producción, fertilización y control de plagas), mientras que en instancias del Amazon Web Services utiliza un SOS por cada parcela y se despliegan las aplicaciones utilizadas en las oficinas administrativas y la planta de procesamiento.

Otro ejemplo del uso de esta implementación son los edificios inteligentes, como "The Edge" (las oficinas de la empresa Deloitte en Ámsterdam), que cuenta con 30.000 sensores funcionando en la actualidad. En este edificio el SOSLite puede desplegarse para cada uno de los pisos como una instancia de FC, mientras que las aplicaciones que analizan y presentan los datos residen en el *Cloud site* privado de la compañía.

Un último ejemplo lo constituyen los sistemas de eSalud distribuidos, en los cuales la salud del paciente es monitorizada cuando se encuentra en casa o en el hospital cuando requiere estar internado por algún tiempo. En estos casos, un pequeño *Fog site* se envía a la casa del paciente crónico, junto con todo el equipo médico de monitoreo y una red de sensores ambientales para monitorizar la casa; en este *Fog site* reside una única instancia del SOSLite, la cual periódicamente realiza una copia incremental en el *Cloud site* del hospital; de esta forma si un medico desea ver las constantes vitales del paciente durante los últimos días, puede acceder a la copia local de forma rápida.

Esta implementación tiene grandes ventajas que hereda de las dos anteriores: integración, adaptabilidad, itinerancia, flexibilidad, rendimiento y proximidad. Sin embargo, su baja movilidad y alto costo, pueden hacer que esta implementación no sea la ideal para abordar una solución.

4.3. Casos de uso

4.3.1. Resumen

Para profundizar en los posibles usos que puede tener el SOSLite dentro de las SN, en esta sección se detallaran casos de uso relacionados, los cuales presentan el despliegue del SOSLite en: un dispositivo limitado, en una instancia de FC y en una instancia de CC. Además, al integrar los tres casos de uso, se puede tener una visión de cómo el SOSLite, es a su vez, un punto de integración entre diversas SN y un mecanismo de interoperabilidad.

Así, el primer caso de uso, exhibe un autobús intermunicipal monitorizado, el cual tiene una red de sensores internos y externos, que se suman a los sensores específicos que integra el bus de fábrica y que son accesibles mediante el protocolo EOBD (On Board Diagnostics). En este caso, el SOSLite se desplegara sobre un dispositivo limitado que estará a bordo del autobús.

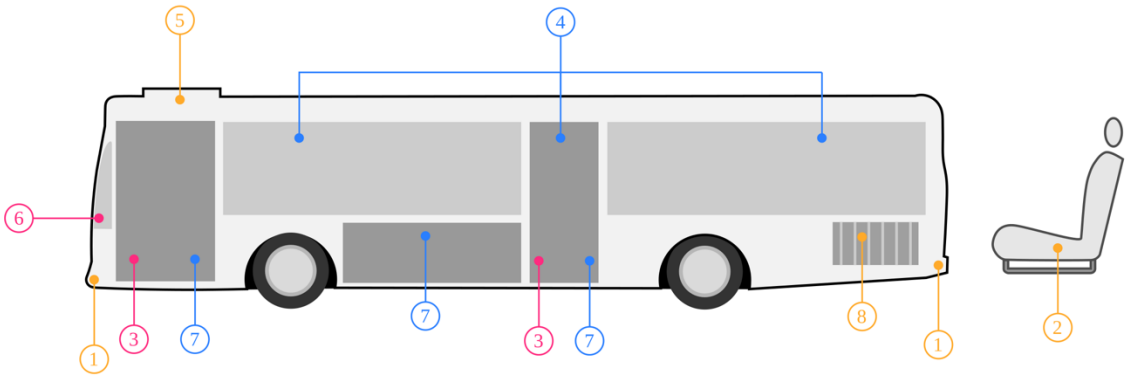
El segundo caso de uso, se centra en una estación de autobuses automatizada, la cual interactúa con los buses que entran en ella, obteniendo una mejora en la eficiencia de los servicios que la estación ofrece. Además, la estación contara con un mini-data center donde el SOSLite se desplegara como instancias de FC, siendo cada una de estas una agrupación los sensores y observaciones según el sistema del que se encarga: acceso, parking, ocupación de bahías, etc.

El último caso de uso, presenta el funcionamiento integrado de varios nodos de transferencia de transporte de personas, los nodos de transferencia son comunes en las grandes ciudades y permite mantener una red de transporte eficiente y descentralizada, que evita el transito innecesario de autobuses por las calles saturadas de las ciudades. En este caso, se opta por el uso de un *Cloud site* privado, que pertenece al ministerio de transporte y que es gestionado por la administración municipal de transporte; en este *Cloud site*, se desplegaran diversas instancias del SOSLite, cada una de ellas corresponderá a uno de los nodos de transporte.

4.3.2. El bus intermunicipal

4.3.2.a. Introducción

Los autobuses monitorizados existen desde hace varios años (véanse las Telematics Gateway Unit); sin embargo, el seguimiento que se hace de estos, suele estar limitado a las condiciones de funcionamiento del vehículo y en algunos casos, se les suma un sistema de localización y movimiento (GPS, giroscopio, acelerómetro). La propuesta de autobús municipal monitorizado del proyecto SOSoB (SOSLite on Board), agrega mediciones de parámetros dentro y fuera del vehículo. Con este fin, se ha ideado una red de sensores que se integra en los autobuses (Figura 60) y que está especialmente ideada para proporcionar valor agregado a los transportes intermunicipales.



1. Sensores de proximidad
2. Sensores piezoeléctricos
3. Sensor de barrera laser
4. Puntos: temperatura, humedad, iluminación, nivel de ruido y calidad de aire
5. Estación: temperatura, humedad, presión barométrica, iluminación y visibilidad
6. Posición/movimiento: GPS, acelerómetro, giroscopio, altitud y brújula magnética
7. Sensores reed switch
8. Conexión EOBD

Figura 60 - Bus intermunicipal inteligente

4.3.2.b. Sensores

Una gran gama de sensores son necesarios para conocer con mayor precisión las condiciones del autobús intermunicipal del proyecto SOSoB, estos incluyen:

- Asistencia en conducción: con sensores de proximidad en los parachoques se puede determinar los niveles de distancia correctos entre los coches que anteceden y preceden el vehículo.

- Nivel de ocupación: mediante sensores piezoeléctricos instalados en cada uno de los asientos, se determina el nivel de ocupación del autobús; además, se emplea un contador de personas mediante sensor de barrera láser en las puertas de acceso a pasajeros.

- Condiciones ambientales internas: utilizando tres puntos de medición con sensores de temperatura, humedad, iluminación, nivel de ruido y calidad de aire (gas, butano, propano, metano, humo, amoniaco, sulfuros y bencenos).

- Condiciones ambientales externas: con una estación instalada en el techo del bus se mide: temperatura, humedad, presión barométrica, iluminación y visibilidad (niebla, neblina, humo, lluvia, nieve, hielo granulado y llovizna).

- Posición y movimiento: el autobús se localiza mediante GPS; mientras que el movimiento del mismo, viene determinado por el acelerómetro, el giroscopio, el sensor de altitud y la brújula magnética.

- Seguridad: en este caso se monitoriza la apertura de las puertas de acceso a pasajeros y las puertas de los comportamientos de equipaje mediante sensores reed switch (interruptores magnéticos).

- EOBD: además de los sensores ya descritos el sistema integra las mediciones que los sensores del vehículo realiza y pone a disposición mediante EOBD, conociendo características propias de la mecánica y electrónica del autobús.

4.3.2.c. Comunicaciones

Con la gran cantidad de sensores que se ha desplegado en el bus, es necesario contar con unas comunicaciones robustas y estándares. Con este fin, los sensores de posición y movimiento están conectados mediante puerto GPIO (General Purpose Input/Output) utilizando I^2C (Inter-Integrated Circuit); de igual forma, se conecta la estación para la medición de condiciones ambientales externas, los sensores reed switch de seguridad, los sensores de proximidad de los parachoques y los sensores de barrera láser de las puertas de acceso a pasajeros.

Por su parte, la conexión de a las tres estaciones de condiciones ambientales internas y al puerto EOBD se realizan mediante bluetooth 4.0. Mientras que, la comunicación de los sensores piezoeléctricos de los asientos, se realiza en dos pasos: en una primera instancia se conectan los sensores, en grupos de 30, a un microcontrolador mediante GPIO/ I^2C y en segunda instancia, el microcontrolador se comunica con el dispositivo limitado haciendo uso de BLE (Bluetooth low energy).

Existe un par de conexiones adicionales que permite al dispositivo limitado comunicarse fuera del autobús: la primera de ellas mediante redes móviles celulares y la segunda utilizando el protocolo 802.11. Estas conexiones son útiles para reportar a una central cuando el vehículo está en carretera o cuando está en un lugar que le provea conexión de área local, o para realizar comunicación vehículo/vehículo o vehículo/carretera.

4.3.2.d. SOSLite

El dispositivo limitado despliega el SOSLite utilizando un sistema operativo basado en GNU/Linux, un servidor NGINX y una base de datos MongoDB. Este SOS ligero registra los datos de todos los sensores que están vinculados con el autobús y permite la consulta de la misma desde otras aplicaciones; por cada uno de los autobuses existe una instancia de SOSLite.

4.3.2.e. Posibles usos

Ante la competencia que se ha generado en la actualidad desde los servicios de economía colaborativa que proveen el servicio de transporte intermunicipal a un menor precio, las compañías de transporte deben responder mediante los valores agregados y los cambios en las tarifas. La propuesta del proyecto SOSoB busca mejorar la eficiencia para disminuir costos y asegurar prestaciones adicionales de seguridad en carretera y un mayor confort a los pasajeros.

El primer uso posible dentro del vehículo es la detección de condiciones anómalas y la activación de alarmas, bien sea en presencia de humo o gases nocivos, identificación de un mal funcionamiento grave o apertura no autorizada de las puertas. Con esta aplicación el conductor podrá actuar rápidamente y mejorar así la seguridad en carretera.

Un segundo uso es asistencia al conductor, sin llegar a ser un sistema automático de conducción, se proveen servicios básicos de asistencia como: indicadores de

modificación de velocidad según el trayecto de carretera que se está transitando y las condiciones ambientales del entorno, advertencia de posible choque posterior, señales para la modificación de parámetros de conducción para el ahorro de combustible y cambio de luces o activación de faros automática según las condiciones ambientales. En este área se puede incluir la comunicación vehículo/vehículo, en la cual, se reporta a otros vehículos el estado de la carretera o el entorno, de forma que se muestren recomendaciones para el manejo en el autobús receptor.

Finalmente, un tercer uso es el aumento en el confort para los pasajeros dentro del vehículo, modificando la configuración del sistema de aire acondicionado y el de entretenimiento a bordo. Además, se brinda un suministro superior de información a los pasajeros, indicando en sus móviles: el tiempo de viaje, el tiempo estimado de llegada y las condiciones ambientales internas y externas del autobús.

4.3.3. La estación de autobuses

4.3.3.a. Introducción

Las estaciones de autobuses, también denominadas terminales de transporte intermunicipal, son núcleos que proveen servicios a las compañías de autobuses y a las personas que desean transportarse. Los servicios a las compañías de autobuses incluye la coordinación, el control y la supervisión operativa, mediante la cual se gestiona los procedimientos de: control de ingreso, plataforma de descenso de pasajeros, plataforma de paquetes y encomiendas, parqueaderos operacionales, entrada a la plataforma de ascenso, plataforma de ascenso de pasajeros, control de salida, módulo de excretas y fallas mecánicas.

Para proveer en puente en la automatización de estos procedimientos y lograr así, una mejora en la prestación del servicio de la estación de autobuses hacia las compañías de transporte, se propone el proyecto iT-Hub (intelligent Transport Hub); el cual, utiliza el SOSLite como repositorio de datos y metadatos de sensores y observaciones, utilizando para ello, un despliegue del SOS como instancia de FC. Las instancias *Fog* residirán en un *mini-datacenter* anclado en la propia estación de autobuses, la red de sensores desplegados en esta se detalla en la siguiente figura (Figura 61).

Nota: Para el funcionamiento ideal del proyecto se requiere que todos los autobuses que utilizan la terminal se encuentren monitorizados, como los provistos por el proyecto SOSoB, aunque un funcionamiento básico es válido haciendo uso de *tags* de radio frecuencia.

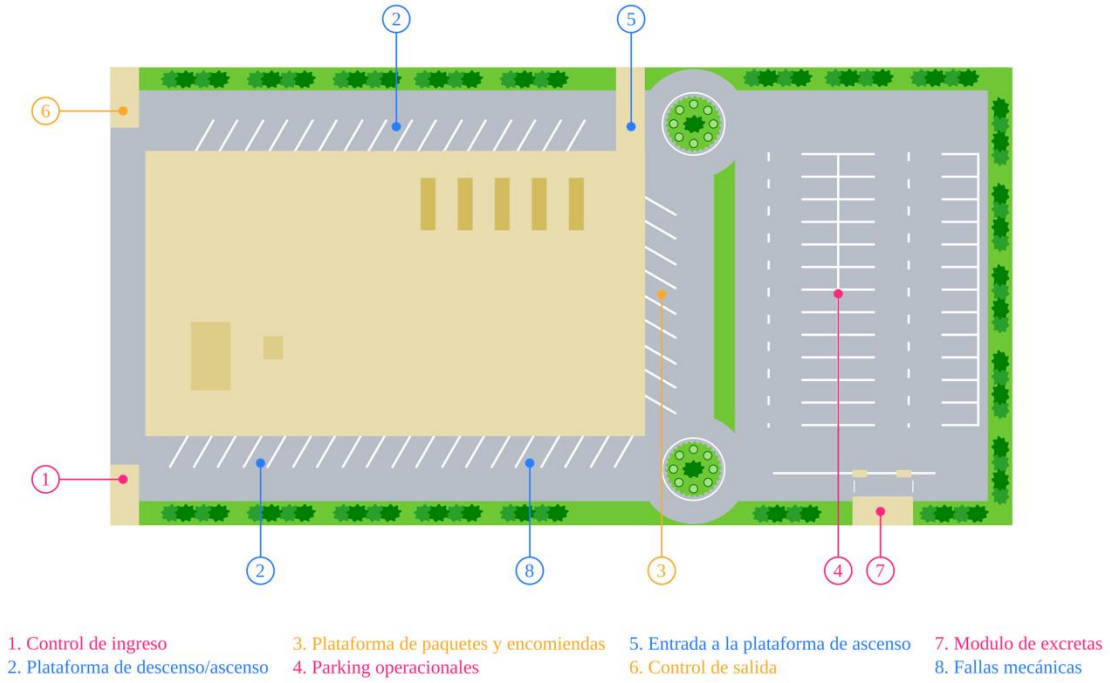


Figura 61 - Estación de autobuses inteligente

4.3.3.b. Sensores

- Control de ingreso: este proceso requiere la identificación del autobús intermunicipal que está arribando a la terminal, para determinar de si se encuentra autorizada su entrada a la estación y entregar un itinerario del tiempo de permanencia del vehículo en las instalaciones, generalmente un bus no puede permanecer más de 2 horas en la estación con la excepción de las entradas al taller mecánico. Para la identificación, se utiliza un sensor de lectura de RFID, que determina la identidad del autobús utilizando una etiqueta pegada en la parte superior del mismo; el sistema integra una segunda opción de identificación mediante una cámara de vídeo que detecta el número de placa. A partir del punto de entrada se brinda conexión mediante el protocolo 802.11 al autobús, a fin de que pueda reportar la información almacenada por el SOS de abordo, misma que es útil en este proceso para programar revisiones o reparaciones en los talleres y agregarlas al itinerario y para reportar el número de pasajeros que han sido transportados.

- Plataforma de descenso/ascenso de pasajeros: en muchas estaciones las bahías de descenso y ascenso de pasajeros están separadas. Para la operación de estas plataformas, se requiere registrar el momento de entrada y el tiempo de permanencia del auto bus en la bahía, esto se realiza por medio de un lector RFID, que se encarga de identificar el autobús, y dos sensores de barrera láser para determinar si la bahía está ocupada.

- Plataforma de paquetes y encomiendas: al igual que las bahías de descenso/ascenso de pasajeros las destinadas a la descarga/carga de paquetes y encomiendas utilizan sensores de RFID y sensores de barrera láser.

- Parking operacionales: estos espacios cuentan con los mismos sensores que las bahías, lector RFID y sensores de barrera láser, en este caso se utilizan para saber con antelación cuales se encuentran disponibles para asignación y si el autobús se utilizado el espacio asignado.

- Entrada a la plataforma de ascenso: tal como en el control de ingreso, cuenta el lector de RFID y una cámara de vídeo por cada una de las entradas. Este proceso determina si el vehículo está autorizado a pasar al espacio de plataformas de ascenso de pasajeros (en aquellas estaciones en donde se separan de las plataformas de descenso).

- Control de salida: funciona de forma análoga al control de ingreso, identificando el autobús que está saliendo de la estación mediante RFID y cámaras de vídeo. Si el bus cuenta con la funcionalidad, se reporta de forma automática la cantidad de pasajeros que están siendo transportados.

- Módulo de excretas: los vehículos que cuentan con baño abordo pueden hacer uso de este módulo el cual abre el compartimiento para los desechos una vez el autobús está en posición; para ello utiliza sensores de barrera láser y lector de RFID.

- Fallas mecánicas: este procedimiento es accesible de dos formas, la primera forma es si hay una autorización, previa a la llegada del autobús, por porte de la compañía de transporte o en el caso de que el vehículo cuente con el SOSoB y reporte automáticamente una avería al momento de su arribo a la estación, de forma que se pueda realizar una programación en el itinerario de su estancia en las instalaciones; la segunda forma es mediante los lectores de puerto EOBD instalados en los bahías de descenso de pasajeros, los cuales debe ser utilizados cuando el bus se encuentra aparcado y que permite determinar si el vehículo es apto para realizar otro viaje.

4.3.3.c. Comunicaciones

Dado que el proyecto IT-Hub está pensado para su utilización en estaciones de autobuses existentes, todas las comunicaciones se realizan mediante redes inalámbricas, utilizando el protocolo 802.11, debido a su ancho de banda disponible y su facilidad de instalación.

Así, en los punto de entrada y salida de la estación, al igual que en los puntos de entrada a la plataforma de ascenso a pasajeros; el lector de RFID se enlaza con el controlador mediante puerto GPIO/ I^2C, mientras que la cámara utiliza un puerto MIPI CSi (Camera Serial Interface) para acoplarse al controlador. Es el controlador el que procesa los datos del RFID y de la cámara para registrar en el SOSLite el identificador de autobús detectado, mediante la red inalámbrica.

Por su parte, las bahías de ascenso/descenso de pasajeros, las de la plataforma de paquetes y encomiendas, y los parking operacionales conectan los lectores de RFID y los sensores de barrera láser mediante los puertos GPIO/ I^2C, el controlador es el encargado de detectar la presencia de un vehículo y leer su identificación para enviar esos datos, usando 802.11, al *Fog site*. Adicionalmente, las bahías de descenso de

pasajeros cuentan con un lector de puerto EOBD que se conecta mediante Bluetooth 4.0 al controlador que se encarga de leer e interpretar los códigos para determinar si es necesario que el bus pase por el taller. Como en las bahías de la estación, en el módulo de excretas los sensores se conectan al controlador mediante los puertos GPIO/ I^2C.

Finalmente, el *Fog site* se conecta a la red mediante Ethernet a alta velocidad y cuenta con salida a Internet. Tanto el mini-datacenter como cada una de las conexiones alámbricas e inalámbricas utilizadas brindan una capa de autenticación y seguridad para preservar la integridad y privacidad de los datos.

4.3.3.d. SOSLite

El servidor en el cual se despliegan las instancias del SOSLite cuenta con un procesador Intel Xeon E5-2650 v3, 32 GB de memoria RAM, 2 TB en disco duro y dos fuentes de alimentación; permite el cambio en caliente de módulos de memoria RAM y de fuente de alimentación. Además cuenta con una UPS de respaldo para mantener la electricidad del servidor. El servidor cuenta con un sistema operativo basado en GNU/Linux y utiliza Docker para el uso de VMs ligeras.

En el *Fog site* eisten siete diferentes instancias de SOSLite, cada una atendiendo a los registros y consultas de una función determinada en la estación: la primera, encargada del control de ingreso y salida de los autobuses en la estación; la segunda, se ocupa de las plataformas de descenso/ascenso de pasajeros; la tercera, de la plataforma de paquetes y encomiendas; la cuarta, de los parking operacionales; la quinta, de la entrada a la plataforma de ascenso; la sexta, del módulo de excretas; y la séptima, de las fallas mecánicas. También existen varias VMs dedicadas a las aplicaciones que utilizan dentro de la estación de autobuses y que hacen uso del SOSLite.

4.3.3.e. Posibles usos

El desarrollo de puesta en marcha de un despliegue de SOSLite con las características citadas, está enfocado en la automatización de la coordinación, el control y la supervisión operativa; sin embargo no se limita solo a prestar esta área, dado que son muchos las servicio que pueden aprovechar los datos que se almacenan en el SOSLite.

El primer servicio que puede hacer uso de los datos recogidos por la red de sensores y almacenados en el SOSLite, es el de información a los pasajeros. Sin la necesidad de realizar grandes procesamientos de datos o de ingresos manuales, los carteles electrónicos de información pueden mostrar los buses que están llegando a la estación y que se disponen a desembarcar pasajeros, con sus respectivas bahías de descenso de pasajeros; al igual, que las alertas indicando los buses que están embarcando y las bahías de las salidas.

Un segundo servicio se centra en la logística de paquetes y encomiendas; consultando el SOS se pueden conocer que buses han llegado a la estación con carga de paquetes, la bahía que le ha sido asignada y el tiempo estimado de arribo a la plataforma de carga/descarga de paquetes; con esta información, una aplicación de programación logística puede asignar recursos para realizar la gestión física de los paquetes que han llegados y los que deben enviarse a las próximas paradas del vehículo.

Por su parte, un agregado de servicios extras a los buses, puede ser una tercera aplicación vinculada a la información que circula por la red de sensores; permitiendo una gestión automatizada de las acciones realizar y los recursos a necesarios para llevarlas a cabo. Por ejemplo, en parking se pueden realizar limpiezas externas e internas al autobús bajo demanda, como también, los cambios de conductor con las sus respectivas comprobaciones y reportes.

De igual forma, un cuarto servicio es el de taller mecánico. Este servicio permite que los buses puedan verificar el funcionamiento de los sistemas mecánicos, hidráulicos y eléctricos del autobús. Además atender a las labores de mantenimiento como la verificación del nivel de los neumáticos, el nivel de aceite en el motor, el nivel de combustible, etc. Así mismo, se incluyen revisiones y arreglos técnico mecánicos, en alineación, balanceo, frenos, etc.

Finalmente, la gestión operativa, objetivo central al cual atiende este caso de uso, mediante el sistema desplegado, permite el control de acceso por zonas, la logística del uso de las bahías y los estacionamientos para los buses intermunicipales, la gestión de residuos generados en viaje y la asistencia mecánica/eléctrica para las compañías de autobuses. Con esto se logra una mayor eficiencia en el uso de los recursos de la terminal, además, se mejora la calidad en el servicio prestado a las compañías transportadoras y al usuario final.

4.3.4. Nodos de transferencia (Transport hub)

4.3.4.a. Introducción

Los nodos de transferencia de transporte intermunicipal de pasajeros, es una estrategia que está entrando en auge en las grandes ciudades, debido a su promesa de lograr una descongestión de las vías de la ciudad, al evitar que los autobuses intermunicipales entren a la misma. Estos nodos se encuentran en la periferia de la ciudad y funcionan como una red descentralizada de transporte, donde los pasajeros descienden de los autobuses intermunicipales y pueden tomar dos acciones: ingresar en la ciudad haciendo uso de los servicios de transporte municipales, o bien, tomar un transporte satélite que los conduzca a otro nodo de transferencia donde puedan continuar su viaje.

Esta aproximación al transporte de pasajeros brinda ventajas a las ciudades, pero puede ser un reto a los pasajeros. Para que se produzca una implantación exitosa, es necesario que se puedan coordinar los diferentes servicios de transporte para que los tiempos de espera sean mínimos; si a esto se le suma: una amplia disponibilidad de información al pasajero y una gestión confiable y automática de los equipajes, se obtiene un servicio que puede llegar a ser superior que la tradicional aproximación centralizada.

Así para dar respuesta a la integración de varios nodos de transferencia de transporte intermunicipal de pasajeros se propone el proyecto ITI (Intercity Transport Interchanges), encargado de la gestión automática de pasajeros entre diversos nodos de transferencia; que basa su funcionamiento en el SOSLite, mediante instancias desplegadas en un *Cloud site* privado.

El funcionamiento distribuido del proyecto, lleva a que exista una instancia del SOSLite atendiendo a cada uno de los nodos, en estas instancias se almacenará la información

proveniente de cuatro SN (Figura 62): la primera encargada del registro de la entrada y la salida de autobuses; la segunda, enfocada en la gestión de la plataforma de descenso y la plataforma de ascenso de pasajeros, la tercera centrada en el sistema de equipaje; y la última, la utilizada para dar información al usuario.

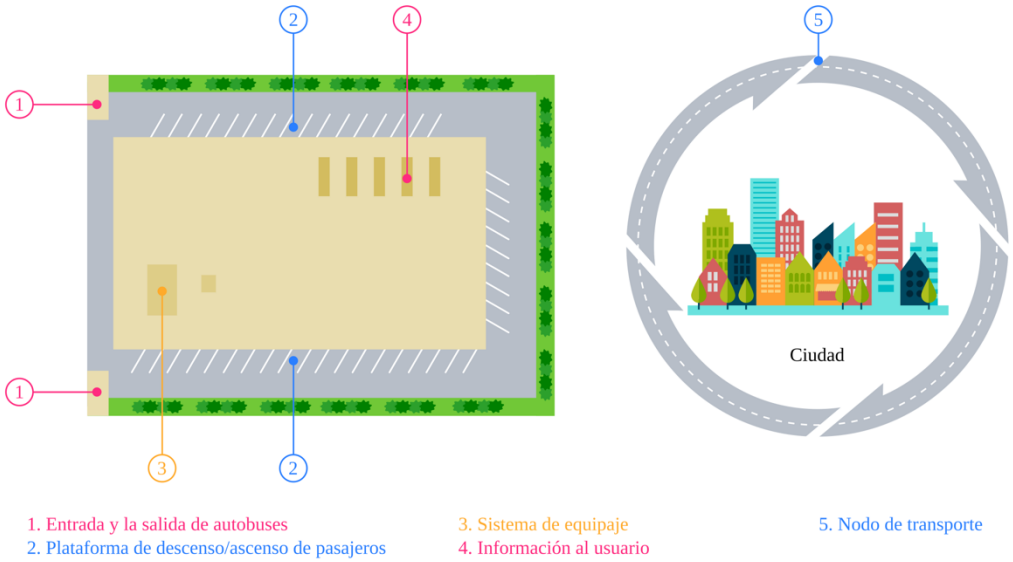


Figura 62 - Nodos de transferencia inteligentes

4.3.4.b. Sensores

Las SN son similares en cada uno de los nodos de transferencia, por lo tanto se detalla los sensores que se encontrarían en un nodo y se entiende que en los otros nodos funcionaran igual:

- Entrada y la salida de autobuses: en cada uno de los nodos de transporte se registra ingreso y la salida de los autobuses, mediante la lectura de su identificador usando RFID o con la lectura de la matrícula del vehículo utilizando una vídeo cámara.

- Plataforma de descenso/ascenso de pasajeros: en cada una de las bahías de las plataformas de descenso/ascenso, existe un lector de RFID encargado de identificar el vehículo estacionado y dos sensores de barrera láser para conocer si una bahía se encuentra ocupada.

- Sistema de equipaje: es uno de los puntos críticos donde los nodos de transferencia deben innovar si desean una aceptación general; en este caso, se plantea un sistema equiparable con el usado en los aeropuertos, mediante el uso cintas de transporte y vehículos livianos de distribución de equipaje. En este sistema, cada maleta es identificada mediante un código de barras que permite que la misma pueda ser enviada inequívocamente desde el puesto de recibo de equipajes hasta el puesto de entrega de equipajes. El proceso es el siguiente:

- Al recibir una maleta se la etiqueta con una cinta que incluye el código de barras, el cual es leído en el puesto de recibo, para asociarlo con la identificación del usuario (su huella dactilar).

- Luego se pasa a un sistema de distribución automático que clasifica las maletas, según el código de barras, y las organiza para ser transportadas al autobús que llevará a la persona.

- En la plataforma de paquetes y encomiendas del nodo, se carga el equipaje de todos los viajeros y se precinta la bodega de equipajes.

- Una vez el autobús llega a el nodo destino, en la plataforma de paquetes se descarga el equipaje.

- El equipaje se distribuye según la lectura del código de barras, asignado el equipaje que debe ser reembarcado en otro autobús y el que se debe llevar las puesto de entrega de equipajes.

- En los puntos de entrega, el equipaje se consigna a la persona cuya huella dactilar se encuentra asociada, utilizando en él un escáner de código de barras y un lector de huellas dactilares.

- Información al usuario: existen varios puntos de información al usuario, los cuales puede consultar utilizando su huella dactilar. En estos puntos se le indicara las condiciones de su viaje actual, incluyendo: la hora de salida, la hora de llegada, la bahía de abordaje, el número de transbordos que debe realizar, etc. También, se incluirán en estos puntos la posibilidad de recibir indicaciones rápidas sobre el trayecto a recorrer en el nodo, de forma que los pasajeros que deben pasar de un bus a otro sepan en todo momento a donde dirigirse, solo con utilizar su huella

4.3.4.c. Comunicaciones

La información de las SN, empleadas en los nodos de transferencia, utiliza diferentes tecnologías de interconexión, incluyendo Internet para la comunicación con el *Cloud site*. En el caso de los registros de entrada y salida de autobuses se emplean una comunicación en tres niveles: en el primero, al controlador se conectan el lector de RFID, mediante GPIO/ I^2C, y la cámara, usando MIPI CSi; en el segundo, el controlador reporta mediante 802.11 a un *Access Point* el cual se encuentra conectado a una red Ethernet; y en el tercero, el *Access Point* envía la información al SOSLite usando la red Ethernet local e Internet.

Por su parte, la información de las plataformas de descenso/ascenso de pasajeros conectan el lector de RFID con el controlador por medio de GPIO/ I^2C, al igual que los sensores de barrera láser. El controlador se conecta a un *Access Point* 802.11 el cual encamina los datos hacia el SOSLite.

En el caso del sistema de equipaje, cada escáner de código de barras se conecta a su contralor mediante USB, el cual envía los datos mediante Ethernet y a través del enrutador hacia el SOSLite. Caso similar se encuentra en los punto de información al

usuario, donde los lectores de huellas dactilares se conectan al controlador mediante USB, mientras que el controlador utiliza la tecnología 802.11 e Internet para realizar el reporte al *Cloud site*.

4.3.4.d. SOSLite

El SOSLite se despliega sobre VMs Docker en un *Cloud Site* privado. Cada una de las instancias corresponde con uno de los nodos de transferencia, agrupando en un solo SOS los datos de: entrada y la salida de autobuses, plataformas de descenso/ascenso de pasajeros, sistema de equipaje e información al usuario.

Así, una instancia puede tener un crecimiento sostenido en el consumo de almacenamiento sin tener limitaciones, gracias a que reside en el *Cloud site*. Esta bondad, facilita la permanencia de la información durante largos periodos de tiempo. Como ventaja adicional, el acercar los datos al lugar donde se van a procesar, influye de manera positiva en los tiempos de respuesta en aplicaciones que requieren del análisis de grandes cantidades de datos.

4.3.4.e. Posibles usos

El uso de CC, provee una capacidad ilimitada de almacenamiento que es aprovechada para la revisión histórica de los datos en busca de tendencias mediante minería de datos y el análisis en tiempo real utilizando *Big Data* para modelar el comportamiento de los nodos y detectar fenómenos anómalos en cada uno de ellos. Además, provee capacidades de procesamiento intensivo bajo demanda, logrando que los análisis sobre los datos y la generación de reportes, se realicen en cualquier momento sin degradar las prestaciones. Adicionalmente, facilita la integración de información de los diferentes nodos de transferencia por parte de las aplicaciones residentes en el mismo *Cloud site*, logrando así una mayor desempeño.

Como en los casos anteriores, existen aplicaciones que pueden aprovechar los datos que se encuentran almacenados en las instancias de SOSLite y que extienden la funcionalidad para la cual ha sido diseñado el sistema. A continuación, se presentan tres ejemplos de esta situación y posteriormente, se revisa el uso principal del proyecto.

Un primer ejemplo, es el análisis del flujo de pasajeros, el cual es fundamental para realizar la planeación de los transportes urbanos y para determinar la cantidad de turistas que tiene la ciudad mensualmente. Estas estadísticas nacen de la agregación de los datos de los diversos nodos de transferencia, sobre los cuales se realizan procesos de minaría de datos y análisis estadístico, de forma que, sean fácilmente aprovechables por la planeación municipal.

Siguiendo con las aplicaciones fuera del diseño original, se encuentra la logística de paquetes y mercancías en los nodos de transferencia; dado que cualquier caja que se encuentre correctamente identificada será encaminada hacia su destino, el servicio de envío de paquetes por vía terrestre puede aprovechar la capacidad extra de los autobuses y el sistema de equipaje en los nodos de transferencia, para lograr llevar de forma eficiente mercancías y paquetes.

El tercer ejemplo es seguimiento de equipaje, un servicio complementario de

información al pasajero, que permite que el pasajero sepa en todo momento la ubicación de su equipaje. Esta información puede hacerse accesible mediante los puntos de información en cada uno de los nodos o mediante una aplicación web/móvil, si se cuenta con un sistema de registro de pasajeros.

Pasando a las aplicaciones para las cuales fue diseñado el proyecto, la información del SOSLite es relevante para la gestión de los nodos de transferencia, permitiendo conocer: la afluencia de pasajeros esperada en cada nodo y la cantidad requerida de autobuses satélite para el transporte de pasajeros entre nodos. Además, es útil para la planeación de: nuevos nodos de transferencia, rutas directas de autobuses entre dos ciudades (de forma que no requieran pasar por los nodos de transferencia), y cursos de acción secundarios en caso de anormalidades en el transporte.

4.3.5. Sistema integrado de transporte intermunicipal

Uniendo los tres proyectos presentados con anterioridad se puede plantear un sistema integral de transporte intermunicipal (Figura 63). En este sistema integrado, todos los buses cuentan con el SOSoB, y las terminales funcionan como nodos de transferencia en las grandes ciudades, uniendo las tecnologías de los proyectos iT-Hub y ITI. En este caso se tendrá un sistema jerárquico de tres niveles: dispositivo limitado, *Fog site* y *Cloud site*; en los cuales la información fluirá entre las diferentes instancias utilizando para ello aplicaciones específicas que hacen llamadas a las operaciones de consulta.

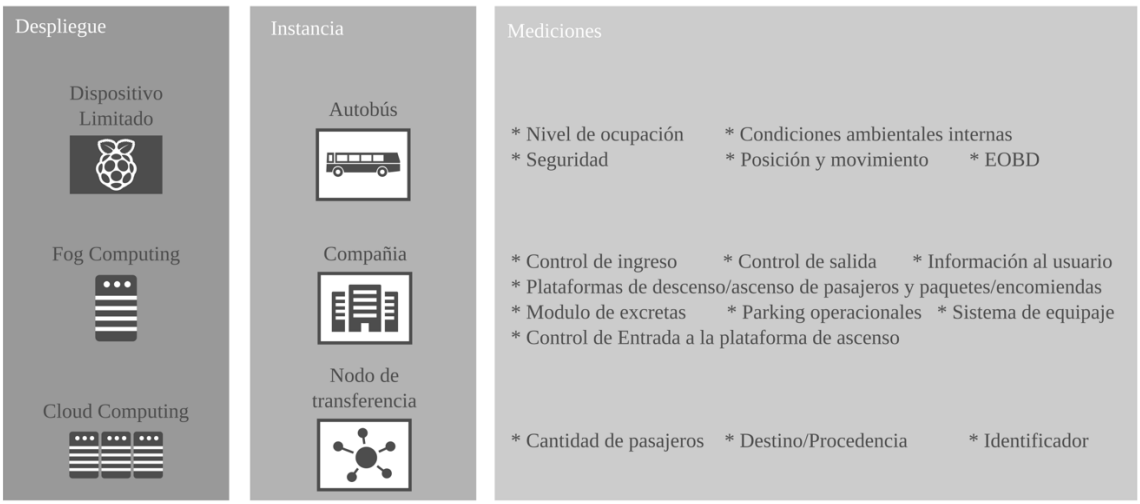


Figura 63 - Sistema integrado de transporte intermunicipal

Así, el SOSLite desplegado en el dispositivo limitado a bordo del autobús registra los datos de la red de sensores del vehículo, incluyendo los leídos desde el puerto EOBD, durante todo el trayecto entre los diferentes nodos de transferencia de las ciudades. Una vez, el bus se encuentra en el punto de control de ingreso del nodo, utiliza la red WiFi para reportar los datos del viaje, incluyendo: nivel de ocupación, condiciones ambientales internas, posición y movimiento, seguridad y EOBD. Por su parte, durante el tiempo que el autobús permanece en el nodo de transferencia, reporta los datos de: nivel de ocupación, seguridad y posición y movimiento.

