Chapter 1

Fog Computing in 5G Networks: An Application Perspective

Harshit Gupta[1], Sandip Chakraborty[1], Soumya K. Ghosh[1], and Rajkumar Buyya[2]

[1]Department of Computer Science and Engineering, IIT Kharagpur, India
[2]CLOUDS Laboratory, University of Melbourne, Australia

Abstract: Fifth generation (5G) cellular network promises to offer to its users sub-millisecond latency and 1 gigabit per second transmission speed. However, the current cloud based computation and data delivery model do not allow these quality of service (QoS) guarantees to be efficiently harnessed, due to the number of hops of wired networks between the 5G base stations and the cloud, that leads to a significant increase in latency. Forwarding all the data generated by devices directly to the cloud may devour the bandwidth and lead to congestion. Therefore, it is necessary that processing be hosted near the devices, close to the source of the data, so that the high speed transmission of 5G can be utilized and data can be processed and filtered out by the time it reaches the cloud. This bringing down of computation, storage and networking services to the network edge opens up many new research areas of applying Fog computing over cellular network architecture. This chapter discusses the advantages of extending the cloud services to the edge by presenting use-cases which can be realized by fog computing over 5G networks.

1.1 Introduction : Overview of Fog computing

The Internet has been evolving from the time it was conceived, and is now going beyond traditional desktop computers. The proliferation of the Internet of Things has brought about a transformation in the way the world interacts on the Internet. The World Wide Web connected computers together, smartphones brought humans into the fold of the Internet, and now IoT is poised to connect devices, people, environments, virtual objects and machines in ways that the world has never known. IoT deployments like smart cities, smart homes and the like - things that were more of fiction - are now becoming a reality, and are projected to affect as many aspects of human life as possible.

Internet of Things

- The number of "things" connected to the Internet surpassed people in 2008. By 2020, the population of Internet-connected things will reach 50 billion, garnering profit and cost savings worth $19 trillion over the next decade.
- General Electric predicts that amalgamation of machines, data, and analytics will become a global industry worth $200 billion in a period of 3 years.
- A whopping 94% of all businesses have seen a return on their IoT investments.

Typical IoT systems consist of a myriad of devices, ranging from sensors embedded in roads to mobile devices like cars and trains. With such a large number of *things* involved in an IoT deployment, the number of devices connected to the Internet is growing by leaps and bounds. Currently the number of endpoints (typically smart phones and laptops) has been estimated to be around 3-4 billion and is expected to grow to a trillion in a few years. Such a lot of devices will generate gigantic volumes of data, in a phenomenon that has been attributed the term *data tsunami*. The applications and the network infrastructure will have to adapt accordingly to such a massive increase in the amount of data they will have to handle given the constraint of the amount of bandwidth available.

The IoT brings a Data Tsunami

Development in IoT has brought about the proliferation of cheap, distributed sensors resulting in a huge volume of data in a short amount of time. Virgin Atlantic's new fleet of highly connected planes is expected to create over half a terabyte of data per flight. According to Cisco Systems most recent Visual Networking Index (VNI), mobile data traffic will grow 10-fold globally between 2014 and 2019, reaching 24.3 exabytes per month worldwide in 2019.

Development in engineering has always aimed at designing systems that can function with as low human intervention as possible. The Internet-of-Things is a perfect platform for designing such applications, particularly because connecting every device to the Internet gives every device the power to make decisions on it's own, thus reducing the need of human intervention. Research on such autonomous systems has revealed that they heavily rely on low response time of the application. IoT systems like smart grids, collaborative object detection, etc. require latency of sub-millisecond order - requirements that the Internet will have to provide for the application to work in the desired manner, failing to do which may defeat the entire purpose of the application.

This change in the nature of devices connected to the Internet and the concomitant increase in the amount of data generated demands an evolution in the network infrastructure as well. The present cloud-model of execution will prove to be inefficient, if at all feasible, for the futuristic applications that development in IoT brings to vision.

1.1.1 Limitations of the current computation paradigm

The current computation paradigm has cloud datacenters as the only point for execution after the basic processing available at the devices. However, such a large number of IoT devices continuously sending data to the cloud for analysis would lead to scalability issues in the core network. Levels of congestion in the backbone network will increase manifold and may lead to aggravated packet loss and delay, spoiling the user experience. Furthermore, sending lots of data to the cloud for processing may lead to the cloud becoming a bottleneck, again leading to increase in response time.

A lot of IoT applications, typically those that run in industrial settings like smart grids, need their devices to react very quickly to an impulse. In such a case, sending the data related to the impulse to the cloud and then getting the response back may not be desired due to the high communication latency involved in the network in between. This latency is unavoidable due to the large number of hops that a packet has to travel through to reach the cloud. Therefore, the *do-it-on-cloud* paradigm of computation will become disruptive with the advent of latency-critical applications for IoT systems. Such a scenario poses the requirement of distributed computation, storage and networking services which are close to the source of data, or, in other words, *Fog computing*.

1.1.2 Fog computing

Fog computing[4] is a term coined by professor Salvatore J. Stolfo [26], that has recently been picked up by Cisco[2]. Fog Computing is a paradigm that extends Cloud computing and services to the edge of the network allowing applications to run in close proximity of users, be highly geo-distributed and support user mobility. Due to such characteristics, Fog cuts down latency of service requests, and improves QoS, resulting in superior user-experience. Fog Computing is a necessity for emerging Internet of Everything (IoE) applications (like industrial automation, transportation, etc.) that demand real-time/predictable latency. Owing to its wide and dense geographical distribution, the fog paradigm is well positioned for real time big data analytics. The data collection points in fog computing are densely distributed, hence adding a fourth axis - *geo-distribution* - to the often mentioned Big Data dimensions (volume, variety, velocity and veracity).

Fog Computing

Fog computing is a non-trivial extension of cloud computing - by providing compute, storage and networking services near the edge of an enterprise’s network. The peculiar characteristics of the Fog are its proximity to end-users, its dense geographical distribution, and its support for mobility.

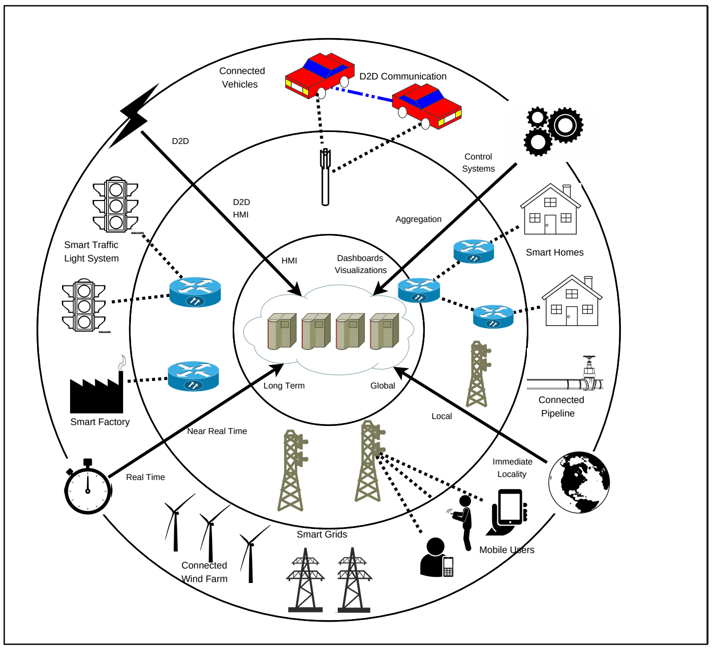

Figure 1.1: Applications supported by fog computing.

Fog provides the same services as the cloud (compute, storage and networking) and shares the same mechanisms (virtualization, multi-tenancy, etc.). These common attributes of the cloud and the fog makes it possible for developers to build applications that utilize the interplay between the fog and the cloud. According to Bonomi et al. [6], fog computing was conceived to support applications whose requirements don’t quite match the QoS guarantees provided by the cloud. Such applications include (as illustrated in Figure **??**)

- Applications having stringent latency requirements, for example mobile gaming, video conferencing, etc. Running these applications on the cloud can mar user experience due to the unreliability of QoS offered by the cloud.

- Geo-distributed applications where the data collection points are distributed over a wide area, for instance, pipeline monitoring, or sensor networks to monitor the environment.

- Fast mobile applications involving highly mobile users (smart connected vehicle, connected rail).

- Large-scale distributed control systems consisting of a vast number of sensors and actuators working in a coordinated manner to improve user experience. For example smart grid, connected rail, smart traffic light systems.

It is important to note that the Fog is not a substitute for the existing cloud computing paradigm, instead, fog is an extension to the cloud, and application built for the fog should be able to exploit both the flexibility and power of the cloud and the real-time capabilities of the fog.

1.2 Fog computing on 5G networks

Fog computing and 5G networks are two concepts having different origins but will soon converge as the promises made by the vision of 5G networks makes it necessary to bring processing down to the edge.

