UNIT-4

IMPLEMENTATION OF IOT USING RASPBERRY PI

The basic implementation of IOT includes usage of a host device, a Remote Controllable Device and connectivity between them. In this paper, the host device can be a computer or a mobile phone and the remote controllable device is a Raspberry Pi, which executes the commands given by the master host. The implementation mechanism can be understood by the following figure

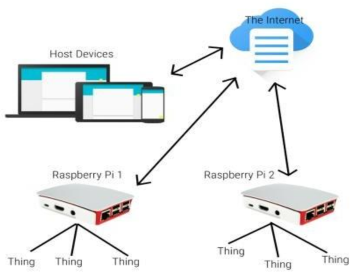


Fig 1: Block diagram of implementing the Internet of Things

The implementation requires a close association with both hardware and software. We will now discuss them briefly.

HARDWARE IMPLEMENTATION:

The system that implements the Internet of Things includes clusters of hardware components that we are familiar with. Firstly, we need a host like a Personal Computer or a mobile phone that can be used to pass commands to a remotely operable device. As the brain of the system we are using a Raspberry Pi that can be used to control and obtain a desired result from a device. The “things” that we use here are basically day-to-day objects like a bulb, a fan, a washing machine etc., Our intention is to show the operation of the Internet of Things in a concise way.

As the Raspberry Pi is more like a compact computer itself, it cannot control “things” directly. It needs an interface to communicate the with them. Fortunately, Raspberry Pi comes with a 40-pin GPIO set that could efficiently be utilized to communicate with the “things”. As we need an interface between them, a “Daughter Board” is to be designed.

This Daughter Board will enable us to dim and glow a light source. Switch ON/OFF electrical devices and receive feedback from sensors.

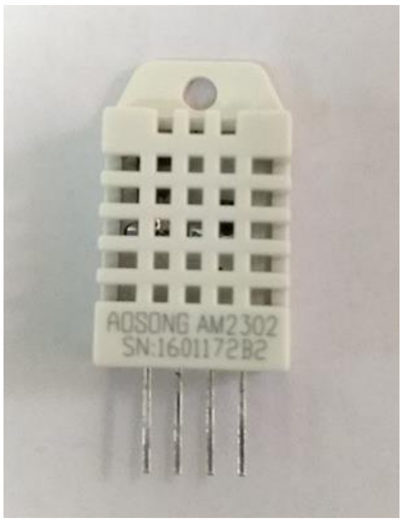
SOFTWARE IMPLEMENTATION:

Hardware without proper software is nothing but a piece a brick. When it comes to Raspberry Pi, an OS must be installed to control and configure it. And in the case of the Daughter Board, python scripts are to be coded to work with the "things". We have,a communications platform for IOT devices that enables device setup and user interaction from mobile devices and the web, can be used to accomplish communication between Host device and the Raspberry Pi.

Implementation of IoT with Raspberry Pi

Program1

IOT: Remote Data Logging

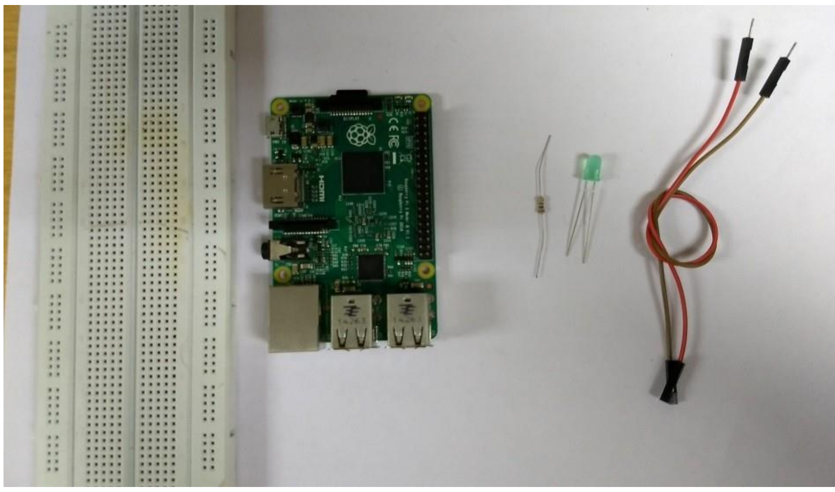


System Overview:

1. A network of Temperature and humidity sensor connected withRaspberry Pi
2. Read data from the sensor
3. Send it to a Server
4. Save the data in the server

Requirements

- **DHT Sensor**
- **4.7K ohm resistor**
- **Jumper wires**
- **Raspberry Pi**



Sending Data to a Server (contd..)

- Client Code: Obtain readings from the sensor

```
def sensordata():
        GPIO.setmode(GPIO.BOARD)
        GPIO.setwarnings(False)
        sensor = Adafruit_DHT.AM2302
        humidity, temperature = Adafruit_DHT.read_retry(sensor,17)
        return(humidity, temperature)
```

This function returns the values from the DHT sensor

Client Code: Connecting to the server and sending the data

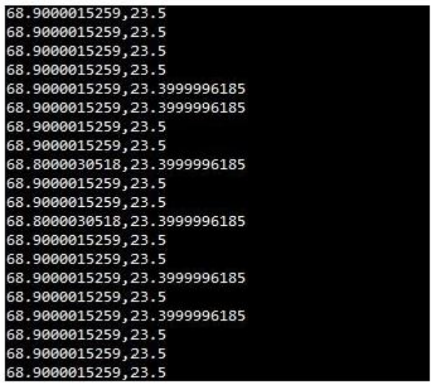
```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)              #create UDP socket
server_address = ('10.14.3.194', 10001)
sock.connect(server_address)
try:
   while (1):
            h,t = sensordata()
            message = str(h)+','+str(t)
            #Send data
            print >>sys.stderr, 'sending "%s"' % message

            sent = sock.sendto(message, server_address)
finally:
   print >>sys.stderr, 'closing socket'
   sock.close()
```

Server Code: Receive data from client and save it

```
                                                                     # Create a UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                                                                     # Bind the socket to the port
server_address = ('10.14.3.194', 10001)
sock.bind(server_address)
while True:
   data, address = sock.recvfrom(4096)
   with open("Datalog.txt","a") as f:
      mess=str(data)
      f.write(mess)
      print mess
   f.close()
```

Result

- The client takes reading from the sensor and sends it to the server
- The server receives the data from the client and saves it in a text file DataLog.txt



```
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.8000030518,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.8000030518,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
```

Program2

DataProcessing Technique

Data from the client needs to be processed before it can be used further

Data splitting/filtering

Data plotting

Data Processing

Data splitting/filtering:

- Data from the client is saved in a text file
- The values are separated by a comma(' , ')

  message = str(h)+','+str(t)

- Split() function can be used to split a string into multiple strings depending on the type of separator/delimiter specified.

  Example:

Data= 'sunday,monday,tuesday'	#Data is a string with 3 words separated by a comma
Data.split(",")	# split the data whenever a "," is found
['sunday','monday','tuesday']	# Gives 3 different strings as output

Plotting the data:

MATPLOTLIB is a python library used to plot in 2D

Plot(x,y): plots the values x and y

xlabel('X Axis'): Labels the x-axis

ylabel('Y Axis'): Labels the y-axis

title("Simple Plot"): Adds title to the plot

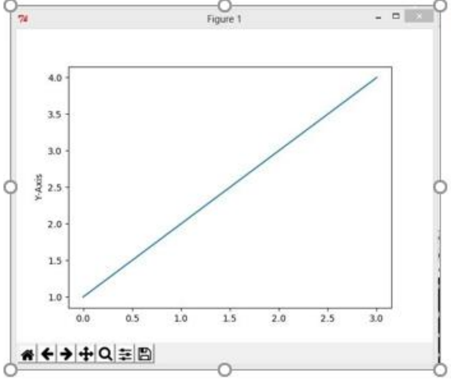
Plotting the data:

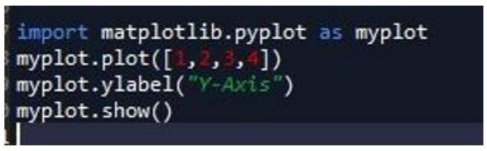
- import matplotlib.pyplot as myplot
  myplot.plot([1,2,3,4])
  myplot.ylabel('Y-Axis')
  myplot.show()

By default the values are taken for y-axis, values for x-axis are generated automatically starting from 0

Data Processing (contd..)

- Basic Plot:



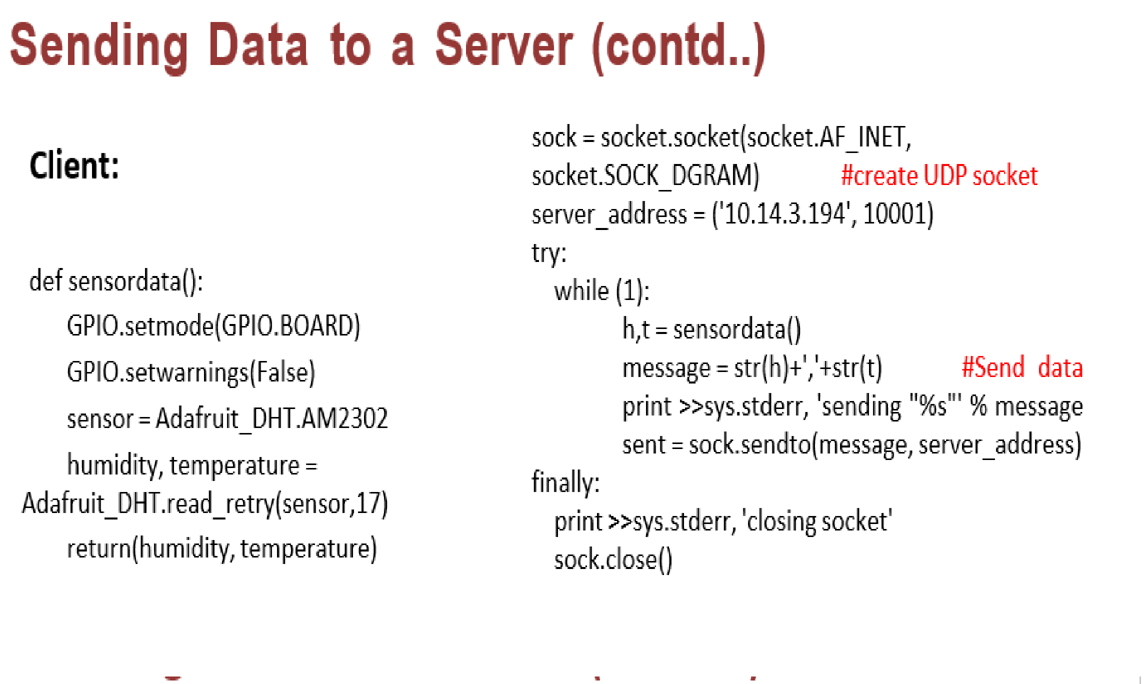


```
import matplotlib.pyplot as myplot
myplot.plot([1,2,3,4])
myplot.ylabel("Y-Axis")
myplot.show()
```

Some other common functions used in plotting:

- figure(): Creates a new figure
- grid(): Enable or disable axis grids in the plot
- ion(): turns on the interactive mode
- subplot(): Adds subplot in a figure

- Close(): Close the current figure window
- Scatter(): make a scatter plot of the given points

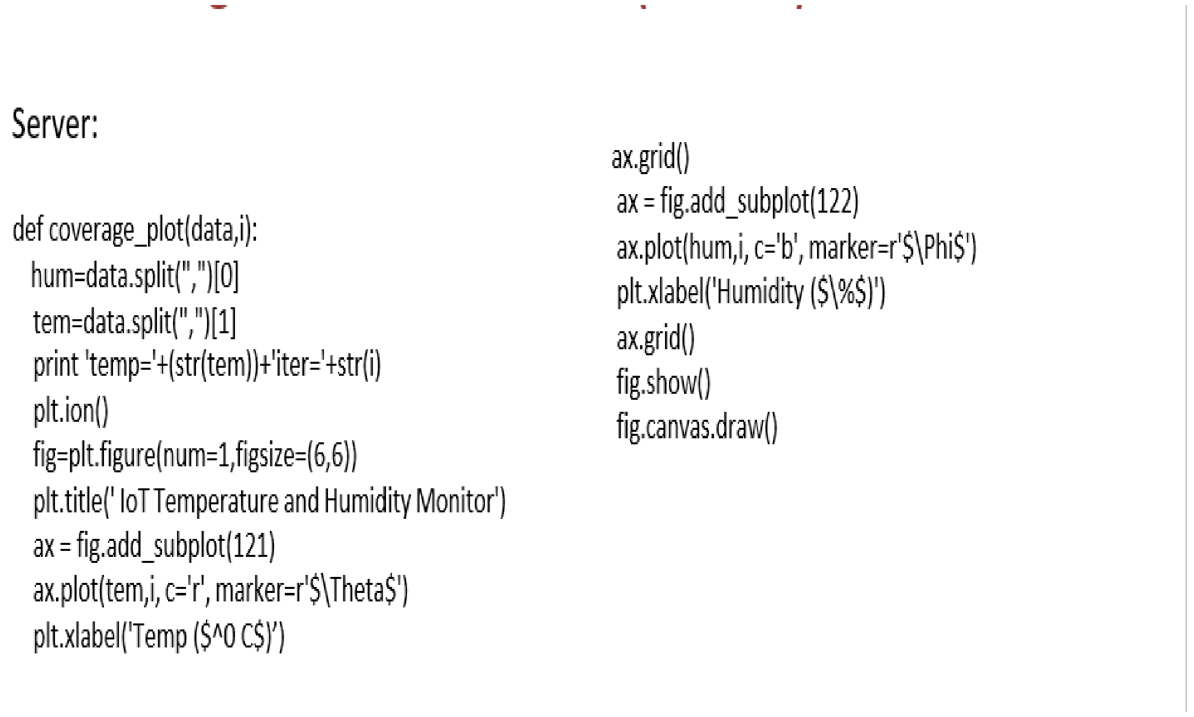


Sending Data to a Server (contd..)

