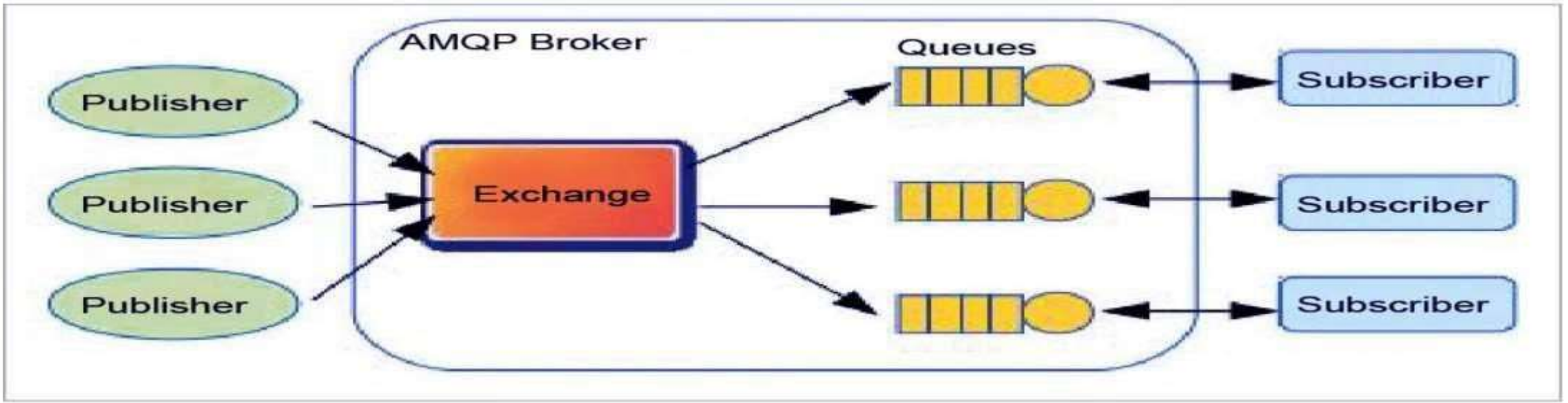
- ✓ In the setup phase, the subscribers and publishers register themselves to the broker and get a master secret key according to their developer's choice of key generation algorithm.
- ✓ When the data is published, it is encrypted and published by the broker which sends it to the subscribers, which is finally decrypted at the subscriber end having the same master secret key.
- ✓ The key generation and encryption algorithms are not standardized.
- ✓ SMQTT is proposed only to enhance MQTT security features.

3. Advanced Message Queuing Protocol (AMQP)

This was evolved by John O'Hara at JP Morgan Chase in London. AMQP is a software layer protocol for message-oriented middleware environment. It supports reliable verbal exchange through message transport warranty primitives like at-most-once, at least once and exactly as soon as shipping.

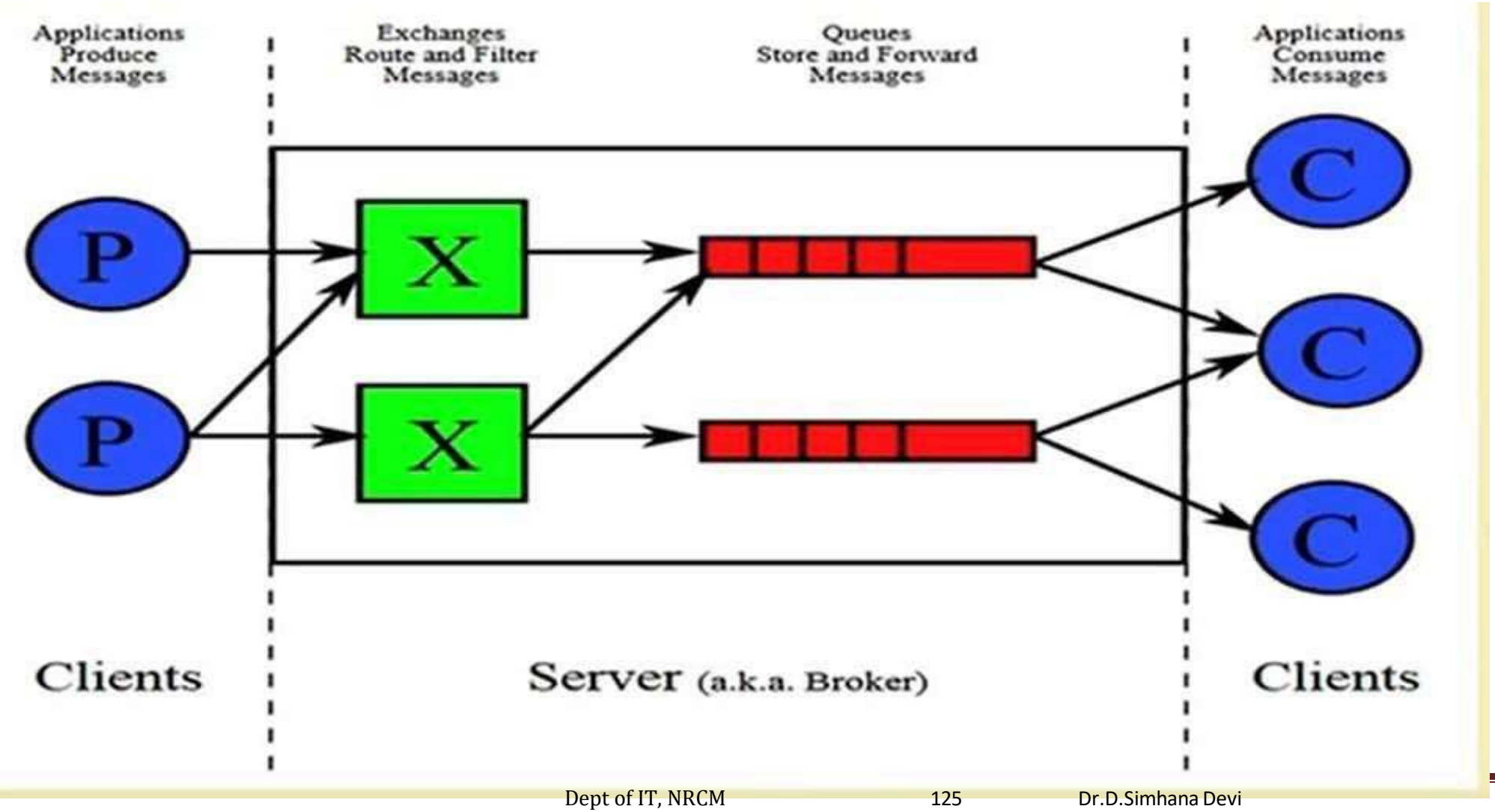
The AMQP – IoT protocols consist of hard and fast components that route and save messages within a broker carrier, with a set of policies for wiring the components together. The AMQP protocol enables patron programs to talk to the dealer and engage with the AMQP model. This version has the following three additives, which might link into processing chains in the server to create the favored capabilities.

- **Exchange:** Receives messages from publisher primarily based programs and routes them to 'message queues'.
- **Message Queue:** Stores messages until they may thoroughly process via the eating client software.
- **Binding:** States the connection between the message queue and the change.