1.2.1 Fog computing - a requirement of 5G networks

Fifth Generation (5G) mobile networks, though not a reality at present, is expected to hit the market by 2020. Communication in 5G networks will be based on high-frequency signals - in the millimeter-wave frequency band - that can allocate more bandwidth to deliver faster, higher-quality video and multimedia content. 5G networks promise to provide millisecond and sub-millisecond latency while offering a data rate of more than 1 gigabit per second [24]. This latency is so small that it eliminates the possibility of the radio interface being the bottleneck. Next generation mobile networks are designed in a way that can handle communications not restricted to humans (where one can possibly mask the latency) - they are built to support reliable and fast machine-to-machine communication as well, a use-case that needs low latency to be effective.

For 5G to be successful, it has to support fog computing, otherwise the low latency radio interfaces will be of no avail. A typical 5G network have mobile users connecting to a base station, which would in turn be connected to the core network through wired links. Requests to a cloud-based application would go through the base station and the core network to finally reach the cloud servers. In such a deployment, even though the low latency radio interfaces enable sub-millisecond communication between the mobile device and base station, but sending the request from the base station to the cloud will lead to a delay increase in orders of magnitude.

The true value of 5G cannot be harnessed by running applications having the cloud as the only processing unit, and it is required to enable the deployment of application code at devices in close proximity to the users. [3]

It is imperative for the 5G networks to be more than just a communication infrastructure. Computation and storage services, if supplied by the network, close to the devices, will allow applications to take benefit of low latency radio to provide very fast end-to-end response time. This will highly benefit both the customers (by giving timely responses) and the provider (by alleviating the load on the backbone network). This descent of processing from the cloud to the edge forms the definition of fog computing, and it would not be wrong to say that 5G networks cannot fulfil it's promises without fog computing. Fog computing is not a feature, as most view it, but a necessary requirement for 5G networks to be able to succeed.

A key element of 5G networks that enables fog computing is small cell (pico and femto cells), also known as micro-cells. Small cells can alleviate the burden on roof-top base stations (macro-cells) by allowing end points to connect to them. A device can connect either to the macro-cell or to a micro-cell. This makes the architecture of 5G networks a hierarchical one - with the core network (cloud) at the apex, followed by macro-cell base stations and micro-cell base stations, and finally end devices. Hence, from the perspective of fog computing, both macro and micro-cell base stations form the fog nodes, i.e. networking nodes providing computation and storage as well. Packets sent uplink by the devices will be analysed at the micro-cell or macro-cell base stations before reaching the core network.

Another major advancement in communications that 5G brings along is efficient device-to-device communication. Application data sent will be sent from the sender device directly to the receiver device, with the base station handling only control information of this transfer. This allows inter-device communication to take place without burdening the base station, thus beatifying fog systems with scalability of handling numerous devices interacting with each other. This will be categorically useful for applications that involve numerous connected points and continuous communication between these points, for example smart homes.

The rest of this section discusses the network architecture of 5G networks and how they will realize fog computing. In addition to this, the architecture of fog applications is also described - a segregation of application logic into components that can harness the services provided by fog computing.

1.2.2 Physical network architecture

The physical network architecture of a fog network over 5G will extend the architecture of the state-of-the-art Heterogeneous Cloud Radio Access Networks [22]. In the traditional HCRAN architecture, all application processing tasks are performed on the cloud inside the core network, which requires billions of end devices to communicate their data to the core network. Such a massive amount of communication may vitiate the fronthaul capacity and may overburden the core network, which will have a detrimental impact on the QoS experienced by the end-users.

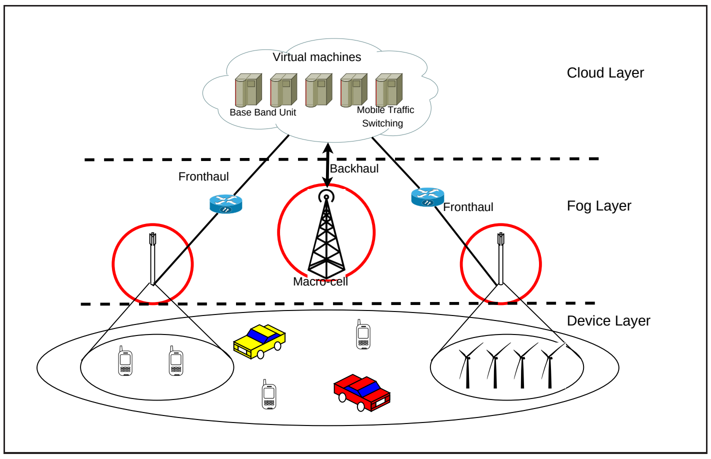

Figure 1.2: Architecture of 5G network with fog computing - a three-layered architecture.

An intuitive solution to this problem is to bring down computation and storage capabilities from the cloud near the edge, so that the need to send all the data generated by end-devices to the cloud is done way with, hence alleviating the fronthaul and the core network of the immense traffic surge. Figure **??** depicts the various locations where this offload of computation and storage can be done. The fog network architecture consists of 3 logical layers which are shown in Figure **??**. The devices in each layer are capable of hosting computation and providing storage, hence making it possible for creating complex processing offload policies.

- **Device Layer** : The device layer subsumes all the end-devices connected to the fog network. The devices include IoT devices like sensors, gateways, etc. and also mobile devices like smartphones, tablets, etc. These devices may be exchanging data directly with the network, or may be performing peer-to-peer communication among themselves. Being the source of all data entering the network and the prime actuators performing tasks, these devices are the lowest tier of fog devices. The device layer hosts computation either by embedded coding (for low-end devices like sensors) or as a software running on the operating system of the device.

- **Fog Layer** : The fog layer consists of intermediate network devices located between the end-devices in the device layer and the cloud layer. The first point of offload in this layer are the RRHs and small cells which are connected by fibre fronthaul to the core network. Processing incoming data here will considerably reduce the burden on fronthaul. Macro cells also form a point of offloading processing which send the processed data to the core network through backhaul links. Both fronthaul and backhaul is realized by Ethernet links and the intermediate devices like router and switches in the path from the radio heads to the core also form potential places where computation and storage tasks can be offloaded.

Deploying applications on these devices is made possible by advances in virtualization technology. Each application is packaged in the form of a virtual machine and is launched on the appropriate device. The application virtual machines run alongside the host OS virtual machine (which performs the original network operations) over a hypervisor on the fog device.

- **Cloud Layer** : This layer forms the apex of the hierarchical architecture, with cloud virtual machines being the computation offload points. The theoretically infinite scalability and high-end infrastructure of the cloud makes it possible to handle processing that requires intensive computation and large storage - which cannot be done at the edge devices. In addition to application layer processing, the cloud layer contains Base Band Units which process data coming from RRHs and small cells via fronthauls and route processed data to application servers.

1.2.3 Application architecture

For an application to be called fog-ready, it must be designed to harness the full potential of the fog. Typically, a fog-ready application should have the following components [6].

- **Device component** : The Device component is bound to the end devices. It performs device level operations, mostly, power management, redundancy elimination, etc. At times, when the end-device is not just a light client, it also hosts application logic demanding very low latency responses as this component is executed on the device itself. However, due to the resource constraints of the underlying device, this component should not contain heavy processing tasks.

- **Fog component** : The fog component of an application performs tasks that are critical in terms of latency and require such processing power that cannot be provided by end-devices. Furthermore, since the fog component is meant to run on fog devices close to the edge, the coverage of this component is not global. Thus this component should host logic that requires only local state information to execute.

  The fog component is not bound to a particular kind of device. It is free to reside in any kind of device between the edge (consisting of end-devices) and the cloud. The mapping of the fog components to devices depends on the points of offload in the path from the edge to the cloud. Depending on the geographical coverage and latency requirements of the application, the fog component can be hosted on any of these points of offload. In fact, placement of fog component on appropriate fog nodes forms an interesting and important area for research.

- **Cloud component** : Cloud component is bounded to the cloud servers in the core network. It contains logic for long-term analytics of the data collected from the lower layers and for operations that don't have any sort of latency constraints per se. Applications tasks requiring large processing power and storage are suitable to be placed in the cloud component, so that they can harness the infinite resources of the cloud. Moreover, since the cloud layer is located at the apex of the network, it receives information from all devices and hence has a global knowledge of the entire system. Thus, application logic requiring knowledge of the global state of the system should be placed in the cloud component of the application.

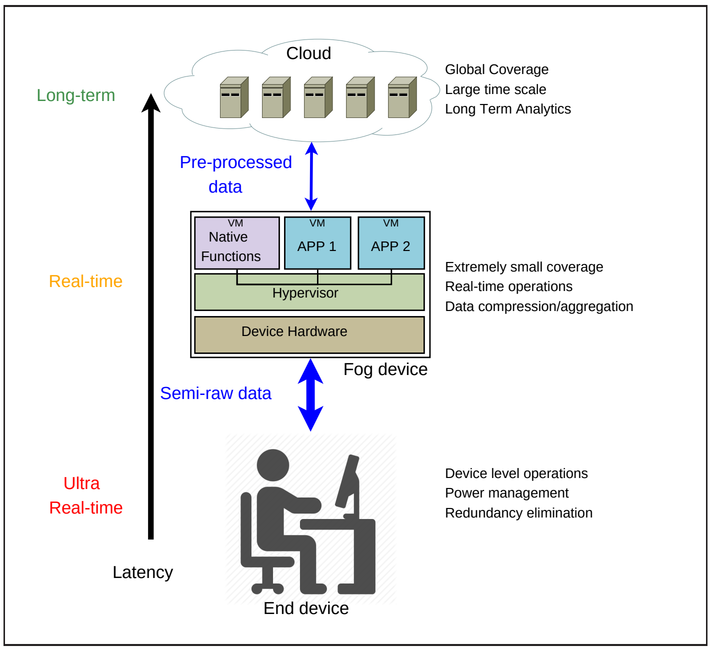

Figure 1.3: Application architecture.

