

Monitorización Inteligente de un Espacio Físico en Tiempo Real



2015-07-26

Índice

| | | |
|------|------------------------------------------------------------|----|
| 1. | Introducción..... | 3 |
| 2. | Motivaciones | 4 |
| 3. | Definición del proyecto y Arquitectura de la solución..... | 5 |
| 3.1. | Fuente de datos | 6 |
| 3.2. | Sensor | 6 |
| 3.3. | Webservice | 7 |
| 3.4. | Sistema de colas | 7 |
| 3.5. | Procesamiento en Streaming | 8 |
| 3.6. | Base de datos | 8 |
| 3.7. | Visualización de datos..... | 9 |
| 3.8. | Despliegue actual del proyecto | 9 |
| 4. | Resultados y conclusiones | 16 |
| 5. | Futuras líneas de trabajo | 17 |
| 6. | Bibliografía..... | 18 |
| 6.1 | Probe Request (Source Data) | 18 |
| 6.2 | Raspberry pi 2 | 18 |
| 6.3 | Kafka Producer API (java)..... | 19 |
| 6.4 | Spark Streaming & Spark SQL | 19 |
| 6.5 | InfluxDB & API (Scala)..... | 19 |
| 6.6 | Grafana | 19 |

1. Introducción

El presente documento representa la memoria del proyecto fin de programa “Experto en Big Data” realizado por Javier Abascal Carrasco bajo la supervisión del tutor Iván de Prado Alonso.

El objetivo del proyecto ha sido realizar una monitorización inteligente en tiempo real de un espacio físico utilizando los conceptos, ideas, diseños y tecnologías aprendidas durante el programa de post-grado. La monitorización del espacio ha consistido en la creación de un panel de control online (Dashboard) y un sistema generador de alarmas. Para ello:

- Se ha diseñado un sensor capaz de monitorizar de forma unívoca a los dispositivos con acceso a Internet (protocolos 802.11x Wi-Fi) y hemos aprovechado el hecho de que actualmente la mayoría de las personas llevan consigo un Smartphone con conectividad a Internet
- Se ha desarrollado una plataforma de ingestión, procesamiento, análisis, almacenamiento de datos y visualización alojada temporalmente en Amazon Irlanda

El piso 4º situado en la Calle Mesón de Paños nº 11 de Madrid capital (CP: 28013) ha sido el espacio físico elegido para la realización. En esta residencia habitan Javier y sus dos compañeros de piso, David y Miguel.

2. Motivaciones

La decisión de proponer y realizar este trabajo en vez de alguno de los proyectos listados en el programa ha estado motivada por los siguientes cuatro puntos:

1. El desafío de realizar un proyecto desde cero y abarcando desde la generación de los datos hasta la visualización de los mismos
2. El reto de aprender y obligarme a usar las herramientas, lenguajes y tecnologías donde poseo menor experiencia (Spark & Scala, Webservices, Scripting, Kafka, InfluxDB, Grafana)
3. La motivación de saber que los datos son generados por señales (IEEE 802.11x) que he aprendido en mis estudios universitarios (Ingeniero de Telecomunicaciones)
4. El orgullo de poder proponer una nueva alternativa al proceso de ingestión realizado actualmente en la Startup Gennion Solutions¹, que a fecha del presente documento, me encuentro trabajando.

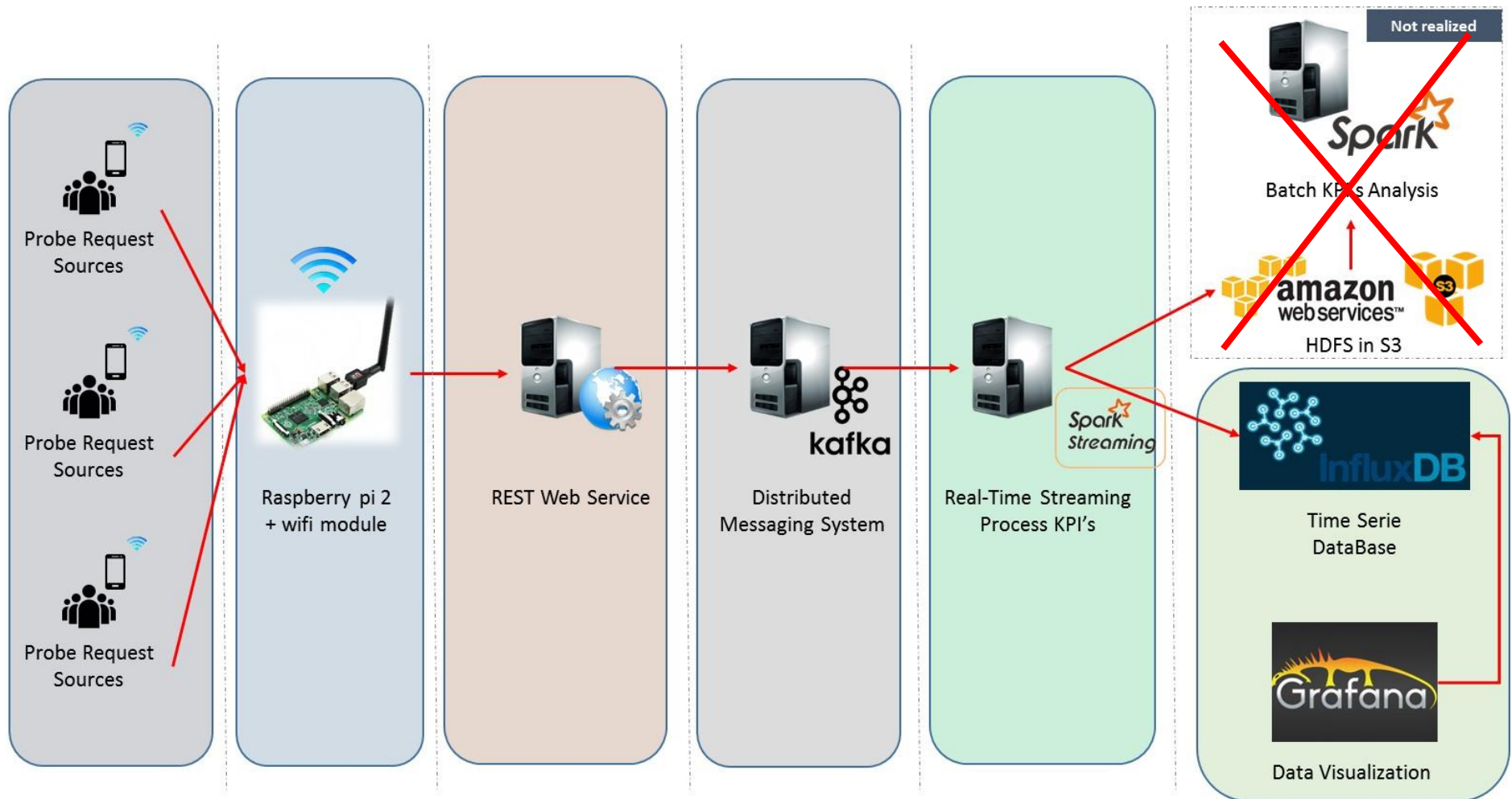
Por último, desde el comienzo he tenido claro que debía compartir mis conocimientos aprendidos y mi experiencia al realizar el proyecto. Por ello, toda la documentación y los futuros pasos dados estarán disponibles libremente en github².

