CSSE7014 Distributed Computing
Assignment 2
Semester 1, 2017

Paul Kogel (44644743), Ramdas Ramani (44743767), Andi Nuruljihad (44159069)

May 13, 2017

Contents

1 Introduction

Fog computing is a new, exciting computing paradigm [**?**] that serves as an extension to cloud computing. It allows for tasks traditionally performed at the cloud server to be performed directly by – or at nodes in closer proximity to – edge devices.

Cloud computing refers to the usage of remote servers for the purpose of storing, managing, or processing data. Cloud computing has enabled the development and proliferation of advanced services that were previously impossible, like the recognition of voice commands on a smart phone; the phone merely serves as a collector of raw audio data from the user then transmits that data to the cloud where it is processed and the information relevant to the user's query is collected to be sent back to the user.

As the number of devices requesting data from the cloud rises, and the Internet of Things (IoT), the inter-networking of everyday objects via the Internet, continues to expand, cloud services are facing the ever-growing challenge of scaling to accomodate the rising number of objects transmitting and receiving data. This puts a strain on bandwidth as the millions of devices are in a state of constant connection with to the cloud.

Inherent in cloud computing is the problem of latency; data is stored and processed at remote servers in off-site locations that could be anywhere in the world. In addition, the quality of a cloud-based service is highly dependent on the network conditions of a region or area. When you lose your connection to the Internet, you lose the ability to use these services. These are two of the greatest problems faced by developers of smart services that rely on low-latency or real-time response and highly-reliable connections, such as health and emergency services. A delay of even a half-second could pose a safety risk or affect treatment.

Fog computing solves these problems by offloading much of the workload of the cloud server to devices that are closer to the edge [**?**]. These devices, or "fog nodes", can be any device with storage and processing power. This means more data is stored and processed locally, reducing the volume of network traffic being sent to and from the cloud, and since more of the data analysis and processing is performed at a location much closer to the user, latency, and subsequently response time, is greatly diminished.

This report will go into detail about the architecture and models of fog computing, common issues in the implementation of fog computing, and real-use applications of the paradigm.

2 Architectures and Models

Fog computing is an emerging cloud paradigm that extends the traditional cloud computing paradigm to the edge of the network to decrease latency and network congestion,enabling the creation of refined and better applications or services. Fog is an edge computing and micro data center (MDC) paradigm for IoTs and wireless sensor networks (WSNs).

In this section, we are going to discuss different basic architectures that are commonly used in fog environments. Fog is still a developing field, and very broad, with many different use cases. To give good overview,we will present two specific architectures namely Software-Defined Fog Network architecture and Gateway Based Fog Computing Architecture. We will then look at the efforts of the Open Fog consortium to provide a standardised framework for building Fog Applications.

2.1 Gateway based Fog Computing Architecture

Internet of Things(IoT) is a huge global information system composed of many objects that can be identified, sensed and processed based on standardized and interoperable communication protocols[**?**].In general,most of the IoT services and applications some of which are namely Connected Vehicle, Smart Grid, Smart Cities, and Wireless Sensors and Actuators Networks(WSANs)require mobility support, location awareness and low latency[**?**]. These features are not readily realisable using the traditional cloud computing paradigm and that is where a computing model such as fog computing model that extends a traditional cloud computing model fits in.

One of the key elements of the Internet of Things is Wireless Sensor Networks where multiple heterogeneous wireless communications coexist: such as Wifi,ZigBee,cellular and Bluetooth. Wireless Sensor Networks are a collection of nodes that are used to sense and interact with the environment in which

they are set up. Actuators however add a new dimension to sensor networks by bringing the ability to control the system. Some applications which require physical actions such as open, close, move and focus can use Actuators to achieve them.Furthermore WSANs are still domain specific and usually deployed to support a specific application[?].

Wangbong lee et.al[?] proposed a Gateway based fog computing architecture for WSANs. The fog computing setup proposed consisted of a set of gateways and a microserver[?].The Gateway and Micro server were connected by Ethernet interface and the Interfaces between gateway nodes were a combination of wired and wireless interfaces such as 3G, LTE, and Ethernet[?].

The nodes could be seen as classified into Slave Nodes and Master nodes. Slave nodes took care of functions such as flow management,virtual gateway and resource management whereas the Master Nodes had control over the functionalities of the Slaves[?]. The gateways had their own role where they functioned as a Master/Slave node.

Flow management for this setup was based on software defined network such as OpenFlow, and virtual function management needed a light version of NFV(Near Field View) architecture such as ClickOS[?] which helped in network function virtualization. The architecture provided helped virtualize WSAN and the model was event driven. It provided virtual event from networked objects and shared them to various applications.

The figure 1 shown below[?] looks at the architecture both from a Physical and Logical View.