Coding logic into the various layers of the a fog-ready application determines the performance of the application. Incorrect placement of logic can cripple an application and make it unable to use the benefits that fog computing has to offer. The following sections discuss several use-cases whose requirements can be satisfied by the unique quality of service provided by fog computing when deployed over a 5G cellular network. For each use-case, we also present a suitable mapping of application logic to application layers for each use-case.

1.3 Use Case 1 (Latency critical application) : Smart Traffic Light System

A Smart Traffic Light System (STLS) is a network of connected traffic lights which intelligently, and in a coordinated fashion take decisions that prevent accidents, reduce traffic congestion, minimize noise and fuel consumption, and give the drivers a better experience by long-term monitoring. The STLS is but a component of the larger vision of Smart Connected Vehicle (SCV) and Advanced Transportation Systems, but it is rich enough to drive some key requirements for fog computing.

1.3.1 Requirements

An STLS needs to take full control of the traffic in an area and perform a broad spectrum of tasks - more than what a traffic policeman would have to - right from accident prevention to flow control. The various use-cases of an STLS have been listed down as follows.

Accident Prevention

The most important concern of any automated system directly affecting humans is user safety. Given the number of traffic accidents that occur daily, accident prevention is one of the key requirements of an STLS, failing to do which can have serious repercussions involving loss of life and property. The STLS should be able to detect vehicles not following traffic rules - for instance, not stopping at a red signal - and should inform vehicles that can potentially be affected by this rogue vehicle (typically those on an orthogonal street). This information can be conveyed by communication between traffic lights on adjacent streets. The orthogonal streets can ask their vehicles too to stop for some time. Also, in the event of a pedestrian crossing a road when he/she should not, and there is a vehicle coming her way, the STLS will calculate, from factors like speed of approaching vehicle and pedestrian, whether an accident may take place and take suitable action. In addition to this, over-speeding vehicles can be asked to stop by these traffic lights in order to avoid accidents[34]. The traffic lights may also determine whether the over-speeding vehicle is an emergency vehicle, like an ambulance and accordingly decide whether to make it stop or let it go.

Re-synchronization and Flow Control

Activation of accident prevention mechanism causes the traffic light cycles in the affected area to go out of synchronization. To dampen this perturbation in traffic light cycles, few neighbouring traffic lights need to re-adjust their cycle. This task is not very critical in terms of latency, since at the most it may lead to prolonged red lights on few lanes causing vehicles to stop more than required. Moreover, this use-case is of a slightly global nature, that is, the actors involved are spread across a few streets.

Flow control is essential to ensure a smooth movement of traffic without having to make the drivers stop too often. An STLS can collect information about the level of traffic in each lane of the city from sensors and based on a routing policy route vehicles to reduce congestion. Traffic lights can coordinate and maintain a *green wave*, reducing the number of times a vehicle would have to stop at traffic signals. By doing this, the STLS can reduce noise and fuel consumption, since vehicles would not have to accelerate often. This will be especially useful for emergency vehicles like ambulances or fire engines, for which the STLS can create green waves on demand, so that they don't have to stop at traffic signals and can reach their destination as soon as possible.

Long-term Monitoring

This use-case is required for monitoring the entire Traffic Light system over a large time scale and looking at ways to enhance the performance of the system. The STLS can improve it's congestion-aware traffic routing policy continuously based on analytics on data collected over a long time. Through long-term analysis on observed pedestrian movement, the STLS would be able to decide the optimal time for which pedestrians should be allowed to cross roads. Policy makers will be able to make decisions such as whether creation of alternate routes is required with the help of long-term analysis of traffic congestion data. The main actors involved in this use-case are policy-makers that analyze the road traffic over a long time period and come up with changes to improve driver experience.

Design Requirements

The use-cases entailed by a smart traffic light system highlight the following design requirements of the application.

- **Low-latency response** : Accident prevention requires a very low response time to alert the involved person in a timely manner, failing to do which will mar the very purpose of the accident prevention mechanism. Furthermore, detecting a rogue vehicle (based on his movement) and alerting the rest of the drivers also requires a quick response, so that the chances of a mishap can be minimized.

- **Handling large volume of data** : An STLS contains a large number of sensors deployed on roads throughout the city - generating data at a high rate. Due to the large volume of data that needs to be analysed, the network should be scalable and robust enough to handle large traffic. Poor network architecture can be a victim of bandwidth over-utilization and become congested, leading to further delay in responses.

- **Heavy processing power and global coverage** : The tuning of traffic routing algorithm and analysis for policy-making requires processing a large amount of data, that too on a large time-scale, which is a computationally intensive task. Moreover, the analysis has to be done on a city level - and thus requires this analysis be done on a device with global coverage.

1.3.2 Deployment Details

It is worth noting that the requirements of an STLS showcase a variety of requirements, both in terms of response time and geographical area affected. We now discuss a model deployment of an STLS on fog infrastructure - that utilizes services provided both by the edge and the cloud.

1.3.2.1 Physical Deployment

The data collection points in an STLS are primarily sensors that are deployed on the roads, like induction loop sensors which can detect a crossing vehicle and speed detection sensors. CCTV cameras installed at intersections also fall under the data collection points of the STLS. Traffic lights are the actuators of the system, as any action performed by the STLS is reflected by a change of traffic lights. Consider a traffic intersection, having a set of traffic lights and two intersecting roads as shown in Figure **??**.

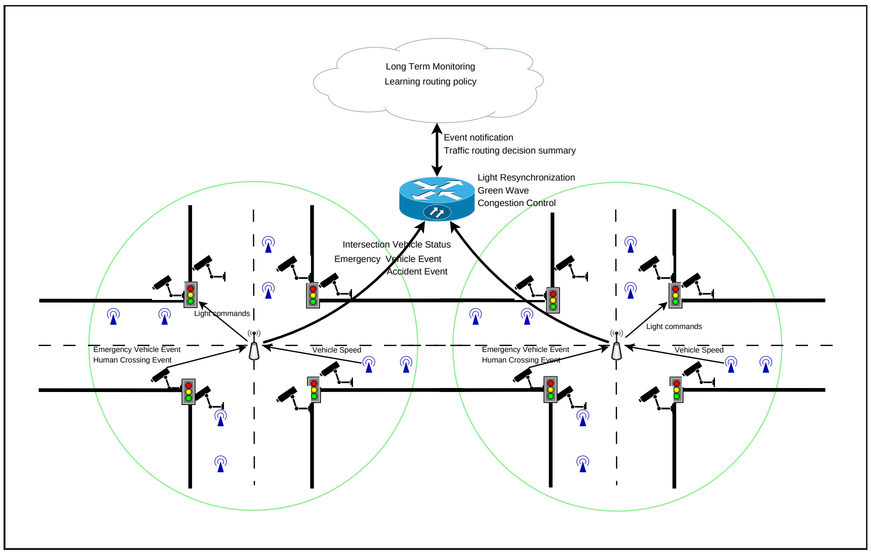

Figure 1.4: Deployment of STLS on fog infrastructure over 5G mobile network.

Each intersection will be equipped with a 5G small-cell which would connect the devices on that intersection together, allowing real-time device-to-device communication among them. The small cell is equipped with compute and storage facilities which will be utilized by the STLS application. The small cell is in turn connected by a high bandwidth connection to the cloud through intermediate network devices (typically belonging to the ISP). These network devices in the STLS deployment are also fog-enabled, meaning that they too are points for offloading application logic. These intermediate devices will be used for communicating between devices belonging to neighbouring intersections.

1.3.2.2 Application Architecture

The application logic of STLS has been broken down into components that can be mapped to the 3-layered architecture of a fog application. This partitioning of application tasks has been described below.

Device component

Since the devices in this system include only sensors, CCTV cameras and traffic lights, the device component of the application is not complicated. For sensors, they need to be able to send updates to the small cell over a 5G network. The logic running in the CCTV cameras should process the recorded video stream in real-time to detect events of interest, such as a human crossing the road or an approaching emergency vehicle, and convey this to the small cell in such a case. As concerns traffic lights, they are the only actuators in the system, as all control decisions taken by the system are ultimately realized via changes in traffic light sequences. The application component running on a traffic light should be able to receive messages from the small cell on the intersection and change light sequence accordingly.

Fog component

The fog component of the application runs on the small cell at each intersection as well as on the intermediate network devices connecting the small cells to the Internet. The application logic running in these devices handles most of the requirements of an STLS.

