Chapter-1
Introduction

1.1 Fog Computing

1.2 Fog Computing Architecture

1.3 Issues Related to Fog Computing

- 1.3.1 Privacy
- 1.3.2 Network Security
- 1.3.3 Network Management
- 1.3.4 Placement of Fog Servers
- 1.3.5 Delay in Computing
- 1.3.6 Energy Consumption

1.4 IoT-Based Architectures and Protocols

- 1.4.1 Three and Five-Layer Architectures
- 1.4.2 IoT Device Connectivity: Architectures and Protocols
- 1.4.3 IoT Protocol Architecture
- 1.4.4 Layer IoT Architecture
- 1.4.5 Five-Layer IoT Architecture
- 1.4.6 Types of IoT Connections

1.5 Cloud And Fog Based Architectures

1.6 Social IoT

1.7 Implication of Fog Computing

1.8 Fog Computing Task Scheduling

- 1.8.1 Static Scheduling Strategy
- 1.8.2 Dynamic task scheduling methods
- 1.8.3 Hybrid task scheduling methods

1.9 Fog Computing Challenges

- 1.9.1 Drones
- 1.9.2 Machine learning
- 1.9.3 Security and Privacy
- 1.9.4 Autonomic Fog Management and Connectivity

To improve services and quality of life for citizens and visitors, several cities have recently made progress toward becoming smart cities. These cities now have improved resource utilization, increased environmental protection, enhanced infrastructure operations and maintenance, and robust safety and security measures. To improve services and performance in their various sectors, smart cities rely on implementing new and existing technologies and various optimization techniques. The IoT[1], FOG computing and cloud computing are a few of the technologies assisting smart city applications. These three can be combined into one system, an integrated IoT-Fog-Cloud system, to create a sophisticated platform for creating and managing various kinds of smart city applications. With the help of this platform, applications will be able to deliver the best functionality and performance possible by utilizing the best features of IoT gadgets, FOG nodes, and cloud services. Numerous opportunities for improving and optimizing applications in the fields of energy, transportation, healthcare, and other industries will be presented by the use of this strong platform. The improvised SMART FOG system design would be the main focus of this research project.

1.1 Fog Computing

Fog computing is referred to as a distributed computing paradigm that essentially extends the cloud's services to the network's edge. According to Cisco, Fog computing is a continuation of the cloud computing paradigm from the network's core to its edges. It makes networking, computing, and storage between end devices and conventional cloud servers easier. Fog computing uses both the cloud and the edge devices that are situated between end devices and cloud servers to run applications rather than only using the cloud for this purpose. Edge and cloud computing are both benefits of fog computing. While making use of edge devices' proximity to the endpoints, it also uses the cloud's on-demand scalability.

By effectively exploiting the resources present at the edge nodes to do partial computing and by performing filtering operations in the nodes, it essentially lessens the strain on the cloud server. Fog computing is typically confused with two ideas in particular. Mobile Edge Computing and Mobile Cloud Computing are these ideas.

[1]Internet of Things

MCC[2] essentially contends that data processing and storage are carried out on a cloud, away from mobile devices. As a result, it transfers data and processing power from individual mobile devices to the cloud. MEC[3] is a network architecture concept that extends cloud computing capabilities to the edge of the network. It brings computation, storage, and networking resources closer to the end-user or device, reducing latency and improving overall system performance. MEC enables the execution of applications and services at the edge of the network, closer to where the data is generated and consumed. A cloudlet, on the other hand, is a concept related to edge computing and MEC. It refers to a small-scale data center or server cluster deployed at the edge of the network, typically near mobile devices or end-users. Cloudlets provide computational resources and services to nearby devices, offering low-latency access to data and applications.

In comparison, MEC is a broader term that encompasses the concept of cloudlets. MEC involves deploying computing capabilities at various points in the network, such as base stations, access points, or edge routers, whereas a cloudlet specifically refers to a small-scale server cluster. Cloudlets are one implementation of MEC, but MEC can also involve distributed edge computing without using dedicated cloudlet infrastructure. It may be viewed as a more focused version of the cloud computing concept. It resembles a cloud server that is situated at the edge of a mobile network. Fog computing combines these two ideas with some of its characteristics to increase its dependability and utility.

1.2 Fog Computing Architecture

The bandwidth, particularly on cellular networks, is a significant issue with cloud computing. As the IoT grows and more physical devices are wirelessly connected, the issue will only become worse. This issue is resolved by Fog computing, which stores data locally on computers and other gadgets known as fog nodes. Any device having computation, storage, and network connection, such as handheld devices, tablets, PCs, routers, etc., can be used as a fog node.

[2]Mobile Cloud Computing
[3]Mobile Edge Computing

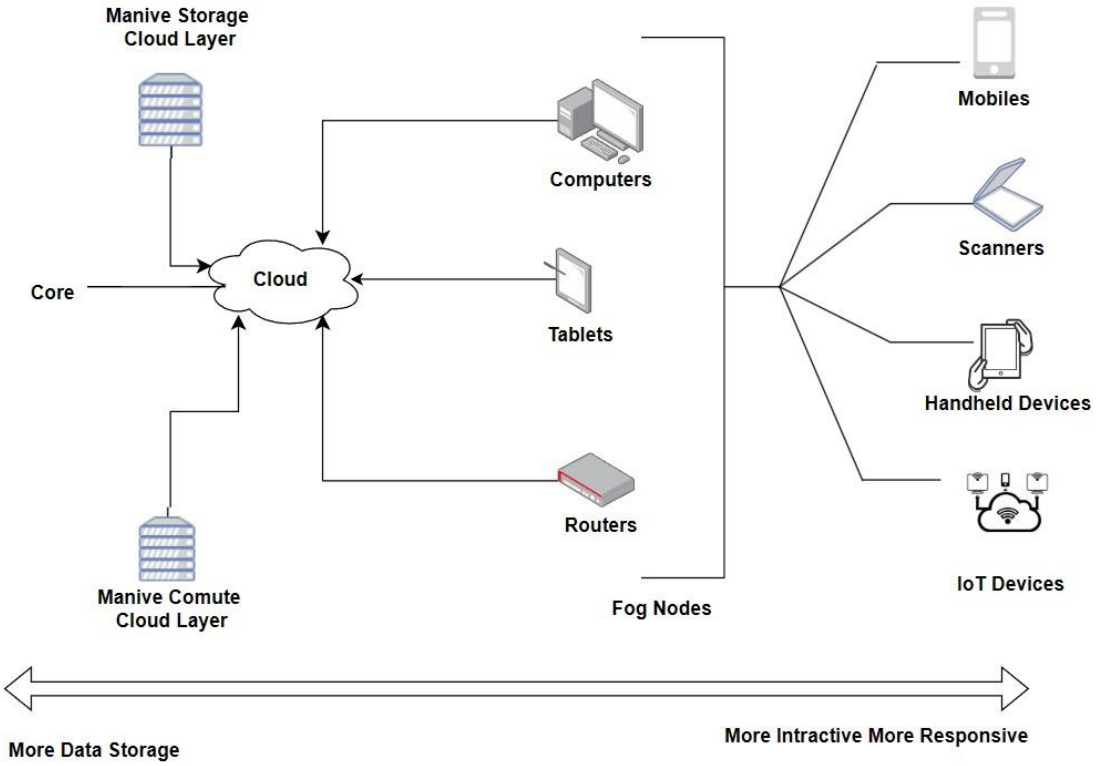

Figure 1.1: Fog Computing Architecture (Lai, 2021)

Figure 1.1 shows Fog-based architecture, fog nodes, also known as edge devices or fog devices, are distributed throughout the network, closer to the data sources and end-users. These fog nodes can be various devices such as routers, switches, access points, edge servers, IoT devices, or other computing resources. The architecture extends the capabilities of cloud computing by providing localized data processing, storage, and analytics at the edge of the network. These fog nodes are controlled by the Fog Data Service, which performs a variety of functions like data reduction, data virtualization, data control and security, and edge analytics. Additionally, data might be uploaded to the cloud for long-term analyses.

Kopras (2023) discussed that the widely adopted cloud computing paradigm is evolving with the integration of fog computing, placing computing nodes in closer proximity to end-users to meet stringent latency requirements. However, effective task offloading, considering transmission and computation energy consumption, poses challenges. Task allocation becomes intricate due to the multitude of arriving tasks with diverse computational, communication, and delay requirements, alongside a variety of computing nodes with differing capabilities. The research work introduces an optimal task allocation procedure aimed at minimizing energy consumption for

wirelessly connected users in a network comprising Fog Nodes located at Access Points and Cloud Nodes. The assignment of Access Points and computing nodes to offloaded tasks, along with Fog Node operating frequencies, is optimized using a Mixed-Integer Nonlinear Programming approach. Realistic energy consumption and delay models, along with their pertinent parameters reflecting device characteristics, are employed. Results indicate the profitability of distributing task processing among multiple Fog Nodes and the cloud, often selecting distinct nodes for transmission and computation. The proposed algorithm demonstrates superior performance, achieving the lowest energy consumption and task rejection rate compared to alternative allocation strategies. Additionally, a heuristic algorithm is presented, decoupling wireless transmission optimization from implemented computations and wired transmission, providing optimal or near-optimal solutions across various scenarios.

