Software Architecture Conference 2016

Large Scale Connectivity Infrastructure for an Internet of Things Platform

Nir Naaman – IBM Haifa Research Lab

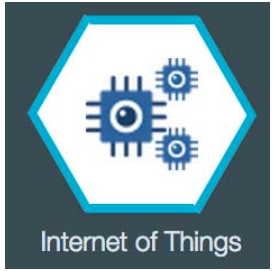
December 19th 2016

IBM

Agenda

- IoT Platform Basics
  - Architecture
  - The MQTT Protocols
  - MQTT Broker
- Large Scale Pub/Sub
  - Clustering
  - Scalable Routing
- Software Architecture
  - Compact subscription representation
  - Fast Lookup
  - Wildcard subscriptions

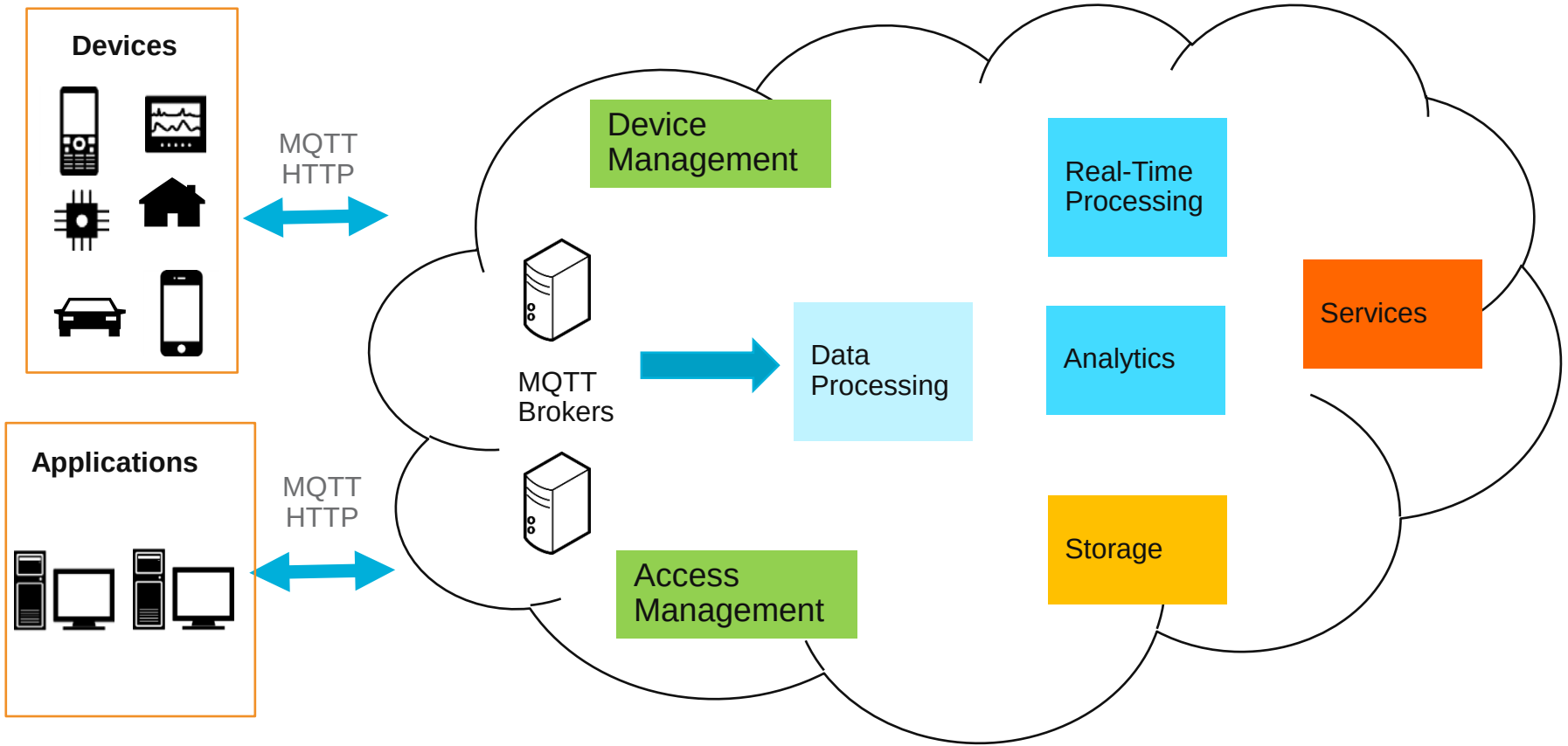
Internet of Things Platform



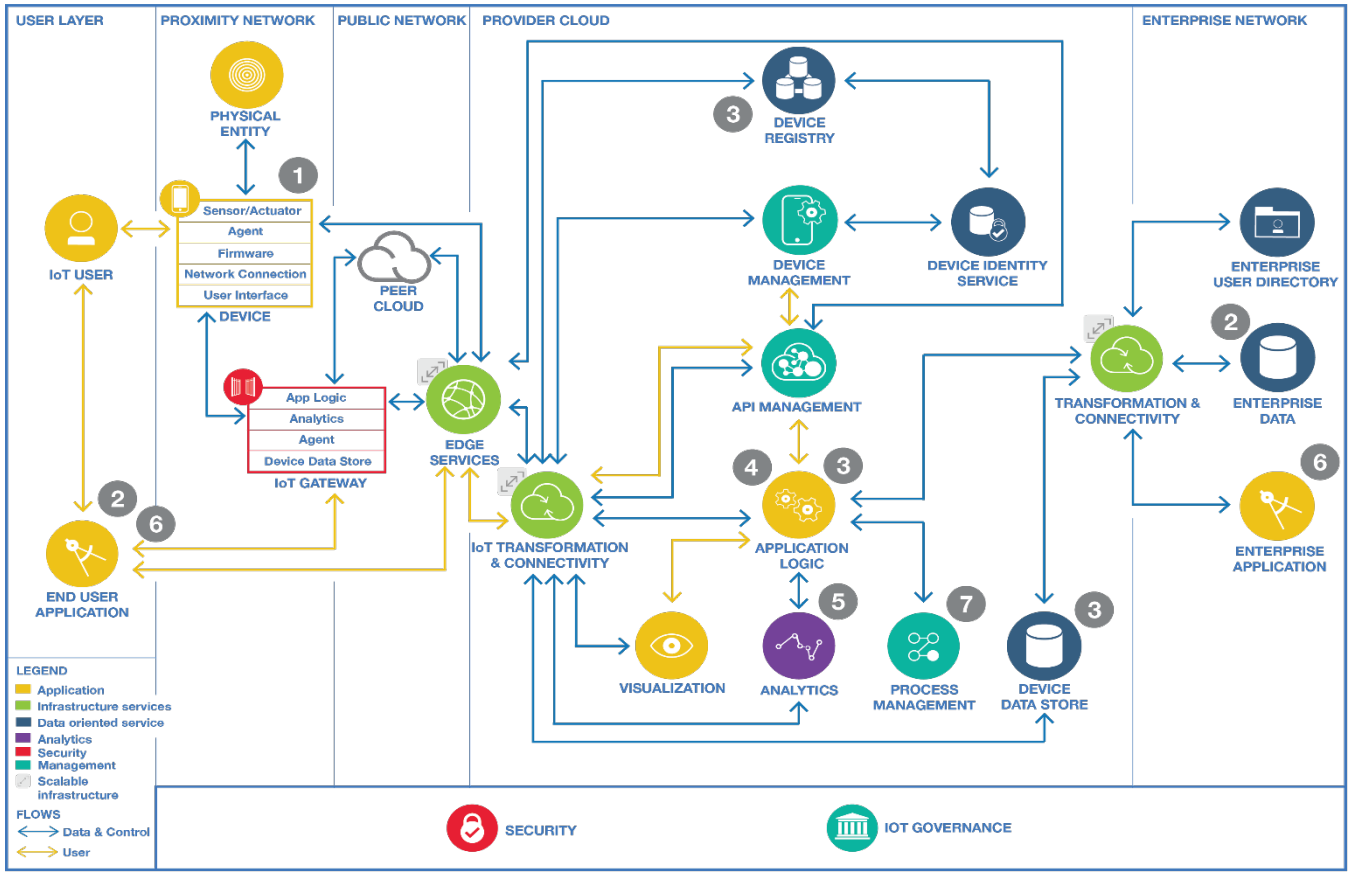
- IoT is expected to grow rapidly in the coming years
  - Over 20 billion IoT units are predicted by 2020
  - Worldwide market for IoT solutions is predicted to grow to trillions of dollars
  - Tens of IoT companies in Israel
- Diverse use cases
  - Connected cars, smart homes, wearables, energy and utility, supply chain, electronics, maintenance, life cycle management, agriculture, smart city, healthcare, and much more
- An IoT Platform offers a common infrastructure on which diverse solutions can be developed
  - Developers can concentrate on building applications/solutions
  - Typically offered as a cloud service
    - Ability to interact with other cloud services
    - Start small and grow
  - Offered by IBM, Amazon, Microsoft