For accident prevention, the fog component needs to handle the event of *human-crossing-road* sent by those CCTV cameras that surveil the lane through which traffic is flowing. In case an accident is possible, the application will send a message to the traffic lights on that lane to change to red immediately so that traffic stops and also blow a horn to alert the human. This task should take place in real time, hence it is handled on small cell itself - that is the place which directly receives the human crossing event from the CCTV cameras. This proximal placement of application logic, coupled with the sub-millisecond latency of 5G transmission, allows the accident prevention mechanism to happen with a very small delay, hence minimizing the risk of a human getting injured.

For re-synchronization of traffic lights to dampen a sudden change in light sequence due to activation of the accident prevention mechanism, there has to be a communication between nearby traffic lights so that they may run a distributed algorithm and re-synchronize their light cycles. The STLS allows smooth traffic flow and maintains green waves by coordinating between multiple traffic lights and maintaining an appropriate traffic light sequence. Research works like [15], [35] and [18] have explored the possibilities of improving the traffic flow and minimizing congestion by running a distributed algorithm on multiple traffic lights based on information collected by sensors. The fog component of the STLS also receives events of an approaching emergency vehicle from the CCTV cameras, to which the system will respond by triggering the traffic light on the vehicle's path green and inform the next neighbouring intersection of the approaching emergency vehicle so that it may take necessary actions. This component requires swift communication between neighbouring traffic lights, however, the latency requirement is not as critical as the accident prevention use-case. Moreover, this use-case requires the knowledge of the state of traffic lights at more than one intersection, a coverage which is more global in nature than accident prevention. Hence, this component is hosted on the intermediate network devices connecting the small cells at intersections to the Internet. Being hosted on a device just a few hops away from the small cells, the small cells are able to send messages to each other with a very low delay and hence are able to control traffic lights in other intersections. In addition to the low delay, running this component in the fog reduces the amount of raw sensor data sent to the cloud, thus alleviating the core network of the risks of congestion and minimizing the consumption of bandwidth.

Cloud component

The cloud component receives data from the small cells about the traffic conditions and events at regular intervals. Small cells aggregate information over a period of time and send it to the cloud, which reduces the volume of data sent. The cloud component of the STLS performs long-term analysis on the incoming data, based on the results of which, experts can infer whether to create a new route for reducing load on existing roads, or whether the crossing time for pedestrians needs to change. Several studies have been conducted on such analysis of traffic data [21], [32], [33]. Through long-term analysis on congestions levels in the city, the STLS can improve the traffic routing policy that runs in the fog component to reduce traffic congestion. This analysis may require heavy processing since it needs to analyze data related to a large time-scale. Moreover, this component needs to know the traffic state on a global scale, and does not require any guarantee on response time. Due to these characteristics of the application logic in the cloud component, it is appropriate for hosting it on the cloud.

1.4 Latency critical application - Mobile Gaming

Cloud gaming, sometimes called gaming on-demand, is a new kind of gaming platform made possible by the proliferation of cloud technologies, allowing physically distant users to play together. Cloud gaming is an efficient and cost-effective way to deliver high-quality gaming experience and has opened up a lot of business opportunities. In a cloud gaming system, computer games run on powerful game servers on the cloud, while gamers interact with the games using thin clients connected to the Internet. The thin clients are light-weight applications and can be hosted on resource-constrained devices, such as mobile devices. Cloud gaming is ubiquitous, allowing gamers to play a game from anywhere and at any point of time, while the game developers can optimize their games for a particular machine configuration.

A cloud gaming system essentially renders a gaming application on cloud servers and streams the scenes of the application as a video sequence back to the player. A player of the game interacts with the game through a thin client, which is responsible for displaying the video received from the cloud server as well as sending the interactions of the player with the game to the cloud. Cloud gaming is one of those applications requiring a strict latency guarantee, failing to provide which will lead to detrimental impact on the user experience. In addition, cloud gamers are also particular about the video quality that is rendered on their light clients. Thus the implementation of a cloud gaming system needs to take resource allocation, scalability and fault tolerance into account as well apart from meeting the gamers' needs.

The traditional implementation of mobile gaming involves hosting all the computation and storage in the cloud, hence making *mobile gaming* synonymous to *cloud gaming*. However, communicating with the cloud for every request may not always be the best practice, especially when latency requirements are stringent. Choy et al. [8] have shown through a large-scale empirical study that contemporary cloud infrastructure cannot meet the stringent latency requirements necessary for acceptable game play for many end-users, thus imposing a limit on the number of potential users for an on-demand gaming service. Based on empirical results, they have concluded that augmenting the cloud infrastructure with edge-servers can significantly increase the feasibility of on-demand gaming, or cloud gaming. Hence, it makes sense to offload some computation involved in the cloud-based game to the edge. They have described three computation approaches : *cloud-only*, *edge-only*, and a *hybrid* approach in [9]. Experiments show that the percentage of users served increased from 70% in an only-cloud deployment to 90% in a hybrid-deployment that used both cloud and edge-servers.

The future of cloud games

"Lets say our industry had never done consoles or consumer clients. Even if we just started out with cloud gaming, you'd actually go in the direction of pushing intelligence out to the edge of the network, simply because its a great way of caching and saving you on network resources." - Gabe Newell, co-founder and managing director of video game development and online distribution company Valve Corporation.

These studies give ample support to the fact that fog computing is an efficient platform for deploying on-demand games, and in this section, we discuss the deployment of a cloud-game on fog infrastructure.

1.4.1 Requirements

Cloud gaming is a highly interactive application posing stringent requirements in terms of latency and video quality, failing to do which can directly affect user experience. The typical requirements of an on-demand game are discussed as follows.

Example Game Type	Perspective	Delay threshold
First person shooter (FPS)	First person	100 ms
Role playing game (RPG)	Third person	500 ms
Real-time strategy (RTS)	Omnipresent	1000 ms

Table 1.1: Delay tolerance in traditional gaming

Interaction Delay

The authors of [27] have performed a categorical analysis of state-of-the-art cloud gaming platforms, and brought out the novelty in their framework design. They have highlighted *interaction latency* and *streaming quality* as the two quality of service requirements of cloud gaming. As for the interaction latency, Table 1.1 lists out the maximum delay allowed for different types of traditional games before the user experience begins to degrade.

However, the latency requirements in cloud gaming are more stringent. Traditional online games can perform the rendering on the local machine and then update the game state in the game server in some time. Hence the player of a traditional online game does not feel the effects of interaction delay. But in case of cloud gaming, the rendering is offloaded to the cloud, thus the thin client does not have the ability to hide the interaction delay from the user. This makes cloud gaming less delay tolerant than traditional online gaming systems. The maximum interaction delay for all cloud-based games should be at most 200 ms. Other games, specifically such action-based games as First Person Shooting games, likely require less than 100 ms interaction delay so that players' quality of experience is not affected.

Video Streaming and Encoding

When a player of a cloud-based game issues a command, the command has to traverse the Internet to the game server in the cloud, be processed by the gaming logic, rendered by the processing unit, compressed by video encoder and streamed back to the player. This encoding/compression and distribution to end users has to take place in a very timely manner in order to prevent degradation of users' QoE. In addition to timeliness in encoding, the quality of the video being streamed is also an important factor in determining user-experience.

Design Requirements

- **Low-latency response** : User experience will be hampered in case of high response time, hence making low-latency response a critical requirement of mobile gaming. For guaranteeing low-response time, the infrastructure should be strong enough that the user inputs reach the game server, be processed by the game logic, and the audio/video be captured, encoded and sent in a timely manner.

- **High bandwidth** : Transferring video streams constitutes most of the data exchanged in a cloud-game. For transferring such a huge amount of data, that too in real-time requires a high bandwidth connection between the game server and client.

- **Global coverage** : To be able to support users from multiple geographical regions, the cloud-game application needs to be accessible from anywhere. Hence, it is imperative for such an application to have a global coverage.

1.4.2 Deployment Details

Bharambe et al., in [5], have presented *Colyseus*, a distributed architecture for hosting interactive multiplayer games on the internet. Colyseus distributes dynamic game-play state and computation to multiple nodes across the Internet, adhering to stringent latency constraints and maintaining communication costs at the same time.

Properties of multiplayer games

- Games can tolerate weak consistency in the game state. Present client-server implementations cut down interaction delay by presenting the player with a weakly consistent view of the game world.
- Game-play is generally driven by a rule-set that makes it easy to predict reads/writes of the shared game state. For instance, most reads and writes of a player relate to objects which are located physically close to the player.

Using Colyseus [5], the game state concerning a player can be located in a node very close to her, so that the interaction delay is minimized for a smooth gaming experience. Each game is described as a collection of game objects - where each object can be a player's avatar or the user's representation in the game (for example a car in a racing game as shown in Figure **??**). Colyseus maintains a primary copy and several replicas of each object, with each device holding primary copies of user objects that are directly connected to it, and replicas of other objects which primary objects interact with. Figure **??** elucidates the concept of primary and secondary objects. Distributed objects in Colyseus follow a single-copy consistency model, i.e., all writes to an object are serialized through exactly one node in the system - the one containing the primary copy of it. This allows low-latency reads and writes at the cost of weak consistency, since most of the communication is made to the player's own object (which is present right at the edge).