1.3 Issues Related to Fog Computing

Cloud computing is expanded by Fog computing, which also affects IoT. These gadgets, also known as fog nodes, can be set up anywhere there is a network connection. Fog computing provides extra storage capabilities at the periphery to handle the demands. As a result, the Fog server must modify its services, which increases administration and maintenance expenses. The operator must also deal with the following problems.

1.3.1 Privacy

Because wireless dominates fog computing, network privacy is a major challenge. The network operator manually creates settings, deploys fog nodes at the edge of the internet, and incurs significant maintenance costs. The exposure of personal information when utilizing networks is receiving more attention. The Fog nodes have easier access to the end consumers. Because of this, Fog nodes gather more sensitive data than faraway clouds. To address these problems, encryption techniques like HAN[4] might be applied.

[4]Home-Area Network

1.3.2 Network Security

Fog networks may be vulnerable to various network-level attacks, such as Denial of Service, Man-in-the-Middle, or network sniffing attacks. It is crucial to implement robust network security measures, including firewalls, intrusion detection systems, and secure communication protocols, to detect and prevent these attacks and protect the integrity and availability of the network. Fog nodes and IoT devices connected to the fog network can be targets for exploitation and compromise. Weak device security, such as default or easily guessable passwords, outdated firmware, or unresolved vulnerabilities, can lead to unauthorized access and control. Implementing secure device configurations, regular security updates, and strong security policies can mitigate these risks.

1.3.3 Network Management

Network management in Fog computing refers to the processes, tools, and strategies used to efficiently control, monitor, and maintain the network infrastructure and devices in a fog computing environment. Fog computing introduces additional complexity to network management due to the distributed nature of the architecture and the heterogeneity of devices involved. Efficient network management in fog computing is crucial to ensure the reliable and secure operation of the fog network. It involves continuous monitoring, optimization of network performance, resource allocation, configuration management, fault handling, and security measures to maintain a robust and scalable fog computing environment. Fog computing environments require continuous monitoring of network performance to ensure efficient and reliable service delivery.

Network administrators need to monitor network traffic, latency, bandwidth utilization, and other performance metrics to identify bottlenecks, congestion, or potential issues that could impact the quality of service. Real-time monitoring tools and analytics are employed to proactively manage and optimize network performance. If SDN[5] and NFV[6] approaches SD are not used, controlling the network, the fog nodes, and the connections between each node would be difficult when linked to heterogeneous devices.

[5] Software-Defined Networking
[6] Network Function Virtualization

1.3.4 Placement of Fog Servers

The placement of fog servers requires careful consideration to ensure optimal performance and cost-effectiveness for the area. One approach to reducing maintenance costs is to thoroughly assess the capabilities and workload of each server node before deployment. Before deploying fog servers, a comprehensive analysis should be conducted to understand the specific needs and requirements of the area. This analysis can involve evaluating factors such as network traffic patterns, latency requirements, data processing demands, and the distribution of edge devices. By examining the workload completed by each server node, it becomes possible to identify the areas where fog servers would be most beneficial. This assessment helps in determining the optimal placement of Fog servers, ensuring that they are strategically located to reduce latency and efficiently process data closer to the source. Additionally, considering the proximity of Fog servers to edge devices can help minimize data transmission delays and enhance real-time processing capabilities. Placing Fog servers near areas with high concentrations of edge devices can improve response times and reduce network congestion. Furthermore, it is essential to assess the scalability and flexibility of Fog server deployments. As the needs of the area evolve, Fog servers should be easily adjustable and expandable to accommodate changing demands.

Effective placement of fog servers involves analyzing the workload of server nodes, considering network traffic patterns, optimizing proximity to edge devices, and ensuring scalability. By carefully considering these factors, it is possible to deploy fog servers in a manner that meets the needs of the area while minimizing maintenance costs.

1.3.5 Delay in Computing

Delays in computing can have significant impacts on the efficiency and performance of various services and applications that rely on data processing. One of the primary reasons for delays is the aggregation of data. When data from multiple sources is collected and combined for processing, it may take time to complete the aggregation process, leading to delays in computing. Additionally, resource overuse can exacerbate the delay issue. Fog servers, which are responsible for processing data locally, may become overloaded with tasks, leading to slower processing times. This

resource constraint can hinder the effectiveness of Fog computing services, making them less responsive and efficient. To address these challenges and reduce delays in computing, it is essential to implement efficient data aggregation techniques. Data should be aggregated in a manner that minimizes processing time while ensuring the accuracy and integrity of the information. This involves optimizing algorithms and strategies for data aggregation to achieve faster processing.

Furthermore, Fog nodes, which are distributed computing resources, should be carefully managed to avoid resource overuse. Scheduling algorithms that prioritize critical tasks and consider the mobility of Fog nodes can help distribute the processing load more effectively. By using a priority and mobility paradigm in scheduling, fog nodes can be dynamically allocated based on their availability and proximity to data sources, reducing delays and improving overall performance. Moreover, optimizing the communication and networking infrastructure between fog nodes and data sources is crucial. Efficient data transmission protocols and network configurations can minimize latency and ensure timely data delivery to Fog servers for processing. Overall, addressing the delay in computing in fog environments requires a comprehensive approach that involves optimizing data aggregation, managing resources effectively, and improving communication infrastructure. By doing so, Fog computing services can offer faster and more responsive data processing, enhancing the overall user experience and system performance.

1.3.6 Energy Consumption

In Fog computing settings where multiple fog nodes are used, the distribution of computing tasks can result in increased energy consumption. To address this issue, reducing energy usage becomes crucial. This can be achieved through various strategies, such as employing energy-efficient hardware components, implementing dynamic resource allocation techniques, utilizing sleep mode and power management features, adopting energy-aware task scheduling algorithms, implementing data compression and aggregation methods, monitoring energy consumption, and exploring the integration of renewable energy sources. By implementing these measures, fog computing environments can minimize energy consumption, improve sustainability, and reduce long-term energy costs.

Fog computing, while offering numerous benefits, faces challenges in terms of network security, privacy, interoperability, resource management, and scalability. Network security and privacy concerns arise due to the distributed nature of fog computing, necessitating robust security mechanisms and encryption techniques to protect sensitive data. The heterogeneity of Fog nodes and edge devices poses interoperability challenges, requiring standardization efforts and protocols for seamless communication and device management. Resource management and load balancing become complex with increasing numbers of devices and applications, necessitating dynamic resource provisioning and monitoring. Additionally, scalability becomes crucial to handle the growing demands of Fog computing, requiring scalable architectures and mechanisms for efficient resource allocation. Addressing these issues through effective security measures, interoperability standards, resource management techniques, and scalable architectures is essential for the successful implementation and operation of Fog computing systems.

Fog computing offers substantial benefits but also faces several issues. Security is a paramount concern as distributing computing resources closer to the edge increases the attack surface. Interoperability challenges persist among diverse IoT devices and fog nodes, hindering seamless data exchange. Resource allocation and load balancing are complex due to dynamic workloads. Privacy issues arise from the vast data generated and processed at the edge. Standardization efforts, security protocols, and robust management systems are crucial to address these challenges and unlock the full potential of fog computing, ensuring it can efficiently support IoT applications while safeguarding data and systems.

1.4 IoT-Based Architectures and Protocols

IoT-based architectures and protocols are essential components that enable the seamless integration and communication of various devices and systems in the IoT ecosystem. These architectures and protocols play a crucial role in ensuring efficient data exchange, interoperability, and security in IoT applications.

1.4.1 Three and Five-Layer Architectures

The IoT is a transformative concept that envisions a network of interconnected devices, sensors, and systems communicating and exchanging data to provide innovative services and valuable insights. In the realm of IoT architecture, two

common frameworks are the Three-Layer Architecture and the Five-Layer Architecture. Accordingly, Figure 1.2 shows Three-Layer Architecture comprises the Perception Layer, where data is collected from IoT devices and sensors; the Network Layer, responsible for facilitating communication between devices and data processing systems; and the Application Layer, where data is processed and transformed into meaningful insights. On the other hand, the Five-Layer Architecture presents a more comprehensive model with the addition of the Middleware Layer, which acts as an intermediary for data normalization and transformation, and the Business Layer, where business logic and decision-making occur based on insights generated from the Application Layer. Both architectures play a crucial role in organizing the flow of data and services within the IoT ecosystem, catering to diverse use cases and providing a structured framework for the successful implementation of IoT solutions. The choice between these architectures depends on the specific requirements and complexity of the IoT application at hand. Three-layer design was first used in the early stages of this field of study. The perception, network, and application layers are its three layers.

The physical layer, which has sensors for sensing and gathering environmental data, is the perception layer. It detects certain physical parameters or locates other intelligent objects in the surrounding area. The network layer is in charge of establishing connections with other intelligent objects, network components, and servers. Additionally, it uses its characteristics to communicate and interpret sensor data.

The Application layer, delivering application-specific services to the user is the responsibility of the application layer. It describes a variety of uses for the IoT, including smart homes, smart cities, and smart health. The three-layer design encapsulates the core concept of the IoT, however research on IoT frequently focuses on its more intricate details, therefore it is insufficient. Because of this, the literature has suggested a lot more layered structures. The first is the five-layer architecture, which also has layers for processing and business. Perception, transport, processing, application, and business layers make up the five layers as shown in Figure 1.2, The perception and application layers play the same role as in a three-layer design.