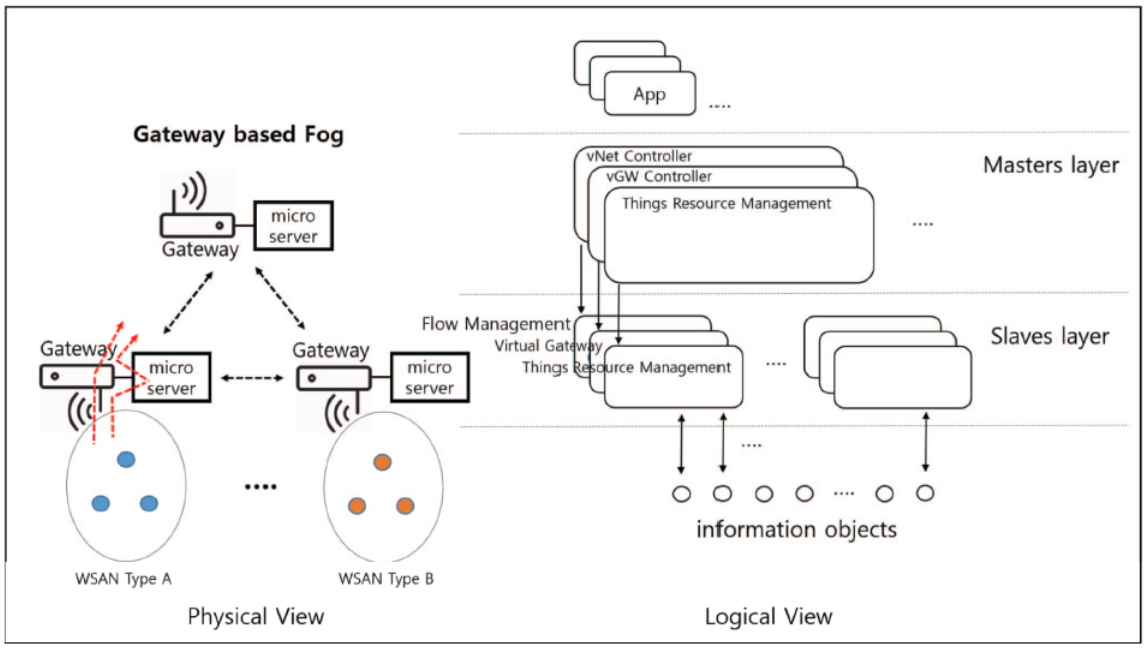


Figure 1. A gateway based fog computing architecture for WSANs

When we look at the amount of data interchange that goes on in WSANs, latency of the application plays a major role and it is desirable to keep it as low as possible. This is not possible to realise in a cloud model where the latency is high. This could be attributed to the fact that the control topology of fog computing is distributed whereas cloud is centralized[?].Furthermore, most important fog applications involve real-time interactions rather than batch processing[?]. Finally, Large-scale sensors networks to monitor the environment is an example of inherently distributed systems, requiring distributed computing and storage resources. Fog computing has very large number of nodes as evidenced in sensor networks in general[?].

2.2 Software-Defined Fog Network Architecture for IoT

Before we discuss the Software Defined Fog Network for IoT, let us look at the traditional IoT architecture and how it is realised. Figure 1 below[?] illustrates the architecture from a high level perspective.

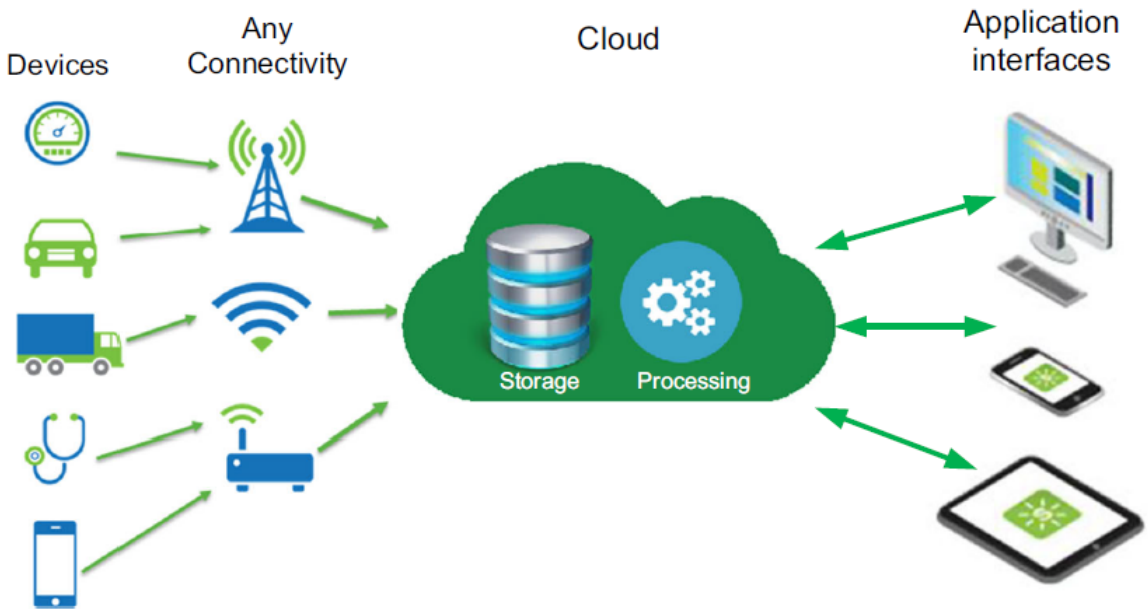


Fig. 1 Traditional IoT architecture

Four main components involved are[?]:

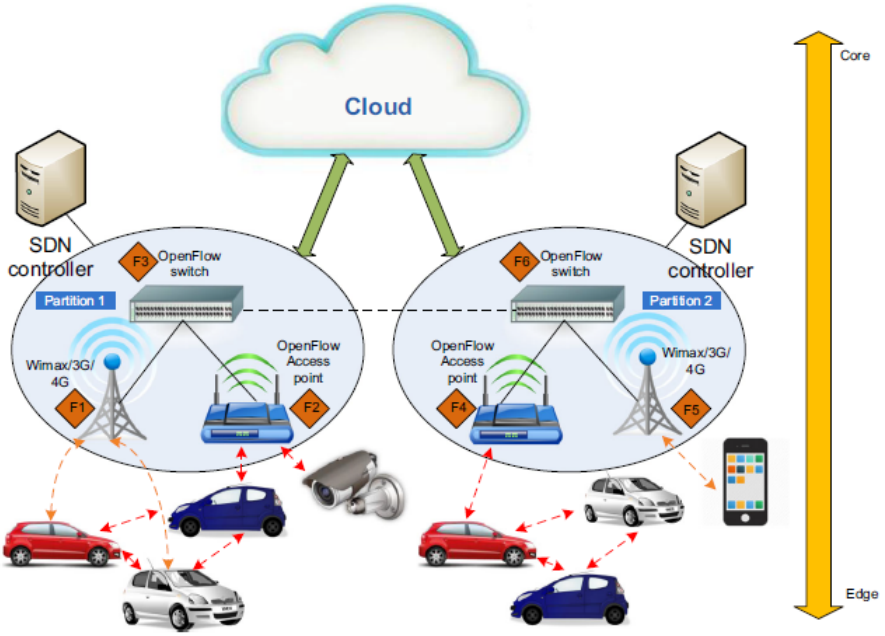
- Sensing devices or as we call them 'things'
- Local communication network
- The Internet Cloud
- The Back-end IoT applications

