Mesh Systems lights up the market with IoT-based Azure solutions

April 30, 2018

This post was authored by Michael Thomassy, Stas Kondratiev, and Ed Price from Microsoft, in conjunction with Bill Craun, Uriel Kluk and TJ Butler from Mesh Systems.

This post is part of a series on customers who've participated in the Azure Service Fabric (https://azure.microsoft.com/en-us/services/service-fabric/) preview program over the last year. We look at why they chose Service Fabric, and take a closer look at the design of their application, particularly from a microservices perspective. You can read the first post in this series, Service Fabric Customer Profile: TalkTalk TV (https://blogs.msdn.microsoft.com/azureservicefabric/2016/03/15/service-fabric-customer-profile-talktalk-tv/) .

In this post, we profile Mesh Systems, their NetLink solution, and how they designed their IoT lighting control application.

Mesh Systems provides IoT software and services for enterprise IoT customers. They have spent over 11 years creating the building blocks, tests, and process to model complex enterprise IoT applications with telemetry and control workflows. Using these core building blocks, Mesh Systems has been able to significantly reduce the time-to-market while increasing reliability for their customers. For example, by migrating from bare metal machines to Azure services, Mesh Systems is now able to provide customer-specific solutions in a fraction of the time, often in weeks rather than months. When Mesh Systems needs to onboard a new customer, their building blocks, which are built on Azure platform as a service, are a big part of what enables them to deliver the solution quickly.

They leveraged their IoT cloud platform, MESHVista, to build a lighting control application for WLS Lighting's NetLink product. This solution reduces energy consumption and the overall operational costs of parking lot lights, as shown in the photos below with a Honda dealership customer.

It does this by providing very granular control of individual light fixtures that were previously controlled simply by photocells. This system allows property owners to contour the lights to meet very specific needs and omit unnecessary lighting, which results in tremendous cost savings. NetLink estimates the system can yield ROI in less than 18 months. The system currently controls around 125,000 light fixtures and is growing rapidly.

" Service Fabric has enabled us to take a big step forward in providing timely and reliable managed cloud services for our customers."

—Uriel Kluk: CTO
Mesh Systems

Figure 1: Mesh Systems' lighting control application in use at a car dealership lot

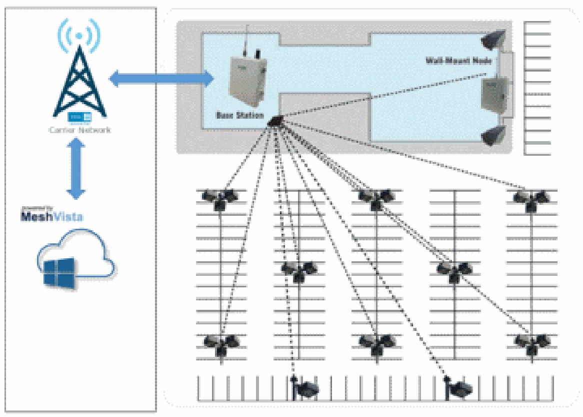

Figure 2: Telemetry about each light is transmitted over carrier networks to Mesh Systems IoT solutions

Mesh Systems developed the system several years ago as a monolithic application on bare metal machines, and the system was nearing its expansion ceiling. Mesh Systems wanted not just scalability but a flexible platform for building and composing microservices with an improved DevOps life cycle.

A microservices architecture for scalability and flexibility

Mesh Systems was already moving in the direction of building smaller, composable services in an effort to make the distributed system significantly more manageable. Mesh Systems used Cloud Services (Azure PaaS compute) for earlier IoT systems, and often saw a small fraction of resources being used across a group of three services, per concern, to meet SLAs.

This approach often resulted in combining concerns to better utilize the cloud service resources, which complicated life-cycle management and added unnecessary dependencies. Service Fabric microservices offered several advantages that addressed these needs. Mesh Systems' reliability and time to market are directly proportional to code reusability and implementing simple, stable components. Service Fabric provided a great foundation on which to build these components and meet those tenets.

Mesh Systems built a new version of the solution using Service Fabric, following an emerging pattern in the IoT space wherein physical devices have a virtual mirror of their physical state in the cloud alongside the code that executes business rules against that state.