Toda la información la información que el vehículo reporta, es almacenada en una

instancia del SOSLite en el *Fog site* del nodo. Por cada una de las empresas que gestionan sus vehículos en un respectivo nodo, se mantiene una instancia diferente; de esta forma, se asegura que los datos de funcionamiento de una empresa se encuentran debidamente aislados de los datos de las otras empresas. Una vez la información del SOS es reportada al nodo de transferencia, el SOSLite del autobús puede liberar espacio de almacenamiento, eliminado observaciones pasadas.

Por su parte, en el nodo de transferencia se registran los datos mencionados con anterioridad: los puntos de control de ingreso, los puntos de control de salida, las bahías de las plataformas de descenso/ascenso de pasajeros, las bahías de las plataforma de paquetes y encomiendas, los parking operacionales, el módulo de excretas y el punto de control de Entrada a la plataforma de ascenso; a estos se les suman los datos de: el sistema de equipaje y la información al usuario. Con este fin se utiliza una instancia del SOSLite por cada una de las funciones del nodo de transferencia, las cuales se despliegan en el *Fog site* y se suman a las instancias donde reportan los autobuses.

Cada nodo de transferencia reporta los datos de los puntos de control de ingreso/salida de autobuses, que indican: el identificador del vehículo, el destino o procedencia y la cantidad de pasajeros. Esta información de almacena en una instancia del *Cloud site*, manteniendo la separación de instancias según el nodo desde donde proceden los datos.

Como se puede deducir de la descripción realizada, no todos los datos medidos en las SN se pasan de un nivel a otro; así se prioriza los datos utilizables en cada una de las instancias, se mantienen un nivel inicial de privacidad de la información y se reduce el ancho de banda necesario para la transmisión de los datos entre niveles. Cada nivel puede seguir ofreciendo aplicaciones propias o pueden surgir aplicaciones que utilicen datos cruzados de los diversos niveles.

Así, una aplicación que desea identificar las emisiones del transporte terrestre intermunicipal puede cruzar la información anónima de las emisiones de cada uno de los buses que se corresponde con las llegadas y salidas de buses en los nodos; utilizando de esta forma, los datos disponibles en el segundo y tercer nivel.

Por su parte, si se cuenta con el acceso remoto de la información del SOS a bordo de los autobuses, se puede aprovechar para simular el flujo de vehículos a las vías de entradas a la ciudad, además, se puede estimar la ocupación de las vías de interconexión entre nodos de transferencia que utilizan los buses satélite; esta información es de suma utilidad para planificar las operaciones de la policía de carreteras. Si además se conoce la ocupación de los nodos de transferencia se pueden desviar algunos de los vehículos de transporte intermunicipal para logran descongestionar las plataformas de descenso/ascenso de un nodo especifico.

4.4. Conclusiones

El SOSLite ha sido diseñado para dar servicio a las SN, y es en estas, donde encuentra su nicho de desarrollo natural. Son muchas las aplicaciones en las cuales un SOS ligero puede ser empleado y grandes las ventajas de desplegarlo sobre un dispositivo limitado o como una micro-instancia.

Cuando se despliega el SOSLite sobre dispositivos limitados se obtienen las ventajas de: movilidad, integración, adaptabilidad, bajo costo y proximidad; esta aproximación es muy útil en los casos en los cuales se conoce de antemano las necesidades de recursos computacionales, tales como: automatizados de confort ambiental, sistemas de registro de consumo energético y agricultura de precisión sobre aperos.

De igual forma, el despliegue del SOSLite como instancia de FC brinda las ventajas de: integración, adaptabilidad, itinerancia, flexibilidad, rendimiento y proximidad; lo que hace que sean ideales en los casos en los cuales se requiera una asignación dinámica de recursos computacionales (dentro de unos límites), el balanceo de carga y los sistemas nómadas. Ejemplos de buen uso de esta implementación son: los sistemas de seguridad, los sistemas de producción automática industrial y los sistemas domóticos nómadas.

Así mismo, el caso del despliegue del SOSLite como instancia de CC tiene ventajas específicas, como son: integración, adaptabilidad, itinerancia, flexibilidad y rendimiento; esto hace que esta implementación sea ideal cuando se requiere almacenar grandes cantidades de información y cuando se van a presentar picos de ingresos o consultas de información. Ejemplos donde esta solución es ideal son: los sistemas de monitoreo de volcanes, los túneles de viento y las ciudades inteligentes.

En este capítulo se ha presentado además casos de uso que reflejan cómo se puede sacar provecho del SOSLite al utilizar despliegues sobre plataformas diferentes. En el caso de los dispositivos limitados, el autobús municipal monitorizado del proyecto SOSoB (SOSLite on Board), enfatiza en la movilidad como la ventaja definitiva para optar por el uso de esta plataforma para desplegar el SOS; además, se puede identificar como el SOSLite permite el correcto funcionamiento de SN con tecnologías de comunicación heterogéneas, sin necesidad de un consumo excesivo de recursos computacionales.

Por su parte, la gestión operativa de una estación de autobuses del proyecto iT-Hub (intelligent Transport Hub), profundiza en el uso de un SOSLite como micro-instancia de FC; destacando las ventajas de flexibilidad y proximidad; al permitir el registro de los datos de gran cantidad de sensores, manteniendo los registros por un periodo de tiempo largo y aislando los diferentes servicios de la estación, facilitando así la gestión del control de acceso a información de las aplicaciones.

De forma análoga, el SOSLite como micro-instancia sobre CC ejemplifica su utilidad en la integración de varios nodos de transferencia de transporte intermunicipal de pasajeros del proyecto ITI (Intercity Transport Interchanges), en el cual cada despliegue del SOS ligero refleja el flujo de autobuses en los nodos de transferencia. En este caso se exaltan las ventajas de flexibilidad y rendimiento; permitiendo que exista un almacenamiento ilimitado de información y brindando la capacidad de análisis de grandes cantidades de datos en tiempo real.

Además, se ha abordado el sistema integral de transporte intermunicipal, que aúna los

tres proyectos anteriores y que profundiza en la utilidad práctica que tiene el SOSLite en despliegues jerárquicos que mezclan CC & FC. Llevando un paso más allá esta idea al ofrecer un nivel adicional en esta jerarquía, mediante los dispositivos limitados; proveyendo ventajas en todos los niveles.

Como se ha observado en los cuatro casos de uso analizados, la heterogeneidad a nivel del hardware de los sensores y tecnologías de comunicación, no es un inconveniente al momento de desplegar servicios que aprovechen los datos que proveen las SN. Esto, debido a que el SOSLite provee un mecanismo de interoperabilidad a nivel semántico que abstrae a las aplicaciones de la complejidad que presentan los ambientes heterogéneos.

Finalmente, el SOSLite muestra su versatilidad para adaptarse a un gran número de servicios, los cuales pueden sacar provecho de sumar diversas instancias ligeras, utilizando para aplicaciones que analicen los datos existentes en varias de estas fuentes. Esta versatilidad del SOSLite lo hace una alternativa ideal para afrontar el reto de la interoperabilidad en multitud de escenarios de IoT y CPS, donde las SN son un componente fundamental.

5. SOSFul

5.1. Introducción

El SOSLite es una versión del estándar SOS que se ha adaptado para dar respuesta a los múltiples casos de uso en los cuales los recursos computacionales se encuentran limitados, y las dimensiones del caso de uso/escenario no requieren de todas las funcionalidades del estándar. En el proceso de adaptación se limitó el alcance del estándar completo, debido a la simplificación en las opciones de las operaciones y los modelos de datos; sin embargo, puede ser usado eficientemente en la mayoría de situaciones donde se utiliza el SOS, manteniendo su ligereza, al tiempo que es compatible con otras versiones del SOS completas y pesadas.

Sin embargo, las virtudes del SOSLite en el uso de recursos computacionales limitados se ven contrastadas con la utilización que realiza del ancho de banda de las redes de comunicación. Y es que el SOS con sus interfaces de comunicación basadas en SOAP/XML y sus modelos de datos construidos mediante XML, es un estándar con gran demanda de ancho de banda.

Mientras que, si se analizan el tamaño de los mensajes que emiten los nodos sensores, el coste energético de cada bit enviado a través de las diferentes tecnologías de comunicación y la coexistencia en un medio compartido por cientos de nodos, es fácil llegar a la conclusión de que él envió de la menor cantidad de información es, para este caso, la mejor estrategia de aprovechamiento eficiente de los recursos de las redes de comunicación y energéticos.

Así, se llega a una dicotomía entre la riqueza semántica, necesaria para la interoperabilidad, que aportan los datos enviados empleado protocolos y modelos de datos basados en XML, y la eficiencia de las estructuras de clave/valor (una de las formas más livianas de enviar información independiente de la tecnologías de hardware). La cual, en la actualidad no tiene una solución estandarizada; pero si, algunos avances que se presentan en las especificaciones de OGC.

Es a partir de los resultados positivos obtenidos de SOSLite y como respuesta al balanceo entre eficiencia y semántica, que se ha llevado a la construcción de una implementación nueva de SOS ligero. La cual aúna el uso eficiente de los recursos computacionales, con el mejor aprovechamiento del ancho de banda. Esta versión se ha denominado SOSFul y se ha especificado y desarrollado para contribuir a la próxima versión del estándar SOS.

Dado que la última versión del estándar fue publicada en 2012 y que el mundo tecnológico actual provee una visión que difiere en gran manera a la que se tenía en ese año, el SOSFul se aleja de las implementaciones actuales y se redefine, para proveer unas interfaces más agiles, una mayor facilidad de utilización, unas operaciones para Crear, Leer, Actualizar y Borrar (Create, Read, Update and Delete – CRUD) *featureOfInterest*, sensores y observaciones en su núcleo, y unos modelos de mensajes simplificados.

Así, el SOSFul se encuentra ampliamente vinculado del estándar SOS, retomando sus principios e integrando las recomendaciones que OGC ha ido poniendo a disposición. En especial se concentra en dos aspectos:

1. Interfaces RESTFul [91]: estas interfaces se encuentra presentes en la versión 2

del estándar SOS [17], expresado como trabajo futuro, y se trata de proveer interfaces que se ajusten al estilo de arquitectura de servicios REST (Representational State Transfer) fundamentado en cuatro principios: operaciones bien definidas, modelo cliente/servidor sin estado, sintaxis universal y uso de hipermedios.

2. Buenas prácticas para el perfil de SOS ligero [77]: es una recomendación publicada por el OGC, sobre el que también se ha fundamentado el SOSLite, y que presenta un conjunto reducido de los estándares SOS, SensorML y O&M adaptado a una gran cantidad de casos en los cuales la complejidad de los estándares sobrepasa las necesidades reales.

De esta forma, se considera al SOSFul como un acercamiento al trabajo futuro planteado en el estándar y que abarca la integración del perfil ligero, con cambios en otros estándares del OGC que son propios de esta propuesta y que en conjunto, brindan una implementación de SOS acorde a los desarrollos tecnológicos actuales, por ejemplo: el uso de REST en las arquitecturas de sistemas orientados a servicios, el uso de JSON como modelo de mensajes para la comunicación entre aplicaciones y la integración con protocolos nuevos de la IoT como el protocolo CoAP.

Así, en este capítulo se aborda la creación del SOSFul. Con este fin se ha utilizado la misma metodología de desarrollo iterativa e incremental (Figura 4) que se utilizó para el desarrollo del SOSLite. Las secciones siguientes detallan el ciclo de desarrollo: análisis, diseño, implementación, pruebas y evaluación.

5.2. Análisis

5.2.1. Requerimientos y no funcionales

Para realizar el SOSFul, se emplea el proceso de gestión de requerimientos de software. Por su parte, para los requerimientos no funcionales se emplearán el estándar IEEE 830; mientras que los requerimientos funcionales serán descritos mediante casos de uso en fase enfocados.

Así mismo, a lo largo del capítulo se establece un paralelismo entre las dos implementaciones de SOS desarrolladas en la tesis; de forma que, sea fácil identificar los casos donde es más adecuado utilizar uno u otro. En general, en el SOSFul se observa una mayor simplificación y una mayor especificidad en los modelos, basada en los despliegues reales que actualmente se encuentran en funcionamiento.

En primer lugar, las suposiciones de partida son las mismas que las del SOSLite (Tabla XXXVIII), con lo cual se tiene un mismo punto de partida que permite que el desarrollo cumpla con las condiciones de ser ligero para funcionar dentro de dispositivos limitados o como micro instancia virtual de FC/CC.

Tabla XXXVIII - Suposiciones de desarrollo del SOSFul

Nombre	Descripción
Dispositivo	El SOSFul se ejecutara en dispositivo o instancia virtual que cuenta con al menos las características de una Raspberry Pi 1 Modelo B+: 512 MB RAM, almacenamiento 4 GB en, procesador ARM a 700 MHz
Conectividad	El SOSFul cuenta con conectividad con un ancho de banda mínimo de 10 Mbps. El mismo puede ser accedido de forma continua desde los nodos sensores y los programas de procesamiento de datos
Alimentación	El dispositivo sobre el cual se despliegue el programa cuenta con alimentación eléctrica continua y estable
Sincronización	El sistema cuenta con un mecanismo de sincronización que permite la programación de los envíos de información desde los sensores

En segundo lugar, se presentan el listado de actores que interactúan con el sistema (Tabla XXXIX). Como se puede apreciar en este punto, se encuentra la primera diferencia entre las dos implementaciones de SOS. En el caso del SOSFul, no se hace uso del sistema de ficheros, haciendo que buena parte de los requerimientos funcionales se vean modificados. Esto implica, además, que los modelos de datos se almacenarán directamente en la base de datos y estén sujetos a las ventajas y desventajas que tenga ésta.

Tabla XXXIX - Listado de actores del SOSFul

ID	Nombre	Información
A01	Proveedor de datos	Cada uno de los dispositivos o aplicaciones que pueden registran información en el sistema. Ejemplo: nodo sensor, simuladores de SN, etc.
A02	Consumidor de datos	Cada una de los dispositivos o aplicaciones que consultan los datos del sistema para procesarlos. Ejemplo: procesador de eventos complejos, procesador de grandes datos, etc.
A03	Base de datos	Base de datos para almacenar la información y meta-información sobre sensores y observaciones

En tercer lugar, se presenta el diagrama de casos de uso (Figura 64), además de evidenciar la falta del sistema de ficheros como uno de los actores, se representa la

existencia de las operaciones CRUD para los tres conceptos principales del SOS: *featureOfInterest*, sensor y observación. A diferencia del estándar SOS 2.0, la propuesta del SOSFul permite la manipulación directa del *featureOfInterest*, que identifica el objeto real sobre el que se hace la observación.

Figura 64 - Diagrama de casos de uso del SOSFul

Una vez se tiene una visión general de la funcionalidad que provee el SOSFul, se puntualizan los requerimientos funcionales, detallados como casos de uso en fase

enfocados. Utilizando esta forma de descripción se puede reconocer como interactúan los actores con el sistema en cada una de las funcionalidades. También se comprueba, que en general, se requieren menos pasos para realizar las funcionalidades del SOSFul en comparación a las del SOSLite.

Tabla XL - RF del SOSFul: Consultar la auto-descripción del sistema

Identificación	**RF01**	
Nombre	Consultar la auto-descripción del sistema	
Fase	Enfocados	
Descripción	Consulta la auto-descripción del SOS desplegado, incluyendo: la identificación del servicio, el proveedor de servicio, los filtros soportados y los metadatos de la información almacenada. Soporta filtros de sección: serviceIdentification, serviceProvider, filterCapabilities, contents.	
Actores	Principales: (A01) Proveedor de datos, (A02) Consumidor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita la auto auto-descripción del sistema	
		2. Procesa los datos de la petición y determina la información que se enviara como respuesta según los filtros pasados en la petición. Si no se encuentra ningún filtro se envía toda la descripción
		3. Consulta la base de datos
		4. Construye la respuesta
		5. Envía la respuesta
Excepciones	N/A	
Casos relacionados	N/A	

Tabla XLI - RF del SOSFul: Insertar feature

Identificación	**RF02**	
Nombre	Insertar feature	
Fase	Enfocados	
Descripción	Agrega un sensor al featureOfInterest, almacenando los datos en la base de datos en formato JSON	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Envía los datos del feature en formato JSON	
		2. Procesa los datos de la petición
		3. Valida los datos del feature pasados en la petición
		4. Almacena los datos del feature en la base de datos y le asigna un identificador
		5. Construye la respuesta
		6. Envía la respuesta
Excepciones	*Los datos del feature enviado son inválidos* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	N/A	

Tabla XLII - RF del SOSFul: Consultar feature por identificador

Identificación	**RF03**	
Nombre	Consultar feature por identificador	
Fase	Enfocados	
Descripción	Consulta la información del feature cuyo identificador es pasado dentro de la petición	
Actores	Principales: (A02) Consumidor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita los datos de un feature cuyo identificador es pasado dentro de la petición	
		2. Procesa los datos de la petición y obtiene el identificador del feature
		3. Consulta los datos del feature en la base de datos
		4. Construye la respuesta
		5. Envía la respuesta
Excepciones	*El feature no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	N/A	

Tabla XLIII - RF del SOSFul: Consultar feature por query

Identificación	**RF04**	
Nombre	Consultar feature por query	
Fase	Enfocados	
Descripción	Consulta la información del feature según la query pasada dentro de la petición. El query soporta el filtro: spatialFilter	
Actores	Principales: (A02) Consumidor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita los datos de los feature que cumplen con el query pasado dentro de la petición	
		2. Procesa los datos de la petición y obtiene el query
		3. Construye la consulta a la base de datos, según los filtros pasados en la petición. Si no se encuentra ningún filtro se envía todos los feature registrados
		4. Consulta los datos de los feature en la base de datos
		5. Construye la respuesta
		6. Envía la respuesta
Excepciones	N/A	
Casos relacionados	N/A	

Tabla XLIV - RF del SOSFul: Actualizar feature

Identificación	**RF05**
Nombre	Actualizar feature
Fase	Enfocados
Descripción	Actualiza los datos del feature pasado en la petición
Actores	Principales: (A01) Proveedor de datos

Guion	Secundarios: (A03) Base de datos	
	Actor	**Sistema**
	1. Envía los datos del feature en formato JSON	
		2. Procesa los datos de la petición
		3. Consulta si el feature existe
		4. Validar los datos del feature pasado en la petición
		5. Actualiza los datos del feature en la base de datos
		6. Construye la respuesta
		7. Envía la respuesta
Excepciones	*El feature no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta *Los datos del feature pasado son inválidos* 4. 5. Construye la respuesta de error indicando la condición que lo genero 6. Envía la respuesta	
Casos relacionados	N/A	

Tabla XLV - RF del SOSFul: Eliminar feature

Identificación	**RF06**	
Nombre	Eliminar feature	
Fase	Enfocados	
Descripción	Elimina los datos del feature con el identificador pasado en la petición	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita la eliminación de los datos de un feature cuyo identificador es pasado dentro de la petición	
		2. Procesa los datos de la petición y obtiene el identificador del feature
		3. Consulta si el feature existe
		4. Elimina los datos del feature de la base de datos
		5. Construye la respuesta
		6. Envía la respuesta
Excepciones	*El feature no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	N/A	

Tabla XLVI - RF del SOSFul: Insertar sensor

Identificación	**RF07**	
Nombre	Insertar sensor	
Fase	Enfocados	
Descripción	Agrega un sensor al sistema, almacenando los datos en la base de datos en formato SensorML simplificado (JSON)	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Envía los datos del sensor en	

	formato SensorML simplificado (JSON)	
		2. Procesa los datos de la petición
		3. Valida los datos del sensor pasado
		4. Valida que el feature exista
		5. Almacena los datos del sensor en la base de datos
		6. Construye la respuesta
		7. Envía la respuesta
Excepciones	*Los datos del sensor pasado son inválidos* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta *El feature no existe en el sistema* 4. 5. Construye la respuesta de error indicando la condición que lo genero 6. Envía la respuesta	
Casos relacionados	N/A	

Tabla XLVII - RF del SOSFul: Consultar sensor por identificador

Identificación	**RF08**	
Nombre	Consultar sensor por identificador	
Fase	Enfocados	
Descripción	Consulta la información del sensor cuyo identificador es pasado dentro de la petición	
Actores	Principales: (A02) Consumidor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita los datos de un sensor cuyo identificador es pasado dentro de la petición	
		2. Procesa los datos de la petición y obtiene el identificador del sensor
		3. Consulta los datos del sensor en la base de datos
		4. Construye la respuesta
		5. Envía la respuesta
Excepciones	*El sensor no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	N/A	

Tabla XLVIII - RF del SOSFul: Consultar sensor por query

Identificación	**RF09**	
Nombre	Consultar sensor por query	
Fase	Enfocados	
Descripción	Consulta la información de los sensores que cumplen con la query pasada dentro de la petición. Los filtros soportados son: spatialFilter, observedProperty y featureOfInterest	
Actores	Principales: (A02) Consumidor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita los datos de los sensores que cumplen con la query pasada dentro de la petición	

		2. Procesa los datos de la petición y obtiene la query
		3. Construye la consulta a la base de datos, según los filtros pasados en la petición. Si no se encuentra ningún filtro se envía todos los sensores registrados
		4. Consulta los datos de los sensores en la base de datos
		5. Construye la respuesta
		6. Envía la respuesta
Excepciones	N/A	
Casos relacionados	N/A	

Tabla XLIX - RF del SOSFul: Actualizar sensor

Identificación	**RF10**	
Nombre	Actualizar sensor	
Fase	Enfocados	
Descripción	Actualiza los datos del sensor pasado en la petición	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Envía los datos del sensor en formato SensorML simplificado (JSON)	
		2. Procesa los datos de la petición
		3. Consulta si el sensor existe
		4. Valida los datos del sensor pasado
		5. Valida que el feature pasado exista en la base de datos
		6. Actualiza los datos del sensor en la base de datos
		7. Construye la respuesta
		8. Envía la respuesta
Excepciones	*El sensor no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta *Los datos del sensor pasado son inválidos* 4. 5. Construye la respuesta de error indicando la condición que lo genero 6. Envía la respuesta *El feature no existe en el sistema* 5. 6. Construye la respuesta de error indicando la condición que lo genero 7. Envía la respuesta	
Casos relacionados	N/A	

Tabla L - RF del SOSFul: Eliminar sensor

Identificación	**RF11**
Nombre	Eliminar sensor
Fase	Enfocados
Descripción	Elimina los datos del sensor con el identificador pasado en la petición
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos

Guion	**Actor**	**Sistema**
	1. Solicita la eliminación de los datos de un sensor cuyo identificador es pasado dentro de la petición	
		2. Procesa los datos de la petición y obtiene el identificador del sensor
		3. Consulta si el sensor existe
		4. Elimina los datos del sensor de la base de datos
		5. Construye la respuesta
		6. Envía la respuesta
Excepciones	*El sensor no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	N/A	

Tabla LI - RF del SOSFul: Insertar observación

Identificación	**RF1**	
Nombre	Insertar observación	
Fase	Enfocados	
Descripción	Agrega una observación al sistema, almacenando los datos en la base de datos en formato O&M simplificado (JSON)	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Envía los datos de la observación en formato O&M simplificado (JSON)	
		2. Procesa los datos de la petición
		3. Valida los datos de la observación pasada
		4. Valida que el sensor exista
		5. Valida que el feature exista
		6. Almacena los datos de la observación en la base de datos
		7. Construye la respuesta
		8. Envía la respuesta
Excepciones	*Los datos de la observación pasada son inválidos* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta *El sensor no existe en el sistema* 4. 5. Construye la respuesta de error indicando la condición que lo genero 6. Envía la respuesta *El feature no existe en el sistema* 5. 6. Construye la respuesta de error indicando la condición que lo genero 7. Envía la respuesta	
Casos relacionados	N/A	

Tabla LII - RF del SOSFul: Consultar observación por identificador

Identificación	**RF13**
Nombre	Consultar observación por identificador

Fase	Enfocados	
Descripción	Consulta la información de la observación que tiene el identificador pasado en la petición	
Actores	Principales: (A02) Consumidor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita los datos de una observación cuyo identificador es pasado dentro de la petición	
		2. Lee la información de la petición y determina el identificador de la observación solicitada
		3. Consulta los datos de la observación en la base de datos
		4. Construye la respuesta
		5. Envía la respuesta
Excepciones	*La observación no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta	
Casos relacionados	N/A	

Tabla LIII - RF del SOSFul: Consultar observación por query

Identificación	**RF14**	
Nombre	Consultar observación por query	
Fase	Enfocados	
Descripción	Consulta la información de las observaciones que cumplen con la query pasada dentro de la petición. Los filtros soportados son: spatialFilter, temporalFilter, sensor, observedProperty y featureOfInterest	
Actores	Principales: (A02) Consumidor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita los datos de las observaciones que cumplen con la query pasada dentro de la petición	
		2. Procesa los datos de la petición y obtiene la query
		3. Construye la consulta a la base de datos, según los filtros pasados en la petición. Si no se encuentra ningún filtro se envía todos las observaciones registradas
		4. Consulta los datos de las observaciones en la base de datos
		5. Construye la respuesta
		6. Envía la respuesta
Excepciones	N/A	
Casos relacionados	N/A	

Tabla LIV - RF del SOSFul: Actualizar observación

Identificación	**RF15**	
Nombre	Actualizar observación	
Fase	Enfocados	
Descripción	Actualiza los datos de la observación pasada en la petición	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**

	1. Envía los datos de la observación en formato O&M simplificado (JSON)	
		2. Procesa los datos de la petición
		3. Consulta si la observación existe
		4. Valida los datos de la observación pasada
		5. Valida que el feature pasado exista en la base de datos
		6. Valida que el sensor exista
		7. Actualiza los datos de la observación en la base de datos
		8. Construye la respuesta
		9. Envía la respuesta
Excepciones	*La observación no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero 5. Envía la respuesta *Los datos de la observación pasada son inválidos* 4. 5. Construye la respuesta de error indicando la condición que lo genero 6. Envía la respuesta *El feature no existe en el sistema* 5. 6. Construye la respuesta de error indicando la condición que lo genero 7. Envía la respuesta *El sensor no existe en el sistema* 6. 7. Construye la respuesta de error indicando la condición que lo genero 8. Envía la respuesta	
Casos relacionados	N/A	

Tabla LV - RF del SOSFul: Eliminar observación

Identificación	**RF16**	
Nombre	Eliminar observación	
Fase	Enfocados	
Descripción	Elimina los datos de la observación con el identificador pasado en la petición	
Actores	Principales: (A01) Proveedor de datos Secundarios: (A03) Base de datos	
Guion	**Actor**	**Sistema**
	1. Solicita la eliminación de los datos de una observación cuyo identificador es pasado dentro de la petición	
		2. Procesa los datos de la petición y obtiene el identificador de la observación
		3. Consulta si la observación existe
		4. Elimina los datos de la observación de la base de datos
		5. Construye la respuesta
		6. Envía la respuesta
Excepciones	*La observación no existe en el sistema* 3. 4. Construye la respuesta de error indicando la condición que lo genero	

	5. Envía la respuesta
Casos relacionados	N/A

Cumplir los requerimientos funcionales asegura que el SOSFul ofrezca las funcionalidades para las cuales ha sido ideado. Sin embargo, es necesario contar con los requerimientos no funcionales para determinar la arquitectura que la implementación debe seguir. De esta forma, se detallan a continuación, los requisitos no funcionales más relevantes para esta implementación de SOS ligero; siendo la usabilidad, el requisito no funcional que diferencia las dos implementaciones abordadas en esta tesis.

Tabla LVI - RNF del SOSFul: Dispositivo

Identificación	**RNF01**
Nombre	Dispositivo
Descripción	El SOSFul se debe adaptar al funcionamiento de un dispositivo limitado con características equivalentes a una Raspberry Pi 1 Modelo B+

Tabla LVII - RNF del SOSFul: Rendimiento

Identificación	**RNF02**
Nombre	Rendimiento
Descripción	Garantizar un mínimo de una operación cada 500 milisegundos

Tabla LVIII - RNF del SOSFul: Portabilidad

Identificación	**RNF03**
Nombre	Portabilidad
Descripción	Asegurar el correcto funcionamiento en cualquier distribución GNU/Linux

Tabla LIX - RNF del SOSFul: Modificabilidad

Identificación	**RNF04**
Nombre	Modificabilidad
Descripción	El SOSFul debe poder ser modificado fácilmente para ampliar sus características

Tabla LX - RNF del SOSFul: Usabilidad

Identificación	**RNF05**
Nombre	Usabilidad
Descripción	El sistema debe poder ser fácilmente desplegable y usable, de forma que la integración con otros sistemas sea sencilla y su uso por parte de desarrolladores claro

5.2.2. Arquitectura

Tal como en el caso del SOSLite, esta implementación de SOS ligero mantiene una arquitectura orientada a servicios. Se diferencia en que el SOSFul deja de lado el tradicional SOAP y opta por el estilo REST (Representational State Transfer) que es mucho más fácil de integrar con otros sistemas (RNF05) y que permite la integración y modificación de las características funcionales de forma simple (RNF04).

Para a implementación del software, se emplean una arquitectura de tres capas (Figura 65): enrutamiento, *middleware* y modelo de datos. A través de este patrón arquitectural, se favorece la portabilidad (RNF03) y la modificabilidad (RNF04); sin embargo, cada capa agrega más tiempo de procesamiento, influenciando de forma negativa el

rendimiento (RNF02) del servicio.

Figura 65 - Arquitectura del SOSFul

En esta arquitectura, la capa de enrutamiento se encarga de recibir las URI (Uniform Resource Identifier) y determinar cuál es el middleware con la responsabilidad de atender a dicha petición; acto seguido, delega la petición al middleware correspondiente. Por su parte, la capa de middleware responde a cada una de las operaciones del CRUD, según el concepto sobre el que tiene responsabilidad. Finalmente, la capa de modelo de datos, se encarga de mantener una representación de los tres conceptos principales en el SOSFul: nodos sensores (Sensor), observaciones (Observation) y áreas de interés (FeatureOfInterest).

Así, para dar cumplimiento a todos los requerimientos no funcionales y revertir el rendimiento negativo aportado por el uso de las capas, se plantean varias tácticas arquitecturales, incluyendo: la reducción del *overhead* computacional y la introducción de concurrencia en el procesamiento de las operaciones.

Con miras a reducir el *overhead* computacional, se utiliza la misma técnica que en el casos del SOSLite, pero llevándola un paso hacia delante mediante adaptaciones que incluyen el uso de JSON en lugar de XML en los modelos de mensajes y el pasar de un protocolo descriptivo como el SOS sobre SOAP a encapsular la riqueza semántica en las URIs (RNF01, RNF02). La selección de JSON sobre XML se debe a que el primero es mucho menos verboso, haciendo los mensajes más livianos, sin perder la facilidad de procesamiento y la expresividad sintáctica del XML.

Por su parte, para la introducción de concurrencia en el procesamiento de las operaciones se emplea un servidor orientado a eventos para el despliegue del servicio, tal como el SOSLite. En este caso, vistas las ventajas de esta aproximación, se ha optado por integrar mejor la aplicación con el servidor, haciendo que la comunicación entre capas también responda mediante orientación a eventos. De esta forma se obtiene un alto desempeño con un bajo consumo de recursos (RNF01, RNF02).

5.2.3. Diagrama de Conceptos

Cuando se revisa el modelo de datos del SOSLite se observa la existencia de los *Procedure*, los cuales abstraen bajo un solo concepto los métodos, algoritmos, instrumentos, sensores o sistemas, que tienen la capacidad de realizar observaciones. A partir de los despliegues reales, y en la búsqueda de cumplir los requerimientos no

funcionales, este concepto se simplifica en el SOSFul; de forma que, solo los sistemas pueden generar las observaciones. Concretamente se toma el concepto del *PhysicalSystem*, utilizado en la recomendación de SOS ligero, y se pasa a llamar Sensor, entendiéndolo como un nodo sensor que puede contener varios sensores reales y que siempre ocupa un espacio físico real. Además, este concepto de Sensor siempre tiene un *FeatureOfInterest* asociado.

Otra simplificación realizada, es la eliminación del concepto *ObservationOffering*, el cual se ve integrado dentro del concepto Sensor. De esta forma, ahora se tiene una relación bidireccional entre el Sensor y las Observaciones, dotando a un Sensor con un conjunto de Observaciones, mientras que, una Observación siempre está vinculada con un Sensor.

Con estas modificaciones, se construye el modelo de conceptos para el SOSFul (Figura 66). El mismo, mantiene la representación de las observaciones mediante el concepto "Observation", el cual siempre tiene relación con un *FeatureOfInterest* y con una *ObservedProperty*. Análogamente, el *FeatureOfInterest* (área de interés) indica el emplazamiento del Sensor que generó la observación.

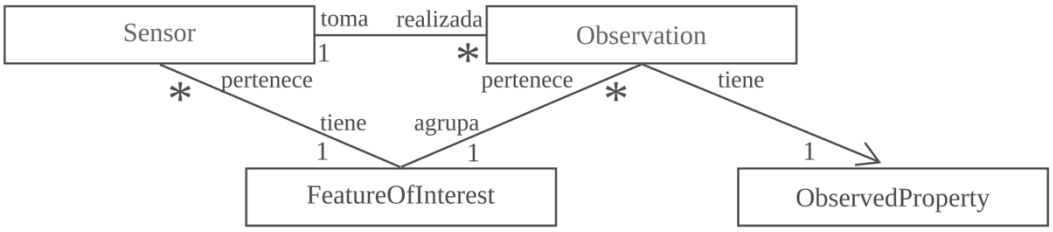


Figura 66 - Diagrama de conceptos SOSFul

Como se observa, este modelo de conceptos es más concreto que el de la anterior implementación. Esto facilita las operaciones que utilizan los conceptos y posibilita una interacción muy cercana al mundo real; siendo los nodos sensores (*Sensor*), las observaciones (*Observation*) y las áreas de interés (*FeatureOfInterest*), representaciones directas de las entidades y fenómenos físicos con los que se encuentran los sistemas basados en sensores.

5.3. Diseño

5.3.1. Scripts del sistema

El SOSFul se ha programado siguiendo un paradigma orientado a eventos, integrando por diseño la concurrencia en el procesamiento de las operaciones, alcanzando un mejor aprovechamiento de los recursos compartidos. Con el paradigma escogido además se acopla de forma natural la arquitectura orientada a servicios del SOS y posibilita la alta cohesión y el bajo acoplamiento; características ideales para facilitar la modificabilidad.

Así, las directrices de programación se ven enmarcadas en el siguiente diagrama (Figura 67); donde se indican las diferentes capas de la arquitectura de programación y se detallan los scripts encargados del enrutamiento, el *middleware* y los modelos de datos. Los scripts están delimitados por su responsabilidad sobre un concepto; así, existe un script para el enrutamiento, un script con las operaciones por cada concepto y un script que modela cada uno de los conceptos del SOSFul.

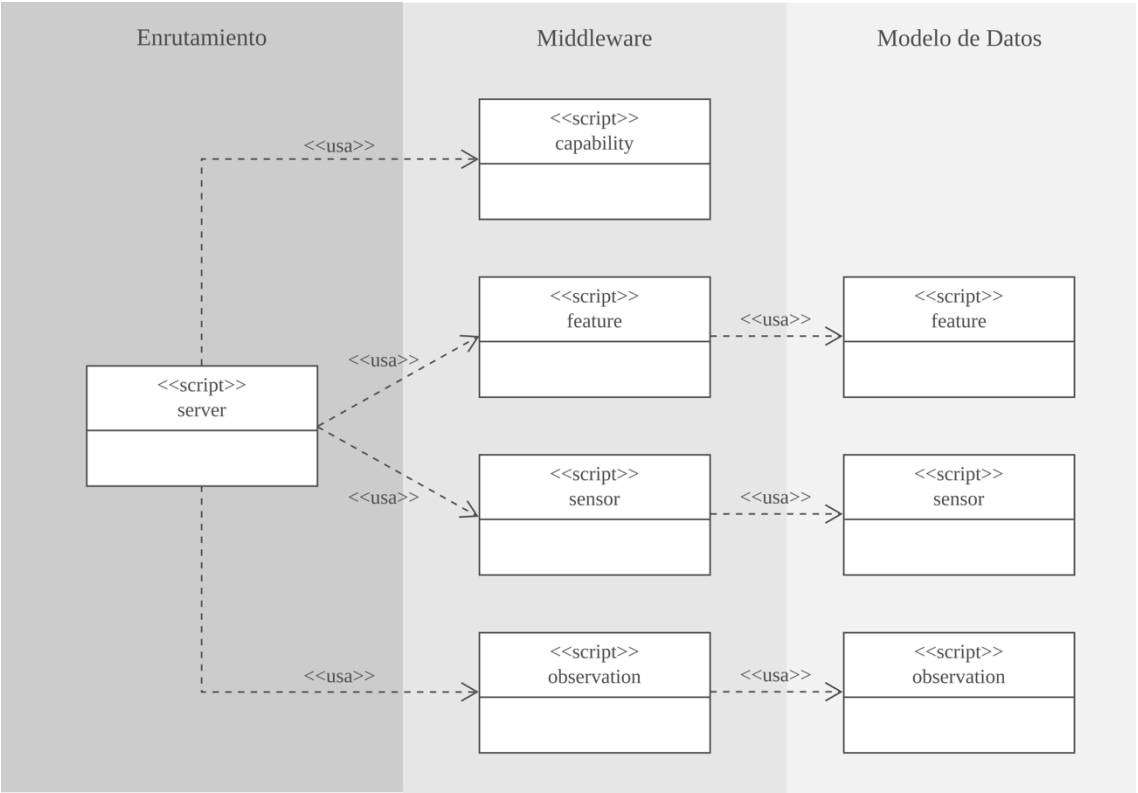


Figura 67 - Scripts del SOSFul

El script "*server*" mantiene un solo procedimiento, este se encarga de inicializar un servidor cuya función es recibir las peticiones realizadas, determinar el *middleware* que debe atender dicha petición y pasarle la ejecución al procedimiento pertinente en el script del *middleware*. Además de la funcionalidad descrita, el servidor también se encarga de manejar los errores no capturados por los middlewares, de forma que siempre se entregue el mensaje correcto a los clientes que se conectan con él.

