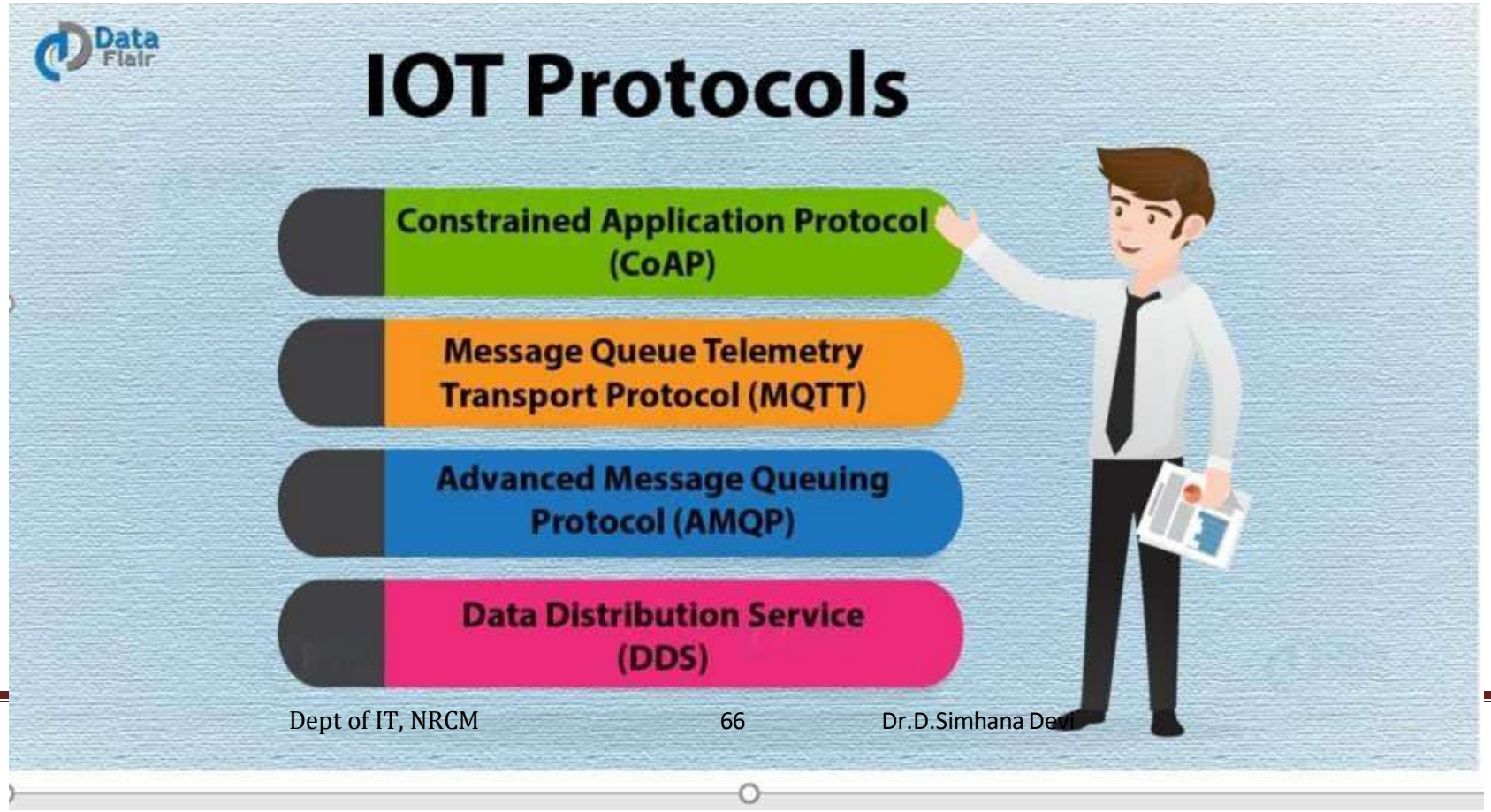
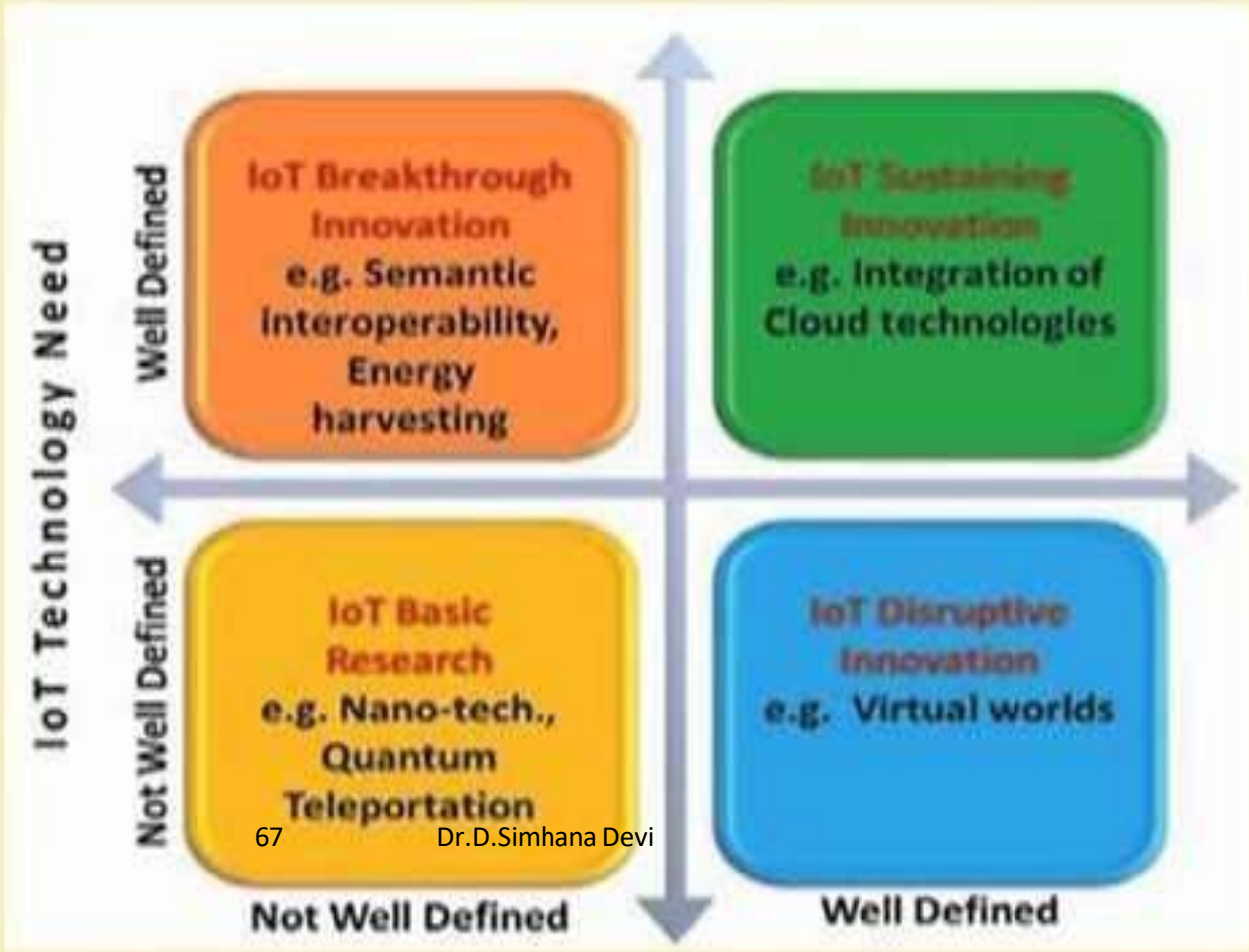
UNIT-1 PART 2

