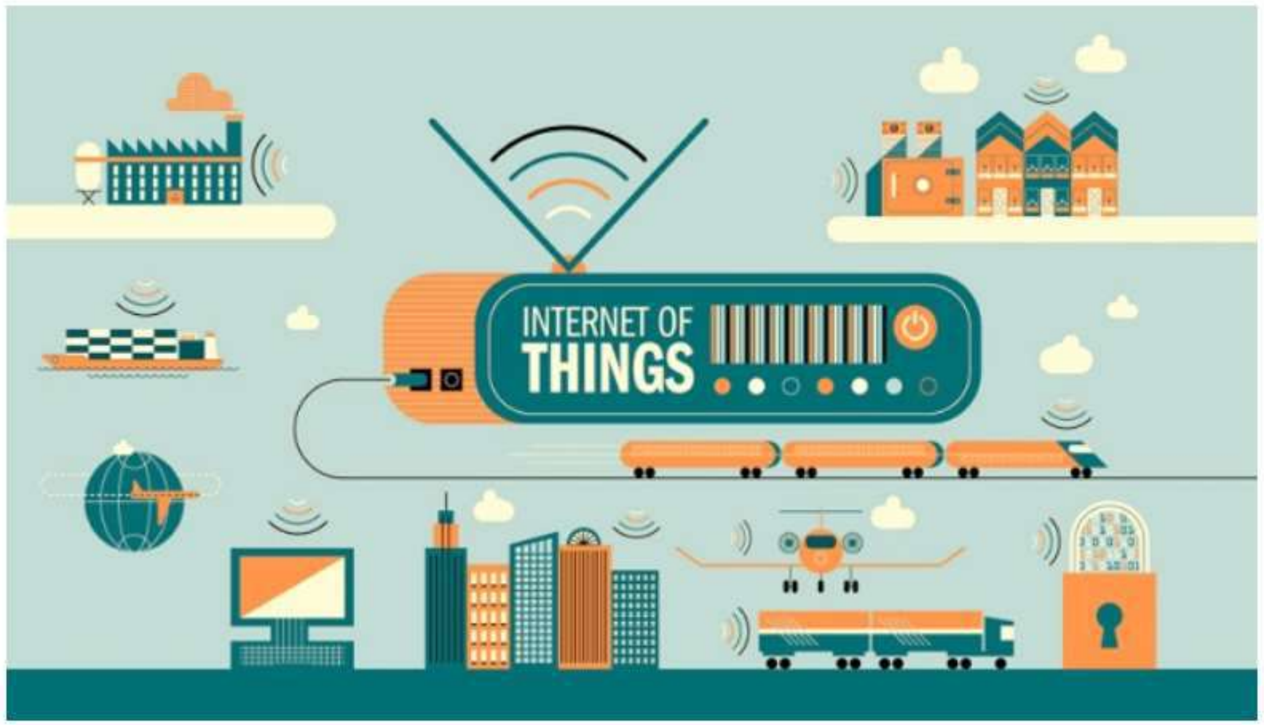
REVA UNIVERSITY



Prof Raghavendran T S
Asst Prof, School of C&IT, Reva University, Bengaluru
Raghavendran.ts@reva.edu.in

B22EK0601

UNIT -3

Data Analytics in IoT

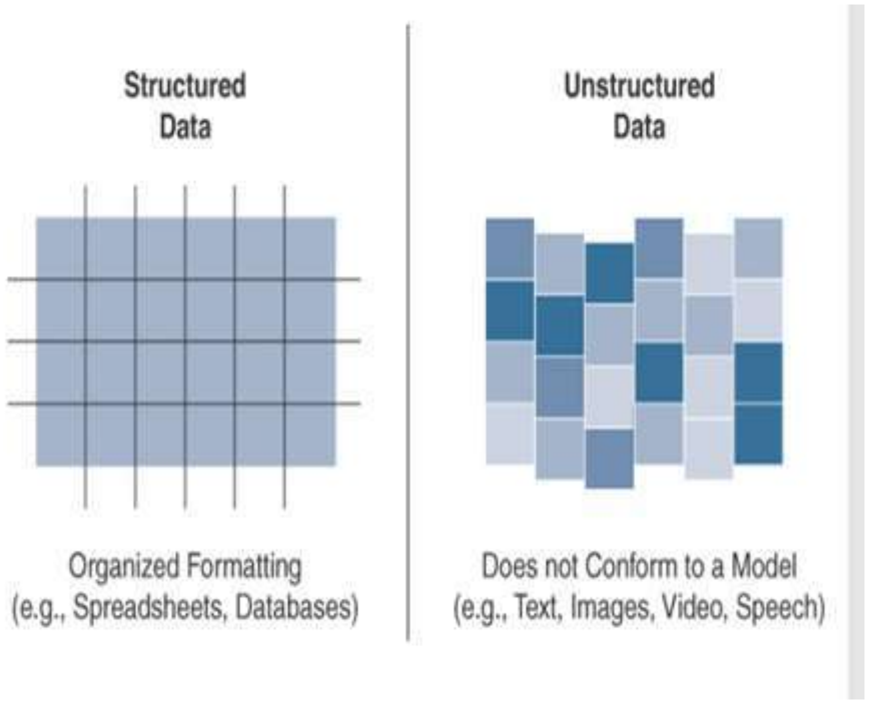
Sensors data from IoT device
Data Analytic
Data Monitoring Applications

Why Data Analytics in IOT?

- One of the biggest challenges in IoT: – Management of massive amounts of data generated by sensors.
- Modern jet engines are fitted with thousands of sensors that generate a whopping 10GB data per second
- A twin engine commercial aircraft with these engines operating on average 8 hours a day will generate over 500TB data daily, and this is just the data from the engines!

Commercial Jet Engine

Types of Data in Data Analytics


Structured Data
Unstructured Data
Organized Formatting (e.g., Spreadsheets, Databases)
Does not Conform to a Model (e.g., Text, Images, Video, Speech)

Structured data :– data follows a model/schema – defines data representation – easily formatted, stored, queried, and processed– e.g. Relational Database Model

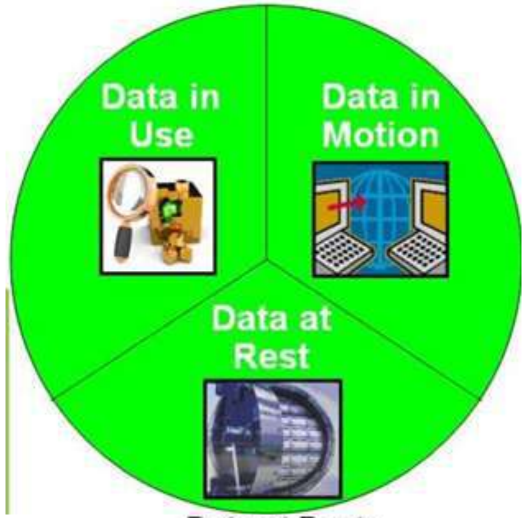
- Has been core type of data used for business decisions
- Wide array of data analytics tools are available

Unstructured data:– lacks of logical schema– Doesn't fit into predefined data model– e.g. text, speech, images

Semi-structured data: − hybrid of structured and unstructured data − Not relational, but contains a certain schema − e.g. Email message: fields are well defined, but body and attachments are unstructured. Other examples include JavaScript Object Notation (JSON) and Extensible Markup Language (XML), which are common data inter change formats used on the web and in some IoT data exchanges.

Different States of Digital Data

Different states of digital data can be
–in transit(data in motion)
–being held/stored(data at rest)
–being processed(data in use)



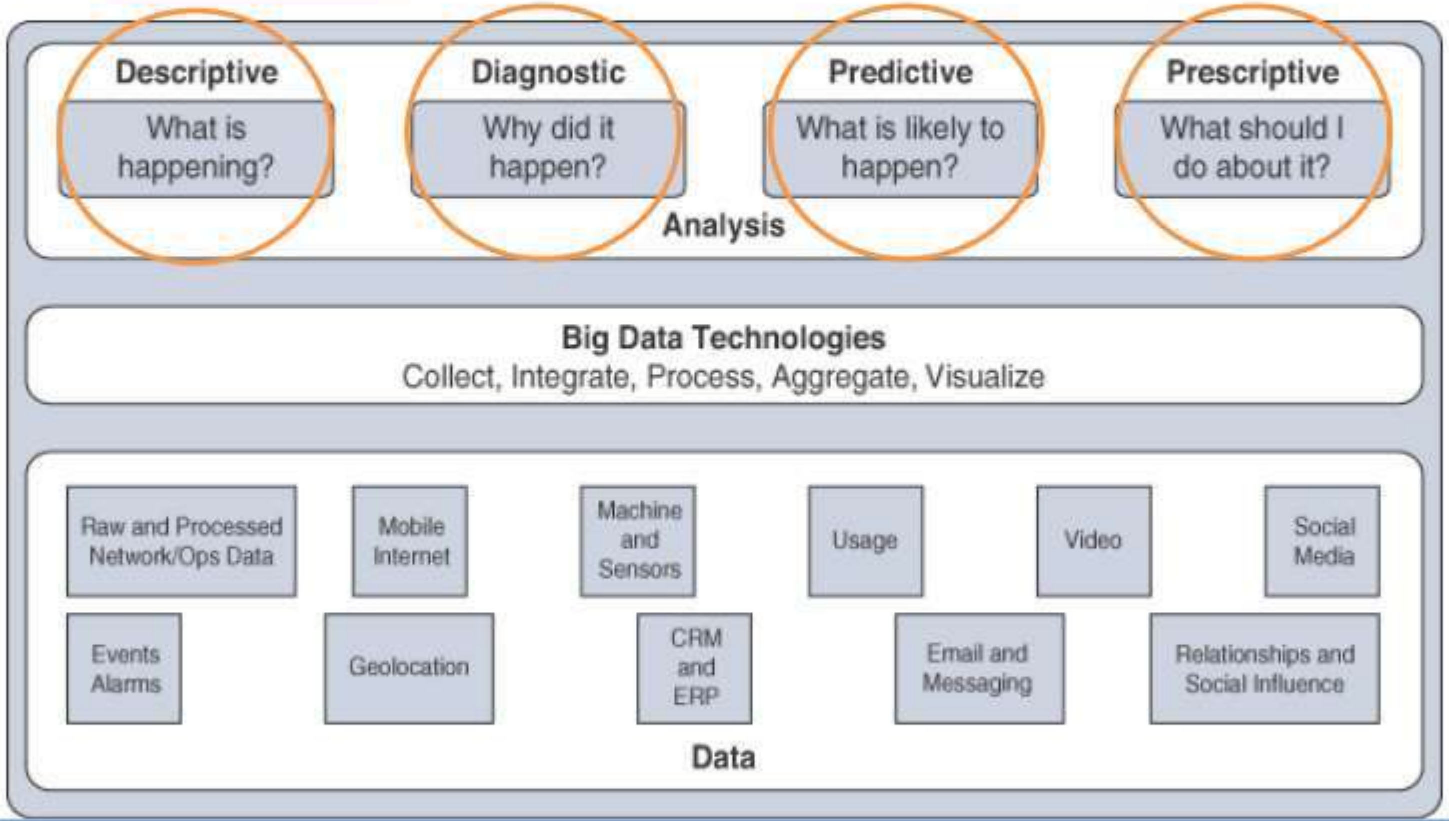
Data in Use: Active data under constant change stored physically in databases, data warehouses, spreadsheets etc.

Data in Motion: Data that is traversing a network or temporarily residing in computer memory to be read or updated.

Data at Rest: Inactive data stored physically in databases, data warehouses, spreadsheets, archives, tapes, off-site backups etc.

Types of Data Analytics

- Data analysis is typically broken down by
  - the types of results that are produced.



1) Descriptive – It tells you what is happening, either now or in the past.
• e.g., thermometer in a truck engine reports temperature values every second.
2) Diagnostic – It can provide the answer to "why" it has happened
• e.g. why the truck engine failed

3) Predictive – It aims to foretell problems or issues before they occur.
• e.g., it could provide an estimate on the remaining life of the truck engine.

4) Prescriptive – It goes a step beyond predictive and recommends solutions for upcoming problems.
• e.g. it might calculate various alternatives to cost-effectively maintain our truck.

IoT Data Analytics - Challenges

1) IoT data places two specific challenges on relational database data:–

a) **Scaling problems**:
- large number of smart objects continually send data,
- relational databases grow incredibly large very quickly.
- Results in performance issues which is costly to resolve.

b) **Volatility of data:**
- In RDBMS, schema is designed from the beginning,
- changing the scheme later creates problem.
- IoT data is volatile in the sense that the data model is likely to change and evolve over time.
- A dynamic schema is often required.

