Although the challenges of connecting Internet of Things (IoT) devices are being addressed, the next wave of IoT innovation will be driven by data analytics. Correspondingly, there has been a proliferation of cloud-based, centralized data analytics platforms, such as Amazon Web Services (AWS) IoT, IBM Bluemix, and Microsoft Azure IoT Suite. Sensors send data over the network to these cloud-based platforms, where all the processing takes place and appropriate decisions are made. This approach reduces application development efforts and maintenance costs by keeping the application logic in one central location and offering preprogrammed services (for example, data visualization and device management) to developers. However, cloud-based solutions have limitations, such as high bandwidth cost and/or high latency. They also assume that sufficient connectivity exists between IoT devices and a cloud service, which does not always hold in reality. Even if we assume that we could address the bandwidth, latency, and connectivity issues by employing a sophisticated infrastructure, a large class of IoT applications might not be suitable because of regulations and the security concerns of sharing data.

Advances in IoT technologies have ensured the design of powerful sensing/actuating devices with increased computing power and storage capacities that enhance their ability to perform a few computation tasks at the device or gateway level rather than through a centralized data analytics approach. Recently, the fog computing concept has gained momentum. The term refers to pushing the data processing tasks to the edge of the network rather than using a centralized cloud-based approach.

In this article, we outline an intelligent approach for IoT analytics that facilitates the automated transitions between edge and cloud depending on the dynamic conditions of the IoT infrastructure and application requirements.

**Intelligence at the Edge: Fog Computing**

Fog computing (a term coined by Cisco) aims to address the limitations of cloud computing. Cisco’s IOx operating system is one of the prominent solutions that provides storage and computation in virtual machines (VMs) on infrastructure nodes. Computational, networking, and storage can be called fog nodes. Fog nodes can be devices such as Raspberry PI, devices with ContikiOS/TinyOS, mobile devices with Android OS/iOS, or routers with Cisco’s IOx operating system. However, a fog node cannot handle a large analytic task or multiple IoT applications competing for resources, which could result in increased processing latency. To overcome such limitations, a fog node leverages on-demand cloud resources and coordinates with other geographically distributed fog nodes, as Figure 1 shows.

In future fog deployments, many fog nodes will have resource constraints and rapidly changing contexts, such as battery level, availability, and mobility. Moreover, there will be a demand to provide flexible environments where fog nodes can dynamically adapt according to context and application requirements rather than just providing simple static services. Therefore, we will need...
an intelligence at the edge to address these requirements.

**Related Concepts**
The concept of intelligence at the edge is not new. Similar concepts include mobile cloud computing (MCC), mobile edge computing (MEC), mist computing, and cloudlets. MCC integrates a centralized cloud service (such as AWS) into the mobile environment to overcome the limitations of mobile devices such as performance (storage, bandwidth, battery life, and so on) and environment (such as heterogeneity, availability, and scalability).\(^2,^3\) MEC advances further and tries to move a large portion of tasks from centralized resources directly to nearby devices and infrastructure.\(^4\) Cloudlets are trusted, resource-rich, and well-connected computers that are available for use by nearby mobile devices.\(^5\)

Mist computing, proposed by Cisco, pushes the computation to the extreme edge of the IoT environment, where sensing and actuating devices are involved. Mist nodes are different from traditional sensor nodes. For instance, a sensor node largely provides preconfigured services such as reporting temperature and humidity values, whereas a mist node provides a more flexible environment in which the mist node can execute an application-specific customized computation. The computation could be a simple application-specific task such as data aggregation, data filtering and preconditioning, or data fusion. Other researchers propose an intelligence at the edge framework, which enables efficient execution of machine perception on resource-constrained IoT devices (for example, a mobile phone or gateway).\(^6\) This approach uses bit-vector encodings to represent domain knowledge and implement the semantic perception algorithms.

The concept of intelligence at fog nodes comes from several of these existing concepts. However, we distinguish it as a more generalized approach for IoT, where the IoT environment is resource constrained. Our focus is on IoT scenarios (as discussed in the next section) in which computation needs to be performed quickly and locally. Moreover, the application logic is not explicitly woven into an underlying infrastructure.

A broad aim of the system is to speed up an analytic task through parallelization. We achieve this...
parallelization by segmenting the task into smaller tasks and offloading these tasks to fog nodes and cloud services while maintaining the system’s objectives. The objectives could be any combination of saving the collective energy of devices, saving network bandwidth, processing or filtering a large amount of data before transferring it outside the network, performing distributed large-scale analytic tasks, targeting low latency for specific IoT scenarios (for example, real-time video analytics, cloud gaming, and smart factories), or achieving application-specific requirements.

Application-specific analytic tasks can be reconfigured at runtime, rather than deciding the data processing strategy at design time. For instance, a fire detection sensor can be configured to send values every 5 seconds in a normal situation, but, if the temperature value is above a certain threshold, it will send values more frequently, irrespective of constraints.