Basics of Networking



Convergence of Domains

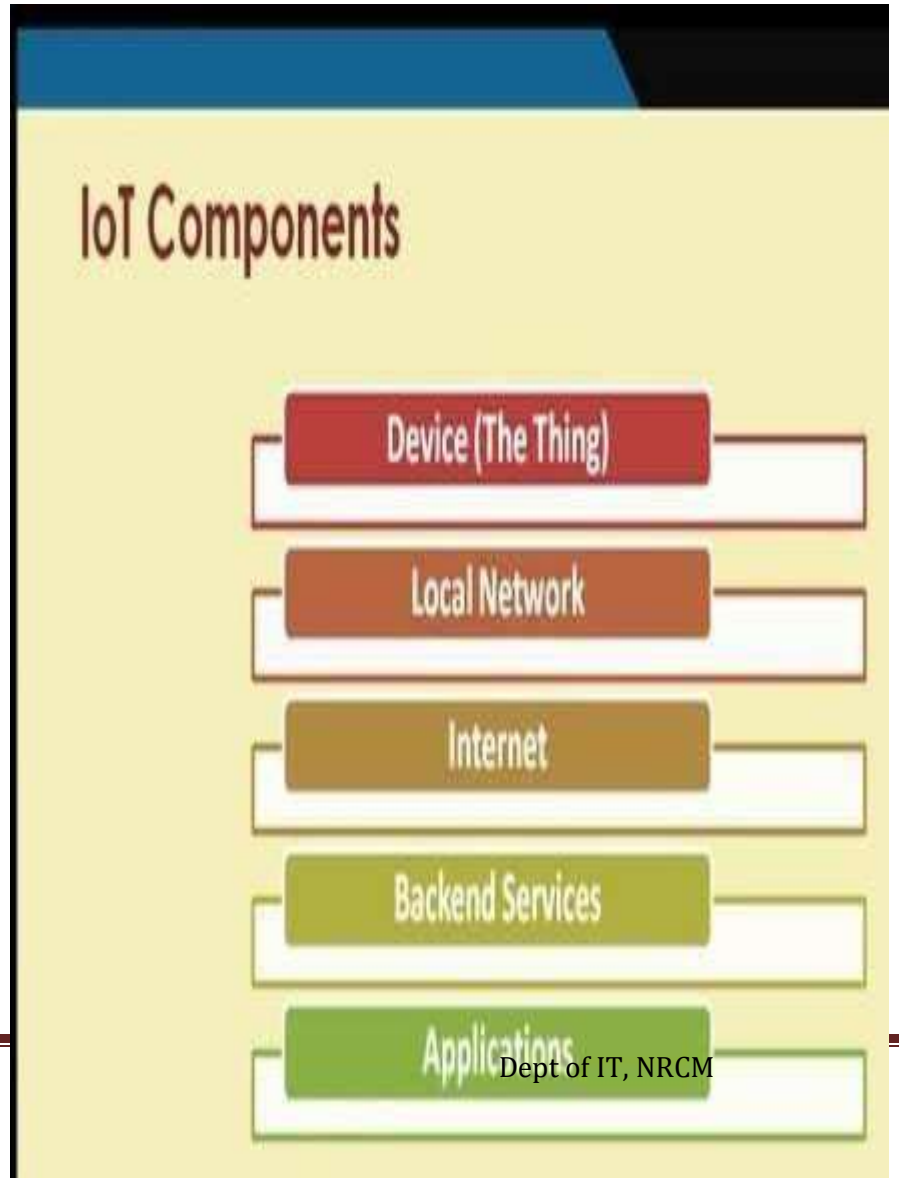


1)IOT Basic research, there has been a lot of research on the nanotechnology, the use of nanotechnology, the use of quantum teleportation. **Quantum teleportation** basically means that how the different information at the atomic level is sent from one point to another.So, it is transported from one point to another at the atomic level and nanotechnology, it involves things like nano IoT, nano nodes, nano networking, nodes, nano sensor nodes and nano networks. So, like this at the nanoscale and for quantum communication, there has been lot of advertisements that has been done for involving basic innovations, basic research innovations.

2)IOT Break through Innovation: Eg:Semantic Interoperability,Energy harvesting.

Semantic Interoperability : There has been lot of research on semantic in for interoperability. For example, let us see that a temperature sensor, it might be given the data as temp, another temperature sensor as temperature, another temperature sensor the third one. So, there has to be interoperability between all these different collisions, but they are all different to the same temperature, right. So, this is

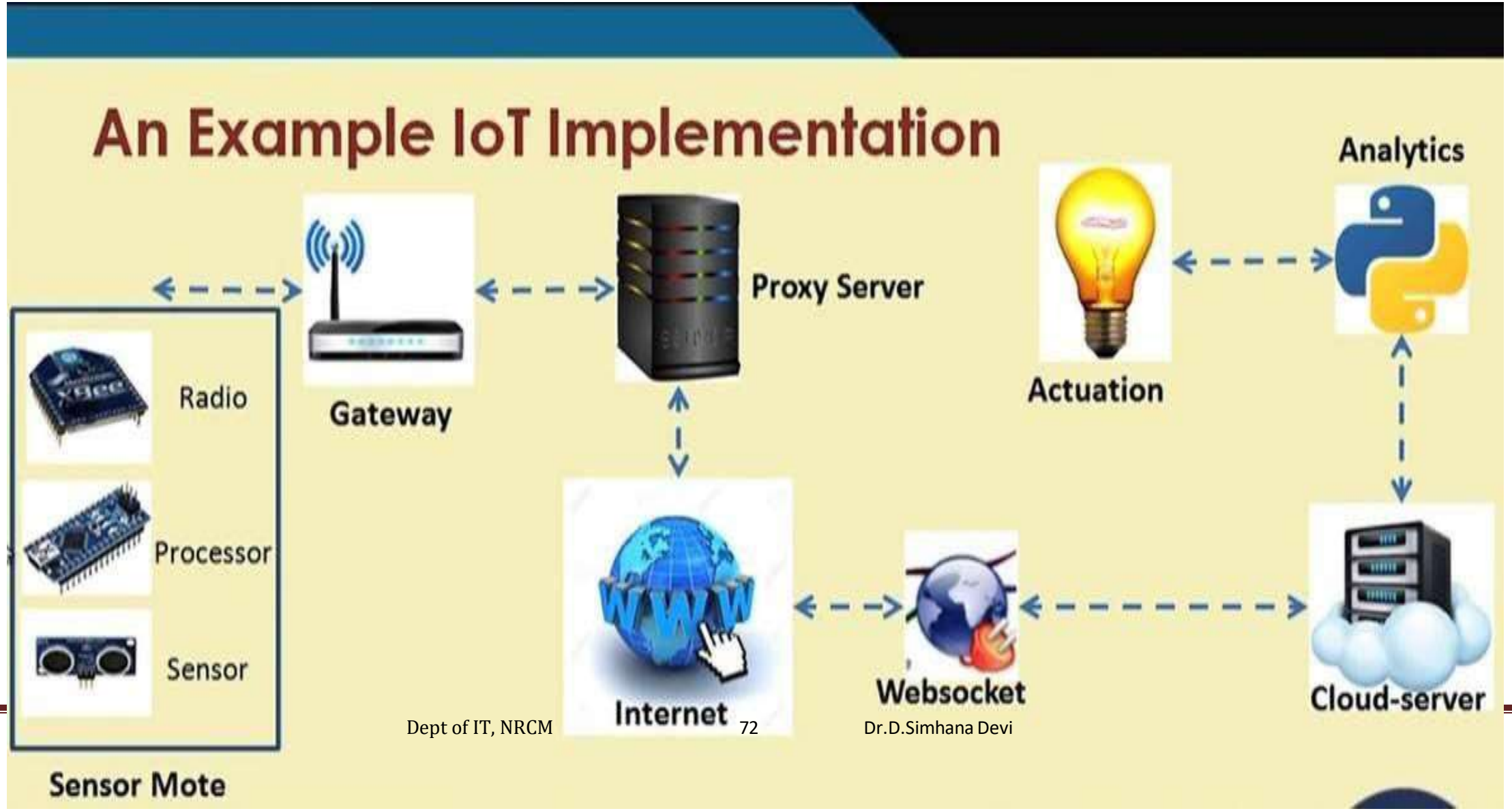
basically taken care of by things like semantic interoperability


IoT Components
Device (The Thing)
Local Network
Internet
Backend Services
Applications