¹ <http://www.gennion.com/>

² <https://github.com/JavierAbascal?tab=repositories>

3. Definición del proyecto y Arquitectura de la solución

En el siguiente esquema se dibuja la arquitectura propuesta para dar solución al objetivo del proyecto:



En general, la arquitectura ha sido diseñada desde un principio para ser escalable horizontalmente ya que en el caso real de uso de la misma las fuentes de datos son muy numerosas y existe un gran número de sensores que recogen esas señales emitidas por los dispositivos (+1000).

De izquierda a derecha, se definen los siguientes elementos:

3.1. Fuente de datos

Las “probe_request” son un tipo de mensaje de saludo del protocolo de comunicaciones inalámbricas que es emitido por todo dispositivo con conectividad WiFi de forma periódica al buscar nuevas redes a las que conectarse. Este mensaje no está cifrado³ e incluye la dirección MAC del dispositivo que es única y permite la identificación dispositivo $\leftarrow \rightarrow$ persona.

No obstante, cabe recordar que todo dispositivo con conectividad WiFi es capaz de emitir “probe_request”. Por ejemplo: portátiles, tablets, Smart TV’s o routers. Debido a que la mayoría de la información a extraer es sobre las personas y no sobre dispositivos inmóviles, será necesario filtrar estos dispositivos con diferentes heurísticas. Para más información sobre esta fuente de datos se recomienda acudir a la bibliografía .

3.2. Sensor

Para ser capaz de capturar las señales emitidas por las fuentes de datos es necesario un componente hardware y un componente software. Por un lado, el sensor debe disponer de un conector de red inalámbrico + antena cuyo chipset permita el modo de monitorización (monitor mode). Por otro lado, es necesario instalar paquetes software que sean capaces de trabajar a niveles de capa de red en el sensor. Ejemplos de este software son Wireshark o tcpdump.

En nuestro caso, hemos utilizado una Raspberry Pi 2 con el OS Ubuntu Mate 15.04, una tarjeta de red inalámbrica TP-Link y los paquetes de tcpdump para capturar las tramas. Para ello, se han generado una serie de scripts (ver código) que generan un archivo con todas las tramas capturadas cada 10 segundos, lo comprime y lo envía al Webservice.

Como ejemplo, se muestra a continuación las líneas de código ya filtradas (la trama original capturada con la instrucción tcpdump es mucho más larga) de un fichero capturado:

```
000001;64:A3:CB:C2:54:48;2015-06-25;20:18:42.535548;-76dB;Broadcast;""
000001;4C:7C:5F:14:FB:D4;2015-06-25;20:18:42.575918;-70dB;Broadcast;"Starbucks-Clientes"
000001;F0:DB:F8:9B:E9:D4;2015-06-25;20:18:42.592240;-67dB;Broadcast;""
000001;4C:7C:5F:14:FB:D4;2015-06-25;20:18:42.605052;-66dB;Broadcast;"Starbucks-Clientes"
```

³ El mensaje no es cifrado ya que emisor y receptor todavía no se conocen

Donde los campos de izquierda a derecho son:

- Id_sensor
- MAC Address
- Timestamp (micro segundos)
- BSSID
- TAG SSID

Nuevamente, se recomienda acudir a la bibliografía en caso de querer tener un mejor conocimiento sobre el software de captura así como las tramas recibidas

3.3. Webservice

El Webservice es el agente que sirve de interconexión entre el proceso de ingestión de los datos y los sensores. Éste componente está encargado de escuchar las peticiones POST de los sensores, recibir los ficheros comprimidos, guardarlos temporalmente, leerlos línea a línea comprobando que se cumple el formato de la trama y no hay errores en ella y enviar línea a línea al *topic* creado en el sistema de colas Kafka.