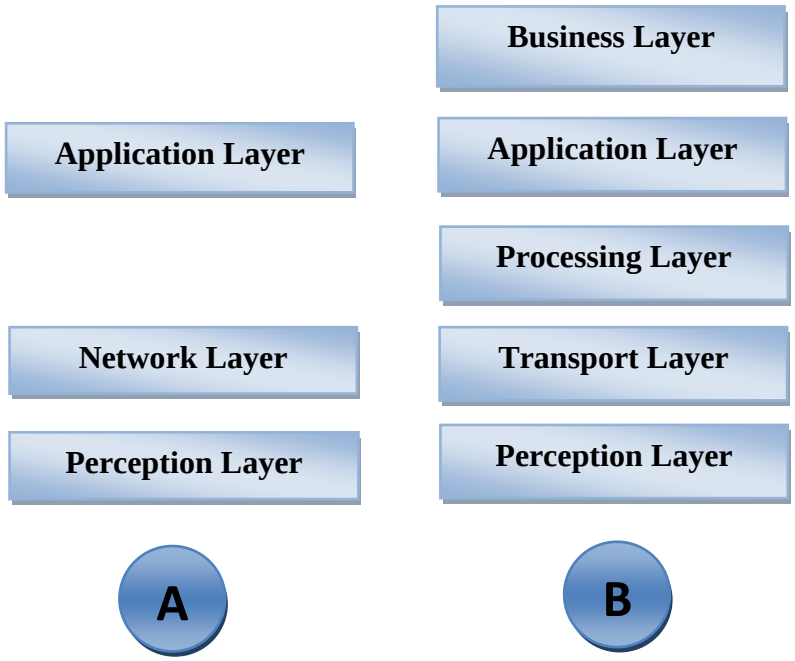

Figure 1.2: Three(A) and Five-Layer(B) Architectures (Lai, 2021)

The Transport layer, through networks including WiFi[7], 3G[8], LAN[9], Bluetooth, RFID[10], and NFC[11], the transport layer moves sensor data from the perception layer to the processing layer and back again. The processing layer, also known as the middleware layer, plays a crucial role in a fog computing architecture. This layer receives a substantial volume of data from the transport layer and performs various tasks such as processing, storing, and analyzing it. It possesses the capability to handle and provide a diverse range of services to the lower tiers. The processing layer leverages different technologies, including modules for big data processing, cloud computing infrastructure, and databases. By utilizing these technologies, the processing layer enhances the overall functionality and performance of the fog computing system. The business layer oversees the whole IoT system, including all applications, revenue streams, and user privacy.

IoT-based architectures and protocols are pivotal in enabling the seamless operation of interconnected devices and systems. These frameworks, including restful APIs,

[7]Wireless Fidelity
[8]3rd Generation
[9]Local Area Network
[10]Radio Frequency Identification
[11]Near Field Communication

MQTT, and CoAP, facilitate efficient communication and data exchange. They play a crucial role in building scalable, interoperable, and secure IoT ecosystems. Selecting the appropriate architecture and protocol depends on specific use cases, emphasizing the need for careful consideration in implementing IoT solutions that align with performance, scalability, and security requirements.

1.4.2 IoT Device Connectivity: Architectures and Protocols

The IoT is only able to function properly and transfer data when all of the connected devices are online and securely linked to a communications network. Standards and protocols for the IoT start to become relevant here. Both IP and non-IP networks can be used to link devices that are part of the IoT. IP network connections are highly complicated and need an increase in memory as well as power from the IoT devices; nevertheless, range is not an issue. On the other hand, non-IP networks have a range constraint and need a far lower amount of power and memory than IP networks do.

1.4.3 IoT Protocol Architecture

The architecture of the IoT is dependent on the functioning and execution of its components in various industries. The IoT is constructed on top of a fundamental process flow, which has two main architectures: a 3-layer architecture and a 5-layer architecture.

1.4.4 Layer IoT Architecture

The most fundamental architecture consists of three distinct layers. It is composed of three layers: the perception layer, the network layer, and the application layer respectively. The physical layer is known as the perception layer, and it is comprised of all of the intelligent sensor-based devices that collect data from their surrounding environment.

The network layer is in charge of establishing connections between the many devices and applications that make up the IoT ecosystem. It is comprised of all of the wireless and wired communication technologies that are currently available. After that, the data is sent to the application layer for processing.

It is the responsibility of the application layer to provide the user with services that are unique to the program. It describes a variety of applications that may be used to implement IoT, including smart homes, smart cities, and health care.

1.4.5 Five-Layer IoT Architecture

The three-layer design has been expanded into a five-layer architecture figure 1.3 shows this by adding two more layers, the processing layer, and the business layer respectively. In the 5-layer design, the perception and application layers function in a manner that is analogous to the 3-layer architecture. Networking technologies such as WiFi, Bluetooth, 3G, RFID, and NFC are utilized by the transport layer to convey the sensor data from the perception layer to the processing layer and vice versa. The processing layer, also known as the middleware layer, is responsible for storing, analyzing, and processing large amounts of data delivered by the transport layer. This layer uses a wide variety of technologies, including databases, cloud computing, and Big Data processing modules. The whole IoT system, including apps, companies, and the privacy of individual users, is managed by the business layer.

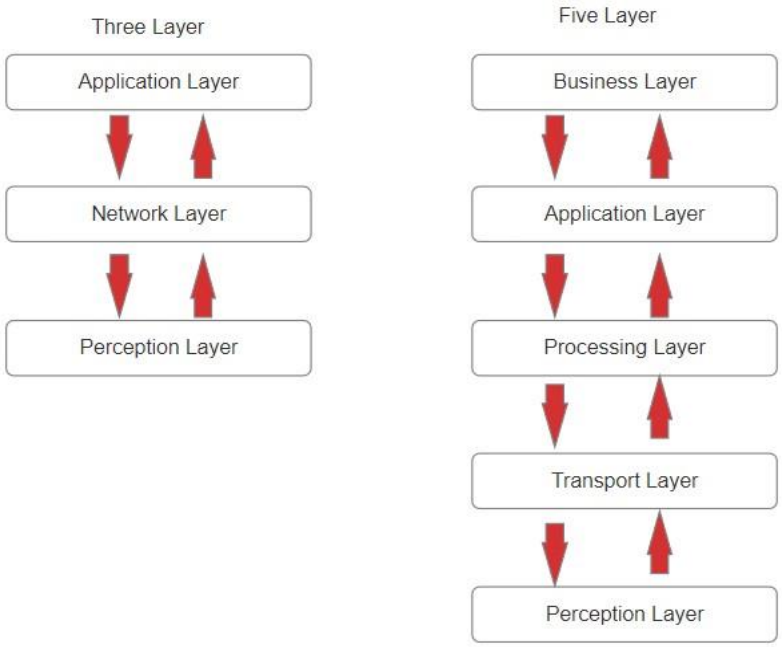

Figure 1.3: IoT Architecture (Lai, 2021)

The Transport layer, through networks including WiFi, 3G, LAN , Bluetooth, RFID, and NFC, the transport layer moves sensor data from the perception layer to the processing layer and back again. IoT architecture can be centralized or decentralized,

depending on the application requirements and scale. The design of the architecture needs to consider scalability, interoperability, data integrity, and energy efficiency to create a robust and reliable IoT ecosystem that can support a wide range of applications and services.

1.4.6 Types of IoT Connections

When it comes to data communication, an IoT system utilizes one of four distinct types of transmission channels. Device-to-device communication, often known as D2D[12] communication enables devices that are physically adjacent to one another to talk to one another via wireless protocols such as Bluetooth, ZigBee, or Z-Wave. By the use of a D2D connection, it is possible to create a link even in the absence of a network in Figure 1.4.

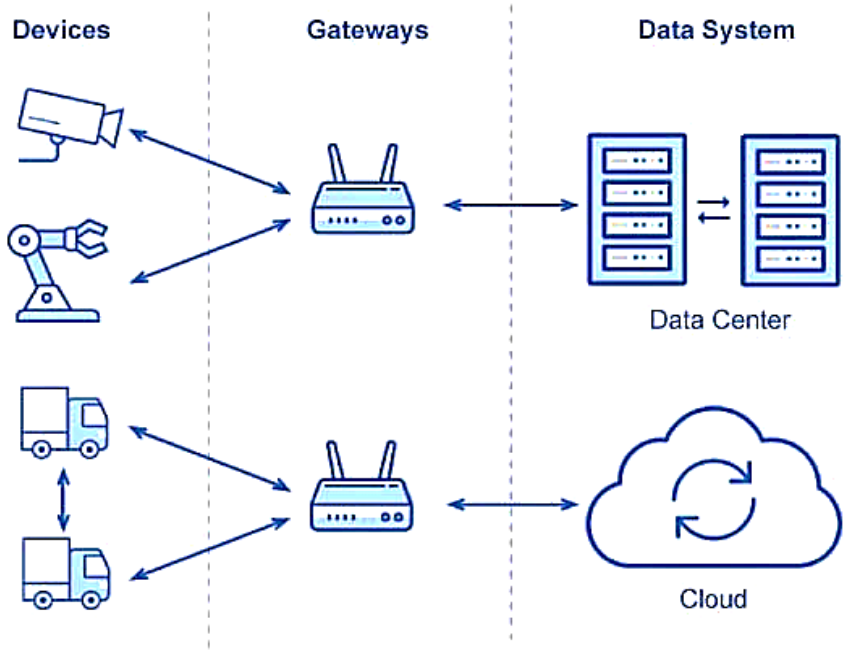

Figure 1.4: Types of IoT Connections (Adel, 2020)

The deployment of an intermediate platform enables communication to occur between devices and gateways at each stage of the network. The majority of the time, gateways are employed for two distinct functions: first, to collect data from sensors and transmit it to the appropriate data system; and second, to evaluate data and transmit it back to the device if any problems are discovered while the data is being analyzed. Both of these functions are essential to the operation of a gateway.

