Measures of Network Centricity for Edge Deployment of IoT Applications

Dinesh C. Verma
IBM T. J. Watson Research Center
Yorktown Heights, NY, U.S.A.
dverma@us.ibm.com

Geeth de Mel
IBM Research UK
Warrington, UK
geeth.demel@uk.ibm.com

***Abstract*—Edge Computing is a scheme to improve the performance, latency and security guidelines for IoT applications. However, edge deployment of an application also comes with additional complexity in management, an increased attack surface for security vulnerability, and could potentially result in a more expensive solution. As a result, the conditions under which an edge deployment of IoT applications delivers a better solution is not always obvious. Metrics which would be able to predict whether or not an IoT application is suitable for edge deployment can provide useful insights to address this question. In this paper, we examine the key performance indicators for IoT applications, namely the responsiveness, scalability and cost models for different types of IoT applications. Our analysis identifies that network centrality of an IoT application is a key characteristic which determines whether or not an IoT application is a good candidate for edge deployment. We discuss the different measures of network centrality that can be used to characterize applications, and the relative performance of edge deployment compared to centralized deployment for various IoT applications.**

Keywords—Edge Computing, IoT, Fog Computing, Performance Analysis, Network Centricity

I. INTRODUCTION

An Internet of Things (IoT) application can be generically characterized as a set of sensors and actuators that are connected together into a network—typically to a cloud based service—running on the Internet; the cloud based service can equivalently be on the private Intranet for an enterprise. Due to the proliferation of sensors and actuators, and the advantages of cloud computing, the applicability of IoT devices have dramatically increased in the recent past for many different industries; thus, several platforms have emerged to support IoT applications [1,2]. Almost all the major technology companies offer an IoT platforms, and equivalent platforms for specific industries are offered by companies that make a variety of equipment, sensors and actuators—e.g., IBM Watson IoT, Microsoft Azure IoT Hub, Amazon Web Services IoT Platform.

Despite all the benefits offered by a cloud based IoT platform, there is an equally strong set of arguments for implementing IoT solutions in a distributed manner. For example, a cloud based approach may not be the best one for many applications because of network bandwidth, latency, data rates, performance, cost or regulatory reasons. Moreover, several IoT applications can be better implemented by following the paradigm of edge computing or fog computing—e.g., a smart lighting system based on localized movements.

Whether an application is better implemented as a cloud-based solution or an edge-based solution depends on the characteristics of the application, and the characteristics of the network in which it is deployed. An approach which would allow the determination of the better approach would be very useful to technical community. It is the goal of this paper to present a general framework for assessing the performance of edge applications in both types of implementation, and to provide an intuitive metrics that can be used to select the right model for implementation. Our analysis shows that the key determinant driving the choice between cloud and edge is the importance that network communication has on the overall performance of the application. In this paper, we present a metrics that can be used to measure this determinant—which we refer to as the network-centricity of the application.

The rest of the paper is structured as follows: In the next section of this paper, we present an overview of IoT applications and how they would look like in a cloud based and an edge-based implementation. We propose an abstract problem formulation that addresses these IoT applications, and consider two broad classes of IoT applications in Section III. In Section IV and Section V we present approaches to compare the performance of cloud- and edge- based implementation of each class of the application. Finally, we summarize our results and provide the conclusion of the analysis in Section VI.

II. IOT APPLICATIONS

IoT applications cover a wide verity of application domains for many industries, and the exact instantiation of an IoT application in any industry reflects different idiosyncrasies of the environment assumed for that industry. In this section, we (1) discuss a few—but typical—IoT applications in known industry domains; (2) explore what their implementation would

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

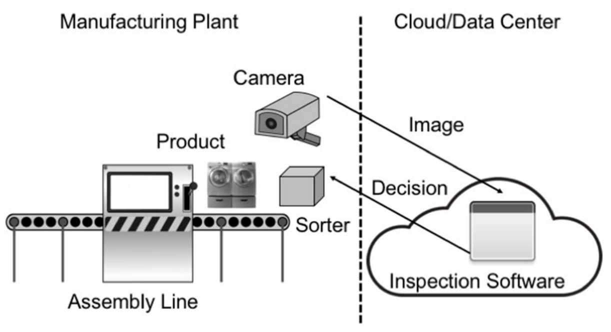

Fig. 1. Centralized Model for Visual Inspection

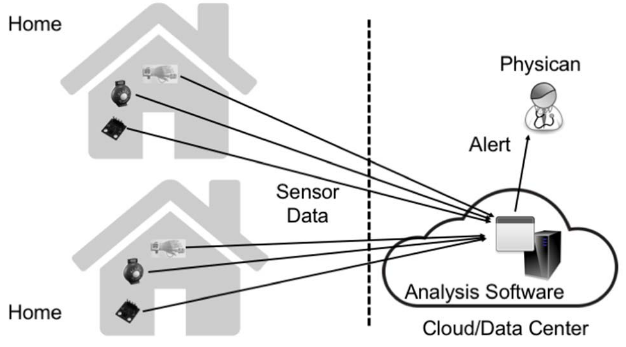

Fig. 3. Cloud Model for Home HealthCare

look like in a cloud computing environment; (3) discuss the reasons why the industry may want to migrate towards an edge/fog computing model and; (4) show the implementation in an edge/fog model. The set of applications in this section is not intended to present a comprehensive view of all different types of IoT applications for an industry—which is out of the scope for this paper—but to provide some illustrative examples of the applications and the motivations for edge computing model.