IoT Platform Architecture (simplified)



Smart Home Insurance Architecture



https://developer.ibm.com/architecture/iot

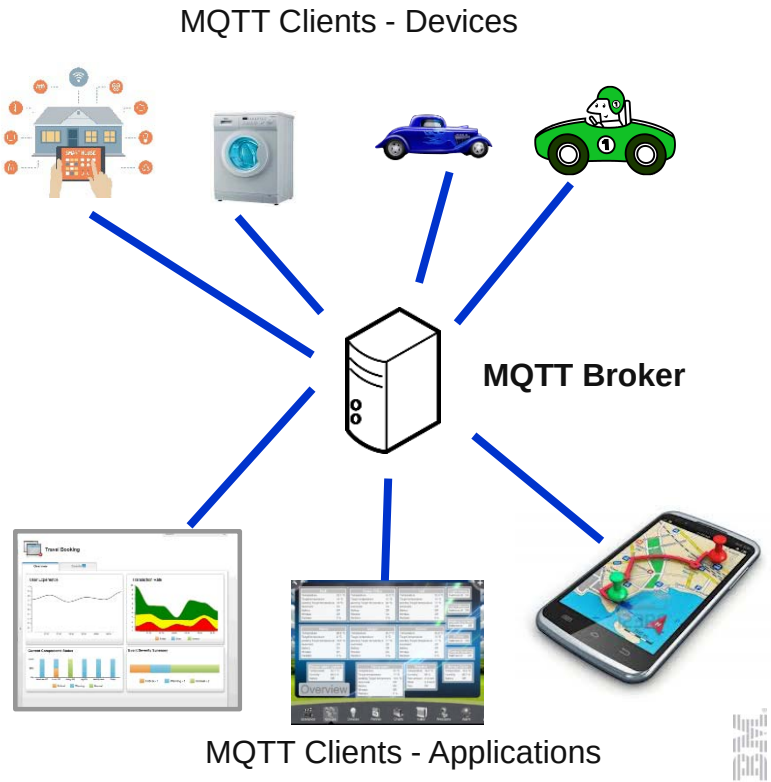
The MQTT Protocol



- A popular IoT and M2M connectivity protocol
  - An OASIS standard
  - Designed for devices with limited resources (bandwidth, memory, power, etc.)
- A lightweight publish/subscribe protocol
  - Publishers publish messages on a specific topic, e.g., /home/monitor/fire/kitchen/sensor5
  - Consumers subscribe to get messages on topics of interest to them
    - Exact subscription: for a single topic
    - Wildcard subscription: for multiple topics matching a certain pattern
- MQTT wildcard subscriptions
  - '+' is used as a single level wildcard, e.g., /home/monitor/+/kitchen/+
    - Can appear multiple times
  - '#' is used as multi level wildcard, e.g., /home/monitor/fire/#
    - Can only appear at the end
- Three levels of Quality of Service (QoS)
  - At most once (QoS 0), at least once (QoS 1), and exactly once (QoS 2)

