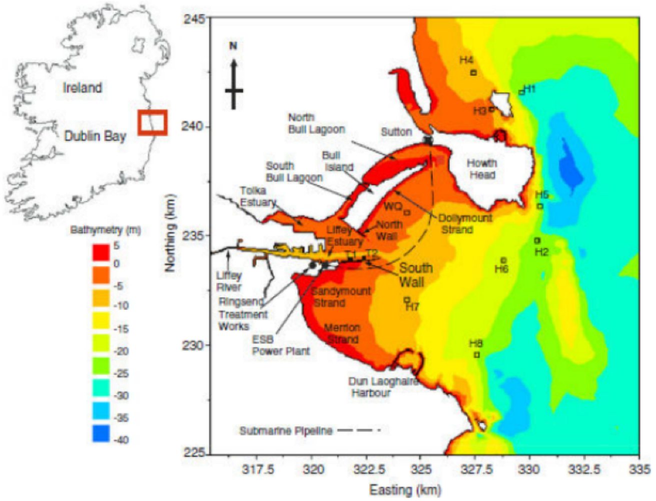
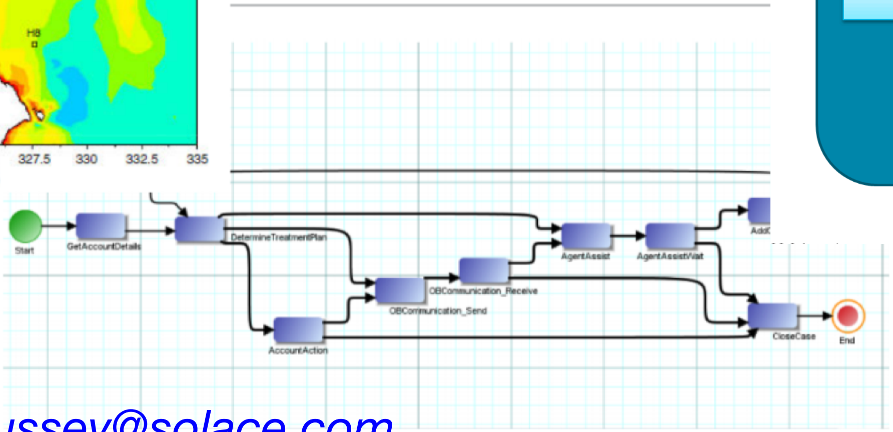
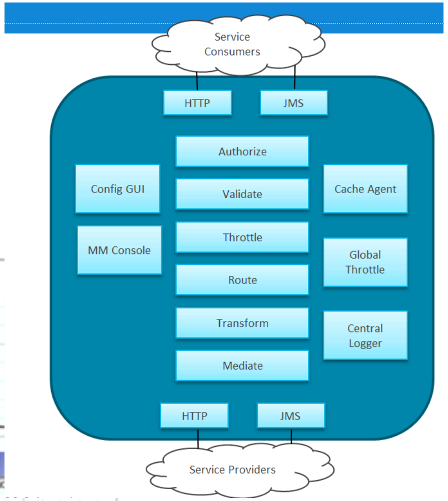
Tasty Topics!

Novel approaches using Topic Filtering

QCon
LONDON







michael.hussey@solace.com
dev.solace.com
https://www.linkedin.com/in/michael-hussey

solace

Pub/Sub revision

- Distributed
- Decoupled
- Fan-in/Fan-out
- Persistence
- Register interest in *Topic*

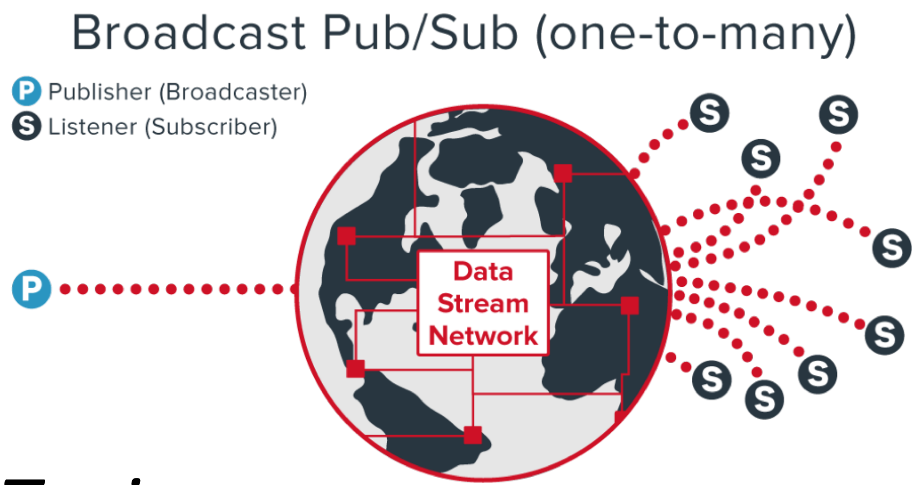
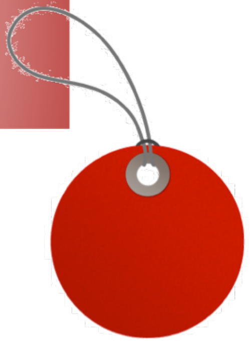


Image credit: pubnub.com

Topics

Topic $\neq Tag$!



food/apple/slices

//slices

=>

food/apple/slices

food/ham/slices

food_apple_slices

List {food_apple_slices, food_ham_slices}

String search?

Who Cares?

- Simpler
- Consistent
- Reduces unnecessary data copies
  - E.g. In IoT reduces unnecessary sensor reads

Use Case 1

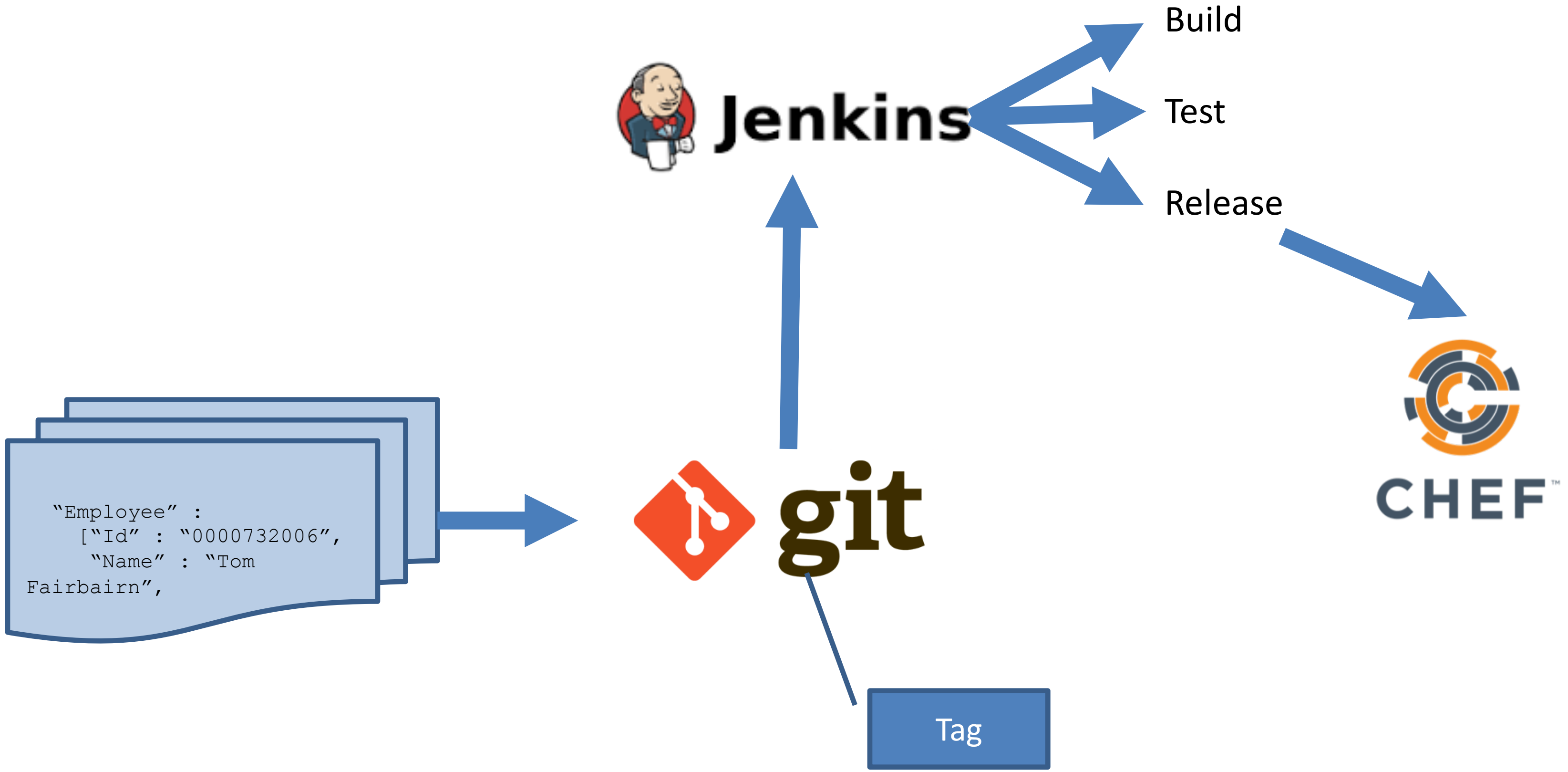
Migrating Your Data Format

Case 1: Migrating Your Data Format

```
{
  “Person” : “Tom”,
  “Team” : “Magicians”,
  “Mobile” : “07746 244422”,
  “EmployeeId” : 6
}
```

```
{
  “Employee” :
    [“Id” : “0000732006”,
     “Name” : “Tom Fairbairn”,
     “PhoneNum” : “+44(0)7746244422”,
     “DirectReports”: [],
     ”ReportsTo”: “Ben Taieb”
    ]
}
```