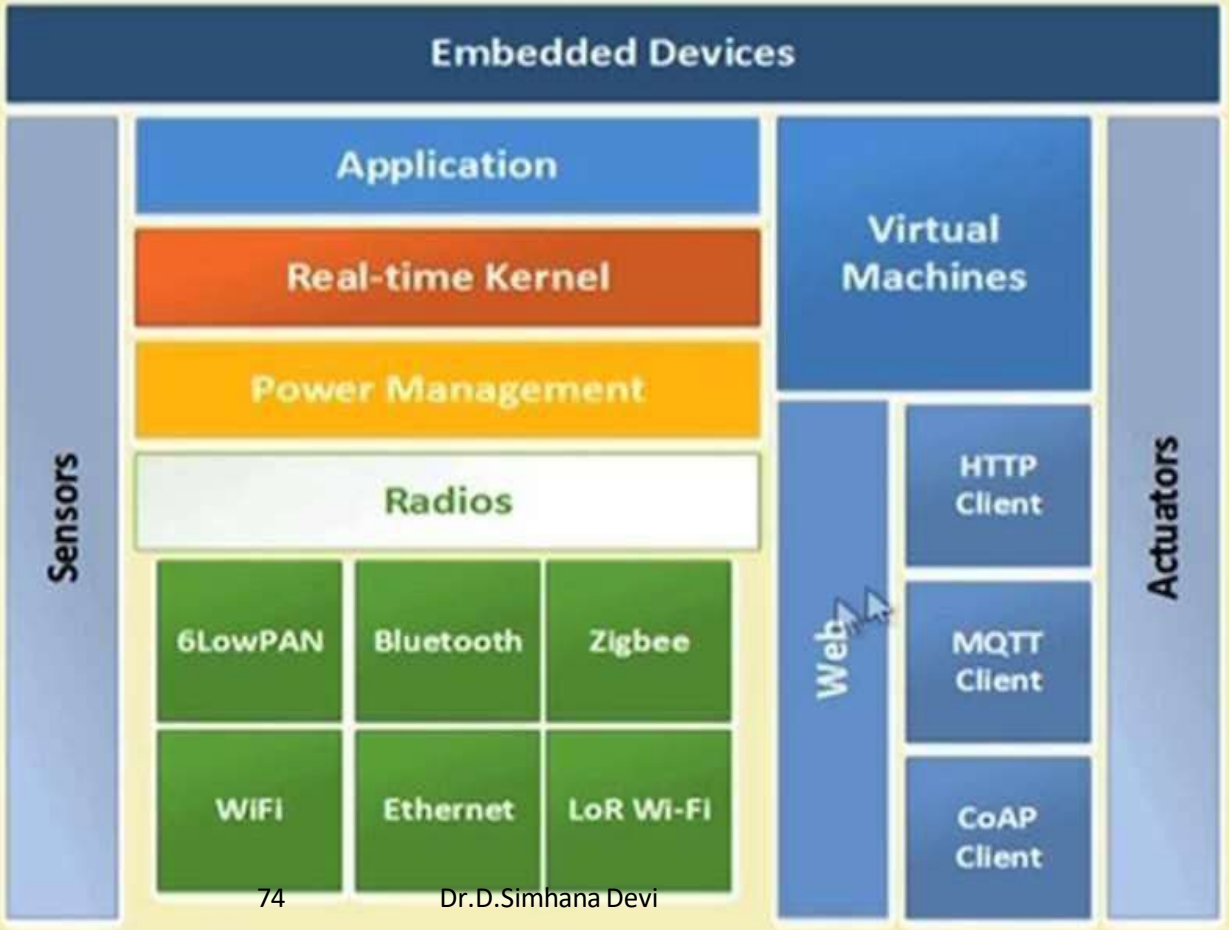

Things
Local Network
Internet
Backend Services
Communication
Processing
Analytics
Server/ Storage

Functional Components of IoT

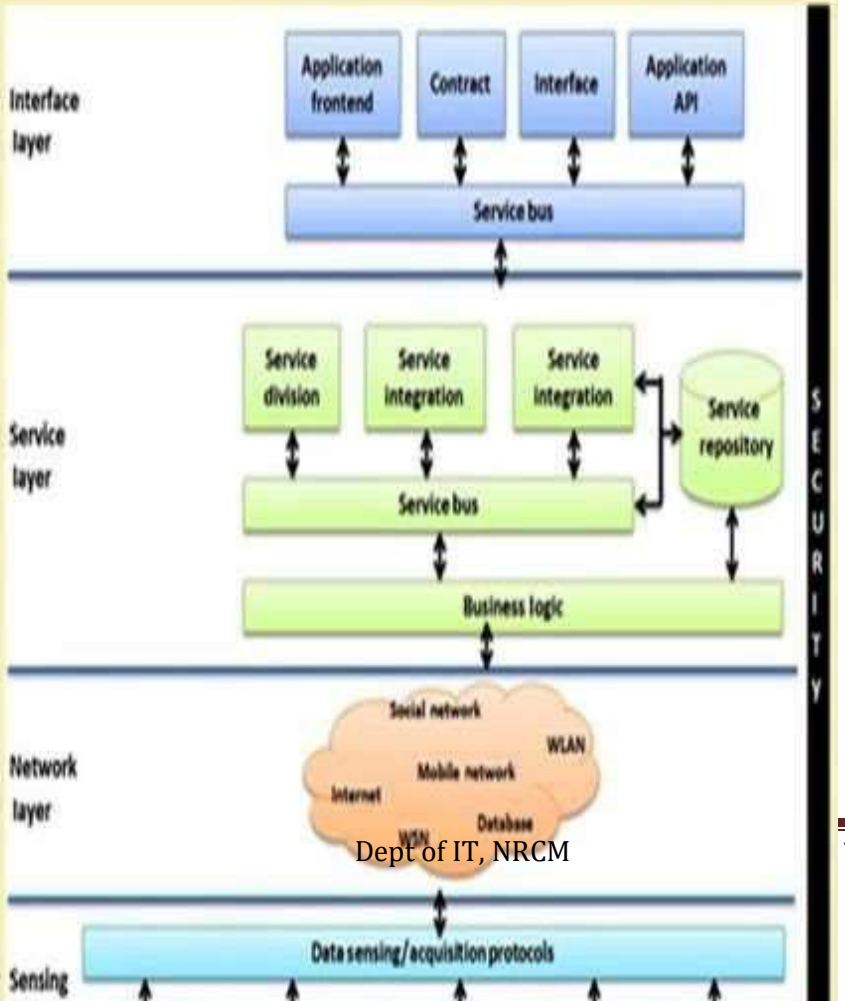
- ✓ Component for interaction and communication with other IoT devices
- ✓ Component for processing and analysis of operations
- ✓ Component for Internet interaction
- ✓ Components for handling Web services of applications
- ✓ Component to integrate application services
- ✓ User interface to access IoT


An Example IoT Implementation
Analytics
Proxy Server
Actuation
Radio
Gateway
Processor
Sensor
Internet
Websocket
Cloud-server
Sensor Mote