A. *Manufacturing*

During any manufacturing process, the products rolling off the assembly line need to be checked for quality assurance post-production. For many products, the quality assurance process requires a visual inspection of the product to see if it contains any defects; products that contain visible defects can be separated from the products that do not contain any visible defects. Using an IoT based approach, the process for visual inspection can be automated and autonomously pre-filter defected products—e.g., a set of cameras can take pictures of the finished product from multiple angles, and these pictures can be analyzed to determine whether or not the product is defective.

In order to determine whether or not an image shows a defect, a machine learning approach can be used. A set of training data, which contains the images of several products which a human being determines to have visual defect or not, can be collected from the assembly line. This training data can then be used to train a machine learning model to distinguish good products from defective products. Because of the complexity and processing power required for the machine learning process, such training is best done in a centralized platform, either a cloud based platform or a central location at a data center of the manufacturer. Once the training process is completed, images can be sent over a network to the central site, which can use the machine learning model it has learnt to decide whether or not a product is defective. The setup is shown in Figure 1.

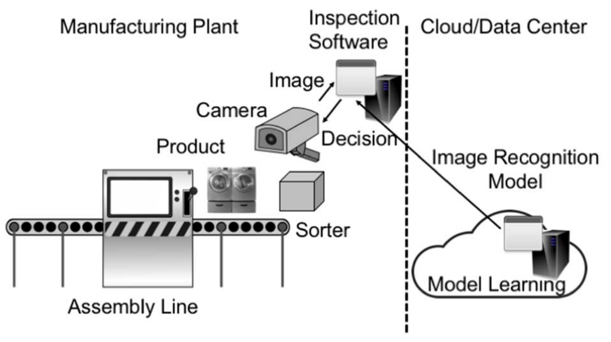

Fig 2. Edge Model for Visual Inspection

While the centralized model is a workable implementation, it has a few challenges. If the centralized system is operated in the cloud, then the manufacturer may not want to send the images of its products outside its organization boundary—e.g., information such as the percentage of defective products in the factory are sensitive information that a manufacturer may want to guard closely. Even if the centralized model is in the data center of the manufacturer, and there are no sensitivity concerns, the centralized model still has some limitations. With typical present-day network connectivity, the latency to go to a central location would be about 100 ms. For larger images requiring multiple round-trips, the response time for an image may be as high as 500 ms. This means that the production rate may be limited to a handful of products per second, which may be less than the optimum desired rate of production.

The edge computing model, as shown in Figure 2, addresses both of these concerns. The inspection software can be run at a server within the plant itself. The latency between the camera and the server would be less than 10 ms if it is within the same premises. Apart from keeping sensitive information on the premises, it also increases the maximum possible production rate that the quality assurance process enforces.

In any real-life manufacturing company, there will be several plants, and the cloud/data center system needs to support all of them. While Figure 1 and 2 are showing only one manufacturing plant, there will be several such plants talking to the software in the cloud/data center.

B. *Home HealthCare*

With the increase in the aging population, providing health-care at home and taking care of senior citizens living independently is becoming popular option. A typical home living arrangement in such conditions may include homes that are instrumented with sensors on cabinets, doors, bathrooms, toilets, and so forth, along with wearable devices that manage the vital signs of senior citizens. The sensors that are included within the household—be it on an individual or on various

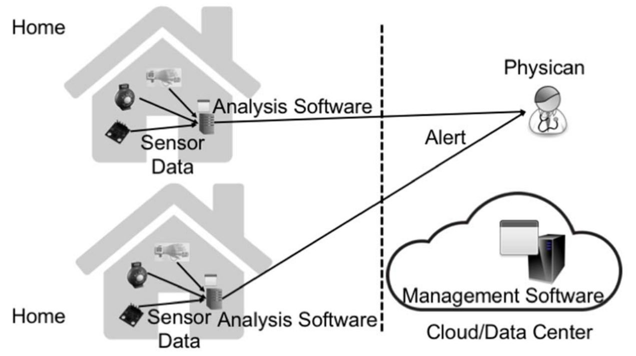

Fig. 4. Edge Model for Home HealthCare

devices— can be analyzed to determine if a person is living a normal life-style, requires some medicine for a chronic illness, or requires immediate medical attention for some acute problem.