Furthermore, Colyseus utilizes the locality and predictability in the movement patterns of players to pre-fetch objects needed for driving game logic computation.

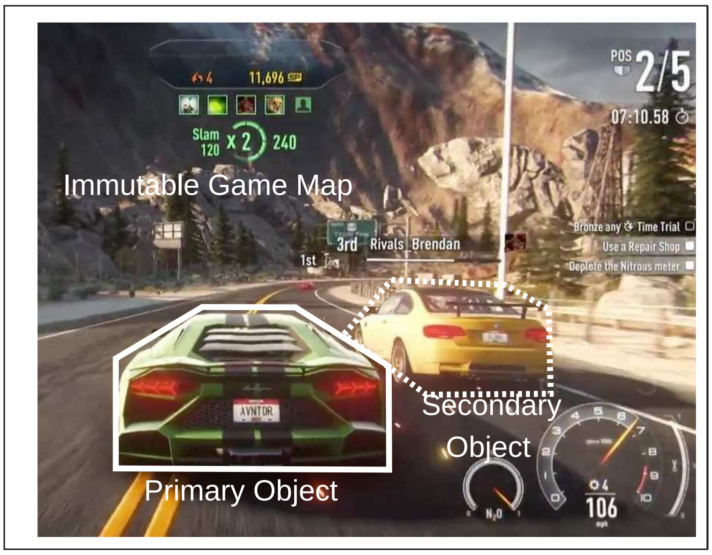

Figure 1.5: Game play showing primary and secondary objects. The game player owns the green car and interacts with the yellow car. The device closest to the owner of the green car would contain the primary copy of the green car object and a replica of the yellow car object. (Courtesy : http://www.metacritic.com)

This pre-fetching of objects hides the delay in communicating with the node containing the primary copy of required object, hence giving a smooth user experience without any lags.

For a successful implementation of a cloud-game on fog infrastructure, the game service provider will have to incorporate a Colyseus-like system that distributes game state across multiple nodes based on proximity to user. In the absence of such a mechanism, communication with the cloud for every request will severely hamper user experience, especially for highly interactive games. In the next subsections, we discuss a model deployment of a cloud-game on fog over a 5G network.

1.4.2.1 Physical Deployment

Gaming clients running on mobile devices host the device component of the game application. They are connected to 5G base stations, which have the ability to host computation and storage. These base stations host the fog component of the gaming application as virtual machines. Base stations are connected to the cloud via high speed Ethernet links. Virtual machines in the cloud are responsible for carrying out the logic described in the cloud component of the application.

1.4.2.2 Application Architecture

To discuss effectively the deployment of a cloud-based game on a fog computing infrastructure, we need to host the aforementioned components of a typical cloud-

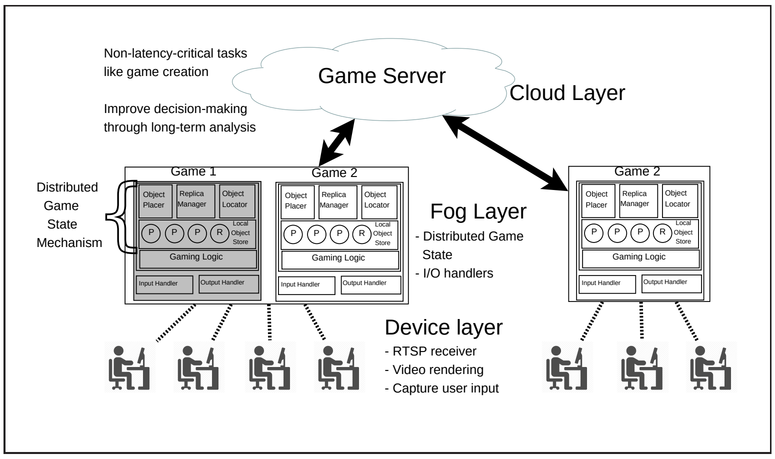

Figure 1.6: Deployment of cloud gaming on fog infrastructure over 5G mobile network.

based game on the three different kinds of computation offload points available. Figure **??** shows the mapping of application components to fog infrastructure.

Device component

The device component, that is the application logic running on game clients, hosts the Real Time Streaming Protocol (RTSP) reception module for receiving incoming video and audio frames. In addition to this, the component needs to have input handler modules for capturing inputs from users' consoles and sending them to the server.

Fog component

The fog component of the application holds most of the computation and storage involved in the distributed game system. This computation hosting is realized by virtualization technology on fog-enabled edge devices. The application component of a particular game runs on the fog devices inside a virtual machine.

The gaming virtual machine contains, as any cloud-gaming server would, an input handling module for receiving events from users and applying them to the gaming logic, and an output module that captures the rendered audio and video, encodes them and sends them to the clients via RTSP. The purpose of these modules is to allow a basic game to function (the way it did on a cloud-based deployment), and are agnostic to the gaming application. In addition to these modules, a fog game server hosts a distributed game state mechanism (Figure **??** shows Colyseus), so that game-play state and computation is distributed among all such fog nodes in the network. This module makes the gaming experience look transparent to the number and geographical distribution of users, by pre-fetching game objects in the area of interest, making users get the impression of a single gaming server while being able to overcome the concomitant latency and scalability issues of a single-server implementation.

The majority of data transfer in the gaming application takes place between the fog component and the device component, since the fog needs to stream the video to the game client. To be able to effectively harness the gigabit speed and sub-millisecond latency of 5G transmission, it is necessary that these components are separated by no more than one hop in 5G. Hence, hosting the fog component on the base stations - to which a game client directly connects - is a requirement. Other communications are not heavy, and mostly pertain to game events - which can be sent on fibre links without considerable delay and bandwidth consumption.

Cloud component

The cloud layer hosts logic that are not required to work under strict latency constraints - like game initiation and hosting static game-maps. The cloud can also assist the gaming virtual machines running on fog devices by learning the optimal control strategy through long-term analytics on the decisions taken by them in the past. Furthermore, to assist communication between edge servers running the fog component of the gaming application, the cloud component will provide a message-passing interface - like a publish-subscribe protocol.

1.5 Use Case 3 : Smart Homes

The proliferation of the Internet of Things (IoT) has given a great boost to smart home automation systems. The smart home market is presaged to cross $44bn in five years from now[17], bringing with it new opportunities for mobile network operators and the rest of the mobile ecosystem.

1.5.1 Requirements

Smart home is an amalgamation of various technologies which collectively improve the lifestyle and experience of the user through coordinated functionalities. A typical smart home applications should fulfil the following requirements.

Energy efficiency

A smart home environment contains a lot of different kinds of devices apart from the appliances normally found in homes. Such devices consume a considerable amount of energy, thus making energy minimization one of the key objectives of an efficient smart home design.

Average American Home Electrical Usage [16]

Heating and cooling accounts for 54% of a household’s electricity bill while lighting consumes 25% of the total use. Standby power leaked by devices accounts for 10% of total electrical use.

”Little changes can make a big difference in a home’s energy consumption. And with modern technology, it’s so easy to automate energy use, so you don’t even have to think about it,” - Adam Justice, founder of **ConnectSense**, a wireless, cloud-based home automation device.

Occupancy sensors installed in smart homes can detect the absence of any activity in an area and turn the lights of that location off in order to save electricity. The same can be done for air conditioners, geysers and room heaters to cut down the expenses incurred on heating etc. Another way to save power is by eliminating phantom energy loss from devices like microwaves even though they are switched off but plugged into the socket. Studies like [25] and [19] have shown the potential of reducing power consumption of a house by cutting down standby power loss. Data about power consumption of a device - detected by power sensors - coupled with the knowledge of whether the device is on or off can be used to detect the phantom energy loss and the outlet powering the device can be switched off. This requirement is not complex in terms of implementation and can save a large amount of electricity, thus making it one of the most popular requirements of a smart home.

Safety

User safety is one of the key concerns of any system in general, and smart homes in particular. A smart home application should be able to detect intruders or any suspicious activity happening around the house. CCTV cameras installed outside the house can detect suspicious activity and send an alert message to the smart home application, which can take action by activating an alarm and turning on the lights of the area. Glass-breakage systems can detect an intruder, while motion sensors can detect movement in the house when the owner is away and inform the owner, and also call the police in case the situation demands it. Products offering these services like *Canary*[1] make use of the computation on both the cloud and edge-devices, but these products use a separate hardware and form a different ecosystem that is difficult to tie with the complete smart home fog ecosystem.

Maintaining home environment

The most important purpose of a smart home is to improve the experience of the house owner by maintaining optimal physical conditions like temperature and humidity inside the home and providing assistance for daily tasks, like preparing coffee on waking up, maintaining optimal lighting by drawing curtains based on time of day and weather. Such application logic works by processing streams of data generated by sensors that sense the physical conditions and detect the activities of the user.