IoT Interdependencies



IoT Service Oriented Architecture



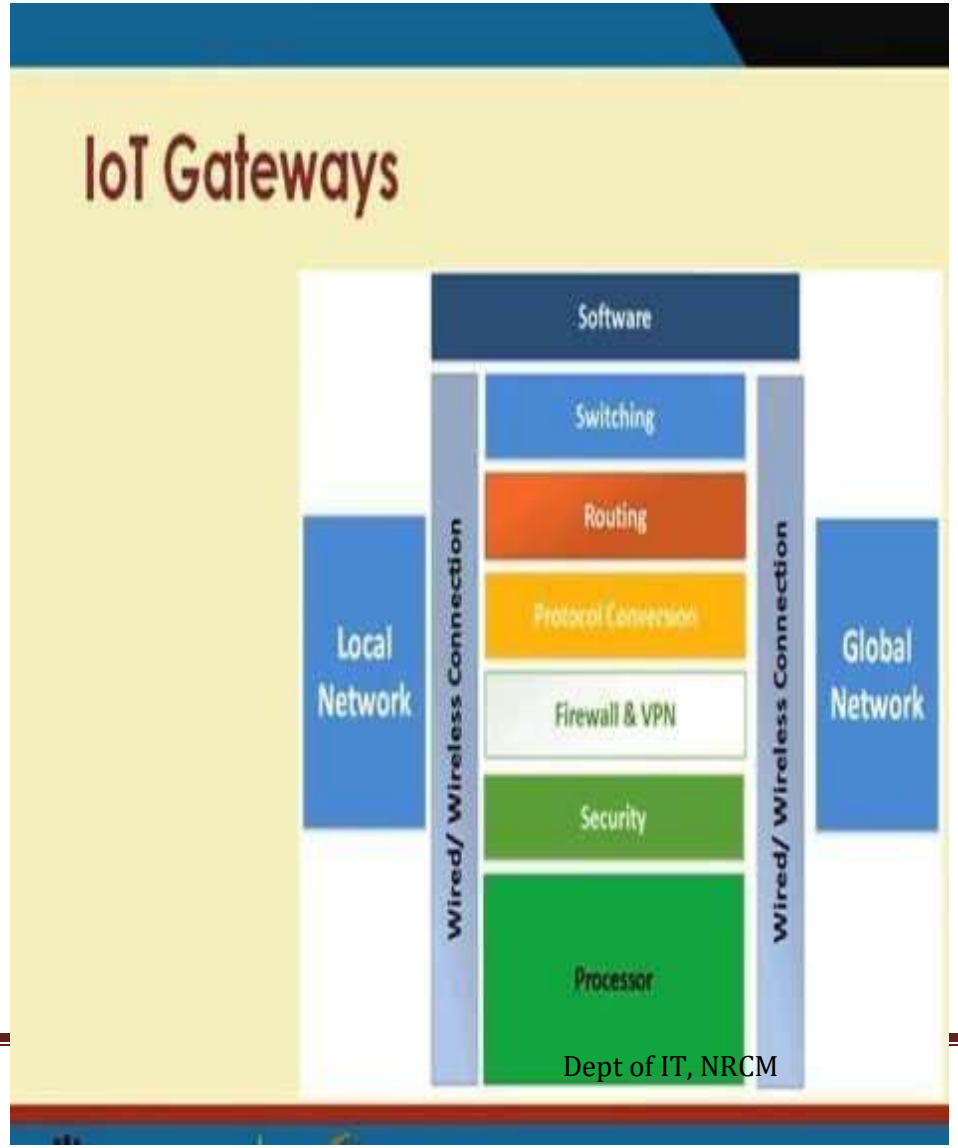
IoT Categories

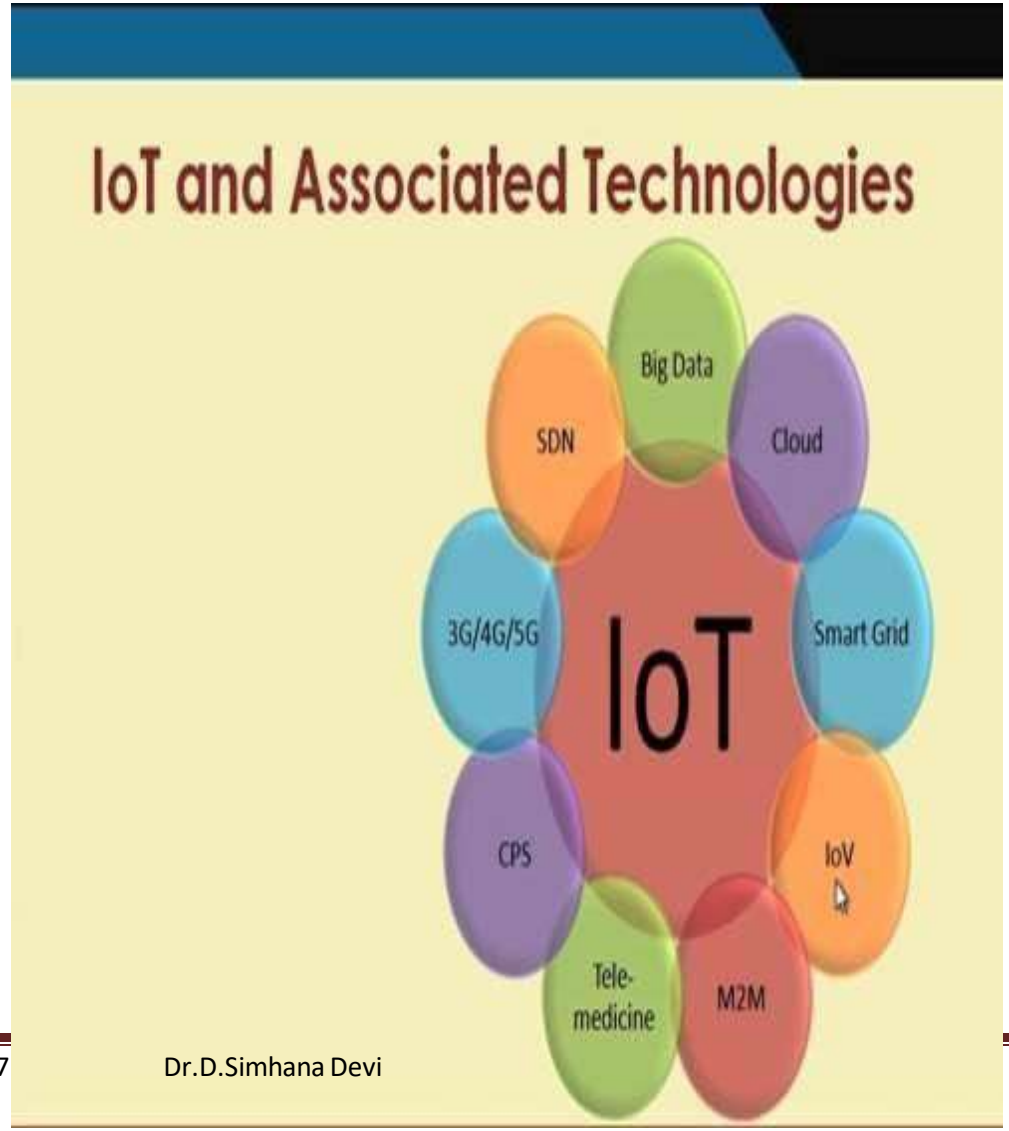
✓ Industrial IoT

- IoT device connects to an IP network and the global Internet.
- Communication between the nodes done using regular as well as industry specific technologies.

✓ Consumer IoT

- IoT device communicates within the locally networked devices.
- Local communication is done mainly via Bluetooth, Zigbee or WiFi.


IoT Gateways
Software
Switching
Routing
Protocol Conversion
Firewall & VPN
Security
Processor
Local Network
Wired/ Wireless Connection
Wired/ Wireless Connection
Global Network


IoT and Associated Technologies
IoT
Big Data
Cloud
SDN
3G/4G/5G
Smart Grid
CPS
IoV
Tele-medicine
M2M

Technical Deviations from Regular Web

IoT Challenges

- ✓ Security
- ✓ Scalability
- ✓ Energy efficiency
- ✓ Bandwidth management
- ✓ Modeling and Analysis
- ✓ Interfacing
- ✓ Interoperability
- ✓ Data storage
- ✓ Data Analytics
- ✓ Complexity management (e.g., SDN)

Considerations

- ✓ Communication between the IoT device(s) and the outside world dictates the network architecture.
- ✓ Choice of communication technology dictates the IoT device hardware requirements and costs.
- ✓ Due to the presence of numerous applications of IoT enabled devices, a single networking paradigm not sufficient to address all the needs of the consumer or the IoT device.

