Module 1:

Overview of Internet of Things:

Syllabus:

IoT Conceptual Framework, IoT Architectural View, Technology Behind IoT, Sources of IoT,M2M communication, Examples of IoT. Modified OSI Model for the IoT/M2M Systems, data enrichment, data consolidation and device management at IoT/M2M Gateway, web communication protocols used by connected IoT/M2M devices, Message communication protocols (CoAP-SMS, CoAPMQ, MQTT,XMPP) for IoT/M2M devices.

Definition:

The **Internet of Things (IoT)** is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

OR

The **Internet of things** refers to a type of network to connect anything with the Internet based on stipulated protocols through information sensing equipments to conduct information exchange and communications in order to achieve smart recognitions, positioning, tracing, monitoring, and administration.

Characteristics of IoT:

The fundamental characteristics of IoT are as follows:

- **Interconnectivity:** In IoT, anything can be interconnected with the global information and communication infrastructure.
- **Things-related services:** The IoT is capable of providing thing-related services within the constraints, such as privacy protection and semantic consistency between physical things and their associated virtual things. In order to provide thing-related services within the constraints of things, both the technologies in physical world and information world will change.
- **Heterogeneity:** The devices in the IoT are heterogeneous as based on different hardware platforms and networks. They can interact with other devices or service platforms through different networks.
- **Dynamic changes:** The state of devices change dynamically, e.g., sleeping and waking up, connected and/or disconnected as well as the

context of devices including location and speed. Moreover, the number of devices can change dynamically.

- **Enormous scale:** The number of devices that need to be managed and that communicate with each other will be at least an order of magnitude larger than the devices connected to the current Internet
- **Safety:** As benefits are more with the usage, safety becomes an ad joint issue. This includes the safety of personal data and the safety of physical well-being. Securing the endpoints, the networks, and the data moving across all of it means creating a security paradigm that will scale.
- **Connectivity:** Connectivity enables network accessibility and compatibility. Accessibility is getting on a network while compatibility provides the common ability to consume and produce data.

Basic IoT Architectural View [only for understanding purpose but forms the base for the next topic]:

IOT architecture consists of different layers of technologies supporting IOT.

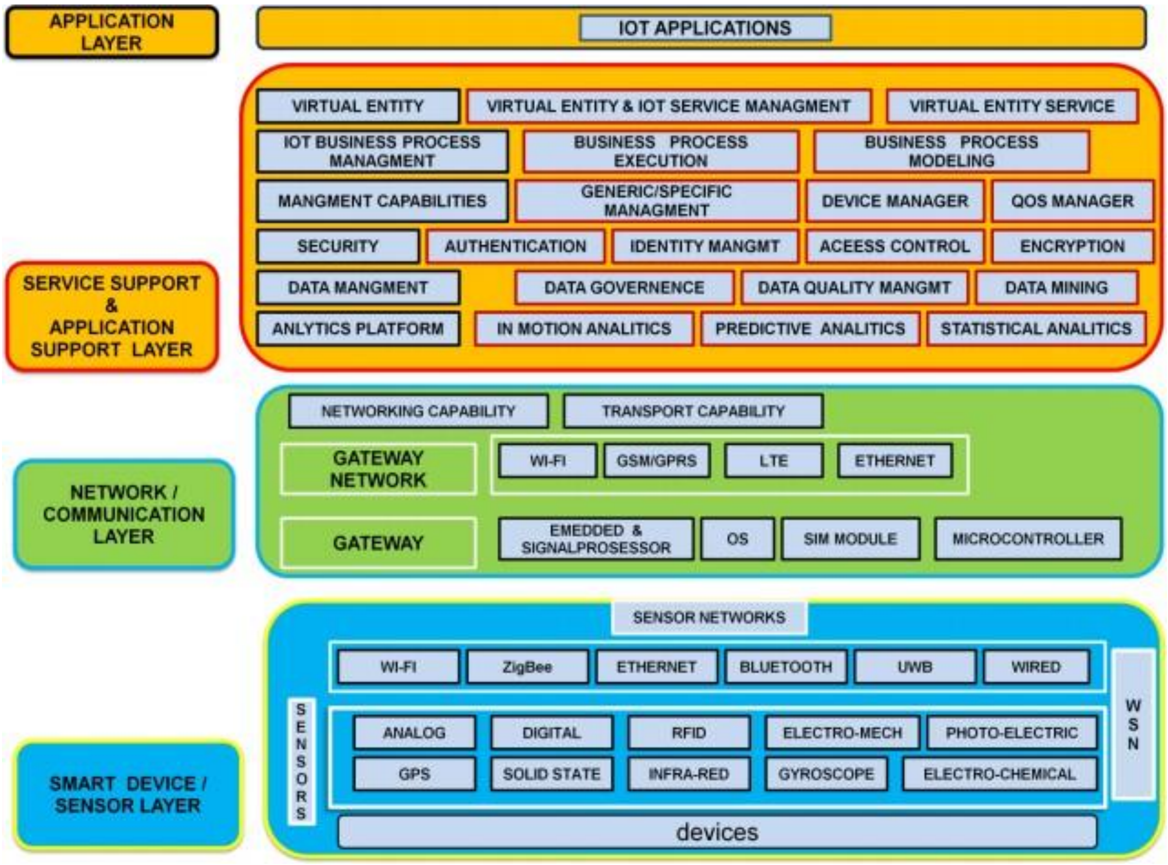


Figure1: Architectural view of IoT

1. **Smart device / sensor layer:**

❖ The lowest layer is made up of smart objects integrated with sensors.

- ❖ The sensors enable the interconnection of the physical and digital worlds allowing real-time information to be collected and processed.
- ❖ There are various types of sensors for different purposes.
- ❖ The sensors have the capacity to take measurements such as temperature, air quality, speed, humidity, pressure, flow, movement and electricity etc. In some cases, they may also have a degree of memory, enabling them to record a certain number of measurements.
- ❖ A sensor can measure the physical property and convert it into signal that can be understood by an instrument.
- ❖ Sensors are grouped according to their unique purpose such as environmental sensors, body sensors, home appliance sensors and vehicle telemetric sensors, etc.
- ❖ Most sensors require connectivity to the sensor gateways. This can be in the form of a Local Area Network (LAN) such as Ethernet and Wi-Fi connections or Personal Area Network (PAN) such as Zigbee, Bluetooth and Ultra Wideband (UWB).
- ❖ Sensors that use low power and low data rate connectivity, they typically form networks commonly known as wireless sensor networks (WSNs).

2. **Gateways and Networks:**
   - ❖ Massive volume of data will be produced by tiny sensors.
   - ❖ It requires a robust and high performance wired or wireless network infrastructure as a transport medium.
   - ❖ Current networks, often tied with very different protocols, have been used to support machine-to-machine (M2M) networks and their applications.
   - ❖ With demand needed to serve a wider range of IOT services and applications such as high speed transactional services, context-aware applications, etc, multiple networks with various technologies and access protocols are needed to work with each other in a heterogeneous configuration.
   - ❖ These networks can be in the form of a private, public or hybrid models and are built to support the communication requirements for latency, bandwidth or security.
   - ❖ Various gateways (microcontroller, microprocessor...) & gateway networks (WI-FI, GSM, GPRS...) are shown in figure 1
3. **Management Service Layer**:
   - ❖ The management service renders the processing of information possible through analytics, security controls, process modeling and management of devices.
   - ❖ One of the important features of the management service layer is the business and process rule engines.

- ❖ IOT brings connection and interaction of objects and systems together providing information in the form of events or contextual data such as temperature of goods, current location and traffic data.
- ❖ Some of these events require filtering or routing to post processing systems such as capturing of periodic sensory data, while others require response to the immediate situations such as reacting to emergencies on patient's health conditions.
- ❖ The rule engines support the formulation of decision logics and trigger interactive and automated processes to enable a more responsive IOT system. In the area of analytics, various analytics tools are used to extract relevant information from massive amount of raw data and to be processed at a much faster rate.

4. **Application Layer**:

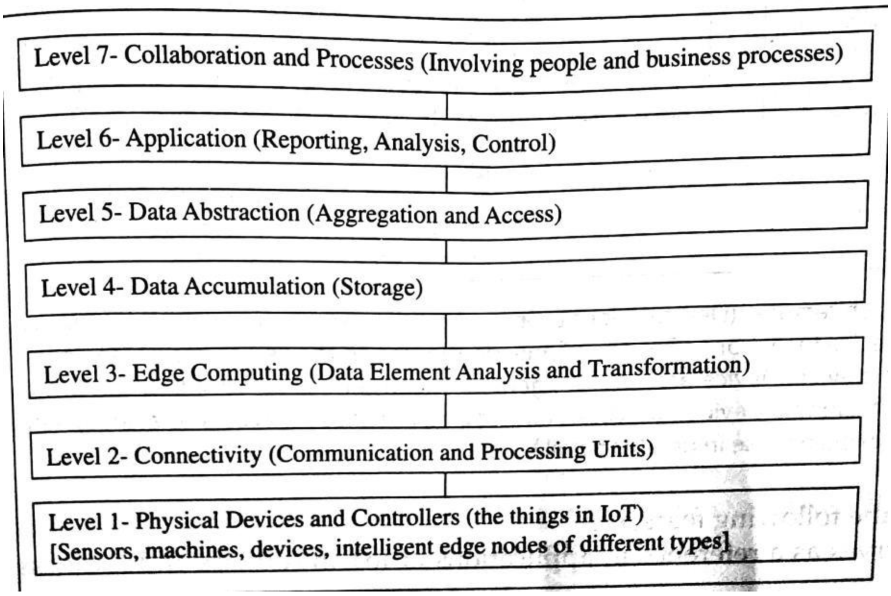
   The IoT application covers "smart" environments/spaces in domains such as: Transportation, Building, City, Lifestyle, Retail, Agriculture, Factory, Supply chain, Emergency, Healthcare, User interaction, Culture and tourism, Environment and Energy.

IoT Architectural View: [As per VTU Syllabus]

The IoT system is defined in different levels called as tiers. A model enables the conceptualisation of the framework.

A reference model can be used to depict the building blocks, successive interactions and integration.

The diagram below depicts the CISCO presentation of a reference model comprising of 7 levels and the functions of each level.



Level 7- Collaboration and Processes (Involving people and business processes)

Level 6- Application (Reporting, Analysis, Control)

Level 5- Data Abstraction (Aggregation and Access)

Level 4- Data Accumulation (Storage)

Level 3- Edge Computing (Data Element Analysis and Transformation)

Level 2- Connectivity (Communication and Processing Units)

Level 1- Physical Devices and Controllers (the things in IoT)
[Sensors, machines, devices, intelligent edge nodes of different types]

Features of the architecture:

- The architecture serves as a reference in the applications of IoT in services and business processes.
- A set of sensors which are smart, capture the data, perform necessary data element analysis and transformation as per device application framework and connect directly to a communication manager.
- The communication management subsystem consists of protocol handlers, message routers and access management.
- Data routes from gateway through the Internet and data centre to the application server or enterprise server which acquires that data.
- Organisation and analysis subsystems enable the services, business processes, enterprise integration and complex processes.

IEEE P2413

IEEE suggested P2413 standard for architecture of IoT. It is a reference architecture which builds upon the reference models. This reference model defines the relationship between various IoT Applications like Transportation and Health Care.
The characteristics of this IEEE standard are as follows:

- ❖ Follows top- down approach.
- ❖ Does not define a new architecture but reinvent existing architectures congruent with it
- ❖ Gives a blue print for data abstraction.
- ❖ Specifies abstract IoT domain for various IoT domains.
- ❖ Recommends quality ‘quadruple’ trust that includes protection, security, privacy and safety.
- ❖ Addresses the documentation of data.
- ❖ Strives for mitigating architecture divergence.

IoT Conceptual Frame Work:

Explain the concept of operation in an IoT System.

Explain the Oracle Conceptual Frame work of IoT

Explain the IBM Conceptual Frame work of IoT

An IoT System has multiple levels as seen in the basic architecture. It can be explained using the equations given below:

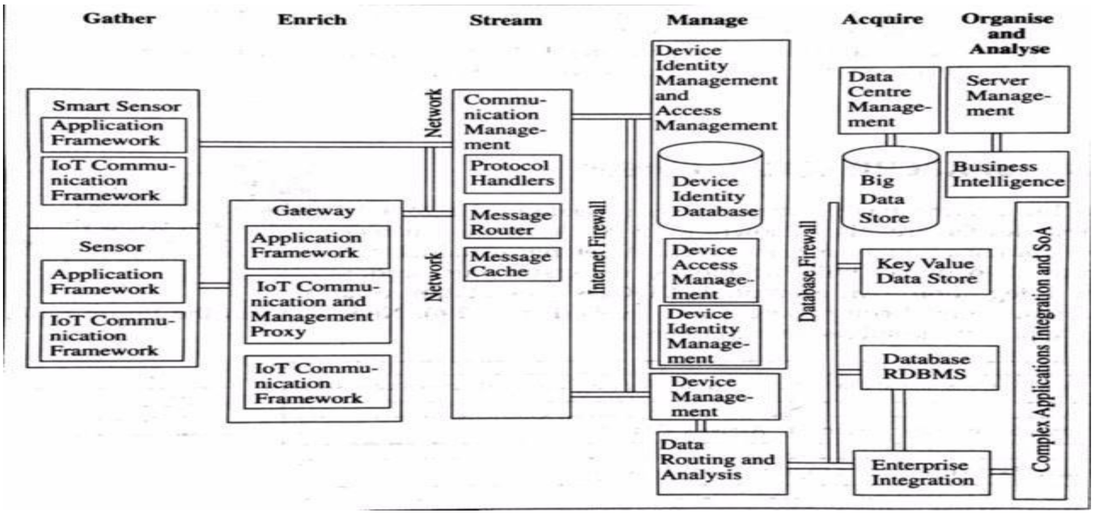
Physical Object+ Controller, Sensor & Actuators+ Internet= IoT---------(1)

IoT is an internetwork of devices and physical objects. The operation of these devices is could be to gather the information or acquire the parameter through a sensor or a controller and an actuator to serve the application.

Ex: A series of street lights communication data to the group controller which connects t the central server using the **Internet.**

Gather + Enrich + Stream + Manage + Acquire + Organise& Analyse =IoT with Connectivity to Data enter, Enterprise or Cloud ------------------(2)

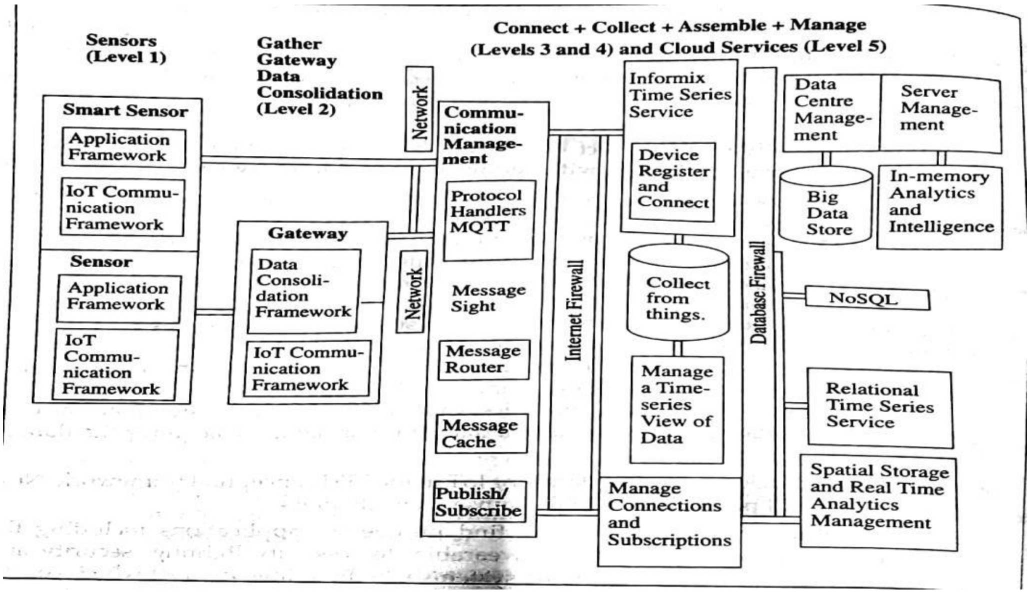
The equation (2) represents the **conceptual frame work and architecture** presented by **Oracle** as in the figure below.
The steps and processes that this architecture follows to communicate the data at different levels in IoT are:



1. Level 1: The data of the devices (things) using sensors are gathered from Internet.
2. Level 2: A sensor connected to the Gateway functions as a smart sensor. The data is then enriched-transcoding at the gateway.
3. Level 3: A communication management subsystem sends and receives the data streams.
4. Level 4: The device management, identity management and access management subsystems receive the device's data.
5. Level 5: The data store or database acquires the data.
6. Level 6: Data routed from the devices and things is organised and analysed.

Gather + Consolidate + Connect + Assemble + Manage& Analyse = IoT With Connectivity to Cloud Services-- (3)

The equation (3) presents an alternate conceptual approach for a complex system proposed by ***IBM***. The framework is as shown below.


Sensors (Level 1)
Gather Gateway Data Consolidation (Level 2)
Connect + Collect + Assemble + Manage (Levels 3 and 4) and Cloud Services (Level 5)
Smart Sensor
Application Framework
IoT Communication Framework
Sensor
Application Framework
IoT Communication Framework
Gateway
Data Consolidation Framework
IoT Communication Framework
Network
Network
Communication Management
Protocol Handlers MQTT
Message Sight
Message Router
Message Cache
Publish/ Subscribe
Internet Firewall
Informix Time Series Service
Device Register and Connect
Collect from things.
Manage a Time-series View of Data
Manage Connections and Subscriptions
Database Firewall
Data Centre Management
Server Management
Big Data Store
In-memory Analytics and Intelligence
NoSQL
Relational Time Series Service
Spatial Storage and Real Time Analytics Management

The steps for the actions and communication of data at the successive levels of IoT are as given below:

1. Levels 1 and 2 consist of a sensor network to gather and consolidate the data.
2. The gateway at level2 communicates the data streams between level 2 and 3. The system uses a communication management subsystem at level 3.
3. An information service consists of connect, collect, assemble and manage subsystems at levels 3 and 4.
4. Real time series analysis, data analytics and intelligence subsystems are at level 4 & 5. A cloud infrastructure, a data store or database acquires the data at level 5.

Various conceptual frameworks for IoT find number of applications. Ex: M2M communications, wearable devices, smart objects, smart automation of the house etc...

Smart systems use the user interfaces (UIs), Application Programming Interfaces(APIs), identification data, sensor data and communication ports to process the data and communicate it to the next level.

Technology behind IoT:

The following entities provide a diverse technology environment and are examples of technologies involved in IoT:

- Hardware: A variety of Hardware play a vital role in communicating the parameters from the IoT to the Publisher or Subscriber.
- The hardware to communicate requires an Integrated Development Environment (IDE) for developing device software, firm ware and APIs.

- Protocols are a means to effectively put the data into format. Ex: RPL, CoAP, Restful, HTTP, MQTT, XMPP.
- Communication: Media of information transfer- Power line Ethernet,RFID,NFC,6LowPAN,UWB,ZigBee,Bluetooth,WiFi,WiMAX,2G3G/4G.
- Network Backbone: IPV4, IPV6, UDP and 6LowPAN.
- Software: RIOT OS, Contiki OS, Thingsquare, Mist Firm ware, Eclipse IoT
- Internetwork Cloud platforms/ Data Centre: Sense, ThingsWorx, Nimbits
- Machine Learning Algorithm and Software

Server End Technology:

IoT Servers are application servers, enterprise servers, cloud servers, data centres and databases.

Servers offer the following components:

1. Online Platforms
2. Devices identification, identity management and their access management.
3. Data accruing aggregation, integration, organising and analysing
4. Use of web applications, services and business process.

Major Components of IoT Systems:

Major Components of IoT devices are as follows:

1. **Physical Object with embedded software into hardware** - Sensors and control units. Sensors are electronic devices that sense the physical environment. Control units commonly are the microcontroller units or a custom chip that can comprise of a processor, memory and several units which are interfaced together.
2. **Hardware** consisting of a microcontroller, firmware, sensors, control unit, actuators and communication module.
3. **Communication module**: Software consisting of device APIs and device interface for middleware for creating communication stacks using CoAP, LWM2M, IPV4, IPV6 and other protocols.
4. **Software** for actions on messages, information and commands which the devices receive and then output to the actuators, which enable actions such as glowing LEDs, robotic hand movement.

Sources of IoT:

Arduino Boards

- E.g. Arduino Yún
- Using Microcontroller ATmega32u4

- Includes Wi-Fi, Ethernet, USB port, micro-SD card slot and three reset buttons
- Runs Linux

Intel Galileo board

- A line of Arduino-certified development boards.
- Intel x86, Intel SOC X1000 Quark based System-On-Chip
- Power over Ethernet (PoE) and 6 Analog Inputs

Beagle Board

- Very low power requirement
- Card like computer, Can run Android and Linux
- Open source Hardware designs and the software for the IoT devices are

Raspberry Pi

- Wi-Fi-connected device
- Included code open source RasWIK

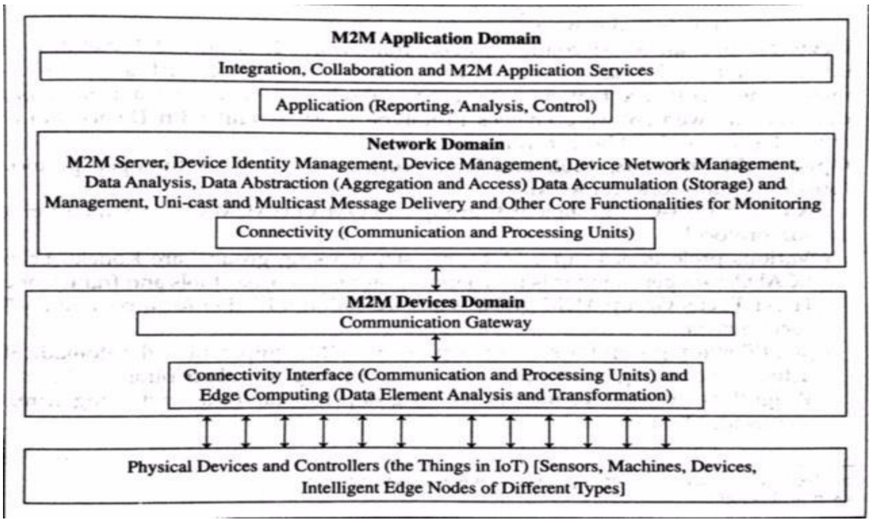
M2M Communication:

M2M refers to a process of communication of a physical object or device at machine with others of same type, mostly for monitoring and control purposes.

M2M to IoT:

- Technology closely relates to IoT which use smart devices to collect data that is transmitted via the Internet to other devices.
- Close differences lies in M2M uses for device to device communication also for coordinated monitoring and control purposes.

M2M Architecture:


M2M Application Domain
Integration, Collaboration and M2M Application Services
Application (Reporting, Analysis, Control)
Network Domain
M2M Server, Device Identity Management, Device Management, Device Network Management, Data Analysis, Data Abstraction (Aggregation and Access) Data Accumulation (Storage) and Management, Uni-cast and Multicast Message Delivery and Other Core Functionalities for Monitoring
Connectivity (Communication and Processing Units)
M2M Devices Domain
Communication Gateway
Connectivity Interface (Communication and Processing Units) and Edge Computing (Data Element Analysis and Transformation)
Physical Devices and Controllers (the Things in IoT) [Sensors, Machines, Devices, Intelligent Edge Nodes of Different Types]

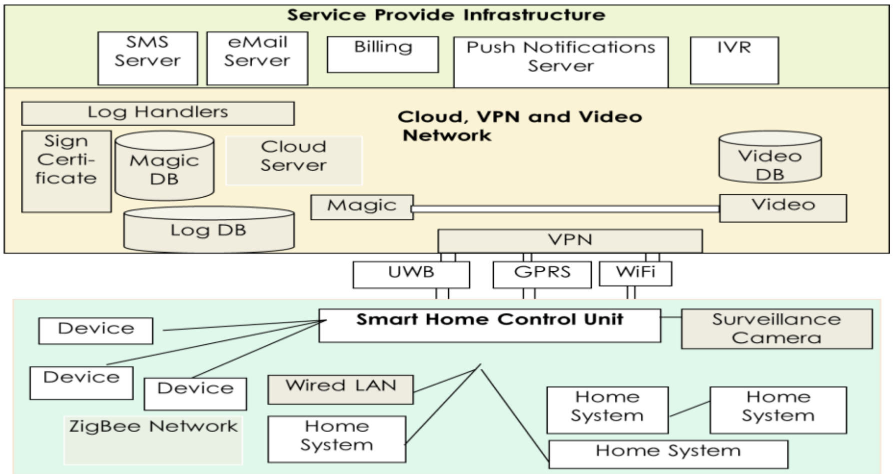
The architecture consists of three domains:

- M2M device domain
- M2M network domain
- M2M application domain

- The device management domain consists of three entities: physical devices, communication interface and gateway.
- Communication interface is a port or a subsystem which receives the input from one end sends the data received to another.
- M2M Network domain consists of M2M server, device identity management, data analytics and data & device management similar to IoT architecture level.

Examples of IoT:

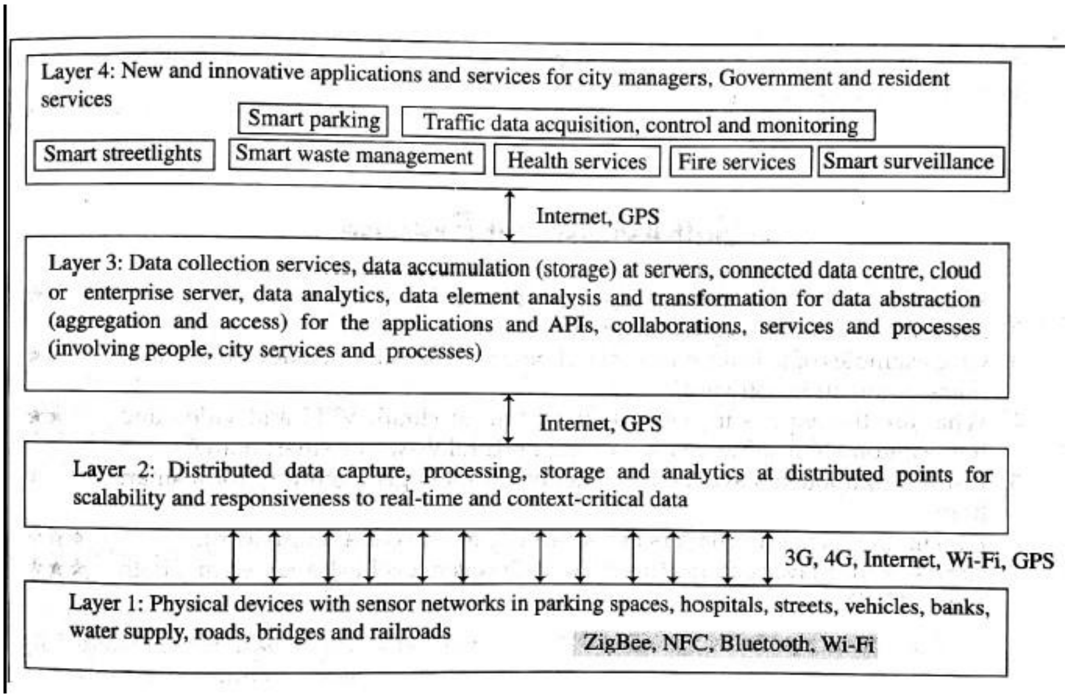
An example for Smart home automation/ Smart Home application:



- ➢ Sensors and actuators manage a smart home with an Internet connection. Wired and Wireless sensors are incorporated into the security sensors, thermostats and many more.
- ➢ In the device layer, the devices that are monitored like the temperature, lighting, power meter and so on are connected to a sensor.
- ➢ The sensor records any change in the operation of the device and communicates to the intermediate layer via UWB, GPRS or Wifi.
- ➢ Using the data is uploaded into cloud through Internet. With proper authentication the user can observe the changes at home.

- The cloud provides the information to the user by sending an email, and SMS, or Push Notifications for which the user could pay the electricity bill, telephone bill, switch off the lights or On the lights accordingly.
- This is an example of smart home automation using IoT.

An example for Smart City application:



Layer 4: New and innovative applications and services for city managers, Government and resident services

Layer 3: Data collection services, data accumulation (storage) at servers, connected data centre, cloud or enterprise server, data analytics, data element analysis and transformation for data abstraction (aggregation and access) for the applications and APIs, collaborations, services and processes (involving people, city services and processes)

Layer 2: Distributed data capture, processing, storage and analytics at distributed points for scalability and responsiveness to real-time and context-critical data

Layer 1: Physical devices with sensor networks in parking spaces, hospitals, streets, vehicles, banks, water supply, roads, bridges and railroads

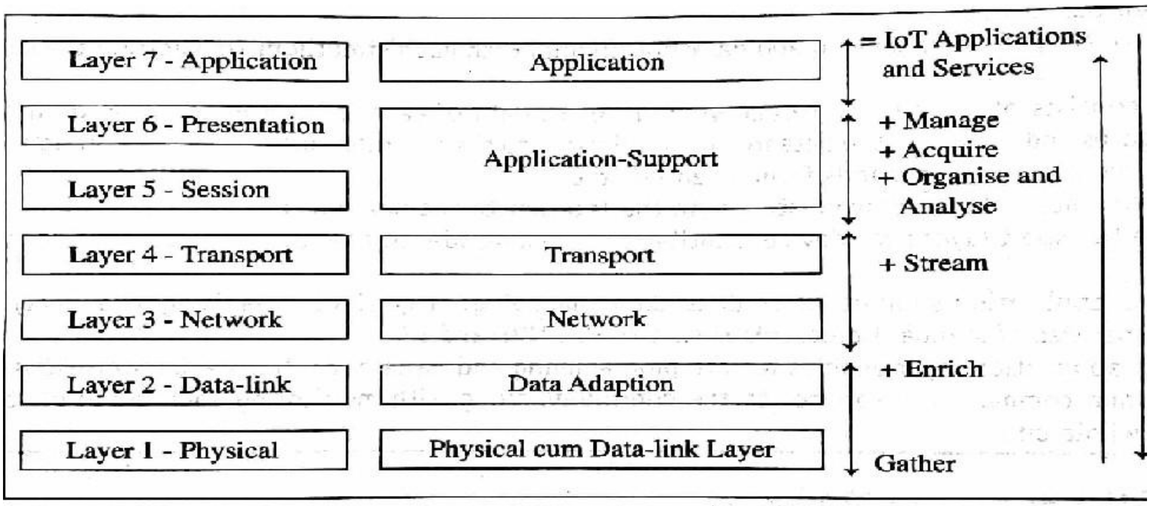
- The IoT technology can be expanded to construct a smart city.
- The feature of this application is that it connects traffic in the city to the hospitals to the schools via Internet.
- Layer 1 describes the physical device level. Here sensors are deployed in the parking space, hospitals, streets, vehicles, banks, water supply, roads, bridges and railroads.
- Layer 2: The data captured from the sensors is integrated and processed with the requirement.
- Layer 3: It is meant for central collection services, connected data centres and cloud.
- Layer 4: Consists of new innovative applications such as waste containers monitoring, WSN for power loss monitoring and to inform the concerned organisation.

Differences between IoT and M2M:

Parameters	M2M	IoT
Definition	M2M solutions contain a linear communication channel between various machines that enables them to form a work cycle. It's more of a cause and effect relation where one action triggers the other machinery into activity.	IoT can be defined as a system where multiple devices communicate with each other through sensors and digital connectivity. They talk to each other, work in tandem, and form a combined network of services.
Interactions	M2M refers to communication and interaction between machines & devices Such Interaction can occur via a cloud computing Infrastructure e.g. devices exchanging information through cloud infrastructure	IoT has broader scope than M2M, since it comprises broader range of interactions, including interactions between devices /things , things and people, things with applications and people with applications. It also enables composition of workflows comprising all of the above interactions
Interactivity	Machine to machine solutions operate by triggering responses based on an action. It's mainly a one-way communication.	The key advantage IoT has over M2M solutions is the ability to add interactivity amongst devices. In this system to and fro communication flows freely. There can be countless scenarios and combinations.
Connectivity Scope	M2M solutions rely primarily on conventional connection tools like wired connection , in wireless wifi , cellular , etc	IoT adds more sophisticated sensors into the mix. its result, Internet of Things based systems have much more flexible and varied connectivity options.
Solutions	M2M solutions, because of their limited scope, are confined to creating a network of machines that work in synchronization.	On the other hand, IoT creates 360° solutions that allow for flexible responses and multi-level communication.
Communications	Point to point communication usually embedded within hardware at customer site	Devices communicate using IP networks, incorporating with varying communication protocols
Integration	Limited integration option , as devices must have corresponding communication standards	Unlimited integration options, but requires a solution that can manage all the communications

IoT/M2M Systems, Layers and Design Standardisation:

Modified OSI Model for the IoT/ M2M Systems:



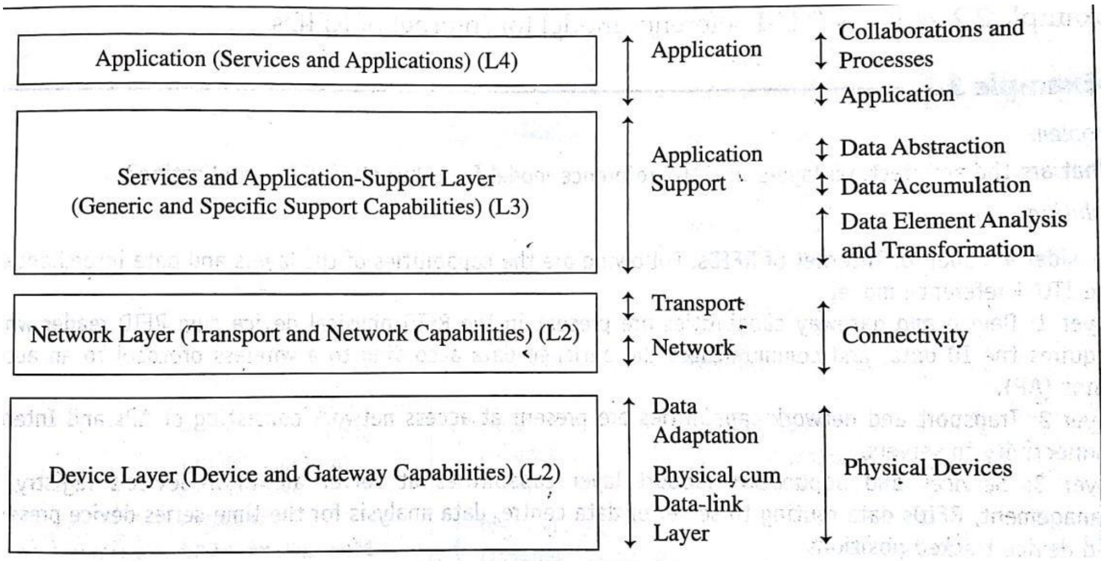
- The above diagram refers to the modified 7 layer OSI model for IoT/ M2M Systems.
- The modifications are proposed by IETF.
- Each layer proposes the received data and creates a new data stack which transfers it to the next layer.
- The processing takes place at the intermediate layers between the functional layer to the top layer.
- Device end also receives the data from the application/ service after processing.
- This shows a similarity to the operation of the equation 2 w.r.t conceptual framework as given below:
  Gather + Enrich + Stream + Manage + Acquire + Organise& Analyse =IoT with Connectivity to Data enter, Enterprise or Cloud

ITU-T Reference Model:

- The diagram below shows the ITU-T Reference Model called as RM1.
- This corresponds to the model with the six layers modified OSI model.
- Layer1: L1 is the device layer and has device and gateway capabilities.
- Layer2:L2 has transport and network capabilities.
- Layer3:L3 is the services and application-support layer. The support layer has two types f capabilities- Generic and specific service or application support capabilities.
- Top Layer: L4 is for applications and services.