Mobile Dashboard and Long-term analysis

A smart home application should have a mobile dashboard using which a user can check the state of his house even from a remote location. The user can use the dashboard to even control objects in the house, like opening/closing doors, talking to people who visit in his absence, or informing the police in case of an emergency. This use-case requires analysis of data generated by the smart home and a user-friendly presentation of information extracted from this analysis. It also demands global coverage since the smart home application may have to communicate with a user at a remote location.

Design Requirements

- **High-speed communication** : Due to the number of devices connected in a smart home ecosystem, there is a need for an efficient M2M communication mechanism that incurs a very small delay. For a smooth user experience, it is necessary for the devices coordinate and perform in real-time - thus requiring high speed M2M communication.

- **Handling high data volume** : Smart homes generate massive volumes of data, particularly due to the number of devices connected. Hence, the device processing the data as well as the connecting network should be able to handle such an immense volume of data.

1.5.2 Deployment Details

The requirements of a smart-home system are peculiarly handled by fog computing owing to its *near-the-edge* processing and resultant low-latency. The deployment of a smart-home system on fog infrastructure over a 5G network is described as follows.

1.5.2.1 Physical Deployment

A smart home system consists of a myriad of connected devices serving a variety of functions, ranging from sensors measuring temperature, humidity, or detecting presence or fire to high-level appliances like smart air-conditioners and CCTV cameras. These devices need to communicate with each other and perform coordinated functions to serve the requirements of a smart home, requiring an efficient and reliable M2M communication. The M2M communication facility provided by 5G mobile networks and it's ability to support a huge number of connected devices makes it an enabling technology for smart home automation systems. Smart devices are connected to a *small cell* hosted inside the house, which in turn is connected to the core network via a high speed broadband connection. This small cell acts as the *smart home gateway* for the devices in the smart home, and serves as a point for offloading computation and storage of smart home applications.

The intermediate network devices (typically belonging to the ISP) connecting a group of smart homes to the Internet also serve as offload points for the smart home application. Owners of smart homes connect to the smart home application through

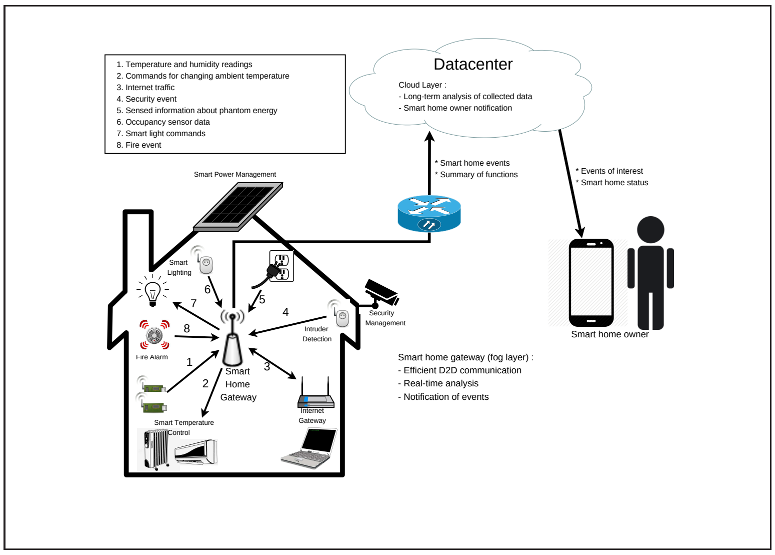

Figure 1.7: Deployment of a smart home.

high-speed 5G mobile network using their smartphone application and can access the smart home dashboard to view/modify the status of the smart home.

1.5.2.2 Application Architecture

Efficient use of the services provided by fog computing is possible only when the smart-home application logic is partitioned into the three components of a fog application. This partitioning is elucidated in the following.

Device component

The data collection points of a smart home are mainly sensors, which need to send the sensed data to the smart home gateway in a timely manner. The application logic deployed in CCTV cameras should process the captured video and detect events of interest in real-time and inform the smart home gateway.

The application component running on actuators - like air-conditioners, fire alarm, etc. - in a smart home should be able to receive commands from the smart home gateway and implement those in real-time.

Fog component : Smart Home Gateway

The *smart home gateway* is the seat of control of the smart home and is responsible for running applications that coordinate the activities of various smart devices to create a holistic smart home experience. In order to fulfil the requirements of a smart home system (as described earlier), the smart home gateway needs to provide the following services :

- **Efficient M2M communication** : The smart home gateway needs to provide interface to an M2M communication interface for the smart home devices to communicate with each other. Energy management requires communication between the sensors and electrical appliances, so that they may be turned off when sensors detect absence of activity. For ensuring safety of the home, CCTV cameras and motion sensors should be able to communicate with the smart home gateway in a timely manner, thus requiring efficient communication between smart home devices. The devices in a smart home are generally resource-constrained, and using standard protocols like HTTP for message passing will be inefficient.

  In recent years, a lot of effort has been made towards developing protocols for M2M communication between resource constrained devices, some of which have yielded popular outcomes like MQTT, COAP and SIP.

MQ Telemetry Transport (MQTT)

MQTT works on an asynchronous publish-subscribe architecture and is realized by sending control packets. MQTT packet headers are kept as small as possible, making this protocol apt for IoT by lowering the amount of data transmitted. Hence, this protocol is suitable for contrained networks (low bandwidth, high latency and fragile connections).

  Daş et al. [13] have demonstrated an example implementation of a smart home application using M2M communication between resource constrained sensors and home appliances. In their implementation, communication between devices and sensors was enabled by a SIP server in the smart home. Drawing parallels from the proposed implementation, the *smart home gateway* will support a number of such protocols which can be used by application developers to build useful smart home applications.

- **Real-time Data analysis** : The smart home application needs to process events, especially those related to safety and energy management, in real time. Works like [37] and [7] have shown how real-time analysis of data can benefit smart homes in terms of energy management and security respectively.

  Processing offload could not get any closer to the appliances than the *smart home gateway*. Such a close vicinity to the device reduces the communication delay of between the gateway and the sensors and appliances, allowing smart home applications to make real-time decisions. In cloud-based smart homes, the control system used to reside in the cloud, giving rise to a high latency between the devices and the control system.

- **Network Traffic Reduction** : Devices in a smart home generate a lot of data, typically because of continuously sensing the environment. In fact, a large portion of the data is redundant or not useful. Smart home applications analyse this little big data to extract valuable information. However, the volume of this data, from a group if not from a single house, can be too huge to be continuously sent to the cloud for processing. The presence of a control system right at the gateway solves this problem by performing the data cleaning and analysis to generate control signals for the appliances. Being located at the very edge, the analysis happens in real-time.

Cloud component

A typical smart home application should allow its users to view and control the status of the smart home from any remote location. Such a use-case requires a global coverage of the smart home application, thus justifying the deployment of this logic in the cloud. This component of the application will receive aggregated summaries of the smart home's status at regular intervals as well as notifications of events requiring immediate attention from the smart home gateway and inform the owner of the house.

This component will also perform long-term analytics on data collected from a smart home as well as data from multiple smart homes to detect any kind of usage patterns that it may leverage to improve the services provided in smart homes.

1.6 Distributed Camera Networks

Distributed system of cameras surveilling an area has garnered a lot of attention in recent years particularly by enabling a broad spectrum of interdisciplinary applications in areas of the likes of public safety and security, manufacturing, transportation, and healthcare. The widespread use of these systems has been made possible by the proliferation of economical cameras and the availability of high-speed wired and wireless networks. Such a large number of cameras makes these systems generate data at very high rates. Monitoring these video streams manually is not practical, if at all feasible, thus engendering the need for tools that automatically analyze data coming from cameras and summarize the results in a way that is beneficial to the end-user.

Centralized tools for analyzing camera-generated data are not desirable in a lot of cases primarily because of the huge amount of data that needs to be sent to the central processing machine. This would not only lead to a high latency in the system, but would also devour the bandwidth. Hence, processing the video streams in a decentralized fashion is a more advisable method of analysis. A number of research works have explored distributed camera networks [14], [23]. The requirements of such a system have been listed down as follows.

1.6.1 Requirements

Distributed camera network involves communication between devices only, and thus poses unique requirements, which have been discussed in [28].

Real-time consensus among cameras

Decisions taken by cameras need to be coordinated in order to attain a consensus about the task they are performing (e.g. activity recognition). Hence, there is a need for communication, that too one with a low delay, between cameras covering overlapping or adjacent regions. Furthermore, arriving at a consensus in real-time demands that the processing of video streams be done in a latency-critical manner. Sending all video streams to the cloud will be inefficient in this case due to the high delay incurred in communicating with the cloud.

Real-time PTZ tuning

In case of fixed cameras, video analysis becomes difficult because the fixed resolution or viewpoint may not be able to capture the target. The distributed camera network should support active sensing allowing cameras' parameters such as pan-tilt-zoom (PTZ) and resolution to be controlled by the video analysis system. This tuning of camera parameters has to be done in real-time in order to effectively capture the target. Apart from functioning in real-time, the parameter tuning of cameras needs to be adaptive, being able to learn from previous decisions and improve.

