Comentario técnico: CTC-086
Componente: **Internet de nuestras cosas (¿IooT?)**
Autor: Sergio R. Caprile, Senior R&D Engineer

Revisiones	Fecha	Comentarios
0	02/04/20	

Existe la Internet y existen las cosas. Algunas de esas cosas tienen conexión a alguna red. El monitorear y controlar de manera remota dispositivos que nos entregan un dato de interés o realizan una tarea de servicio es tan viejo como las comunicaciones digitales mismas; la tecnología actual ayuda un poco más, pero muchos conceptos tienen ya varias décadas.
El concepto de la IoT nace como la evolución del M2M, pero las cosas no hablan entre sí todavía, simplemente porque no hay un lenguaje común, aún, pero cuando comiencen a hacerlo van a tener que resolver algunos problemas de logística para encontrarse entre sí.
Fuera del concepto como propulsor de mercado y generador de ingresos, el hecho es que nosotros hoy queremos hablar con las cosas o que nos digan algo, y tal vez que hablen entre sí, y para que eso suceda *nosotros* debemos diseñar una red. Lo que en realidad queremos, quienes tenemos una relación con la electrónica como para leer este artículo, es comunicarnos con nuestras cosas y saber cómo están, cómo las usan, y agregarle valor a eso para transformarlo en mejores productos y más ingresos.
La labor de los desarrolladores se inclina entonces a escuchar a sus clientes y diseñar la red que más se acerca a sus necesidades, previendo escalabilidad, crecimiento y flexibilidad, sin hacerlos pagar hoy por lo que tal vez vayan a usar mañana.

Índice de contenido

Breve descripción del problema

En algunos ámbitos de divulgación se habla de protocolos y dilemas de elección. Visto que no existe cosa física como “una IoT”, no hay por ende “un protocolo” o grupo de éstos que nos sirva para "conectarnos a la IoT" o "hacer cosas de IoT"; ni cosas que hacer, al menos no ahora. No existe entonces un dilema entre la selección de uno u otro protocolo sino una decisión arquitectónica para diseñar la red que mejor se ajusta a las necesidades del cliente.
Cuando comenzamos a diseñar una red, siempre dentro del ámbito de aplicabilidad comercial dentro de lo habitual en esta región del planeta, en principio las consideraciones suelen dividirse en el espectro entre dos grandes modelos arquitectónicos.

Seguramente comenzaremos analizando si los equipos requieren una interfaz de operación remota, y/o si reportan información; esto nos inclinará más hacia uno u otro lado. Luego llegaremos a la conclusión de que necesitaríamos un concentrador, el cual debería estar en un lugar accesible por todos con mínimas complicaciones de ruteo, luego evaluaríamos los problemas de NAT y las direcciones privadas cuando queremos acceder a los equipos, y finalmente evaluaremos los problemas de escalabilidad de dicho sistema. Nos haremos preguntas como: ¿Qué pasa si los dispositivos quieren hablar entre sí o si hay dispositivos o usuarios que necesitan consumir información de otros? ¿No sería interesante que los actuadores pudieran recibir la información de los sensores y obrar sin intervención de una inteligencia central? ¿Cómo manejamos el acceso, un sitio central que recibe y deriva o cada sensor tiene una lista de a quiénes envía la información? ¿Podríamos por este medio resolver además el problema del acceso a los dispositivos? ¿Qué pasa si en el momento que necesitamos enviar una orden, no tenemos comunicación con el dispositivo?
Como vemos, en general pensamos en un modelo cliente-servidor con un esquema de pedidos y respuestas. Este esquema resulta fuertemente acoplado para lo que deseamos y deberíamos buscar otra solución arquitectónica.