CI/CD



```
  "Employee" :
    ["Id" : "0000732006",
     "Name" : "Tom
Fairbairn",
```

Data Format – read/write

```
Gson gson = new Gson();
empolyeeData = gson.fromJson(data, employee.class);
```

```
public class employee {
  private String Person;
  private String Team;
  private String Mobile;
  public int EmployeeId;
…
}
```

Tag: v1.0

```
public class employee {
  private class employeeData {
    private String Id;
    private String Name;
    private String PhoneNum;
    private String[] DirectReports;
    private String ReportsTo;
  }
…
}
```

Tag: v2.0

Data Format topic

```
private String versionedTopic =
      “london/employee/json/$GIT_TAG_NAME/[…]”;

session.subscribe(versionedTopic);

producer.send(message, versionedTopic);
```

Use Case 2

Monitoring

Monitoring

A quick diversion

{ REST }

https://ip/endpoint

MQTT.ORG

AMQP

Websockets

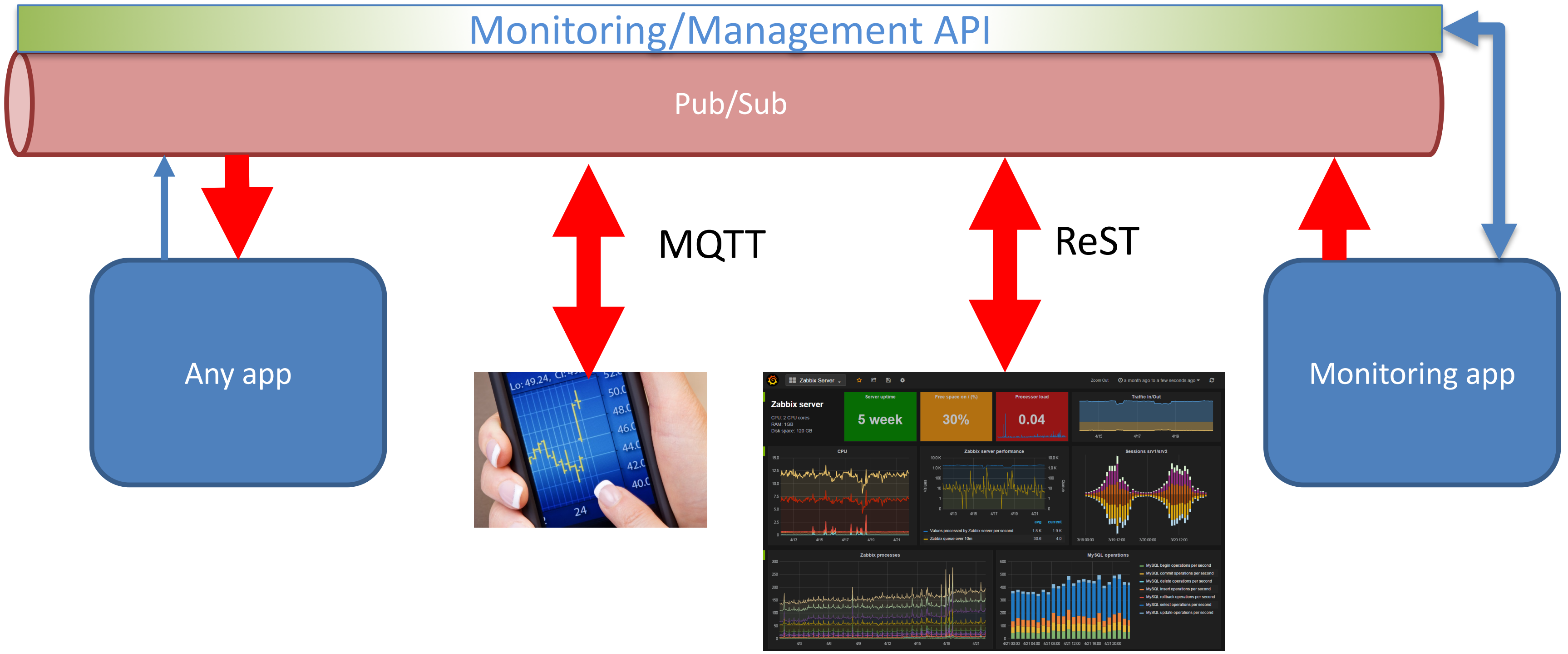
A quick diversion -again

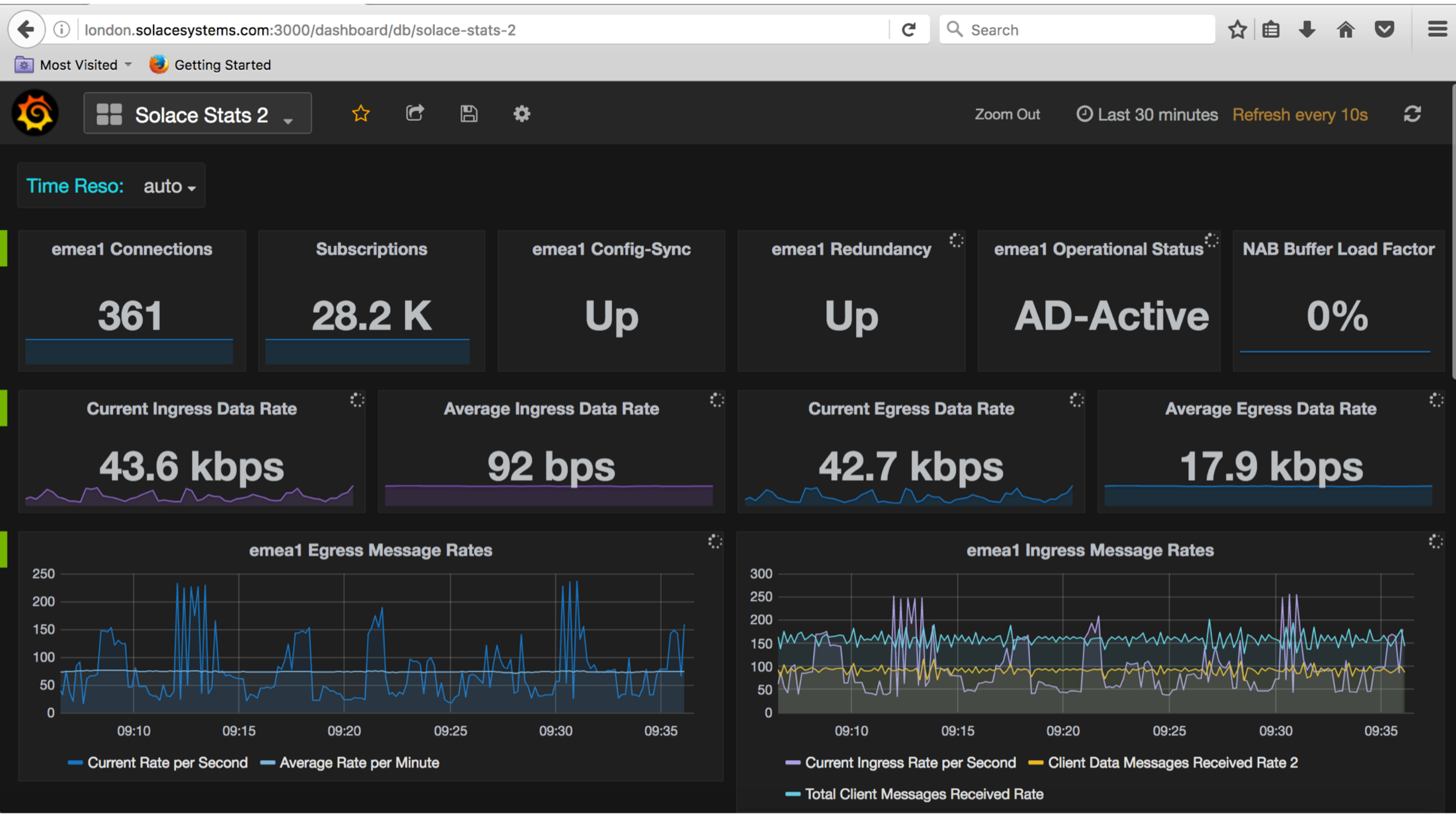
V 1

V 5

V 3

Pub/Sub Monitoring over Pub/Sub!