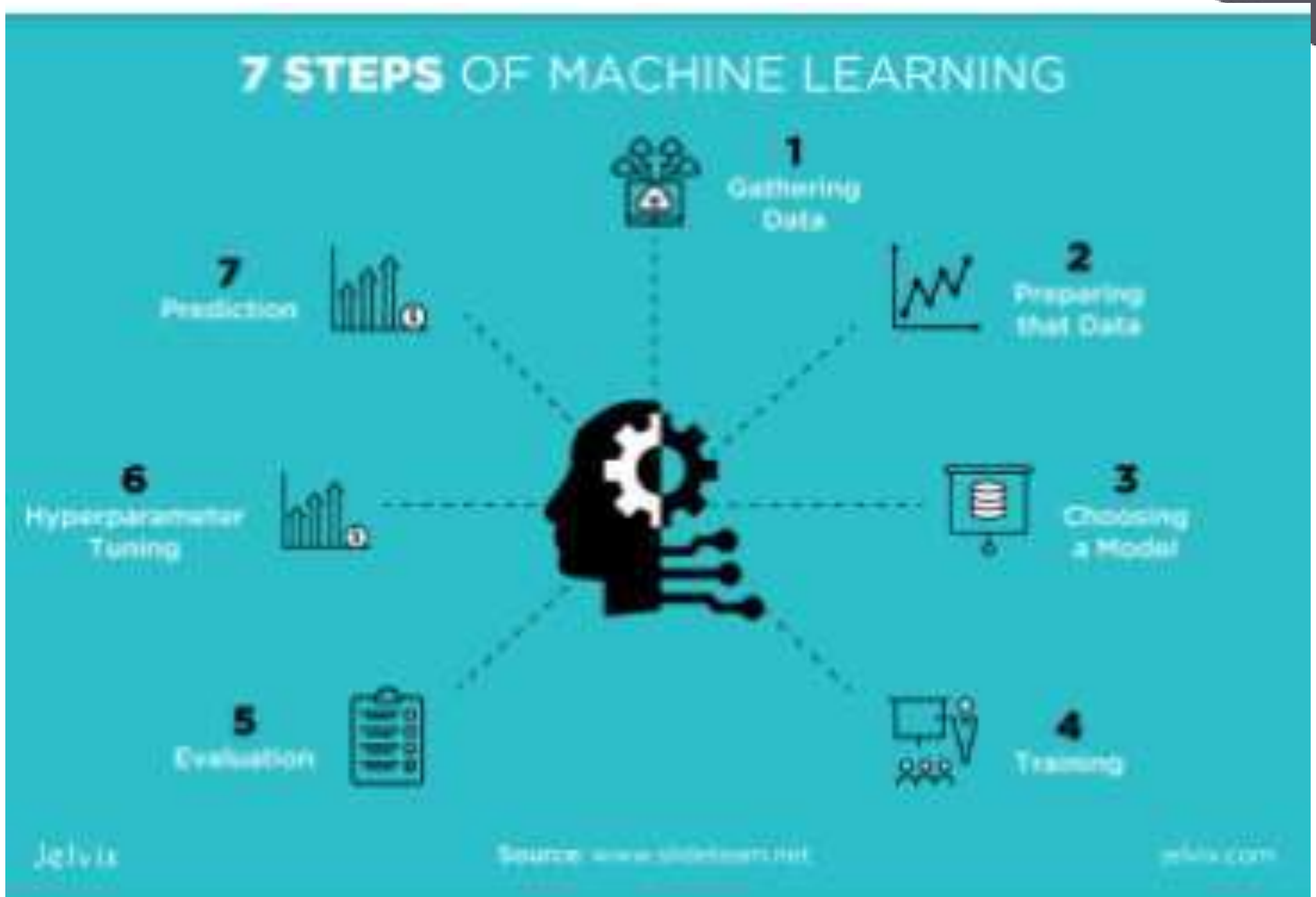
IoT Data Analytics - Challenges

2) IoT brings challenges to streaming and network analytics

Real-time analysis of streaming data allows you to detect patterns or anomalies that could indicate a problem or a situation that needs some kind of immediate response. To have a chance of affecting the outcome of this problem, you naturally must be able to filter and analyze the data while it is occurring, as close to the edge as possible.

Another challenge that IoT brings to analytics is in the area of network data, which is referred to as network analytics. With the large numbers of smart objects in IoT networks that are communicating and streaming data, it can be challenging to ensure that these data flows are effectively managed, monitored, and secure.

Machine Learning



www.reva.edu.in

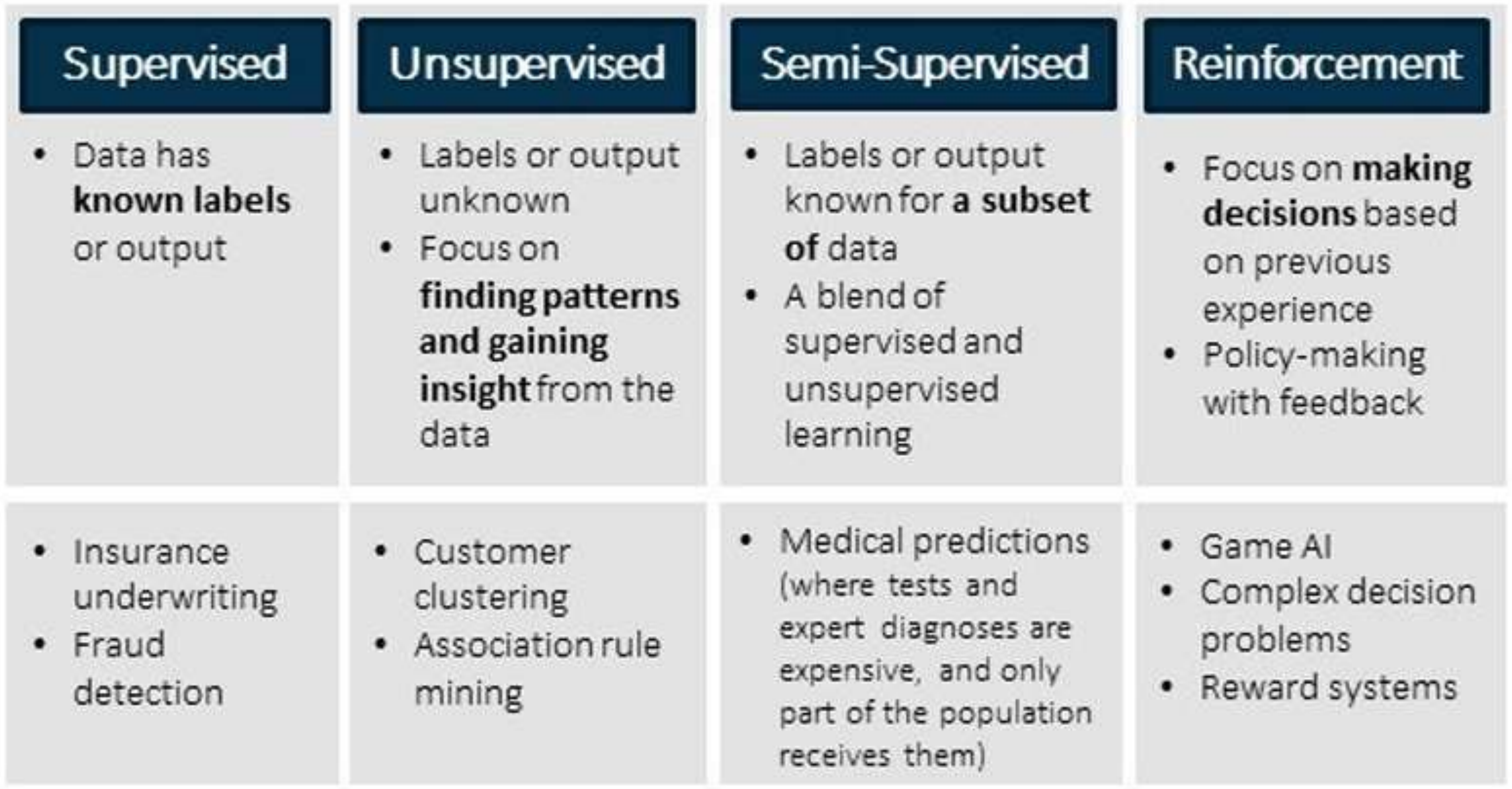
Machine Learning

MLis a part of a larger set of technologies commonly grouped under the term artificial intelligence (AI).

AI includes any technology that allows a computing system to mimic human intelligence– e.g., an App that can help you find your parked car.– e.g., a GPS reading of your position at regular intervals calculates your speed

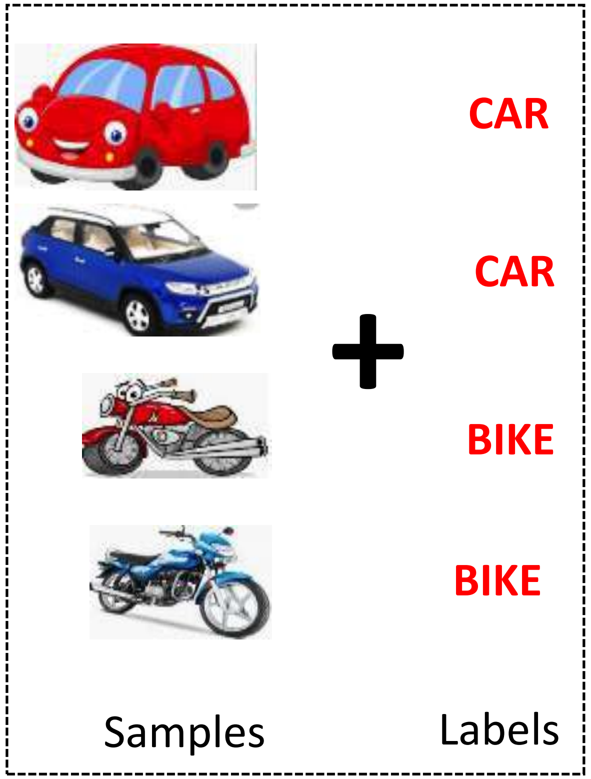
Is Machine Learning necessary for IoT?
Yes, it is required to have Machine learning for IoT. Whenever an IoT system brings dependency on humans, it fails. Therefore, it becomes necessary to support machine learning to avoid errors and smooth the process.

Types of ML



Supervised	Unsupervised	Semi-Supervised	Reinforcement
• Data has **known labels** or output	• Labels or output unknown • Focus on **finding patterns and gaining insight** from the data	• Labels or output known for **a subset of** data • A blend of supervised and unsupervised learning	• Focus on **making decisions** based on previous experience • Policy-making with feedback
• Insurance underwriting • Fraud detection	• Customer clustering • Association rule mining	• Medical predictions (where tests and expert diagnoses are expensive, and only part of the population receives them)	• Game AI • Complex decision problems • Reward systems

What is Supervised Learning?



= **Training Dataset**

$f(\quad, \qquad)=$

[Given a labelled dataset, the task is to devise a function which takes the dataset, and a new sample, and produces an output value.]

What is Supervised Learning?

CAR

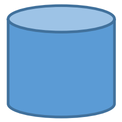
CAR

BIKE

BIKE

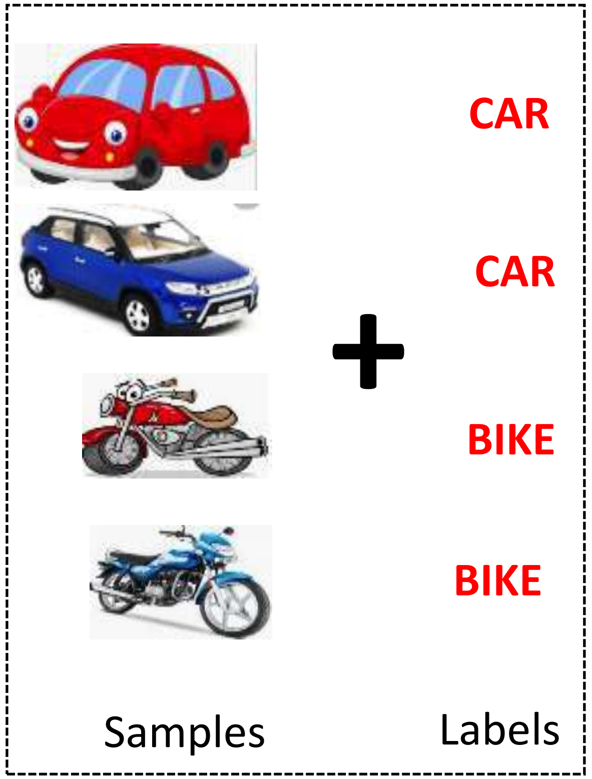
Samples Labels

+ = **Training Dataset**

$f($  , $)=$

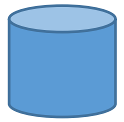
[Given a labelled dataset, the task is to devise a function which takes the dataset, and a new sample, and produces an output value.]

What is Supervised Learning?



