PROYECTO FINAL DE INGENIERÍA

ARQUITECTURA PARA LA REPLICACIÓN PROACTIVA DE CONJUNTOS DE CONTENIDOS EN REDES CON RETRASOS Y/O INTERRUPCIONES

Prieto Galardo, Nahuel Agustín – LU1046126

Ingeniería en Informática

Tutor:
Giaccio, Gustavo, UADE

9/3/2021

UNIVERSIDAD ARGENTINA DE LA EMPRESA

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

Agradecimientos

A Gustavo Giaccio (UADE y Cisco), mi tutor, por las revisiones, los consejos, y el apoyo incondicional desde el principio del proyecto.

A Gabriel Sakata (BVS España) por su revisión y punto de vista holístico de la solución en contraste con CDN, Edge Computing y Fog Computing.

A David J. Edell (Johns Hopkins University), Scott C. Burleigh (Caltech/NASA JPL), y Carlo Caini (Università di Bologna), por el soporte brindado para el correcto uso de la tecnología ION en este proyecto.

A Matías Miguenz, Mauro Della Chiesa y Pablo Gotthelf (Spark Digital), por haber revisado el proyecto y por los consejos brindados para su mejora antes de la presentación final.

A Deepti Agarwal (Kaplan) y Mauricio Salafia (Spark Digital), por su comprensión y apoyo durante la etapa más difícil del desarrollo del proyecto.

A Mauro Cappella, Alexis Flores, Pablo Micieli y Jorge Aiello (YPF), por el tiempo dedicado a proveer sus perspectivas sobre la posible aplicación del proyecto en entornos reales de producción energética.

A Rodolfo Álvarez (YPF) por proveer y explicar documentación técnica existente sobre uno de los posibles escenarios de aplicación del proyecto.

A Nicolás Rueda (CGC) por su perspectiva sobre la potencial aplicación del proyecto en los campos ubicados al sur de Río Gallegos, en Santa Cruz.

A Juan Kulichevsky (estudiante de Ingeniería Aeroespacial en UNLP) y Sara Rios Ruiz (estudiante de Física en FCEN) por el apoyo y la guía en las revisiones de los temas del proyecto relevantes a sus campos de estudio.

A Lucas Schuft (Spark Digital y exalumno UADE), Nahir Jara (Taligent) y Ezequiel Haydossian (Agnostic) por las revisiones técnicas brindadas a lo largo del proyecto.

A todos los que ayudaron directa o indirectamente al progreso del proyecto.

A mis padres, por su apoyo sin igual desde el inicio de la carrera.

Resumen

A medida que la expansión territorial del ser humano va alcanzando niveles cada vez más elevados, mantener la calidad de las comunicaciones entre los distintos puntos del territorio se va haciendo progresivamente más complicado. Nuevos desafíos se presentan constantemente debido a los difíciles entornos en los que seres humanos y máquinas deben desempeñar su labor. En particular, hay dos desafíos que son los que hacen al alcance del objetivo de este trabajo, los cuales son las crecientes distancias entre punto y punto de las comunicaciones, en especial en la industria espacial, y los medios poco estables que provocan interrupciones de forma intermitente en el flujo físico de datos, como ocurre en entornos mineros o submarinos, entre otros. Ante las dificultades mencionadas, se ofrece como mitigación la solución presentada en este documento. La misma propone que los datos y la información necesaria sean entregados a cada nodo de forma anticipada, antes de ser accedidos, y que se mantengan siempre tan actualizados como sea posible. Esta solución es presentada en forma de modelo teórico, y se incluye una implementación no productiva como prueba de concepto, que demuestra que la arquitectura propuesta puede desempeñarse en contextos interplanetarios. El espíritu del proyecto es el de abrir una discusión sana sobre la forma en la que las personas colaboran con la tecnología que tienen al alcance, y sobre la necesidad de adoptar una nueva perspectiva para concebir modalidades más universales de hacerlo.

Abstract

As the territorial expansion of humanity reaches ever higher levels, maintaining the quality of communications between the different points of our territory becomes increasingly complicated. New challenges constantly arise because of the difficult environments in which humans and machines must perform their duties. In particular, there are two challenges which closely relate to the goal of this project, which are the increasing distances between point and point of communications, especially in the space industry, and the unstable connections that cause intermittent interruptions in the physical flow of data, as occurs in mining or underwater environments, among others. In view of these difficulties, the solution presented in this document is offered as mitigation. It proposes that the necessary data and information be delivered to each node in advance, before they are accessed, and that they be kept as up-to-date as possible. This solution is presented in the form of a theoretical model, and includes a non-productive implementation as proof of concept, which proves that the proposed architecture can be used in interplanetary contexts. The spirit of the project is to open a healthy discussion about how people collaborate with the technology at their disposal, and about the need to adopt a new perspective to devise more universal ways to do so.

Contenidos

Introducción

En la vida cotidiana del humano moderno, las comunicaciones instantáneas son algo que se da por hecho, y la gente está acostumbrada a la interacción con personas y máquinas que se sitúan en lugares posiblemente remotos. Sería ingenuo creer que este es el estado natural de las cosas: este es sólo un estado particular, y muy especial, al que no conviene acostumbrarse mucho, por dos factores limitantes.

En primer lugar, hay zonas en las que aún no existe una infraestructura de red que soporte un servicio de alta calidad. En ciertos yacimientos petroleros, como los ubicados en Vaca Muerta, Argentina, el despliegue de redes de 4G privadas (El Cronista, 2021) y LPWAN (Más Energía, 2020) para lograr conectividad local están aún en etapas tempranas. En algunos casos, debido a la falta de infraestructura local, la única opción disponible para conectarse a internet es a través de un enlace satelital. Por ello, las conexiones a internet que se pueden lograr desde estas zonas sufren interrupciones reiteradas. A pesar de ser esto indeseable, muchas veces el costo de una inversión en infraestructura local para remediar la situación supera los posibles beneficios. En la terminología existente, para referirse a este factor limitante se hace uso de la palabra “disrupción”, debido a dichas interrupciones (en algunas referencias bibliográficas también se menciona el término “conectividad episódica”, para referirse a la forma en que la conectividad se establece y se pierde de forma crónica). A pesar de que, como fue mencionado, muchas veces el costo de la inversión supera su potencial beneficio, este factor limitante es, en teoría, posible de eliminar, a diferencia del segundo factor limitante, el cual se examina a continuación.

Contemple ahora el lector una limitación natural que, con el conocimiento científico existente, no puede ser esquivada: el intercambio útil de información a una velocidad mayor que la de la luz no es posible (Wynne, 2002). Y la velocidad de la luz, a diferencia del espacio ocupado por la civilización, es, lamentablemente, una constante universal. Con el conocimiento científico actual, no hay forma, ni aún con todo el dinero del mundo invertido en tareas de ingeniería, de que entes en planetas distintos mantengan una comunicación interactiva. Simplemente no es posible. El planeta que más cerca llega a estar de la Tierra está a $38.2 \times 10^6\ km$ de distancia en su punto más próximo a esta (Williams, 2018). Esto significa que cualquier respuesta a un mensaje enviado se haría esperar, como mínimo, poco más de unos

4 minutos aproximadamente, asumiendo que la señal se desplace a la velocidad de la luz. El término establecido en la industria para este segundo factor limitante es “retraso”, debido a lo que tardan los mensajes en recorrer grandes distancias.

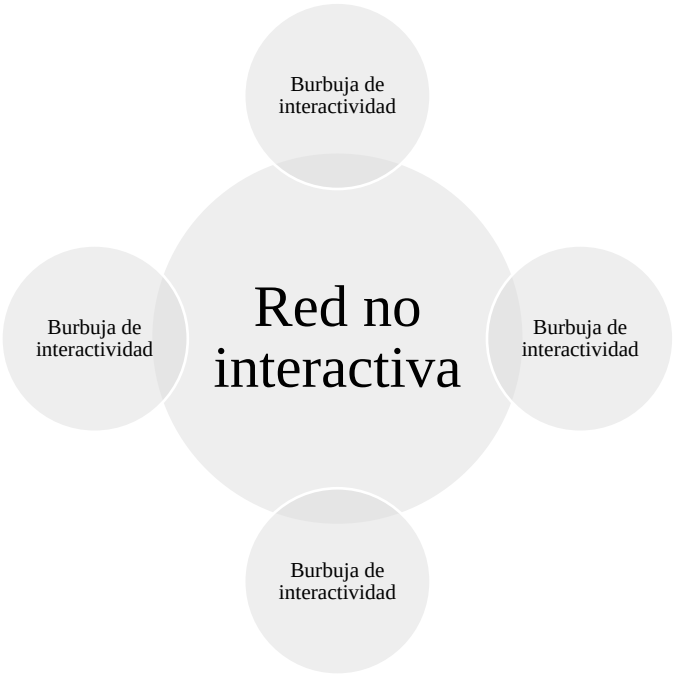

Figura 1: Red no interactiva de burbujas de interactividad

Mientras que el primer factor mencionado podría ser pasado por alto con obras de infraestructura (indistintamente de la rentabilidad de estas), la universalidad del segundo factor es lo que hace tan particular y escasa a la comodidad que se encuentra hoy en día en la colaboración interactiva. Si la humanidad logra establecerse como una civilización interplanetaria, la posibilidad de colaborar interactivamente con cualquier individuo de la sociedad humana dejará de existir en el instante en que personas comiencen a vivir en otros planetas, lunas, u objetos situados a más de unos pocos segundos luz de la Tierra. A partir de ese momento, la colaboración interactiva[1] será posible solo con un subconjunto de la civilización, dejando al resto “inalcanzable” a menos que sea por diferido. Desde ese momento, suponiendo una expansión progresiva en el espacio, la humanidad quedaría “fragmentada”,

[1] La “interactividad” no se refiere a aquella que se da entre el usuario y la interfaz de usuario, la cual es fundamental para el uso de cualquier aplicación, sino que se refiere a aquella que se da entre el emisor y el destinatario de un mensaje (en este caso separados ampliamente en tiempo).

formándose cada vez más "burbujas de interactividad[2]", que sólo se pueden comunicar entre sí por métodos totalmente asincrónicos. Ante esta posible situación, la única manera de mantener una universalidad en las comunicaciones, sacrificando la interactividad, es conectando estas burbujas en una red no interactiva, como se muestra en la Figura 1. Nótese que una locación muy afectada por disrupciones en la red (por ejemplo, una plataforma de petróleo en el océano) también puede ser considerada como una burbuja de interactividad, ya que la cantidad de reintentos necesarios para poder comunicarse con el mundo exterior conllevan una espera que derrota la posibilidad de que se efectivice la interacción.

Cuando se utilizan protocolos como HTTP a diario, los usuarios pueden esperar fácilmente un tiempo de ida de la solicitud desde la terminal al servidor, y de vuelta de la respuesta desde el servidor a la terminal. Es aceptable para ellos esperar estos tiempos ya que tanto la terminal como el servidor operan dentro de la misma burbuja de interactividad, como muestra la Figura 2.

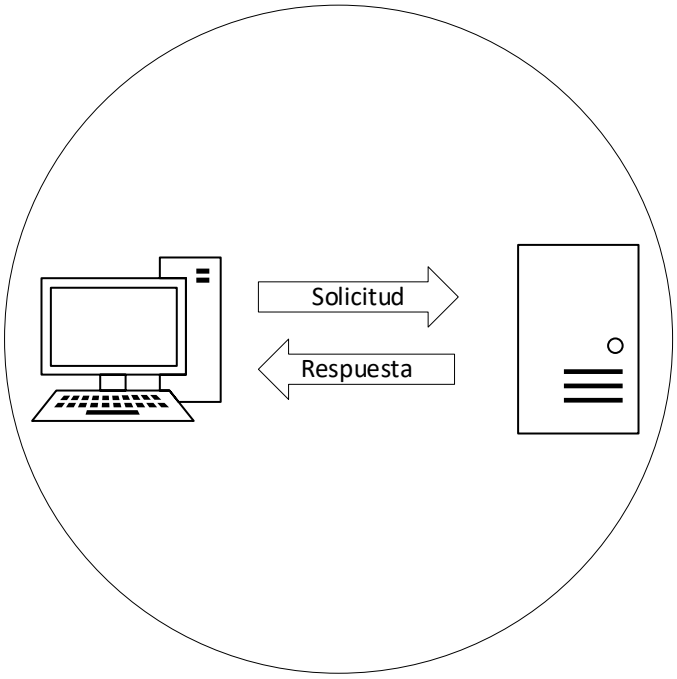

Figura 2: Ida y vuelta dentro de la misma burbuja de interactividad

[2] En inglés, el término técnico "continuum" que se ve en algunas referencias bibliográficas y estándares internacionales, se alinea con este concepto.

Cuando el contenido al que se desea acceder se encuentra en otra burbuja de interactividad, la situación se vuelve algo más compleja. En la Figura 3 se representa este segundo escenario.

