Following are things to keep in mind:

1. IoT Hub hashes by deviceID, making sure all messages from the same device are sent to the same partition ID; A partition therefore can contain multiple messages from multiple device IDs unlike Event hubs messages are not automatically distributed to partitions based on round robin (in absence of a partitionKey).
2. There are three common types of patterns:
   1. One where there a few number of devices that send large amounts of data
      1. Such as field gateways, aggregating and compressing data, which is sent to them every 10 seconds/minutes
   2. Second where there are large number of devices sending small amounts of data
      1. Such as monitoring wells, sending data once a day/a few hour etc.
   3. Thirdly, where there are large number of devices sending large amounts of data
      1. Such as light weight sensors sending data every 10 seconds

**Immediate ask from Engineering**

Is there a way to predict the partition ID from Device ID? What is the hashing algorithm? If we know this we can have the admin app reading new DeviceID registries and generating new instances with the DeviceID. (Pattern A).

**Changes we could make:**

1. *Modify the ingestor router service - catering to pattern b*

The existing application architecture’s ingestor router services will work well for pattern c & b. Where it scales based on partitions and incoming data is mostly distributed evenly.

Patten **a** is also very common (very few partitions), we should tweak the Ingestor router service to extend across the 20 consumer groups, and partitions. Since one can have at most five connections per partition per consumer group. The ingestor will be able to scale better.

1. *Modify the Admin application – to remove the tenant complexity*

I think we should have the first version without the tenant complexity, we can have “data processor” endpoint to which data is pushed to. And have some common data processing services available which can be deployed based on a configuration. Have these extendable so anyone can extend it to suit their needs.

1. *Modify the Admin application – to extend to 20 consumer groups*

The admin application could create instances specifying deviceID & consumer group, hence scaling beyong a single partition – single consumer group

1. *Add “pre-built” data processing service types*

We could have a couple of pre-built data processor services such as:

1. De-compression
2. De-aggregation
3. De-compression & de-aggrgation
4. Storage – Blob
5. Storage - Table
6. Storage – Document DB

IoT Ingestor Goals:

Goals:

1. Getting a Service Fabric ingestor for IoT that can be used in hackfests as well as something we can recommend users to go live with if they choose
2. It needs work with any amount of partitions and consumer groups. So the team with one partition and one consumer group can use it as well as the one with dozens (or hundreds).
3. It needs to be easy to use and documented well.
4. It needs to be tested with lots of events
5. It needs to be extensible via custom modules that can easily be plugged in. I’m thinking a new function is a new Service Fabric service. For example, if I wanted my main “reader” to get the messages and then send off to another integration point that should be easy. We could even look at the new workflow SDK from Microsoft.
6. Decide where to host this on github. Iot-samples or in a new repo. New repo would be easier to track issues and PRs for it and it won’t be muddled with the rest of the samples. But having in iot-samples get us more visibility for people that already know about iot-samples.