Advancing IoT Architectures Using Collaborative Computing Paradigms for Dynamic and Scalable Systems

Prashant G Joshi and Bharat M. Deshpande
Department of Computer Science & Information Systems, BITS Pilani, K K Birla Goa Campus, Goa, India
{*p20160413, bmd*}*@goa.bits-pilani.ac.in*

Keywords: Collaborative Computing Paradigms, CCP, IOT System, IOT System Architecture, IOT Reference Architectures, Distributed Computing, System Software Architecture, Scalable IOT Architectures, Dynamic IOT Environments.

Abstract: The Internet of Things (IoT) continues to push the boundaries of traditional system architecture models, necessitating more agile, scalable, and efficient frameworks to handle complex, large, real-time applications. While conventional architectures, like layered, provide a structured approach, they often suffer from limitations in adaptability, latency, and dynamic resource utilisation. This paper presents a comprehensive design and implementation a Collaborative Computing Paradigms (CCP)-based IoT Reference Architecture, defined by five core attributes: interconnection and interplay across paradigms, dynamic distribution of data processing, computing fluidity enabling seamless transitions across layers, collaborative data storage and management, and scalability and extensibility of computational resources. To validate the proposed architecture, experiments were conducted in Data Center Management System, Automotive Telematics, Smart Building (Fire Safety, Environment) and Asset Tracking (Shipping carts in a mall). Comparative analysis against traditional layered IoT architectures reveals significant improvements in latency reduction, resource utilisation, scalability under variable workloads, and interoperability between computational layers. The CCP-based architecture leverages dynamic orchestration of Edge, Fog, and Cloud paradigms, allowing for adaptive load balancing, real-time analytics, and distributed decision-making, thereby overcoming the rigidity of layered models. The results highlight the superiority of CCP architectures in enabling low-latency processing, high fault tolerance, and dynamic resource optimisation in highly fluid and demanding IoT environments. We believe this work underscores the paradigm shift towards collaborative architectural models, establishing CCP as a benchmark for next-generation IoT system design, particularly in domains requiring high degrees of responsiveness and cross-layer integration.

1 INTRODUCTION

The Internet of Things (IoT) has rapidly transformed how we interact with technology, embedding intelligence into everyday devices, and enabling a seamless flow of data. From smart homes to industrial automation, IOT promises to revolutionise our lives by creating hyper-connected environments (Ericsson, 2024). However, the success of IoT systems hinges on efficient data processing, communication, extandability and scalability (Joshi and Deshpande, 2024a).

Collaborative computing paradigm (CCP), as discussed by the authors (Joshi and Deshpande, 2024a), has emerged as an advanced solution to bridge the gap between IoT devices and today's real-world applications. Unlike centralised systems, collaborative computing paradigms leverage distributed resources, enabling devices to work together, share computational loads, and make real-time decisions. This approach has potential to not only optimise the performance, but also enhance fault tolerance, resource management, and energy efficiency, all critical for IoT deployment.

In addition to sensing and transmitting data, IoT systems encompass several other critical features that are key to enabling real-world applications. Data processing allows for real-time decision-making by analysing and extracting actionable insights from vast streams of information. Storage plays a vital role in preserving data for future use, support advanced analytics, and train predictive models. Furthermore, actuation enables IoT systems to achieve control by triggering physical actions based on processed data, such as adjusting a thermostat or activating machin-

Joshi, P. G. and Deshpande, B. M.
Advancing IoT Architectures Using Collaborative Computing Paradigms for Dynamic and Scalable Systems.
DOI: 10.5220/0013388900003896

In *Proceedings of the 13th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2025)*, pages 109-121
ISBN: 978-989-758-729-0; ISSN: 2184-4348

ery. These features, when integrated seamlessly, empower IoT to deliver intelligent, responsive, and automated solutions across diverse domains.

We achieve this by designing key components from Collaborative Computing Paradigms (CCP), using the CCP-IOT-RA, a reference architecture validated by (Joshi and Deshpande, 2024b), which aligns with the specific requirements of a dynamic IoT environment. The CCP-IOT-RA is adopted to address real-world IoT challenges, ensuring its applicability in diverse scenarios. Furthermore, we build and validate these components through practical implementation and experimentation, demonstrating their effectiveness in enabling scalable and efficient IoT solutions. We believe that, this structured approach, ensures that the proposed paradigms are not only theoretically sound but also practically viable for real-world applications.

In this paper, we explore collaborative computing paradigms, tailored for the IoT, and their role in building scalable, robust, and real-world applications. By addressing both the technical and practical aspects, this paper aims to provide a comprehensive framework for future IoT development through collaborative computing.

This paper is organised as follows. Section 2 provides a brief summary of the CCP and CCP-IOT-RA. Section 3 provides the details of the methodology developed and used for overall experiment design and processes. Section 4, we provide the details of the design, and implementation of the CCP based IOT, detailing the devices and nodes, and their functionality. Section 5 describes in brief the experiments conducted, and their results. We summarise the lessons learned in Section 6. Section 7 provides a conclusion, and Section 8 closes with the future work.

2 CCP BASED IOT ARCHITECTURE: A SUMMARY

As discussed by the authors (Joshi and Deshpande, 2024a), CCP with its five characteristics is used to build a RA which is detailed in (Joshi and Deshpande, 2024b). The CCP employs a three-pronged approach:

- Build an architecture of infrastructure which can enables and creates opportunities for dynamism, and expansion of services
- Build the system using well-established standards to enable ease of operation, and interoperability
- Build using such an infrastructure software system, that brings functional dynamism

A typical system architecture, of the CCP (CCP-IOT-RA), is depicted in Figure 1.