+ = Training Dataset

Classification

$f($  , $) = \text{CAR}$

[Predefined classes means, it will produce output only from the labels defined in the dataset. For example, even if we input a bus, it will produce either CAR or BIKE]

What is Unsupervised Learning

~~CAR~~

~~CAR~~

~~BIKE~~

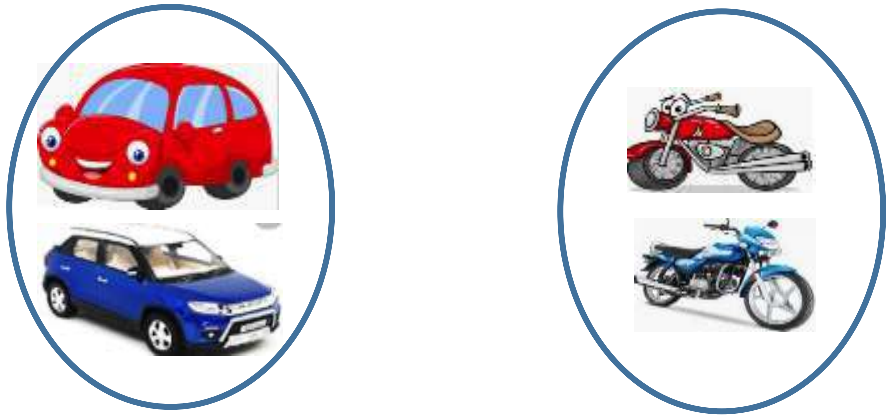
~~BIKE~~

Dataset

[In the unsupervised learning, we do not need to know the labels or Ground truth values]

What is Unsupervised Learning

Dataset

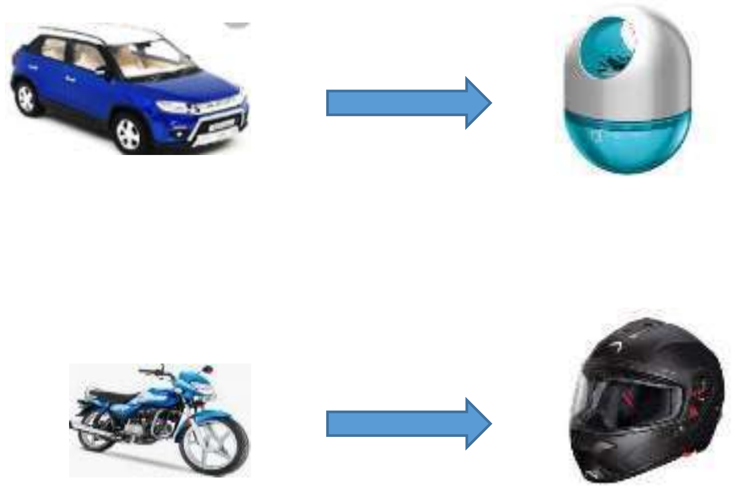


Clustering

[The task is to identify the patterns like group the similar objects together]

What is Unsupervised Learning

Dataset

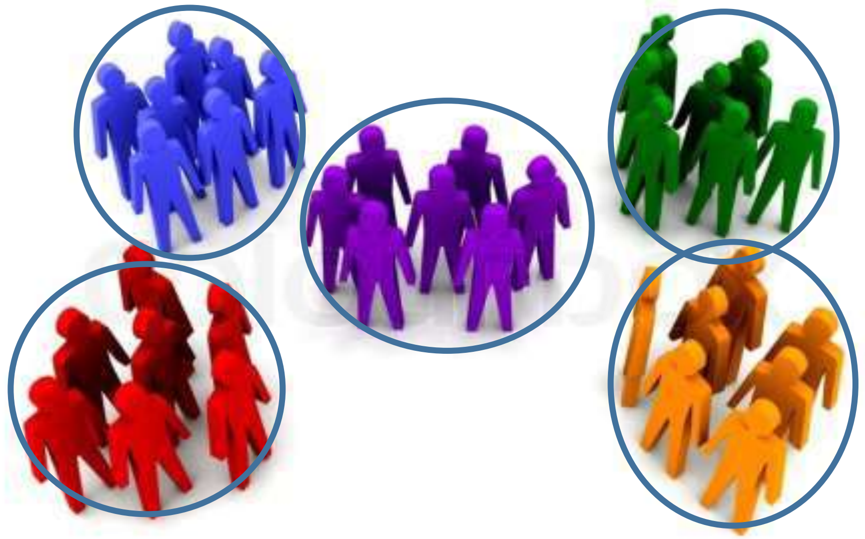


Association Rules Mining

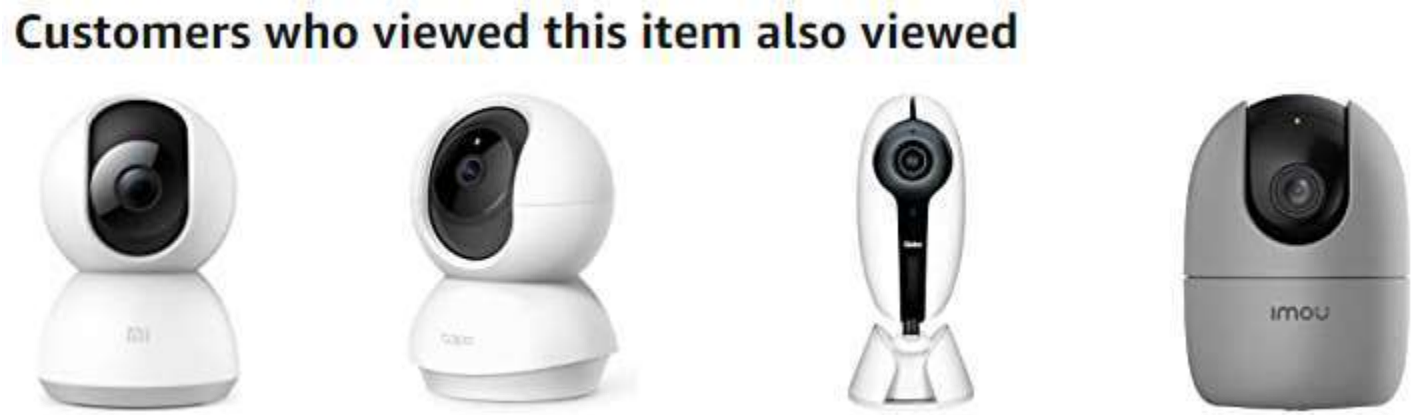
[Association rules like]

More Example Unsupervised Learning

Dataset

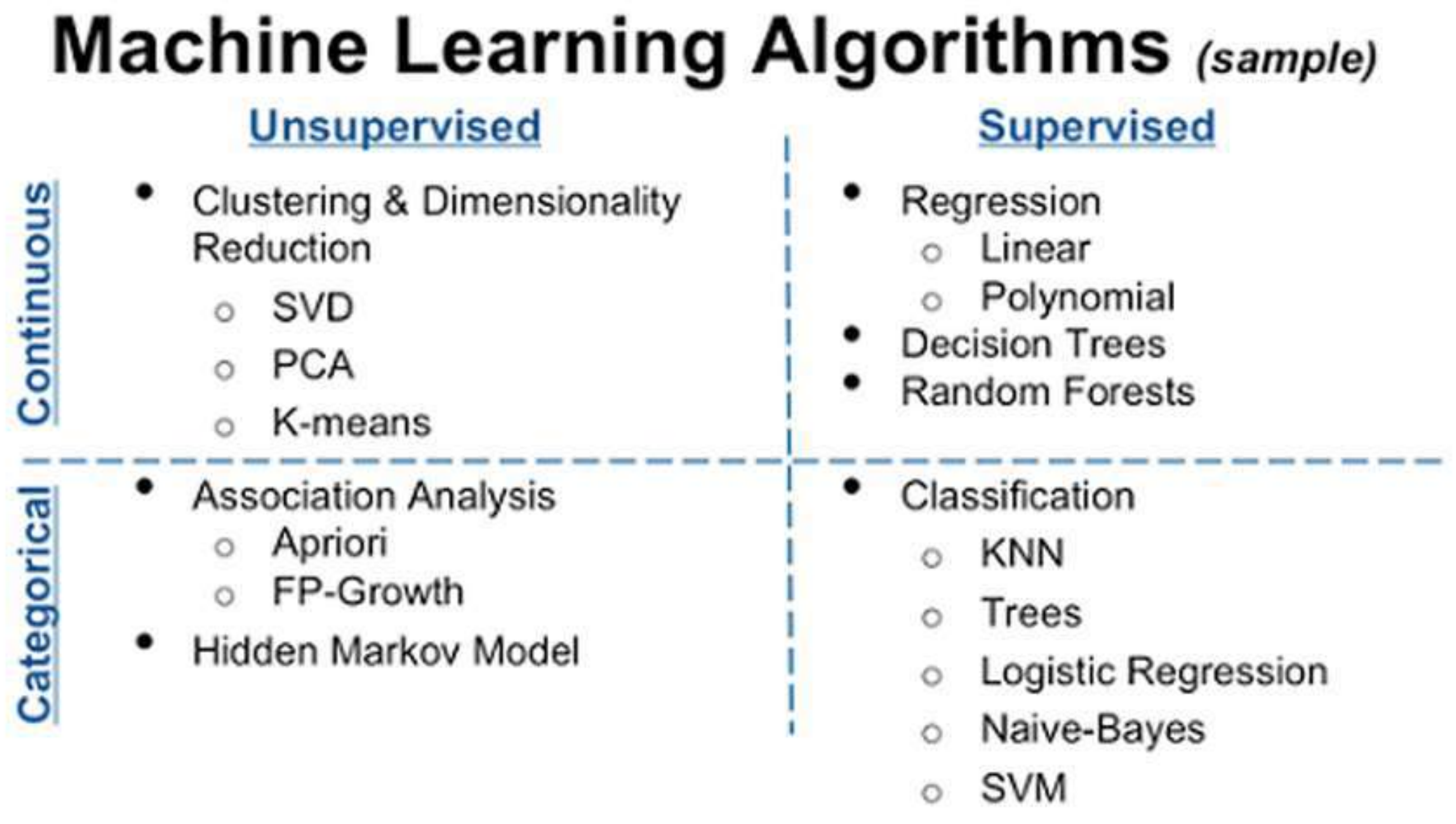


More Example Unsupervised Learning



Customers who viewed this item also viewed

Few ML Algorithms



Machine Learning Algorithms *(sample)*

	Unsupervised	Supervised
Continuous	• Clustering & Dimensionality Reduction ○ SVD ○ PCA ○ K-means	• Regression ○ Linear ○ Polynomial • Decision Trees • Random Forests
Categorical	• Association Analysis ○ Apriori ○ FP-Growth • Hidden Markov Model	• Classification ○ KNN ○ Trees ○ Logistic Regression ○ Naive-Bayes ○ SVM

Examples from IOT Application

Supervised Learning

- Suppose you are training a system to recognize when there is a human in a mine tunnel.

- Process:
  - sensor equipped with a basic camera can capture shapes
  - send them to a computing system.

  - hundreds or thousands of images are fed into the machine.
  - each image is labelled as human or nonhuman in this case

  - An algorithm is used to determine common parameters and common differences between the images.
  - This process is called *training*.

  - Each new image is compared with "good images" of human as per training model
  - This process is called *classification*.

  - the machine should be able to recognize human shapes.
  - the learning process is not about classifying in two or more categories but about finding a correct value.

  - regression predicts numeric values, whereas classification predicts categories.

Cont’d

Unsupervised Learning

- Consider a factory manufacturing small engines.
- You know that about 0.1% of the produced engines on average need adjustments to prevent later defects.
- Your task is to identify them before they shipped away from the factory.

- Process:
  - you can test each engine
  - record multiple parameters, such as sound, pressure, temperature of key parts, and so on.

  - Once data is recorded, you can graph these elements in relation to one another.
  - You can then input this data into a computer and use mathematical functions to find groups.

  - A standard function to operate this grouping, *K-means clustering*
  - Grouping the engines this way can quickly reveal several types of engines that all belong to the same category.

  - There will occasionally be an engine in the group that displays unusual characteristics
  - This is the engine that you send for manual evaluation
  - This determination process is called *unsupervised learning*.

Application Domains for ML in IOT

It revolves around four major domains:

I. Monitoring
- ML can be used with monitoring to detect early failure conditions or to better evaluate the environment

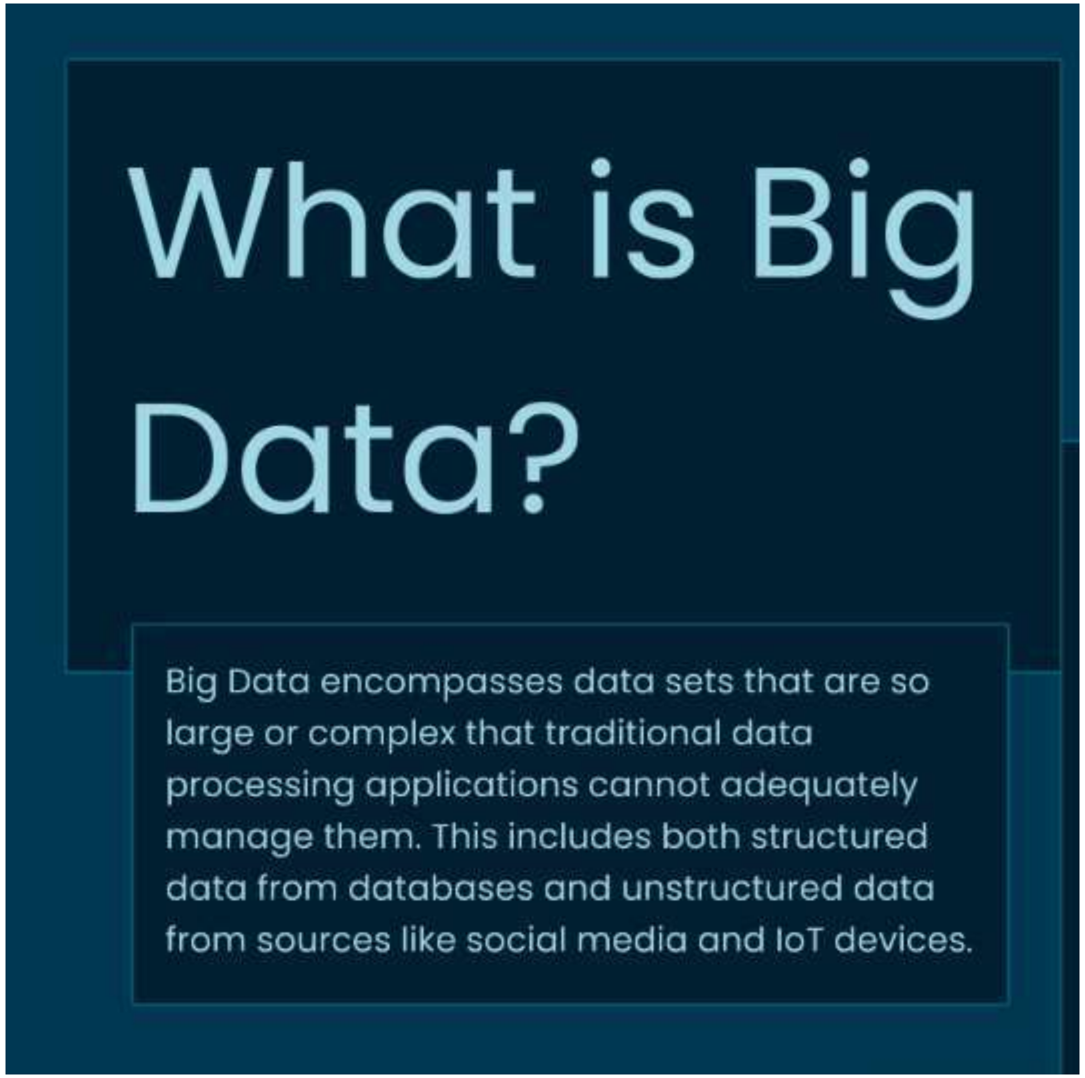
II. Behaviour control
- Monitoring commonly works in conjunction with behaviour control.
- When a given set of parameters reach a target threshold, monitoring functions generate an alarm OR would trigger a corrective action

