Unit- 4

Solution framework for IOT APPLICATIONS :

An IoT solution framework provides a structured approach to building and managing IoT applications. It typically involves several key components:

1.Device Layer:

Sensors and Actuators: These are the physical components that collect data (sensors) or perform actions (actuators) in the real world.

Connectivity: This refers to the technology used to connect devices to the network, such as Wi-Fi, Bluetooth, cellular, or LoRa.

2.Network Layer:

Connectivity Protocols: These are the standards that govern communication between devices and the network, such as MQTT, HTTP, or CoAP.

Network Infrastructure: This includes the physical infrastructure (e.g., routers, gateways) and the network protocols used to connect devices to the cloud.

Network Security: This involves protecting the network from unauthorized access and data breaches.

3.Data Ingestion Layer:

Data Collection: This involves collecting data from IoT devices and storing it in a suitable format.

Data Pre-processing: This involves cleaning, filtering, and transforming data to prepare it for analysis.

Data Storage: This refers to the storage of IoT data in various formats, such as time-series databases, NoSQL databases, or data lakes.

4.Data Processing Layer:

Data Analytics: This involves applying various analytical techniques to extract insights from IoT data, such as machine learning, data mining, and statistical analysis.

Data Visualization: This involves presenting data in a visual format to facilitate understanding and decision-making.

5.Application Layer:

IoT Applications: These are the specific software applications that utilize IoT data to provide value to users, such as smart home systems, industrial automation, or healthcare monitoring.

User Interface: This is the interface through which users interact with IoT applications.

6.Security Layer:

Authentication and Authorization: This involves ensuring that only authorized users can access IoT data and applications.

Data Privacy: This involves protecting sensitive data from unauthorized access and disclosure.

Security Best Practices: This includes following industry best practices for IoT security, such as using strong encryption, regular updates, and vulnerability management.

7.Management Layer:

IoT Platform: This is a centralized platform that provides tools for managing IoT devices, data, and applications.

Device Management: This involves managing the configuration, updates, and security of IoT devices.

Data Management: This involves managing the storage, processing, and analysis of IoT data.

By following a structured IoT solution framework, organizations can ensure that their IoT applications are scalable, secure, and deliver the desired value.

Implementation of device integration :

There are several standards for device integration, especially for consumer and industry devices. That is, these standards are compactly and constantly optimized toward integrating a wide variety of distributed, decentralized, and disparate devices. However, the ultimate target is to establish smarter environments that readily link cross domain automation modules.

Hugely successful SOA paradigm is being Tweaked toward service-oriented device architecture (SODA) to enable service-based device integration. Devices and machines need to be instrumented and interconnected with one another In order to communicate and collaborate to showcase intelligent behaviour. Device integration Empowers devices to share their unique service capabilities with one another. Service requests And responses are getting fulfilled through SODA.

Device integration using Service-Oriented Device Architecture (SODA) in IoT revolves around treating each device function as a modular service that can interact with other services.

By following steps we can integrate devices using SODA.

1.Service Abstraction: Each device is abstracted as a service provider.

2.Registration: When devices are installed in the building, they register their services with the Central Control System.

3.Dynamic Discovery: The Central Control System discovers all registered services, allowing it to understand which devices are available and what functions they offer.

4.Inter-Service Communication: Devices communicate using standard protocols like MQTT or HTTP.

5.Event-Driven Actions: Services react to events. If the motion sensor detects movement (event), it can trigger the security cameras to start recording.

6.Orchestration: The Central Control System orchestrates the interactions. It ensures that services integrate data from all devices to provide a cohesive view.

7.Flexibility and Scalability: New devices can be easily integrated with existing services.

Example : Soil Moisture Sensors send soil data to the Central Control System. The Central Control System checks weather forecasts from the Weather Stations. If no rain is expected, it triggers the Irrigation System to start watering. The Irrigation System starts watering and updates the Central Control System. If rain is expected, the Central Control System stops the Irrigation System to conserve water.

SODA is an emerging concept for the device world. There are several standards.

1.WSDL (Web Services Description Language): This standard provides a machine-readable description of web services. It outlines how to call the service, the parameters required, and the structure of the responses.

2.SOAP (Simple Object Access Protocol): SOAP is a protocol for exchanging structured information in web services via XML. It ensures messages are sent and received consistently and securely.