Los *middlewares* están compuestos por dos tipos de procedimientos: los que atienden a las peticiones de clientes y los de apoyo. Los procedimientos de apoyo realizan operaciones que son comunes a varios procedimientos que atienden peticiones;

optimizando la reutilización de código. Por su parte, los procedimientos encargados de las peticiones encapsulan las funcionalidades; leyendo las peticiones, determinando los parámetros pasados, realizando las operaciones sobre la base de datos y creando las respuestas a los clientes.

Finalmente, los modelos se encargan de precisar los conceptos principales del SOSFul, determinando las propiedades y el tipo de datos que mejor se adapta a ellos. Estos modelos permiten una interacción más directa con la base de datos, ya que sirven como interprete entre los documentos de la base de datos y los objetos manipulables dentro del *middleware*.

5.3.2. Modelo de datos

En los despliegues reales de SOS se ha determinado que las operaciones de inserción y consulta son las más utilizadas. Por este motivo, se da prioridad a este tipo de operaciones sobre las de actualización y eliminación. Así, el comportamiento esperado en el uso del SOSFul es realizar el ingreso insipiente de las áreas de interés y de los sensores; para posteriormente realizar ingresos y consultas masivas sobre observaciones.

A diferencia que en el SOSLite, los modelos de datos se realizaran sobre JSON y serán pasados directamente a la base de datos; no existen ficheros inmutables, lo que permite actualizaciones más agiles; con lo que esta operación también se ve beneficiada en esta nueva implementación; aunque sea poco utilizada.

El no realizar escrituras y lecturas sobre el sistema de ficheros; y en su lugar, emplear la base de datos tiene un impacto amplio sobre el tiempo de procesamiento; siempre y cuando no sea necesario unir datos de dos conceptos diferentes, es decir, que el modelo sea desnormalizado y poco relacional. Para lograr esta mejora, todas las operaciones sobre a la base de datos están optimizadas para emplear un solo concepto por consulta.

Como se vio en el capítulo del SOSLite, es fácil obtener un modelo desnormalizado y poco relacional que promueva el alto rendimiento en las operaciones de inserción y consulta. En este caso el modelo de datos (Figura 68) se aproxima bastante al modelo de conceptos (Figura 67); aunque no por esto, es una adaptación directa de este; en especial se integra el concepto *ObservedProperty* al concepto *Observation*, en uno solo.

Tal como en la implementación anterior, optar por una base NoSQL basada en documentos permite aprovechar su rendimiento en operaciones de inserción y consulta, junto con la flexibilidad de las estructuras de almacenamiento. También es muy útil en bases de datos distribuidas y proveen una alta tolerancia a fallos.

Como se puede observar (Figura 68), este modelo de datos se encuentra compuesto de tres colecciones de documentos: *FeatureOfInterest*, *Sensor* y *Observation*. Como sus nombres lo indican se corresponden de forma directa con cada uno de los conceptos, tratados en la sección anterior, con la única diferencia del *ObservedProperty* que pasa a ser una de las propiedades de la colección de observaciones.

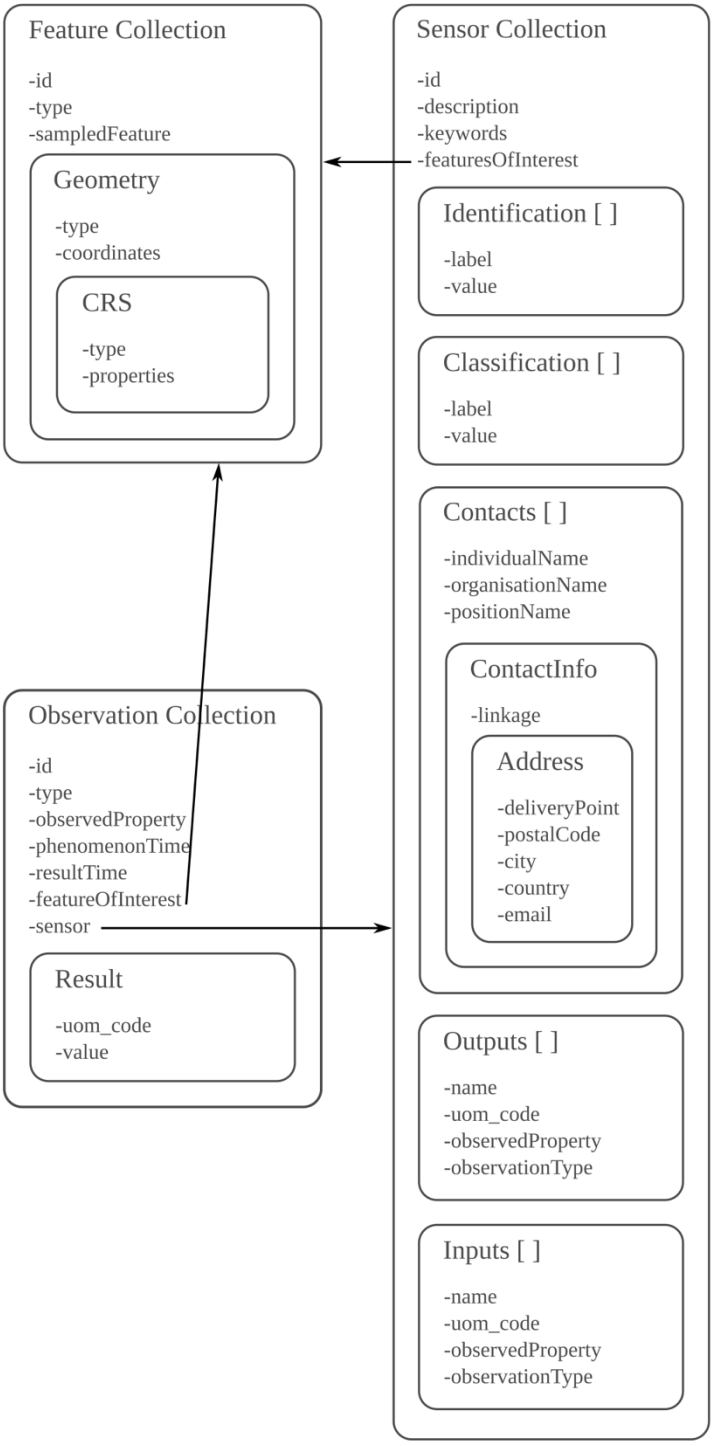


Figura 68 - Modelo de datos SOSFul

5.3.3. Diagramas de actividad

El funcionamiento del SOSFul es sencillo y eficiente; enfocado en la simplicidad y la concreción. Sin embargo, entender como ha sido implementado requiere conocer su

visión estática y dinámica. La visión estática, ya se ha revisado mediante: el diagrama de casos de uso, el modelo de conceptos y el modelo de datos. Mientras que la visión dinámica, se expondrá en esta sección mediante los diagramas de actividad.

Utilizar herramientas de diseño de software diferentes entre el SOSFul y el SOSLite se debe al paradigma de programación escogido. Con la programación orientada a objetos, el diagrama de clase y los diagramas de secuencia resultan ser adecuados para la descripción de la aplicación; sin embargo, se tornan incorrectos o complejos en el caso de la programación orientada a eventos. Así, describir la visión dinámica mediante los diagramas de actividad presenta una aproximación más adecuada para la programación orientada a eventos. Sin embargo, no se puede tener un equivalente de los diagramas de clases en este paradigma, habiéndose empleado el diagrama de scripts presentado (Figura 67), el cual no está basado en estándares pero aporta claridad en cuanto la responsabilidad que se asume en cada script.

Para los diagramas de actividad que se presentan a continuación, el lector encontrara una referencia clave a los actores proveedor y consumidor, como también a los scripts de las capas de enrutamiento y de *middleware*. Así, resulta fácil seguir como las decisiones presentadas del análisis y del diseño, enmarcan todo el desarrollo realizado en el SOSFul y finamente como el conjunto impacta en el rendimiento medido en la evaluación de prestaciones. Para presentar los diagramas de actividad se seguirá el orden de los casos de uso funcionales y en ellos se identifican las operaciones CRUD que des despliegan en el núcleo del SOSFul.

5.3.3.a. Consultar la auto-descripción del sistema

La auto-descripción del sistema puede ser pedida por el proveedor o el consumidor; empleando el verbo GET sobre la raíz de la instalación del SOSFul. La petición llega al script server, el cual se encarga del enrutamiento hacia el *middleware* adecuado según la URI pasada o de enviar un mensaje de error, en caso de no contar con ningún *middleware* que responda a la petición realizada. Por su parte, el *middleware capability* se encarga de determinar los filtros requeridos, consultar la base de datos y responder al generador de la petición. En este caso no existe un modelo *capability*, si no que se emplean información cargada en el SOSFul y los modelos *Sensor* y *Observation*.

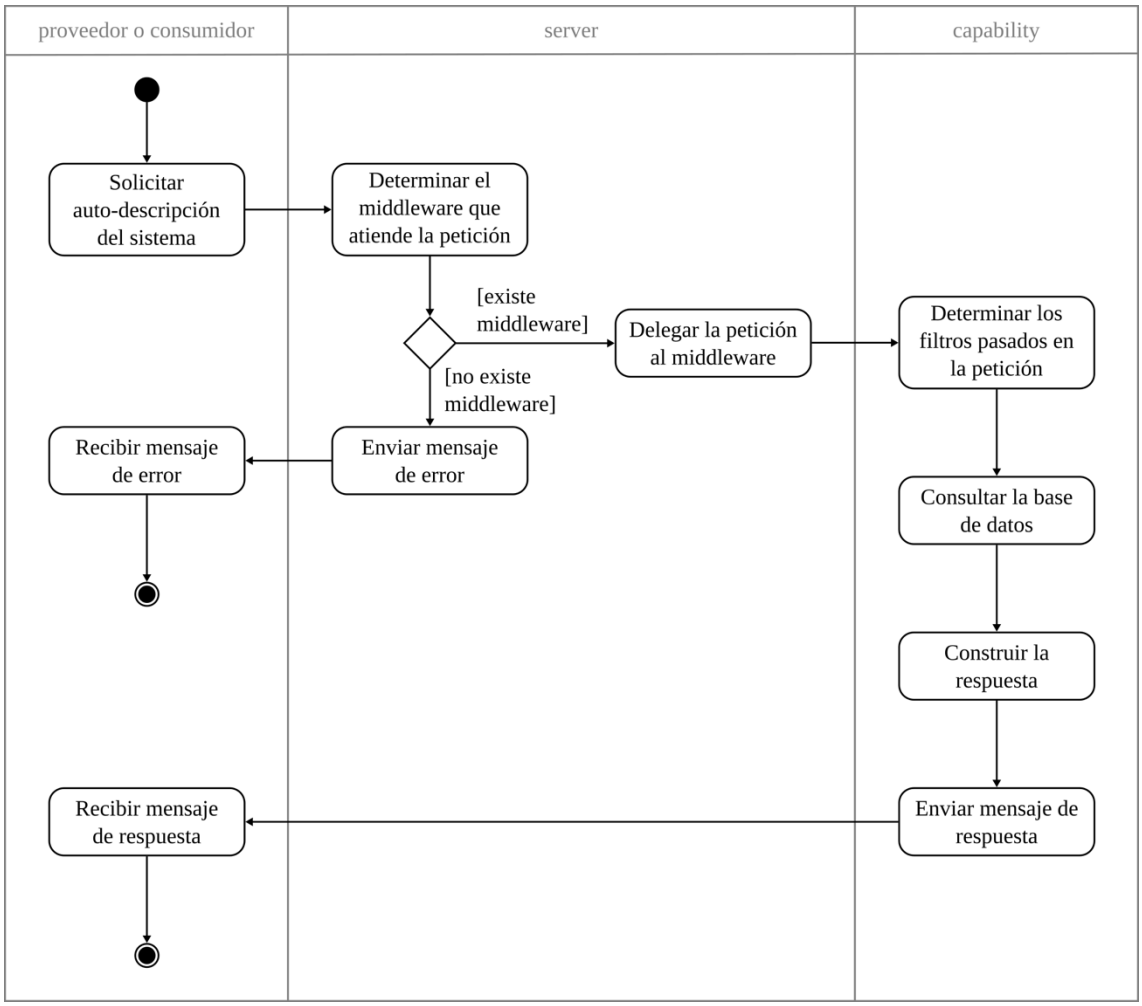


Figura 69 – DA del SOSFul: Consultar la auto-descripción del sistema

5.3.3.b. Insertar feature

La operación de agregar un área de interés (*FeatureOfInterest*) es iniciada por el proveedor, enviado para esto la información del *feature* en formato JSON en el cuerpo de la petición; utilizando el verbo POST sobre el directorio /feature. Empleado la URI utilizada por el proveedor, el server redirige la petición hacia el *middleware* de *feature*, quien tiene la responsabilidad de todas las operaciones de este concepto, incluyendo el agregar nuevos objetos. Para esto el *middleware* obtiene los datos desde la petición, los validad e inserta en la base de datos; empleando de forma implícita el modelo de datos *feature*; posteriormente crea la respuesta con el identificador asignado en la base de datos y lo envía al proveedor.

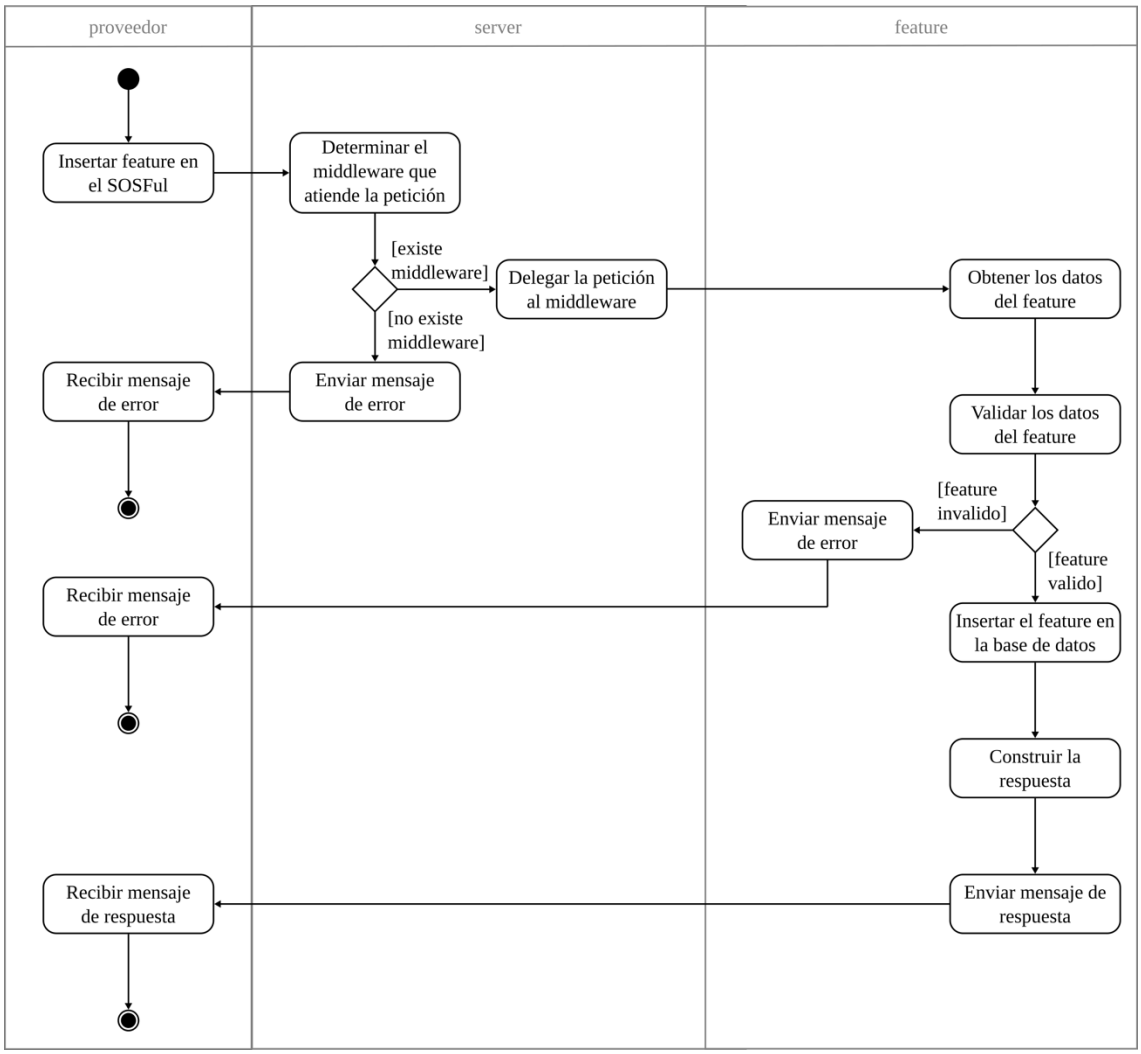


Figura 70 - DA del SOSFul: Insertar feature

5.3.3.c. Consultar feature por identificador

A diferencia de la operación anterior, las consultas son iniciadas por los consumidores; los cuales, envían en la URI el identificador del feature que quiere consultar, así: /feature/:id; y empleado el verbo GET. El server redirige la petición al *middleware feature* quien obtiene el ID y realizan la consulta a la base de datos. Con los datos que devuelve la base de datos, el *script feature* construye la respuesta, que incluye el *feature* en formato JSON, para enviarla al consumidor de datos.

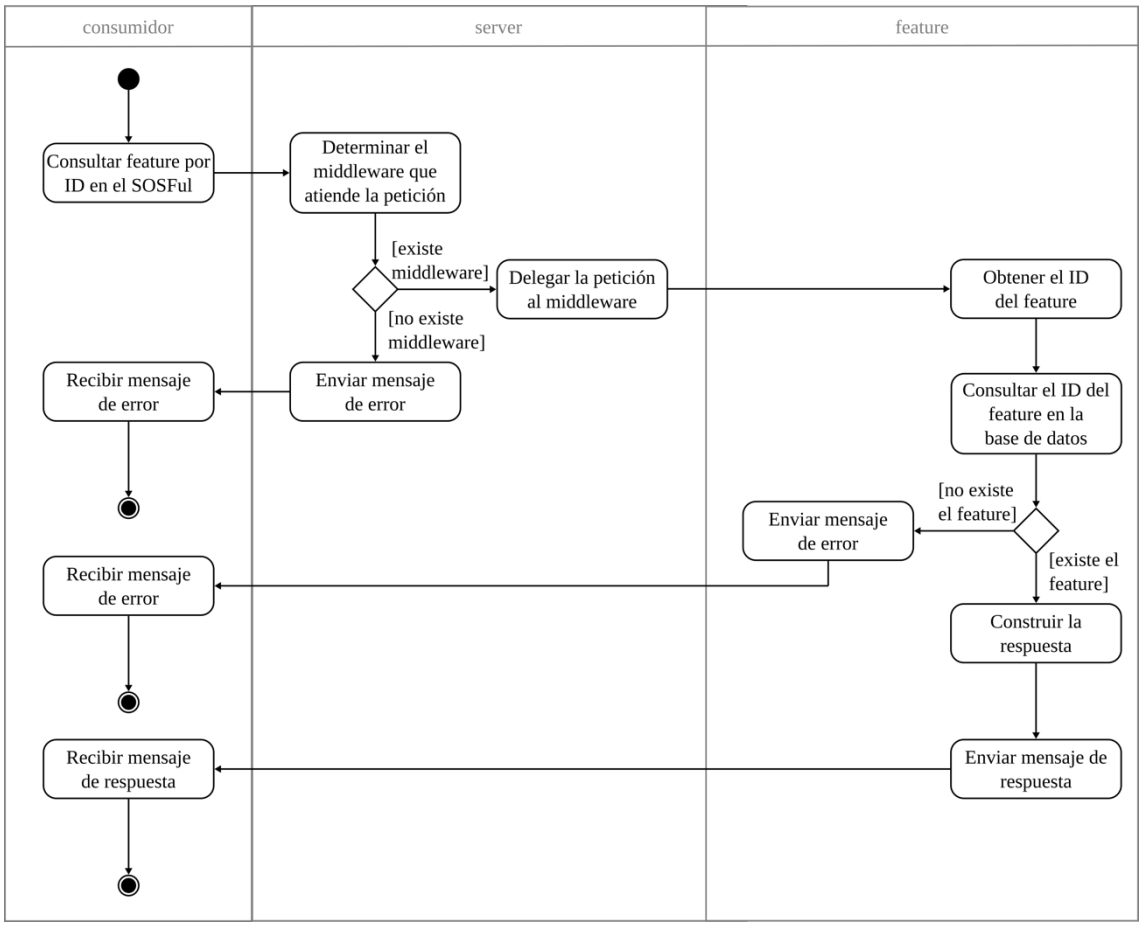


Figura 71 - DA del SOSFul:Consultar feature por identificador

5.3.3.d. Consultar feature por query

Como en la operación anterior el consumidor inicia la petición y es el destinatario de la respuesta. En esta operación existe la posibilidad de que se envíen un filtro espacial (*spatialFilter*); el cual, permite obtener todas las *feature* que se encuentran contenidos dentro de un polígono pasado como parámetro del filtro. Así, el consumidor utilizando el verbo GET sobre la URI /feature puede pasar el filtro dentro del parámetro "q" (de *query*). El *middleware* encargado construye la consulta en la base de datos basada en los filtros y retorna la respuesta con un arreglo JSON con los datos de los *feature*; si no encuentra ningún *feature* que cumpla con los filtros se envía un arreglo vacío.

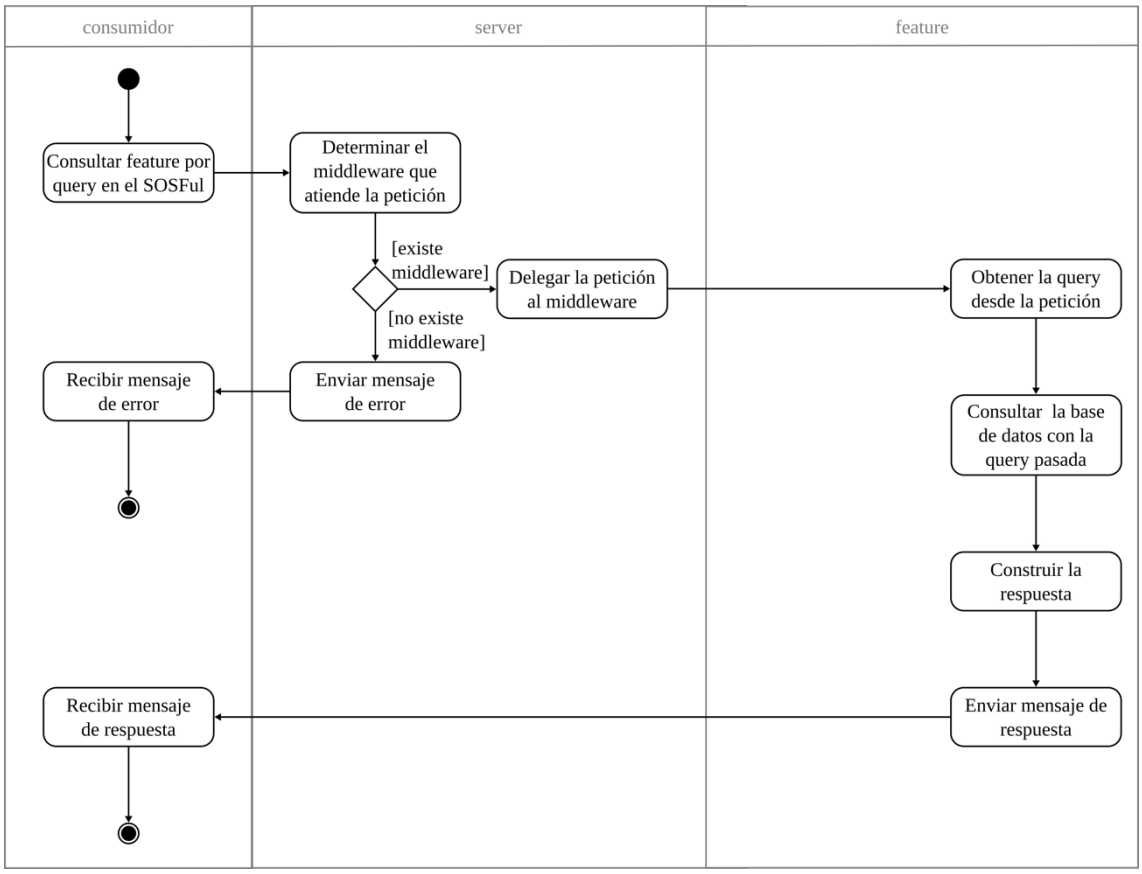


Figura 72 - DA del SOSFul: Consultar feature por query

5.3.3.e. Actualizar feature

Las operaciones de actualización son las menos comunes dentro de las redes que emplean en la actualidad alguna implementación de SOS. Como se observa en este diagrama y en los correspondientes a las actualizaciones de sensores y observaciones son las que más pasos conllevan; sin embargo, al no hacer uso del sistema de ficheros, dentro del SOSFul, estas operaciones son realmente rápidas. En el caso de la actualización de *feature*, el proveedor inicia la operación pasando en la URI el identificador del feature que desea actualizar y en el cuerpo de la petición lleva la nuevos datos que tendrá el feature; para esto emplea el verbo PUT sobre la URI /feature/:id. Cabe anotar que en las operaciones de actualización se remplaza todo el documento de la base de datos.

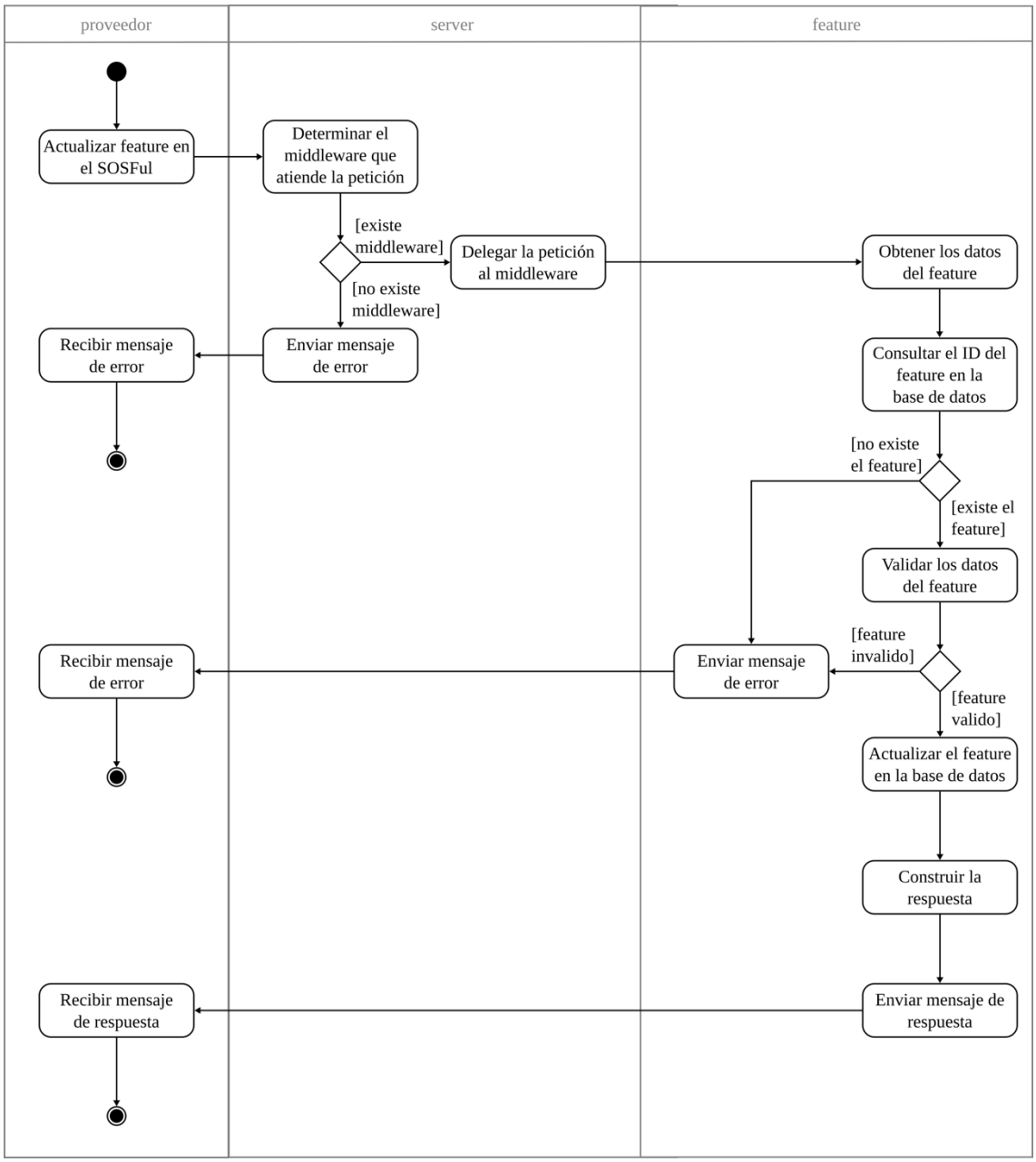


Figura 73 - DA del SOSFul: Actualizar feature

5.3.3.f. Eliminar feature

La última operación sobre *feature* es la de eliminación. Un proveedor de datos puede iniciar esta operación, pasando el identificador del *feature* que desea eliminar y como respuesta obtiene un mensaje de confirmación; se emplea el verbo DELETE sobre la URI /feature/:id. Esta información es recibida por el server que delega la petición al *middleware feature* quien se encarga de eliminar el *feature* de la base de datos y construir la respuesta de confirmación.

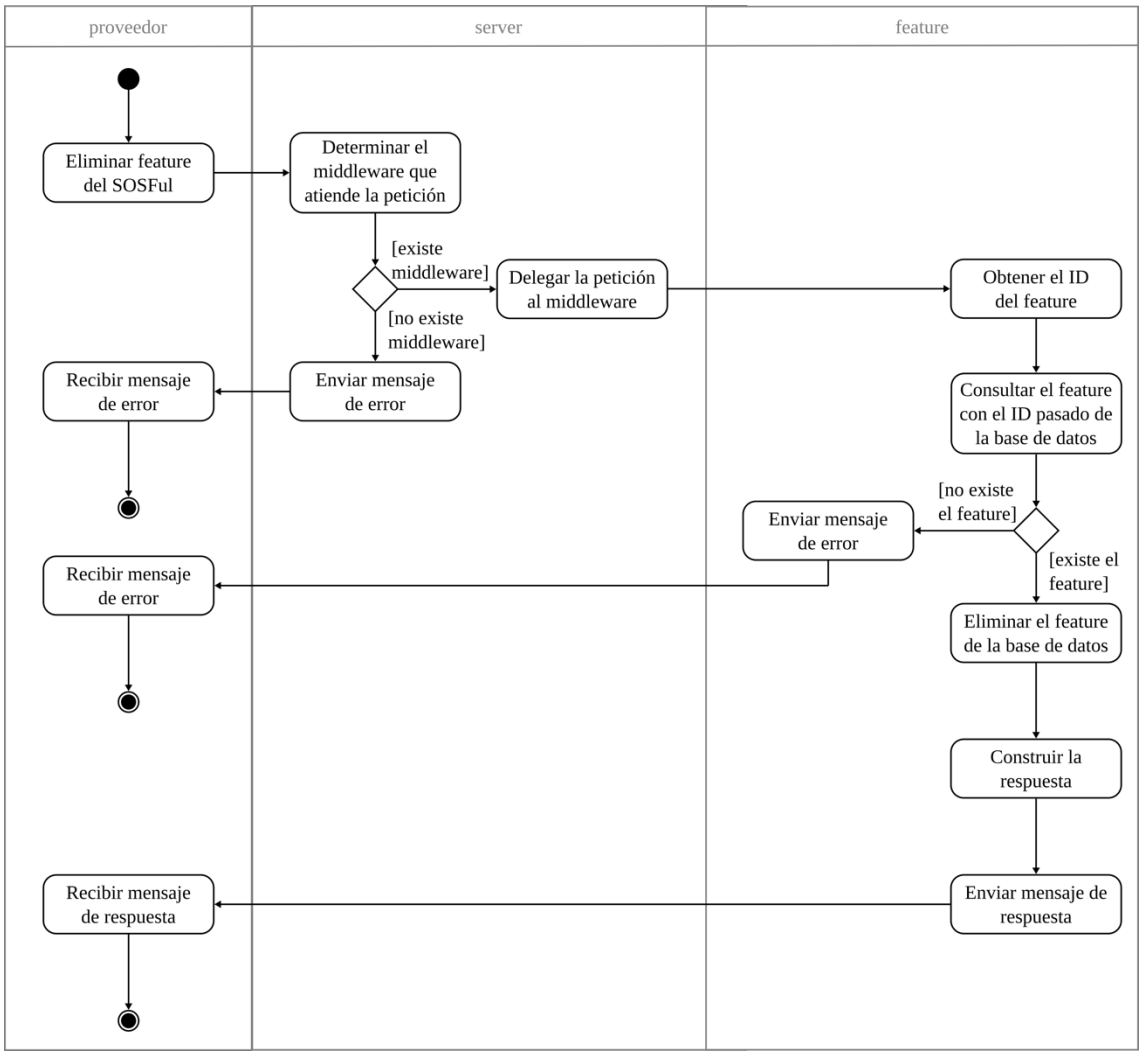


Figura 74 - DA del SOSFul: Eliminar feature

5.3.3.g. Insertar sensor

Insertar un sensor no difiere en gran manera del ingreso de un *feature*. En una primera instancia el proveedor envía los datos del sensor en formato JSON, empleando el verbo POST sobre la URI /sensor. Con esta URI el servidor reencamina la petición hacia el *middleware* sensor, el cual valida los datos JSON pasados en la petición, verifica que el feature asociado este creado en la base de datos y acto seguido agrega el sensor a la base de datos. Finalmente crea una respuesta con el identificador asignado al sensor y se la envía al proveedor quien está en capacidad de recibir esta respuesta.

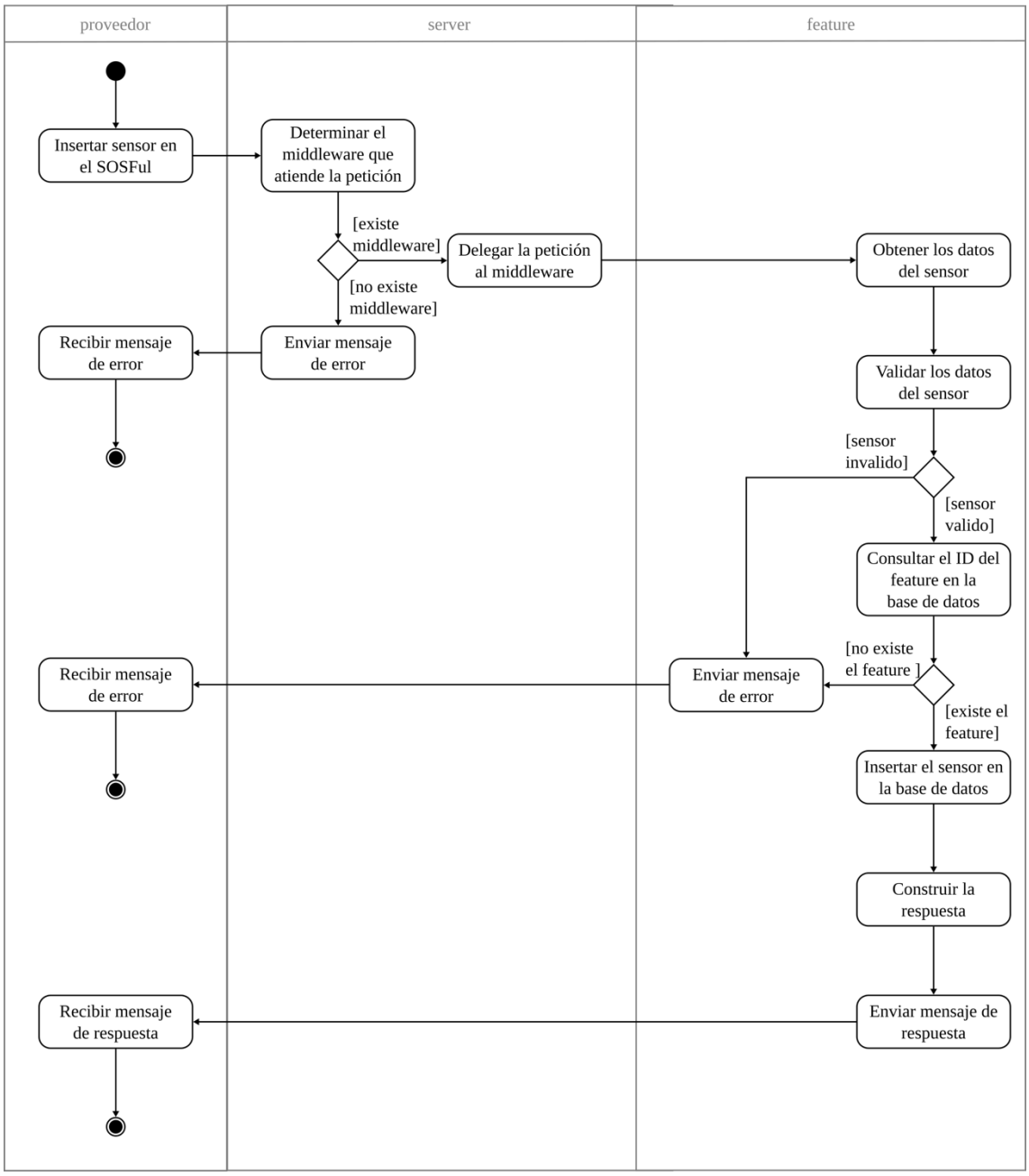


Figura 75 - DA del SOSFul: Insertar sensor

5.3.3.h. Consultar sensor por identificador

Las consultas son realizadas por el consumidor, en este caso se envía el identificador codificado dentro de la URI, así: /sensor/:id; además, se emplea el verbo GET para codificar esta operación. En este caso el *middleware* sensor es quien tiene la responsabilidad de responder a esta petición; para esto, lee el ID pasado y realiza la consulta en la base de datos para construir una respuesta con los datos del sensor devueltos en formato JSON por la base de datos.