AMQP Broker
Queues
Publisher
Publisher
Publisher
Exchange
Subscriber
Subscriber
Subscriber

Introduction

- ✓ **Advanced Message Queuing Protocol.**
- ✓ **Open standard for passing business messages** between applications or organizations.
- ✓ Connects between systems and business processes.
- ✓ It is a binary application layer protocol.
- ✓ Basic unit of data is a *frame*.
- ✓ ISO standard: **ISO/IEC 19464**


Applications
Produce
Messages
Exchanges
Route and Filter
Messages
Queues
Store and Forward
Messages
Applications
Consume
Messages
P
P
X
X
C
C
C
Clients
Server (a.k.a. Broker)
Clients


AMQP Features
Organizations
Technologies
Time
Space
Connects across
Features
Security
Reliability
Interoperability
Routing
Queuing
Open standard

AMQP Frame Types

- ✓ Nine AMQP frame types are defined that are used to initiate, control ar tear down the transfer of messages between two peers:
  - Open (connection open)
  - Begin (session open)
  - Attach (initiate new link)
  - Transfer (for sending actual messages)
  - Flow (controls message flow rate)
  - Disposition (Informs the changes in state of transfer)
  - Detach (terminate the link)
  - End (session close)
  - Close (connection close)

Message Delivery Guarantees

- ✓ *At-most-once*
  - each message is delivered once or never
- ✓ *At-least-once*
  - each message is certain to be delivered, but may do so multiple times
- ✓ *Exactly-once*
  - message will always certainly arrive and do so only once


Components
Exchange
• Part of Broker
• Receives messages and routes them to Queues
Queue
• Separate queues for separate business processes
• Consumers receive messages from queues
Bindings
• Rules for distributing messages (who can access what message, destination of the message)


AMQP Exchanges
Direct
Fan-out
Topic
Header

AMQP Features

- ✓ Targeted QoS (Selectively offering QoS to links)
- ✓ Persistence (Message delivery guarantees)
- ✓ Delivery of messages to multiple consumers
- ✓ Possibility of ensuring multiple consumption
- ✓ Possibility of preventing multiple consumption
- ✓ High speed protocol

Applications

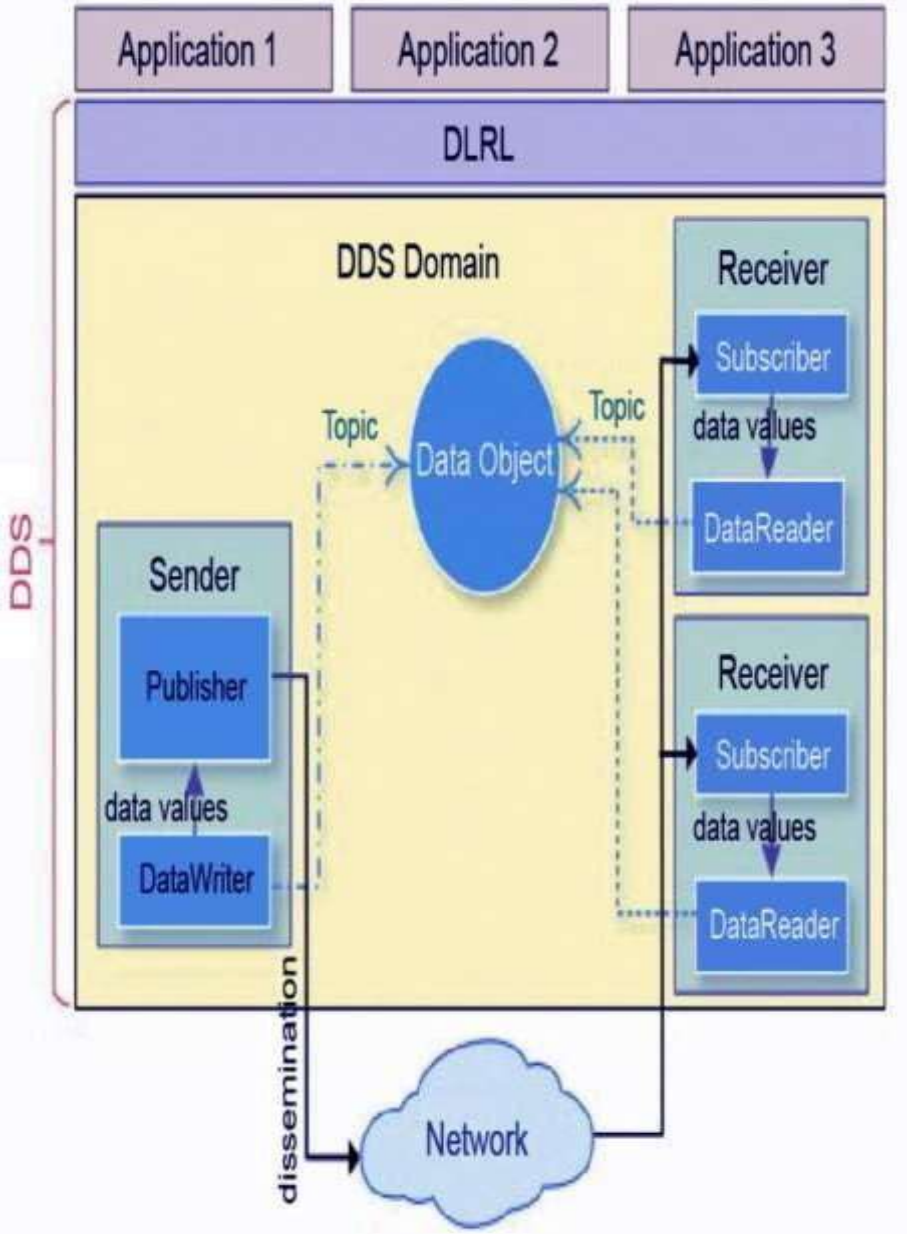
- ✓ Monitoring and global update sharing.
- ✓ Connecting different systems and processes to talk to each other.
- ✓ Allowing servers to respond to immediate requests quickly and delegate time consuming tasks for later processing.
- ✓ Distributing a message to multiple recipients for consumption.
- ✓ Enabling offline clients to fetch data at a later time.
- ✓ Introducing fully asynchronous functionality for systems.
- ✓ Increasing reliability and uptime of application deployments.

4. Data Distribution Service (DDS)

It enables a scalable, real-time, reliable, excessive-overall performance and interoperable statistics change via the submit-subscribe technique. DDS makes use of multicasting to convey high-quality QoS to applications.

DDS is deployed in platforms ranging from low-footprint devices to the cloud and supports green bandwidth usage in addition to the agile orchestration of system additives.

The DDS – IoT protocols have fundamental layers: facts centric submit-subscribe (dcps) and statistics-local reconstruction layer (dlrl). Dcps plays the task of handing over the facts to subscribers, and the dlrl layer presents an interface to dcps functionalities, permitting the sharing of distributed data amongst IoT enabled objects.



UNIT-1 PART 2 (CONT)

