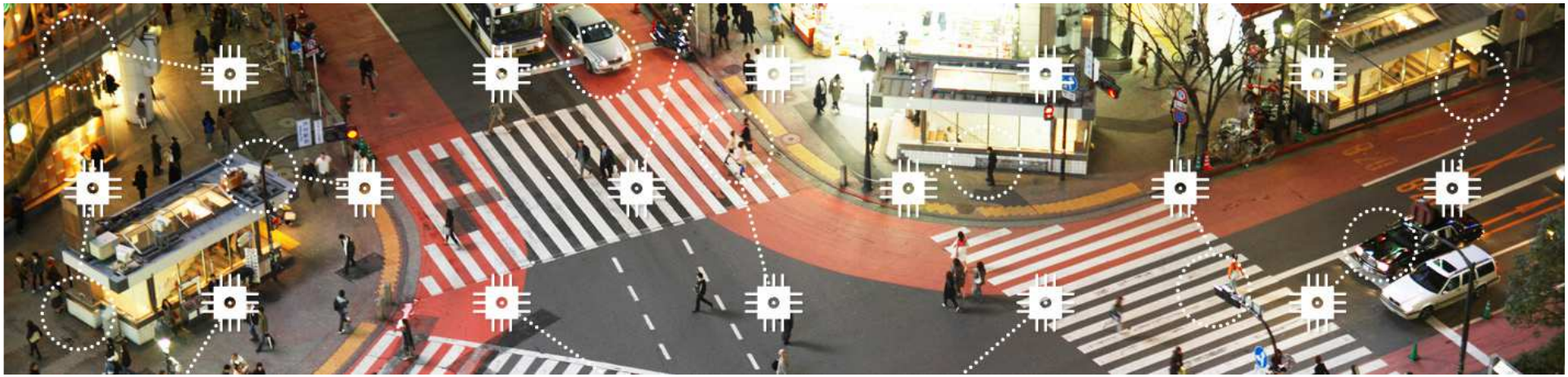
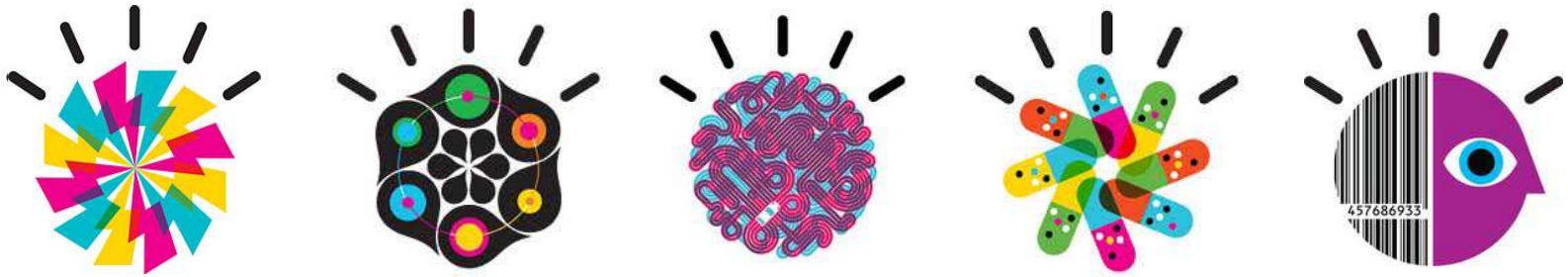
IBM MessageSight

Joel Gauci

Certified IT Specialist,IBM Connectivity Appliances

gauci@fr.ibm.com

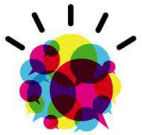




POTs MQTT & MessageSight

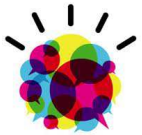
> Tuesday, Sept. 10th 2013
>> Friday, Oct. 18th 2013
>>> Thursday, Nov. 21st 2013
>>>> Thursday, Dec. 12th 2013

Agenda



- Internet of Things (IoT)
- MQTT
- IBM MessageSight
- Use Cases for MessageSight
- *Demo*

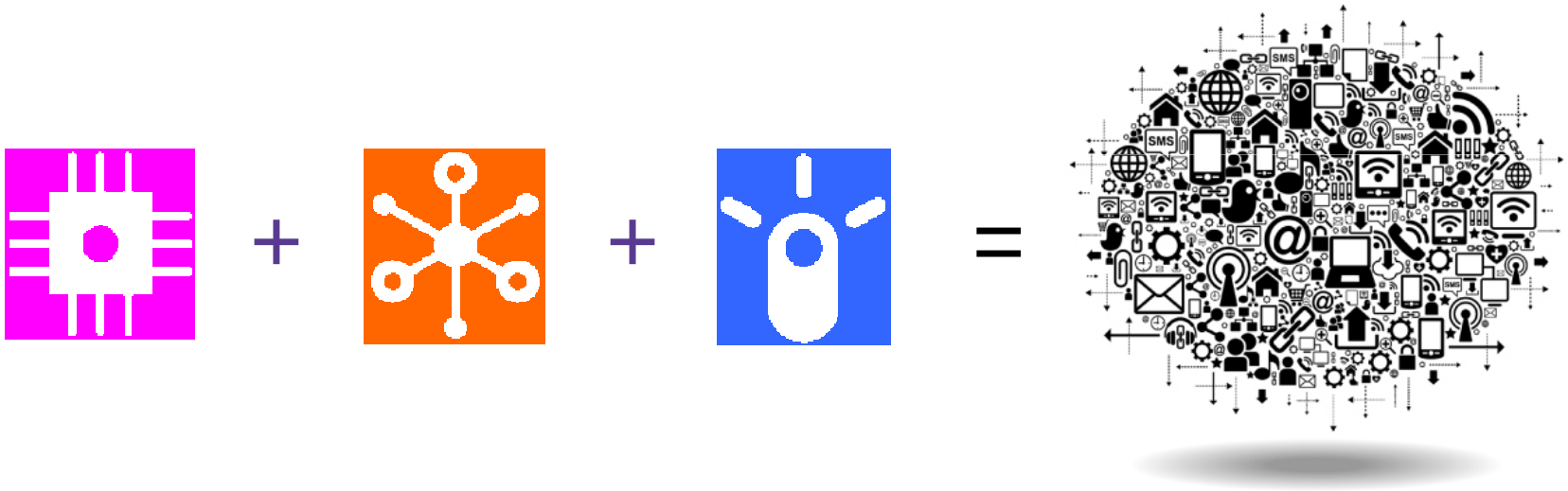
Agenda



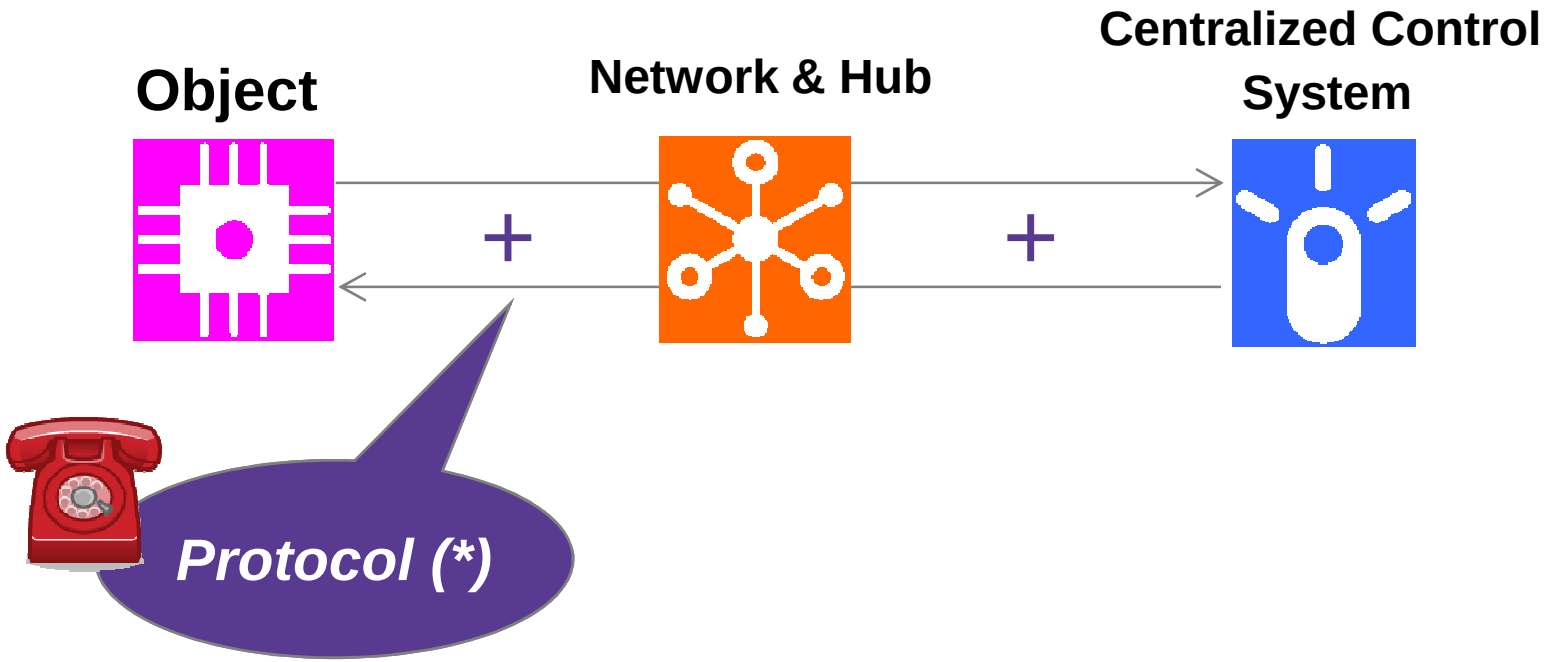
- Internet of Things (IoT)
- MQTT
- IBM MessageSight
- Use Cases for MessageSight
- *Demo*

Internet of Things

- **The Interconnection of Everything**

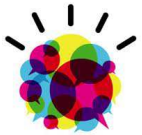


Internet of Things



(*): a standardized and « simple » protocol ... please !