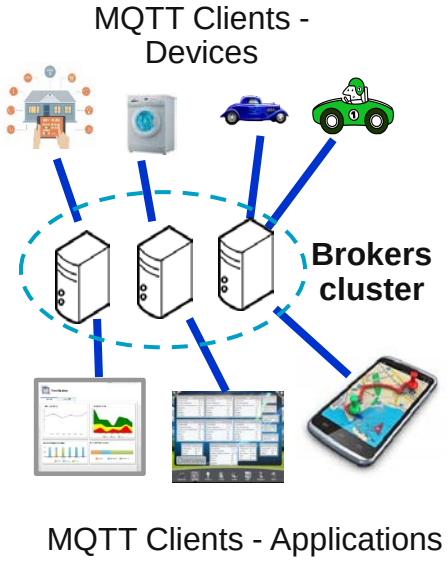
MQTT Broker

- MQTT clients connect to an MQTT broker
  - Hub and spoke architecture
  - A client may publish and/or subscribe
  - Interoperability implementations
- MQTT Broker responsibilities
  - Route messages between publishers and subscribers
  - Store messages for cleanSession=false clients



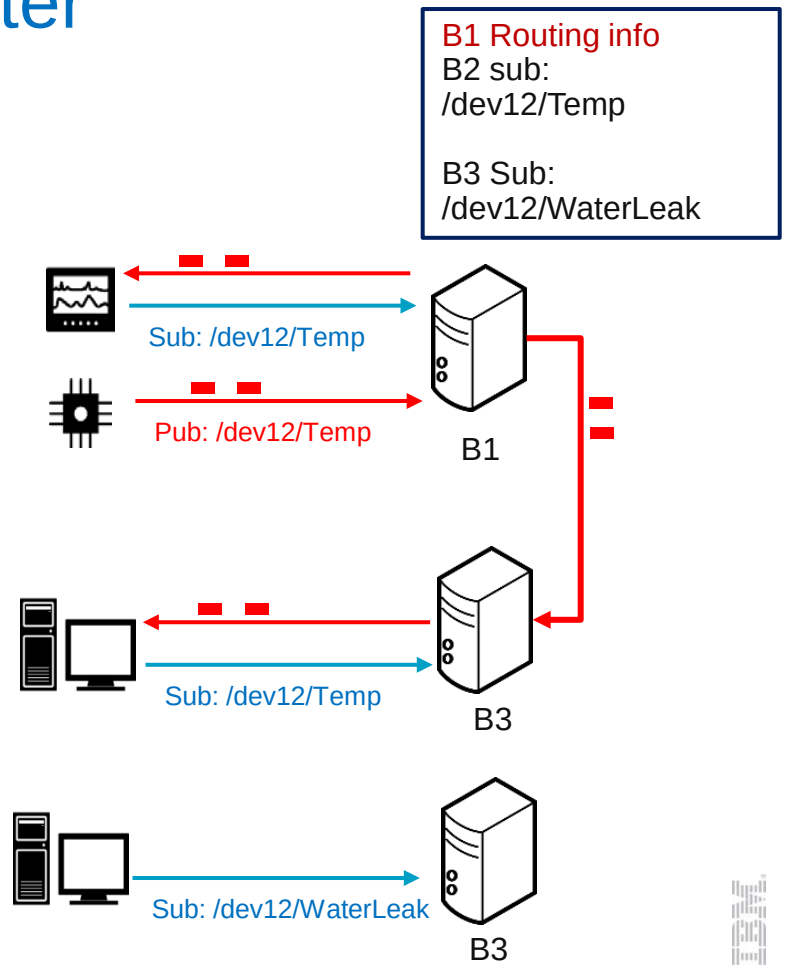
MQTT Brokers Cluster

- The IoT Platform should be able to service a huge number of devices and applications
  - An application may have to service millions of connected devices
- Problem: A single MQTT broker cannot service all the clients
  - Limited in number of connections, memory, bandwidth, etc.
- Solution: Use a cluster of MQTT brokers to share the load
  - An MQTT client connects to the cluster which appears as a single large broker
  - The cluster provides membership and data exchange between brokers
- Many challenges in the transition from a single broker to a cluster
  - Create a common topic space across all brokers in the cluster