. The sensing devices collect/sense data from the physical environment which is then used in applications such as Smart Transportation and so on to provide a desirable service to end customers. Since IoT devices are in general characterized by very limited memory and computational resources, IoT application usually takes advantage of services offered by the Cloud for data storage and processing.Furthermore,in order to reach the Cloud, sensing devices rely on different communication technologies[?]. Many emerging IoT applications require real-time interaction and mobility support (e.g. smart traffic lights and target tracking systems), which makes network latency an important limiting factor. Latency introduced in the network is not only a consequence of long distance between IoT devices and the Cloud but its also caused by queuing delay, which is non-negligible on the congested links.[?].

In order to address these challenges,Slavica Tomovic et.al[?] proposed a model of IoT architecture which took advantage of both the Software Defined Network(SDN) and Fog computing paradigms.

Novelty introduced with SDN as opposed to a traditional communications network was a clear separation of the control plane(signalling traffic, performing route calculation, system configuration and management) and the data plane (transport of packets to destination)[?]. SDN control plane is placed on a logically centralized controller, which maintains a global view of the network,interacts with simple forwarding devices and provides a programming interface for network management applications. In this way, SDN allowed network managers to configure and optimize network resources dynamically via automated programs[?].

The architecture proposed by Slavica Tomovic et.al[?]was intended to realise the IoT scenario where features of both technologies(SDN and Fog) were combined together in one integrated system. The setup(See Figure below)[?] involved end devices with multiple wireless communication solutions, SDN controllers, heterogeneous Fog infrastructure (virtualized servers, routers, access points, etc.) and Cloud in the network core[?].



SDN architecture for IoT based on Fog computing

The fog nodes exposed a set of APIs (Application Programming Interfaces) for application deployment and development, resource management and control.Since IoT applications may be geo-spatially distributed, they assumed hierarchical deployment of Fog network. Mobile Fog programming model[**?**] was used in the development of IoT applications using hierarchically deployed and heterogeneous Fog resources. The final application consisted of multiple processes that performed different tasks based on the device capability and position in the network hierarchy.

For example, tasks of large-scale video surveillance application were organized in three levels: motion detection at IP camera, face recognition at edge Fog nodes and aggregation of identities at Cloud server[**?**].

As discussed earlier, since IoT applications deal with dynamic workload due to periodic or event-driven data delivery models, they should be transparently scaled at the runtime without resource over-provisioning. In order to achieve that, Slavica Tomovic et.al[**?**] proposed logical centralization of orchestration functionality at the SDN controller. To achieve that,the design of SDN controller was modified compared to traditional one used in DC networks[**?**]. So, the basic function of the traditional SDN controller was modified to perform

- Fog Orchestration
- Speading routing Logic into SDN-enabled network elements
- Optimally selecting access points for IoT devices.

In order to perform the above tasks, the controller collected and maintained information about fog nodes such as available RAM, storage, running Operating Systems,software applications,state and interconnectivity of the network elements and characteristics of the connected smart devices. The fog orchestration was then performed according to the business policies defined by the application service providers and the policies were stored in the SDN controllers and the fog nodes hosting the provider's application[**?**].

SDN controller therefore provided dynamic,policy-based management of Fog services. It could thus track the moving devices and predict their potential destinations in the near future. This enabled seamless handover to a new Fog node at the network edge [**?**]which would otherwise not be possible using a traditional cloud because of its centralised topology.

Besides Fog orchestration, SDN controller also performed traffic control and connectivity management for IoT devices. Another Main component that was introduced in the architecture was a software OpenFlow switch.The application traffic always went through this component before being sent, which allowed the control plane elements (i.e. Fog nodes at the edge and SDN controllers) to identify the access of traffic flows into the network[**?**].

Important Use cases of such a system would be Smart Transportation, Video Surveillance etc where low latency and real time streaming takes preference.Besides Fog computing, the presented system model can also exploit benefits of SDN to dynamically assign higher priority to some traffic flows in emergency situations, and hence guarantees low-latency. The applications are discussed in detail under the Applications section.

2.3 OpenFog Reference Architecture

The OpenFog Reference Architecture is the product of the OpenFog Architecture Workgroup, co-chaired by Charles Byers (Cisco) and Robert Swanson (Intel)[**?**]. It represents the collaborative work of the global membership of the OpenFog Consortium[**?**]. The OpenFog Consortium was formed in November 2015 and is based on the principle that an open fog computing architecture is necessary in todays increasingly connected world[**?**].

The main idea behind OpenFog RA was to help business leaders, software developers, silicon architects, and system designers create and maintain the hardware, software and system elements necessary for fog computing and it provides a medium- to high-level view of system architectures for fog nodes and networks.

