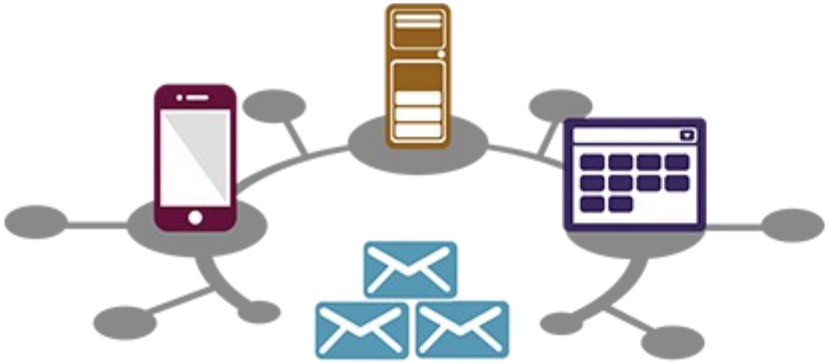
Mobile Messaging

Andrew Schofield
IBM Software Group



IBM Messaging

Why messaging for mobile?

- **The HTTP standard revolutionized how we consume data**
  - A single simple model: Send a request, read the response
  - Available via any tablet, laptop, phone, PC etc.

- **Mobile and M2M applications have additional challenges**
  - HTTP remains ideal for requesting data from a known source
    - e.g. a mobile user *requesting* info
  - But we also need support for *transactions* and an *event-oriented* paradigm:
    - Reliably and securely completing *mobile business transactions* over *unreliable networks*
    - *Pushing* information over *unreliable networks*
    - Transmitting information *one to many*
    - Listening for events *whenever they happen*
    - Distributing minimal packets of data in *huge volumes*

Additional communication challenges in the mobile environment

- Volume (cost) of data being transmitted
- Power (battery) consumption
- Responsiveness (near real-time delivery)
- Reliable delivery over fragile connections
- Scalability

Make it easier for mobile app developers to access enterprise data

Clients for Mobile and M2M Messaging
- Including Simple Javascript Messaging API

JavaScript
Messaging for **Hybrid apps**

Apple iOS
Messaging for native iOS apps

Android
Messaging for native Android apps

Worklight developers
Developer experience augmented with mobile messaging

IBM MessageSight
- Edge-of-network Messaging Appliance
- Highly-scalable MQTT and WebSockets support

MQTT
- Reliable messaging protocol
- Access to enterprise
- Reliable delivery
- Conserve device power
- Reduce network traffic

ESB and MQ messaging
- Universal Enterprise Messaging
- Provide access to enterprise apps and data already connected by MQ
- Pub/sub model to enable dynamic distribution of notifications

MQTT: Key Features

Open

- Open published spec designed for the world of “devices”
  - MQTT client code (C and Java) donated to the Eclipse "Paho" M2M project
  - Standardization underway at OASIS

Lean

- Minimized on-the-wire format
  - Smallest possible packet size is 2 bytes
  - No application message headers
- Reduced complexity/footprint
  - Clients: C=50Kb; Java=100Kb

Reliable

- Three qualities of service:
  - 0 – at most once delivery
  - 1 – assured delivery but may be duplicated
  - 2 – once and once only delivery
- In-built constructs to support loss of contact between client and server.
  - “Last will and testament” to publish a message if the client goes offline.
- Stateful “roll-forward” semantics and “durable” subscriptions.

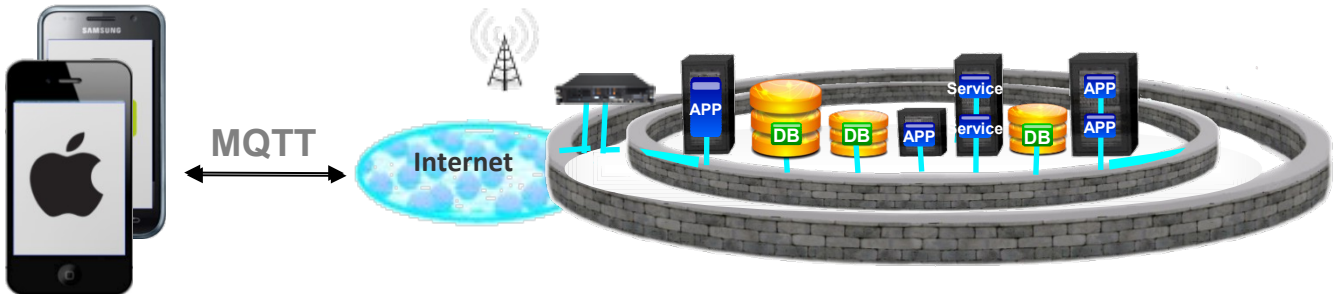
Simple

- Simple / minimal pub/sub messaging semantics
  - Asynchronous (“push”) delivery
  - Simple set of verbs -- connect, publish, subscribe and disconnect.