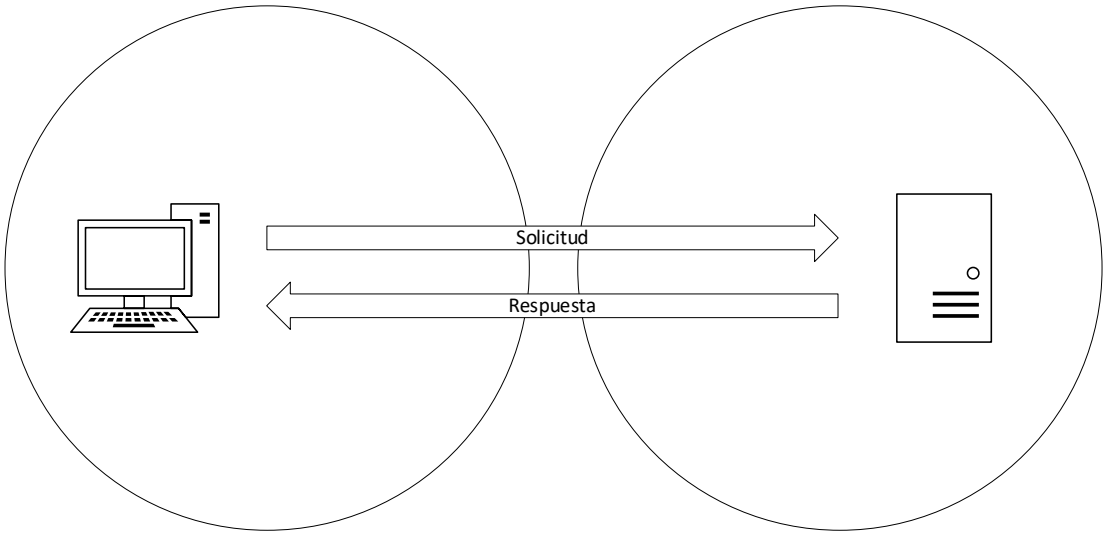

Figura 3: Ida y vuelta entre dos burbujas de interactividad diferentes

La pila de protocolos de red más ampliamente usada en el mundo, conocida como TCP/IP, no puede operar cuando cualquiera de los dos factores limitantes mencionados anteriormente entra en juego (Durst, y otros, 2002) y (Burleigh, y otros, 2019). Tanto retrasos como interrupciones severas hacen que los mensajes se terminen perdiendo o descartando. En otras palabras, los protocolos de la pila TCP/IP no pueden cruzar la frontera de una burbuja de interactividad. Por este motivo, el escenario que presenta la Figura 3 simplemente no puede ser resuelto utilizando estos protocolos tradicionales. Ya existen en la industria estándares e implementaciones de pilas de protocolos que pueden operar bajo estas condiciones (véase la sección de Antecedentes). Éstos hacen posible el escenario que se muestra en la Figura 3. Sin embargo, a pesar de ser posible, recordemos que la comunicación entre burbujas de interactividad diferentes no es aparentemente instantánea (lo que sí ocurre dentro de una misma burbuja). Por este motivo, trabajar con este esquema se vuelve impráctico, ya que para acceder a un determinado contenido hay que aguardar, como mínimo, en tiempo luz, el doble de la distancia que separa entre sí a ambas burbujas, desde el momento en que se expresó el deseo de acceder al mismo mediante la emisión de la solicitud.

Como el título de este informe lo indica, y como se detallará en la sección "Descripción", este proyecto se centra en la utilización de tecnologías y técnicas existentes para la replicación de conjuntos de contenido entre diferentes burbujas de interactividad. La misma es la solución que se propone a lo planteado en el párrafo anterior, ya que, como se muestra en la Figura 4, habilita a que quien esté usando la terminal para acceder a contenidos originados en otra burbuja de interactividad, pueda hacerlo tan pronto como sea posible, sin tener que aguardar tiempos adicionales a los estrictamente necesarios.

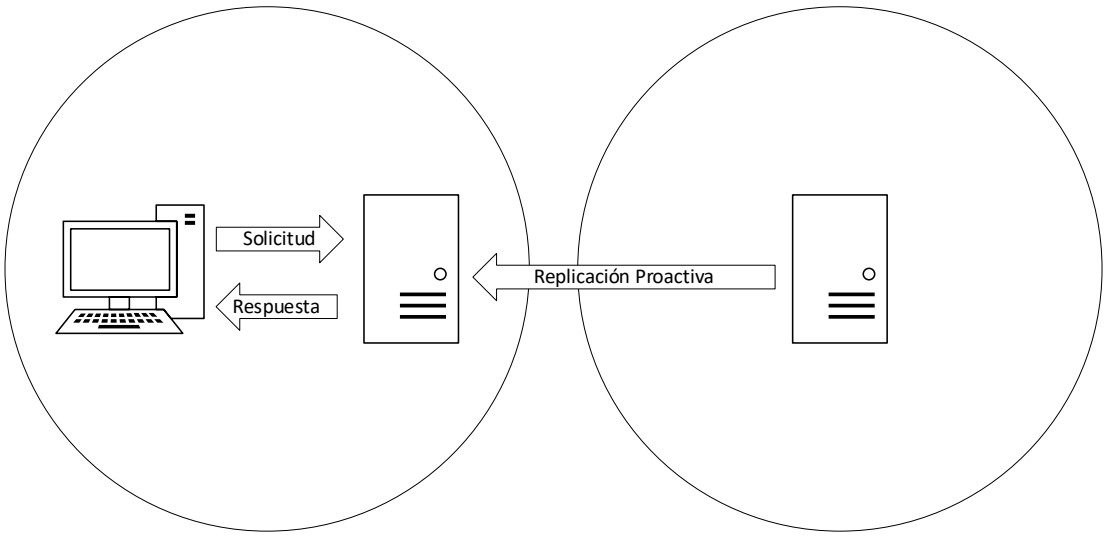

Figura 4: Replicación proactiva entre burbujas de interactividad

Esta ventaja temporal es la misma que push caching tiene sobre pull caching (Kumar, et al.), pero multiplicada por el crecimiento de la distancia entre las burbujas de interactividad.

En este modelo, la decisión de qué enviar sobre el enlace de replicación es crítica. Los enlaces con retrasos o interrupciones, y en este caso entre burbujas de interactividad, suelen ser escasos y su uso debe ser optimizado.

La arquitectura que este informe propone para hacer realidad el modelo presentado en la Figura 4 tiene los siguientes objetivos de diseño:

1. Proveer un mecanismo de replicación que deje la información disponible en las otras burbujas de interactividad lo más rápido que sea posible, ya que determinados tipos de información pierden valor económico con el paso del tiempo.

2. Proveer una alternativa más económica a la de invertir en infraestructura local en los casos en los que un enlace con disrupciones ya existe (como, por ejemplo, en yacimientos de petróleo remotos).
3. Optimizar el uso del enlace que conecta dos burbujas de interactividad, ya que es un recurso escaso que debe ser aprovechado al máximo.

A pesar del gran logro que es poder comunicarse con personas y máquinas en lugares poco accesibles, no hay que olvidar que esto no cambia el hecho de que, en las grandes distancias, no es posible la comunicación interactiva instantánea y que, por el momento, no hay nada que hacer al respecto. Por eso, un cambio de mentalidad es necesario en cuanto a lo que es universal y lo que no. La colaboración interactiva e instantánea no es universal. La colaboración en general, por diferido, sí puede serlo, y es posible construir ese futuro. La arquitectura propuesta en este documento simplemente acompaña este cambio de mentalidad haciendo práctico lo que es posible, pero no cambia lo que es, con el conocimiento científico actual, imposible.

Lo que queda de este informe comienza con la sección “Antecedentes”, en la que se exploran los avances tecnológicos que establecieron las bases necesarias para el desarrollo de este proyecto.

En la sección “Descripción”, se explora en detalle la solución propuesta. En particular, se comienza por establecer el alcance abarcado por la solución; una vez establecido el alcance, se presentan los componentes arquitectónicos que componen la solución, y luego las interacciones planeadas entre ellos. Luego, se exponen como ejemplo dos posibles implementaciones de la solución, indicando en detalle la composición de éstas y las interacciones entre los componentes para esos casos particulares; el objetivo de esas implementaciones de ejemplo es el de familiarizar al lector con la arquitectura y proveer la intuición necesaria para lograr un entendimiento más profundo de las decisiones de diseño que se tomaron, más allá de la descripción estrictamente técnica.

Luego de la descripción, en la sección “Metodología de desarrollo”, se detallan las técnicas que se usaron durante la concepción del marco teórico, incluyendo una explicación de la importancia que tuvieron el desarrollo iterativo y la creación de una prueba de concepto para la mejora progresiva del resultado del proyecto.

En la sección “Pruebas Realizadas”, se presenta la prueba de concepto que fue desarrollada y desplegada para demostrar la viabilidad de la solución, y que realimentó el marco teórico, mejorándolo con lecciones aprendidas. Esta prueba de concepto cubre uno de los escenarios explicados en la sección de descripción, lo cual facilita su entendimiento. Además de presentarse la prueba de concepto, se brinda acceso al código fuente que la hizo posible.

En la sección “Discusión”, se indaga con más profundidad en las fortalezas y debilidades de la solución presentada, y, para algunas de las debilidades, se describen brevemente ideas alternativas de mitigación.

Finalmente, en la sección “Conclusiones”, se observan lecciones aprendidas y se transita una breve exploración de las posibles interpretaciones filosóficas del proyecto.

En “Anexo A: Componentes Originales” se explica la arquitectura original que se pensó al principio del proyecto, y se enumeran las debilidades que se encontraron para que el lector comprenda cómo esa arquitectura evolucionó a la que se terminó presentando en el informe escrito.

En “Anexo B: Declaración Inicial de Aporte Esperado” se provee una copia textual de la declaración del aporte esperado que forma parte de la propuesta de tema que dio origen a este esfuerzo. Se espera que su inclusión en este documento agregue contexto a los objetivos planteados para aquellos lectores que quieran tener una visión más amplia del aporte que se espera del proyecto.

Antecedentes

A principios de los años 2000 se comenzó a usar el acrónimo “DTN” (Redes tolerantes a retrasos y disrupciones, en inglés) para referirse a la forma de encarar el problema de retrasos y disrupciones en redes, proveyendo tecnologías y métodos de redes que permitan el envío confiable de mensajes en estas circunstancias. Hoy en día es un campo de estudio con su propia comunidad de investigadores, e incluso tiene su propio grupo de trabajo en la IETF. En (IETF, 2015) se enumeran una lista de casos de uso de las tecnologías emergentes de DTN, que se alinean con los problemas anteriormente planteados. Si el lector está interesado en conocer DTN sin contar con conocimiento previo específico sobre el tema, puede consultar (Warthman, 2015).

En este campo de estudio, inicialmente en la IRTF y luego en la IETF, se comenzó a trabajar, en la misma época, una serie de RFC. La RFC de central relevancia para el campo es la RFC 5050, publicada en 2007, luego de más de 4 años desde su borrador inicial, que especifica un protocolo denominado BP, cuyo principal propósito es proveer un método confiable de envío de datos a destinatarios que no pueden ser alcanzados mediante enlaces que no sufren retrasos ni interrupciones. En (Muri, et al., 2013) se provee una comparación entre el rendimiento de distintas configuraciones de BP y TCP/IP ante anomalías de red simuladas.

Más tarde durante esa misma década, NASA y otras agencias espaciales y gubernamentales del mundo comenzaron a realizar pruebas de esta y otras tecnologías relacionadas usando infraestructura espacial y terrestre preexistente y nueva (NASA, 2012). Las pruebas fueron exitosas, y demostraron formalmente que es posible comunicar a pesar de no tener los enlaces confiables de los que gozan los habitantes de las zonas urbanas.

Otro logro relevante hacia el final de esa misma década fue la estandarización, por parte de CCSDS (Consultative Committee for Space Data Systems), de un protocolo de transferencia de archivos con un enfoque de DTN. Dicho protocolo, llamado CFDP, cumple funciones similares a las de FTP, pero usando técnicas provenientes del campo de estudio de DTN. Tanto es así, que (CCSDS, 2007) menciona que el protocolo en cuestión tiene buen potencial al ser usado en conjunto con BP (mencionado más arriba). El estándar, originalmente de CCSDS, tiene su equivalente idéntico publicado por ISO bajo el número de norma 21323.

Tanto BP como CFDP hacen uso del concepto de "transferencia de custodia". La "custodia" es la responsabilidad de entregar un elemento a su destinatario. Dicha custodia puede ser transferida de un responsable a otro. El concepto de "transferencia de custodia", en el contexto de DTN, consiste en ir traspasando la custodia del mensaje, de forma de acercarla progresivamente al destinatario final del mensaje, con el objetivo de que, en caso de que una retransmisión sea necesaria, la misma tenga que recorrer la menor cantidad de saltos posible, en lugar de nacer desde el nodo de origen, como sería el caso en una conexión TCP (Warthman, 2015).

Otro protocolo para operar en DTN, también desarrollado por CCSDS, es AMS, apuntado a aplicaciones complejas distribuidas entre varios "continuos" ("burbujas de interactividad") que necesitan publicar y consumir mensajes relativamente cortos de manera organizada. La transferencia de un archivo, por ejemplo, puede ser lograda de una forma más elegante por CFDP, mientras que la difusión de eventos de aplicación, tales como cambios de estado o indicaciones a componentes, son escenarios más cercanos a los que AMS fue diseñado para cubrir. Si el lector está interesado en conocer más sobre AMS, puede consultar el estándar en (CCSDS, 2011) y un reporte informativo emitido por la misma agencia en (CCSDS, 2012).

A partir de la especificación del protocolo BP fueron naciendo implementaciones varias. En (Burleigh, et al., 2020) se mencionan las siguientes:

- uPCN
- pyDTN
- Terra
- dtn7-go
- dtn7-rs
- ION

Algunas de estas incluyen más que lo pedido por BP, implementando otros protocolos de DTN. Notablemente, ION, mantenida por NASA JPL/Caltech implementa también CFDP y AMS.

Con la popularización del uso de protocolos como CFDP en la industria espacial, se comenzaron a desarrollar herramientas para permitir a usuarios de otras áreas de conocimiento

la utilización de estas tecnologías, tales como la herramienta “Trek” de NASA Marshall Space Flight Center (NASA, 2020).

Descripción

En esta sección se explicará en detalle el marco teórico propuesto y la arquitectura resultante. Para ello, primero se define formalmente el alcance del problema que la solución propuesta abarca.

Alcance Abarcado

Los avances descriptos más atrás abren nuevas posibilidades; nuevas aplicaciones pueden ser desarrolladas, para los escenarios específicos de DTN, cuya naturaleza es, recordemos, no interactiva. Entre estos escenarios, pueden mencionarse los de:

- Comando remoto (por ejemplo, de robots o aeronaves). Hablamos, por supuesto, de comandos que no son críticos en cuanto a tiempo. Para una aeronave, las tareas pertinentes a mantener un vuelo recto y nivelado son críticas en cuanto a tiempo; mientras que la selección de un destino, en coordenadas, para el piloto automático, puede esperar, aunque sea unos minutos.
- Monitoreo remoto (por ejemplo, de robots autónomos o satélites).
- Consumo de contenido (detallado a continuación).

Este documento se centra en el consumo de contenido, y, en particular, en instancias específicas del mismo. Para poder expresar el tipo de escenarios alcanzados por la solución propuesta, primero es necesario analizar cómo está compuesto un escenario de consumo de contenido. En toda actividad de consumo de un contenido remoto, se pueden identificar claramente los siguientes elementos:

E1. El consumidor
E2. El contenido
E3. El origen del contenido
E4. El sistema de distribución y entrega del contenido

La arquitectura que se propone es aplicable a escenarios de consumo específicos; dichos escenarios deben cumplir con las siguientes restricciones:

R1. La relevancia del contenido es atemporal (es decir, no aumenta ni disminuye con el paso del tiempo) o no cambia de manera significativa en escalas de tiempo

del mismo orden de magnitud que los retrasos más graves del sistema de distribución.