To sustain IoT momentum,which usually comes with high data velocity and volume,the OpenFog Consortium is defining an architecture to address infrastructure and connectivity challenges by emphasizing information processing and intelligence at the logical edge which is Fog Computing.

The OpenFog RA describes a generic fog platform that is designed to be applicable to any vertical market or application. This architecture is applicable across many different markets including, but not limited to, transportation, agriculture, smart-cities, smartbuildings, healthcare, hospitality, financial services, and more, providing business value for IoT applications that require real-time decision making, low latency, improved security, and are network-constrained[**?**].

The OpenFog RA is driven by a set of core principles called pillars[**?**]. The pillars can be thought of as the key attributes that a system needs to adhere to to embody the OpenFog definition of a horizontal, system-level architecture that provides the distribution of computing, storage, control, and networking functions closer to the data source[**?**].

The Pillars[**?**] are

- Security - Security in OpenFog RA describes all of the mechanisms that can be applied to make a fog node secure. All fog nodes must employ a hardware-based immutable root of trust. The Hardware Root of Trust is a trusted hardware component which receives control at power-on. It then extends the chain of trust to other hardware, firmware, and software components. The root of trust should then be attestable by software agents running within and throughout the infrastructure.
- Scalability - Since the hierarchical nature of Fog adds Scaling opportunities, this pillar addresses the dynamic technical and business needs behind fog deployments. Because of the variability in the use cases for fog computing, the OpenFog RA enables elastic scaling based on Demand.
- Openness - Openness as a basic principle enables fog nodes to exist anywhere in a network and span across networks thereby enabling pooling by discovery, which means that new software-defined fog nodes can be dynamically created to solve a business mission.
- Autonomy - This pillar enables fog nodes to continue to deliver the intended functionality in the face of the external service failures. In this architecture, autonomy is supported throughout the hierarchy and therefore the decision making will be made at all levels of a deployments hierarchy including near the device or higher order layers. It does not rely upon a centralized system for operation (e.g. a backend cloud)
- Programmability - This pillar enables highly adaptive deployments including support for programming at the software and hardware layers. This means that we can completely automate re-tasking of a fog node or a cluster of fog nodes. The fog nodes' inherent programmability interfaces is made use of in such a case.
- Reliability, Availability and Serviceability (RAS) - Hardware, software, and operations are the three main areas of the RAS pillar and as such takes on great importance in the OpenFog RA. A reliable deployment will therefore continue to perform as expected under all conditions. Availability ensures continuous management and orchestration, which is usually measured in uptime while Servicing a fog deployment ensures correct operation.
- Agility - This pillar addresses the business operational decisions for an OpenFog RA deployment. It also deals with the highly dynamic nature of fog deployments and the need to respond quickly to change.

- Hierarchy - This pillar ensures that the OpenFog RA is complementary to traditional cloud architectures. Depending on the scale and nature of the scenario being addressed, the hierarchy may be a network of smart and connected partitioned systems arranged in physical or logical layers, or it may collapse into a single physical system (scalability pillar).

2.3.1 OpenFog RA description - Views and Perspectives

The OpenFog RA description is a composite representation of various stakeholder concerns which are referred to as views[?]. The stakeholders and their associated views are identified because they are required to facilitate any successful fog based deployment[?].

The functional viewpoint of the architecture describes how the OpenFog architectural elements and views are applied to satisfy the stakeholders requirements/concerns on a given scenario. It is however important to note that these change over time.

The Deployment viewpoint addresses how the fog software and fog systems are deployed in order to satisfy a given stakeholder scenario. The deployments generally happen in a Hierarchical based model where the number of tiers will be dictated by the scenario requirements[?]. The scenarios might be the amount and type of work required by each tier or the Number of sensors required for the deployment and so on.

The abstract architecture comprising of the views and perspectives is shown below[?]

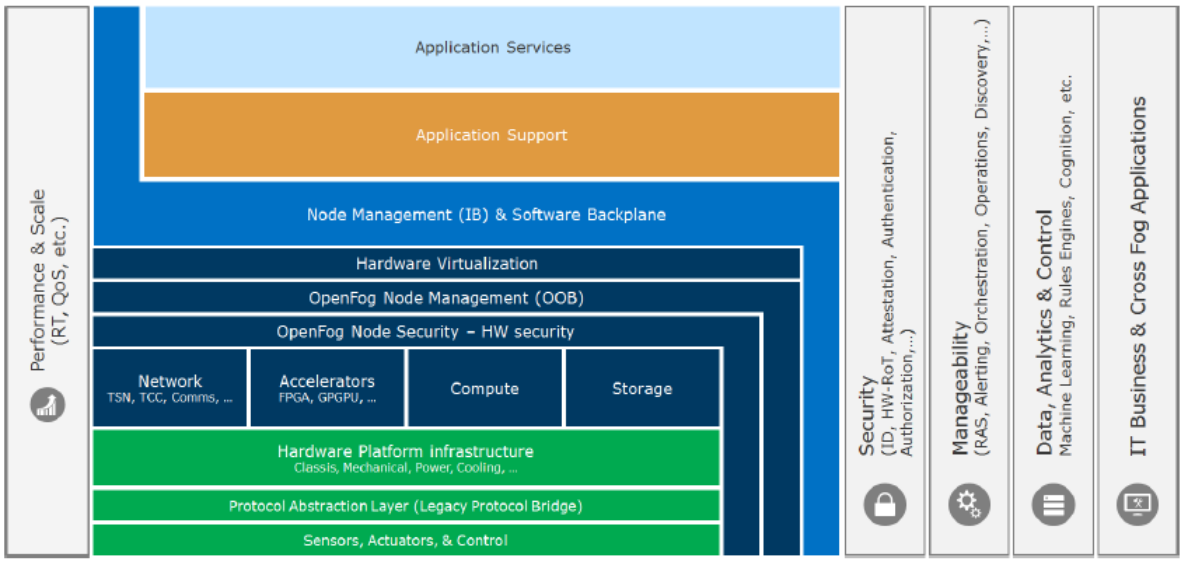


Figure – Architecture Description with Perspectives

The above figure includes grey vertical bars on the sides of the views which are referred to as Perspectives[?].