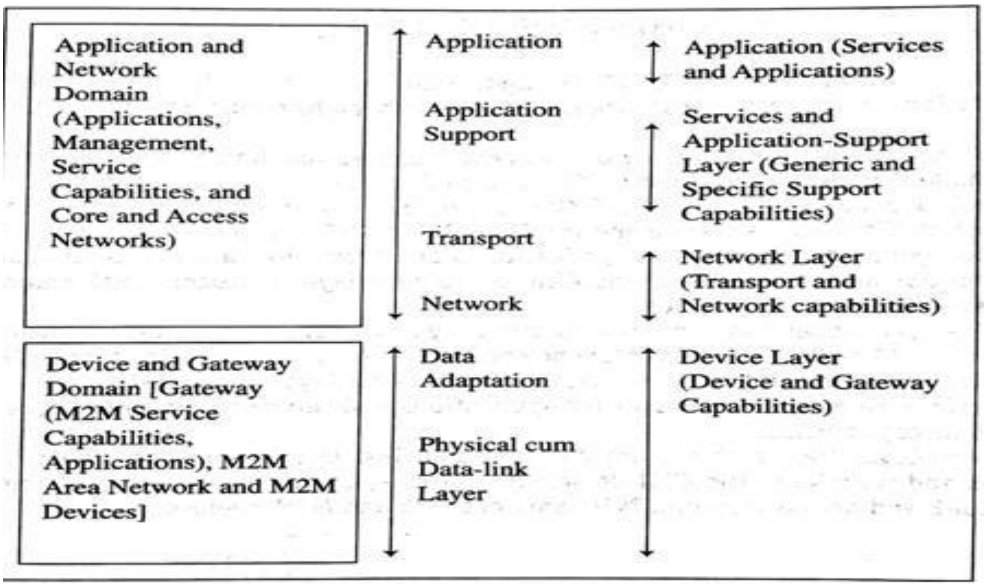
  Comparison with the CISCI IoT reference model:
- L4 capabilities are similar to the Cisco Reference Model and processes and applications are of top two levels.

- L3 functions are similar to that of the middle level function of data abstraction, accumulation, analysis and transformation.
- L2 layer capabilities are similar to the connectivity in the Cisco Model.
- L1 device layer capabilities are similar to the physical devices level.



Application (Services and Applications) (L4)
Application
Collaborations and Processes
Application
Services and Application-Support Layer (Generic and Specific Support Capabilities) (L3)
Application Support
Data Abstraction
Data Accumulation
Data Element Analysis and Transformation
Network Layer (Transport and Network Capabilities) (L2)
Transport
Network
Connectivity
Device Layer (Device and Gateway Capabilities) (L2)
Data Adaptation
Physical cum Data-link Layer
Physical Devices

ETSI M2M Domain Architecture



Application and Network Domain (Applications, Management, Service Capabilities, and Core and Access Networks)
Application
Application Support
Transport
Network
Application (Services and Applications)
Services and Application-Support Layer (Generic and Specific Support Capabilities)
Network Layer (Transport and Network capabilities)
Device and Gateway Domain [Gateway (M2M Service Capabilities, Applications), M2M Area Network and M2M Devices]
Data Adaptation
Physical cum Data-link Layer
Device Layer (Device and Gateway Capabilities)

- Like ITU-T , ETSI specifies the functional areas, a high level architecture and reference model for communicating the data from and to the IoT/ M2M devices
- The above diagram shows the ETSI M2M domains and architecture and the high level capabilities of each domain.
- It also depicts the architectural correspondence with the 6 layer modified OSI model and 4 layer of the ITU-T Reference model.
- The ETSI Network Domain has 6 capabilities and functions:
    - M2M Applications

    - ❖ M2M Service Capabilities
    - ❖ M2M Management functions
    - ❖ Network Management Functions
    - ❖ CoRE network ex: 3G and IP Networks
    - ❖ Access network ,WLAN and Wi Max.
- ➢ ETSI device and gateway have the following functional units:
    - ❖ Gateway between M2M area network, CoRE and access network , processing M2M service capabilities.
    - ❖ M2M area network(Bluetooth, ZigBee, NFC, PAN,LAN)
    - ❖ M2M Devices

- **Explain M2M ETSI domains and high level architecture for applications and services ATMs to bank servers.**
- **What are the architecture layers in ITU-T reference model for Internet of RFIDs application?**
- **What are the architectural layers in IoT? List the applications and advantages of IoT.**

Data Enrichment, Data Consolidation and Device Management at Gateway:

- ❖ A gateway at the data adaptation layer has several functions.
- ❖ These are data privacy, data security, data enrichment, data consolidation, transformation and device management.

Data Management and Consolidation Gateway:

- ➢ Gateway includes the following functions:
    - ❖ Transcoding
    - ❖ Privacy, Security
    - ❖ Integration
    - ❖ Compaction and fusion
- ➢ **Transcoding:** It means conversion and change of protocol, format or code using software.
- ➢ The gateway renders the web response the web and messages in formats and representations required and acceptable at an IoT device.
- ➢ IoT device requests are adapted, converted and changed into required formats acceptable at the server by the transcoding software.
- ➢ Ex. conversion from ASCII to Unicode at the server.
- ➢ A transcoding proxy can execute itself on the client system or the application server.
- ➢ It has conversional, computational and analysing capabilities while the gateway has conversion and computational capabilities only.

- **Privacy:** The data such as medical records, logistics, and inventories of a company may need privacy and protection.
- The following are the components of privacy model:
  - Devices and applications identity management
  - Authentication
  - Authorization
  - Trust
  - Reputation
- A suitable encryption method ensures data privacy.
- The data is decrypted and analysed and is an input to the application service or process.
- **Secure data access:** Access to data needs to be secured. The design needs to ensure the authentication of a request from a service or application.
- End to end security is a feature which uses a security protocol at each layer.
- **Data gathering and Enrichment:** IoT applications involve actions such as Data gathering(Acquisition), Validation, Storage ,Processing, Retention and analysis.
- Data gathering is to acquire the data from the devices or device networks. Four modes of Acquisition are:
  - **Polling:** Refers to the data sought by addressing the device[Its operated like the polling by a computer to access the control of a channel to transfer data or to check if there is a data addressed to it]
  - **Event Based:** The data acquired from the device on an event like a NFC or a card reader.
  - **Scheduled Interval:** The data acquired from the device at selected intervals. Ex: changes in the lighting condition of street lights.
  - **Continuous Monitoring:** Refers to the data sought from the device continuously. Ex: Data for traffic monitoring.
- **Data Dissemination: (Dissemination means to distribute, broadcast, diffuse or spread)**

  There are three steps in the data enrichment before data dissemination
  - **Aggregation:** Refers to the process of joining together present and previously received data frames after removing redundant or duplicate data.
  - **Compaction:** means making information short without changing the meaning or context; ex. transmitting only the incremental value of the data so that the information is short.
  - **Fusion:** formatting the information received in parts through various data frames and several types of data, removing redundancy in the received data.

- When the data transmission takes place in the wireless environment the energy dissipation or power consumption is a criteria. This is due to the battery life in the WSNs.
- Energy efficient computations can be made use of by using the concepts of data aggregation, compaction and fusion.
- **Data Source and Data Destination: ID:** Each device and resource is assigned an ID for specifying the data of source and a separate ID for data destination.
- **Address:** Header fields add the destination address.
- **Data Characteristics, Formats and structures:** Data characteristics can be in terms of temporal data i.e. dependent on time, Spatial Data i.e. dependent on location, real time data i.e. generated continuously and acquired continuously at the same pace, real world data i.e. ex: traffic or streetlight, Proprietary data i.e. data reserved with copy rights to authorised enterprises and Big Data i.e. unstructured voluminous data.
- Data received from the devices can be in different formats for further communication like: XML, JSON, TLV. The structure implies the ways for arranging the data bytes in sequences with size limit.
- **Device Management (DM) at gateway:** DM means provisioning the device ID or address which is distinct from other sources, device activating, configuring, registration, deregistering, attaching and detaching.
- DM also means accepting subscription for its resources.
- **Open Mobile alliance** (OMA)-DM and several standards for device management.
- OMA-DM model suggests the use of a DM server which interacts with devices through a gateway in case of IoT/M2M application.
- Gateway functions for device management are:
    - Forwarding the data/ request when the DM server and device interact without structuring.
    - Protocol conversion when the device and DM server use distinct protocols.
    - Proxy functions in case of intermediate pre fetch is required in a lossy environment or network environment needs.

Web communication protocols used by connected IoT/M2M devices

- An IoT/M2M device network gateway needs connectivity to web services.
- A communication gateway enables web connectivity, while IoT/M2M methods and protocols enable the web connectivity for a connected device network.

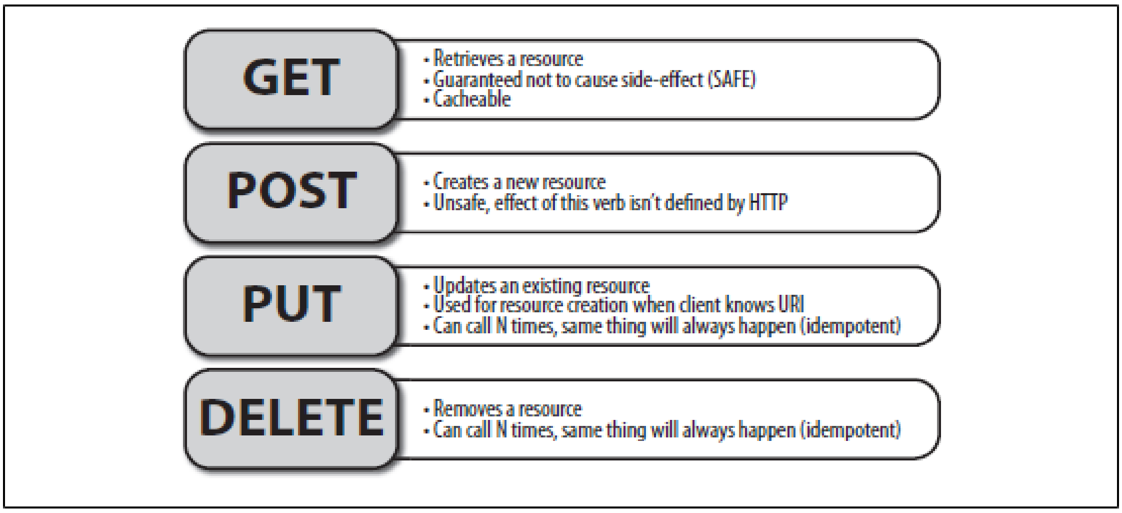
- Following are the key terms used in communication:
  - **Application or APP**: refers to software for applications for creating and sending an SMS, measuring and sending the measured data.
  - **Application Programming Interface(API):** refers to a software component, which receives messages from one end. It might consist of GUIs (Button, Checkbox and Dialog Box).
  - **Web service:** refers to a servicing software which web protocols, web objects utilize ex. Weather reports, traffic density.
  - **Object:** refers to collection of resources.
  - **Object Model:** defined as the usage of objects for values, messages, data or resource transfer, and creation of one or more object instances.
  - **Class:** It creates one or more instances.
  - **Communication Gateway:** functions as a communication protocol translator.
  - **Client:** refers to a software object which makes request for data, messages, resources or objects.
  - **Server:** is defining as software which sends a response on a request.
  - **Web Object:** That retrieves a resource from the web object at other end using a web protocol.
  - **Broker:** denotes an object which arranges the communication between two end point devices.
  - **Proxy:** an application which receives a response from the server for usage of the client or application and which also requests from the client for the responses retrieved or saved at proxy.
  - **Communication protocol:** defines the rules and conventions for communication between the web server and web clients.
  - **Web protocol:** that defines the rules and conventions for communication between the web server and web clients. It is a protocol for web connectivity of web objects, clients, servers and intermediate servers or firewalls.
  - **Firewall:** is one that protects the server from unauthentic resources.
  - **Universal Resource Locator:** is generally used for retrieving resources by a client.
  - **Representational State Transfer (REST):** is a software architecture referring to ways of defining the identifiers for the resources, methods, access methods and data transfer during interactions.
  - REST also refers to usage of defined resource types when transferring the objects between two ends-URIs or URLs for representations of the resource.

- ❖ REST also refers to the usage of use verbs(commands), POST, GET, PUT and DELETE.
- ❖ RESTful refers to one which follows REST constraints and characteristics.

Web Communication Protocols for Connected Devices:

REST-Representation State Transfer

- ➢ REST (Representational State Transfer) is an architectural style for developing web services.
- ➢ REST is popular due to its simplicity and the fact that it builds upon existing systems and features of the internet's HTTP in order to achieve its objectives, as opposed to creating new standards, frameworks and technologies.
- ➢ It is implemented by using the following to fetch, maintain, enrich, update and append the data.

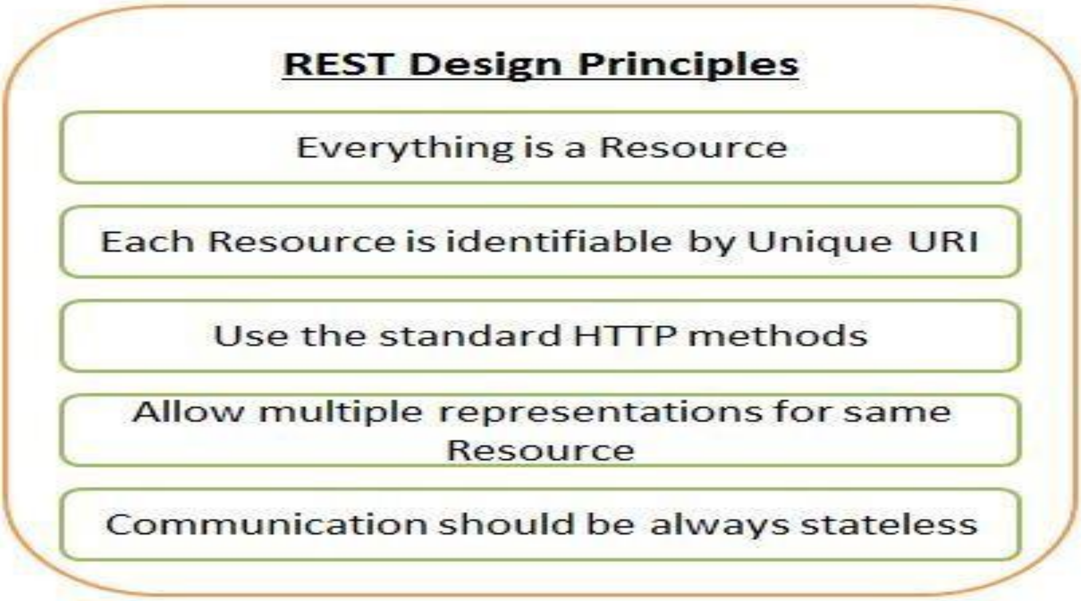


GET
- Retrieves a resource
- Guaranteed not to cause side-effect (SAFE)
- Cacheable

POST
- Creates a new resource
- Unsafe, effect of this verb isn't defined by HTTP

PUT
- Updates an existing resource
- Used for resource creation when client knows URI
- Can call N times, same thing will always happen (idempotent)

DELETE
- Removes a resource
- Can call N times, same thing will always happen (idempotent)



REST Design Principles

Everything is a Resource

Each Resource is identifiable by Unique URI

Use the standard HTTP methods

Allow multiple representations for same Resource

Communication should be always stateless

CoRE: Constrained RESTful Environment:

IoT devices or M2M devices communicate between themselves in a Local Area Network

```
 _________                                                  __________
|         |                                                |          |
| Thing  (C)-----------------------------------------------(S)  Origin |
|        (S)                                               |   Server |
|_________| \                                              |__________|
 (Sensor)    \    __________                        (Resource Directory)
              \ |          |
               (C)  Thing  |
                |__________|
               (Controller)
```

- ➢ Nodes in IoT systems often implement both roles.
- ➢ Unlike intermediaries, however, they can take the initiative as a client (e.g., to register with a directory, such as CoRE Resource Directory or to interact with another thing) and act as origin server at the same time (e.g., to serve sensor values or provide an actuator interface).

Features:

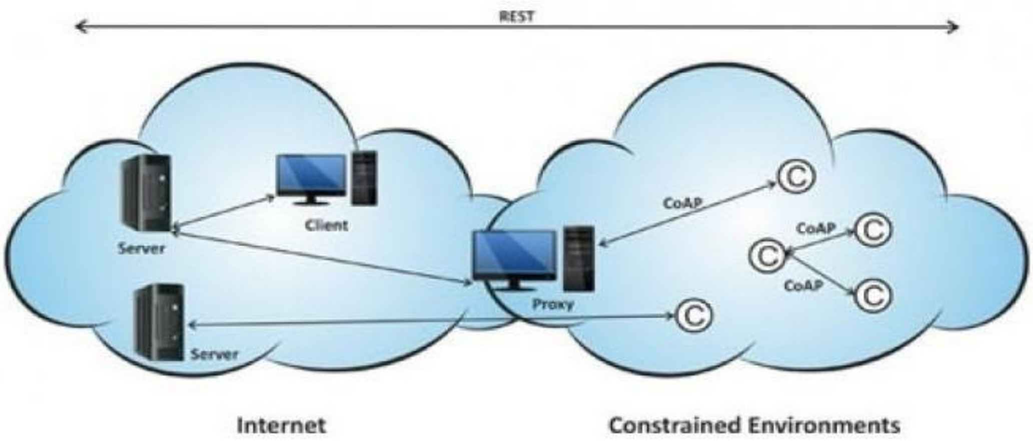
- ➢ Devices have a constraint in the sense that their data is limited in size compared to when data interchange between web clients and web servers takes place using HTTP, TCP and IP.
- ➢ Data Routing is another constraint when **Routing Over a Network of Low Power and (data) Loss- ROLL**.
- ➢ ROLL network is a low power wireless network.
- ➢ The devices may sleep most of the time in a low power environment and awaken on an event or when required.

Unconstrained Environment:

- ➢ Web applications use HTTP and RESTful HTTP for web client and web server communication.
- ➢ A web object consists of 1000s of bytes.
- ➢ Data routes over IP networks for the Internet.

Constrained Application Protocol

- ➢ Constrained Application Protocol (aka CoAP) is a specialized web transfer protocol for use with constrained nodes (low power sensors and actuators) and constrained networks (low power, lossy network).
- ➢ It enables those nodes to be able to talk with other constrained nodes over Internet.
- ➢ The protocol is specifically designed for M2M applications such as smart energy, home automation and many Industrial applications.



- CoAP protocol is necessary because traditional protocols such as TCP/IP are considered “too heavy” for IoT applications that involves constrained devices.
- CoAP protocol runs on devices that support UDP protocol. In UDP protocol, client and server communicate through connectionless datagrams.
- As it is a web transfer protocol, it is based on RESTful architecture which provides a request/response interaction model between application endpoints and supports built-in discovery of services and resources.
- Like HTTP, Servers make resources under URL and clients access those resources using methods such as GET, PUT, POST and DELETE.

The CoAP protocol has the following features

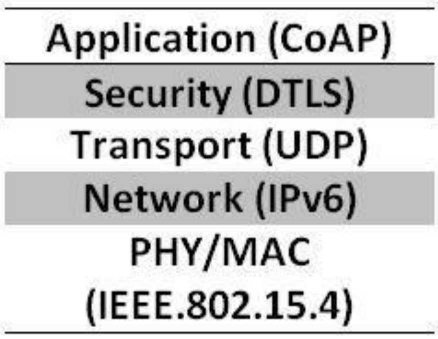
- It provides M2M communication in constrained environment.
- It provides security of data by datagram transportation layer security (DTLS).
- Asynchronous message exchange.
- Low header overhead and parsing complexity
- URI and content type support
- UDP binding with optional reliability supporting unicast and multicast requests.

- The CoAP is different from other protocols.
- When compared with HTTP, CoAP is implemented for IoT and M2M environment to send messages over UDP protocol.
- To compensate for the unreliability of UDP protocol, CoAP defines a retransmission mechanism and provides resource discovery mechanism with resource description.

CoAP should be on priority for the following three factors

  - Quality of service with confirmable message
  - When multicast support is needed
  - Very low overhead and simplicity.

- CoAP follows a client-server communication model.
- Client makes request to the server and the server sends back the responses to the client.
- Client can GET, PUT, POST or DELETE the resources on network.
- CoAP improves the HTTP request model with the ability to observe a resource.
- In HTTP, the server needs to do polling again and again to check where there is any state changes to the client or not.
- Whereas in CoAP, the observe flag is set on the CoAP GET request, the server continues to reply after the initial document has been transferred.
- This allows servers to stream the state changes to clients as they occur. Any end can stop the observation.
- The CoAP defines a standard mechanism for resource discovery.
- Servers provide a list of their resources, along with metadata about them, at /.well-known/core. For Quality of Service (QoS), Requests and response messages may be marked as confirmable or non-confirmable.
- Confirmable messages must be acknowledged by the receiver. Non-confirmable messages are “fire and forget” type.



Application (CoAP)
Security (DTLS)
Transport (UDP)
Network (IPv6)
PHY/MAC
(IEEE.802.15.4)

CoAP Protocol at Application Layer in Network Architecture

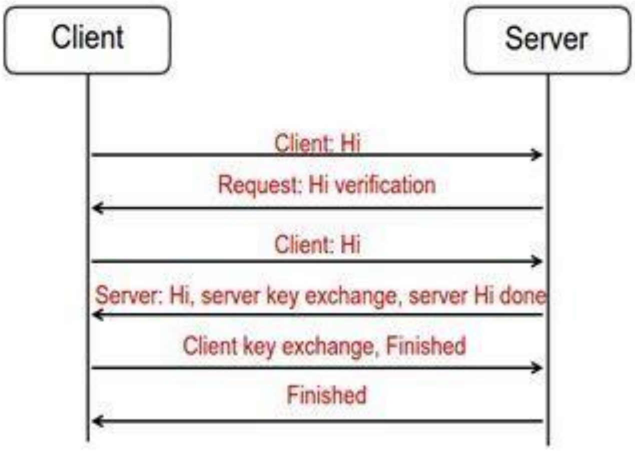
CoAP Protocol Security

- The main concern from security point of view is to provide Data Integrity, Data Authentication and Data Confidentiality.
- The CoAP provides security over Datagram Transportation Layer Security in Application layer.
- As CoAP runs over UDP protocol stack, there are chances of data loss or data disordering. But with DTLS security, these two problems can be solved.

DTLS security adds three implementations to CoAP

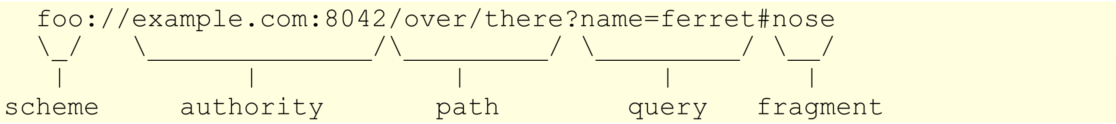
1) Packet retransmission
2) Assigning sequence number within handshake
3) Replay detection

- The security is designed to prevent eavesdropping, tampering or data forgery at any cost.
- Unlike network layer security protocols, DTLS in application layer protect end-to-end communication.
- DTLS also avoids cryptographic overhead problems that occur in lower layer security protocols.
- There is a Secured Handshake Mechanism in DTLS as shown in image below


Client
Server
Client: Hi
Request: Hi verification
Client: Hi
Server: Hi, server key exchange, server Hi done
Client key exchange, Finished
Finished

DTLS Secured Handshake Mechanism for CoAP

- The CoAP can also be implemented over TCP and over TLS.
- Check out the following official documentation for CoAP implementation over TCP and TLS.
- An important part of RESTful API design is to model the system as a set of resources whose state can be retrieved and/or modified and where resources can be potentially also created and/or deleted.
- Uniform Resource Identifiers (URIs) are used to indicate a resource for interaction, to reference a resource from another resource, to advertise or bookmark a resource, or to index a resource by search engines.



```
  foo://example.com:8042/over/there?name=ferret#nose
  \_/   \______________/\_________/ \_________/ \__/
   |           |            |            |        |
scheme     authority       path        query   fragment
```

CoAP SMS

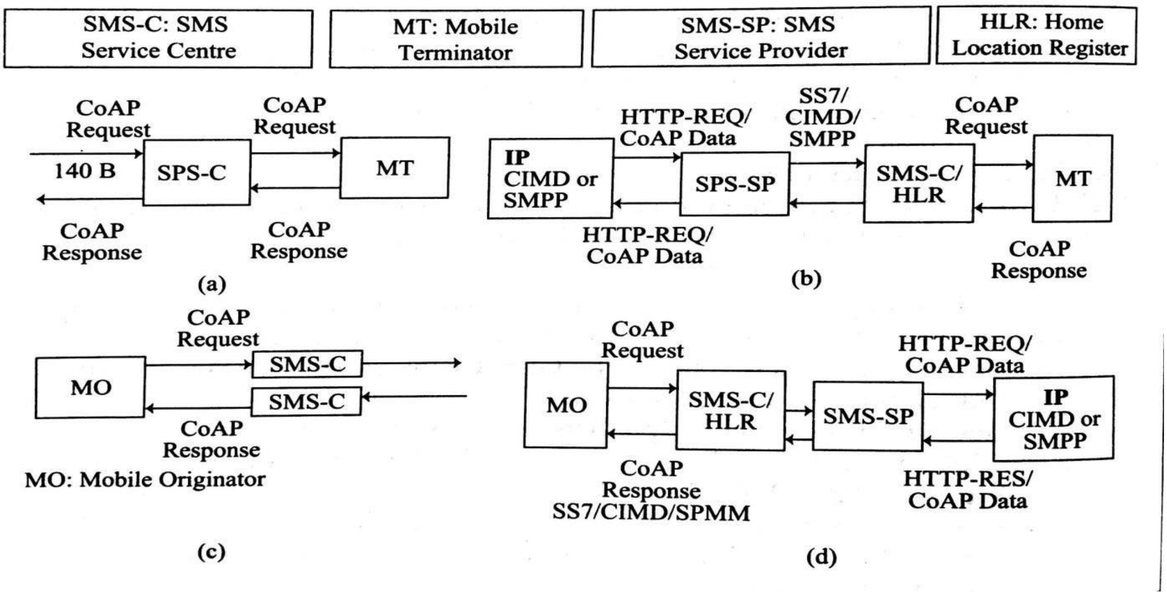
- ❖ Is a protocol when CoAP object uses IP with networks and uses SMS.
- ❖ SMS is used instead of UDP+DTLS by CoAP client server.
- ❖ Client communicates to a mobile terminal(MT) endpoint over GPRS,HSPA or LTE using CoAP-SMS protocol.

CoAP-SMS features:

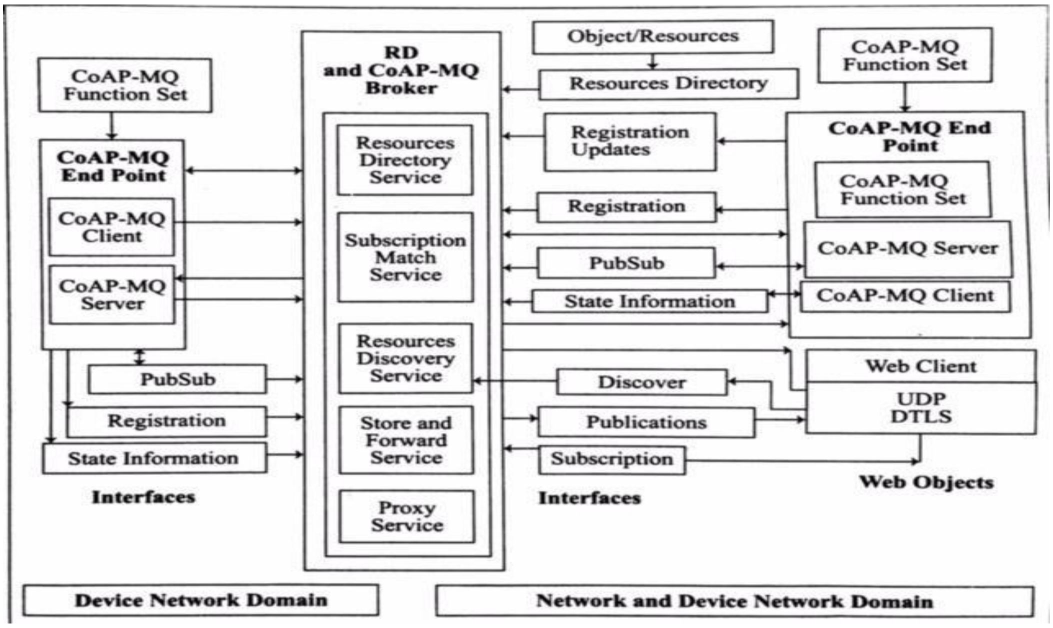
- ❖ An URI(Universal Resource Identifier) is used to send specified telephone number.

  Example: coap+sms://telNum/…….
- ❖ CoAP msg consists of 160 character in 7-bits/8 bits
- ❖ CoAP works with SIM(subscriber Identity Module) for SMS in cellular networks.
- ❖ Does not support multi-casting
- ❖ Two options are availble RUH(Response to URI-Host) and RUP(Response to URI-Port) for initiating CoAP client to know about the alternative interface are CIMP and SMPP
- ❖ MSISDN and SIM based security is used during SMs data exchange.

- ➢ CoAP request or response communication to a machine, IoT device or mobile terminal (MT) fig(a).
- ➢ A computer or machine interface using IP communication to a mobile service provider for data interchange with terminal fig(b)
- ➢ A machine or IoT device or mobile origin (MO) communication of CoAP request or response communication fig(c)
- ➢ An origin communication using SS7/CIMD/SMPP with a computer or machine interface using IP communication.



CoAP MQ:



CoAP-MQ feature

- It is message queue protocol.
- CoAP provides resource-subcription, from publishers.
- The device objects communicate using the CoAP client and server protocol and CoAP web object using DTLS as security protocol
- UDP for CoAP APIs.

Lightweight Machine to machine Communication Protocol:

- It is a Communication protocol at the application layer.
- Specified by OMA-Open Mobile Alliance for transfer of data/ message.
- Why Light Weight Management: It is widely used for mobile devices, low cost remote management and service enabled mechanism that works on wireless connection.
- It provides data management as well as application data handling.

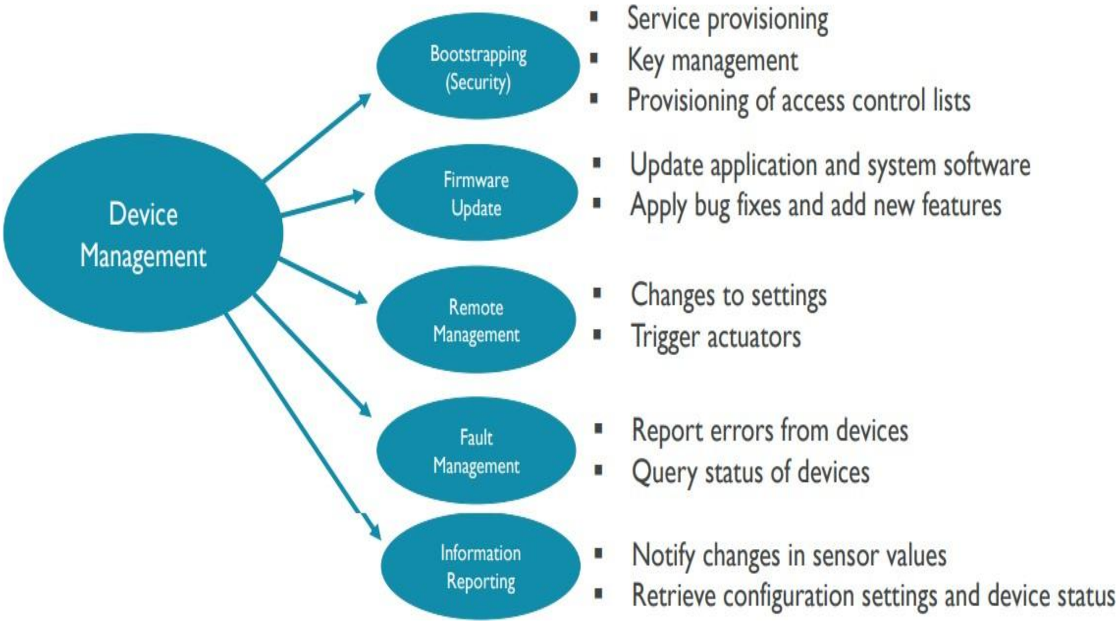
Features:

- An object or resource use CoAP, DTLS and UDP or SMS protocols for sending a request or response.
- Use of plain text for a resource or use of JSON during a single data transferor binary TLV format data transfer.
- An object or its resource access using an URI.
- It uses 3 types of Interface functions:
  - Bootstrapping

- ❖ Registration
- ❖ Report

Advantages:

- ❖ Enables plug and play solution between an increasing variety of M2M
- ❖ Enables independent innovation of M2M applications and M2M platforms



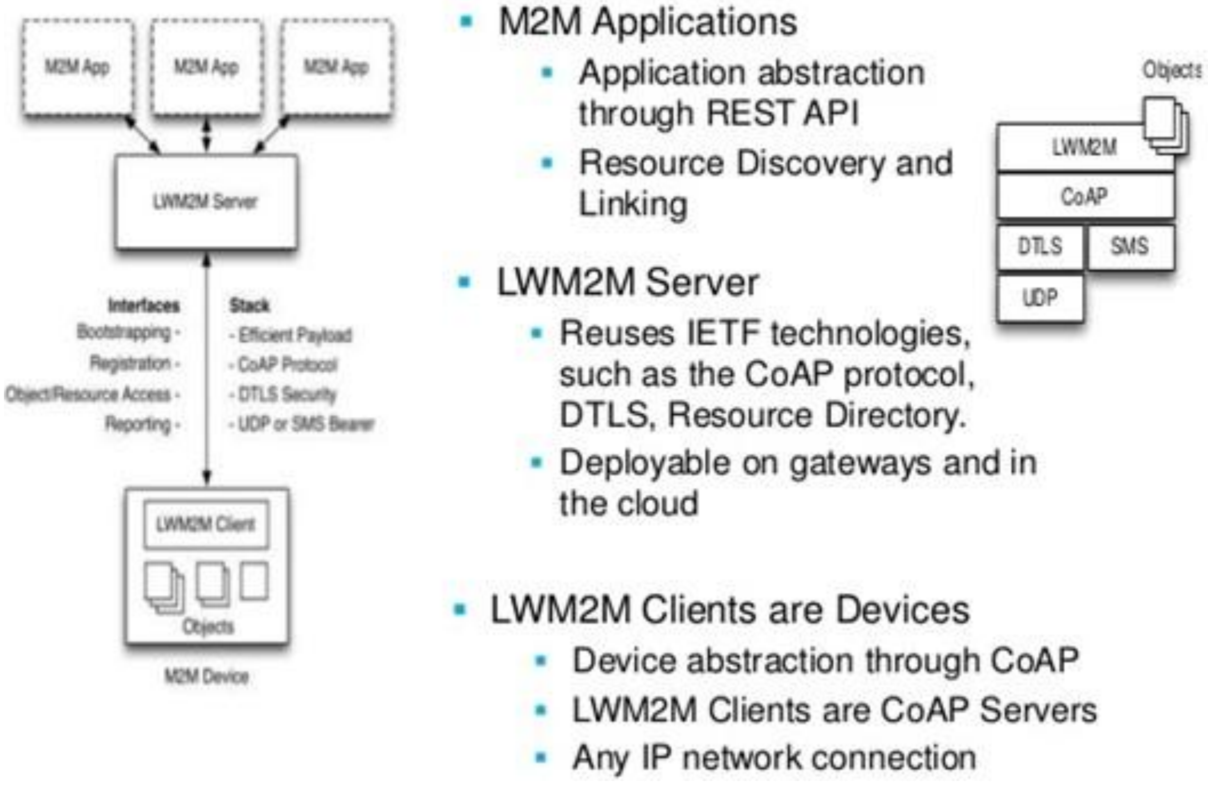
- Service provisioning
- Key management
- Provisioning of access control lists

- Update application and system software
- Apply bug fixes and add new features

- Changes to settings
- Trigger actuators

- Report errors from devices
- Query status of devices

- Notify changes in sensor values
- Retrieve configuration settings and device status



- M2M Applications
  - Application abstraction through REST API
  - Resource Discovery and Linking
- LWM2M Server
  - Reuses IETF technologies, such as the CoAP protocol, DTLS, Resource Directory.
  - Deployable on gateways and in the cloud
- LWM2M Clients are Devices
  - Device abstraction through CoAP
  - LWM2M Clients are CoAP Servers
  - Any IP network connection

MQTT- Message Queue Telemetry Transport:

- An open source protocol for machine-to-machine (M2M)/"Internet of Things" connectivity
- Created by IBM
- The objects communicating using the Connected devices network protocols, such as ZigBee.
- Web objects also using MQTT library functions and communicate using IP network and SSL and TLS security protocols

MQTT Features

- Constrained environment protocol.
- PubSub messaging architecture in place of request-response client-server architecture
- publisher (message sender at the device domain or web object at network and application domain) sending the messages on a topic.
- Subscriber (message receiver at the device domain or web object at network and application domain) receiving the messages on a subscribed topic
- Lightweight, running on limited resources of processor and memory processor or memory resources
- Header of fixed-length header and two bytes only
- M2Mqtt library providing a set of functions for coding
- M2Mqtt library functions in Java needing just 100 kB and in C# is 30 kB,
- Minimum number of exchanges, and therefore lessening the network traffic
- Three Quality of Services
- MQTT TCP/IP Connectivity
- Broker-based publish/subscribe messaging protocol
- publish/subscribe functions enable one-to-many message distribution decoupled with the applications (unconcerned about the payload)
- Notifying on an abnormal disconnection of a client, notified all nodes subscribing to the message
- The last will specifying the final action to be taken on failure to send the messages.