Client:

```
def sensordata():
      GPIO.setmode(GPIO.BOARD)
      GPIO.setwarnings(False)
      sensor = Adafruit_DHT.AM2302
      humidity, temperature =
Adafruit_DHT.read_retry(sensor,17)
      return(humidity, temperature)
```

```
sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)          #create UDP socket
server_address = ('10.14.3.194', 10001)
try:
   while (1):
          h,t = sensordata()
          message = str(h)+','+str(t)          #Send  data
          print >>sys.stderr, 'sending "%s"' % message
          sent = sock.sendto(message, server_address)
finally:
   print >>sys.stderr, 'closing socket'
   sock.close()
```



Server:

```
def coverage_plot(data,i):
  hum=data.split(",")[0]
  tem=data.split(",")[1]
  print 'temp='+(str(tem))+'iter='+str(i)
  plt.ion()
  fig=plt.figure(num=1,figsize=(6,6))
  plt.title(' IoT Temperature and Humidity Monitor')
  ax = fig.add_subplot(121)
  ax.plot(tem,i, c='r', marker=r'$\Theta$')
  plt.xlabel('Temp ($^0 C$)’)
```

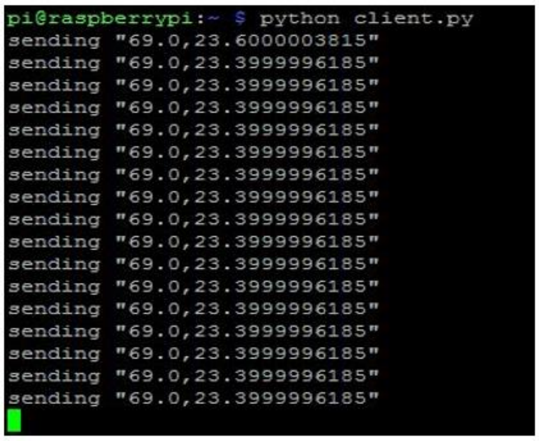
```
ax.grid()
ax = fig.add_subplot(122)
ax.plot(hum,i, c='b', marker=r'$\Phi$')
plt.xlabel('Humidity ($\%$)')
ax.grid()
fig.show()
fig.canvas.draw()
```

Server:

```
# Bind the socket to the port

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_address = ('10.14.3.194', 10001)
sock.bind(server_address)

i=0
while True:
  data, address = sock.recvfrom(4096)
  with open("DataLog.txt","a") as f:
    mess=str(data)
    f.write(mess)
    coverage_plot(mess,i)
    print mess
    i+=1
  f.close()
```

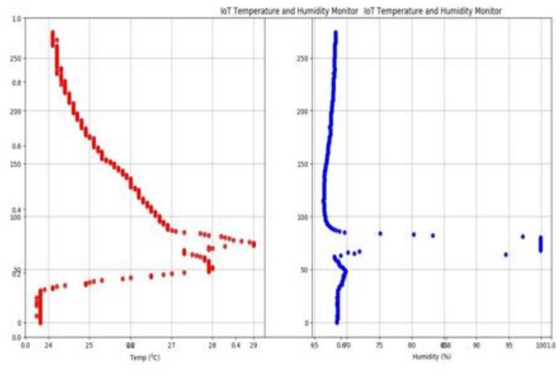
Output

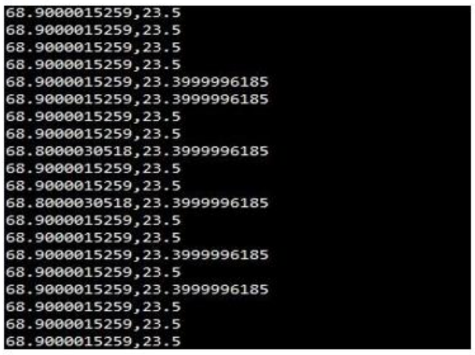
- The Reading from the sensor is sent to the Server and saved in a text file.
- Two different plots for temperature and humidity data



```
pi@raspberrypi:~ $ python client.py
sending "69.0,23.6000003815"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
```

Output





```
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.8000030518,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.8000030518,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
```

Introduction to SDN

Software-defined networking (SDN) is an architecture designed to make a network more flexible and easier to manage. SDN centralizes management by abstracting the control plane from the data forwarding function in the discrete networking devices.

SDN elements

An SDN architecture delivers a centralized, programmable network and consists of the following:

- A controller, the core element of an SDN architecture, that enables centralized management and control, automation, and policy enforcement across physical and virtual network environments
- Southbound APIs that relay information between the controller and the individual network devices (such as switches, access points, routers, and firewalls)
- Northbound APIs that relay information between the controller and the applications and policy engines, to which an SDN looks like a single logical network device

Origin of SDN

2006: At Stanford university, a team proposes a clean-slate security architecture (SANE) to control security policies in a centralized manner instead of doing it at edges.

2008: The idea of *software-defined network* is originated from OpenFlow project(*ACM SIGCOMM 2008*).

2009: Stanford publishes OpenFlow V1.0.0 specs.

June 2009: Nicira network is founded.

March 2011: Open Networking Foundation is formed.

Oct 2011: First Open Networking Summit. Many Industries (Juniper, Ciscoannounced to incorporate.

•

In order to understand software defined networks, we need to understand the various planes involved in networking.

1.Data plane:
All the activities involving as well as resulting from data packets sent by the end user belong to this plane. This includes:

- Forwarding of packets
- Segmentation and reassembly of data
- Replication of packets for multicasting

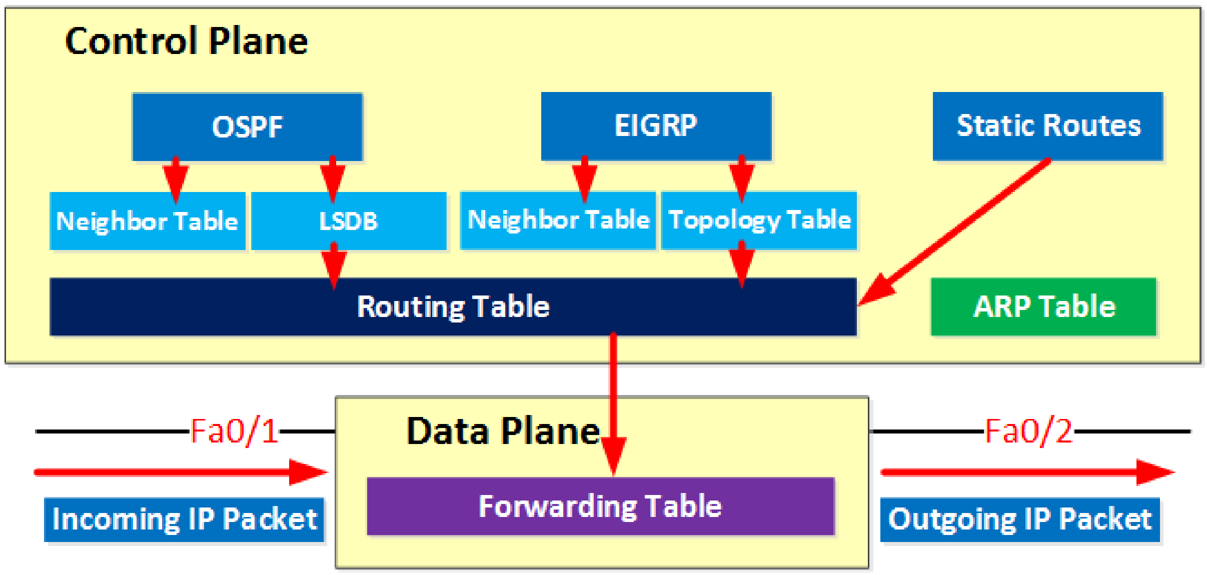
2.Control plane:
All activities necessary to perform data plane activities but do not involve end user data packets belong to this plane. In other words, this is the brain of the network. The activities of the control plane include:

- Making routing tables
- Setting packet handling policies

3.Management Plane

The management plane is used for access and management of our network devices. For example, accessing our device through telnet, SSH or the console port.

When discussing SDN, the control and data plane are the most important to keep in mind. Here's an illustration of the control and data plane to help you visualize the different planes:



SDN Architecture

SDN attempts to create network architecture that is simple, inexpensive, scalable, agile and easy to manage. The below figure shows the SDN architecture and SDN layers in which the control plane and data plane are decoupled and the network controller is centralized. SDN controller maintains a unified view of the network and make configurations, management and provisioning simpler. The underlying infrastructure in SDN uses simple packet forwarding hardware.

Network devices become simple with SDN as they do not require implementations of a large no of protocols. Network devices receive instructions from SDN controller on how to forward packets. These devices can be simpler and costless as they can be built from standard hardware and software components.

SDN architecture separates the network into three distinguishable layers, i.e., applications communicate with the control layer using northbound API and control layer communicates with data plane using southbound APIs. The control layer is considered as the brain of SDN. The intelligence to this layer is provided by centralized SDN controller software. This controller resides on a server and manages policies and the flow of traffic throughout the network. The physical switches in the network constitute the infrastructure layer.

An SDN architecture contains six major components.

First is the **management plane**, which is a set of network applications that manage the control logic of a software-defined network. Rather than using a command line interface, SDN-enabled networks use programmability to give flexibility and easiness to the task of implementing new applications and services, such as routing, load balancing, policy enforcement, or a custom application from a service provider. It also allows orchestration and automation of the network via existing APIs [1].

Second is the **control plane** that is the most intelligent and important layer of an SDN architecture. It contains one or various controllers that forward the different types of rules and policies to the infrastructure layer through the southbound interface [1].

Third, the **data plane**, also known as the infrastructure layer, represents the forwarding devices on the network (routers, switches, load balancers, etc.). It uses the southbound APIs to interact with the control plane by receiving the forwarding rules and policies to apply them to the corresponding devices [1].

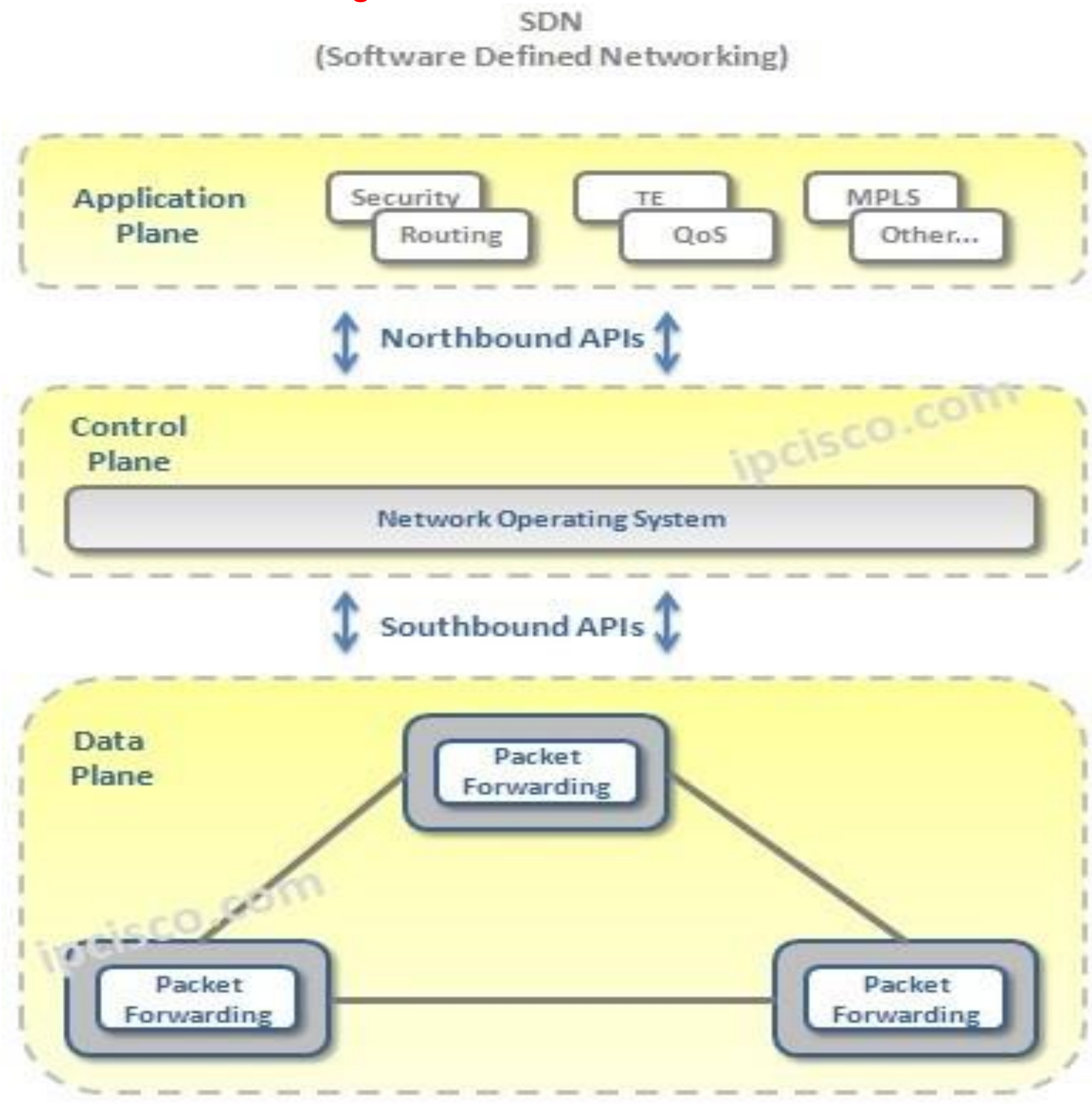
Fourth, the **northbound interfaces** that permit communication between the control layer and the management layer are mainly a set of open source application programming interfaces (APIs) [1].