emea1 Connections: 361

Subscriptions: 28.2 K

emea1 Config-Sync: Up

emea1 Redundancy: Up

emea1 Operational Status: AD-Active

NAB Buffer Load Factor: 0%

Current Ingress Data Rate: 43.6 kbps

Average Ingress Data Rate: 92 bps

Current Egress Data Rate: 42.7 kbps

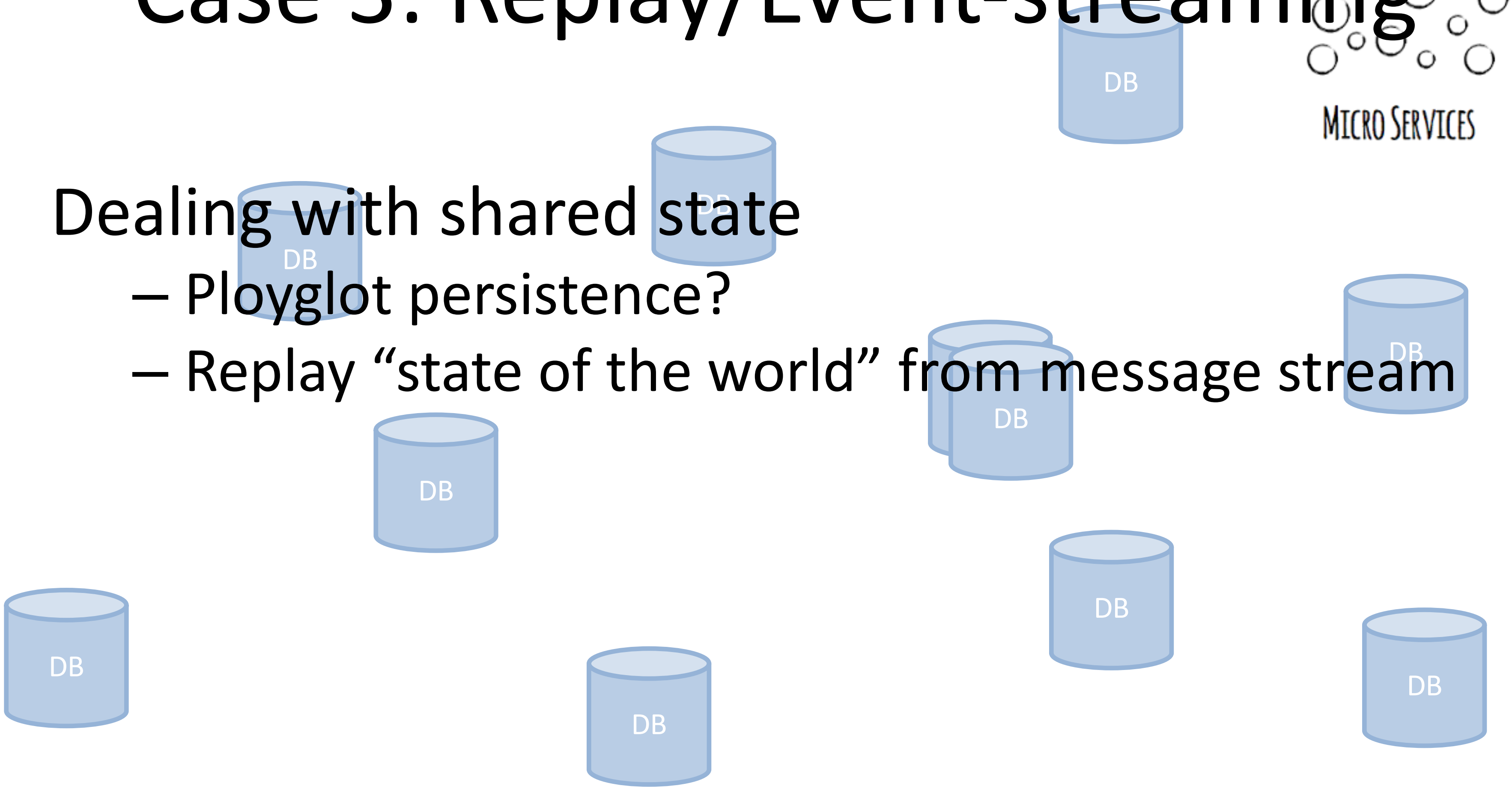
Average Egress Data Rate: 17.9 kbps

emea1 Egress Message Rates
250
200
150
100
50
0
09:10
09:15
09:20
09:25
09:30
09:35
Current Rate per Second
Average Rate per Minute
emea1 Ingress Message Rates
300
250
200
150
100
50
0
09:10
09:15
09:20
09:25
09:30
09:35
Current Ingress Rate per Second
Client Data Messages Received Rate 2
Total Client Messages Received Rate