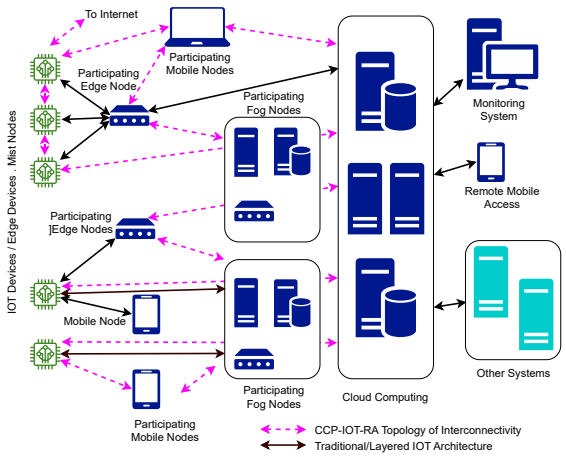

Figure 1: Collaborative Computing Paradigms (CCP-IOT-RA) - Systems Architecture.

The characteristics of the Collaborative computing paradigms software system architecture are detailed in TABLE 1.

By establishing a collaborative systems architecture, the software can leverage available computing paradigms depending on the level of dynamism it can accommodate, such as in distributing tasks across different paradigms.

3 METHODOLOGY FOR DESIGN & IMPLEMENTATION

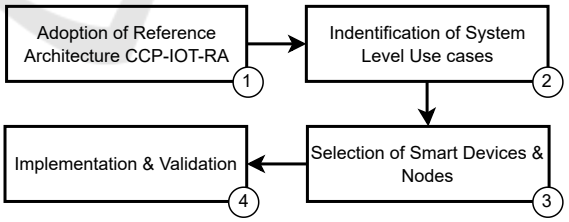

Figure 2: Methodology and Steps for Design and Implementation.

The implementation of a tailored reference architecture, for IoT systems, involves several structured phases to ensure robustness, scalability, and relevance to critical applications. This section outlines the systematic approach for tailoring the CCP-IOT-RA framework, identifying relevant use cases, and validating the proposed architecture. The process steps are depicted in Figure 2.

Our prime reference for the architecture is the

Table 1: Collaborative Computing Paradigms- Software Systems Architecture Characteristics (Joshi and Deshpande, 2024a).

No.	Characteristic	Description and details
1	Inter-connection and interplay	Overcome the limitations imposed by layered approach by interconnection and interplay between smart devices, and various computing paradigms.
2	Dynamic distribution of data processing	Dynamic distribution, free from strict layering and predefined computing allocation, enhances flexibility, efficiency, and overall computing capacity
3	Fluidity of computing across paradigms	Task allocation is dynamic, determined by the paradigm's suitability, contextual requirements, and available resources.
4	Storage and data management across participating paradigms	All participating paradigms will require the necessary data for processing and thus, the data will be distributed across paradigms. This data, both input data and processed data, will need to be collected and transmitted to a single store of data, typically cloud storage.
5	Scalability and extendability of the architecture	It supports the integration of newer devices seamlessly, allowing the architecture to accommodate additional components to monitor subsystems, analyse data, and ensure the scalability and reliability of the overall system.

(H. Washizaki, 2024)

In this section we detail the process steps and the high level topics in each of the process steps for a successful implementation of the IOT system using the CCP.

3.1 Adoption of the Reference Architecture CCP-IOT-RA

The CCP-IOT-RA serves as a foundational framework for IoT systems, offering modularity, interoperability, and scalability. To meet the specific requirements of the targeted application, the following customisations are considered:

- **Component Selection:** Key components such as sensors, smart devices, nodes - edge, fog and cloud, communication protocols, data processing and storage nodes, and security modules are selected and tailored to match the application context.
- **Adaptation to Use Cases:** System-level use cases, such as sensing, transmission, processing, actuation and applications are created to ensure seamless integration with critical functionalities.
- **Scalability Considerations:** The architecture is designed to accommodate increasing the device connections, high intervals of data transmission, data volume, and computational demands without compromising performance.
- **Integration of Advanced Features:** Smart computing capabilities, edge processing, and adaptive data storage mechanisms are integrated into the architecture.

3.2 Identification of System Level Use Cases

Among the identified use cases, critical ones are prioritised based on the need for robust computing power, data processing, storage capabilities, and scalability. The selection criteria include:

- **Computing Requirements:** Applications that demand extensive edge or cloud-based computation, such as real-time decision-making systems in smart cities.
- **Data Processing Needs:** Systems requiring efficient data preprocessing, such as machine learning-based anomaly detection in industrial IoT.
- **Data Storage Demands:** Applications requiring long-term storage and retrieval of large datasets, such as environmental monitoring logs in agriculture.
- **Scalability:** Use cases that require seamless scaling of resources, such as growing smart grid networks or expanding patient monitoring systems.

3.3 Selection of Smart Devices & Nodes

To achieve the functional goals of the architecture, appropriate IoT devices and nodes are identified based on the requirements of the critical use cases:

- **Sensors and Actuators:** Devices such as temperature, humidity, motion sensors, and actuators, such as relays, for physical control in industrial and healthcare environments.
- **Edge Devices:** Microcontrollers (e.g., Raspberry Pi, ESP32 SoC), and edge computing nodes for local processing and data filtering.

- **Communication Nodes:** Protocols such as MQTT, CoAP, and WiFi for efficient communication between IoT devices and central systems.
- **Storage Systems:** Cloud-based platforms and distributed data stores for data persistence and retrieval.
- **Device and Node Selection:** It ensures compatibility with the customised CCP-IOT-RA and provides optimal resource utilisation.

3.4 Implementation & Validation

This phase involves implementing the tailored architecture and validating its performance against critical use cases. The following steps are performed:

- **System Deployment:** The selected devices and nodes are integrated into the CCP-IOT-RA framework. Use cases are implemented in real or simulated environments.
- **Performance Metrics Evaluation:** Key performance indicators (KPIs), such as latency, throughput, computational efficiency, and energy consumption, are measured.
- **Scalability Testing:** Tests are conducted to assess system performance under increasing workloads, and number of device connections.
- **Data Validation and Accuracy**: Data is collected from sensors and processed by nodes are validated for accuracy and reliability.
- **Use Case Validation:** Critical use cases, such as industrial monitoring and smart healthcare systems, are evaluated for their ability to meet functional requirements.