R2. Existe un servicio de notificación de cambios, y un servicio de diferenciación de datos, que operan sobre el contenido en su origen (ambos servicios se explican en la sección “Nodos Emisores”).

R3. El sistema de distribución está soportado por una red DTN que hace uso del protocolo BP en su capa de “integración[3]”.

R4. El consumidor no puede acceder al contenido mediante un método interactivo inmediato.

Se debe tener en cuenta que no todas estas restricciones implican que el escenario no va a poder ser alcanzado por la solución. En este aspecto, las restricciones R2 y R3 son “salvables”, mientras que las restricciones R1 y R4 no lo son. Por ejemplo, cuando no se cumple R2, ambos servicios se pueden agregar o desarrollar para poder adaptar el escenario y de esa forma alcanzarlo con la solución aquí presentada. Asimismo, cuando no se cumple R3, se pueden hacer modificaciones o agregados a la red subyacente para que la restricción se cumpla. Por supuesto, esto implicaría gastos adicionales.

Además, se impone como limitación explícita sobre el alcance de la solución, que:

L1. El contenido podrá ser modificado únicamente en su origen: el consumidor remoto no podrá modificar el contenido; sus accesos al mismo serán de solo lectura.

La siguiente es una lista de ejemplos no exhaustiva del tipo de contenido que cumple con la restricción R1, asumiendo retrasos máximos de un mes:

- Documentación corporativa
- Documentos académicos
- Estándares
- Manuales de maquinaria
- Entradas de una enciclopedia
- Clases filmadas en video

[3] En las referencias bibliográficas en inglés, a esta capa se la conoce como “Bundling Layer”.

- Material de entretenimiento (películas, libros, etc.)

Solución

A la solución aquí presentada se le otorga el nombre de DCSR (DTN Content Set Replication; inglés para Replicación de Conjuntos de Contenido en DTN).

En esta sección se exploran en detalle la composición y el funcionamiento de la arquitectura que se presenta para los escenarios abarcados por el alcance.

Composición

En esta sección se presenta cada uno de los elementos que intervienen en la dinámica de la arquitectura con una breve explicación de cada uno y una figura, que será la misma que se usará durante el resto del documento para referirse a cada uno de los conceptos presentados.

Burbujas de interactividad

Ya anticipadas en la descripción, las burbujas de interactividad (Figura 5) son conjuntos de actores, ya sean humanos o máquinas, que pueden interactuar entre sí en vivo. Las burbujas de interactividad no son realmente un componente de la arquitectura propuesta, sino una realidad impuesta por el entorno ya sea debido a las distancias involucradas o a la infraestructura local de red disponible.

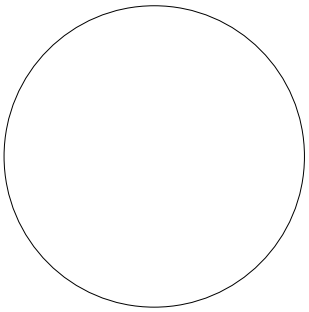

Figura 5: Burbuja de interactividad

Contenidos

Un contenido (Figura 6) en el contexto de este documento, es un elemento identificable, que un consumidor puede consumir. El concepto de contenido aquí presentado se alinea directamente con el elemento E2 presentado en la subsección “Alcance Abarcado”. Un artículo

de Wikipedia, por ejemplo, puede ser considerado un contenido en este contexto, así como puede serlo una película de Netflix o un documento disponible en una intranet corporativa.

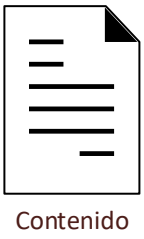

Figura 6: Contenido DCSR

Nodos

Un nodo (Figura 7), en esta arquitectura, es un nodo de una red DTN, utilizando protocolo BP, que cumple por lo menos una de las dos siguientes funciones:

NF1. Propiedad de conjunto(s) de contenidos.

NF2. Replicación de conjunto(s) de contenidos.

A aquellos nodos que cumplan la función NF1, vamos a llamarlos “Nodos Emisores”, y a aquellos que cumplan la función NF2, vamos a llamarlos “Nodos Receptores”.

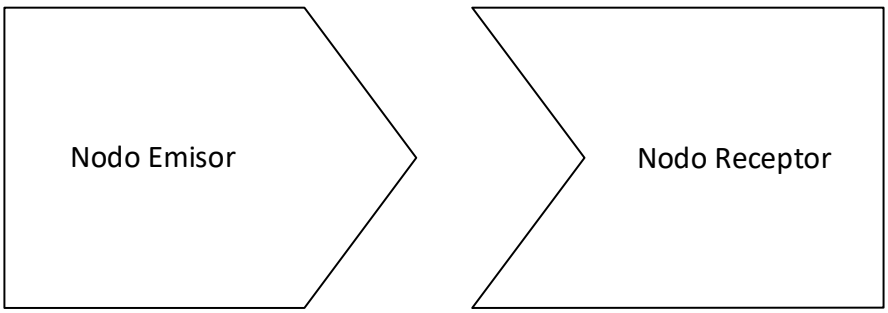

Figura 7: Nodos DCSR: a la izquierda, un nodo emisor; a la derecha, un nodo receptor

No todos los nodos tienen que cumplir ambas funciones. Sin embargo, ambas funciones pueden ser cumplidas por un mismo nodo, en cuyo caso estaríamos en presencia de un nodo emisor-receptor.

Esta separación ofrece dos ventajas principales:

1. En caso de que realmente solo se desee consumir contenido desde una determinada locación, sin producirlo, o viceversa, se puede tener sólo los subcomponentes necesarios

para la función elegida, en lugar de contar con todos los subcomponentes utilizando sólo un subconjunto de ellos.

2. Incluso cuando se deseen ambas funciones para un mismo nodo, esta separación ofrece a los dueños del nodo elegir libremente implementaciones de autores distintos para cada una de las funciones, si así lo desearan por cualquier motivo.

Este concepto de nodo está alineado con el concepto de burbuja de interactividad explicado más atrás. En el contexto de DCSR, cada nodo vive en una burbuja de interactividad. Esto no impide que dentro de una misma burbuja se instale más de un nodo, lo cual a veces puede ser conveniente, como veremos en la sección “Discusión”.

Conjuntos de Contenidos

Todo conjunto finito de contenidos (tal como se definen más atrás) pertenecientes al mismo nodo puede ser considerado un conjunto de contenidos (Figura 8) en el contexto de este documento. El conjunto de artículos de Wikipedia en español, por ejemplo, podría ser un conjunto de contenidos, así como podría serlo el conjunto de archivos que se encuentran en los servidores de Wikipedia, que hacen posible el funcionamiento del sitio. También pueden considerarse conjuntos de contenidos los conjuntos de películas de plataformas de video como Netflix, conjunto de documentos disponibles en una intranet corporativa, o hasta el conjunto de archivos binarios que componen una aplicación que corre en un servidor de aplicaciones.

Conjunto de Contenidos

Figura 8: Conjunto de Contenidos DCSR

Subconjuntos de Contenidos

Dentro de los conjuntos de contenidos, existe la posibilidad de agrupar contenidos relacionados, de forma jerárquica. Un subconjunto de contenidos (Figura 9) es a los contenidos lo que una carpeta es a los archivos.

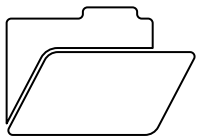

Figura 9: Subconjunto de Contenidos DCSR

Orígenes

El origen se define a partir del contenido. El origen de un contenido es el nodo dueño del conjunto de contenidos al cual el contenido en cuestión pertenece. Un contenido puede estar replicado en muchos nodos, junto con el resto del conjunto de contenidos al que pertenece, pero, a pesar de estar replicado, su origen es siempre el mismo.

Nodos Emisores

Esta sección muestra la arquitectura propuesta para todos aquellos nodos que deban cumplir la función NF1 (emisores). Se requiere, de los nodos emisores, que provean un "Servicio de Replicación", el cual será consumido por nodos receptores.

En la Figura 10 se pueden ver los componentes necesarios para cumplir la función NF1. La tarea del notificador de cambios es la de avisar al replicador emisor cada vez que un cambio ocurre en el el conjunto de contenido siendo observado. Esto incluye las siguientes operaciones:

- Adición de un contenido nuevo
- Cambios a algún contenido existente
- Cambios a los metadatos de algún contenido
- Eliminación de un contenido
- Adición de un subconjunto de contenidos
- Cambios a los metadatos de algún subconjunto de contenidos existente
- Eliminación de un subconjunto de contenidos

Nótese que estas operaciones son equivalentes directos a los casos de uso exhibidos en la Figura 12 más adelante.

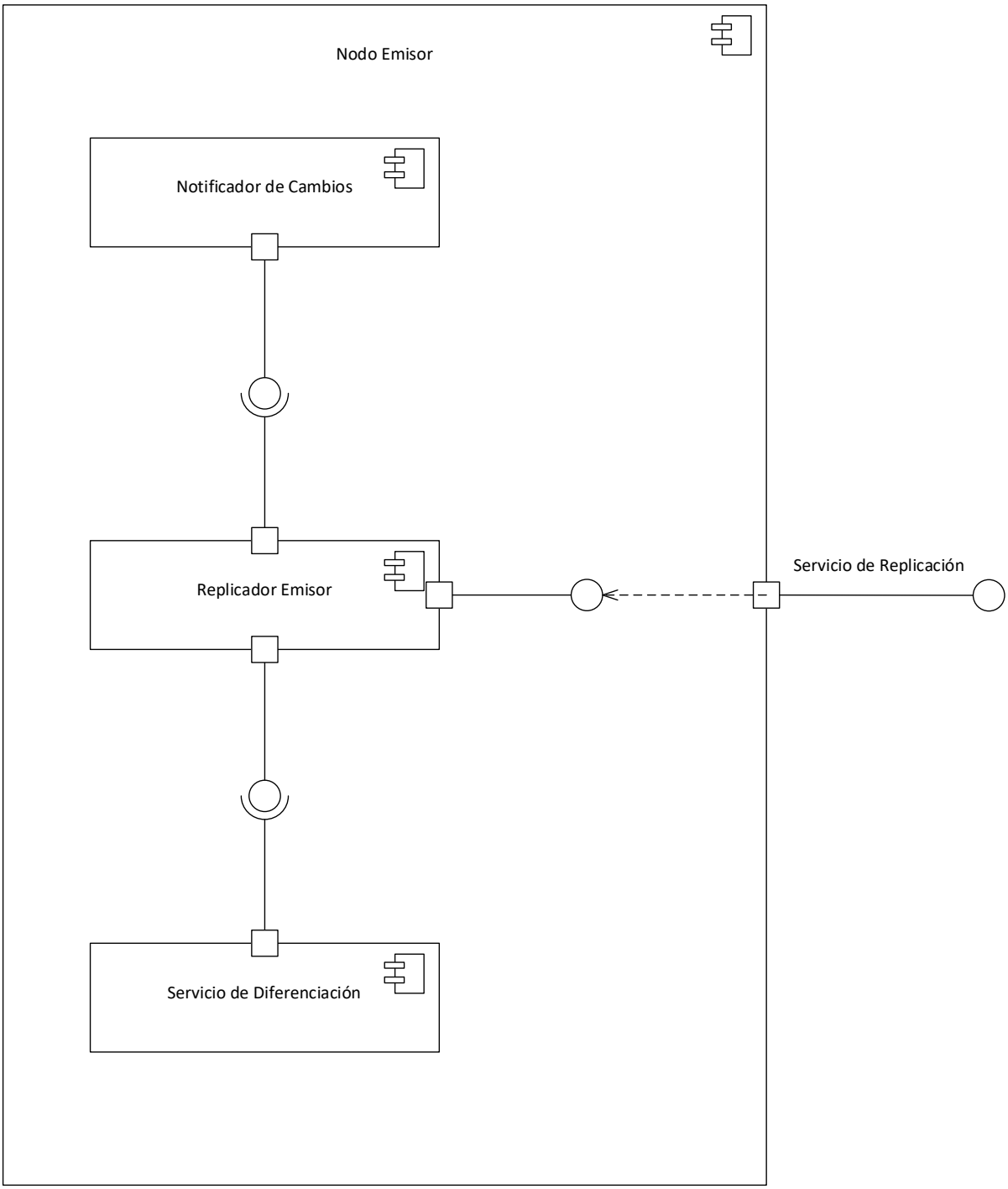

Figura 10: Componentes necesarios para poder cumplir con la función NF1

El servicio de diferenciación le indica al replicador emisor (cuando éste se lo solicite) cuáles fueron los cambios que ocurrieron desde la última vez que se lo consultó. Esto se logra mediante identificadores de cambios. Cada vez que el servicio de diferenciación es consultado, éste entrega un identificador de cambio que identifica el estado actual del conjunto de

contenidos; luego, un tiempo más tarde, en una llamada subsiguiente, el cliente le entrega al servicio, como parte de la solicitud, ese mismo identificador, el cual el servicio utiliza para saber cuáles son los cambios que debe devolverle al invocador. Esta definición está basada directamente en el mecanismo que Microsoft Graph[4] usa para reportar cambios incrementales a los recursos a los que dicha API brinda acceso (Microsoft, 2017).

El replicador emisor es una aplicación que es invocada por el notificador de cambios cada vez que se produce una modificación del conjunto de contenidos, sea cual sea. El replicador responde a este estímulo consultando al servicio de diferenciación para obtener los cambios ocurridos desde la invocación anterior; para lograr este propósito, el replicador provee un identificador de cambio (si es que ya tiene uno) lo cual pone en funcionamiento el mecanismo explicado en la sección anterior.

El principal trabajo del replicador es interpretar los cambios obtenidos, traducir el conjunto de cambios a un conjunto de operaciones que serán indicadas a través del servicio de replicación, y enviar representaciones de estas operaciones, junto con los contenidos que haya que agregar o modificar, a los nodos receptores. Nótese que el replicador tiene la libertad de elegir no propagar algunos de los cambios locales, o darle prioridad a cambios específicos en caso de que el negocio los priorice. Este concepto se explica con más detalle en la subsección “Extensibilidad”.

Más detalles sobre la comunicación resultante pueden encontrarse en la subsección “Comunicación entre Nodos”.

Nodos Receptores

Esta subsección muestra la arquitectura propuesta para todos aquellos nodos que deban cumplir la función NF2 (receptores). Los nodos receptores cumplen esta función consumiendo servicios de replicación ofrecidos por nodos emisores.