Use Case 3

Replay

Case 3: Replay/Event-streaming

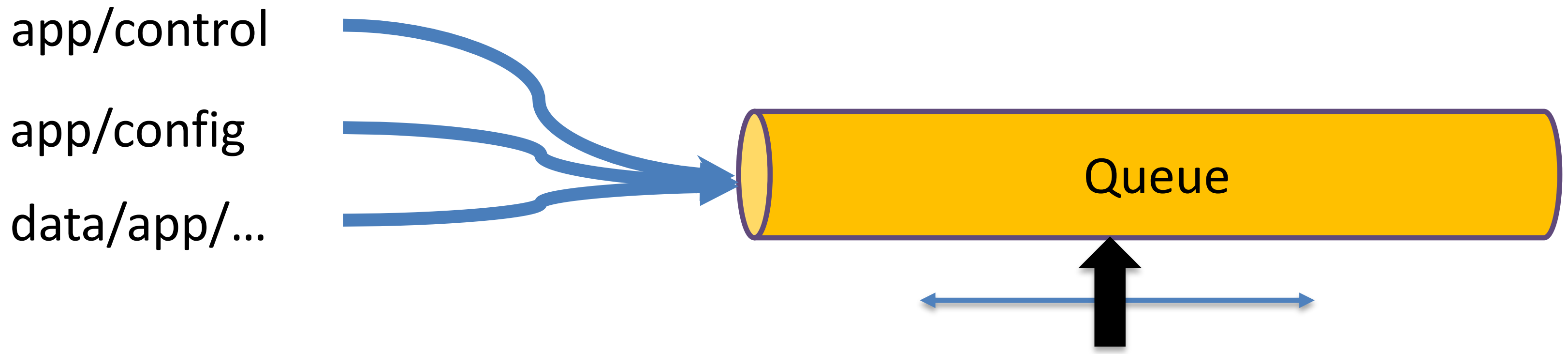


Dealing with shared state

- Ployglot persistence?
- Replay “state of the world” from message stream

Replay – queues that can subscribe

MICRO SERVICES

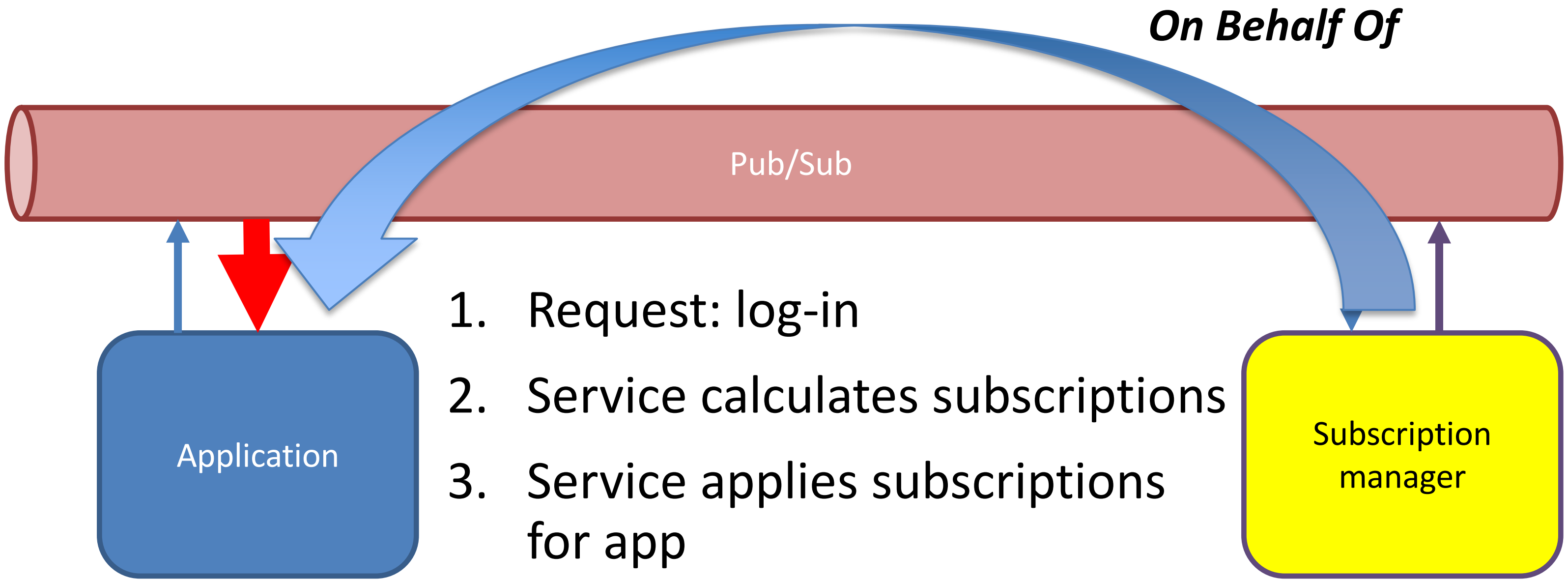


- Queue Browser
- TTL
- LVQ

Use Case 4

Authorisation

Case 4: Authorisation

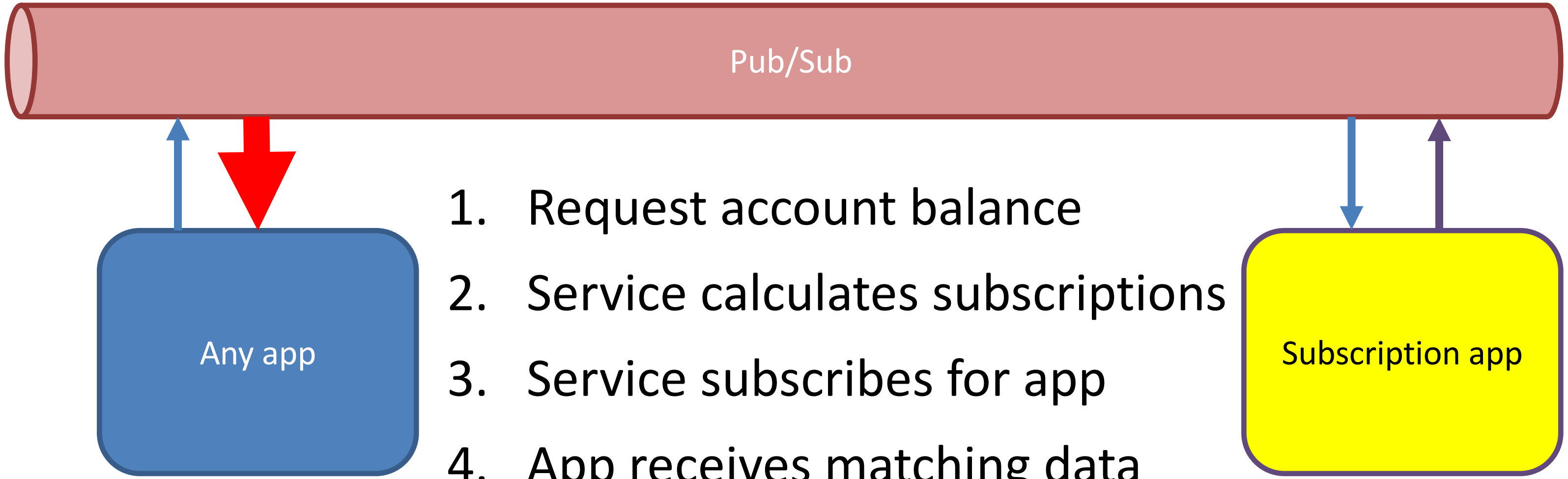


1. Request: log-in
2. Service calculates subscriptions
3. Service applies subscriptions for app
4. Reply with OK
5. App receives matching data

On Behalf Of

- Client has no awareness of topics/services
  - No chance to guess other services
  - No work/exposure at client
- Fully pluggable architecture

Authorisation



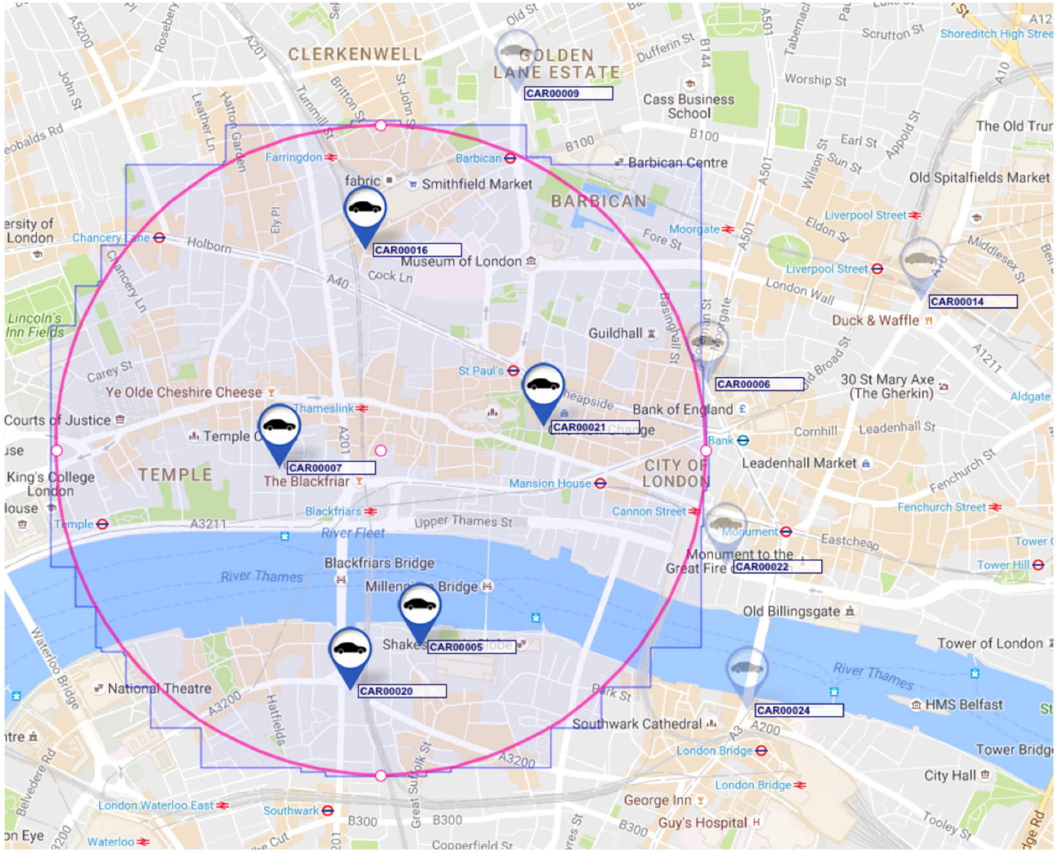
1. Request account balance
2. Service calculates subscriptions
3. Service subscribes for app
4. App receives matching data

Use Case 5

Find the nearest…

Geo-location using topics

Case 2:Find The Nearest… In Real Time



Source: ICOMP 2016, A.L. Lee, Ranged Filtering of Streaming Numeric Data… using Topic-Based Pub/Sub Messaging

Geo-filtering topic

Publish to topic with location:

```
<app>/<type>/<lat>/<long>/<vehicle>/<id>
geo/sim/51.520150/-00.097330/CAR/00021
```

Where is CAR00021?