xmpp

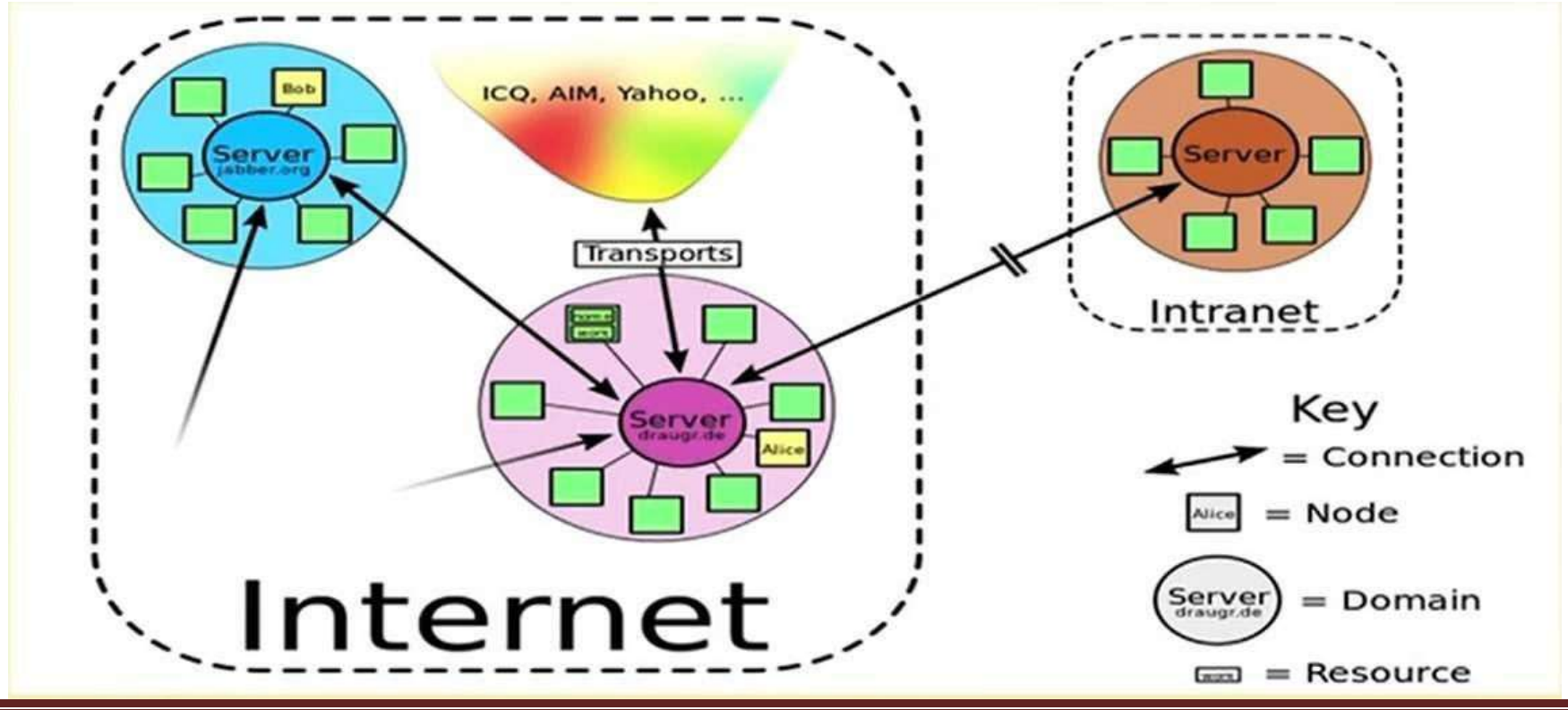
Introduction

- ✓ **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.

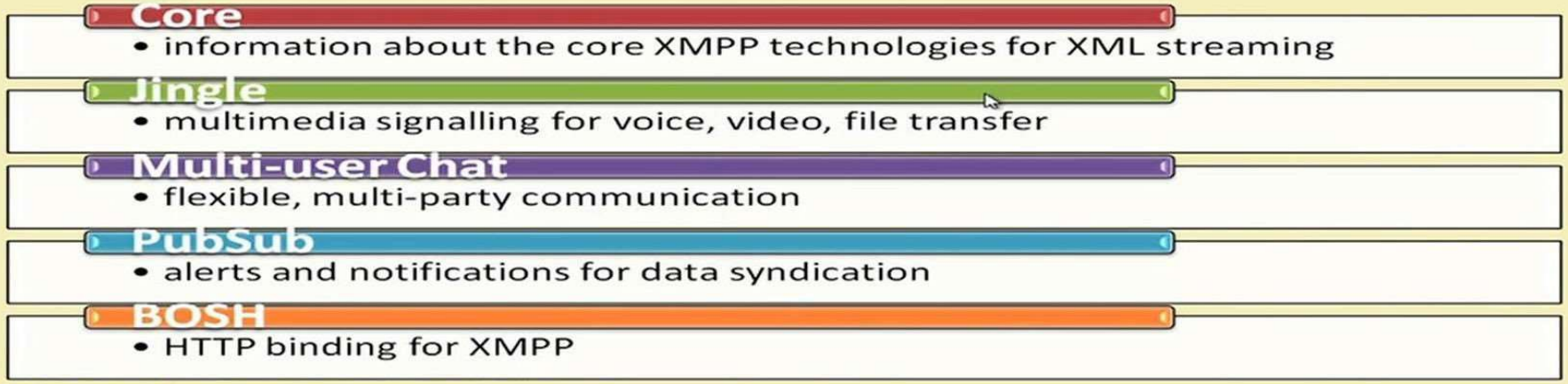
- ✓ XMPP uses a **client-server architecture**.
- ✓ As the model is **decentralized**, no central server is required.
- ✓ XMPP provides for the **discovery of services** residing locally or across a network, and the **availability information** of these services.
- ✓ Well-suited for cloud computing where 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



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.

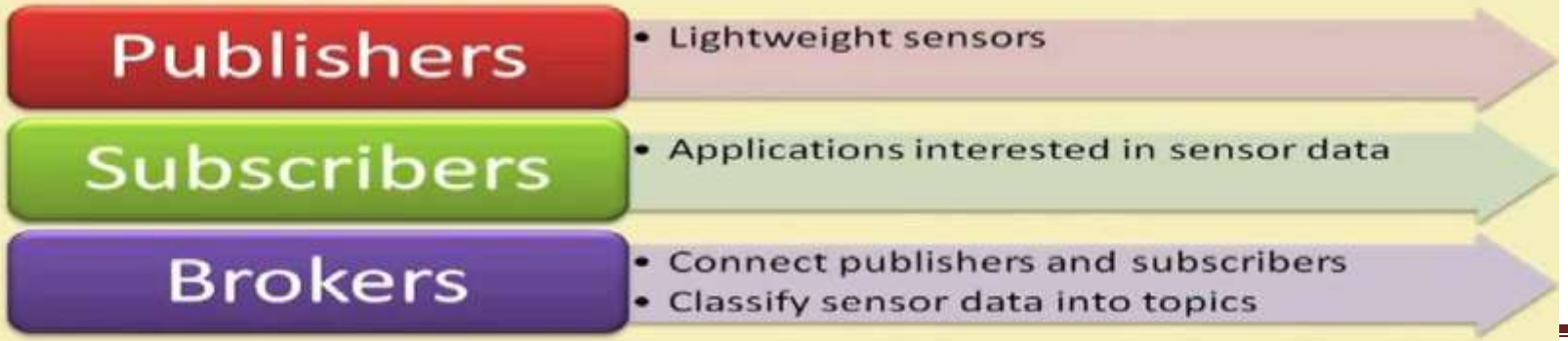
These IoT protocols include 3 foremost additives: subscriber, publisher, and dealer. The writer generates the information and transmits the facts to subscribers through the dealer. The dealer guarantees safety by means of move-checking the authorization of publishers and subscribers.