Agenda

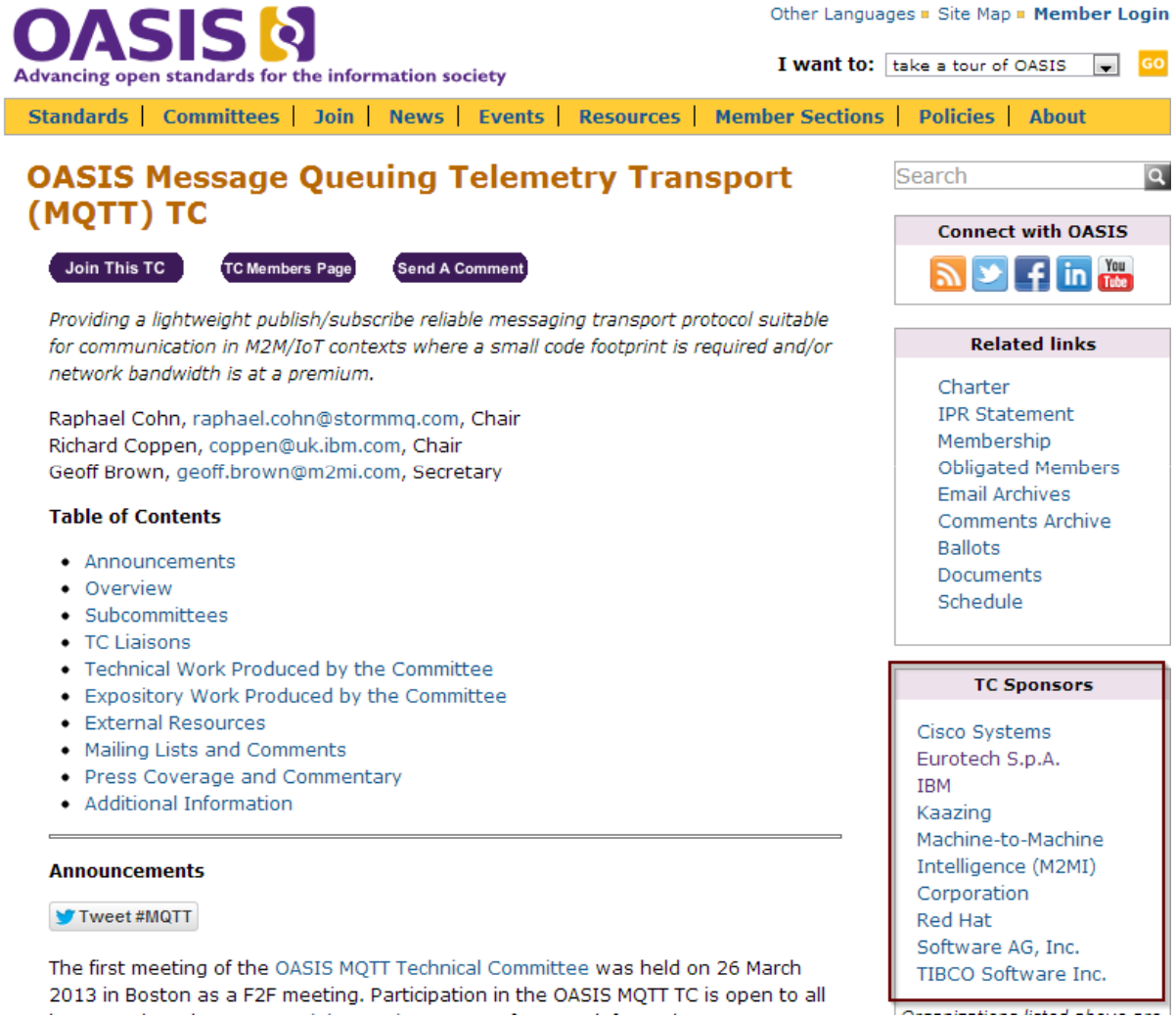


- Internet of Things (IoT)
- MQTT
- IBM MessageSight
- Use Cases for MessageSight
- *Demo*

MQTT is a Protocol !

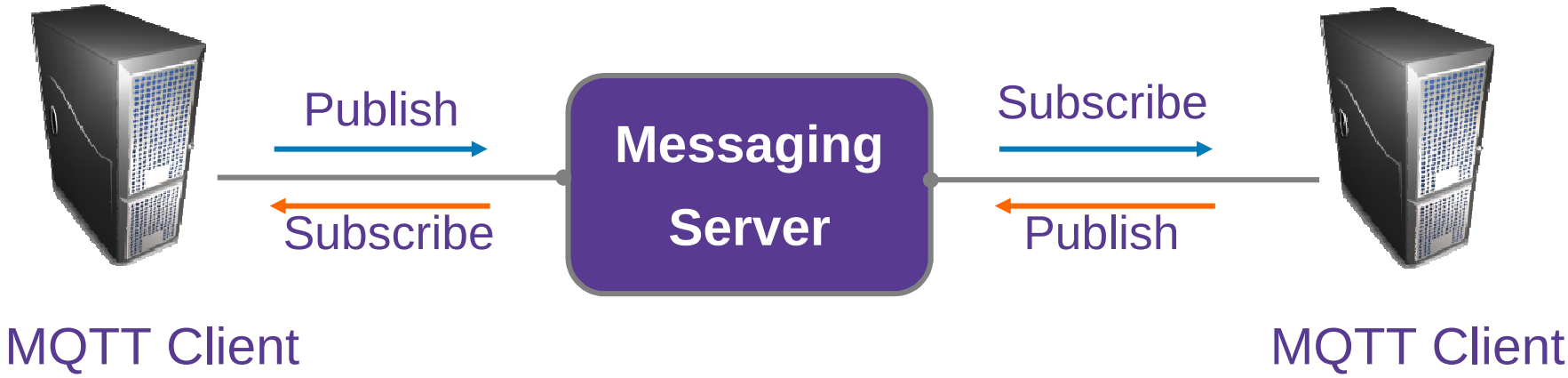
- **MQTT = MQ Telemetry Transport**
- MQTT is an extremely simple and lightweight messaging protocol.
- Its **publish/subscribe** architecture is designed to be open and easy to implement.
- **MQTT is defined as the standard « IoT protocol » by OASIS**
- The MQTT protocol includes the following benefits:
  - Extends connectivity beyond enterprise boundaries to smart devices.
  - Offers connectivity options optimized for sensors and remote devices.
  - Delivers relevant data to any intelligent, decision-making asset that can use it.
  - Enables massive scalability of deployment and management of solutions.

MQTT & OASIS

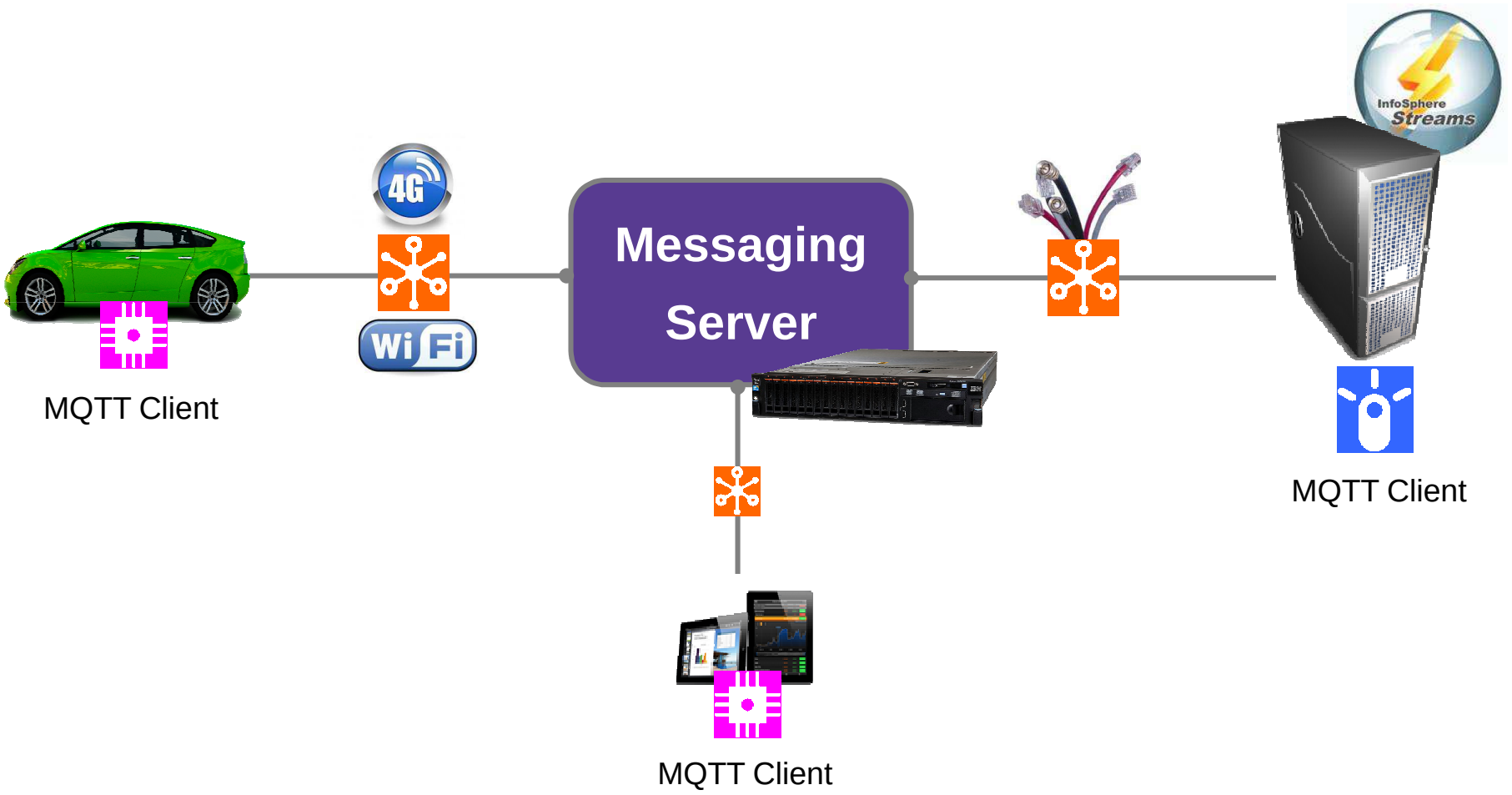


- Cisco Systems
- Eurotech S.p.A.
- IBM
- Kaazing
- Machine-to-Machine Intelligence (M2MI)
- Corporation
- Red Hat
- Software AG, Inc.
- Tibco Software Inc.