The key design tenet Mesh Systems followed was to develop a microservices architecture with a clear separation of concerns. It was critical to deploy granular changes to the overall system, so Mesh Systems broke the legacy monolithic application down into small units of isolated functionality. This made it possible to version them independently of one another, which allowed for more agility and incremental deployments of functionality in a controlled and predictable way.

The advantages of Service Fabric and microservices

IoT systems involve several steps of closely related, back-end processing. For example, the critical path for a typical system involves payload processing, business rules execution, data archiving, and notifications, just to name a few. The engines that run each functionality often have their own life cycles and performance/scalability requirements. The legacy system was a monolithic engine that required planned downtime for the entire system to update just one of the components.

Mesh Systems chose Service Fabric to break apart their legacy application into microservices, and achieve the following:

- **Isolated functionality:** By isolating the functionality, the services have well-defined inputs and outputs, which make unit and load testing significantly more practical. Isolation also facilitates separate life cycle management. Mesh Systems can now update one service without impacting another.
- **Scalability:** Using microservices allows Mesh Systems to manage each process independently. For example, they can scale out the payload processing independent of notifications. They can update any part of the process without downtime.
- **Flexibility of the Azure cloud platform:** This allows the cluster itself to be scaled dynamically, according to demand. Nodes in the cluster can be scaled up and out independently of the applications running on those nodes. Applications can be scaled on the infrastructure in a way that maximizes the available resources.
- **Incremental deployments:** Applications can be deployed in a controlled manner: one at a time, in groups, or all at once, depending on the stage of deployment. It's possible to automate complex deployments by using ARM (Azure Resource Manager (https://azure.microsoft.com/en-us/features/resource-manager/)) templates.
- **Reduced costs:** For service-dense applications, it's possible to reduce the overall cost of compute required to achieve high availability. The existing Mesh Systems' implementation used five traditional cloud services types; it would have required three instances of each cloud service compute node to achieve the desired SLA. However, with Service Fabric, Mesh Systems was able to deploy only five nodes, have arguably better SLA, and actually reduce the cost for compute.
- **Improved resilience:** Service Fabric provides something similar to RAID functionality for applications with its stateful services. Applications can failover from a failed node to a hot standby replica very quickly. Groups of applications can be migrated from one node to another during maintenance such as for patching or planned restarts.

Integrating Azure Services

Mesh Systems IoT solutions leverage many Azure services like a cloud toolbox. Solutions are customized to match the business requirements of each system.

The following diagram demonstrates the Azure services leveraged by Mesh System’s MeshVista platform to build the NetLink solution.

“ Service Fabric is a perfect fit to our building blocks, decoupling the responsibilities into very well-defined tasks.”