MQTT Broker Functions

- Store and forward
- Clients publish topics and receives topics on subscription
- Recovers subscriptions on reconnect after a disconnection, unless client explicitly disconnected
- Acts as a broker between publisher of the topics and subscribers of the topics
- Finds client disconnection until DISCONNET message receives, keeps message alive till explicit disconnection
- retains the last received message from a publisher for a new connected subscriber on same topic, when retain field in the header is set.

XMPP (Extensible Messaging and Presence Protocol)

- XMPP is the Extensible Messaging and Presence Protocol, a set of open technologies for **instant messaging,** presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data.
- XMPP was originally developed in the Jabber open-source community to provide an **open, decentralized alternative to the closed instant messaging services** at that time.
- XMPP offers several key advantages over such services:
  - **Open** — the XMPP protocols are free, open, public, and easily understandable; in addition, multiple implementations exist in the form clients, servers, server components, and code libraries.
  - **Standard** — the Internet Engineering Task Force (IETF) has formalized the core XML streaming protocols as an approved instant messaging and presence technology.
  - **Decentralized** — the architecture of the XMPP network is similar to email; as a result, anyone can run their own XMPP server, enabling individuals and organizations to take control of their communications experience.
  - **Secure** — any XMPP server may be isolated from the public network (e.g., on a company intranet)
  - **Extensible** — using the power of XML, anyone can build custom functionality on top of the core protocols; to maintain interoperability, common extensions.
  - **Flexible** — XMPP applications beyond IM include network management, content syndication, collaboration tools, file sharing, gaming, remote systems monitoring, web services, lightweight middleware, cloud computing, and much more.
  - **Diverse** — a wide range of companies and open-source projects use XMPP to build and deploy real-time applications and services; you will never get “locked in” when you use XMPP technologies.

MODULE 2 CHAPTER-1 ARCHITECTURE AND DESIGN PRINCIPLES FOR IOT.

Nischitha.S
Asst. Professor

• INTERNET CONNECTIVITY

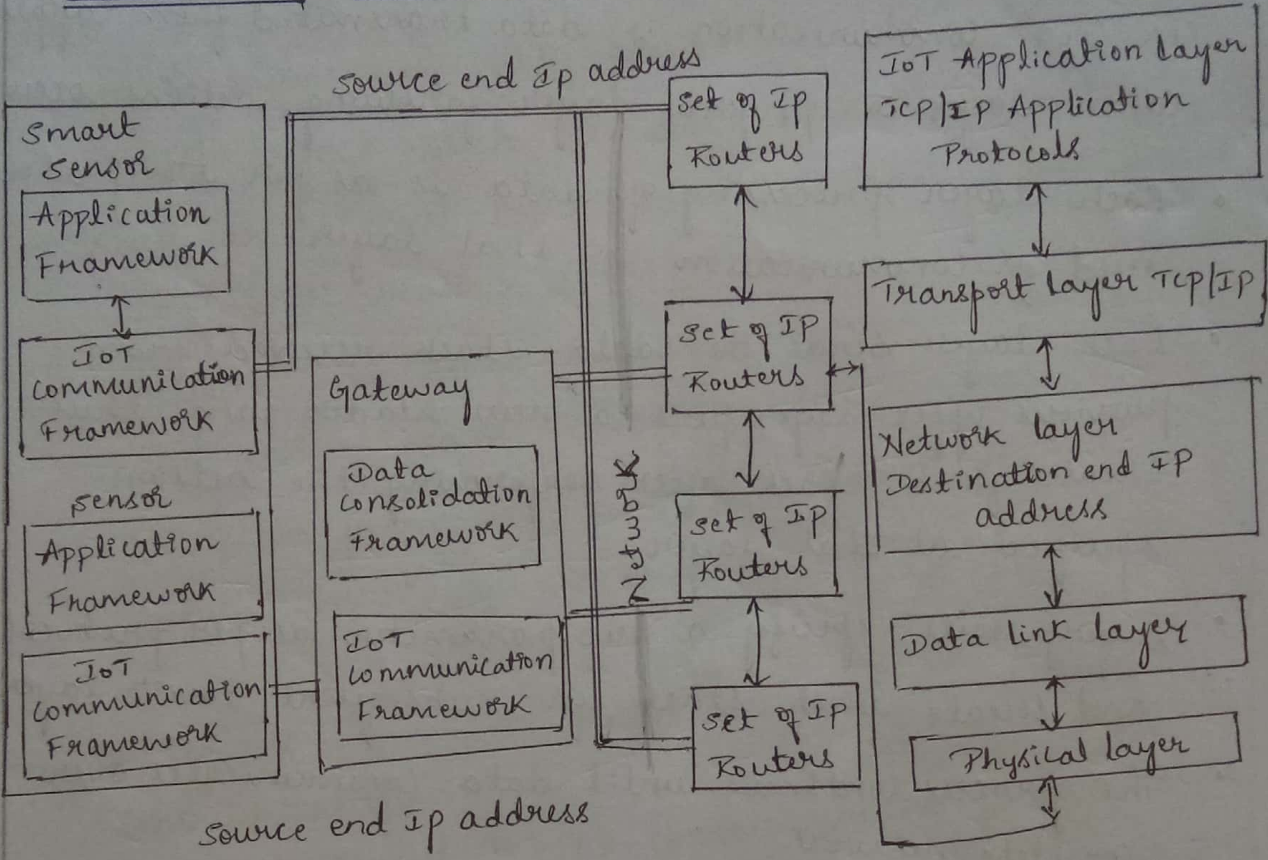


Fig:- source end N/w layer connected through a set of Ip Routers for data packets from an Ip address and communicating with IoT/M2M IoT application and service layer using TCP/IP suite of application protocols.

→ Figure shows that source end network layer connected to the destination through a set of IP routers

→ The communication framework uses an IP address and communicates with the IoT/M2M IoT application and services layer using TCP/IP suite of application protocol to destination IP address.

→ Internet connectivity is through a set of routers in global network of routers which carry data packets as per IP protocol from a source end to destination and vice versa. using IETF standard. exchanges is done.

INTERNET BASED COMMUNICATION

→ When data transmits from layer$_i$ to next layer$_j$ [$i>j$] In IoT communication if data transmitted from application layer to physical layer following actions occur

- Each layer processing of data is as per the protocol used for communication by that layer
- Each layer sends the data stack received from previous upper layer plus a new header and thus creates fresh stack after performing the actions specified at that layer
- Layer$_j$ will specify a new parameter as per protocol and create fresh stack for subsequent lower layer
- The process continues until data communicates over complete network

→ When data received at next layer$_i$ from a layer$_j$ that is IoT device physical layer to IoT application following actions are performed

- Each layer performs the processing as per header field bits which are received according to protocol to be used for decoding the fields for required actions at that layer
- Each layer receives data stack from previous lower layer and after the required actions it re ves the header words and creates a new stack specified for next higher layer

- The process continues until the data is received at the port on the highest application layer.

Note:-

→ upper layers uses only header words.

→ Lower layers such as data link layer protocol, such as Ethurnet 802.3 provides for tailing bits inclusion along with header words

→ Trailing bits usage can be as error - control bits and end of frame indicating bits.

→ Maximum frame size used by Ethurnet is 1518 B which consists of 32 trailing bits.

↳ 4 layers from OSI model is used for Internet communication

Internet based TCP/IP communication protocol uses application layer L7, transport L4, Internet L3 and link L2 layers.

Figure shows the protocol data units [PDU] at the layers.

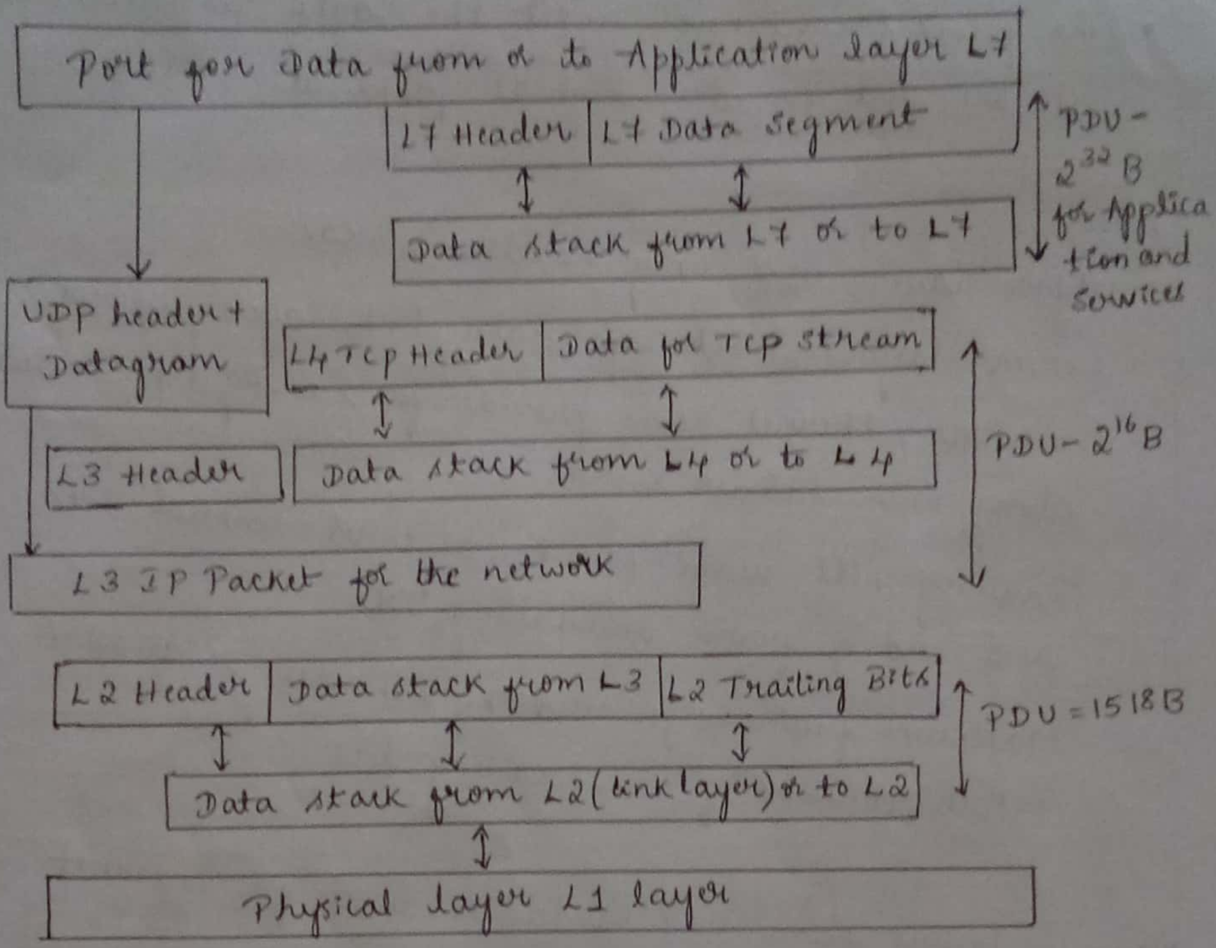

Port for Data from or to Application layer L7
L7 Header
L7 Data Segment
Data stack from L7 or to L7
PDU - 2^{32} B for Application and Services
UDP header + Datagram
L4 TCP Header
Data for TCP stream
L3 Header
Data stack from L4 or to L4
PDU - 2^{16} B
L3 IP Packet for the network
L2 Header
Data stack from L3
L2 Trailing Bits
PDU = 1518 B
Data stack from L2 (link layer) or to L2
Physical layer L1 layer

Fig:- TCP/IP suite four layers generating data stack for network and for physical layer during Internet communication

→ If layer 4, transport layer accepts a data segment (maximum of 2^{32}B per segment) from layer 7 and generates TCP stream

↳ L3 converts the stream into packets

→ If layer 4, accepts a data segment (maximum of 2^{16}B per segment from layer 7.

If layer4 is used for UDP usage it generates a UDP datagram (maximum 2^{16} B). L3 converts stream into packets [maximum of 2^{16} B including header]

→ Internet layer uses Ip protocol. Packet routing is done by routers.
Each router has information about path to destination
When a number of paths are there to reach destination
each packet can take different paths, the destination
reassembles the packets and deliver it to IOT
application layer

→ Data link layer uses a protocol each in its sublayers
Example Ethernet 802.3, MAC[media access control]
PPP[Point to point protocol], ARP[Address Resolution
Protocol], NDP[Network discovery protocol]

INTERNET PROTOCOLS [IPV4, IPV6]

(•) IPV4 [Internet protocol version 4]

31 16 len(IP packet length words)	15 8 service Type and Precedence	7 4 Service Type and Precedence	3 0 IP version
63 51 Fragment offset	50 47 Flags	46 32 First Byte Sequence Number in stream	
95 80 checksum	79 72 Type of Protocol	71 64 TTL (Time to live)	
127 96 Source IP Address			
159 128 Destination Ip Address			
q 160 option header words and fields plus the words as padding before data			
v q Data stack of $V = [(v-q)/32]$ words			

Header
Packet Length
$len = \frac{v}{32}$ words

Extended header

$q = (32 \times n - 1)$,
n → number of words
n = 5 words for header plus padding words for extension

Data packet (stack) from a to Transport layer
(maximum size 2^{14} words = 2^{16} B)

Maximum:
$v = (2^{14} - n) \times 32 - 1$

The features of IPV4 are

- The header has 5 words. The header can extend when using option and padding words. Data stack to the network layer has maximum $v = (n + len)$ words where $v <= (2^{14} - n)$
- Header first, second and third word fields are shown in figure.
- Header fourth and fifth words are source IP address and destination IP address
- IP protocol transport is half duplex unacknowledged data flow from Internet layer at one end to internet layer at other end.
- Each IP layer data stack is called IP packet and this packet is not guarnated to reach destination if UDP protocol is used, and guaranteed to reach destination when transport layer protocol is TCP.
- one packet communicates in one direction at an instance.

(*) Internet Protocol Version 6 [IPv6]

The features of Ipv6 are

- It provides larger address space [Ipv6 addresses 128bits]
- Permits hierarchical address allocation and this root aggregation across Internet and keeps the routing table small
- Provides additional optimization for delivery of services using routers, subnets and interfaces
- Manages device mobility, security and configuration aspects
- Expanded and simple use of multicast addressing
- Provisions jumbo grams (big size datagrams)
- Extensibility of options
- Ipv6 address provides a numerical label. It identifies a network interface of a node or other networks nodes and subnets in Ipv6 Internet

(*) RPL [IPv6 Routing Protocol for Low Power Lossy Networks (LLNs)]

→ Low power lossy network refers to a constrained nodes network which has low data transfer rate a low packet delivery rate and unstable links

→ RPL is a non-storing routing mode

→ IoT/M2M low power lossy environment uses RPL protocol

→ Data flow between the nodes is as per Destination Oriented Directed Acyclic Graph [DODAG] model

→ Directed acyclic graph is a data flow model between nodes.

Destination oriented means either upwards or down ward directed in a tree like structure of DODAGs

→ The dataflow directs downwards in an RPL instance from root at transport layer to child nodes and from child node to leaf node at physical layer device node

→ The dataflow directs upwards in an RPL instance from a leaf/child node to another child node and then to the root.

DODAGs are disjoint [no share nodes]

→ The features of RPL are

- A routing protocol for LLNs
- A RPL controlled message are destination advertisement object (DAO), DODAG information object [DIO] and DAG information object [DIO]
- Transfers data at an RPL instance data point to point, point to multipoint or multipoint to point.
- Sends or receives multiple DODAGs at each instance

- Sends data from DODAGs to DODAG root or DODAG leaves to support low data transfer rate compared to IP low packet delivery rate.
- Supports unstable links
- Has optimization objective for an RPL instance.
- Supports coexistence of different optimization objective using multiple RPL instances
- Supports nodes, inform parents of their presence and reach ability to descendants using DIOs.
- Supports soliciting (seeking) information of DODAG using DIS.
- Supports destination, inform nodes for discovery of an RPL instance, know about configuration parameters and select DODAG parents using DIOS.

(*) 6LOWPAN [IPV6 over Low Power Wireless Personal Area Network]

→ IPV6 receives and transmits from/to adaptation layer.

→ The data stack uses 6LOWPAN protocol at adaptation layer before transmit to IPV6

→ 6LOWPAN is an adaptation layer protocol for IEEE 802.15.4 network devices.

→ Features of 6LOWPAN are header compression, fragmentation and a reassembly.

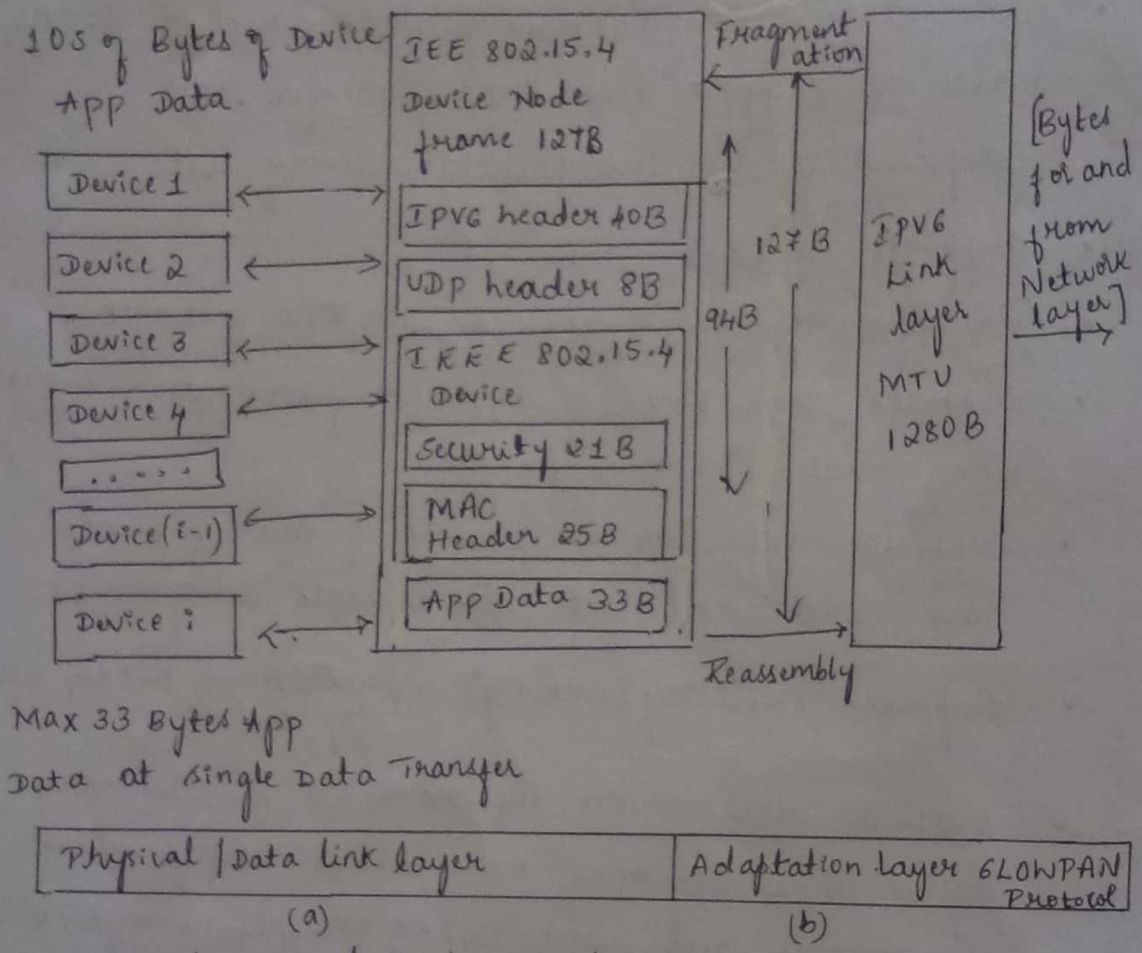

10s of Bytes of Device App Data
Device 1
Device 2
Device 3
Device 4
. . . . .
Device (i-1)
Device i
IEE 802.15.4 Device Node frame 127B
IPV6 header 40B
UDP header 8B
IEEE 802.15.4 Device
Security 21B
MAC Header 25B
App Data 33B
Fragmentation
127B
94B
Reassembly
IPV6 Link layer MTU 1280B
[Bytes for and from Network layer]
Max 33 Bytes App Data at single Data Transfer
Physical / Data link layer
Adaptation layer 6LOWPAN Protocol
(a)
(b)

Fig:- (a) Networked i devices at physical layer in IEEE 802.15.4 WPAN

(b) Adaptation layer 6LOWPAN protocol 127B (maximum) fragmented frames reassembly into IPv6 maximum 1280B for fragmentation of IPv6 MTU 1280B into 127 B frames for transfer to a device

→ Figure shows IPv6 over IEEE 802.15.4 standard network node uses the headers, security and application data as follows

→ IPv6 header = 40B

→ UDP header = 8B

→ Device node MAC address = 25B

→ AES - 128 security = 21B

→ Total device node frame size = 127B (maximum)

→ Link layer MTU [Maximum transmission unit] = 1280B so fragmentation has to be done for transmissions

↳ 6 LOWPAN has following features

- Specifies the IETF recommended methods for reassembly of fragments, IPV6 and UDP headers compression, neighbour discovery
- support mesh routing.

IP ADDRESSING IN THE IOT

→ An IP header has source and destination addresses called IP address

→ Internet uses IPv4 address

→ IOT | M2M use IPv6 address

(*) [IPV4 address] IP Address

↳ IPv4 address has 32 bits.

↳ It can be considered as four decimal numbers seperated by dots.

↳ Each decimal number is decimal value of an octet (=8 bits)

↳ The addresses can be between 0.0.0.0 to 255.255.255.255

↳ Total number of addresses is 2^{32}

↳ An IP address uniquely identify an individual network interface of host.
The address is used for routing

↳ Internet address is visible to outside world to all the routers on internet.

↳ Subnet is a subnetwork consisting of number of hosts or nodes or devices or machines
Subnet address is used for forwarding packets and are not visible to outside world [outside network routers]

↳ (Subnet address) = (Subnet Mask) AND operation (32 bit IP address)

↳ (Host Identifier on subnet) = (Complent of subnet mask) AND operation (IP address)

The subnet address and host identifier on subnet is found by above 2 operations

(*) Static Ip address

- This address is assigned by service providers.
- Two types of addressing methods are used if there are group of hosts
  - classful addressing
  - classless addressing
- In classful addressing the service provider based on the number of hosts in a company's network will allocate class A, B, C, D addresses.
- In classless addressing a block of addresses are allocated based on the requirement, The address will be further subnetted using mask value
  Ex:- 198.136.56.0/4 4→ mask value

(*) Dynamic Ip address

- When a device connects to internet through router the device has to be allocated with an address
- The router and device use DHCP [Dynamic host configuration protocol] which assigns Ip address to device which is called dynamic Ip address
- When a device disconnects or switches off or router boots again then dynamic Ip address is lost and new allocation takes place when device reconnects.

(*) DNS

→ Consider an Ip address 192.136.56.2 this address with number is difficult to remember or use.

The domain name for this address is rajkamal.org

→ Access to web server is made by using website name http://www.rajkamal.org/.

• com , . org, . in and . us are called top level domain (TLD)

A TLD can be further divided into as

. co , . in or .gov.uk

→ A registrar provides domain names at certain cost per year.

↳ The DNS server at the registrar has control panel (cpanel). a

↳ Domain names system (DNS) is an application which provides an Ip address for corresponding service from named domain service

(*) DHCP

When a sensor, actuator or IoT device or node needs to connect to Internet a server called DHCP server provides a dynamic Ip address, subnet mask, ARP and RARP caches.

↳ Dynamic host configuration protocol (DHCP) is a protocol to dynamically provide new Ip addresses and set subnet masks for connected node so that it can use the subnet server and subnet router at communication framework

↳ DHCP enables the process of configuring IP addresses automatically at start up

↳ A node has software component for sending requests to DHCP server and receiving responses. The component is called DHCP client. DHCP client communicates with a server

↳ steps in DHCP protocol are

(1) The DHCP client broadcast a discover request known as DHCPDISCOVER

(2) A DHCP server listens to DHCPDISCOVER and finds the configuration which can be offered to client. Server sends the configuration parameter including an IP address not presently in use at subnet. The configuration parameters are in DHCPOFFER

(3) The selected DHCP server creates and manages bindings the DHCP server also sets time interval during which the offered IP address will be valid for DHCP client node

(4) The DHCP server confirms the binding through a message. It sends DHCPACK after creating the binding.

(5) when the node with DHCP client computer leaves the subnet, it sends a DHCPRELEASE message
If the client doesnot send a DHCPRELEASE within specified time interval, server frees created binding

(6) The server and client also use authentication protocols before considering DHCPDISCOVER from a client and before accepting a DHCP OFFER respectively.

The DHCP protocol takes care that any assigned network address at a given instant is in use by only one DHCP client.

(*) IPV6 Address

↳ IPV6 uses 128 bit address.
A Hexadecimal digit represent 4 bit
128 bit address has 32 hexadecimal digits
Each set of 4 hex-digits are each seperated by a colon or dot in an IPV6 address.
Ex:- 40a0:0acb:8a00:b312:0000:0000:0000:0000

↳ IPV6 addresses are classified into 3 classes
- unicast address is for a single network interface
48 bit or more in unicast specify routing prefix
16 bit or less specify subnet id.
64 bit are interface identifiers.

(•) Anycast address means address of a group of nodes or interfaces.
A packet sent to an anycast address is delivered to just one of the member interfaces one may be nearest host. Nearest host is decided by routing protocol's definition of distance

(•) Multicast address means an address used by multiple hosts, which obtains the multicast address destination by participating in multicast distribution protocol among network routers.
A packet with multicast address delivers to all interfaces that have joined the corresponding multicast group

Application Layer Protocols: HTTP, HTTPS, FTP, TELNET and others

→ A port uses a protocol to send and receive messages

→ A TCP/IP messages must be sent from right port at transmission and to the right port at receiver end else receiver port doesnot listen.

HTTP and HTTPS ports

- HTTP uses a default port number 80
  A web HTTP server listens to port 80 only and responds to port 80 only
  HTTP port number can be specified in URL as
  http:// www. mheducation. com :80/

- HTTPS(HTTP over secure socket layer) uses a port number 443
  Each port at application layer uses a different protocol
  port is assigned a number based on protocol used for transmission and reception.

Important features of HTTP

(•) HTTP is the standard protocol for requesting a URL defined web page resources and for sending response to web server

(•) HTTP is a stateless protocol. The current exchange in HTTP request is independent of previous exchange each request is a fresh request.

(•) HTTP is a file transfer like protocol. HTTP is simple and there are no commands line overheads It follows request, reply paradigm

(•) HTTP protocol is very light (small format) and thus faster compared to other protocols such as FTP

(•) HTTP is flexible. If a connection breaks, client can start by simply reconnecting.

(•) HTTP protocol is based on object oriented programming system (oops)

(•) Multimedia file access is feasible due to provision for MIME (Multipurpose Internet mail extension) type field definition for HTTP 1.0 and 1.1 version

(•) 8 HTTP specific, specified methods and extension methods are included from HTTP 1.1 version

1) GET 2) POST 3) HEAD 4) CONNECT 5) PUT 6) DELETE

7) TRACE 8) OPTIONS

(•) user Authentication is provided along with basic authentication

(•) Digest Access Authentication prevents transmission of username and password as HTML

(•) A host header field adds support to gets and virtual hosts that do not accept or send IP packets

(•) An absolute URL is acceptable to servers

(•) A message during a request from client or during a response from a server has 2 parts

(1) Start line with none or several message headers and empty line

(2) body of message

Message headers used are

- Common (general) headers are added when requesting a server and when responding to a client

- Request headers are for request and client information

- Entity headers contain information about entity body contained in the message

- Response headers are present in response for a server information to a client.

(•) Status codes add in response and caching of a resource provided at a server. Ex: 404 means URL resource not found by server

(•) Byte range specification helps an HTTP server to send large responses in parts.

(•) Selection among various characteristics on retrieval by client is possible when a server sends a responce to client request. Example language and encoding. The client can request for the required language and encoding scheme, the server forwards in that language without changing the content.

(•) Other Ports

- IANA reserves port numbers. The numbers are between 0 and 1023. They are well known ports system process & software connects to them
  Port number 0 is host itself
- Internet Assigned Number Authority (IANA) registers a registered port number which is according to specific protocol.
  Registered numbers (Ports) are between 1024 and 49151.
- 2^{12} numbers between 49152 are unregistered, left for assignment and use by users.
- A user unregistared server has port number above 5000
- A list of widely used ports are available at https://en.wikipedia.org/wiki/List_of_TCP_and UDP_port_numbers

Examples of registered port numbers

Port Number	Protocol	Port Number	Protocol	Port Number	Protocol
17	User list	110	POP3	5223	XMPP-SSL
21	FTP	161	SNMP	5683	CoAP
23	Telnet	389	LDAP		
25	SMTP	443	HTTP		
53	DNS	547	DHCP server		
79	Finger	1293	IPsec		
80	HTTP	2095	cPanel		
108	SNA-GAS	5222	XMPP		

MODULE - 2 DATA COLLECTION, STORAGE AND COMPUTING USING A CLOUD PLATFORM

CHAPTER - 2

Neachitha. S
Asst. Prof

(•) INTRODUCTION

Few Methods for data collection and storage are

- Saving data at a local server for device nodes
- Communicating and saving device data in files locally on removable media such as micro SD cards and computer hard disks.
- Communicating and saving data and results of computations in a dedicated data store.
- Communicating and saving data at a local node which is a part of distributed DBMS
- Communicating and saving data at remote node in distributed DBMS
- Communicating on Internet and saving at a data store in a web server
- Communicating on Internet and saving at data centre for an enterprise

↳ cloud is a new generation method for data collection, storage and computing.

Cloud Computing paradigm for Data Collection, Storage and computing.

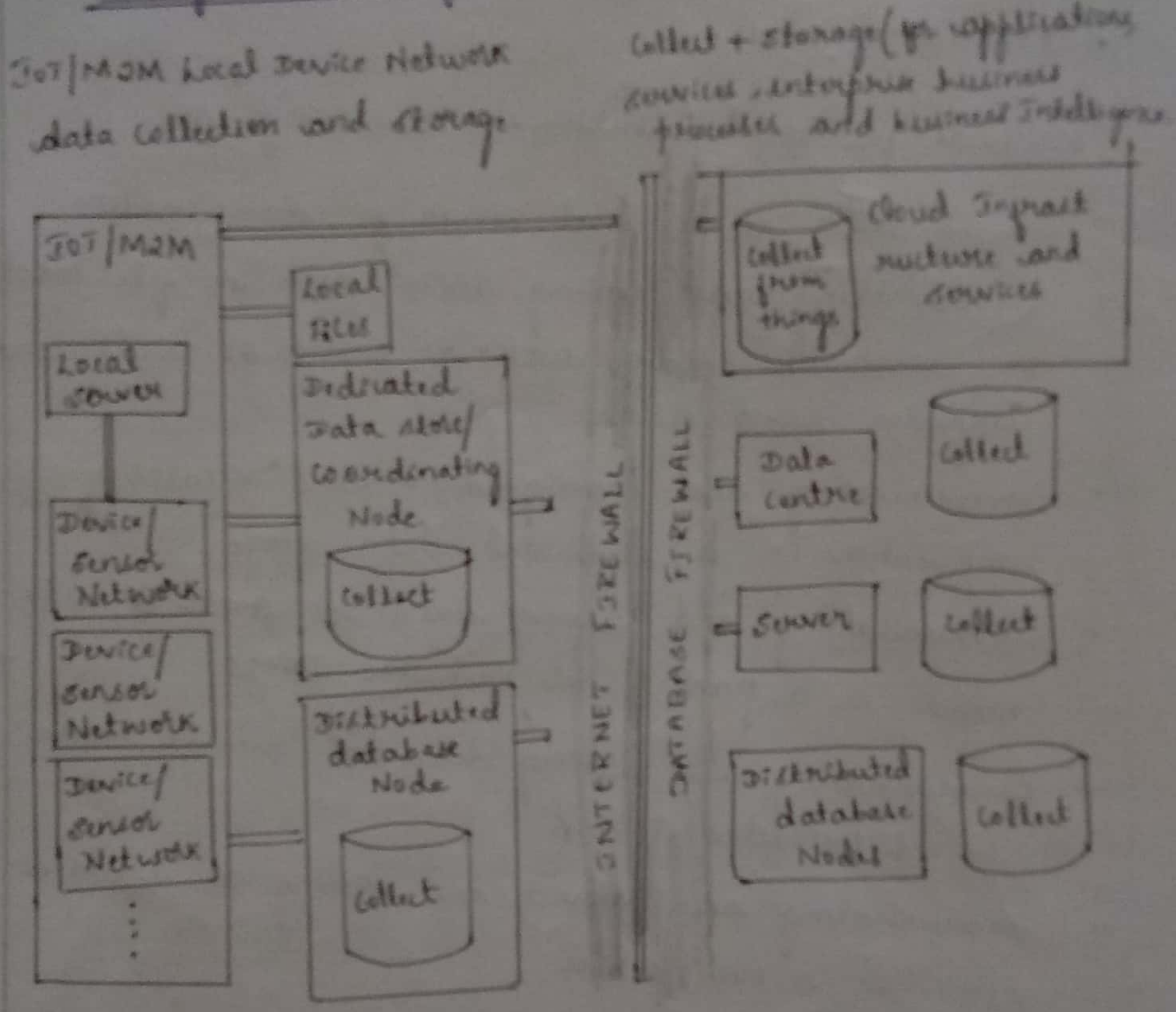

IoT/M2M Local Device Network data collection and storage
Collect + storage (for applications services, enterprise business processes and business Intelligence
IoT/M2M
Local server
Device/sensor Network
Device/sensor Network
Device/sensor Network
Local files
Dedicated Data store/coordinating Node
Collect
Distributed database Node
Collect
INTERNET FIREWALL
DATABASE FIREWALL
Collect from things
Cloud Infrastructure and services
Data centre
Collect
Server
Collect
Distributed database Node
Collect

Figure: Different methods of data collection, storage and computing

Figure shows

(i) Devices or sensors network data collection at device web server

(ii) local files

(iii) Dedicated data store at coordinating node

(iv) Local node in distributed DBMS

(v) Internet connected data centre

(vi) Internet connected server

(vii) Internet connected distributed DBMS nodes

(viii) Cloud infrastructure and services.