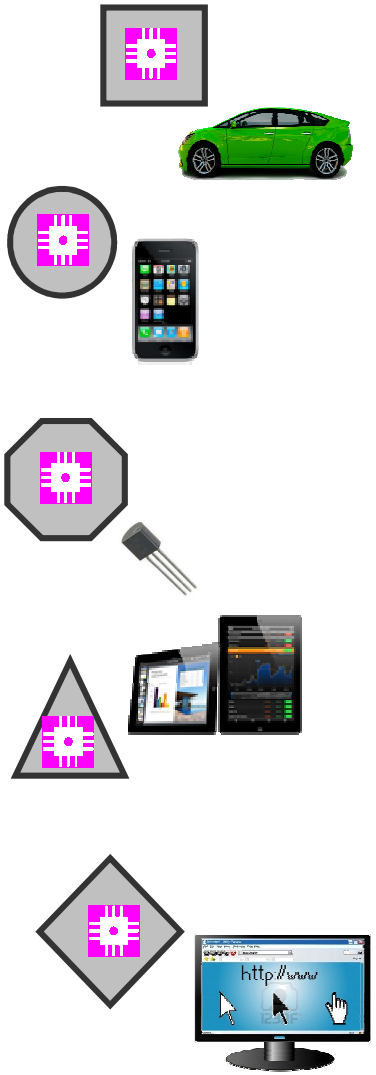
Actors of the MQTT Protocol



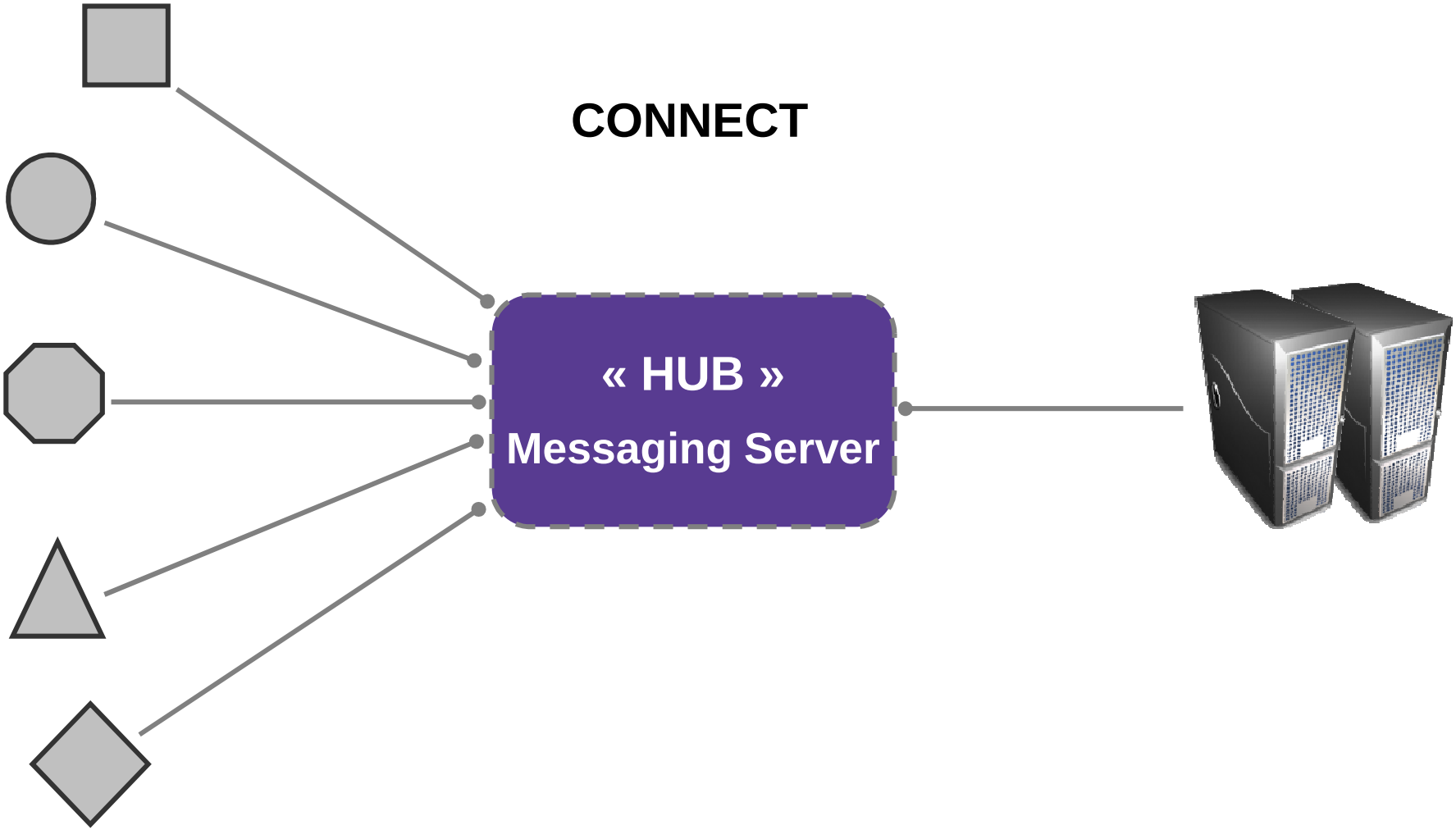
Example of real Actors



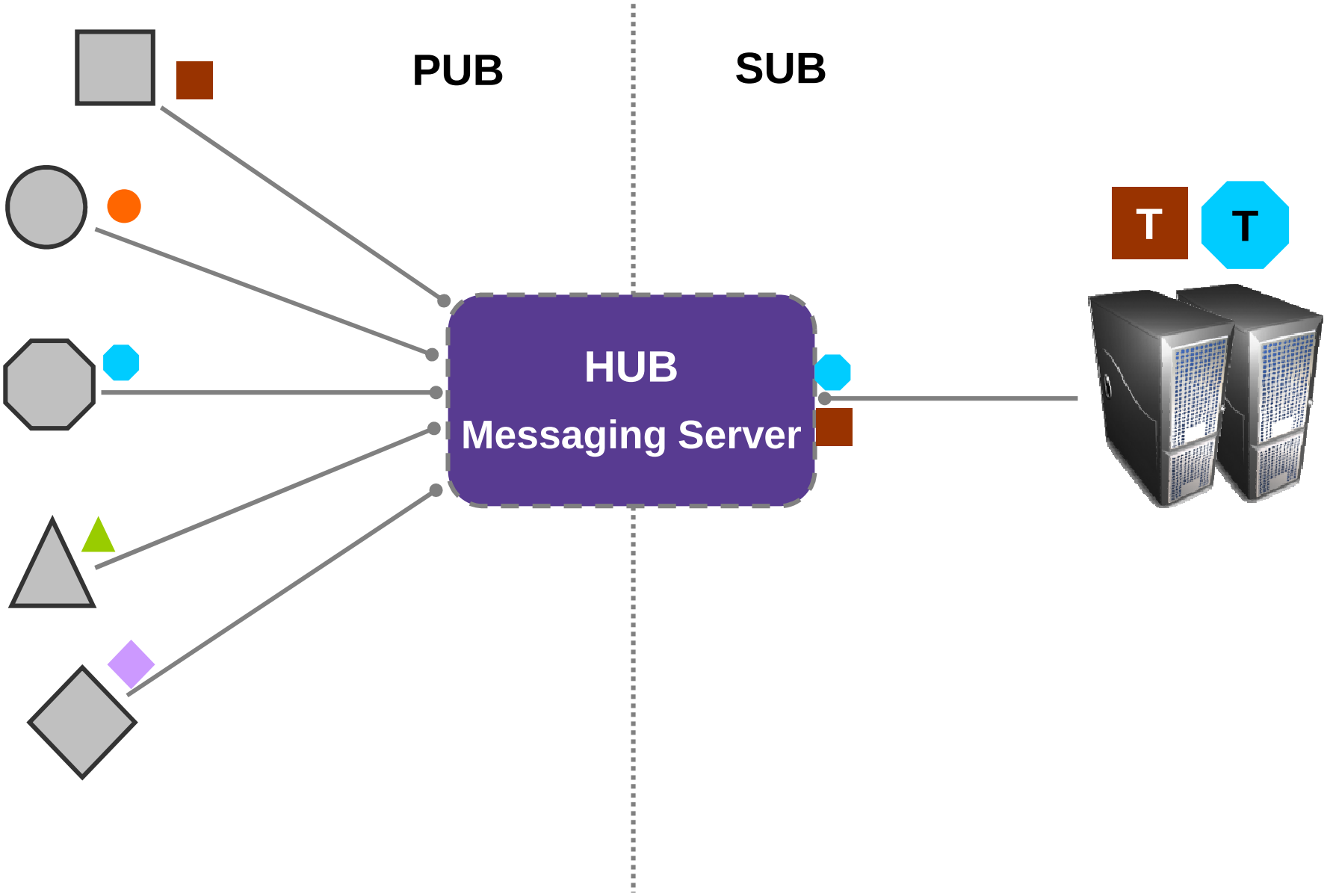
Publish/Subscribe



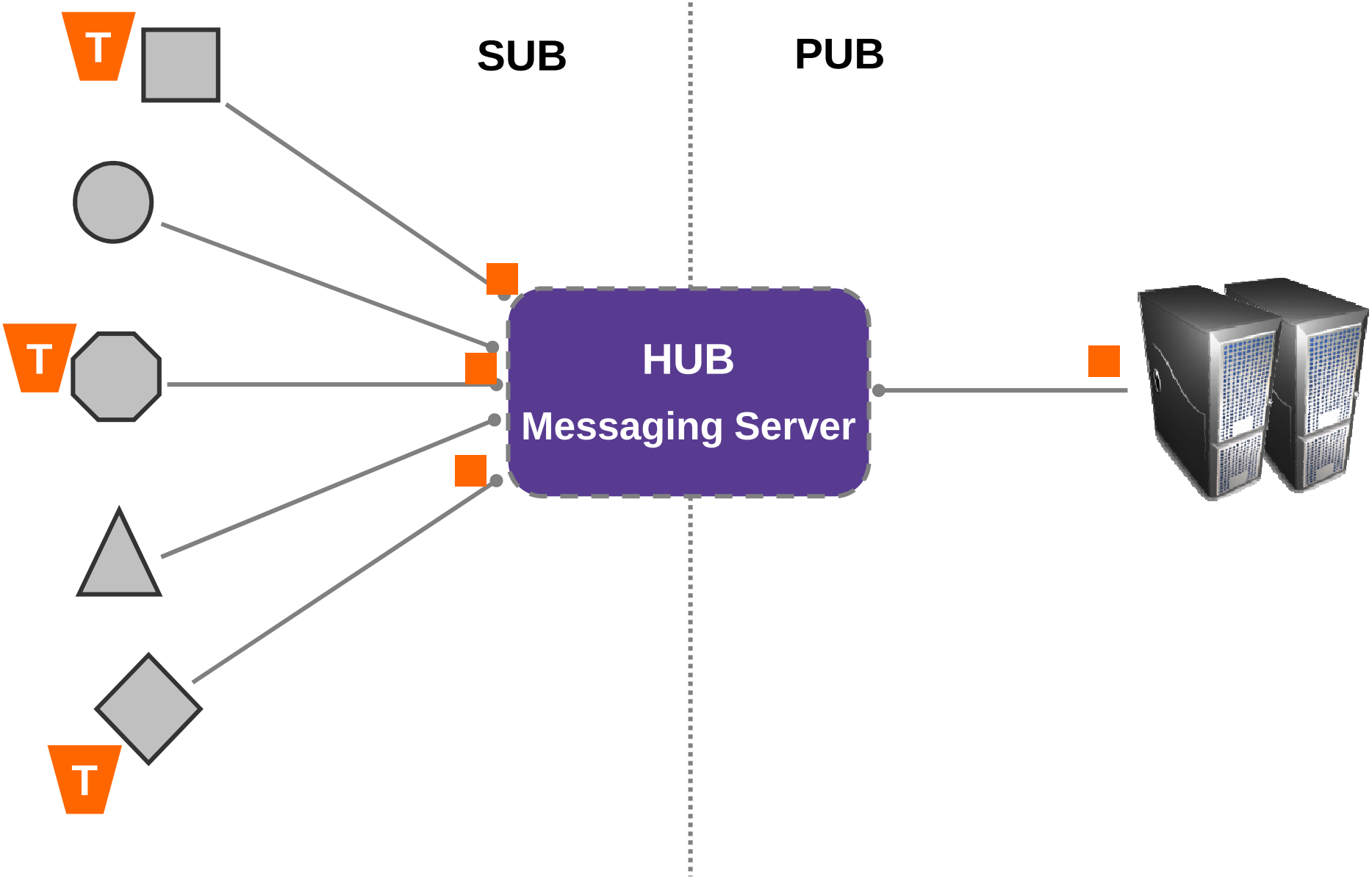
Publish/Subscribe



Publish/subscribe



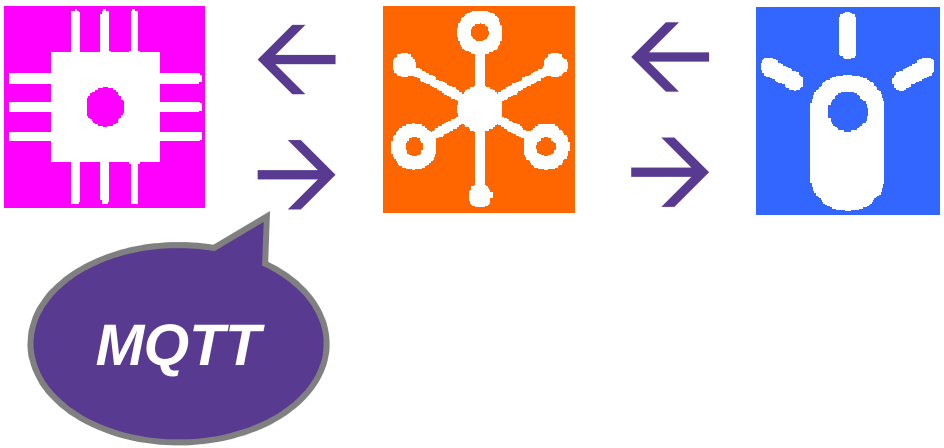
Publish/subscribe



MQTT usage

- The MQTT messaging protocol is designed for devices in constrained environments
  - Embedded systems with limited processing ability and memory.
  - Systems that are connected to unreliable networks.