4 DESIGN & IMPLEMENTATION OF CCP BASED IoT

In this section, we provide the components, subsystems, and systems built for the experiments to validate the CCP. The reference architecture based on (Joshi and Deshpande, 2024b), for the system, is shown in Figure 1. To ensure we are able to validate the architecture using various applications, we need to build some of the smart devices and computing nodes. Each device or node are made of the characteristic hardware and software to participate in the IoT solution.

We refer to (Espressif, 2024), (Sensortec, 2024) and (Pi, 2024) for the details on SoCs and sensors used for our experiments. These components were built using the process steps and methodology, from 3 as a guideline and those from (Bass et al., 2012) (H. Washizaki, 2024)

4.1 Sensor Nodes: Smart Devices

Each of the device, sensor and smart, is characterised specifically based on their capabilities. In the context, sensor device maybe capable of sensing and digitisation or sensing, digitisation and transmission to the smart device. A smart device is capable of common features - OLED Display over I2C for a readout, WiFi for connecting to internet or private network, and Bluetooth for peer-to-peer communication, and certain quantity of Local Storage. All the smart devices, and nodes are summarised in Table 2.

4.1.1 Smart Device: Environment Sensor (T10i)

Our smart device employs the ESP32-WROVER-E System-on-Chip (SoC) as its core processing unit. The ESP32 is interfaced with the BME680 sensor via an I2C communication bus. Developed by Bosch Sensortec, the BME680—recently succeeded by the BME690—is a multi-functional sensor capable of measuring gas, pressure, temperature, and humidity, making it ideal for environmental monitoring. In addition, the device supports seamless connectivity with nodes over both WiFi and Bluetooth, enable robust and flexible communication within IoT networks.

4.1.2 Smart Device: Telematics CAN-OBD (TX1)

This device integrates an ESP32-WROVER-E System-on-Chip (SoC) paired with a CAN transceiver, allowing it to interface with a vehicle's CAN network via the On-Board Diagnostics (OBD) port. This enables the device to access and monitor vehicular data for various automotive applications. To further enhance its capabilities, a GPS receiver is incorporated into the design, providing precise location tracking. The system supports connectivity over both WiFi and Bluetooth, enabling communication with other nodes, and facilitating real-time data transmission and analysis.

4.1.3 Smart Device: Temperature Sensor (T10)

The device leverages the ESP32-WROVER-E SoC, a versatile and efficient microcontroller, as its main platform. Integrated with a DS1820 temperature sensor over a single wire interface, the system is designed to monitor multiple environmental DS1820 sensors, connected via the digital IOs of ESP32. Moreover, the ESP32's built-in capabilities allow it, along with local

Table 2: Summary of Smart Devices and Nodes.

No	Device or Node	Description and details	Storage	WiFi	Bluetooth
1	Smart Device - Environment Sensor (**T10i**)	Sensor: BME680/BME690 (Temperature, Humidity, Pressure, Air Quality)	Yes	Yes	Yes
2	Smart Device - Telematics CAN-OBD (**T10X1**)	Sensor: CAN/CAN-FD Transceiver (CAN Protocol), GPS (Location)	Yes	Yes	Yes
3	Smart Device - Temperature sensor (**T10**)	Sensor: DS1820 (Single wire semiconductor temperature sensor).	Yes	Yes	Yes
4	Smart Device - Electrical Energy Metering (**T10E**)	Interface: RS485 MODBUS for connecting with a smart electrical energy meter	Yes	Yes	Yes
5	Edge Gateway - ESP32 (**N32**)	Multi-interface Edge Gateway	Yes	Yes	Yes
6	Edge Gateway - Raspberry Pi (**NPi**)	Multi-interface Edge Gateway	Yes	Yes	Yes
7	Fog Node - Dell Mini PC (**NmPC**)	Multi-interface Fog Node with medium computing and storage	Yes	Yes	Yes

storage, to connect effortlessly to other devices or networks via WiFi and Bluetooth, ensuring efficient data transmission and enabling interoperability within IoT ecosystems.

4.1.4 Smart Device: Electrical Energy Metering (T10E)

The device incorporates the ESP32-WROVER-E System-on-Chip (SoC) alongside an RS485 transceiver, enabling communication over the RS485 interface using the MODBUS protocol. This setup facilitates seamless integration with smart energy meters for efficient data exchange and monitoring. Leveraging the ESP32's advanced capabilities, the system also supports wireless connectivity via WiFi and Bluetooth, ensuring compatibility with other network nodes for flexible and scalable deployment.

4.2 Node: Edge Computing

4.2.1 Edge Gateway: Raspberry Pi (NPi)

An edge computing node is developed using a Raspberry Pi, which serves as a central communication hub for smart devices. Functioning as an Access Point (AP), the Raspberry Pi enables devices to connect via WiFi for data exchange and coordination. Each fog node is equipped with internet connectivity, and supports communication protocols such as MQTT and HTTP (RESTful web services), allowing seamless interaction with smart devices and other computing nodes across both intranet and internet networks.

4.2.2 Edge Gateway: ESP32 (N32)

An edge computing node is developed using an ESP32, which serves as a central hub for communication with smart devices. The ESP32 node operates as an Access Point (AP), enabling smart devices to connect seamlessly over WiFi for data exchange and coordination. Each fog node is connected to the internet and supports communication protocols such as MQTT and HTTP (RESTful web services), allowing smooth interaction with smart devices and other computing nodes across both intranet and internet networks. Additionally, the device is equipped with two analog inputs, four digital inputs, two output relays, and an RS485 transceiver, further expanding its capabilities for interfacing with sensors, actuators, and other industrial systems.

4.3 Node: Fog Computing (NmPC)

A fog computing node is implemented using a Dell Mini PC. This node acts as an intermediary between edge nodes and smart devices, and other fog nodes, facilitating seamless communication over WiFi networks. Whether connected through an intranet or the internet, the fog node ensures efficient data transmission, processing, storage and coordination, enabling robust interactions across the IoT ecosystem.