Common Topic Space in a Cluster

- MQTT clients
  - Each client is connected to a single broker
  - A client may publish and/or subscribe
- Subscriptions
  - Each broker may have millions of subscriptions
  - Subscriptions may change frequently
- Routing
  - A published message must be delivered to all subscribers in the cluster
  - Message routing between brokers is required
- Challenges
  - Compact subscription storage
  - Fast lookup
  - Efficient routing (avoid flooding)



Routing using Bloom Filters

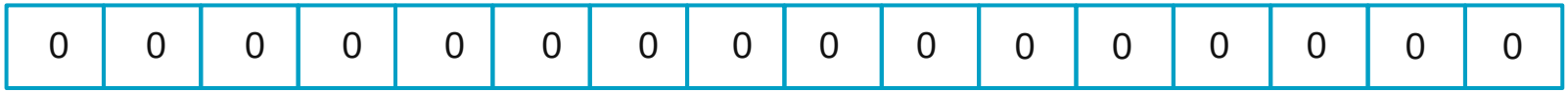
- Each broker records its local subscriptions using a Bloom Filter (BF) and distributes it to all other members in the cluster
  - A counting Bloom filter is used to handle unsubscribe operations
  - The BF size is adjusted to maintain low false positive probability
- Each broker records the latest BF of every other broker in the cluster
- A local message published on topic T is forwarded to all brokers for which T is in their BF
  - Small amount of false positive is possible; such messages are simply discarded by the receiving broker
- Naïve implementation
  - Go over all remote brokers and for each remote broker
    - If T is in the broker's BF add the broker as a forwarding destination for the message
  - For a large cluster this can be expensive



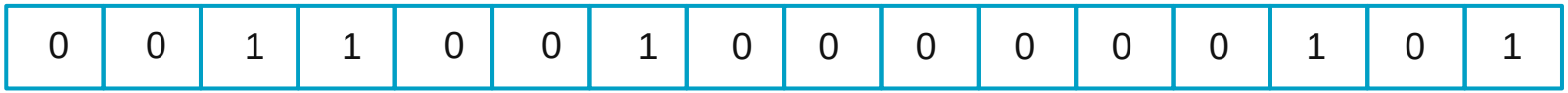
Bloom Filter

m bits in a bit array
k hash functions $H_1,\ldots H_k$

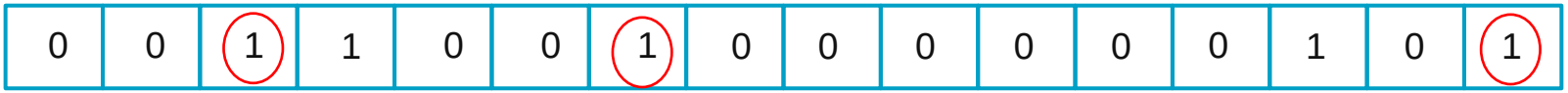
Start with all *m* bits set to zero



To insert an item T set the *k* bits corresponding to $H_1(T), \ldots, H_k(T)$ in the bit array



An item T is in the BF if all *k* bits corresponding to $H_1(T), \ldots, H_k(T)$ are 1



It is possible to have false positive; an item S is not in the BF but all *k* bits are 1

Bloom Filter Properties

- The false-positive probability *p* depends on *k*, *m*, and *n* the number of items in the BF
- Using *k*=6 and *m*=10*n* result in false positive rate of about 1%
- Counting Bloom filter for easy insert and delete

To insert an item T add 1 to the *k* positions corresponding to $H_1(T)$, …, $H_k(T)$ in the bit array. To delete an item subtract 1 in each position