- It provides the robust messaging features that are needed to communicate with remote systems and devices while consuming just a small portion of network bandwidth.



Quality of service levels

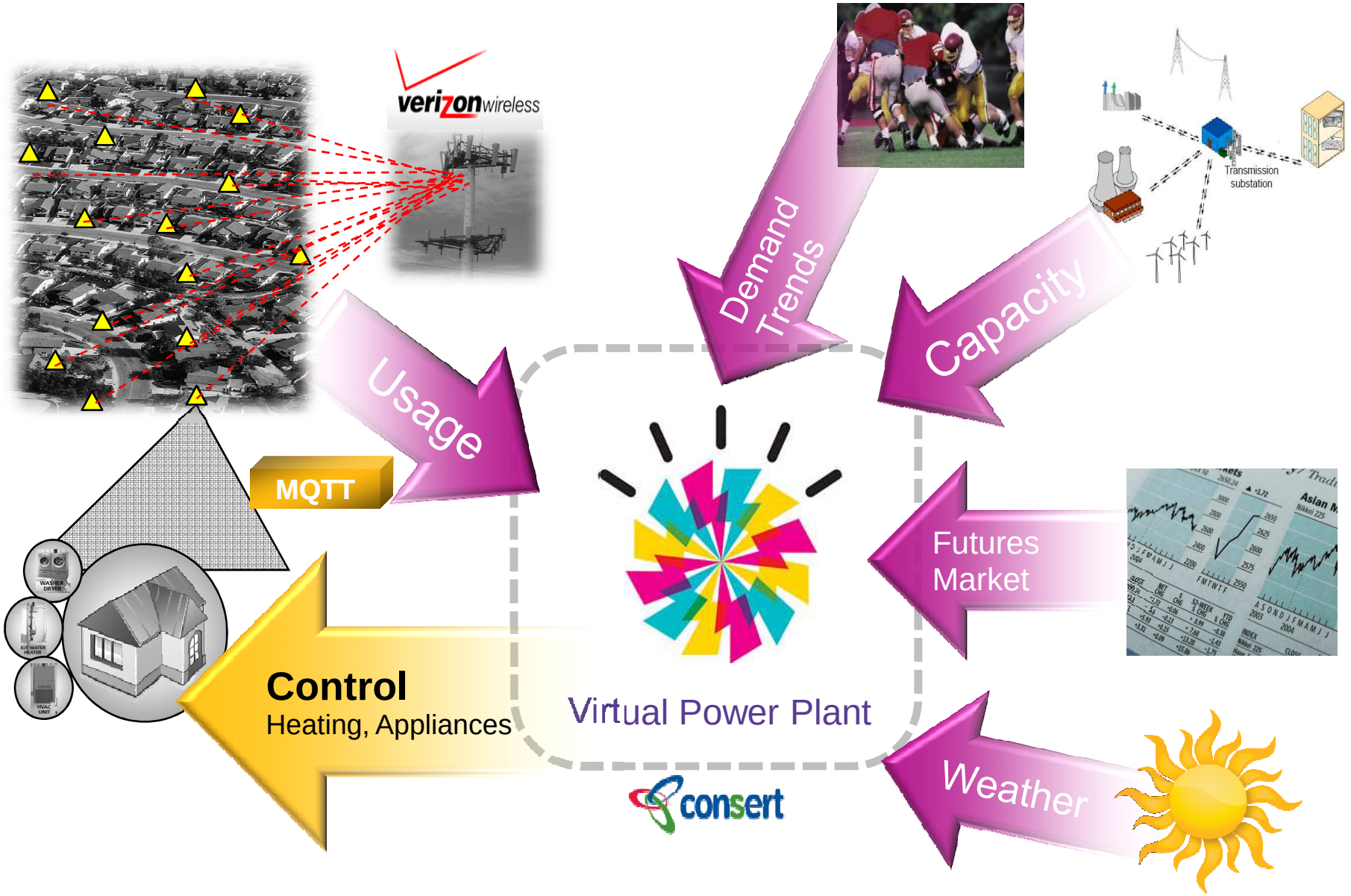
- MQTT defines three quality of service (QoS) for message delivery
- Each level designates a higher level of effort by the server (Hub) to ensure that the message gets delivered
  - 0: delivery of message at best effort
  - 1: message delivered at least once
  - 2: message delivered exactly once
- Higher QoS levels ensure more reliable message delivery but
  - might consume more network bandwidth
  - subject the message to delay due to issues such as latency