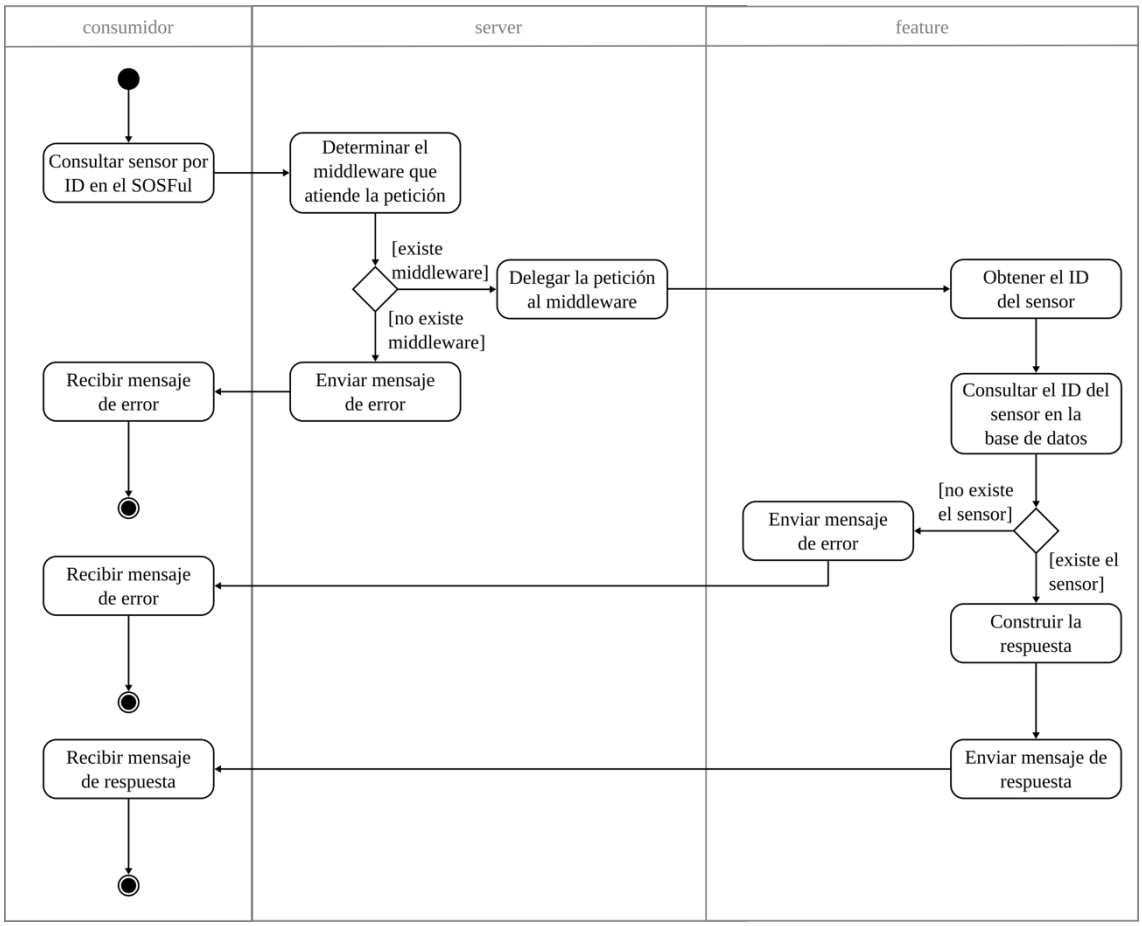


Figura 76 - DA del SOSFul: Consultar sensor por identificador

5.3.3.i. Consultar sensor por query

Como en el caso anterior, el proveedor se encarga de iniciar la operación empleado el verbo GET y empleado la URI /sensor; a la cual, se le pasa el parámetro q (de *query*) en el cual se presentan los filtros que se desean aplicar a la consulta, los cuales pueden ser: *spatialFilter*, *observedProperty* y *featureOfInterest*. El filtro espacial (*spatialFilter*) permite obtener todos los sensores que se encuentren dentro de un polígono pasado como parámetro; por su parte, el filtro de propiedad observada (*observedProperty*) permite obtener todos los sensores que en sus salidas (*outputs*) contengan una de las propiedades pasadas en el filtro; finalmente, el filtro de área de interés (*featureOfInterest*) permite obtener todos los sensores que miden una de las áreas de interés pasadas. Los filtros pueden usarse de forma combinada y en caso de no existir el parámetro q en la URI se envían todos los sensores registrados en la base de datos como un arreglo de sensores en formato JSON. Por su parte, si no existen sensores que cumplan con el filtro la operación retorna un arreglo JSON vacío.

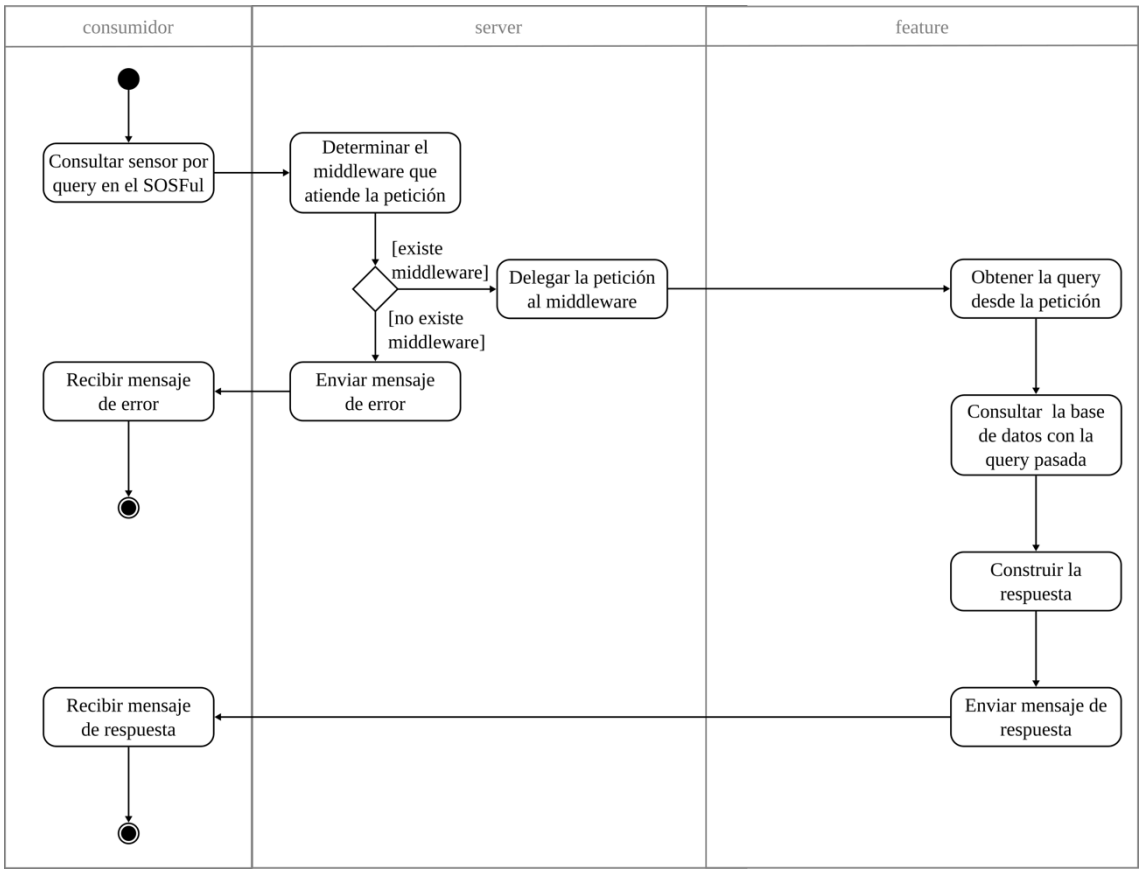


Figura 77 - DA del SOSFul: Consultar sensor por query

5.3.3.j. Actualizar sensor

Para actualizar un sensor el proveedor de datos envía como cuerpo de la petición los nuevos datos del sensor y en la URI indica el identificador del sensor a modificar; emplea el método PUT sobre la URI /sensor/:id. El *middleware* sensor, verifica que el identificador exista en la base de datos, comprueba que los datos pasados en el cuerpo de la petición sean valido y verifica que el feature asociado exista de antemano; para posteriormente, hacer la actualización y enviar la respuesta confirmando el cambio.

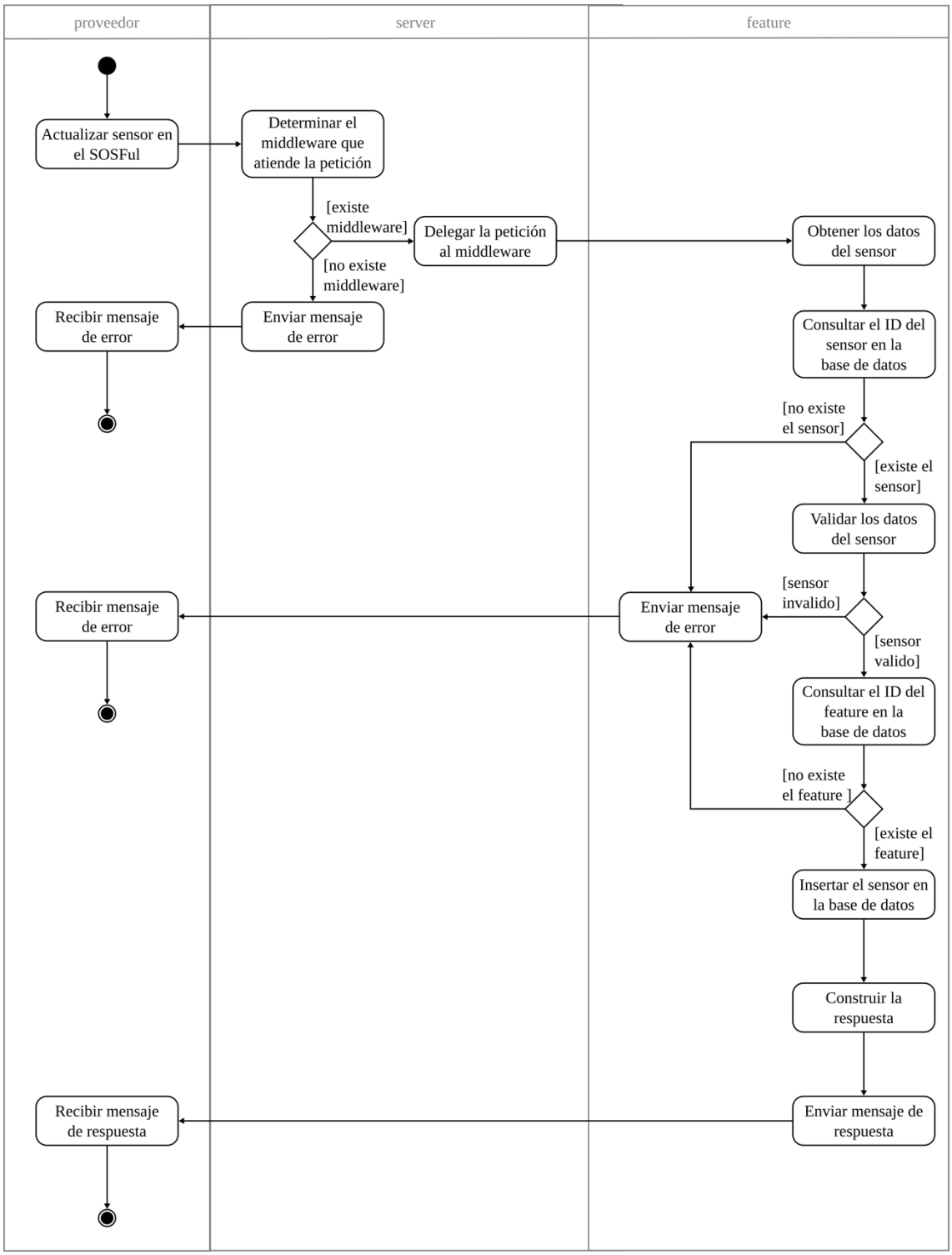


Figura 78 - DA del SOSFul: Actualizar sensor

5.3.3.k. Eliminar sensor

Para la eliminación del sensor se emplea el verbo DELETE sobre la URI /sensor/:id; con esta URI el server delega la petición al *middleware* sensor, quien lee el ID del sensor que se desea eliminar y procede a su borrado de la base de datos y finalmente envía el mensaje de confirmación al proveedor quien inició la operación.

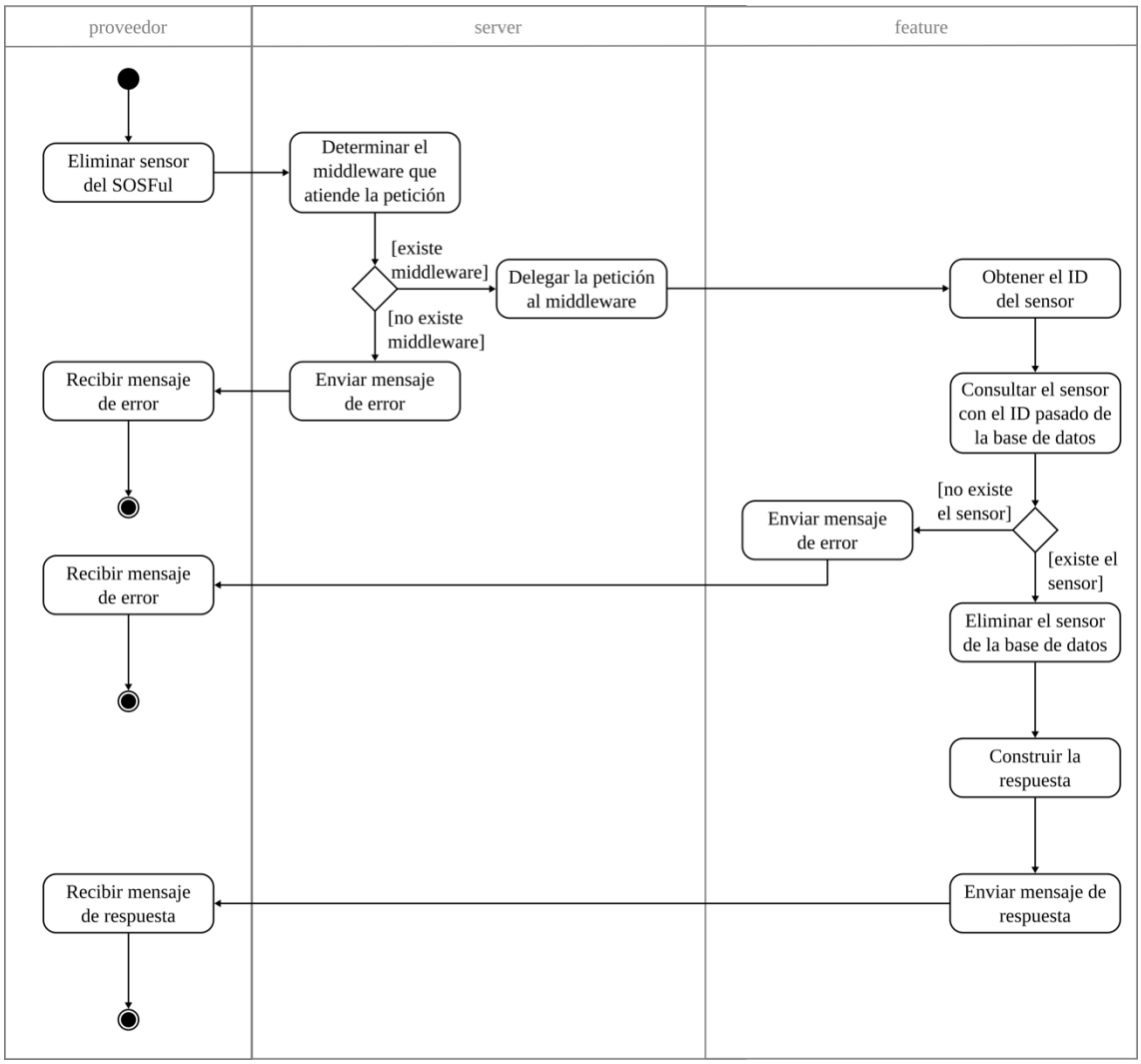


Figura 79 - DA del SOSFul: Eliminar sensor

5.3.3.l. Insertar observación

La inserción de la observación es iniciada por el proveedor de datos, quien usando el verbo POST sobre la URI /observation, envía los datos de la observación en forma JSON, a través del cuerpo de la petición. En este caso el *middleware* observation es a quien el server encamina la petición; este *middleware* valida los datos pasados en la petición y verifica la existencia del feature y del sensor asociados. Posteriormente, agrega la observación a la base de datos y responde al proveedor con el ID asignado a la observación.

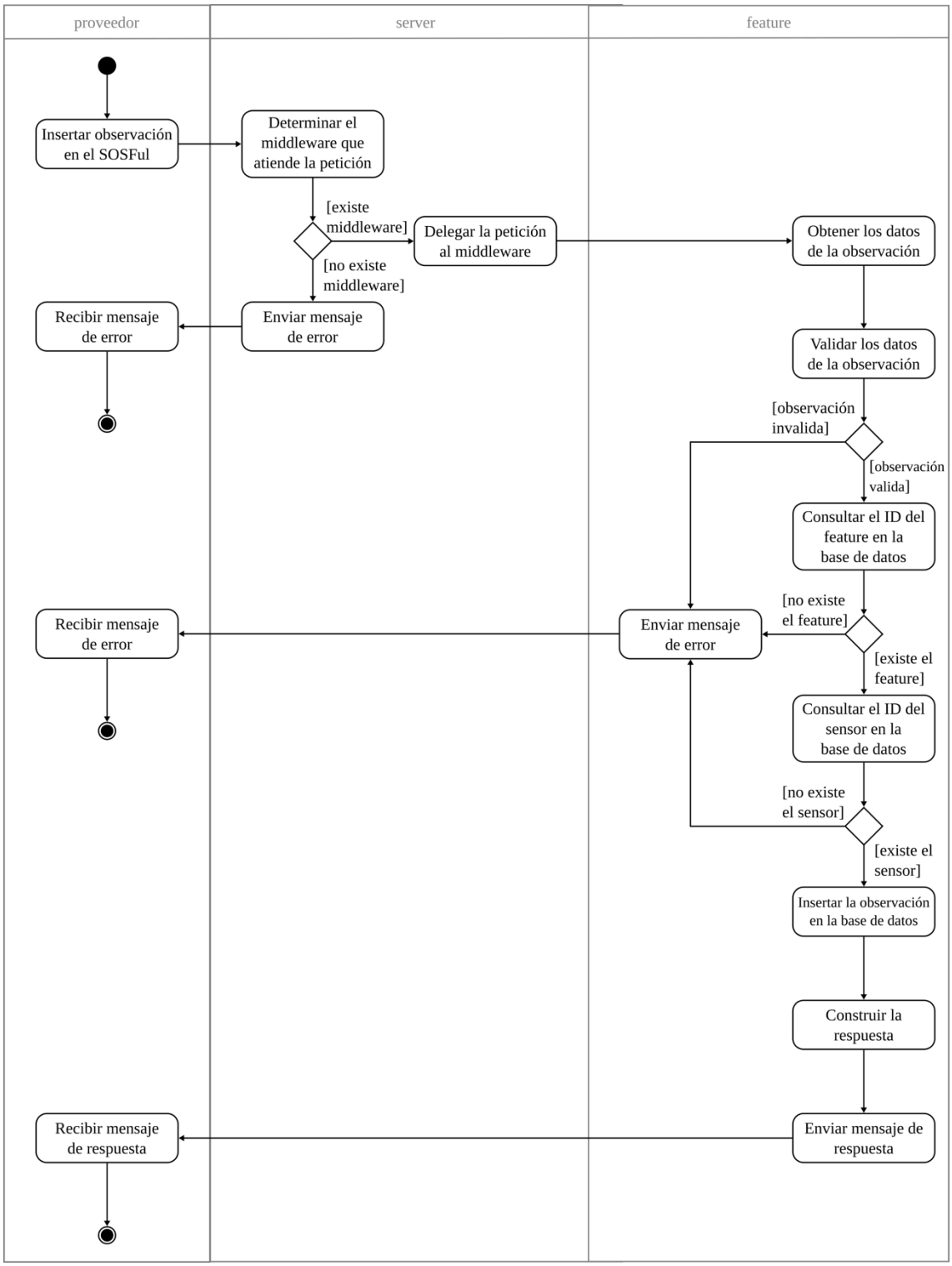


Figura 80 - DA del SOSFul: Insertar observación

5.3.3.m. Consultar observación por identificador

Como en operaciones de consulta anteriores, la complejidad de esta operación es mínima; en este caso el consumidor realiza una petición empleado el verbo GET a la URI /observation/:id con la cual el servidor delega la responsabilidad de la atención sobre el *middleware* observation, el cual se encarga de hacer la consulta a la base de datos y enviar la respuesta en forma JSON al generador de la petición.

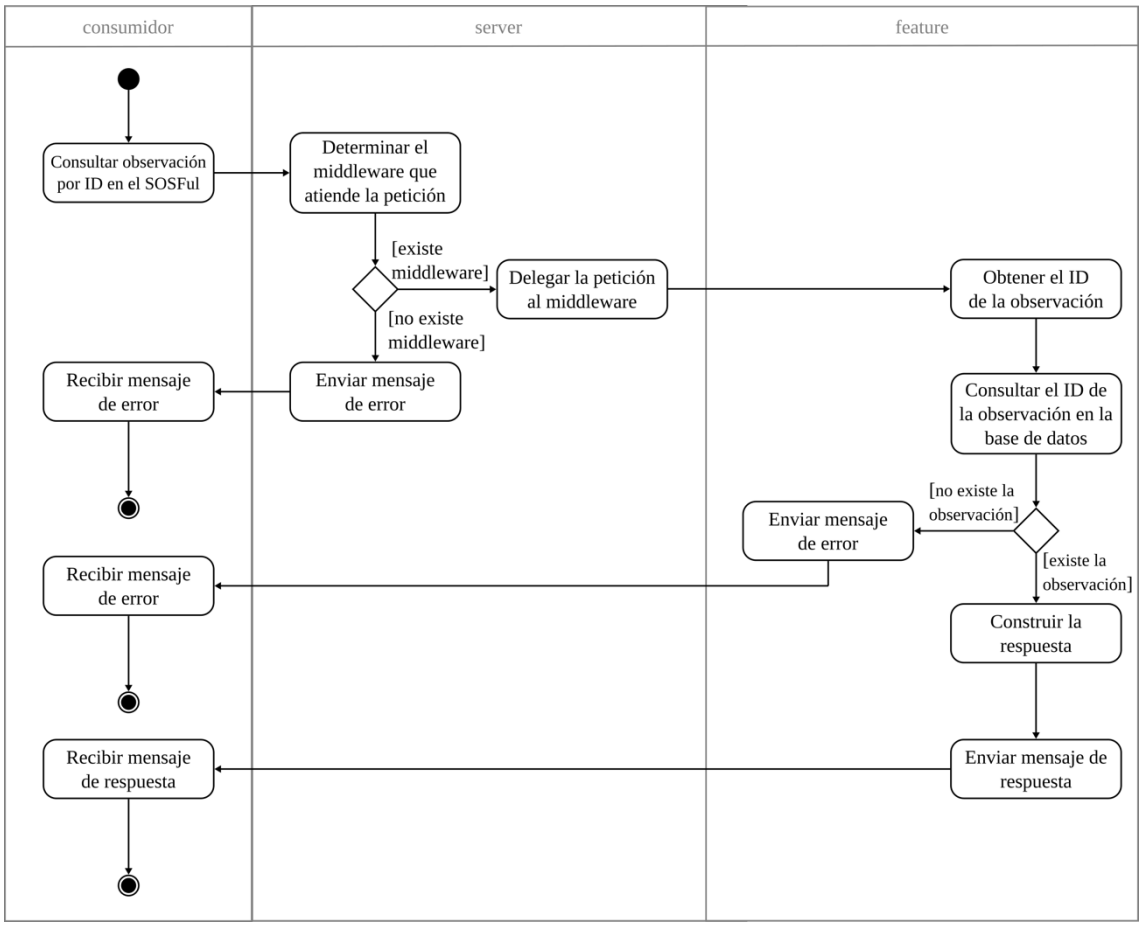


Figura 81 - DA del SOSFul: Consultar observación por identificador

5.3.3.n. Consultar observación por query

Sin duda una de las operaciones más interesantes con las que cuenta el SOSFul, dado que soporta los filtros: *spatialFilter*, *temporalFilter*, *sensor*, *observedProperty* y *featureOfInterest*. Esta riqueza de filtros permite consulta con mucha flexibilidad. El filtro espacial (*spatialFilter*), retorna todas las observaciones realizadas dentro de un polígono pasado como parámetro; el filtro temporal (*temporalFilter*), permite varias configuraciones para obtener el arreglo de todas las observaciones en las cuales el *phenomenonTime* o el *resultTime* (según se indique en el parámetro) se encuentren entre dos fechas (con precisión de milisegundo) o se hayan presentado en un instante preciso (con precisión de milisegundo); por su parte, el filtro de nodo sensor (sensor) retorna todas las observaciones realizadas por los sensores pasados como parámetro; el filtro de propiedad (*observedProperty*) permite obtener las observaciones sobre una propiedad determinada y finalmente el filtro de área de interés (*featureOfInterest*) permite consultar todas las observaciones sobre las áreas de interés pasadas.

Los filtros se pueden emplear en conjunto obteniendo consultas complejas del tipo: quiero todas las observaciones de temperatura y humedad (*observedProperty*), realizadas en Valencia y Barcelona (*featureOfInterest*) entre las 22:34 del 20 de Junio de 2016 y las 14:21 del 22 de Septiembre de 2016 (*temporalFilter*). Si no se encuentran observaciones que cumplan con los filtros se responde con un arreglo vacío; mientras que, si no se pasa el parámetro q con los filtros se responde con un arreglo que contiene todas las observaciones registradas en el SOSFul.

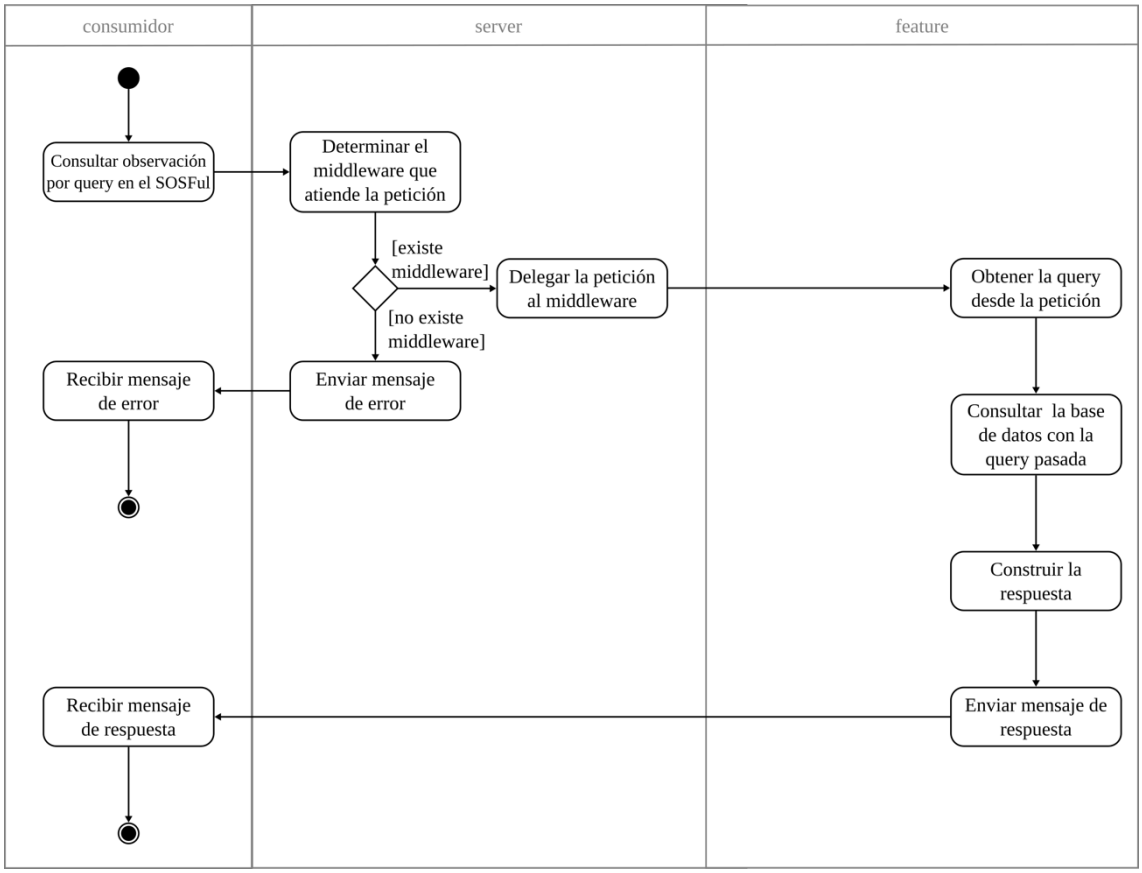


Figura 82 - DA del SOSFul: Consultar observación por query

5.3.3.o. Actualizar observación

La actualización de observaciones es iniciada por el proveedor de datos quien envía el identificador y los datos de la observación que remplazaran los anteriores. Para esto emplea el verbo PUT sobre la URI /observation/:id. El *middleware observation* es el delegado por el servidor para atender la petición, y se encarga de leer los datos pasados, validad que son correctos y verificar la existencia del *feature* y del *sensor* asociados; antes de realizar la actualización en la base de datos.

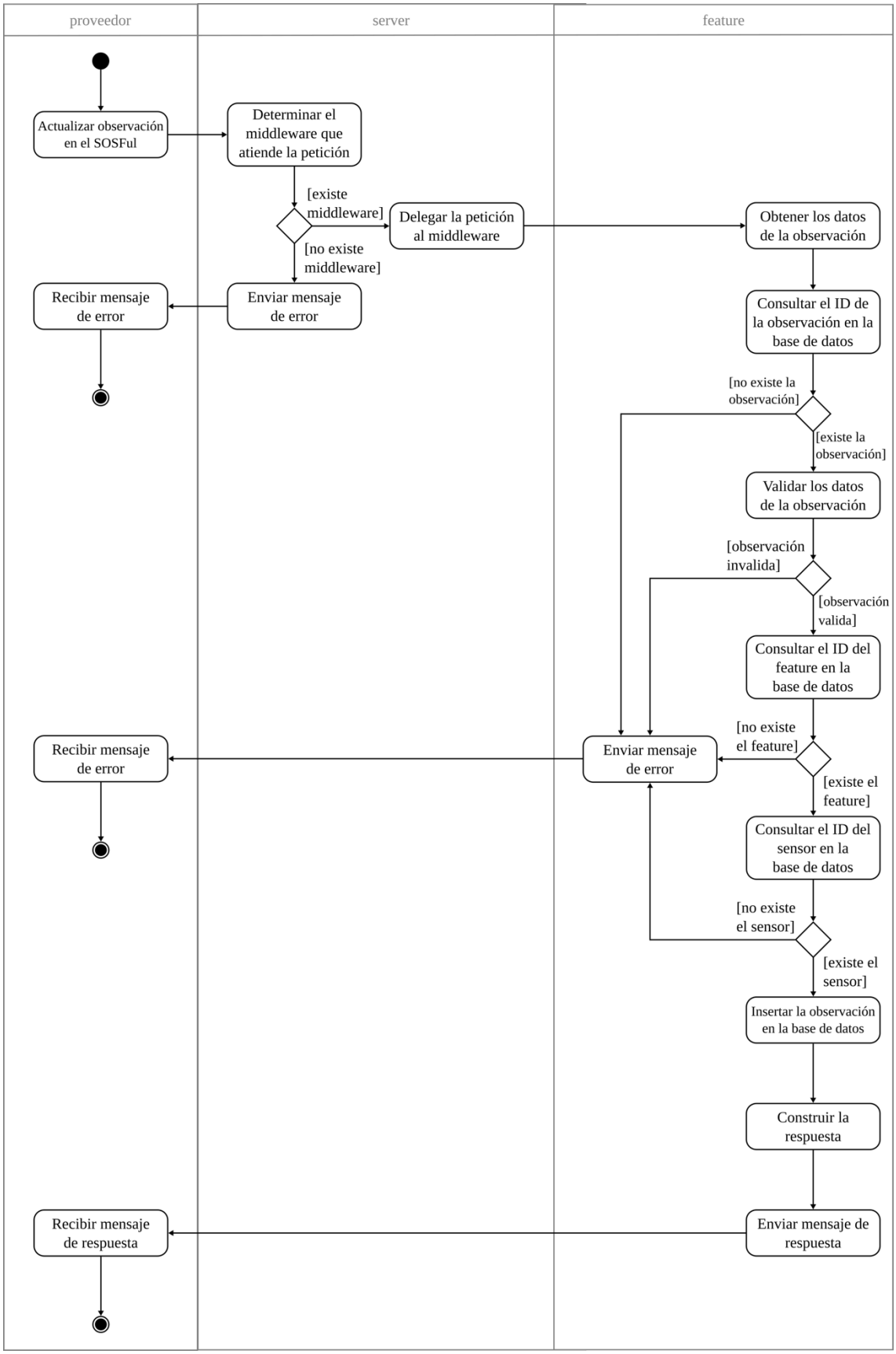


Figura 83 - DA del SOSFul: Actualizar observación

5.3.3.p. Eliminar observación

La última operación que soporta el SOSFul en su núcleo es la de eliminación de observaciones; en este caso basta con el proveedor emplee el verbo DELETE sobre la URI /observation/:id para que *middleware observation* pueda leer el identificador y realizar la eliminación de la base de datos. El *middleware* responde con la confirmación de la operación.

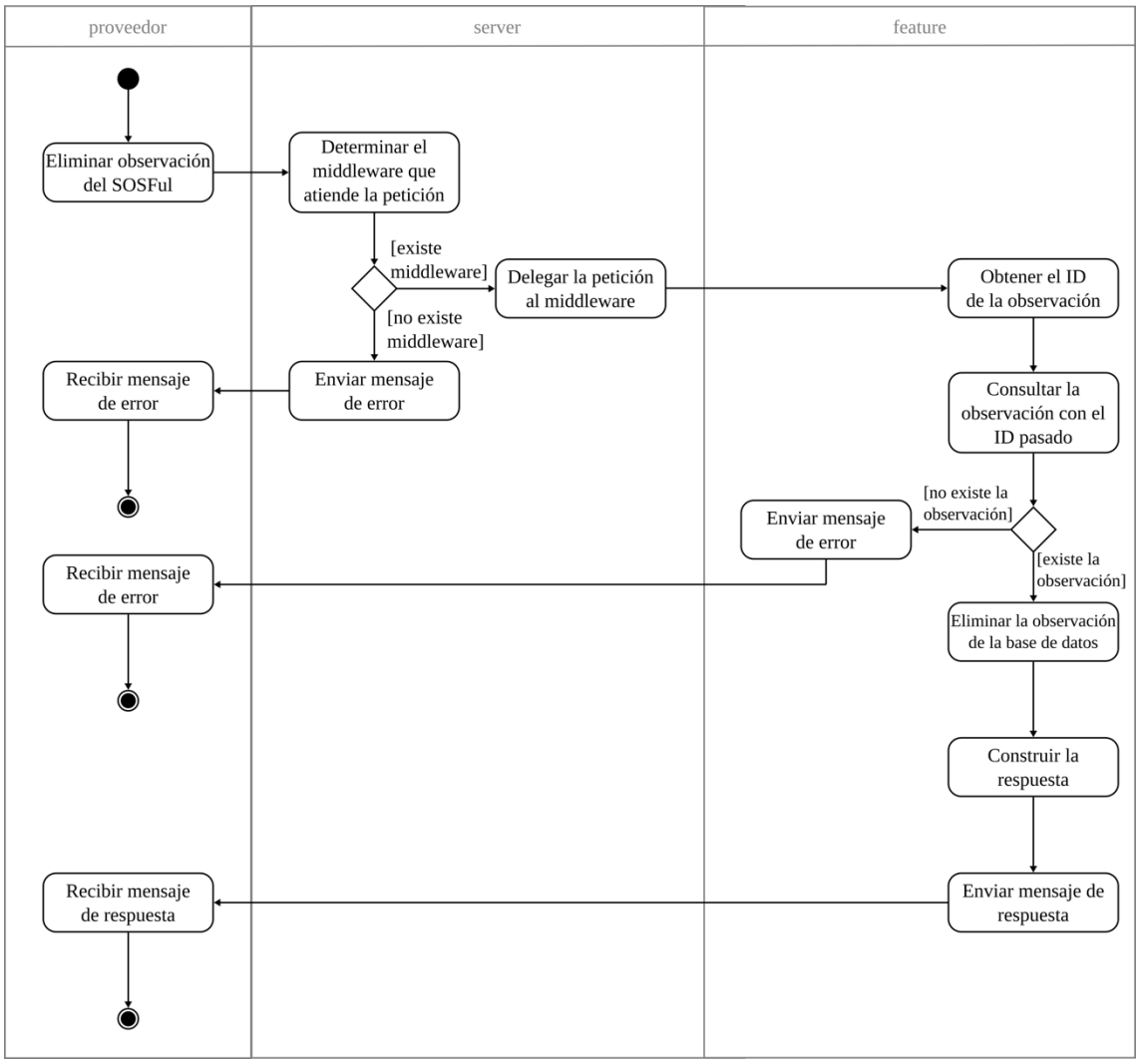


Figura 84 - DA del SOSFul: Eliminar observación

5.4. Implementación

5.4.1. Adaptación del SOS a REST/JSON

El SOSFul toma como referencia la documentación del trabajo futuro planteado en la versión 2.0 del estándar SOS e integra el perfil ligero para crear un servicio de SOS que se adecua a las necesidades actuales de las redes que emplean esta tecnología. Además, se ciñe a los despliegues reales y los resultados obtenidos por el SOSLite para determinar las necesidades reales que debe cubrir y las operaciones que debe brindar. De esta forma, el SOSFul ofrece operaciones CRUD sobre áreas de interés, nodos sensores y observaciones, desde su despliegue. Para encontrar las similitudes y diferencias del SOSFul con el estándar y la implementación de este realizada en esta tesis, la siguiente tabla (Tabla LXI) concreta las operaciones disponibles en estas.