Por lo anterior, se ha diseñado un REST Webservice implementado en Java que contiene la API Producer Java de Kafka. Para la seguridad, se ha decidido utilizar (temporalmente y por simplicidad) un token fijo por sensor.

3.4. Sistema de colas

El Sistema de colas es un elemento importante e indispensable para el caso de uso de esta arquitectura. Las colas nos permiten conectar las comunicaciones asíncronas provenientes de los diferentes sensores⁴ con las comunicaciones síncronas del procesamiento de datos en Streaming posterior.

La elección de Kafka ha sido realizada por su escalabilidad, simpleza, robustez y una gran integración con Spark. El *topic* creado “kafkaPrueba” guarda los mensajes durante 48h y tiene un factor de réplica unidad y una única partición

⁴ Por factores económicos, en este proyecto se ha utilizado un único sensor. Sin embargo, el diseño está pensado para soportar muchos sensores generadores de datos

3.5. Procesamiento en Streaming

Spark ha sido la tecnología elegida frente a Storm para satisfacer los requisitos de monitorización de un espacio inteligente en tiempo real. El motivo de esta decisión se debe a que la exigencia de “tiempo real” en este caso es de una actualización cada 20-60 segundos (no es necesario y no tiene sentido una tasa de refresco mayor).

Spark ha sido configurado para realizar micro-batches de análisis cada 20 segundos. En cada una de estas tareas programadas, el sistema:

1. Lee de la fuente de datos de Kafka las muestras “probe_request” capturadas en ese tiempo por los sensores
2. Parsea los datos y los cachea en un *dataframe* haciendo uso de Spark-SQL
3. Realiza unas *queries* de agregación sobre ese dataframe
4. Hace uso de la API de Influx para guardar la agregación generada en la base de datos (InfluxDB)
5. Guarda el *raw data* en AWS S3 como back_up para futuros análisis
6. Comprueba las direcciones MACs recibidas y el último instante de tiempo donde se ha detectado a las personas que viven en el piso (David, Miguel o Javier) para saber si acaban de llegar y activar el sistema de alarma que enviará una notificación (email) de llegada al piso

Adicionalmente, Spark ha sido configurado para que guarde un punto de recuperación frente a caídas inesperadas.

3.6. Base de datos

La naturaleza del dato analizado en este proyecto es de serie temporal. Por ello, se ha decidido utilizar como BB.DD principal a InfluxDB cuya principal característica es que está orientada a series temporales y facilita las consultas sobre este tipo de series. InfluxDB es una base de datos Open Source de muy nueva creación y que permite el cálculo de diferentes agregaciones según se van insertando los datos (*continuing queries*). Influx ha sido diseñada desde un principio para escalar horizontalmente y en alta disponibilidad⁵ y encaja perfectamente en la realización de este proyecto

Por último, no debemos olvidar que aunque por cuestiones de tiempo no ha sido posible realizar análisis más complejos de la monitorización inteligente del espacio físico, los datos están siendo guardados en el HDFS Amazon S3 con el fin de poder realizar a futuro análisis más complejos sobre largos periodos de tiempo.

⁵ La configuración de alta disponibilidad de InfluxDB se encuentra en fase BETA

3.7. Visualización de datos

La visualización de datos de este proyecto es el lugar donde mostrar los resultados de todo lo realizado anteriormente y el lugar de comprobación de que todo está funcionando como debe y los datos están siendo cargados en InfluxDB. Se ha pensado en él desde el principio y es uno de los elementos más importantes en todo proyecto de Big Data (Sin visualización, no hay resultados)

Grafana ha sido la herramienta elegida para crear el Dashboard objetivo y mostrar la monitorización inteligente realizada en el espacio físico. Grafana es otra herramienta Open Source y posee una muy buena integración con la base de datos InfluxDB