4.4 Node: Cloud Computing

In the experimental setup, server instances are deployed on SSD Nodes, which are standard Linux Ubuntu Servers. These instances support communication via MQTT and HTTP (RESTful web services) over the internet, enabling reliable and efficient data

and message transfer. To handle multiple server instances and distribute workloads efficiently, a load balancer is implemented using NGINX, ensuring optimal resource utilisation and system performance.

4.5 Node: Mobile Computing

Smartphones and laptops serve as versatile computing devices within the system. A smartphone equipped with the Android operating system and a custom-designed application, using standard interfaces, enables mobile computing with extensive connectivity options. The application facilitates communication over WiFi, 3G, or 4G networks, as well as Bluetooth, ensuring seamless data exchange between nodes and devices. With ample memory for data storage, sufficient computational power, and a built-in GPS sensor, the smartphone provides real-time location data and acts as a dynamic node within the IoT ecosystem. Similarly, laptops equipped with required applications enhance connectivity and computational capabilities, serving as central hubs for data monitoring, analysis, and control.

4.6 Features of Computing Nodes

4.6.1 Software Platform for Data Collection, Communication, and Storage

All computing nodes are equipped with a software stack that includes Apache Spring Boot (for Web Services APIs), MySQL Database, Mosquitto MQTT broker, Eclipse Paho MQTT Client Library, and Apache Tomcat. Each node runs a scaled-down version of the software, tailored to its computing capacity, storage space, and memory available for data processing.

4.6.2 Push and Pull Mechanism

All computing nodes utilise a pull mechanism through request-response over MQTT or HTTP. Additionally, each node has the capability to push messages via MQTT or Web Sockets.

4.6.3 Device Management

For the purpose of device and nodes Onboarding (bootstrapping) and Deboarding (unbootstrapping) the cloud maintains a registry of all devices and nodes, while each node must be provisioned with the devices or nodes it connects to, establishing a topology and trust among all participants.

4.6.4 Requests and Responses

All requests and responses use standard JSON formats. Each request or response, whether pull or push, is structured with the following elements: device identifier, action to be performed, source of the request, expected destination of the response, appropriate timestamp, and the payload.

5 EXPERIMENTS FOR APPLICATION DOMAINS

This section describes and provides details of the experiments performed, to evaluate the CCP, to apply the CCP in four applications. These are,

- Multi-location Data Center Cooling Management System
- Automotive Telematics - Driver & Vehicle Behaviour
- Smart Building - Environment, Fire Safety, Electrical Energy
- Asset Tracking - Track carts at shopping malls

Each application encompasses multiple use cases for measurement, monitoring, and control. As we implement IoT using CCP in these systems, we evaluate CCP against the required use cases: (a) Notifications for events (e.g., smoke, fire, asset loss, overspeeding), (b) Data collection for predictive health monitoring (e.g., cooling system optimisation, monitor driver behavior), and (c) Scalability and Extendability, dynamic addition of devices or configurations (e.g., adding smoke detectors or extinguishers to a building, or air quality monitors in a vehicle cabin). Our validation criteria is as follows:

- **VQ1:** Scalability of the system due to CCP: How scalable is the IOT system with CCP?
- **VQ2:** Consistent functioning of the Use Case: How reliably does the system perform the use case?
- **VQ3:** Unique advantage of CCP: What advantage does CPP provide for the IOT system?

All applications are built using the devices described in Section 4

5.1 Multi-Location Data Center Management System

5.1.1 Introduction

Data centers must maintain a specific temperature to ensure the optimal functioning of servers and other

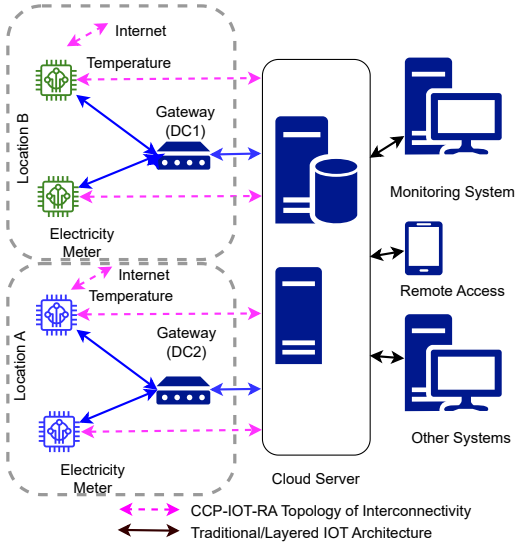

Figure 3: Multi-location Data Center Management System.

network equipment. In a typical setup, servers and equipment are installed in racks, and data centers contain multiple server or equipment racks. Temperature control within the acceptable range (typically 22 to 25°C) is achieved using air conditioners (ACs). The primary goal is to provide efficient cooling while minimising electricity consumption. To achieve this, temperature and electricity consumption are regularly monitored. Authors have described the use cases in detail in (Joshi and Deshpande, 2024a)

5.1.2 Experimental Set-Up

The architectural set-up of the system is shown in Figure 3. The set-up is such that the configuration can be updated to run the experiment in traditional IOT (Layered) architecture and in an IOT with CCP. A T10 and an N32 is used in the set-up along with the cloud server.

5.1.3 AC with IoT (Traditional Layered Approach)

As described by the authors in (Kaushik and Naik, 2023), the data for temperature and electricity consumption is collected periodically. This data is transmitted to the cloud server over the wired or wireless network interface. At the cloud node the data is preprocessed, and stored in the database. The data is also analysed, and based on the model of operation, commands are sent to the smart device to operate the AC system. Thus, switching based on a cooling model ensures optimal AC usage and electricity consumption.

During regular operations, the AC system is continuously monitored, and any fault codes are reported to the server. On occurrence of a fault, notification(s) are sent to the operator by the cloud server. Notifications are sent on email and/or web sockets.