[12]Device-to-Device

When someone refers to "gateway-to-data systems communication," they are referring to the process in which data is sent from a gateway to the appropriate data system. Communication between the many different data systems might take place either within a data center or within the cloud itself. For this sort of connection, the protocols need to be easy to put into action and uncomplicated to include programs that are already in existence. They are required to have high availability, appropriate capacity, and trustworthy disaster recovery capabilities.

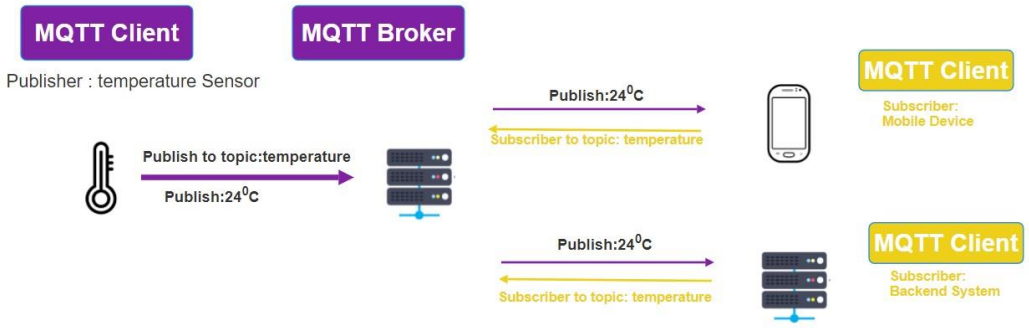

Figure 1.5: Publish / Subscribe Architecture (Ansari, 2018)

Figure 1.5 shows MQTT[13] Publish / Subscribe Architecture there are two types of IoT protocols: Protocols for the network layer: and IoT network protocols that link devices requiring medium to high amounts of electricity to the network. With this protocol, it is possible to communicate data from one end of the network to the other within the network. A few of the most common network protocols for the IoT are HTTP[14], LoRaWAN[15], Bluetooth, and Zigbee.

Data protocols for the IoT: Data protocols for the IoT link low-power IoT devices. These protocols are capable of providing end-to-end communication with the hardware even in the absence of any Internet connection. Connection in the data protocols of the IoT can be accomplished by either a wired or cellular network. MQTT, CoAP[16], AMQP[17], XMPP[18], DDS[19] are some common IoT data protocols.

[13] Message Queuing Telemetry Transport
[14] Hypertext Transfer Protocol
[15] Long Range Wide Area Network
[16] Constrained Application Protocol
[17] Advanced Message Queuing Protocol
[18] Extensible Messaging and Presence Protocol
[19] Data Distribution Service

IoT protocols and network standards: There is a wide variety of IoT protocols available to cater to a variety of applications and needs. Yet, each has its own set of benefits and drawbacks for a variety of IoT use cases. This research work will go through some of the IoT protocols that are the most popular overall.

1.5 Cloud and Fog-Based Architectures

Recently, there has been a shift toward fog computing, a system architecture in which network gateways and sensors perform some of the data processing and analytics. Cloud and fog-based architectures are fundamental paradigms in modern computing. Cloud computing involves centralized data processing in remote data centers, while fog computing disperses computing resources to the edge of the network. Both play key roles in supporting a wide range of applications, balancing data processing, and enabling scalability in an increasingly interconnected world.

Cloud and Fog computing architectures are advanced paradigms for distributed computing. Cloud computing typically involves centralized data centers for resource-intensive tasks, while Fog computing extends this concept to the edge of the network, closer to end-users and devices. Both offer unique advantages.

Cloud computing provides scalability, cost-efficiency, and vast resources for data processing and storage. Fog computing complements this by enabling low-latency, real-time processing and reducing network congestion, making it ideal for applications like IoT and autonomous vehicles. These architectures also foster data security and privacy concerns, which require careful management. Recent research delves into optimizing the integration of Cloud and Fog, ensuring seamless coordination between central and edge resources. This involves developing efficient data transfer, task offloading, and orchestration techniques. Additionally, AI and machine learning are integrated to enhance decision-making processes in these architectures, paving the way for more intelligent, context-aware applications in diverse domains like healthcare, smart cities, and Industry 4.0.

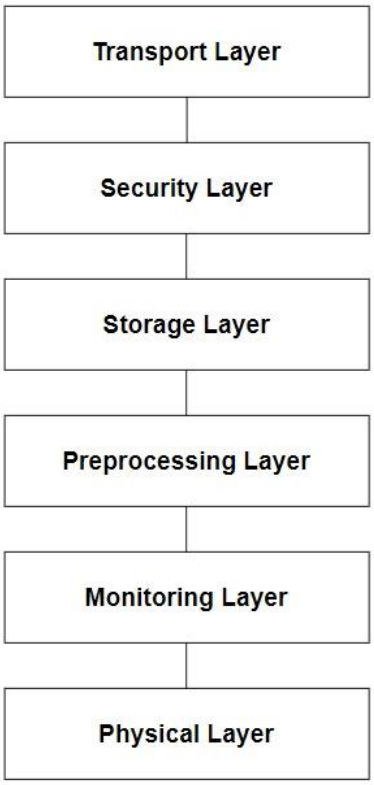

Figure 1.6: Cloud and Fog-Based Architectures (Gupta, 2016)

Figure 1.6 shows illustrate Fog architecture's tiered approach, inserting security, monitoring, and pre-processing layers between the physical and transport levels. Power, resources, and services are all tracked by the monitoring layer. Filtering, processing, and analytics of sensor data are carried out by the pre-processing layer. Data replication, dissemination, and storage are just a few of the storage capabilities offered by the temporary storage layer. The security layer also assures data integrity and privacy and performs encryption and decryption. On the network's edge, monitoring and pre-processing are carried out before data is sent to the cloud. The temporary storage layer in fog computing provides various storage capabilities, including data replication, dissemination, and storage. It serves as a crucial component for managing data within the fog environment. Additionally, the security layer plays a vital role in ensuring data integrity and privacy. It performs encryption and decryption operations to safeguard sensitive information.

Cloud and Fog-based architectures offer versatile solutions for diverse computing needs. Cloud provides centralized, scalable, and reliable data processing, while fog extends computing to the network edge, reducing latency.

1.6 Social IoT

SIoT[20] evaluate social interactions between objects in the same manner that social relationships between people are considered. SIoT represents the integration of IoT technology with social networks and human interactions. It enables smart devices to collect and share data, enhancing user experiences, enabling collaborative decision-making, and fostering a deeper connection between the physical and digital worlds through social engagement and data sharing. The three basic components of a SIoT system are as follows:

- One can navigate the SIoT. We can begin with a single device and browse all of the devices that are linked to it. New devices are simple to find, and services utilize an IoT social network like this.
- There is a requirement for reliability between gadgets
- To analyse the social networks of IoT devices, we can use models similar to those used for researching human social networks.

SIoT refers to the integration of social networking principles and techniques into the IoT paradigm. It combines the power of social interactions and networked devices to enhance communication, collaboration, and information sharing among IoT devices and users.

[20]Social Internet of Things.