—Uriel Kluk: CTO
Mesh Systems

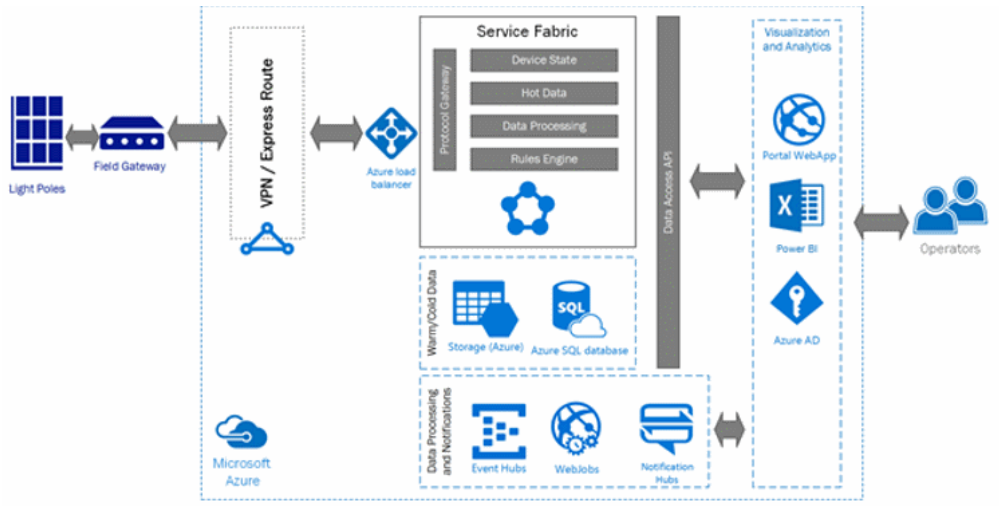

Figure 3: The Azure services leveraged by Mesh Sytems for the NetLink solution

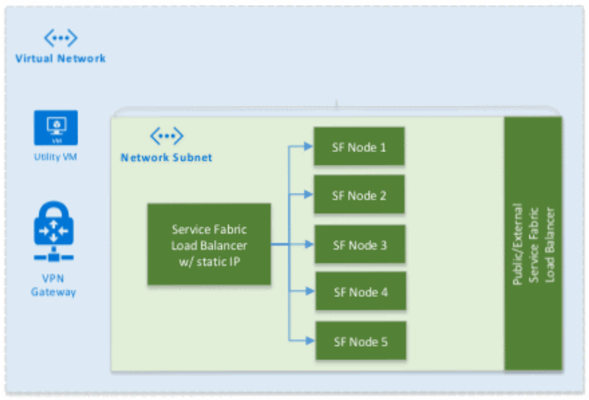

Figure 4: Mesh Systems' two resource groups (RGs)

Mesh Systems took the approach of evolving their existing Azure solution to move to Service Fabric to reduce cost as well as complexity. Their final solution uses the following Azure services:

- **Service Fabric** hosts the microservices applications that are primarily responsible for the back-end data processing, state management, and rules engines. The service here act on and store hot data.
- **Event Hubs** (https://azure.microsoft.com/en-us/services/event-hubs/) acts as a message broker by decoupling the inbound message producers from the consumers.
- **SQL Azure Database** (https://azure.microsoft.com/en-us/services/sql-database/) ensures that metadata and hot data is available for relational queries, which typically feed an on-demand UI such as a web or mobile interface.
- **Storage** (https://azure.microsoft.com/en-us/services/storage/) is probably the most commonly used service. It facilitates a combination of warm, and cold path operations. Blobs are the core of a high scale, low-cost *cold* data archival. Table storage is a scalable no-SQL service for storing data in a fast access, low-cost repository.
- **Virtual machines** are the foundation for compute workloads, such as Service Fabric clusters and Elasticsearch for monitoring and diagnostics.
- **Virtual networks, VPN/Gateways, and Express Route**bridge the gap between physical devices and existing cloud services. Many solutions are leveraging the cellular network for WAN connectivity. Azure provides several options for making the hop between on-premises and the cloud. They also help connect services from one Azure regional data center to another.
- **Notification queue** notifies subscribed mobile users of anomalies via native push methods.
- **Azure Active Directory** (https://azure.microsoft.com/en-us/services/active-directory/) facilitates service principal-based authentication between Azure applications.
- **Web Apps** are the primary user interface for IoT portals. In the case of NetLink, end users interact with the portal to modify lighting schedules and get energy savings reports.
- **Power BI** provides advanced users, typically the commercial product manufacturers, with a means to build self-service, ad-hoc reports.

Mesh Systems uses Azure services as building blocks to create tailored IoT solutions that meet the specific business needs of each product. Smaller proof-of-concept systems start off with the minimum services in place, while large, full-scale production applications typically leverage the full stack of services.

Azure deployment with resource groups

The Azure ARM model is used to deploy resource groups for the infrastructure required for the IoT lighting solutions.

As shown in the diagram above, Mesh Systems created two ARM templates to deploy two resource groups that manage infrastructure with separate life cycles.

- **Virtual Network Resource Group (RG):** This facilitates a persistent connection to a cellular carrier VPN. It also:
  - Establishes the parent network.
  - Connects the VPN Gateway to the carrier to communicate telemetry from lights.
  - Has a reserved NIC and a static IP for use by the Service Fabric load balancer that's created by the second template.
- **Network Subnet/Service Fabric Resource Group (RG):** This primarily facilitates the Service Fabric cluster. It also:
  - Creates the subnet within the parent network that's created by the first template.
  - Creates the load balancer tied to the NIC that's created by the first template.
  - Creates the 5 VMs that make up the Service Fabric cluster.