Tabla LXI - Adaptación de las operaciones del SOS

SOSFul			SOS 2.0			
URI	**Verbo**	**Filtro**	**Cuerpo**	**Operación**	**Extensión**	**SOSLite**
/	GET	- section	N/A	GetCapabilities	Core	Si
/feature	GET	- spatialFilter	N/A	GetFeatureOfInterest	Enhanced	No
/feature	POST	N/A	área de interés	N/A	N/A	No
/feature/:id	GET	N/A	N/A	N/A	N/A	No
/feature/:id	PUT	N/A	área de interés	N/A	N/A	No
/feature/:id	DELETE	N/A	N/A	N/A	N/A	No
/sensor	GET	- spatialFilter - observedProperty - featureOfInterest	N/A	DescribeSensor	Core	Si
/sensor	POST	N/A	sensor	InsertSensor	Transactional	Si
/sensor/:id	GET	N/A	N/A	DescribeSensor	Core	Si
/sensor/:id	PUT	N/A	sensor	UpdateSensorDescription	Transactional	Si
/sensor/:id	DELETE	N/A	N/A	DeleteSensor	Transactional	Si
/observation	GET	- spatialFilter - temporalFilter - sensor - observedProperty - featureOfInterest	N/A	GetObservation	Core	Si
/observation	POST	N/A	observación	InsertObservation	Transactional	Si
/observation/:id	GET	N/A	N/A	GetObservaitonById	Enhanced	No
/observation/:id	PUT	N/A	observación	N/A	N/A	No
/observation/:id	DELETE	N/A	N/A	N/A	N/A	No

El SOSFul cubre más operaciones que las provistas por el estándar y las implementadas en el SOSLite. Cabe aclarar que una implementación completa del estándar, difiere del alcance del SOSFul al abarcar una cantidad mayor de casos de uso, basados en la potencialidad completa del estándar y los modelos de datos; sin embargo, tanto el SOSLite como el SOSFul tienen una auto-restricción al estar enfocados en dar servicios a casos más específicos.

Es necesario resaltar que en el SOSFul, al ser un servicio basado en REST, las peticiones son una intersección entre las URI, los verbos (GET, POST, PUT & DELETE), los parámetros de consulta (“q” de query) y el cuerpo del mensaje que contienen las entidades en formato JSON.

Para dar una idea al lector sobre como las operaciones en el SOSFul difieren de las del SOSLite y en definitiva, como son más sencillas y menos verbosas, se presentan a continuación, las mismas operaciones que se evaluaron en el capítulo del SOSLite, para ofrecer una comparación directa se emplearán los nombres de las operaciones del

estándar; se sugiere revisar la tabla anterior (Tabla LXI).

5.4.1.a. GetCapabilities

Es la operación de auto-descripción del sistema que permite conocer las funcionalidades que este ofrece. En esta operación como parámetro "q", se pueden enviar las secciones que se desean como respuesta a la petición, además dado que es una consulta se emplea el verbo GET; mientras que, el formato tanto del parámetro "q", como el de la respuesta, es JSON.

Tabla LXII - SOSFul GetCapabilities atributos petición/respuesta

Mensaje	Filtro o Cuerpo	Descripción
Petición	Sections	Indica que secciones quieren recibirse en la respuesta. Si no se indica ningún valor la respuesta incluye todas las secciones. Soporta los valore: serviceIdentification, serviceProvider, filterCapabilities y contents
Respuesta	serviceIdentification	Esta sección incluye el título, la descripción, el tipo de servicio, las restricciones de acceso, la versión y un arreglo de las operaciones soportadas (incluyendo su URL y el verbo a emplear)
	serviceProvider	Indica los datos de la organización que despliega el servicio, incluyendo: el nombre de la organización, la URL de la página web y la información de contacto
	filterCapabilities	Contienen la información sobre los filtros soportados sean temporales o espaciales
	Contents	Agrupa la información registrada en el SOS incluyéndola relacionada con: feature, sensor y observation

La siguiente ejemplo muestra como un proveedor o consumidor puede llamar la operación, cuando el SOSFul se encuentra desplegado en la maquina local, y la respuesta completa (con formato sangrado para fácil lectura) que puede obtener.

```
GET http://localhost:3000/
?q={"section":[
 "serviceIdentification"
 ,"serviceProvider"
 ,"filterCapabilities"
 ,"contents"
]}
```

```
1. {
2.   "serviceIdentification":{
3.     "title":{
4.       "eng":"SOSful"
5.     },
6.     "abstract":{
7.       "eng":"SOSful REST/JSON Sensor Observation Service"
8.     },
9.     "serviceType":"VLC:SOS",
10.      "accessConstraints":["NONE"],
11.      "versions":["1.0.0"],
12.      "operations":[
13.        {
14.          "url":"/",
15.          "verbs":[
16.            {
17.              "name":"GET",
18.              "description":{
19.                "eng":"Return the service description"
20.              }
```

```
      }
    ]
  },
  {
    "url":"/sensor",
    "verbs":[
      {
        "name":"GET",
        "description":{
          "eng":"Return all sensors"
        }
      },
      {
        "name":"POST",
        "description":{
          "eng":"Create a new sensors"
        }
      }
    ]
  },
  {
    "url":"/sensor/:id",
    "verbs":[
      {
        "name":"GET",
        "description":{
          "eng":"Return the sensors whit the id"
        }
      },
      {
        "name":"PUT",
        "description":{
          "eng":"Update the sensors whit the id"
        }
      },
      {
        "name":"DELETE",
        "description":{
          "eng":"Remove the sensors whit the id"
        }
      }
    ]
  },
  {
    "url":"/observation",
    "verbs":[
      {
        "name":"GET",
        "description":{
          "eng":"Return all observations"
        }
      },
      {
        "name":"POST",
        "description":{
          "eng":"Create a new observation"
        }
      }
    ]
  },
  {
    "url":"/observation/:id",
    "verbs":[
      {
        "name":"GET",
        "description":{
```

```
87.             "eng":"Return the observation whit the id"
88.           }
89.         },
90.         {
91.           "name":"PUT",
92.           "description":{
93.             "eng":"Update the observation whit the id"
94.           }
95.         },
96.         {
97.           "name":"DELETE",
98.           "description":{
99.             "eng":"Remove the observation whit the id"
100.          }
101.        }
102.      ]
103.    },
104.    {
105.      "url":"/feature",
106.      "verbs":[
107.        {
108.          "name":"GET",
109.          "description":{
110.            "eng":"Return all feature"
111.          }
112.        },
113.        {
114.          "name":"POST",
115.          "description":{
116.            "eng":"Create a new feature"
117.          }
118.        }
119.      ]
120.    },
121.    {
122.      "url":"/feature/:id",
123.      "verbs":[
124.        {
125.          "name":"GET",
126.          "description":{
127.            "eng":"Return the feature whit the id"
128.          }
129.        },
130.        {
131.          "name":"PUT",
132.          "description":{
133.            "eng":"Update the feature whit the id"
134.          }
135.        },
136.        {
137.          "name":"DELETE",
138.          "description":{
139.            "eng":"Remove the feature whit the id"
140.          }
141.        }
142.      ]
143.    }
144.  ]
145. },
146. "serviceProvider":{
147.   "name":"Universitat Politecnica de Valencia (UPV)",
148.   "site":"http://www.upv.es",
149.   "contacts":[
150.     {
151.       "individualName":"Juan Pradilla",
152.       "organisationName":"Universitat Politecnica de Valencia (UPV)",
```

```
      "positionName":"Ph.D. Student",
      "contactInfo":{
        "address":{
          "deliveryPoint":"Camino de Vera, s/n",
          "postalCode":"46022",
          "city":"Valencia",
          "country":"Spain",
          "electronicMailAddress":"juaprace@teleco.upv.es"
        },
        "linkage":"upv.es"
      }
    }
  ]
 },
 "filterCapabilities":{
  "spatial":{
   "operands":["Polygon"],
   "operators":{
    "within":["Polygon"]
   }
  },
  "temporal":{
   "operands":["TimeInstant","TimePeriod"],
   "operators":{
    "during":["TimePeriod"],
    "equals":["TimeInstant"]
   }
  }
 }
}
```

5.4.1.b. InsertSensor

Las operaciones de inserción emplean el verbo POST y según la URI pasada serán atendidas por un *middleware* u otro. En el caso de la inserción de sensores, la petición lleva en su cuerpo la entidad JSON que desea registrar; de forma que no tiene ningún filtro a aplicar. De igual forma, la respuesta se entrega en el cuerpo del mensaje.

Tabla LXIII - SOSFul InsertSensor atributos petición/respuesta

Mensaje	Filtro o Cuerpo	Descripción
Petición	sensor	Contiene el modelo del sensor en formato JSON
Respuesta	identifier	Es el identificador asignado por el sistema al sensor que se ha agregado

Para ejemplificar el uso de esta operación en el SOSFul por parte de un proveedor, se presenta el caso donde se desea ingresar un sensor nuevo al sistema. Como se observa la petición es solo la combinación verbo+URI, mientras la entidad viaja en el cuerpo del mensaje; observe el ahorro de datos transmitidos en la petición en comparación con el SOSLite. Por su parte, la respuesta es solo el identificador del sensor ingresado que se envía en el cuerpo del mensaje.

POST http://localhost:3000/sensor

```
{
   "identifier": "56ab90322c697f063d243bc4"
}
```

5.4.1.c. DescribeSensor

Obtener sensores desde el SOSFul se puede realizar mediante dos operaciones: consulta por ID y consulta por query. En la primera de ellas solo se requiere usar el verbo GET y pasar la URI con el identificador, la misma responderá con un sensor en formato JSON; por su parte, la segunda permite, realizar filtros para obtener un conjunto de sensores.

Tabla LXIV - SOSFul DescribeSensor por ID atributos petición/respuesta

Mensaje	Filtro o Cuerpo	Descripción
Petición	identificador	Es el identificador del sensor del cual se quieren recuperar los datos, se pasa mediante la URI
Respuesta	sensor	Contiene el modelo del sensor en formato JSON y viaja en el cuerpo del mensaje

Tabla LXV - SOSFul DescribeSensor por query atributos petición/respuesta

Mensaje	Filtro o Cuerpo	Descripción
Petición	spatialFilter	Polígono dentro del cual deben encontrarse los sensores solicitados
	observedProperty	Arreglo de propiedades presentes en las entradas o salidas de los sensores que se desean recuperar
	featureOfInterest	Arreglo de áreas de interés a las cuales tienen referencia los sensores pedidos
Respuesta	Arreglo de sensores	Contiene los modelos de los sensores que cumplen con los filtros en formato JSON como cuerpo de a respuesta

Como ejemplo se presentan las peticiones en las dos operaciones para la obtención de sensores. La respuesta se puede ver en la sección de SensorML (JSON), en primer caso se retorna un único sensor, mientras que la segunda operación responde con un arreglo de sensores.

```
GET http://localhost:3000/sensor/56ab90322c697f063d243bc4
GET http://localhost:3000/sensor/
?q={
  "spatialFilter":{
            "operator": "within"
            , "operand": "Polygon"
            , "parameter": {
                        "coordinates": [[[50.0, 7.0],[50.0,11.0],[56.0,11.0],[56.0,7.0],[50.0,7.0]]]
            }
  }
  ,"observedProperty": [5, 4]
  ,"featureOfInterest": ["56ab8fee2c697f063d243bc2", "56ab900f2c697f063d243bc3"]
}
```

5.4.1.d. UpdateSensorDescription

La actualización de un sensor es muy parecida a la operación de ingreso de estos. En la petición se envía como cuerpo del mensaje los datos del sensor que remplazar a los existentes y en la URI se envía el identificador del sensor que se desea actualizar. La respuesta no lleva cuerpo para saber si la operación fue exitosa se emplea el código del mensaje de respuesta.

Tabla LXVI - SOSFul UpdateSensorDescription atributos petición/respuesta

Mensaje	Filtro o Cuerpo	Descripción

Petición	identificador	Es el identificador del sensor del cual se quieren actualizar los datos, se pasa mediante la URI
	sensor	Contiene el modelo del sensor en formato JSON
Respuesta	N/A	N/A

A continuación, se puede ver el ejemplo de una petición para actualizar un sensor en el SOSFul usando el protocolo HTTP, el cuerpo de la petición lleva el modelo del sensor en formato JSON (ver la sección de SensorML). En este caso una respuesta con código 204 en http, indica una operación exitosa sin cuerpo en el mensaje de respuesta.

PUT http://localhost:3000/sensor/56ab90322c697f063d243bc4

5.4.1.e. DeleteSensor

Para eliminar un sensor se pasa el identificador dentro de la URI y se emplea el método DELETE. Como en el caso anterior, la respuesta viene determinada por el código del mensaje de respuesta y no se incluye ningún cuerpo de mensaje.

Tabla LXVII - SOSFul DeleteSensor atributos petición/respuesta

Mensaje	Filtro o Cuerpo	Descripción
Petición	identificador	Es el identificador del sensor del cual se quieren actualizar los datos, se pasa mediante la URI
Respuesta	N/A	N/A

El siguiente ejemplo muestra la petición de eliminación de un sensor del SOSFul local usando HTTP. En este caso una respuesta con código 204 en http, indica una operación exitosa y que no se incluye un cuerpo en el mensaje de respuesta.

DELETE http://localhost:3000/sensor/56ab90322c697f063d243bc4

5.4.1.e. InsertObservation

La operación de ingreso de observaciones es una de las más empleadas por los sistemas que despliegan un SOS en la actualidad. Esta operación envía el modelo de la observación en formato JSON como cuerpo del mensaje y recibe el identificador asignado en el cuerpo del mensaje de respuesta.

Tabla LXVIII - SOSFul InsertObservation atributos petición/respuesta

Mensaje	Filtro o Cuerpo	Descripción
Petición	observación	Contiene el modelo de la observación en formato JSON
Respuesta	identifier	Es el identificador de la observación asignado en el sistema

Para ejemplificar, la siguiente instrucción inserta una observación en el SOSFul, el cuerpo sigue el formato O&M/JSON (ver la sección O&M más adelante).

POST http://localhost:3000/observation

```
1.  {
2.     "identifier": "ab90322c697f063d243bc456"
3.  }
```

5.4.1.f. GetObservation

Como en el caso de los sensores existen dos operaciones en el SOSFul que retornan los datos de las observaciones. La primera de ellas consulta una observación puntual por su identificador pasado en la URI. La segunda emplea filtros para determinar un arreglo de observaciones que se está solicitando. Como respuesta se obtiene una o varias observaciones que tienen el identificador pasado o que cumplen con los filtros empleados.

Tabla LXIX - SOSFul GetObservation por ID atributos petición/respuesta

Mensaje	Filtro o Cuerpo	Descripción
Petición	identificador	Es el identificador de la observación de la cual se quieren recuperar los datos, se pasa mediante la URI
Respuesta	observación	Contiene el modelo de la observación en formato JSON y viaja en el cuerpo del mensaje de respuesta

Tabla LXX - SOSFul GetObservation por query atributos petición/respuesta

Mensaje	Atributo	Descripción
Petición	spatialFilter	Filtro espacial que deben cumplir las observaciones que se están consultando. Indica un polígono dentro del cual deben estar las observaciones que se desean consultar
	temporalFilter	Filtro temporal que indica un momento exacto o un intervalo de tiempo en el cual debe encontrarse el phenomenonTime o el resultTime (según se indique) de la observación
	sensor	Es el conjunto de identificadores de todos los sensores para los que se quieren obtener las observaciones
	observedProperty	Conjunto de propiedades que tienen en común las observaciones que se están solicitando
	featureOfInterest	Identificadores de las áreas de interés que contienen las observaciones que se desean consultar
Respuesta	Arreglo de observaciones	Conjunto de observaciones que cumplen con los filtros pasados en la petición, en formato O&M (JSON)

El siguiente ejemplo presenta el uso de las dos operaciones dentro del SOSFul mediante sus respectivas peticiones. La respuesta viene dada en formato JSON siguiendo las indicaciones de la sección O&M (JSON).

POST http://localhost:3000/observation/ab90322c697f063d243bc456

POST http://localhost:3000/observation/
?q={
 "spatialFilter":{
 "operator": "within"
 ,"operand": "Polygon"
 ,"parameter": {
 "coordinates": [[[50.0, 7.0],[50.0,11.0],[56.0,11.0],[56.0,7.0],[50.0,7.0]]]
 }
 }
 ,"temporalFilter":{
 "operator": "equals"
 ,"operand": "TimeInstant"
 ,"parameter": {
 "attribute": "phenomenonTime"
 ,"time": ["2016-01-27T13:00:01.000Z"]
 }
 }
 ,"sensor": ["56ab90322c697f063d243bc4", "56ab90322c697f063d243bc2"]
 ,"observedProperty": [1,2]

```
 ,"featureOfInterest": ["56ab8fee2c697f063d243bc2", "56ab900f2c697f063d243bc3"]
}
```

5.4.2. Adaptación del SensorML a JSON

El SensorML empleado por el SOSLite y recomendado por el estándar SOS puede resultar excesivamente verboso; lo que hace que el consumo del ancho de banda en la red de transporte de datos sea alto. Encontrar un equilibrio entre la riqueza sintáctica que ofrece este modelo de datos y un uso comedido del ancho de banda llevo a transformar los mensajes basados en XML en unos fundamentados sobre JSON. Estos nuevos mensajes simplifican estructuras de XML, como son los arreglos, y evitan realizar importaciones de esquemas.

Para el caso del SOSFul, todas las operaciones de inserción, consulta y actualización de sensores emplean este lenguaje modificado de SensorML/JSON. Los atributos que tiene un sensor modelado bajo la propuesta de esta tesis y empleado dentro del SOS ligero son (Tabla LXXI):

Tabla LXXI – Atributos SensorML/JSON

Atributo	Descripción
description	Es un texto corto que describe el nodo sensor
keywords	Listado de palabras clave que describen el nodo sensor
identification	Conjunto de términos con los cuales se puede identificar el nodo sensor. Puede contener un nombre corto y un nombre largo que faciliten la identificación por parte de las personas
classification	Listado de términos que pueden facilitar la clasificación de un nodo sensor. Por ejemplo, el tipo de sistema de sensores que representa
contacts	Contiene la información de la organización que está operando el nodo sensor; incluyendo la información de contacto y el nombre de la organización
featuresOfInterest	Listado con los identificadores de las featuresOfInterest a las cuales el nodo sensor se encuentra asociado
outputs	Es un listado de todas las posibles salidas del nodo sensor, donde cada una de ellas especifica un nombre, la observableProperty a la que está asociada, la observationType que describe el tipo de dato utilizado y el código UOM que especifica la unidad de medición

A continuación, se presenta un ejemplo, muy cercano al mostrado en el capítulo del SOSLite, que representa un nodo sensor en el lenguaje SensorML/JSON (no se ha realizado ninguna simplificación).

```
1.  {
2.     "description" : "Estación meteorológica de la UPV con sensores de: temperatura y humedad",
3.     "keywords" : ["estación meteorológica", "temperatura", " humedad "],
4.     "identification" : [
5.        {
6.           "label" : "short name",
7.           "value" : "UPV estación"
8.        },
9.        {
10.          "label" : "long name",
11.          "value" : "Estación meteorológica de la UPV"
12.       }
13.    ],
14.    "classification" : [
15.       {
16.          "label" : "sensorType",
```

```
            "value" : "estación meteorológica"
        }
    ],
    "contacts" : [
        {
            "individualName" :  "Juan Vicente Pradilla",
            "organisationName" :  "Universitat Politècnica de València",
            "positionName" : "Investigador Asociado",
            "contactInfo" : {
                "address" : {
                    "deliveryPoint" : "Camino de Vera, s/n",
                    "postalCode": "46022",
                    "city" : "Valencia",
                    "country" : "España",
                    "electronicMailAddress" : "juaprace@teleco.upv.es"
                },
                "linkage": "upv.es"
            }

        }
    ],
    "featuresOfInterest" : ["56b478b7d3cdd69a173af21e"],
    "outputs" : [
        {
            "name" : "humidity",
            "uom_code" : "%",
            "observedProperty": "humidity",
            "observationType" : "Number"
        },
        {
            "name" : "temperature",
            "uom_code" : "Cel",
            "observedProperty": "temperature",
            "observationType" : " Number "
        }
    ]
}
```

5.4.3. Adaptación del O&M a JSON

El modelo O&M basado en XML es empleado dentro del SWE y dentro del SOSLite. Es extensible pero para proveer riqueza semántica emplea una gran cantidad de atributos que lo hacen pesado para la transferencia eficiente de datos en las redes que emplean el SOS. Por este motivo, el SOSFul realiza una adaptación del O&M a JSON, partiendo del perfil ligero, utilizado en el SOSLite, se crea una estructura liviana y semánticamente equivalente, que representa las observaciones realizadas por los nodos sensores.

Las tipos de observaciones se encuentran recogidos en la Tabla LXXII, mientras que los atributos se presentan en la Tabla LXXIII.

Tabla LXXII - Tipos de observaciones del SOSFul

Tipo	Descripción	Ejemplo
Number	Numérico	75
Date	Fecha con precisión de milisegundos	2016-01-27T13:00:01.000Z
Boolean	Booleano	true
String	Cadena de texto	“id:ax500”
Buffer	Objeto en formato binario	01001000010001010100110001000

Tabla LXXIII – Atributos O&M/JSON

Atributo	Descripción
type	Tipo de observación
sensor	Contiene el identificador del nodo sensor que genero la observación
observedProperty	Es el identificador de la propiedad que está monitorizando
featureOfInterest	Representa el identificador del featureOfInterest al cual está vinculada la observación
phenomenonTime	Describe el instante de tiempo de los datos medidos que contiene la observación
resultTime	Determina el tiempo en el cual los datos estuvieron disponibles. Mayoritariamente es idéntico al *phenomenonTime*
result	Agrupa el valor medido y el código UOM que especifica la unidad de medición

El siguiente modelo en O&M/JSON representa una medición de tipo numérica y es equivalente a la presentada en el capítulo del SOSLite. En esta se hace palpable la concreción aplicada en aras de economía del ancho de banda, al tiempo que la semántica se mantiene.

```
1.  {
2.      "type": "Number",
3.      "sensor": "56ab90322c697f063d243bc4",
4.      "observedProperty": "temperature",
5.      "featureOfInterest": "56ab8fee2c697f063d243bc2",
6.      "phenomenonTime": "2016-05-02T17:47:15+00:00",
7.      "resultTime": "2016-05-02T17:47:15+00:00",
8.      "result": {
9.        "uom_code": "Cel",
10.       "value": 25
11.     }
12. }
```

5.4.4. Herramientas

A partir de la experiencia recolectada en la implementación y pruebas del SOSLite, se realizó una segunda evaluación sobre las herramientas a emplear para la creación de esta nueva versión de SOS ligero. En especial se privilegió la integración con la programación orientada a eventos, la arquitectura orientada a servicios basada en REST y los modelos de datos en JSON. Bajo este marco se seleccionó Javascript como lenguaje de programación, Node.js como servidor de aplicaciones y MongoDB para la base de datos orientada a documentos (Figura 85).

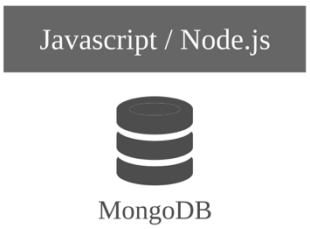


Figura 85 - Herramientas SOSFul

5.4.4.a. Javascript

Basado en la especificación abierta de lenguaje de programación ECMAScript, Javascript comenzó siendo empleado en las páginas web del lado del cliente para brindar dinamismo y permitir una mejor interacción; posteriormente se empleó como base para las aplicaciones web y en la actualidad, ha encontrado un fuerte nicho en la creación de aplicaciones del lado del servidor. Dado su utilización en las páginas web distribuidas empleado el protocolo HTTP, de origen permiten una vinculación directa con la arquitectura REST y por supuesto, con JavaScript Object Notation (JSON).

5.4.4.b. Node.js

Es posiblemente el servidor de aplicaciones con mayor crecimiento en la actualidad y el cual ha impulsado el uso de Javascript de lado del servidor. Es una plataforma de código abierto para la ejecución de lenguajes basados en el estándar ECMAScript y que emplea el motor de ejecución V8 (inicialmente creado para el navegador Chrome). Node.js es un servidor orientado a eventos ideal para manejar un gran número de clientes manteniendo un consumo de recursos bajo.

5.4.4.c. MongoDB

Base de datos basada en documentos enfocada en el rendimiento, el almacenamiento distribuido y la alta disponibilidad. Se aleja de las bases de datos relacionales y almacena los documentos en formato BSON (Binary JSON), el cual, permite una interacción directa con los modelos de datos basados en JSON. Tiene un desempeño excepcional en las operaciones de consulta e inserción cuando los datos se encuentran desnormalizados.

5.5. Pruebas y Evaluación

5.5.1. Escenario de pruebas

En el caso del SOSFul las pruebas se han centrado en la comparativa entre la implementaciones: 52North SOS, SOSLite y SOSFul. Para esto se ha ideado un caso de prueba intensivo en operaciones, con el objetivo de evidenciar el comportamiento en el uso de recursos computacionales y de red, los cuales constituyen los recursos del núcleo de los CPS.

Así, las pruebas están centradas en dos operaciones: InserObservation y GetObservation, o sus equivalentes en el SOSFul (Tabla LXI). Cada una de las pruebas están compuestas por 10 hilos, encargados de hacer las operaciones cada 100 ms, durante 300 segundos, obteniendo así un total de 30.000 muestras por cada una de las operaciones e implementaciones de SOS (Figura 86).

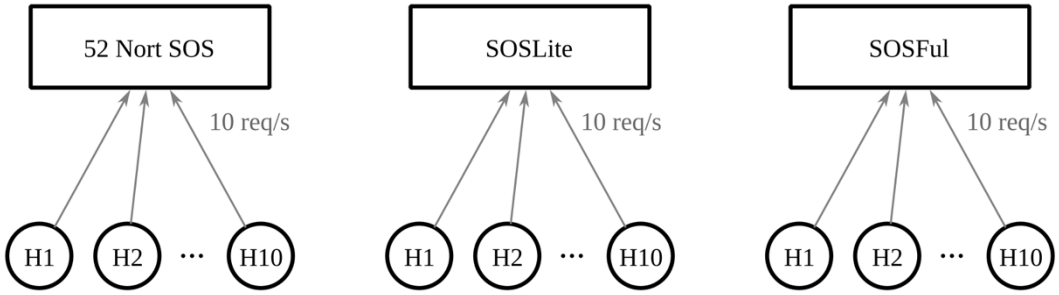


Figura 86 - Escenario de pruebas SOSFul

De esta forma, el tráfico modelado en las operaciones de inserción, coincide con una red de sensores que reporta las observaciones a un nodo central. Por su parte las operaciones de consulta son propias de servicios de análisis de datos en tiempo real, como lo pueden ser los procesadores de eventos complejos (Complex Event Processing – CEP).

En cada uno de los escenarios se mide: el uso de procesador (%), el uso de memoria RAM (%), el *throughput* (kbps), transacciones por segundo (#/s) y tiempo de respuesta (ms). Con estas medidas se caracteriza el rendimiento de las tres implementaciones de SOS en un ambiente de pruebas, con datos generados de forma automática y que siguen la estructura y complejidad propia de los datos que circularían en una red de sensores real, teniendo en cuenta que tanto los modelos XML como los JSON son equivalentes.

5.5.2. Docker

El ambiente de pruebas se ha configurado sobre contenedores Docker, una aproximación más liviana al aislamiento que brindan las VMs. Las pruebas se realizan de forma independiente, haciendo un reinicio programado con cada una de ellas, asegurando así, que el ordenador anfitrión siempre tiene la menor carga de trabajo posible antes de iniciar la ejecución de los contenedores.

Cada una de las implementaciones de SOS fue probada usando una composición de contenedores, donde en uno de ellos se ejecuta el SOS y en el otro su respectiva base de datos (Figura 87). El ordenador anfitrión utiliza como sistema operativo la distribución

Debian 8.3 con un kernel GNU/Linux 3.16, misma distribución que se utiliza en cada uno de los contenedores. Por su parte las versiones utilizadas dentro de las pruebas para los diferentes servicios son: SOSFul 1.0, SOSLite 1.0, SOS 52 North 4.3.6, JAVA 8u45, PHP 5, Tomcat 9.0, NGINX 1.6, Node.js 5.10, PosgreSQL 9.4 y MongoDB 3.2.

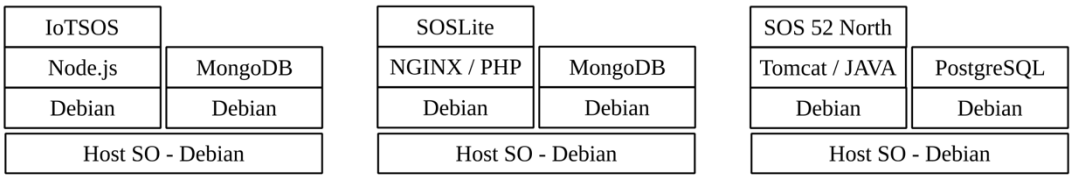


Figura 87 - Composición de contenedores para pruebas de SOSFul

Como herramienta de generación de tráfico y medición de rendimiento se ha utilizado Apache JMeter (versión 2.13) y los conjuntos de extensiones "standard" y "extras" en su versión 1.3.1. De forma que, en el ordenador anfitrión se generan las operaciones y sus respectivos datos, al tiempo que, se miden los parámetros de red; mientras que en cada uno de los contendedores se despliega una sonda para monitorizar el uso de recursos computacionales de los servicios utilizados.

5.5.3. Resultados

5.5.3.a. Uso del procesador

El uso del procesador evidencia un excelente desempeño para el SOSFul, siendo de aproximadamente el 10% para la operación de consulta de observaciones (Figura 88) y de 30% para la inserción de observaciones (Figura 89); a su vez, se mantiene con pocas variaciones a lo largo de la prueba. Por su parte, el SOSFul presenta un buen comportamiento, con un consumo promedio de 40% de CPU para las dos operaciones; aunque en las consultas muestra una mayor variabilidad. Finalmente, el 52North SOS consume mayores recursos que las otras opciones evaluadas para las dos operaciones, cercano 50%.

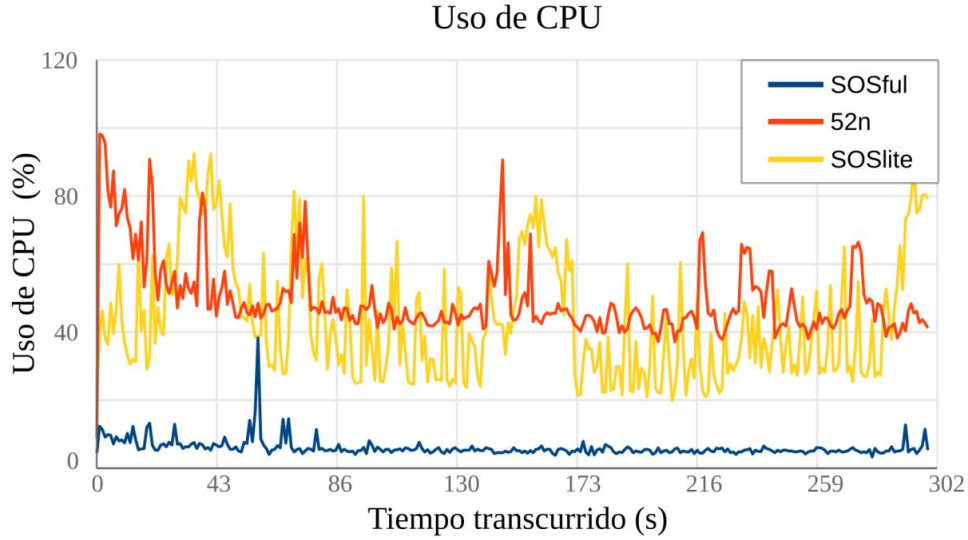


Figura 88 - Uso de procesador: GetObservation

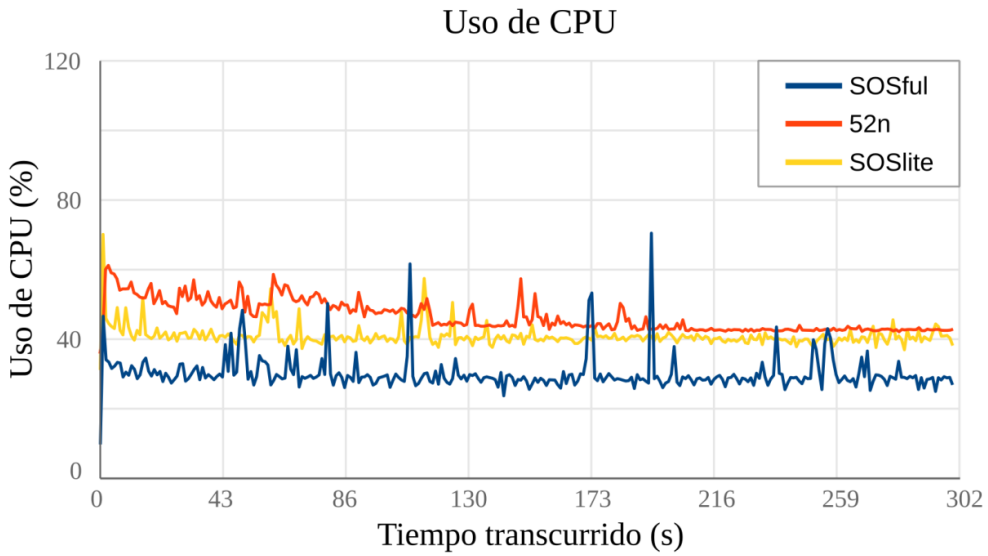


Figura 89 - Uso de procesador: InsertObservation

5.5.3.b. Uso de memoria RAM

En el apartado de uso de memoria de procesamiento, tanto el SOSFul como el SOSLite tienen un consumo muy similar, entorno al 15% para la operación de consulta, mientras que el 52 North consume un 25% en promedio para la misma operación. Para la operación de inserción existen unas ligeras variaciones, siendo el SOSFul un 3% más eficiente que en la operación anterior.