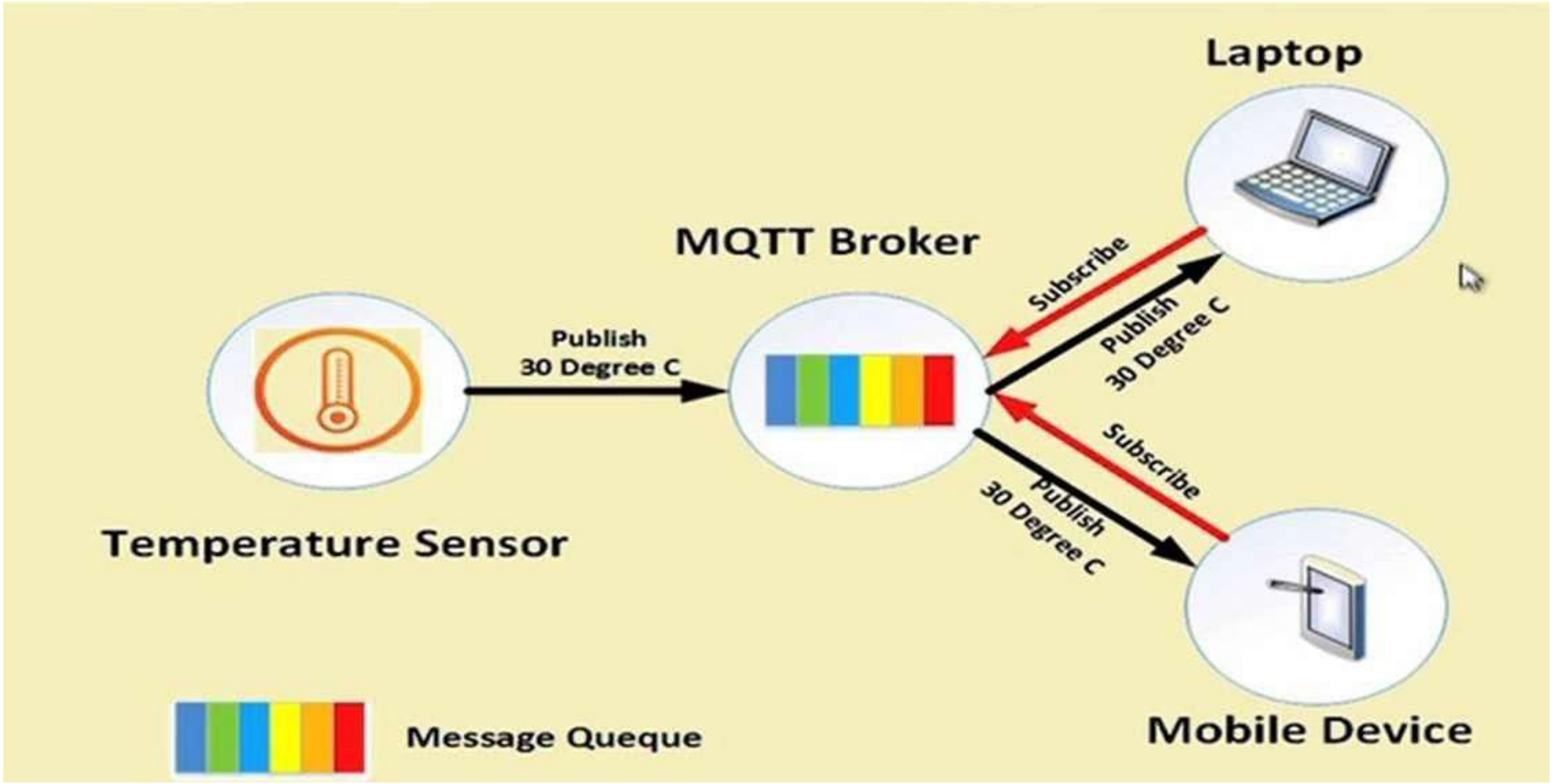
Introduction

- ✓ **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.

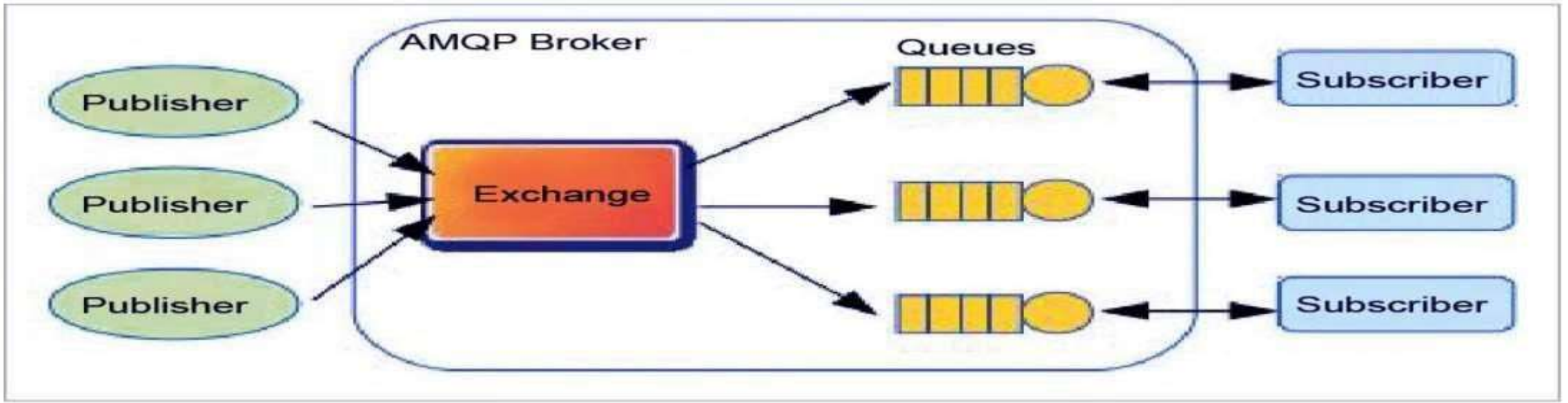
- ✓ In the setup phase, the subscribers and publishers register themselves to the broker and get a master secret key according to their developer's choice of key generation algorithm.
- ✓ When the data is published, it is encrypted and published by the broker which sends it to the subscribers, which is finally decrypted at the subscriber end having the same master secret key.
- ✓ The key generation and encryption algorithms are not standardized.
- ✓ SMQTT is proposed only to enhance MQTT security features.

3. Advanced Message Queuing Protocol (AMQP)

This was evolved by John O'Hara at JP Morgan Chase in London. AMQP is a software layer protocol for message-oriented middleware environment. It supports reliable verbal exchange through message transport warranty primitives like at-most-once, at least once and exactly as soon as shipping.

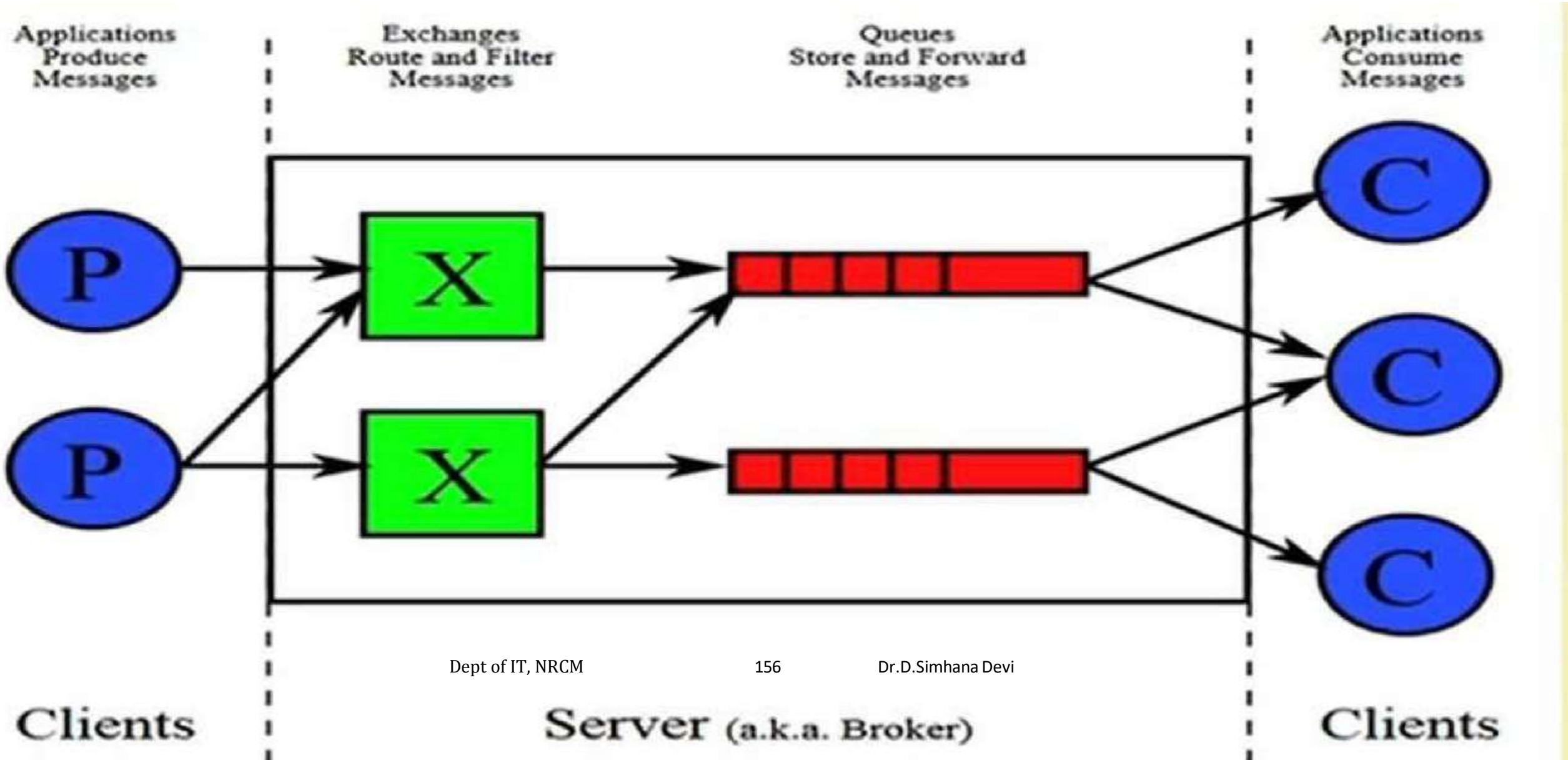
The AMQP – IoT protocols consist of hard and fast components that route and save messages within a broker carrier, with a set of policies for wiring the components together. The AMQP protocol enables patron programs to talk to the dealer and engage with the AMQP model. This version has the following three additives, which might link into processing chains in the server to create the favored capabilities.