En la Figura 11, podemos ver los componentes necesarios para cumplir la función NF2.

[4] “Graph”, “MS Graph” y “Microsoft Graph” son todas maneras de referirse al mismo producto y se usan de forma intercambiable a lo largo del informe.

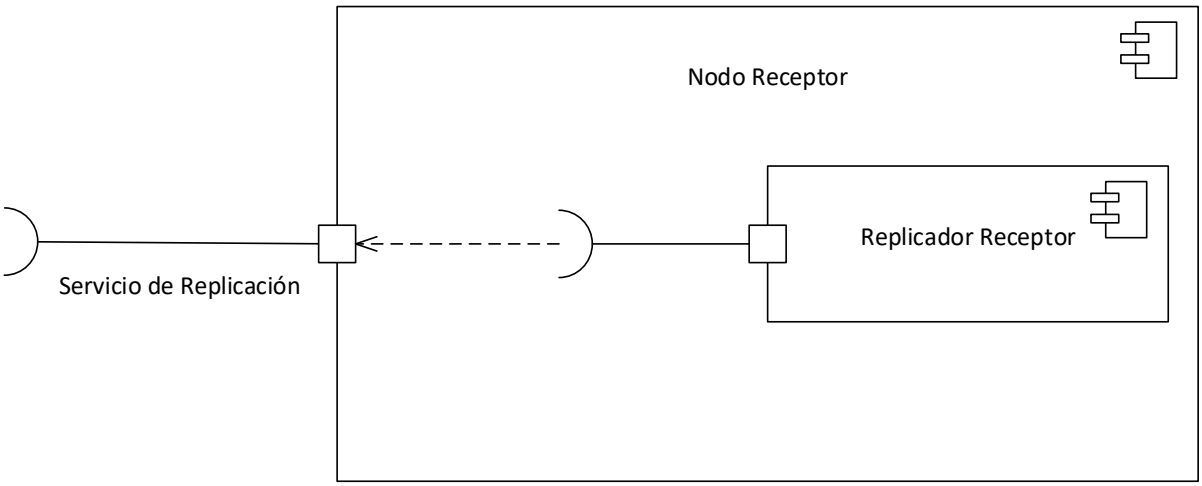

Figura 11: Componentes necesarios para poder cumplir con la función NF2

Como se mostrará en la sección "Pruebas Realizadas", con las tecnologías existentes, hay casos en los que no haría falta escribir código adicional para construir un nodo receptor: en la mayoría de los casos sólo se requerirá instalar paquetes existentes y configurarlos apropiadamente.

Comunicación entre Nodos

Dado que los nodos emisores (NF1) ofrecen una interfaz de servicio de replicación, y que los nodos receptores (NF2) consumen dichas interfaces, la comunicación siempre será entre nodos de tipos distintos, ya que dos nodos del mismo tipo no pueden ser interconectados mediante ninguna interfaz.[5]

La Figura 12 muestra los casos de uso que se deben poder propagar a través del servicio de replicación. Dependiendo del protocolo de capa de aplicación utilizado para concretar esta comunicación, una mayor o menor cantidad de casos de uso deberán ser implementados con código de usuario. El protocolo CFDP, mencionado en la sección Antecedentes, ofrece primitivas que cubren todos los casos de uso, de manera que, de utilizarlo en la implementación, no es necesario escribir código adicional para desplegar en los nodos NF2, ya que la mera presencia de un servicio CFDP que se encuentre activo cumple todas las funciones necesarias (excepto, por supuesto, que la implementación de CFDP que se esté utilizando no adhiera

[5] Es por este motivo que, si se desea lograr una replicación bidireccional entre dos locaciones distintas, es necesario instalar ambos nodos en cada una de ellas. Esto no es recomendado sobre un mismo conjunto de contenidos ya que violaría la limitación L1 y puede ocasionar problemas que se exploran en la sección "Discusión".

completamente al estándar internacional que especifica el protocolo). En estos casos, se considera que la instancia de CFDP activa en el nodo receptor es el replicador receptor.

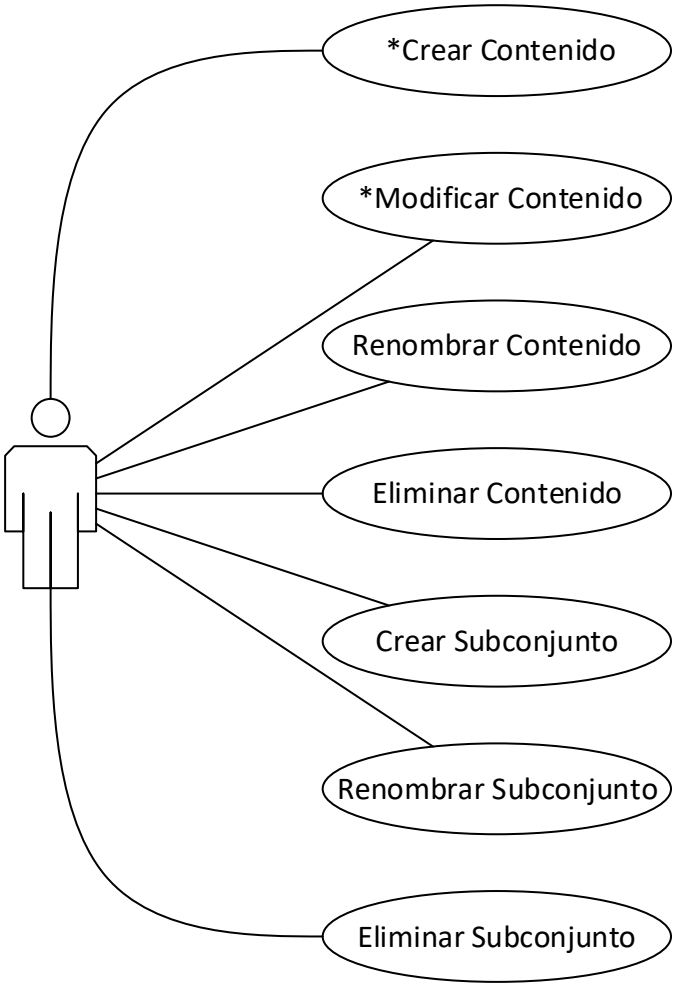

Figura 12: Casos de uso del Servicio de Replicación. Los asteriscos indican operaciones que requieren transferencia de contenidos.

Explorando más en detalle la comunicación necesaria entre un nodo replicador emisor y un nodo replicador receptor, encontraremos que hay dos tipos distintos de datos que deberían ser intercambiados entre ambos:

TD1. Metadatos que representan las diferencias incrementales de los cambios que le van ocurriendo al conjunto de contenidos en su origen (estos tipos de datos son mencionados como “deltas” en algunas referencias bibliográficas).

TD2. Contenidos propiamente dichos, en los casos en los que se crea un contenido nuevo o se modifica uno existente, en el origen.

Esta distinción es importante ya que, por tener una naturaleza diferente, el implementador podría decidir utilizar protocolos de comunicación diferentes para cada uno de los tipos de datos mencionados.

Extensibilidad

Se propone que el replicador emisor conceda la posibilidad de instalar "filtros", con código de usuario, que puedan intervenir en las acciones que se llevan a cabo para cada cambio que el servicio de diferenciación entrega. Algunos ejemplos de este tipo de acciones son:

- Prevenir la replicación del cambio (ver la subsección "Replicación Selectiva" en la sección "Discusión").
- Agregar prefijos o sufijos a los nombres de los contenidos replicados.
- Escribir en un log la operación de replicación que se está ejecutando.

Esta lista no es exhaustiva; está para dar una idea de la utilidad de proveer el punto de extensibilidad en las implementaciones.

Escenarios de Ejemplo

Se presentan en esta sección dos escenarios de ejemplo para ayudar al lector a comprender el marco teórico, llevándolo a un plano práctico.

Monitoreo de Yacimiento de Petróleo

Una empresa petrolera cuenta con un campo de trabajo en la zona de Vaca Muerta, y desea comenzar a monitorear los sitios de extracción mediante captura de imágenes, para mejorar el tiempo de reacción ante incidentes y el diagnóstico de problemas, que actualmente se basa únicamente en telemetría no multimedial. El tipo de acceso a internet con el que se cuenta en estos sitios es mediante enlaces de radio o satelitales. Ambos enlaces son susceptibles a condiciones del entorno, notablemente, el clima. Este tipo de enlace es suficiente para la telemetría no multimedial que se transmite hoy en día, pero no va a ser suficiente para el envío de imágenes de la maquinaria de extracción.

El equipo táctico a cargo del proyecto tiene la opción de presentar a la alta gerencia una opción de presupuesto que incluya la instalación de accesos cableados a cada una de las máquinas a ser monitoreadas en los sitios de extracción. Sin embargo, esta opción resulta ser

muy costosa y con DCSR se podrían aprovechar los enlaces ya existentes si se adopta una arquitectura como la mostrada en la Figura 13.

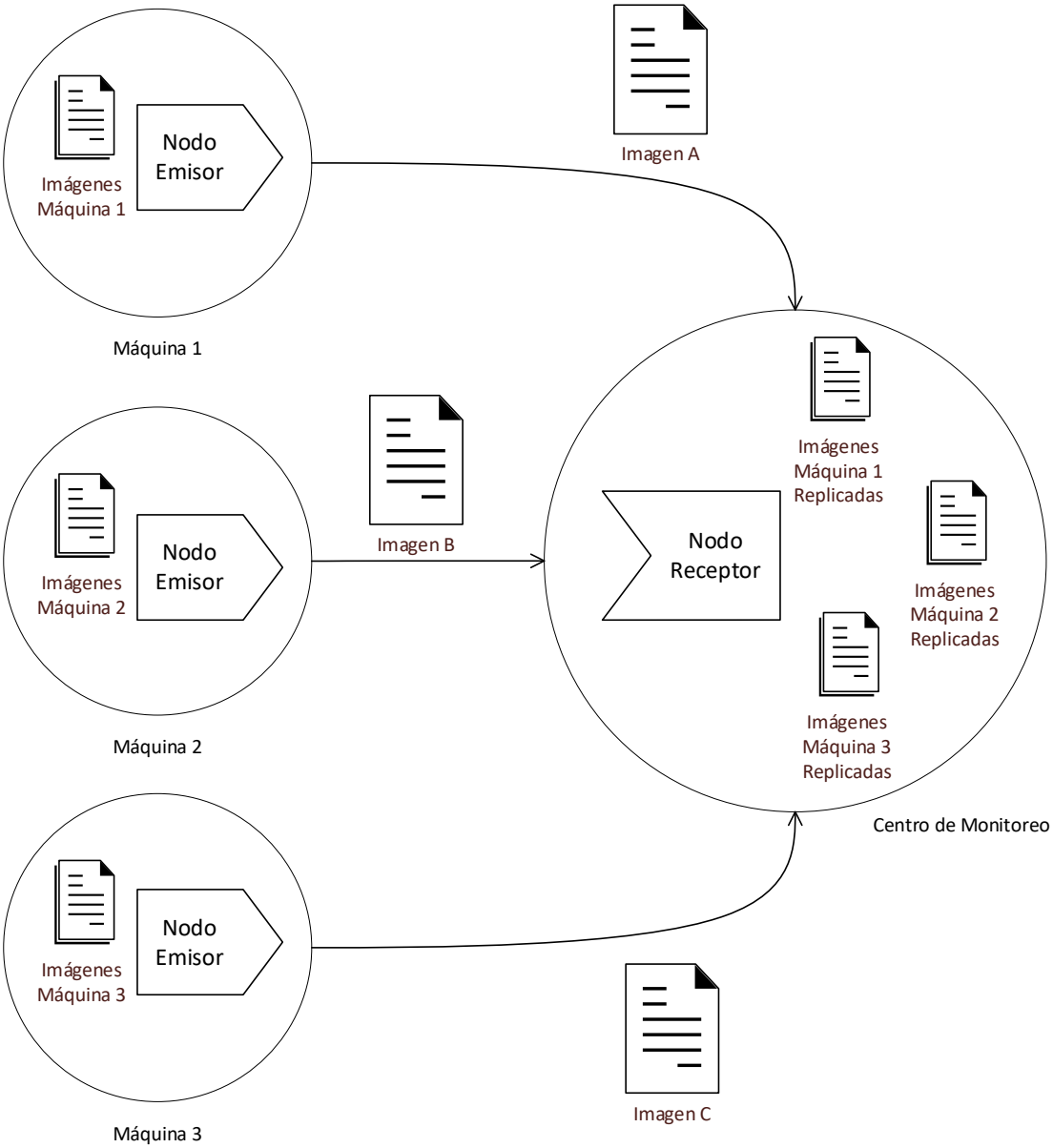

Figura 13: Solución al escenario presentado en DCSR

Como puede apreciarse, en este caso varios nodos emisores se conectan a un mismo nodo receptor, el cual tendrá los conjuntos de contenidos replicados de cada una de las máquinas de extracción. Si se decidiese usar protocolos DTN para aprovechar los enlaces existentes, pero sin una solución como DCSR harían falta empleados de tiempo completo que emitan solicitudes

a las máquinas para obtener imágenes de forma constante, ya que los protocolos por sí solos no hacen envíos automáticos de archivos.

El resultado de utilizar DCSR es la obtención de las imágenes más recientes lo antes posible, lo cual cumple el objetivo de agilizar la reacción ante incidentes y el diagnóstico mediante medios visuales, al mismo tiempo evitando una costosa instalación de enlaces cableados y los costos de empleo de personal dedicado a la solicitud manual de imágenes mediante herramientas de líneas de comando o de interfaz gráfica.

Misión a Marte

En este caso, vamos a imaginar una misión de exploración en Marte, similar a las que se llevaron a cabo en la Luna a principio de los años ’70, con la diferencia de que, en este escenario, la comunicación interactiva con el centro de control de la misión no es posible (es decir, hay dos burbujas de interactividad separadas), y la estadía es prolongada en el tiempo, sin retorno inmediato.

En la Tierra, un comité elige cómo distribuir el tiempo de misión en tareas científicas, y produce itinerarios diarios para indicar a los científicos en Marte qué tareas se espera que lleven a cabo un determinado día. Por lo general, producen estos itinerarios con una semana de antelación. Los itinerarios son simples documentos digitales, en formato DOCX, que el comité almacena en una nube privada en Washington DC. Además de estos documentos, hay miles de otros que científicos en la Tierra mantienen y actualizan para detallar en profundidad y complementar los requisitos listados en los itinerarios.