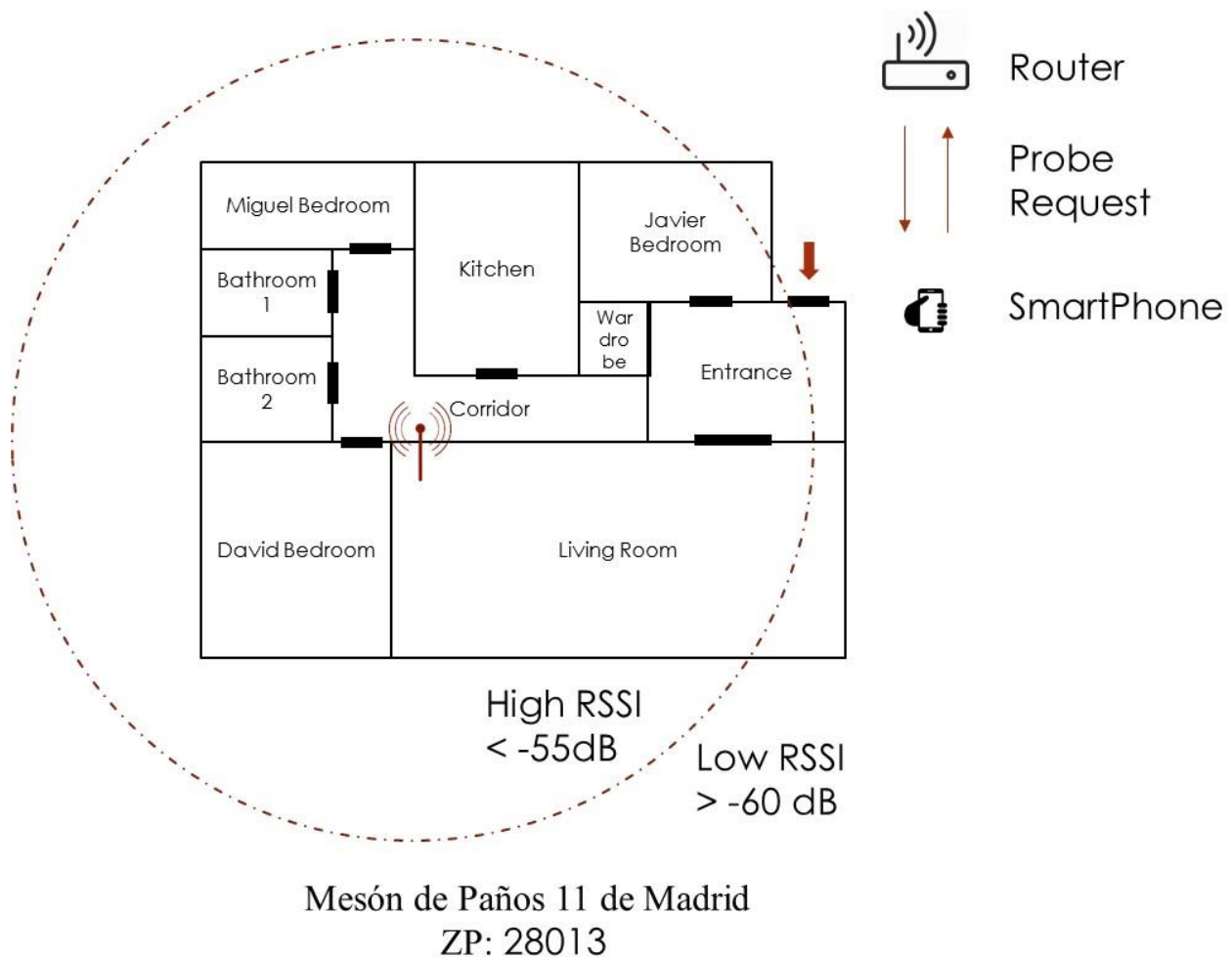
3.8. Despliegue actual del proyecto

El proyecto se encuentra actualmente corriendo en máquinas ofrecidas por los servicios de Amazon Irlanda a excepción de la Raspberry Pi 2 (el sensor) que se encuentra en la posición más centrada del piso que hemos encontrado⁶ (Una silla del salón) y con conexión a nuestra red WiFi doméstica



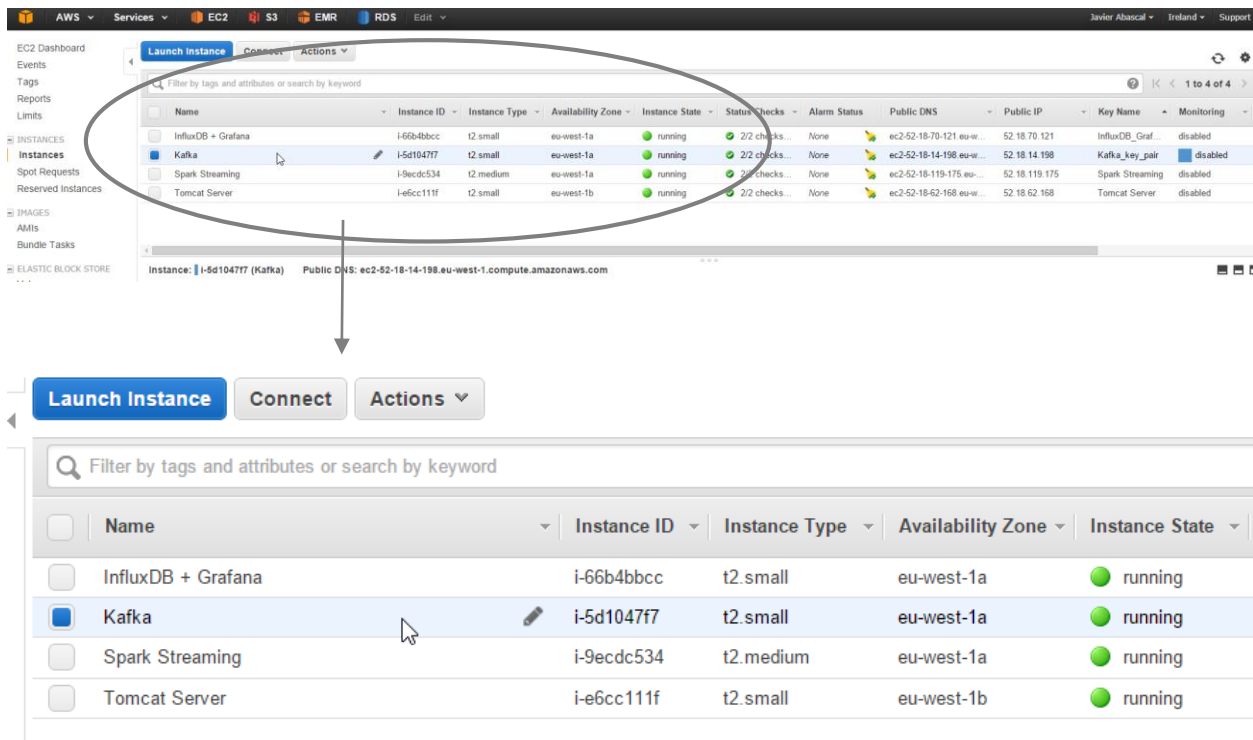
⁶ Pese a que no se ha explicado en profundidad, la posición del sensor es muy importante debido a que influye en los niveles de señal recibidos en las lecturas. Además, la antena de recepción de la tarjeta de red conectada a la placa raspberry es omnidireccional, es decir, captura con la misma ganancia en todas direcciones y puesto que el espacio físico objetivo a monitorizar es el piso, una ubicación central garantiza una buena distribución en estos niveles de señal recibidos.