- **Exchange:** Receives messages from publisher primarily based programs and routes them to 'message queues'.
- **Message Queue:** Stores messages until they may thoroughly process via the eating client software.
- **Binding:** States the connection between the message queue and the change.


AMQP Broker
Queues
Publisher
Publisher
Publisher
Exchange
Subscriber
Subscriber
Subscriber

Introduction

- ✓ **Advanced Message Queuing Protocol.**
- ✓ **Open standard for passing business messages** between applications or organizations.
- ✓ Connects between systems and business processes.
- ✓ It is a binary application layer protocol.
- ✓ Basic unit of data is a *frame*.
- ✓ ISO standard: **ISO/IEC 19464**


Applications
Produce
Messages
Exchanges
Route and Filter
Messages
Queues
Store and Forward
Messages
Applications
Consume
Messages
P
P
X
X
C
C
C
Clients
Server (a.k.a. Broker)
Clients

AMQP Features

Features

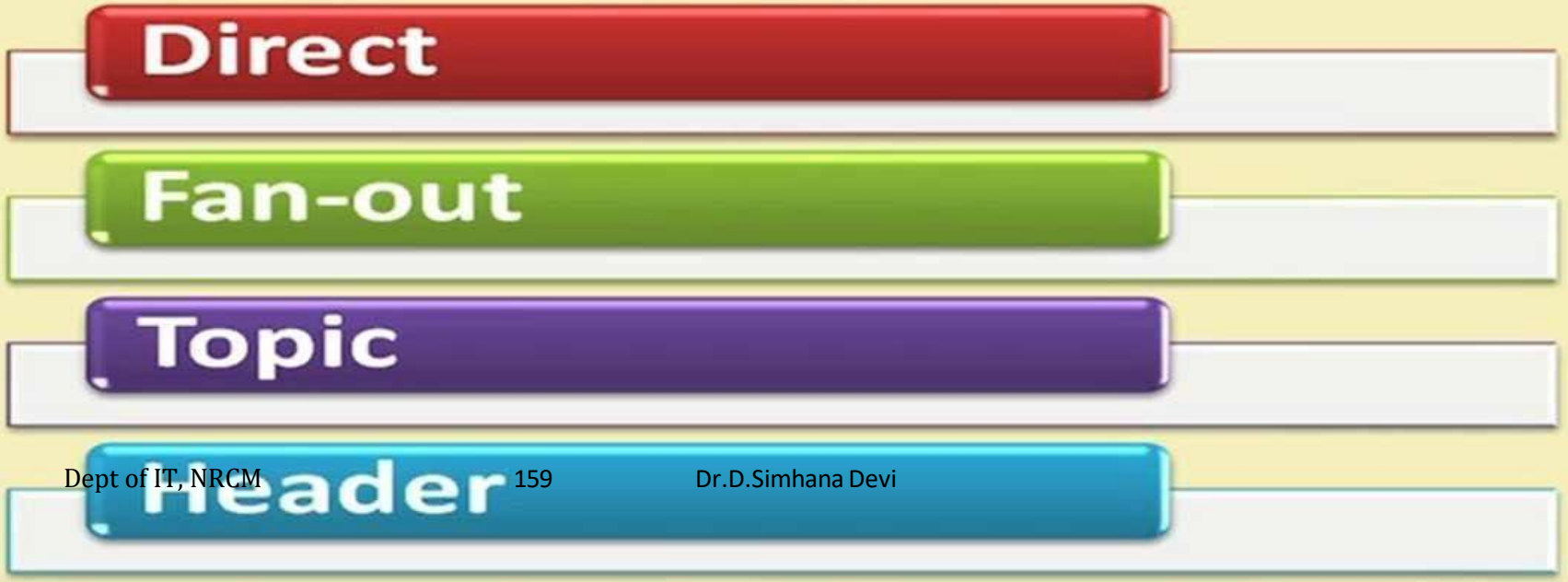
- Security
- Reliability
- Interoperability
- Routing
- Queuing
- Open standard

AMQP Frame Types

- ✓ Nine AMQP frame types are defined that are used to initiate, control ar tear down the transfer of messages between two peers:
  - Open (connection open)
  - Begin (session open)
  - Attach (initiate new link)
  - Transfer (for sending actual messages)
  - Flow (controls message flow rate)
  - Disposition (Informs the changes in state of transfer)
  - Detach (terminate the link)
  - End (session close)
  - Close (connection close)

Components

AMQP Exchanges



AMQP Features

- ✓ Targeted QoS (Selectively offering QoS to links)
- ✓ Persistence (Message delivery guarantees)
- ✓ Delivery of messages to multiple consumers
- ✓ Possibility of ensuring multiple consumption
- ✓ Possibility of preventing multiple consumption
- ✓ High speed protocol

Applications

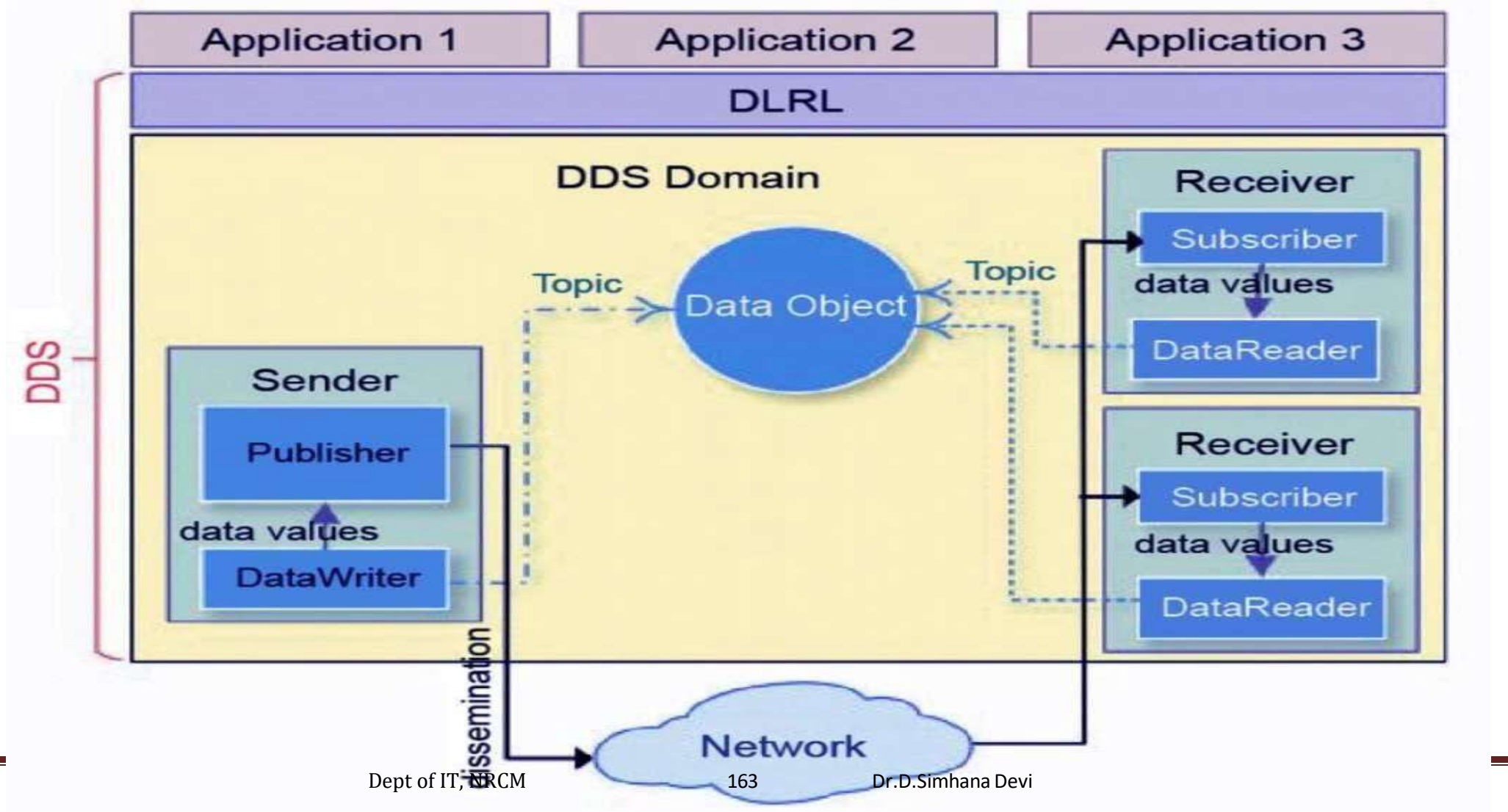
- ✓ Monitoring and global update sharing.
- ✓ Connecting different systems and processes to talk to each other.
- ✓ Allowing servers to respond to immediate requests quickly and delegate time consuming tasks for later processing.
- ✓ Distributing a message to multiple recipients for consumption.
- ✓ Enabling offline clients to fetch data at a later time.
- ✓ Introducing fully asynchronous functionality for systems.
- ✓ Increasing reliability and uptime of application deployments.