Comparison between MQTT & HTTP

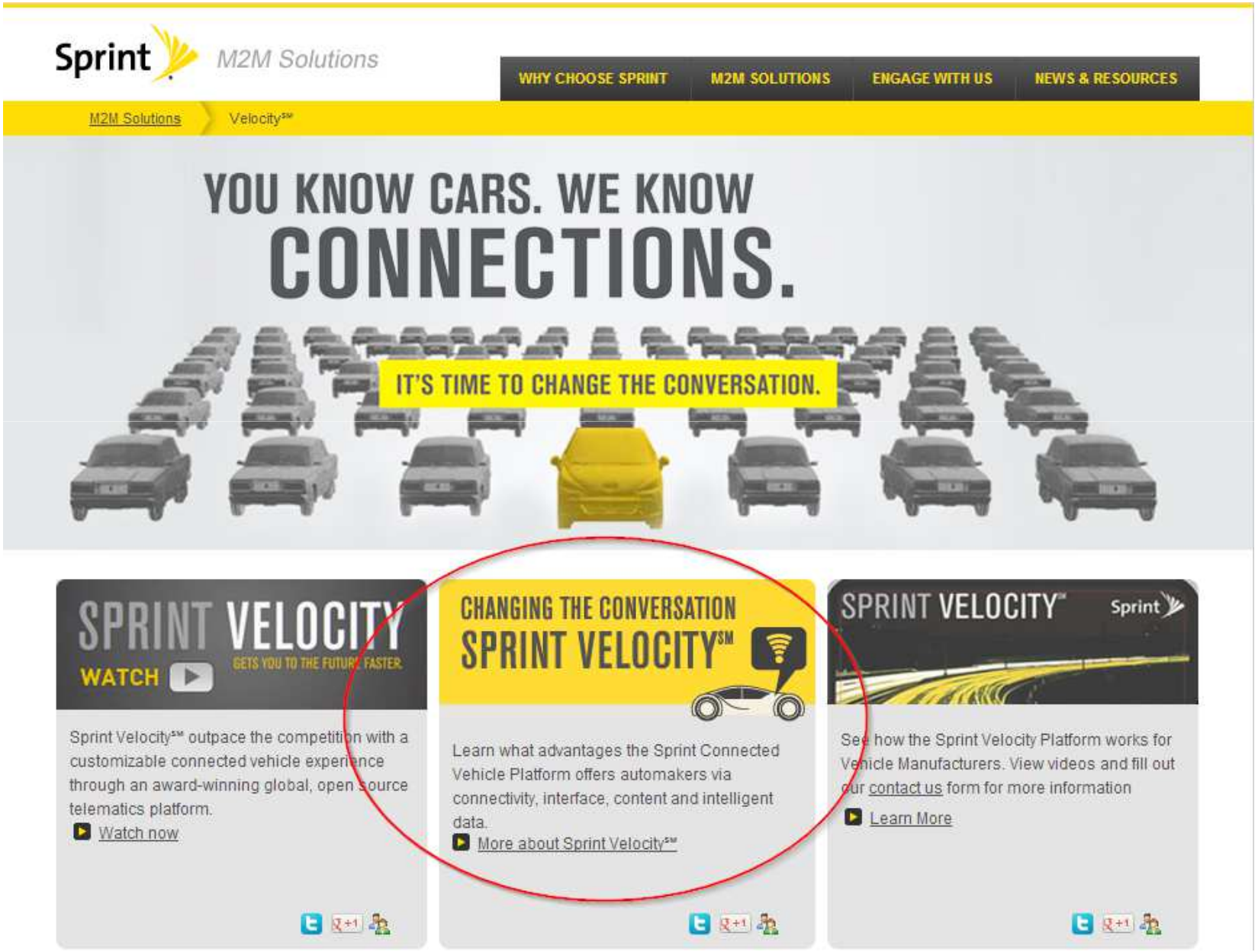
	MQTT	HTTP
Design orientation	Data centric	Document centric
Pattern	Publish/subscribe	Request/response
Complexity	Simple	More complex
Message size	Small, with a compact binary header just two bytes in size	Larger, partly because status detail is text-based
Service levels	Three quality of service settings	All messages get the same level of service
Extra libraries	Libraries for C (30 KB) and Java (100 KB)	Depends on the application (JSON, XML), but typically not small
Data distribution	Supports 1 to zero, 1 to 1, and 1 to n	1 to 1 only

Implementation Example of MQTT

Implementation Example of MQTT



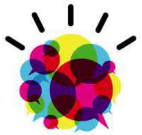
MQTT & MessageSight Example



Benefits of using MQTT

- ✓ Simplicity
- ✓ Use of a publish/subscribe model
- ✓ Minimal maintenance
- ✓ Limited on-the-wire footprint
- ✓ Limited battery consumption
- ✓ Continuous session awareness
- ✓ Local message processing
- ✓ Message persistence
- ✓ Agnostic regarding data types

Agenda



- Internet of Things (IoT)
- MQTT
- IBM MessageSight
- Use Cases for MessageSight
- *Demo*

IBM MessageSight - Introduction

- An appliance-based **messaging server** built on special purpose hardware
- Supports very large numbers of connected clients and devices
- Processes high volumes of messages
- Maintains consistent latency
- Can extend existing infrastructure or be used as a standalone messaging server

- Sits at the edge of the enterprise and connects users and devices to enable use cases like mobile and telemetry
- Supports high quality, timely notification of data and events to massive number of external users
- Enables business agility and intelligence by delivering real-time events from external devices to analytics engines and big data
- Provides a secure point of entry into the enterprise

IBM MessageSight Appliance

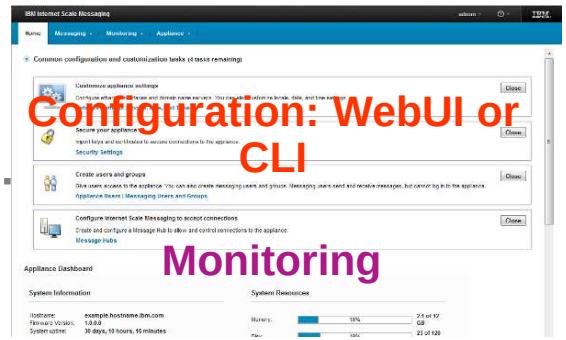
Hardened appliance; DMZ ready
→ No user accessible Operating System
→ Signed and encrypted firmware

Multiple Netwok Interfaces

8 x 1GbE ports
→2 mgmt
→6 data

4 x 40GbE ports

Firmware

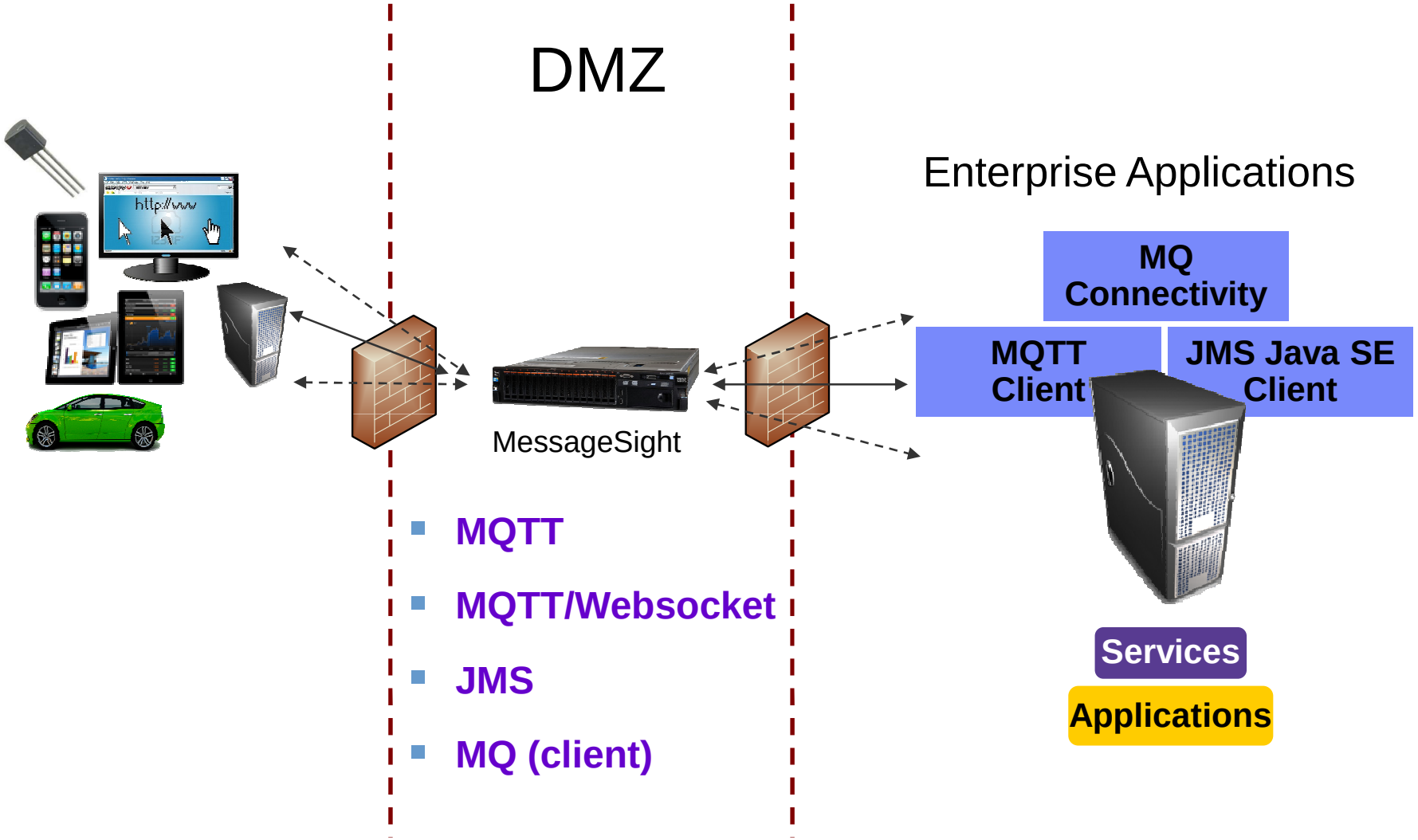


Configuration: WebUI or CLI

Monitoring

- 2U form factor appliance
- Unique hardware optimizations
- 4 x 900GB hard drives