One approach to perform this analysis is to stream all the sensor information from the home and the wearables to a service in the cloud. The service in the cloud would analyze the readings from various sensors to determine the current condition of the patient. When the analysis software detects a situation that needs medical attention (e.g., readings divert from the normal or perceived acceptable ranges), it sends an alert to a physician. This alert may be a text message, an automated phone call, or another notification means. This deployment model is as shown in Figure 3. However, due to privacy issues, some people may find it objectionable to have such sensor information sent to the cloud; additionally, a cloud based solution may violate privacy regulations in some jurisdictions.

An alternative approach is to have a home gateway which collects the private information and performs the analysis in the house of every senior citizen. The home gateway would raise an alert to a physician or emergency care personnel depending on the result of the analysis. However, the details of the vital signs are not sent over to the cloud, and remain completely within the control of the senior citizen living in the home. This configuration is shown in Figure 4.

In addition to maintaining privacy of the senior citizen, the edge approach also has some cost benefits. It reduces the amount of data that needs to be stored at the cloud, or transmitted to the cloud. If the network connectivity between the house and the cloud is via a cellular network, this can save a significant amount of money in networking expenses. Furthermore, the cost of the cloud infrastructure can be reduced since less traffic is sent to the cloud, and lower capacity machines can be used there to support the equivalent workload.

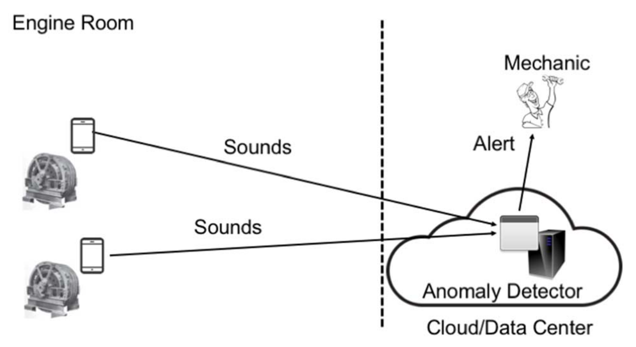

Fig. 5. Centralized Model for Equipment Monitoring

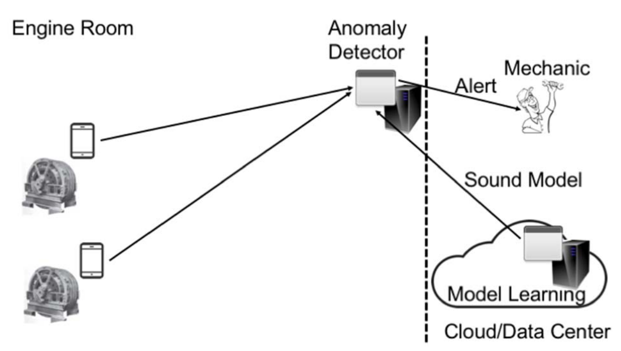

Fig. 6. Edge Model for Building Equipment Monitor

C. Building Equipment Monitoring

Modern commercial (and residential) buildings are complex structures, and include many different engines, motors, compressors, heating, ventilation, air-conditioning and an assortment of other equipment. Like any equipment, these require constant monitoring and regular maintenance to function smoothly. Building equipment also have a long life-span, and can last for several decades. While new equipment comes equipped with several sensors, old equipment often lacks such capabilities.

One way to keep tabs on the health of aging equipment in buildings is to monitor them using a microphone and images [3]. Inexpensive cameras can record images of the equipment as well as sounds. However, sounds are often a better indicator of equipment failure since they can record noises from moving parts that are typically covered and hidden from view. Those sounds can be analyzed using machine learning techniques to determine if any piece of equipment is behaving abnormally. The normal baseline for the equipment can be determined by monitoring the sounds for a few days, after which the system can be trained to look for anomalies.

Such monitoring can be sent over to a centralized or cloud service for analysis, in a setup shown in Figure 5. For the task of training a neural network (or any other AI model) over a large number of collected sound samples, the required processing capacity may only be available in the cloud. However, for the detection of the abnormal sounds, the transmission of sounds can impose a significant load on the central server.

An alternative way to monitor the sounds is to only use the cloud for learning of the normal patterns of the sounds. Once the normal patterns have been learnt, the detection of abnormality can be done by a server that is located on the premises of the building. This eliminate the need for network communication over the cloud during the normal maintenance of the equipment. For buildings that are in remote areas, such an arrangement can set significantly on the costs.