Event notification

The distributed camera network should inform the security personnel monitoring the area about the occurrence of an event. This use-case requires a global coverage, since the user may be present at a remote location.

Design Requirements

- **Low-latency communication** : For effective object coverage, the PTZ parameters of multiple cameras need to be tuned in real-time based on the captured image. This requires ultra low latency communication between the cameras and the seat of camera control strategy.
- **Handling voluminous data** : Video cameras continuously send captured video frames for processing, which amounts to a huge traffic, especially when all cameras in a system are taken into account. It is necessary to handle such a large amount of data without burdening the network into a state of congestion.
- **Heavy long-term processing** : The camera control strategy needs to be updated constantly so that it learns the optimal PTZ parameter calculation strategy. This requires analysis of the decisions taken by the control strategy over a long-period of time, which makes this analysis computationally intensive.

1.6.2 Deployment Details

There have been several studies like [14] and [23] that cover distributed sensing in camera networks. Of particular relevance to fog computing is the work by Peng et

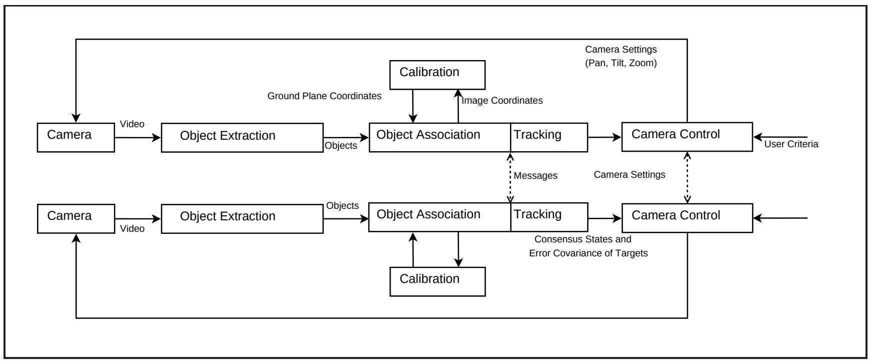

Figure 1.8: Schematic diagram of a distributed camera analysis system.

al. [23] in which they have proposed the use of camera servers physically close to the cameras for processing real-time queries on networks of distributed camera networks. Based on the concept presented by Peng et al., we discuss a typical deployment of distributed camera analysis system on fog infrastructure in the following.

1.6.2.1 Physical Deployment

The data generation points in a distributed camera network are the numerous surveillance cameras which generate data in the form of captured frames at a constant rate. Cameras are also the prime actuators in this system as they need to constantly change the PTZ parameters in order to get the best coverage of the target. Cameras in a DCN are connected via high bandwidth 5G connection to a small cell located in physical proximity of the cameras. The small cell connects the DCN to the cloud via high-speed Ethernet.

1.6.2.2 Application Architecture

We now discuss the application architecture, i.e. the placement of application logic into components that can be deployed at different offload points in the fog network.

Device Component

The device component of a distributed camera network application runs on a camera and contains the code for handling it. It essentially consists of two modules - **video sender** and **command receiver**. The video sender module sends the recorded frames to the associated small cell at a constant rate. The command receiver module receives instructions to change the camera parameters from the small cell and applies them to get a better coverage of the target. The encoding of video and sending it should take place in real-time, as well as the PTZ change commands received from small cells should be applied in real-time so as to bring down the response time of the DCN to real-time domain.

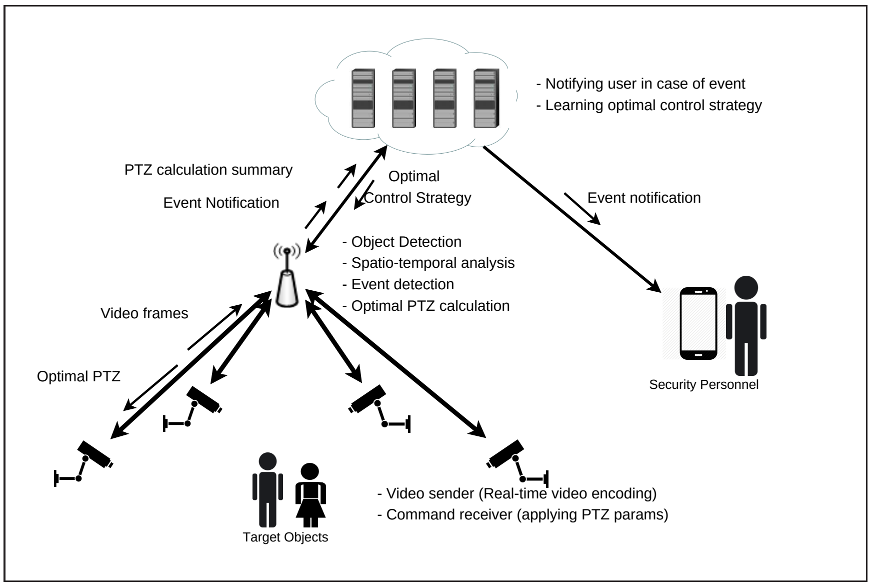
- Notifying user in case of event
- Learning optimal control strategy
PTZ calculation summary
Event Notification
Optimal
Control Strategy
Event notification
- Object Detection
- Spatio-temporal analysis
- Event detection
- Optimal PTZ calculation
Video frames
Optimal PTZ
Security Personnel
- Video sender (Real-time video encoding)
- Command receiver (applying PTZ params)
Target Objects

Figure 1.9: Deployment of a distributed camera network on fog over 5G network.

Fog Component

The fog component of the application is responsible for detecting events based on spatio-temporal relations between objects across video streams coming from different cameras. The application logic first filters out objects of interest from the live camera feeds by using image processing techniques. It then uses the spatio-temporal relations between the detected objects to detect if an event has occurred. Works like [23], [36], and [12] have proposed techniques performing event detection in real-time from live camera feeds. In case an event is detected, the fog component informs the cloud component of the application so that users of the system, even at remote locations, can get notified of the occurrence of the event.

The fog component of a DCN application is also responsible for the camera control strategy, that is tuning the parameters of the cameras in order to optimize the scene acquisition capabilities of the cameras. Based on the camera feeds, the application calculates the optimal PTZ parameters for each camera. In addition to the PTZ parameters, the application also responds to scene complexity by determining the optimal resolution for the camera to capture, for example, to capture a higher resolution video when the scene is relatively empty and lower the resolution as the number of objects in the scene increases. Starzyk et al. has elucidated on optimal camera parameter calculation in [30] and [29], the calculation being based on images captured by the cameras. These optimal parameters are sent to the cameras in real-time which apply them to improve the quality of the captured scene. The control strategy hosted by the fog component is adaptive and tries to improve itself based on previous decisions. For this, the fog component sends an aggregate of camera control decisions taken in the past to the cloud component for determining the optimal control strategy, which is then communicated back to the fog component.

In centralized systems, video streams had to be sent to the cloud for processing, and cameras would receive instructions for tuning camera parameters from the cloud, both these communications exhibiting an unpredictable delay which could mar the purpose of the system. Furthermore, sending live video streams to the cloud at all times would devour the bandwidth and may lead to congestion, further delaying frame delivery. In a fog setting, it is apt to place the fog component (which takes video streams as input) on the small cells connecting a group of cameras. Placing this component at the very edge, close to the source of data and actuators, greatly cuts down the delay. The only communication between the small cells and the cloud takes place when an object of interest is detected or when the control strategy of the fog component needs to be updated - which is relatively inexpensive in terms of bandwidth consumed.

Cloud Component

Research works like [29] suggest that the control strategies of cameras can be learnt using online learning algorithms. Since the latency requirement of updating the control strategies is not that stringent, the cloud component can perform the learning based on the information about previous decisions taken by the fog component. The cloud component uses advanced learning tools to determine the optimal control strategy at every time and updates the control strategy currently running on the fog component. This task is computationally intensive, hence making it a good fit to be run on the cloud.

The cloud component of a DCN application enables security personnel present at remote locations to monitor the activities in the area surveilled by the DCN by sending notifications pertaining to events of interest to the security personnel, who can respond accordingly. In special cases, the cloud component may also stream video related to the event so that security personnel may have a look at the actual situation. This part of the application demands a wide coverage as the users monitoring the activity in the surveilled region may not be located in vicinity of the cameras or small cells. Hence, it makes a lot of sense to deploy this application logic to the cloud which has a global coverage.