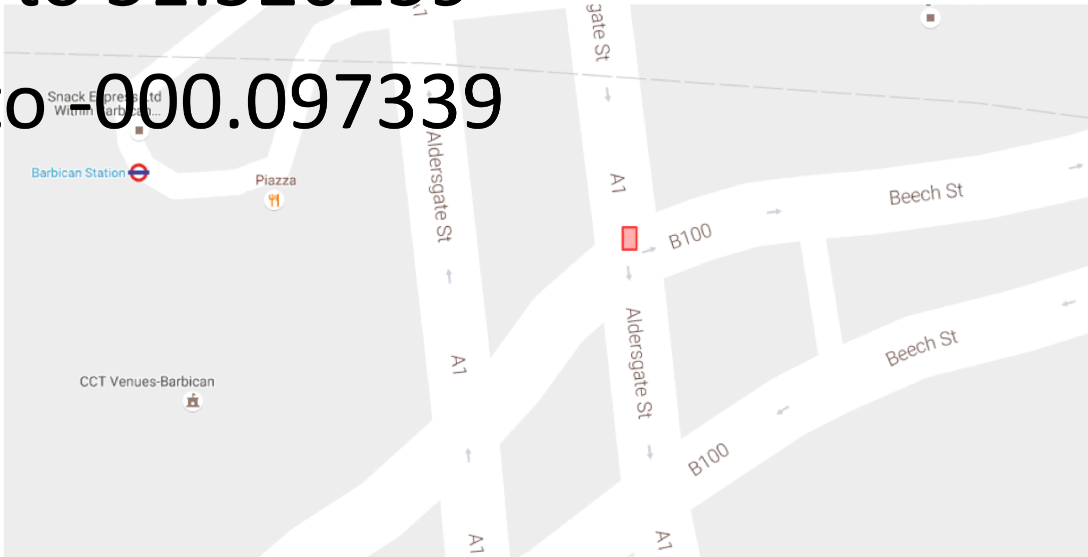
```
subscribe(“geo/sim/*/*/CAR/00021”);
```

Geo-filtering location

```
subscribe(“geo/sim/51.52015*/-00.09733*/>”);
```

Match: lat 51.520150 to 51.520159

long -000.097330 to -000.097339

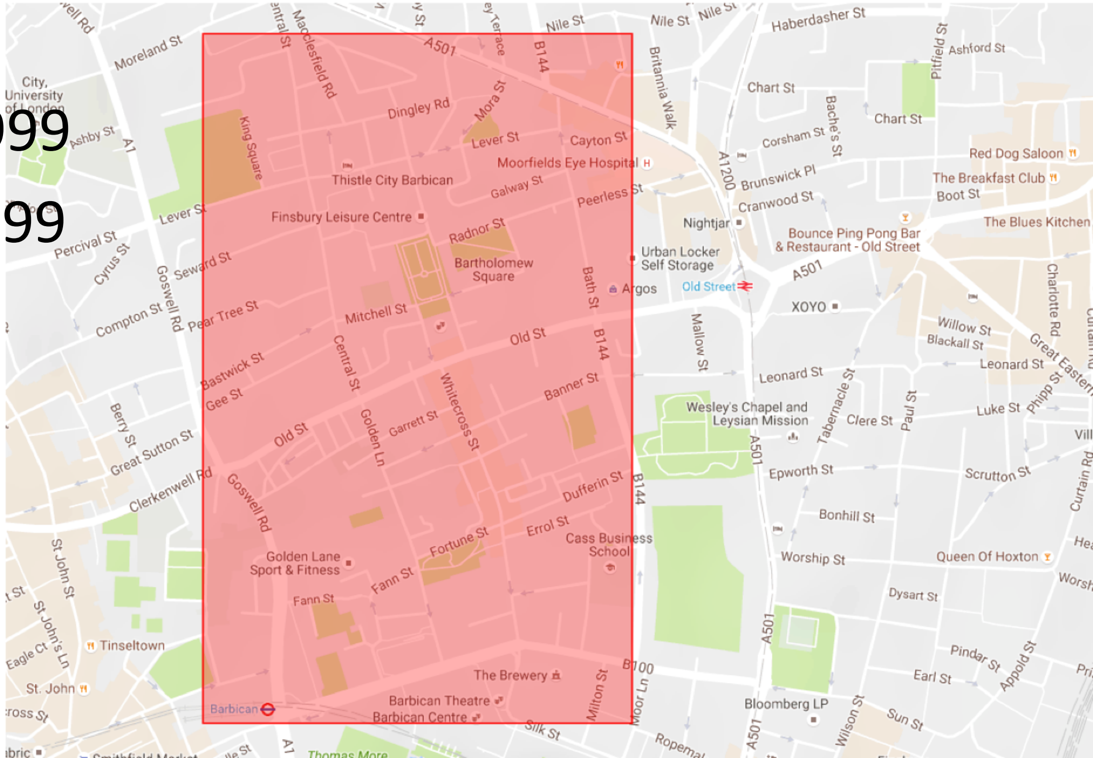


Geo-filtering location

```
subscribe("geo/sim/51.52*/-00.09*/>");
```

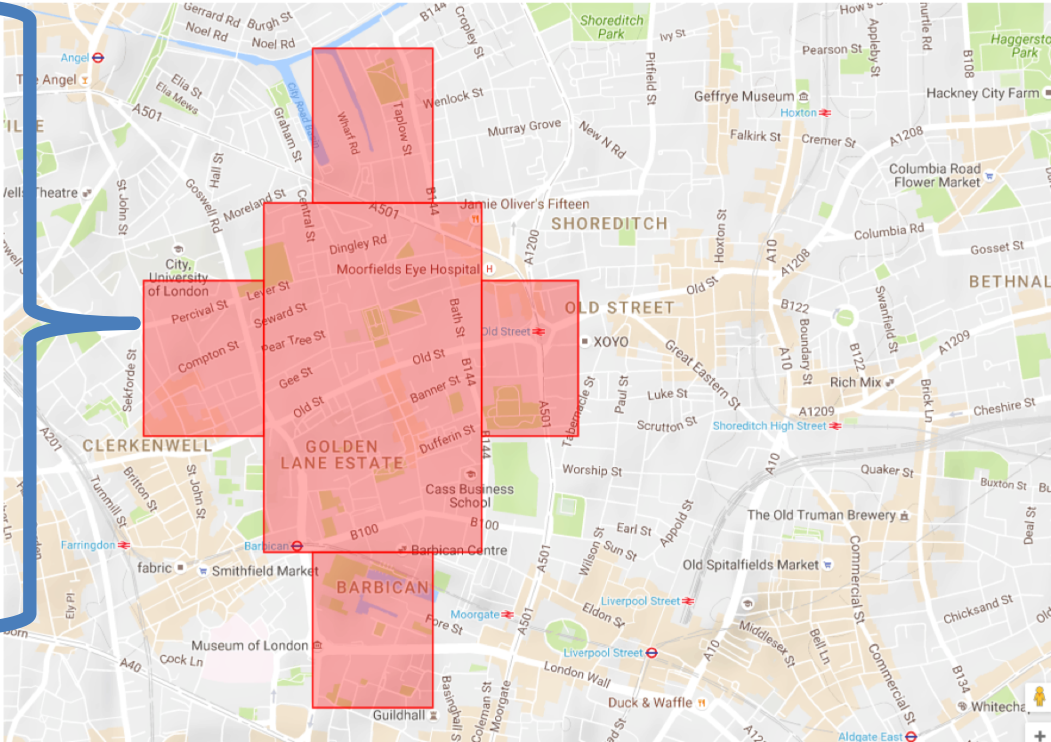
Match: lat 51.520 to 51.529999

long -0.090 to -0.099999



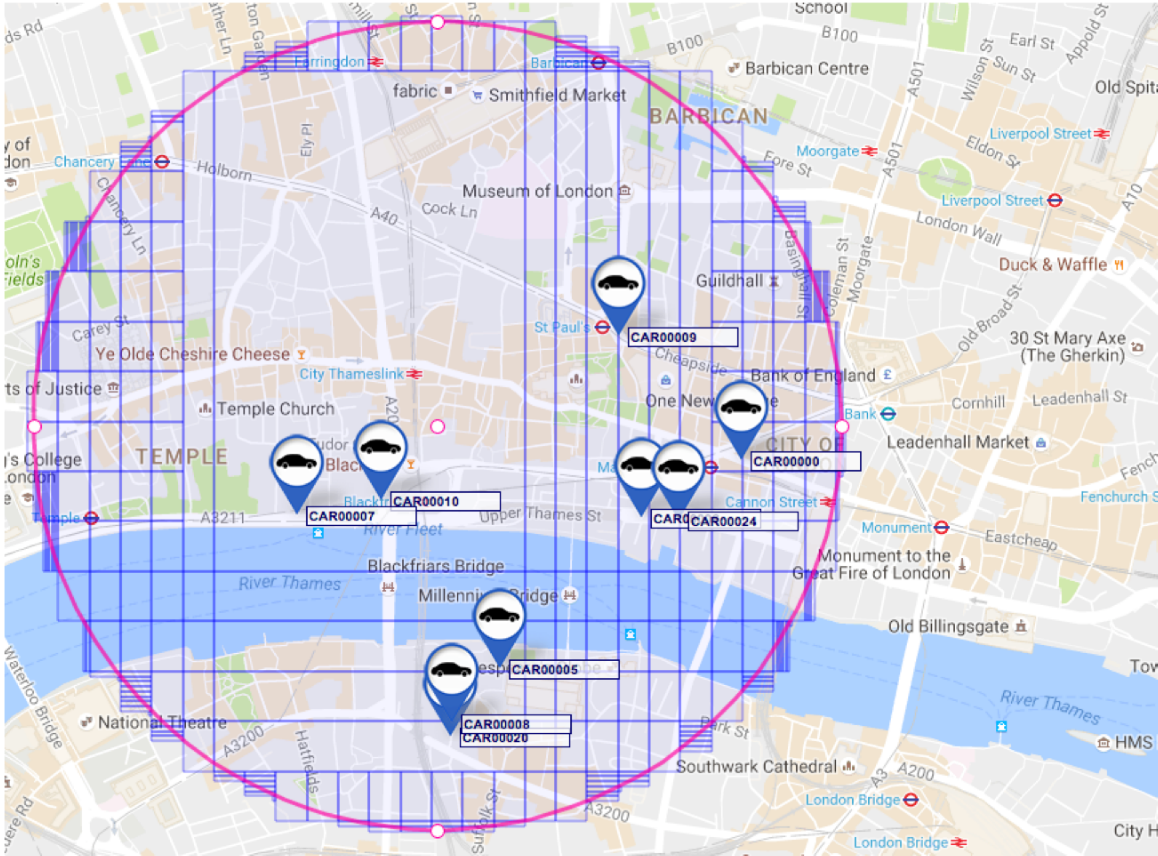
Geo-filtering location

```
subscribe("geo/sim/51.52*/-00.09*/>",

"geo/sim/51.516*/-00.092*/>",
"geo/sim/51.516*/-00.093*/>",
"geo/sim/51.516*/-00.094*/>",
"geo/sim/51.516*/-00.096*/>",

"geo/sim/51.517*/-00.092*/>",
"geo/sim/51.517*/-00.093*/>",
"geo/sim/51.517*/-00.094*/>",
"geo/sim/51.517*/-00.096*/>",

// repeat for 51.518 and .519
);
```