Perspectives can be thought of as concerns that are to be employed in the fog implementation.They are listed below.

- Performance - This is a cross cutting concern as it impacts system and deployment scenario. Eg Low Latency
- Security - Data integrity is a special aspect of security for devices that currently lack adequate security. This includes intentional and unintentional corruption.
- Manageability - Managing all aspects of fog deployments, which include RAS, DevOps etc.
- Data Analysis and Control - The Autonomy of fog nodes requires localized data analytics coupled with control.
- IT Business and Cross Fog Applications - Ability to migrate and properly operate at any level of a fog deployments hierarchy.

The views(representation of stakeholder concerns)[?] described in the OpenFog RA description include

- Software View - Represented in the top 3 layers shown in figure above and Include Application Services, Application Support, and Node Management (IB) and Software Backplane.
- System View - Represented in the middle Layers shown in fig above and include Hardware Virtualization down through the Hardware Platform Infrastructure.
- Node View - Represented in the bottom layers shown in fig above and includes the Protocol Abstraction Layer and Sensors, Actuators, and Control.

2.3.2 OpenFog Reference Architecture Deployment Models

In most fog deployment models,there are usually several tiers (N-tiers) of nodes. Depending on the use-case/scenario, multiple fog and cloud elements may collapse into a single physical deployment. Each fog element may also represent a mesh of peer fog nodes in use cases like connected cars, electrical vehicle charging, and closed loop traffic systems. The models discussed below[**?**] use a combination of fog and cloud deployment to address various domain scenarios as framed by the layered view of IoT systems.

- - Fog Model Independent of the Cloud This model is mainly used in cases where cloud cant be used. The reasons might be response time concern, compliance to regulations, security and privacy, and unavailability of a centralised cloud. Use Cases include armed forces combat systems, drone operations, some healthcare systems, hospitals, and ATM banking systems[**?**].
- - Model using Cloud for Business Support This model uses cloud for Business Support (information processing related to decision making). However the actual Operation-centric information processing is done by fog deployments located close to the infrastructure/process being managed. Use Cases include commercial building management, commercial solar panel monitoring, and retail[**?**].
- - Model using Cloud for Business and Operational Support This model uses local fog infrastructure for time-sensitive computation, while the cloud is used for the balance of operational and business-related information processing. Use Cases include commercial UPS device monitoring, mobile network acceleration, and content delivery networks (CDNs) for Internet acceleration[**?**].
- - Model using Minimal/no Fog infrastructure This model leverages the cloud for the entire process due to the constrained environments in which the deployment of fog infrastructure may not be feasible or economical. However fog nodes that are at device layer may get some monitoring and control functions for safety related control. Use cases include agriculture, connected cars, and remote weather stations[**?**]

The models discussed above might be summarised using the figure given below [**?**].