4. Data Distribution Service (DDS)

It enables a scalable, real-time, reliable, excessive-overall performance and interoperable statistics change via the submit-subscribe technique. DDS makes use of multicasting to convey high-quality QoS to applications.

DDS is deployed in platforms ranging from low-footprint devices to the cloud and supports green bandwidth usage in addition to the agile orchestration of system additives.

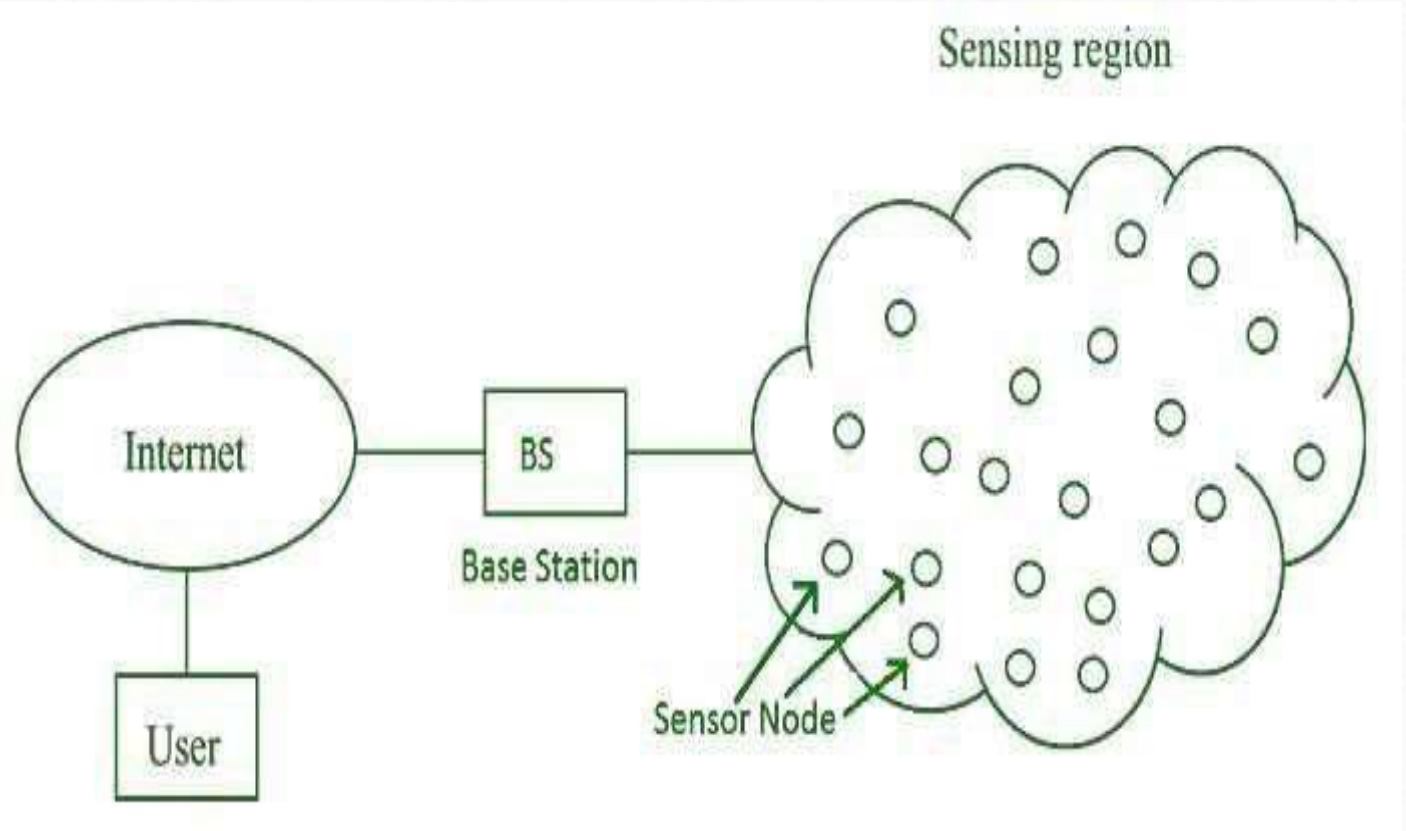
The DDS – IoT protocols have fundamental layers: facts centric submit-subscribe (dcps) and statistics-local reconstruction layer (dlrl). Dcps plays the task of handing over the facts to subscribers, and the dlrl layer presents an interface to dcps functionalities, permitting the sharing of distributed data amongst IoT enabled objects.


Application 1
Application 2
Application 3
DLRL
DDS
DDS Domain
Topic
Data Object
Topic
Sender
Publisher
data values
DataWriter
Receiver
Subscriber
data values
DataReader
Receiver
Subscriber
data values
DataReader
Dissemination
Network

SENSOR NETWORKS

Wireless Sensor Networks (WSNs)

- Consists of a large number of sensor nodes, densely deployed over an area.
- Sensor nodes are capable of collaborating with one another and measuring the condition of their surrounding environments (i.e. Light, temperature, sound, vibration).
- The sensed measurements are then transformed into digital signals and processed to reveal some properties of the phenomena around sensors.
- Due to the fact that the sensor nodes in WSNs have short radio transmission range, intermediate nodes act as relay nodes to transmit data towards the sink node using a multi-hop path.