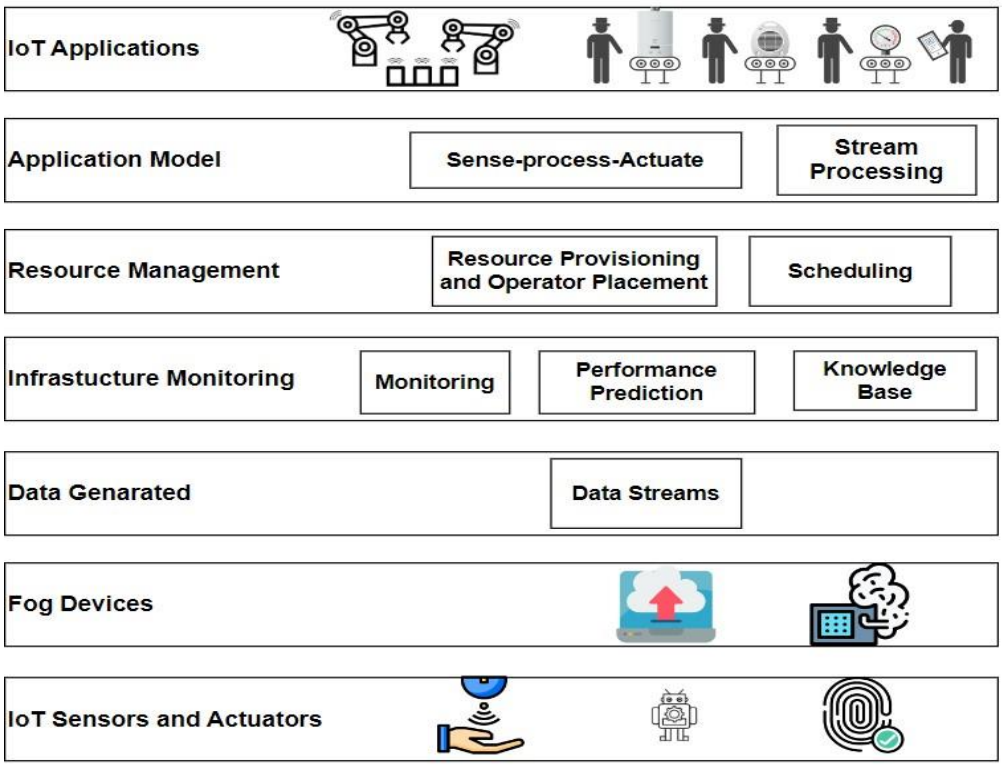

Figure 1.7: IOT & FOG Computing (Gupta, 2016)

In SIoT, devices are considered social entities, and relationships between devices are established based on trust, reputation, and user preferences. This social aspect enables devices to interact and collaborate in a more intelligent and context-aware manner. SIoT offers several benefits. It enables efficient device discovery, where devices can be easily found and connected based on their social relationships. It promotes information sharing and collaborative decision-making among devices, leading to improved efficiency and productivity. Additionally, it enhances user experience by providing personalized and socially influenced services. However, SIoT also presents challenges such as security and privacy concerns, managing complex social networks of devices, and developing appropriate social networking models for IoT environments. Overall, the integration of social aspects into the IoT ecosystem through SIoT has the potential to revolutionize the way devices interact, collaborate, and share information, paving the way for more intelligent and socially aware IoT applications in Figure 1.7.

Social IoT bridges the gap between the physical and digital realms by connecting IoT devices with social interactions. It transforms data sharing, fosters collaborative decision-making, and enriches user experiences. By integrating technology with human connections, Social IoT has the potential to drive innovation, improve communication, and create more personalized and interconnected digital ecosystems.

1.7 Implication of Fog Computing

Fog Computing is a promising paradigm that complements cloud computing by extending computing and storage capabilities to the network edge. This methodology focuses on leveraging fog computing to enhance the infrastructure of a smart city. Smart cities are urban environments that integrate information and communication technologies to optimize the efficiency of various systems, such as transportation, energy, waste management, and public services. By utilizing advanced technologies and data analytics, smart cities aim to improve the quality of life for citizens, enhance sustainability, and enable better resource management.

The smart city concept opens up a new area to explore and It also brings new challenges to implement and design it as a sustainable solution. The smart city has great potential for economic growth and lifting the quality of life in cities. As increasing numbers of citizens migrate to cities, the demand for services and resources continues to increase. The World Bank predicts that over the next two decades, India’s urban population will more than double to 33 % of the total population. The emerging IoT introduces many challenges that cannot be handled by today’s cloud computing. In this research work, we deal with the IoT Environment features like low latency, high distribution, large-scale sensor network, mobility support, and device heterogeneity. This proposed SMART FOG system allows us to create a collaborative environment for IoT networks. In the proposed system, we are going to implement a SMART FOG protocol-based technique which will allow Fog nodes to share computing and storage power to IoT devices that have low computational power within IoT network. The proposed system will be able to schedule the tasks assigned to fog node for easy processing and efficient resource management. The proposed work is focused on creating a resilient environment using SMART FOG which will create trust in fog computing. As fog computing is in its infancy, there are still many open challenges present. The SMART FOG will create

trust between fog clients and fog environment by providing fault-tolerant and secure techniques for fog computing. This research will identify some of these challenges and try to find the solution in proposed system.

Fog computing has attracted by huge number of researchers, so it is a trending topic for research. The literature study motivates research in Fog Computing by introducing a bright future and application of it. The researchers stated that Fog Computing will the how today's IoT and cloud computing are working. The researchers also stated the challenges to be faced in the implementation of Fog Computing in real-life applications. Currently, researchers are working on the implementation of fog for commercial applications. The challenge for further studies and solutions from experts is that we need to keep ourselves updated for online publications and updates from OpenFog consortium related to Fog Computing.

According to author Sheikh (2023), Fog Computing's dynamic nature demands innovative solutions for effective task scheduling. Integrating K-Means clustering with fuzzy logic, addresses Fog's resource constraints, offering adaptability in task allocation. Leveraging machine learning, our methodology optimizes execution time, response time, and network usage by intelligently assigning tasks to Fog nodes.

1.8 Fog Computing Task Scheduling

Fog computing task scheduling refers to the process of efficiently allocating computational tasks to fog nodes in a fog computing environment. It plays a crucial role in optimizing resource utilization, reducing latency, and improving overall system performance. Task scheduling in fog computing involves determining which tasks should be executed, where they should be executed, and when they should be executed. This decision-making process takes into account various factors such as the computational requirements of tasks, the availability and capabilities of fog nodes, network conditions, and user requirements. Efficient fog task scheduling involves several considerations. These include load balancing, where tasks are evenly distributed among fog nodes to avoid overloading or underutilization of resources.

Proximity-aware scheduling considers the geographic proximity of fog nodes to edge devices to minimize communication delays and improve real-time processing capabilities. Furthermore, energy-aware task scheduling focuses on optimizing energy

consumption by intelligently allocating tasks to energy-efficient fog nodes or selectively activating certain nodes based on workload requirements.

Deadline-aware scheduling ensures that tasks with time constraints are scheduled promptly, meeting their deadlines. Various scheduling algorithms and techniques are employed in fog computing, such as heuristic-based algorithms, optimization algorithms, and machine learning-based approaches. These algorithms aim to balance the trade-offs between task performance, resource utilization, energy efficiency, and other system objectives.

The study by Aimal (2022) addresses the challenges posed by traditional task scheduling methods in Fog computing for latency-critical applications. By introducing the "Critical Task First Scheduler", which prioritizes tasks based on their nature, particularly focusing on critical tasks with larger MIPs[21] sizes, the proposed methodology aims to reduce latency, energy consumption, and network utilization. Implemented in a healthcare scenario using the Fog simulator, the Critical task First Scheduler, scheduler demonstrates superior performance compared to First Come First Served, Shortest Job First, and cloud-only approaches. Simulation results highlight the efficacy of the Critical task First Scheduler approach in enhancing latency, energy efficiency, and network utilization for critical tasks