Geo-filtering location

- Create any polygon
  - Accuracy at metre level
  - Circles, arcs…
- Subscriptions generated *once*
- Matches then stream in with no extra computation

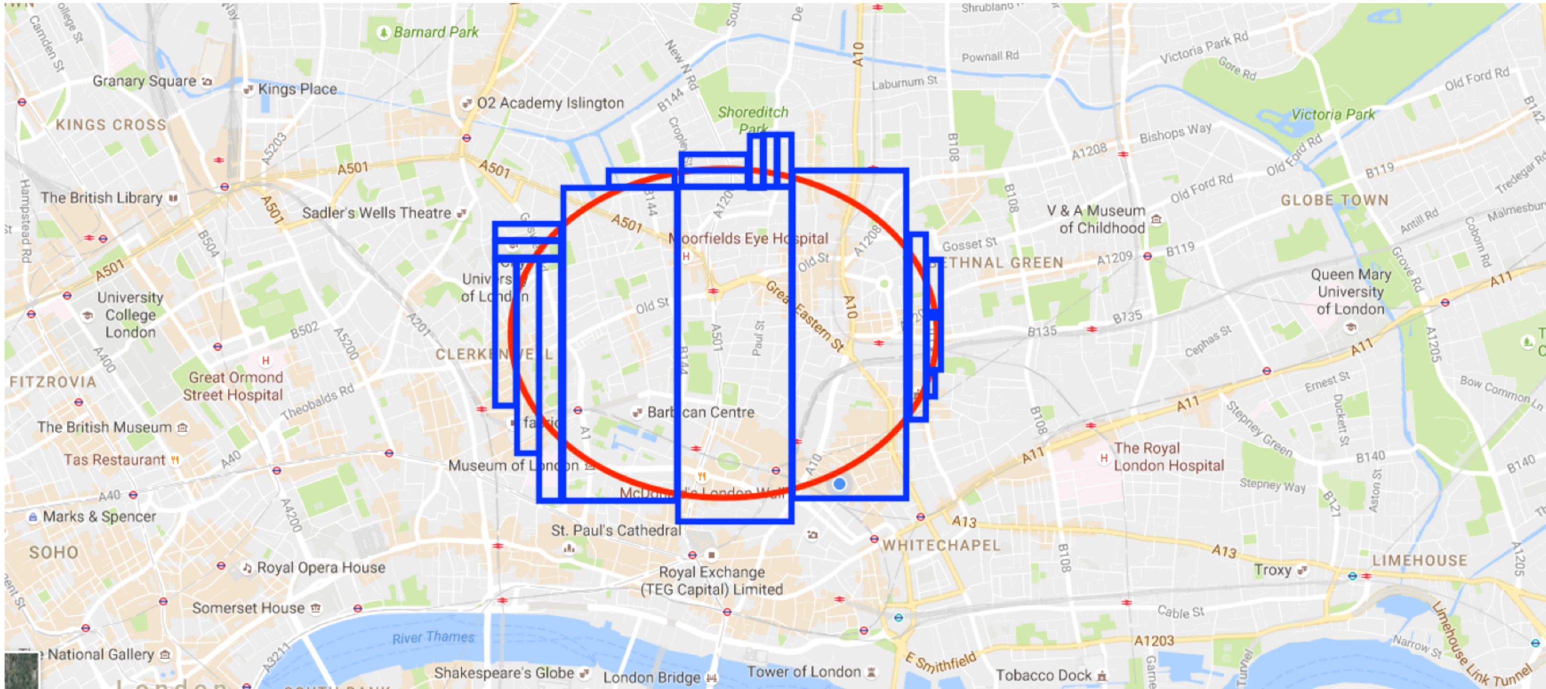


Geo-filtering location algorithm

- Divide space into rectangles aligned to subscriptions
- Throw away rectangles with no match

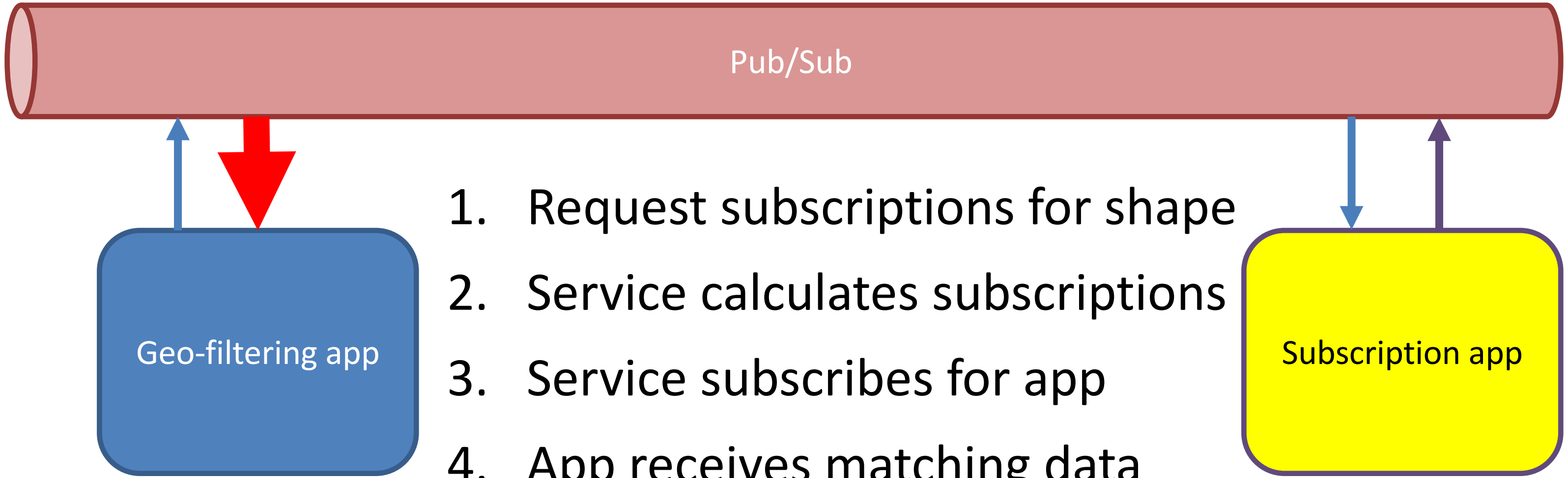
Geo-filtering location algorithm

- Repeat: divide remaining rectangles by 10
- Throw away rectangles with no match



Geo-filtering location algorithm deployment

- Library?