Commands from the cloud must always be transmitted to the device for switching operations. Thus, in case of connectivity issues with the cloud, the commands were not received and the operation was affected.

5.1.4 AC with IoT with CCP

Now, we configure the system to include a smart gateway, capable of autonomous operation between the smart sensor and cloud. While the computation of the model for cooling continues to be computed on the cloud, the switching schedule was transmitted to the gateway which continued the operation using the switching times which were provided by the cloud server. Till the time that the cooling model does not change the entire switching operation was performed automatically by the smart gateway. Thus, there is an evident reduction on the computing resources of the cloud server. Further, smart gateway and cloud server establish an interplay; wherein any increase in the number of sensors or smart devices was handled by the gateway. Data so sent to the server included the new set of sensor devices. A mobile phone, with an app, was used to remotely monitor the system and receive alerts. With the smart sensor, even in event of gateway downtime, it is connected with the server and operation continued without any interruptions.

Thus, in conclusion the Multi-location Data Center Management System, was equipped to perform better, scalable and reliable even in anomalies of disconnected. System performed better with CCP IOT than with only IOT with cloud. The comparison is depicted in Table 3.

5.1.5 Conclusion

Based on the validation criteria, the assessment is:

- **VQ1:** Addition of multiple sensors or locations is handled by the CCP with the capacity shared between edge gateway and the cloud.
- **VQ2:** With the edge gateway, operations continue seamlessly without cloud connectivity, functioning autonomously until switching times require adjustment.
- **VQ3:** In the event of connectivity loss, the edge gateway continues to operate. If the edge gateway fails or experiences connectivity issues, the smart device connects directly to the cloud and sends failure notifications to the intended user via MQTT channels.

Table 3: Multi-Location Data Centers - IOT (Layered), IOT with CCP.

Design of cooling system	Temperature Control	Electricity Optimisation	Scalability
Without IoT	YES. Completely Localised	None	None
IoT (Layered)	YES. Cloud Controlled	YES. Cloud Controlled	YES. Need more capacity on the cloud with increase in locations. Mobile node only for remote monitoring.
IoT CCP	YES. Edge Gateway/Cloud Participation	YES. Edge/Cloud Participation	YES. Not all times Cloud Resources required as Edge Gateway takes the processing load. Mobile node participated for configuration changes.

5.2 Automotive Telematics: Vehicle & Driver Behaviour

5.2.1 Vehicle and Driver Behaviour

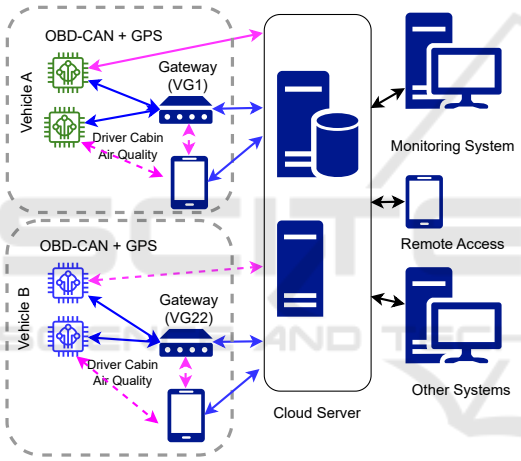

Figure 4: Automotive Telematics - CAN OBD.

5.2.2 Introduction

To enhance the safety of vehicles and drivers on the road, numerous IoT-based solutions have been implemented, utilising data collected from the On-Board Diagnostics (OBD) port. Key vehicle parameters such as speed, engine RPM, odometer readings, coolant temperature, fuel level, fuel consumption, and gear position are frequently gathered from the Electronic Control Units (ECU) for driver and vehicle behavior analysis. OBD fault codes offer additional insights into potential service or maintenance needs for the vehicle. Data is also collected from other sensors and smart devices, such as GPS and cabin environment, which provide critical information about the vehicle's location and driver's cabin environment. Using the data, additional attributes like hard braking, acceleration, deceleration, and compliance with speed limits can be calculated, offer valuable feedback to drivers. Fault codes and engine data further assists in assessing the vehicle's operational condition and overall fitness. Additionally, the collected data can be processed for predictive analytics to support preventive maintenance strategies. We refer to the work done by the authors in (Malekian and et al., 2015) (Türker and et al, 2016).

5.2.3 Experimental Set-Up

The architectural set-up of the system is shown in Figure 4. The set-up is such that the configuration can be updated to run the experiment in traditional IOT (Layered) architecture and in an IOT with CCP. A TX1 along with a N32 is used in the set-up with the cloud servers. An App, capable of connecting to the cloud server and also the smart device was used in the set-up.

5.2.4 Vehicle OBD with IoT (Traditional IoT and a Layered Approach)

Data fusion combines data from the CAN OBD, and GPS devices to capture vehicle parameters. This data is transmitted to the cloud for processing, where alerts for over-speeding, hard-braking, driver behavior, or vehicle faults are generated and sent to the driver's mobile device.

Since all processing occurs on the server, vehicle and driver behavior data must be computed in the cloud, leading to continuous use of server resources with no possibility of offloading.

5.2.5 Vehicle OBD with IoT with CCP

In the set-up a smart device, a mobile app and a gateway when operational provided capacity for computing at the edge. Over-speeding, hard-braking data was computed on the mobile phone connected to the smart CAN-OBD device with alerts to the driver in real-time.

Thus, with offloading of this computing the capacity and dependency on the cloud was removed. Local capacity to compute and collect data increased. When the cabin environment monitor was added, the same was seamlessly connected at the gateway. Thus, in this case, Mobile Computing, Fog Computing and Cloud computing were the paradigms participating in the overall application.

Data collected from all sources by all computing paradigms was ultimately sent to the cloud, where it was collected, processed and stored for future use, insights and analysis.

Thus, in conclusion the Automotive Telematics - Vehicle & Driver Behaviour, was equipped to perform better, scalable and reliable even in anomalies of disconnected. System performed better with CCP IOT than with only IOT with cloud. The comparison is depicted in Table 4.