Complexity of Networks

- ✓ Growth of networks
- ✓ Interference among devices
- ✓ Network management
- ✓ Heterogeneity in networks
- ✓ Protocol standardization within networks

Wireless Networks

- Traffic and load management
- Variations in wireless networks – Wireless Body Area Networks and other Personal Area Networks
- Interoperability
- Network management
- Overlay networks

Scalability

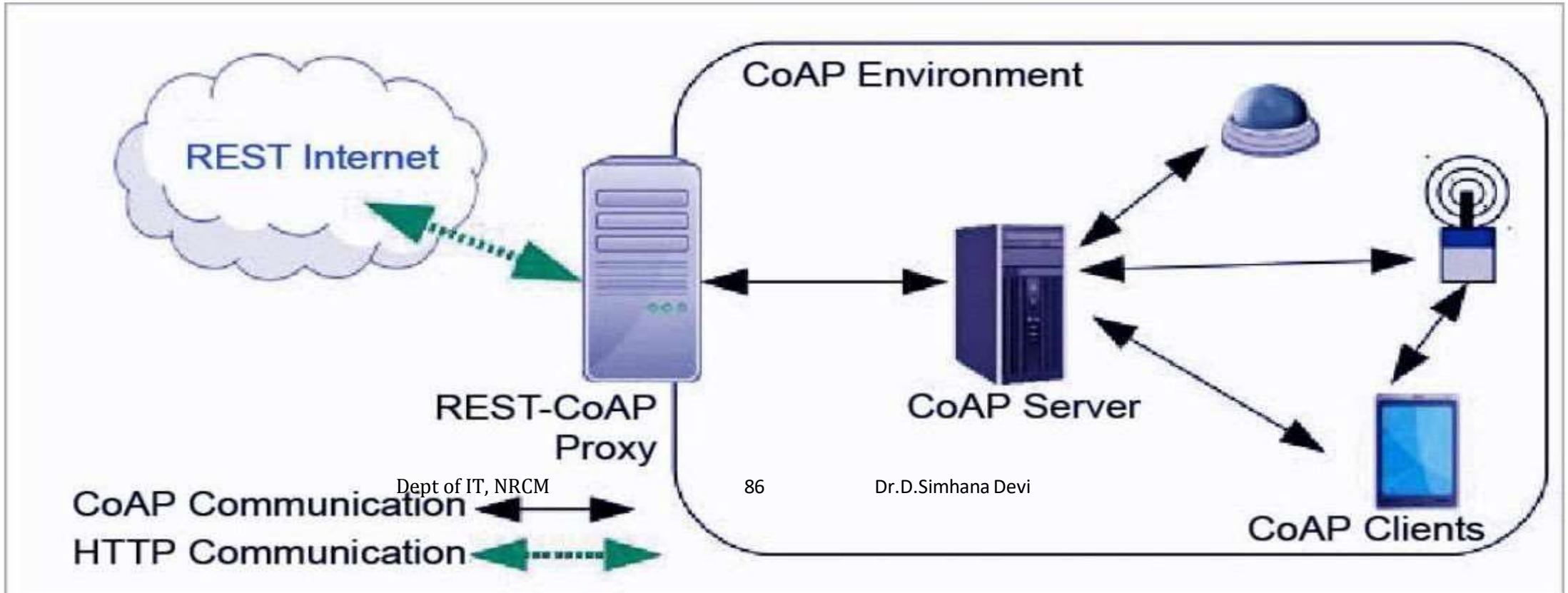
- Flexibility within Internet
- IoT integration
- Large scale deployment
- Real-time connectivity of billions of devices

Functionality-based IoT Protocol Organization

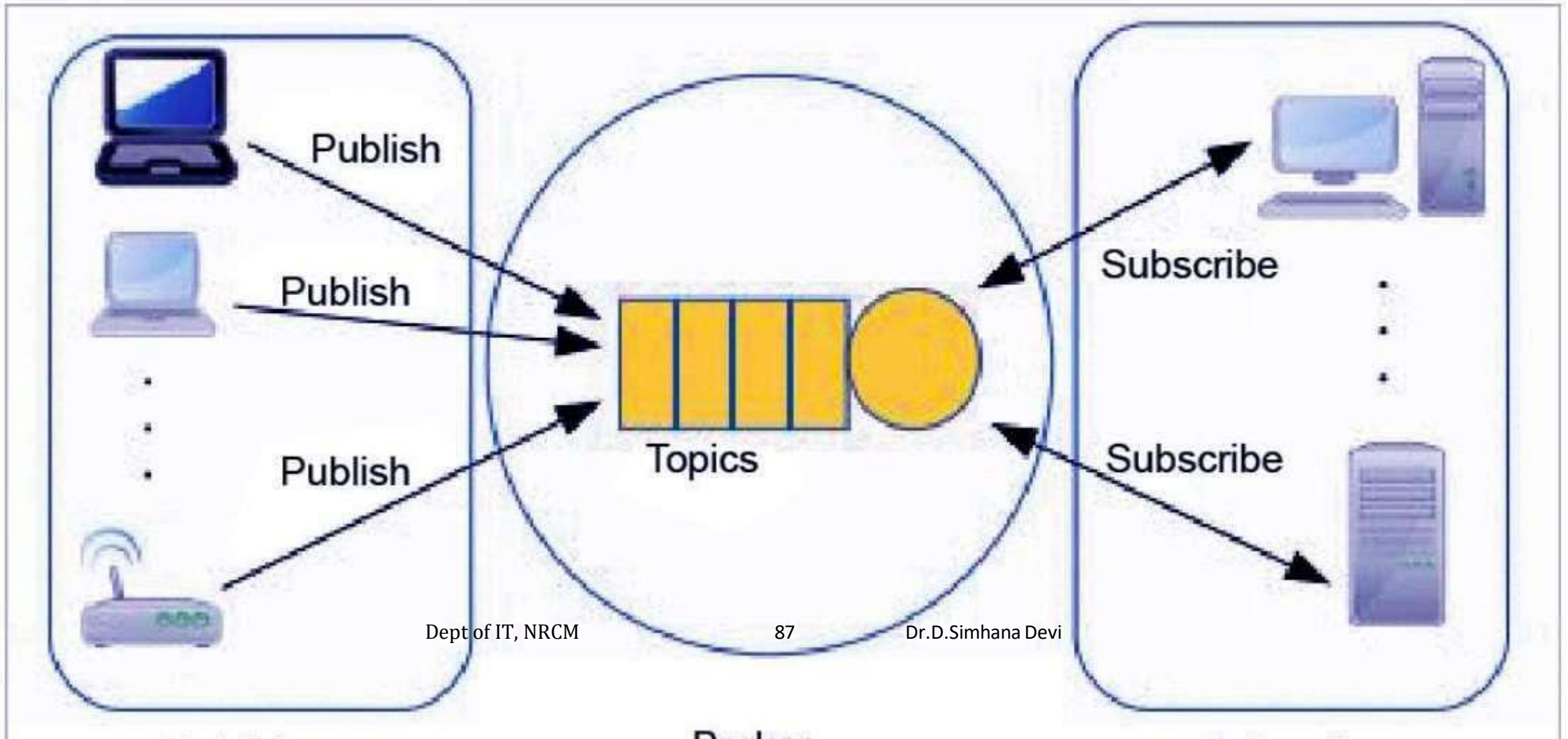
- ✓ **Connectivity** (6LowPAN, RPL)
- ✓ **Identification** (EPC, uCode, IPv6, URIs)
- ✓ **Communication / Transport** (WiFi, Bluetooth, LPWAN)
- ✓ **Discovery** (Physical Web, mDNS, DNS-SD)
- ✓ **Data Protocols** (MQTT, CoAP, AMQP, Websocket, Node)
- ✓ **Device Management** (TR-069, OMA-DM)
- ✓ **Semantic** (JSON-LD, Web Thing Model)
- ✓ **Multi-layer Frameworks** (Alljoyn, IoTivity, Weave, Homekit)

1. Constrained Application Protocol (CoAP)

CoAP is an internet utility protocol for constrained gadgets. It is designed to enable simple, constrained devices to join IoT through constrained networks having low bandwidth availability. This protocol is primarily used for machine-to-machine (M2M) communication and is particularly designed for IoT systems that are based on HTTP protocols.



CoAP makes use of the UDP protocol for lightweight implementation. It also uses restful architecture, which is just like the HTTP protocol. It makes use of dtls for the cozy switch of statistics within the slipping layer.