Una posible solución

Lo que proponemos es desacoplar los bloques y abstraernos a una idea arquitectónica llamada *publish/subscribe*, *publicar/suscribirse*.
En esta arquitectura, existen unidades abstractas que son generadores de información; dispositivos o parte de ellos que proveen la información necesaria para desarrollar la tarea. Estos entes publican dicha información en un sitio central denominado *broker*, en forma de unidades temáticas que podemos llamar *tópicos*. Por otra parte, existen otras unidades abstractas que son consumidores de información; dispositivos o parte de ellos que toman la información generada por otros y la utilizan. Los generadores publican en tópicos acordados con los consumidores, quienes se suscriben y obtienen la información cuando es publicada. De este modo, la tarea de logística de distribución y mantenimiento de la información y el acceso a ésta se distribuye entre los consumidores. El sitio central, el *broker*, mantiene una lista dinámica de sus suscriptores y entrega los anuncios a medida que son publicados.
Agreguémosle a esto alguna que otra garantía de recibir la información al menos una vez, o exactamente una vez, y tenemos MQTT (MQ Telemetry Transport), protocolo desarrollado por IBM como ejemplo e implementación de esta arquitectura.
Como corolario tenemos el hecho de que los actuadores pueden además estar suscriptos a un tópico propio determinado, y entonces el enviar un mensaje a un actuador se reduce a publicar en dicho tópico. Es decir, no sólo reciben datos suscribiéndose a los tópicos de los sensores que les interesan, sino que a la vez pueden recibir “mensajes manuales” suscribiéndose a un tópico que les es propio. Eligiendo cuidadosamente el esquema de tópicos, podemos además pensar en direccionar grupos de dispositivos en un mismo mensaje, es decir, mandar un mensaje a todos los dispositivos con determinadas características u obtener los datos de todos los dispositivos con determinadas características.
Nos quedan dos puntos a resolver.

- cómo administrar los tópicos, es decir, cómo sabe cada actuador a qué sensor escuchar.
- interfaces de usuario. La comunicación en esta arquitectura es inherentemente en un solo sentido.

El primero lo resolveremos como parte del diseño de la red; si necesitamos algún mecanismo dinámico deberemos proveerlo por encima de ella, dado que ya tenemos resuelto el mecanismo de comunicación.
Para el segundo, una solución inmediata es diseñar la arquitectura de modo que no exista respuesta directa sino colectiva[1]. De ser imprescindible recibir respuestas directas existen estrategias para proveer algún modo de identificar las preguntas o indicar en ellas la dirección de respuesta[2].

1 El dispositivo publica su estado luego de cada cambio, todos los interesados ven el resultado de una operación
2 Dos estrategias clásicas son incluir un token que se replica en las respuestas en un canal común, o incluir en la pregunta el tópico donde se debe publicar la respuesta. Si bien esto es en principio engorroso, cuando se embebe como transporte de un protocolo de nivel superior, pasa a ser anecdótico y queda oscurecido dentro de las otras razones que afectan la performance

Dado que se trata de actividades de baja frecuencia, las ventajas de olvidarse de NAT y compañía superan a las desventajas.
En síntesis, cuando la red comienza a crecer, requerimos conversaciones entre los nodos, o la complejidad excede a lo que estamos acostumbrados a manejar, conviene pasar al nivel siguiente de abstracción y aplicar una solución arquitectónica escalable como *publish/subscribe*. Podemos entonces implementar MQTT para la recolección de la información y como transporte para el resto.

Una solución genérica escalable

Describimos aquí una solución general que de acuerdo al caso podemos ir haciendo particular de acuerdo a nuestras necesidades en costo y prestaciones. Los detalles de seguridad enunciados los podemos analizar más adelante.
El esquema en una primera aproximación se reduce a:

- MQTT para obtener los datos y redistribuir a los interesados en tiempo real
- Guardar según el tipo de dato en tablas (no relacionales) o bases de datos relacionales. Debido a costo y frecuencia de actualización partimos en 3 grandes grupos: telemetría, estado y configuración
- Tomar los datos de las tablas/bases para presentación y para analytics

Luego, a medida que hilamos fino, nos podemos plantear y evaluar alternativas, por ejemplo para el ingreso de datos:

- Autenticación por certificados
- Encripción en tránsito por TLS
- Costo de comunicación por tamaño de mensaje y por destino reenviado

Para las bases de datos:

- Encripción de la base de datos por certificados
- Costo de storage por tipo de almacenamiento

Para pre-procesamiento y también presentación y analytics:

- Costo de procesamiento por servicio invocado

En cuanto al dimensionamiento de una solución de estas características, tanto red como servidores pueden modelarse mediante la teoría de colas, en particular con tráfico siguiendo la distribución de Poisson y los modelos de Markov; sin embargo consideramos más conveniente en nuestro ámbito de acción el contratar los servicios de Amazon, Google u otros. Estos servicios ofrecen escalado dinámico, es decir, podemos contratar un servicio mínimo e ir ampliándolo a medida que vamos requiriendo más recursos para satisfacer una mayor demanda de nuestro servicio.
Si tenemos know-how en sistemas pero no en redes, podemos contratar los servicios de brokers MQTT como por ejemplo CloudMQTT[3]. De todos modos, no es fácil manejar la cantidad de datos de un sistema que progresa, por lo que es recomendable alojar también los sistemas en un proveedor en la Internet como por ejemplo Heroku[4]. Ambos tienen planes escalables y un grado de integración mutua.
Para aplicaciones más limitadas en tamaño o que requieran un manejo más en bajo nivel; o incluso para una pequeña prueba de concepto, disponemos de esquemas completos como por ejemplo myDevices[5] y su Cayenne.
Finalmente, si queremos y podemos ocuparnos de casi todo, existen brokers MQTT que podemos instalar y manejar; open source como Mosquitto[6] y comerciales como HiveMQ[7]. Este último además ofrece un servicio similar al de un broker MQTT brindando no solo su software sino la infraestructura para tenerlo operativo: HiveMQ Cloud[8].