Programming example – MQTT in JavaScript

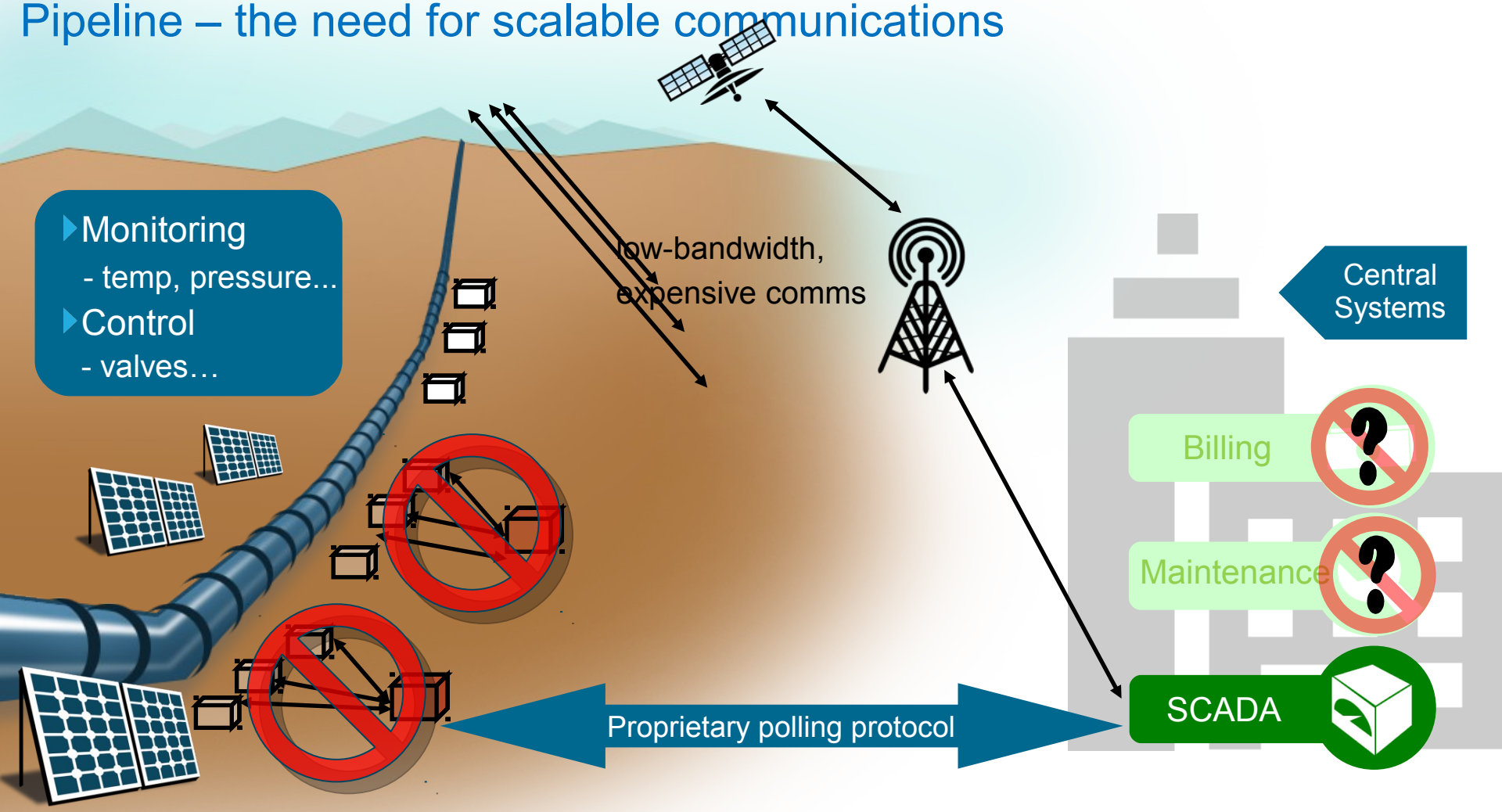
```
/* Sample connect function */
function connect(form) {
    try {
        client = new Messaging.Client(form.host.value, Number(form.port.value), form.clientId.value);
    } catch (exception) {
        alert("Exception: " + exception);
    }
    client.onConnect = onConnect;
    client.onMessageArrived = onMessageArrived;
    client.onConnectionLost = connectionLostCallback;
    client.connect();
}

/* Sample send function */
function send(form) {
    message = new Messaging.Message(form.textMessage.value);
    message.destinationName = form.topicName.value;
    client.send(message);
}

/* Sample subscribe */
function subscribe(form) {
    client.subscribe(form.subscribeTopicName.value);
}
```