III. Operations optimization
- The objective is not merely to pilot the operations but to improve the efficiency and the result of these operations.
  - e.g., Smart system for a water purification plant in a smart city estimate the best chemical and stirring mix for a target air temperature

IV. Self-healing, self-optimizing
- The system becomes self-learning and self-optimizing.
- ML engine can be programmed to dynamically monitor and combine new parameters, and automatically deduce and implement new optimizations

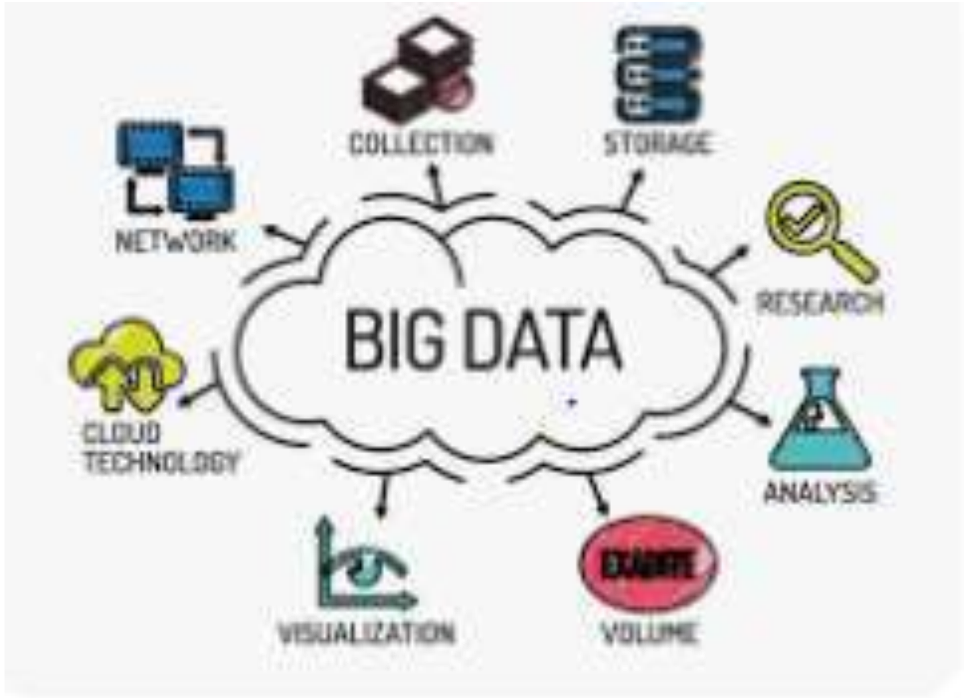
BigData Analytics

COLLECTION
STORAGE
NETWORK
RESEARCH
BIG DATA
CLOUD TECHNOLOGY
ANALYSIS
VISUALIZATION
EXABYTE
VOLUME

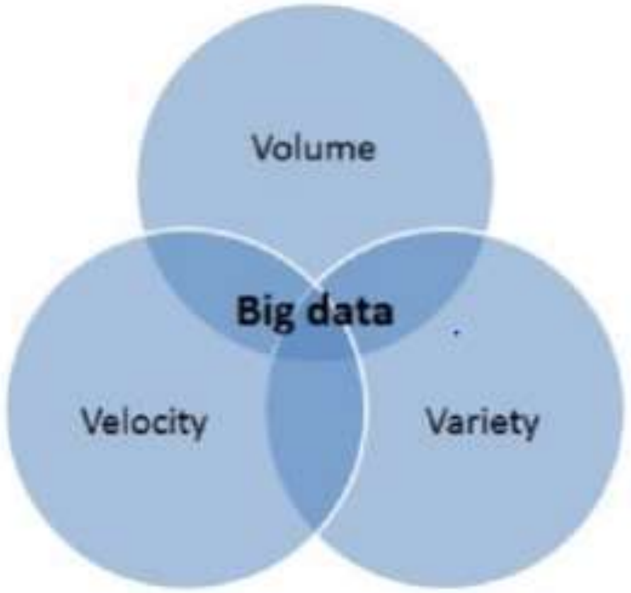


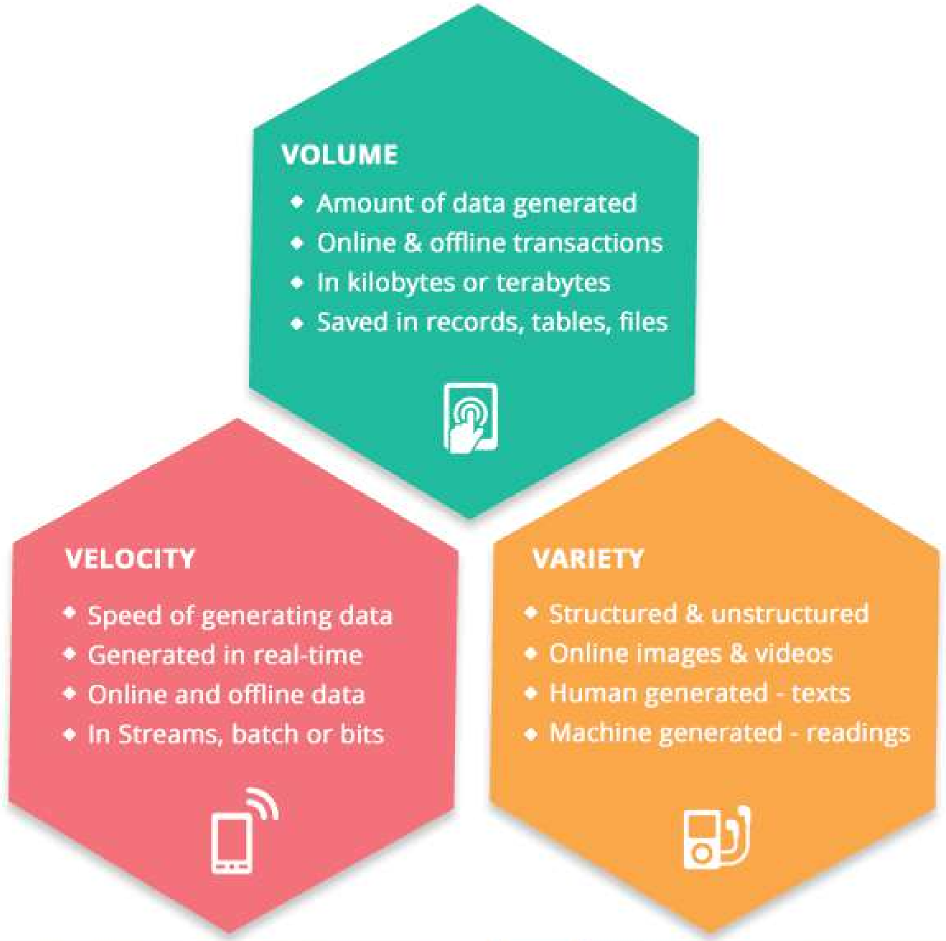
What is Big Data?

Big Data encompasses data sets that are so large or complex that traditional data processing applications cannot adequately manage them. This includes both structured data from databases and unstructured data from sources like social media and IoT devices.



3 V’s of Big Data





VOLUME

- Amount of data generated
- Online & offline transactions
- In kilobytes or terabytes
- Saved in records, tables, files

VELOCITY

- Speed of generating data
- Generated in real-time
- Online and offline data
- In Streams, batch or bits

VARIETY

- Structured & unstructured
- Online images & videos
- Human generated - texts
- Machine generated - readings

Characteristics of Big Data

- Machine data or Sensor data -Generated by IoT devices and is typically unstructured data.
- Transactional data:- from the sources that produce data from transactions on the systems, and, have high volume and structured.
- Social data - which are typically high volume and structured.
- Enterprise data-data that is lower in volume and very much structured

Database Technologies in IoT for BigData Analytics

- **NoSQL (Not Only SQL):-** is a class of databases that support semi-structured and unstructured data. It includes type of databases as below:-

-> **Document Stores-** This type of data base stores semi-structured data such as XML, JSON.

-> **Key Value Stores**:- This type of database stores associative arrays where a key is paired with an associated value. These databases are easy to build and easy to scale

->**Wide-column stores:** This type of database stores similar to a key-value store, but the formatting of the values can vary from row to row, even in the same table.

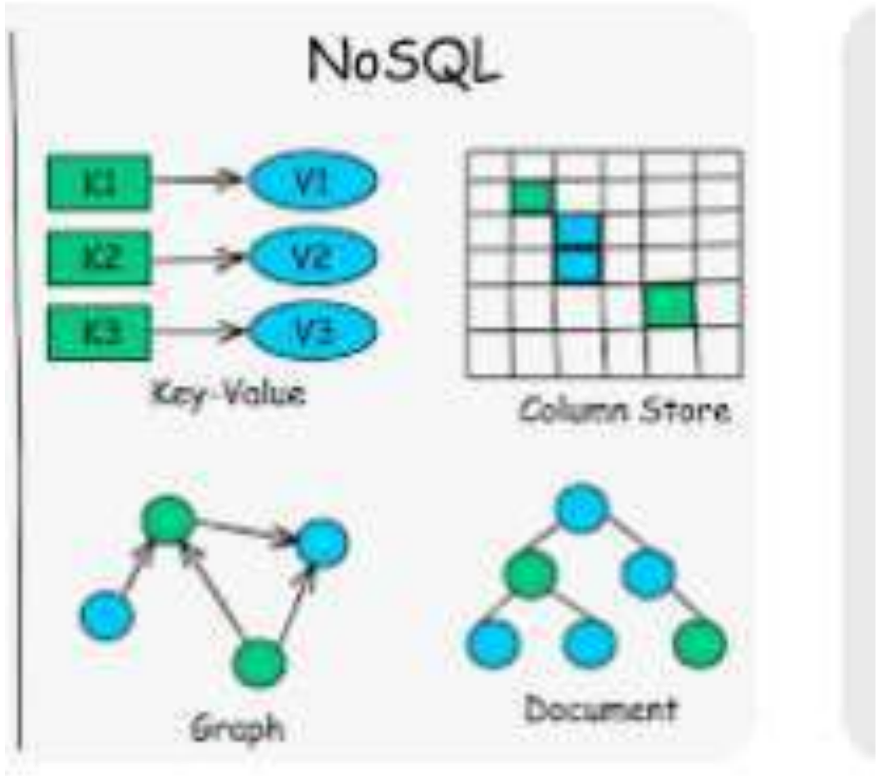
->Graph stores: This type of database is organized based on the relationships between elements. Graph stores are commonly used for social media or natural language processing, where the connections between data are very relevant.

NoSQL

- NoSQL was developed to support the →high-velocity,

-> urgent data requirements of modern web applications that typically do not require much repeated use.

-> The original intent was to quickly ingest rapidly changing server logs and clickstream data generated by web scale applications that did not neatly fit into the rows and columns required by relational databases.

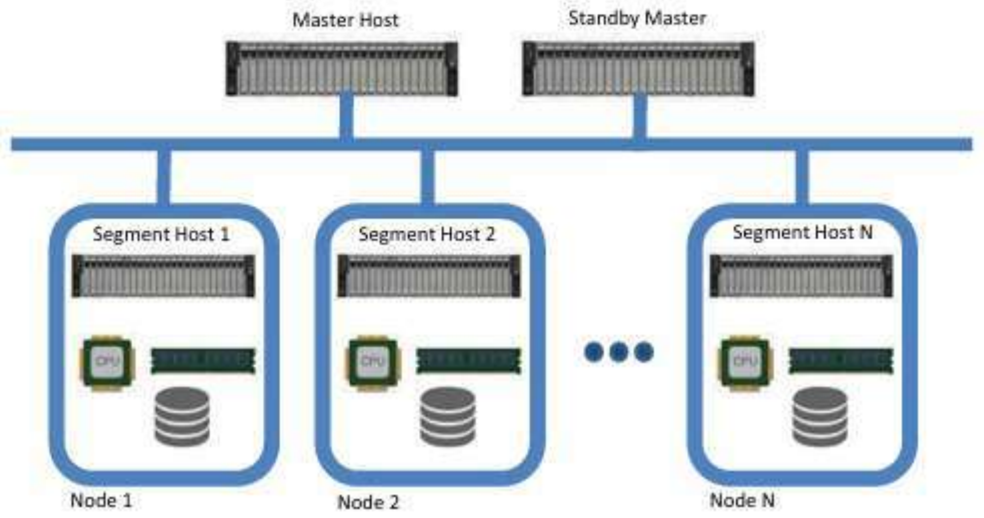


Massively Parallel Processing Database

Massively parallel processing (MPP) databases were built on the concept of the relational data warehouses but are designed to be much faster, to be efficient, and to support reduced query times.

To accomplish this, MPP databases take advantage of multiple nodes (computers) designed in a scale-out architecture such that both data and processing are distributed across multiple systems.