5.2.6 Conclusion

Based on the validation criteria, the assessment is:

- **VQ1:** Addition of sensor devices can be handled by the CCP with the capacity shared between mobile phone, edge gateway and the cloud.
- **VQ2:** With the presence of the edge gateway and mobile app, the operation continues even in absence of the connectivity to the cloud. Till such a point that analytics are indeed required from the cloud, the functioning can continue autonomously at the edge using gateway and mobile phone.
- **VQ3:** In event of a connectivity loss, the edge gateway and mobile app continues to operate. With the smart device and mobile app connectivity to the internet and hence the cloud, the overall communication of events and data is possible.

5.3 Smart Building: Environment, Fire Safety, Electrical Energy

5.3.1 Introduction

IoT based solutions for smart buildings enable efficient monitoring and management of critical systems such as electricity consumption, fire safety equipment, and air quality. Smart sensors installed across the building track electricity usage in real time, identifying energy-intensive areas and providing insights to optimize consumption. Automated systems can adjust lighting, HVAC, and other electrical equipment based on occupancy or pre-set schedules, reducing energy waste. IoT-enabled smoke detectors continuously monitor for fire hazards and send instant alerts to building management or emergency

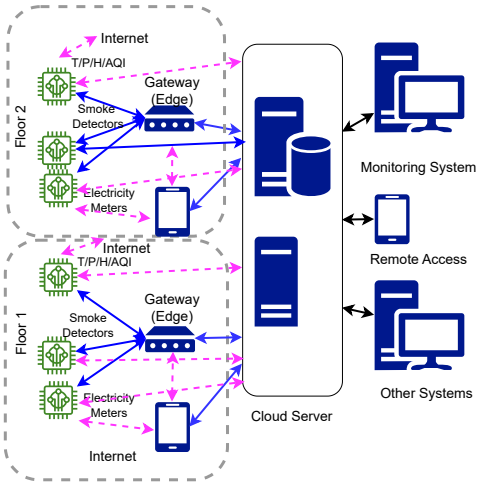

Figure 5: Smart Building - Env., Fire Safety, Elect. Energy.

services in case of anomalies. These devices can also perform self-diagnostics to ensure they are operational, improving fire safety reliability. Air quality sensors measure parameters like CO_2 levels, humidity, temperature, and the presence of pollutants, ensuring a healthy indoor environment. Data collected from these sensors can trigger ventilation systems to maintain optimal air quality. Centralised dashboards display real-time data, enabling facility managers to make informed decisions and respond promptly to issues. Predictive analytics, powered by IoT, can identify patterns helping with preventive maintenance of equipment and avoiding costly failures. Overall, IoT enhances energy efficiency, safety, and comfort in smart buildings, creating a sustainable and secure environment for occupants.

5.3.2 Experimental Set-Up

The architectural set-up of the system is shown in Figure 5. The set-up is such that the configuration can be updated to run the experiment in traditional IOT (Layered) architecture and in an IOT with CCP. A T10i, T10E and T10 (simulates the smoke detector) along with a N32 is used in the set-up with the cloud servers. An App, capable of connecting to the cloud server and also the smart device was used in the set-up.

5.3.3 Smart Building with IoT (Traditional IoT and a Layered Approach)

In the traditional set-up, all the sensors were sending the data to the cloud, we had, per floor, temperature, pressure, humidity, air-quality, smoke sensor and electricity consumption sent to the cloud. At all times, the data was compared with the parameter thresholds and the commands were issued to control example:

Table 4: Automotive Telematics (Fleet Management) - IOT (Layered), IOT with CCP.

Design of Vehicle/Fleet Mgmt.	Vehicle Behaviour	Driver Behaviour	Scalability
Without IoT	YES. Completely Localised	Yes. Completely Localised	None
IoT (Layered)	YES. Cloud Controlled	YES. Cloud Controlled	YES. Need more capacity on the cloud with increase in locations.
IoT CCP	YES. Edge Gateway/Cloud Participation	YES. Edge/Cloud Participation	YES. Not all times Cloud Resources required as Edge Gateway takes the processing load.

ON/OFF for HVAC were send by the cloud. Computation of the received data and control based on the processing of the data was done at the cloud.

Addition of devices to simulate floors or additional equipment added the computing load on the server. Since the processing was done only at the server end, all the processing was to be computed on the cloud. Thus, the server computing power is used continuously with no offloading possibility.

5.3.4 Smart Building with IoT with CCP

In the set-up a smart device, a mobile app, and a gateway, when operational, provided capacity for computing at the edge. Sensing and control based on ON/OFF for temperature were done autonomously by the gateway. The gateway continuously transmitted data to the cloud, where the air quality or temperature model was computed on the cloud server.

Offloading operations to the gateway reduced the cloud server's load and eliminated dependency on the cloud. A mobile app was used for configuration, verification, and updates. Field data was observed and correlated by facility management with server-analysed data. All smart devices, equipped with internet connectivity, could send notifications and messages about anomalies if links to the gateway or server were disrupted. For example, a smart smoke detector could directly notify the facility manager.

It is clear, from the experiment, that the cloud, edge and mobile computing in a collaborative manner, can be very effective and efficient. System performed better with CCP IOT than with only IOT with cloud. The comparison is depicted in Table 5.

5.3.5 Conclusion

Based on the validation criteria, the assessment is as follows:

- **VQ1:** With CCP addition of sensor devices can be handled effectively by sharing the capacity across mobile phone, edge gateway and the cloud.
- **VQ2:** With the edge gateway and mobile app, operations continue even without cloud connectivity. The system can function autonomously at the edge using the gateway and mobile phone until cloud-based analytics are needed. The mobile platform offers portability and real-time insights in the field.
- **VQ3:** In event of a connectivity loss, the edge gateway and mobile app continues to operate. With the smart device and mobile app connectivity to the internet and hence the cloud, the overall communication of events and data is possible.