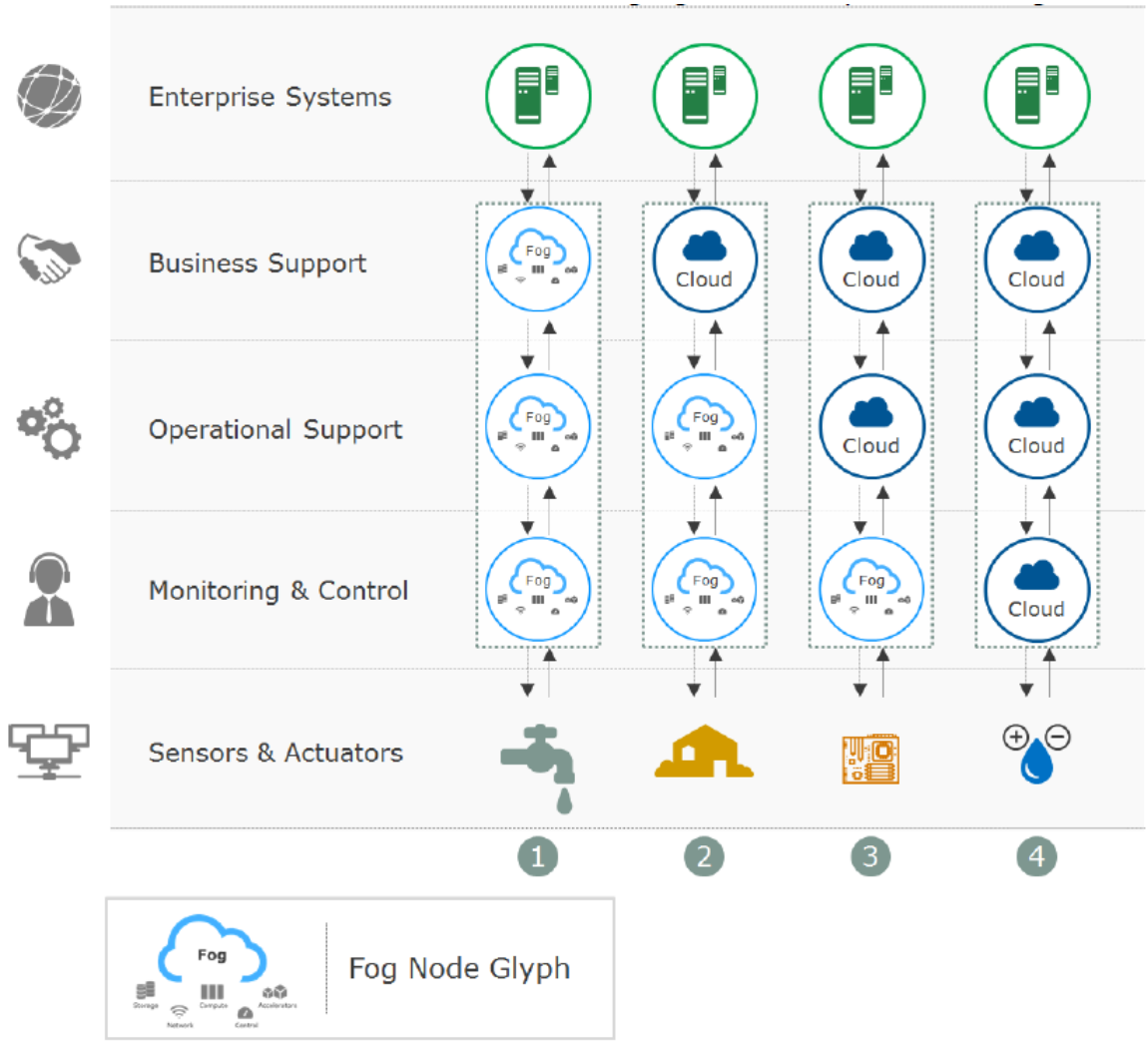


Figure – IoT System Deployment Models

2.4 Fog Computing Vs Cloud Computing vs Edge Computing

Fog computing is a highly virtualized platform which provides computation, storage between the end nodes in the IoT and traditional clouds[?].

There are some differences between fog and cloud computing.The key difference is the fogs proximity to the underlying accessing nodes. In contrast to the cloud, which is more centralized,fog computing targets the services and applications with widely distributed deployments[?]and hence is more localized. Fog can therefore be seen as a descended cloud. The Quality of the core network hence becomes richer when compared to a cloud as fog is available locally. Fog also provides improved Mobility as the nodes extend close to the edge of the network and this in turn provides Location awareness as well which is not provided by traditional cloud. Furthermore, Access methods of cloud are fixed and wireless while that of fog is mostly wireless.

Fog computing also is often erroneously called edge computing, but there are key differences. Fog works with the cloud, whereas edge is defined by the exclusion of cloud[?]. Fog is hierarchical, where edge tends to be limited to a small number of layers. In additional to computation, fog also addresses networking, storage, control and acceleration[?].

A quick use case example to illustrate the value of fog[?]: Consider an oil pipeline with pressure and flow sensors and control valves. One could transport all those sensor readings to the cloud (perhaps using expensive satellite links), analyze the readings in cloud servers to detect abnormal conditions, and send commands back to adjust the positon of the valves. There are several problems with this scenario: the bandwidth to transport the sensor and actuator data to and from the cloud could cost many thousands of dollars per month; those connections could be susceptible to hackers; it may take

several hundred milliseconds to react to an abnormal sensor reading (during which time a major leak could spill significant oil); and if the connection to the cloud is down, or the cloud is overloaded, control is lost.

Now, consider having a hierarchy of local fog nodes near the pipeline. They can connect to sensors and actuators with inexpensive local networking facilities. Fog nodes can be highly secure, lessening the hacker threat. Fog nodes being closer, can react to abnormal conditions in milliseconds, quickly closing valves to greatly reduce the severity of spills. Local control in the fog nodes produces a more robust control system. Moving most of the decision-making functions of this control system to the fog, and only contacting the cloud occasionally to report status or receive commands, creates a superior control system.

3 Common Issues

Though still an emerging field, previous research, such as Yi et al.'s "survey on fog computing" [**?**], has been able to identify multiple potential issues related to fog computing. In this section, we summarise their main findings, and provide additional discussion and research related to them whenever possible.

To improve clarity, we organise issues around 5 main areas: networking, optimal use of resources, fault tolerance, application development, and security and privacy. Note that we focus purely on technical issues. Business-related aspects, such as implementation of a viable business model, and billing mechanisms, are not covered.

3.1 Networking

In order for the fog to function properly, the network has to provide nodes with connectivity, and additional network services, such as routing. The particular nature of the fog, though, makes the implementation of these functions difficult.

For example, the network has to be highly scalable, providing support for a large number of potential nodes. In addition, it should account for constant topology changes due to node mobility. Ensuring that the system is fully distributed is also an important aspect [**?**].

Using virtualisation mechanisms, such as SDN, has been deemed as viable solution to these issues [**?**]. In their proposal of a general architecture for the fog, Bonomi et al. [**?**] state that the fog should use virtualisation for "key resources", including networking. Providing an implementation of SDN for the fog, however, is still an open issue [**?**]. Partly, this appears to be the case because SDN itself does not put "a high emphasis" on distribution [**?**].

3.2 Optimal resource use

As stated before, the fog is highly heterogeneous. This heterogeneity is greatly reflected by different degrees of resource availability throughout the system. Important resources are storage, computation power and bandwidth. For example, in parts of the system, available bandwidth might be high due to the presence of more powerful network links, while in others, it can be a scare resource.

Naturally, these resources should be "optimally" used. However, the actual optimisation goal is highly dependent on the use case: for example, in a real-time application, the main objective is to ensure a small delay. Using more bandwidth or computation power to meet this goal is a valid trade-off. For a computation-heavy application running on a mobile device, in contrast, reducing the amount of computation performed locally on the device might be most important.