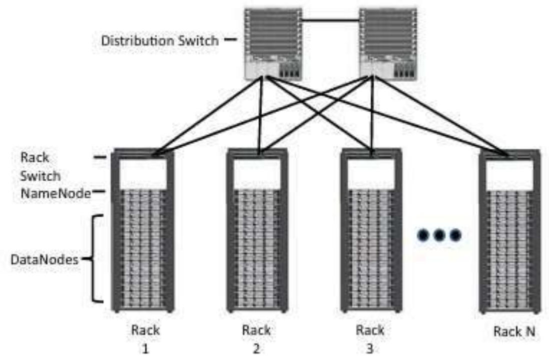
An MPP architecture (typically contains a single master node that is responsible for the coordination of all the data storage and processing across the cluster. It operates in a “shared-nothing” fashion, with each node containing local processing, memory, and storage and operating independently.



Hadoop

- Most popular choice as a data repository and processing engine
- Originally developed as a result of projects at Google and Yahoo!– original intent was to index millions of websites and quickly return search results for open source search engines.
- Initially, the project had two key elements:– Hadoop Distributed File System (HDFS): A system for storing data across multiple nodes.
- MapReduce:A distributed processing engine that splits a large task into smaller ones that can be run in parallel.

Key Features of HDFS



Key Features of HDFS:

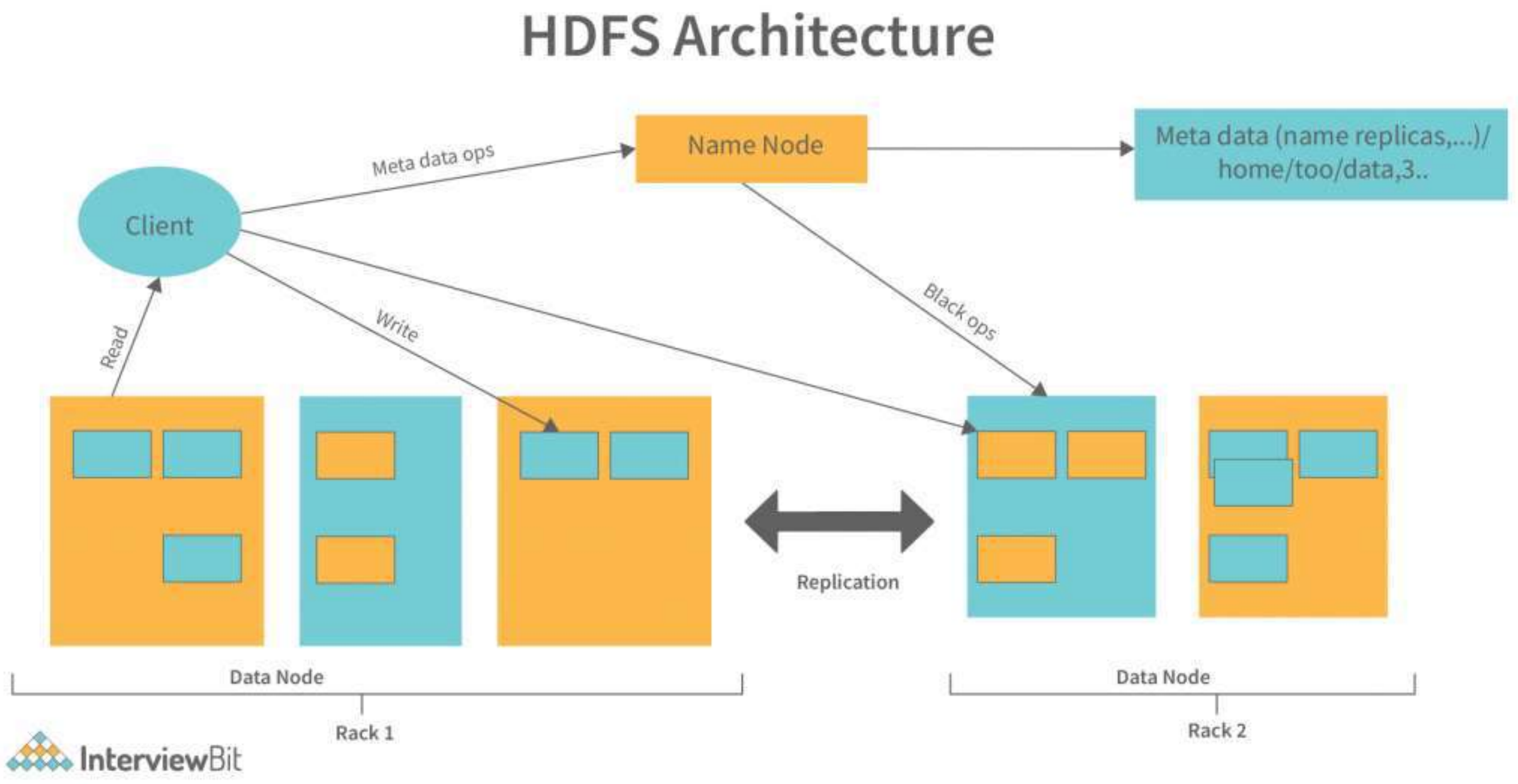
- **Scalability**: HDFS can scale horizontally by adding more nodes to the cluster, thus allowing the storage of petabytes of data.
- **Fault Tolerance**: Data in HDFS is replicated across multiple nodes (by default, three replicas) to ensure that if one node fails, data can still be retrieved from other nodes.
- **High Throughput**: HDFS is optimized for large datasets and batch processing, making it ideal for tasks that require reading and writing large files.

Cont’d

- **Streaming Data Access**: It’s designed more for streaming reads and writes rather than random reads and writes, favoring batch processing over interactive use.
- **Data Blocks**: HDFS splits large files into blocks (typically 128 MB or 256 MB) and distributes these blocks across different nodes in the cluster.

- **Master-Slave Architecture**:

1. **NameNode**: The master node that manages metadata and file system operations.
2. **DataNodes**: The worker nodes responsible for storing actual data blocks.



Much like the MPP and NoSQL systems, Hadoop relies on a scale-out architecture that leverages local processing, memory, and storage to distribute tasks and provide a scalable storage system for data.

NameNode

NameNode **stores metadata** about all files and directories in the file system.

What It knows:

- The **hierarchical structure** (folders/files)
- The **location of data blocks** on DataNodes
- The **replication factor** of blocks

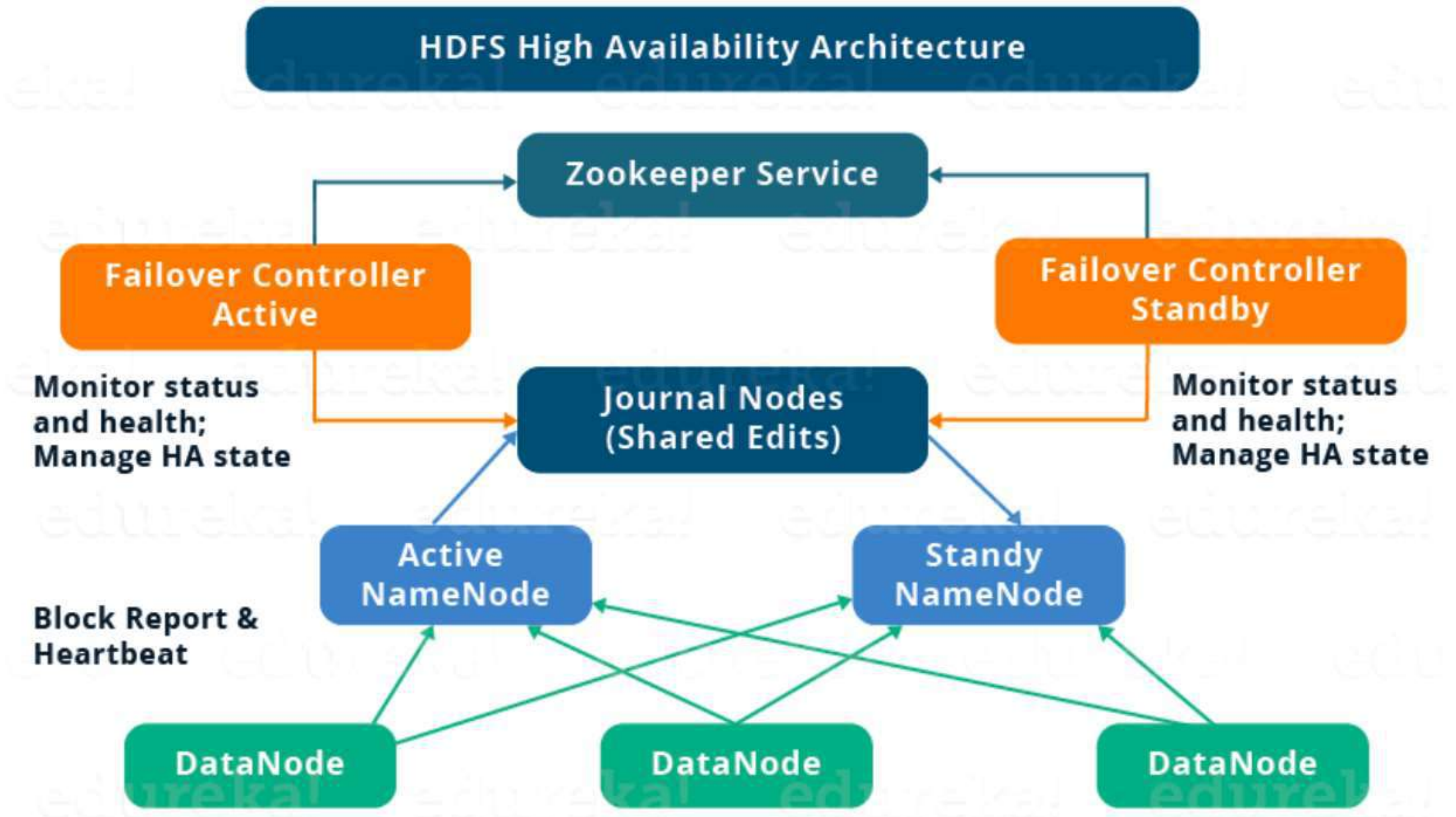
Keeps track of:

- File names, permissions, and locations
- Mapping of file blocks to DataNodes

If the NameNode fails...

- The file system becomes **inaccessible**.
- Modern HDFS setups use **High Availability (HA)** with **active and standby NameNodes**.

HA with HDFS


HDFS High Availability Architecture
Zookeeper Service
Failover Controller Active
Failover Controller Standby
Monitor status and health; Manage HA state
Journal Nodes (Shared Edits)
Monitor status and health; Manage HA state
Active NameNode
Standy NameNode
Block Report & Heartbeat
DataNode
DataNode
DataNode

Data Node

- DataNodes store every data.
- It handles all of the requested operations on files, such as reading file content and creating new data, as described above.
- All the instructions are followed, including scrubbing data on DataNodes, establishing partnerships, and so on.

Processing Data with HADOOP

- Processing data with Hadoop primarily involves the **MapReduce** programming model.
- which works in conjunction with the **Hadoop Distributed File System (HDFS)** to handle large-scale data processing across distributed systems.

Core Components of Hadoop:

- **HDFS**: Stores large datasets across multiple nodes.
- **MapReduce**: The computational model used for processing data in parallel.
- **YARN (Yet Another Resource Negotiator)**: Manages and schedules resources for running tasks on the Hadoop cluster.

Steps in Processing Data with Hadoop (MapReduce Model):

1. Data Storage in HDFS:

- Data is first loaded into HDFS, which breaks it into blocks and distributes them across different nodes. Since the data is spread across the cluster, it can be processed in parallel.

2. MapReduce Programming Paradigm:

MapReduce consists of two main functions:

- **Map Function**: It processes input data and transforms it into key-value pairs. This is the first stage of computation.
- **Reduce Function**: It takes the output from the Map function, groups it by key, and then performs a summary operation (e.g., counting, summing) to produce the final output.

Contd

3. Execution in Parallel:

MapReduce jobs are executed in parallel across the cluster. Hadoop’s YARN framework manages resource allocation to ensure efficient processing.

4. Fault Tolerance:

If a node fails during execution, Hadoop reruns the failed task on another node, ensuring fault tolerance. Data replication in HDFS also guarantees that the data is always available.

5. Data Output:

Once the Reduce phase is complete, the output is typically written back to HDFS for further analysis or storage.

How MapReduce Works in Hadoop

- **Input Splitting**: Hadoop divides large datasets into smaller chunks, known as input splits. Each chunk is processed by a different mapper.
- **Task Distribution**: YARN assigns tasks (map or reduce) to available nodes in the cluster based on the location of the data to reduce data transfer.
- **Task Monitoring**: Hadoop continuously monitors running tasks. If a task fails, it restarts the task on another node.