5.4 Asset Tracking: Track Carts at Multiple Shopping Malls

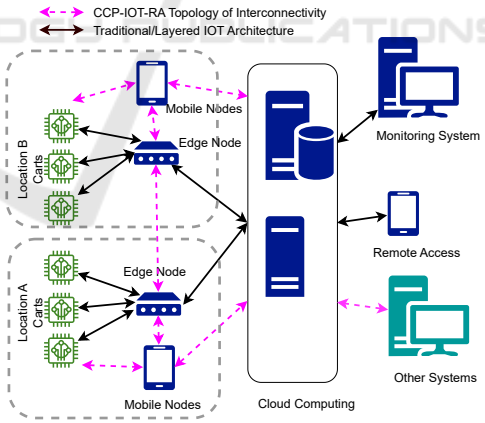

Figure 6: Asset Tracking System - Carts at mall(s).

5.4.1 Introduction

An asset tracking system, for carts at malls, utilises IoT devices with GPS to efficiently monitor and manage cart movement. Real-time data, and alerts enable staff to retrieve carts promptly, improving operational efficiency. Usage patterns are analyzed to identify high-demand areas, and optimize cart distribution. Maintenance alerts ensure carts remain in good

Table 5: Smart Building - IOT (Layered), IOT with CCP.

Design of BMS	Air Quality	Electricity	Fire Safety	Scalability
Without IoT	YES. Fully localised	YES. Fully localised	YES. Fully localised	None
IoT (Layered)	YES. Cloud based	YES. Cloud based	YES. Cloud based	YES. Need more capacity on the cloud with increase in locations.
IoT CCP	YES. Edge Gateway/Cloud/Mobile Participation	YES. Edge/Cloud/Mobile Participation	YES. Edge Gateway/Cloud/Mobile Participation	YES. Cloud resources are not always required, as the Edge Gateway handles the processing load.

Table 6: Asset Tracking (Carts in malls) - IOT (Layered), IOT with CCP.

Design of Cart Monitor	Cart Tracker	Scalability
Without IoT	YES. Totally Localised	None
IoT Only	YES. Cloud based	YES. Increased locations require additional cloud capacity.
IoT CCP	YES. Edge Gateway/Cloud/Mobile Participation	YES. Cloud resources are not always required, as the Edge Gateway handles the processing load.

condition, while a centralised dashboard allows seamless oversight of cart availability and performance.

5.4.2 Experimental Set-Up

The architectural set-up of the system is shown in Figure 6. The set-up is such that the configuration can be updated to run the experiment in traditional IOT (Layered) architecture, and in an IOT with CCP. A TX1 (CAN-OBD) is repurposed to track only the location with a fog node NmPC and a cloud server was used in the set-up.

5.4.3 Asset Tracking with IoT (Traditional IoT and a Layered Approach)

Fusion of beacon/ping of presence, and location information by GPS is done per cart. This data is then transmitted to the cloud for further processing. All alerts for number of carts at a location are generated by the cloud and then sent to the mobile app of the facility staff.

Since processing was done solely on the server, all cart data computations occurred in the cloud, resulting in continuous server resource usage with no option for offloading.

5.4.4 Asset Tracking with IoT with CCP

In the setup, a smart device, mobile app, and gateway, enabled edge computing. Beacon/pings and GPS location data were processed on the mobile phone connected to the smart device, providing real-time alerts to facility staff. This offloaded computing from the cloud, reducing dependency and increasing local capacity. An environment monitor was added. It seamlessly connected to the gateway, integrating Mobile Computing, Edge Computing, and Cloud Computing into the application. Data from all sources was ultimately transmitted to the cloud for storage and analysis, offering insights into cart usage by location.

In conclusion, the Asset Tracking system for shopping mall carts became more reliable even during disconnections. The CCP IOT performed better than traditional IoT with cloud-only reliance, as shown in Table 6.

5.4.5 Conclusion

In conclusion,

- **VQ1:** The CCP efficiently manages the addition of sensor devices by distributing capacity across the mobile phone, edge gateway, and cloud.
- **VQ2:** The edge gateway and mobile app ensure uninterrupted operation, even without cloud connectivity, while mobile platforms offer portability and real-time insights.
- **VQ3:** In case of connectivity loss, the edge gateway and mobile app maintain functionality, and upon internet restoration, smart devices facilitate data and event communication with the cloud.

6 LESSONS LEARNED

The design and experimentation with the CCP-based IoT RA have yielded valuable insights into the architectural requirements of dynamic IoT environments:

- **Computation as the Core of Architecture:** A key takeaway is the importance of placing computation at the core of IoT architectural design. Designing with computational efficiency and adaptability in mind ensures that system components are aligned to perform optimally under varying workloads. The Collaborative Computing Paradigms (CCP)-based architecture demonstrated that organising computational tasks across Edge, Fog, and Cloud is critical for meeting the demands of modern IoT systems. This includes enabling real-time processing, adaptive workload distribution, and dynamic resource allocation. Future IoT architectures must be conceived with computation as the organising principle, driving decisions around hardware selection, data flow management, and system scalability.
- **Use Cases as the Driving Force Behind Architectural Design:** Another critical insight is the centrality of use cases in guiding architectural decisions. The experiments highlighted that designing IoT systems with specific ”vertical” applications—such as building automation, telematics, or fire safety—at the forefront ensures that the architecture aligns with the functional and performance requirements of real-world scenarios. Starting with the use cases ensures the selection of appropriate components, computing paradigms, and resource allocation strategies. This user-driven approach prevents overengineering while ensuring that the architecture is tailored to meet practical demands.
- **Leveraging System on Chip (SoC) for Retrofitting Existing Solutions:** When integrating CCP into existing IoT systems, the use of advanced System on Chip (SoC) devices emerged as a practical and efficient solution. The high-performance capabilities, energy efficiency, and protocol support offered by modern SoCs allow them to retrofit and upgrade legacy systems with minimal disruption. By facilitating edge analytics, real-time data processing, and seamless communication across computational layers, SoCs bridge the gap between outdated architectures and advanced frameworks like CCP. This learning underscores the value of SoC-enabled nodes in transforming existing IoT deployments into more scalable, responsive, and efficient systems.
- **Use of Standardized Technologies Enhances Interoperability:** The adoption of standardised technologies played a crucial role in ensuring interoperability and ease of integration across diverse components in the IoT ecosystem. By leveraging widely accepted protocols and frameworks, such as MQTT, HTTP REST, and WiFi-based connectivity, the architecture facilitated seamless communication and coordination between devices, computational layers, and external systems. This learning highlights the importance of building IoT solutions with standardisation at their core, as it not only simplifies system implementation but also ensures compatibility and scalability in complex and heterogeneous environments. Systems designed around standardised technologies are inherently more robust and adaptable, allowing for smoother integration with both existing infrastructure and future advancements.