The edge setup where sounds are sent to a local server for tracking anomalies and altering a mechanic is shown in Figure 6. If the sensing device is programmable, e.g. a phone or

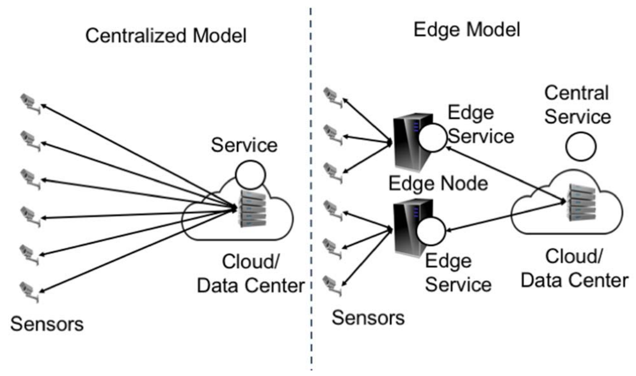

Fig. 7. Abstracted Representation of Implementations

Raspberry Pi is used as the information collector, the anomaly detection can be done within the sensing device as well.

The aforementioned examples elaborated above arise in many other industries and scenarios related to IoT, including automotive applications, telecommunications industry, forestry, mining, finance, healthcare, construction, retail and so on. The edge computing model confers some advantages, but could possibly require more complexity in management and operation. Therefore, it is important to understand the circumstances where edge computing provides a technical advantage, and when it does not. In the next section, we abstract our problem so that the performances of the central and edge approaches could be analyzed formally.

III. ABSTRACT PROBLEM DEFINITION

In order to analyze and compare the performance of a centralized model versus the edge model, we postulate an abstract model for representing the central and the edge model for implementing any IoT applications. Applications discussed in Section II, along with several other applications, can be represented in an abstract model as shown in Figure 7. The Figure shows the centralized model on the left and the edge model for deploying IoT applications on the right. Sensor information is generated by the different sensors, and they are sent for processing to a service, which is running on a server in the data center in the centralized model. The server may be located on a cloud service, instead of a data center as well. In the edge model, the sensors send their information to an edge server running an edge service. The edge service may need occasionally to contact the service at the cloud/data center, but would be able to process most of the requests on their own.

In general, there will be many edge nodes which would be responsible for handling a smaller number of sensors. The assignment of sensors to edge nodes is usually based on the geographic proximity of the nodes and the sensors. In most of the application cases, organizational boundaries or network connectivity characteristics determine the location of the edge node, and the set of sensors that an edge node will handle.

Parameter	Meaning
L_0	Latency between sensor and Central Site
L_1	Latency between sensor and Edge Site
L_2	Latency between Edge and Central Sites
N	Number of Edge Sites

TABLE I. PARAMETERS TO CHARACTERIZE THE ENVIRONMENT

	Visual Inspection
L_0	Latency between camera and data center running inspection service
L_1	Latency between camera and local computer system with inspection service
L_2	Latency between local computer system and the data center
N	Number of manufacturing sites
	Home Health Center
L_0	Latency between sensor and cloud site
L_1	Latency between sensors and gateway in the same home
L_2	Latency between gateway and cloud site
N	Number of homes
	Building Equipment Monitor
L_0	Latency between sensor and cloud site
L_1	Latency between sensors and computer in the same building
L_2	Latency between computer in building and cloud site
N	Number of buildings in the system

TABLE II. PARAMETERS MEANING FOR SPECIFIC APPLICATIONS

To compare the performance between the centralized model and the edge model, we introduce some parameters to characterize the environment. Table 1 lists the parameters that will be used to characterize the different aspects of the environment along with their definitions. The latency mentioned in the table refers to round trip latencies between the different locations.

From the perspective of the different examples of IoT applications discussed in Section II, the meaning of the different parameters would be as shown in Table 2. For other applications, the parameters can similarly be mapped to specific characteristics of the environment in the application.

IoT applications can be characterized as belonging to one of two types, which we define as service oriented applications and message oriented applications. Service oriented applications used a protocol such as REST to exchange information with the services on either the centralized or edge site. These applications expect a response on each such call. REST is designed around the concept of a request-response pair, and each request from the sensor to the edge service/cloud service results in a response [4].

Message oriented applications would use a protocol such as MQTT or COAP to transfer the information from the sensor to the cloud service or application. In a message oriented paradigm, the service does not necessarily send a response on every message that is received. However, when something interesting is observed in the message, the service may take an action.

Any application can be implemented in either a message oriented paradigm or a service oriented approach. From a capability perspective, both approaches are equivalent [5]. However, from a performance analysis perspective, the relative advantages of edge computing compared to cloud computing depends on the implementation paradigm that is selected.

The next two sections provide an assessment of the relative performance of edge applications implemented according to each of the two paradigms.

IV. SERVICE ORIENTED IOT APPLICATIONS

For a service oriented application, interactions between any two components in the system (sensors, edge service, and cloud service) can be characterized as a request response exchange. Such a system can be characterized by several Key Performance Indicators (KPIs) and below we identify 3 critical KPIs for our evaluations.