Example Use Cases of Hadoop with MapReduce:

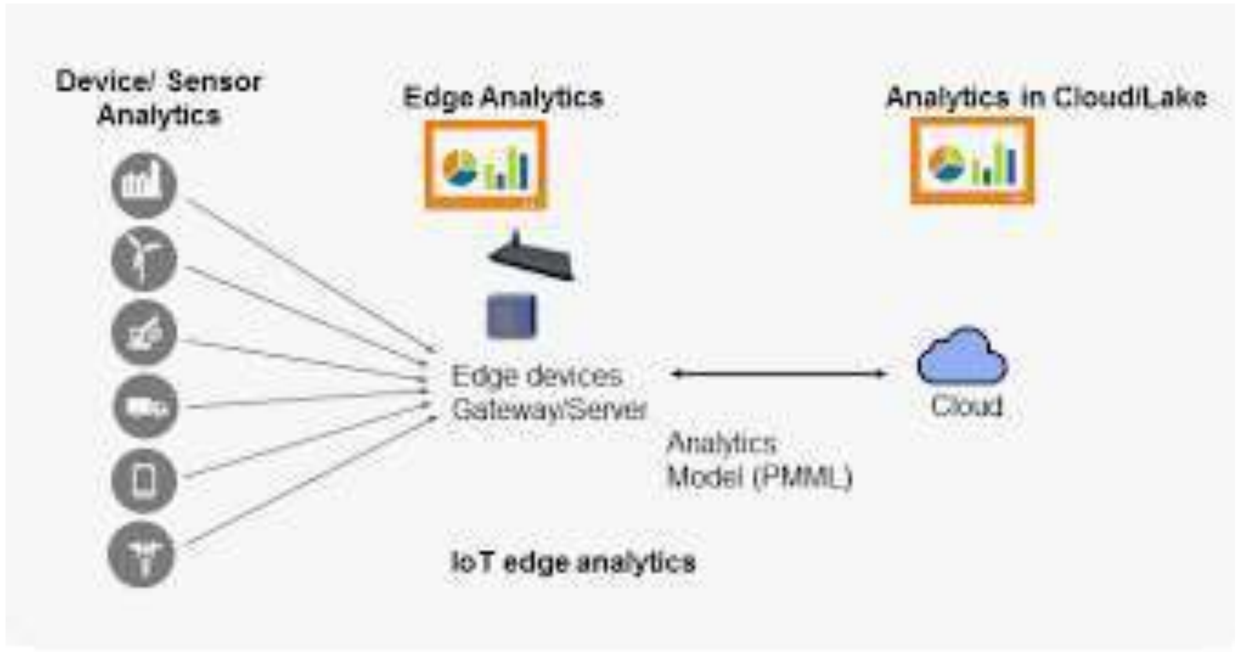
- **Log Analysis**: Processing and analyzing large server logs to extract information like error rates, traffic patterns, or other metrics.
- **Data Aggregation**: Summing sales data from multiple regions to compute total sales or identifying top-selling products.
- **ETL (Extract, Transform, Load)**: Hadoop is used to transform raw data into a more structured format for downstream analysis.

Hadoop Ecosystem

- Hadoop Ecosystem comprises of more than 100 software projects under the Hadoop umbrella
  - Capable of every element in the data lifecycle,
    - from data collection,
    - to storage,
    - to processing,
    - to analysis, and
    - to visualization

- Several of these packages
  - **Apache Kafka**
  - **Apache Spark**
  - **Apache Storm**
  - **Apache Flink**
  - **Lambda Architecture**

Edge Analytics



Edge Streaming Analytics

- In the world of IoT vast quantities of data are generated on the fly
  - Often they are time sensitive i.e. needs immediate attention,
  - waiting for deep analysis in the cloud simply isn't possible.
  - e.g., automobile racing industry
    - Formula One racing car has 150-200 sensors that generate more than 1000 data points per second
    - enormous insights leading to better race results can be gained by analyzing data on the fly
- Big Data tools like Hadoop and MapReduce are not suitable for real-time analysis
  - because of distance from the IoT endpoints and the network bandwidth requirement

- **Streaming analytics** allows you to continually monitor and assess data in real-time so that you can adjust or fine-tune your predictions as the race progresses.

- In IoT, streaming analytics is performed at the edge
  - either at the sensors themselves or very close to them such as gateway

- The edge isn't in just one place. The edge could be highly distributed.

Key Features of Edge Streaming Analytics

- Does the streaming analytics replaces big data analytics in the Cloud?
  - Answer: Not at all.
  - **Big data analytics** is focused on large quantities of **data at rest**,
  - **Edge analytics** continually processes streaming flows of **data in motion**.

Key Features:

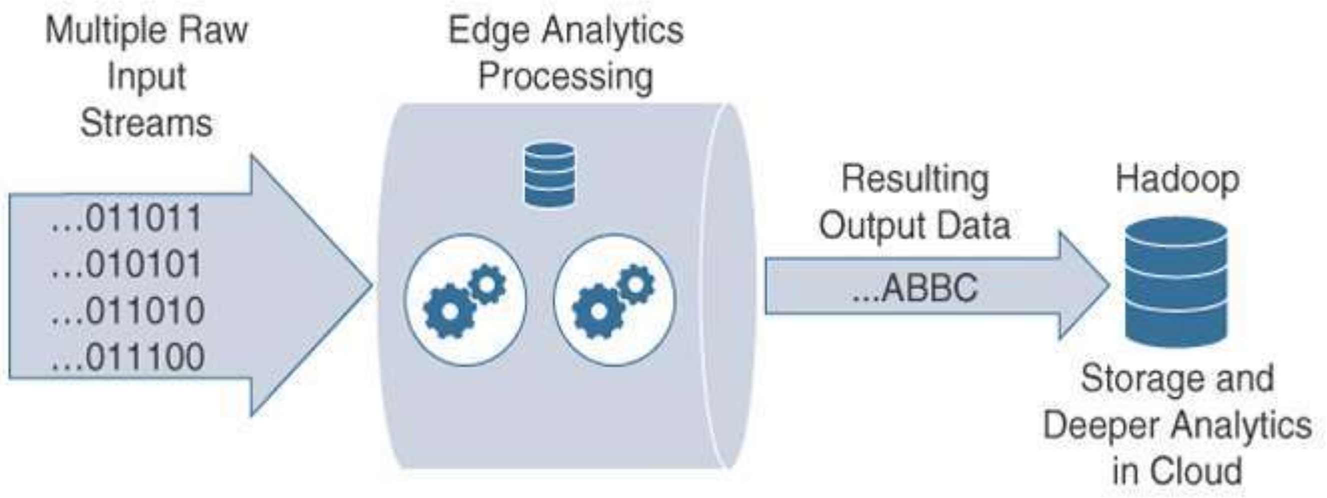
- Reducing data at the edge
  - Passing all data to the cloud is inefficient and is unnecessarily expensive in terms of bandwidth and network infrastructure.

- Analysis and Response at the edge
  - Some data is useful only at the edge and for small window of time
  - e.g., Roadway sensors combined with GPS wayfinding apps may tell a driver to avoid a certain highway due to traffic. This data is valuable for only a small window of time.

- Time sensitivity
  - When timely response to data is required, passing data to the cloud for future processing results in unacceptable latency.

Edge Analytics Core Functions

- Raw input data
  - This is the raw data coming from the sensors into the analytics processing unit.
- Analytics processing unit (APU)
  - The APU filters and combines (or separates) data streams, organizes them by time windows, and performs various analytical functions.
- Output streams
  - The data that is output is organized into insightful streams and passed on for storage and further processing in the cloud.

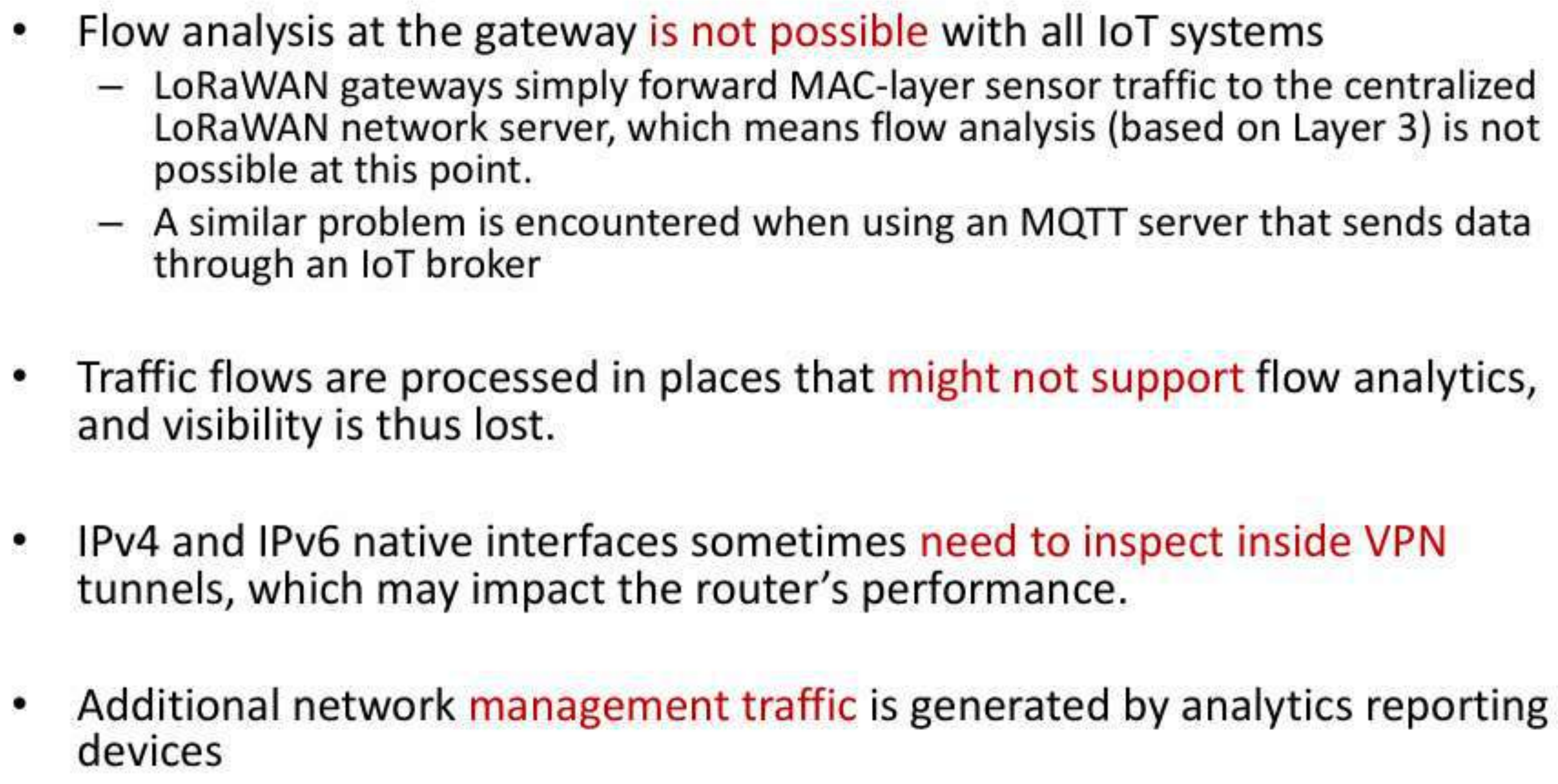

Multiple Raw Input Streams
...011011
...010101
...011010
...011100
Edge Analytics Processing
Resulting Output Data
...ABBC
Hadoop
Storage and Deeper Analytics in Cloud

Network Analytics

- Network analytics: concerned with discovering patterns in the communication flows
  - It is network-based analytics
  - power to analyze details of communications patterns made by protocols
  - correlate this pattern across the network
  - allows to understand what should be considered normal behavior in a network

Challenges of Network Analytics

- Challenges with deploying flow analytics tools in an IoT network.



- Flow analysis at the gateway is not possible with all IoT systems
  - LoRaWAN gateways simply forward MAC-layer sensor traffic to the centralized LoRaWAN network server, which means flow analysis (based on Layer 3) is not possible at this point.
  - A similar problem is encountered when using an MQTT server that sends data through an IoT broker
- Traffic flows are processed in places that might not support flow analytics, and visibility is thus lost.
- IPv4 and IPv6 native interfaces sometimes need to inspect inside VPN tunnels, which may impact the router's performance.
- Additional network management traffic is generated by analytics reporting devices

Security in IoT

Introduction

- IoT security is the act of securing IoT devices and the networks they're connected to.
- From the Internet of Things devices, Attackers may utilize remote access to steal data by using a variety of strategies, including credential theft and vulnerability exploitation.
- As IoT devices become increasingly integrated into critical systems, **security** has emerged as one of the most significant challenges in the IoT ecosystem.

Security Requirements

Fundamental Requirements:

- Confidentiality
  - refers to protecting information from being accessed by unauthorized parties
- Integrity
  - Data must not be changed in transit.
- Availability
  - is a guarantee of reliable access to the information by authorized people.
- Authentication
  - merely ensures that the individual is who he or she claims to be
- Non-repudiation
  - is the assurance that someone cannot deny the validity of something
- Resilience
  - is the ability to prepare for, respond to and recover from an attack.