↳ Cloud computing paradigm is a great evolution in Information and communication technology (ICT). The new paradigm uses XAAS at internet connected clouds for collection, storage and computing.

Key terms used in cloud computing

(•) Resource: It refers to one that can be read, written or executed.
A path specification can also be a resource.
The resource is atomic, which is usable during computation.
A resource may have single or multiple instances.
A data pointer, pointer, data, object, data store or method can also be a resource.

(•) System resource: It refers to an operating system, memory, network, server, software or application.

(•) Environment: It refers to an environment for programming, program execution or both.
Example: Cloud 9 online provides an open programming environment for BeagleBone board for development of IoT devices.
Google app engine environment for creation and execution of web application in python or Java.
Windows environment for creation and execution of application.

(*) Platform refers to basic hardware, operating system and network and is used for software applications & services over which programs can be run or developed

A platform may provide a browser and APIs which can be used as base on which other applications can be run or developed

(*) Edge Computing: It is a type of computing that pushes computing applications data and services away from centralized nodes to IoT data generating nodes

Pushing computations from centralized nodes enable usage of resources at device nodes

(*) Distributed Computing: It refers to computing and usage of resources which are distributed at multiple computing environments over the Internet The resources are logically related, they can communicate among themselves using message passing and transparency concepts, cooperating and movable without affecting computations

(*) Service: It is a software which provides capabilities, logically grouped and encapsulated functalities. A service is called by an application for using the capabilities. Service binds to service level agreement between service [provider] and application(end user). One service can call another service

(*) Web service:- It is an application identified by a URI, described and discovered using XML based web service description language [WSDL].
A web service interacts with other services and application using XML messages.
It exchanges objects using Internet protocols.

(*) Service oriented architecture:- It consists of components which are implemented as independent services the components can be dynamically bonded, orchestrated and they possesses loosely coupled configurations communication between components is through messages

(*) Web computing:- It refers to computing using resources at computing environment of web server or web services over Internet.

(*) Grid computing:- It refers to computing using pooled interconnected grid of computing resources and environments in place of web servers

(*) utility computing:- It refers to computing using focus on service levels with optimum amount of resources alloted, It takes the help of pooled resources ashe and environment for hosting applications when required.

(*) Cloud computing:- It refers to computing using a collection of services available over the Internet. Service provider provides computational functionalities and infrastructure for connected systems.
It also provides distributed grid computing and utility computing.

(*) Key performance indicator (KPI) :- It refers to set of values, including minimum, maximum and average values which indicate a scale.

- A service has to be fast, reliable and secure.
- KPIs monitor the fulfillment of these objectives.
- Example a set of values may relate to quality of service characteristics such as bandwidth availability, data backup capability, peak and average workload handling capacity, ability to handle defined volume of demand at different times of day and ability of to deliver defined volume of service
- A cloud service should fulfill the defined minimum, average and maximum KPI values agreed in SLA.

(*) Localisation :- It means localization of QOS level and KPIs are used to monitor cloud computing usage.

(*) Seamless cloud computing :- It means during computing the content of usage and computation continue without any break when service usage moves to location with similar QOS level and KPIs.

(*) Elasticity:- It denotes that an application can be used for local or remote applications and realease them after usage.

(*) Measurability:- It refers to a resource or service which can be measured for controlling or monitoring. It enables report of delivery of resource or service.

(*) Homogeneity:- Homogeneity of different computing nodes in a cluster refers to integration with kernel providing the automatic migration of processes from one to other homogeneous nodes

(*) Resilient computing:- It refers to the ability of offering and maintaining the accepted QOS and KPIs in presence of identified challenges, defined and appropriate resilience metrics, and protecting the service

(*) Scalability:- Scalability in cloud services refers to ability of using which an application can utilize smaller local resource as well as remotely distributed servers and then increase or decrease the usage while paying the increased usage cost

(*) Maintainability:- In cloud service it refers to storage applications, computing, infrastructure, services data centres and servers maintenances which are responsibility of remotely connected cloud services with no costs to user.

(*) XAAS:- It is a software architectural concept that enables deployment and development of applications and offers services using web and SOA.

(*) Multitenant:- Multitenant cloud model refers to accessibility to a cloud platform and computing environment by multiple users who pay as per agreed QoS and KPIs which are defined at separate SLAs with each user.

Cloud computing Paradigm

- cloud computing means a collection of services available over internet
- cloud provides computational functionality
- cloud computing deploys infrastructure of a cloud service provider
- the infrastructure deploys on utility or grid computing or web services environment that includes network, system, grid of computers or servers or data centres

Cloud platform services

(i) Infrastructure for large data storage of devices RFIDs, industrial plant machines, automobiles and device networks

(ii) Computing capabilities such as analytics, IDE.

(iii) Collaborative computing and data store sharing.

Cloud Platform usages

Cloud platform are used for connecting devices data, APIs, applications and services, persons, enterprise business and XAAS.

The following equation describes a simple conceptual framework of internet cloud

< internet cloud + clients = user applications and services with no boundaries and no walls >

An application or service executes on platform which includes the operating system, hardware and network.

Multiple application may be designed to run on diversified platforms.

Applications and services need to integrate them on a common platform and running environment.

Cloud storage and computing environment provides a virtualised environment

Virtualized environment refers to a running environment which appears to be one environment but physically there may be two or three more running environment and platforms may be present.

Virtualization

- virtualized environment allows applications and services to execute in an independent execution environment.

- Each one of them stores and executes in isolation on same platform, though in fact it may actually execute or access to a set of data centres or servers or distributed environment services and computing systems.

- Applications or services which are hosted remotely and are accessible using Internet can be utilized at a user application or service in a virtualized environment through internet

- Application need not be aware of platform just internet connectivity to platform called cloud platform is required.

  The storage is called cloud storage. The computing is called cloud computing.

  The services are called cloud services in line with web services which host on web service

- virtualization of storage means user application or service accesses physical storage using abstract database or file system or logical drive or disk drive, storage may be accessible using multiple interfaces or servers

  example: Apple icloud

- Network function virtualization (NFV) means a user application or service accesses the resource appearing as just one network, though network access to resource may be through multiple resources and networks.

- Virtualization of server means a user application accesses not only one server but in fact accesses multiple server

- Virtualized desktop means the user application can change and deploy multiple desktops though the access by user is through their own computer platform (OS) that in fact may be through multiple OS and platforms & remote computers

Cloud computing Features and Advantages

Essential features of cloud storage and computing are

→ Elasticity

→ Scalability

→ Homogenity

→ Resilient computing

→ Low cost

→ Massive scale availability

→ Maintainability

→ Virtualization

→ Advanced security

→ Resource pooling in multitenant model

→ on demand self service to users for provisions of storage, computing servers, software delivery and server time

→ Broad network accessibility in virtualized environment to heterogeneous users, clients, systems and devices

→ Interconnectivity platforms with virtualized environment for enterprises and provisioning of in between service level agreements (SLAs)

Cloud computing concerns

Concerns in usage of cloud computing are.

→ Requirement of a constant high speed internet connection

→ Limitations of services available

→ possible data loss

→ Non delivery as per defined SLA specified performance

→ Different APIs and protocols used at different clouds

→ Security in multi tenant environment needs high trust and low risks

→ Loss of users' control

Cloud deployment models

(*) Public cloud :- It is for general usage. This model can be used by educational institutions industries, government institutions or business or enterprises and is open for public use.

(*) Private cloud :- This model is exclusive for use by institutions, industries, businesses or enterprises and is meant for private use in organisation by employees and associated users only

(*) Community cloud :- This model is exclusive for use by a community formed by institutions, industries, business, enterprises and for use within the community organization, employees and associated users. The community specifies security and compliance considerations.

(*) Hybrid cloud :- A set of 2 or more distinct clouds with distinct data stores and applications that bind between them to deploy the proprietary or standard technology

Cloud platform architecture is a virtualized network architecture consisting of a cluster of connected servers over data centres and service level agreements (SLAs) between them.

A cloud platform controls and manages resources and dynamically provisons the networks, servers and storage

Cloud platform applications and network services are utility, grid and distributed services.

Examples q cloud platforms are Amazon EC2, Microsoft Azure, Google Apple Engine, Xively, Nimbits

Everything as a service and cloud service models

- cloud services can be considered as distribution service - a service for linking the resources and for provisions q coordinating between the resources

- cloud computing can be considered by simple equation

  cloud computing = Saas + Paas + Iaas + Daas

- figure shows 4 cloud service models and examples

- Saas means software as a service. the software is made available to an application or service on demand.

  Saas is a service model where the application or services deploy and host at the cloud and are made available through internet on demand by service user.

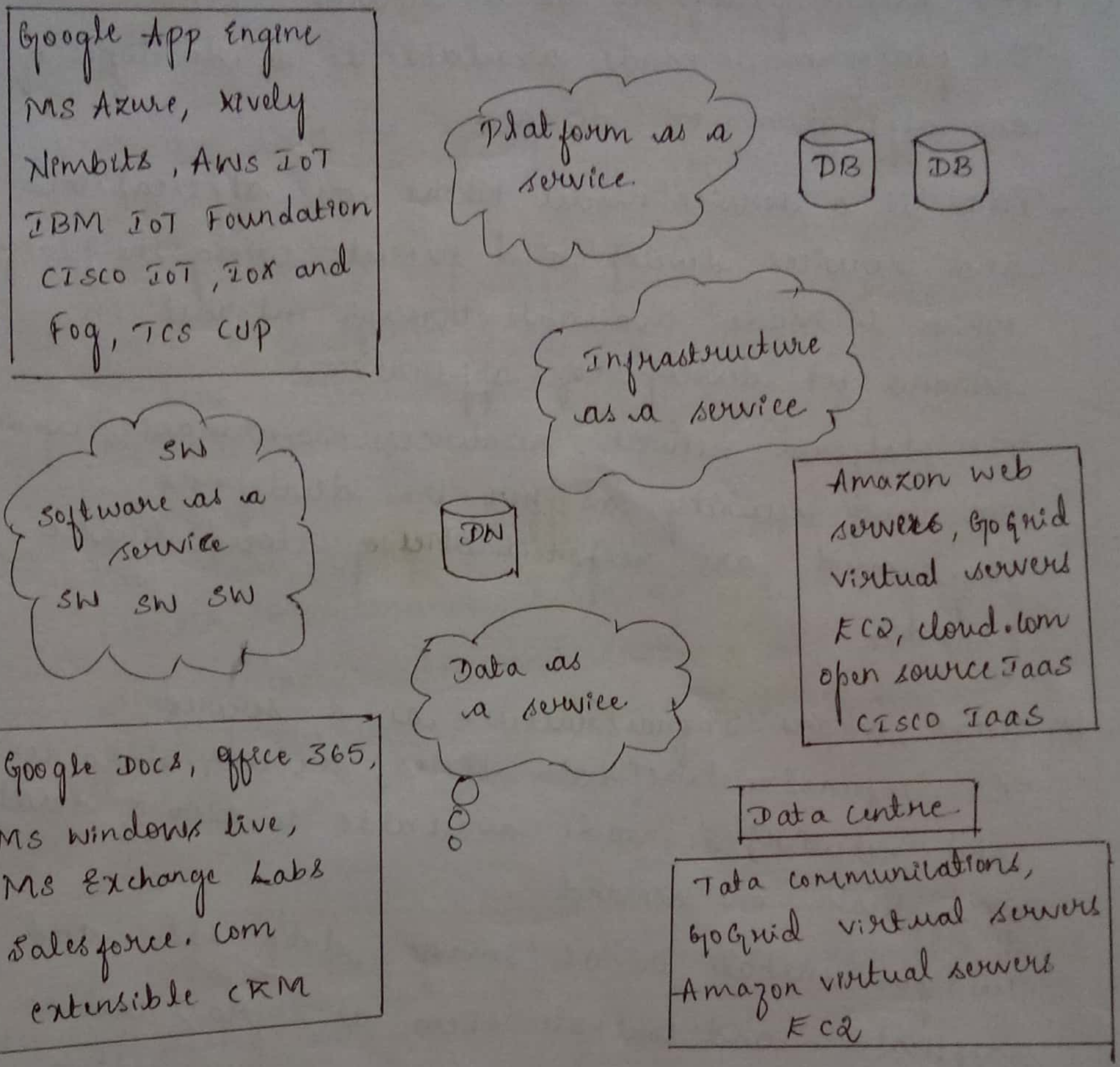


Fig:- Paas, Saas, Iaas and Daas cloud service model

- The software control, maintenance, updation to new version and infrastructure, platform and resources requirements are responsibilities of cloud service provider.

(*) Paas means platform as a service

- The platform is made available to a developer of an application on demand.

- Paas is a service model where the applications and services develop and execute using the platform which is made available through internet on demand for developer of applications.

- The platform, network, resources, maintenance, updation and security as per the developer's requirement are responsibilities of cloud service provider

(*) Iaas means Infrastructure as a service.

The infrastructure [data stores, servers, data centres and network] is made available to user or developer of application on demand.

Developer installs the OS image, data store and application and controls them at infrastructure

- Iaas is a service model where the application develop or use the infrastructure which is made available through Internet on demand or rent by a developer or user.

- Iaas computing systems, networks and security are the responsibility of cloud service provider

(*) Daas means data as a service

Data at a data centre is made available to a user or developer of application on demand

- Data at data centre is made available to a user or developer of application on demand
- Daas is a service model where the data store or data warehouse is made available through the Internet on demand or sent to an enterprise.
- The data centre management, 24x7 power, control network, maintenance, scale up, data replicating and mirror nodes and systems as well as physical security are responsibilities of data centres service provider.

IoT Cloud Based Data Collection, Storage and Computing services using Nimbits.

- Nimbits enables IoT on an open source distributed uted cloud
- Nimbits functions as an M2M system, data store, data collector and logger with access to historical data.
- Nimbits architecture is a cloud based google App engine.

- Nimbits Paas service offers following features

- It supports multiple programming languages including Arduino, new Arduino library, push functions from Arduino cloud, Java script, HTML & Nimbits. io java library

- Edge computing locally on embeded systems built up of local applications. It runs the rules and pushes important data up to cloud running. when connected over internet and an instance of Nimbits server hosts at device nodes which is then enabled

- Nimbits server functions as backend platform. Nimbits data point can relay data between the software systems or hardware devices such as Arduino, using the cloud as a backend.

- An open source java library called nimbits. io enables easy development of java, web and android solutions.

- It provides rule engine for connecting sensors, persons and software to cloud and one another Rules can be for calculations, statistics, email alerts, xmpp messages, push notifications and more

- It provides a data logging service and access and stores the historical data points and data objects.

- Storage in any format that can be serialised into a string such as JSON or XML
- It filters the noise and important changes sent to another larger central instance
- It process specific type of data and can store it
- Time or geostamping of data
- Nimbits clients provide over Internet, data collection in real time, charts, chart and graphical plots of collected data and data entry
- Data visualisation for data of connected sensors to IOT devices
- Supports the alerts subscription, generation and sending in real time over internet
- It creates streams of data objects and stores them ~~at~~ in a data point series
- Data accessibility and monitoring from anywhere and is used to shape the behaviour of connected devices and software.
- It supports the mBed™, Arduino, Raspberry pi based and other hardware platform based IOT devices
- Web service APIs are easy to implement on device hardware acting as clients to Nimbits web service and connect to web service and send data.
- It deploys software on Google App engine, any J2EE server on Amazon EC2 or on a Raspberry Pi.

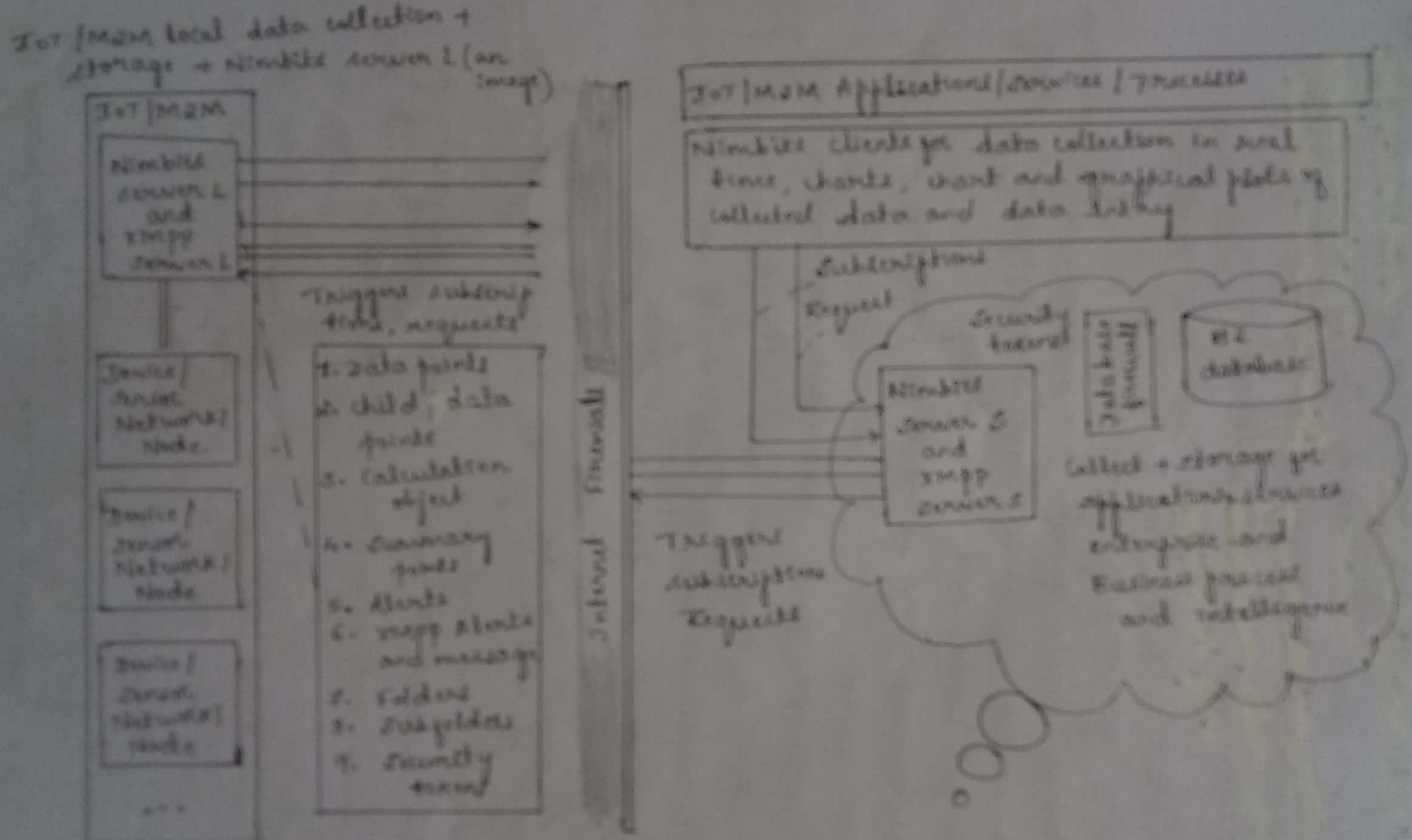

IoT/M2M local data collection + storage + Nimbits server L (an image)
IoT/M2M
Nimbits server L and XMPP server L
Device/sensor network node
Device/sensor network node
Device/sensor network node
...
Triggers subscriptions, requests
1. Data points
2. Child data points
3. Calculation object
4. Summary points
5. Alerts
6. XMPP alerts and message
7. Folders
8. Subfolders
9. Security token
Internet firewall
IoT/M2M Applications/services/processes
Nimbits clients for data collection in real time, charts, chart and graphical plots of collected data and data listing
Subscriptions Request
Security token
Database firewall
BI database
Nimbits server S and XMPP server S
Triggers subscriptions Requests
Collect + storage for applications, services enterprise and Business process and intelligence

Fig: Connected devices, sensor nodes, network data points, Nimbits server, deployment at the device network nodes and networked with Nimbits server at Cloud for application and services

→ Figure shows connected devices, sensor nodes, network data points, Nimbits server, deployment at device network nodes, and networked with Nimbits server [Paas, Saas and Iaas service] at cloud. for applications and services.

→ Architecture shown in figure shows a Nimbits serverL which deploys at each device node and is at instance of Nimbit servers at the cloud Each Nimbit serverL of the device node generates the calculation objects for device nodes.

→ Each node also hosts an XMPPserverL an instance of XMPPserverC at cloud.

→ XMPP serverL deploys at each device node and generates the data feed channels for XMPP messages and alerts. Each XMPPserverL sends feeds to XMPP Server C

→ Data points:-

↳ A data point means collected value of a sensor in a group of sensors.

↳ Data point organise data in a number of ways.

↳ Data points can have child points [subpoints]

↳ Example:- If light level is a point then light on & off is a child point, light level above or below threshold can be another child point.

↳ points can be in a folders. The folders can go as deep as like a tree [folders ⇒ subfolders < subfolders subfolders]

↳ Any type of document can upload and organize them with the points. Files can be shared publicly or with connections

↳ A subscription data feed is a special point of each user that logs system messages, events, alerts from other points which are subscribed by a service and more

→ Data channels:-

↳ A user can create a data feed channel which shows the system events and messages that also shows data alerts which are subscribed to shows up in the feed.

↳ The user can subscribe to data point of other users also and configure subscription to send messages to feed

↳ The user can observe the idle, high or low alerts here in real time.

↳ The user data feed is just another Nimbits data point.

→ Using Advanced Features

↳ An application can create a connection to another Nimbits application or service

↳ The application sends an invitation and if the invitee approves then application can see inviter's points and data in their tree

↳ Nimbits 3.6.6 introduced H2 database engine
Nimbits 3.8.10 includes H2 database engine
H2 is a java SQL database
APIs are pure java
The main features of H2 are

- very fast, open source, JDBC API
- Embeded and server modes; in memory databases
- Encrypted database
- ODBC driver
- Full text search
- Multi version concurrency
- Browser based console application
- Small footprint [around 1.5 MB jar file size]

↳ MySQL is not in pure Java and have no provision for in memory & encrypted databases.
Footprint [Dynamically linked library] is 4.5MB

→ Security Tokens
Nimbits 3.9.6 provide security tokens in a new way

→ Breakthrough performance and Data integrity
↳ Nimbits server 3.9.10 version launched in June 2015
.breakthrough and data integrity

→ Alerts
↳ A filter means applying some rule to get new data for a data point. The filter item in tree called "ah" is for XMPP alerts

↳ A user application can have many Jabber IDs [JIDs] for a single point - alerts and messages can be sent over XMPP using custom JID of points

→ Jabbing

↳ It means pushing the alerts or messages down quickly or pushing repeatedly.

↳ Each type of alert or message is assigned a Jabber ID, called JID

↳ JID has 3 main parts
- node identifier (optional), resource identifier (optional)
- domain identifier (required)

↳ JID is written in notation as
<JID>:= [<node>"@"]<domain>["/"<resource>].

→ Subscriptions

↳ user can create many subscriptions for single point

↳ user can subscribe to one of points, other user or anyone's public point to get alerts.

↳ The user get alerts when the point goes into an alarm state or receives new data.

↳ Alarm state means reaching present value

↳ subscriptions are alternative to configuration of alert

↳ A subscription to a point creates on configuring the subscription - how an application is programmed to get alerts and what events trigger the alert.

↳ An application can have many subsuiptions to a point. It may subscribe to another user's point to which the user is connected to find a point using the search engine on nimbits.com or may subscribe to a public point from another user who is not even known.

→ Summary point:-

A user creates summary point which can compute averages, minimum, maximum, standard deviations variance and sums of another point on a specific interval basis

→ Calculations:

↳ user can create calculation object for a point.

↳ The object can organise in a tree and a user can apply many formulas for a single data point

Example:- In temperature sensor data point one formula is for increase over the last value while the other is for increase over a normal value, each time a new temperature data point is recorded.

Module - 3
Chapter - 1

Nischitha . S
Asst. Professor

Prototyping Embeded Device software

- Prototype development of program requires bootloader OS and IDE. Software embeds into device platform
- First level in IOT architectural concept is gathering and consolidating

↳ Second level in IOT architectural concept is connection to Internet

- An IDE enables development of software for functions at first and second levels.
- Bootloader is a system software that is loaded & embeded into a microcontroller chip or computing platform to allow the system to start its functions

↳ Bootloader enables communication with a computer having an IDE.

- The IDE in general has API's, libraries, compilers RTOS, simulator, editor, assembler, debugger, emulator logic analyser, code burner and other software for integrated development of system.

↳ An IDE may be open source. Example :- Arduino has an open source IDE.

↳ IDE enables development of codes on computer and later on downloading (pushing) of codes on to embeded device.

The code burner places the codes into flash memory or EEROM or EPROM.

Programming Embeded Device Arduino Platform using IDE

- Arduino board can be programmed using avr-gcc tools Arduino board has pre installed bootloader embeded into the firmware.
- Arduino programmer develops the codes using graphical cross platform IDE. Arduino board is based on processing language which makes the programming easy.
- The board connects to computer which runs the IDE. The bootloader program enables hand overs the control and enables running of the loader. Loader loads the required OS functions and software into the system hardware and networking capabilities into the board.
- The Arduino bootloader allows multitasking by the usage of interrupt handling functions for each task. when an instruction for interrupt for example INT n executes, then interrupt handling function n is called for exeution.
- The IDE consists of a set of software modules which provide the software and hardware environment for developing and prototyping the software for a specific device platform.
- The bootloader enables the computer to push the developed codes into a board using Arduino IDE through a USB cable or a labelled serial port.

Development of codes

- Arduino IDE functions as a file editor for codes using processing environments and library functions The editor provides automatic indentation, highlights the syntax of codes and matches braces. The edited file compiles, checks and lists errors if no errors, it allows pushing of codes for embedding onto the board through serial or USB port.

- In Arduino only 2 functions are necessary to define executable program functions for the board namely setup() and loop().

setup() → Runs at start and is used for initialising setting and

& loop() → has a program in endless loop using statement while (true) { statements; } which runs till power off.

- Serial monitor at the IDE enables messages from the embeded software for the microcontroller into the computer screen ~~which~~ where IDE is set up. The messages are required during testing and debugging the downloaded software during testing stages.

Programming for Arduino controlled traffic control lights (TLs) at a road junction.

Case :- port LEDs are ON-OFF programmed so that north and south pathways directed roads and traffic is switched ON and east and west pathways traffic switched OFF.

→ Assume Arduino UNO board as an embeded device platform.

3 ~~traffic~~ lights Red, Yellow, Green needs to be controlled on each of four north, south, east & west clockwise pathways.

Let 12 GPIO pins on UNO connect 12 number externally connected LED's R0, Y0, G0, R1, Y1, G1, R2, Y2, G2, R3, Y3, G3 the port LED's represent the traffic lights during the prototype development and testing stage

Step 1:- Declaring data types, constants, variables and functions used.

/* Assume 12 digital IO ports assigned to external LEDs are port 2 to 12 and port 14.

Pin 13 connects the board LED and be used for indicating successful running of developed codes during testing phases.

```
int internalLED = 13;
int ledR0, ledY0, ledG0, ledR1, ledG1, ledY1, ledR2,
  ledY2, ledG2, ledR3, ledY3, ledG3;
ledR0=2; ledY0=3; ledG0=4; ledR1=5; ledY1=6; ledG1=7;
ledR2=8; ledY2=9; ledG2=10; ledR3=11; ledG3=14; ledY3=12;
```

```
Void north-south-Green() {
digitalWrite (ledR0, LOW); digitalWrite (ledY0, LOW);
digitalWrite (ledG0, HIGH);
digitalWrite( ledR2, LOW); digitalWrite( ledY2, LOW);
digital write( led G2, HIGH);
};
Void east-west-Red () {
digitalWrite (led R1, HIGH); digitalWrite (ledY1, LOW);
digitalWrite (led G1, LOW);
digitalWrite (ledR3, HIGH); digitalWrite (ledY3, LOW);
digitalWrite (led G3, LOW);
};
Step 2:- Void setup () {
 /* Assign each pin mode as output */
pinmode (ledR0, OUTPUT);
pinmode (ledY0, OUTPUT);
   :
pinmode(ledG3, OUTPUT);
pinmode (internalLED, OUTPUT);
/* Initialise start of the board and sequence */
 digitalWrite (internal LED, HIGH);
/* Let UART mode baud rate = 9600 */
Serial.begin (9600);
Serial.println ("Arduino project.program for controlling
three traffic lights Red, Yellow, Green at four pathways");
Serial.println("Arduino board LED glows when cycle starts
for the sequence of lights turning high and turns off
for brief interval in order to indicate the successful
completion of the cycle");
Serial.println( "Twelve 12 external LED's, R0, Y0, G0, R1, Y1, G1
R2, Y2, G2, R3, Y3, G3 corresponds to 12 traffic lights at
north, east, west four pathways");
}
```

Step 3:- /* Loop function which endlessly runs*/

```
void loop() {
/* Assume no right turn from pathways or left turn from
  pathways permitted */
  north_south_Green();
  east_west_Red();
}
```

Programming for Arduino controlled traffic-control lights at a road junction with control of an intervals

→ Assume Arduino UNO board as an embeded device platform

3 traffic lights red, Yellow, Green needs to be controlled on each of four north, south, east and west controlled clockwise pathways.

Let 12 GPIO pins on UNO connect 12 number externally connected LED's R0, Y0, G0, R1, Y1, G1, R2, Y2, G2, R3, Y3, G3. The port LED's represent the traffic lights during the prototype development and testing stage.

Step 1:- Declaring data types, constants, variables and functions used.

/* Assume 12 digital IO ports assigned to external LEDs are port 2 to 12 and port 14. pin 13 connects the board LED and be used for indicating successful running of developed codes during testing phases.

```
int internalLED = 13;
int ledR0, ledY0, ledG0, ledR1, ledY1, ledG1, ledR2,
ledY2, ledG2, ledG3, ledY3, ledG3;
ledR0 = 2; ledY0 = 3; ledG0 = 4; ledR1 = 5; ledY1 = 6; ledG1 = 7;
ledR2 = 8; ledY2 = 9; ledG2 = 10; ledR3 = 11; ledY3 = 12; ledG3 = 14;
void_north_south_Green(){
digitalwrite(ledR0, LOW); digitalWrite(ledY0, LOW);
digitalwrite(ledG0, HIGH);
digitalwrite(ledR2, LOW); digitalwrite(ledY2, LOW);
digitalwrite(ledG2, HIGH);
};
void_east_west_Red(){
digitalwrite(ledR1, HIGH); digitalwrite(ledY1, LOW);
digitalwrite(ledG1, LOW);
digitalwrite(ledR3, HIGH); digitalwrite(ledY3, LOW);
digitalwrite(ledG3, LOW);
};
void_north_south_Yellow(){
digitalwrite(ledR0, LOW); digitalwrite(ledY0, HIGH);
digitalwrite(ledG0, LOW);
digitalwrite(ledR2, LOW); digitalwrite(ledY2, HIGH);
digitalwrite(ledG3, HIGH);
};
void_east_west_Green(){
digitalwrite(ledR1, LOW); digitalwrite(ledY1, LOW);
digitalwrite(ledG1, HIGH);
digitalwrite(ledR3, LOW); digitalwrite(ledY3, LOW);
digitalwrite(ledG3, HIGH);
};
void_north_south_Red(){
digitalwrite(ledR0, HIGH); digitalwrite(ledY0, LOW);
digitalwrite(ledG0, LOW);
digitalwrite (ledR2, HIGH); digital write( led Y2, LOW);
digitalwrite (ledG2, LOW);
};
void _ east_ west _ Yellow () {
digital write (ledR1, LOW); digital write( led Y1, HIGH);
digitalwrite (ledG1, LOW);
digital write ( ledR3, LOW); digitalwrite ( ledY3, HIGH);
 digitalwrite ( led G3, LOW);
};
```

Step2:- void setup() {

```
/* Assign each pin mode as output */
pinmode ( ledR0, OUTPUT);
pinmode ( led Y0, OUTPUT);
  :
  :
pinmode ( ledG3, OUTPUT);
pinmode ( internalLED, OUTPUT);
/* Initialise start of the board and sequence */
digitalwrite( internal LED, HIGH);
/* Let VART mode baud rate = 9600 */
  serial . begin (9600);
  serial. println ("Arduino project. program for controlling
  three traffic lights Red, Yellow, Green at four pathways):
  serial. println(" Arduino board LED glows when cycle
  starts for the sequence of lights turning high and
  turns off for brief interval in order to indicate.
  the successful completion of the cycle!");
serial.println (" Twelve 12 external LED's R0, Y0, G0, R1, Y1, G1,
R2, Y2, G2, R3, Y3, G3, corresponds to 12 traffic lights
at northa, east, west four pathways");
}
```

Step 3:- void loop() {

```
/* Assume no right turn from pathways or left turn
 from pathways permitted */
 north_south_Green();
 east_west_Red();
 delay(30000); //wait for 30 seconds;
 north_south_yellow();
 delay(10000);
 east_west_Green();
 north_south_Red();
  delay(30000);
  east_west_Yellow();
  delay(10000);
/* Let internal board LED flash off for 6s before the
 sequence of lights repeat */
// statements for test;
digitalwrite(internalLED, LOW); delay(6000);
serial.println("one sequence completed");
digitalwrite(internalLED, HIGH);
}.
```

Devices, Gateways, Internet and web/cloud services software - Development

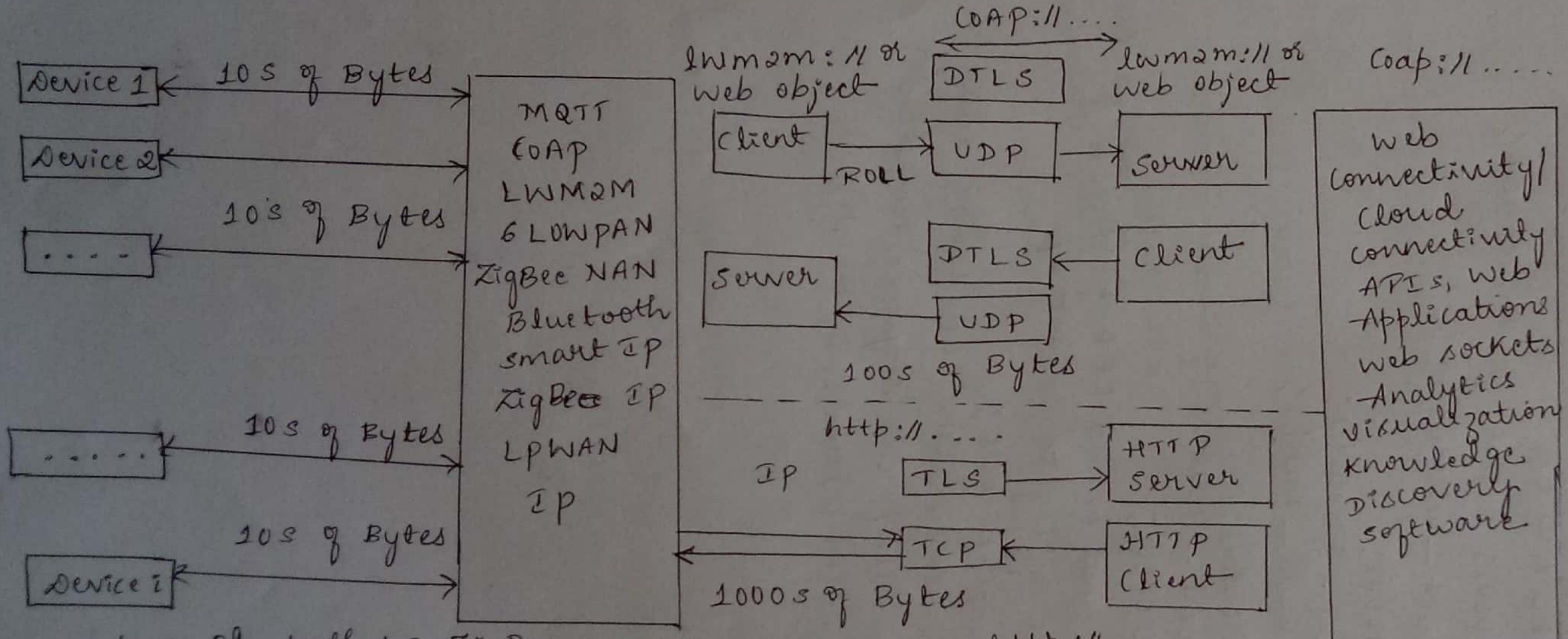
1. Physical/Data link and Adaptation layers software using IDE

2: IoT or M2M area Local area network and gateway software

3: Network and Transport Layers software

4: Application support layer APIs/software.