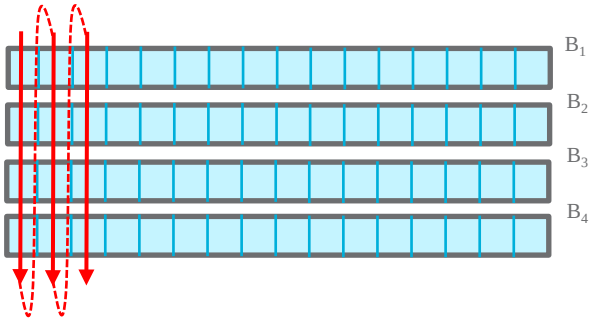
0	1	4	0	0	3	1	0	0	2	0	0	1	0	0	1

A Bloom filter is easily derived from the counting Bloom filter

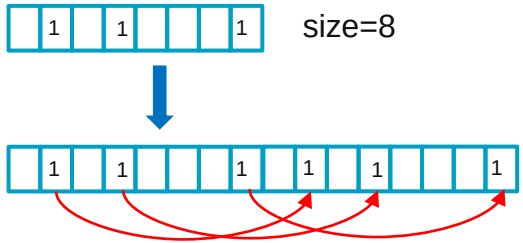
0	1	1	0	0	1	1	0	0	1	0	0	1	0	0	1

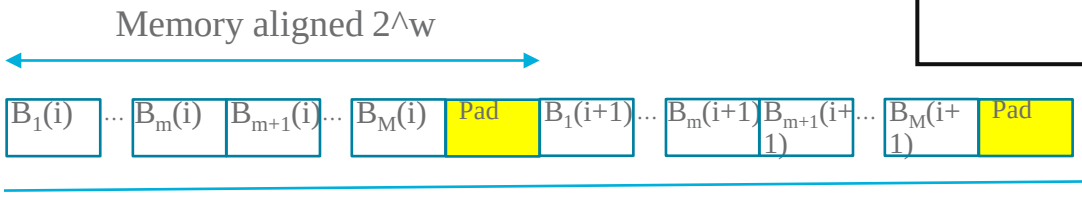
Fast Lookup using Bloom Filter Interleaving

- Bloom filters of all M remote brokers are interleaved
  - [$B_1(1)|B_2(1)|\ldots|B_M(1)|pad|B_1(2)|B_2(2)|\ldots|B_M(2)|pad|\ldots$]
  - In each memory access we obtain the values of up to 64 brokers
  - Bitwise AND on the *k* vectors produces a final vector with the matched brokers



- Extending Bloom filter size
  - A BF of size 2^x can be extended to size 2^(x+1) by replicating the BF





Wildcard Subscriptions

- Should we expect a very large number of wildcard subscriptions?
  - Applications tend to make few wide wildcard subscriptions
    - Best handled by standard topic tree approach
  - If devices use wildcard subscriptions the number can be large
    - Topic tree approach may be inefficient
- The number of wildcard subscriptions **patterns** is expected to be small
  - The number of levels where a wildcard may appear is small
  - For example, devices subscribe to the same information but with a different device ID
    - /deviceId/device0000001/command/#
    - /deviceId/device0000002/command/#
    - Same pattern – # at level 4
- We use a Wildcard Bloom Filter (WBF) to represent a large number of wildcard subscriptions that match the same pattern (wildcard positions)
  - If the number of wildcard subscriptions is small we use the topic tree approach
  - A broker may need to use several WBFs to represent its subscriptions

Wildcard Bloom Filter (WBF)

- Preparing the WBF
  - The WBF consists of the pattern P and the BF
  - A wildcard subscription is inserted to a BF as is
  - The pattern P describes what levels should be replaced with a '+' and the '#' level
    - Example 1: P={2, 4; -1} meaning '+' at levels 2 and 4 and no '#'
    - Example 2: P={-1; 3} meaning no '+' and '#' at level 3
  - Send the WBF to all remote brokers
- Using the WBF
  - To check if a published topic T matches a WBF perform the following
  - Apply the pattern to the topic T' = P(T)
    - Example 1: T=/aa/bb/cc/dd/ee and P={2, 4; -1} ➔ T'=P(T)=/aa/+/cc/+/ee
    - Example 2: T=/aa/bb/cc/dd/ee and P={-1; 3} ➔ T'=P(T)=/aa/bb/#
  - Match T' against the BF
  - In case there are more patterns repeat for each pattern (or a match is found)

Summary

- An IoT Platform provides the infrastructure for divers IoT use cases
  - Provides connectivity, storage, analytics, security, management, and more
- Building a scalable IoT platform presents many challenges
- One of the challenges is supporting very large scale publish/subscribe
  - Use a cluster of brokers that should appear as a single large server
- Presented methods for compacting subscription representation
  - Both exact and wildcard subscriptions
- Maintaining high performance at scale

THANK YOU