In their paper, Yi et al. [**?**] present several strategies that might be used to optimise resource use under different circumstances. Firstly, they suggest that the adequate placement of data can help optimising bandwidth use. In the previously given example of a real-time application, storing data on nodes that are well-connected to the consumer could significantly reduce delay. This placement, however, has to take the dynamic nature of the fog into account. If a node changes location, for example, data placement on the same node can result in small latencies at one time, but introduce great delays at another time. Even if the location of the consumer does not change, available bandwidth at a link might, e.g. if more nodes are interested in the data. Besides placing data, the authors also suggest to place computation. Using "computation offloading", an operation can be partly or fully delegated to a different node in the network.

In the aforementioned example of the computation-heavy application, for instance, a more powerful node could perform most of the computation-intense work. Determining which parts of the computation to offload to which nodes, though, can be challenging. As for data placement, this is largely due to the dynamic nature of the fog. Lastly, they present different methods based on the concept of adjusting the network topology. For example, they suggest that effectively choosing the relay nodes for one or multiple endpoints could help reducing delay, while increasing throughput. Again, however, constant changes in the environment, especially the topology, make an implementation challenging.

Focusing less on choosing locations for data and operations, and more on resources actually available at a given node, Aazam and Huh [**?**] present a resource management method that has been developed especially for the dynamic environment of the fog. At its core, their method predicts the resources required by a consumer for the use of a specific service, and uses these predictions to give "guarantees" about resource availability. For example, a consumer might get a guarantee of 80% availability for a particular service, meaning that it will have access to all resources required to run the service for most of the time. Predictions are largely dependent on past consumer behaviour. If the node in the previous example had, for instance, frequently disconnected from the service provider, its resource guarantee would be lower. Basically, this means that if not sufficient resources are available, they would preferably be given to nodes that make better use of them. As it can be easily seen, this makes resource allocation considerably fair.

3.3 Fault tolerance

As describe before, the fog mainly uses unreliable wireless network links. In addition, nodes are highly mobile. Being able to ensure availability of services, and provide reliability in general are therefore important aspects.

To improve service availability, Yi et al. [**?**] suggest to adjust the network topology (see section 3.2). For instance, they present the idea of dividing a network into several clusters, with each cluster centred around a "rich-resource" node.

Traditionally, reliability in a distributed system can be provided by the means of techniques such as checkpointing or rescheduling (see [**?**]). According to Yi et al. [**?**], though, most of these techniques are unfit for the fog, as they introduce too much delay. They conclude that replication might work, but they expect it to be difficult to implement due to the distributed nature of the system. Additional research on the topic does not seem to exist. Madsen et al. [**?**] claim to provide such, but fail to give any actual fog computing-related insights.

3.4 Application Development

As stated in section XX, the fog is dynamic in regards to network topology, and resource availability. In addition, fog nodes might run on different platforms and system architectures [**?**]. Developing applications that are able to run in this environment, and provide high compatibility, can be expected to be difficult.

To ease development, Bonomi et al. [**?**] propose a "fog abstraction layer" that hides the underlying heterogeneity, and provides developers with a "uniform and programmable interface". Yi et al. make a similar suggestion by calling for a "unified interfacing and programming model" [**?**].

Due to issues mentioned in the beginning, though, we expect that the implementation of such a layer is challenging.

3.5 Security and Privacy

Many applications that have been proposed for fog computing are safety-critical, and/or process sensitive data. For example, in vehicle-to-vehicle communication, an insecure system that allows attackers to remotely control the car could have disastrous consequences. In home automation, users might be worried about giving third parties insights into their daily routine.

Stojmenovic and Wen [**?**] find that providing authentication throughout the system is one of the "main security issues" for fog computing. As an example, they describe a smart meter that is modified by a user, and reports then, due to a lack of proper authentication, false readings. As a possible solution, they suggest encryption at node-level. For this, the meter would encrypt its data, and another node would

decrypt it before further forwarding the data. Similarly to this, the OpenFog consortium deems access control (to which it counts authentication) as "key to building a secure system" [**?**].

In addition to advocating access control, the consortium's reference architecture for fog computing also defines a hardware component called "root of trust" that is "at the heart of the [...] security of the fog node". This component is tamper proof, and required to be implemented by every fog node. It provides security by creating a "chain of trust", i.e., selecting other components such as hardware, software, or other nodes that it considers trustworthy. If a component is compromised, like the smart meter in the example above, it would not gain trust from the root, and therefore not do any harm.

Though access control and the chain of trust promise to provide a solid foundation to a secure fog system, they are both rather general methods. In order to improve security in a given context, it has been suggested in [**?**] to select additional measures based on the specific use case.

To protect privacy, Yi et al. [**?**] suggest to run "privacy-preserving" algorithms before data is transferred from the fog to the cloud. As examples, they mention techniques based on differential privacy and homomorphic encryption. Gerla [**?**] makes an interesting point by stating that moving processing from the cloud to mobile devices alone gives users more control over their data. It can be easily seen, however, that this requires the implementation of adequate control mechanisms. The aforementioned reference architecture [**?**] vaguely describes "privacy attributes" that a user can assign to his/her data, suggesting that these might be used to control its use.

4 Applications

Fog computing was conceptualized as an extension of the cloud to address services and applications for which the cloud paradigm is not entirely suitable [**?**]. As a relatively new model, the potential applications and likely infrastructure and design challenges for the fog are still being explored. However, there is a wide range of possible uses of a paradigm that enables real-time, low-latency processing, reduces bandwidth costs, with the benefits of improved security and governance. This section will detail some possible applications of fog computing in various fields.