Mobile message exchange patterns – beyond simple request/response



Reliable asynchronous transactions	User submits a transaction. One or more responses may come back over time.	MQTT provides **reliability** and **store/forward** of requests and responses if needed – reducing the amount of application code
Continuous update of real-time information	Server-side data is “streamed” to the device and used to update the UI. In most cases this is only required when the app is in the foreground	Small MQTT header size reduces **battery consumption** and **network traffic.** **One->many** publish/subscribe reduces load on application
Notification	Sending alert or other informational message to the device. The app may or may not be running at the time.	Avoidance of polling reduces **battery consumption** and **network traffic**. **Store/forward** of important notifications if app/device is not contactable
Collection of data from device	Data sent to the server coming either from User Interface, of from onboard sensors or from devices attached to the phone	Small MQTT header size reduces **battery consumption** and **network traffic.** **Store/forward** of messages.

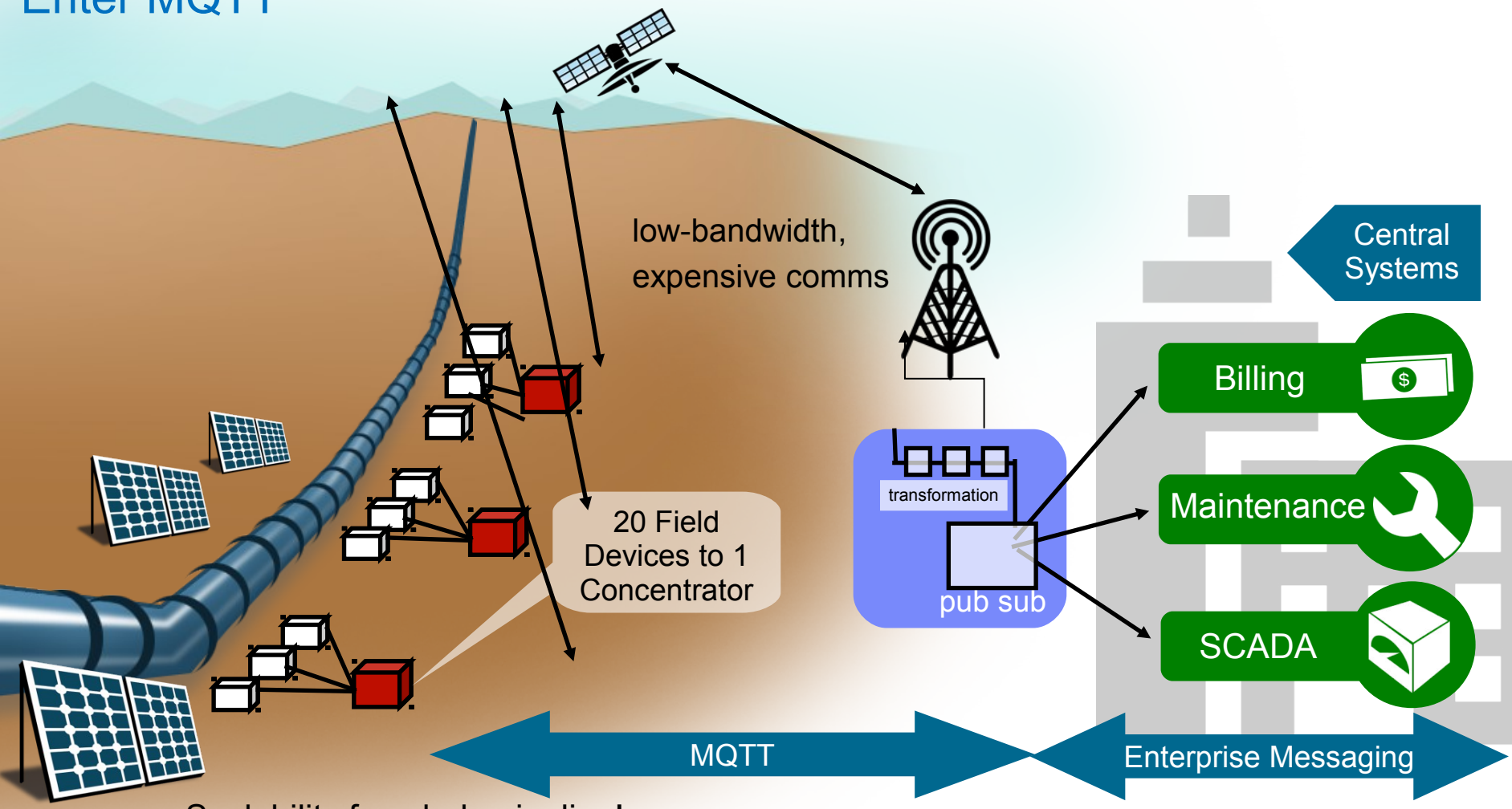
Pipeline – the need for scalable communications