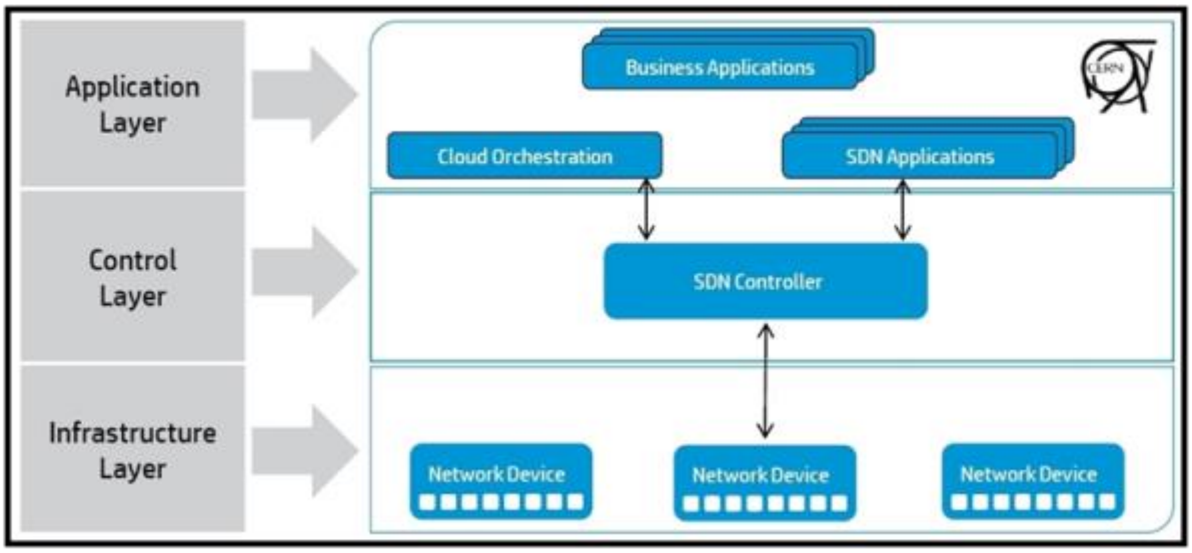
Fifth, the **east-west interfaces**, which are not yet standardized, allow communication between the multiple controllers. They use a system of notification and messaging or a distributed routing protocol like BGP and OSPF.

Sixth, the **southbound interfaces** allow interaction between the control plane and the data plane, which can be defined summarily as protocols that permit the controller to push policies to the forwarding plane**. The OpenFlow protocol is the most widely accepted and implemented southbound API for SDN-enabled networks.**

OpenFlow is normalized by the Open Networking Foundation (ONF) OpenFlow is just an instantiation of SDN

SDN Architecture Diagram.





More about OPENFLOW Architecture

In an OpenFlow-enabled network, flow can be represented as a transmission control protocol (TCP) connection. Flows can also be packets with a matching MAC address, an IP address, a VLAN tag, or a switch port [6].

The OpenFlow switch has one or more flow tables. A flow table is a set of flow entries. A flow entry is used to match and process packets. It consists of many matching fields to match packets, a set of encounters to track packets, and instructions to apply. The OpenFlow switch uses an OpenFlow channel to communicate with the OpenFlow controller.

The OpenFlow channel is a secure channel between the OpenFlow switch and the OpenFlow controller. It permits communication by allowing the control plane to send instructions, receive requests, or exchange information. All messages are encrypted, using transport layer security.

The OpenFlow channel has three types of messages.

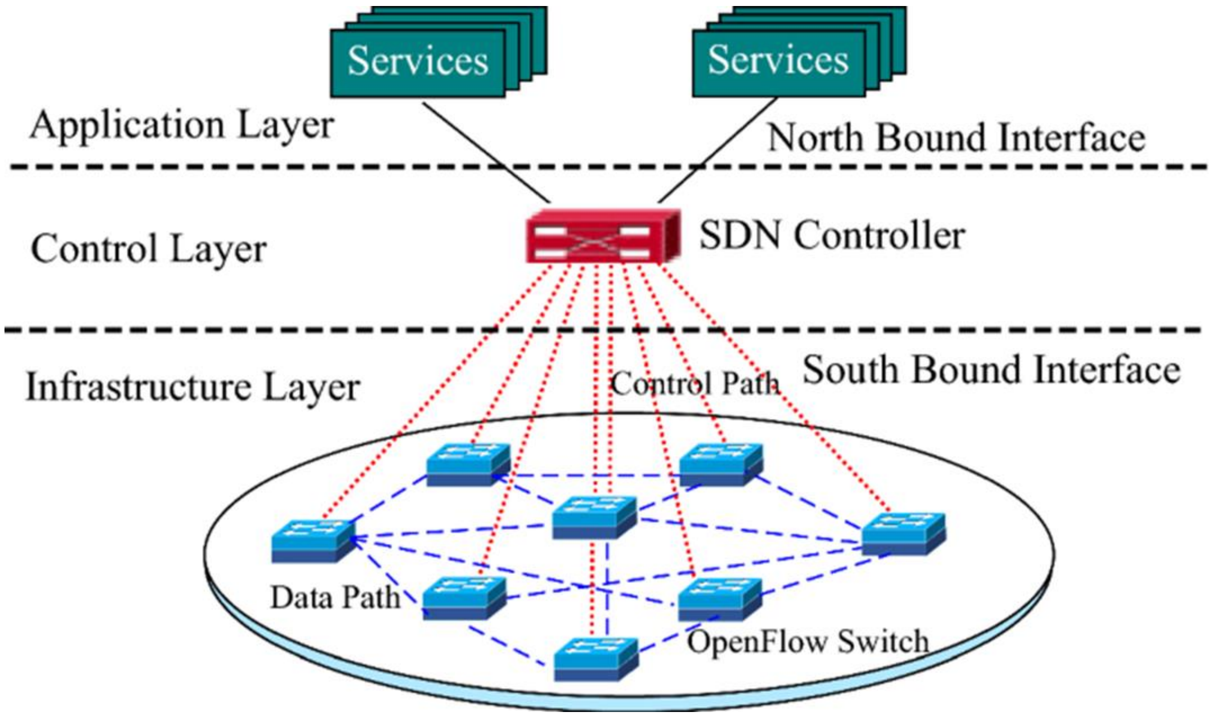
The controller/switch message is initiated by the controller and may not require a response from the switch.

The asynchronous message informs the controller about a packet arrival, a switch state change, or an error.

The symmetric message can be sent in both directions for other purposes .

The OpenFlow controller handles flow tables inside the switch by adding and removing flow entries. It uses the OpenFlow channel to send and receive information [9]. It can be considered as an operating system that serves the whole network.

The OpenFlow protocol is the southbound interface that permits communication between the OpenFlow controller and the OpenFlow switch via the OpenFlow channel



The OpenFlow switch may be programmed to:

(1) identify and categorize packets from an ingress port based on a various packet header fields.

(2) Process the packets in various ways, including modifying the header; and,

(3) Drop or push the packets to a particular egress port or to the OpenFlow Controller.

The OpenFlow instructions transmitted from an OpenFlow Controller to an OpenFlow switch are structured as "flows". Each individual flow contains packet match fields, flow priority, various counters, packet processing instructions, flow timeouts and a cookie. The flows are organized in tables. An incoming packet may be processed by flows in multiple "pipelined" tables before exiting on an egress port. The OpenFlow protocol standard is evolving quickly with release 1.6 as the current revision at the time of this blog being published.

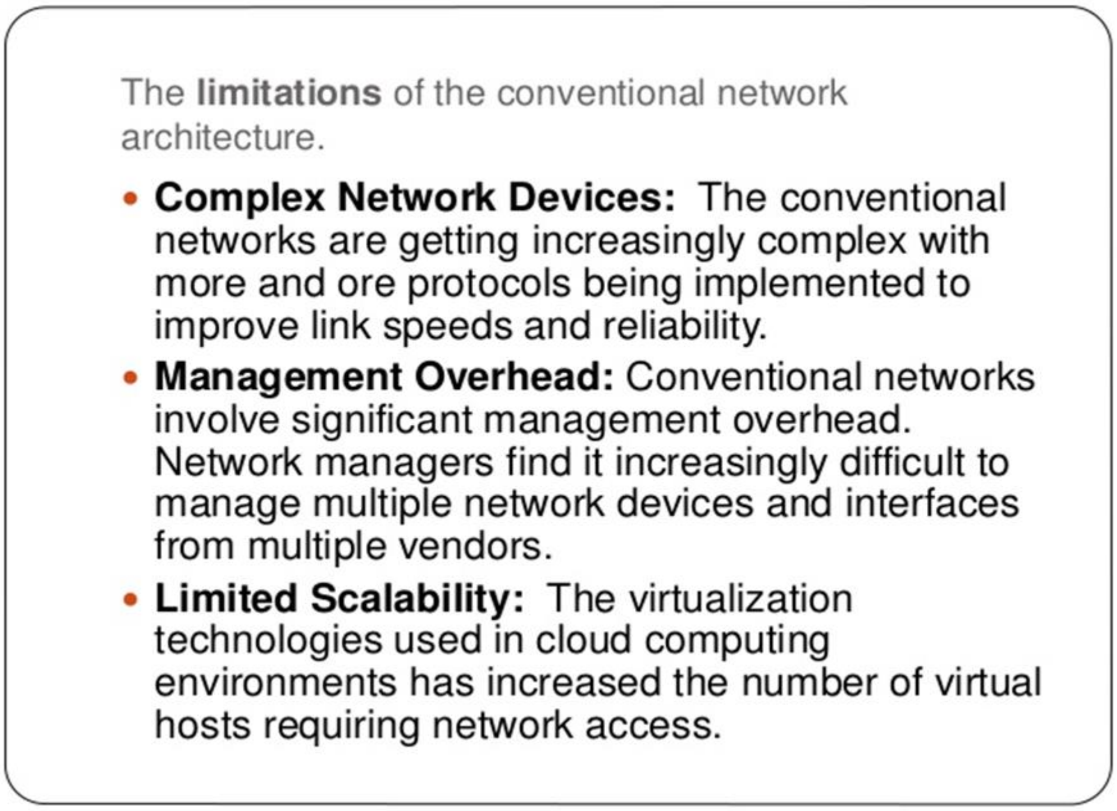
The OpenFlow Network Architecture consists of three layers:

(1) One or more OpenFlow virtual and/or physical switches;
(2) One or two OpenFlow controller(s); and,
(3) One or more OpenFlow application(s). For an illustration, see figure 2 below.

The OpenFlow controller maintains the OpenFlow protocol communications channels to the OpenFlow switches, maintains a local state graph of the OpenFlow switches and exposes a northbound API to the OpenFlow applications. The northbound API may be viewed as an abstraction of the network and allows the OpenFlow applications to read the state of the network and to instruct the network to perform various tasks.

How SDN is different from Conventional or Traditional Architecture

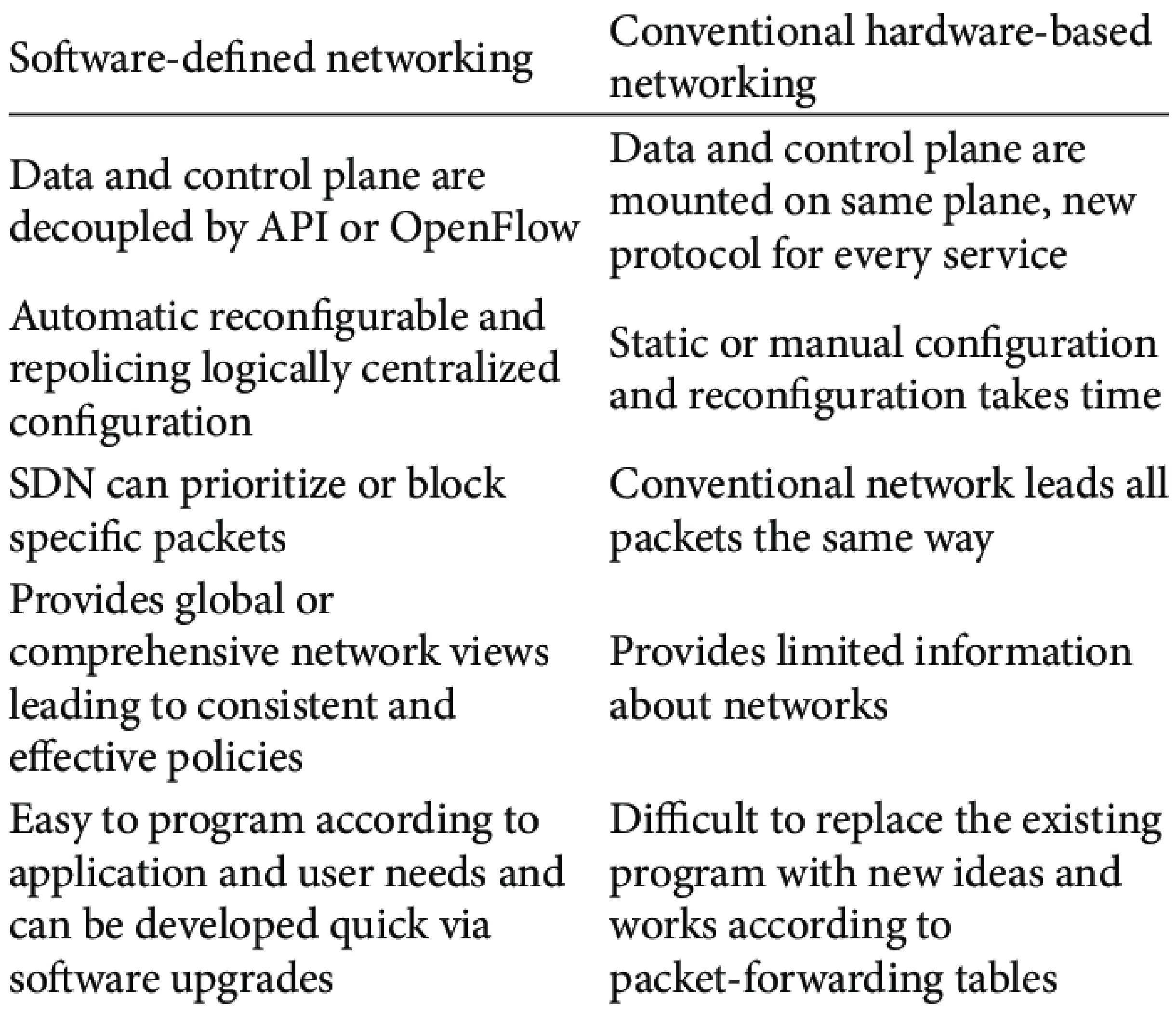
In Conventional architecture the control plane and data plane are coupled . Control plane is part of the network that carries the signalling and routing message traffic while the data plane is part of the network that carries the payload traffic.