1.7 Conclusion: Open Challenges and Future Trends

In this chapter, few peculiar use-cases apt for fog computing on 5G networks have been discussed. The requirement of edge-processing for 5G networks have been pointed out and the deployment of use-cases on a model 5G network has been shown. However, large-scale successful deployment of fog computing systems on 5G mobile networks is bound by research on a number of areas. Fog computing in 5G networks is as much of a vision today as 5G networks itself and a number of challenges need to be addressed to make it a reality. These challenges are described as follows :

- **Computation offloading in network base stations** : Fog applications run in the form of virtual machines (or containers) on virtualized fog-devices. This would require shifting the network functions - originally implemented in dedicated hardware - to software (a concept called Network Function Virtualization). However, implementing NFV on such a heterogeneous network as a fog-enabled 5G network is still not lucid. Works like [10] and [11] have explored into the implementation of network virtualization on mobile networks. Further advancements need to be made in this domain for efficiently realizing fog computing on 5G networks.
- **Energy efficiency** : Fog computing on 5G network requires the base stations to be enabled with virtualization for running applications. Running applications on a hypervisor a higher energy requirement due to the heavier processing involved in virtualization. This carbon footprint would be amplified to a great extent by the numerous fog-devices in a network with a dense geographical distribution. Minimizing energy consumption is a key challenge that needs to be addressed for successful commercialization of fog computing on 5G.
- **Pricing policies** : A fog ecosystem will consist of two kinds of stakeholders : 1) Internet Service Providers who construct the fog infrastructure 2) The application service providers who want to extend their applications to the edge. Thus for enabling *pay-as-you-go* pricing model, it is necessary to decide the price for resources and the division of payment for different parties. This will be difficult given the widely distributed network of fog devices. For realizing this utilization-based pricing policy, accounting and management systems working at a very fine granularity of the network are required.
- **Resource management** : Efficient resource provisioning and management has been a strong reason for the success of cloud computing - and will continue to be so for fog computing as well. However, the problem of resource management is even tougher, because of the added dimension of network latency involved. Besides, the vast number of heterogeneous devices in the network further complicates resource management. Ottenwälder et al. have proposed MigCEP[20] and operator migration policy for fog infrastructure so as to optimize on end-to-end latencies as well as bandwidth consumption and their work is one of the few contributions to resource management on fog.
- **Privacy and Security** : Fog computing virtualizes the network and decouples network functionality from the hardware provider. Hence, fog applications process application data on third-party hardware, which poses strong concerns about visibility of data to the third-party. 5G networks handle voice and data packets in the same manner which may lead to leakage of sensitive voice data. This makes privacy measures even more necessary for fog computing on 5G networks. Stojmenovic et al. have discussed the security issues in fog computing in their work[31].

Only by addressing these challenges will fog computing on 5G networks be successful commercially. Both 5G networking and fog computing technologies are compatible with each-other and their amalgamation will be the enabler of applications that define the future of Internet. Due to its close affinity for use-cases on IoT, fog computing will help realize applications that will help us build a greener, more sustainable future for mankind.

Bibliography

[1] Canary: The first smart home security device for everyone. `http://canary.is/`, visited 2015-12-06.

[2] Cisco delivers vision of fog computing to accelerate value from billions of connected devices. `http://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1334100`, visited 2015-12-08.

[3] Why edge computing must be a companion to nfv and 5g ? `http://www.telecomtv.com/articles/mec/why-edge-computing-must-be-a-companion-to-nfv-and-5g-13043/`, visited 2015-12-06.

[4] Jonathan Bar-Magen. Fog computing: introduction to a new cloud evolution. In *Escrituras silenciadas: paisaje como historiografía*, pages 111–126. Servicio de Publicaciones, 2013.

[5] Ashwin R Bharambe, Jeff Pang, and Srinivasan Seshan. *A distributed architecture for interactive multiplayer games*. Citeseer, 2005.

[6] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 169–186. Springer, 2014.

[7] Wen-Tsuen Chen, Po-Yu Chen, Wei-Shun Lee, and Chi-Fu Huang. Design and implementation of a real time video surveillance system with wireless sensor networks. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 218–222. IEEE, 2008.

[8] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In *Proceedings of the 11th annual workshop on network and systems support for games*, page 2. IEEE Press, 2012.

[9] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia Systems*, 20(5):503–519, 2014.

[10] Xavier Costa-Pérez, Jorg Swetina, Tao Guo, Rajesh Mahindra, and Sampath Rangarajan. Radio access network virtualization for future mobile carrier networks. *Communications Magazine, IEEE*, 51(7):27–35, 2013.

[11] Jose Costa-Requena, Jesus Llorente Santos, Vicent Ferrer Guasch, Kimmo Ahokas, Gopika Premsankar, Sakari Luukkainen, Ijaz Ahmad, Madhusanka Liyanage, Mika Ylianttila, Oscar Lopez Perez, et al. Sdn and nfv integration in generalized mobile network architecture. In *Networks and Communications (EuCNC), 2015 European Conference on*, pages 154–158. IEEE, 2015.

[12] Abir Das, Anirban Chakraborty, and Amit K Roy-Chowdhury. Consistent re-identification in a camera network. In *Computer Vision–ECCV 2014*, pages 330–345. Springer, 2014.

[13] Resul Daş and Gurkan Tuna. Machine-to-machine communications for smart homes.

[14] Chong Ding, Bi Song, Akshay Morye, Jay Farrell, Amit K Roy-Chowdhury, et al. Collaborative sensing in a distributed ptz camera network. *Image Processing, IEEE Transactions on*, 21(7):3282–3295, 2012.

[15] Sébastien Faye, Claude Chaudet, and Isabelle Demeure. A distributed algorithm for multiple intersections adaptive traffic lights control using a wireless sensor networks. In *Proceedings of the first workshop on Urban networking*, pages 13–18. ACM, 2012.

[16] Ilana Greene. Ilana greene. `http://www.huffingtonpost.com/ilana-greene/smart-houses-help-reduce-_b_4472919.html`, visited 2015-11-26.

[17] GSMA. Vision of smart home : The role of mobile in the home of the future. `http://www.gsma.com/connectedliving/wp-content/uploads/2012/03/vision20of20smart20home20report.pdf`, visited 2015-11-20.

[18] Stefan Lämmer and Dirk Helbing. Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(04):P04019, 2008.

[19] Ing Nils F Nissen. Eup preparatory study lot 6 standby and off-mode losses. *Compiled by: Fraunhofer Institute for Reliability and Microintegration, IZM, Berlin*, 2007.

[20] Beate Ottenwälder, Boris Koldehofe, Kurt Rothermel, and Umakishore Ramachandran. Migcep: operator migration for mobility driven distributed complex event processing. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 183–194. ACM, 2013.

[21] FT Park. Traffic data collection and analysis. 1980.

[22] Mugen Peng, Yuan Li, Jiamo Jiang, Jian Li, and Chonggang Wang. Heterogeneous cloud radio access networks: A new perspective for enhancing spectral and energy efficiencies. *Wireless Communications, IEEE*, 21(6):126–135, 2014.

[23] Rui Peng, Alex J Aved, and Kien A Hua. Real-time query processing on live videos in networks of distributed cameras. *Research, Practice, and Educational Advancements in Telecommunications and Networking*, page 27, 2012.

[24] Zhouyue Pi, Farooq Khan, and Jianzhong Zhang. Techniques for millimeter wave mobile communication, October 29 2010. US Patent App. 12/916,019.

[25] K Roth, K McKenney, R Ponoum, and C Paetsch. Residential miscellaneous electric loads: energy consumption characterization and savings potential in 2006 and scenario-based projections for 2020. *Final Report by TIAX LLC to US Department of Energy*, 121, 2008.

[26] Noah Shachtman. Feds look to fight leaks with fog of disinformation’. `http://www.wired.com/2012/07/fog-computing/`, visited 2015-12-12.

[27] Ryan Shea, Jiangchuan Liu, Edith Ngai, and Yong Cui. Cloud gaming: architecture and performance. *Network, IEEE*, 27(4):16–21, 2013.

[28] Bi Song, Chong Ding, Ahmed T Kamal, Jay Farrell, Amit K Roy-Chowdhury, et al. Distributed camera networks. *Signal Processing Magazine, IEEE*, 28(3):20–31, 2011.

[29] Wiktor Starzyk and Faisal Z Qureshi. Learning proactive control strategies for ptz cameras. In *Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on*, pages 1–6. IEEE, 2011.

[30] Wiktor Starzyk and Faisal Z Qureshi. Multi-tasking smart cameras for intelligent video surveillance systems. In *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pages 154–159. IEEE, 2011.

[31] Ivan Stojmenovic and Sheng Wen. The fog computing paradigm: Scenarios and security issues. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pages 1–8. IEEE, 2014.

[32] Michael AP Taylor and William Young. *Traffic Analysis: new technology and new solutions*. 1988.

[33] Michael AP Taylor, William Young, and Peter W Bonsall. *Understanding traffic systems: data, analysis and presentation*. 1996.

[34] The Telegraph. Smart traffic lights to stop speeders. `http://www.telegraph.co.uk/motoring/news/8521769/Smart-traffic-lights-to-stop-speeders.html`, visited 2015-12-12.

[35] Marco Wiering, Jelle Van Veenen, Jilles Vreeken, and Arne Koopman. Intelligent traffic light control. *Institute of Information and Computing Sciences. Utrecht University*, 2004.

[36] Shu Zhang, Yingying Zhu, and Amit Roy-Chowdhury. Tracking multiple interacting targets in a camera network. *Computer Vision and Image Understanding*, 134(C):64–73, 2015.

[37] Suyang Zhou, Zhi Wu, Jianing Li, and Xiao-ping Zhang. Real-time energy control approach for smart home energy management system. *Electric Power Components and Systems*, 42(3-4):315–326, 2014.