[21] Million Instructions Per Seconds

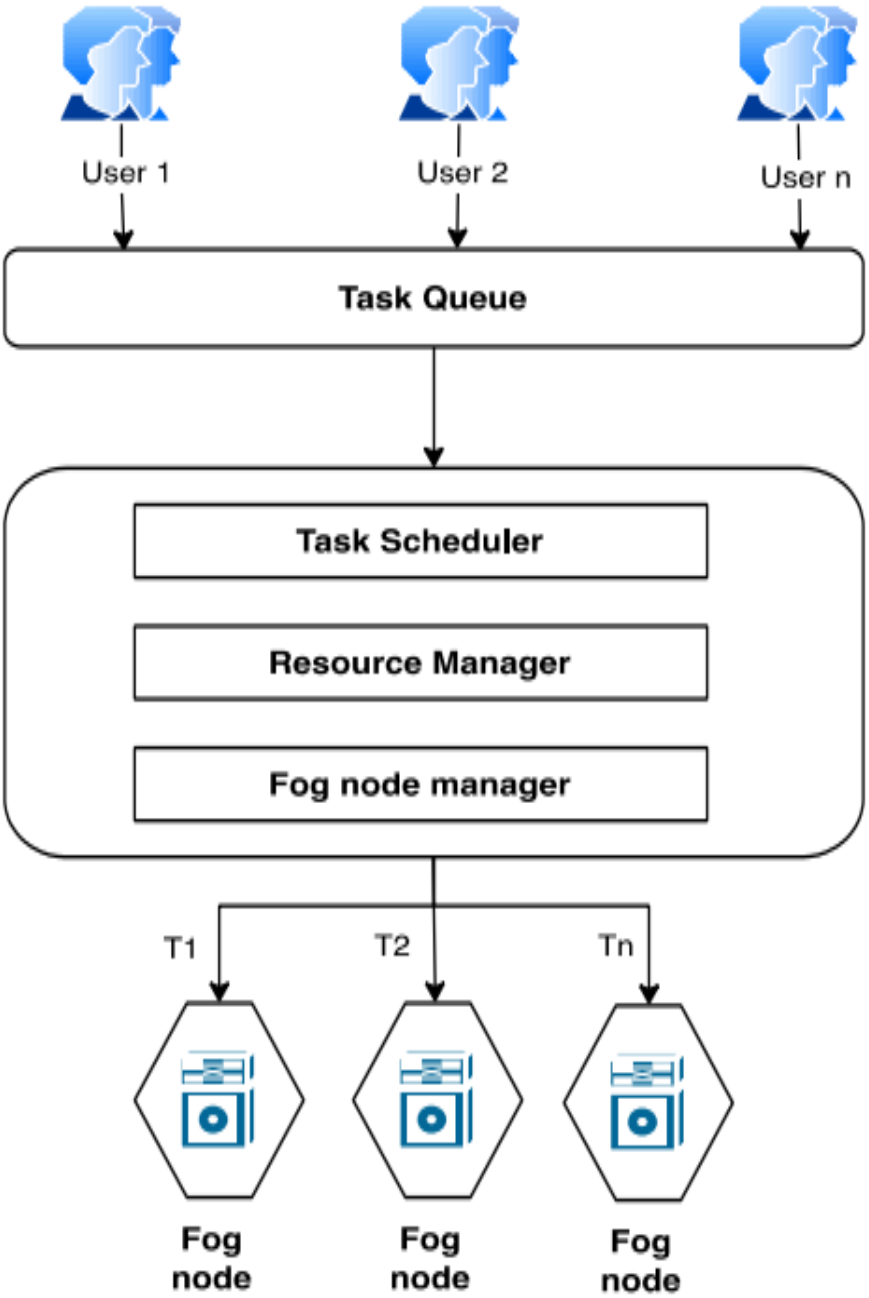

Figure 1.8: Fog Computing Task Scheduling (Alizadeh, 2020)

Figure 1.8 shows the following categories can be used to categorize task scheduling techniques in a fog computing environment are as follows:

- Static task scheduling methods
- Dynamic task scheduling methods
- Hybrid task scheduling methods

Overall, fog computing task scheduling plays a critical role in achieving efficient and effective utilization of fog resources. By carefully managing task allocation and considering various factors, fog computing systems can provide low-latency, energy-efficient, and responsive services to edge devices, enabling a wide range of applications in areas such as IoT, real-time analytics, and edge computing.

1.8.1 Static Scheduling Strategy

The task requirements must be available to the task scheduler before the initial cutting scheduling strategy for static task scheduling approaches to work. Before beginning any scheduling procedure, the task scheduler calculates the needs for each job. In this case, the tasks are sent to the system without regard to the availability of computing resources or the statuses of those resources. The First Come First Serve scheduling approach and the round-robin method are the two most popular task-scheduling techniques in this category. There is different static scheduling strategies as follows:

First Come First Served Method

The First Come First Serve CPU scheduling algorithm processes jobs in the order that they arrive in the ready queue. Newly arrived processes are added to the tail of the FIFO queue. The first process in the queue is scheduled first and removed from the queue.

Max Min Method

Performs a linear transformation on the original data. This technique gets all the scaled data in the range (0,1). The formula to achieve this is that Min-max normalization preserves the relationships among the original data values.

Minimum Completion Time Method

This algorithm locates the task with minimum execution time and allocates the task to the resource on a first come first served basis. Severe load imbalance is the major issue in this algorithm. It does not consider the resource availability and its load.

Opportunistic Load Balancing Method

Is a static load balancing algorithm. OLB keeps all nodes busy, so don't think about previous loads. However, OLB[22] does not consider the execution time of the task on this node.

Round Robin Method

reduce a multi-class problem to multiple two-class problems by learning one classifier for each pair of classes, using only training examples for these two classes, and ignoring all others

Figure 1.9 shows static task scheduling approaches, the task requirements must be known to the task scheduler before initiating any scheduling strategy. The scheduler calculates the resource needs for each job before the scheduling process begins. Consequently, tasks are sent to the system without considering the availability or status of computing resources. Within this category, several task-scheduling techniques are commonly employed.

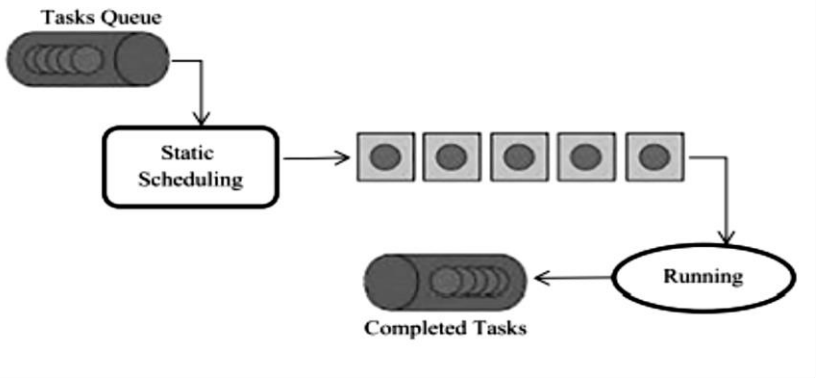

Figure 1.9: Static Scheduling Strategy (Alizadeh, 2020)

One such method is the First Come First Served approach, where tasks are executed in the order they arrive, with no consideration for their resource requirements or priorities. Another technique is the round-robin method, which assigns each task a fixed time slice for execution in a cyclic manner, regardless of their resource needs. Additionally, there are other methods like the Max-Min Method, which allocates resources to tasks based on maximum possible resource utilization, and the Min-Min Method, which focuses on minimizing the completion time of the smallest tasks. Furthermore, the Minimum Completion Time Method prioritizes tasks with the

[22] Opportunistic load balancing

shortest expected completion times, and the Opportunistic Load Balancing Method dynamically allocates resources based on real-time availability and task demands. Each method has its advantages and limitations, and the choice of a specific task scheduling technique depends on the nature of the tasks, the system's resource capabilities, and the desired performance objectives.

1.8.2 Dynamic Task Scheduling Methods

Based on the task's arrival at a specific time and the status of the system machine, dynamic scheduling methods are created. These techniques might take into account a single task at a time or several tasks at once.

Cross Entropy

Cross-entropy, also known as logarithmic loss or log loss, is a popular loss function used in machine learning to measure the performance of a classification model. Namely, it measures the difference between the discovered probability distribution of a classification model and the predicted values. As per-word cross-entropy is the average number of bits required per word, which has the advantage that you can interpret it without knowing. randomly. Perplexity is closely related to per-word cross-entropy; it just undoes the log. One advantage is that you can interpret it without knowing the base of the log.

Genetic Algorithm

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that are based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. In computer science and operations research, a genetic algorithm GA is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms EA. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover, and selection. Some examples of GA applications include optimizing decision trees for better performance, solving sudoku puzzles, hyperparameter optimization, causal inference, etc.

Immune Algorithm

The immune algorithm is a new optimization algorithm imitating the immune system to solve the multimodal function optimization problem. This paper offers a newly modified immune algorithm based on several former immune algorithms and shows its ability to solve the multimodal function optimization problem. A digital immune system is a software development practice for safeguarding applications and services from software bugs and security flaws.

Particle Swarm Optimization

An iterative optimization technique that was inspired by the behavior of social animals such as birds or fish. It involves a group of particles, or agents, that move through a search space and try to find the optimal solution to a given problem. In computational science, particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution concerning a given measure of quality.

Ant Colony Optimization

In computer science and operations research, the ant colony optimization algorithm is a probabilistic technique for solving computational problems that can be reduced to finding good paths through graphs. Artificial ants stand for multi-agent methods inspired by the behavior of real ants. The pheromone-based communication of biological ants is often the predominant paradigm used. Combinations of artificial ants and local search algorithms have become a method of choice for numerous optimization tasks involving some sort of graph.

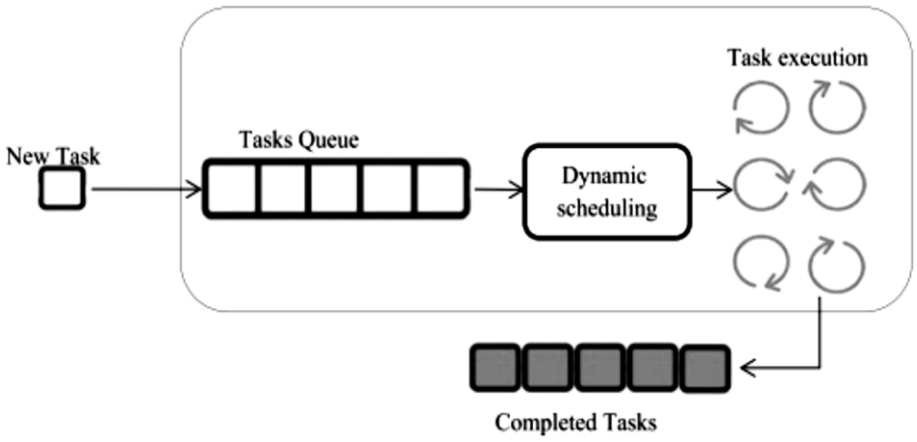

Figure 1.10: Dynamic Task Scheduling (Alizadeh, 2020)

According to Figure 1.10, the dynamic scheduling approach reduces computing costs and long-term service latency. It utilized both a double deep Q learning-based task scheduling method and the reinforcement learning technique. The allocation of user tasks to virtual machines has previously been studied through studies that took into account the propagation, waiting, transmission, and execution delays of various activities. The experimental findings supported the methodology as superior to the existing algorithms.