Fog nodes could aim together to optimize certain objectives as a whole. In their work, Cássio Prazeres and Martin Serrano present an example to describe this behavior: a fog node can serve up to X simultaneous requests (REQ), and other fog nodes notice that requests are made beyond the fog node’s capacity (that is, REQ > X). This increases the latency time, thus affecting an application’s performance. To avoid this, fog nodes could replicate the same functionality and offer it to achieve REQ < X.

Fog nodes can adapt themselves to provide a better quality of experience to users. For instance, fog nodes could perform analytics locally and send the data to the cloud to serve remote users. For local users, the system can offer the same service through localized protocols (such as Wi-Fi and Bluetooth). This can serve the needs of both remote and local users and could provide better quality experience to local users.

**Potential Use Cases**

Data analytics have been used successfully to design innovative smart applications that facilitate large communities, including smart cities, smart enterprises, and smart buildings as well as industry needs such as smart manufacturing, smart farming, and business services.

**Smart City Applications**

Smart city initiatives are driven by the deployment of IoT infrastructure within various cities around the world and by building IoT applications over this infrastructure. Examples include challenges related to traffic congestion, public services, energy management, sustainability, and public safety and security. Despite all the advancements in telecommunication technologies and network coverage, cities still face great difficulty providing support for all smart city applications. The analytics will reduce both network congestion by building an ad hoc local network and the dependency on the central network. Additionally, by having an on-demand tradeoff option between the cloud and the edge, applications can make on-the-fly decisions related to data transmission and computing tasks.

**Security Surveillance**

Live streams from surveillance cameras often result in heavy network traffic. Intelligence at the edge can play a key role in reducing network traffic and saving bandwidth by deploying different analytical algorithms such as anomaly detection from crowd behavior. The algorithms can support devices such as surveillance cameras to locally process video images and only dispatch relevant information based on contextual requirements. For example, analytical algorithms can detect anomalies locally after processing the video streams and can reconfigure the camera settings (video quality, camera angle, operational configurations, and so on) and/or trigger actuations to switch between analytical algorithms deployed at the edge or the cloud.

**Smart Manufacturing**

Industry 4.0 and similar initiatives are embracing the concept of “fog in the factory,” where the decision-making
process moves from the cloud to the edge. Analytical algorithms can help realize such concepts, since cloud-based solutions are not optimal for meeting the high latency demands of manufacturing processes and devices, mainly because of the large volume of data produced at a rapid rate. Data analytics algorithms can be adopted for smart manufacturing and deployed at the edge, fog, or cloud. In the smart manufacturing domain, a key requirement is to gain performance and operational efficiencies by providing a quick decision-making process for automated devices operating on the factory floor.

### Design Considerations for Intelligent System at the Edge

Figure 2 shows a general intelligent system architecture in a fog paradigm. Such a system receives a request from an application, then, based on the system context and user’s objectives, it divides the analytic task into segments, which it can offload to surrogates, expecting that the offloaded task will be executed on surrogates and they will return the results to the user’s application.

#### Roles
Each element has one or more roles, depending on each fog node’s functionality. Like any emerging area, fog roles are not completely new; instead, they are an evolved version of concepts found in the existing literature.\(^7\)\(^8\)\(^9\)

**Surrogate.** A surrogate is a fog node that can be used to offload a portion of a large analytic task. The offloaded task are executed on a surrogate, and the results sent back to the initiating fog node.\(^8\)

**Context manager.** The context manager node periodically monitors the status of surrogates and provides the context information to other components for decision making. An intelligent fog computing system that performs computation offloading needs a certain awareness of a surrogate’s status, such as its availability, battery level, CPU, RAM, bandwidth, and free memory. The system uses this information to decide whether a surrogate is qualified for an offloading task.

A surrogate can be mobile, which changes its availability and connectivity over time. For a mobile surrogate, the context manager could use prediction techniques to forecast the device’s connectivity, location, link quality,
and mobility. The context manager could leverage a machine learning library such as Google’s TensorFlow (www.tensorflow.org/mobile) to implement learning on devices such as a Raspberry PI and Android phone. Sometimes, collecting data from surrogates and learning about their behavior is a computation-intensive task and it might not be possible on fog nodes due to resource constraints; therefore, the context manager could move the data collection and learning tasks to a cloud service, which could build prediction models and return them to the context manager.

**Partitioner.** The partitioner generates a bundle of components that can be used by the offloading process, which we discuss next. Hadoop automates the partition phase, which takes place after the map phase and before the reduce phase (https://hadooptutorial.wikispaces.co/Custom+partitioner). The partition can be performed by developers, too. Developers explicitly build applications such that part is executed locally and another part is executed at a surrogate (for example, some user interface-driven tasks are performed locally while computation-intensive tasks are performed at a powerful surrogate). A recent practice is to adopt a microservices architecture (https://martinfowler.com/articles/microservices.html), a more distributed architecture where some application functions are explicitly divided at design time. Although this approach requires extra development effort, it also optimizes some parameters because it is fine-grained; that is, an application can execute remotely on only those subparts that benefit from remote execution.