Sensing region
Internet
BS
Base Station
User
Sensor Node

TYPES OF SENSOR NODE

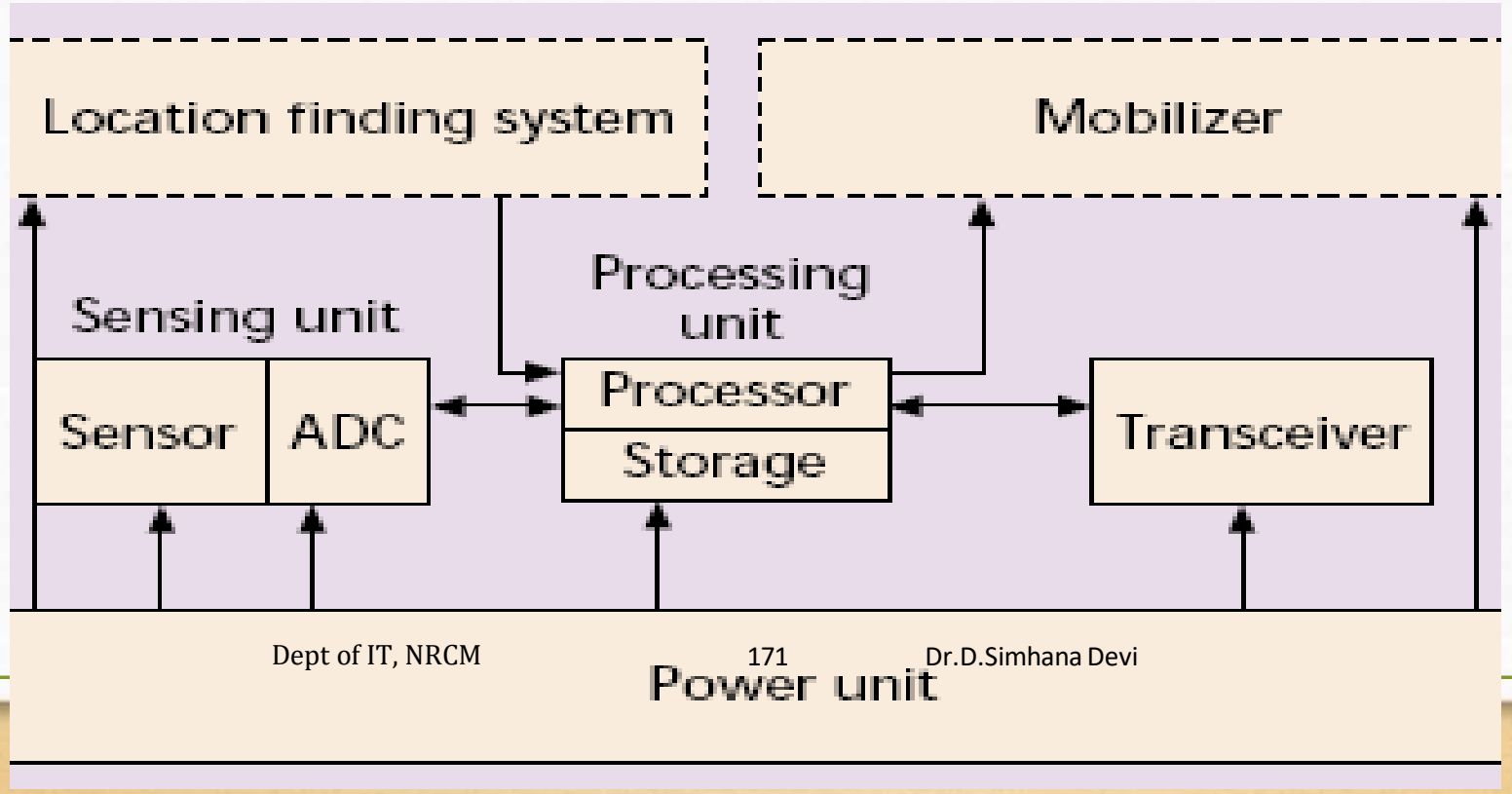
- SATIONARY SENSOR NODE
- MOBILE SENSOR NODE
  1. AERIAL SENSOR NODE
  2. TERESTRIAL SENSOR NODE
  3. UNDERWATER SENSOR NODE

SENSOR NODE

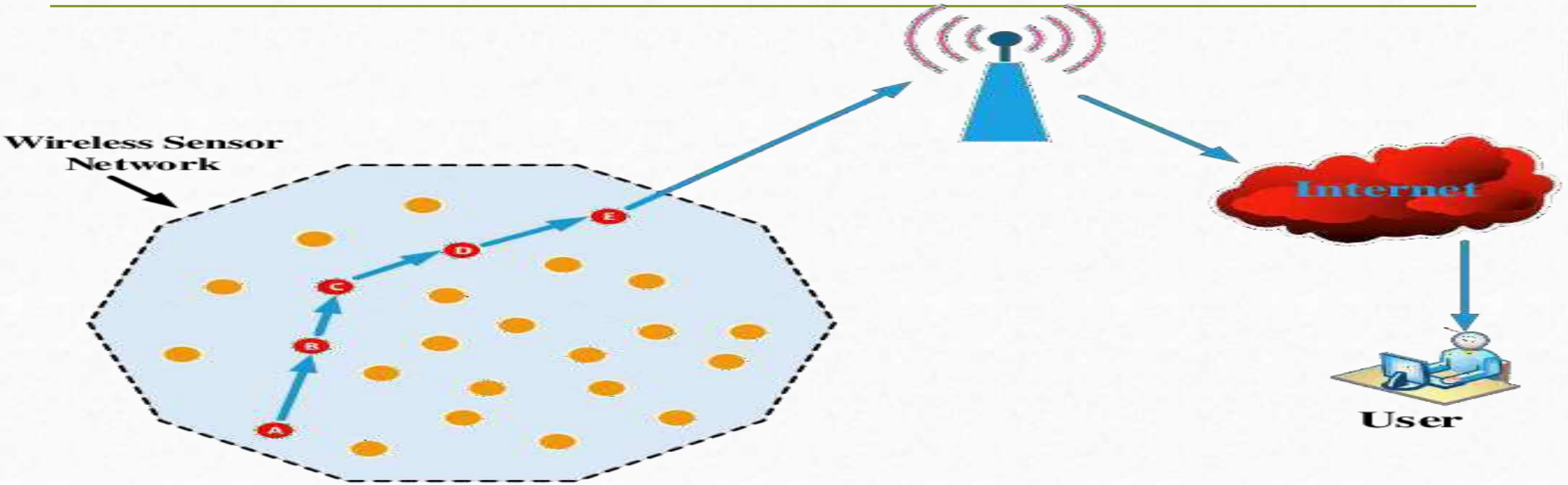
- SENSING UNIT
- COMMUNICATION UNIT (TRANS-RECEIVER)
- PROCESSING UNIT
- STORAGE UNIT
- ANALOG TO DIGITAL CONVERTER
- OPTIONAL UNITS (LOCATION FINDING SYSTEMS Eg. GPS)

- POWER (BATTERY)

BASIC COMPONENTS OF SENSOR NODE



MULTI-HOP PATH



Sensor Nodes

- **Multifunctional**
  - The number of sensor nodes used depends on the application type.
- **Short transmission ranges**
- **Have OS** (e.g., TinyOS).
- **Battery Powered** – Have limited life.