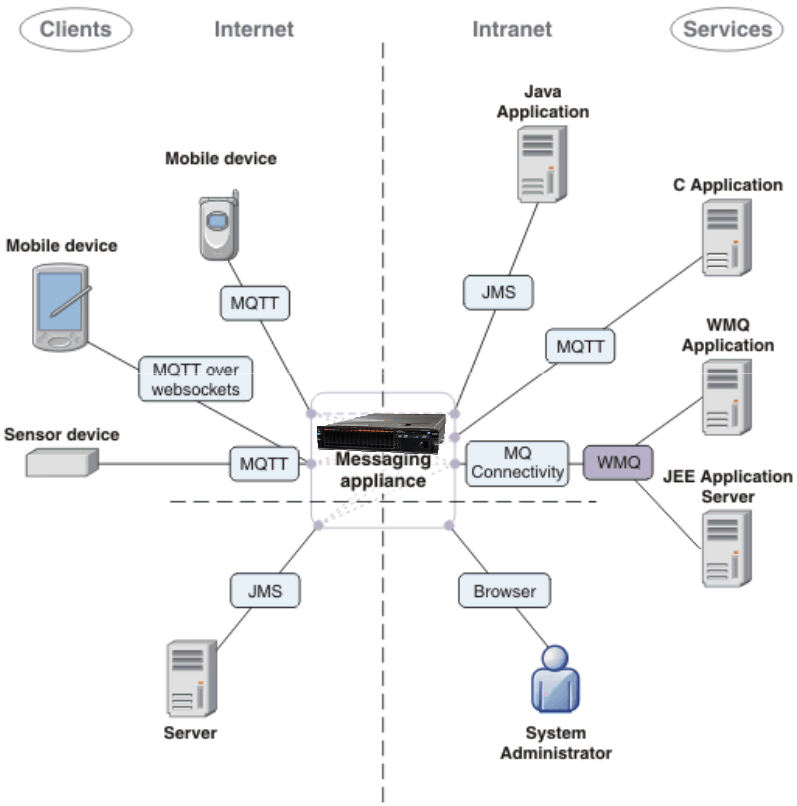
Edge Connectivity



MessageSight supported protocols

Asynchronous Messaging that supports

- Publish and Subscribe (topic-based)
- Point-to-point (queue-based)
- Persistent and non-Persistent qualities of service
- MQ Telemetry Transport (MQTT) v3.1 specifications
- MQTT over HTML5 WebSockets
- Java Messaging Service (JMS) 1.1



IBM MessageSight: Performance and Reliability

Massive Scale

Orders of Magnitude Improvements

Performance

- Delivers high throughput for persistent and non-persistent messages
- Supports vast numbers of connected devices

Reliability

- Assured delivery of messages
  - Support for MQTT qualities of service (QoS) 0, 1, and 2 - best effort, at least once and exactly once
  - Local transaction support
  - Support for high availability configurations

IBM MessageSight: Performance examples

- **Pub/sub performance results**
  - Persistent JMS: ~200K msg/sec
  - Non-persistent JMS: ~1.4M msg/sec
  - Persistent MQTT QoS1: ~500K msg/sec
  - Non-persistent MQTT QoS0: 13M msgs/sec
- **Scalability results**
  - #Connections: 1M

IBM MessageSight: Security

Secure and Reliable

DMZ Ready Appliance

- Authentication via local user store or external Lightweight Directory Access Protocol (LDAP) servers
- Secure Sockets Layer (SSL) and Transport Level Security (TLS) support (SSL v3 and TLS 1.0, 1.1 and 1.2)
- Federal Information Processing Standard (FIPS) 140-2 Level (1-certified cryptographic module)
- Fine-grained messaging authorization policies which restrict access based on combinations of:

  → User or group, Client identifier, Protocol, Network interface, Listening address and/or port, Client IP address or range, Destination (topic and queue) name

IBM MessageSight: Integration

Easy to Integrate

Open Standards

- Extends and connects to WebSphere MQ infrastructures

  → Supports mapping appliance destinations to and from WebSphere MQ destinations

- Supports connectivity to WebSphere Message Broker via JMS and or MQTT nodes
- Integrates with Java environments
- Integrates with rich HTML5-based web applications
- Allows to develop interactive mobile messaging applications with IBM Worklight Studio

IBM MessageSight: Developer-friendly APIs and libraries

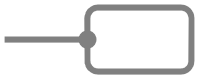
Developer Friendly

Simple Programming Model

- MQTT clients and libraries for a variety of platforms (C and Java- Based APIs)
- Libraries for Google Android and Apple iOS
- JMS client libraries
- JavaScript API for HTML5-based applications
- PhoneGap MQTT plug-ins with JavaScript API for use with IBM
- Worklight, Apache Cordova, and Adobe PhoneGap

Implemeting an MQTT Client in JavaScript

Connect



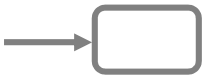
```
function connect(form) {
    try {
        client = new Messaging.Client(hostName, port, clientId);
    } catch (exception) {
        alert("Exception:"+exception);
    }
    client.onMessageArrived = onMessageArrived;
    client.onConnectionLost = connectionLostCallback;

    client.connect({onSuccess: onSuccessCallback});
}
```

Create client

Set callbacks

Connect to the server

Send



```
function doSend(form) {
    if (form.textMessage.value == "") {
        message = new Messaging.Message("");
    } else {
        message = new Messaging.Message(form.textMessage.value);
    }

    message.destinationName = form.topicName.value;

    client.send(message);
}
```

Create Message object

Set Topic

Send the message

Subscribe

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

Subscribe to a topic

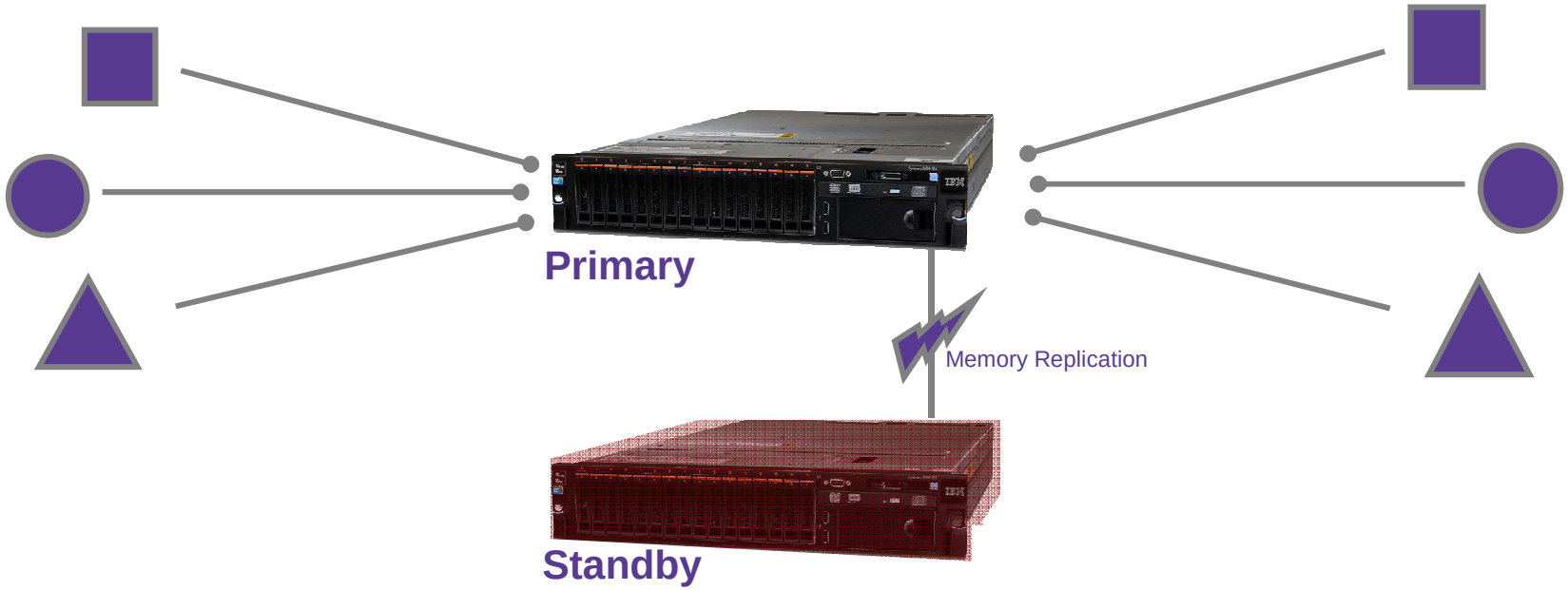
Receive

```
function onMessageArrived(message) {
    var form = document.getElementById("basic");
    form.receivedMessage.value = message.payloadString;
}
```