3 https://www.cloudmqtt.com/
4 https://www.heroku.com/
5 https://mydevices.com/
6 https://mosquitto.org/
7 https://www.hivemq.com/
8 https://www.hivemq.com/cloud/

Haremos ahora la salvedad de indicar que MQTT requiere que los dispositivos estén conectados a la red permanentemente. Soporta desconexiones y reconexiones, obviamente, pero en esencia el modelo asume la conexión permanente y con un transporte como TCP. Para aplicaciones que no tienen estas características, existe otro tipo de redes a diseñar. En principio, para replicar este modelo existe MQTT-SN, completamente operativo y en proceso de estandarización al momento de escribir este texto.

Seguridad

La seguridad se analiza antes, o durante el diseño de la red; no al final. En esta descripción consideramos más simple y acertado plantearla luego de haber introducido los otros conceptos y la arquitectura en sí.

Seguridad en el transporte de los datos

¿Qué tanto podemos garantizar que lo que se recibe es lo que se envió y que corresponde a tráfico real de la red?. Podemos desde firmar los mensajes hasta encriptar todo el tráfico, dependiendo de nuestra arquitectura y de las capacidades del hardware.
En el caso de MQTT, la especificación 3.1.1 indica que es posible correrlo sobre TLS o Websockets (si el hardware lo soporta...).
En general, además, casi siempre es posible introducir algún esquema dentro de la aplicación misma.
En todos los casos, debemos preguntarnos qué sucede si los datos son vistos por terceros no deseados, y qué sucede si alguien modifica datos en curso o introduce datos inválidos. Esto, sumado a la privacidad, nos contestará si lo necesitamos o no. En general, suele ser recomendable, pero en casos como por ejemplo la temperatura de un lugar público, no es necesario.

Seguridad en el acceso a la red

Si alguien logra introducir datos considerados válidos, puede afectar nuestras mediciones y por consiguiente nuestras acciones. En una red que es sólo telemetría, podría alterar lo que vemos y hasta generar dispositivos que no existen. También, un usuario no autorizado podría operar sobre un actuador, en una red en la que existe la capacidad de ello.
Si esto representa un riesgo, deberemos proveer seguridad en la información y un mecanismo para validar los dispositivos. En los casos que involucran TLS es posible distribuir certificados que identifican a los integrantes de la red. En otros casos se trata de tokens o puede ser simplemente un login con nombre de usuario y contraseña. Esto último es soportado directamente por MQTT.
Es posible también que no exista la posibilidad de operación sobre ningún dispositivo, y que lo máximo que un atacante pueda realizar es introducir datos inválidos o incluso dispositivos inválidos. Podemos oscurecer la información (mantenerla propietaria) y esperar que dado que el esfuerzo necesario es superior a los posibles beneficios obtenidos, nadie se tomará el trabajo de molestarnos. Sin embargo, no es buena práctica de diseño si nos conectamos a algo tan amplio como la Internet.

Seguridad en el almacenamiento de los datos

Resulta claro que quien puede acceder a donde se almacenan los datos tiene posibilidad de acceder a los mismos. Y aquí nos adentramos en otro terreno que es el de la propiedad de dichos datos, quién es el dueño. Una vez saldado el debate entre proveedor del servicio y generador de contenidos, existen mecanismos para garantizar ambas cuestiones, como por ejemplo la encripción con certificados.

Disponibilidad

Sí, un pequeño detalle que dejamos para el final. Todo esto debe operar y debe hacerlo 24/7. En algunos casos los clientes exigirán una cierta disponibilidad e incluso deberemos firmar un acuerdo de calidad de servicio (SLA, Service Level Agreement), lo cual es muy difícil de lograr para quienes no disponen del know-how específico de ambos rubros, para quien no es un datacenter. Por esta razón inicialmente hemos recomendado

aprovechar a Google y Amazon (por ejemplo) que no sólo ya tienen razones de por sí para estar operativos y cubrir todas las áreas del mundo, sino que además tienen planes para utilizar los servicios con varias zonas de acuerdo a la disponibilidad que requerimos y el precio que podemos pagar.