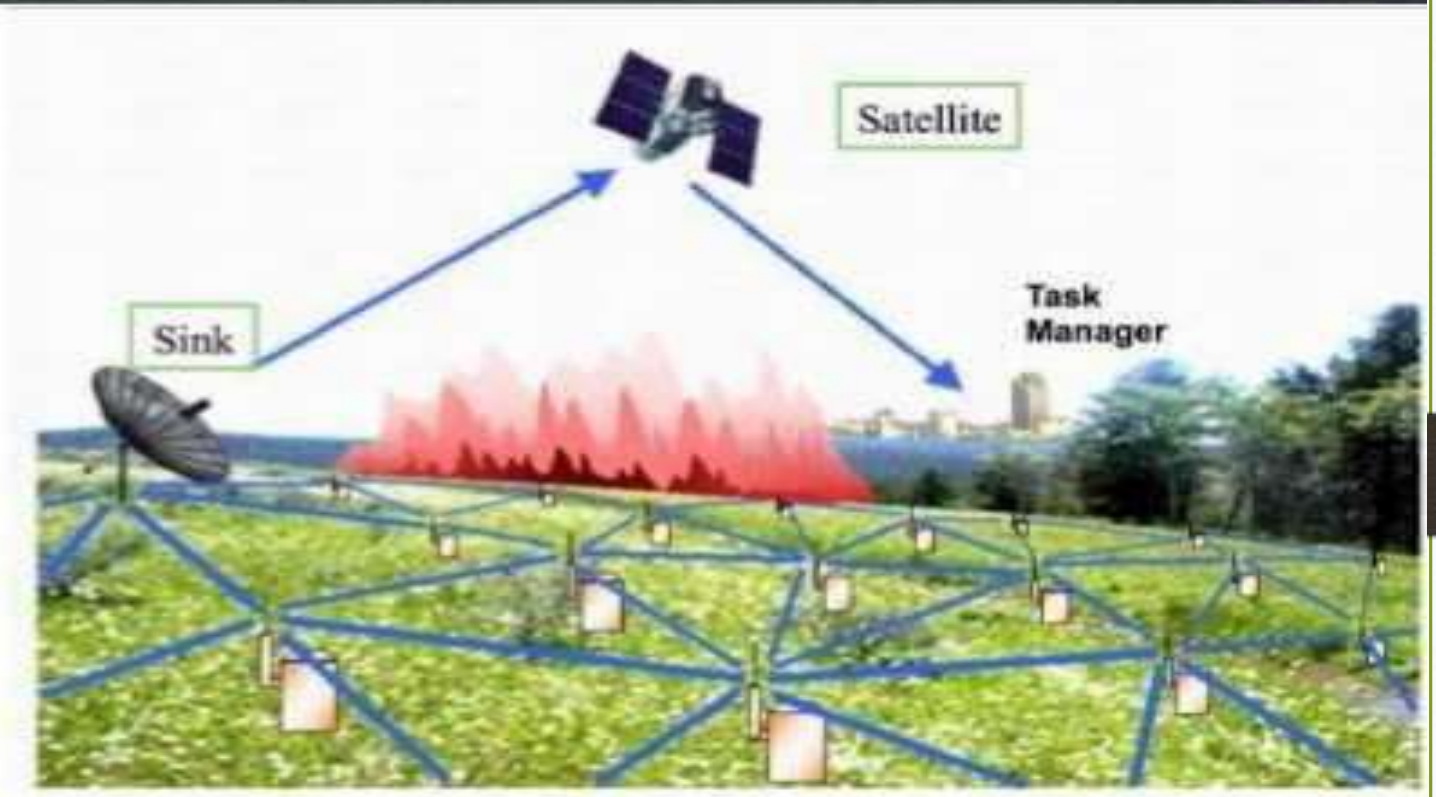
Image source: Wikimedia Commons

CONSTRAINTS OF SENSOR NODES

- Small size, typically less than a cubic cm.
- Must consume extremely low power.
- Operate in an unattended manner in a highly dense area.
- Should have low production cost and be dispensable.
- Be autonomous
- Be adaptive to the environment.

Applications of WSN-

- The Military Applications
- The Medical Applications
- Environmental Monitoring
- Target tracking
- Industrial Application
- Infrastructure and protection application



CHALLENGES

SCALABILITY

- Providing acceptable levels of service in the presence of large number of nodes.
- Typically, throughput decreases at a rate of 1/root N, N=number of nodes

QUALITY OF SERVICE

- Offering guarantees in terms of bandwidth, delay, jitter, packet loss probability.
- Limited bandwidth, unpredictable changes in RF channel characteristics.

ENERGY EFFIENCY

- Nodes have limited battery power
- Nodes need to cooperate with other nodes for relaying their information

SECURITY

- Open medium
- Nodes prone to malicious attacks, infiltration, eavesdropping, interference.

Sensor Web

WNS: Web Notification Services

SCS: Sensor Collection Services

SPS: Sensor Planning Services

SensorML: Sensor Modeling language

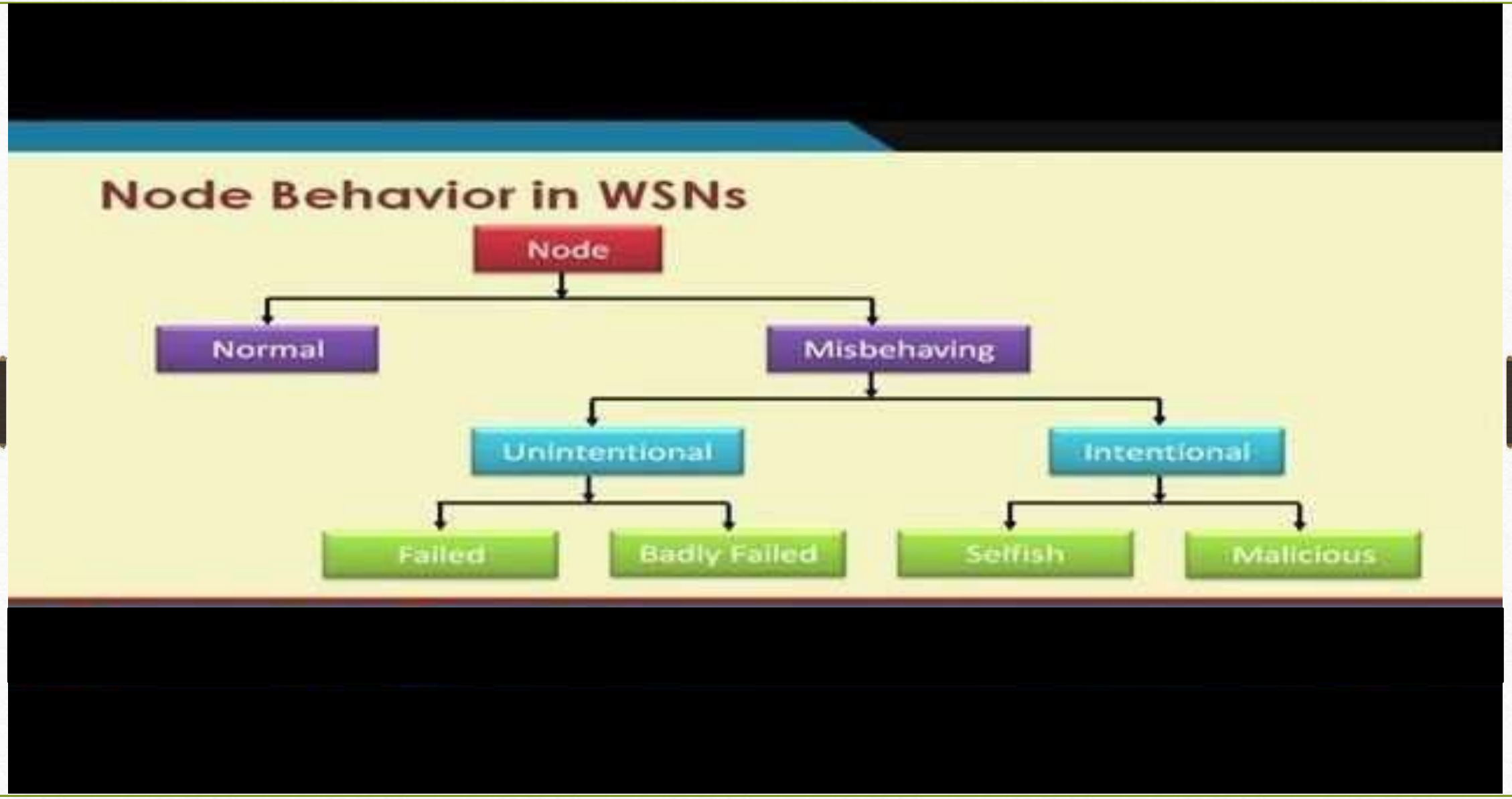
Source: X. Chu and R. Buyya, "Service Oriented Sensor Web", Sensor Networks and Configuration, Springer, 2007, pp. 51-74.

Sensor Web Entanglement

- Observations & measurements (O&M)
- Sensor model language (sensorml)
- Transducer model language (transducerml or TML)
- Sensor observations service (SOS)
- Sensor planning service (SPS)
- Sensor alert service (SAS)
- Web notification services (WNS)

Node Behaviour in WSN

- In order to have communication among different sensor nodes, these nodes they basically have to cooperate with one another.
- In order for the network to function if the nodes do not cooperate they will not be able to function.
- To promote cooperation we need to understand the behavior of the different nodes in the network.


Node Behavior in WSNs
Node
Normal
Misbehaving
Unintentional
Intentional
Failed
Badly Failed
Selfish
Malicious

Node Behavior in WSNs (contd.)

- Normal nodes work perfectly in ideal environmental conditions
- Failed nodes are simply those that are unable to perform an operation; this could be because of power failure and environmental events.
- Badly failed nodes exhibit features of failed nodes but they can also send false routing messages which are a threat to the integrity of the network.

Node Behavior in WSNs (contd.)

- Selfish nodes are typified by their unwillingness to cooperate, as the protocol requires whenever there is a personal cost involved. Packet dropping is the main attack by selfish nodes.
- Malicious nodes aim to deliberately disrupt the correct operation of the routing protocol, denying network service if possible.