En Marte, un grupo de científicos colonizadores debe leer el itinerario del día y los integrantes llevar a cabo las tareas indicadas en el mismo, asignándolas entre ellos de acuerdo con las habilidades de cada uno. Pueden usar los documentos complementarios, si así lo desean, en casos que consideren necesarios. Sin DCSR, esto podría concretarse de varias maneras:

- Tener uno o más empleados dedicados, en Washington DC, a utilizar una herramienta (ya sea de línea de comandos o de interfaz gráfica) para transmitir los archivos que considera que van a necesitar en Marte. Si la gente en Marte necesita documentos adicionales, estos deben enviar un mensaje que será recibido con retraso en la Tierra, y entonces el personal dedicado utilizaría la misma herramienta para enviar los archivos solicitados.

- Instalar un servidor en Washington DC que reciba solicitudes de archivos y envíe a Marte archivos solicitados desde allí, con el retraso de la ida y la vuelta.

Con DCSR y su replicación proactiva, en Marte pueden acceder a cualquier contenido del conjunto de forma inmediata, sin necesidad de emplear a nadie en Washington DC. La Figura 14 muestra cómo la arquitectura DCSR modela la solución a este escenario.

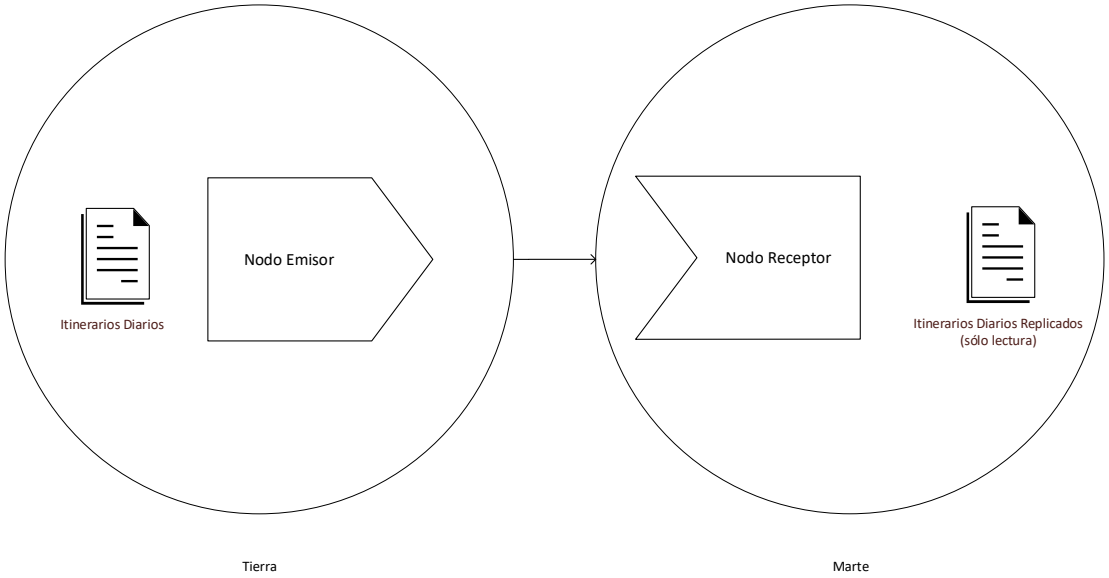

Figura 14: Solución al escenario presentado en DCSR

Nótese, en particular, que en ningún momento hay una comunicación interactiva. La colaboración es eficiente y funcional, pero es totalmente en diferido.

Metodología de desarrollo

Desde la idea exhibida originalmente en la propuesta de tema formal de este proyecto, hasta la arquitectura final presentada en este informe, hubo muchos grandes cambios y ajustes menores. Se utilizó una metodología iterativa para conciliar, progresivamente, la idea original con un producto fácilmente implementable pero que a la vez cumpliera con las expectativas y exigencias de calidad que caracterizan a un proyecto final.

En el Anexo A: Componentes Originales se describe brevemente la arquitectura tal como fue concebida en el inicio del proyecto, y las debilidades que se fueron encontrando en ese modelo original, para que si el lector así lo desea pueda entender el camino de razonamiento que condujo a la arquitectura final presentada.

Como parte central del desarrollo iterativo, tanto el informe escrito como la prueba de concepto se fueron enriqueciendo mutuamente de forma gradual. El autor fue alternando entre una tarea y la otra de manera regular, mejorando el informe escrito mediante lecciones aprendidas durante el desarrollo y la depuración de la prueba de concepto, e implementando en la prueba de concepto mejoras que fueron originalmente concebidas en el informe escrito y en diagramas UML y otras herramientas de diseño visual.

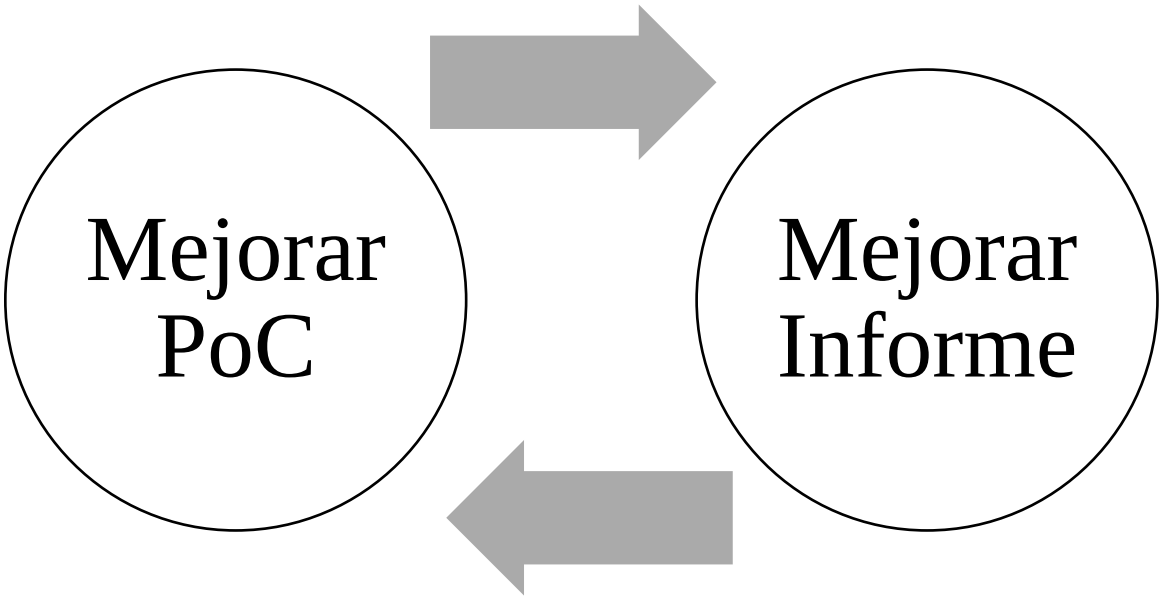

Figura 15: Desarrollo iterativo con apoyo de una prueba de concepto

A pesar de que cualquier prueba de concepto es, en el fondo, por más sencilla que sea, una implementación, al dejar que la misma influya en la especificación de una arquitectura genérica se corre el riesgo de incluir detalles de implementación en la descripción de esta. Es difícil separar por completo ambas cosas, pero se observó cuidadosamente la necesidad de no contaminar la descripción de la arquitectura con estos detalles (excepto en notas a los implementadores para guiarlos en la toma de decisiones de alto nivel). Por eso se observará que la sección "Descripción" es de naturaleza mucho más genérica que la sección "Pruebas Realizadas", la cual escruta cuidadosamente a nivel de clases, funciones y atributos los elementos de la prueba de concepto implementada. Varios de los párrafos que pueden encontrarse hoy en estas dos secciones fueron cambiados de lugar varias veces hasta que logró tomarse una decisión sobre si ser incluidos en la descripción como parte de la especificación formal de la arquitectura, o si ser mostrados como detalles particulares de la prueba de concepto que otros implementadores pueden ejecutar de manera diferente.

El objetivo de este proyecto es presentar una arquitectura funcional para la replicación proactiva de conjuntos de contenidos en redes DTN; y el objetivo de la prueba de concepto es el de demostrar que la arquitectura presentada es viable y práctica. Dado que no se espera de la prueba de concepto una calidad ni una performance productiva, las premisas que guiaron la elección de tecnologías fueron la facilidad de uso y la rapidez de configuración. Siendo que en UADE los alumnos gozan de acceso a una serie de herramientas, estas se aprovecharon para el desarrollo de la prueba de concepto siempre que fue posible. Entre estas se destacan los productos de la suite Microsoft 365, en particular Office 365, Azure, Microsoft Graph y Windows 10 Education. Más detalles sobre cómo estos productos fueron utilizados para la prueba de concepto pueden encontrarse en la sección "Pruebas Realizadas".

En cuanto a la elección de la implementación del protocolo BP (que es una dependencia directa de la arquitectura, como fue indicado en la restricción R3, explicada en la subsección Alcance Abarcado de la sección Descripción), se eligió ION, de NASA JPL/Caltech, ya que:

- Es de código abierto (lo cual posibilita su depuración, de ser necesario)
- Tiene una comunidad activa hasta la fecha (lo cual ayudó en cuestiones de soporte)

- Contiene, además de BP, una implementación de CFDP, lo cual, como se explica en varias ocasiones en este documento, simplifica enormemente la implementación.
- Fue probada y está en uso productivo en misiones espaciales.
- Tiene APIs accesibles desde C/C++, lenguajes soportados por Visual Studio, que fue el IDE utilizado para el resto de la solución (lo cual facilitó el desarrollo al necesitar de un único entorno).

Por último, se entrevistaron y solicitaron constantemente revisiones de este informe a profesionales que trabajan en las industrias relacionadas para representar de manera fiel los puntos de dolor que este proyecto ayuda a solucionar en el corto plazo. Las personas consultadas están listadas en la sección “Agradecimientos”.

Pruebas Realizadas

El producto esperado de este proyecto es el modelo teórico que se presentó en la descripción, por lo que la manera en la que se probó el modelo fue desarrollando una implementación para un caso de ejemplo, que sirve de prueba de concepto. Todo el código fuente de la prueba de concepto está disponible en https://github.com/nprieto95/dcsr.git.

El diagrama de componentes de la Figura 16 brinda una visión global de la estructura arquitectónica de la implementación de la prueba de concepto. Nótese que esta composición en particular no es un requisito, sino el resultado de una serie de decisiones que se tomaron para esta implementación. Siempre y cuando se cumplan las funciones explicadas en la sección de Descripción, la elección de los componentes para cumplir las funciones es libre.

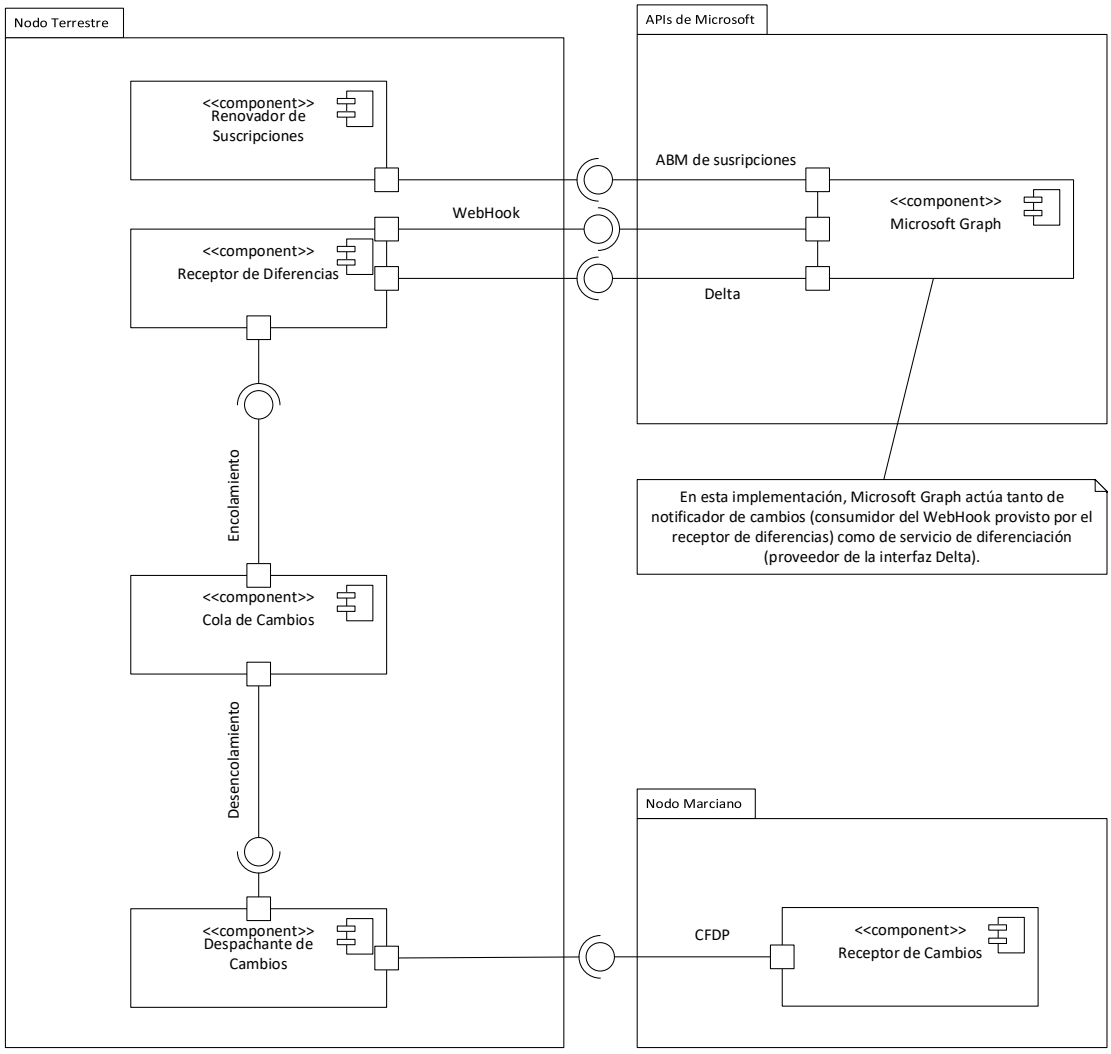

Figura 16: Estructura arquitectónica de la implementación de la prueba de concepto.

Microsoft Graph provee dos servicios que se alinean con dos de las funciones que deben cumplirse en un nodo NF1: la notificación de cambios y un servicio de diferenciación.