The key purpose of using ARM templates for the NetLink implementation was to facilitate an immediate need to create the Azure networking infrastructure well ahead of and/or in parallel to the Service Fabric cluster deployment. It was important to maintain the connection with these resources even during development, while the Service Fabric cluster deployment was changing. One team was able to work on establishing a stable connection to the cellular carrier while the other developed the Service Fabric cluster. The second group provided an easy way to create, destroy, and recreate the Service Fabric cluster during early iterations. In addition, this group was used to create an exact duplicate of the cluster for load testing purposes. In a disaster recovery situation, it would be possible to leverage the ARM template to recreate the exact deployment in a very short timeframe.

Mesh Systems has since embraced ARM templates using them to setup test environments and create reusable infrastructure contained, deployed and monitored as cohesive groups.

Mesh Systems applications and microservices

The MeshVista microservices are packaged as a Service Fabric application with multiple services as illustrated in the diagram below. Note that Event Hubs serve as message brokers and ingress buffers to decouple communication between the services, making them loosely-coupled units of code.

- **Protocol Gateway service:** A lightweight stateless service that exposes Web API endpoints via OWIN, which acts as a front door through which the devices send telemetry data. It does the communication protocol handshaking, validates telemetry transmission packets, and queues them up in a message broker—Event Hub—to be processed further by downstream systems
- **Payload Processor service**: A stateless service, built on top of EventProcessorHost (https://msdn.microsoft.com/library/azure/microsoft.servicebus.messaging.eventprocessorhost.aspx) , responsible for inbound payload routing from Event Hub. It persists the payloads to a table storage for auditing and logging purposes, and, at the same time, it dispatches them to the Device Actors service.
- **Device Avatars actors service**: Stateful actors are used to model a hierarchy of physical devices deployed in the field. The relationship between the physical devices and their digital counterparts is key. There are device type actors and data point actors. It takes a collection of these actors to comprise the digital mirror of the physical device in the field. For example, each physical device contains multiple sensors that produce a collection of analog and digital telemetry values. Each sensor value is modeled as a data point actor. Actor-specific business logic can be triggered as an event flows through the hierarchy.

"I sleep better knowing that I have programmatic infrastructure sitting in source control. ARM templates provide the documentation and programmatic recovery that highly distributed applications demand."

—TJ Butler: Chief Software Architect
Mesh Systems

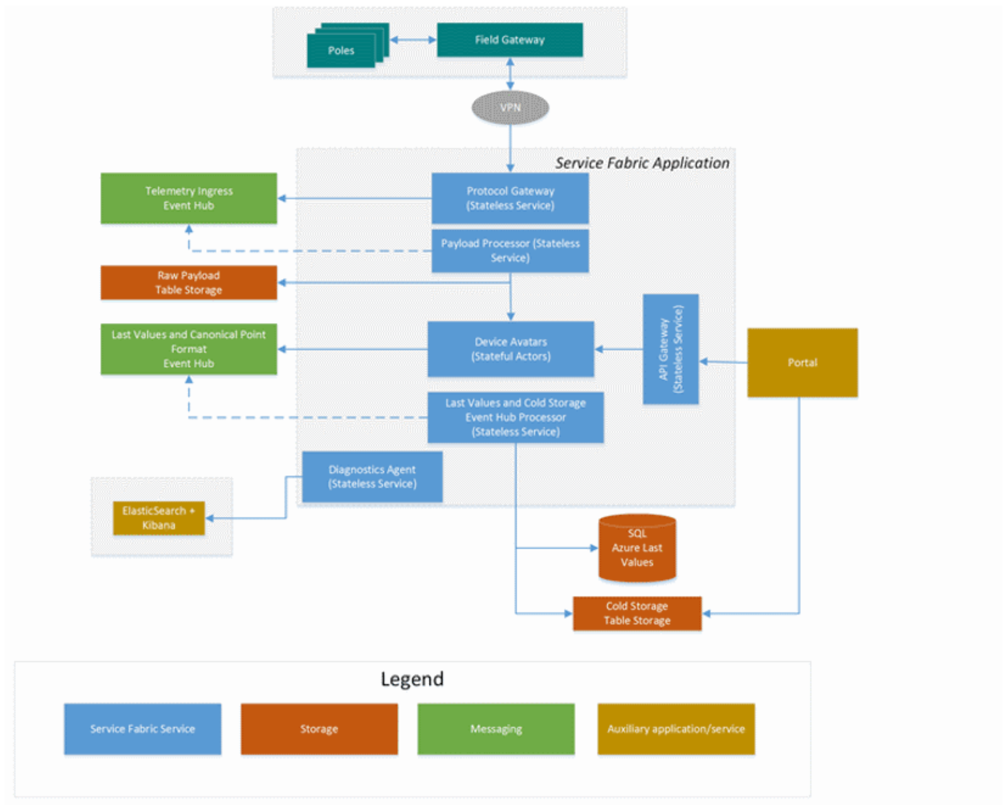
Poles
Field Gateway
VPN
Service Fabric Application
Telemetry Ingress Event Hub
Protocol Gateway (Stateless Service)
Payload Processor (Stateless Service)
Raw Payload Table Storage
Last Values and Canonical Point Format Event Hub
Device Avatars (Stateful Actors)
API Gateway (Stateless Service)
Portal
Last Values and Cold Storage Event Hub Processor (Stateless Service)
Diagnostics Agent (Stateless Service)
ElasticSearch + Kibana
SQL Azure Last Values
Cold Storage Table Storage
Legend
Service Fabric Service
Storage
Messaging
Auxiliary application/service