1. *Responsiveness of the system:* The responsiveness of the system can be measured by the total time it takes for the exchange to be completed.
2. *Scalability of the system:* scalability measures the maximum number of sensors that the system could handle within a required threshold of responsiveness.
3. *Costs associated with the system:* The relative cost of the edge solution versus centralized solution is another KPI, as well as the amount of network bandwidth that is used by the system.

In order to compare the performance of the edge model and the centralized model, we make a couple of assumptions in the modeling process. The two key assumptions are as follows:

- *Efficient Implementation Assumption*: We assume that the system is implemented in a manner such that the edge site can quickly make the determination as to whether the request can be handled locally or requires transferring to the cloud/data center. In practice, this means that the following condition holds true in the system.

$$L_1 + L_2 = L_0$$

- *Equivalent Processing Time Assumption*: This assumes that the real-time required for processing an application does not depend on whether it is processed. The real-time used for any request processing depends on many factors such as the design of the application, resource contention on the system etc. We assume that these values do not depend on the location of the processing, and the clock time added to handling of the request is same. In practice, it means that we can use a single parameter P for processing time instead of considering different edge and cloud service values for the same.

The equivalent processing time assumption may appear to be unrealistic since edge devices are usually less powerful than centralized devices. A typical edge device in the home would have the processing power equivalent to a laptop, while the centralized system could be a massive server. However, we should keep in mind that the edge device is required to process a much smaller number of requests. In any of the scenarios described above, the edge device is only processing a fraction of the requests that the centralized device needs to process. Similarly, for many applications, the locality of the data access patterns helps the edge device. While the edge devices are less powerful in general, they also need to do less work. As a result, the equal processing time assumption is a reasonable approximation to the reality on the ground.

In the edge computing model, some of the exchanges can be handled by the edge site, while some of the exchanges may require reaching out to the cloud site. As an example, an exchange that may require that data be uploaded to the learning system would typically be sent to the cloud, whereas an anomaly detection can be handled largely at the edge site, but on rare occasion may need to be sent to the cloud site. The fraction that can be handled at the edge site will have a significant impact on the responsiveness of the system.

Parameter	Meaning
f	Fraction of requests that edge can handle
M	Amount of data exchanged in an exchange
P	Processing time at edge or cloud site
K	Factor to map latency to data transfer time
Q_e	Threshold for concurrent sensor feeds at Edge
Q_c	Threshold for concurrent sensor feeds at Central site
C_e	Cost of processing cycles at edge site
C_c	Cost of processing cycles at central site
C_n	Cost of network bandwidth

TABLE III. PARAMETERS FOR SERVICE ORIENTED IOT APPLICATIONS

Each system has a capacity for handling a given number of requests while maintaining an acceptable processing time. The chart of performance against load in any context looks like a hockey-stick, and there is a threshold beyond which performance degrades heavily. This threshold will be different for an edge site and the centralized site. In general, we would expect the threshold to be much larger for the centralized location which needs to handle all the sensors, instead of the edge site which needs to handle a smaller number of sensors.

In order to compare the cost of the solution, we consider three cost measures, the cost of a unit of processing on the cloud/data center site, the cost of a unit of processing on the edge site, and the cost of a unit of data transferred on the network between the edge and the cloud/data center. In most of the IoT application contexts, the network between the sensor and the edge can be assumed to be free, and not adding additional costs.

The latency of any network link has an impact on the throughput rate sustainable by applications on that link. The relationship between throughput and network latency is complex, depending on many factors such as loss rates, configuration parameters of the protocol and so forth. However, using the commonly used simple model for characterizing TCP performance [6], we can assume that the time needed for network data transfer is directly proportional to the latency of the link—i.e. communication on a link of latency L would use up time KL, where K is a constant dependent on the network parameters.

The list of all the parameters that are needed to model the service oriented applications are summarized in Table III. With the parameters from Table I and Table III, we can compute the values of the KPIs for both the edge and centralized model of IoT applications.

A. *Responsiveness Gain*

For responsiveness, the response time of a transaction R_c for the centralized model can be computed as:

$$R_c = KL_0 + P$$

The response time of the same transaction at the edge R_e can be computed to be:

$$R_e = f(KL_1 + P) + (1-f)\ (KL_0 + P)$$

The relative gain in responsiveness by using the edge model is the ratio $(R_c-R_e)/R_c$. Simplifying the previous equations, we get that the gain in responsiveness (G_r) from the edge model is:

$$G_r = f \frac{L_2/L_0}{1 + P/KL_0}$$

The fact that the responsiveness depends on the probability of handling request and the edge, and the relative reduction in latency from processing at the edge is not surprising. The parameter f is a characteristic of the application, and the ration of L_2/L_0 is a characteristic of the network. However, the denominator consists of a term which combines the attributes of the application and the network, and which has an important impact on gain in responsiveness. This leads to our first measure of the interaction between an application and the network characteristics, which can be defined as the basic network centricity of an application.