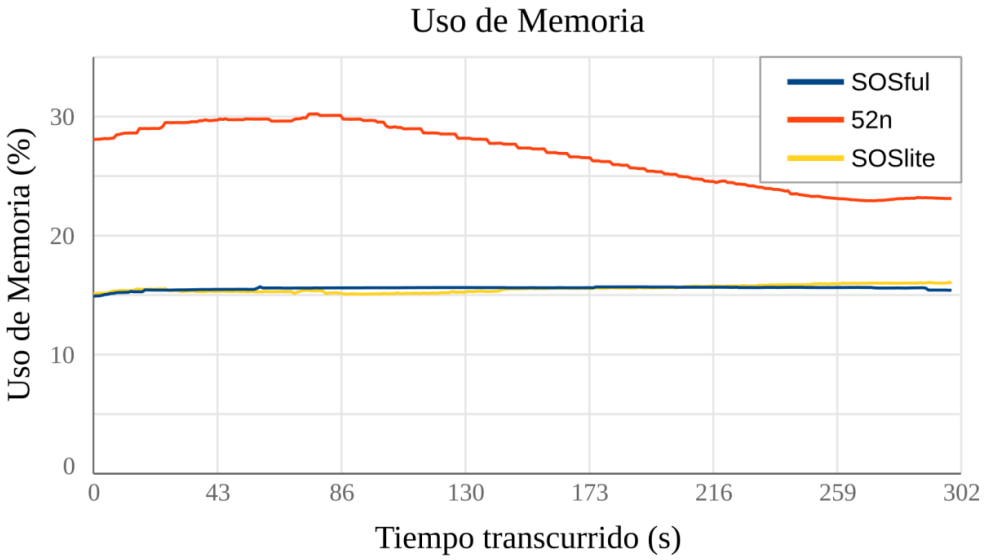


Figura 90 - Uso de memoria RAM: GetObservation

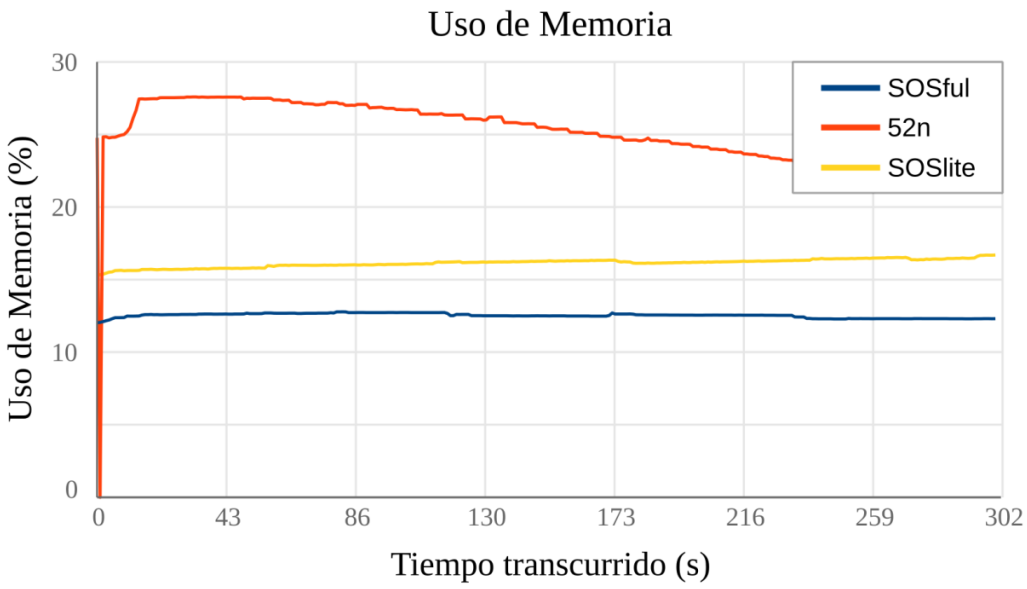


Figura 91 - Uso de memoria RAM: InsertObservation

5.5.3.c. Throughput

En las mediciones del throughput se observa el efecto directo de la modificación de los modelos de mensajes, evidenciando un uso de ancho de más comedido en el SOSFul, 400 kbps para la operación de consulta y 220 kbps en la operación de inserción. Por su parte, el SOSLite y el 52 North en la operación de consulta emplea cerca de 1280 kbps. Mientras que, el SOSLite tiene un throughput de 300 kbps y el 52 North SOS cerca de 360 kbps para la operación de inserción de observaciones.

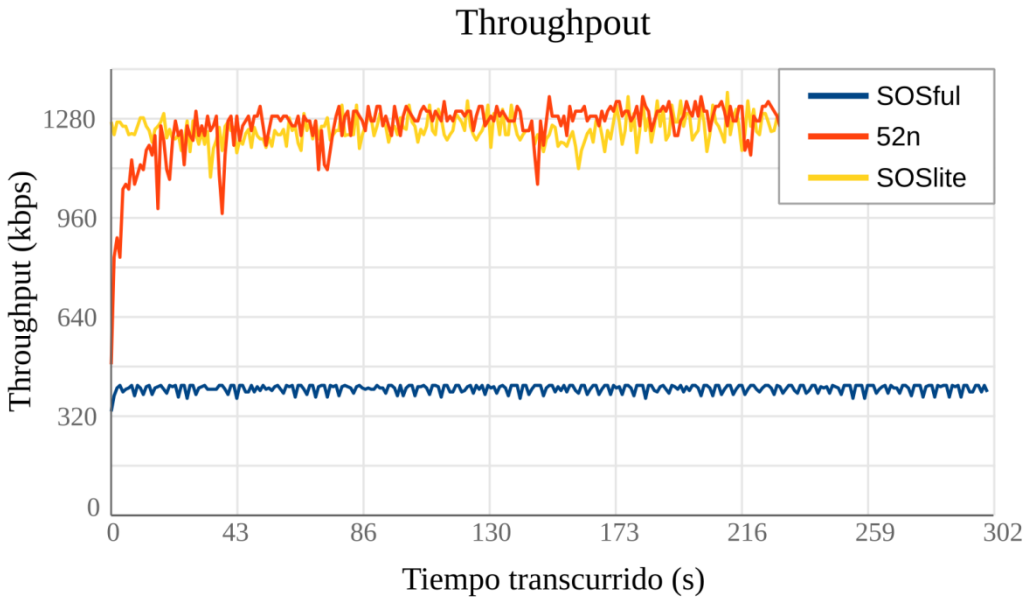


Figura 92 – Throughput: GetObservation

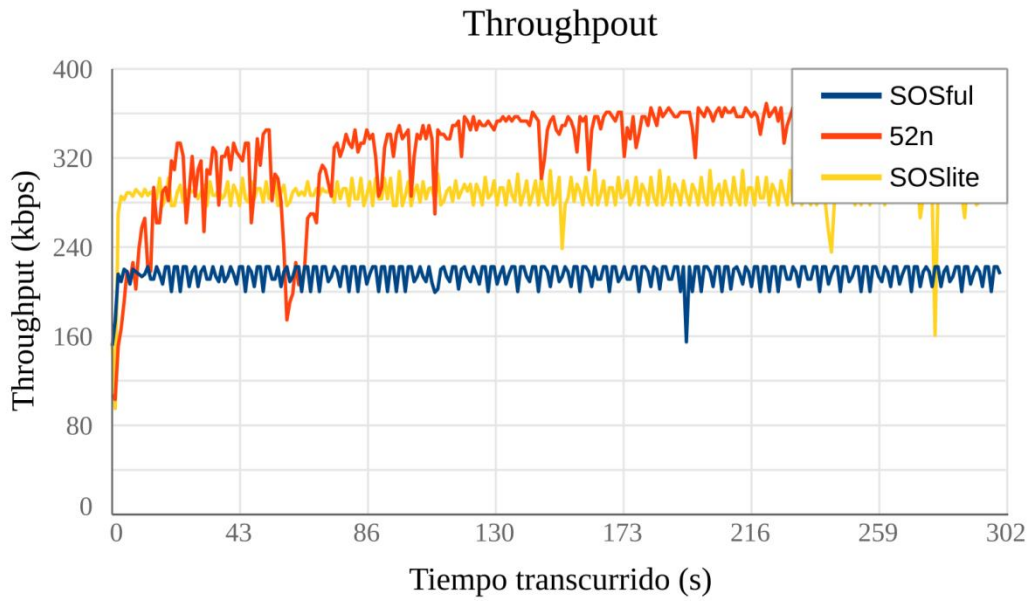


Figura 93 – Throughput: InsertObservation

5.5.3.d. Transacciones por segundo

Las pruebas están planteadas para realizar 100 peticiones por segundo, como se observa el SOSFul atiende dichas solicitudes de forma estable cerca de este límite en las dos operaciones; por su parte el SOSLite tiene un desempeño ligeramente inferior con aproximadamente 95 transacciones por segundo en las dos operaciones; así mismo, el 52 North SOS tiene un desempeño de aproximadamente 90 transacciones por segundo, aunque presenta una mayor variedad que las otras dos opciones evaluadas.

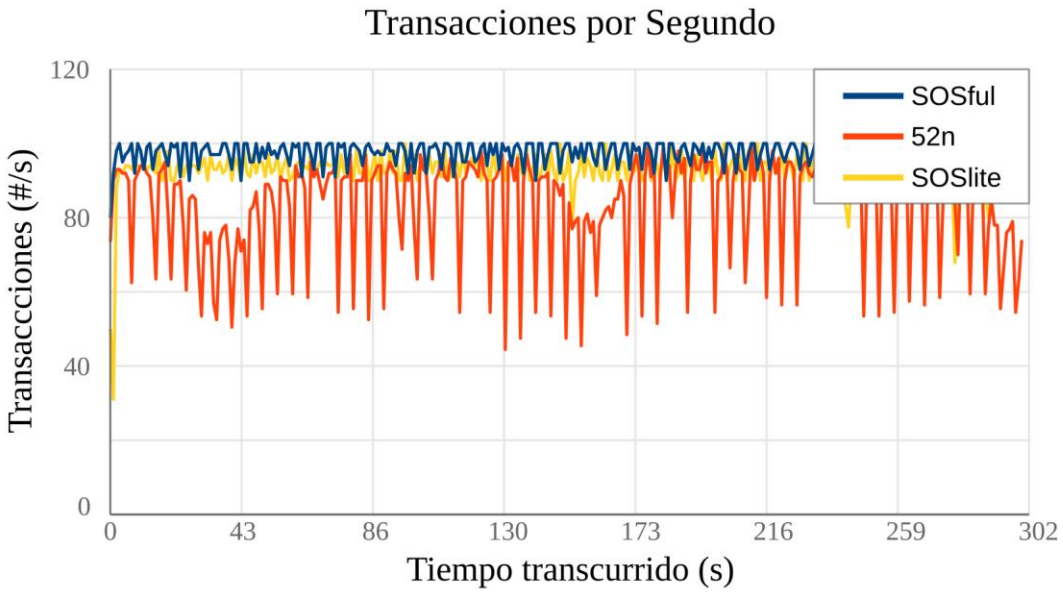


Figura 94 - Transacciones por segundo: GetObservation

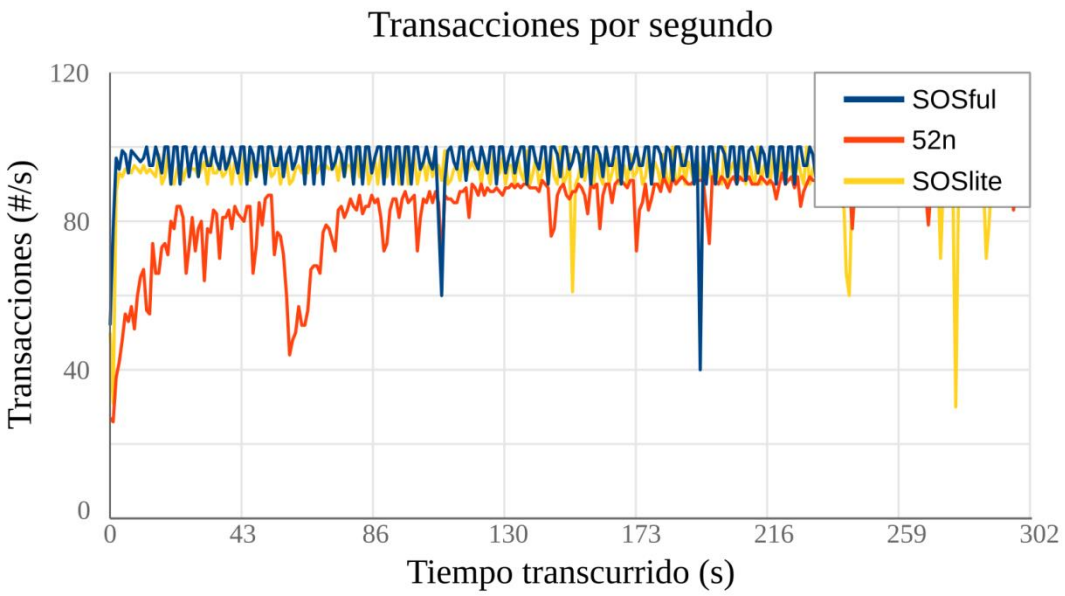


Figura 95 - Transacciones por segundo: InsertObservation

5.5.3.e. Tiempo de respuesta

La última medición realizada esta vinculada al tiempo de respuesta, que en las tres opciones se mantiene por debajo de los 50 ms para las dos operaciones; sin embargo, se debe recalcar que el SOSFul tiene un tiempo de respuesta más bajo y estable; mientras que el 52 North es el que tiene tiempos de respuesta mayores y con más variabilidad.

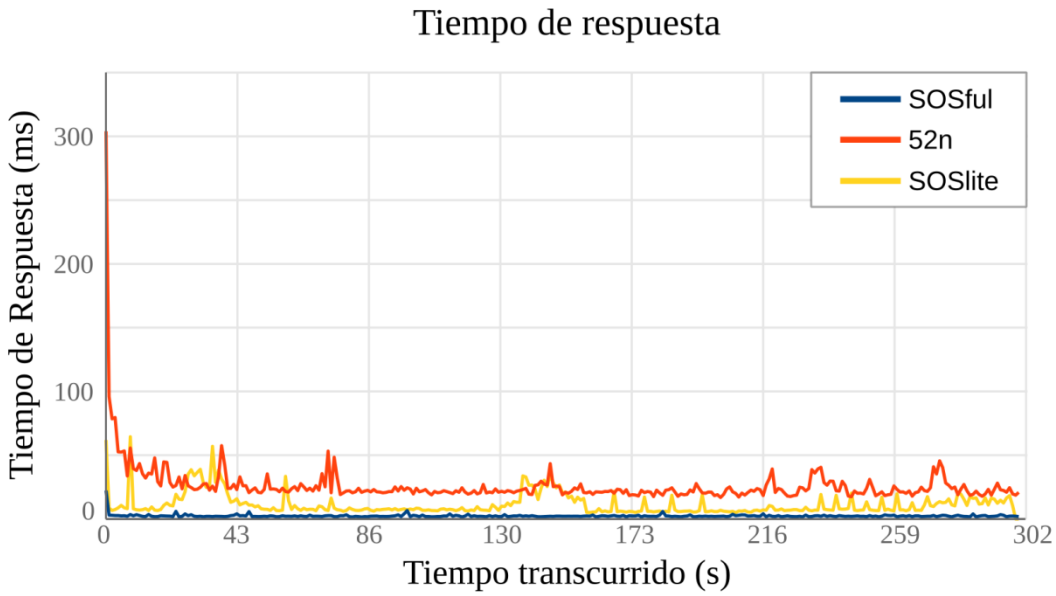


Figura 96 - Tiempo de respuesta: GetObservation

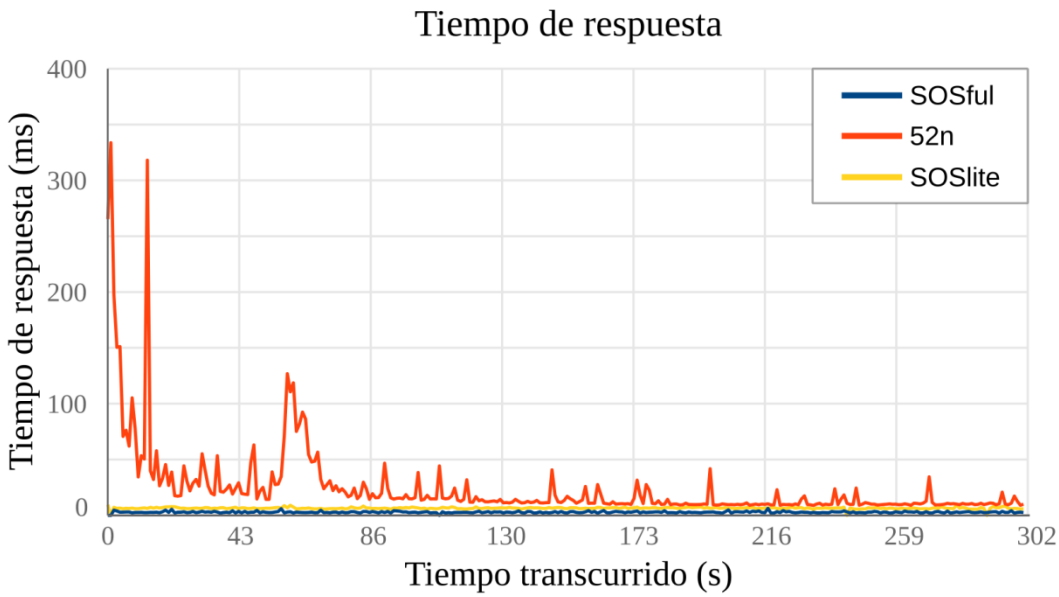


Figura 97 - Tiempo de respuesta: InsertObservation

5.6. Conclusiones

A partir de la experiencia obtenida en el desarrollo del SOSLite, se ha ideado un nuevo SOS ligero que aprovecha de forma integral el núcleo de los CPS, tanto en el uso de recursos computacionales como los de las redes de comunicación. Apartándose del estándar actual, que ha ido quedando rezagado en cuanto al desarrollo de los servicios web, pero sin abandonar la esencia de este servicio, posibilitando una migración inmediata a esta nueva propuesta. El SOSFul es una implementación y a su vez una guía para nuevas versiones del estándar SOS.

El proceso de desarrollo del SOSFul es guiado por una metodología iterativa e incremental que ha permitido hacer mejoras progresivas, desembocando en un producto de software afinado. Todas las decisiones de diseño han sido dirigidas por el análisis realizado del estándar SOS y los productos disponibles en la actualidad; encontrando en el rendimiento y el fácil despliegue dos de las mayores falencias de estos. Por tal motivo, la integración de (1) la arquitectura REST y los mensajes JSON con (2) la simplificación de operaciones y los modelos de mensajes; brinda un modelo de cambio que se adapta a las nuevas necesidades, sin dejar de lado el suplir las ya presentes.

Pero el SOSFul no se queda solo en mejorar el rendimiento, además brinda un conjunto completo de operaciones desde su despliegue, pudiendo manipular los tres conceptos principales del SOS según la necesidad. Aunque pueda parecer trivial, esta característica hace el SOSFul ideal para los entornos distribuidos y aumenta los casos de uso en los cuales se puede emplear.

Para dar respaldo a las mejoras realizadas se han realizado pruebas de laboratorio, donde es notorio el uso comedido de recursos de cómputo y de red que hace el SOSFul. La comparación ha sido realizada con el SOSLite y con una de las implementaciones más empleadas en la actualidad, el 52 North SOS; este último tiene un mayor alcance, pero como evidencian las pruebas, requiere de mayores recursos en el núcleo de los CPS.

Finalmente, el SOSFul es una evolución sobre el SOSLite y el propio estándar SOS; la misma, está encaminada a convertir el SOSFul en el referente para el almacenamiento de los datos de los sensores en los CPS. Para ejemplificar este uso, el siguiente capítulo abordara el funcionamiento del SOSFul dentro de la IoT, uno de los CPS con mayor auge en la actualidad.

6. SOSFul en Internet de las Cosas (IoT)

6.1. Introducción

La implementación SOSFul desarrollada en el marco de la presente tesis, actualiza las implementaciones del estándar SOS, a partir de los resultados obtenidos en las pruebas del SOSLite; para ello simplifica las interfaces de comunicación y los modelos de mensaje. Logrando así, un repositorio de datos y metadatos de sensores y observaciones, diseñado para utilizar eficientemente los recursos del núcleo de los CPS, bien sean de procesamiento o de red; al tiempo que, facilita el despliegue y la integración del mismo con otras tecnologías.

En este capítulo, se profundiza en cómo aprovechar las ventajas del SOSFul dentro de uno de los CPS con mayor auge en la actualidad, la IoT. Para esto, se detalla una arquitectura de referencia de IoT y se determina como el SOSFul se puede integrar dentro de la misma. Finalmente, se lleva el fundamento a un caso práctico por medio de un caso de uso centrado en las ciudades inteligentes (SmartCities).

El propósito del SOSFul es proporcionar un mecanismo que facilite la coexistencia eficiente de los dispositivos y tecnologías heterogéneas vinculados a los sensores de la IoT. Y es que, los sensores son parte fundamental de los objetos inteligentes que conforman la IoT, permitiendo que las aplicaciones y servicios que se ejecutan sobre la IoT perciban el mundo real.

Son varias las implicaciones de lograr exitosamente la interoperabilidad en la IoT, siendo remarcable el impacto en los beneficios económicos potenciales que se vinculan con este reto, que llegan hasta un 40% para el año 2020 [11]. Con esta motivación, abordar la interoperabilidad en la IoT es una de las principales líneas de investigación de la actualidad; tal como en el caso de las SN, comienzan a surgir soluciones a los diferentes niveles, siendo la propuesta de este capítulo una contribución hacia la interoperabilidad semántica.

6.2. SOSFul en la IoT

6.2.1. Arquitectura de referencia para la IoT

Para profundizar en el papel del SOSFul como mecanismo de interoperabilidad en la IoT, se hace necesario emplear una arquitectura de referencia. De esta forma, se cuenta con un marco de desarrollo y unos términos comunes que facilitan el entendimiento y ponen de manifiesto las interacciones del SOS con los otros elementos que convergen en la IoT.

Figura 98 - Arquitectura de referencia IoT

Aunque existen muchas arquitecturas de IoT propuestas, se ha escogida una de cinco capas (Figura 98) [57] por la claridad que aporta, esta separación, al entendimiento general de la IoT y al funcionamiento de SOS dentro de la misma. Esta arquitectura está compuesta por las capas de: percepción, transporte, procesamiento, aplicación y negocio. Las cuales se detallan a continuación:

- Capa de percepción: la principal tarea de esta capa es percibir las propiedades físicas de los objetos mediante sensores y convertir esta información en señales digitales para su fácil transmisión por medio de la red transporte.

- Capa de transporte: se encarga de transmitir los datos recibidos de la capa inferior a los centros de procesamiento, mediante varias redes de comunicaciones.

- Capa de procesamiento: principalmente almacena, analiza y procesa la información de los objetos recibidos mediante la capa de transporte.

- Capa de aplicación: basándose en los datos procesados por la capa anterior integra diferentes aplicaciones de propósito específico para cada industria.

- Capa de negocio: se encarga la gestión de IoT, incluyendo la administración de objetos, redes de comunicaciones, aplicaciones y usuarios.

Cuando se analiza la IoT y se emplea una arquitectura de referencia, es lógico pasar a pensar en cómo diferentes paradigmas de la Internet tradicional se ajustan a la propuesta de la IoT. En el caso del CC, se integra con la IoT para proveer capacidades de procesamiento y almacenamiento virtualmente ilimitados. Esta cualidad es especialmente deseable en el ámbito de la IoT, debido a la enorme cantidad de datos que se generan y deben ser almacenados, procesados y analizados [59]. Por su parte, el FC

ha llegado a apoyar las aplicaciones IoT con necesidades de baja latencia, almacenamiento y procesamiento distribuido, seguridad diferenciada y uso eficaz del ancho de banda de los canales de comunicación.

Así, el CC usado dentro de la IoT permite: la virtualización y composición de componentes IoT, la simulación de partes de la IoT, la asignación dinámica de recursos, las funciones de gobernanza, las capacidades de monitorización y análisis de rendimiento de la infraestructura y aplicaciones, y la coordinación en la asignación de recursos [92]. Por su parte, el uso de FC en la IoT, cumple las mismas funciones que el CC pero siendo más próxima a los nodos de acceso, aumentando la seguridad de los datos sensibles y estando más localizada que el CC [93].

Claramente el FC y CC son paradigmas que ofrecen servicios que quedan enmarcados dentro de las capas de: procesamiento, aplicación y negocio, de la arquitectura propuesta. Es decir, se integra de forma horizontal varias de las capas, ofreciendo: la posibilidad de asignación dinámica de recursos a los componentes de estas y la posibilidad de distribuir sus componentes entre diferentes *sites* (Figura 99).

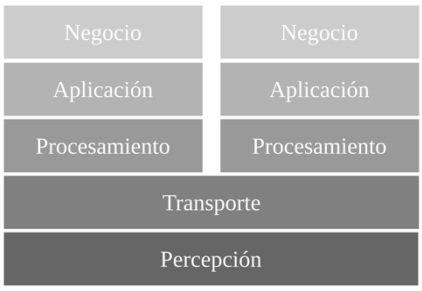


Figura 99 - Arquitectura de IoT con CC/FC

Analizando las funcionalidades correspondientes al SOSFul y el papel que desempeña dentro de la IoT, se puede determinar que pertenece a la capa de procesamiento, dentro de la arquitectura de referencia escogida. Esto es debido a que es un servicio de almacenamiento de datos que provee la capacidad de guardar, actualizar, consultar y eliminar información de los sensores y sus observaciones.

Desde esta capa, el SOS puede ser utilizado como mecanismo de interoperabilidad semántica, habilitando el uso de un gran número de tecnologías en la capa de transporte como: Zigbee, Bluetooth, WiFi, Sigfox, LTE-M, LoRa, LoRaWAN, etc. Y un enorme conjunto de sensores en la capa de percepción, independientemente del fabricante, capacidad y funcionalidad.

El SOSFul desarrollado en la presente tesis puede desplegarse en una variedad amplia de hardware, que abarca desde los dispositivos limitados hasta los *clusters* en *datacenters*; y su funcionamiento ser ofrecido como un servicio sobre el dispositivo o una instancia virtual *Fog* o *Cloud*.

La combinación de hardware y forma de ofrecimiento presenta un conjunto amplio de opciones; sin embargo existen tres casos que son de especial interés, dado que son los más comunes en las implementaciones reales, estos son: servicio sobre dispositivo limitado, instancia FC en un micro-*datacenter* e instancia CC en un *datacenter*. A continuación, se presenta cada uno de estos casos en más detalle.

6.2.2. SOSFul sobre dispositivos limitados

La IoT engloba diferentes tecnologías incluyendo una gran variedad de nodos sensores que perciben el mundo físico y lo transforman en medidas y en una representación virtual. Esta representación se fundamenta en el compendio de observaciones, mediciones de un parámetro de la realidad en un tiempo y espacio delimitados, las cuales son almacenadas y gestionadas para poder obtener información y conocimiento a partir de ellas.

La implementación de SOSFul brinda el repositorio de almacenamiento y los mecanismos de gestión, para los datos y metadatos de las observaciones y los sensores que las generaron. Este servicio de SOS ligero puede desplegarse de muchas formas, incluyendo los dispositivos con recursos computacionales limitados.

Estos dispositivos limitados, forman parte de las redes más inmediatas a los sensores, proveyendo de capacidades diversas a los mismos, tales como ser el *gateway* de acceso a la infraestructura de red o ser el middleware entre diferentes tecnologías de red de los nodos sensores. Que estos dispositivos ya hagan parte de las redes actuales y que su valor de incorporarlos o actualizarlos sea bajo, es una característica fundamental cuando se piensa en la masificación de la IoT.

La implementación de SOSFul ha sido optimizada para aprovechar al máximo los recursos de procesamiento, almacenamiento y ancho de banda; manteniendo a su vez, la independencia tecnológica y la riqueza semántica. Estas particularidades hacen del SOSFul, el servicio de almacenamiento de datos de sensores y observaciones, que más se ajusta a los dispositivos limitados y a las redes donde estos se despliegan.

Por su parte, el resto de componentes que forman parte de las tres capas superiores de la arquitectura de referencia suelen encontrarse en dispositivos diferentes, bien sea en la red local o en Internet. Caso bastante usual es el que las aplicaciones y el procesamiento de datos basado en *BigData*, se encuentren en *datacenters* accesibles mediante Internet, mientras que los datos locales de sensores permanecen de forma temporal en el SOS local.

El SOSFul desplegado sobre dispositivos limitados aplicado a la IoT es un mecanismo intermedio que brinda: interoperabilidad, abstrayendo la heterogeneidad de los nodos sensores mediante mensajes livianos e interfaces de fácil interacción; seguridad y privacidad, de los datos sensibles medidos por los sensores; almacenamiento distribuido, de las observaciones durante un periodo de tiempo finito; y acceso bajo demanda, de los datos recogidos según la necesidad de la aplicación o servicio que los requiera.

Para lograr una mayor comprensión sobre el uso del SOSFul en dispositivos limitados dentro de la IoT, se presentan tres ejemplos de utilización:

- El primero de ellos son los sistemas de seguridad en casa vinculados a empresas prestadoras del servicio de vigilancia; en este caso los datos medidos por los sensores en el hogar del suscriptor del servicio solo deben ser reportados a la

empresa prestadora del servicio cuando se ha detectado una situación anómala. Antes de seguir con el ejemplo, es necesario recordar que una característica de la IoT es su capacidad de funcionar sin la intervención directa de las personas, aunque estas pueden ser uno de las entidades a las que se les hace seguimiento (human-in-the-loop). Cuando se tiene un sistema de seguridad en IoT el proceso de detección de brechas de seguridad es automático y no requiere de que se active/desactive el sistema de forma manual. Así, los sensores funcionan de forma continuada, pero solo generan una alerta en caso de detectar la presencia de una persona no autorizada dentro del recinto monitorizado. Por lo tanto, de todas las observaciones registradas en el SOSFul solo se enviaran la empresa de seguridad, mediante Internet, las que se identifican como brechas de seguridad. Esto permite que se tenga un mayor control de la privacidad de los suscriptores.

- Un segundo ejemplo del uso del SOSFul en la IoT desplegado sobre dispositivos limitados, es el de sistemas colaborativos de medición de tráfico. En estos, el SOS Ligero desplegado en un dispositivo limitado a bordo de los vehículos de particulares, pueden reportar la posición y velocidad a la cual están transitando, de forma anónima, a un servicio público para conocer el estado de las carreteras en una ciudad. En una primera instancia la posición medida por un GPS y las aceleraciones/desaceleraciones del automóvil medidas por un acelerómetro son almacenados dentro del SOSFul del dispositivo limitado; de forma periódica, una aplicación, dentro del dispositivo limitado, lee los datos y calcula la velocidad, para enviar al servicio público, mediante Internet, los tramos recorridos y las velocidades a las cuales se circuló por estos. Una segunda instancia de la aplicación, residente en la entidad prestadora del servicio, se encarga de consolidar los datos y ponerlos a disposición en forma gráfica mediante un GIS (sistema de información geográfica - geographical information system), accesible por Internet.

- Finalmente, otro excelente ejemplo de la utilización del SOSFul son los Ambient-assisted living (AAL) centrados en la ayuda a personas mayores a vivir de forma independiente [94]. Estos sistemas están encargados de proporcionar múltiples servicios enfocados en proveer bienestar y condiciones de salud a las personas mayores. Varios de estos servicios se encuentran enfocados en la seguridad física de las personas, mediante sistemas móviles de respuesta a emergencias y detección automática de caídas. En varios de estos AAL de propósito específico, los registros de observaciones se mantienen de forma local en el SOSFul desplegado en el dispositivo limitado de su casa, los cuales son analizados en tiempo real y en caso de detectar una condición anómala realizar la llamada a los servicios de emergencias.

De estos tres ejemplos se evidencian algunas ventajas del despliegue del SOSFul sobre un dispositivo limitado:

- Movilidad: tal como se presenta en el segundo ejemplo, un dispositivo limitado con conexión a Internet inalámbrica es un repositorio móvil para las observaciones recogidas por objetos móviles, como vehículos y personas.

- Integración: es la base del SOSFul, dado que abstrae la complejidad de tener una gran diversidad de nodos sensores, de formar que se habilita la interconexión a

otros sistemas mediante interfaces simplificadas y mensajes concisos.

- Adaptabilidad: el SOSFul es un servicio genérico, que gracias al uso de JSON y MongoDB lleva los formatos de mensaje a tener una máxima flexibilidad, abarcando una gran cantidad de casos de uso; usando dispositivos limitados hay que tener en cuenta las limitaciones de procesamiento y almacenamiento; haciendo que los sistemas distribuidos o/y jerárquicos sean los que se beneficien de mayor manera de estos dispositivos.

- Coste: la mayor diferencia con las otras alternativas que se evalúan en este capítulo, ya que los dispositivos limitados son los más económicos; si se enfoca en la masificación de la IoT, esta es una de las características disruptivas.

- Proximidad: como se observa en los tres ejemplos, el SOSFul sobre dispositivo limitados se encuentran en la misma red donde se generan las observaciones; esto tiene ventajas en los tiempos de transito de la información, en la capacidad de análisis local y en la seguridad/privacidad.

Por su parte, esta combinación de SOSFul como servicio y dispositivo limitado, cuenta con algunas limitaciones:

- Itinerancia: esta característica esta limita al traspaso tácito de la información mediante las interfaces del SOSFul, como se observa posteriormente, las otras opciones al ser VMs ligeras pueden ser transmitidas de forma íntegra, generando una itinerancia más directa.

- Flexibilidad: dado que los recursos computacionales de los que se disponen son limitados y en el caso del procesamiento difícilmente actualizables; el SOSFul desplegado en un dispositivo de estas características, tiene una restricción intrínseca en cuanto a la asignación de recursos utilizables.

- Rendimiento: como se sabe un dispositivo limitado se diferencia en gran manera de un mini-datacenter o un datacenter en los recursos disponibles; aunque en la mayoría de escenarios para los que se ha diseñado el SOSFul, el rendimiento en un dispositivo limitado es suficiente, es necesario aclarar que en este caso, el rendimiento está restringido a las capacidades y en comparación será inferior al de las otras alternativas revisadas.

6.2.3. SOSFul sobre FC

Utilizar el paradigma de FC dentro de la IoT es una de las ideas más interesantes sobre las cuales se está trabajando en la actualidad. Almacenar y procesar la información en *sites* cercanos a donde se origina y consume, permite mejorar en los tiempos de respuesta y distribuir la gran cantidad de información que se produce en la IoT.

Cuando el SOSFul se despliega como una instancia de FC, permite que exista un contenedor de información completamente aislado de los otros servicios que brinda el *site*. Este hecho brinda mucha flexibilidad en el uso del SOS ligero, permitiendo que pueda atender a una gran cantidad de casos de uso, incluyendo aquellos que tienen variaciones de recursos computacionales en tiempos cortos.

Esta flexibilidad de la que se dota al SOSFul cuando se despliega como instancia virtual, es ideal en la arquitectura de referencia, ya que, muchos de los componentes que constituyen las tres capas superiores, pueden compartir recursos en un *site* y utilizarlos de forma eficiente. Esto se logra al asignar los dinámicamente según las necesidades de cada componente; por ejemplo, se puede programar los análisis de datos cuando hay una menor demanda de recursos desde las aplicaciones.

Así, el SOSFul actúa como repositorio de datos y metadatos, que se encuentra distribuido a través de Internet y que gracias a su despliegue sobre *Fog*, permite una utilización eficiente de los recursos compartidos por los componentes de la IoT. Al tiempo que, abstrae a las capas superiores y a los otros componentes de la capa de procesamiento de la heterogeneidad propia de los sensores que sirven para percibir la realidad física.

Con miras a clarificar el uso del SOSFul como instancia de FC, a continuación, se describen tres ejemplos de uso y posteriormente se enumeran las ventajas y desventajadas de este despliegue.

- El primer ejemplo es un sistema de monitorización de incendios forestales. El mismo, divide las zonas forestales grandes en secciones más pequeñas para realizar las mediciones. Cada sección cuenta con un conjunto de nodos sensores; donde cada nodo sensor puede medir: gases de combustión, presencia de llama, dirección de viento, velocidad de viento, temperatura y humedad. Los nodos sensores están distribuidos con una densidad de un nodo sensor por hectárea en cada una de las secciones. Además, cada sección cuenta con un mini-*datacenter* donde está desplegada la instancia del SOSFul que registra todos los datos que los nodos sensores reportan mediante LoRaWAN y cuenta con una conexión a Internet a través de redes móviles celulares. En los mini-*datacenter* residen además una instancia virtual, encargada de analizar los datos y determinar si está ocurriendo un incendio; en caso afirmativo se realizan dos procesos: el primero, envía, mediante Internet, alertas a los organismos de gestión de emergencias y avisos a los encargados de las secciones forestales aledañas (por si estos deben tomar acciones preventivas); el segundo, aumenta los recursos computacionales al SOSFul, aumenta la frecuencia de muestreo en los nodos circundantes del fuego detectado, y además, inicia una instancia virtual adicional encargada de determinar la dirección y velocidad de propagación del incendio.

- Un segundo ejemplo del uso del SOSFul como instancia de FC, viene de la mano del posicionamiento *indoor* en los almacenes de las grandes superficies. En estos sistemas [95][96], diversos nodos sensores detectan el dispositivo de usuario mediante BLE (Bluetooth Low Energy) cuando este se encuentra cerca de ellos y registran en el SOS el identificador del dispositivo, la intensidad de la señal recibida y el tiempo de ida y vuelta del mensaje. Además de la instancia virtual del SOSFul, en el *Fog site* de cada almacén, existen una instancia virtual que ejecuta redes neuronales para determinar la posición cada segundo de cada uno de los dispositivos de usuario. De igual forma, una tercera instancia virtual despliega el CRM (Customer relationship management) de la compañía. Finalmente una instancia, que se funciona solo en la noche, se encarga de realizar el proceso de BI (Business Intelligence), tomando la información del

CRM y la posición de los dispositivos de usuario a lo largo del día; mientras tanto, la instancia virtual del SOSFul permanece apagada.

- Finalmente, dentro del área de los *Smart Building* se encuentran los sistemas de monitorización de la salud estructural (SHM) en edificios, los cuales se constituyen en un excelente ejemplo del uso del SOSFul como instancia de FC. En este caso, un compendio de construcciones, como las pertenecientes a un campus universitario, registran los datos de los sensores (inclinómetros, fisurómetros, piezómetros, extensómetros, células de presión, extensímetros, etc.) distribuidos en cada construcción dentro de la instancia virtual de SOS. Para este ejemplo, cada una de las construcciones del campus tiene su propia instancia de SOSFul; mediante una instancia independiente encargada de detectar anomalías en las mediciones, se determina si se deben asignar más recursos computacionales a una determinada instancia de SOS de forma que se obtenga un mayor detalle en las mediciones realizadas. La información de todos los SOSFul puede ser accedida desde Internet por la empresa encargada de la vigilancia y seguridad estructural de las construcciones.

Mediante estos ejemplos, se pueden observar varias de las ventajas que ofrecen las instancias *Fog* de SOSFul:

- Integración: el SOSFul desplegado como instancia de *Fog* facilita el funcionamiento conjunto con otras instancias SOSFul y con instancias que brinden otros servicios o aplicaciones.