Como símil, se podría pensar en el router WiFi de casa, que se intenta colocar en un punto central de la casa para ofrecer la mayor cobertura posible



En la siguiente imagen se puede ver el esquema elegido y el tipo de máquina para cada uno de los componentes del proyecto:

- Una máquina t2.small con Amazon Linux para el RESTWebservice levantado en tomcat8
- Una máquina t2.small con Ubuntu para levantar Kafka (0.8.2.1) y ZooKeeper (Aunque esta máquina está sobredimensionada, la instalación de Kafka exige 2GB RAM)
- Una máquina t2.medium con Ubuntu donde corre la tarea de Spark Streaming (1.3.1)
- Una máquina t2.small con Ubuntu para InfluxDB (0.8) y el componente visualizador Grafana (2.0)



A continuación, se incluyen algunas capturas de pantallas de los logs / paneles webs de monitorización comprobados durante la realización del proyecto que han sido claves para el desarrollo y entendimiento del mismo:

Logs de envío Raspberry Pi 2

```
MasterBigData_macCounter scanning
magictaly@magictaly-desktop:~/work$ cd scanning/
magictaly@magictaly-desktop:~/work/scanning$ pwd
/home/magictaly/work/scanning
magictaly@magictaly-desktop:~/work/scanning$ ls -lah
total 1,8M
drwxr-xr-x 3 root root 4,0K jul 22 23:16 .
drwxrwxr-x 4 magictaly magictaly 4,0K jul 22 08:15 ..
-rw-r--r-- 1 root root 0 jul 22 23:16 dump_2015-07-22-23:16:52
-rw-r--r-- 1 root root 1,8M jul 22 23:16 log
drwxr-xr-x 2 root root 4,0K jul 22 23:16 upload
magictaly@magictaly-desktop:~/work/scanning$ tail -f log
2015-07-22_23-16-33: Compressing probe 2015-07-22_23-16-32_000001.log ...
2015-07-22_23-16-33: Moving /home/magictaly/work/scanning/upload/probe_2015-07-22_23-16-32_000001.log.gz to server...
2015-07-22_23-16-42: Generating text dump on /home/magictaly/work/scanning/dump_2015-07-22-23:16:32 ...
reading from file /home/magictaly/work/scanning/dump_2015-07-22-23:16:32, link-type IEEE802_11_RADIO (802.11 plus radiotap header)
2015-07-22_23-16-43: Compressing probe 2015-07-22_23-16-42_000001.log ...
2015-07-22_23-16-43: Moving /home/magictaly/work/scanning/upload/probe_2015-07-22_23-16-42_000001.log.gz to server...
2015-07-22_23-16-52: Generating text dump on /home/magictaly/work/scanning/dump_2015-07-22-23:16:42 ...
reading from file /home/magictaly/work/scanning/dump_2015-07-22-23:16:42, link-type IEEE802_11_RADIO (802.11 plus radiotap header)
2015-07-22_23-16-53: Compressing probe 2015-07-22_23-16-52_000001.log ...
2015-07-22_23-16-53: Moving /home/magictaly/work/scanning/upload/probe_2015-07-22_23-16-52_000001.log.gz to server...
2015-07-22_23-17-02: Generating text dump on /home/magictaly/work/scanning/dump_2015-07-22-23:16:52 ...
reading from file /home/magictaly/work/scanning/dump_2015-07-22-23:16:52, link-type IEEE802_11_RADIO (802.11 plus radiotap header)
2015-07-22_23-17-03: Compressing probe 2015-07-22_23-17-02_000001.log ...
2015-07-22_23-17-03: Moving /home/magictaly/work/scanning/upload/probe_2015-07-22_23-17-02_000001.log.gz to server...
2015-07-22_23-17-12: Generating text dump on /home/magictaly/work/scanning/dump_2015-07-22-23:17:02 ...
reading from file /home/magictaly/work/scanning/dump_2015-07-22-23:17:02, link-type IEEE802_11_RADIO (802.11 plus radiotap header)
2015-07-22_23-17-13: Compressing probe 2015-07-22_23-17-12_000001.log ...
2015-07-22_23-17-13: Moving /home/magictaly/work/scanning/upload/probe_2015-07-22_23-17-12_000001.log.gz to server...
2015-07-22_23-17-22: Generating text dump on /home/magictaly/work/scanning/dump_2015-07-22-23:17:12 ...
reading from file /home/magictaly/work/scanning/dump_2015-07-22-23:17:12, link-type IEEE802_11_RADIO (802.11 plus radiotap header)
2015-07-22_23-17-23: Compressing probe 2015-07-22_23-17-22_000001.log ...
2015-07-22_23-17-23: Moving /home/magictaly/work/scanning/upload/probe_2015-07-22_23-17-22_000001.log.gz to server...
```