The **limitations** of the conventional network architecture.

- **Complex Network Devices:** The conventional networks are getting increasingly complex with more and ore protocols being implemented to improve link speeds and reliability.
- **Management Overhead:** Conventional networks involve significant management overhead. Network managers find it increasingly difficult to manage multiple network devices and interfaces from multiple vendors.
- **Limited Scalability:** The virtualization technologies used in cloud computing environments has increased the number of virtual hosts requiring network access.

Differnce between SDN and Conventional/Traditional architecture



Software-defined networking	Conventional hardware-based networking
Data and control plane are decoupled by API or OpenFlow	Data and control plane are mounted on same plane, new protocol for every service
Automatic reconfigurable and repolicing logically centralized configuration	Static or manual configuration and reconfiguration takes time
SDN can prioritize or block specific packets	Conventional network leads all packets the same way
Provides global or comprehensive network views leading to consistent and effective policies	Provides limited information about networks
Easy to program according to application and user needs and can be developed quick via software upgrades	Difficult to replace the existing program with new ideas and works according to packet-forwarding tables

**Key Elements of SDN

1.Centralized Network controller :

With decoupled control and data planes and centralized network controller , the network administrators can rapidly configure the network. SDN applications can be deployed through programmable open APIs. This speed up innovation as the network administrator no longer need to wait for the device vendors to embed new features in their proprietary hardware.

2.Programmable Open API's:

SDN architecture supports programmable open API's for interface between the SDN application and control layers.(North bound Interface), with these open APi's various network services can be implemented., such as routing ,Quality of service ,access control etc.

3.Standard communication interface API's :

SDN architecture uses a standard communication interface between the control and infrastructure layers(southbound interface).

OpenFlow, which is defined by the Open Networking Foundation (ONF) is the broadly accepted SDN protocol for the Southbound Interface.

With Open flow, the forwarding Plane of the network device can be directly accessed and manipulated.

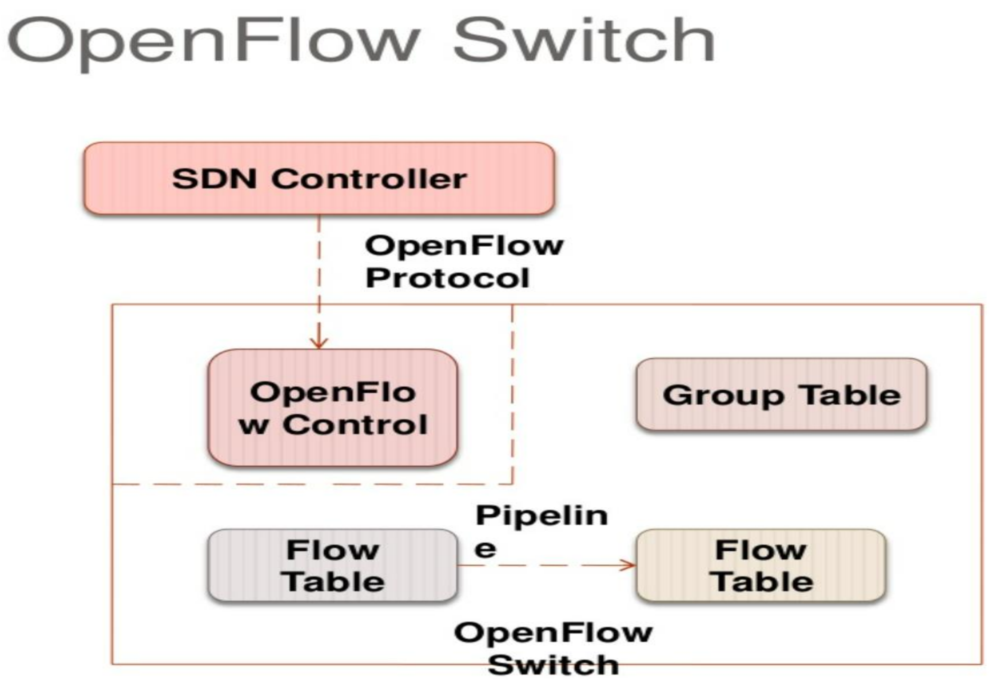
Open Flow uses the concept of flows to identify network traffic based on pre-defined match rules. Flows can be programmed statically or dynamically by the SDN control software.

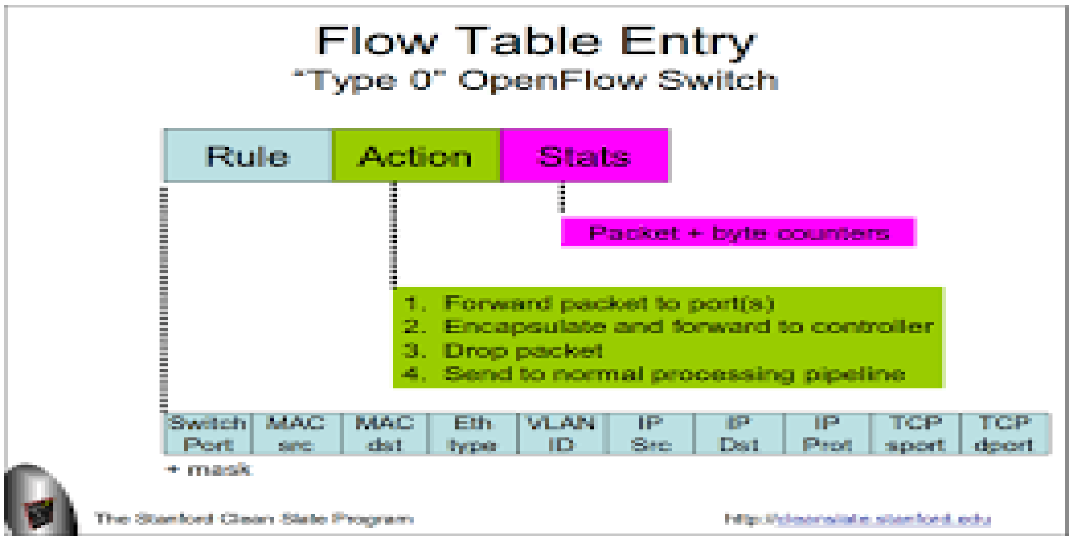
Open Flow protocol is implemented on both sides of the interface between the controller and the network devices.

The controller manages the switch via the OpenFLow Switch Protocol

The controller can add ,update, delete flow entries in the flow table .

Each Flow entries contains the match fields , counters and set of instructions to apply matching packets





Challenges to implement SDN(RULE Placement and CONTROLLER placement)

SDN Rule Placement Problem

A rule placement solution deals with which rules must be deployed in the network and where.

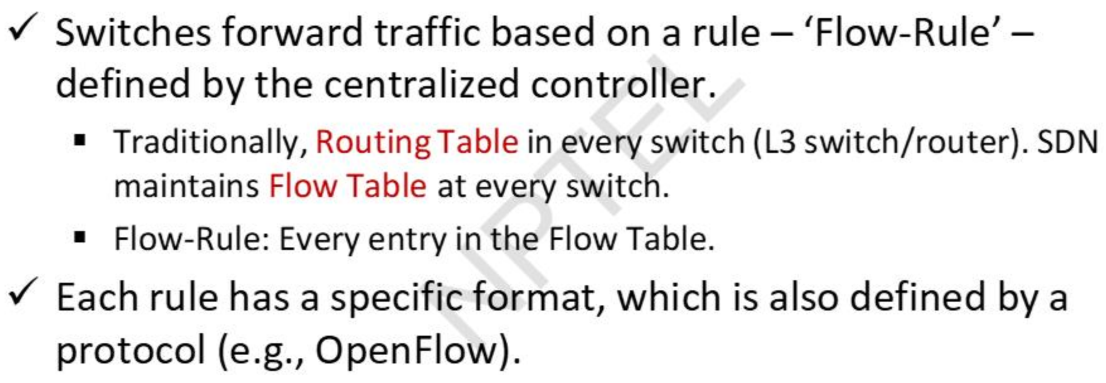
SDN Controller Placement Problem

•How many controllers are required

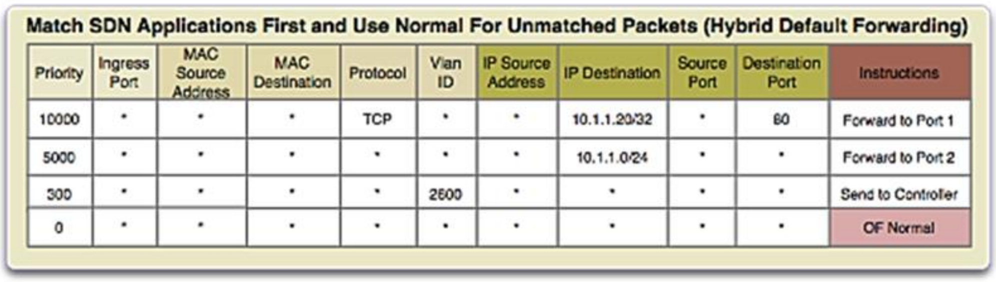
• Where to place them

• What is the controller domain (switches controlled by each controller)

SDN Rule Placement



- ✓ Switches forward traffic based on a rule – 'Flow-Rule' – defined by the centralized controller.
  - Traditionally, Routing Table in every switch (L3 switch/router). SDN maintains Flow Table at every switch.
  - Flow-Rule: Every entry in the Flow Table.
- ✓ Each rule has a specific format, which is also defined by a protocol (e.g., OpenFlow).



Match SDN Applications First and Use Normal For Unmatched Packets (Hybrid Default Forwarding)

Priority	Ingress Port	MAC Source Address	MAC Destination	Protocol	Vlan ID	IP Source Address	IP Destination	Source Port	Destination Port	Instructions
10000	*	*	*	TCP	*	*	10.1.1.20/32	*	80	Forward to Port 1
5000	*	*	*	*	*	*	10.1.1.0/24	*	*	Forward to Port 2
300	*	*	*	*	2600	*	*	*	*	Send to Controller
0	*	*	*	*	*	*	*	*	*	OF Normal

TCAM is used to store the flow rules received from controller at switch

✓ Size of ternary content-addressable memory (TCAM) is limited at the switches.

- Limited number of rules can be inserted.

✓ Fast processing is done using TCAM at the switches.

✓ TCAM is very cost-expensive.

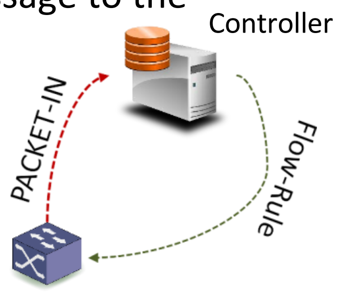
If flow rule is not available in switch then steps taken are:

✓ On receiving a request, for which no flow-rule is present in the switch, the switch sends a *PACKET-IN* message to the controller.

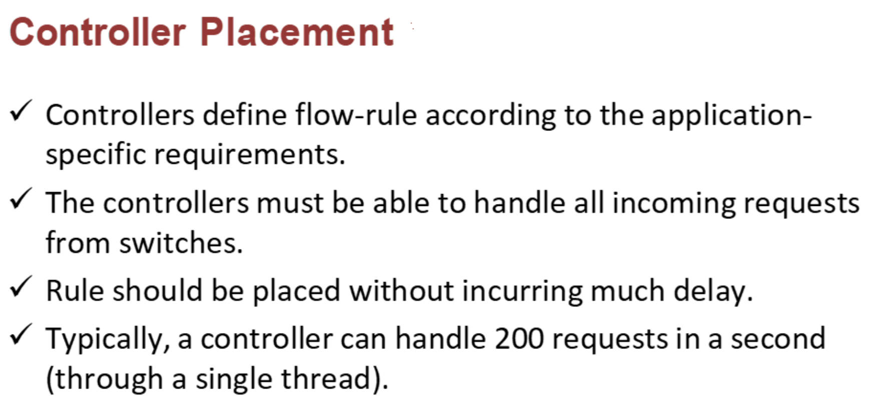
✓ The controller decides a suitable flow-rule for the request.

✓ The flow-rule is inserted at the switch.

✓ Typically, 3-5ms delay is involved in a new rule placement



Controller placement



Controller Placement

✓ Controllers define flow-rule according to the application-specific requirements.

✓ The controllers must be able to handle all incoming requests from switches.