- Adaptabilidad: debido al dinamismo en la asignación de recursos, se abre una gama amplia de casos de uso, en los cuales bajo ciertas condiciones el SOSFul tiene una mayor disponibilidad de almacenamiento o procesamiento.

- Itinerancia: toda instancia virtual puede ser transmitida fácilmente desde un *site* a otro, con este se obtiene una mayor sencillez en las migraciones, los respaldos y los sistemas nómadas.

- Flexibilidad: aunque se encuentra limitada por los recursos del *Fog site*, las instancias del SOSFul puede hacer uso de recursos computacionales según la necesidad, permitiendo ofrecer servicio diferencial según el caso de uso al que atiende.

- Rendimiento: en comparación al SOSFul desplegado en dispositivos limitados, las instancias virtuales de SOSFul pueden acceder a una mayor cantidad de recursos computacionales, sean de forma temporal o definitiva.

- Proximidad: gracias a que el *Fog site* se encuentra próximo a la red donde se generar y consumen los datos de los sensores, se puede disfrutar de un gran desempeño en términos de retardo.

Algunas desventajas que tiene el SOSFul como instancia de FC, en comparación con las otras analizadas en esta sección, debido a las características inherentes a este modelo de computación, son:

- Movilidad: los *Fog sites* puede ser móviles; sin embargo, al comparar la facilidad de los dispositivos limitados para afrontar esta situación, se puede considerar que esta implementación se encuentra en mayores requerimientos para poder lograrlo.

- Coste: la mayor diferencia que tienen las soluciones basadas en instancias virtuales es que requieren de mayores costes de implantación (debido a su mayor cantidad de recursos computacionales disponibles).

- Rendimiento: surge tanto en las ventajas como en las deficiencias debido a que se encuentra en un punto intermedio entre los dispositivos limitados y las instancias *Cloud*, generalmente asociadas con grandes *datacenters.*

- Proximidad: como en el caso anterior esta es una cualidad que se encuentra en el intermedio de los dispositivos limitados y los *datacenters*, siendo lo más habitual que las instancias de FC se ejecuten en nano, micro o mini *datacenters.*

6.2.4. SOSFul sobre CC

A diferencia de la solución anterior, las instancias virtuales de CC suelen utilizarse sobre grandes *datacenters*. Esto les permite disponer de capacidades de almacenamiento y procesamiento virtualmente ilimitadas. Sin embargo, los grandes costes de esta implementación conllevan a que sus recursos sean compartidos y se encuentren distantes del lugar donde se genera la información de los sensores, con la consiguiente degradación del retardo.

Como en los casos anteriores, el SOSFul cumple una funcionalidad de repositorio y de componente de interoperabilidad, bien sea, distribuido o centralizado; donde se almacenan los datos de los sensores fraccionados según la utilización que se le vaya a dar (por localización geográfica, por categoría, por contexto, por cliente, etc.) o como un bloque de datos conjunto de una gran cantidad de sensores.

Esta implementación aventaja a las dos anteriores, en sus capacidades de almacenamiento y procesamiento; pudiendo atender casos de uso con una gran cantidad de sensores o aquellos en los cuales se realizan mediciones con un alta tasa de muestreo. Dentro de la IoT es de especial interés cuando se quieren realizar análisis consolidados de gran cantidad de observaciones.

A continuación, se presentan tres ejemplos que permiten apreciar el uso del SOSFul desplegado como instancia de CC:

- El primero de ellos, se enmarca en las *SmartCities* y sus servicios de seguridad, concretamente el subsistema de localización de eventos violentos. En este caso, una gran cantidad de sensores de sonido se encuentran distribuidos por toda la ciudad, la información de un posible sonido de disparo de arma de fuego se envía a una instancia *Cloud* del SOSFul, donde el registro se mantiene por un periodo de 1 hora. En tiempo real una instancia diferente que contiene una aplicación de análisis, revisa la información del SOS para determinar si se ha presentado un disparo y realizar la localización mediante triangulación. En caso

de detectar un disparo, las observaciones que la registraron pasan a una segunda instancia del SOSFul, donde permanecerán hasta por 100 años y que sirve como prueba para los procesos judiciales, y además, se genera una alarma a los servicios policiales.

- Un segundo ejemplo lo constituye el sistema de análisis de calidad del agua en ríos que tienen las entidades de gestión ecológica de la ciudad. En estos sistemas, diversos nodos que analizan la calidad del agua se distribuyen a lo largo del rio, desde que entra a la ciudad y hasta que sale de esta. Toda la información recolectada desde los nodos se envía por Internet a una instancia del SOSFul, cada rio cuenta con su propia instancia, la cual almacena de forma permanente todas observaciones. Este tipo de aplicaciones es intensiva en el uso de almacenamiento ya que los datos deben almacenarse por décadas para analizar tendencia en el largo plazo.

- Un último ejemplo del despliegue del SOSFul como instancia de CC, surge de las nuevas legislaciones que están siendo adoptadas por varias ciudades a lo largo del mundo, con el fin de restringir la circulación de vehículos en determinadas zonas geográficas o periodos de tiempo. En este caso, las autoridades de transito aprovechan los sensores de identificación de placas para registrar: la zona, el momento y el número de la placa, en el SOSFul. Una aplicación analiza posteriormente la información almacenada y determina si se ha incurrido en una falta de tránsito, y en caso de que se detecte, envía la información para que se realice el proceso de penalización. Los datos que no presentan una falta de transito se eliminan del SOS, para liberar espacio de almacenamiento.

Como se puede entrever, este tipo de implementación brinda grandes ventajas, como son:

- Integración: como se observa en los ejemplos, las instancias *Cloud* interactúan de forma natural, al tiempo que mantiene un aislamiento que asegura el uso adecuado de recursos, la tolerancia del sistema a errores y la seguridad.

- Adaptabilidad: es la configuración que permite un mayor rango de casos de uso, desde pequeños sistemas de IoT hasta el nivel de las *SmartCities*.

- Itinerancia: dado que esta implementación se encuentra fundamentada en las instancias virtuales, el respaldo y migración de las mismas se encuentra asegurado. Esta funcionalidad resalta en los proveedores de *Cloud* distribuidos, donde se pueden tener mecanismos que acerquen los *Cloud site* a los lugares donde se generan y consumen los datos almacenados en el SOSFul.

- Flexibilidad: brindando recursos virtualmente ilimitados, el SOSFul puede almacenar ingentes cantidades de observaciones que puede ser útil para el análisis temporal, la inteligencia de negocios, la minería de datos y el *BigData.*

- Rendimiento: la utilización de una arquitectura basada en CC, proporciona recursos y capacidad de procesamiento bajo demanda, que puede incluir gran cantidad de procesadores y memoria RAM de una forma elástica.

Como en las anteriores implementaciones, se pueden puntualizar algunas desventajas de la solución, debidas a las características que aportan las arquitecturas de *Cloud Computing*:

- Movilidad: los *Cloud sites* generalmente suelen estar vinculados a grandes *datacenters* que no están diseñados para ser móviles, aunque existen algunas excepciones.

- Coste: de las tres implementaciones que se han revisado en este capítulo, es la que mayor coste conlleva.

- Proximidad: en general, las instancias *Cloud* correrán en sitios alejados de donde se genera y consume la información, esto hace que sea la aproximación con mayor tiempo de retardo de transmisión.

6.3. Casos de uso: SmartCity

6.3.1. Introducción

Las ciudades se han afianzado como centros de influencia social y económica; en los cuales, actualmente, viven más del 50% de la población mundial y se espera que para el 2050 pase a un 70%. De la gran cantidad de personas que habitan en las ciudades y la economía que se genera en las transacciones entre estas personas, surge la necesidad de contar con una gestión sostenible y eficiente.

Como respuesta a esta necesidad las TIC plantean las *SmartCities*. Estas se definen como: "el uso de las Tecnologías de la información y de la comunicación para monitorizar, analizar e integrar la información clave de los servicios que ejecutan en las ciudades" [97] con el objetivo de proveer una infraestructura que garantice un incremento en la calidad de vida de los ciudadanos.

Con base en esta definición, las *SmartCities* se fundamentan en la cooperación y en el intercambio de información. Para brindar estos fundamentos, es necesario utilizar tecnologías y estándares que posibiliten obtener, transmitir, procesar, almacenar y compartir, grandes cantidades de datos en tiempo real, para prestar servicios a la ciudadanía.

El análisis de una *SmartCity* se aborda considerando seis características [98]; las cuales identifican áreas de interés en las cuales una ciudad puede integrar las TIC, y con su ayuda mejorar los servicios al ciudadano. Cada una de las características incluye algunos rasgos específicos (Figura 100); estos aportan una mejor comprensión sobre el desarrollo de las características en la ciudad y su nivel de madurez en cada una de ellas. Una *SmartCity* no tiene por qué abordar todas las características de forma simultanea o alcanzar el máximo nivel de madures en todas ellas; basta con que la ciudad aborde las características de mayor impacto social y económico, y en ellas construya servicios adaptados a sus necesidades, para ser considerada como una *SmartCity*.

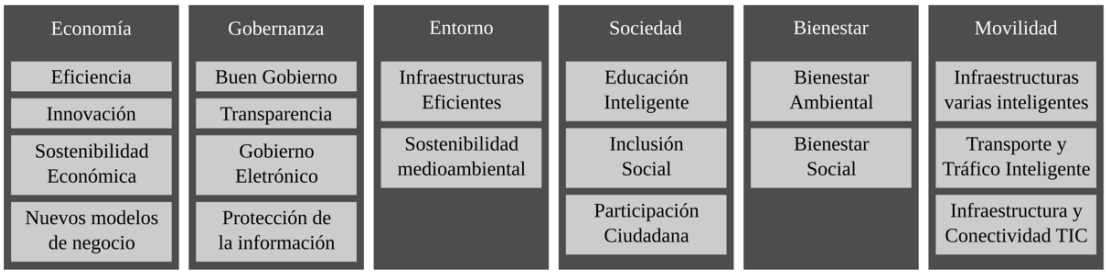


Figura 100 - Características de SmartCity

Aunque no todas las características deben desarrolladas en una SmartCity, todas ellas comparten una propiedad: están basadas en las TIC. En especial, todas las *SmartCities* cuentan con una o varias SN que les sirven como mecanismo de percepción de la realidad física de la ciudad. Así, los sensores y las redes que conforman, están intrínsecamente relacionadas con toda aproximación de *SmartCity*. Sin embargo, con la IoT no es así; dado que el utilizar Internet es opcional y se puede optar en su lugar por redes dedicadas exclusivamente para dar servicio a la ciudad.

No obstante, el uso de IoT es deseable en muchos casos, dado que aprovecha la

infraestructura y servicios presentes en la ciudad, los cuales suelen ser provistos por operadores de telecomunicaciones independientes. De esta forma, se emplea una red de gran calidad y con buen nivel de servicio; al tiempo, que se emplean los recursos públicos de forma eficiente. Bajo estas consideraciones, esta sección se centra en las *SmartCities* que utilizan IoT como mecanismo de comunicaciones para el funcionamiento de sus servicios.

Para lograr una mayor comprensión de las *SmartCities* que emplean IoT, se ha detallado la arquitectura de referencia de IoT (Figura 101). De forma que, se pueden observar algunos componentes típicos que componen los sistemas de estas ciudades; incluyendo: los nodos sensores de la capa de percepción, la Internet en la capa de transporte, algunos servicios de la capa de procesamiento, algunas aplicaciones de la capa de aplicación y algunas aplicaciones de gestión en la capa de negocio.

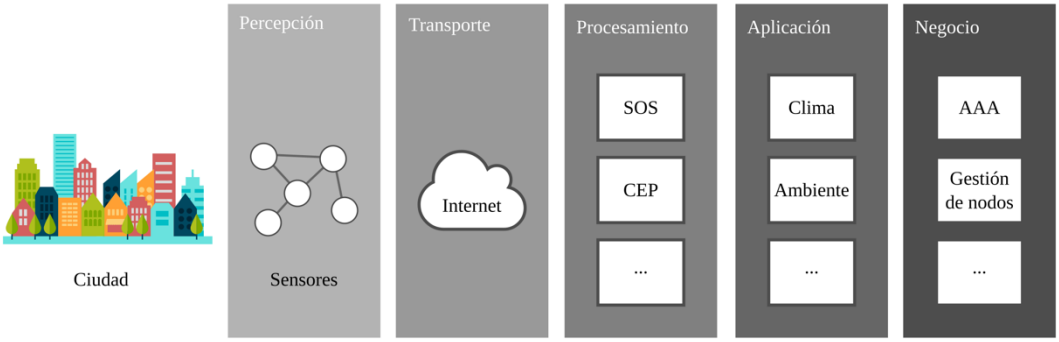


Figura 101 - Características de IoT aplicado a SmartCities

Como se observa en la figura anterior, el SOS forma parte de la capa de procesamiento, siendo el repositorio de la información obtenida mediante la capa de percepción. Como servicio, el SOS permite a una *SmartCity* poder mantener un historio de las observaciones medidas por los nodos sensores diseminados por toda la ciudad; estos datos pueden ser aprovechados para determinar variables: ambientales, sociales, económicas, logísticas, etc.; las cuales son fundamentales para tomar decisiones estratégicas como: políticas ambientales, planeación del desarrollo urbanístico, gestión efectiva de los transportes públicos, creación de zonas comerciales, etc.

Para explotar el uso del SOSFul como servicio de la capa de procesamiento para las *SmartCities*, se ha planteado un caso de uso centrado en la integración de dos SN. Estas redes están disponibles en las ciudades, pero que se encuentran desagregadas y sus datos son poco aprovechables en tiempo real. El caso de uso abarca la implementación de todas las capas de la arquitectura de referencia de IoT, sin embargo, se hará hincapié en el SOSFul como mecanismo de interoperabilidad a nivel semántico.

6.3.2. Implementación

El caso de uso (Figura 102) representan la integración de cuatro SN: dos dedicadas al registro de la contaminación y dos especializadas en la monitorización medioambiental. Los nodos sensores emplean tecnologías de comunicación heterogéneas, incluyendo: Ethernet, WiFi, Bluetooth y ZigBee; además, los sensores provienen de diversos fabricantes, mientras que el tipo y formato de las observaciones que realizan son de diversa índole. Por su parte, los datos y metadatos de los sensores y observaciones se

almacenan en el SOSFul y se hacen disponibles mediante un *Context Broker*. Finalmente, se despliega una aplicación donde se puede visualizar las observaciones registradas por las cuatro SN; cada observación se encuentra georeferenciadas, lo que permite su representación sobre el mapa de la ciudad.

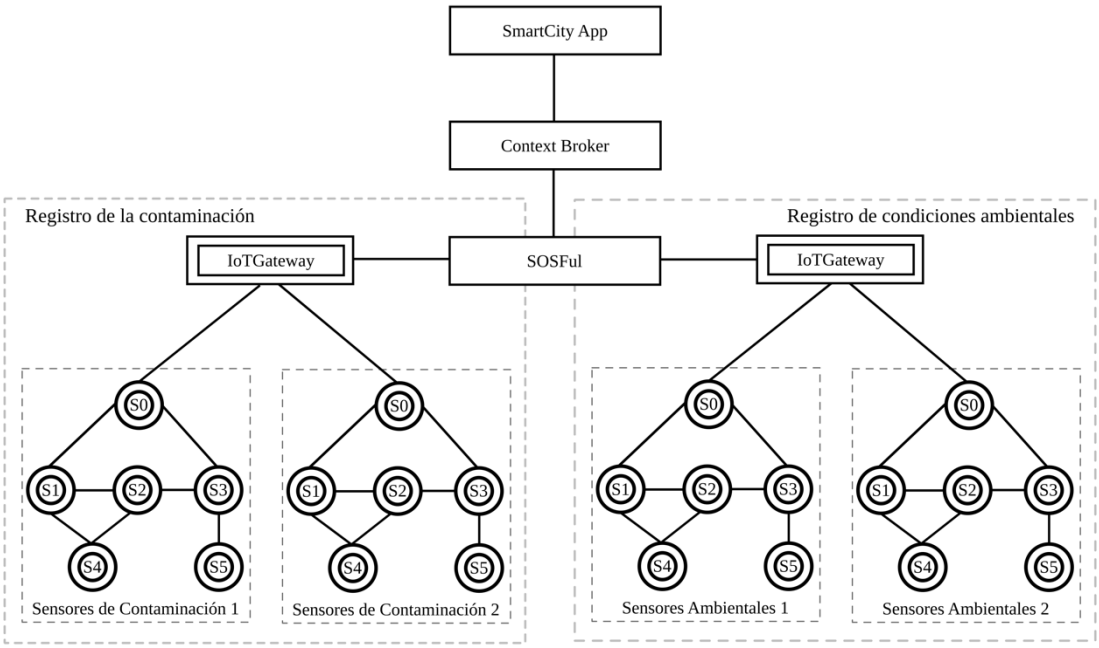


Figura 102 - Caso de uso de SmartCity

Para profundizar en la descripción del caso de uso, es necesario identificar los dispositivos que han utilizado y como el SOSFul ha sido desplegado; además, se debe reconocer la heterogeneidad inherente a las redes y protocolos de comunicación. Así, en primer lugar se profundiza en el diagrama de despliegue del caso de uso, en segundo lugar se presentan las tecnologías de red empleadas y finalmente, se identifican los flujos de información.

En el siguiente diagrama (Figura 103), se puede detallar cada uno de los componentes del caso de uso y los dispositivos en los cuales se han desplegado. En primera instancia se aprecian los nodos sensores, los cuales están compuestos por los sensores y una placa de desarrollo; seguidamente se ve el IoTGateway, el cual funciona sobre un ordenador de placa reducida (Single Board Computer - SBC); de ahí, para el SOSFul se emplea un mini-*datacenter* que brinda un *Fog site* privado; y finalmente, el *Context Broker* y la *SmartCity App* que se han desplegado sobre un sistema de *Cloud* público.

Figura 103 - Componentes del caso de uso

Por su parte, la identificación de las tecnologías de comunicación empleadas, se puede observar en el siguiente diagrama (Figura 104). Aquí, se identifica que el enlace entre

los nodos sensores y el IoTGateway se realiza mediante: Ethernet, WiFi, Bluetooth o ZigBee; mientras tanto, el enlace entre el IoTGateway y el SOSFul se hace utilizando Ethernet; análogamente, el enlace entre el SOSFul y el ContextBroker, y el enlace entre el ContextBroker y la *SmartCity App* se hacen usando Internet.

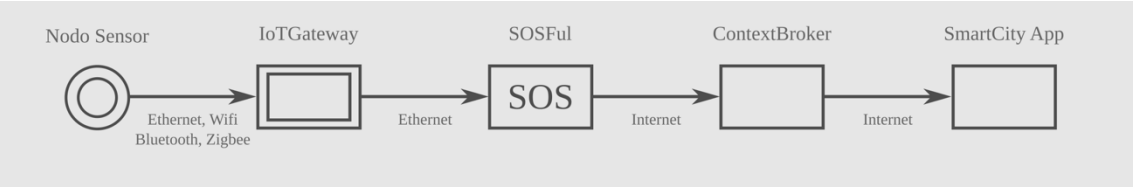


Figura 104 - Tecnologías de comunicación del caso de uso

Mientras tanto, el diagrama presentado a continuación (Figura 105), determina los protocolos utilizados en cada enlace y el formato de los datos transmitidos. Así, el enlace entre el nodo sensor y el IoTGateway emplea el protocolo MQTT sobre TCP y envía las observaciones en formato JSON; mientras que, el enlace entre el IoTGateway y el SOSFul, emplean el protocolo SOS (versión ligera modificada REST), sobre CoAP/UDP, con los mensajes basados en el modelo O&M ligero, adaptados a JSON (formato propuesto por el SOSFul); por su parte, el enlace entre el SOSFul y el ContextBroker se realiza utilizando HTTP/TCP y empleando mensajes JSON con el modelo de datos de ContextEntity (del estándar NGSI9/ NGSI10); y finalmente, el enlace entre el ContextBroker y la SmartCity App emplea HTTP/TCP con formato de mensajes JSON.

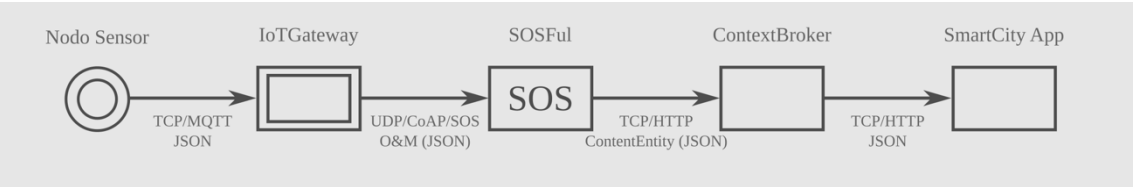


Figura 105 - Protocolos del caso de uso

Los elementos empleados para realizar el prototipo en el laboratorio; se adecuan a los diagramas presentados. En primer lugar, se detallan los componentes que conforman los nodos sensores empleados; en segundo lugar, se da paso a los dispositivos de red locales; en tercer lugar, se describe el *Fog site* donde se desplego el SOSFul; y finalmente, se detalla el servicio de *Cloud* empleado.

Los nodos sensores están compuestos por: los sensores, las placas de desarrollo y los módulos de comunicación. Dependiendo la red a la que corresponden y la tecnología de comunicación que van a implementar, se seleccionan los componentes (Tabla LXXIV) para conformarlos (Figura 106). Por su parte, los nodos que despliegan el IoTGateway (Figura 107) están compuestos por la un ordenador de placa reducida y los módulos para conectarse a través de todas las tecnologías de comunicación del caso de uso, estos componentes también aparecen en la

Tabla LXXIV.

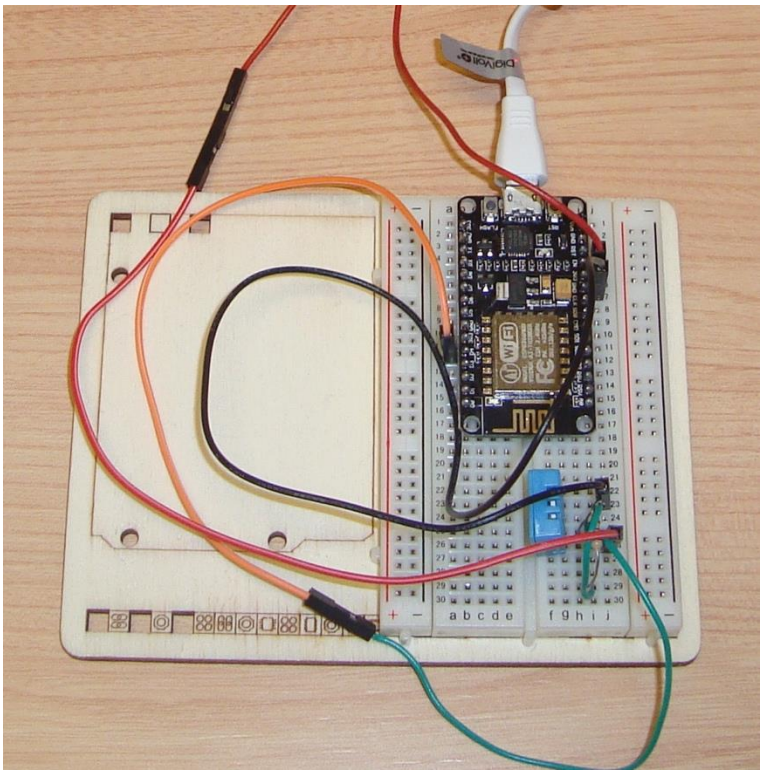
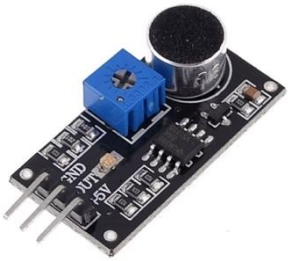
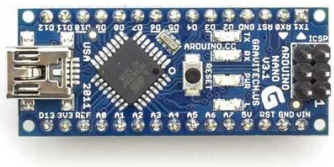


Figura 106 - Ejemplo de nodo sensor del caso de uso

Figura 107 - Nodo de IoTGateway del caso de uso

Tabla LXXIV - Componentes para el caso de uso

SENSORES	
	Sensor de temperatura y humedad Hace parte de los nodos sensores de la red de medición de condiciones ambientales y envía los datos en formato digital a la placa de desarrollo a la cual está conectado.
	Sensor de gotas de lluvia Como el sensor anterior, hace parte de la red de condiciones ambiental, se encarga de detectar si ha comenzado una lluvia y envía una señal analógica a la placa de desarrollo a la que esté conectado.
	Detector de sonido con comparador Este sensor se encarga de enviar una señal digital a la placa de desarrollo a la que se encuentre conectada en el momento en que el ruido del entorno supere un umbral determinado. Este sensor es empleado para determinar la contaminación acústica y por ende hace parte de la red de medición de contaminación.
	Sensor de polución Se encarga de la medición de hidrógeno, alcohol y monóxido de carbono del ambiente. Hace parte de la red de detección de contaminación con una sensibilidad desde 10 ppm para el CO.
	Sensor de calidad del aire Detecta gran cantidad de gases contaminantes como: amoniaco, sulfuros, bencenos, metano, propano y butano. Envía una señal digital a la placa desarrollo a la cual está conectada y hace parte de la red de detección de contaminación.
PLACAS DE DESARROLLO	
	NodeMCU LUA V3 Placa de desarrollo empleada para conformar los nodos sensores. Es compatible con la plataforma Arduino y cuenta con un puerto USB, lo que facilita su programación. Como ventaja, esta placa viene con WiFi de fábrica; de forma que, los nodos que utilizan esta tecnología de comunicación se construyen sobre esta placa.

	Arduino Nano Placa de desarrollo Arduino con puerto USB para fácil programación. Dentro del montaje es empleada en los nodos sensores que utilizan tecnologías de comunicación diferentes al WiFi.
ORDENADOR DE PLACA REDUCIDA	
	Raspberry Pi 2 modelo B Se utiliza para desplegar el IoTGateway, siendo un ordenador en formato reducido que corre el sistema operativo GNU/Linux. Cuenta con conexión Ethernet de fábrica y mediante módulos de comunicación dispone de WiFi, Bluetooth y Zigbee.
MÓDULOS DE COMUNICACIÓN	
	Módulo de Bluetooth para Arduino Soporta Bluetooth 4.0 y BLE para la interconexión de los nodos sensores. Es empleada por los nodos que se quieren conectar con el IoTGateway a través de esta tecnología. Está presente tanto en la red de contaminación como en la de condiciones ambientales.
	Módulo de Ethernet para Arduino Habilita a los nodos sensores de las dos redes, la posibilidad de conectarse mediante Ethernet al IoTGateway. Provee una conexión 10BASE-T.
	Módulo de ZigBee para Arduino y Raspberry Pi Módulo que provee a los nodos sensores basados en Arduino la posibilidad de conectarse utilizando la tecnología de ZigBee. Mediante adaptador, también se emplea este módulo para la Raspberry Pi donde se despliega el IoTGateway.
	Módulo de Bluetooth para Raspberry Pi Módulo de interconexión Bluetooth que se conecta mediante puerto USB a la Raspberry Pi. Facilita esta tecnología de comunicación al IoTGateway desplegado.
	Módulo de WiFi para Raspberry Pi Mediante USB este módulo provee la posibilidad de conectarse mediante WiFi a la Raspberry Pi; permitiendo así, que el IoTGateway soporte esta tecnología de comunicación.

Una vez la información llega al IoTGateway, es trasmitida mediante redes cableadas Ethernet hasta el SOSFul, de igual forma para enviar la información a la nube se utiliza el acceso a Internet. Para estas comunicaciones se emplean varios dispositivos de conectividad de red local. Los más destacados se presentan a continuación (Tabla LXXV):

Tabla LXXV - Dispositivos de conectividad para el caso de uso

DISPOSITIVOS DE CONECTIVIDAD	
	Access Point Permite la interconexión de los dispositivos mediante WiFi, es empleado por los nodos sensores que desean conectarse al IoTGateway por medio de esta tecnología.
	Switch Permite la conexión mediante la red cableada, es utilizado por los IoTGateway para enviar la información al *Fog site* local.
	Router Sirve como interconexión de la red local con Internet, posibilitando la comunicación entre el *Fog site* y el servicio *Cloud*, es decir, entre el SOSFul y el ContextBroker.

Por su parte, se ha hecho uso de un *Fog site* privado que brinda la capacidad de desplegar VMs ligeras sobre un sistema operativo GNU/Linux. En este caso se ha empaquetado el SOSFul como dos instancias de Docker, una para la base de datos MongoDB y otra con el servidor de aplicaciones Node.js con el SOSFul en el. El site no ha sido manipulado de forma directa, por el contrario, se ha brindado un acceso remoto que permite utilizar el servicio bajo demanda, bajo unas restricciones de uso de los recursos computacionales; contando con hasta 2 GB de memoria RAM, 1 TB de disco duro y el 50% de uno de los procesadores reales de la maquina donde reside la instancia virtual.

Tabla LXXVI - Fog site para el caso de uso

FOG SITE	
	*Fog Site* Cuenta con dispositivos de conectividad y un grupo de servidores de capacidad baja y media; sobre los cuales se brinda los recursos computacionales en un esquema de servicio y que permite el despliegue de VMs basadas en la tecnología Docker.

Finalmente, tanto el ContextBroker como la SmartCity App están desplegadas como instancias virtuales basadas en Docker dentro de *Amazon Web Services*, específicamente dentro del *Amazon Elastic Compute Cloud* (Amazon EC2). Así las dos aplicaciones están desplegadas en instancias t2.micro.

Tabla LXXVII - Servicio de Cloud para el caso de uso

CLOUD SERVICE	
amazon web services	*Cloud service* Para ContextBroker y la SmartCity App se ha optado por el despliegue sobre un servicio público de *Cloud*, el Amazon EC2. Cada una de las instancias cuenta con recursos limitados asignados pero puede extenderlos según sus necesidades. Debo a que las pruebas son controladas, se ha estimado que las capacidades de una instancia t2.micro es suficiente para dar servicio efectivo.

A continuación, se puede observar una captura de pantalla de la aplicación en funcionamiento (Figura 108). En ella se puede ver como se representan los distintos nodos sensores geoposicionados; además muestra las últimas observaciones registradas por uno de los nodos sensores al estar seleccionado.

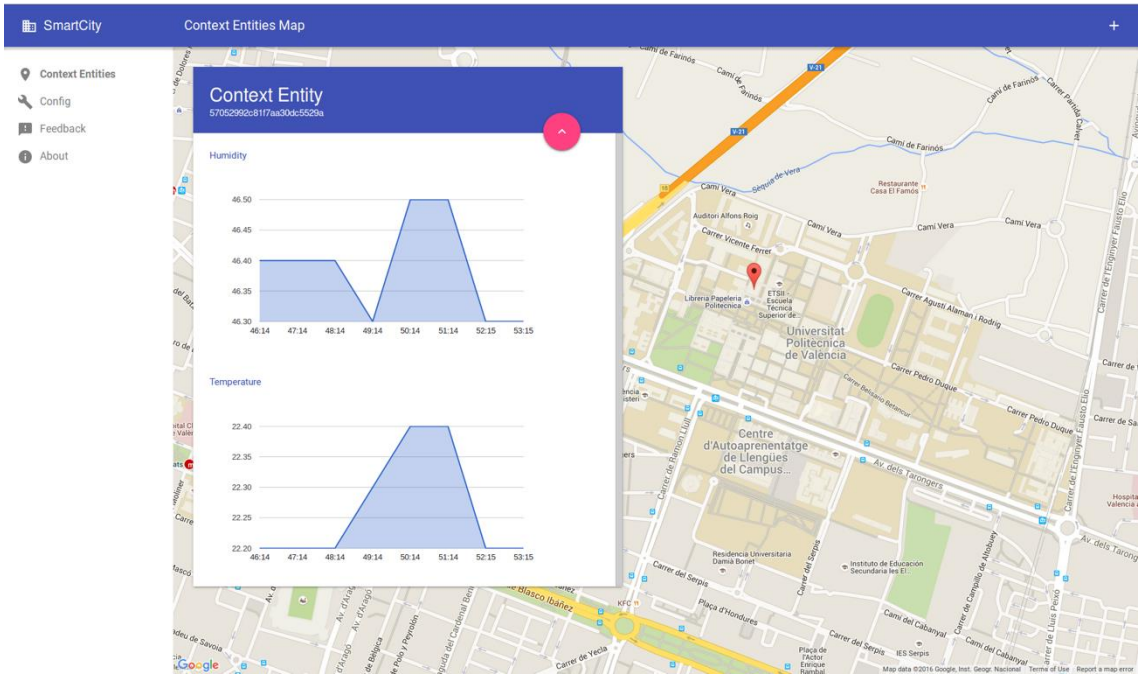


Figura 108 - Captura de pantalla de la aplicación

6.4. Conclusiones

El SOSFul se presenta como una alternativa para la interoperabilidad semántica en la IoT. Centrado su funcionamiento en los datos de sensores, los cuales conforman la base de toda IoT; convirtiéndose así, en un punto de partida interesante para abordar otras entidades que conforman esta nueva red de redes.

Debido a sus interfaces y mensajes, el SOSFul se integra de forma efectiva a los desarrollos actuales que conforman la IoT. Como se observa en el caso de uso, aunque no se profundizo en los componentes empleados fuera del SOS, la integración con tecnologías ya existentes es posible; por ejemplo, el uso del protocolo CoAP, el uso de la tecnología ZigBee o la integración con un ContextBroker.

A su vez, el SOSFul puede integrarse dentro de una arquitectura de referencia de IoT. Esto facilita el poder aplicarlo a un gran número de casos de uso; así cuando se desea analizar las plataformas que se ajustan a un caso determinado, se puede empezar a revisar la arquitectura de referencia, los componentes y como estos se integran; siendo el SOSFul, el componente que provee de interoperabilidad semántica, que habilita la abstracción de la heterogeneidad de los niveles inferiores.

Por su parte, como se detalla en el capítulo, las SmartCities son cada día más relevantes en el ámbito de aplicación de las TIC. Éstas pueden ejecutarse utilizando IoT, aprovechando así, las infraestructuras presentes en gran parte de estas ciudades. Al utilizar IoT y dado la relevancia de los sensores dentro de una SmartCity, el uso del SOSFul facilita el contar con un repositorio de datos y metadatos de las observaciones; dando la oportunidad de construir rápidamente una plataforma que supla las necesidades técnicas, sin necesidad de desarrollar una solución que enfrente la heterogeneidad de los sensores.

Finalmente, el caso de uso brinda un claro ejemplo de la utilidad del SOSFul al emplearlo dentro de la IoT. Como componente de interoperabilidad, permite que las aplicaciones pueda utilizar la información de las observaciones, sin conocer las tecnologías empleadas por los nodos sensores.

7. Conclusiones y líneas de trabajo futuras

7.1. Conclusiones finales

En esta tesis se ha presentado, estudiado, implementado y evaluado un mecanismo de interoperabilidad basado en los estándares SWE del OGC, que puedan ser aplicadas en arquitecturas distribuidas a gran escala y que se integren con los paradigmas de FC/CC. Las conclusiones generales son las siguientes:

7.1.1. Estado del Arte

- Los CPS son uno de los campos de investigación y desarrollo que se proyectan con mayor potencial de crecimiento, ya que como disciplina permite una visión integradora de gran cantidad de sistemas existentes y la proyección de los conocimientos adquiridos hacia sistemas en desarrollo. De igual forma, los CPS están captando gran cantidad de recursos para su desarrollo; por ejemplo, la agricultura de precisión está teniendo una aceptación amplia a lo largo del mundo, lo que ha conllevado una inversión en compra y actualización de maquinaria por parte de los productores agrícolas; de igual forma, la IoT es una de las tecnologías en las cuales gobiernos y empresas están realizando mayores inversiones, bastan con revisar los proyectos que está llevando a cabo por parte de Google, Amazon o Qualcomm para reconocer el interés generalizado que se ha producido en torno a este tema

- Como en el caso de los CPS, las apropiaciones de CC y CF se están presentando de forma asidua a nivel global; de la mano de Cisco, Google, Amazon y Microsoft son cada vez más las empresas y estados que emplean alguna de estas tecnologías. Pero es la integración entre CPS y *Cloud/Fog* lo que genera un mayor interés a nivel investigativo, debido a que el crecimiento esperado de los CPS solo puede ser atendido mediante la integración con CC y FC. En este campo, los esquemas distribuidos de *Cloud* y los sistemas jerárquicos con *Cloud* y *Fog* representan los frentes de mayor interés; y es por eso motivo que se han introducido como parte de esta tesis.

- Pero la integración CPS con CC y FC se quedaría solo en proyecciones sobre su potencial si no se encuentran mecanismos apropiados para la interoperabilidad. Y es que los CPS son sistemas agrupadores de diversos dispositivos, fabricantes, protocolos y desarrollos tecnológicos, lo que los convierte en candidatos perfectos para que se presenten incompatibilidades a todos los niveles. Bajo este panorama, han comenzado a surgir modelos, arquitecturas y plataformas para la interoperabilidad; siendo la propuesta de esta tesis un mecanismo que aborda esta problemática, con el objetivo claro de extenderse, desde las SN hacia la IoT y luego hacia otros CPS, para dar respuesta a la interoperabilidad semántica de forma transversal en los diferentes CPS.

- Así, sé a buscado una marco de referencia ampliamente usado dentro de uno de los CPS con mayor relevancia y desarrollo, como lo son las SN, para comenzar a construir una solución integradora para la interoperabilidad. De esta búsqueda se seleccionó el SWE debido a su amplia aceptación dentro de las implementaciones de SN, la gran cantidad de empresas que se encuentran asociadas a el organismo estandarizados, OGC; y la flexibilidad que se

encuentra inmersa en todos los estándares y recomendaciones asociados a este marco de trabajo.