Logs del REST Webservice recibiendo los datos enviados por el sensor e introduciéndolos en Kafka

```
[root@ip-172-31-11-140 ec2-user]# tail -f /usr/share/tomcat8/logs/catalina.out
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:28.325636;-76dB;Broadcast;"JAZZTEL_52AD"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:28.389636;-77dB;Broadcast;"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:28.452476;-79dB;Broadcast;"JAZZTEL_52AD"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:28.453164;-80dB;Broadcast;"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:28.980865;-77dB;Broadcast;"JAZZTEL_52AD"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:28.981584;-77dB;Broadcast;"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:29.108197;-81dB;Broadcast;"
Total Time: 0.079897467
  ReadFile Time: 7.161790000000001E-4
  SendKafka Time: 0.079181288
File uploaded correctly to : /tmp/probe_2015-07-22_21-14-40_000001.log.gz

000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:29.637756;-78dB;Broadcast;"JAZZTEL_52AD"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:29.638435;-79dB;Broadcast;"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:29.703037;-77dB;Broadcast;"JAZZTEL_52AD"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:29.704449;-77dB;Broadcast;"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:29.764587;-80dB;Broadcast;"JAZZTEL_52AD"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:30.294503;-77dB;Broadcast;"JAZZTEL_52AD"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:30.295210;-77dB;Broadcast;"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:30.357882;-77dB;Broadcast;"JAZZTEL_52AD"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:30.358628;-77dB;Broadcast;"
000001;00:61:71:42:FF:CF;2015-07-22;21:14:30.404884;-82dB;Broadcast;"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:30.421299;-78dB;Broadcast;"JAZZTEL_52AD"
000001;8C:C5:E1:89:4C:17;2015-07-22;21:14:30.422012;-79dB;Broadcast;"
```


Logs de conexiones de Kafka / Zookeeper

```
[ec2-user@ip-172-31-44-0 logs]$ tail -f server.log
[2015-07-22 21:21:40,130] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:21:40,239] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:21:40,429] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:21:40,431] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:21:41,011] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:21:41,013] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:21:44,155] INFO Closing socket connection to /52.18.62.168. (kafka.network.Processor)
[2015-07-22 21:21:44,233] INFO Closing socket connection to /52.18.62.168. (kafka.network.Processor)
[2015-07-22 21:21:54,168] INFO Closing socket connection to /52.18.62.168. (kafka.network.Processor)
[2015-07-22 21:21:54,241] INFO Closing socket connection to /52.18.62.168. (kafka.network.Processor)

[2015-07-22 21:22:00,122] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:22:00,235] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:22:00,441] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:22:00,442] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:22:00,998] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)
[2015-07-22 21:22:01,009] INFO Closing socket connection to /47.62.79.216. (kafka.network.Processor)

[2015-07-22 21:22:04,222] INFO Closing socket connection to /52.18.62.168. (kafka.network.Processor)
[2015-07-22 21:22:04,304] INFO Closing socket connection to /52.18.62.168. (kafka.network.Processor)
```

Panel Web del clúster StandAlone de Spark recién creado

 **Spark Master at spark://ip-172-31-37-185:7077**

URL: spark://ip-172-31-37-185:7077
 RESTURL: spark://ip-172-31-37-185:6066 (cluster mode)
 Workers: 3
 Cores: 3 Total, 0 Used
 Memory: 3.0 GB Total, 0.0 B Used
 Applications: 0 Running, 0 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

Workers


| Worker Id | Address | State | Cores | Memory |
|-------------------------------------------------------------------------|---------------------------------------------------|-------|------------|------------------------|
| worker-20150722212326-ip-172-31-37-185.eu-west-1.compute.internal:53818 | ip-172-31-37-185.eu-west-1.compute.internal:53818 | ALIVE | 1 (0 Used) | 1024.0 MB (0.0 B Used) |
| worker-20150722212329-ip-172-31-37-185.eu-west-1.compute.internal:36045 | ip-172-31-37-185.eu-west-1.compute.internal:36045 | ALIVE | 1 (0 Used) | 1024.0 MB (0.0 B Used) |
| worker-20150722212332-ip-172-31-37-185.eu-west-1.compute.internal:45520 | ip-172-31-37-185.eu-west-1.compute.internal:45520 | ALIVE | 1 (0 Used) | 1024.0 MB (0.0 B Used) |

Notificación de Alarma de Llegada

New FlatMate at Home



raspberrypi2.sensor.system@gmail.com (raspberrypi2.sensor.system@gmail.com) 