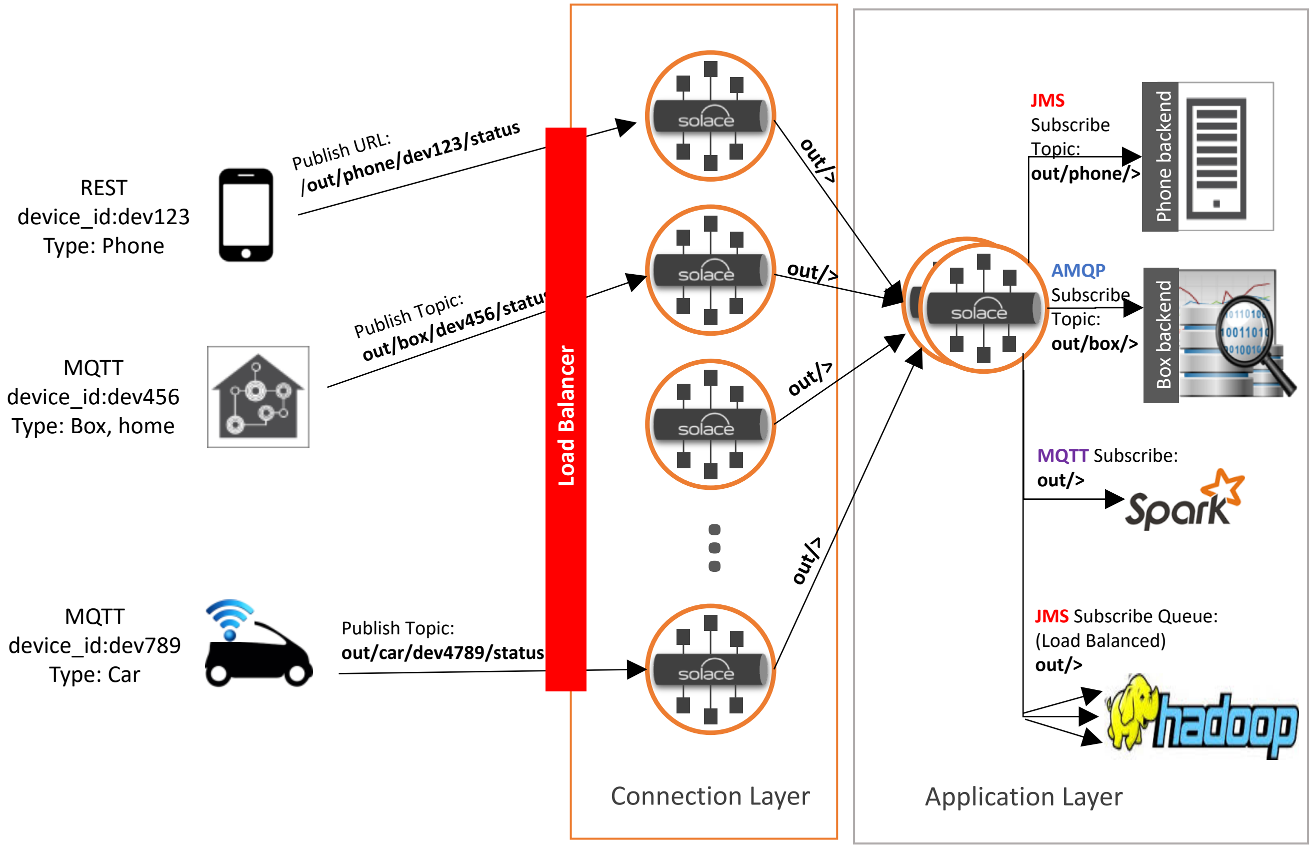
1. Request subscriptions for shape
2. Service calculates subscriptions
3. Service subscribes for app
4. App receives matching data

Use Case 6

Addressing millions of things…

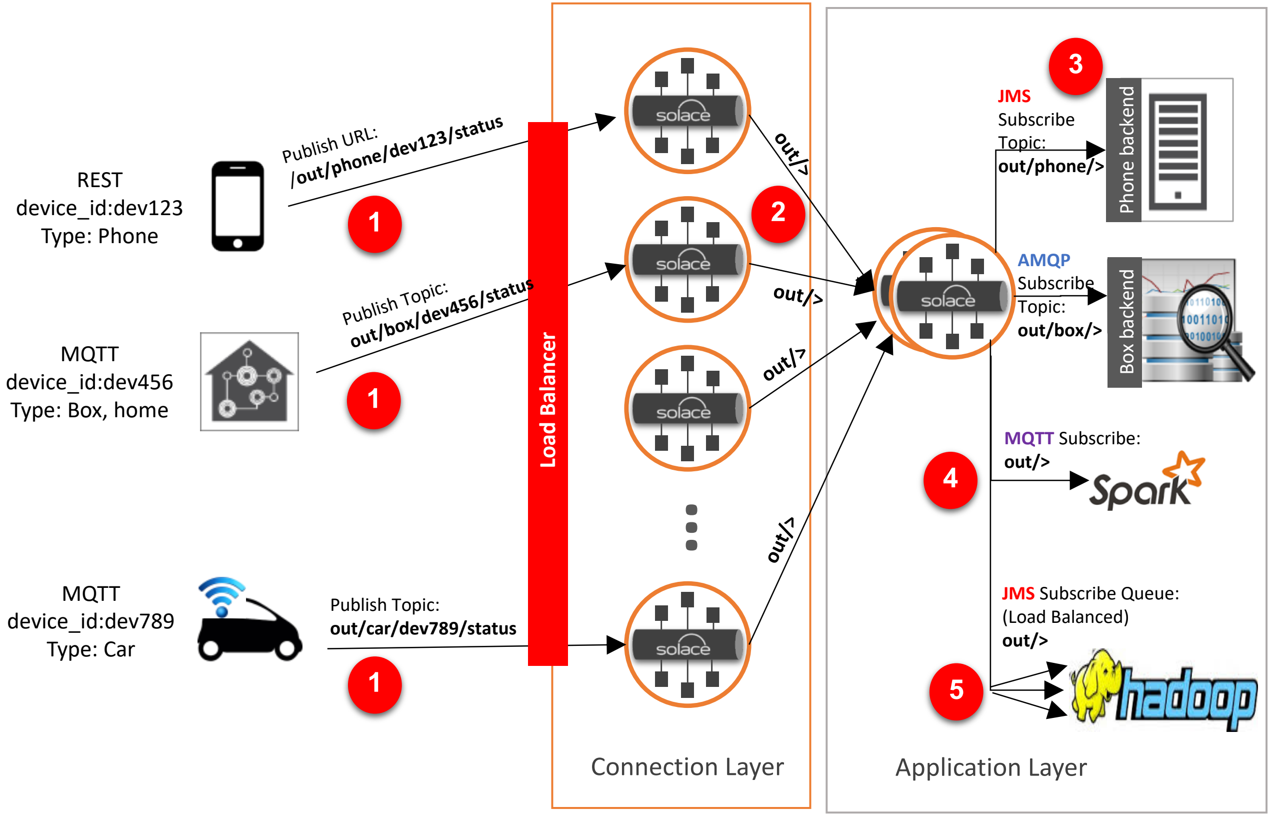
IOT at scale using topics

IOT Edge to Core connectivity



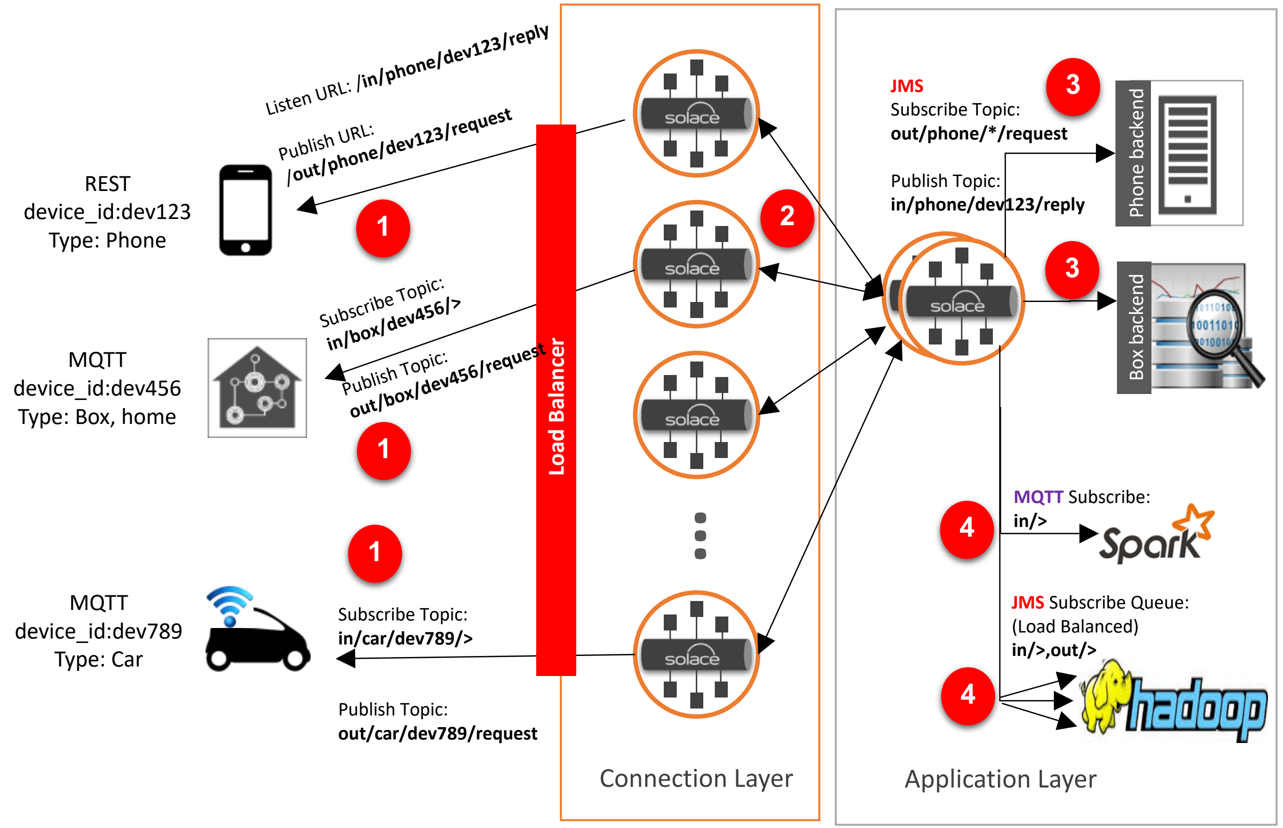
- For IoT at scale, a 2 tier architecture is applied
- The connection layer terminates device connections and is “wide and shallow”
- The core application layer aggregates and queues data and is “narrow and deep”, and communicates with business logic and analytics applications