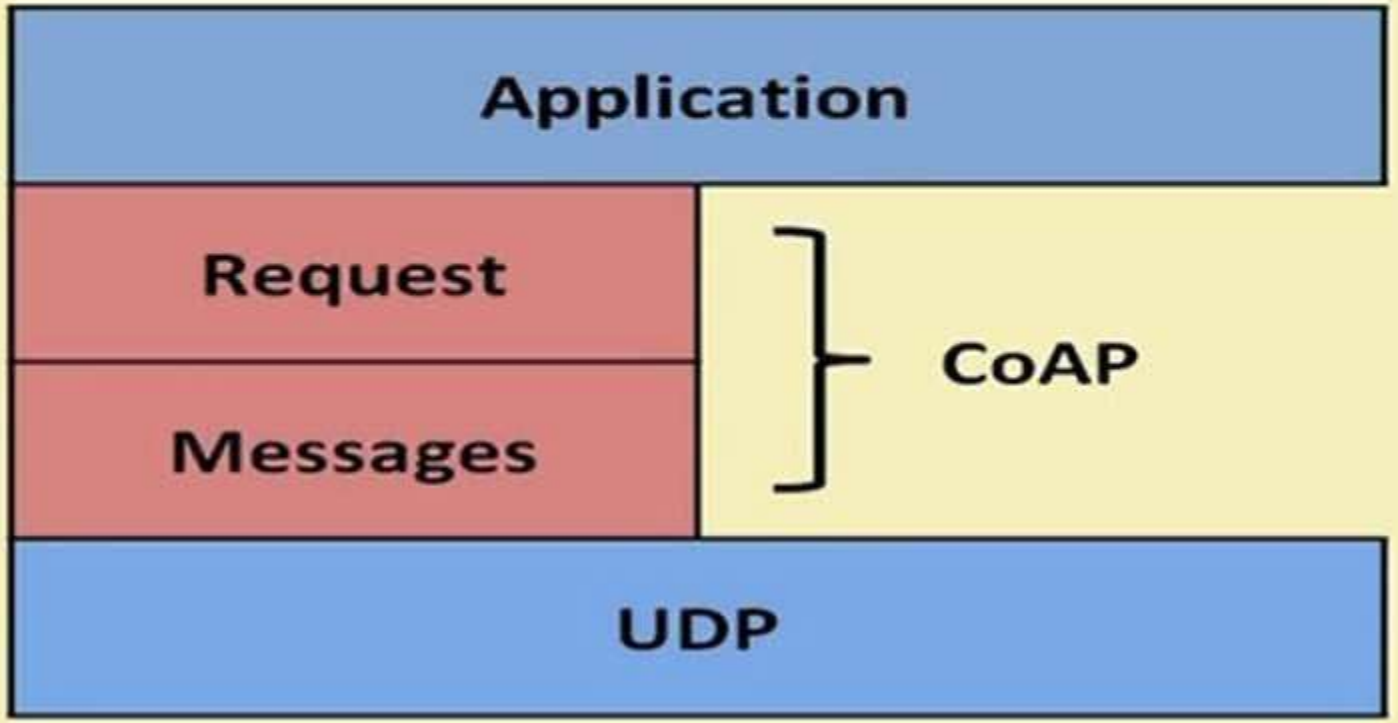
Introduction

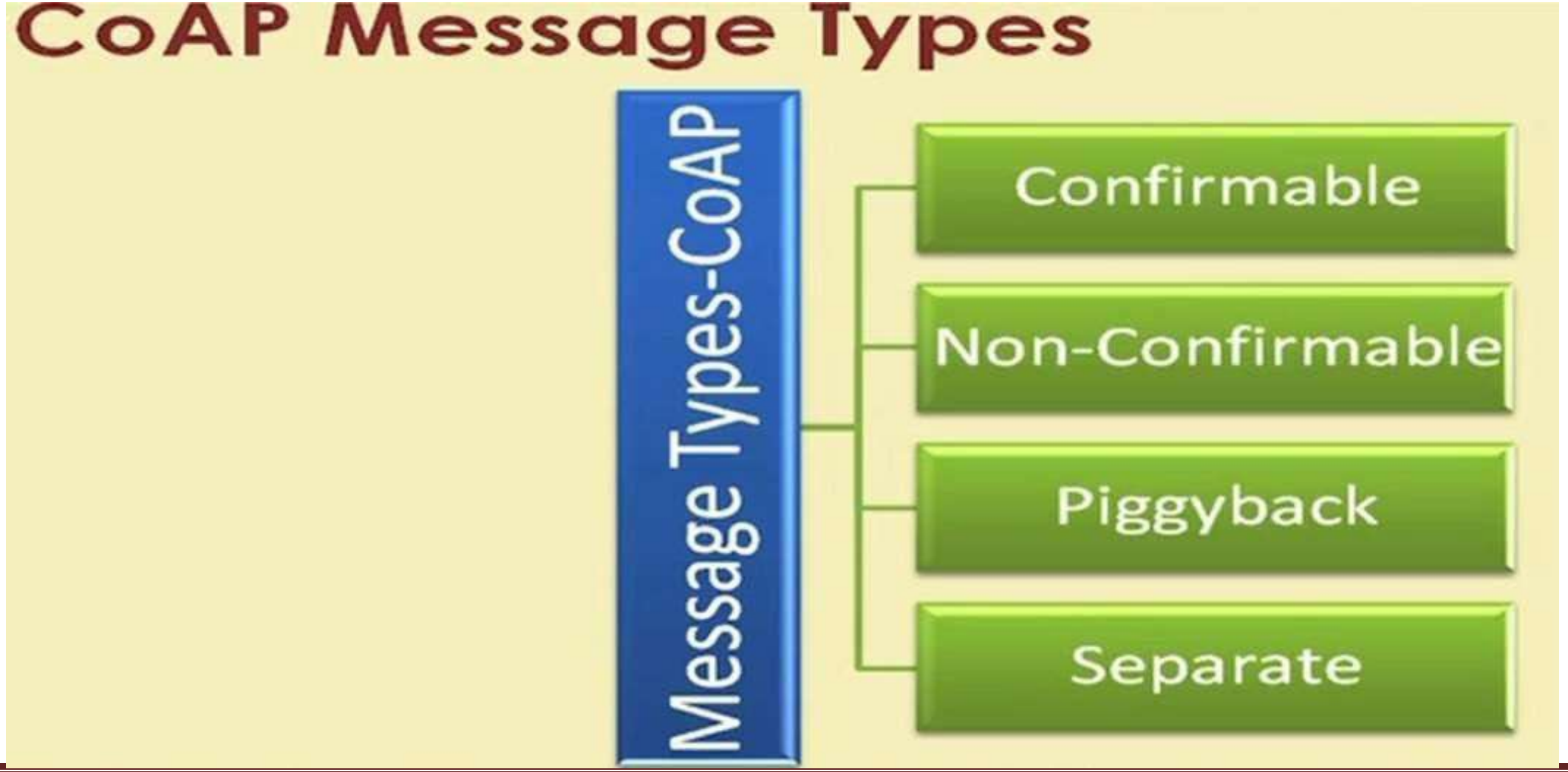
- ✓ CoAP – **Constrained Application Protocol**.
- ✓ **Web transfer protocol** for use with constrained nodes and networks.
- ✓ **Designed for Machine to Machine** (M2M) applications such as smart energy and building automation.
- ✓ Based on **Request-Response model** between end-points
- ✓ Client-Server interaction is **asynchronous over a datagram oriented transport protocol** such as UDP