4.1 Real-time Health Monitoring

Wireless Body Area Networks (WBAN) is an important technology in healthcare Internet-of-Things (IoT) applications that allows for the unobtrusive monitoring and recording of various vital signs of a patient in real-time. In current cloud-enabled health-monitoring systems, these WBAN devices are in a state of constant connection with a cloud server, sending patient health data to the cloud around the clock [**?**]. However, all of this data being transmitted and received by so many devices on a network can lead to high network bandwidth usage, and consequently, high latency and slow response times from the health monitoring system. In addition, proper analysis can only be performed after relevant data from all WBAN devices in a patient are received by the system. With each individual WBAN transmitting its raw data to the cloud via Wi-Fi, bottlenecks in the analysis might be caused by any single device that is slow or fails to upload data [**?**]. A small dip in the vitals of a patient on life support can quickly turn into a full-on drop within minutes, thus a model that can effectively and reliably monitor this health data in real-time is absolutely critical.

Gia et al., in a case study exploring a practical application of fog computing for WBAN health data feature extraction, suggest the "provision of an extra layer in between a conventional gateway and a remote cloud server"[**?**]. This added layer, described as the "fog layer" in the study, is essentially a fog gateway that pre-processes health data from the various sensor nodes, reducing the volume of data transmitted to the cloud and expediting the response time of vital applications as a result. As detailed in the study, instead of hundreds of WBAN devices each sending patient health data directly to a cloud server, the WBAN data is sent to a fog gateway. The fog gateway aggregates all of the WBAN data and groups together data that is relevant for each user. The gateway pre-processes this data, storing recent results in a distributed fog database where it may be accessed by the patient or their doctor as permitted by the hospital. The pre-processed data is then sent to the cloud server for storage and to be related to data from other patients.

The fog computing model proposed by Gia et al. showcases how fog computing extends the cloud to allow for vital processes to be executed much closer to the edge. Since the raw WBAN data is collected and pre-processed at the fog layer, the number of devices communicating with the cloud is cut down from several hundred WBAN devices to a single fog gateway. As a result, the volume of data transmitted by the system is drastically decreased, affecting overall latency and response time. The model proposed in the study successfully reduced the size of data being sent to the cloud by at least 93% while reducing latency more than 48.5% in busy Wi-Fi networks [**?**].

4.2 Security and Surveillance

Surveillance and security camers generate massive volumes of data; a single camera may create over one terabyte of high-definition video data per day [**?**]. An effective security system would require several security cameras in a number of locations each sending large amounts of video data to a cloud server at a remote location. This puts a great strain on the network, effectively forcing the system to dedicate a significant portion of its bandwidth solely for video transmission, resulting in high latency and delays in busy networks. Security decisions must be made rapidly as situations may arise where a delay in response can result in a risk to public safety. If a person carrying a suspicious package is discovered on a camera, the system must be able to reliably track the package's movements. Delays in the system can result in out-of-date information being relayed to the authorities, making a timely response impossible.

The Open Fog Consortium claims fog computing may serve as a solution for this scenario [**?**]. The paper recommends the deployment of fog nodes that can be designed to intelligently partition video processing between the various edge devices and the cloud. In contrast to traditional cloud models, this design

allows video analytics to be performed at fog nodes that are physically close to the site, reducing latency and allowing for quicker response times in emergency situations. Relevant data can then be sent to the cloud to allow for historical analysis over long periods of time and enable data sharing between multiple locations [**?**], for example, the sharing of data between airports or multiple locations of a hotel chain.

The addition of an extra layer between the edge and the cloud allows for added security protocols to be implemented at the fog gateways where the raw camera data is processed before being sent to the cloud. This raw camera data can be analyzed at the fog gateways, where rules may be established to determine which data may be sent to the cloud and which can be stored locally for security analysis purposes [**?**].

4.3 Smart Cities - Traffic Congestion Management

Traffic management is an ever-growing challenge faced by major cities; lost worker productivity, slower response times from emergency services, and high carbon emissions can all be attributed to traffic congestion. Various factors that contribute to traffic congestion simply can not be predicted, such as accidents. Even the predictablity of how certain weather affects road conditions can vary wildly between different areas. Furthermore, traffic management spans multiple jurisdictions; the development and implementation of potential solutions will probably take place in isolation within each department, hindering information sharing and integration initiatives [**?**].

According to the Open Fog Consortium, proposed cloud solutions in Smart Cities, while suggesting a single cloud that connects these many departments, in reality involve multiple clouds for traffic management [**?**]. The paper notes that these departments are each responsible for the implementation of their own solutions to the traffic congestion problem for areas under their jurisdiction and, as a result, may potentially work with any number of clouds. This segmented approach, where each piece of the solution is developed and deployed in isolation, can result in the accumulation of redundant data and is an obstacle to data sharing between departments which in turn creates delays in response.

In a paper published by the Open Fog Consortium, a fog-enabled model is proposed to solve these problems [**?**]. CCTV, camera, electronic signage, and traffic light data will be captured by roadside or local fog nodes to be analyzed and used locally. These local fog nodes process this local data and transmit emergency messages (such as flooding or roadblock notices) at the local level. This local information would then be sent to a regional fog node where it would be related to data from other local fogs to piece together a picture of traffic conditions at a larger scale. These fog nodes at the regional and city level may be used to share information across individual networks, allowing for all relevant departments to share a single, synchronized picture of road-side conditions as they change. In the future, as manufacturers install fog nodes in their vehicles, each vehicle could potentially function as an edge device that connects with other devices on the road. All of this data in aggregate can be used by municipalities to provide a far more detailed view of traffic conditions, allowing for greater control of road conditions and much quicker response times. In addition, all of this node information, from local fog data up to city fog data, can be sent to any number of clouds, effectively allowing for synchronization of information between multiple jurisdictions that may employ different cloud services without disrupting their existing cloud-enabled implementations.