4000 devices integrated, need to add 8000 more BUT:

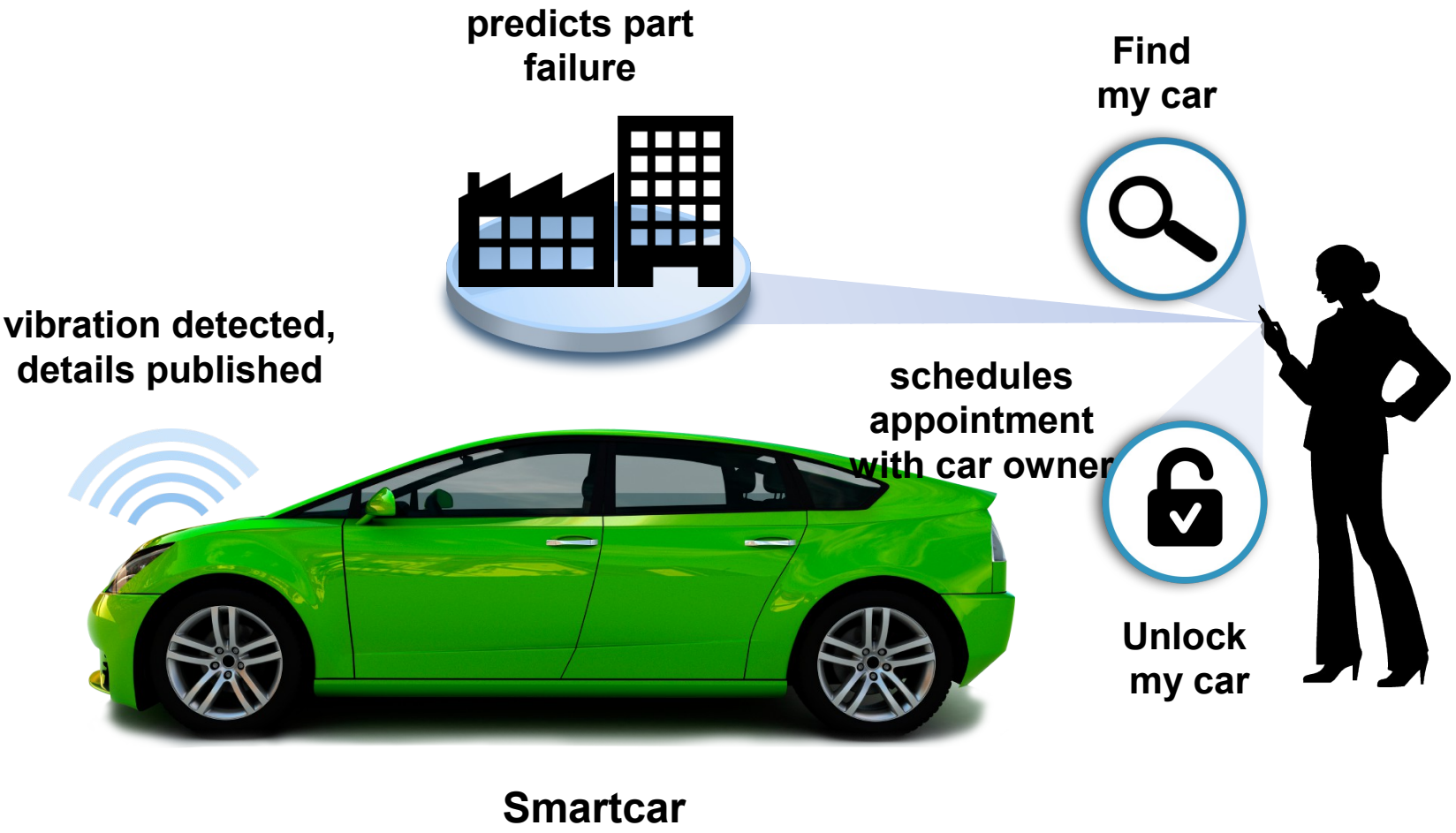
- Satellite network saturated due to polling of device
- VALMET system CPU at 100%
- Other applications needed access to data ("SCADA prison")