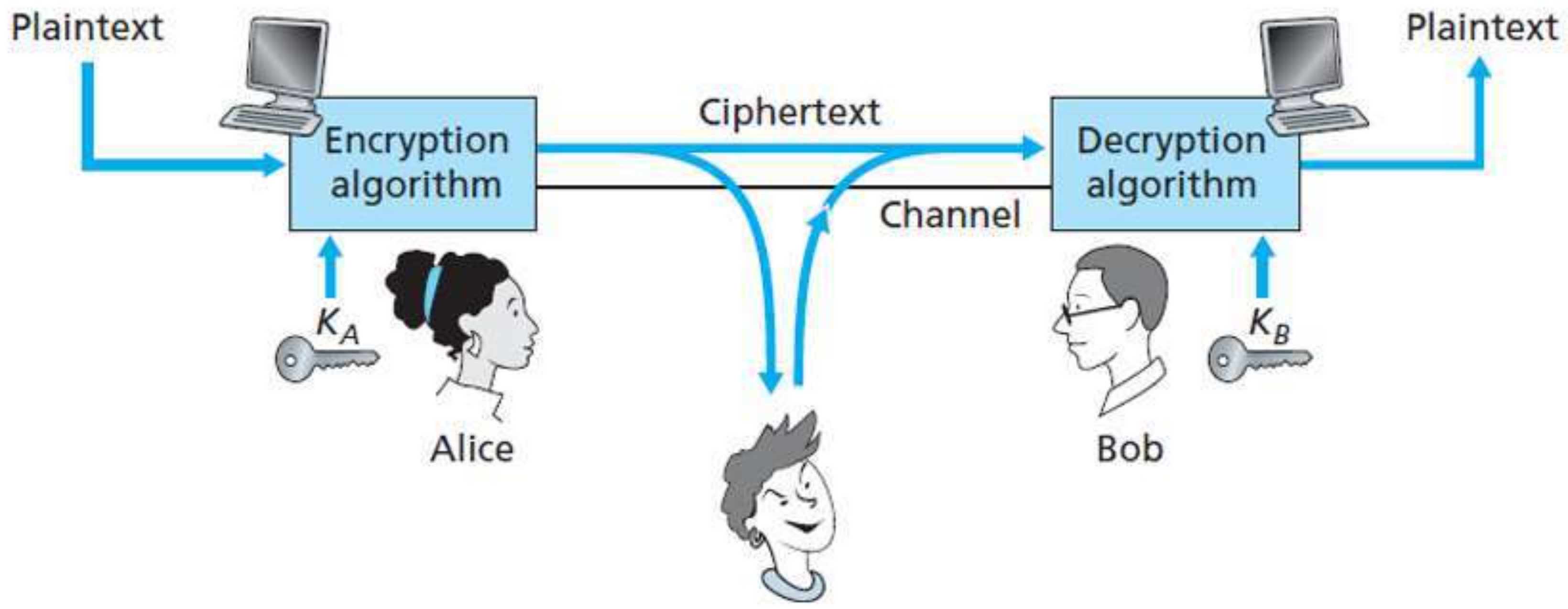
Few more Requirements - Privacy ; Anonymity; Accountability; Trust

Common Security Attacks

- *Snooping :* unauthorized access or interception of data.
- *Traffic Analysis :* obtain some other types of information by monitoring online traffic.
- *Modification :* modifies the information to make it beneficial to the attacker
- *Masquerading* or *spoofing* : the attacker impersonates somebody else.
- *Replaying :* the attacker replays the obtained/snooped copy of a message later on.
- *Repudiation :* The sender of the message might later deny that it has sent the message OR the receiver of the message might later deny that it has received the message.
- *Denial of Service* (DoS)*:* attacker may slow down or totally make unavailable the service of a legitimate system.

Cryptography for Confidentiality

- Cryptographic techniques allow a sender to disguise data so that an intruder can gain no information from the intercepted data

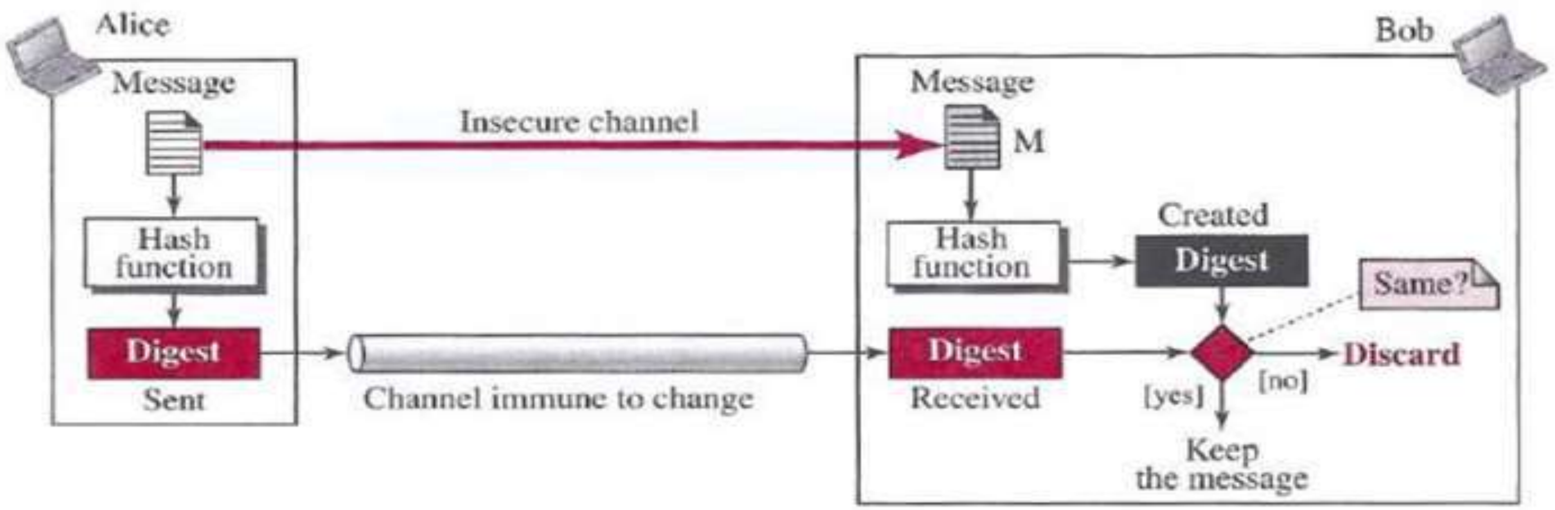


- To create the ciphertext from the plaintext, Alice uses an encryption algorithm and a key.
- To create the plaintext from cipher text, Bob uses a decryption algorithm and a key
- Based on type of key
- – Symmetric key systems : Alice's and Bob's keys are identical and are secret
- – Asymmetric / public key systems : a pair of keys is used. One of the keys is known to both Bob and Alice (indeed, it is known to the whole world). The other key is known only by either Bob or Alice (but not both)

Message Digest for Integrity

- We must ensure integrity: the message should remain unchanged.
- One way to preserve the integrity of a document is through the use of a fingerprint.
- A **Message Digest** is a fixed-size hash value generated from data (a message) using a **hash function**. It plays a key role in ensuring **data integrity**, meaning the data has not been altered or tampered with during transmission or storage.

Message Digest Working



Sender Side:

•A hash function (like **SHA-256**) processes the message to produce a **message digest**.

•The digest is sent along with the message.

Receiver Side:

•The receiver runs the same hash function on the received message.

•It compares the newly calculated digest with the received digest.

•If they match, the message has not been altered (integrity is preserved).

•If not, the data was likely tampered with or corrupted.

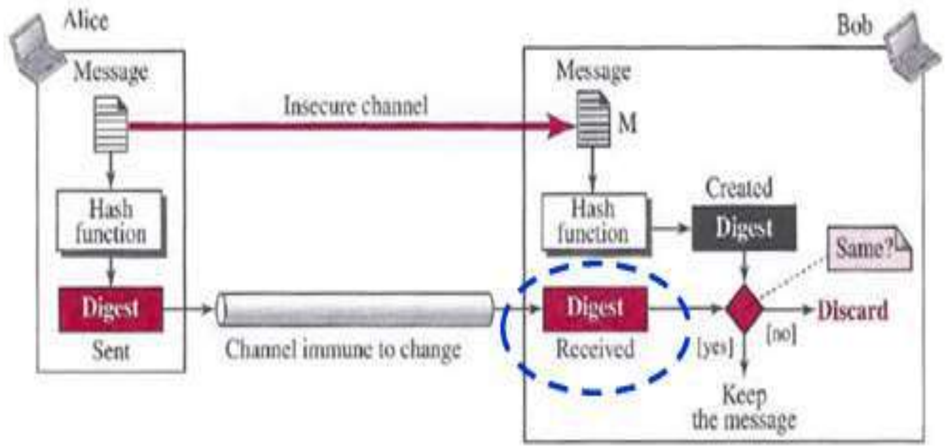
Message Authentication

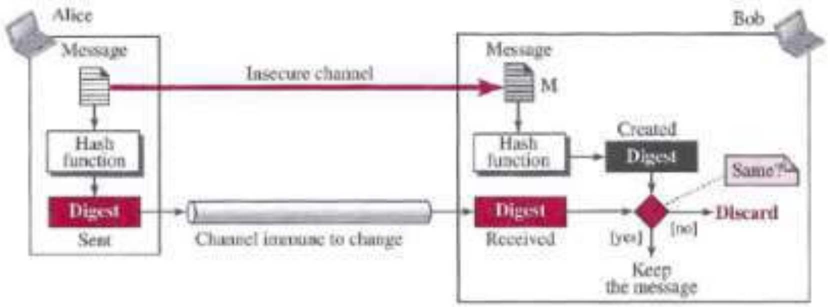
- A digest can be used to check the integrity of a message.
- But, how to ensure the authentication of the message - the message indeed originated from Alice

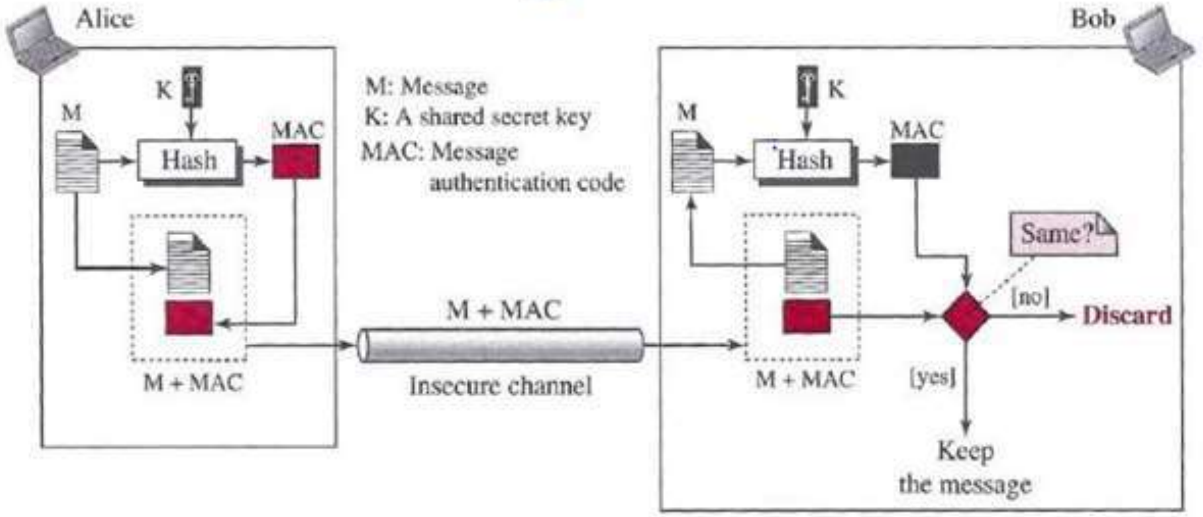
Solution: we need to include a secret shared between Alice and Bob.

– This shared secret, which is nothing more than a string of bits, is called the authentication key

– Message digest of message and authentication key is called Message Authentication Code (MAC)



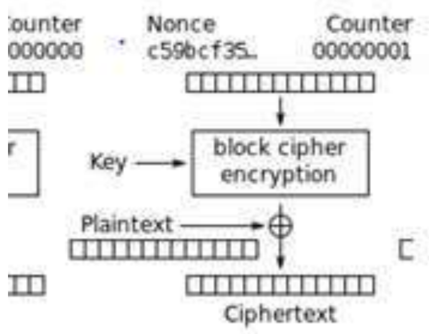




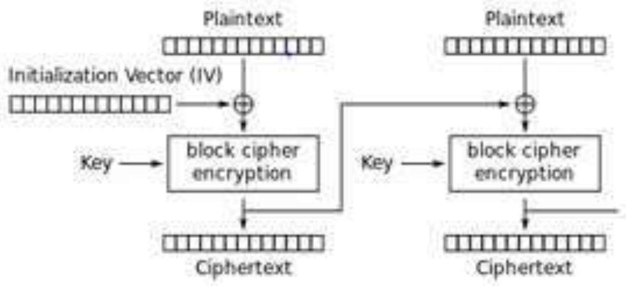
- One nice feature of a MAC is that it does not require an encryption algorithm

Security in IoT PHY & MAC

CTR Mode



CBC Mode

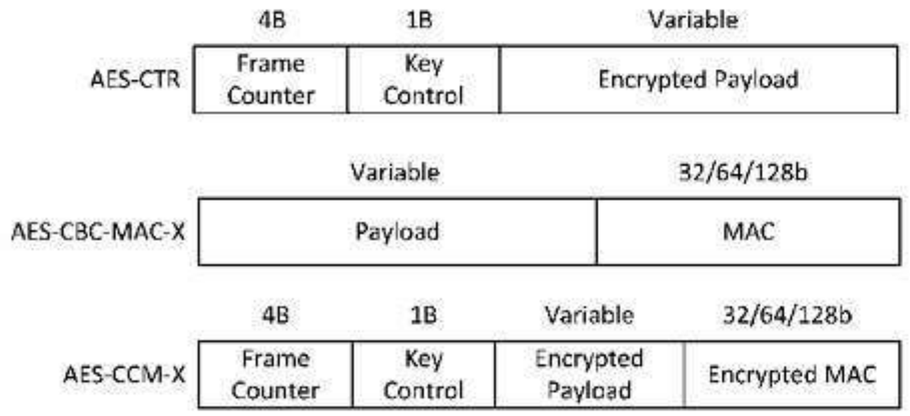


Security Suites	Achieved Security Requirements
AES-CTR	Confidentiality
AES-CBC-MAC-32, AES-CBC-MAC-64, AES-CBC-MAC-128	*Data Authenticity, Integrity,*
AES-CCM-32, AES-CCM-64, AES-CCM-128	*Confidentiality, Data Authenticity, Integrity*

AES: Advanced Encryption Standard
CBC: Cypher Block Chaining
MAC: Message Authentication Code
CTR: Counter Mode
CCM: CTR + CBC

Security-related information

- The various *security suites* require transportation of security-related information for different configurations



Security Suites	Achieved Security Requirements
AES-CTR	Confidentiality
AES-CBC-MAC-X	*Data Authenticity* *Integrity*
AES-CCM-X	*Confidentiality* *Data Authenticity* *Integrity*

Cont’d

The frame counter is used to:

•Keep track of how many data frames (packets) are sent or received.