Figure 5: Mesh Systems microservices on Azure Service Fabric

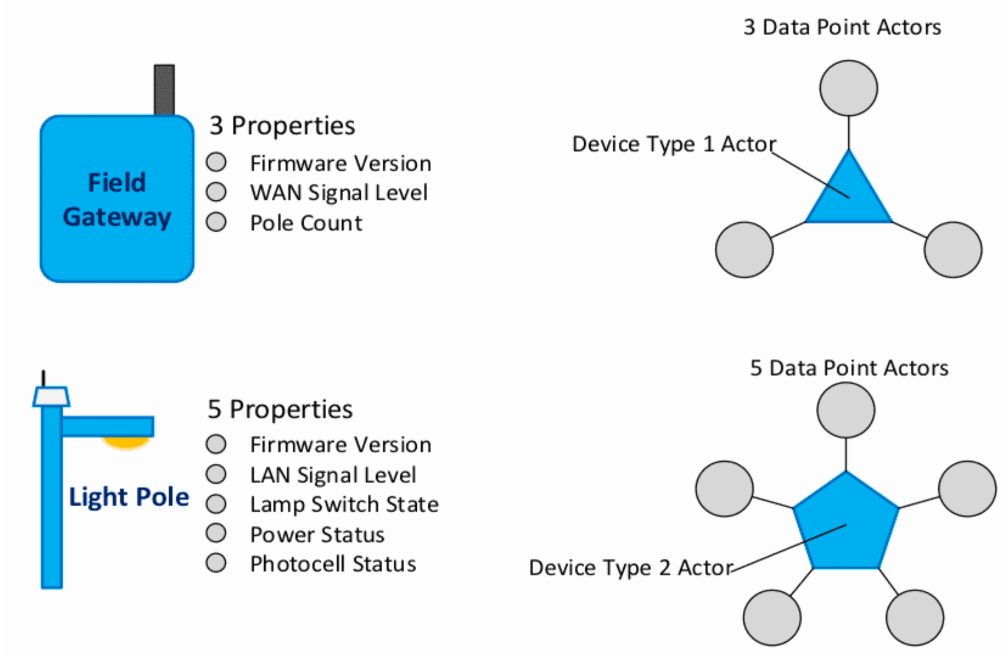

Figure 6: Mesh Systems physical device actor models

This design is pretty granular, and it comes with certain trade-offs, including a need for more memory and CPU to accommodate all the actors. Nevertheless, Mesh Systems decided to create this system to easily run business rules against each metric within a data point actor. For example, a data point actor checks if an inbound value hits a certain threshold, and if it does, a notification is generated and queued for an external SMS service that promptly notifies Mesh Systems' operators about abnormal conditions in the field. The operator can then use the portal to query—via API Gateway service—the current state of the collection of actors to either resolve the situation remotely or gather troubleshooting data to make a site visit as efficient as possible.

As the last values are updated within point actors, they're routed to another message broker, the Last Values and Canonical Point Format Event Hub service.