**Offloading.** An offloading process is divided into two steps: mapping and migration.

After a partitioner segments an analytic task at design time, the mapping process decides during runtime which part should be offloaded to surrogates and which part should be executed locally. The mapping process receives information about the static and dynamic contexts from the context manager and determines an optimal deployment strategy. The decision is not limited to a single goal; users can have multiple goals that conflict with one another. Based on these parameters, the mapper decides which partitioned segment should be deployed on which surrogates.

---

**Application development for IoT in fog computing is a challenging task because of the associated dynamicity; the goal is to allow an analytic task to dynamically scale based on context in the fog.**

After the partitioner decides to map a segment to a surrogate, this technique migrates the segmented bundles to the surrogates. The bundle’s granularity could be a full VM, in which an individual application can be migrated to a fog node. However, this approach might be too “heavy” for resource-constrained surrogates. To address this problem, a lightweight container, such as a docker (www.docker.com/what-docker) can be applied. Unlike VMs, these containers do not bundle a full operating system; rather, they bundle a docker image, which contains the necessary runtime and libraries required to execute the bundle on a fog node. The docker image is efficient, lightweight, and self-contained, and it guarantees that an application’s code will always run the same, regardless of where it is deployed. A docker image uses a centralized distribution strategy, where a cloud-based central repository (for example, Docker store; https://store.docker.com) in the cloud holds bundles and the fog nodes fetch the bundles necessary for execution.

**Gaps in Current Technology and Requirements**

Some important gaps in current technologies and requirements required to build an intelligent system at the edge remain.

**Programming Models**

Application development for IoT in fog computing is a challenging task because of the associated dynamicity; the goal is to allow an analytic task to dynamically scale based on context in the fog. It is hard for developers to orchestrate such a dynamicity. The orchestration becomes more challenging over numerous heterogeneous fog nodes distributed over a wide area. Existing distributed stream and data-processing programming frameworks (such as Hadoop and Apache storm) might not be suitable for fog computing because their architectures are based on static configurations. Technologies such as Google’s Kubernetes (https://kubernetes.io) automate scaling and the management of containerized applications. However, these technologies have been developed largely for cloud computing scenarios, where resources are typically homogeneous and tasks are resource intensive.
Heterogeneity
A fog infrastructure consists of a broader variety of heterogeneous nodes than a cloud computing ecosystem. A fog infrastructure implements heterogeneous protocols and hardware resources that exhibit different operating conditions. The heterogeneity further raises interoperability issues among nodes. To address these issues, a common technique is to design a common framework/middleware and provide abstractions on top of it. A good example to fill this gap is Eclipse’s ioFog (https://projects.eclipse.org/projects/iot.iofo). Eclipse ioFog is a set of technologies on a fog computing layer that can be installed on any fog node running a Linux operating system. The ioFog universal runtime allows microservices to run on fog nodes.

Deployment
Existing technologies such as containers and VMs provide suitable abstractions for application deployment. However, these technologies have largely been developed for cloud computing scenarios, where resources are generally homogeneous and tasks are resource intensive. However, future fog deployments will contain a wide variety of fog nodes with resource limitations. Therefore, technologies that can accommodate a wide variety of fog nodes are needed. Some early efforts such as microclouds attempt to bring docker containerization techniques to devices like Raspberry Pi. However, more efforts are required to bring these technologies into industrial practices.

Fog computing can definitely provide benefits over cloud. However, it is not a replacement for cloud computing. We believe that some scenarios (more specifically in industrial Internet/Industry 4.0 domains) will use fog along with cloud.

References

Pankesh Patel is a research scientist at ABB Corporate Research, India. His research interests include Internet of Things and software engineering. Patel has a PhD in computer science from the University of Paris VI, France. He is a member of ACM and IEEE. Contact him at dr.pankesh.patel@gmail.com.

Muhammad Intizar Ali is an adjunct lecturer, research fellow, and research unit leader of the Reasoning, Querying, and IoT Data Analytics Unit at the Insight Centre for Data Analytics, National University of Ireland, Galway. His research interests include Semantic Web, Internet of Things, and data analytics. Ali has a PhD in computer science from Vienna University of Technology, Austria. Contact him at ali.intizar@insight-centre.org.

Amit Sheth is the LexisNexis Ohio Eminent Scholar and executive director of Kno.e.sis, the Ohio Center of Excellence in Knowledge-Enabled Computing at Wright State University. His research interests include physical cyber-social computing and Web 3.0. Sheth has a PhD in computer science from Ohio State University. He is an IEEE Fellow. Contact him at amit@knoesis.org.

myCS Read your subscriptions through the myCS publications portal at http://mycs.computer.org