3.RESTful (Representational State Transfer): REST is an architectural style that uses standard HTTP methods like GET, POST, PUT, DELETE for communication. It’s known for being lightweight and scalable, ideal for stateless interactions in web services.

4.BPEL (Business Process Execution Language): BPEL is used for orchestrating multiple web services. It defines business processes that interact with various web services, specifying how they’re composed to achieve a particular task.

5.WS-Security (Web Services Security): This standard adds security to SOAP messages. It provides mechanisms for ensuring message integrity, confidentiality, and authentication.

DPWS is a device-specific and SODA-compliant standard that comprehensively addresses discovery, description, security, and control of devices and services on local as Well as remote networks. DPWS (Devices Profile for Web Services) is a OASIS standard that enables web services on smart and resource-constrained devices, which are fundamental to IOT. It provides a lightweight and efficient way to integrate devices into the web, ensuring seamless communication and interoperability.

<u>Data acquisition and integration :</u>

<u>Data acquisition</u>

Data acquisition in IoT applications involves collecting data from various sensors and devices, processing it, and then transmitting it to a central system for analysis.

Data gathering refers to data acquisition from the devices/devices network. Four modes of Gathering data are:

1.Polling : refers to the data sought from a device by addressing the device. For example, Waste container filling information in a waste management system.

2.Event-based gathering : refers to the data sought from the device on an event. For example A motion sensor detects movement and sends data to the security system only when motion is detected.

3.Scheduled interval : refers to the data sought from a device at select intervals. For example, Soil moisture sensors in a smart farm send readings every hour to a central control system to monitor soil conditions.

4.Continuous monitoring refers to the data sought from a device continuously. For example Wearable fitness trackers continuously monitor heart rate and activity levels, sending data in real-time to a connected app.

Application can configure sending of data after filtering or enriching at the gateway At the data-adaptation layer. The gateway in-between application and the devices can Provision for one or more of the following functions—transcoding, data management and Device management. Data management may be provisioning of the privacy and security, And data aggregation, compaction and fusion.

Aggregation refers to the process of joining together present and previously received Data frames.

Compaction means making information short without changing the meaning or context; For example, transmitting only the incremental data so that the information sent is short.

Fusion means formatting the information received in parts through various data frames And several types of data (or data from several sources), removing redundancy in the Received data.

<u>Data integration</u>

Data integration refers to the process of combining and harmonizing data from multiple sources into a unified, coherent format that can be put to use for various analytical, operational and decision-making purposes.

Data integration involves a series of steps and processes that brings together data from disparate sources and transforms it into a unified and usable format. Here's an overview of how a typical data integration process works:

1.Data source identification: The first step is identifying the various data sources that need to be integrated.

2.Data extraction: Next, data is extracted from the identified sources using extraction tools or processes.

3.Data mapping: Different data sources may use different terminologies, codes or structures to represent similar information. Creating a mapping schema that defines how data elements from different systems correspond to each other ensures proper data alignment during integration.

4.Data validation and quality assurance: Validation involves checking for errors, inconsistencies and data integrity issues to ensure accuracy and quality. Quality assurance processes are implemented to maintain data accuracy and reliability.

5. Data transformation: At this stage, the extracted data is converted and structured into a common format to ensure consistency, accuracy and compatibility. This might include data cleansing, data enrichment and data normalization.

6. Data loading: Data loading is where the transformed data is loaded into a data warehouse or any other desired destination for further analysis or reporting.

7. Data synchronization: Data synchronization helps ensure that the integrated data is kept up to date over time, via periodic updates.

8. Data governance and security: When integrating sensitive or regulated data, data governance practices ensure that data is handled in compliance with regulations and privacy requirements.

9.Metadata management: Metadata, which provides information about the integrated data, enhances its discoverability and usability so users can more easily understand the data's context, source and meaning.

10.Data access and analysis: Once integrated, the data sets can be accessed and analyzed using various tools. This analysis leads to insights that drive decision making and business strategies.

Device storage :

Device storage in IoT applications refers to the memory capacity within an IoT device, used to store data collected from sensors and other inputs. This storage can be volatile (like RAM) or non-volatile (like flash memory), and it plays a crucial role in ensuring the device can operate independently and store data when there's no immediate internet connection available.

Example: Imagine a smart home security system. Each security camera in the system has internal storage where it records video footage. If the internet connection goes down, the cameras can still record and save footage locally. Once the connection is restored, the data can be uploaded to the cloud for remote access and analysis. This local storage ensures continuous operation and data retention even during network interruptions.