The basic network centricity (NC_b) of an application is the ratio of the time spent by an application communicating over the network to the processing time spent in the service, i.e. the metric (KL_0/P). NC_b can range from 0 to ∞, with 0 indicating applications running in an environment where the processing time required by the service is significantly larger than the time spent by the application exchanging information over the network. This will be hold for the model learning phase of any AI application which is significantly compute-intensive. The network centricity of ∞ would characterize the applications which spend the bulk of their time exchanging data over the network, with relatively little processing. Applications which perform relatively simple operations over large volumes of data, e.g. checking for a sensor crossing a threshold (e.g. whether a light sensor has indicated lack of illumination, or an image has changed) will fall in the latter category. Such applications are good candidates for implementing in the edge computing model.

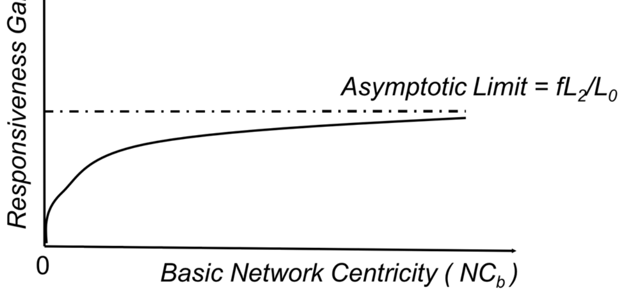

Fig. 8. Responsiveness Gain from Basic Network Centricity

Figure 8 shows a plot of the relative gain of the edge-centric model as a function of the basic network centricity. As one can see from the plot, the gains increase asymptotically initially but flatten out eventually. The maximum gain that an application can get from an edge deployment model would be $f(L_2/L_0)$, and is shown by the asymptotic line in Figure 8.

We can also adapt the definition of network centricity so that it can be simplified and be measured in a manner similar to the other terms in the equation characterizing G_r. The other two factors that impact G_r are f and the ratio L_2/L_0, both of which are values that range from 0 to 1. A better measure of network centricity can be obtained by converting it into a measure which also ranges between 0 and 1.

This enhanced measure of Network Centricity (*NCe*) can be defined to the ratio $KL_0/(KL_0+P)$. Instead of just measuring the relative contribution of the network and the service to the response time of the application, this measure determines the relative contribution of the network to the overall response time of the application. *NCe* is a measure which ranges from 0 to 1. With this measure, we can simplify the expression of the responsiveness gain as

$$G_r = f.(L_2/L_0).\ NCe.$$

This expression shows that the three measures that impact the improvement in the responsiveness of IoT edge applications are the probability that the edge can handle the request, the reduction in network latency introduced by the edge, and the enhanced network centricity.

B. *Scalability Gain*

We have defined the scalability of the system as a measure of the number of sensors the solution can support with acceptable performance. In the centralized model, the system can support S_c sensor feeds which will be given by:

$$S_c = Q_c/(KL_0 + P)$$

Similarly, the number of concurrent sensor feeds at any edge site can be computed. However, since there are N such edge sites, the number of sensor feeds in the system that can be supported using the edge computing model will be:

$$S_e = NQ_e/\ [f(KL_1 + P) + (1-f)\ (KL_0 + P)]$$

This expression can be simplified with a bit of mathematical manipulation, and their ratio calculated. This gain in scalability (G_s) can be seen to be:

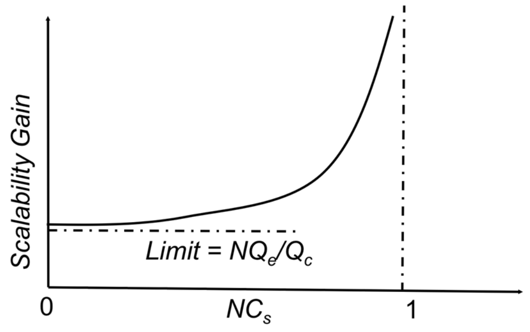

Fig. 9. Impact on Network Centricity for Scalability

$$G_s = \frac{N\,{}^{Q_e}/_{Q_c}}{1 - \frac{kfL_2}{kfL_0 + P}}$$

The numerator can be viewed as the additional processing power that is introduced in the network due to the large number of edge processing nodes. It depends on the number of edge processing nodes as well as the relative capacity of the edge and the core nodes. In some cases, if only a few edge nodes are deployed, and the edge nodes have significantly lower threshold for concurrent sessions than the centralized nodes, edge computing may result in reduced scalability.

The denominator is another metric that combines attributes from both the network and the application characteristics. This can be viewed as another measure of network centricity of an application, which we refer to as the Network Centricity for Scalability (*NC_s*). Because the numerator for this measure is *KL_2* instead of *KL_0*, *NC_s* is different than *NC_e*. We define *NC_s* to be $\frac{KL_2}{KL_0+P}$, and this measure has the desired property that it will be between 0 and 1.