Show the payload in a field

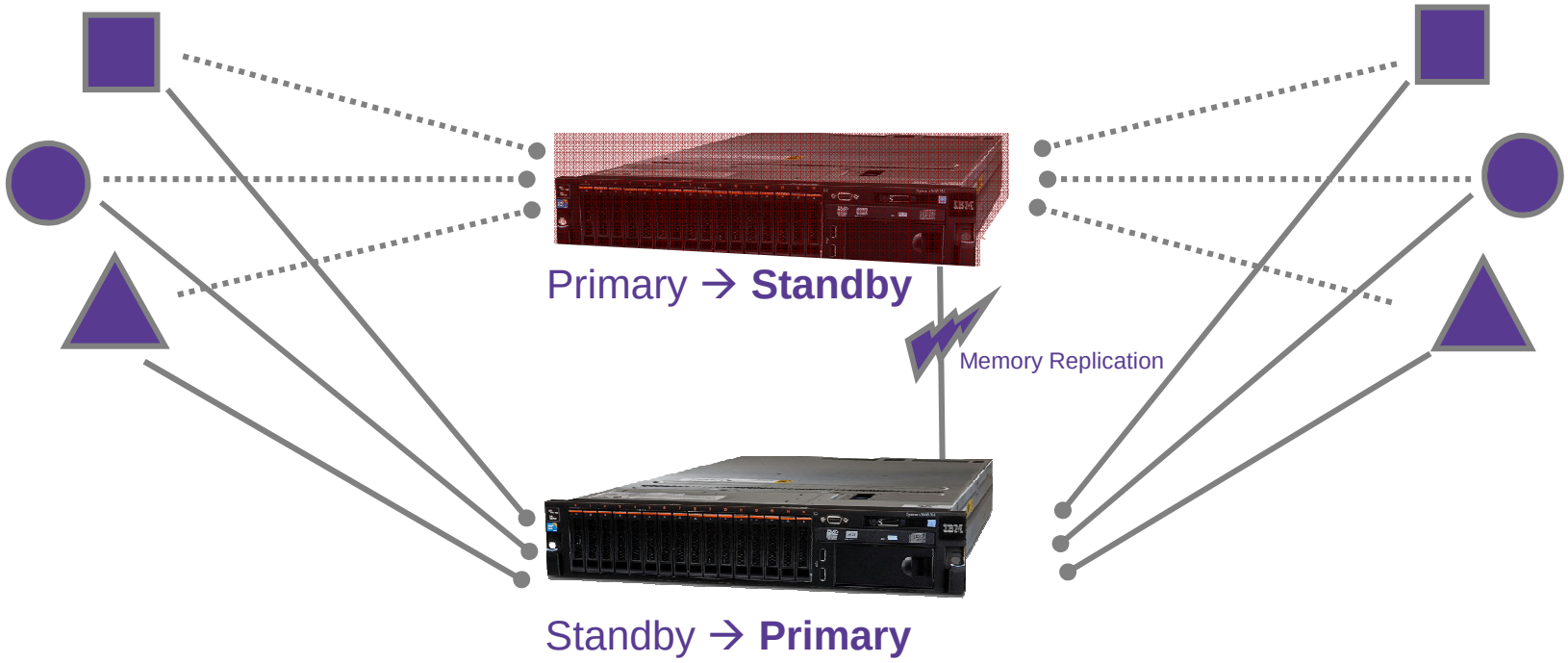
IBM MessageSight: High Availability

- Failover using RDMA over Converged Ethernet (RoCE)
  - RDMA: Remote Direct Memory Access

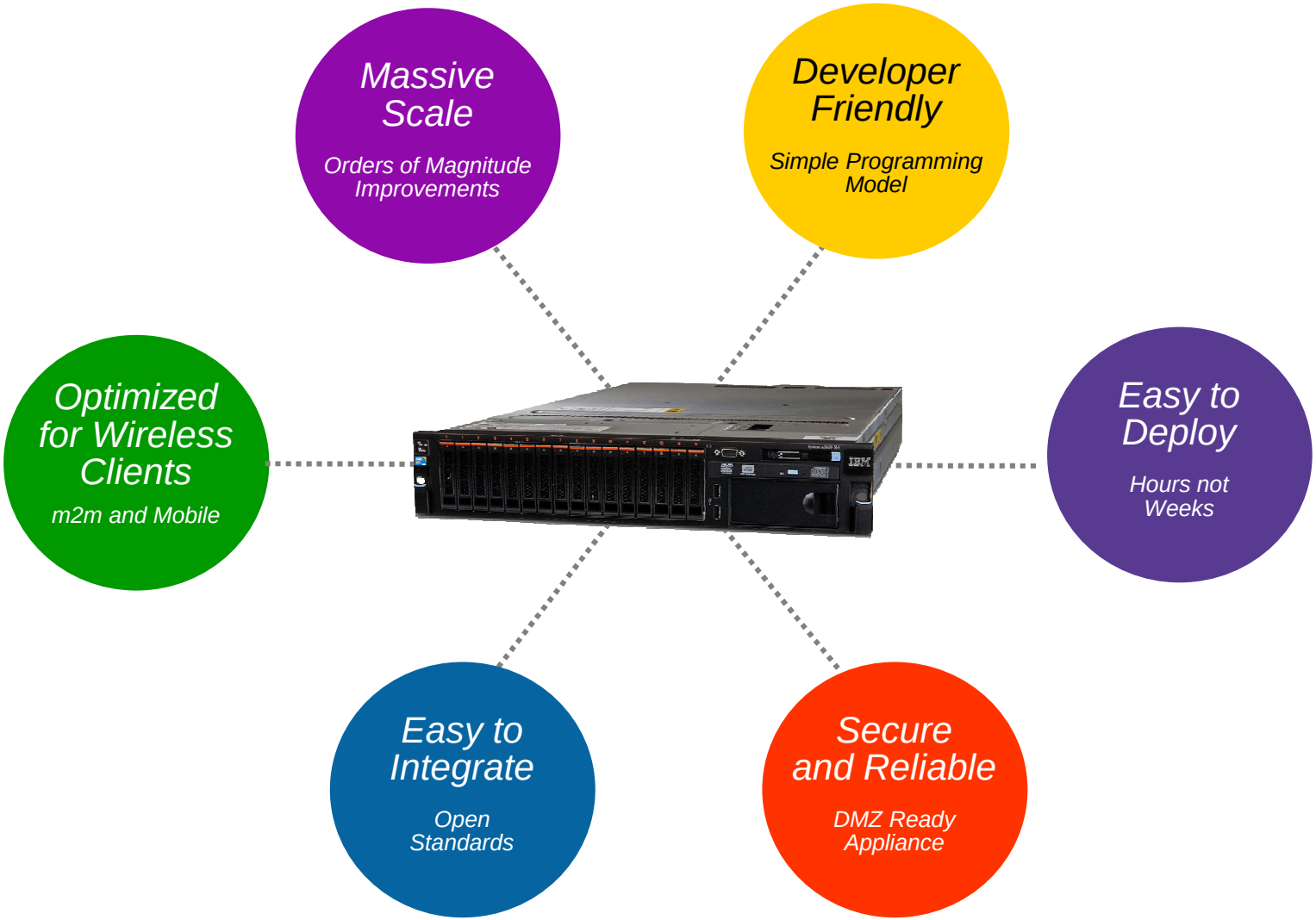


IBM MessageSight: High Availability

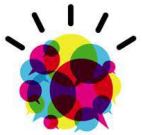
- Failover using RDMA over Converged Ethernet (RoCE)
  - RDMA: Remote Direct Memory Access



IBM MessageSight: Summary



Agenda



- Internet of Things (IoT)
- MQTT
- IBM MessageSight
- Use Cases for MessageSight
  - IOT & API Management
- *Demo*

Use Cases for MessageSight

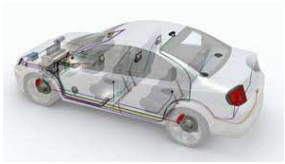
HEALTHCARE
- Patient monitoring,
- Automated patient records

SMARTER CITIES
Traffic monitoring and control

MOBILE
- Event notification
- Data gathering
- Collaboration
- Payments

VEHICLES
- Telematics
- Monitoring/recall
- Security
- Routing

ENERGY/UTILITY
- Metering (energy, temperature)
- Control
- Intelligent prediction
- Analytics and broking