5: Application layers API's/software



wireless Bluetooth LE, ZigBee, LPWAN
Wi-Fi or wired: UART, USB, I2C, SPI
SDIO or Ethernet

Fig: Five levels for. software development for application and services for IoT or M2M

Module - 3
Chapter 2

Neechitha. S
-Asst. Professor

Introduction to IoT Privacy and Security.

(*) INTRODUCTION

→ International organizations are making a number of efforts to include trust, data security and privacy in IoT design.

↳ Trust is important. Trust in IoT means dependen dability, accuracy, quality of data from multiple sources for applications and services.
An organisation called open trust alliance, established IoT trustworthy group[ITWG] for recognising the priority from begining of product development and addressing holistically

↳ Security is important. Example ATM messages should be communicated securely over internet otherwise it will cause problem

↳ Privacy is important. The video clips communicate on internet in a smart home security application. If clips reach other parties/entities it can lead to serious breach of security.

Definitions | Meanings

(*) **Message**:- It is a string that represent data or client request & server response which communicate between sender and receiver objects.

(*) **Hash**:- It is a collection or bundle which gives an irreversible result after many operation on data and operations are just one way. Example:- secure hash algorithm, where hash value [128 & 256 bit] is exchanged between sender and receiver. The receiver compares the received hash value if it matches then authenticates the sender message

(*) **Digest**:- It is a process which gives irreversible result involving many operations. example: Message digest 5 (MD5) which is similar to hash algorithm in operation gives digest value in place of hash value

(*) **Encryption**:- It is a process of generating new data using a secret key known only to sender and receiver. The sender and receiver exchange the key before exchanging messages. The key is usually 128, 192 & 256 bit

(*) **Decryption**:- It is a process of retrieving the data from encrypted data

(*) <u>Use Case</u>:- It means a list of event steps or actions which define interactions between two ends in which one end is playing a role and other is system. use case define the required behaviour of software under development. use case describe the details of usage of software and its normal behaviour

(*) <u>Misuse case</u>:- It defines the behaviour which is not required from software under development It defines the behaviour Which should not happen It also specifies threats. Misuse case gives information and help in identifying the requirement of new use cases for prevention of attack and find out what should not happen.

(*) <u>Layer</u>:- It is a stage at which actions takes place as per specific protocol or method and then result passes to next layer until the set of actions complete. Layer's model representation of design represents a set of systematic actions which are followed sequentially for accomplishing a task.

(*) <u>Sublayer</u>:- It it a layer of consisting of various sublayers in a model to provide set of actions sequentially taking place at the layer

(*) Firewall :- It is a software interface. It interconnects all network with differing trust, and It provides perimeter defence and prevents penetration of unauthorised data.

It functions as choke point for controlling and monitoring

It does auditing and provides controlled access

It allows only authorised traffic and applies restrictions on network services

It can raise alarms for abnormal behaviours

Vulnerabilities, Security Requirements and Threat Analysis

→ Privacy :-

Message privacy means that the message should not reach the hands of unrelated entity

Message generated from a device platform should reach the targeted goal only

Privacy means no interference or disturbance from other.

IoT requires privacy policy. A privacy policy has to determine how much of IoT devices data and which data needs complete privacy and which need limited privacy

National Institute of Standards and Technology (NIST) USA is developing the standards for privacy

→ Vulnerabilities of IoT

↳ Vulnerability means weak without complete protection, weakness to defend oneself & can be easily influenced from surrounding unwanted things from itself

↳ There are many vulnerabilities due to participation of number of layers, hardware sublayers and software in applications and services

↳ IoT can be vulnerable to - - eavesdropping. the third party may send a fake request and the server may assume it as part of device and may respond

↳ Eavesdropping can be prevented by using secret key encryption.

↳ open web Application security project [OWASP] has considered the security issues of IoT for purpose of helping developers, manufacturers and consumers OWASP is open source and has free to use licensing policy
OWASP has undertaken a number of security related subprojects such as ones for defining the top vulnerabilities, attack surface areas and testing guides

↳ OWASP has listed top ten vulnerabilities in IoT applications / services which are as follows

1) Insecure web interface
2) Insufficient authentication or authorization
3) Insecure network services

4) Lack of transport encryption/integrity verification
5) privacy concerns
6) Insecure cloud interface
7) Insecure mobile interface
8) Insufficient security configurability
9) Insecure software or firmware
10) Poor physical security

↳ CERT coordination centre [CERT/cc] at carnegie Mellon university (CMU) has taken the initiative for coordinating the vulnerabilities in IOT devices

(*) Security Requirements

↳ IOT reference architecture means a guide for one or more concrete architect

↳ IOT reference architecture is a set of 3 architectural views - functional, information and deployment & operational.

↳ Security is one of functional groups [FG] of functional view

↳ FG for security has security functions between the applications and devices

↳ Security FG has 5 sets of functions which are required for providing security and privacy

↳ 5 functional components of security are defined in IOT reference architecture

1) Identity management 2) Authentications 3) Authorization
4) Key exchange and management 5) Trust and Reputation

↳ Figure lists the functions of security function group in functional view in IoT reference architecture

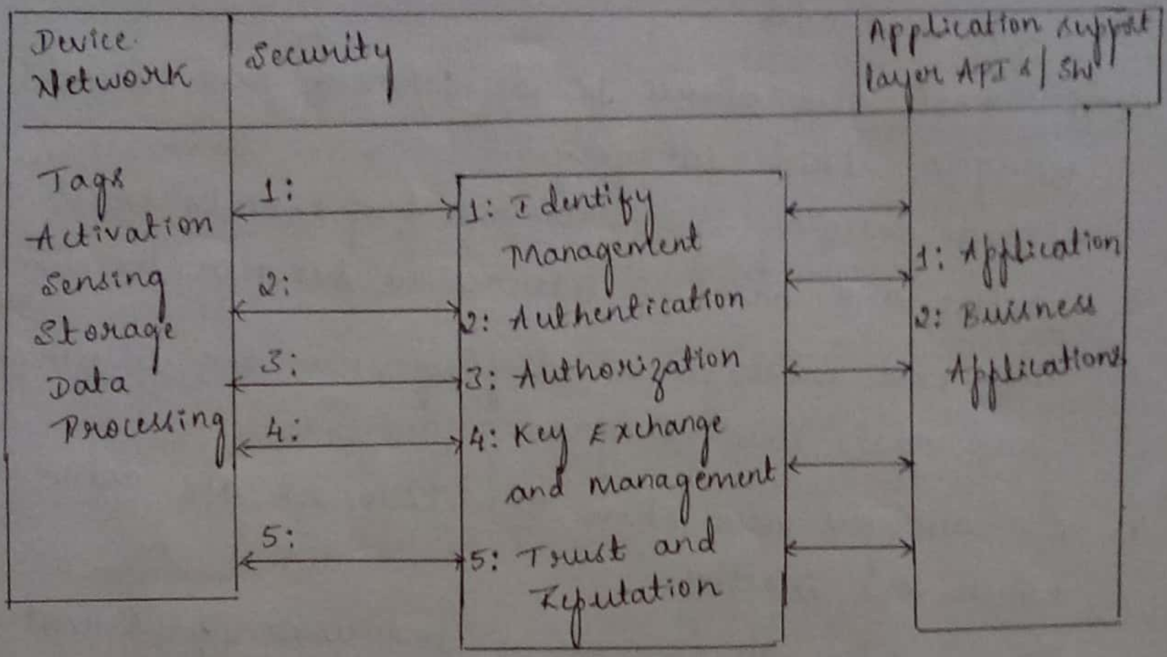


Fig:- Security function group components in functional view in IoT reference architecture.

→ Threat Analysis

↳ A threat analysis tool first generates threats and analyses system for threats.

↳ Threat analysis means discovering the security design flaws after specifying stride category, data flow diagram, elements between that interactions occuring during the stride and processes which are activated for analysis.

Example:- Microsoft Threat Modelling Tool 2014 - The model is designed for threat analysis with definitions of strides and elements. The interacting elements are processes, data store, flow, boundary or may be external specified elements in system

↳ The tool has 3 components

- Getting started guide
- create a model
- open a model.

↳ The tool also allows for definitions of new threats using a stride category.

Stride category generates a list of threats which are active and based on interactions between the elements

↳ The tool messages on display show the vulnerabilities and data flow diagram

↳ The analysis view show threats which are active and which are inactive

↳ Figure shows an example of tool usage for threat analysis during a web service interaction between the application and web

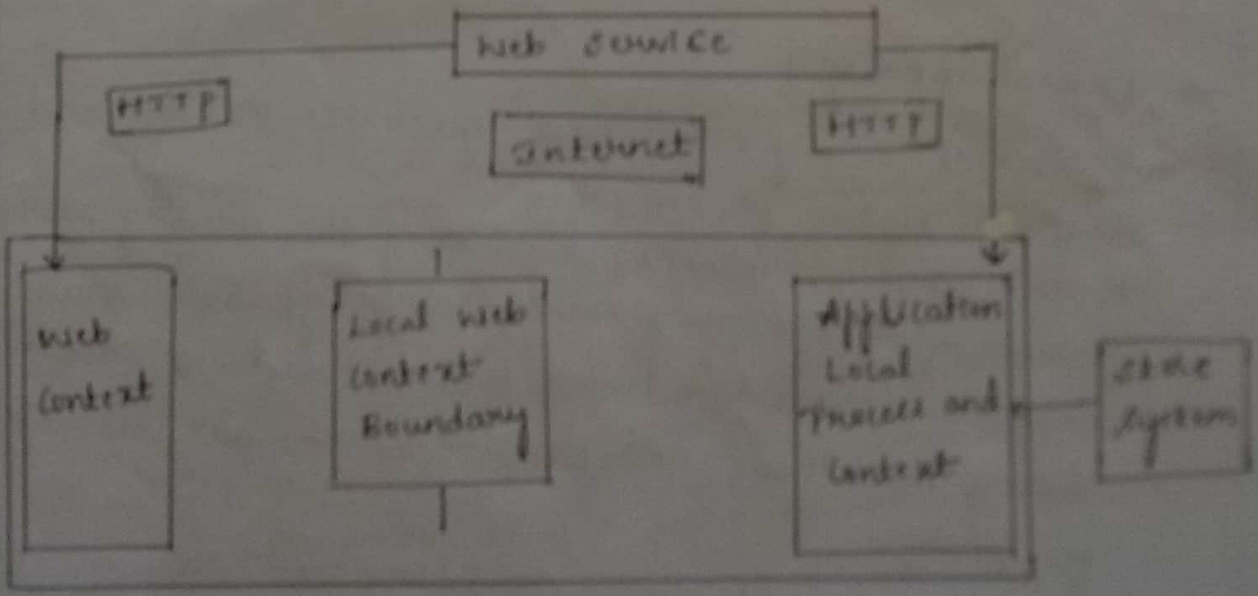

Web service
HTTP
Internet
HTTP
Web Content
Local Web Content Boundary
Application Local Threats and Content
File System

Fig: An application threat model in Microsoft threat modelling tool display when diagram item selected

IoT Security Tomography And Layered Attacker Model

→ Computational tomography means a computing method of producing a 3 dimensional picture of internal structures of an object, by observation and recording of the difference in effects on passing energy waves on those structures

→ Computational security in complex set of networks uses network tomography procedures of identifying the network vulnerabilities this helps to design an efficient attack strategies.

↳ Network tomography refers to study of vulnerabilities and security aspects for network monitoring in a complex system such as WSNs, RFIDs & IoT networks and allocating resources and making network reliable and secure

→ Monitoring individual nodes is not fast and it is impractical.
Network tomography helps in observing each network section and subsections

→ Security tomography means finding attack vulnerable sections/subsections from observations for behaviours using a finite number of objects or threats in complex set of subsystems.

→ Layered Attacker Model

6: Applications/services	vulnerabilities in applications/ services can be exploited through attacks such as SQL injection, where the developer has failed to ensure that user input is validated against defined schema.
5: Applications Support	
4: Transport	vulnerable ports.
3: Network	Packet sniffing and DoS attacks such as ping floods and ICMP attacks.
2: Data Adaptation	un-encrypted data store, tampering & sniffing
1: Physical cum data link layer	Insecure in protocols DHCP & STP, LAN node attack using MAC flooding & ARP poisoning physical & service disruption/ attacks on wireless networks

Fig: Layered attacker model and possible attacks using IETF six layer modified model for IoT/M2M

→ Following are the suggested solutions for reducing the attacks on layers [OSI modified 6 layered IoT architecture]

(*) Layer1 Attacks solution

Solution depends on device used.

Solution depends on provisioning of security uses

Example; link level provisioning of security uses BT LE link level AES CCM 128 authenticated encrypt-ion algorithm for confidentiality and authenti-cation and Zigbee at link level security using AES-CCM-128

(*) Layer 2 Attacks solution

↳ Programming the network switches to prevent internal node attacks during use of DHCP & spanning tree protocol (STP)

↳ Addition control may include ARP inspection, disabling unused ports and applying effective security on VLAN

(*) Layer 3 Attacks solution

use of temper resistant router, use of packet filtering and controlling routing messages and packets data between layers 3 and 4 through a firewall

(*) Layer 4 Attacks solution

↳ Port scanning method is a solution which identifies vulnerable ports.

↳ A solution is to open a network ports and effectively configure firewall and locking down ports only to those required.

↳ A solution is DTLS between layer 5 and 4. The DTLS has provisions for 3 types of security services - integrity, authentication and confidentiality.

↳ A solution is include SASL (simple Authentication and security layer) for security when using XMPP protocol

(*) Layer 5 and 6 Attacks solutions

↳ Above layer 4, application level attacks are result of poor coding practices.

↳ For example attacker injects the SQL input to extract data from database. When application fails to validate the injection the query extracts the data.

↳ web Application / service can use HTTP communication link. The features of S-HTTP [Secure HTTP] are

- Application level security (HTTP specific)
- Content privacy domain header
- Allows use of digital signatures and encryption various encryption options
- Server client negotiations
- Cryptographic scheme is a property assigned for link
- Specific algorithm is value assigned
- Direction specification is done, one way or two way security

→ CISCO suggested layered framework provides following solutions

- Layers 1-6 : Role based security
- Layers 1-4 : Anti temper and detection based security
- Layers 1-6 : Data protection and confidentiality
- Layers 1-6 : Ip protection.

(*) Layer 2 Attacks solution

↳ Programming the network switches to prevent internal node attacks during use of DHCP or spanning tree protocol (STP).

↳ Addition control may include ARP inspection, disabling unused ports and applying effective security on VLAN.

(*) Layer 3 Attacks solution

use of temper resistant router, use of packet filtering and controlling routing messages and packets data between layers 3 and 4 through a firewall.

(*) Layer 4 Attacks solution

↳ Port scanning method is a solution which identifies vulnerable ports.

↳ A solution is to open a network ports and effectively configure firewall and locking down ports only to those required.

↳ A solution is DTLS between layer 5 and 4. The DTLS has provisions for 3 types of security services - integrity, authentication and confidentiallty.

↳ A solution is include SASL (simple Authentication and security layer) for security when using XMPP protocol.

MODULE 4

4.1 Challenges for WSNs

Handling such a wide range of application types will hardly be possible with any single realization of a WSN. Nonetheless, certain common traits appear, especially with respect to the characteristics and the required mechanisms of such systems. Realizing these characteristics with new mechanisms is the major challenge of the vision of wireless sensor networks.

1.4.1 Characteristic requirements

The following characteristics are shared among most of the application examples

Type of service The service type rendered by a conventional communication network is evident – it moves bits from one place to another. For a WSN, moving bits is only a means to an end, but not the actual purpose. Rather, a WSN is expected to provide meaningful information and/or actions about a given task: Additionally, concepts like *scoping* of interactions to specific geographic regions or to time intervals will become important. Hence, new paradigms of using such a network are required, along with new interfaces and new ways of thinking about the service of a network.

Quality of Service Closely related to the type of a network's service is the quality of that service. Traditional quality of service requirements – usually coming from multimedia-type applications– like bounded delay or minimum bandwidth are irrelevant when applications are tolerant to latency or the bandwidth of the transmitted data is very small in the first place. In some cases, only occasional delivery of a packet can be more than enough; in other cases, very high reliability requirements exist. In yet other cases, delay *is* important when actuators are to be controlled in a real-time fashion by the sensor network. The packet delivery ratio is an insufficient metric; what is relevant is the amount and quality of information that can be extracted at given sinks about the observed objects or area.Therefore, adapted quality concepts like reliable detection of events or the approximation
quality of a, say, temperature map is important.

Fault tolerance Since nodes may run out of energy or might be damaged, or since the wireless communication between two nodes can be permanently interrupted, it is important that the WSN as a whole is able to tolerate such faults. To tolerate node failure, redundant deployment is necessary, using more nodes than would be strictly necessary if all nodes functioned correctly.

Lifetime In many scenarios, nodes will have to rely on a limited supply of energy (using batteries). Replacing these energy sources in the field is usually not practicable, and simultaneously, a WSN must operate at least for a given mission time or as long as possible. Hence, the **lifetime** of a WSN becomes a very important figure of merit. Evidently, an energy-efficient way of operation of the WSN is necessary. As an alternative or supplement to energy supplies, a limited power source (via power sources like solar cells, for example) might also be available on a sensor node. Typically, these sources are not powerful enough to ensure continuous operation but can provide some recharging of batteries. Under such conditions, the lifetime of the network should ideally be infinite. The lifetime of a network also has direct trade-offs against quality of service: investing more energy can increase quality but decrease lifetime. Concepts to harmonize these trade-offs are required.

The precise *definition of lifetime* depends on the application at hand. A simple option is to use the time until the first node fails (or runs out of energy) as the network lifetime. Other options include the time until the network is disconnected in two or more partitions, the time until 50% (or some other fixed ratio) of nodes have failed, or the time when for the first time a point in the observed region is no longer covered by at least a single sensor node (when using redundant deployment, it is possible and beneficial to have each point in space covered by several sensor nodes initially).

Scalability Since a WSN might include a large number of nodes, the employed architectures and protocols must be able scale to these numbers.

Wide range of densities In a WSN, the number of nodes per unit area – the *density* of the network – can vary considerably. Different applications will have very different node densities. Even within a given application, density can vary over time and space because nodes fail or move; the density also does not have to homogeneous in the entire network (because of imperfect deployment, for example) and the network should adapt to such variations.

Programmability Not only will it be necessary for the nodes to process information, but also they will have to react flexibly on changes in their tasks. These nodes should be programmable, and their programming must be changeable during operation when new tasks become important. A fixed way of information processing is insufficient.

Maintainability As both the environment of a WSN and the WSN itself change (depleted batteries, failing nodes, new tasks), the system has to adapt. It has to monitor its own health and statusto change operational parameters or to choose different trade-offs (e.g. to provide lower quality when energy resource become scarce). In this sense, the network has to maintain itself; it could also be able to interact with external maintenance mechanisms to ensure its extended operation at a required quality.

4.2Enabling technologies for wireless sensor networks

Building such wireless sensor networks has only become possible with some fundamental advances in enabling technologies. First and foremost among these technologies is the miniaturization of hardware. Smaller feature sizes in chips have driven down the power consumption of the basiccomponents of a sensor node to a level that the constructions of WSNs can be contemplated. This is particularly relevant to microcontrollers and memory chips as such, but also, the radio modems, responsible for wireless communication, have become much more energy efficient. Reduced chip size and improved energy efficiency is accompanied by reduced cost, which is necessary to make redundant deployment of nodes affordable.

Next to processing and communication, the actual sensing equipment is the third relevant technology. Here, however, it is difficult to generalize because of the vast range of possible sensors.

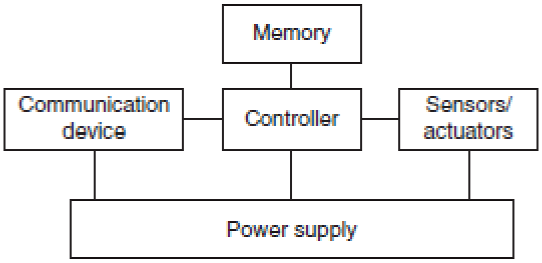
These three basic parts of a sensor node have to accompanied by power supply. This requires, depending on application, high capacity batteries that last for long times, that is, have only a negligible self-discharge rate, and that can efficiently provide small amounts of current. Ideally, a sensor node also has a device for **energy scavenging**, recharging the battery with energy gathered from the environment – solar cells or vibration-based power generation are conceivable options.

Such a concept requires the battery to be efficiently chargeable with small amounts of current, which is not a standard ability. Both batteries and energy scavenging are still objects of ongoing research.

The counterpart to the basic hardware technologies is software. The first question to answer here is the principal division of tasks and functionalities in a single node – the architecture of the operating system or runtime environment. This environment has to support simple retasking, cross-layer information exchange, and modularity to allow for simple maintenance. This software architecture on a single node has to be extended to a network architecture, where the division of tasks between nodes, not only on a single node, becomes the relevant question – for example, how to structure interfaces for application programmers. The third part to solve then is the question of how to design appropriate communication protocols.

4.3 Hardware components

A basic sensor node comprises five main components



Controller A controller to process all the relevant data, capable of executing arbitrary code.

Memory Some memory to store programs and intermediate data; usually, different types of memory are used for programs and data.

Sensors and actuators The actual interface to the physical world: devices that can observe or control physical parameters of the environment.

Communication Turning nodes into a network requires a device for sending and receiving information over a wireless channel.

Power supply As usually no tethered power supply is available, some form of batteries are necessary to provide energy. Sometimes, some form of recharging by obtaining energy from the environment is available as well (e.g. solar cells).

4.4 Energy consumption of sensor nodes

4.41 Operation states with different power consumption

As the previous section has shown, energy supply for a sensor node is at a premium: batteries have small capacity, and recharging by energy scavenging is complicated and volatile. Hence, the energy consumption of a sensor node must be tightly controlled. The main consumers of energy are the controller, the radio front ends, to some degree the memory, and, depending on the type, the sensors.

To give an example, consider the energy consumed by a microcontroller per instruction. A typical ball park number is about 1 nJ per instruction.To put this into perspective with the battery capacity numbers from Section 2.1.6, assume a battery volume of one cubic millimeter, which is about the maximum possible for the most ambitious visions of "smart dust". Such a battery could store about 1 J. To use such a battery to power a node even only a single day, the node must not consume continuously more than $1/(24 \cdot 60 \cdot 60)$ Ws/s ≈ 11.5 μW. No current controller, let alone an entire node, is able to work at such low-power levels.

One important contribution to reduce power consumption of these components comes from chip-level and lower technologies: Designing low-power chips is the best starting point for an energy-efficient sensor node. But this is only one half of the picture, as any advantages gained by such designs can easily be squandered when the components are improperly operated.

The crucial observation for proper operation is that most of the time a wireless sensor node has nothing to do. Hence, it is best to turn it off. Naturally, it should be able to wake up again, on the basis of external stimuli or on the basis of time. Therefore, completely turning off a node is not possible, but rather, its operational state can be adapted to the tasks at hand. Introducing and using multiple states of operation with reduced energy consumption in return for reduced functionality is the core technique for energy-efficient wireless sensor node. In fact, this approach is well known even from standard personal computer hardware, where, for example, the Advanced Configuration

and Power Interface (ACPI) introduces one state representing the fully operational machine and four sleep states of graded functionality/power consumption/wakeup time (time necessary to return to fully operational state). The term Dynamic Power Management (DPM) summarizes this field of work .

These modes can be introduced for all components of a sensor node, in particular, for controller, radio front end, memory, and sensors. Different models usually support different numbers of such sleep states with different characteristics; some examples are provided in the following sections. For a controller, typical states are "active", "idle", and "sleep"; a radio modem could turn transmitter, receiver, or both on or off; sensors and memory could also be turned on or off. The usual terminology is to speak of a "deeper" sleep state if less power is consumed.

While such a graded sleep state model is straightforward enough, it is complicated by the fact that transitions between states take both time and energy. The usual assumption is that the deeper the sleep state, the more time and energy it takes to wake up again to fully operational state (or to another, less deep sleep state). Hence, it may be worthwhile to remain in an idle state instead of going to deeper sleep states even from an energy consumption point of view.

Figure illustrates this notion based on a commonly used model. At time $t1$, the decision whether or not a component (say, the microcontroller) is to be put into sleep mode should be taken to reduce power consumption from $P\text{active}$ to $P\text{sleep}$. If it remains

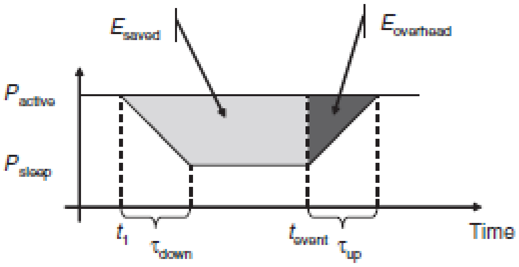
active and the next event occurs at time $t\text{event}$, then a total energy of $E\text{active} = P\text{active}(t\text{event} - t1)$ has be spent uselessly idling. Putting the component into sleep mode, on the other hand, requires a time τdown until sleep mode has been reached; as a simplification, assume that the average power consumption during this phase is $(P\text{active} + P\text{sleep})/2$. Then, $P\text{sleep}$ is consumed until $t\text{event}$. In total,

$\tau\text{down}(P\text{active} + P\text{sleep})/2 + (t\text{event} - t1 - \tau\text{down})P\text{sleep}$ energy is required in sleep mode as opposed to

$(t\text{event} - t1)P\text{active}$ when remaining active. The energy saving is thus

$E\text{saved} =(t\text{event} - t1)P\text{active} - (\tau\text{down}(P\text{active} + P\text{sleep})/2 +(t\text{event} - t1 - \tau\text{down})P\text{sleep})$

$E\text{overhead} = \tau\text{up}(P\text{active} + P\text{sleep})/2$



Once the event to be processed occurs, however, an additional overhead of is incurred to come back to operational state before the event can be processed, again making a simplifying assumption about average power consumption during makeup. This energy is indeed an overhead since no useful activity can be undertaken during this time. Clearly, switching to a sleep mode is only beneficial if

$E_{overhead} < E_{saved}$ or, equivalently, if the time to the next event is sufficiently large:

$$(t_{event} - t_1) > \frac{1}{2}\left(\tau_{down} + \frac{P_{active} + P_{sleep}}{P_{active} - P_{sleep}}\tau_{up}\right).$$

4.5 Operating systems and execution environments

4.5.1 Embedded operating systems

The traditional tasks of an operating system are controlling and protecting the access to resources (including support for input/output) and managing their allocation to different users as well as the support for concurrent execution of several processes and communication between these processes. These tasks are, however, only partially required in an embedded system as the executing code is much more restricted and usually much better harmonized than in a general-purpose system.

Also, as the description of the microcontrollers has shown, these systems plainly do not have the required resources to support a full-blown operating system. Rather, an operating system or an execution environment – perhaps the more modest term is the more appropriate one – for WSNs should support the specific needs of these systems. In particular, the need for energy-efficient execution requires support for energy management, for example, in the form of controlled shutdown of individual components or Dynamic Voltage Scaling (DVS) techniques. Also, external components – sensors, the radio modem, or timers – should be handled easily and efficiently, in particular, information that becomes available asynchronously (at any arbitrary point in time) must be handled.

All this requires an appropriate programming model, a clear way to structure a protocol stack, and explicit support for energy management – without imposing too heavy a burden on scarce system resources like memory or execution time. These three topics are treated in the following sections, with a case study completing the operating system considerations.

4.5.2 Programming paradigms and application programming interfaces

Concurrent Programming

One of the first questions for a programming paradigm is how to support concurrency. Such support for concurrent execution is crucial for WSN nodes, as they have to handle data communing from arbitrary sources – for example, multiple sensors or the radio transceiver – at arbitrary points in time. For example, a system could poll a sensor to decide whether data is available and process the data right away, then poll the transceiver to check whether a packet is available, and then immediately process the packet, and so on. Such a simple sequential model would run the risk of missing data while a packet is processed or missing a packet when sensor information is processed. This risk is particularly large if the processing of sensor data or incoming packets takes substantial amounts of time, which can easily be the case. Hence, a simple, sequential programming model is clearly insufficient.

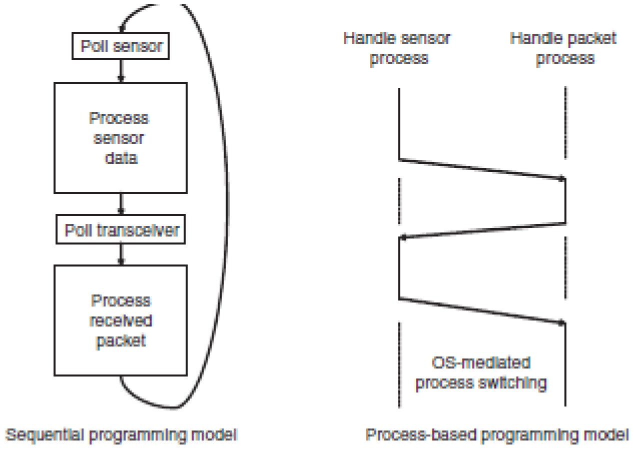
Process-based concurrency

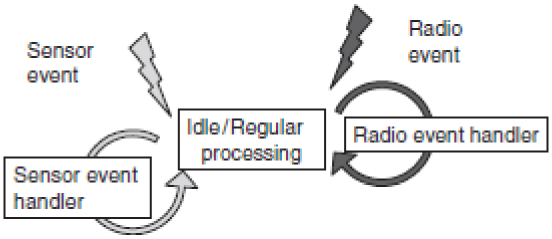
Most modern, general-purpose operating systems support concurrent (seemingly parallel) execution of multiple processes on a single CPU. Hence, such a process-based approach would be a first candidate to support concurrency in a sensor node as well; it is illustrated in (b) of Figure While indeed this approach works in principle, mapping such an execution model of concurrent processes to a sensor node shows, however, that there are some granularity mismatches : Equating individual protocol functions or layers with individual processes would entail a high overhead in switching from one process to another. This problem is particularly severe if often tasks have to be executed that are small with respect to the overhead incurred for switching between tasks – which is typically the case in sensor networks. Also, each process requires its own stack space in memory, which fits ill with the stringent memory constraints of sensor nodes.

Event-based programming

For these reasons, a somewhat different programming model seems preferable. The idea is to embrace the reactive nature of a WSN node and integrate it into the design of the operating system. The system essentially waits for any event to happen, where an event typically can be the availability of data from a sensor, the arrival of a packet, or the expiration of a timer. Such an event is then handled by a short sequence of instructions that only stores the fact that this event has occurred and stores the necessary information – for example, a byte arriving for a packet or the sensor's value – somewhere. The actual processing of this information is not done in these event handler routines, but separately, decoupled from the actual appearance of events. This **event-based programming** model is sketched in Figure

Such an event handler can interrupt the processing of any normal code, but as it is very simpleand short, it can be required to **run to completion** in all circumstances without noticeably disturbing other code. Event handlers cannot interrupt each other (as this would in turn require complicatedstack handling procedures) but are simply executed one after each other.





Event Based Programming Model

4.6 Design principles for WSNs

Appropriate QoS support, energy efficiency, and scalability are important design and optimization goals for wireless sensor networks. But these goals themselves do not provide many hints on how to structure a network such that they are achieved. A few basic principles have emerged, which can be useful when designing networking protocols; the description here follows partially references Nonetheless, the general advice to always consider the needs of a concrete application holds here as well – for each of these basic principles, there are examples where following them would result in inferior solutions.

4.6.1 Distributed organization

Both the scalability and the robustness optimization goal, and to some degree also the other goals, make it imperative to organize the network in a distributed fashion. That means that there should be no centralized entity in charge – such an entity could, for example, control medium access ormake routing decisions, similar to the tasks performed by a base station in cellular mobile networks.

The disadvantages of such a centralized approach are obvious as it introduces exposed points of failure and is difficult to implement in a radio network, where participants only have a limited communication range. Rather, the WSNs nodes should cooperatively organize the network, using distributed algorithms and protocols. **Self-organization** is a commonly used term for this principle. When organizing a network in a distributed fashion, it is necessary to be aware of potential short comings of this approach. In many circumstances, a centralized approach can produce solutions that perform better or require less resources (in particular, energy). To combine the advantages, one possibility is to use centralized principles in a localized fashion by dynamically electing, out of the set of equal nodes, specific nodes that assume the responsibilities of a centralized agent, for

example, to organize medium access. Such elections result in a hierarchy, which has to be dynamic: The election process should be repeated continuously lest the resources of the elected nodes be overtaxed, the elected node runs out of energy, and the robustness disadvantages of such – even only localized – hierarchies manifest themselves. The particular election rules and triggering conditions for reelection vary considerably, depending on the purpose for which these hierarchies are used.