•Prevent **replay attacks** by ensuring that an old frame (with a lower counter) is not accepted again.

A **Key Control Frame** is a special type of control message/frame used to:

•Manage **encryption keys** (distribution, refresh, or acknowledgment).

•Initiate or maintain **secure connections** between devices.

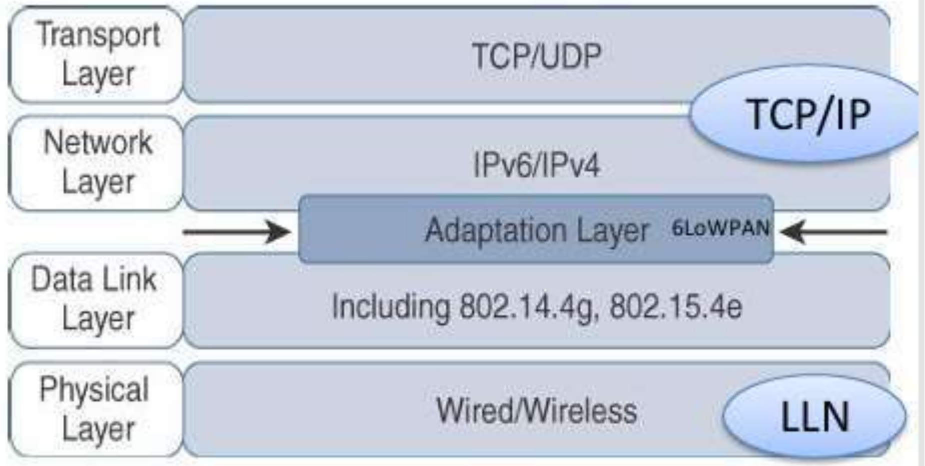
•Coordinate **handshakes** or **authentication** processes.

Adaptation Layer Protocol

- 6LoWPAN is currently a key technology to support Internet communications in the IoT

- It defines:
  - Header compression
    - Compress the NET and TRN layer headers
  - Neighbor discovery
  - Address auto-configuration

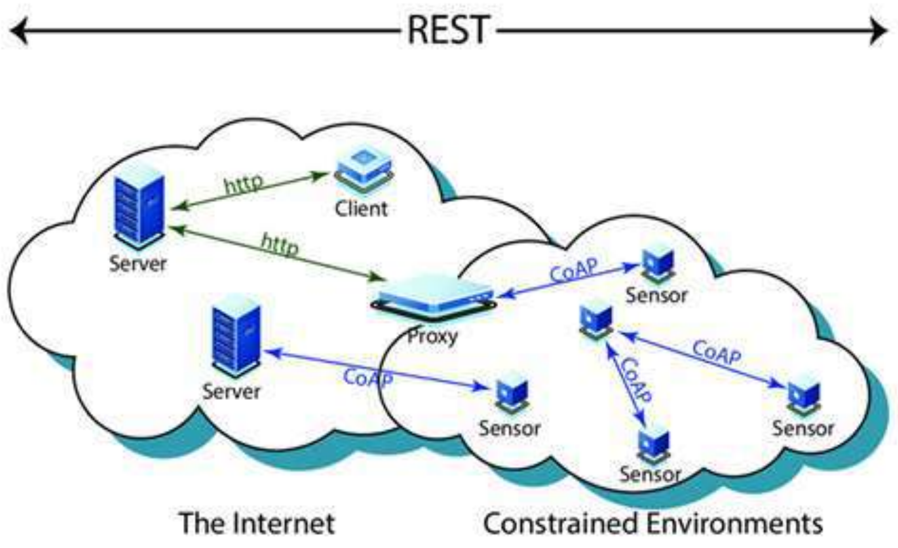
- It supports four header types:
  - Not 6LoWPAN
  - Dispatch
  - Mesh Addressing
  - Fragmentation



4 Header Types

- Not a 6LoWPAN Packet (Type 00):- This indicates that the packet is not a 6LoWPAN packet. It helps identify and separate 6LoWPAN traffic from other types.
- Dispatch Header:- Used to identify the type of 6LoWPAN packet, such as IPv6 header compression, mesh headers, or fragmentation headers. It essentially tells how the payload should be interpreted.
- Mesh Addressing Header:- Enables multi-hop delivery by including source and destination short addresses. It's crucial for routing packets across multiple 6LoWPAN nodes (mesh networks).
- Fragmentation Header:- Supports the fragmentation and reassembly of packets. Since IEEE 802.15.4 frames are small (127 bytes max), large IPv6 packets need to be broken down into fragments.

CoAP



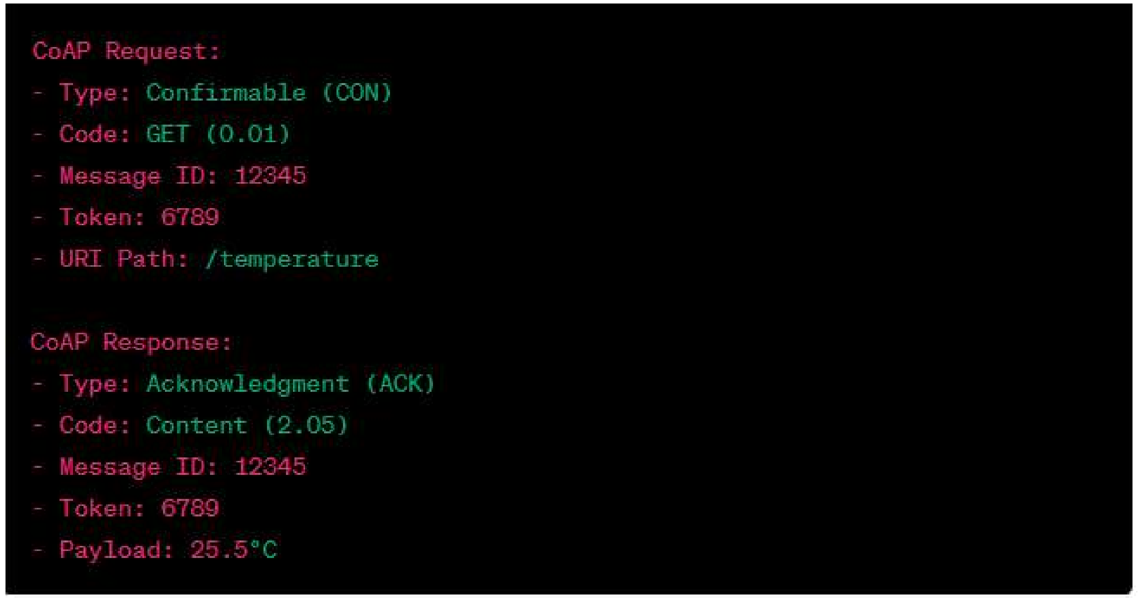
CoAP: Constrained Application Protocol

- For constrained and low-power networks.
- Follows the request-response messaging pattern
- The request is sent using Confirmable (CON) or Non-Confirmable (NON) message.
  - Uses stop-and-wait mechanism with exponential backoff to achieve reliability
- Response is sent using several scenarios (e.g. piggy-backed, separate response, etc)

Other Features:

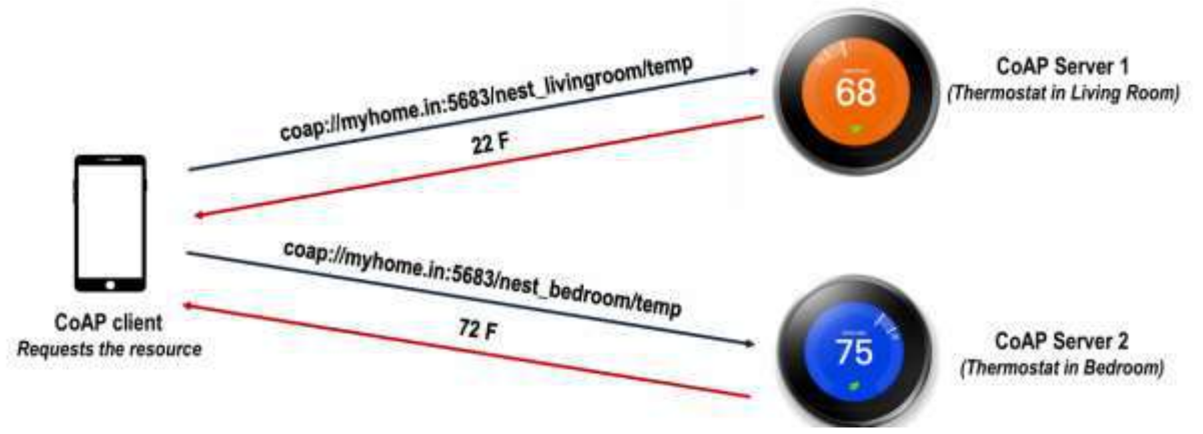
- ➤ Very efficient RESTful protocol (i.e. uses REST: Representational State Transfer architecture)
- ➤ Low header overhead and parsing complexity
- ➤ Uses both asynchronous & synchronous messaging
- ➤ Mainly uses for UDP communications

CoAP Protocol Example



```
CoAP Request:
- Type: Confirmable (CON)
- Code: GET (0.01)
- Message ID: 12345
- Token: 6789
- URI Path: /temperature

CoAP Response:
- Type: Acknowledgment (ACK)
- Code: Content (2.05)
- Message ID: 12345
- Token: 6789
- Payload: 25.5°C
```


coap://myhome.in:5683/nest_livingroom/temp
22 F
68
CoAP Server 1
(Thermostat in Living Room)
CoAP client
Requests the resource
coap://myhome.in:5683/nest_bedroom/temp
72 F
75
CoAP Server 2
(Thermostat in Bedroom)

Coap://myhome.in:5683/nest_livingroom/temp
Coap:- Protocol,
myhome:- Domain
5683:- Port number
Nest_living room:- Name of the Device
Temp:- Parameter

CoAP Security

- CoAP Protocol defines bindings to DTLS (Datagram Transport-Layer Security) to transparently apply security to all CoAP messages

- Security Modes in CoAP:
  - *NoSec*
    - CoAP messages are transmitted without security applied.
  - *PreSharedKey*
    - sensing devices that are pre-programmed with the symmetric cryptographic keys required to support secure communications
  - *RawPublicKey*
    - A given device must be pre-programmed with an asymmetric key pair
    - supports authentication based on public-keys
    - The device has an identity calculated from its public key, and a list of identities and public keys of the nodes it can communicate with.
    - This is mandatory for implementing CoAP
  - *Certificates*
    - The device has an asymmetric key pair with an X.509 certificate
    - supports authentication based on public-keys

CoAP Security

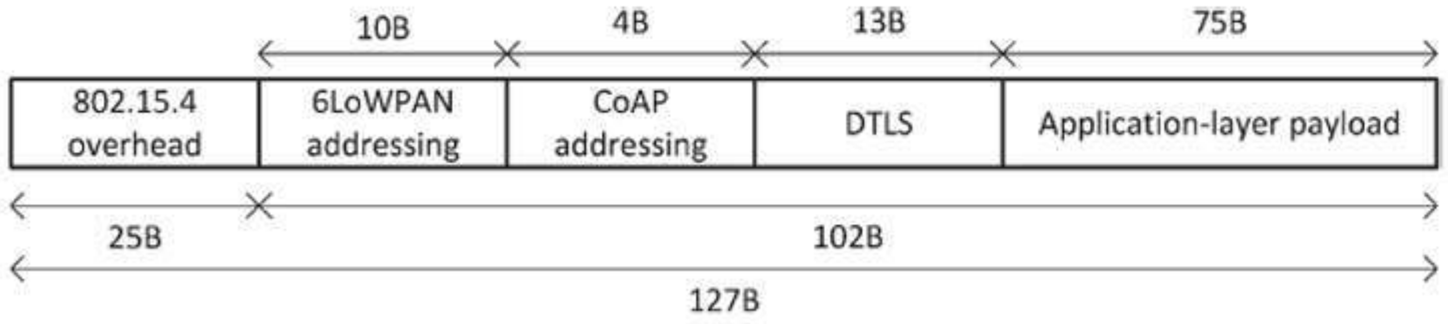


Fig: Payload space with DTLS on 6LoWPAN environments

- Elliptic Curve Cryptography (ECC) is adopted to support the ***RawPublicKey*** and ***Certificates*** security modes
  - ✓ ECC supports device authentication using the Elliptic Curve Digital Signature Algorithm (ECDSA), and also key agreement using the Elliptic Curve Diffie-Hellman Algorithm with Ephemeral keys (ECDHE).
- ***PreSharedKey*** security mode requires the TLS_PSK_WITH_AES_128_CCM_8 suite, which supports authentication using pre-shared symmetric keys and 8-byte nonce values, and encrypts and produces 8-byte integrity codes.

Operational Technology (OT) Level

- Security Requirements from three Operation Levels

1) **Information Level**

-Integrity: received data should not been altered during the transmission

-Anonymity: identity of the data source should remain hidden

- Confidentiality: Data cannot be read by third parties

-Privacy: The client's private information should not be disclosed.

2) **Access Level**

-Access Control: only legitimate users can access to the devices and the network for administrative tasks

- Authentication: checks whether a device has the right to access a network and vice-versa

- Authorization: ensures that only the authorized devices /users get access to the network services or resources.

Operational Technology (OT) Level

- Security Requirements from three Operation Levels

3) **Information Level**

Functional Level

- Resilience: refers to network capacity in case of attacks and failures
- Self Organization: denotes the capability of an IoT system to adjust itself in order to remain operational even in case of partial failure or attack