We developed a scalable, extensible, and flexible IoT solution by reimagining traditional system design. With CPP, we moved beyond the limitations of layered architectures, and embraced a computation-centric perspective, prioritised real-world use cases, utilised advanced System on Chip technologies, and focused on architectural innovation. We learned that achieving such a solution requires balancing dynamic resource allocation, adaptability to varying workloads, and seamless integration of new devices and services while maintaining a focus on interoperability and scalability. Our experiments emphasised the need to break traditional boundaries in order to design IoT systems that meet the evolving demands of modern, interconnected environments.

We believe these learnings empower designers to shape next-gen IoT solutions driven by CCP.

7 CONCLUSION

This study validates the Collaborative Computing Paradigms (CCP)-based IoT Reference Architecture as a superior alternative to traditional layered architectures for dynamic IoT environments. By incorporating critical characteristics such as interconnection and interplay between computational paradigms, dynamic processing distribution, computational fluidity and scalability and extensibility, the CCP architecture addresses key limitations of layered models, including rigidity, high latency, and suboptimal resource utilisation.

Experimental evaluations in Building Automation (fire safety systems, air quality monitoring, and HVAC control) and Telematics applications highlight

measurable performance improvements with the CCP architecture. Specifically, it demonstrated enhanced latency reduction, adaptive workload distribution possibilities, real-time analytics capabilities at the edge. These results underscore its ability to dynamically optimize resource allocation and decision-making processes across heterogeneous and fluid IoT environments. There will also be a need to quantify the results in terms of performance parameters.

Thus, the CCP-based architecture offers a foundational enhancement, and a fundamentally different approach to IoT system design, moving beyond the constraints of static, layered frameworks. By enabling highly distributed, context-aware, and collaborative processing, this architecture establishes itself as a viable and scalable framework for next-generation IoT deployments. Future work may focus on extending the CCP paradigm to broader domains, further validating its potential to redefine IoT architectures in increasingly complex and data-driven scenarios.

8 FUTURE WORK

Building on the demonstrated advantages of the CCP-based IoT Reference Architecture, future research will focus on extending its applicability across diverse IoT domains and industry verticals. Customized models will be developed to address domain-specific requirements in areas such as smart cities, healthcare, industrial automation, agriculture, and energy management. Each vertical presents unique challenges—ranging from real-time decision-making and scalability, to data security, and compliance—that will require adapting the CCP framework to meet operational demands.

Additionally, the integration of AI and ML within CCP architectures will be a major focus. By embedding predictive analytics, anomaly detection, and adaptive resource allocation capabilities, CCP-based systems can evolve into intelligent ecosystems capable of self-optimisation. Research will also explore the deployment of CCP, in ultra-low latency and mission-critical applications, such as autonomous vehicles, robotics, and remote healthcare, to assess its suitability for high-stakes environments.

Finally, standardising CCP models for various industries, enhancing security and privacy mechanisms, and creating simulation tools for validation and benchmarking will drive broader adoption. Future studies will also evaluate the scalability and performance of CCP architectures in globally distributed IoT environments, ensuring their readiness for large-scale, heterogeneous deployments. These efforts aim to solidify CCP as a cornerstone for next-generation IoT systems, delivering adaptive, secure, and highly efficient solutions for increasingly complex interconnected ecosystems.

ACKNOWLEDGMENTS

We sincerely thank Leap&Scale®, India, for providing access to their Sensoyo® IoT Platform and IoT devices, which facilitated our experiments and tests.

REFERENCES

Bass, L., Clements, P., and Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley Professional, 3rd edition.

Ericsson (2024). Iot connections forecast - accessed on 10 dec 2024. In *https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook*.

Espressif (2024). Esp32 wifi & bluetooth soc - accessed on 10 dec 2024. In *https://www.espressif.com*.

H. Washizaki, e. (2024). Guide to the software engineering body of knowledge (swebok guide), version 4.0. In *SEWBOK, IEEE Computer Society*.

Joshi, P. and Deshpande, B. (2024a). Collaborative computing paradigms: A software systems architecture for dynamic iot environments. In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering - Volume 1: MODELSWARD*, pages 297–306. INSTICC, SciTePress.

Joshi, P. and Deshpande, B. (2024b). A reference architecture for dynamic iot environments using collaborative computing paradigms (ccp-iot-ra). In *Proceedings of the 19th International Conference on Software Technologies - Volume 1: ICSOFT*, pages 193–202. INSTICC, SciTePress.

Kaushik, K. and Naik, V. (2023). Making ductless-split cooling systems energy efficient using iot. In *International Conference on Communication Systems*. COMSNETS.

Malekian, R. and et al. (2015). Design and implementation of a wireless obd ii fleet management system. In *IEEE Sensors Journal, vol. 17, no. 4, pp. 1154-1164*. IEEE.

Pi, R. (2024). Products, accessed on 10 dec 2024. In *https://www.raspberrypi.com/products/*.

Sensortec, B. (2024). Products - accessed on 10 dec 2024. In *https://www.bosch-sensortec.com/products/*.

Türker, G. F. and et al (2016). Survey of smartphone applications based on obd-ii for intelligent transportation systems. In *International Journal of Engg Research and Applications*. IJERA.