Enter MQTT

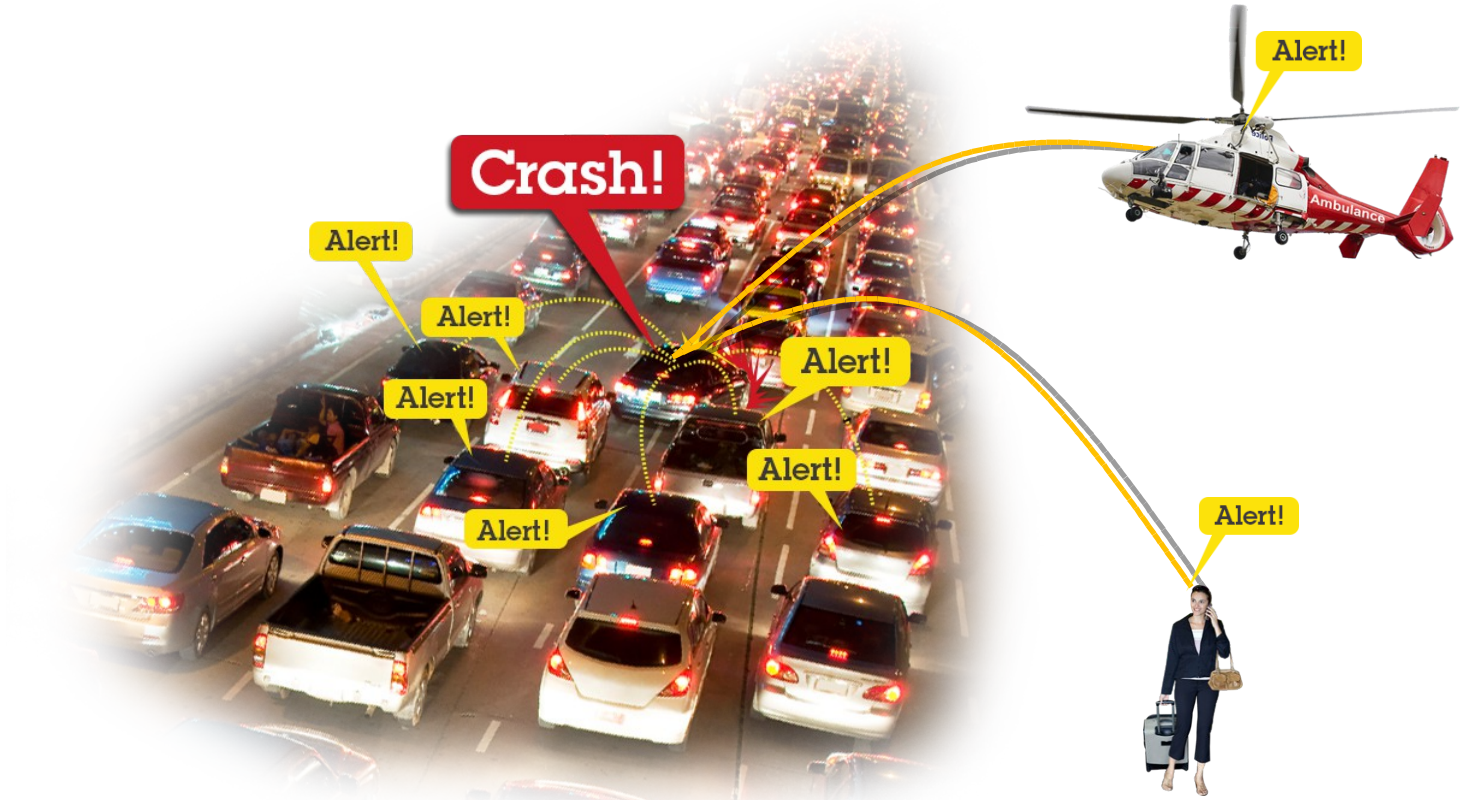


- Scalability for whole pipeline!
- Network traffic much lower - events pushed to/from devices and report by exception
- Network cost reduced
- Lower CPU utilization
- Broken out of the SCADA prison – data accessible to other applications