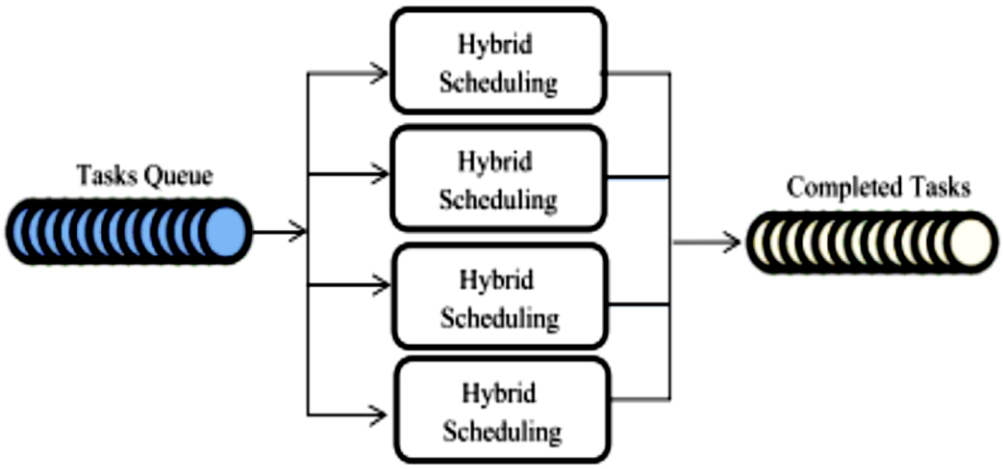

Figure 1.11: Hybrid Task Scheduling Methods (Wang, 2019)

The combination of both Static Scheduling Strategy and Dynamic task scheduling is shown in Figure 1.11. Static scheduling is a strategy where tasks are assigned to resources before program execution and remain fixed during runtime, while dynamic task scheduling involves assigning tasks based on real-time conditions and workload variations. Static scheduling strategies, such as round-robin or block scheduling, provide predictable execution patterns but may not adapt well to dynamic changes. On the other hand, dynamic task scheduling strategies, like work-stealing or task prioritization, dynamically adjust task assignments to optimize performance, considering factors like load balancing and task dependencies. Dynamic scheduling offers flexibility but introduces overhead due to runtime decisions and coordination.

1.8.3 Hybrid Task Scheduling Methods

Hybrid task scheduling methods in fog computing combine multiple approaches or techniques to optimize task allocation and resource utilization. These methods leverage the strengths of different scheduling strategies to address the unique challenges and requirements of fog computing environments.

One common approach is to combine centralized and decentralized scheduling techniques. In centralized scheduling, a central controller or orchestrator is responsible for making task allocation decisions based on a global view of the system. Decentralized scheduling, on the other hand, distributes the task allocation decision-making process among fog nodes themselves. Hybrid methods may use a combination of both approaches, with the central controller handling high-level task allocation decisions and individual fog nodes making local decisions based on their local knowledge and resources.

Another hybrid approach is to combine static and dynamic scheduling. Static scheduling involves pre-determining task assignments based on static parameters such as task characteristics and node capabilities. Dynamic scheduling, on the other hand, adjusts task assignments in real-time based on changing system conditions and workload demands. Hybrid methods can utilize static scheduling for long-term task allocation planning while incorporating dynamic scheduling to adapt to dynamic changes in the system.

Furthermore, hybrid methods may integrate heuristic algorithms with optimization techniques. Heuristic algorithms provide fast and approximate solutions by utilizing predefined rules or guidelines. Optimization techniques, such as genetic algorithms or particle swarm optimization, aim to find optimal solutions by exploring the search space. Hybrid methods leverage the speed and simplicity of heuristics for initial task allocation and use optimization techniques to refine and improve the initial solutions. Hybrid task scheduling methods in fog computing are designed to strike a balance between efficiency, scalability, adaptability, and system performance. By combining different scheduling approaches, these methods can effectively handle the complexity and variability of fog computing environments, leading to optimized task allocation, reduced latency, improved resource utilization, and enhanced overall system performance.

1.9 Fog Computing Challenges

Fog computing faces several challenges that need to be addressed for its successful implementation and operation. Overcoming the challenges requires collaborative efforts from researchers, industry stakeholders, and standardization bodies to innovate and develop solutions that maximize resource utilization, enhance security, promote interoperability, scale the system, reduce energy consumption, and optimize task allocation. By addressing these challenges, fog computing can realize its potential in enabling efficient and reliable edge computing solutions for various applications. These challenges include:

1.9.1 Drones

Drones can be used in ITS[23] applications not just as dumb sensors but also as smart fog nodes, with external devices like the Raspberry Pi, Intel Edison, and ROCK64 placed on the top of the drone to aid in traffic monitoring by seeing and locating errant cars. Similar to this, a drone can serve as a flying policeman in tele-surveillance applications, able to identify and apprehend criminals. Therefore, further research must be done to determine how drones can be used in a fog computing architecture.

1.9.2 Machine learning

Applications like ITS, healthcare, and tele-surveillance require real-time data processing and speedy replies, which might be given by implementing machine learning in fog nodes. To make judgments based on the information gathered from the sensors, the fog nodes must be intelligent enough. We are proposing a more robust approach that integrates drones with machine learning and extends it to capture the misbehaving cars and the driver's face identification. An earlier study recommended utilizing machine learning in fog nodes to anticipate busy locations.

1.9.3 Security and Privacy

The sensors' limited resources prevent a large computation cryptography approach from being used. The current priority is to secure the system and duty of the fog nodes to prevent the spread of clouds with harmful packets. Using the Diffie-Hellman problem for cryptography regarding the use of hash collision cryptography for traffic security ITS applications that use a light system and fog devices signal light.

[23]Intelligent Transportation System

Additionally, fog nodes must confirm the queries made by IoT devices when the devices themselves must confirm the security of the fog node.

1.9.4 Autonomic Fog Management and Connectivity

To meet the real-time processing needs of ITS, tele-surveillance, and healthcare applications, fog devices must be able to control themselves independently. Additionally, it poses a problem to maintain a smooth connection between all deployed devices in the fog computing architecture because of the expectation that they would be diverse.

Fog computing faces several challenges that require collaborative efforts to address and overcome. These challenges include the integration of drones as smart fog nodes for applications such as traffic monitoring and tele-surveillance, the implementation of machine learning in fog nodes for real-time data processing, ensuring security and privacy despite limited sensor resources, and achieving autonomic fog management and connectivity for efficient and smooth operations. Overcoming these challenges is crucial for realizing the potential of fog computing in enabling efficient edge computing solutions for various domains. Further research, innovation, and standardization efforts are necessary to tackle these challenges and unlock the full capabilities of fog computing.

1.10 Machine Learning Algorithms

To avoid imprecise or erroneous predictions, the data collected / generated must go through pre-processing, merging, modifying, and learning. the computational intensity and speed of a specific technique are two significant characteristics to consider while employing ML. techniques. The best algorithm is chosen based on the user application and should be fast enough to track changes in the input data and provide the desired output in a reasonable amount of time. ML algorithms create a mathematical model using sample data, known as "training data," on which to make predictions or choices. The training phase of supervised ML classifier development involves training a specific classifier from a set of labeled data. As the size of the training data increases, so do the classifiers. Some of the most popular ML algorithms are detailed further below.

1.10.1 Naive Bayes

Based on Bayes' theorem, a naive Bayes classifier is a probabilistic classifier that works by assuming that no pair of features are dependent. Naive Bayes is a simple but powerful machine learning algorithm based on Bayes' theorem and the assumption of independence between features. Despite its simplicity, Naive Bayes is often effective and computationally efficient, so it is often used in a variety of classification tasks. It is particularly suitable for text classification and spam filtering.

1.10.2 Logistic Regression

Logistic regression is a machine learning algorithm commonly used for binary classification tasks, where the goal is to predict whether an instance belongs to one of two classes. Despite its name, logistic regression is more of a classification algorithm than a regression algorithm. Logistic regression is a fundamental machine learning algorithm that is widely used in various applications such as medical diagnostics, spam detection, and credit scoring due to its simplicity, interpretability, and effectiveness. Although it is designed for binary classification, it can be extended to handle multiple classes through techniques such as one-vs-rest regression and softmax regression.

1.10.3 Sequential Minimal Optimization

SMO is a machine learning algorithm designed to train SVMs in supervised learning. SVM is used for classification and regression tasks, and SMO is a specific algorithm used to efficiently solve the optimization problems associated with training these models. Although SMO is an important algorithm for SVM training, there are alternative approaches and optimizations to solve SVM problems, such as the widely used libsvm library that implements more general optimization techniques. Still, understanding SMO provides insight into the support vector machine training process.