To: javier.abascal@hotmail.com, davsucar@gmail.com, a_mi_no_mestafes@hotmail.com 

Miguel Angel Amo has arrived at 2015/07/22 19:34:19

Panel Web de InfluxDB

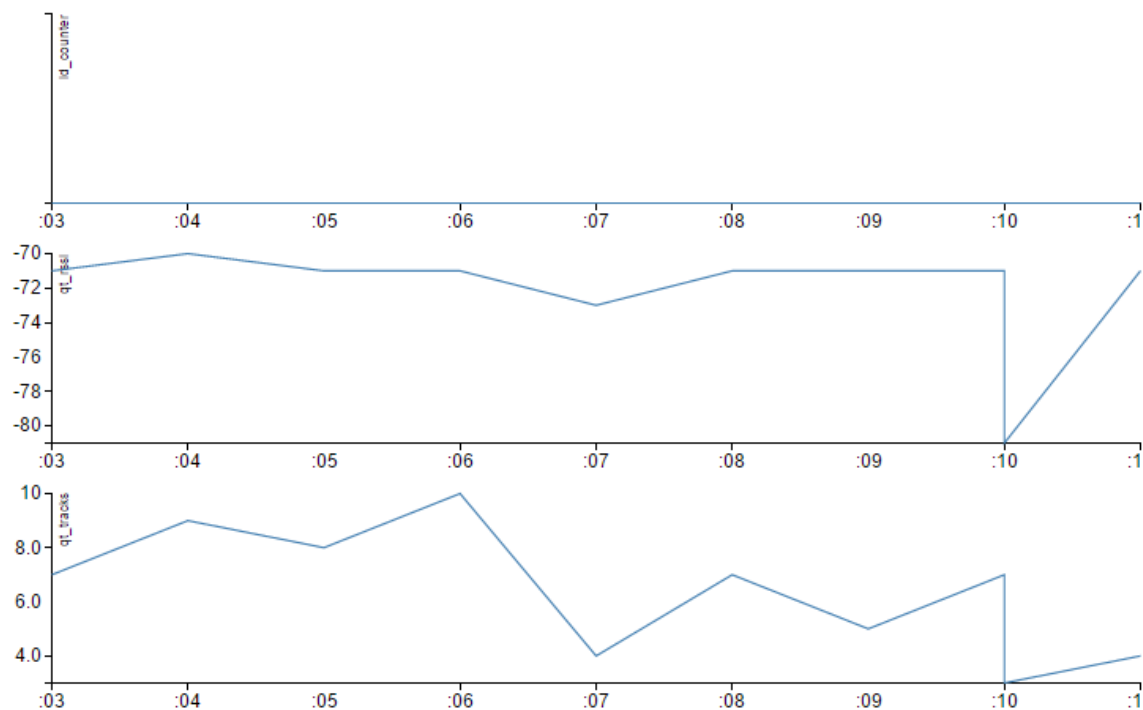

Databases Cluster Admins Cluster
52.18.70.121: Disconnect

Query

InfluxDB features a SQL-like query language

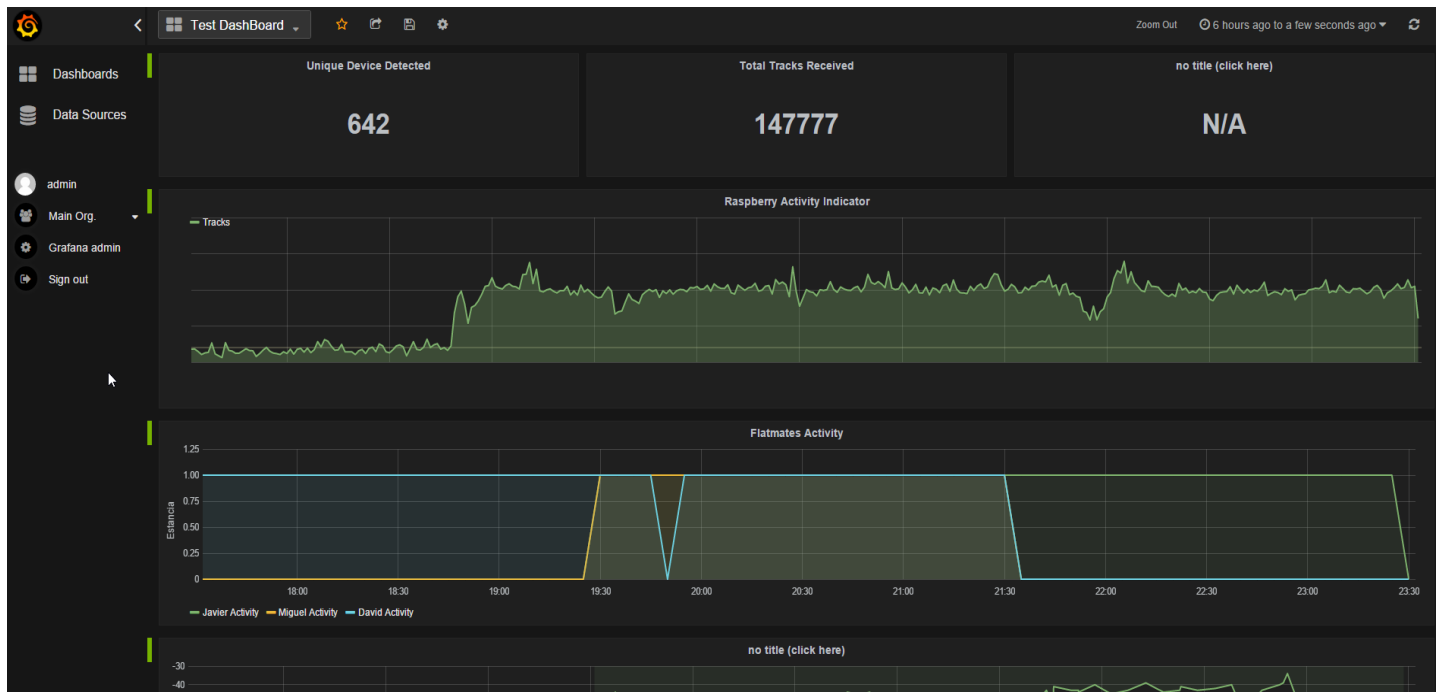
Execute Query

track_seconds



| time | sequence_number | id_counter | co_mac | qt_rssi | qt_tracks |
|---------------|-----------------|------------|-------------------|---------|-----------|
| 1437600551000 | 575010001 | 1 | 8C:C5:E1:89:4C:17 | -71 | 4 |
| 1437600550000 | 575030001 | 1 | C0:EE:FB:21:52:6A | -81 | 3 |
| 1437600550000 | 575020001 | 1 | 8C:C5:E1:89:4C:17 | -71 | 7 |
| 1437600549000 | 575040001 | 1 | 8C:C5:E1:89:4C:17 | -71 | 5 |
| 1437600548000 | 575050001 | 1 | 8C:C5:E1:89:4C:17 | -71 | 7 |

DashBoad Grafana



4. Resultados y conclusiones

Pese a haber quedado pendiente la realización de un análisis batch con Spark de los datos guardados en el HDFS de Amazon S3, los resultados y el alcance obtenido en este proyecto superan ampliamente mis expectativas iniciales. Hemos conseguido una monitorización inteligente en tiempo real de un espacio físico a través de un panel Dashboard y un sistema de alarmas que tiene muchas aplicaciones prácticas a un caso real de **negocio**.