One can also notice the that

$$NC_s = (L_2/L_0)\ NC_e.$$

Thus, there is a relationship between responsiveness gain and the measure of network centricity for scalability

$$G_r = f\,NCs.$$

Therefore, the equation for scalability gain can also be simplified as

$$G_s = \frac{\mathrm{N}\,{}^{Q_e}/_{Q_c}}{1 - fNC_s}.$$

As *NC_s* approaches 0, the edge model scalability decreases, asymptotically reaching the limit defined by the numerator which effectively measures the additional processing power introduced by the edge nodes. As it approaches 1, the scalability for the system increases, and the edge model can support a much larger number of sensors than the centralized model can.

C. Cost Analysis

Because the edge devices run on smaller machines than the centralized core system, computing cost per instruction at the edge can be higher than the computing cost per instruction at the core. The computing cost includes the sum total of amortized fixed costs and the operational costs of the systems at the edge and the cloud. Because of the additional complexity of managing a large number of edge devices distributed across several locations, the management costs of edge implementation can be high, resulting in a higher per instruction cost for the edge deployment model.

It therefore becomes interesting to explore whether the edge model would be rendered infeasible because of cost reasons, even if it has a better scalability and provides a better response-time. The cost of the centralized solution for every request response exchange is going to be:

$$M.C_n + P.C_c.$$

The equivalent cost of the edge solution will be

$$f.\mathrm{P}.C_e + (1\text{-}f).M.C_n + (1\text{-}f).P.C_c$$

We can now compute the conditions under which the edge computing approach will have a lower cost than the centralized approach. By comparing the two costs defined above, some mathematical manipulation leads to the result that the edge is better than the centralized solution when

$$C_e < C_c + C_n M/P$$

In other words, the edge solution can be more cost effective than the centralized solution as long as the increased cost of edge cycles are offset by the savings in network bandwidth. The savings in costs of the network bandwidth are determined by the ratio M/P, which is the ratio of network bytes exchanged to the processing time required at the service. *M/P* is another measure of network centricity for service oriented IoT applications.

It follows that the best cost justification for edge computing will be in environments where the network costs are non-trivial (e.g. IoT solutions relying on cellular or satellite connectivity), and for those applications which have a relatively higher volume of network data exchange compared to processing requirements at the server.

V. Message oriented IoT Applications

For the message oriented IoT applications, there is no well-defined request-response exchange. Instead, the sensors produce a stream of messages that traverse the network, being processed either at the centralized service or at the edge service instance. If an interesting pattern is found in this stream of messages, the service may generate additional activities, e.g. raising an alert or sending a new message on the network.

In order to analyze the effectiveness of edge solutions for message oriented IoT applications, we need to characterize them in a different manner. While the KPIs of scalability and solution cost for message oriented IoT applications would be the same as that for service oriented IoT applications, KPI measures such as responsiveness would need to be defined differently. For message oriented applications, we define responsiveness as the time lag between an interesting event being captured by the sensors, and the ability of the application to take an action (e.g. send an alert) based on the processing of the interesting event.

To analyze the message oriented applications, we need to introduce a new set of parameters that characterize these applications. Those parameters are shown in Table IV. Furthermore, we will reuse the parameters of *K, Q_e, Q_c, C_n, C_c* and C_e used for analysis of service oriented applications. As in the previous section, we would compare the responsiveness

Parameter	Meaning
q	Probability that a message has interesting event
M	Size of a message
P	Processing time at edge or cloud site for a message

TABLE IV. PARAMETERS FOR MESSAGE ORIENTED APPLICATIONS

gains, the scalability gains and the relative costs of the centralized and edge models.

A. Responsiveness Gain

In the centralized approach, once the interesting pattern is generated by a sensor, it will be acted upon the system after the message is processed at the central service. This lag is going to be $P+KL_0/2$. The division by 2 reflects the fact that we had defined the original measure as round-trip latency, and only one half of this time will be needed from one way of the transmission. Equivalently, the lag for edge model would be $P+KL_1/2$.

Comparing the increase in responsiveness and measuring it relative to the original lag shows that the increase in responsiveness of the edge solution can be characterized as:

$$NC_r = \frac{KL_2}{2P + KL_0}$$

This measure, which includes a combination of attributes of both application and the network, is a new measure of network centricity for message oriented IoT applications.

B. Scalability Gain

The analysis for scalability gain can be done in a manner analogous to that for stream oriented applications, with the differences introduced due to the difference in the equation for responsiveness. By comparing the scalability gain due to edge computing, we see that the scalability gain for this class of applications is given by:

$$G_S = \frac{\mathrm{N}\, {Q_e}/{Q_c}}{\frac{KL_1 + 2P}{KL_0 + 2P}}$$

The numerator of the gain is the same as that for service oriented applications, and subject to the same discussion as was made in the previous section. The denominator is different, reflecting the fact that the characteristic equations for response time is different.