Mobile use case: Connected car



Mobile use case: Connected city



Introducing IBM MessageSight

Introducing the IBM MessageSight appliance

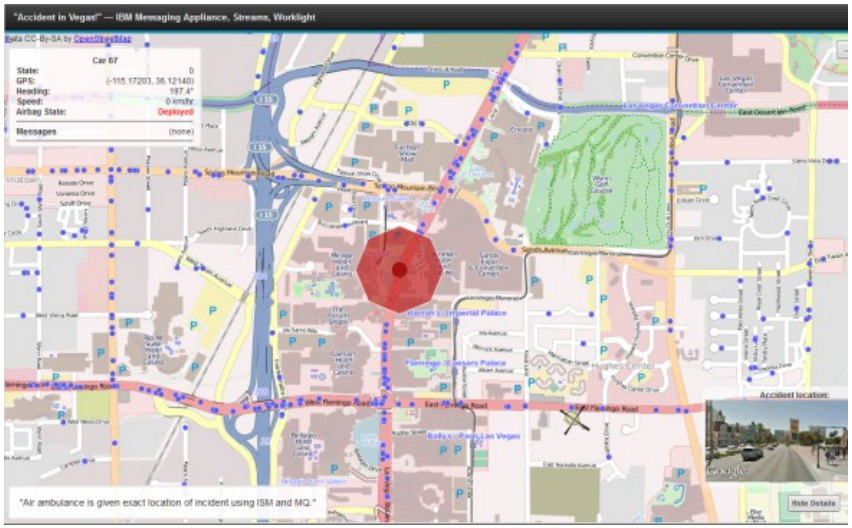
- Extends the IBM Messaging family with a secure, easy to deploy appliance-based messaging server
- Optimized to address the massive scale requirements of machine to machine (m2m) and mobile use cases
- Exploits hardware acceleration for performance
- Designed to sit at the edge of the enterprise

- Can extend your existing messaging infrastructure or be used standalone
- Complements WebSphere MQ - provides an offload/accelerator for edge of enterprise scenarios

Demo: Real-time analytics with InfoSphere Streams

“Emergency in Vegas”

- 1000s of cars circulate in Las Vegas and subscribe to a “Smart Car” notification service
- Cars update the notification service in real time
- Based on events, they can also be warned or notified



High-level architecture of the demo

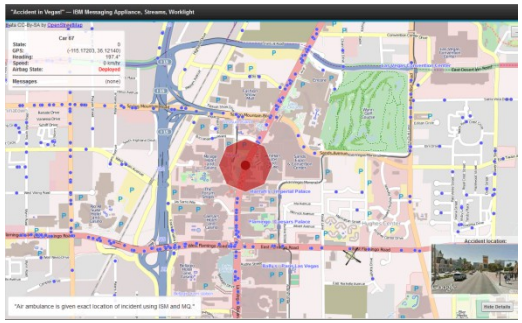
Cars: Embedded MQTT applications
In the dash board

MQTT

IBM MessageSight

MQ
Connectivity

Existing Mainframe
Emergency Notification System



MQTT over
Websockets

HTML5 Web UI
Using WebSocket Client

MQTT
adapter

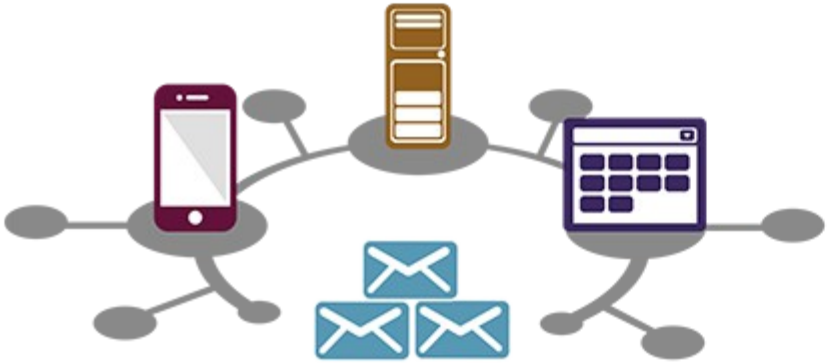
Geospatial Analytics

InfoSphere Streams

Tablet
With Worklight App.

Video

Mobile Messaging



IBM Messaging