4.6.2 In-network processing

When organizing a network in a distributed fashion, the nodes in the network are not only passing on packets or executing application programs, they are also actively involved in taking decisions about how to operate the network. This is a specific form of information processing that happens in the network, but is limited to information about the network itself. It is possible to extend this concept by also taking the concrete data that is to be transported by the network into account in this information processing, making **in-network processing** a first-rank design principle.

Several techniques for in-network processing exist, and by definition, this approach is open to an arbitrary extension – any form of data processing that improves an application is applicable

Aggregation

Perhaps the simplest in-network processing technique is aggregation. Suppose a sink is interested in obtaining periodic measurements from all sensors, but it is only relevant to check whether the average value has changed, or whether the difference between minimum and maximum value is too big. In such a case, it is evidently not necessary to transport are readings from all sensors to the sink, but rather, it suffices to send the average or the minimum and maximum value The name **aggregation** stems from the fact that in nodes intermediate between sources and sinks, information is aggregated into a condensed form out of information provided by nodes further away from the sink (and potentially, the aggregator’s own readings).

Distributed source coding and distributed compression

Aggregation condenses and sacrifices information about the measured values in order not to have to transmit all bits of data from all sources to the sink. Is it possible to reduce the number of transmitted bits (compared to simply transmitting all bits) but still obtain the *full* information about all sensor readings at the sink?

While this question sounds surprising at first, it is indeed possible to give a positive answer. It is related to the coding and compression problems known from conventional networks, where a lot of effort is invested to encode, for example, a video sequence, to reduce the required bandwidth

The problem here is slightly different, in that we are interested to encode the information provided by several sensors, not just by a single camera; moreover, traditional coding schemes tend to put effort into the encoding, which might be too computationally complex for simple sensor nodes.

How can the fact that information is provided by multiple sensors be exploited to help in coding? If the sensors were connected and could exchange their data, this would be conceivable (using relatively standard compression algorithms), but of course pointless. Hence, some implicit, joint information between two sensors is required. Recall here that these sensors are embedded in a physical environment – it is quite likely that the readings of adjacent sensors are going to be quite similar; they are *correlated*. Such **correlation** can indeed be exploited such that not simply the sum of the data must be transmitted but that overhead can be saved here.

Distributed and collaborative signal processing

The in-networking processing approaches discussed so far have not really used the ability for *processing* in the sensor nodes, or have only used this for trivial operations like averaging or finding the maximum. When complex computations on a certain amount of data is to be done,it can still be more energy efficient to compute these functions on the sensor nodes despite their limited processing power, if in return the amount of data that has to be communicated can be reduced.

An example for this concept is the distributed computation of a Fast Fourier Transform (FFT). Depending on where the input data is located, there are different algorithms available to compute an FFT in a distributed fashion, with different trade-offs between local computation complexity and the need for communication. In principle, this is similar to algorithm design for parallel computers. However, here not only the latency of communication but also the energy consumption of communication and computation are relevant parameters to decide between various algorithms. Such distributed computations are mostly applicable to signal processing type algorithms.

Mobile code/Agent-based networking

With the possibility of executing programs in the network, other programming paradigms or computational models are feasible. One such model is the idea of **mobile code** or **agent-based networking**.

The idea is to have a small, compact representation of program code that is small enough to be sent from node to node. This code is then executed locally, for example, collecting measurements, and then decides where to be sent next. This idea has been used in various environments; a classic example is that of a software agent that is sent out to collect the best possible travel itinerary by hopping from one travel agent's computer to another and eventually returning to the user who has posted this inquiry. There is a vast amount of literature available on mobile code/software agents in general. A newer take on this approach is to consider biologically inspired systems, in particular, the **swarm intelligence** of groups of simple entities, working together to reach a common goal.

4.7 Service interfaces of WSNs

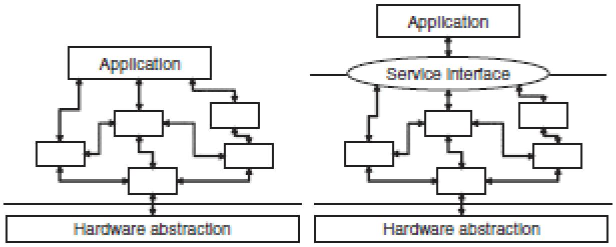
4.7.1 Structuring application/protocol stack interfaces

Component-based operating system and protocol stack already enables one possibility to treat an application: It is just another component that can directly interact with other components using whatever interface specification exists between them (e.g. the command/event structure of TinyOS). The application could even consist of several components, integrated at various places into the protocol stack. This approach has several advantages: It is streamlined with the overall protocol structure, makes it easy to introduce application-specific code into the WSN at various levels, and does not require the definition of an abstract, specific service interface. Moreover, such a tight integration allows the application programmer a very fine-grained control over which protocols (which components) are chosen for a specific task; for example, it is possible to select out of different routing protocols the one best suited for a given application by accessing this component's services.

But this generality and flexibility is also the potential downside of this approach. The allowing of the application programmer to mess with protocol stacks and operating system internals should not be undertaken carelessly. In traditional networks such as the Internet, the application programmer can access the services of the network via a commonly accepted interface: sockets. This interface makes clear provisions on how to handle connections, how to send and receive packets,and how to inquire about state information of the network. This clarity is owing to the evident tasksthat this interface serves – the exchange of packets with one (sometimes, several) communication peers. Therefore, there is the design choice between treating the application as just another component or designing a service interface that makes all components, in their entirety, accessible in a standardized fashion. These two options are outlined by Figure. A service interface would allow to raise the level of abstraction with which an application can interact with the WSN – instead of having to specify which value to read from which particular sensor, it might be desirable to provide an application with the possibility to express sensing tasks in terms that are close to the semantics

of the application. In this sense, such a service interface can hide considerable complexity and is actually conceivable as a "middleware" in its own right.

Clearly, with a tighter integration of the application into the protocol stack, a broader optimization spectrum is open to the application programmer. On the downside, more experience will be necessary than when using a standardized service interface. The question is therefore on the one hand the price of standardization with respect to the potential loss of performance and on the other hand, the complexity of the service interface. In fact, the much bigger complexity and variety of communication patterns in wireless sensor networks compared to Internet networks makes a more expressive and potentially complex service interface necessary. To better understand this trade-off, a clearer understanding of expressibility requirements of such an interface is necessary.



4.7.2 Expressibility requirements for WSN service interfaces

The most important functionalities that a service interface should expose include:

- Support for simple request/response interactions: retrieving a measured value from some sensor or setting a parameter in some node. This is a synchronous interaction pattern in the sense that the result (or possibly the acknowledgment) is expected immediately. In addition, the responses can be required to be provided periodically, supporting periodic measurement-type applications.
- Support for asynchronous event notifications: a requesting node can require the network to inform it if a given condition becomes true, for example, if a certain event has happened. This is an asynchronous pattern in the sense that there is no a priori relationship between the time the request is made and the time the information is provided. This form of asynchronous requests should be accompanied by the possibility to cancel the request for information. It can be further refined by provisions about what should happen after the condition becomes true; a typical example is to request periodic reporting of measured values after an event.
- For both types of interactions, the addressees should be definable in several ways. The simplest option is an explicit enumeration of the single or multiple communication peers to whom a (synchronous or asynchronous) request is made – this corresponds to the peer address in a socket communication.
- Location – all nodes that are in a given region of space belong to a group.
- Observed value – all nodes that have observed values matching a given predicate belong to a group. An example would be to require the measured temperature to be larger than 20◦C. Along with these groups, the usual set-theoretic operations of intersection, union, or difference between groups should be included in the service interface as well. Because of this natural need for a service interface semantics that corresponds to the publish/ subscribe concept, this approach is a quite natural, but not the only possible, fit with WSNs.
- In-networking processing functionality has to be accessible. For an operation that accesses an entire group of nodes, especially when reading values from this group (either synchronously or asynchronously), it should be possible to specify what kind of in-network processing should be applied to it. In particular, processing that modifies the nature of the result (i.e., data fusion)

must be explicitly allowed by the requesting application. In addition, it can be desirable for an application to be able to infuse its own in-network processing functions into the network. For example, a new aggregation function could be defined or a specific

mobile agent has to be written by the application programmer anyway.

In-network processing and application-specific code may also be useful to detect **complex events**: events that cannot be detected locally, by a single sensor, but for which data has to be exchanged between sensors.

• Related to the specification of aggregation functions is the specification of the required accuracy of a result. This can take on the form of specifying bounds on the number of group members that should contribute to a result, or the level of compression that should be applied. Hand in hand with required accuracy goes the acceptable energy expenditure to produce a given piece of information.

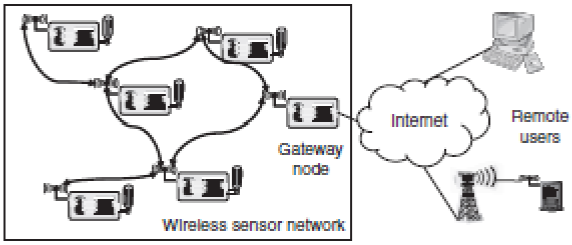
• Timeliness requirements about the delivery of data is a similar aspect. For example, it may be possible to provide a result quickly but at higher energy costs (e.g. by forcing nodes to wakeup earlier than they would wake up anyway) or slowly but at reduced energy costs (e.g. by piggy-backing information on other data packets that have to exchanged anyway).

4.8 Gateway concepts

4.8.1 The need for gateways

For practical deployment, a sensor network only concerned with itself is insufficient. The network rather has to be able to interact with other information devices, for example, a user equipped with a PDA moving in the coverage area of the network or with a remote user, trying to interact with thesensor network via the Internet (the standard example is to read the temperature sensors in one’s home while traveling and accessing the Internet via a wireless connection). Figure shows this networking scenario. To this end, the WSN first of all has to be able to exchange data with such a mobile device or with some sort of gateway, which provides the physical connection to the Internet. This is relatively straight forward on the physical, MAC, and link layer – either the mobile device/the gateway is equipped with a radio transceiver as used in the WSN, or some (probably not all) nodes in the WSN

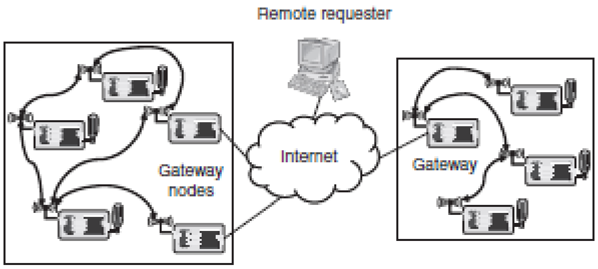
support standard wireless communication technologies such as IEEE 802.11. Either option can be advantageous, depending on the application and the typical use case. Possible trade-offs include the percentage of multitechnology sensor nodes that would be required to serve mobile users in comparison with the overhead and inconvenience to fit WSN transceivers to mobile devices like PDAs.

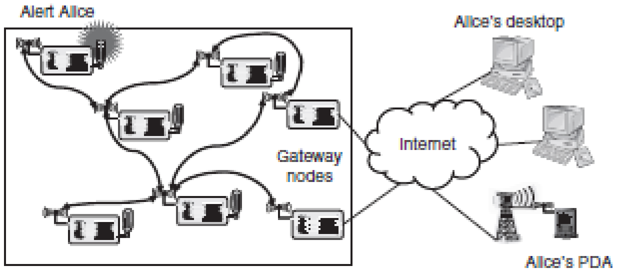


WSN to Internet communication

Assume that the initiator of a WSN–Internet communication resides in the WSN– for example, a sensor node wants to deliver an alarm message to some Internet host. The first problem to solve is akin to ad hoc networks, namely, how to find the gateway from within the network. Basically, a routing problem to a node that offers a specific service has to be solved, integrating routing and service discovery. If several such gateways are available, how to choose between them? In particular, if not all Internet hosts are reachable via each gateway or at least if some gateway should be preferred for a given destination host? How to handle several gateways, each capable of IP networking, and the communication among them? One option is to build an IP overlay network on top of the sensor network.

How does a sensor node know to which Internet host to address such a message? Or even worse, how to map a semantic notion ("Alert Alice") to a concrete IP address? Even if the sensor node does not need to be able to process the IP protocol, it has to include sufficient information (IP address and port number, for example) in its own packets; the gateway then has to extract this information and translate it into IP packets. An ensuing question is which source address to use here – the gateway in a sense has to perform tasks similar to that of a Network Address Translation (NAT).





Internet to WSN communication

The case of an Internet-based entity trying to access services of a WSN is even more challenging. This is fairly simple if this requesting terminal is able to directly communicate withthe WSN, for example, a mobile requester equipped with a WSN transceiver, and also has all the necessary protocol components at its disposal. In this case, the requesting terminal can be a direct part of the WSN and no particular treatment is necessary. The more general case is, however, a terminal "far away" requesting the service, not immediately able to communicate with any sensor node and thus requiring the assistance of a gateway node.

First of all, again the question of service discovery presents itself – how to find out that there actually is a sensor network in the desired location, and how to find out about the existence of a gateway node? Once the requesting terminal has obtained this information, how to access the actual services?Clearly, addressing an individual sensor (like addressing a communication peer in a traditional Internet application) both goes against the grain of the sensor network philosophy where an individual sensor node is irrelevant compared to the data that it provides and is impossible if a sensor node does not even have an IP address. The requesting terminal can instead send a properly formatted request to this gateway, which acts as an application-level gateway or a proxy for the individual/set of sensor nodes that can answer this request; the gateway translates this request into the proper intrasensor network protocol interactions. This assumes that there is an application-level protocol that a remote requester and gateway can use and that is more suitable for communication over the Internet than the actual sensor network protocols and that is more convenient for the remote terminal to use. The gateway can then mask, for example, a data-centric data exchange within the network behind an identity-centric exchange used in the Internet.

It is by no means clear that such an application-level protocol exists that represents an actual simplification over just extending the actual sensor network protocols to the remote terminal, but there are some indications in this direction. For example, it is not necessary for the remote terminalto be concerned with maintaining multihop routes in the network nor should it be considered as “justanother hop” as the characteristics of the Internet connection are quite different from a wireless hop.

In addition, there are some clear parallels for such an application-level protocol with so-called Web Service Protocols, which can explicitly describe services and the way they can be accessed. The Web Service Description Language (WSDL) ,in particular, can be a promising starting point for extension with the required attributes for WSN service access – for example, required accuracy, energy trade-offs, or data-centric service descriptions. Moreover, the question arises as to how to integrate WSN with general middleware architectures or how to make WSN services accessible from, say, a standard Web browser (which should be an almost automatic by-product of using WSDL and related standards in the gateway). However, research here is still in its early infancy.

Module – 5

Module-5 covered by chapters 4, 5, 7, 10 and 11 from the prescribed text book "*Protocols and Architectures for Wireless Sensor Networks*" by Holger Karl and Andreas Willig

Chapter 4: Physical Layer (Chapter 4: 4.3, Page 103 - 108)

- Physical Layer and Transceiver Design Considerations in WSN

Chapter 5: MAC Protocols (chapter 5: 5.1.3, 5.2, 5.3, 5.4, Page 119 - 139)

- MAC protocols for wireless sensor networks
- Low Duty Cycle Protocols and Wakeup Concepts
  - *S-MAC*
  - *The Mediation Device Protocol*
  - *Wakeup Radio Concepts*
- Contention based protocols : CSMA, PAMAS
- Schedule based protocols: LEACH, SMACS, TRAMA

Chapter 7: Naming and addressing. (Chapter 7: 7.1, 7.2, 7.3, Page182- 189)

- Fundamentals
- Address and name management in wireless sensor networks
- Assignment of MAC addresses

Chapter 10: Topology control (Chapter 10: 10.4, Page 274- 284)

- Hierarchical networks by clustering

Chapter 11: Routing Protocols for WSN. (Chapter 11: 11.3, 11.5, Page 295-304 & Page 316-327)

- Energy-Efficient Routing
- Geographic Routing

Chapter 4: Physical Layer of WSN

Introduction: The physical layer is mostly concerned with modulation and demodulation of digital data; this task is carried out by transceivers. In sensor networks, the challenge is to find modulation schemes and transceiver architectures that are simple, low cost, but still robust enough to provide the desired service.

Physical Layer and Transceiver Design Considerations in WSN: Some of the most crucial points influencing physical layer design in wireless sensor networks are:

- *Low power consumption.*
- *Small transmit power and thus a small transmission range.*
- *Operate at Due to low duty cycle.*
- *Keep most of the hardware should be switched off or operated in a low-power standby mode most of the time.*

- *Comparably low data rates, on the order of tens to hundreds kbps required.*
- *Low implementation complexity and costs.*
- *Low degree of mobility.*
- *A small form factor for the overall node.*

Energy usage profile: The choice of a small transmit power leads to an energy consumption profile different from other wireless devices like cell phones. The radiated energy is small but the overall transceiver consumes much more energy than is actually radiated, for example. for the Mica motes, 21 mW are consumed in transmit mode and 15 mW in received mode for a radiated power of 1 mW.

- Strive for good power efficiency at low transmission power
  - *Some amplifiers are optimized for efficiency at high output power*
  - *To radiate 1 mW, typical designs need 30-100 mW to operate the transmitter (RF and CMOS transceiver).*
- Receiver can use as much or more power as transmitter at these power levels.
- Many practical transmitter designs have efficiencies below 10% at low radiated power.
- For small transmit powers, the transmit and receive modes consume more or less the same power; therefore it is important to put the transceiver into sleep state instead of idle state.
- This rises the problem of startup energy/ startup time which a transceiver has to spend upon waking up from sleep mode, for example, to ramp up phase locked loops or voltage controlled oscillators; during this startup time, no transfer of data is possible; for example, the µAMPS-1 transceiver needs 466 µs and a power dissipation of 58 mW; therefore, going into sleep mode is unfavorable when the next wakeup comes fast.
- For the WIN nodes, 1500 to 2700 instructions can be executed per transmitted bit.
- Computation is cheaper than communication: the ratio is hundreds to thousands of instructions/ 1 transmitted bit.

Choice of modulation scheme: The choice of modulation scheme depends on several aspects, including technological factors, packet size, and target error rate and channel error model. The higher the data rate offered by a transceiver/modulation, the smaller the time needed to transmit a given amount of data and, consequently, the smaller the energy consumption. Power consumption can depend on modulation scheme. The power consumption of a modulation scheme depends much more on the symbol rate than on the data rate; it leads to desire of high data rates at low symbol rates which ends to m – ary modulation schemes; trade – offs:

- M – ary modulation schemes require more hardware than 2 – ary schemes
- M – ary modulation schemes require for increasing m an increased Eb/N0 ratio

- Generally, in WSN applications most packets are short; for them, the startup time dominates overall energy consumption making the other efforts irrelevant; Dynamic modulation scaling is necessary(see Table 4.3)

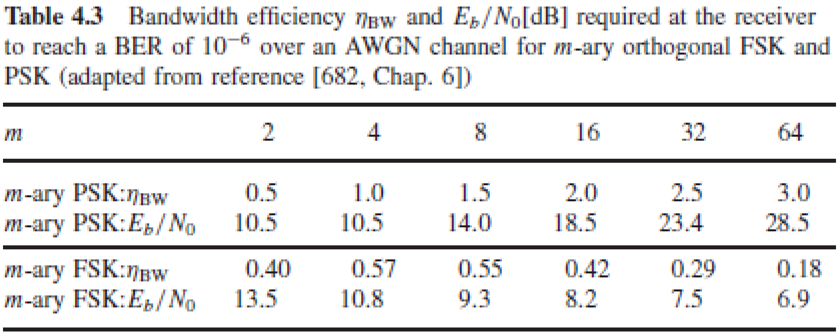
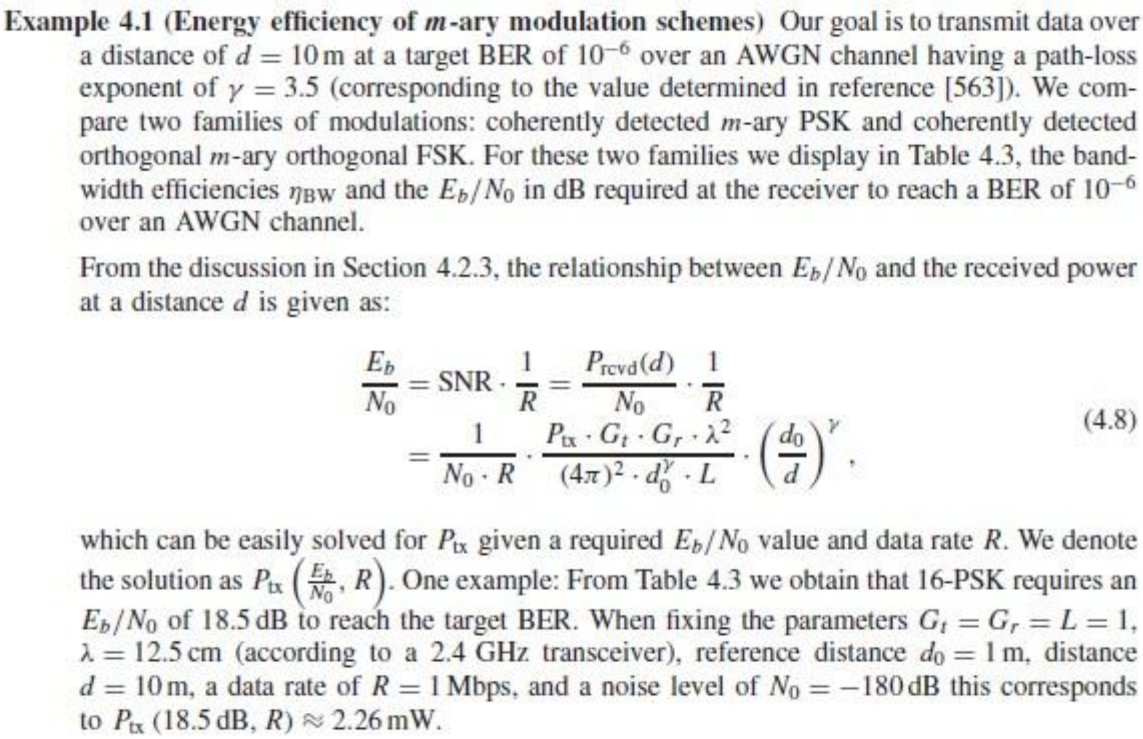


Table 4.3 Bandwidth efficiency η_{BW} and E_b/N_0[dB] required at the receiver to reach a BER of 10^{-6} over an AWGN channel for m-ary orthogonal FSK and PSK (adapted from reference [682, Chap. 6])

m	2	4	8	16	32	64
m-ary PSK:η_{BW}	0.5	1.0	1.5	2.0	2.5	3.0
m-ary PSK:E_b/N_0	10.5	10.5	14.0	18.5	23.4	28.5
m-ary FSK:η_{BW}	0.40	0.57	0.55	0.42	0.29	0.18
m-ary FSK:E_b/N_0	13.5	10.8	9.3	8.2	7.5	6.9



Example 4.1 (Energy efficiency of *m*-ary modulation schemes) Our goal is to transmit data over a distance of $d = 10$ m at a target BER of 10^{-6} over an AWGN channel having a path-loss exponent of $\gamma = 3.5$ (corresponding to the value determined in reference [563]). We compare two families of modulations: coherently detected m-ary PSK and coherently detected orthogonal m-ary orthogonal FSK. For these two families we display in Table 4.3, the bandwidth efficiencies η_{BW} and the E_b/N_0 in dB required at the receiver to reach a BER of 10^{-6} over an AWGN channel.

From the discussion in Section 4.2.3, the relationship between E_b/N_0 and the received power at a distance d is given as:

$$\begin{aligned}\frac{E_b}{N_0} &= \mathrm{SNR} \cdot \frac{1}{R} = \frac{P_{\mathrm{rcvd}}(d)}{N_0} \cdot \frac{1}{R} \\ &= \frac{1}{N_0 \cdot R} \cdot \frac{P_{\mathrm{tx}} \cdot G_t \cdot G_r \cdot \lambda^2}{(4\pi)^2 \cdot d_0^{\gamma} \cdot L} \cdot \left(\frac{d_0}{d}\right)^{\gamma},\end{aligned} \tag{4.8}$$

which can be easily solved for P_{tx} given a required E_b/N_0 value and data rate R. We denote the solution as $P_{\mathrm{tx}}\left(\frac{E_b}{N_0}, R\right)$. One example: From Table 4.3 we obtain that 16-PSK requires an E_b/N_0 of 18.5 dB to reach the target BER. When fixing the parameters $G_t = G_r = L = 1$, $\lambda = 12.5$ cm (according to a 2.4 GHz transceiver), reference distance $d_0 = 1$ m, distance $d = 10$ m, a data rate of $R = 1$ Mbps, and a noise level of $N_0 = -180$ dB this corresponds to P_{tx} (18.5 dB, R) $\approx$ 2.26 mW.

Dynamic modulation scaling

- To achieve delay constraints or high throughput need higher modulation schemes.
- *Problem*: higher modulation levels need higher radiated energy.
- *Solution*: Consider methods to adapt the modulation scheme to the current situation. Such an approach, called “dynamic modulation scaling”.
- Both energy per bit and delay per bit are minimized for the maximum symbol rate.

Antenna considerations

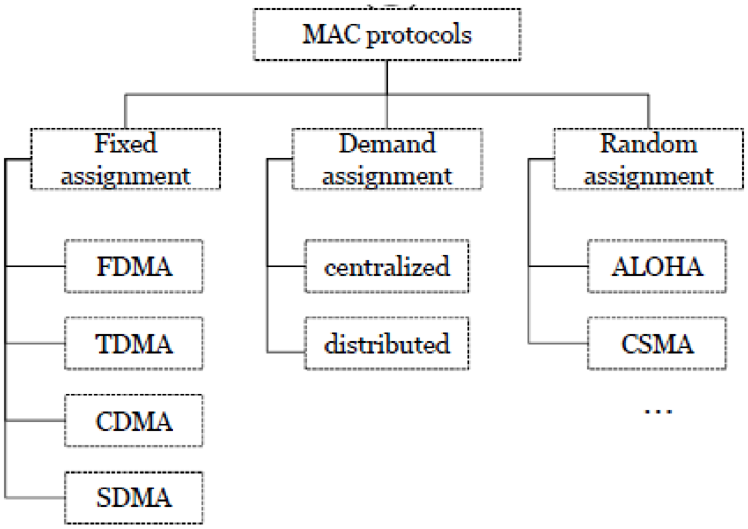
- Sensor nodes size restricts the size and the number of antennas and antenna diversity.
- If the antenna is much smaller than the carrier’s wavelength, it is hard to achieve good antenna efficiency and transmitted energy must increase.

- In case of multiple antennas, they should be spaced apart at least 40 – 50% of the wavelength used to achieve good effects; for ex. for 2.4 GHz, a spacing of 5 – 6 cm between the antennas is necessary, which is difficult to be accepted.
- Radio waves emitted from antennas close to the ground, typical in some applications, are faced with higher path – loss coefficients than the common value of α = 2; a typical value, considering the obstacles too, is α = 4;
- Nodes randomly scattered on the ground, deployed from an aircraft, will land in random orientations, with the antennas facing the ground or being otherwise obstructed; this can lead to nonisotropic propagation of the radio wave, with considerable differences in the strength of the emitted signal in different directions.

Chapter 5: MAC Protocols

The fundamental task of any MAC protocol is to regulate the access of a number of nodes to a shared medium.

Important classes of MAC protocols



A huge number of (wireless) MAC protocols have been devised. They can be roughly classified into the following classes

1. **Fixed assignment protocols**: *In this class of protocols, the available resources are divided between the nodes such that the resource assignment is long term and each node can use its resources exclusively without the risk of collisions. Typical protocols of this class are TDMA, FDMA, CDMA, and SDMA..*
2. **Demand assignment protocols**: *Here protocols exclusive allocation of resources to nodes is made on a short-term basis, typically the duration of a data burst. This class of protocols can be broadly subdivided into centralized and distributed protocols. Example HIPERLAN/2 protocol, DQRUMA, the MASCARA protocol and polling schemes.*
3. **Random access protocols**: The nodes are uncoordinated, and the protocols operate in a fully distributed manner. Examples ALOHA, CSMA, CSMA-CD, CSMA-CA.

MAC protocols for wireless sensor networks: The fundamental task of any MAC protocol is to regulate the access of a number of nodes to a shared medium. In this section, narrow down the specific requirements and design considerations for MAC protocols in wireless sensor networks.

- ***Requirements:*** New requirements are imposed by WSNs, the main one being the energy efficiency; Typical performance figures like fairness, throughput or delay have a minor role in SNs; fairness is not important since the nodes in a WSN do not represent individuals competing for bandwidth, but they collaborate to achieve a common goal; Other important requirements:
  - *Scalability*: is obvious when considering very dense sensor networks with dozens or hundreds of nodes in mutual range
  - *Robustness against frequent topology changes*: are caused by nodes powering down temporarily to replenish their batteries by energy scavenging, mobility, deployment of new nodes or death of existing nodes.
  - *Low complexity operation*: Sensor nodes are simple, cheap and have limited hardware resources; therefore computational expensive operations like complex scheduling algorithms should be avoided. Very tight time synchronization would require frequent resynchronization of neighboring nodes, which can consume significant energy.

- ***Energy problems on the MAC layer******: Transmitting is costly, receive costs often are as transmit costs, idling can be cheaper but also about as expensive as receiving and sleeping costs almost nothing but results in a deaf node. The following are the major energy waste problems are related to MAC protocols:
  1. **Collision:** Collision occurs when two sensor nodes transmit their packets at the same time. Retransmissions of the packets increase both energy consumption and delivery latency.
  2. **Overhearing:** Overhearing occurs when a sensor node receives packets that are destined for other nodes. Overhearing such packets results in unnecessary waste of energy and such waste can be very large when traffic load is heavy and node density is high.
  3. **Idle Listening**: The node will stay in an idle state for a long time, which results in a large amount of energy waste. There are reports that idle listening consumes 50 – 100% of the energy required for receiving data traffic.
  4. **Control Overhead:** A MAC protocol requires sending, receiving, and listening to a certain necessary control. Which also consumes energy not for data communication

Low duty cycle protocols and wakeup concepts***

Low duty cycle protocols try to avoid spending much time in the idle state and to reduce the communication activities of a sensor node to a minimum; In an ideal case, the sleep state is left only when a node is about to transmit or receive packets; In several protocols, a periodic wakeup scheme is used; one flavor is the cycled receiver approach. It is illustrated in Figure 5.4. In this approach, nodes spend most of their time in the sleep mode and wake up periodically to receive packets from other nodes.

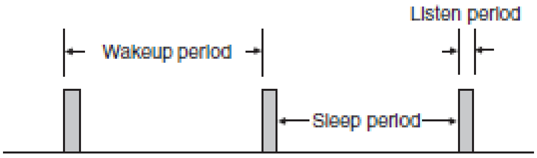


Figure 5.4 Periodic wakeup scheme

- A node A listens onto the channel during its listen period and goes back into sleep mode when no other node communicates with it.
- A potential transmitter B must know about A's listen periods and send its packet at the right time; this is a so-called *rendezvous*.
- A *rendezvous* can be implemented by letting node A to send a beacon at the beginning of its listen period or letting node B to send frequent request packets until one of them is sensed by node A.
- If node A wants to send a packet, it must also know the target's listen period.
- A wakeup period = sleep period + listen period.
- The node's duty cycle = listen period/ wakeup period.
- ***Observations:***
  - By choosing a small duty cycle, the transceiver is in sleep mode most of the time, avoiding idle listening and conserving energy.
  - By choosing a small duty cycle, the traffic directed from neighboring nodes to a given node concentrates on a small time window (the listen period) and in heavy load situations significant competition can occur.
  - Choosing a long sleep period induces a significant per – hop latency, since a prospective transmitter node has to wait an average of half a sleep period before the receiver can accept packets; in the multihop case, the per – hop latencies add up and create significant end – to – end latencies.
  - Sleep phases should not be too short lest the start – up costs outweigh the benefits;
  - In other protocols, there is also a periodic wakeup but nodes can both transmit and receive during their wakeup phases; when all nodes have their wakeup phases at the same time, there is no need for a *rendezvous*.

Sparse topology and energy management (STEM)***

The goal of topology management is to coordinate the sleep transitions of all the nodes, while ensuring adequate network connectivity, such that data can be forwarded efficiently to the data sink. This protocol tries to save energy due to idle listening. This protocol does not provide a complete MAC protocol, however a MAC protocol can be used along with it to give a complete MAC protocol.

- This protocol proposes to use two channels
    1. *Wake up channel:* Wake up channel is used to inform the receiver that a transmitter wants to transmit data to it
    2. *Data channel*: Data channel is used to transmit data, underlying MAC protocol is used for this data transmission.
- STEM is designed for applications which wait for an event and report that event, when the event takes place. In other words STEM is applicable where nodes have two states
    1. *Monitor state*, where nodes monitor and no event takes place.
    2. *Transfer state*: where event is detected and data has to be transmitted.
- On the Wake up channel time is divided into *sleep period* and *listen period*, these together are called wake up period. This can be seen in the diagram below

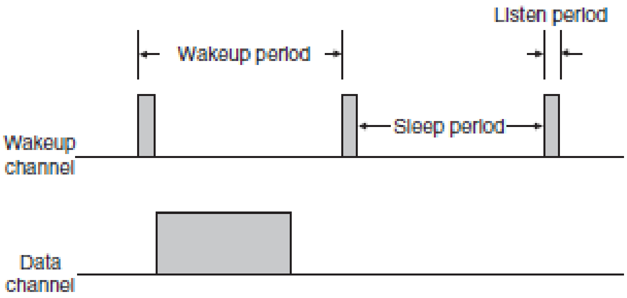


Figure 5.5 STEM duty cycle for a single node [742, Fig. 3]

- ***Channel in STEM***

    There will be two transceivers in every sensor node. One is for wake up channel and other is for data channel.

    - *The transceiver of the data channel:* It will always be in sleep mode until some has to received or transmitted by the node
    - *The transceiver of the wake up channel:* It will be sleep in sleep period and be active in listen period. During the listen period the wake up channel receiver is switched on and the node waits to check if any data is to be received if so the data channel transceiver is switched on or else the wake up channel transceiver goes to sleep.
    - The STEM protocol has two flavors; they are STEM-B and STEM-T.
    - *In STEM-B:* a node which wishes to transmit to another node, sends beacons periodically on the wake up channel. This beacon contains the address of transmitter and receiver. The receiver detects the beacons during its listen period and acknowledges the transmitter, and then both shift to data channel and exchange data.

- *In STEM-T*: The transmitter sends busy tone on wake up channel for a long enough time to hit the receivers listen period. As there is no address of the receiver in the busy tone all neighboring nodes which hear busy shift to data channel, however on receiving the data, only the node for which the data was intended will reply and all others go back to sleep. ***Advantages***: Lower Latency. ***Disadvantages***: More complex, High energy consumption

---

S-MAC ***

AC stands for Sensor MAC. This protocol tries to reduce energy consumption due to overhearing, idle listening and collision. S-MAC adopts a duty-cycle approach. In this protocol also every node has two states, *sleep sta*te and *active state*. Unlike STEM, S-MAC does not use two channels. A node can receive and transmit data during its listen period.