MQTT / WebSockets

Subscribe / Publish

BUSINESS ANALYTICS
- Business agility
- Responsiveness

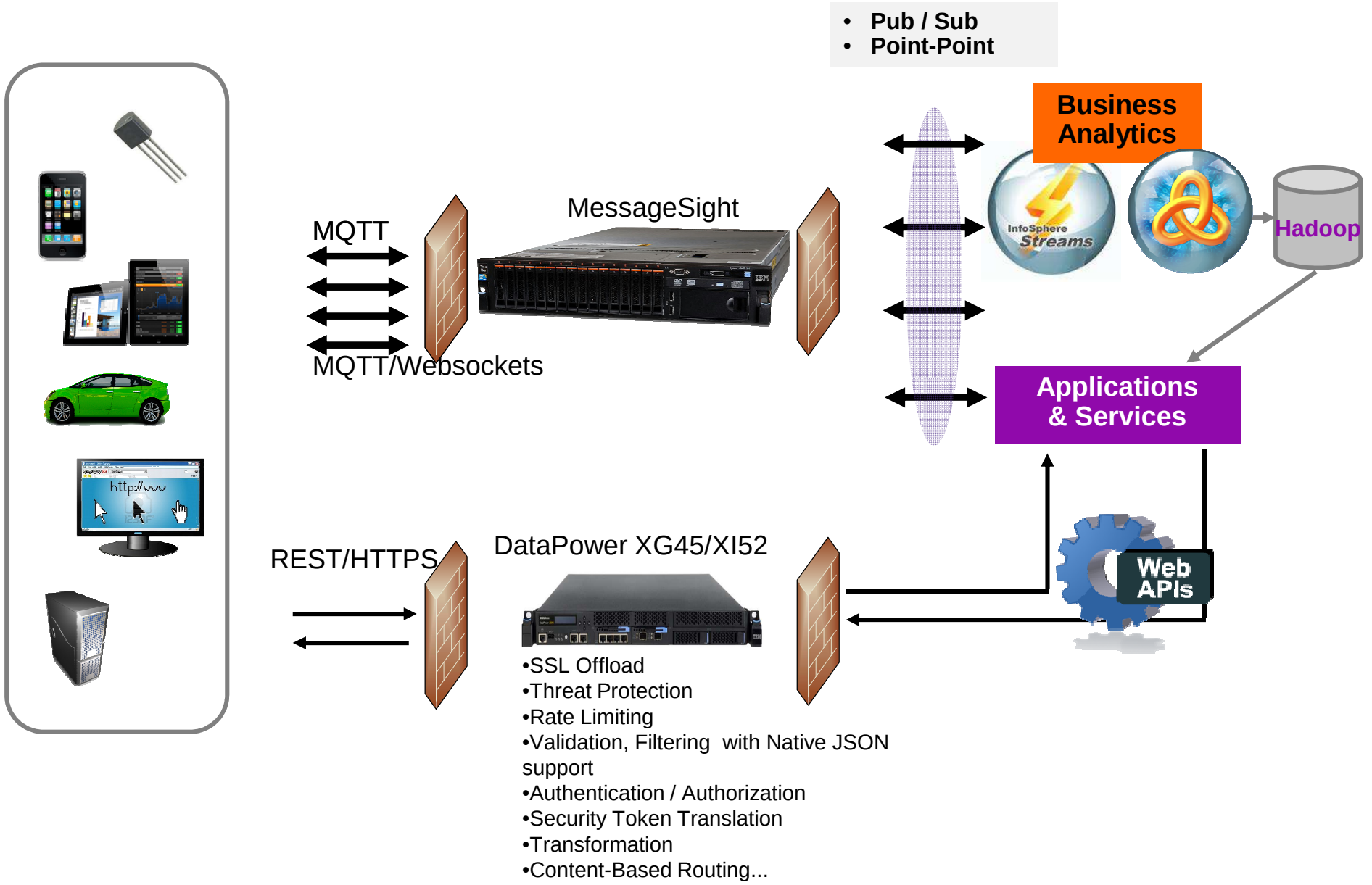
Pub / Sub and Point-Point

Applications

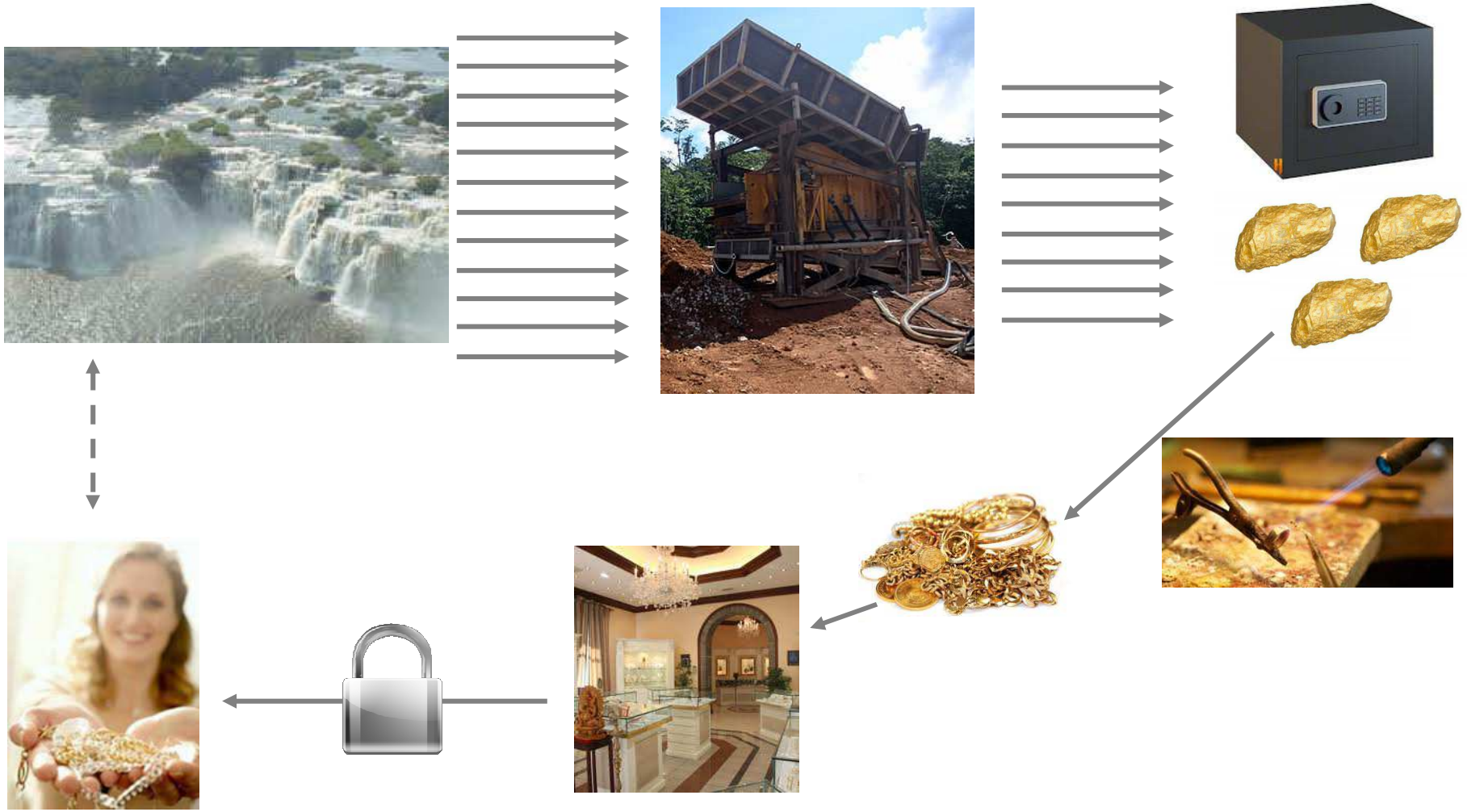
ESB

Events

Internet Of Things & API Management



A new way of making business...

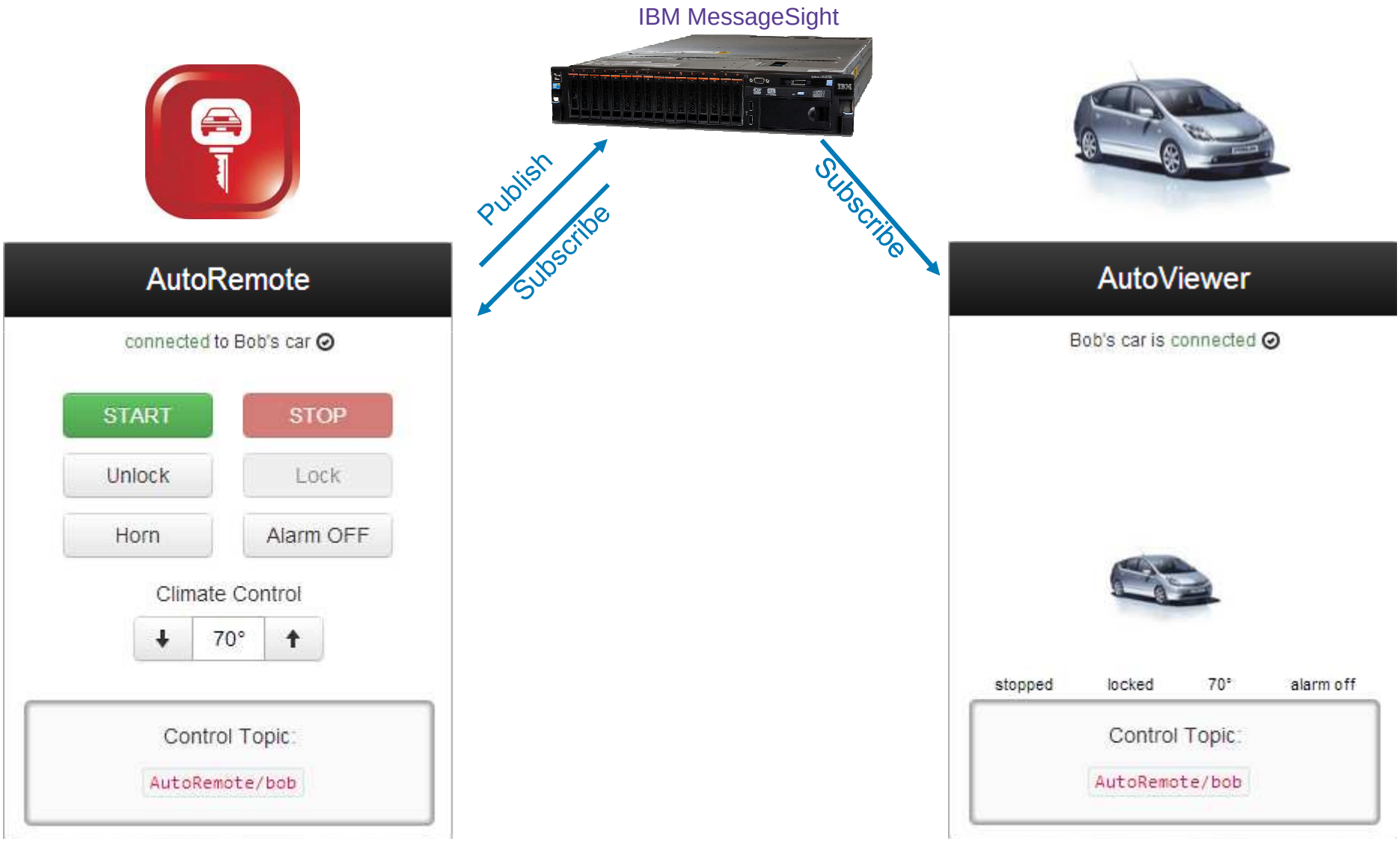


Resources

- MQTT.org
  - videos
    - http://www.mqtt.org
- OASIS & MQTT
  - https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt
- MQTT Specification (V3.1)
  - http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html
- IBM Redbooks
  - http://www.redbooks.ibm.com/abstracts/sg248054.html
- IBM MessageSight
  - Intro: http://www.youtube.com/watch?v=xnOT-c0Qhys
  - Use Cases: http://www.youtube.com/watch?v=bT9vutd6cAc
  - Installation: http://www.youtube.com/watch?v=RZJbqWLj1Zc

Demo

IBM MessageSight Demo



Thanks for your participation !