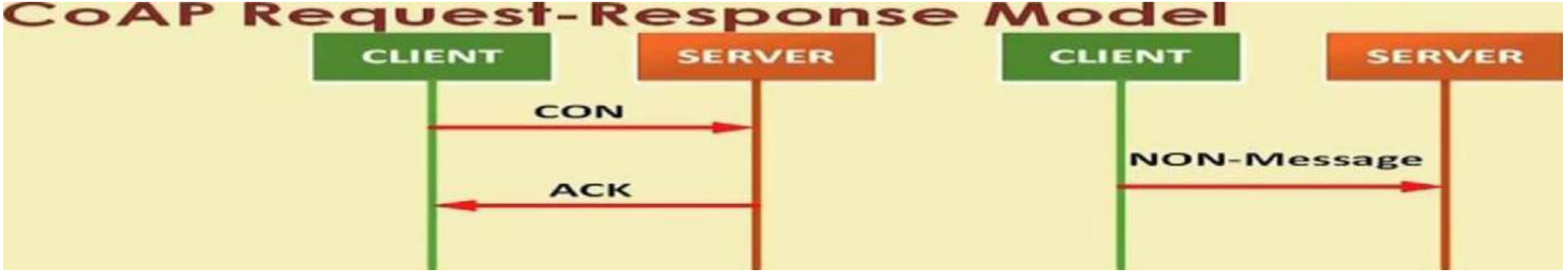
- ✓ The Constrained Application Protocol (CoAP) is a session layer protocol designed by IETF Constrained RESTful Environment (CoRE) working group to provide lightweight RESTful (HTTP) interface.
- ✓ Representational State Transfer (REST) is the standard interface between HTTP client and servers.
- ✓ Lightweight applications such as those in IoT, could result in significant overhead and power consumption by REST.
- ✓ CoAP is designed to enable low-power sensors to use RESTful services while meeting their power constraints.

- ✓ Built over UDP, instead of TCP (which is commonly used with HTTP) and has a light mechanism to provide reliability.
- ✓ CoAP architecture is divided into two main sub-layers:
  - Messaging
  - Request/response.
- ✓ The messaging sub-layer is responsible for reliability and duplication of messages, while the request/response sub-layer is responsible for communication.
- ✓ CoAP has four messaging modes:
  - Confirmable
  - Non-confirmable
  - Piggyback

CoAP Position




CoAP Message Types
Message Types-CoAP
Confirmable
Non-Confirmable
Piggyback
Separate



- Confirmable and non-confirmable modes represent the reliable and unreliable transmissions, respectively, while the other modes are used for request/response.
- Piggyback is used for client/server direct communication where the server sends its response directly after receivinj/ the message, i.e., within the acknowledgment message.
- On the other hand, the separate mode is used when the server response comes in a message separate from the acknowledgment, and may take some time to be sent by the server.
- Similar to HTTP, CoAP utilizes GET, PUT, PUSH, DELETE messages requests to retrieve, create, update, and delete, respectively

CoAP Request-Response Model

Piggyback Message

Separate Message

Features

- ’" Reduced overheads and parsing complexity.
- +" URL and content-type support.
- ’" Support for the discovery of resources provigeg by known CoAP services.
- '^ Siiviple subscription for a resource, and resulting push notificatious.

xmpp

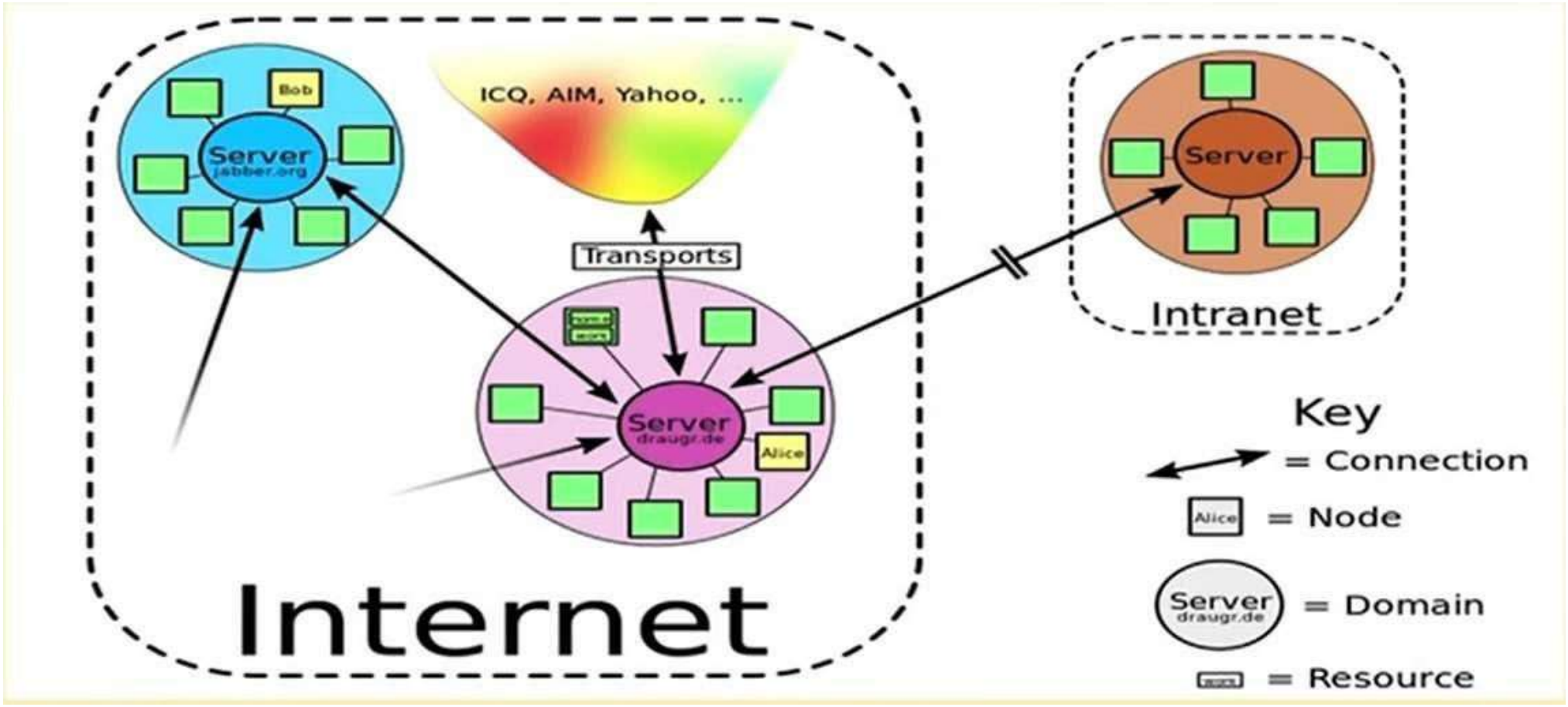
Introduction

- ✓ **XMPP – Extensible Messaging and Presence Protocol.**
- ✓ A communication protocol for **message-oriented middleware** based on XML (Extensible Markup Language).
- ✓ Real-time exchange of structured data.
- ✓ It is an open standard protocol.

- ✓ XMPP uses a **client-server architecture**.
- ✓ As the model is **decentralized**, no central server is required.
- ✓ XMPP provides for the **discovery of services** residing locally or across a network, and the **availability information** of these services.
- ✓ Well-suited for cloud computing where virtual machines, networks, and firewalls would otherwise present obstacles to alternative service discovery and presence-based solutions.
- ✓ Open means to support machine-to-machine or peer-to-peer communications across a diverse set of networks.

Highlights

- ✓ Decentralization – No central server; anyone can run their own XMPP server.
- ✓ Open standards – No royalties or granted permissions are required to implement these specifications
- ✓ Security – Authentication, encryption, etc.
- ✓ Flexibility – Supports interoperability


Bob
Server
jabber.org
ICQ, AIM, Yahoo, ...
Transports
Server
draugr.de
Alice
Server
Intranet
Internet
Key
= Connection
Alice
= Node
Server
draugr.de
= Domain
= Resource

Core XMPP Technologies

Core
- information about the core XMPP technologies for XML streaming

Jingle
- multimedia signalling for voice, video, file transfer

Multi-user Chat
- flexible, multi-party communication

PubSub
- alerts and notifications for data syndication

BOSH
- HTTP binding for XMPP

Weaknesses

- ✓ Does not support QoS.
- ✓ Text based communications induces higher network overheads.
- ✓ Binary data must be first encoded to base64 before transmission.