La notificación de cambios está implementada mediante un modelo de suscripción, en el que los suscriptores manifiestan interés en un determinado tópico enviando a Graph un HTTP POST, y, a su vez, Graph envía un HTTP POST por cada novedad a cada suscriptor interesado en el tópico. Si el lector desea conocer más en detalle estos mecanismos, puede consultar (Microsoft, 2020).

Debido a que las suscripciones tienen una duración máxima de 72 horas (límite impuesto por el proveedor del servicio), las mismas deben ser renovadas periódicamente para ratificar que el suscriptor sigue interesado en recibir novedades sobre el tópico en cuestión. Por este motivo, se incluyó en esta implementación un componente llamado “Renovador de Suscripciones”, cuyo motivo de ser es el de renovar las suscripciones para que el receptor de diferencias (explicado más abajo) continúe recibiendo notificaciones indefinidamente. El renovador de suscripciones está implementado como una Azure Function que se ejecuta cada 36 horas. Si el lector desea familiarizarse con el servicio mencionado, puede consultar (Microsoft, 2021).

El servicio de diferenciación ofrece, por cada recurso disponible, un punto de acceso que la aplicación interesada puede invocar mediante un HTTP GET, ofreciendo, opcionalmente, un identificador de cambio. Si el identificador de cambio no se incluye, Graph responde al HTTP GET con un objeto expresado en JSON que representa el recurso entero, junto con un identificador de cambio que la aplicación puede usar en solicitudes subsiguientes. Si se incluye un identificador de cambio en la solicitud, Graph devuelve un objeto expresado en JSON que representa sólo aquellas partes del recurso que fueron modificadas desde el cambio identificado por el solicitante. Microsoft Graph implementa este patrón sobre sendos recursos a los que ofrece acceso; las unidades de almacenamiento de OneDrive están entre estos. Si el lector desea conocer más en detalle estos mecanismos, en particular para las unidades de almacenamiento de OneDrive, puede consultar (Microsoft, 2017).

El receptor de diferencias es quien recibe las notificaciones de cambios que Graph envía, y, ante dicho estímulo, consulta cuáles son las diferencias que se produjeron en la unidad de almacenamiento de OneDrive desde la última vez que se produjo un cambio; logra esto

almacenando siempre el último identificador de cambio en memoria de aplicación (lo cual significa que, si por cualquier motivo el receptor de diferencias se reinicia, la primera vez que llegue un cambio se concretará una sincronización completa). Una vez obtenidas las diferencias, las mismas se añaden a una cola de diferencias pendientes para propagar a los otros nodos. La misma está implementada como cola de Azure Storage. Si el lector desea familiarizarse con el servicio mencionado, puede consultar (Microsoft, 2019). Para consolidar lo explicado en estos últimos párrafos, la Figura 17 muestra la secuencia de interacciones que ocurren entre Microsoft Graph, el receptor de diferencias, y la cola de cambios, cada vez que Graph envía una notificación.

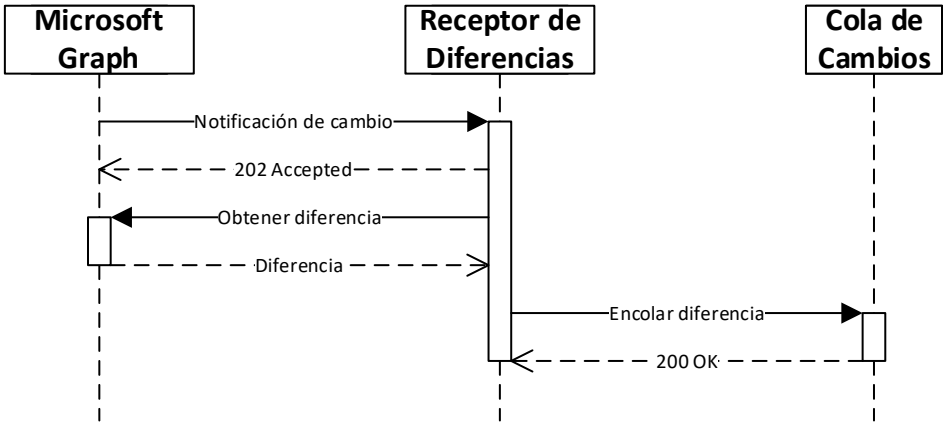

Figura 17: Diagrama de secuencia de la recepción de diferencias.

El despachante de cambios es una aplicación de línea de comandos, escrita en C++, que sondea periódicamente la cola de cambios. Se decidió usar C++ para el despachante de cambios ya que la implementación de CFDP utilizada provee interfaces accesibles desde C/C++, haciendo de C++ el lenguaje de más alto nivel capaz de acceder a estas interfaces de forma directa. Cada vez que el despachante encuentra uno o más cambios en la cola, los descarga en formato JSON y crea instancias de clases que corresponden a cada uno de los cambios. Las clases que representan estos cambios heredan de una clase base, abstracta, denominada “Cambio”. La misma define un método abstracto llamado “Aplicar”, cuyas implementaciones deben ser capaces de transmitir, mediante CFDP, el cambio a los nodos NF2 listados en la configuración de la aplicación. Una vez creados los objetos que representan los cambios encontrados en la cola, se invoca el método Aplicar en cada uno de ellos, de esta forma concretando el inicio de la operación asincrónica de la replicación, mediante CFDP, de los

cambios que ocurrieron en el nodo local. Estas interacciones se exhiben en el diagrama de secuencia mostrado en la Figura 18.

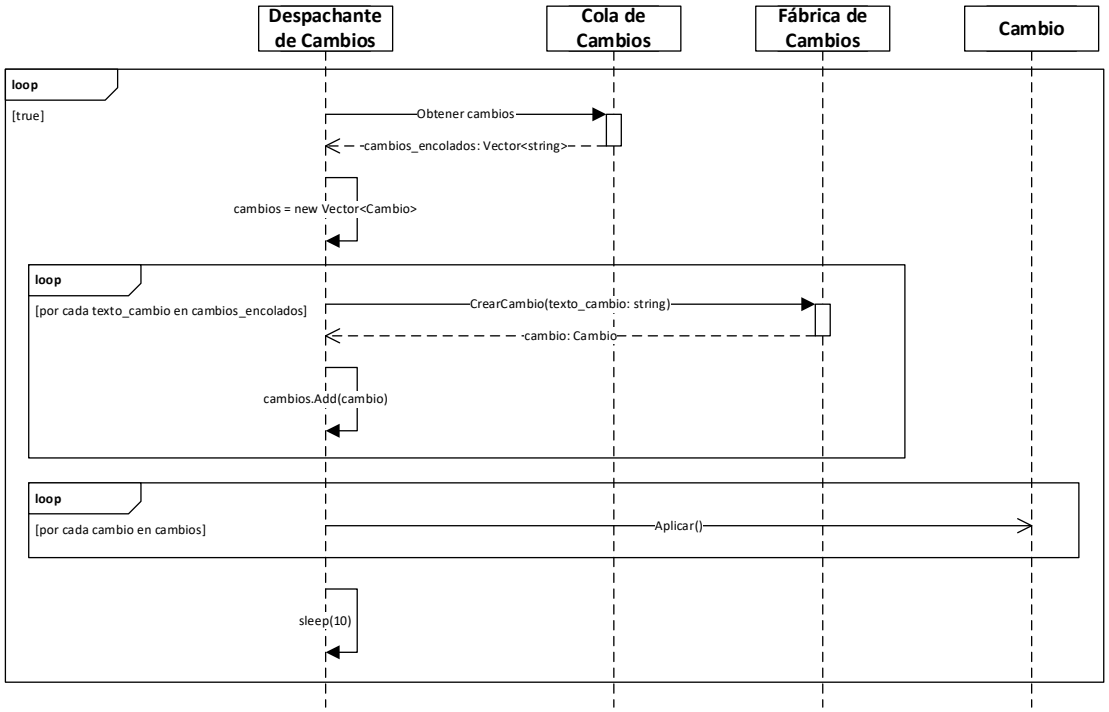

Figura 18: Diagrama de secuencia del despacho de cambios

Para representar los cambios, se diseñó la jerarquía de clases mostrada en la Figura 19. La fábrica de cambios es una fábrica simple según se define en (Freeman, et al., 2004), cuyo objetivo es abstraer la creación de los cambios concretos del flujo principal de la aplicación. Dicha fábrica recibe en formato JSON la representación del cambio, y devuelve una instancia de Cambio. La clase concreta que es devuelta depende del tipo de cambio representado por el texto pasado como parámetro al método creador.

Se identificaron 7 operaciones de cambio posibles. Las operaciones identificadas son las siguientes:

OR1. Archivo creado
OR2. Archivo actualizado
OR3. Archivo renombrado
OR4. Archivo eliminado
OR5. Directorio creado
OR6. Directorio renombrado

OR7. Directorio eliminado

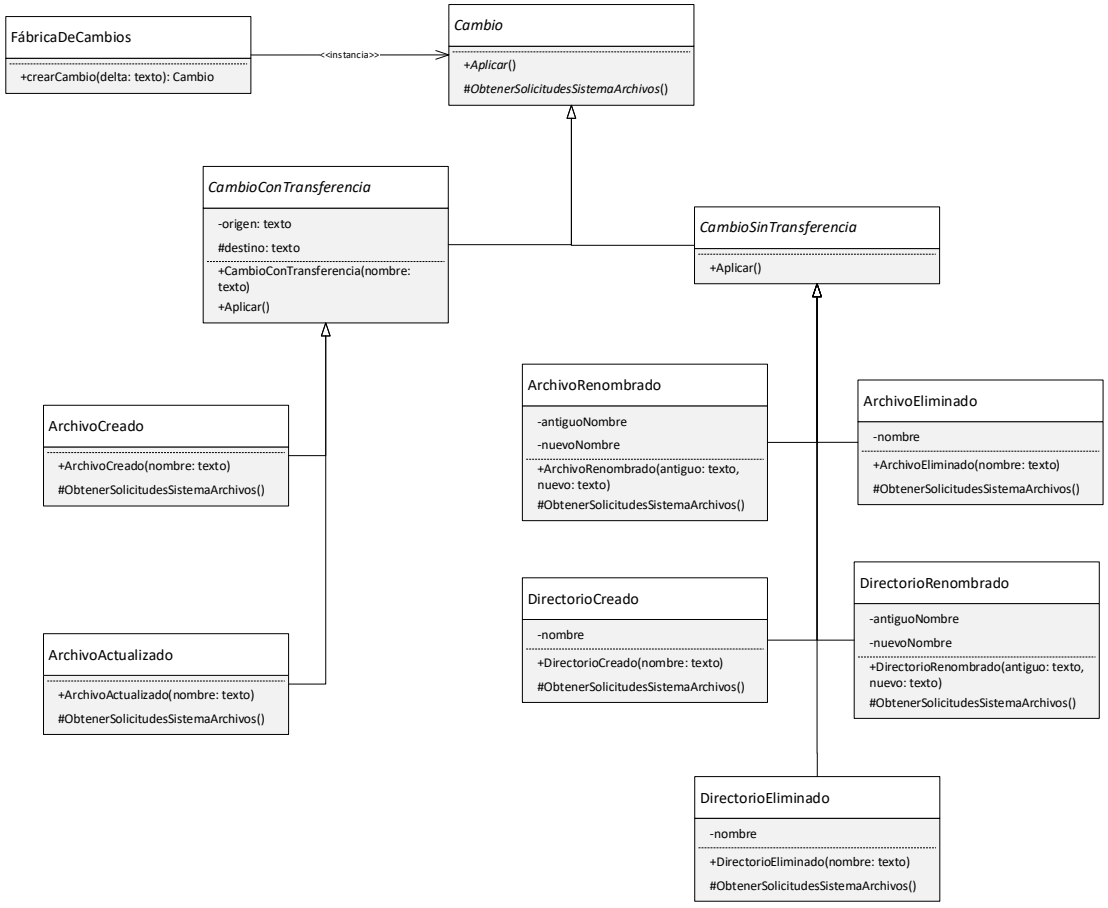

Figura 19: Jerarquía de clases del Despachante de Cambios

Cada una de ellas tiene distintas necesidades de código para poder concretar la replicación de los cambios, ya que se necesitan invocar diferentes funcionalidades de CFDP para poder cumplir con ella. Sin embargo, en base a la estructura general de los procedimientos requeridos, podemos agrupar las operaciones mencionadas en dos grupos: el grupo de aquellas que requieren la transferencia de los contenidos de un archivo, y el grupo de aquellas que no la requieren. Es por este motivo que la jerarquía de clases se dividió en dos ramas representando a cada uno de estos grupos. Cada una de ellas está encabezada por una clase base que hereda de la clase “Cambio”, y define polimórficamente su propia implementación del método “Aplicar”. Dentro de cada grupo, se requieren sólo pequeñas variaciones para poder concretar las operaciones de replicación requeridas, por lo que se hizo uso del patrón de diseño “template method”, tal como se expresa en (Gamma, et al., 1994), para preservar la estructura general del algoritmo y delegar los detalles cambiantes a las clases hijas.

En esta implementación, cada replicador (tanto emisor como receptor) es un sistema Linux (específicamente la distribución Ubuntu). Para facilitar la administración y configuración de los sistemas, cada uno de ellos fue instalado en una máquina virtual. Como motor de virtualización se eligió Hyper-V, ya que es parte de Windows 10 Education, que la universidad provee a sus estudiantes como recurso para el aprendizaje, y no presentó limitaciones que impidieran su uso para la prueba de concepto.

El conjunto de contenidos de origen es una unidad de almacenamiento en OneDrive for Business, que es parte de los recursos de aprendizaje provistos por UADE a sus estudiantes.

En cuanto al notificador de cambios y al servicio de diferenciación, ambos fueron implementados con características existentes de Microsoft Graph, una API de Microsoft que sirve, entre otras cosas, para acceder a recursos de OneDrive for Business (que recordemos que es el medio de almacenamiento que se utilizó para el conjunto de contenidos de origen). En esta prueba de concepto, las suscripciones de Microsoft Graph actúan de notificador de cambios, y la funcionalidad de solicitudes “delta” actúa de servicio de diferenciación.