Actualmente, mis compañeros de piso y yo recibimos un correo electrónico cada vez que alguno de nosotros entra en casa, y disponemos de un panel web donde somos capaces de visualizar a tiempo real (20 segundos de actualización):

1. El número de dispositivos distintos en el alcance del sensor
2. El número de dispositivos distintos dentro del espacio físico (el piso)
3. El tiempo de estancia de los inquilinos del espacio físico (Javier, David, Miguel)
4. Los tiempos de llegada / salida del espacio físico
5. La serie temporal de actividad en la zona (a mayor actividad, mayor tráfico humano)
6. Los niveles de intensidad recibidos y la serie temporal de detección de cualquiera MAC capturada

Adicionalmente, y con el uso de pequeñas consultas, agregaciones o adiciones de más sensores se puede llegar a obtener datos y análisis mucho más ricos sobre la utilización de un espacio físico (frecuencia de visita de los dispositivos, diferenciación máquina/persona, tiempos de estancia, movimiento de las personas dentro del espacio...). Los resultados mostrados aquí no son más que la punta del iceberg de las aplicaciones reales de esta tecnología⁷

También es muy importante recordar que toda esta monitorización se produce de forma anónima y pasiva (para aplicaciones reales la MAC Address es Hasheada), es decir, con el único requisito de que la persona monitorizada lleve encima un smartphone con el WiFi activado.

En el ámbito personal, estoy muy satisfecho de haber sido capaz de desarrollar en un corto periodo de tiempo una arquitectura completa con tecnologías desconocidas por mí hace tan sólo 6 meses y haber abarcado desde la generación de los datos hasta la visualización de los mismos. Por último, quiero decir que esta experiencia me ha servido para aprender muchísimo y conocer a un mayor nivel de detalle los pros y contras de las tecnologías. Ha sido un reto difícil, pero la recompensa ha merecido la pena.

Para terminar, me gustaría agradecer el *feedback* recibido por mi tutor Iván a la hora de proponer el proyecto y la arquitectura del mismo, a las personas que me han ayudado a resolver los miles de problemas que me han ido surgiendo (David, Mariano y Javier Portugal) y a mis compañeros de piso por cederme sus MACs y el piso para realizar el proyecto

⁷ Actualmente existen numerosas empresas repartidas por todo el mundo y dedicadas casi en exclusiva a este tipo de monitorización. Ejemplos: Gennion Solutions, Euclid Analytics, Bitcarrier o World W Sensing

5. Futuras líneas de trabajo

Aunque el alcance del proyecto se ha acotado para cumplir con los plazos de entrega del proyecto, todavía queda trabajo muy interesante por hacer y con el que se puede aprender mucho. Los siguientes puntos a trabajar serán:

- Solucionar los problemas de estabilidad de las tecnologías (He tenido varias caídas del RESTWebservice e InfluxDB debido a errores varios)
- Aprovechar al máximo el uso de las *continuing queries* de InfluxDB para sacar más indicadores
- Realizar los análisis batch con Spark de los datos con el histórico guardado en AWS S3 e incluir en la estructura una base de datos relacional para guardar los resultados (Ya que los datos generados en este proceso no serán masivos)
- Añadir más sensores al proyecto y generar con ellos nuevos sistemas de alarmas e indicadores. Un ejemplo, sería una alarma de salida del edificio. Para ello se piensa en usar ventanas temporales de datos en Spark Streaming
- Aprender a utilizar una herramienta de integración continua como CircleCI para los Webservices
- Aprender a utilizar los contenedores de *docker*, con el fin de montar todo el sistema en una sola máquina, ahorrar costes y ser capaz de montar desarrollos más rápidamente

6. Bibliografía.

En el siguiente apartado se escriben los enlaces a la información más importante que ha sido consultada a la hora de resolver este proyecto. Muchas de las fuentes no son más que la documentación oficial al respecto. En caso de tener cualquier tipo de duda, por favor, no dude en contactar conmigo en:

Javier.abascal@hotmail.com

6.1 Probe Request (Source Data)

Proceso de asociación Router & Dispositivo:

https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/802.11_Association_process_explained

Tipos de paquetes (mensajes) en el protocolo WLAN 802.11

http://www.wildpackets.com/resources/compendium/wireless_lan/wlan_packet_types

Estudios de monitorización de la actividad en un espacio físico utilizando “probe_request”



Pedestrian
Monitoring System us



Discovering Human
Presence Activities wit



An Experimental
Study of WiFi Probe R

Documentación sobre la instrucción tcpdump:

http://www.tcpdump.org/tcpdump_man.html

6.2 Raspberry pi 2

Documentación oficial:

<https://www.raspberrypi.org/products/raspberry-pi-2-model-b/>

Ubuntu Mate para la raspberry pi 2:

<https://ubuntu-mate.org/raspberry-pi/>

Adaptador de red USB con chipset disponible para modo monitor (capaz de capturar las probe_request):

<http://www.tp-link.es/products/details/?model=tl-wn722n>

6.3 Kafka Producer API (java)

Documentación oficial:

<http://kafka.apache.org/documentation.html>

6.4 Spark Streaming & Spark SQL

Documentación oficial:

<https://spark.apache.org/docs/latest/streaming-programming-guide.html>

<https://spark.apache.org/docs/latest/sql-programming-guide.html>

6.5 InfluxDB & API (Scala)

Documentación oficial:

<https://influxdb.com/>

6.6 Grafana

Documentación oficial:

<http://grafana.org/>