Applications

- ✓ Publish-subscribe systems
- ✓ Signaling for VoIP
- ✓ Video
- ✓ File transfer
- ✓ Gaming
- ✓ Internet of Things applications
  - Smart grid
  - Social networking services

2. Message Queue Telemetry Transport Protocol (MQTT)

MQTT (*Message Queue Telemetry Transport*) is a messaging protocol developed with the aid of Andy Stanford-Clark of IBM and Arlen Nipper of Arcom in 1999 and is designed for M2M communication. It's normally used for faraway tracking in IoT. Its primary challenge is to gather statistics from many gadgets and delivery of its infrastructure. MQTT connects gadgets and networks with packages and middleware. All the devices hook up with facts concentrator servers like IBM's new message sight appliance. MQTT protocols paintings on top of TCP to offer easy and dependable streams of information.

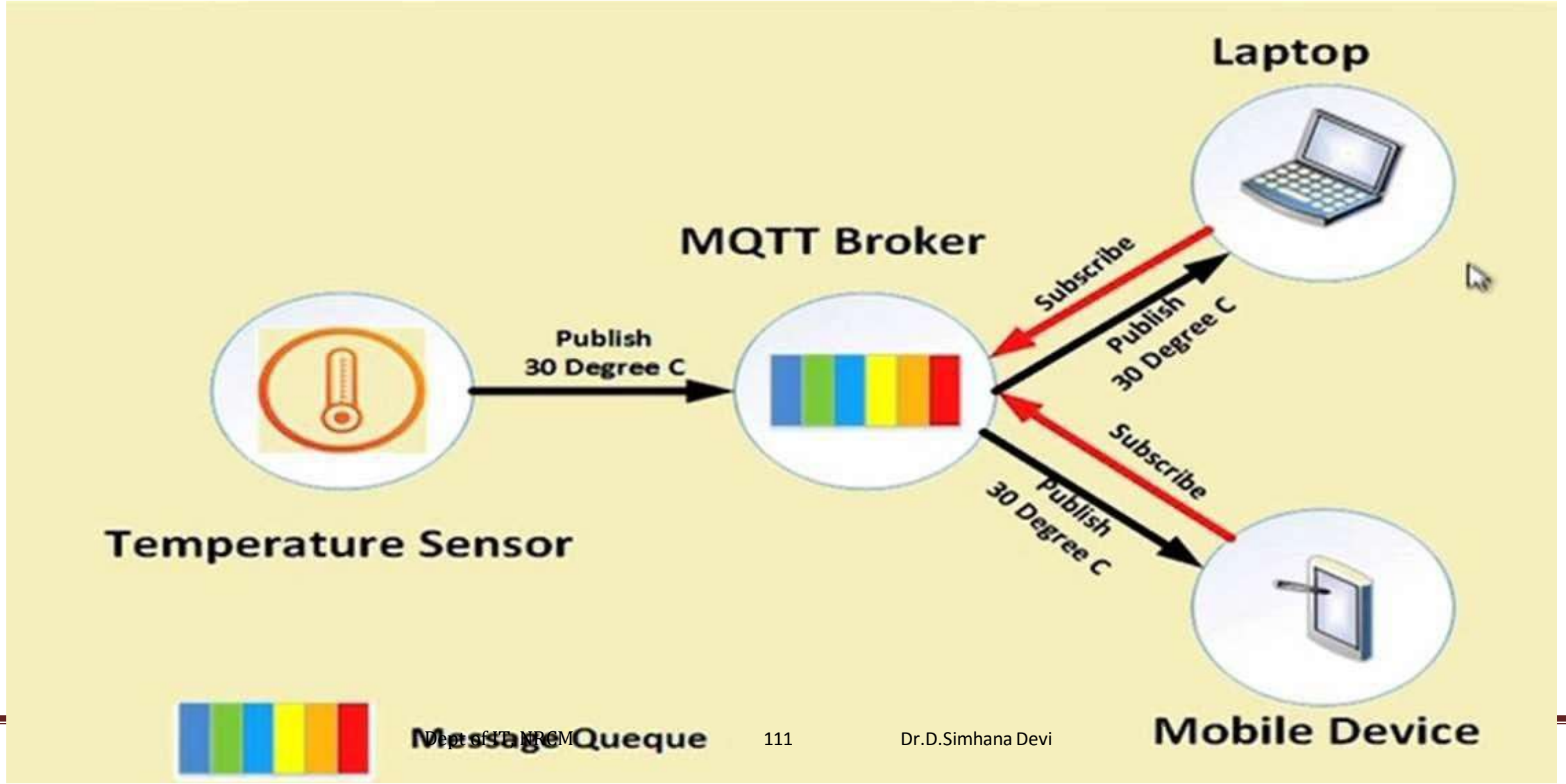
These IoT protocols include 3 foremost additives: subscriber, publisher, and dealer. The writer generates the information and transmits the facts to subscribers through the dealer. The dealer guarantees safety by means of move-checking the authorization of publishers and subscribers.

Introduction

- ✓ **Message Queue Telemetry Transport.**
- ✓ ISO standard (ISO/IEC PRF 20922).
- ✓ It is a publish-subscribe-based lightweight messaging protocol for use in conjunction with the TCP/IP protocol.
- ✓ MQTT was introduced by IBM in 1999 and standardized by OASIS in 2013.
- ✓ Designed to provide connectivity (mostly embedded) between applications and middle-wares on one side and networks and communications on the other side.

- ✓ A message broker controls the publish-subscribe messaging pattern.
- ✓ A topic to which a client is subscribed is updated in the form of messages and distributed by the message broker.
- ✓ Designed for:
  - Remote connections
  - Limited bandwidth
  - Small-code footprint

MQTT Components


Laptop
MQTT Broker
Subscribe
Publish
30 Degree C
Publish
30 Degree C
Subscribe
Publish
30 Degree C
Temperature Sensor
Message Queque
Mobile Device

Communication

- ✓ The protocol uses a **publish/subscribe** architecture (HTTP uses a request/response paradigm).
- ✓ Publish/subscribe is **event-driven** and enables messages to be pushed to clients.
- ✓ The central **communication point is the MQTT broker**, which is in charge of dispatching all messages between the senders and the rightful receivers.
- ✓ Each client that publishes a message to the broker, includes a **topic** into the message. The **topic is the routing information for the broker**.

- ✓ Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client.
- ✓ Therefore the clients don't have to know each other. They only communicate over the topic.
- ✓ This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.

MQTT Topics

- ✓ A topic is a **simple string** that can have more hierarchy levels, which are separated by a slash.
- ✓ A sample topic for sending temperature data of the living room could be *house/living-room/temperature*.
- ✓ On one hand the client (e.g. mobile device) can subscribe to the exact topic or on the other hand, it can use a **wildcard**.

- ✓ The subscription to *house/+/temperature* would result in all messages sent to the previously mentioned topic *house/living-room/temperature,* as well as any topic with an arbitrary value in the place of living room, such as *house/kitchen/temperature*.
- ✓ The plus sign is a **single level wild card** and only allows arbitrary values for one hierarchy.
- ✓ If more than one level needs to be subscribed, such as, the entire sub-tree, there is also a **multilevel wildcard** (#).
- ✓ It allows to subscribe to all underlying hierarchy levels.
- ✓ For example *house/#* is subscribing to all topics beginning with *house*.

Applications

- ✓ **Facebook Messenger** uses MQTT for online chat.
- ✓ **Amazon Web Services** use Amazon IoT with MQTT.
- ✓ **Microsoft Azure** IoT Hub uses MQTT as its main protocol for telemetry messages.
- ✓ The **EVRYTHNG IoT platform** uses MQTT as an M2M protocol for millions of connected products.
- ✓ **Adafruit** launched a free MQTT cloud service for IoT experimenters called Adafruit IO.

SMQTT

- ✓ **Secure MQTT** is an extension of MQTT which uses encryption based on lightweight attribute based encryption.
- ✓ The main advantage of using such encryption is the broadcast encryption feature, in which one message is encrypted and delivered to multiple other nodes, which is quite common in IoT applications.
- ✓ In general, the algorithm consists of four main stages: setup, encryption, publish and decryption.