1.10.4 Instance-Based Learner

IBk is a machine learning algorithm used for classification and regression tasks. It is part of the family of k-NN[1] algorithms, where the prediction of a new instance is based on the majority class for classification or mean for regression of the k-nearest neighbors in a function space. The main feature of the IBk algorithm is Instance-based learning. This means that no explicit model is created during training. Instead, save the training instance and use it to make predictions for new instances. In K-NN Predictions for new instances are determined by examining the class labels for

classification or values for regression of the k-nearest neighbors in the training data set. Small values of k give the model that is more flexible and sensitive to noise, and large values of k gives the model that is smoother and less sensitive. Regression uses the average of the k nearest neighbor target values as the prediction. IBk can be computationally expensive, especially for large datasets, as it must calculate the distance for each prediction. It is often more efficient when the dataset is small. IBk performance can be sensitive to feature scaling. Therefore, it is often recommended to normalize or standardize features to obtain a similar scale. IBk is a simple but effective algorithm, especially in situations where the decision boundary is complex and not easily captured by parametric models. It is widely used in various fields such as pattern recognition, classification, and regression.

1.10.5 K-Star

K Star was developed in 2009. K Star was originally implemented as part of DiPro toolset for generating counterexamples in probabilistic model checking. K. Star A directed search algorithm also called K. It Finds the k shortest paths between the given pair of vertices in the given directed weighted graph. K Star works on the fly. This means that the graph does not have to be made explicitly available and stored in main memory. K Star can be also be controlled using a heuristic function.

1.10.6 Multi Class Classifier

A multiclass classifier is a type of machine-learning algorithm that can assign instances to one of three or more classes. Unlike binary classifiers, which distinguish between two classes such as positive or negative, multiclass classifiers handle scenarios where there are multiple possible classes. Some of the common Multi Class algorithms are Support vector machine, Random Forest, K Nearest Neighbours, Neural Networks and Decision Trees. The choice of algorithm often depends on factors such as the size and type of the dataset, computational efficiency, and the desired interpretability of the model.

1.10.7 Random Forest

A decision tree-based supervised machine learning approach called RF depends on values from a random vector that is sampled separately and with the same distribution

across all of the trees in a forest. By averaging the results, this ensemble method lowers over-fitting and bias-related error, leading to superior outcomes. Random Forest is a powerful and versatile machine learning algorithm that belongs to the ensemble learning category.

Ensemble learning combines the predictions of multiple models to create a more robust and accurate model. Random forests are particularly effective for both classification and regression tasks. The main features and characteristics of the Random Forest algorithm are: Ensemble of Decision Trees: Random Forest is an ensemble of Decision Trees. A decision tree is a discrete model that makes predictions based on a series of hierarchical decisions. Random forests create multiple decision trees and combine their predictions during the training phase. During the training process. Random Forest randomly selects a subset of the training data (with permutations) to train each decision tree. This process is called bootstrapping. Additionally, at each decision point in the tree, a random subset of features is also considered. Random Forest uses a technique called bagging, where each decision tree is trained independently on a different subset of the data. The final prediction is determined by aggregating the predictions of all trees. By training multiple decision trees on different subsets of data and features, random forests become more robust and less prone to overfitting compared to a single decision tree. Overfitting occurs when a model learns the training data well enough but is unable to generalize to new, unseen data. Random Forest provides a measure of feature importance.

Analysing the contribution of each feature across multiple trees can help determine which features have the greatest impact on predictions. The training of individual decision trees in a random forest can be performed in parallel, resulting in a scalable algorithm that can efficiently process large amounts of data. Random forests tend to be less sensitive to outliers in a dataset. Because each tree is trained on a subset of the data, the impact of outliers is reduced. Random Forest has been implemented in various machine learning libraries such as Scikit-Learn in Python, making it highly accessible and widely used.

1.10.8 Random Tree

Random Tree is a term often associated with two different machine learning algorithms, Random Forest and Highly Randomized Trees Extra Trees. Both

algorithms fall into the category of ensemble learning and are used for classification and regression tasks. Both Random Forest and Extra Trees are powerful algorithms that leverage the concept of ensemble learning to improve predictive performance. They are widely used in various applications such as classification, regression, and feature importance analysis. The choice between random forests and extra trees may depend on the specific properties of your data and the desired trade-offs between computational efficiency and model accuracy.

1.10.9 Multi-Layer Perceptron

It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network, we combine neurons so that the outputs of some neurons are inputs of other neurons. A multi-layer perceptron has one input layer and for each input, there is one neuron (or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes.

1.10.10 k-Nearest Neighbors

The k-nearest neighbor algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today. While the KNN algorithm can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

1.10.11 Supervised

Giving training data that has previously been "known" or "labeled" with the proper response and consists of N input-output pairs (X,Y) is how supervised learning functions. The ANN then generates an output 2 for each unknown X, which is then compared against Y using an error cost or distance function. Finally, an iterative process is used to minimize this mistake. Image Classification: Training with image/label datasets are examples of supervised learning methods. A new image is then presented later with the hope that the computer will pick up on the new object.

Regression: Giving the system marked historical data so it can forecast the future result of an identical circumstance.

1.10.12 Unsupervised

Using unsupervised learning methods, it self-organizes and finds hidden patterns in unlabeled input data to create neural networks. It can analyse data without sending an error signal so that the potential fix can be assessed. Unsupervised learning can occasionally be useful since it allows the algorithm to search the past for patterns that weren't previously taken into account. Unsupervised learning is necessary because manually inspecting huge datasets like those for speech recognition is highly expensive. Clustering is a very basic but well-known example of unsupervised learning.

1.10.13 Semi-Supervised

This category is a hybrid of the previous two. The algorithm is trained on a dataset that contains both labeled and unlabeled data. It works by taking enormous amounts of input data and labeling only a subset of it as training data. Reinforcement learning, a related strategy, provides feedback to guide the computer program in interacting with a dynamic environment. In this approach, a model is deployed using a small set of labeled samples and a larger set of unlabeled samples. The goal is to use labeled data to make predictions about unlabeled data and use the additional information to improve model performance.

1.11 Fog Computing Real-Time Applications

Fog computing offers significant advantages in real-time applications. It is often utilized in IoT applications that need real-time data. It functions as a more advanced kind of cloud computing. It serves as a conduit between end users and the cloud. It may be utilized in both scenarios—between humans and machines or between machines and machines.

1.11.1 Mobile Big Data Analytics

Data acquired by IoT devices is gathered in large quantities, making cloud storage ineffective. Fog computing, which uses nodes that are considerably closer to end systems than cloud computing, is advantageous in such circumstances. It also gets rid of additional issues like delays, traffic, processing speed, delivery time, response time,

data processing, data storage, and data transportation. IoT applications of the future may use fog computing.

1.11.2 Dams Safety

Dam sensors transmit data to the cloud, where it is examined and if there are any anomalies then officials are notified the issue here is the potentially deadly information delay. Fog is utilized to address this, and because it is located close to the end systems, it is simpler to send data, evaluate it, and provide immediate response. In dam monitoring scenarios, sensors play a vital role in collecting data related to dam conditions, such as water level, pressure, temperature, and structural integrity. Traditionally, this data was transmitted directly to the cloud for analysis and decision-making. To address this challenge, fog-based architecture, also known as edge computing, is employed. Fog nodes, placed near the dam sensors, act as local processing hubs. These fog nodes receive the data from the sensors and perform real-time analysis and anomaly detection locally. By doing so, they significantly reduce the data transmission time to the cloud and enable swift evaluation of dam conditions.

1.11.3 Smart Utility Service

Here, saving time, money, and energy is the major goal. Data analysis must be conducted every minute on current data. Since end users are primarily involved, cloud computing may not be useful. These programmers daily notify users of which appliances utilize the least amount of energy. Fog is an excellent option since IoT generates a lot of network traffic that makes it difficult to transfer other data.

1.11.4 Health Data

When information needs to be shared between hospitals, strict security, and data integrity are requirements. Fog may be used to achieve this because the data is conveyed locally. The laboratories may utilize these fog nodes to update the patient's lab information, which the adjacent hospitals can simply access. Since any clinician may access this unified information, patients do not need to carry hard copies of their medical histories or health concerns.

1.11.5 Smart Cities

The idea of a "smart city" has generated a great deal of attention in recent years because it promises to improve the quality of life. An urban setting known as a "Smart

City" is one in which several sectors work together to produce sustainable outcomes by analyzing real-time data. Building smart cities presents the problem of assuring accuracy and speed in reaction times when assessing the condition of infrastructure components like gas and oil pipelines, subways, and roadways. Additionally, the enormous amount of data the sensors produce creates problems with big data processing.

1.11.6 Tele-Surveillance

The concept of placing fog nodes next to CCTV[24] cameras at shopping malls and railway stations to get data from them to identify hazards like trespassing in security zones and gunshots. A video content management system is employed in the fog nodes to process and store the footage for the threat detection process.

Fog computing offers numerous benefits for real-time applications, particularly in the context of the IoT. It serves as an advanced form of cloud computing, acting as a bridge between end users and the cloud. Fog computing finds relevance in various scenarios, including human-machine and machine-machine interactions. Some notable real-time applications of fog computing include mobile big data analytics, ensuring dam safety through immediate data analysis and response, smart utility services for efficient energy consumption, secure health data exchange between hospitals, the development of smart cities for sustainable outcomes, and tele-surveillance systems for threat detection. Fog computing provides advantages such as reduced delays, improved processing speed, enhanced data storage and transportation, and localized data communication, making it a valuable solution in these real-time scenarios.

[24]Closed-Circuit Television