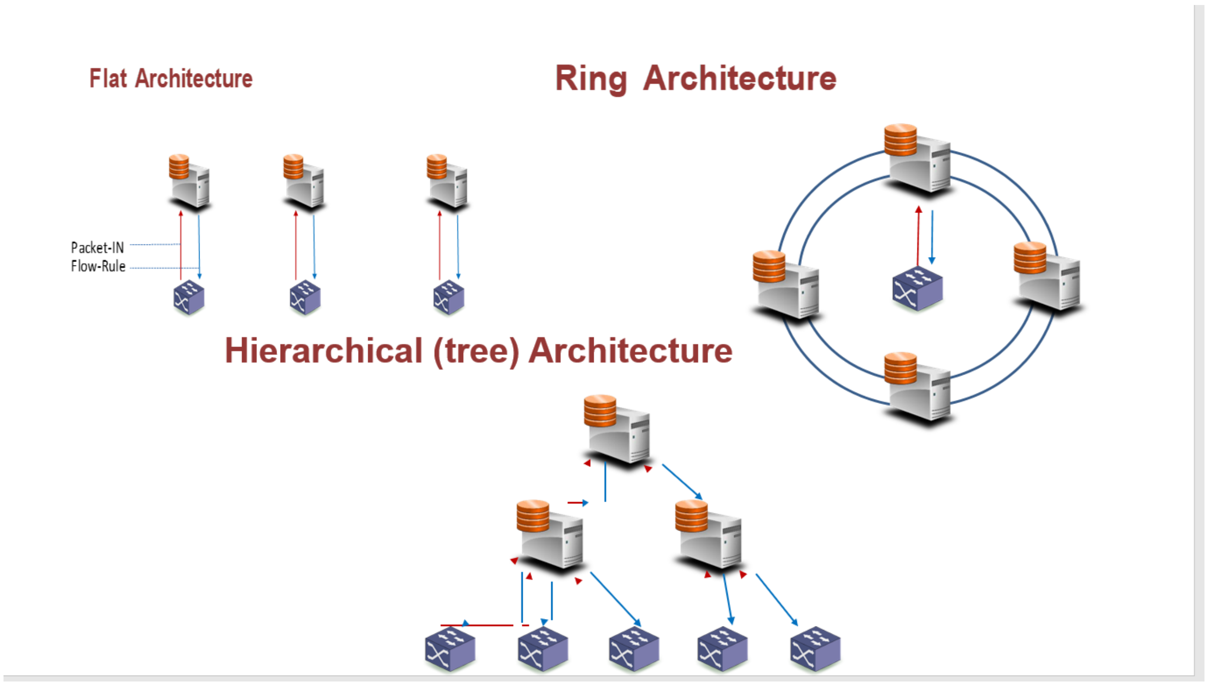
✓ Rule should be placed without incurring much delay.

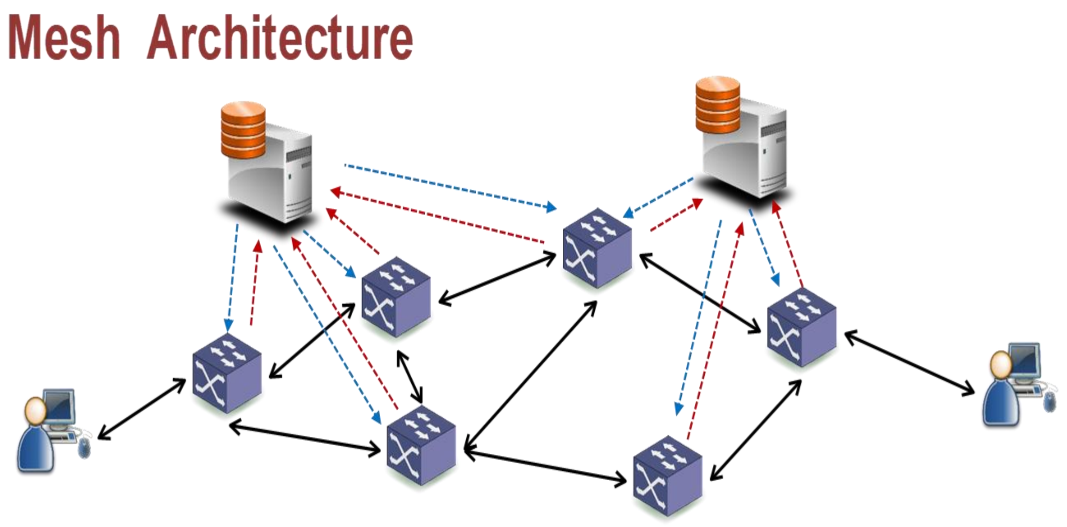
✓ Typically, a controller can handle 200 requests in a second (through a single thread).

✓ **The controllers are logically connected to the switches in <u>one-</u> <u>hop</u> distance.**

- **Physically, they are connected to the switches in multi-hop distance.**

✓ **If we have a very small number of controllers for a large network, the network might be congested with control packets (i.e., PACKET-IN messages).**





In a flat or horizontal architecture, the controllers are positioned horizontally on one single level. In other words, the control plane consists of just one layer, and each controller has the same responsibilities at the same time and has a partial view of its network.
In a hierarchical or vertical architecture, the controllers are positioned vertically. They are portioned among multiple levels, which means that the control plane has several layers, generally two or three. The controllers have different

responsibilities, and they can take decisions based on a partial view of the network.

These two methods have many advantages and disadvantages; for example, both of these approaches can improve the switch/controller latency in comparison to a single controller architecture or a multicore architecture. In flat design, the network provides more resilience to failures. However, the task of managing controllers becomes harder. On the other hand, a hierarchical design gives a simpler way to manage the controllers, but the problem of a single point of failure remains, because of the upper layer of the control plane. To explain more this final idea, in a hierarchical architecture, we usually have about three layers. Each layer contains a type of controllers. Typically, the bottom layer contains the local controllers, while the upper layer contains one root controller, which means that we have the problem of a single point of failure, even if it concerns just one layer of the control plane.

Backup Controller

- ✓ **If a controller is down, what will happen?**
  - ▪ **Backup controller is introduced**
  - ▪ **Replica of the main controller is created**
  - ▪ **If the main controller is down, backup controller controls the network to have uninterrupted network management.**

SDN for IOT

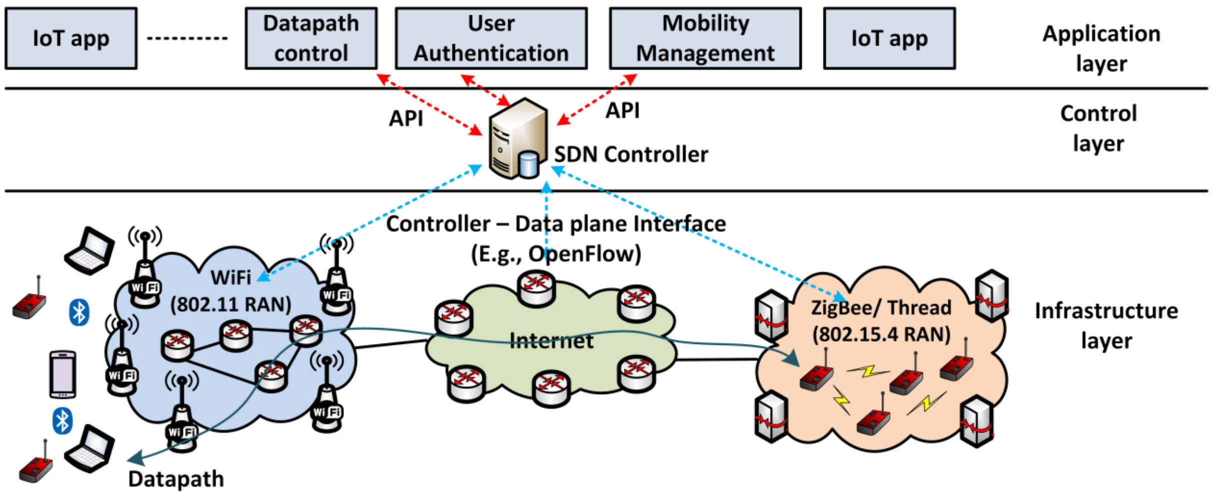
Benefits of Integrating SDN in IOT

1.Intelligent routing decisions can be deployed using SDN

2.Simplification of information collection, analysis and decision making

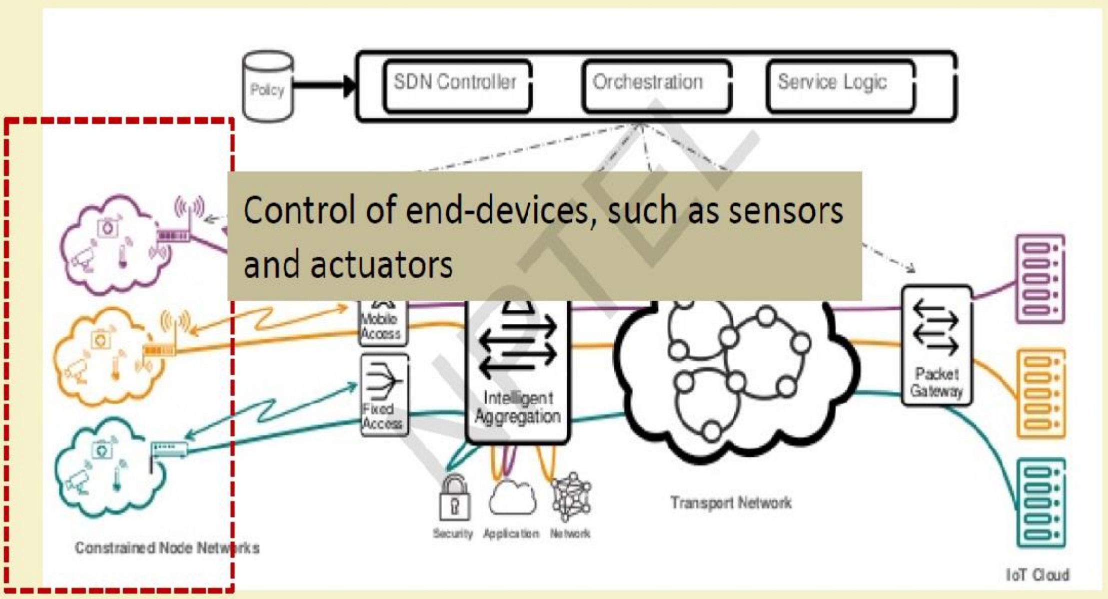
3.Visibility of network resources –network management is simplified based on user, device and application-specific requirements

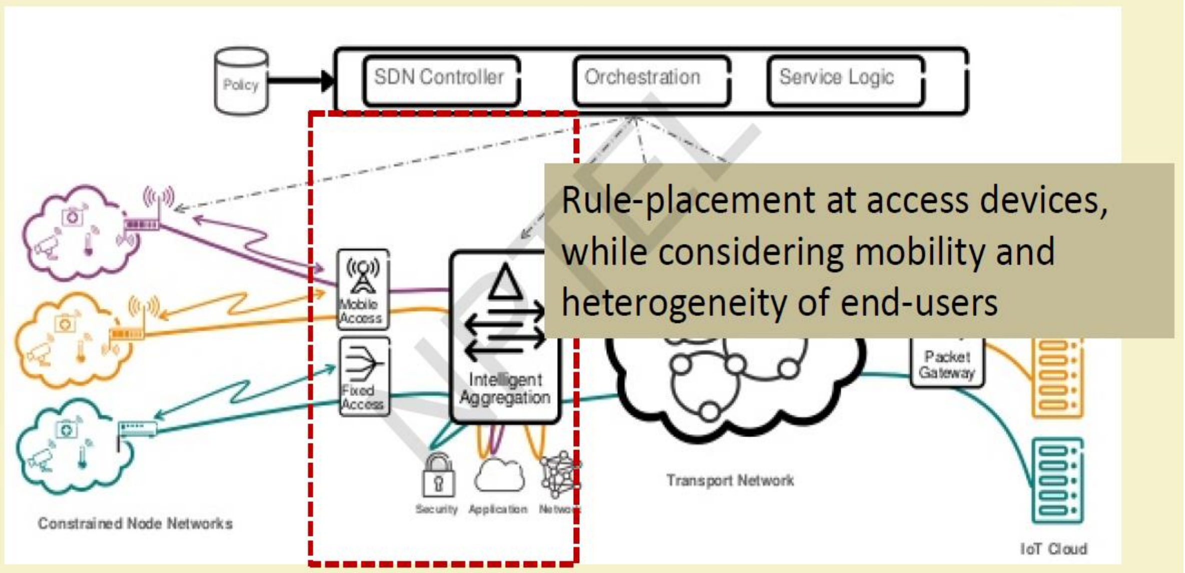
4.Intelligent traffic pattern analysis and coordinated decisions