La comunicación entre nodos fue lograda con ION, la implementación de NASA de la pila de protocolos DTN. ION ofrece más de un protocolo de capa de aplicación, entre los cuales se encuentran AMS y CFDP (explicado en Antecedentes). Recordemos, tal como se estableció en la sección “Descripción”, que entre los nodos viajarán 2 tipos de mensajes: los mensajes de diferenciación (TD1), que expresan diferencias incrementales a los conjuntos de contenidos, y los contenidos en sí (TD2), que son necesarios ante una creación o modificación de contenidos. Dado que CFDP es capaz tanto de movilizar archivos como de indicar solicitudes a sistemas de archivos que cubren todos los casos de uso indicados en la Figura 12, el uso de este protocolo es suficiente para cubrir las necesidades del servicio de replicación.

La implementación de BP de ION puede trabajar sobre protocolos tradicionales como TCP y UDP. Dado que UDP no trabaja con confirmación de recibo, se utilizó UDP como capa de transporte debajo de BP. Claro está que en un caso productivo se debería usar un protocolo DTN en su lugar; esto no se hizo en la prueba de concepto para simplificar su implementación, ya que BP provee la misma interfaz a sus consumidores sin importar el protocolo que utilice en la capa inferior.

El retraso en la comunicación entre las máquinas virtuales se logró mediante una característica nativa de Linux llamada “Network Emulator”, documentada en (The Linux Foundation, 2021). Esta característica permite la retención de paquetes UDP provenientes de direcciones especificadas durante un período de tiempo determinado antes de ser entregados a sus destinatarios.

Discusión

La arquitectura documentada en este informe cumple, de acuerdo con lo demostrado en la prueba de concepto, con los objetivos planteados en la sección “Introducción”. Sin embargo, hay determinadas cuestiones que vale la pena observar.

Relación con CDN, Edge Computing y Fog Computing

En primer lugar, cabe destacar que en CDN, Edge Computing y Fog Computing, se descentraliza una arquitectura a priori centralizada, en donde ya existe un modelo cliente-servidor al que se le mejora el rendimiento. El modelo de este proyecto, en cambio, hace posibles escenarios que no lo serían de otro modo, entre nodos de los cuales no se puede establecer claramente una jerarquía. En CDN, Edge Computing y Fog Computing, existe el concepto de “borde” porque se considera a la nube como el centro de la arquitectura, siendo el borde el punto de contacto con los clientes. En el modelo de este proyecto, no existe tal borde ya que se considera que el nodo emisor y el receptor pueden estar ubicados en el mismo nivel jerárquico; incluso se puede dar que el nodo emisor sea el de menor nivel jerárquico, como ocurre en el ejemplo de los pozos petroleros que se presenta en la sección “Descripción”.

El proyecto presenta similitudes en cuanto a utilidad a aquella que ofrecen las redes de distribución de contenido (CDN), tomando como referencia el sistema Akamai explicado en (Dilley, et al., 2002), excepto por la distribución de contenido dinámico. Si el proyecto se hubiera planteado como uno de replicación en redes tradicionales en lugar de redes tolerantes a retrasos y disrupciones, no ofrecería ninguna ventaja que CDN no tenga. El valor agregado del proyecto por sobre lo que CDN ya ofrece, al plantear la replicación sobre redes tolerantes a retrasos y disrupciones, es el de portar las ventajas de las redes modernas CDN a redes subyacentes que no pueden ser operadas con la pila TCP/IP, agregando conceptos y analizando situaciones que no se presentan en la infraestructura sobre la que trabaja CDN, tales como las burbujas de interactividad.

Por otro lado, el modelo presentado no es, ni intenta ser un reemplazo para Edge Computing y Fog Computing, ya que estos no solo llevan datos sino también tareas de procesamiento al borde, delegando tareas de cómputo de las cuales la nube es originalmente responsable (Abdelshkour, 2015) y (Hamilton, 2018).

Colaboración Interplanetaria

En la Tierra, cuando se colabora en, por ejemplo, la autoría de documentos o de código fuente, cuando dos individuos hacen modificaciones en la misma región del mismo archivo, luego estas diferencias deben ser conciliadas (esto se conoce como "resolución de conflictos" en la jerga de los sistemas de control de versión). Esta resolución de conflictos es posible porque existe un servidor central al que ambos autores pueden acceder de forma pseudo instantánea. Sin embargo, en los escenarios en los que existen más de una burbuja de interactividad, tal servidor central ya no puede existir, ya que nunca podría ser accedido de forma pseudo instantánea por ambos autores: si puede ser accedido de forma pseudo instantánea por uno de los autores, dado que el otro autor está situado en otra burbuja de interactividad, este último no podrá gozar del mismo tipo de acceso. Esta es una debilidad innata de este paradigma de colaboración, y lo que presentamos a continuación es una perspectiva alternativa que habilita a la colaboración a pesar de no poder contar con la resolución de conflictos de los sistemas de colaboración tradicionales.

Para ilustrar mejor la idea, se usa como ejemplo a Wikipedia, una enciclopedia en línea que se alimenta de la colaboración de sus usuarios. Suponga el lector que se cuenta con dos burbujas de interactividad: una en la Tierra y la otra en Marte. Para que los individuos habitando el segundo planeta puedan acceder a los contenidos de la enciclopedia, basta con configurar un nodo emisor en la Tierra, y un nodo receptor en Marte, y que el conjunto de contenidos replicado sea el conjunto de archivos de código fuente de cada una de las páginas de la enciclopedia. De esta manera, cada vez que se modifica un artículo en la Tierra, a los pocos minutos estará disponible la nueva versión para ser consumida por los usuarios de Marte.

En Marte no podrían colaborar en la edición, porque el acceso a conjuntos de contenido replicados es siempre de solo lectura. Sin embargo, esto no impediría a los usuarios en Marte crear un nuevo conjunto de contenidos, del cual ellos sean dueños. Inicialmente, lo que pueden hacer, es copiar el conjunto de contenidos replicado a un servidor nuevo, y ese puede ser el conjunto inicial de artículos. De ahí en más, ellos pueden modificar cuando deseen los artículos locales, de los cuales sí son dueños.

En principio, si quieren que su enciclopedia siga recibiendo las actualizaciones de la enciclopedia de la Tierra, pueden automatizar el proceso de actualización informáticamente.

Cuando haya algún conflicto entre actualizaciones provenientes de la Tierra y actualizaciones hechas por autores locales, podrán resolver los mismos manualmente tal como se hace hoy en día para cualquier documento o archivo de código fuente en la Tierra.

Nótese que la versión del artículo resultante de la resolución de conflictos aún no estará disponible para la Tierra. Si los autores de Marte desean que en la Tierra tengan acceso a su versión de la enciclopedia, se deberá configurar un nuevo nodo emisor, en Marte, y un nuevo nodo receptor, en la Tierra. Una vez configurados los nodos y funcionando, en la Tierra tendrán acceso a la versión marciana de la enciclopedia.

De igual manera, los usuarios terrestres tendrán la opción de incorporar en su versión de la enciclopedia los cambios hechos en Marte, o no hacerlo. Esta dinámica es similar a la que ocurre entre enciclopedias de distintos idiomas dentro de Wikipedia: cuando un artículo es editado en un idioma, los hablantes de un segundo idioma siempre tienen la opción de observar los nuevos cambios e incorporarlos a su versión traducida, pero esto no es obligatorio. El resultado es una serie de enciclopedias que divergen, pero se enriquecen entre sí.

Distribución Interplanetaria de Software

Uno de los aspectos más interesantes de esta arquitectura es que posibilita la distribución y actualización de software entre planetas diferentes.

Suponga el lector que una empresa de la Tierra brinda como servicio una aplicación que informa el pronóstico del clima. Ya establecidos en el mercado terrestre, desean comenzar a operar también en Marte. En este segundo planeta, hay que tener en cuenta determinados factores, como, por ejemplo, la ausencia de un servicio de posicionamiento global. Sin embargo, excepto por algunas cuestiones como la mencionada, la mayoría del comportamiento de la aplicación es exactamente el mismo, por lo que sería conveniente mantener un único repositorio y no tener “bifurcaciones” (es decir, dos copias distintas del repositorio de código, mantenidas por dos equipos diferentes).

Lo que el equipo decide hacer es usar compilación condicional basada en una macro llamada “Planeta”. Cuando esta macro tiene un valor de “Tierra”, se compilará el código que usa el servicio de GPS para averiguar en dónde se encuentra el usuario; cuando la macro tiene un valor de “Marte”, se compilará el código que solicita al usuario su ubicación mediante un cuadro de diálogo.

A continuación, el equipo configura un nodo emisor en la Tierra, y un nodo receptor en Marte, y establece como conjunto de contenidos a replicar el repositorio de código fuente de la aplicación. Por último, configura un servidor de compilación en Marte, que actúa sobre el conjunto de contenidos replicado y compila el código definiendo la macro “Planeta” con el valor “Marte”. De más está decir que, en el servidor de compilación de la Tierra, la macro “Planeta” se define con el valor “Tierra” a la hora de compilar el código.

Una vez completada esta configuración, el equipo de desarrollo de la Tierra puede actualizar la aplicación en ambos planetas en todo momento, sin que esto implique contratar desarrolladores adicionales.

Si fuera necesario, el equipo puede además incluir en cada planeta un archivo de entorno que incluya datos tales como el nombre del planeta, la duración del día, etc. Estos archivos serían ignorados por la replicación ya que son característicos del entorno (lo mismo ocurre con, por ejemplo, archivos de entorno con configuraciones de producción o desarrollo en aplicaciones tradicionales).

Replicación Selectiva y Priorizada

Algunos conjuntos de contenidos ocupan más espacio de almacenamiento del que se puede disponer en lugares remotos; de hecho, con un enfoque realista, este será probablemente el caso la mayoría de las veces. Además, cuando el espacio de almacenamiento es escaso, algunos contenidos son más críticos en su entrega que otros.

Por estos motivos, se sugiere a los implementadores brindar la posibilidad de extender los replicadores con código de usuario que determine si un cambio debe ser replicado o no, y que le pueda asignar un puntaje de prioridad a cada cambio a ser replicado.

Al brindar esta posibilidad de extensión, se abre la puerta al uso de tecnologías tales como la inteligencia artificial para determinar qué contenidos es más probable que necesiten en la burbuja de interactividad remota, y de esta manera servir mejor a los usuarios de esta, evitando priorizar cambios en contenidos que rara vez son consumidos.

Múltiples Emisores, Múltiples Receptores, y Encadenamiento

Este modelo no impone restricciones sobre la posibilidad de tener múltiples nodos emisores para un mismo nodo receptor y múltiples nodos receptores para un mismo nodo emisor, ni tampoco restringe la posibilidad de encadenar varios nodos en secuencia.

Todos estos mecanismos pueden ser usados a favor, para generar implementaciones arquitectónicas redundantes como la que se muestra en la Figura 20.

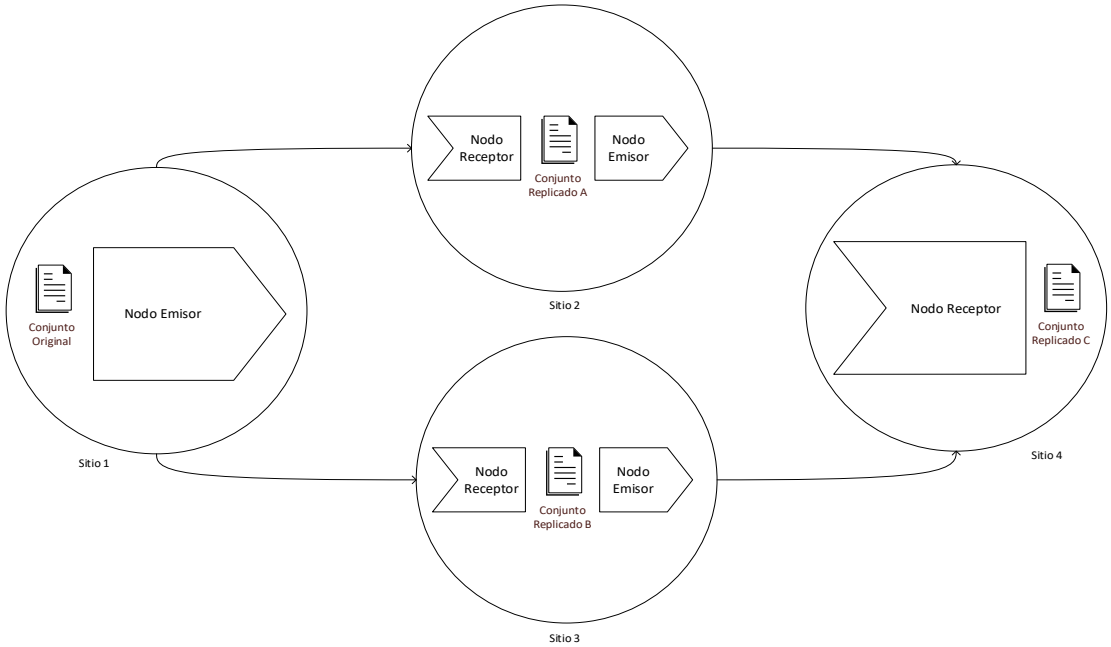

Figura 20: Redundancia a través de múltiples emisores, receptores y encadenamiento.

En una implementación como la mostrada, si por algún motivo el sitio 2 o el sitio 3 dejan de operar temporalmente, el sitio 4 seguirá recibiendo las actualizaciones. Este sistema puede ser útil para aquellos casos en los que las redes subyacentes que soportan las conexiones entre los distintos sitios no cuenten ya con redundancia suficiente.

Conclusiones

La arquitectura que se presentó en este trabajo, además de facilitar el monitoreo de sitios industriales o la gestión de recursos digitales en misiones espaciales, puede ser usada de forma más cotidiana. Usuarios de internet en países subdesarrollados y zonas con conectividad de baja calidad, podrían instalar nodos receptores en sus computadoras personales, y suscribirse a nodos emisores que pueden ser configurados por la comunidad. Imaginemos un estudiante adolescente de una zona con acceso a internet hogareño de baja calidad, que desea aprender sobre una temática particular leyendo una enciclopedia online. Si existiera una implementación que sea receptora de DCSR, el estudiante podría seleccionar temas que son de su interés, y DCSR se encargaría de aprovechar la conectividad disponible para almacenar en su dispositivo los artículos relevantes para el tema. Cuando el estudiante se decida a comenzar a leer, no necesitará tener conexión estable en ese mismo momento, porque la conexión disponible, aunque de baja calidad, ya fue aprovechada por DCSR en momentos de inactividad. El deseo de aprender podría ser satisfecho sin barreras, lo cual es algo positivo para el estudiante, y para el resto de la comunidad.