As in the case for service oriented applications, the denominator can be treated as a measure of network centricity for message oriented applications. We will refer to this measure as NC_q.

C. Cost Analysis

For the analysis of costs, we make the assumption that when the edge sees an interesting event, it flows an extra message of size E to the central site. This extra message will be processed by the central site. As an example, an edge site may be used to suppress a video feed which is not of interest, and send the video back to the central site only when something interesting is seen. In this case, the cloud cycles are invoked only when something interesting happens and the extra message is sent to the cloud.

When we compute the per message cost, the cost of the central solution will be:

$$M.C_n + P.C_c.$$

The equivalent cost of the edge solution will be

$$P.C_e + q(P.C_c + E.C_n)$$

Comparing the two costs, we come to the result that the edge can save costs over the central solution under the following condition:

Environment	L_2/L_0	Justification
Visual Inspection	Close to 1	Latency within machines on plant is negligible compared to latency to cloud/data center
Home HealthCare	Close to 1	Latency within machines in same home is very low
Equipment Monitoring	Close to 1	Latency between two machines on same building will be low.

TABLE V. NETWORK PARAMETERS IN DIFFERENT ENVIRONMENTS

$$C_e < (1-q)\, C_c + C_n\, (M+qE)/P$$

For most applications, q will be fairly small, and a more expensive cycle for edge computing can be supported because they will save a significant amount of network bandwidth. For the limiting case of $q \rightarrow 0$, the condition becomes the same as that for service oriented applications.

The performance metrics for message oriented applications are similar to service oriented applications, albeit with some subtle differences. The network latency contributes only half as much in the equations as it does for service oriented applications, which need to go through a round trip exchange. Also, the metrics for network scalability have extra processing term, which is not exactly equivalent to the corresponding metric for service oriented applications.

VI. DISCUSSION

Having examined different measures of network centricity, it is useful to revisit the original set of illustrative IoT applications and examine them to see if they are suitable candidates for edge model in different types of networks. All three of those applications can be implemented in either a service oriented model or a message oriented model. However, it will be more natural to implement the visual inspection application as a service oriented approach, and the other two, which looks for patterns and generate alerts, in a message oriented paradigm. In each of these applications, there will be a training component in which data from sensors is used to build a model for use subsequently, and there would be a inference components in which a sensor reading from an IoT device is used to decide whether an alert needs to be raised, or what response to provide to the client reporting a specific sensor input.

Let us first consider the parameters that characterize either the network or the application alone, without taking their interplay into account. For each environment, the network is primarily characterized by the ratio L_2/L_0. The qualitative assessment of the values for them will be as shown in Table V.

The visual inspection, being designed as a service oriented application will be characterized by the parameter f, while the other two will be characterized by the parameter q. If the images will have a high degree of locality, f would be fairly high. After all, there are only a few types of common problems in the production process. Similarly, the parameter q will be close to 0 for the home health-care and equipment monitoring since the patterns/abnormal situations are not likely to happen frequently.

Looking at the value of network centricity, we can see that the measures of network centricity are strong for each of these applications. Measures such as NC_b or NC_s will typically tend to be close to 1 for these applications, indicating that they are good candidates for edge computing model.

VII. Conclusions

In this paper, we have analyzed the importance of systematically deciding the strategy for deploying an IoT application—be it centrally or edge-based. We have motivated the work by means of multiple domain scenarios from manufacturing to healthcare. We then abstracted the problem of deciding the central versus edge deployment so that best and worst-case performance analysis could be done formally. In order to compute the performance details, we have considered three key performance indicators and proposed models to compute them in both central and edge deployments. The network centrality of an IoT application is a key characteristic which determines whether or not an IoT application is a good candidate for edge deployment. We have identified three definitions of network centricity which enables the comparison of the performance of cloud versus edge deployments in a quantitative, as opposed to a qualitative manner.

References

[1] L. Xu, W. He and S. Li, "Internet of things in industries: A survey," IEEE Transactions on industrial informatics, vol. 10, no. 4, pp.2233-2243, November 2014.

[2] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises", Business Horizons, vol 58, no. 4, pp. 431-440, August 2015.

[3] B. Ko et. al., "Acoustic Signal Processing for Anomaly Detection in Machine Room Environments", Prof. 3rd ACM International Conference on Systems for Energy-Efficient Built Environments, pp. 213-214, November 2016.

[4] T. Erl, B. Carlyle, C. Pautasso and R. Balasubramanian, "SOA with REST:Principles, Patterns &Constraints for Building Enterprise Solutions with REST", Prentice Hall Press, 2012.

[5] M. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, "Service-oriented computing: State of the art and research challenges", vol 40 no 11, November 2007.

[6] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation", Proc. ACM SIGCOMM, vol. 28, no 4., pp. 303-314, October 1998.