- *Strategy*: The sensor node periodically goes to the fixed listen/sleep cycle. A time frame in S-MAC is divided into two parts: one for a *listening session* and the other for a *sleeping session*.
- SMAC adopts a periodic wake up scheme. SMAC tries to synchronize the listen periods of neighboring nodes.

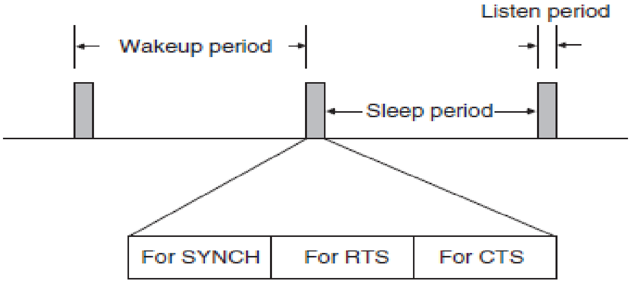


Figure 5.6 S-MAC principle

- The listen period of a node is divided into three phases as shown figure 5.6. The listen period is the time during which a node is awake, rest of the time node is sleeping. The listen and sleep periods in the S-MAC are fixed intervals.
- ***Three phases of listen period in S-MAC:***

1. ***Sync phase:*** *In this phase* node x accepts SYNCH packets from its neighbors. These packets, the neighbors describe their own schedule and x stores their schedule in a table. Node x's SYNCH phase is subdivided into time slots and x's neighbors contend according to a CSMA scheme with additional back-off. SYNC packet is used to synchronize periodically. The SYNC packet contains senders address and time of its next sleep. The next sleep time is according to the sender, the receiver will adjust its timers after it receives the SYNC packet and updates the neighbor's schedule.
2. ***RTS phase:*** During this phase x listens for RTS packets from neighboring nodes. In S-MAC, the RTS/CTS handshake is used to reduce collisions of data packets. Due to hidden-terminal situations. Again, interested neighbors contend in this phase according to a CSMA scheme with additional back-off.

3. ***CTS phase:*** In the third phase, node x transmits a CTS packet if an RTS packet was received in the previous phase. After this, the packet exchange continues, extending into x's nominal sleep time.

- In SMAC long data messages are fragmented and sent form transmitter to receiver. The receiver has to acknowledge for every fragment, else it is retransmitted. A series of fragments are sent with only one CTS and RTS message. This method is called as message passing. A protocol called T-MAC is proposed which is similar to S-MAC but with variable Listen and Sleep periods, this will help to suit the listen and sleep periods according to the load in the network.
- The main concept in SMAC is that, all the neighboring nodes form virtual clusters and synchronize their sleep and listen periods. They communicate during their listen periods and sleep rest of the time. The immediate neighbors of nodes, which are transmitting and receiving, sleep until the communication is completed.
- The NAV mechanism can be readily used to switch off the node during ongoing transmissions to avoid overhearing. When transmitting in a broadcast mode.

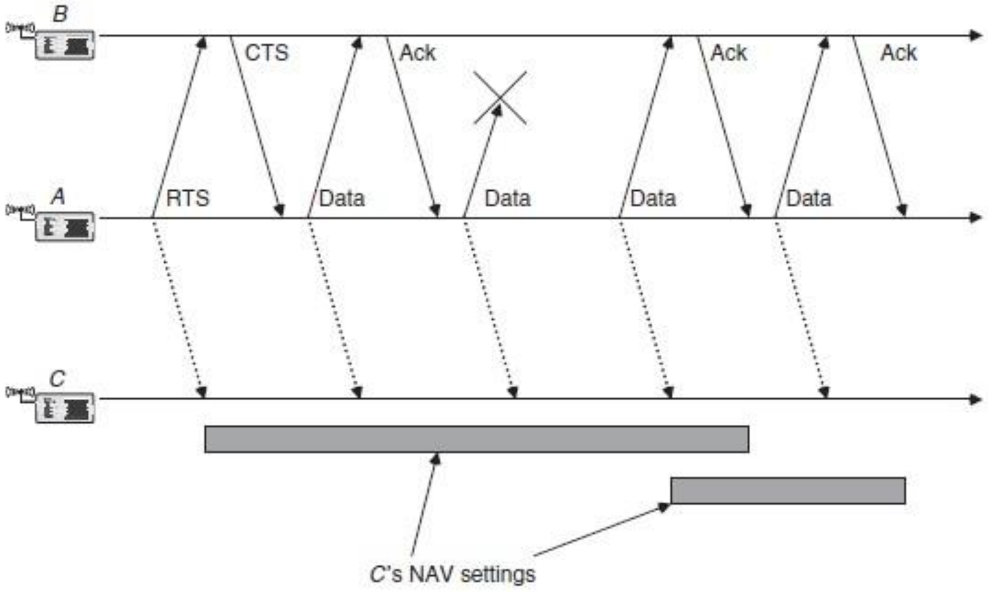


Figure 5.7 S-MAC fragmentation and NAV setting

- S-MAC also adopts a message-passing approach (illustrated in Figure 5.7), where a message is a larger data item meaningful to the application.
- In-network processing usually requires the aggregating node to receive a message completely. S-MAC includes a fragmentation scheme working as follows.
  - A series of fragments is transmitted with only one RTS/CTS exchange between the transmitting node A and receiving node B.
  - After each fragment, B has to answer with an acknowledgment packet. All the packets (data, ack, RTS, CTS) have a duration field and a neighboring node C is required to set its NAV field accordingly.

  - In S-MAC, the duration field of all packets carries the remaining length of the whole transaction, including all fragments and their acknowledgments. Therefore, the whole message shall be passed at once. If one fragment needs to be retransmitted, the remaining duration is incremented by the length of a data plus ack packet, and the medium is reserved for this prolonged time.
- S-MAC contributes in these ways;
  - Reduction of idle listening(as nodes sleep and not stay in idle state),
  - Collision and overhearing avoidance by using RTS and CTS, and saving energy and time, by sending a series of fragments of a long message together, rather than going for contention after sending every fragment.
- ***Advantages of S-MAC*** :
  - The battery utilization is increased implementing sleep schedules.
  - This protocol is simple to implement, long messages can be efficiently transferred using message passing technique.
- ***Disadvantages of S-MAC***:
  - RTS/CTS are not used due to which broadcasting which may result in collision.
  - Adaptive listening causes overhearing or idle listening resulting in inefficient battery usage.
  - Since sleep and listen periods are fixed variable traffic load makes the algorithm inefficient.

The mediation device protocol

- Compatible with the P2P communication mode of the 802.15.4 WPAN standard.
- Each node go into sleep periodically and wakeup for short periods to receive packets from neighbors.
- No global time, each node has its own schedule. Does not take care of its neighbors sleep schedules.
- At wakeup, a node transmits a short query beacon with its node address; no packets? Go back to sleep.
- The dynamic synchronization approach achieves this synchronization without requiring the transmitter to be awake permanently to detect the destinations query beacon. To achieve this, a mediation device (MD) is used.
- The MD is not energy constrained and can be active all the time; this scenario is illustrated in Figure 5.8. Because of its full duty cycle, the MD can receive the query beacons from all nodes in its vicinity and learn their wakeup periods.

Working of MD protocol:

- Assume if node A wants to transmit a packet to node B, then Node A announces this to the MD by sending periodically request to send (RTS) packets, which the MD captures.

- Node A listens for answers from the MD has received A’s RTS packet, it waits for B’s next query beacon.

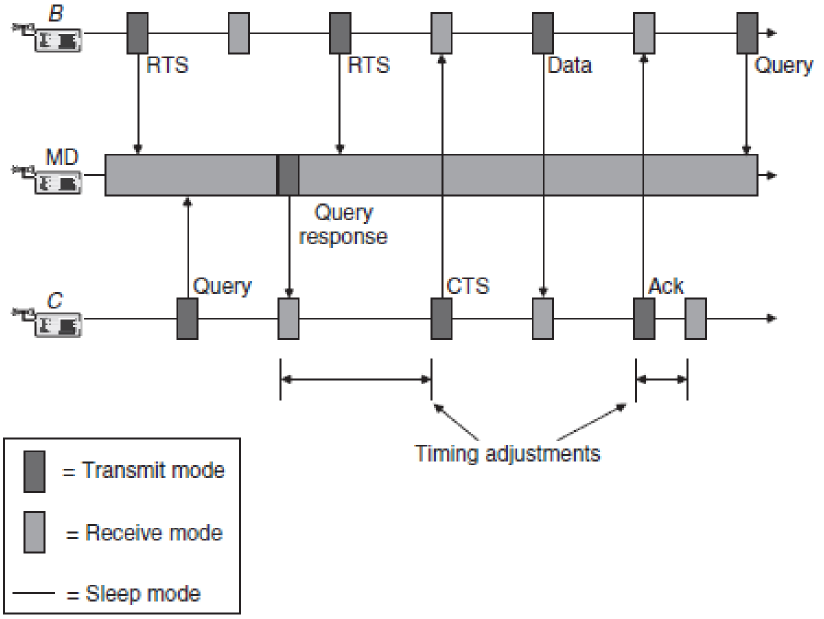


Figure 5.8 Mediation device protocol with unconstrained mediators [115, Chap. 4, Fig. 3]

- The MD answers this with a query response packet, indicating A’s address and a timing offset, which lets B know when to send the answering clear to send (CTS) to A such that the CTS packet hits the short answer window after A’s next RTS packet.
- Therefore, B has learned A’s period. After A has received the CTS packet, it can send its data packet and wait for B’s immediate acknowledgment.
- After the transaction has finished, A restores its periodic wakeup cycle and starts to emit query beacons again.
- Node B also restores its own periodic cycle and thus decouples from A’s period.

Advantages MDP:

- It does not require any time synchronization between the nodes.
- It is a power efficient protocol, since most of the energy burden is shifted to the power unconstrained Median Device.
- All nodes can be in the sleep state most of the time and have to spend energy only for the periodic beacons.
- Transmitter can synchronize with the receiver with very low duty cycles.

Drawbacks:

- Bacon signal and ongoing transmissions may collide repeatedly when nodes have the same period and their wakeup periods overlap
- The mediation device is energy unconstrained, this is against to WSN concept.
- Need sufficient mediation devices to cover all nodes.

Wakeup Radio concepts (WuR):

- Wake up Radios are the basic concept for the on-demand communications scheme.
- The WuR handles the sending and receiving of wake up messages that switch on the main processing unit or the main radio of the required node.
- The ideal situation is to avoid idle state; A wakeup receiver is necessary: it does not need power but can detect when a packet starts to arrive; for example it suffices for it to raise an event to notify other components of an incoming packet; upon such an event, the main receiver can be turned on and perform the reception of the packet.
- The wakeup radio concept tries to attend the ideal situation by using the wakeup receiver idea;
- One of the proposed MAC protocol assumes the presence of several parallel data channels, separated either in frequency (FDMA) either in codes (CDMA); a node wishing to transmit a data packet randomly picks one of the channels and performs a carrier – sensing operation; if the channel is busy, the operation is repeated; after a certain number of tries the node backs off for a random time and starts again; if the channel is idle, the node sends a wakeup signal to the receiver indicating also the channel to use; the receiver wakes up its main data receiver, tunes to the indicated channel and data transfer can proceed; afterwards, the main receiver is sent back to its sleep mode.
- ***Advantages:***
  - Only the low – power wakeup transceiver has to be switched on all the time;
  - The scheme is naturally traffic adaptive; the MAC is more and more active as the traffic load increases.
- ***Disadvantages:***
  - Difficult hardware solution for such an ultralow power wakeup receiver.
  - The range of the wakeup radio and the data radio should be the same.
  - MAC address should be unique within its two – hop neighborhood.
  - This schemes critically relies on the wakeup channel's ability to transport useful information like node addresses and channel identifications.

Contention-based protocols

- These protocols do not rely on transmission schedules, instead they require other mechanisms to resolve contention when it occurs.
- The main advantage of contention-based techniques is their simplicity compared to most schedule-based techniques.
  - Schedule-based MAC protocols must save and maintain schedules or tables indicating the transmission order.
  - Most contention-based protocols do not require to save, maintain, or share state information.

    - This also allows contention-based protocols to adapt quickly to changes in network topologies or traffic characteristics.
- However, they typically result in higher collision rates and overheads due to idle listening and overhearing (overheads usually refer to additional bits in a packet or additional packets such as control packets)
- They may also suffer from fairness issues (i.e., some nodes may be able to obtain more frequent channel accesses than others
- A contention-based protocol (CBP) is a communications protocol allow many wireless users to use the same radio channel without pre-coordination.
- The "listen before talk" is the basic concept of contention-based protocol.
- Contention – based protocols are appropriate in case of a network that is idle for long times.
- Two major contention based protocols discussed here
    1. ***CSMA Protocols***
    2. ***PAMAS (Power Aware Multi-access with Signaling) Protocol***

1. CSMA Protocols:

- CSMA protocols are contention-based, where neighbors try their luck to transmit their packets.
- Procedure for CSMA
    - Whenever have data, sense the channel first
    - If the channel is Idle, transmit data immediately.
    - If channel is busy, wait for some random amount of time, again sense the channel.
    - If channel is free transmit data immediately.
- ***Steps involved in CSMA protocol***
    1. Node gets a new packet for transmission it starts with a random delay and initializes trial counter with zero.
    2. The purpose of the random delay for node synchronization by the external event. During this random delay, the node's transceiver can be put into sleep mode.
    3. During the following listen period, the node performs carrier sensing. If the medium is found to be busy and trials counter incremented and the node goes into the backoff mode.
    4. In the backoff mode, the node waits a random amount of time, which can depend on the number of trials and during which the node can sleep
    5. The backoff mode finishes, the node listens again. If the medium is busy and the node has exhausted its maximum number of trials, the packet is dropped.
    6. If the medium is idle, the node transmits an RTS packet and enters the "Await CTS" state.
    7. In case no CTS packet arrives or a CTS packet for another transaction is received, the node either enters the backoff mode or drops the packet, depending on the value of num retries.

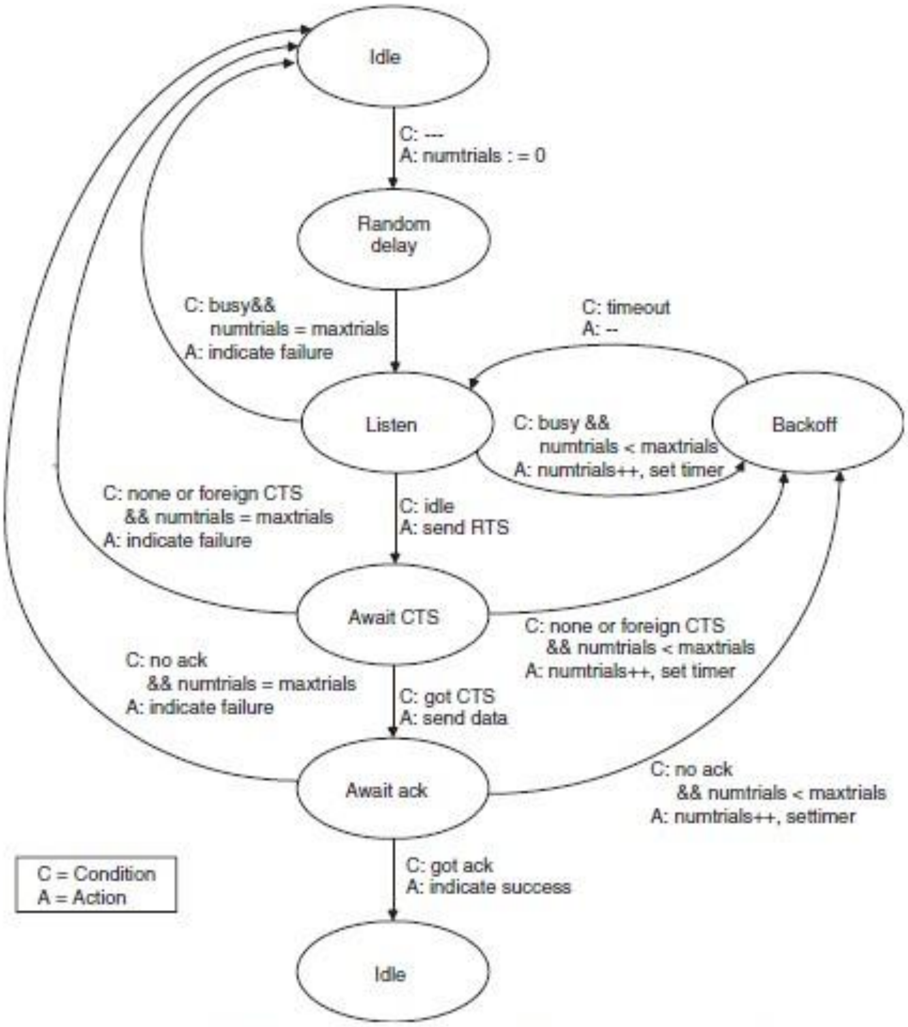
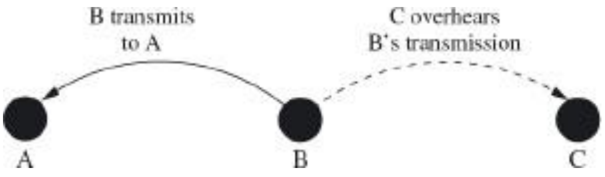


Figure 5.9 Schematic of the CSMA protocol presented in reference [888]

8. If the CTS packet arrives, the node sends its data packet and waits for an acknowledgment.
9. This acknowledgment can be either an explicit acknowledgment packet, or the parent node piggybacks the acknowledgment
10. Several variants of this skeleton (no random delay vs. random delay, random listening time vs. constant listening time, and fixed window backoff vs. exponentially increasing backoff vs. exponentially decreasing backoff vs. no backoff) have been investigated in a single-hop scenario with a triggering event.

2. PAMAS (Power Aware Multi-access with Signaling) Protocol:****

- PAMAS is originally designed for ad hoc networks.
  - The PAMAS protocol attempts to avoid unnecessary energy expenditure caused by overhearing.



- It provides detailed overhearing avoid mechanism and No idle listening solution
- It combines busy tone with RTS/CTS handshaking
- Uses two separate channels to prevent collision among data transmissions: Data channel and control channel
- All the signaling packets (RTS, CTS, busy tones) are transmitted on the control channel, while the data channel is reserved for data packets.
- The separate signaling channel allows nodes to determine when and how long to power down their wireless transceivers.

The state diagram outlining the behavior of our protocol is illustrated in Figure 4.

- As indicated in the figure, node may be in any one of six states , They are
  1. **Idle state**: *When a node is not transmitting or receiving a packet, or does not have any packets to transmit, or does have packets to transmit but cannot transmit (because a neighbor is receiving a transmission) it is in the Idle state.*
  2. **Await-CTS state:** *When it gets a packet to transmit it transmits a RTS and enters the AwaitCTS state.*
  3. **BEB (Binary Exponential Backoff) state:** *If the awaited CTS does not arrive the node goes into binary exponential backed (the BEB state in the figure).*
  4. **Await Packet:** *The intended receiver upon transmitting the CTS enters the Await Packet state*
  5. **Receive Packet state:** *If the packet does not begin arriving within one roundtrip time (plus processing time) it returns to the Idle state. If the packet does begin arriving it transmits a busy tone over the signaling channel and enters the Receive Packet state.*
  6. **Transmit Packet state**: *If a CTS does arrive it begins transmitting the packet and enters the Transmit Packet state.*

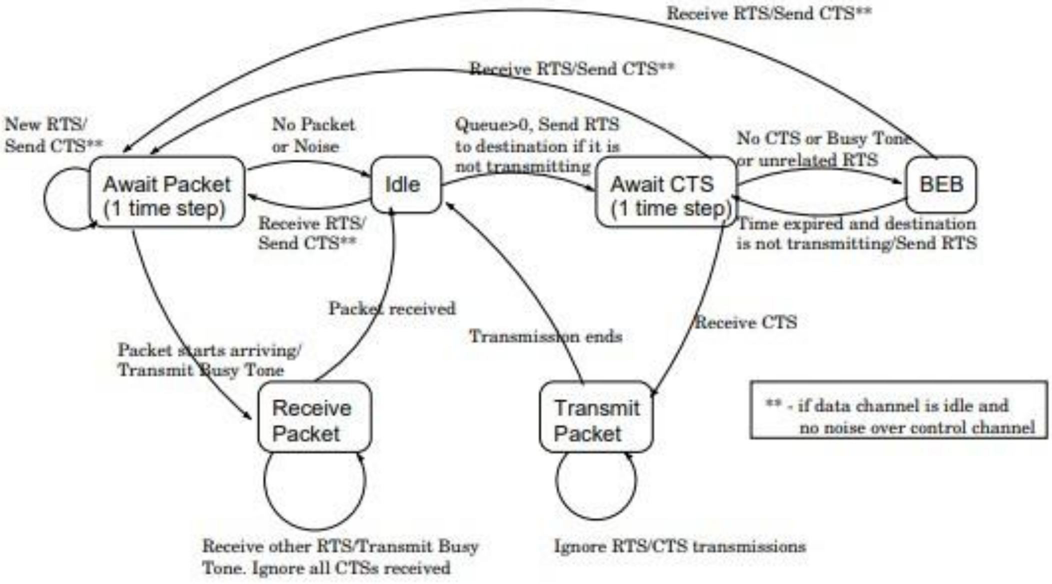
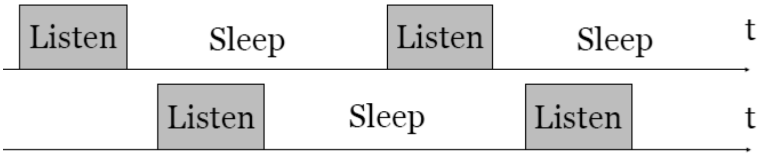


Figure 4: The PAMAS Protocol.

- **Initiating a data transfer:**
  - PAMAS device sends an RTS message over the control channel to the receiver.
  - Receiver responds with CTS if it does not detect activity on the data channel and has not overheard other recent RTS or CTS messages.
  - If the source does not receive a CTS within a specific timeout interval, it will attempt to transmit again after a back-off time (exponential back-off algorithm).
  - Otherwise, it begins data transmission and the receiver node issues a busy tone over the control channel (whose length is greater than twice the length of a CTS).
  - The receiver device also issues a busy tone over the control channel whenever it receives an RTS message or detects noise on the control channel while it receives a frame.
    - *Done to corrupt possible CTS message replies to the detected RTS, thereby blocking any data transmission of the receiver's neighbors.*
  - Every node in a PAMAS network independently decides when to power off its transceiver.
  - Specifically, a node decides to turn off its transceiver whenever one of two conditions holds:
    - a neighbor begins a transmission and the node has no frames to transmit.
    - a neighbor transmits a frame to another neighbor, even if the node has frames to transmit.
  - A node can easily detect either condition by :
    - Overhearing its neighbor's transmissions (condition 1) or
    - Overhearing its neighbor's busy tone (condition 2)
  - A node can identify how long to power down its transceiver by embedding the size or expected transmission duration into messages.
  - However, when a transmission begins while a node is still asleep, it does not know how long this transmission will last and how long it should continue to sleep.
  - Therefore, the node issues a probe frame (containing a certain time interval) over the control channel to all transmitting nodes in its neighborhood.
    - All transmitters that will complete during this interval respond with their predicted completion time.
    - If such a response is received by the awakening node without collision, it can return to the sleep mode until the completion time indicated by the transmitting node.
    - If multiple transmitters respond and their responses collide, the node reissues the probe frame with a shorter time interval.
    - Similarly, if the node did not receive a response, it can reissue the probe with a different time interval.
    - In effect, the node chooses time intervals to perform a binary search to identify when the last ongoing transmission will end.

Schedule-based protocols***

In Scheduled Based MAC Protocols, schedule nodes onto different sub-channels. Examples: TDMA, FDMA, CDMA



- ***Advantages***
  - Collision-Free
  - Low idle listening and overhearing overheads
  - Explicitly assign transmission and reception opportunities to nodes and let them sleep all other times.
- ***Drawback***
  - Requires time synch and not robust to changes.
  - Low throughput and high latency even during low contention.
  - Wake up and listen only during its neighbor transmission.
  - The schedule of a node may require a significant amount of memory.
- Following are energy efficient schedule based protocols for WSN, since they consume less energy hence they do not waste energy in collision and idle listening. They are
  - **LEACH (Low Energy Adaptive Clustering Hierarchy) protocol**
  - **SMACS (Self-Organizing Medium Access Control for Sensor Networks) protocol**
  - **TRAMA (Traffic-Adaptive Medium Access) protocol**

LEACH (Low Energy Adaptive Clustering Hierarchy) protocol*****

- LEACH protocol is a TDMA based MAC protocol.
- It is a less energy adaptive clustering hierarchy protocol.
- It is the first protocol of hierarchical routing which proposed data fusion; it is of milestone significance in clustering routing protocol.

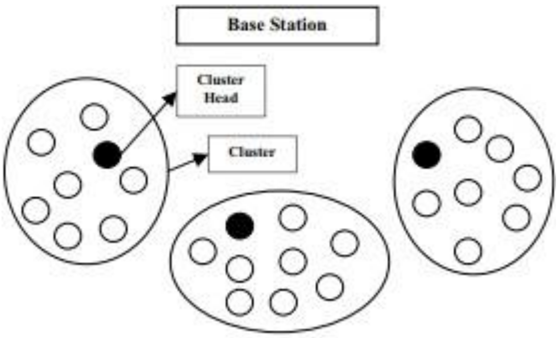


Fig. 2 Cluster Formation in LEACH.

- The main goal of cluster based sensor networks is to decrease system delay and reduce energy consumption.
- LEACH for micro sensor networks which achieves energy efficient, scalable routing and fair media access for sensor nodes.
- The principal aim of this protocol is to improve the lifespan of wireless sensor networks by lowering the energy consumption required to create and maintain Cluster Heads
- This protocol using randomized rotation of cluster heads
    - Cluster heads: collect data and applies process and aggregation on data before forwarding it to base station.
- ***Network organization in LEACH***:
    - LEACH partitions the nodes into clusters and in each cluster a dedicated node (see fig 2), the cluster head, is responsible for creating and maintaining a TDMA schedule; all the other nodes of a cluster are member nodes. To all member nodes, TDMA slots are assigned, which can be used to exchange data between the member and the cluster head; there is no peer-to-peer communication. With the exception of their time slots, the members can spend their time in sleep state. The cluster head aggregates the data of its members and transmits it to the sink node or to other nodes for further relaying.
    - After the clusters have been formed, each cluster head picks a random CDMA code for its cluster, which it broadcasts and which its member nodes have to use subsequently. This avoids a situation where a border node belonging to cluster head A distorts transmissions directed to cluster head B, shown in Figure 5.10.

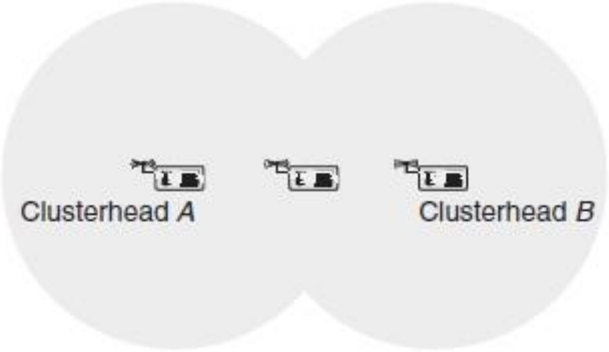


Figure 5.10 Intercluster interference

- Typically ≈5% cluster heads are designated to achieve energy and BER tradeoff.
- LEACH can achieve a seven to eight times lower overall energy dissipation compared to the case where each node transmits its data directly to the sink, and between four and eight times lower energy than in a scenario where packets are relayed in a multi-hop fashion.
- In addition, since LEACH distributes the cluster head role fairly to all nodes, they tend to die at about the same time.

- Operation of leach protocol consists of several rounds with two phases in each round. They are Set-up phase and Steady phase (See Figure 5.11).

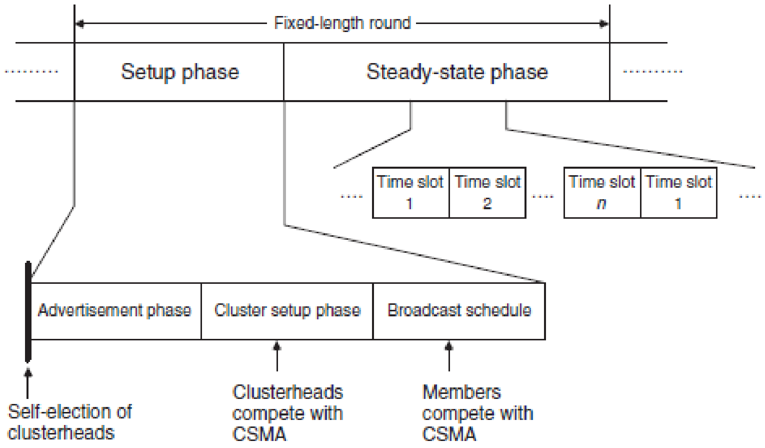


Figure 5.11 Organization of LEACH rounds

Phases of leach protocol are as follows:

A. Set-up phase: In the set-up phase, the main goal is to make cluster and select the cluster head for each of the cluster by choosing the sensor node with maximum energy. Set-up phase has three fundamental steps:

1. *Cluster head advertisement*
2. *Cluster set up*
3. *Creation of transmission schedule*

B. Steady phase: In steady phase, cluster nodes send their data to the cluster head. The member sensors in each cluster can communicate only with the cluster head via a single hop transmission. Cluster head aggregates all the collected data and forwards data to the base station either directly or via other cluster head along with the static route defined in the source code. After predefined time, the network again goes back to the set-up phase.

- ***The advantages of LEACH protocol are***:

  1. Reduced the traffic in the entire network due to data aggregation by cluster Heads.
  2. Single hop routing from nodes to cluster head it results in saving energy.
  3. It increases the lifetime of the sensor network.
  4. Location information of the nodes to create the cluster is not required.
  5. It does not need any control information from the base station.

- ***Demerits which are as follows are***:

  1. It does not give any idea about the number of cluster heads in the network.
  2. Due to any reason Cluster head dies, the cluster will become useless because the data gathered by the cluster nodes would never reach its destination.
  3. Uneven distribution of Clusters cause an increase in energy consumption.
  4. It not be able to cover large geographical areas of some square miles or more.

SMACS (Self-Organizing Medium Access Control for Sensor Networks) protocol***

- SMACS is a distributed protocol suite which enables a collection of nodes to discover neighbors and establish schedules for communicating with them without the need of a "master" node.
- It is an infrastructure-building protocol that forms a flat topology for sensor networks.
- It combined neighbor discovery and channel assignment phases.
- ***SMACS—Assumptions:***
  - The available spectrum is subdivided into many channels and each node can tune its transceiver to an arbitrary one.
  - Most of the nodes are stationary and remain as such for a long time
  - Each node divides its time locally into fixed-length super frames of T_{frame} length
  - Super frames are subdivided into timeslots.
- ***Link organization between SMACS nodes:***
  - The goal of SMACS is to detect neighboring nodes and to set up exclusive links to these.
  - A link is bidirectional TDMA link with receive slot and a transmit slot between the nodes and transmit all packets through this link.
  - The assignment of links shall be such that no collisions occur at receivers. To achieve this, SMACS takes care that for a single node the time slots of different links do not overlap.
  - Furthermore for each link randomly one out of a large number of frequency channels/CDMA codes is picked and allocated to the link.
  - It is not required that a node and its neighbors transmit at entirely different times. In this case, however, they must transmit to different receivers and have to use different frequencies/codes.
  - After link setup, the nodes wake up periodically (once per superframe) in the respective receive time slots with the receiver tuned to the corresponding frequency or with the correct CDMA code at hand; the transmit time slots are only used when required.
- ***Illustration of Neighbor discovery and link setup in SMACS:***

  It is explained by consider four different cases. See figure 5.12

Case1: Suppose that nodes x and y want to set up a link. Assume x is turns on first and listens on a fixed frequency band for a random amount of time. If nothing is received during this time, node x sends an invitation message TYPE1(x, unattached) message, indicating its own node ID and the number of attached neighbors, which so far is zero. When any neighbor z of node x receives this message, it waits for a random but bounded amount of time and answers with a TYPE2(x, z, n) message, indicating its own address, x's address and its number of neighbors n. Now, suppose that the so-far unconnected node y answers first – with TYPE2(x, y, unattached) – and x receives this message properly. Since y sent the first answer, x invites y to construct a link by sending a TYPE3(y, –) message, carrying the identification of the "winning" node y and no further parameters. Now, node y knows that (i) it has been selected, and (ii) it can pick any time slot it wants since neither x nor y has any link allocated so far. Node y answers to node x with a link specification, that is, two time slot specifications and a frequency/code, using a TYPE2(x, y, LinkSpec) message.

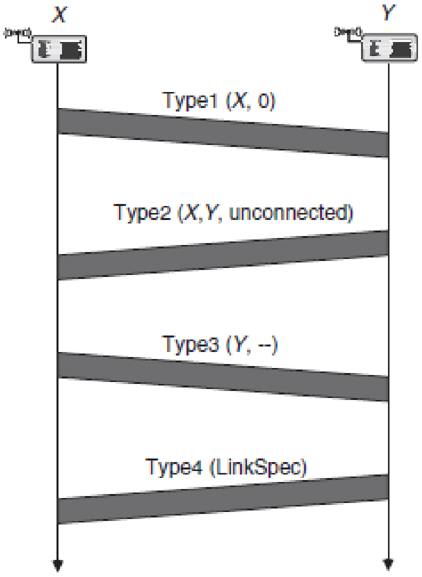


Figure 5.12 SMACS: link setup for two lonesome nodes

- Any other node z loosing against y goes back into sleep mode and tries again at some later time. The nodes repeat their invitations periodically using TYPE1 (· , ·) messages.

Case 2*:* Assume where node x already has some neighbors but the winning node y has none so far. Therefore, x sends a TYPE1(x, attached) message and y manages to answer first with its TYPE2(x, y, unattached) message. After this, node x knows that it can schedule the connection to y freely, since y has no obligations so far. Node x picks two convenient time slots and a frequency and sends a TYPE3(y, LinkSpec) message to y. Again, since y has no neighbors so far, y adopts the superframe phase of x. Finally, node y answers with TYPE2(x, y, –) message, carrying an empty link specification (meaning that x's link specification is adopted).

Case3: In this case, node x does not have any neighbor yet, but y has. Therefore, y answers to x's TYPE1(x, unattached) with a TYPE2(x, y, attached) message. Node x proceeds with sending a TYPE3(y, –) message without link specification to y, and it is y's turn to pick the time slots and frequency. Accordingly, y sends back a TYPE2(x, y, LinkSpec) to x.

Case 4: In the final case, both x and y are already attached to other nodes and their superframes are typically not aligned. Accordingly, x sends a TYPE1(x, attached) message and y answers with a TYPE2(x, y, attached) message. Node x answers with a TYPE3(y, Schedule) message, which contains its entire schedule as well as timing information allowing y to determine the phase shift between x and y's superframes. After receiving this information, node y determines time slots that are free in both schedules, and which are not necessarily aligned with any time slot boundaries in either schedule.