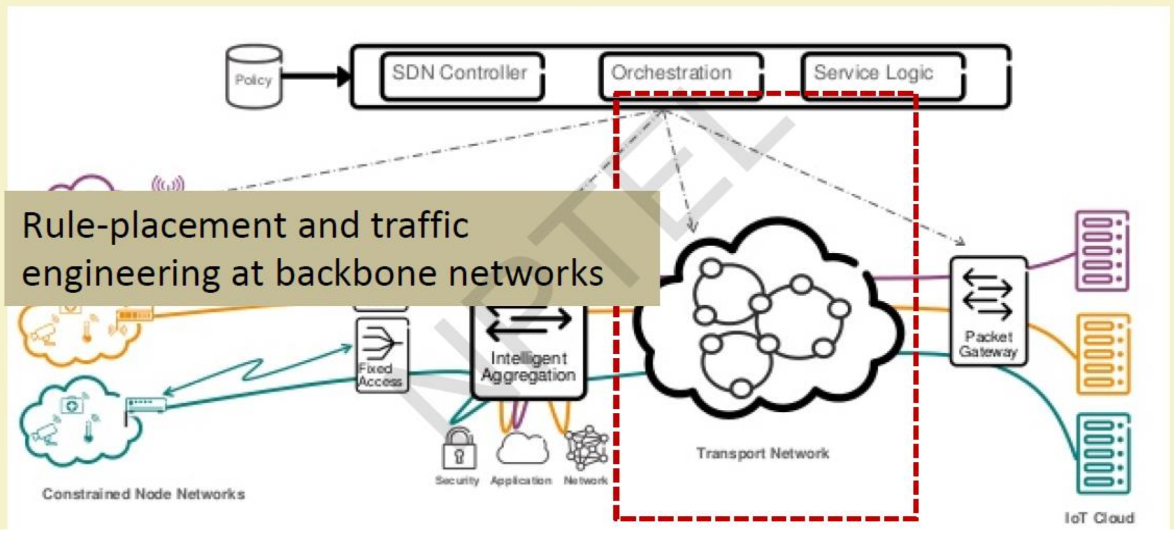


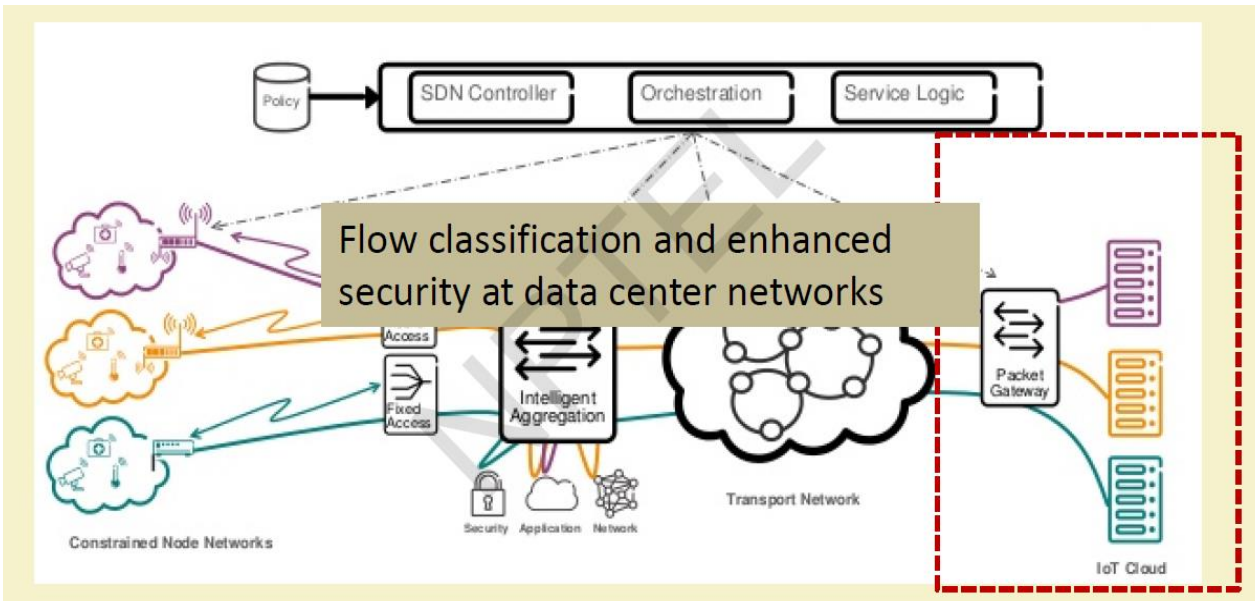
software-defined networking (SDN) is a promising approach to control the network in a unified manner using rule-based management. The abstractions provided by SDN enable holistic control of the network using high-level policies, without being concerned about low-level configuration issues. Hence, it is advantageous to address the heterogeneity and application-specific requirements of IoT.

The above image shows how sdn is implemented in IOT


Control of end-devices, such as sensors and actuators


Rule-placement at access devices, while considering mobility and heterogeneity of end-users


Rule-placement and traffic engineering at backbone networks



In the above fig we can see different type of data is transferred from different IOT devices through subnetworks or fixed axis and the data that is acquired is transmitted to data aggregator here all the data aggregation is done by utilizing the rule placement available at access device that is provided by SDN controller and passes the data packets through transport network where it passes through different Gateways and finally the packets are sent to the end sources.

DATA HANDLING AND ANALYTICS

Internet of Things devices can open a whole new world of data for organizations.

What is Data Handling

Data handling

Ensures that research data is stored, archived or disposed off in a safe and secure manner during and after the conclusion of a research project

Includes the development of policies and procedures to manage data handled electronically as well as through non-electronic means.

In recent days, most data concern

1. Big Data
2. Due to heavy traffic generated by IoT devices
3. Huge amount of data generated by the deployed sensors

Data Handling

In Internet of Things (IoT)

According to Techopedia, IoT "*describes a future where every day physical objects will be connected to the internet and will be able to identify themselves to other devices.*"

For example lets consider sensor devices where these Sensors are embedded into various devices and machines and deployed into fields. These sensors transmit sensed data to remote servers via Internet. The Continuous data acquisition from mobile equipment, transportation facilities, public facilities, and home appliances are produced now huge challenge is how to handle all the data that is received from various devices and how to store this huge data.

Data handling at data centers

Storing, managing, and organizing data.
Estimates and provides necessary processing capacity.
Provides sufficient network infrastructure.
Effectively manages energy consumption.
Replicates data to keep backup.
Develop business oriented strategic solutions from big data.
Helps business personnel to analyze existing data.
Discovers problems in business operations.

What is Big Data

Definition1:

"Big data technologies describe a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling the high-velocity capture, discovery, and/or analysis."
[Report of International Data Corporation (IDC)]

Definition2

"Big data shall mean the data of which the data volume, acquisition speed, or data representation limits the capacity of using traditional relational methods to conduct effective analysis or the data which may be effectively processed with important
horizontal zoom technologies."
[National Institute of Standards and Technology (NIST)]

Types of Data

Structured data

1.Data that can be easily organized.
2.Usually stored in relational databases.
3.Structured Query Language (SQL) manages structured data in databases.

4.It accounts for only 20% of the total available data today in the world.

Unstructured data

1.Information that do not possess any pre-defined model.
2. Traditional RDBMSs are unable to process unstructured data.
3. Enhances the ability to provide better insight to huge datasets.
4.It accounts for 80% of the total data available today in the world

Characteristics of Big Data

Big Data is characterized by 7 Vs –

1. **V**olume
2. **V**elocity
3. **V**ariety
4.**V**ariability
5.Veracity
6. **V**isualization
7.Value

Volume

Quantity of data that is generated
Sources of data are added continuously

Example of *volume* -

1. 30TB of images will be generated every night from the Large Synoptic Survey Telescope

(LSST)
2. 72 hours of video are uploaded to YouTube every minute

Velocity

*Refers to the speed of generation of data
*Data processing time decreasing day-by-day in order to provide real-time services
*Older batch processing technology is unable to handle high velocity of data
Example of *velocity* –
1.140 million tweets per day on average (according to a survey conducted in 2011)
2.New York Stock Exchange captures 1TB of trade information during each trading
Session

Variety

* Refers to the category to which the data belongs
*No restriction over the input data formats
*Data mostly unstructured or semi-structured
Example of *variety* –
1. Pure text, images, audio, video, web, GPS data, sensor data, SMS, documents, PDFs, flash
etc.

Variability

*Refers to data whose meaning is constantly changing.
*Meaning of the data depends on the context.
*Data appear as an indecipherable mass without structure
Example:
Language processing, Hashtags, Geo-spatial data, Multimedia, Sensor events

Veracity

*Veracity refers to the biases, noise and abnormality in data.

*It is important in programs that involve automated decision-making, or feeding the data into an unsupervised machine learning algorithm.
*Veracity isn’t just about data quality, it’s about data understandability

Visualization

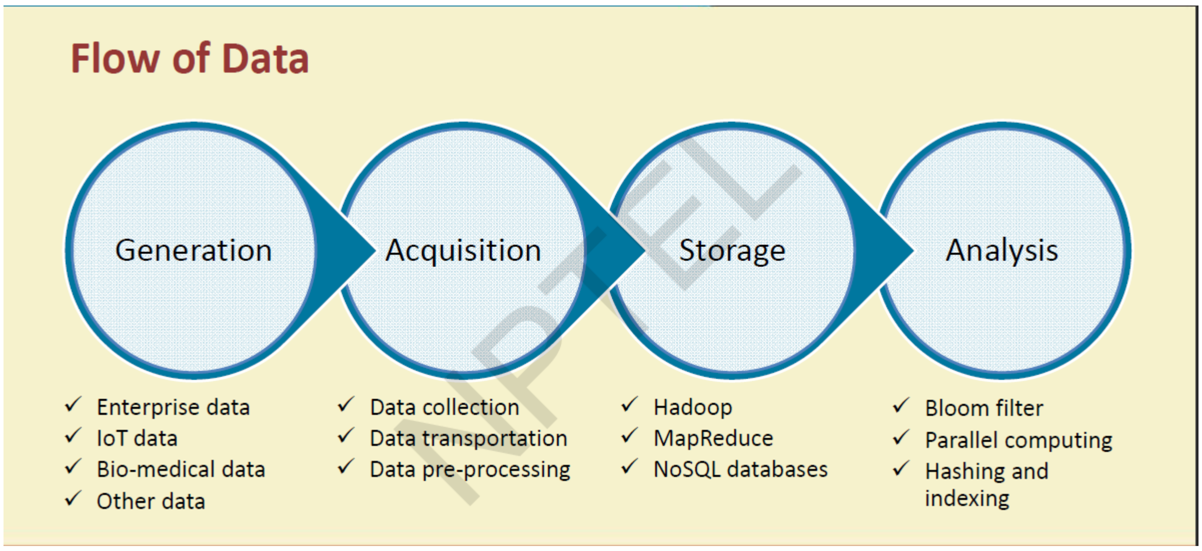
*Presentation of data in a pictorial or graphical format
Enables decision makers to see analytics presented visually
Identify new patterns

Value

*It means extracting useful business information from scattered data.
*Includes a large volume and variety of data
*Easy to access and delivers quality analytics that enables informed decisions

Platform-as-a-Service (PaaS)
Software-as-a-Service (SaaS)

Flow of data



The above fig shows how data flows from generation to analysis .

Data Sources /Data Generation

- Enterprise data

Online trading and analysis data.
/Production and inventory data.
Sales and other financial data.

- IoT data

Data from industry, agriculture,traffic, transportation
Medical-care data,
Data from public departments, and
families.

- Bio-medical data

Masses of data generated by gene
sequencing.
Data from medical clinics and medical
R&Ds.

- Other fields

Fields such as – computational biology,
astronomy, nuclear research etc

Data Acquisition

Data collection

Log files or record files that are automatically generated by data sources to record activities for further analysis, that has been collected from devices like Sensory data such as sound wave, voice, vibration, automobile, chemical, current, weather, pressure, temperature etc , and even Complex and variety of data collection through mobile devices. E.g. – geographical location, 2D barcodes, pictures, videos etc.

Data transmission

1.After collecting data, it will be transferred to storage system for further processing and analysis of the data.
2. Data transmission can be categorized as – Inter-DCN transmission and Intra-DCN transmission

Data pre-processing

1. Collected datasets suffer from noise, redundancy, inconsistency etc., thus, preprocessing of data is necessary.
2.Pre-processing of relational data mainly follows – integration, cleaning, and redundancy mitigation
3.Integration is combining data from various sources and provides users with a uniform view of data.
4. Cleaning is identifying inaccurate, incomplete, or unreasonable data, and then modifying or deleting such data.
5.Redundancy mitigation is eliminating data repetition through detection, filtering and compression of data to avoid unnecessary transmission.

Data Storage : Data can be stored in Filesystems or Databases

File system

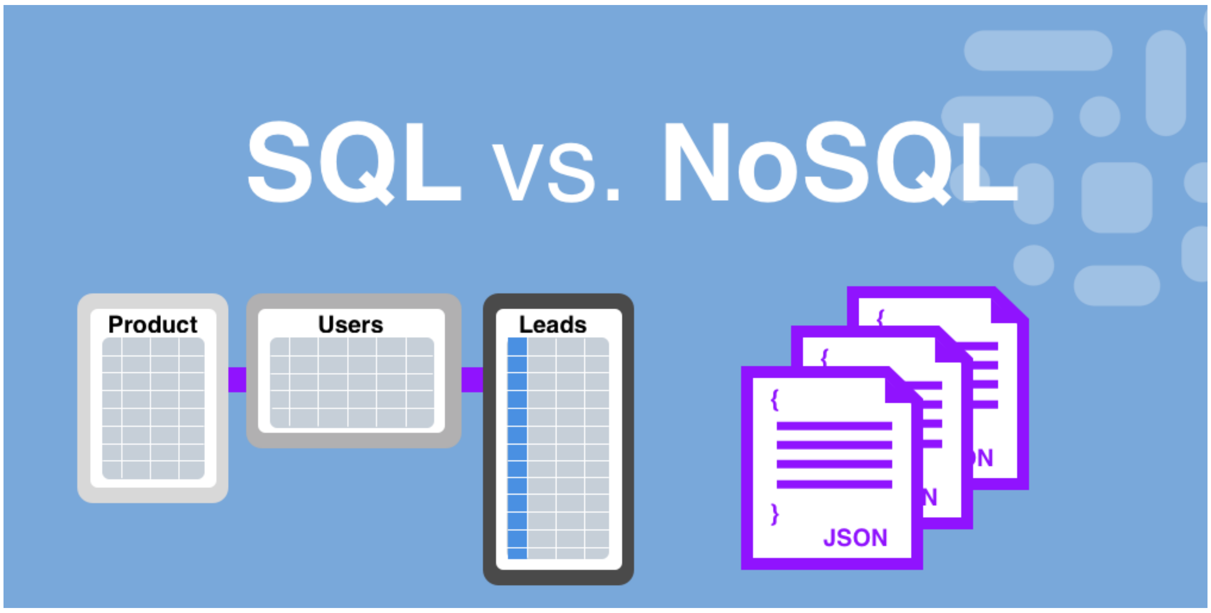
1.Distributed file systems that store massive data and ensure – consistency, availability,and fault tolerance of data.
2. GFS is a notable example of distributed file system that supports large-scale file system, though it's performance is limited in case of small files
3.Hadoop Distributed File System (HDFS) and Kosmosfs are other notable file systems, derived from the open source codes of GFS.