MEP:Device to Cloud, In Only



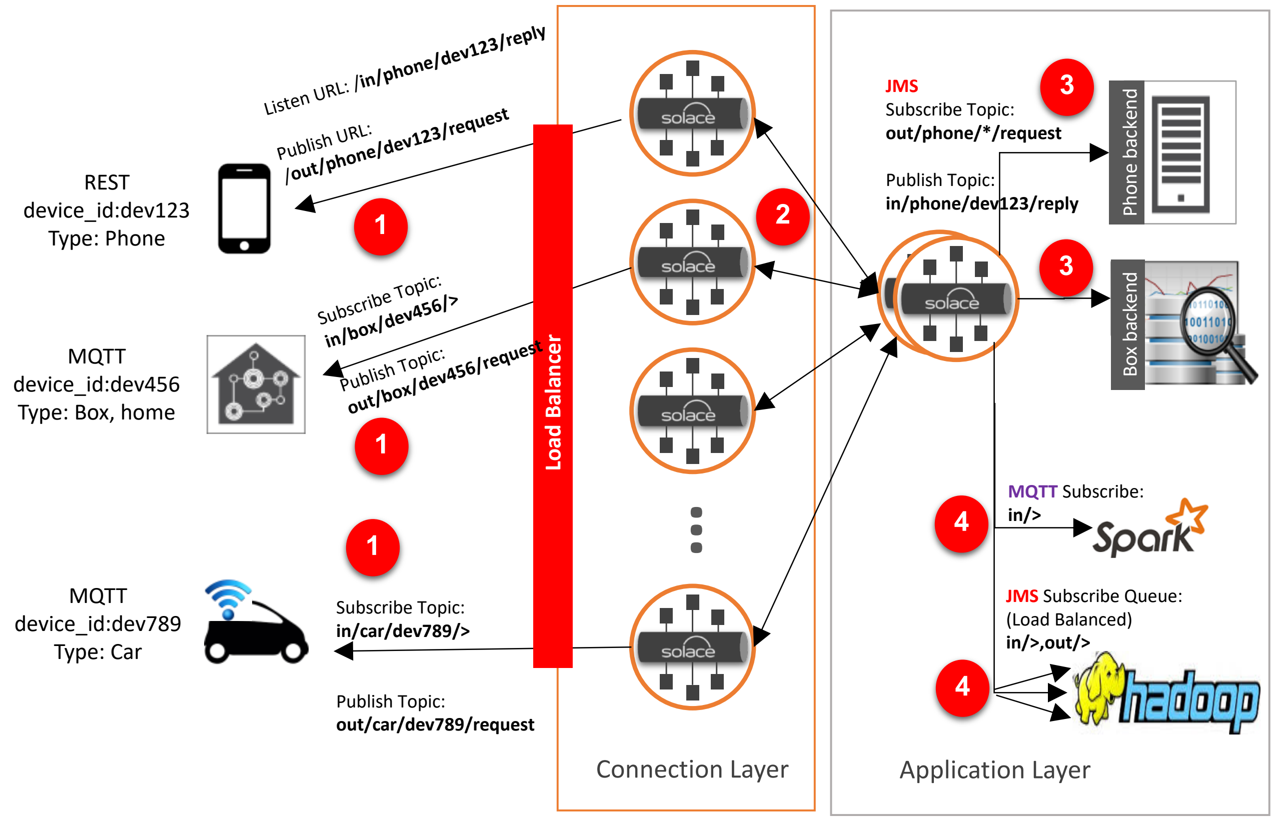
1. Publish topics/URLs should have the chosen namespace for "out" for out from devices, "in" as in to devices, or other similar/multiple verbs
   - Messages land at the connection layer message broker
2. The connection layer message broker is bridged to the application layer broker
   - "out/>" or any other relevant topics are mapped to bridges for the data to flow from connection tier to application tier. Any other verbs/with more levels, wildcards, static subscriptions can be used for more sophisticated routing/filtering
3. The Core Application message brokers deliver messages to backend systems based on their subscriptions (note the phone and box wildcards).
4.
5.

MEP: Device to Cloud request reply


Listen URL: /in/phone/dev123/reply
Publish URL: /out/phone/dev123/request
REST
device_id:dev123
Type: Phone
1
Subscribe Topic: in/box/dev456/>
Publish Topic: out/box/dev456/request
MQTT
device_id:dev456
Type: Box, home
1
1
MQTT
device_id:dev789
Type: Car
Subscribe Topic: in/car/dev789/>
Publish Topic: out/car/dev789/request
Load Balancer
solace
2
Connection Layer
JMS
Subscribe Topic: out/phone/*/request
Publish Topic: in/phone/dev123/reply
3
Phone backend
3
Box backend
4
MQTT Subscribe: in/>
Spark
JMS Subscribe Queue: (Load Balanced) in/>,out/>
4
hadoop
Application Layer

1. Publish topics/URLs should have the chosen namespace for "out" for out from devices, "in" as in to devices, or other similar/multiple verbs
   - Publisher publishes the request message on the show topic. Messages land at the connection layer message broker
2. The connection layer broker is bridged to the application layer message broker bi-directionally
3. The request is routed to the appropriate subscribing backend system
   - The backend system replies using the reply destination sent in the request. This ensures the reply is routed to the sending device
4. The same information, which is going to the devices can also be captured for analytics and audit by passive listeners such as Hadoop and Spark over various protocols

MEP: Device to Cloud request reply


Listen URL: /in/phone/dev123/reply
Publish URL: /out/phone/dev123/request
REST
device_id:dev123
Type: Phone
1
Subscribe Topic: in/box/dev456/>
Publish Topic: out/box/dev456/request
MQTT
device_id:dev456
Type: Box, home
1
1
MQTT
device_id:dev789
Type: Car
Subscribe Topic: in/car/dev789/>
Publish Topic: out/car/dev789/request
Load Balancer
solace
2
Connection Layer
JMS
Subscribe Topic: out/phone/*/request
3
Publish Topic: in/phone/dev123/reply
Phone backend
3
Box backend
4
MQTT Subscribe: in/>
Spark
JMS Subscribe Queue: (Load Balanced) in/>,out/>
4
hadoop
Application Layer

1. Publish topics/URLs should have the chosen namespace for "out" for out from devices, "in" as in to devices, or other similar/multiple verbs
   - Publisher publishes the request message on the show topic. Messages land at the connection layer message broker
2. The connection layer broker is bridged to the application layer message broker bi-directionally
3. The request is routed to the appropriate subscribing backend system
   - The backend system replies using the reply destination sent in the request. This ensures the reply is routed to the sending device
4. The same information, which is going to the devices can also be captured for analytics and audit by passive listeners such as Hadoop and Spark over various protocols

рахмат
danke
謝謝
ngiyabonga
spas
teşekkür ederim
Баярлалаа
mersi
kia ora
barka
welalin
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
blagodaram
dank je
misaotra
paldies
grazzi
спасибі
matondo
kiitos
dankie
thank you
gracias
хвала
nanni
nandri
dhanyavad
mauruuru
asante
manana
bedankt
hvala
köszönöm
aćiū
dankon
obrigada
murakoze
tenki
enkosi
bayarlalaa
gràçie
akun
djiere dieuf
tau
mochchakkeram
дякую
mamnun
dziękuję
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
chokrane
sobodi
děkuji
sukriya
kop khun krap
taiku
obrigado
sagolun
arigatô
takk
dakujem
trugarez
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
terima kasih
rahmet
shukriya
mercé
мерси
kam sah hamnida
rahmat
ευχαριστώ
diolch
dhanyavadagalu
তোমাকে ধন্যবাদ
감사합니다
xièxie
merci