Structured and unstructured data

The sensed data needs to be stored and continually updated in memory locations for further processing. As per requirement, a user has the option of utilizing the sensed information as either structured or unstructured data.

These are typically text data that have a pre-defined structure. Structured data Are associated with relational database management systems (RDBMS). These are Primarily created by using length-limited data fields such as phone numbers, social Security numbers, and other such information. Even if the data is human or machine generated, these data are easily searchable by querying algorithms as well as human generated queries. Common usage of this type of data is associated with flight or Train reservation systems, banking systems, inventory controls, and other similar Systems. Established languages such as Structured Query Language (SQL) are used For accessing these data in RDBMS. However, in the context of IoT, structured data Holds a minor share of the total generated data over the Internet.

In simple words, all the data on the Internet, which is not structured, is categorized as Unstructured. These data types have no pre-defined structure and can vary according To applications and data-generating sources. Some of the common examples of Human-generated unstructured data include text, e-mails, videos, images, phone recordings, chats, and others [2]. Some common examples of machine-generated Unstructured data include sensor data from traffic, buildings, industries, satellite Imagery, surveillance videos, and others. As already evident from its examples, this Data type does not have fixed formats associated with it, which makes it very difficult For querying algorithms to perform a look-up. Querying languages such as NoSQL Are generally used for this data type.

<u>Unstructured data storage on cloud/local server :</u>

<u>On cloud</u>

In the realm of IoT, a vast amount of unstructured data is generated from diverse sources like sensors, cameras, and other devices. This data, often in the form of images, videos, audio recordings, and text logs, lacks a predefined structure, making it challenging to store and analyze using traditional relational databases.

Cloud platforms offer scalable and cost-effective solutions for storing and managing massive volumes of unstructured IoT data. Here are some popular cloud-based storage options:

1. Object Storage:

Scalability: Easily scale storage capacity to accommodate growing data volumes.

Durability: Ensures data durability through redundant storage and automatic backups.

Accessibility: Provides global access to data from anywhere.

Examples: Amazon S3, Google Cloud Storage, Azure Blob Storage

2. Data Lakes:

Centralized Repository: Stores raw, unstructured data from various sources.

Schema-on-Read: Allows flexible data analysis and processing.

Big Data Analytics: Supports advanced analytics techniques like machine learning and AI.

Examples: AWS Lake Formation, Azure Data Lake Storage

3. NoSQL Databases:

Flexible Schema: Accommodates diverse data structures.

High Performance: Handles high-throughput, real-time data ingestion and retrieval.

Scalability: Easily scales to meet increasing data demands.

Examples: MongoDB, Cassandra, Couchbase

Example: Smart City Application

Consider a smart city application where IoT devices like cameras, sensors, and smart meters generate a massive amount of unstructured data.

Gemini was just updated. See update

Example: Smart City Application

Consider a smart city application where IoT devices like cameras, sensors, and smart meters generate a massive amount of unstructured data. This diverse set of unstructured data can be efficiently stored and analyzed using cloud-based storage solutions. For instance:

Video footage captured by surveillance cameras can be stored in object storage like Amazon S3 for easy retrieval and analysis.

Sensor data collected from environmental sensors to measure air quality, noise levels can be ingested into a data lake for long-term storage and batch processing.

Meter readings collected from smart meters to measure energy consumption can be stored in a NoSQL database for real-time analysis and visualization.

By effectively managing and analyzing this unstructured data, city planners and policymakers can make informed decisions to improve urban infrastructure, optimize resource allocation, and enhance citizen experiences.

<u>On local server</u>

Storing unstructured data on local servers for IoT applications involves using various hardware and software to manage the vast and varied data generated by IoT devices. The steps are following

Data Collection

IoT Devices: Sensors, cameras, and other IoT devices generate data continuously.

Protocols: Data is transmitted using protocols like MQTT, HTTP, or CoAP.

Edge Computing

Edge Devices: Initial data processing happens at the edge, close to the data source. This reduces latency and bandwidth usage.

Local Storage on Edge: Data is temporarily stored on edge devices before being sent to central servers.

Network Attached Storage (NAS)

NAS Systems: These devices are connected to a local network and provide shared storage that multiple users and devices can access.

File Systems: NAS uses file systems like NFS (Network File System) or SMB (Server Message Block) to manage the stored data.