- En especial, el SOS, se alza como una solución de interoperabilidad semántica dentro de las SN. Siendo ampliamente utilizado en despliegues actuales, se pueden encontrar puntos de mejora y extensión para abarcar un mayor número de CPS. El SOS y su asociación con los modelos de datos SensorML y O&M, son una línea base para afrontar la interoperabilidad en las SN actuales. Basados en esa primera aproximación, se ha detectado que se necesita poder desplegar este servicio en dispositivos limitados en capacidades computacionales y de conectividad; para esto se ha determinado que una simplificación de las operaciones y los modelos puede contribuir a suplir la necesidad detectada. A partir de estas premisas, se construye el mecanismo en el que se profundiza en la tesis y que se ha denominado SOSLite.

- Sin embargo, un mecanismo de interoperabilidad basado en el estándar SOS, no se adapta a todas las aplicaciones; y cuando se quiere dar respuesta a la necesidad de reducir al máximo el ancho de banda empleado para el intercambio de información en CPS con interoperabilidad semántica, se hace necesario realizar unas simplificaciones de las interfaces de comunicación. Con este fin se desarrolla el SOSFul, el cual no solo atiende a esta necesidad manifiesta, sino que, se adapta a los nuevos protocolos empleados en los CPS actuales.

- Dado que el estándar SOS fue lanzado hace varios años y que las tecnologías de desarrollo de arquitecturas, servicios y aplicaciones han seguido avanzando, tanto el SOSLite como el SOSFul buscan abordar las necesidades a las cuales atienden, mientras emplean tecnologías apropiadas que se han masificado en los últimos dos años. Por ejemplo, el uso de una base de datos NoSQL atienten al hecho de que la mayoría de los datos que circulan por las SN y el IoT son de naturaleza simple, inmutable (no cambian con el tiempo) y no relacional (el concepto que modela los datos puede ser entendido como una entidad auto-contenida); de igual forma, el uso de servidores *Event driven* se adecua al comportamiento de productores y consumidores de información en las SN y la IoT, siendo este en su mayoría orientado a eventos; sin dejar de lado, la atención a los que siguen una orientación procedimental.

- Para determinar la potencialidad de la propuesta de esta tesis, se seleccionaron dos temáticas de amplia importancia a nivel global: la logística de pasajeros y las ciudades inteligentes. Reconociendo que la movilidad eficiente es uno de los frentes en los cuales las sociedades modernas están enfocadas, con miras a: mejorar el servicio, disminuir la contaminación asociada a la actividad y reducir los costos operativos; se centró la aplicación al transporte terrestre intermunicipal de pasajeros y sobre este se plantearon casos de uso a las tecnologías desarrolladas. De igual forma, las ciudades inteligentes son el foco de grandes inversiones gubernamentales y empresariales, vaticinando que para el año 2050 gran parte de la población vivirá en las ciudades y que son éstas las que deben dar respuesta para brindar servicios públicos de forma efectiva; se desarrolló un segundo caso de uso donde se aprovecharan los beneficios del mecanismo de interoperabilidad desarrollado.

7.1.2. SOSLite

- El SOSLite es una implementación de SOS ligero, creada a partir de los estándares SOS, SensorML y O&M; y de las recomendación de perfil ligero para sensores estacionarios realizada por la OGC. Cuyo objetivo es brindar un mecanismo de interoperabilidad semántica en las SN, que pueda ser desplegado sobre dispositivos limitados. Sus requerimientos se basan en el análisis realizado por OGC, y que da lugar a la recomendación para el perfil ligero, y en el reconocimiento de las necesidades de las SN que ha realizado el grupo de investigación a lo largo de los años. Así, se parte de una propuesta teórica de SOS ligero, en la cual se han determinado las funcionalidades necesarias y la forma de dar respuesta a esta; para dar paso a una implementación compatible con el estándar pero con un consumo de recursos computacionales reducido.

- De igual forma, el SOSLite atiende a las operaciones de los perfiles *core* y *transactional* del estándar SOS, simplificándolas para hacerlas más ligeras. Para lograr esta simplificación, se toma como referencia los cambios propuestos por la OGC para el perfil ligero, los cuales están centrados en el perfil *core*, y se extienden al perfil *transactional*; abarcando de esta forma la posibilidad de realizar consultas, adiciones y eliminaciones en los conceptos principales del SOS: sensores y observaciones.

- Análogamente, el SOSLite emplea los modelos de datos SensorML y O&M para él envió y recepción de la información de sensores y observaciones. Estos dos modelos de datos también se han simplificado siguiendo las indicaciones de la recomendación para el perfil ligero y luego se han adaptado para soportar las operaciones del perfil *transactional*.

- Tanto las simplificaciones en las operaciones, como las aplicadas a los modelos logran disminuir la complejidad para procesar las peticiones que se realizan al servicio. Esto hecho, repercute en la cantidad de memoria y procesador que tiene que emplear el SOSLite para su funcionamiento, haciéndolo menos exigente en la cantidad de recursos computacionales que debe tener disponibles. Así mismo, otras decisiones de la implementación tienen influencia en el desempeño general de la misma. El uso de una base de datos orientada a documentos mejora las prestaciones en las operaciones vinculadas a las consultas e inserciones (que a su vez son las operaciones más comunes en las SN); mientras que el uso de un servidor *Event driven* mejora el comportamiento frente a las peticiones simultáneas.

- El SOSLite, ha sido puesto a disposición de los investigadores de forma gratuita y bajo código libre, en la plataforma GitHub. El objetivo es que el uso de este mecanismo de interoperabilidad se extienda de forma que: (i) se convierta en un referente de implementación de perfil ligero de SOS, (ii) sea mejorado de forma colaborativa por la comunidad investigadora, (iii) sea estudiado en profundidad, (iv) las soluciones vinculadas a su desarrollo sean copiadas a otras implementaciones, (v) sus funcionalidades sean extendidas y (vi) se pruebe en un mayor número de casos.

- Por otra parte, el SOSLite está enfocado en las SN, pero su uso puede extenderse a otros CPS si se enmarca dentro de una plataforma de interoperabilidad. Esta implementación de SOS ligero, es ideal si se quiere emplear en conjunción con otras implementaciones de SOS, ya que al cumplir con el estándar podrán operar de manera conjunta. Este enfoque da lugar a un sistema de almacenamiento de datos y metadatos de sensores y observaciones, distribuido y jerarquizado; donde el SOSLite se despliega cerca de los sensores y otra implementación de SOS más completa se despliega en un servidor centralizado con grandes recursos.

- Aunque el enfoque abordado en el punto anterior es el más inmediato, no es el único escenario donde el SOSLite se ajusta de forma correcta. Ya que esta implementación puede ser usada como:

    - Micro-instancias de CC y FC, en las cuales se brinde almacenamiento de datos y metadatos de SN en un esquema de PAAS o SAAS.

    - Sistema almacenamiento distribuido, donde se prescinde de la estructura jerárquica y se potencia la disponibilidad de información empleando un almacenamiento redundantes.

    - Instancia nómada, el SOSLite así utilizado puede migrar de un dispositivo a otro de forma transparente y manteniendo sus configuraciones y los datos almacenados.

    - Servicio centralizado, aunque el SOSLite no fue creado con este propósito, no hay impedimento en utilizarlo de esta forma, siempre y cuando se asegure que no se necesita de una implementación de SOS completo para abordar el caso de uso.

- Las pruebas del SOSLite han permitido perfilar el comportamiento esperado de esta implementación bajo cuatro parámetros básicos: (i) uso de CPU, (ii) uso de Memoria RAM, (iii) *Throughput* (rendimiento) y (iv) *Delay* (retardo); y en el marco de dos situaciones extremas: (i) una SN que contiene pocos sensores que realizan muchas mediciones y (ii) una SN con muchos sensores que realizan pocas mediciones. Estas dos situaciones modelan bastante bien el uso de las SN para: (i) escenarios altamente variables y (ii) escenarios complejos. Así, de las pruebas realizadas se puede deducir que:

    - El SOSLite desplegado en un dispositivo limitado puede atender de forma excelente a los escenarios altamente variables; con un consumo de recursos acotado, un *Throughput* cercano a 6 peticiones/segundo y con un retardo cercano bajo los 500ms.

    - En el caso de los escenarios complejos, el SOSLite sobre un dispositivo limitado hace un uso comedido de los recursos, tiene un *Throughput* cercano a las 6 peticiones/segundo, pero el *Delay* alcanza cerca de 10 segundos; este hecho hace que se limite el uso que se puede dar al SOSLite en estos escenarios, de forma que, si se desea asegurar el buen servicio, se debe limitar el uso de este servicio a aquellos casos en los

cuales los sensores registran los datos a intervalos superiores a los 10 segundos.

- Por su parte, el uso del SOSLite como micro-instancia de CC y FC, presenta un comportamiento aceptable para los escenarios altamente variables; con un uso de recursos inferior al 50% para CPU y RAM, con un *Throughput* cercano a las 2 peticiones/segundo y un *Delay* que alcanza cerca de 4 segundos. De nuevo, los casos de uso donde se puede aplicar el SOSLite deben ser evaluados para determinar si es la solución adecuada para ofrecer un buen nivel de calidad de servicio.

- Finalmente, en los escenarios complejos el uso del SOSLite como micro-instancia sigue un patrón similar al observado en los dispositivos limitados, con un excelente uso de recursos pero con un tiempo de retardo que se encuentra en torno a los 13 segundos; limitando los casos donde se puede utilizar este mecanismo de interoperabilidad.

- En general el SOSLite presenta un comportamiento muy estable a lo largo del tiempo y en las dos situaciones planteadas; esto permite que cuando se determinan los casos de uso en los cuales se va a emplear, como solución de interoperabilidad, se pueda definir si el SOSLite atiende correctamente a los requerimientos del caso y esperar ofrecer un buen nivel de calidad de servicio.

7.1.3. SOSFul

- El SOSFul es la segunda implementación de SOS ligero que se realizó en esta tesis, ha sido creada con base en los aprendizajes del SOSLite y algunos de los trabajos futuros planteados por la OGC. Fue especialmente diseñada para brindar interoperabilidad a nivel semántico en la IoT y para facilitar su uso sobre dispositivos de recursos limitados y como instancia de CC y FC. Parte de los mismos requerimientos que el SOSLite, siendo una implementación ligera que requiere de pocos recursos computacionales para su funcionamiento y además, realiza un uso más eficiente de los recursos de red. Para su construcción se tomaron las mismas referencias del SOSLite: SOS, SensorML, O&M y el perfil ligero; además de las construcciones propias del SOSLite. A partir de allí, se simplificaron las interfaces cambiando las usadas por el estándar, SOAP/XML, por unas REST/JSON. Este mecanismo de interoperabilidad no es compatible con las otras versiones de SOS aunque los conceptos empleados y las operaciones ofrecidas son equivalentes. El SOSFul permite hacer CRUD sobre: sensores, observaciones y áreas de interés.

- Así, aprovechando el trabajo futuro descrito en el estándar SOS, que vislumbra el uso de interfaces REST en lugar de las SOAP sobre las que está construido; el SOSFul plantea unas interfaces para realizar operaciones de consulta, eliminación, modificación y creación de sensores, observaciones y áreas de interés, siendo en este aspecto más completo que la unión de los perfiles *core* y *transactional*. Así mismo, el SOSFul es el encargo de administrar los identificadores que asocia a cada concepto, evitando problemas de duplicación. También, las operaciones pasan a estar estructuradas mediante una combinación

de URL + Verbo REST (GET, PUT, POST y DELETE) y los parámetros de la configuración de las operaciones, son pasados como *query* de la URL en formato JSON.

- Para la simplificación de los modelos de datos se parte de los modelos empleados en el SOSLite y se hace un cambio de lenguaje, desde XML hasta JSON, que es un lenguaje más compacto pero a su vez muy flexible. Así, SensorML y O&M encuentran una representación en JSON que ocupa menos bits y que permite un uso más acotado del ancho de banda disponible, al tiempo que facilita el análisis sintáctico de los mensajes.

- La unión de unas interfaces simplificadas y unos modelos de datos adaptados del SOSFul, tienen una influencia positiva sobre el uso de recursos, tanto computacionales como de red. Haciendo que toda petición realizada al SOSFul ocupe menos bits y se atienda más rápidamente. Con miras a lograr el máximo rendimiento con el mínimo consumo de recursos se emplea una base de datos orientada a documentos y un servidor Event driven, los cuales se acoplan de forma ideal con el uso de REST/JSON.

- En el caso del SOSFul se ha optado por ponerlo a disposición de la comunidad investigadora bajo licencia GPL, siendo entonces un servicio gratuito y de código abierto. Se ha empleado GitHub como repositorio de código y se ha creado una VM ligera, usando Docker, para facilitar la instalación y distribución del servicio. Así se promueve que: (i) se extiendan las funcionalidades, (ii) se convierta en un punto de referencia para el próximo estándar de SOS, (iii) se pruebe en nuevos casos de uso para comprobar su potencialidad y versatilidad, (iv) se estudie el código y las mejoras realizadas se copien a otros desarrollos, (v) se realicen herramientas de integración con el SOSFul y (vi) se mejore el propio servicio.

- El SOSFul está ideado para funcionar dentro de la IoT como mecanismo de interoperabilidad semántica bajo esquemas centralizados o distribuidos. Puede ser desplegado en un amplio rango de plataformas: desde dispositivos limitados hasta entornos ilimitados en la nube. Se espera que su uso se extienda en primer lugar en sistemas distribuidos de almacenamiento de datos de sensores y observaciones, donde herramientas modernas, como los *ContextBroker*, puedan aprovecharlos para construir plataformas de interoperabilidad para IoT. Sin embargo, el SOSFul puede servir de almacenamiento centralizado de la información de las SN que pertenecen a múltiples CPS como las *SmartGrids*, las *SmartCities* o los *Drones*.

- Las pruebas del SOSFul se centraron en comparar su rendimiento frente a dos alternativas basadas en el estándar SOS: 52North SOS y SOSLite. En lugar de revisar casos extremos se mantuvo un escenario unificado que presenta un caso que común dentro de la IoT: atención de múltiples clientes (10 en este caso) que realizan operaciones de forma rápida (10 operaciones por segundo). Con esto se modelan situaciones en las cuales varios sensores ingresan sus mediciones de forma rápida, es decir que el sistema contiene pocos sensores que realizan muchas mediciones para manejar los entorno variables. De igual forma, modela el caso en el que múltiples servicios desean acceder a la información almacenada

en el SOSFul. A partir de las pruebas comparativas se desprende:

  - El SOSFul realiza un mejor uso del procesador que las dos alternativas evaluadas, hecho vinculado a sus interfaces y modelos simplificados que permiten realizar menos operaciones de procesamiento.

  - Tanto el SOSFul como el SOSLite aprovechan al máximo el uso de la memoria RAM, esto se desprende del uso de servidores *Event drive*, que facilita la atención de operaciones simultanea sin una asignación desmedida de memoria para cada petición.

  - El Throughput en el SOSFul es cerca de un 50% mejor en comparación a las otras alternativas evaluadas. En este caso se puede atribuir a una mezcla de factores: (i) interfaces y modelos simplificados, (ii) servidor *Event drive* y (iii) uso de base de datos orientada a documentos con integración con JSON.

  - Tiempo de respuesta del SOSFul se encuentra por debajo de los 50 ms y permanece estable a lo largo del tiempo, lo que implica un número máximo de transacciones por segundo con muy poca variabilidad, superando en este aspecto a los otros dos servicios.

- A partir de las pruebas realizadas se puede determinar que el SOSFul es un servicio muy estable y que mejora las prestaciones del sistema, al tiempo que usa pocos recursos computacionales y de red; convirtiéndolo en una excelente opción para emplear como mecanismo de interoperabilidad semántica en las IoT y en otros CPS.

7.1.4. Conclusiones generales

- El desarrollo de los CPS ha comenzado a ser relevante en el ámbito de la investigación en los últimos años; sin embargo, estos sistemas llevan años siendo utilizados. Este paradigmático comportamiento obedece al desarrollo como disciplina que busca organizar, estructurar y sistematizar el compendio de conocimientos que se han desarrollado a lo largo del tiempo, de forma que se tenga una aproximación metodológica integral hacia estos sistemas complejos que abordan desde sencillos dispositivos hasta grandes redes.

- Sin duda alguna el CPS de mayor auge investigativo y comercial es la IoT, que ha alcanzado el estado de madures necesario para abordar los mercados masivos. Sin embargo, en la actualidad la IoT y otros tantos CPS se enfrentan al problema de interoperabilidad. Son muchos los grupos de investigación y las empresas que están abordando este tema desde diferentes aproximaciones; esta tesis se centra en proveer un mecanismo que permita dicha interoperabilidad a nivel semántico, partiendo de los desarrollos que existen actualmente y que son ampliamente aceptados y utilizados, considerando que esta aproximación puede ser más provechosa y que brinda un avance solido en el estado del arte actual.

- Tanto el SOSLite como el SOSFul se construyen sobre el concepto de SOS

ligero como mecanismo de interoperabilidad para los sistemas que emplean SN. Las dos implementaciones son aproximaciones en desarrollo pero que han mostrado sus ventajas, teniendo aun un camino por recorrer para ser un producto de mercado que atienda a todas las necesidades de interoperabilidad de los CPS.

- Así, el SOSLite es una evolución para el estándar SOS, debido a que se apropia de ideas novedosas como: el uso de servidores *Event drive*, la utilización de una base de datos orientadas a documento, la simplificación de los modelos de datos y de las operaciones. Por su parte, SOSFul es una revolución, encaminada a renovar un estándar que resulta pesado para las necesidades actuales de los CPS; cambiando las interfaces, haciéndolas ligeras y fáciles de integrar con otras soluciones.

7.2. Líneas futuras de investigación

Si bien es posible ampliar el estudio realizado en esta tesis en múltiples direcciones, en esta sección se ofrecen algunas opciones, tanto en el ámbito del SOSLite y el SOSFul, como en los casos de uso presentados.

7.2.1. SOSLite

- *Extender la experimentación del SOSLite*, para lo cual se plantean las pruebas de todas las operaciones en diversos escenarios que modelen casos de uso comunes en las SN y en IoT; incluyendo escenarios de tiempo real y escenarios con sensores en movimiento.

- *Evaluación experimental*, emplear el SOSLite en casos de uso integrados dentro de las SN; por ejemplo: mediciones ambientales en entornos cerrados en oficinas y fábricas, y estaciones meteorológicas móviles para agricultura de precisión.

- *Refinamiento*, optimizar el funcionamiento del SOSLite para que realice un menor consumo de recursos en el procesamiento de los datos y mejorar el desempeño frente a escenarios con gran cantidad de sensores.

- *Ampliación de las funcionalidades*, incluyendo los perfiles *Enhanced* y *Result Handling* que puedan ser activados de forma opcional en la configuración de la implementación.

- *Consolidación*, realizar una mayor divulgación del proyecto a nivel nacional e internacional, incentivando el uso de la implementación a nivel investigativo. Presentar el SOSLite a la OGC como implementación de referencia para el perfil ligero.

- *Uso en proyectos europeos*, emplear el SOSLite en el marco de los proyectos europeos y nacionales que está llevando el grupo de investigación de forma que se emplee en entornos reales y que aumente la proyección a nivel internacional.

7.2.2. SOSFul

- *Extender la experimentación de la plataforma*, probar la totalidad de las operaciones, modelando diferente situaciones, como las propuestas para el SOSLite, de forma que se tenga un abanico de pruebas que valide la versatilidad de la implementación.

- *Evaluación experimental*, extender el caso de uso de ciudades inteligentes y probarlo a mayor escala para validar los resultados obtenidos. Además, integrarlo a nuevos proyectos de objetos inteligentes donde se evalué la

adaptación del SOSFul.

- *Refinamiento*, mejorar la implementación con una versión posterior en la cual se optimice al máximo la base de datos de forma que las consultas y las inserciones de observaciones tengan un aumento en rendimiento. Además, mejorar el sistema de extensiones para facilitar el desarrollo de nuevas funcionalidades.

- *Ampliación de las funcionalidades*, incluir nuevas extensiones, como el ya desarrollado para atender al protocolo CoAP, comenzando con la extensión para el uso de MQTT.

- *Consolidación*, mejorar la documentación y ampliar la divulgación para llamar la atención sobre el proyecto. Presentarlo a la OGC como referencia para la próxima versión del estándar SOS.

- *Uso en proyectos de empresariales* y europeos, realizar alianzas con empresas para comenzar pruebas piloto de utilización de la implementación en entornos reales; así mismo, utilizar el SOSFul dentro de los proyectos nacionales y europeos que está desarrollando el grupo de investigación.

7.2.3. Casos de Uso

- *Logística de transporte intermunicipal de pasajeros*, presentar el caso de uso desarrollado en el 7 congreso de transporte intermunicipal de pasajeros de la ANDI (Asociación Nacional de Industriales - Colombia) y buscar socios para llevar a cabo la prueba piloto.

- SmartCities, consolidar la línea de investigación en SmartCity + CPS y presentar proyectos de investigación para desarrollar el caso de uso en un entorno real. Avanzar hacia la propuesta de inteligencia integrada en las ciudades que permite unificar los diferentes sistemas inteligentes de la ciudad y generar nuevos servicios al ciudadano.

8. Referencias

8.1. Bibliografía

[1] I. Stojmenovic, “Machine-to-Machine Communications with In-network Data Aggregation, Processing and Actuation for Large Scale Cyber-Physical Systems,” *IEEE Internet Things J.*, vol. PP, no. 99, pp. 1–1, 2014.

[2] P. Soulier, L. Depeng, J. R. Williams, D. Li, J. R. Williams, L. Depeng, J. R. Williams, D. Li, and J. R. Williams, “A survey of language-based approaches to Cyber-Physical and embedded system development,” *Tsinghua Sci. Technol.*, vol. 20, no. 2, pp. 130–141, 2015.

[3] R. Rajkumar, “A cyber-physical future,” *Proc. IEEE*, vol. 100, no. SPL CONTENT, pp. 1309–1312, 2012.

[4] M. Matin, "*Wireless Sensor Networks - Technology and Protocols,"* InTech, 2012.

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–105, 2002.

[6] European Research Cluster on IoT - IERC, “Internet of Things,” 2015. [Online]. Available: http://www.internet-of-things-research.eu/about_iot.htm.

[7] M. Serrano, P. Barnaghi, F. Carrez, P. Cousin, O. Vermesan, and P. Friess, “Internet of Things IoT Semantic Interoperability: Research Challenges, Best Practices, Recommendations and Next Steps," European Research Cluster on IoT - IERC, 2015.

[8] H. Van Der Veer and A. Wiles, “Achieving Technical Interoperability,” the ETSI Approach, no. 3, p. 29, 2008.

[9] T. S. Cinotti, “Semantic Interoperability Architecture for Pervasive Computing and Internet of Things 1,” vol. 2, 2014.

[10] K. Rose, S. Eldridge, and C. Lyman, “The internet of things: an overview,” no. October, p. 53, 2015.

[11] Mckinsey Global Institute, “the Internet of Things : Mapping the Value Beyond the Hype,” 2015.

[12] C. Perera, S. Member, A. Zaslavsky, and P. Christen, “Context Aware Computing for The Internet of Things : A Survey,” *IEEE Commun. Surv. Tutor.*, vol. X, no. X, pp. 1–41, 2015.

[13] G. Percivall, C. Reed, and J. Davidson, “Open Geospatial Consortium Inc . OGC White Paper OGC ® Sensor Web Enablement : Overview And High Level Architecture,” *2007 IEEE Autotestcon*, vol. 4540, no. December, pp. 1–14, 2007.

[14] Swetina, Jorg, et al., “Toward a standardized common M2M service layer platform: Introduction to oneM2M,” IEEE Wireless Communications 21.3 (2014): 20-26.

[15] The Open Group, “Open Data Format (O-DF),” 2014.

[16] The Open Group, “Open Messaging Interface (O-MI),” 2016.

[17] A. Bröring, C. Stasch, and J. Echterhoff, “OGC® Sensor Observation Service Interface Standard,” 2012.

[18] A. Bröring, J. Echterhoff, S. Jirka, I. Simonis, T. Everding, C. Stasch, S. Liang,

and R. Lemmens, “New generation Sensor Web Enablement,” *Sensors*, vol. 11, no. 3, pp. 2652–2699, Jan. 2011.

[19] Norad, “The Logical Framework Approach (LFA) - Handbook for objectives-oriented planning,” *Zhurnal Eksp. i Teor. Fiz.*, p. 107, 1999.

[20] Instituto Nacional de Tecnologías de la Comunicación. Inteco, “Ingeniería del Software: Metodologías y Ciclos de Vida,” *Inteco*, p. 83, 2009.

[21] L. G. Roberts, “Beyond Moore’s law: Internet growth trends,” *Computer (Long. Beach. Calif).*, vol. 33, no. 1, pp. 0–2, 2000.

[22] C. A. MacK, “Fifty years of Moore’s law,” *IEEE Trans. Semicond. Manuf.*, vol. 24, no. 2, pp. 202–207, 2011.

[23] I. Ferain, C. a. Colinge, and J.-P. Colinge, “Multigate transistors as the future of classical metal–oxide–semiconductor field-effect transistors,” *Nature*, vol. 479, no. 7373, pp. 310–316, 2011.

[24] B. K. Ahmed and K. Schuegraf, “Rival architectures face off in a bid to keep Moore ’ s Law alive,” pp. 1–7, 2014.

[25] T. Coughlin, “Keeping Data For A Long Time,” *Forbes*, pp. 1–6, 2014.

[26] F. Pivec, "Measuring the information society," vol. 8, no. 3. 2014.

[27] I. Society, “Internet Society Global Internet Report 2014,” p. 146, 2014.

[28] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, “Vision and Challenges for Realising the Internet of Things,” European Union, 2010.

[29] International Telecommunication Union, "*Measuring the information society: The ICT development index"*. 2015.

[30] R. Drath and A. Horch, “Industrie 4.0: Hit or hype? ,” *IEEE Ind. Electron. Mag.*, vol. 8, no. 2, pp. 56–58, 2014.

[31] B. Vogel-Heuser and D. Hess, “Guest Editorial Industry 4.0-Prerequisites and Visions,” *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 411–413, 2016.

[32] V. G. Hamrita Takoi K., S. Durrence Jeffrey, “Precision Farming Practices,” *Ieee Ind. Appl. Mag.*, vol. 15, pp. 34–42, 2009.

[33] A. Bleicher, “Farming by the numbers,” *IEEE Spectr.*, vol. 50, no. 6, pp. 38–44, 2013.

[34] R. Poovendran, K. Sampigethaya, S. K. S. Gupta, I. Lee, K. V. Prasad, D. Corman, and J. L. Paunicka, “Special Issue on Cyber-Physical Systems,” *Proc. IEEE*, vol. 100, no. 1, pp. 6–12, 2012.

[35] J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang, “Toward a science of cyber-physical system integration,” *Proc. IEEE*, vol. 100, no. 1, pp. 29–44, 2012.

[36] F. J. Villanueva, J. Albusac, L. Jimenez, D. Villa, and J. C. Lopez, “Architecture for smart highway real time monitoring,” *Proc. - 27th Int. Conf. Adv. Inf. Netw. Appl. Work. WAINA 2013*, pp. 1277–1282, 2013.

[37] S. Pagadarai, B. A. Lessard, A. M. Wyglinski, R. Vuyyuru, and O. Altintas, “Vehicular communication: Enhanced networking through dynamic spectrum access,” *IEEE Veh. Technol. Mag.*, vol. 8, no. 3, pp. 93–103, 2013.

[38] L. Januszkiewicz, J. Kawecki, R. Kawecki, and P. Oleksy, “Wireless indoor positioning system with inertial sensors and infrared beacons,” *2016 10th Eur. Conf. Antennas Propag.*, pp. 1–3, 2016.

[39] S. Subedi, G.-R. Kwon, Seokjoo Shin, Suk-seung Hwang, and Jae-Young Pyun, “Beacon based indoor positioning system using weighted centroid localization approach,” *2016 Eighth Int. Conf. Ubiquitous Futur. Networks*, pp. 1016–1019, 2016.

[40] L. Fegaras, “Incremental Query Processing on Big Data Streams,” *CoRR*, vol. 4347, no. c, pp. 1–14, 2015.

[41] G. C. Fox, S. Kamburugamuve, S. Ekanayake, M. Pathirage, and G. Fox, “Towards High Performance Processing of Streaming Data in Large Data Centers Towards High Performance Processing of Streaming Data in Large Data Centers,” no. May, pp. 1637–1644, 2016.

[42] Y. Sun, X. Wang, and X. Tang, “Hybrid deep learning for computing face similarities,” *Int’l Conf. Comput. Vis.*, vol. 38, no. 10, pp. 1997–2009, 2013.

[43] L. J. Dwd and D. Q. G. Hhs, “Big Data and Deep Learning,” no. 2015, pp. 11–16, 2016.

[44] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks : a survey,” vol. 38, no. 5, pp. 393–422, 2002.

[45] M. Ilyas and I. Mahgoub, "*Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*." CRC Press, 2004.

[46] IERC, “Internet of Things,” *IERC*, 2014. [Online]. Available: http://www.internet-of-things-research.eu/about_iot.htm. [Accessed: 20-Jun-2007].

[47] Gartner, “Internet of Things,” *Gartner*, 2012. [Online]. Available: https://www.gartner.com/it-glossary/internet-of-things/. [Accessed: 20-Jun-2007].

[48] K. Ashton, “That ‘Internet of Things’ Thing,” *RFID Journal*, 2009. [Online]. Available: http://www.rfidjournal.com/articles/view?4986. [Accessed: 01-Jul-2016].

[49] U. S. Profile, “The Digital Universe in 2020 : Big Data , Bigger Digital Shadows, and Biggest Growth in the Far East,” *IDC iView IDC Anal. Futur.*, vol. 2007, pp. 1–7, 2013.

[50] D. Evans, “The Internet of Things - How the Next Evolution of the Internet is Changing Everything,” *CISCO white Pap.*, no. April, pp. 1–11, 2011.

[51] D. Lee, D. Lough, S. Midkiff, N. Davis, and P. Benchoff, “The next generation of the internet: aspects of the ipv6,” *IEEE Netw.*, vol. 12, no. April, pp. 28–33, 1998.

[52] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and Marrs, “Disruptive technologies: Advances that will transform life, business, and the global economy,” *McKinsey Glob. Insitute*, no. May, p. 163, 2013.

[53] Beecham Research, “IoT Sector Map,” *Beecham Research*, 2011. [Online]. Available: http://www.beechamresearch.com/article.aspx?id=4. [Accessed: 01-Jul-2016].

[54] Zhihong Yang, Yufeng Peng, Yingzhao Yue, Xiaobo Wang, Yu Yang, and Wenji Liu, "Study and application on the architecture and key technologies for IOT," *2011 Int. Conf. Multimed. Technol.*, pp. 747–751, 2011.

[55] L. Tan, "Future internet: The Internet of Things," *2010 3rd Int. Conf. Adv. Comput. Theory Eng.*, pp. V5-376-V5-380, 2010.

[56] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[57] M. Wu, T.-J. T. Lu, F.-Y. Ling, J. Sun, and H. Du, "Research on the architecture of Internet of Things," *2010 3rd Int. Conf. Adv. Comput. Theory Eng.*, pp. V5-484-V5-487, 2010.

[58] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," *Proc. - 10th Int. Conf. Front. Inf. Technol. FIT 2012*, pp. 257–260, 2012.

[59] A. Al-fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," vol. 17, no. 4, pp. 2347–2376, 2015.

[60] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," *Natl. Inst. Stand. Technol. Inf. Technol. Lab.*, vol. 145, p. 7, 2011.

[61] L. Wang, M. T??rngren, and M. Onori, "Current status and advancement of cyber-physical systems in manufacturing," *J. Manuf. Syst.*, 2015.

[62] M. Firdhous, O. Ghazali, and S. Hassan, "Fog Computing: Will it be the Future of Cloud Computing?," *Third Int. Conf. Informatics Appl.*, pp. 8–15, 2014.

[63] J. E. Smith and R. Nair, "The architecture of virtual machines," *Computer (Long. Beach. Calif).*, vol. 38, no. 5, pp. 32–38, 2005.

[64] A. Manzalini, R. Minerva, F. Callegati, W. Cerroni, and A. Campi, "Clouds of virtual machines in edge networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 63–70, 2013.

[65] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs containerization to support PaaS," *Proc. - 2014 IEEE Int. Conf. Cloud Eng. IC2E 2014*, pp. 610–614, 2014.

[66] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," *2015 IEEE Int. Symp. Perform. Anal. Syst. Softw.*, pp. 171–172, 2015.

[67] S. Havens, "Transducer Markup Language Implementation Specification," 2006.

[68] M. Botts and A. Robin, "OGC® SensorML," 2014.

[69] S. Cox, "Observations and Measurements - XML Implementation," 2011.

[70] I. Simonis, "Sensor Alert Service," 2006.

[71] I. Simonis, "Sensor Planning Service Implementation Specification," 2007.

[72] I. S. y Echterhoff, "Draft OpenGIS® Web Notification Service Implementation Specification," 2006.

[73] D. N. y A. Whiteside, "OpenGIS Catalog Services Specification," 2005.

[74] 52 North, “52 North. Sensor Web Community.” [Online]. Available: http://52north.org/communities/sensorweb/index.html. [Accessed: 20-Jun-2007].

[75] Jeff McKenna, “Web de MapServer SOS.” [Online]. Available: http://mapserver.org/ogc/sos_server.html. [Accessed: 20-Jun-2007].

[76] “Web de Deegree SOS.” [Online]. Available: http://wiki.deegree.org/deegreeWiki/deegree3/SensorObservationService.

[77] S. Jirka, C. Stasch, and A. Bröring, “OGC® Best Practice for Sensor Web Enablement Lightweight SOS Profile for Stationary In-Situ Sensors,” 2014.

[78] R. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” University of California, Irvine, 2000.

[79] C. Stasch and B. Practice, “Open Geospatial Consortium OGC ® Best Practice for Sensor Web Enablement Lightweight SOS Profile for Stationary In-Situ Sensors,” 2014.

[80] et all Rogério dos Santos Alves; Alex Soares de Souza, “Qué es una aplicación,” *Igarss 2014*, no. 1, pp. 1–5, 2014.

[81] D. Kulak and E. Guiney, *Use Cases: Requirements in Context*. Addison-Wesley, 2004.

[82] I. Sommerville and M. I. A. Galipienso, "*Ingeniería del software"*. Pearson Educación, 2005.

[83] I. of E. and E. Engineers, “Especificacion de Requisitos segun el estandar de IEEE 830,” p. 27, 2008.

[84] N. M. Josuttis, *SOA in Practice: The Art of Distributed System Design*. O’Reilly Media, 2007.

[85] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Pearson Education, 2012.

[86] J. Dongarra, L. T. Yang, O. F. Rana, and B. Di Martino, *High Performance Computing and Communications: First International Conference, HPCC 2005, Sorrento, Italy, September, 21-23, 2005, Proceedings*. Springer Berlin Heidelberg, 2005.

[87] H. Hayder, *Object-Oriented Programming with Php5*. Packt Publishing, Limited, 2007.

[88] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software (Adobe Reader)*. Pearson Education, 1994.

[89] K. Chodorow and M. Dirolf, *MongoDB: The Definitive Guide*. O’Reilly Media, 2010.

[90] B. Dayley, *NoSQL with MongoDB in 24 Hours, Sams Teach Yourself*. Pearson Education, 2014.

[91] K. Janowicz, A. Bröring, C. Stasch, S. Schade, T. Everding, and A. Llaves, “A RESTful proxy and data model for linked sensor data,” *Int. J. Digit. Earth*, vol. 6, no. 3, pp. 233–254, 2013.

[92] H. Truong and S. Dustdar, “Principles for Engineering IoT Cloud Systems,” 2015.

[93] M. Aazam and E.-N. Huh, “Fog Computing: The Cloud-IoT\/IoE Middleware Paradigm,” *IEEE Potentials*, vol. 35, no. 3, pp. 40–44, 2016.

[94] P. Rashidi and A. Mihailidis, “A survey on ambient-assisted living tools for older adults,” *IEEE J. Biomed. Heal. Informatics*, vol. 17, no. 3, pp. 579–590, 2013.

[95] R. Faragher and R. Harle, “An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications,” *Proc. 27th Int. Tech. Meet. Satell. Div. Inst. Navig. (ION GNSS+ 2014)*, pp. 201–210, 2014.

[96] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 6, pp. 1067–1080, 2007.

[97] K. Su, J. Li, and H. Fu, “Smart city and the applications,” *2011 Int. Conf. Electron. Commun. Control. ICECC 2011 - Proc.*, pp. 1028–1031, 2011.

[98] European Parliament, “Mapping Smartcities in the EU,” 2014.

9. Anexo 1. Glosario

9.1. Términos y acrónimos

CPS	Cyber-Physical System
IoT	Internet of Things
SN	Sensor Network
CC	Cloud Computing
CF	Fog Computing
SWE	Sensor Web Enablement
OGC	Open Geospatial Consortium
SOS	Sensor Observation Service
SensorML	Sensor Model Language
O&M	Observation & Measurements
WWW	World Wide Web
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
VM	Virtual Machine
LXC	Linux Containers
SOA	Service Oriented Architecture
SOSoB	SOSLite on Board
iT-Hub	intelligent Transport Hub
ITI	Intercity Transport Interchanges
M2M	Machine to Machine
CoAP	Constrained Application Protocol
TIC	Tecnologías de la Información y las comunicaciones
RF	Requerimiento Funcional
RNF	Requerimiento no Funcional
DS	Diagrama de Secuencia
DA	Diagrama de Actividad