Databases

1. Emergence of non-traditional relational databases (NoSQL) in order to deal with the characteristics that big data possess i.e unstructured data .
2. Nosql uses 3 different type of databases
1.keyvalue database
2.coloumn oriented database

3.Document oriented database

Nosql doesn't uses the table module instead data is stored in single document file.



Data Handling Using Hadoop

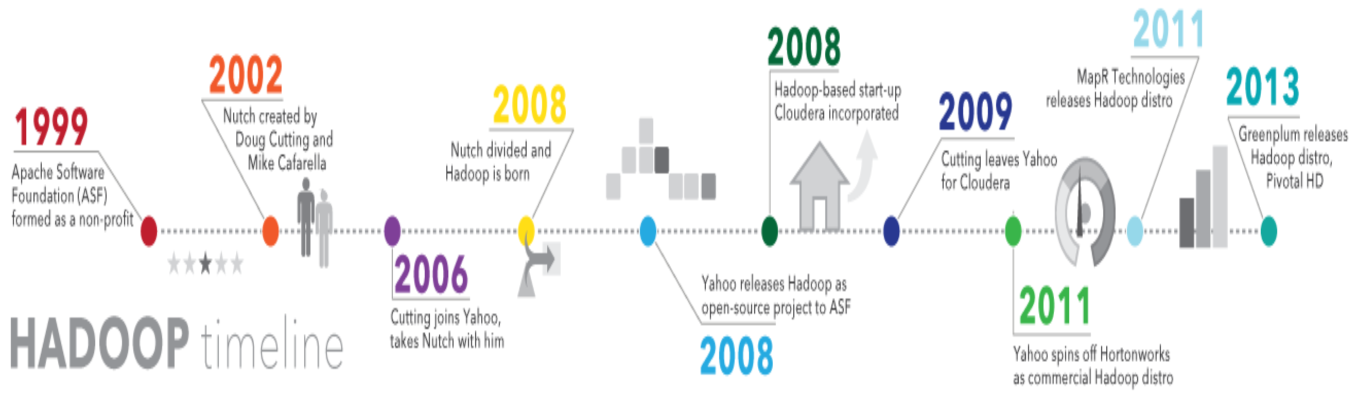
Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

Why is Hadoop important?

- **Ability to store and process huge amounts of any kind of data, quickly.** With data volumes and varieties constantly increasing, especially from social media and the Internet of Things (IoT), that's a key consideration.
- **Computing power.** Hadoop's distributed computing model processes big data fast. The more computing nodes you use, the more processing power you have.
- **Fault tolerance.** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically.

- **Flexibility.** Unlike traditional relational databases, you don't have to preprocess data before storing it. You can store as much data as you want and decide how to use it later. That includes unstructured data like text, images and videos.
- **Low cost.** The open-source framework is free and uses commodity hardware to store large quantities of data.
- **Scalability.** You can easily grow your system to handle more data simply by adding nodes. Little administration is required.

Hadoop History


HADOOP timeline
1999
Apache Software Foundation (ASF) formed as a non-profit
2002
Nutch created by Doug Cutting and Mike Cafarella
2006
Cutting joins Yahoo, takes Nutch with him
2008
Nutch divided and Hadoop is born
2008
Yahoo releases Hadoop as open-source project to ASF
2008
Hadoop-based start-up Cloudera incorporated
2009
Cutting leaves Yahoo for Cloudera
2011
Yahoo spins off Hortonworks as commercial Hadoop distro
2011
MapR Technologies releases Hadoop distro
2013
Greenplum releases Hadoop distro, Pivotal HD

How Is Hadoop Being Used?

IoT and Hadoop

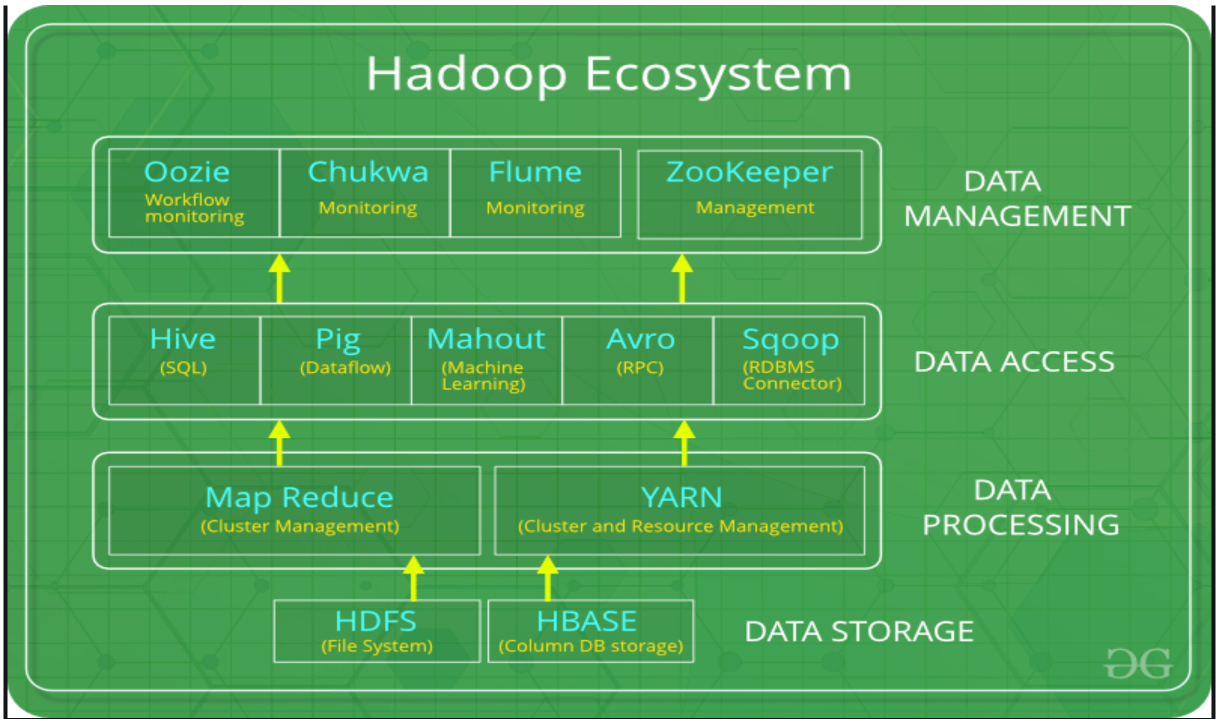
Things in the IoT need to know what to communicate and when to act. At the core of the IoT is a streaming, always on torrent of data. Hadoop is often used as the data store for millions or billions of transactions. Massive storage and processing capabilities also allow you to use Hadoop as a sandbox for discovery and definition of patterns to be monitored for prescriptive instruction. You can then continuously improve these instructions, because Hadoop is constantly being updated with new data that doesn't match previously defined patterns.

Because Hadoop was designed to deal with volumes of data in a variety of shapes and forms, it can run analytical algorithms. Big data analytics on Hadoop can help your organization operate more efficiently, uncover new opportunities and derive next-level competitive advantage.

The sandbox approach provides an opportunity to innovate with minimal investment.

Hadoop Ecosystem

Hadoop Framework was developed to store and process data with a simple programming model in a distributed data processing environment. The data present on different high-speed and low-expense machines can be stored and analyzed. Enterprises have widely adopted Hadoop as Big Data Technologies for their data warehouse needs in the past year. The trend seems to continue and grow in the coming year as well. Companies that have not explored Hadoop so far will most likely see its advantages and applications.



Currently, four core modules are included in the basic framework

Hadoop Common – the libraries and utilities used by other Hadoop modules.

Hadoop Distributed File System (HDFS) – the Java-based scalable system that stores data across multiple machines without prior organization.

YARN – (Yet Another Resource Negotiator) provides resource management for the processes running on Hadoop.

MapReduce – a parallel processing software framework. It is comprised of two steps. Map step is a master node that takes inputs and partitions them into smaller subproblems and then distributes them to worker nodes. After the map step has taken place, the master node takes the answers to all of the subproblems and combines them to produce output.

Other software components that can run on top of or alongside Hadoop

Ambari	**A web interface for managing, configuring and testing Hadoop services and components.**
Cassandra	**A distributed database system.**
Flume	**Software that collects, aggregates and moves large amounts of streaming data into HDFS.**
HBase	**A nonrelational, distributed database that runs on top of Hadoop. HBase tables can serve as input and output for MapReduce jobs.**
HCatalog	**A table and storage management layer that helps users share and access data.**
Hive	**A data warehousing and SQL-like query language that presents data in the form of tables. Hive programming is similar to database programming.**
Oozie	**A Hadoop job scheduler.**
Pig	**A platform for manipulating data stored in HDFS that includes a compiler for MapReduce programs and a high-level language called Pig Latin. It provides a way to perform data extractions, transformations and loading, and basic analysis without having to write MapReduce programs.**
Solr	**A scalable search tool that includes indexing, reliability, central configuration, failover and recovery.**
Spark	**An open-source cluster computing framework with in-memory analytics.**
Sqoop	**A connection and transfer mechanism that moves data between Hadoop and relational databases.**

Zookeeper	**An application that coordinates distributed processing.**

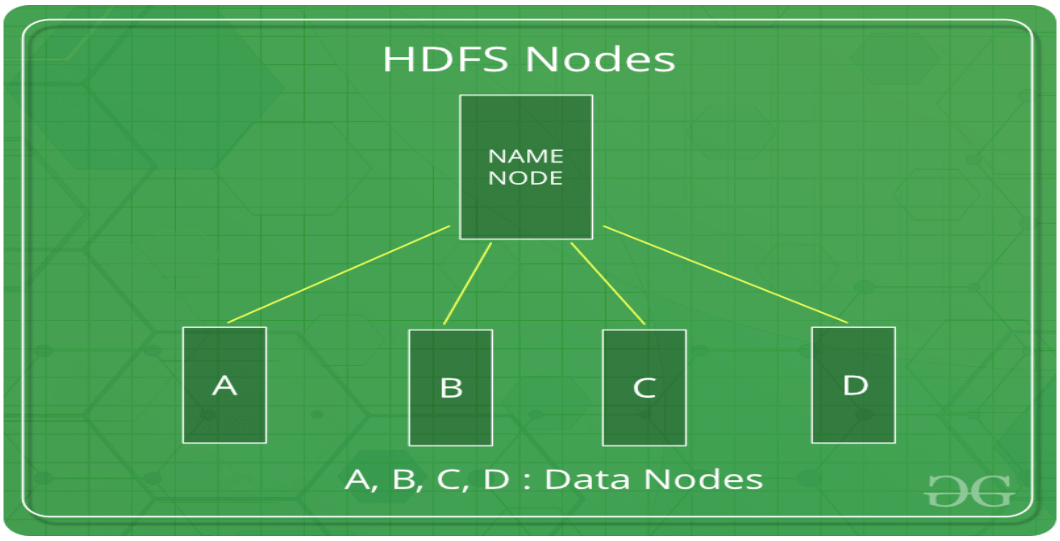
Hadoop Distributed File System (HDFS) Architecture

Apache HDFS or **Hadoop Distributed File System** is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines. Apache Hadoop HDFS Architecture follows a *Master/Slave Architecture*, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes). HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.

All these toolkits or components revolve around one term i.e. *Data*. That's the beauty of Hadoop that it revolves around data and hence making its synthesis easier.

HDFS:

- HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of three core components i.e.
  1. Name node
  2. Data Node
  3. Job tracker and Task tracker
- Name Node is the prime node which contains metadata
- Maintains two in-memory tables, to map the datanodes to the blocks, and vice versa
- Data nodes that stores the actual data. These data nodes can talk to each other to rebalance and replicate data.
- Data nodes update the name node with the block information periodically.
- Before updating data nodes verify the checksums
- Undoubtedly, making Hadoop cost effective.
- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.



3. Job Tracker and Task Tracker

Job Tracker is our contact for applications and Hadoop, and when we submit code to the Hadoop cluster, it determines the execution plan, including deciding which files to process, assigning different tasks to each node (which is actually assigned to task Tracker, and then forwarding), and monitor all tasks that are running. This process typically runs on the primary node of the cluster.