On-Premises Servers

Local Servers: Physical servers located within a facility store and process unstructured data over the long term.

Data Management Software: Tools like Hadoop or Elasticsearch manage and analyze unstructured data on these servers.

Data Management

Metadata: Use metadata to tag and organize data, making it easier to search and retrieve.

Directory Structures: Organize data using logical directories to enhance data management.

Data Synchronization

Hybrid Solutions: Combine local and cloud storage solutions for backup and additional processing. Tools like AWS Storage Gateway or Azure Stack facilitate this.

Regular Backups: Ensure data redundancy and durability by synchronizing local data with cloud storage periodically.

Example: Smart Agriculture

Data Collection: Sensors collect data on soil moisture, temperature, and weather conditions.

Edge Processing: Initial data processing is done on edge devices to filter and analyze data in real-time.

Local Storage: Data is stored on a NAS system for immediate access by farmers. This includes both processed data and raw data.

Data Management: Data is organized using metadata tags and directory structures to facilitate easy retrieval.

Local Analysis: Farmers use local servers to run analytics software, identifying patterns and making data-driven decisions.

Synchronization with Cloud: Data is periodically synchronized with cloud storage for long-term storage and advanced analytics, ensuring data durability and enabling deeper insights using machine learning models.

<u>Authentication and Authorization of devices</u>

<u>Authentication</u>

Authentication of devices in the Internet of Things (IoT) is crucial for ensuring security and trust in interconnected systems.

1. Types of Authentication Mechanisms

a. Password-Based Authentication

Devices use a username/password pair.

Vulnerable to attacks (e.g., brute force, phishing).

Best for low-risk environments.

b. Certificate-Based Authentication

Devices are issued digital certificates by a trusted certificate authority (CA).

Uses Public Key Infrastructure (PKI) to establish trust.

Ensures the authenticity and integrity of devices.

c. Token-Based Authentication

Uses tokens (like JSON Web Tokens) to validate device identity.

Tokens are usually short-lived, reducing risks of interception.

Allows stateless sessions, ideal for resource-constrained devices.

d. Biometric Authentication

Some IoT devices use biometric data (like fingerprints) for user verification.

Applicable primarily in consumer devices (e.g., smart locks).

2. Authentication Protocols

a. OAuth 2.0

Delegates access, allowing devices to interact with APIs securely.

Issues tokens for authentication rather than sharing passwords.

b. MQTT with Authentication

A lightweight messaging protocol that can use username/password or certificates.

Ideal for constrained devices and low-bandwidth scenarios.

3. Device Identity Management

a. Unique Device Identification

Every device is assigned a unique identifier (e.g., MAC address, UUID).

Facilitates tracking and management in large networks.

b. Device Provisioning

The process of securely adding devices to a network.

Often involves assigning credentials and establishing trust relationships

4. Key Management

a. Symmetric vs. Asymmetric Keys

Symmetric keys are shared among devices but pose challenges for distribution.

Asymmetric keys (public/private pairs) enhance security but require more processing power.

b. Key Rotation and Renewal

Regularly updating cryptographic keys to minimize risks from compromised credentials.

Essential for long-term security in IoT ecosystems.

5. Mutual Authentication

Both the device and the server verify each other’s identities.

<u>Authorization</u>

Authorization is the next key layer after authentication in IoT security. Once a device’s identity is verified, authorization determines what the device is allowed to do within the network.

Authorization in IoT involves granting permissions to authenticated devices to access resources, services, or data based on predefined policies.

Key Components of Authorization

Access Control Policies: Define what actions a device can perform, such as reading data, sending commands, or configuring settings.

Roles and Permissions: Assign roles (e.g., admin, user) to devices, with each role having specific permissions.

Access Control Lists (ACLs): Lists that define which devices have access to specific resources or services.

Common Authorization Methods

Role-Based Access Control (RBAC)

How it Works: Devices are assigned roles, and each role has predefined permissions.

Advantages: Simplifies management by grouping permissions.

Challenges: Requires careful role definition to prevent privilege escalation.

Attribute-Based Access Control (ABAC)

How it Works: Access decisions are based on attributes (e.g., device type, location, time).

Advantages: Highly flexible, dynamic access control.

Challenges: Complex to implement and manage.

Policy-Based Access Control (PBAC)

How it Works: Uses policies to define rules for access.

Advantages: Flexible and can enforce complex access rules.

Challenges: Requires robust policy management and enforcement mechanisms.