Como se mencionó en la declaración del aporte esperado[6] en la propuesta de tema cuando este proyecto nació, además del objetivo de corto plazo, que es el de proveer una arquitectura para solucionar escenarios como los mencionados en la sección “Descripción”, o como el noble escenario del párrafo anterior, también se estableció como objetivo uno de largo plazo, que es el de ofrecer un marco de trabajo que pueda ser utilizado en un futuro lejano, en el que la civilización sea de naturaleza interplanetaria. Dos requisitos se mencionaron en su momento para llegar a tal objetivo.

El primer requisito mencionado fue que la solución propuesta cuente con cualidades arquitectónicas tales como extensibilidad y escalabilidad. En la prueba de concepto se demostró que la arquitectura presentada tiene el potencial de funcionar sin importar qué tanta distancia se interponga entre los colaboradores. La solución propuesta funciona en escenarios interplanetarios.

[6] Si el lector desea examinar la declaración original, una copia textual se incluye en el Anexo B: Declaración Inicial de Aporte Esperado.

El otro requisito mencionado fue que este proyecto abra un debate sano y constructivo sobre la necesidad de una nueva mentalidad en la forma en la que se colabora mediante tecnología. Este proyecto es solo un primer paso en esa dirección. Como se estableció en la propuesta de tema y como se reitera en este informe, el mayor aporte de este proyecto a la humanidad no será la arquitectura presentada para mejorar la situación actual, sino los debates que se comiencen a generar a partir de su difusión. Lo mejor que puede ocurrir una vez concluido este proyecto, es que profesionales y estudiantes examinen en profundidad la arquitectura que se propuso y encuentren oportunidades de mejora o incluso arquitecturas alternativas y superiores a la presentada.

Bibliografía

Abdelshkour, Maher. 2015. IoT, from Cloud to Fog Computing. *Cisco Blogs.* [En línea] 25 de Marzo de 2015. [Citado el: 1 de Marzo de 2021.] https://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing.

Burleigh, Scott, Fall, Kevin and Birrane, Edward J. 2020. Bundle Protocol Version 7. *IETF.* [Online] Octubre 27, 2020. [Cited: Febrero 16, 2021.] https://tools.ietf.org/html/draft-ietf-dtn-bpbis-27.html.

Burleigh, Scott, Scott, Keith y Snell, Mike. 2019. Motivation for DTN: Terrestrial Use Cases. *InterPlanetary Networking Special Interest Group.* [En línea] 13 de Mayo de 2019. [Citado el: 30 de Marzo de 2020.] http://ipnsig.org/2019/05/13/motivation-for-dtn-terrestrial-use-cases/.

CCSDS. 2012. Asynchronous Message Service - Informational Report. *CCSDS.* [Online] December 2012. [Cited: 12 29, 2020.] https://public.ccsds.org/Pubs/735x0g1.pdf. 735.0-G-1.

—. 2011. Asynchronous Message Service - Recommended Standard. *CCSDS.* [Online] September 2011. [Cited: Diciembre 29, 2020.] https://public.ccsds.org/Pubs/735x1b1.pdf. 735.1-B-1.

—. 2007. *CCSDS File Delivery Protocol (CFDP)—Part 3: Interoperability Testing Final.* 2007.

Dilley, John, et al. 2002. Globally Distributed Content Delivery. *University of Massachusetts Amherst.* [Online] Septiembre/Octubre 2002. [Cited: Marzo 1, 2021.] https://people.cs.umass.edu/~ramesh/Site/PUBLICATIONS_files/DMPPSW02.pdf.

Durst, Robert, Feighery, Patrick y Scott, Keith. 2002. Why not use the Standard Internet Suite for the Interplanetary Internet? *InterPlanetary Networking Special Interest Group.* [En línea] 2002. [Citado el: 30 de Marzo de 2020.] http://www.ipnsig.org/reports/TCP_IP.pdf.

El Cronista. 2021. Telefónica se une a la petrolera de Galuccio para proveer Internet en Vaca Muerta. *El Cronista.* [En línea] 1 de Marzo de 2021. [Citado el: 7 de Marzo de 2021.]

https://www.cronista.com/apertura-negocio/empresas/Telefonica-se-une-a-la-petrolera-de-Galuccio-para-proveer-Internet-en-Vaca-Muerta-20190626-0005.html.

Freeman, Eric, et al. 2004. *Head First Design Patterns.* s.l. : O'Reilly Media, Inc., 2004. 9780596007126.

Gamma, Erich, et al. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley Professional, 1994. 978-0201633610.

Hamilton, Eric. 2018. What is Edge Computing: The Network Edge Explained. *Cloudwards.* [Online] Diciembre 27, 2018. [Cited: Marzo 1, 2021.] https://www.cloudwards.net/what-is-edge-computing/.

IETF. 2015. Internet Engineering Task Force. *DTN Use Cases.* [En línea] 27 de Febrero de 2015. [Citado el: 30 de Marzo de 2020.] https://trac.ietf.org/trac/dtn/wiki/DtnUseCases.

Kumar, Vimal, et al. Push-Pull Caching. *University of Texas at Arlington.* [Online] [Cited: Marzo 1, 2021.] http://crystal.uta.edu/~kumar/cse6306/papers/team3a.pdf.

Más Energía. 2020. Conectividad en Vaca Muerta: se unen Tecosur y Grupo Datco. *Más Energía.* [En línea] 17 de Septiembre de 2020. [Citado el: 7 de Marzo de 2021.] https://mase.lmneuquen.com/conectividad/vaca-muerta-se-unen-tecosur-y-grupo-datco-n733181.

Microsoft. 2021. Azure Functions documentation. *Microsoft Docs.* [En línea] 10 de Febrero de 2021. [Citado el: 15 de Febrero de 2021.] https://docs.microsoft.com/en-us/azure/azure-functions/.

—. 2019. Azure Queue storage documentation. *Microsoft Docs.* [En línea] 12 de Diciembre de 2019. [Citado el: 15 de Febrero de 2021.] https://docs.microsoft.com/en-us/azure/storage/queues/.

—. 2017. Track changes for a Drive. *Microsoft Docs.* [En línea] 10 de Septiembre de 2017. [Citado el: 15 de Febrero de 2021.] https://docs.microsoft.com/en-us/graph/api/driveitem-delta?view=graph-rest-1.0&tabs=http.

—. 2020. Use the Microsoft Graph API to get change notifications. *Microsoft Docs.* [En línea] 22 de Septiembre de 2020. [Citado el: 15 de Febrero de 2021.] https://docs.microsoft.com/en-us/graph/api/resources/webhooks?view=graph-rest-1.0.

Muri, Paul and McNair, Janise. 2013. *A Performance Comparison of DTN Protocols for High Delay Optical Channels.* Gainesville : s.n., 2013.

NASA. 2012. Disruption Tolerant Networking. *National Aeronautics and Space Administration.* [En línea] 6 de Diciembre de 2012. [Citado el: 30 de Marzo de 2020.] https://www.nasa.gov/content/disruption-tolerant-networking/.

—. 2020. TREK CCSDS FILE DELIVERY PROTOCOL (CFDP) USER GUIDE. *Telescience Resource Kit.* [En línea] Marzo de 2020. [Citado el: 18 de Febrero de 2021.] https://trek.msfc.nasa.gov/Documents/trek_5_2_1/trek_cfdp_user_guide.pdf. TREK-USER-0003.

The Linux Foundation. 2021. netem. *Linux Foundation DokuWiki.* [Online] Febrero 5, 2021. [Cited: Febrero 18, 2021.] https://wiki.linuxfoundation.org/networking/netem.

Warthman, Forrest. 2015. Introduction to Delay & Disruption Tolerant Networking (DTN). *InterPlanetary Networking Special Interest Group.* [En línea] 14 de Septiembre de 2015. [Citado el: 30 de Marzo de 2020.] http://ipnsig.org/introducing-delay-disruption-tolerant-networking-dtn/.

Williams, David Richard. 2018. Venus Fact Sheet. *NASA Space Science Data Coordinated Archive.* [En línea] 27 de Septiembre de 2018. [Citado el: 30 de Marzo de 2020.] https://nssdc.gsfc.nasa.gov/planetary/factsheet/venusfact.html.

Wynne, Klaas. 2002. Causality and the Nature of Information. 2002.

Anexo A: Componentes Originales

A continuación, se enumeran los componentes que se divisaron al inicio del proyecto, en contraposición a los que se terminaron exponiendo en la versión final de este informe.

- Receptor: se encarga de recibir actualizaciones de contenido y almacenarlas en el búfer de acciones para que luego sean aplicadas por el procesador.
- Búfer de acciones: se encarga de almacenar acciones aún no aplicadas en el almacenamiento.
- Procesador: procesa las acciones todavía no aplicadas del búfer de acciones (ej.: creación de archivo, modificación de archivo, eliminación de archivo, etc.) haciendo los cambios necesarios en el Storage; luego de procesar una acción, la marca como aplicada localmente en el búfer de acciones.
- Almacenamiento: tiene las copias locales de los archivos que se desean consultar, para que el usuario tenga acceso instantáneo a los mismos.
- Compartidor: reenvía los paquetes del búfer de acciones a los nodos vecinos configurados, para propagar los cambios en los mismos (se considera otro nodo a otro equipo con los mismos 7 componentes de software que también tiene acceso dificultado a la red); el receptor del nodo vecino se encargará de recibir el mensaje que el compartidor del nodo local le envía.
- Retroalimentador: se encarga de recibir las confirmaciones de recepción de acciones que el compartidor compartió con nodos vecinos; por cada confirmación recibida de un nodo vecino, marca en el búfer de acciones esa acción como propagada a ese nodo.
- Limpiador: se encarga de buscar continuamente en el búfer de acciones las acciones que fueron marcadas como procesadas localmente y compartidas con todos los nodos vecinos y las elimina del búfer de acciones.

La primera debilidad que tiene este sistema es que todos los nodos deben implementar todos los componentes, y no existe la distinción entre un nodo emisor y un nodo receptor.

En segundo lugar, esta división original está prescribiendo demasiados detalles de implementación que podrían ejecutarse de otra manera. Un ejemplo de esto es el “búfer de acciones”. En la prueba de concepto presentada en este informe, un rol similar es el que cumple

la “cola de cambios”; pero en otra implementación se podría decidir no contar con un búfer y replicar los cambios ni bien son recibidos, delegando la garantía de entrega a un componente posterior. En consecuencia, a esto, el “limpiador” también se vuelve opcional, ya que si no se implementa un búfer de acciones tampoco hace falta ningún componente para limpiarlo.

Por último, el “retroalimentador” obliga a todos los nodos a ser en todo momento conscientes de qué contenidos fueron replicados exitosamente en cuáles nodos vecinos, y cuáles aún no. Aunque podría ser una implementación válida para garantizar la entrega, existen otras opciones, tales como la custodia de entrega de los protocolos como BP y CFDP, por lo que prescribir un componente como el retroalimentador resulta inapropiado, ya que se puede solucionar el problema sin su estricta presencia.

Anexo B: Declaración Inicial de Aporte Esperado

La siguiente es una copia textual de la declaración del aporte esperado que se aprobó como parte de la propuesta de tema del proyecto. Se incluye como referencia para aquellos lectores que deseen apreciar el informe con una visión más rica del contexto.

“Muchas veces en la industria de la tecnología, y en particular en el caso de algunos protocolos de red, nos encontramos con limitaciones que fueron inadvertidamente dejadas por nuestros antecesores y que nos fuerzan a invertir grandes cantidades de tiempo (lo cual generalmente se traduce en dinero) en cambios y actualizaciones que podrían haberse evitado con sólo adoptar un pensamiento más previsor. De esto debemos aprender todos y trabajar duro para evitar que las soluciones que creamos hoy causen pérdidas de tiempo y esfuerzo en el mañana.

Con eso en mente, me esforzaré en solucionar el problema que divisé de la forma más extensible posible. Dedicaré las horas que tengo asignadas para la materia en brindar una solución que se adapte al presente pero también al futuro, y, una vez concluido el proyecto, dedicaré las horas que sean necesarias para que, juntos, podamos todos mejorar esta solución tanto como sea necesario, de tal manera que su uso en el mundo real y en entornos que lo necesitan se vuelva una realidad.

Idealmente, la solución mejorará la experiencia de los usuarios en dos escenarios claves, uno en el futuro inmediato, y el otro en el futuro lejano.

En el futuro inmediato, usuarios de países subdesarrollados y en lugares remotos con el acceso a Internet dificultado, podrán gozar de obtención aparentemente instantánea de recursos que pueden ser pesados, como imágenes o videos, incluso en momentos en los que no tienen conexión.

Como vara de calidad, mirando al futuro lejano y tomando como desencadenante las recientes abismales reducciones en los costos de los lanzamientos espaciales y los planes de algunas potencias mundiales de expandir su presencia al resto del sistema solar a partir del 2030, me propongo que la solución pueda funcionar, al menos de forma teórica y respaldada por el prototipo, en un escenario en el que, desde otros planetas, se pueda acceder instantáneamente a contenido del Internet de la Tierra.

Hoy eso puede sonar descabellado o innecesariamente futurista, pero, retomando lo que dije al principio de esta sección, es mi responsabilidad y la de todos aprender de nuestros errores y pensar en el futuro para que las soluciones que creamos hoy no necesiten ser reemplazadas por nuestros hijos si eso se puede evitar ahora con un pensamiento apenas más global y una inversión de tiempo relativamente marginal.

Por último, me gustaría aclarar que el principal objetivo intangible y no medible del tema que estoy proponiendo, es el de abrir una discusión que creo firmemente que necesita ser abierta. Con este proyecto no intento llegar con una solución mágica a salvar el día, sino que me esfuerzo por estimular la imaginación y la creatividad grupal de la industria de la tecnología para que se inicie un debate que me parece que va a ser beneficioso para muchos usuarios actuales y por venir. Aclaro esto porque es mi convicción que las grandes cosas no se logran individualmente, sino que son el resultado de un esfuerzo repetitivo y constante de un grupo de personas que enriquecen el debate con su participación individual. Por este motivo, si luego de publicado este desarrollo se abre la conversación sobre cómo lograr que usuarios con acceso a Internet muy limitado puedan obtener los recursos que necesitan de forma instantánea, voy a sentirme personalmente realizado, y voy a sentir que mi esfuerzo valió la pena.

Considero que la apertura de este debate es sin ninguna duda el aporte más importante que voy a estar realizando para la humanidad, incluso más importante que el diseño de la solución propuesta en sí.”