- **Last Values and Canonical Point Format Event Hub service**: This is another stateless service, which fetches last values from the Event Hub and routes them down both the warm path to a SQL Azure Database (that stores data for the last few weeks) and the cold path to Table Storage (that stores them indefinitely).
- **Diagnostics Agent service**: With logging and instrumentation for any complex application, it's critical to identify and resolve anomalies. For diagnostics, Mesh Systems adopted Elasticsearch for many of its systems, and also integrated it with Service Fabric applications. They did this by running a single stateless service on each node that collects the ambient data, like performance counters and ETW traces. The diagnostic agent application collects the log information and pumps it to their Elasticsearch cluster.

Leveraging ETW traces allows Mesh Systems to decouple log messages from the logging engine. The core applications can move freely from one node to another in the cluster and the diagnostic agents just keep pumping messages regardless of which apps happen to be on that particular node.

Key Service Fabric features

Actors

Act as in-memory device avatars. This physical/digital mirror of device state provides a decoupled state, along with the ability to run logic that pertains to a single device. Applications that need access to that state can do so without querying the device itself. Service Fabric's actor model enables developers to implement this in parallel across a massive amount of devices.

Service Fabric provides an infrastructure that helps safely guide development of highly asynchronous processes. The technology that enables the device avatar and its long-term life cycle management was the primary reason Mesh Systems chose to use Service Fabric. Replication of the actor state across replicas of the node cluster gives them the consistency and reliability they need to provide high-level SLAs that their customers expect from a cloud solution.

Stateful actors are used for device state and logic. Actors are created on demand and spread across available nodes automatically. Again, the replication of their state across nodes is a key benefit.

Stateless actors are used for elastic processes around data ingestion, event processing, and persistence to external databases such as Azure database. These actors provide very tight elastic scale that helps balance spikes or lulls in load. This is particularly well suited for high-scale, dynamic load from IoT devices in the wild. When load spikes, the stateless actors activate quickly upon instantiation and they dissolve just as soon as their work is completed.

Lifecycle management

Application-independent updates, on-the-fly, is another reason Mesh Systems is leveraging Service Fabric. This feature is absolutely critical for maintaining a highly distributed application. Service Fabric has made the process for making changes to a production system significantly less painful and significantly less risky. The DevOps experience of publishing and watching the updates roll through the cluster is a huge leap forward in how they manage interrelated applications, especially for updates that affect multiple services.

Stateful actors, which are used for device avatars, can maintain state across upgrades or horizontal scale-out of the entire cluster as demand grows. This significantly lowers the risk and impact of maintenance.

Application-specific horizontal scalability

Apps and the infrastructure on which they run can be scaled independently. This allows Mesh Systems to maximize the resources within a cluster by adding more application instances to a cluster VM rather than programmatically trying to get as much asynchronous functionality out of a single cloud service as possible. When the cluster approaches capacity, they simply add another node to the cluster and Service Fabric redistributes the applications, taking advantage of the newly available resources.

This can be extended further using node placement constraints, which allow Mesh Systems to contour the nodes of a cluster to the needs of the applications they'll host. For example, they can have three nodes of premium compute for mission-critical services sitting in the same cluster as three smaller, cheaper nodes that support user interface services.

Reliable collections

To minimize the load on SQL Azure Databases, Mesh Systems chose to implement a SQL broker that periodically caches the most heavily-accessed metadata tables. Service Fabric’s reliable collections provide a local, replicated cache of frequently-accessed items, in memory.

This has been a challenging issue, which typically resulted in hosting a cache service, such as Redis, to make this information accessible to more than one distributed component of the solution. Now, the replicated collections are available to every application within the Service Fabric cluster. Mesh Systems considered exposing the same collection to applications external to Service Fabric via an API application sitting inside the cluster.

Summary

Mesh Systems started out with a monolithic application that was too complex, and unable to accommodate the needs of their growing business. Also, Mesh Systems’ earlier IoT system often used only a small fraction of their available resources. They wanted scalability, and a flexible platform with an improved DevOps life cycle. And of course, they wanted to implement smarter use of resources.

So Mesh Systems built a new version on top of Service Fabric. They created a microservices architecture with a clear separation of services and small units of isolated functionality, which allowed for more agility and incremental deployments in a controlled and predictable way.

By breaking their legacy application into microservices, Mesh Systems has been able to reduce the complexity of services, improve scalability, take advantage of the flexibility of the Azure cloud platform, and reduce costs.

“ The ability to provide rolling updates to a distributed production system while maintaining state is game-changing.”

—Bill Craun: Principal Software Engineer
Mesh Systems