Job Tracker –

Runs with the Namenode

Receives the user's job

Decides on how many tasks will run (number of mappers)

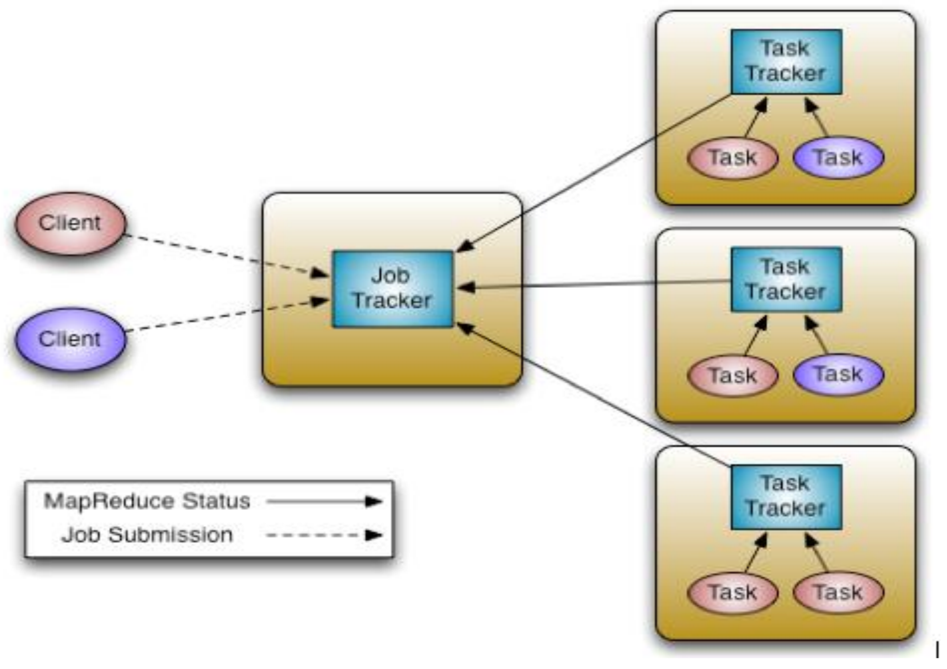
Decides on where to run each mapper

Task Tracker –

Runs on each datanode

Receives the task from Job Tracker

Always in communication with the Job Tracker reporting progress



Hadoop Master/Slave Architecture

Master-slave shared-nothing architecture

Master

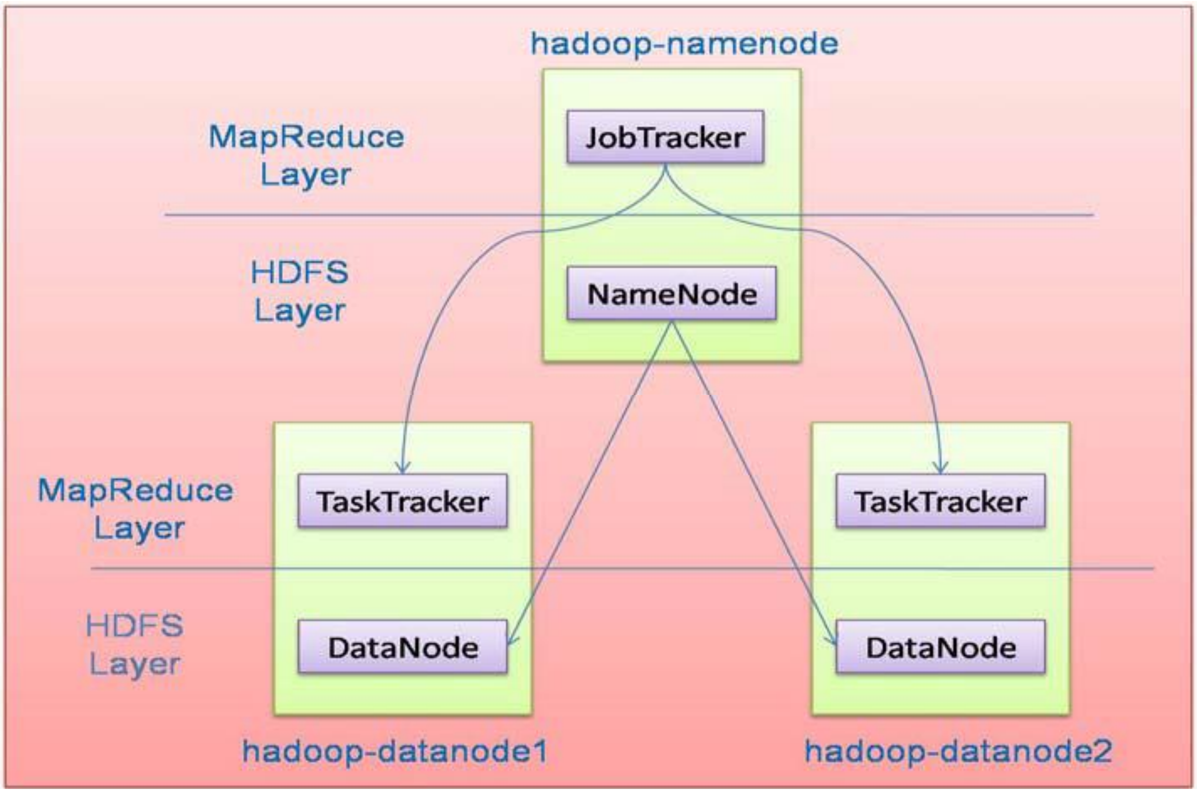
Executes operations like opening, closing, and renaming files and directories.

Determines the mapping of blocks to Datanodes.

Slave

Serves read and write requests from the file system's clients.

Performs block creation, deletion, and replication as instructed by the Namenode



YARN:

- Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- Consists of three major components i.e.
  1. Resource Manager
  2. Nodes Manager
  3. Application Manager
- Resource manager has the privilege of allocating resources for the applications in a system whereas Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager. Application manager works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.

MapReduce:

- By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.
- MapReduce makes the use of two functions i.e. `Map()` and `Reduce()` whose task is:
  1. *Map()* performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a

key-value pair based result which is later on processed by the Reduce() method.

2. *Reduce()*, as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.

Data Analytics

What is Data Analytics

"Data analytics (DA) is the process of examining data sets in order to draw conclusions about the information they contain, increasingly with the aid of specialized systems and software. Data analytics technologies and techniques are widely used in commercial industries to enable organizations to make moreinformed business decisions and by scientists and researchers to verify or disprove scientific models, theories and hypotheses."

Types of Data Analysis

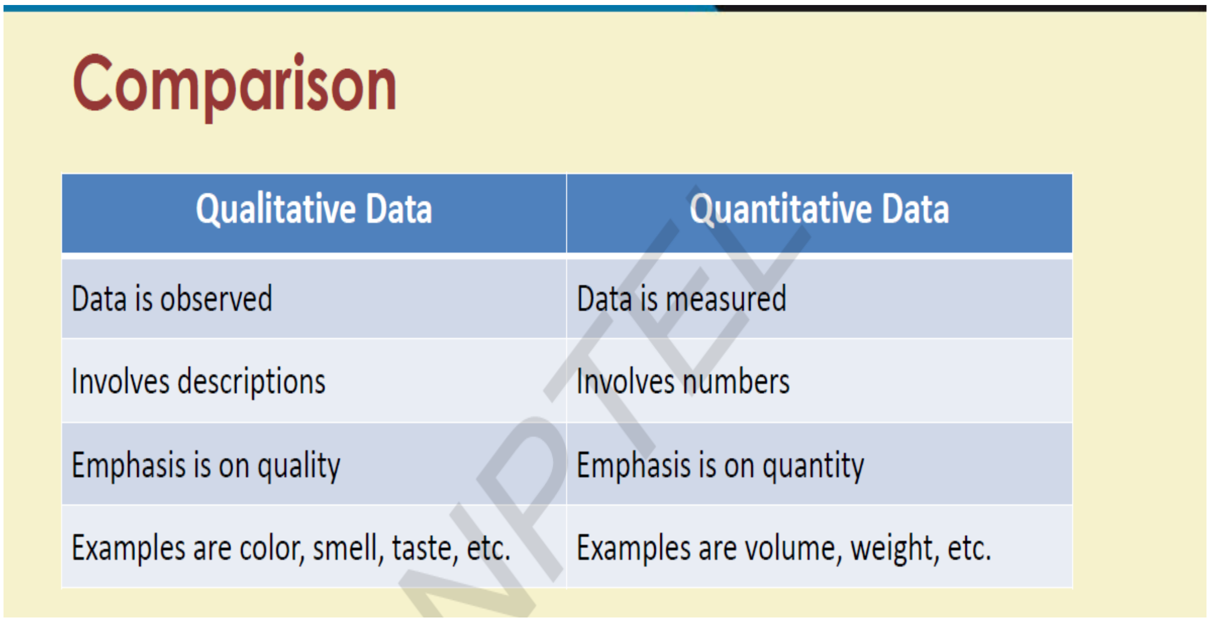
Two types of analysis

Qualitative Analysis

Deals with the analysis of data that is categorical in nature

Quantitative Analysis

Quantitative analysis refers to the process by which numerical data is analyzed



Comparison

Qualitative Data	Quantitative Data
Data is observed	Data is measured
Involves descriptions	Involves numbers
Emphasis is on quality	Emphasis is on quantity
Examples are color, smell, taste, etc.	Examples are volume, weight, etc.

Advantages of Data Analytics

1. Allows for the identification of important (and often mission-critical) trends
2. Helps businesses identify performance problems that require some sort of action
3. Can be viewed in a visual manner, which leads to faster and better decisions
4. Better awareness regarding the habits of potential customers
5. It can provide a company with an edge over their competitors

Brief Description on Qualitative and Quantitative Analysis

. Qualitative Analysis

Data is not described through numerical values
Described by some sort of descriptive context such as text
Data can be gathered by many methods such as interviews, videos and audio recordings, field notes
Data needs to be interpreted
The grouping of data into identifiable themes
Qualitative analysis can be summarized by three basic principles (Seidel, 1998):

- Notice things
- Collect things
- Think about things

Quantitative Analysis

Quantitative analysis refers to the process by which numerical data is analyzed
Involves descriptive statistics such as mean, media, standard deviation
The following are often involved with quantitative analysis:
1.Statistical models
2.Analysis of variables
3.Data dispersion
4.Analysis of relationships between variables

5.Contingence and correlation
6.Regression analysis
7.Statistical significance
8.Precision
9.Error limits

1.Statistical models

Statistical modeling is the process of applying **statistical analysis** to a dataset. A **statistical model** is a mathematical representation (or mathematical **model**) of observed **data**.

1. The **statistical model** is defined as the mathematical equation that are formulated in the form of relationships between variables.
2. A statistical model illustrates how a set of random variables is related to another set of random variables.
3. A statistical model is represented as the ordered pair (X , P)
   X denotes the set of all possible observations
   P refers to the set of probability distributions on X

Statistical models are broadly categorized as

1. Complete models
2. Incomplete models

Complete model does have the number of variables equal to the number of equations
An incomplete model does not have the same number of variables as the number of equations

In order to build a statistical model
 Data Gathering
 Descriptive Methods
 Thinking about Predictors
 Building of model
 Interpreting the Results

2.Analysis of variance

 Analysis of Variance (ANOVA) is a parametric statistical technique used to compare datasets.
 ANOVA is best applied where more than 2 populations or samples are meant to be compared.
 To perform an ANOVA, we must have a continuous response variable and at least one categorical factor (e.g. age, gender) with two or more levels (e.g. Locations 1, 2)
 ANOVAs require data from approximately normally distributed populations

Properties to perform ANOVA –
 Independence of case
 The sample should be selected randomly
 There should not be any pattern in the selection of the sample
 Normality
 Distribution of each group should be normal
 Homogeneity
 Variance between the groups should be the same (e.g. should not compare data from cities with those from slums)

Analysis of variance (ANOVA) has three types:

- One way analysis
- One fixed factor (levels set by investigator). Factors: age, gender, etc.
- Two way analysis
- Factor variables are more than two
- K-way analysis
- Factor variables are k

Analysis of variance

- **Total Sum of square**

- In statistical data analysis, the total sum of squares (TSS or SST) is a quantity that
appears as part of a standard way of presenting results of such analyses. It is defined

as being the sum, over all observations, of the squared differences of each observation from the overall mean.

- **F –ratio**
- Helps to understand the ratio of variance between two data sets
- The F ratio is approximately 1.0 when the null hypothesis is true and is greater than
1.0 when the null hypothesis is false.
- **Degree of freedom**
- Factors which have no effect on the variance
- The number of degrees of freedom is the number of values in the final calculation of a
statistic that are free to vary.

3.Data dispersion

A measure of statistical dispersion is a nonnegative real number that is zero if all the data are the same and increases as the data becomes more diverse.

Examples of dispersion measures:

- Range
- Average absolute deviation
- Variance and Standard deviation

Range

The range is calculated by simply taking the difference between the maximum and minimum values in the data set.

Average absolute deviation

The average absolute deviation (or mean absolute deviation) of a data set is the average of the absolute deviations from the mean.

Variance

Variance is the expectation of the squared deviation of a random variable from its mean

Standard deviation

Standard deviation (SD) is a measure that is used to quantify the amount of variation or dispersion of a set of data values.

4.Contingence and correlation

-→ In statistics, a contingency table (also known as a cross tabulation or crosstab) is a type of table in a matrix format that displays the (multivariate) frequency distribution of the variables.
-→Provides a basic picture of the interrelation between two variables
-→A crucial problem of multivariate statistics is finding (direct-)dependence structure underlying the variables contained in high-dimensional contingency tables

→Correlation is a technique for investigating the relationship between two quantitative, continuous variables
→Pearson's correlation coefficient (r) is a measure of the strength of the association between the two variables.
→Correlations are useful because they can indicate a predictive relationship that can be exploited in practice

5.Regression analysis

1.In statistical modeling, regression analysis is a statistical process for estimating the relationships among variables
2. Focuses on the relationship between a dependent variable and one or more independent variables
3. Regression analysis estimates the conditional expectation of the dependent variable given the independent variables
4.The estimation target is a function of the independent variables called the regression function
5. Characterize the variation of the dependent variable around the regression function which can be described by a probability distribution
6.Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning
7. Regression analysis is also used to understand which among the independent variables are related to the dependent variable

Statistical significance

1. Statistical significance is the likelihood that the difference in conversion rates between a given variation and the baseline is not due to random chance
2.Statistical significance level reflects the risk tolerance and confidence level
3. There are two key variables that go into determining statistical significance:
    Sample size
    Effect size
4.Sample size refers to the sample size of the experiment
5. The larger your sample size, the more confident you can be in the result of the experiment (assuming that it is a randomized sample)
6.The effect size is just the standardized mean difference between the two groups
7. If a particular experiment replicated, the different effect size estimates from each study can easily be combined to give an overall best estimate of the effect size

Precision and Error limits

1.Precision refers to how close estimates from different samples are to each other
2. The standard error is a measure of precision
3.When the standard error is small, estimates from different samples will be close in value and vice versa
4.Precision is inversely related to standard error
5.The limits of error are the maximum overestimate and the maximum
underestimate from the combination of the sampling and the non-sampling errors
6. The margin of error is defined as –
    Limit of error = Critical value x Standard deviation of the statistic
    Critical value: Determines the tolerance level of error