Traffic-Adaptive Medium Access (TRAMA) Protocol***

- TRAMA reduces energy consumption by providing collision-free transmissions and low-power idle state
- Assumes single time-slotted channel and uses a distributed election scheme to determine which node can transmit at a particular slot.
- The schedules are constructed in a distributed manner and on an on-demand basis.
- The protocol assumes that all nodes are time synchronized and divides time in to
  - ***Random access period***: signaling slots
  - ***Scheduled access period***: transmission slots
- A random access period followed by a scheduled-access period is called a cycle.
- The protocol itself consists of three different components:
  1. ***The Neighborhood Protocol(NP):***
  2. ***The Schedule Exchange Protocol (SEP)***:
  3. ***The Adaptive Election Algorithm (AEA):***

1. The Neighborhood Protocol (NP) in TRAMA:

- Propagates one-hop neighbor information among neighboring nodes during random access period (contention based channel acquisition and signaling).
- TRAMA starts in random access mode where each node selects a slot randomly.
- Nodes can only join the network during the random access periods (occur more often in dynamic networks).
- NP gathers neighborhood information by exchanging small signaling packets, carrying incremental neighborhood updates.
- If no updates, the signaling packets serve as “keep-alive” beacons.
- A node times out its neighbor if it does not hear from it for a certain period of time
- The updates are transmitted to ensure 99% probability of success.

2. The Schedule Exchange Protocol (SEP):

- Exchange traffic-based information, or schedules (information on traffic originating from a node), with neighbors.
- Establishes and maintains traffic-based schedule information required by the transmitter (e.g. slot re-use) and the receiver (i.e. sleep state switching)
- A node’s schedule captures a window of traffic to be transmitted by the node; schedules have timeouts.
- Nodes announce their schedule via schedule packets.
- The intended receiver information is conveyed using a bitmap.
- A schedule summary is also send during data transmission to minimize effects of packet loss

in schedule dissemination.

- Nodes maintain schedule information for their one-hop neighbors, which is consulted when needed.
- An unused slot is called Changeover slot; all nodes listen during the Changeover slot of the transmitter to synchronize their schedule.

3. The Adaptive Election Algorithm (AEA):

- Selects transmitters and receivers to achieve collision-free *transmission using the information from NP and SEP.*
  - *Random transmission -> collisions*
  - *Transmitters without receivers -> energy waste*
- At any given time slot t during the scheduled access period, the state of a node u is determined based on its two-hop neighborhood information and the schedules of it's one-hop neighbors; possible states are: transmit (TX), receive (RX), or sleep (SL)
  - Node u is in TX state if (1) u has the highest priority among its contending set and (2) u has data to send
  - Node u is in RX state when it is the intended receiver of the current transmitter
  - Otherwise the node can be turned off to SL state
- ***TRAMA—Winners:***
  - The state of a node u depends on the Absolute Winner and the schedules of its one-hop neighbors
  - From node u's perspective, the Absolute Winner at a time slot t can be:
    - Node u itself
    - Node v that lies in the two-hop neighborhood of node u in which case the Alternate Winner atx(u) is to be considered if hidden from node v
    - Node w that lies in node u's one-hop neighborhood
  - The Absolute Winner is the assumed transmitter unless the Alternative Winner is hidden from the Absolute Winner and it belongs to the Possible Transmitter Set.
- ***In more complicated situation***: TRAMA energy saving depends on load situation. It is depicted in Figure 5.13.

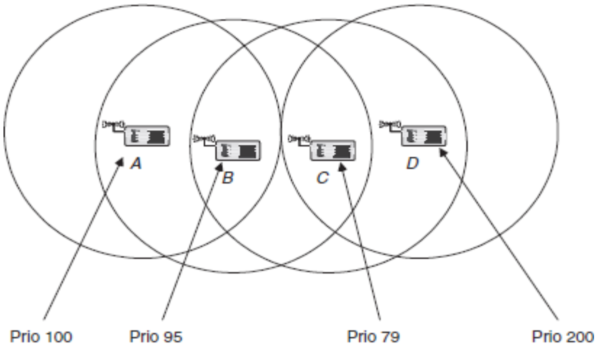


Figure 5.13 TRAMA: conflict situation

- Here, node D has the highest priority in B's two hop neighborhood, but, on the other hand node, A has highest priority in its two-hop neighborhood.
- The adaptive election algorithm of TRAMA provides approaches for resolving this situation and also for allowing nodes to reuse their neighbors' unused winning slots.
- ***Advantage of TRAMA Protocol***
  - *Higher percentage of sleep schedules and collision free transmissions are achieved compared to CSMA based protocols.*
  - *Higher maximum throughput than contention-based protocols.*
  - *TRAMA protocol is suitable for applications require high energy efficiency and throughput.*
- ***Disadvantage of TRAMA Protocol***
  - Not suitable delay sensitive application, since it takes more delay.
  - TRAMA is a feasible solution only if the sensor nodes have sufficient resources.

Chapter 7: Naming and addressing

Fundamentals: Naming and addressing schemes are used to denote and to find things. In networking, names and addresses often refer to individual nodes as well as to data items stored in them.

- **Name:** Denote/refer to "things"
  - *Nodes, networks, data, transactions, …*
  - *Often, but not always, unique (globally, network-wide, locally)*
  - *Ad hoc: nodes –WSN: Data!*
- **Addresses:** Information needed to find these things
  - *Street address, IP address, MAC address*
  - *Often, but not always, unique (globally, network-wide, locally)*
  - *Addresses often hierarchical, because of their intended use in, e.g., routing protocols*
- **Services to map between names and addresses**
  - *E.g., DNS*
- **Sometimes, same data serves as name and address**
  - *IP addresses are prominent examples*

Use of addresses and names in (sensor) networks

1. ***A unique node identifier (UID)*****:** *UID might be a combination of a vendor name, a product name, and a serial number, assigned at manufacturing time.*
2. ***MAC address, Network address, Network identifiers, Resource identifiers***

Assignment of MAC addresses***

- ***Address allocation****: Assign an entity an address from a given pool of possible addresses*
  - *Distributed address assignment (centralized like DHCP does not scale)*
- ***Address deallocation****: Once address no longer used, put it back into the address pool*
  - *Because of limited pool size*
  - *Graceful or abrupt, depending on node actions*
- ***Address representation***
  - *Format for representing addresses must be negotiated and implemented*
- ***Conflict detection & resolution (Duplicate Address Detection)***
  - *What to do when the same address is assigned multiple times?*
  - *Can happen e.g. when two networks merge*
    - *Unnecessary Deallocation/Reallocation should be avoided*
- ***Binding***
  - *Map between addresses used by different protocol layers*
  - *E.g., IP addresses are bound to MAC address by ARP*
- ***Why not globally unique addresses?***
  - Globally unique addresses significantly simplify address management.
  - Must be judged relative to the impact on overhead
  - E.g., Ethernet 48-bit MAC address
    - *500-octet frame (e.g., IP): overhead of 1.2%*
    - *4-octed frame (e.g., WSN): overhead of 150%*
  - MAC addresses only need to be unique within 2-hop neighborhood.

Distributed assignment of network wide addresses

Option 1: Let every node randomly pick an address

- For given size of address space, unacceptable high risk of duplicate addresses

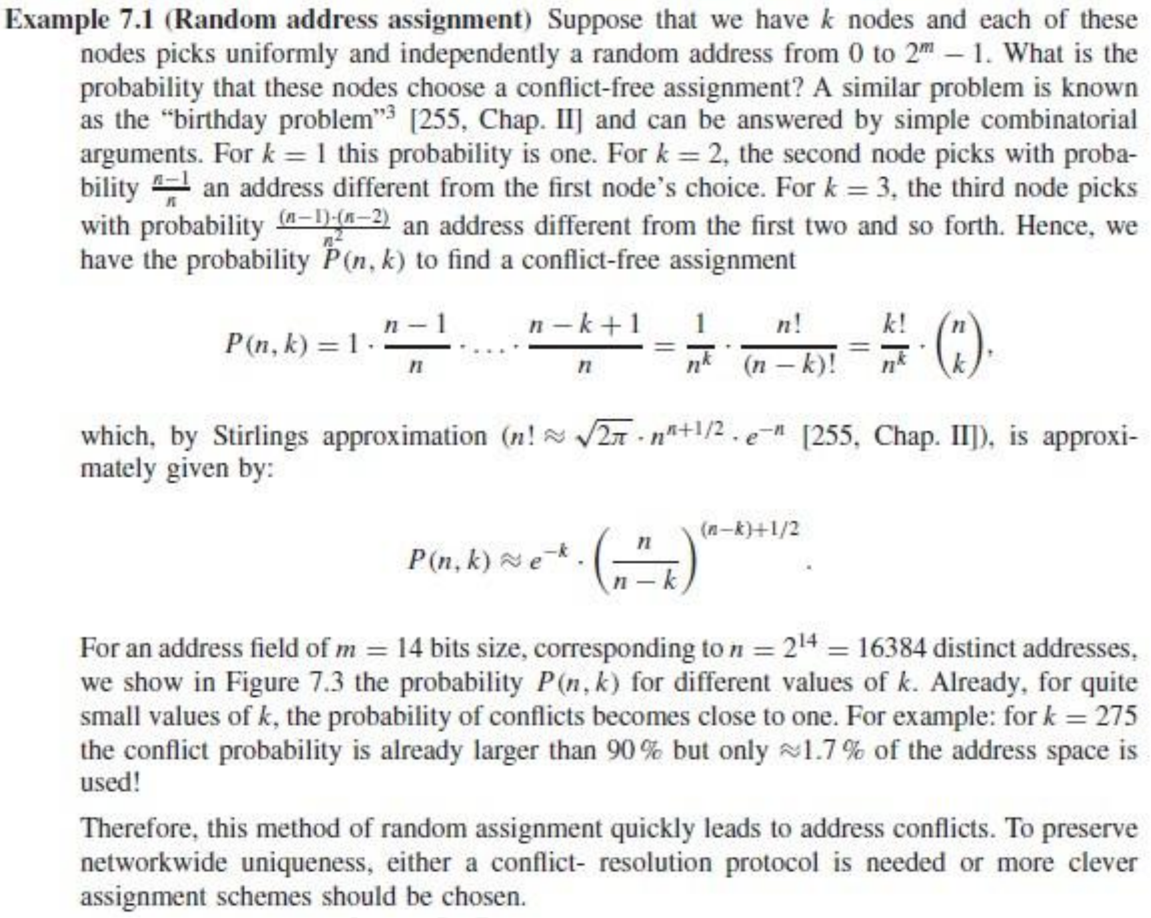
Option 2: Avoid addresses used in local neighborhood

Option 3: Repair any observed conflicts

- *Temporarily pick a random address from a dedicated pool and a proposed fixed address*
- *Send an address request to the proposed address, using temporary address*
- *If address reply arrives, proposed address already exists*
- *Collisions in temporary address unlikely, as only used briefly*

Option 4: Similar to 3, but use a neighbor that already has a fixed address to perform requests

- A node randomly picks an address from a given address range between 0 and $2^m - 1$ and an address can thus be represented with m bits. The address space has a size of $n = 2^m$ addresses.
- A node chooses its address without any prior information, leads problems, as is shown in the following example 7.1.



Example 7.1 (Random address assignment) Suppose that we have k nodes and each of these nodes picks uniformly and independently a random address from 0 to $2^m - 1$. What is the probability that these nodes choose a conflict-free assignment? A similar problem is known as the "birthday problem"[3] [255, Chap. II] and can be answered by simple combinatorial arguments. For $k = 1$ this probability is one. For $k = 2$, the second node picks with probability $\frac{n-1}{n}$ an address different from the first node's choice. For $k = 3$, the third node picks with probability $\frac{(n-1)\cdot(n-2)}{n^2}$ an address different from the first two and so forth. Hence, we have the probability $P(n, k)$ to find a conflict-free assignment

$$P(n,k) = 1 \cdot \frac{n-1}{n} \cdot \ldots \cdot \frac{n-k+1}{n} = \frac{1}{n^k} \cdot \frac{n!}{(n-k)!} = \frac{k!}{n^k} \cdot \binom{n}{k},$$

which, by Stirlings approximation ($n! \approx \sqrt{2\pi} \cdot n^{n+1/2} \cdot e^{-n}$ [255, Chap. II]), is approximately given by:

$$P(n,k) \approx e^{-k} \cdot \left(\frac{n}{n-k}\right)^{(n-k)+1/2}.$$

For an address field of $m = 14$ bits size, corresponding to $n = 2^{14} = 16384$ distinct addresses, we show in Figure 7.3 the probability $P(n, k)$ for different values of k. Already, for quite small values of k, the probability of conflicts becomes close to one. For example: for $k = 275$ the conflict probability is already larger than 90 % but only ≈1.7 % of the address space is used!

Therefore, this method of random assignment quickly leads to address conflicts. To preserve networkwide uniqueness, either a conflict- resolution protocol is needed or more clever assignment schemes should be chosen.

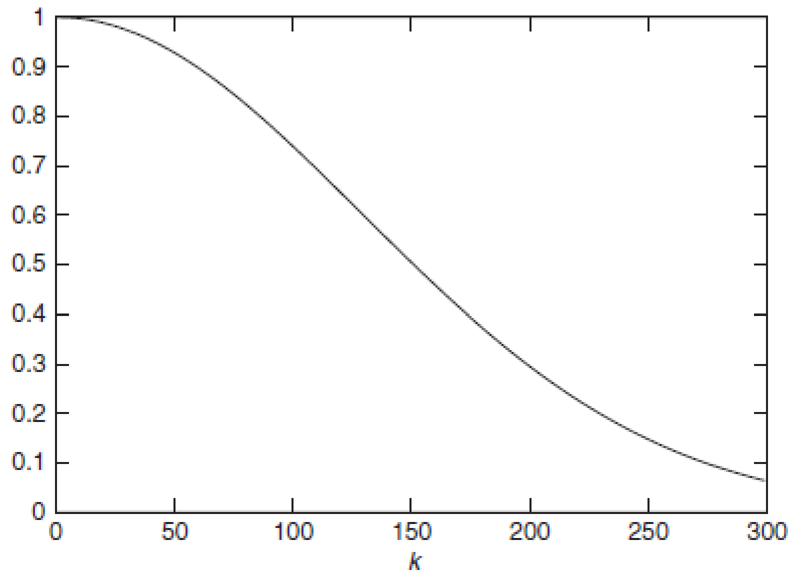


Figure 7.3 "Birthday probability" that k out of $n = 2^{14}$ station pick random addresses without conflicts

Chapter 10: Topology control

10.1 Motivation and basic ideas: Typical characteristic of WSN is the possibility of deploying many nodes in a small area leads to dense network deployment see fig 10.1.

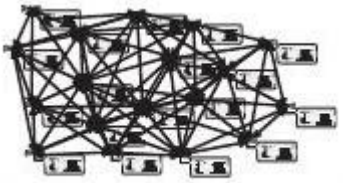
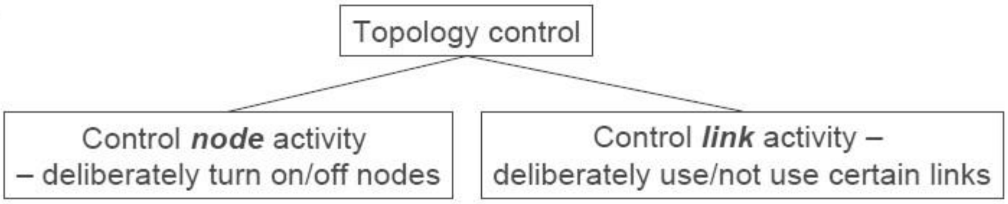


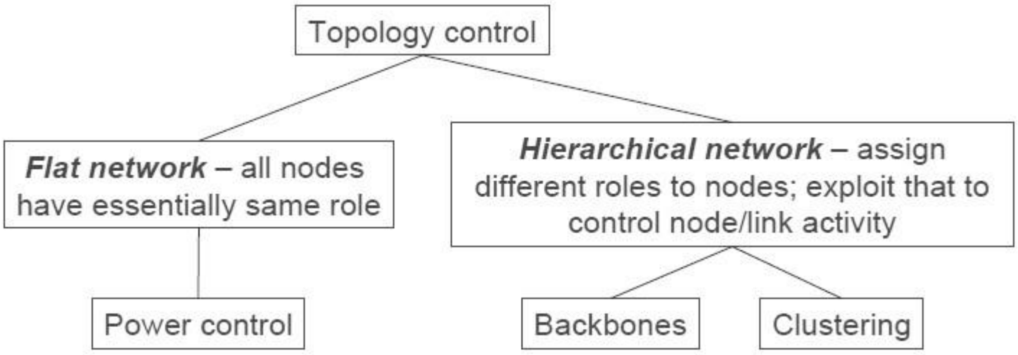
Figure 10.1 Topology in a densely deployed wireless sensor network

- In a very dense networks, too many nodes might be in range for an efficient operation. This result following problems
  - *Too many node collisions*
  - *Many nodes interfere with each other*
  - *Too complex operation for a MAC protocol,*
  - *Too many paths to choose from for a routing protocol,*
  - *Nodes might needlessly use large transmission power to talk to distant nodes directly.*
  - *Nodes might needlessly use large transmission power to talk to distant nodes directly*.

- Some of these problems can be overcome by ***topology-control techniques***
  - ***Idea:*** Make topology to reduce complex
    - *Topology:* Which node is able/allowed to communicate with which other nodes
    - Topology control needs to maintain invariants, e.g., connectivity.

- **Options for topology control**


Topology control
Control *node* activity – deliberately turn on/off nodes
Control *link* activity – deliberately use/not use certain links


Topology control
Flat network – all nodes have essentially same role
Hierarchical network – assign different roles to nodes; exploit that to control node/link activity
Power control
Backbones
Clustering

Hierarchical networks by clustering***

Due to scarce resources in WSN, direct communication of sensor node with BS or multihop communication of sensor nodes towards BS is not practical as energy consumption is high which results in early expiry of sensor nodes as shown in Figure 10.1.

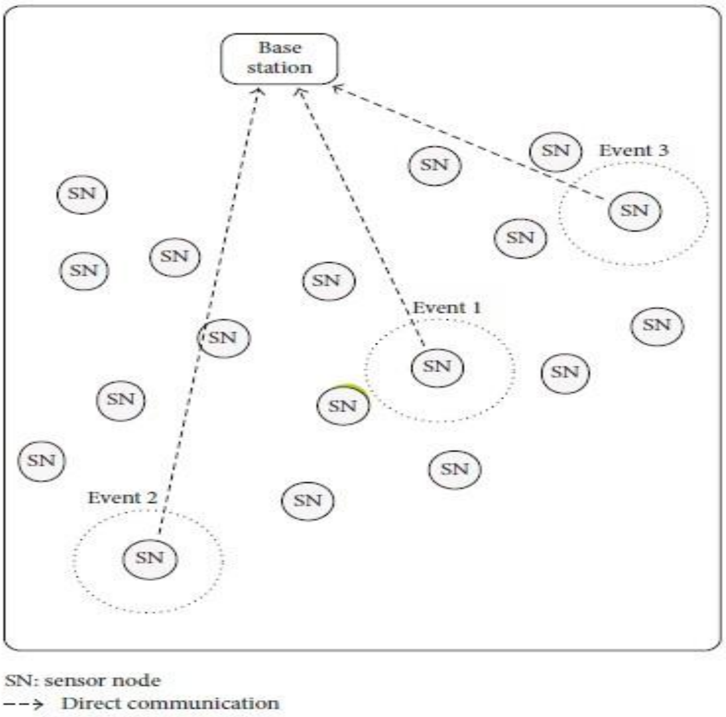


Figure 10.1: Direct communication in WSN

- Direct communication has its disadvantages:
  - *High energy consumption.*
  - *Duplication of data (SN that were close to each other, sending data with very small variation).*
  - *Farthest nodes dying quickly*.
  - To overcome above problems, hierarchical cluster approach is used.

Definition of clusters:

- Partition nodes into groups of nodes called ***clusters***
- One controller/representative node per cluster called ***Clusterheads***

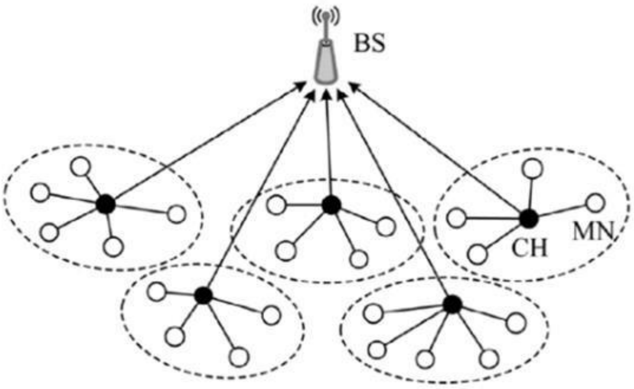


Figure 10.2: Cluster approach in WSN

- Formally, given a graph G = (V ,E), clustering is simply the identification of a set of subsets of nodes V_i , i = 1, . . . , n such that ∪i=1,...,nVi = V .
- Each node in exactly one group
- Except for nodes "bridging" between two or more group
- Typically: all nodes in a cluster are direct neighbors of their clusterhead
- Clusterheads are also a dominating set, but should be separated from each other – they form an independent set
- ***Are there clusterheads?*** for each set V_i there is a unique node C_i , the clusterhead, so Groups can have clusterheads
- ***May clusterheads be neighbors?*** Yes, often desirable to have clusterheads separated.
  - o Formally, clusterheads should form an independent set:

$$\forall c_1, c_2 \in C : (c_1, c_2) \notin E$$

  - o Typically: clusterheads form a maximum independent set. See figure10.7

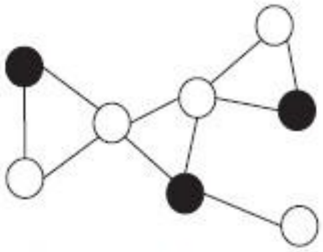


Figure 10.17 An example graph with a maximum independent set [59]

- ***May clusters overlap? Do they have nodes in common?***

To create maximum independent cluster set cluster may overlap and non-clusterhead nodes may common for both clusters. Figure 10.18 highlights these possibilities.

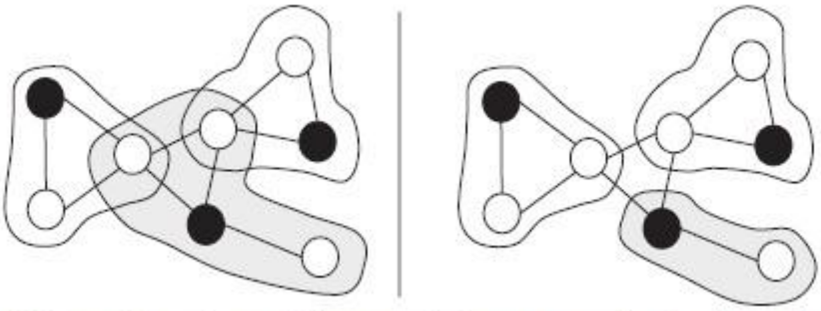


Figure 10.18 Maximum independent set induces overlapping or nonoverlapping clusters (example adapted from reference [59])

- ***How do clusters communicate?*** They communicate through gateways**.** Some nodes need to act as gateways between clusters. If clusters may not overlap, two nodes need to jointly act as a distributed gateway.

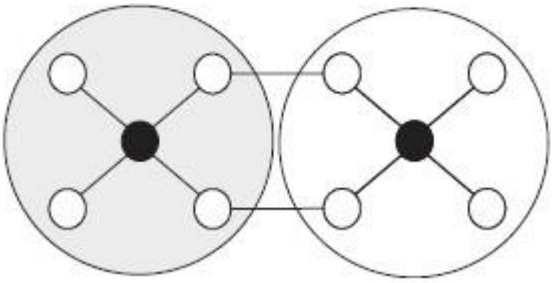


Figure 10.19 Two clusters connected by two distributed gateways

- ***How many gateways exist between clusters?***

  Depending on the optimization goal for the eventual connected dominating set.
- ***What is the maximal diameter of a cluster?*** *Depends on nuber of hops between two clusters.*
- ***Is there a hierarchy of clusters?*** *Clusterheads impose a hierarchy of nodes onto the network.*

A basic idea to construct independent sets

- Independent sets exploits the inherently local nature of being independent.
- The idea is thus for every node to communicate with its neighbors and to locally select nodes to join the set of independent nodes (to become clusterheads in the end).
- Distributed algorithm used to compute independent sets starts out by marking all nodes as being ready to become clusterheads.
- In the first step, each node determines its local ranking property and exchanges it with all of its neighbors.
- Once this information is available, a node can decide to become a clusterhead if it has the largest rank. Among all its as-yet-undecided neighbors. It changes its state accordingly and announces its new state to its neighbors
- The algorithm terminates once all nodes have decided to become either a clusterhead or a cluster member.
- This algorithm is illustrated with a simple linear network in Figure 10.20.

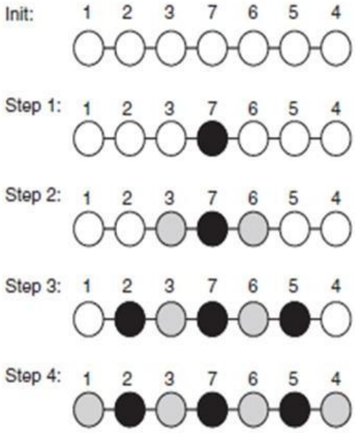


Figure 10.20 Basic algorithm for determining independent sets, using node identifiers as rank (white nodes are undecided, black nodes are clusterheads, gray ones are cluster members)

- In step 1, nodes 2 and 5 cannot become clusterheads because their neighboring nodes 3 and 6 have not yet decided and would, potentially, take precedence over them.
- Once nodes 3 and 6 have learned about node 7 being a clusterhead in their vicinity, they decide to become cluster members and propagate this information to nodes 2 and 5. Then, these nodes can become clusterheads in step 3. Use some attribute of nodes to break local symmetries.
- Make each node a clusterhead that locally has the largest attribute value. Node identifiers, energy reserve, mobility, weighted combinations (all types of variations have been looked at) are used evaluate the attribute.
- Once a node is dominated by a clusterhead, it abstains from local competition, giving other nodes a chance

Connecting clusters

- Once the clusterheads have been found, it necessary to determine the gateways between the clusters.
- How to connect the clusters, how to select gateways? Put simply, this problem is reduced again to the Steiner tree problem.
- It suffices for each clusterhead to connect to all other clusterheads that are at most three hops. Resulting backbone is connected.
- While for some networks, this might mean more connections than necessary, but there are networks where all this links are needed to ensure connectivity.
- In addition to this basic connectivity consideration, other aspects like load balancing between multiple gateways can be considered.

Rotating clusterheads

- Serving as a clusterhead can put additional burdens on a node for MAC coordination, routing and other services.
- Let this duty rotate among various members. Periodically reelect cluserhead based on energy reserves are used as discriminating attribute.
- LEACH protocol used to determine an optimal percentage P of nodes to become clusterheads in a network with following procedure.
  - Use 1/P rounds to form a period
  - In each round, nP nodes are elected as clusterheads
  - At beginning of round r, node that has not served as clusterhead in this period becomes clusterhead with probability P/(1-p(r mod 1/P))

Multihop clusters.

- Clusters with diameters larger than 2 can be useful, e.g., when used for routing protocol support. Formally: Extend "domination" definition to also dominate nodes that are at most d hops away. Goal: Find a smallest set D of dominating nodes with this extended definition of dominance. Only somewhat complicated heuristics exist.

- *Different tilt*: Fix the size (not the diameter) of clusters. *Idea*: Use growth budgets – amount of nodes that can still be adopted into a cluster, pass this number along with broadcast adoption messages, reduce budget as new nodes are found.

Passive clustering:

- In terms of energy consumption, one of the most expensive operations in a network is flooding.
- Flooding can incur considerable overhead. This clustering overhead can be reduced if the information flow that is happening anyway during a flooding operation is leveraged to compute a clustering structure on the fly. Actively sending out any message for clustering as such is avoided; the approach called passive clustering.
- The necessary information exchange is achieved by adding state information about each sender into any packet that is sent anyway, namely “initial”, “clusterhead”, “gateway”, and “ordinary node”. This distributes information about the state of neighboring nodes; it suffices to build a clustering structure that well approximates maximum independent sets with optimal gateway choice and is competitive with ID-based or degree-based algorithms.
- The procedure works as follows:
  - Node to start a broadcast: Initial node
  - Nodes to forward this first packet: Clusterhead
  - Nodes forwarding packets from clusterheads: ordinary/gateway nodes and so on. Finally clusters will emerge at low overhead.

- ***Conclusion for topology control:***
  - *Topology control – namely, power control, backbones, and clustering – is a powerful means to change the appearance and properties of a network for other protocol layers*
  - *Various approaches exist to trim the topology of a network to a desired shape.*
  - *Most of them bear some non-negligible overhead.*
  - *Judicious use of topology control can significantly improve operational aspects of a network, such as lifetime. However, determining an optimal topology is usually prohibitively expensive and appropriate approximations and heuristics have to be used instead.*

*Chapter 11. Routing protocols for WSN****

The design of routing protocols for WSNs is challenging because of several network constraints. WSNs suffer from the limitations of several network resources, for example, energy, bandwidth, CPU and storage. The design issues for routing protocols are *limited energy capacity, Sensor locations, Limited hardware resources, Massive and random node deployment, Network* *characteristics and unreliable environment, Data Aggregation, Diverse sensing application requirements, Scalability etc..*

In a multihop network, intermediate nodes have to relay packets from the source to the destination node. Such an intermediate node has to decide to which neighbor to forward an incoming packet not destined for itself. Typically, routing tables that list the most appropriate neighbor for any given packet destination are used. The construction and maintenance of these routing tables is the crucial task of a distributed routing protocol. Routing in wireless sensor networks differs from conventional routing in fixed networks in various ways. There is no infrastructure, wireless links are unreliable, sensor nodes may fail, and routing protocols have to meet strict energy saving requirements. Many routing algorithms were developed for wireless networks in general. All major routing protocols classes proposed for WSNs. Some of them are

1. **Energy-Efficient Routing**
2. **Geographic Routing**

Energy-efficient and unicast routing***

Overview: In energy-efficient unicast routing consider a the network graph, assign to each link a cost value that reflects the energy consumption across this link, and pick any algorithm that computes least-cost paths in a graph. Modified Dijkstra's shortest path algorithm to obtain routes with minimal total transmission power. Particularly interesting and good cost performance metric: is energy efficiency. Figure 11.3 shows an example scenario for a communication between nodes A and H including link energy costs and available battery capacity per node. Following are the goal to find energy efficient metrics

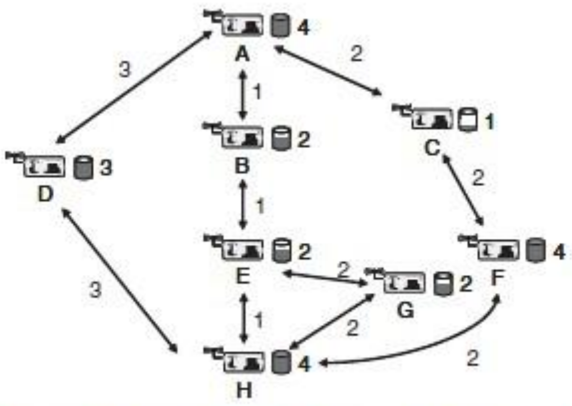


Figure 11.3 Various example routes for communication between nodes A and H, showing energy costs per packet for each link and available battery capacity for each node (adapted from reference [17])

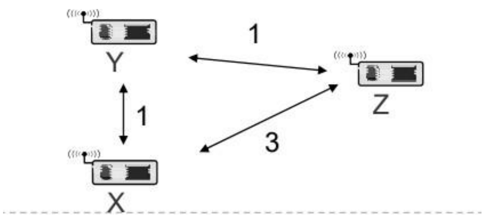
- ***Minimize energy per packet (or per bit):*** Total energy required to transport a packet over a multihop path from source to destination. The goal is then to minimize, for each packet, this total amount of energy by selecting a good route. This cost metric can be easily included in standard routing algorithms. It can lead to widely differing energy consumption on different nodes. In the example of Figure 11.3, the minimum energy route is A-B-E-H, requiring 3 units of energy. The minimum hop count route would be A-D-H, requiring 6 units of energy.

- ***Maximize network lifetime: T***he network should be able to fulfill its duty for as long as possible. Several options exist
  1. ***1.*** *Time until the first node fails.*
  2. *Time until there is a spot that is not covered by the network*
  3. *Time until network partition (when there are two nodes that can no longer communicate with each other)*
- ***Routing considering available battery energy:*** The finite energy supply in nodes' batteries is the limiting factor to network lifetime, it stands to reason to use information about battery status in routing decisions. Some of the possibilities are:
  1. *Maximum Total Available Battery Capacity:* Choose that route where the sum of the available battery capacity is maximized. Path metric: Sum of battery levels. Example: A-C-F-H
  2. *Minimum Battery Cost Routing (MBCR*): Here routing cost can be measured as the reciprocal of the battery capacity. Path metric: Sum of reciprocal battery levels. Example: A-D-H.
  3. *Min–Max Battery Cost Routing (MMBCR*): Instead of using the sum of reciprocal battery levels, simply the largest reciprocal level of all nodes along a path is used as the cost for this path. Then, again the path with the smallest cost is used. In this sense, the optimal path is chosen by minimizing over a maximum. Example of Figure 11.3, route A-D-H (1/3) and ACFG (1/1) will be selected.
  4. *Conditional Max–Min Battery Capacity Routing (CMMBCR*): If there are routes along which all nodes have a battery level exceeding a given threshold. Then select the route that requires the lowest energy per bit. If there is no such route, then pick that route which maximizes the minimum battery level
  5. *Minimize variance in power levels*: To avoid some nodes prematurely running out of energy and disrupting the network. Hence, routes should be chosen such that the variance in battery levels between different routes is reduced
- ***Minimum Total Transmission Power Routing (MTPR):*** Goal: guarantee that transmissions are successful. A given transmission is successful if its SINR exceeds a given threshold. The goal is to find an assignment of transmission power values for each transmitter (given the channel attenuation metric) such that all transmissions are successful and that the sum of all power values is minimized.

Some example unicast protocols:

1. ***Attracting routes by redirecting***
   - Idea*: nodes can overhear packet exchanges between other nodes*
   - Process:
     - *Energy requirement is included in the packet*

- *When communication between two adjacent nodes X 31 and Z proceeds, a third node Y can decide whether it can offer a more energy-efficient route.*



2. ***Distance vector routing on top of topology control***
    - o Here lends itself to a formulation of an energy-efficient routing problem. Bellmann–Ford–type algorithm is used to find paths with minimal power consumption in the enclosure graph.
3. ***Maximizing time to first node outage as a flow problem***
    - o It is a flow problem: Normal maximum flow algorithm are not applicable.
    - o Two approximation algorithms:
        - *First algorithm*: find a generalized description of the "costs" of a link (consider energy cost, initial and residual battery capacity.)
        - *second algorithm:* is a flow redirection algorithm
        - The core result is that system lifetime can be extended 34 up to 60%
4. ***Maximizing time to first node outage by a max–min optimization***
    - o There are two algorithms
        - *The max min zPmin approximation:* The minimal remaining power in all nodes is the largest.

          Property: Require knowledge of battery power level. May pick a very expensive path.

          Sol: Pick a path having at most a power consumption of zPmin.
        - The zone routing approximation can work without this information at only slightly reduced performance.
5. ***Maximizing number of messages***
    - o The goal is to maximize the number of messages that can be sent over a network before it runs out of energy

- **Bounding the difference between routing protocols:**
    - o The graph is partitioned into "spheres" Si that include all the nodes that are reachable from the base station in at most i hops. Then, all traffic has to go through the nodes of sphere S1, and because there are relatively few of these nodes, they limit the lifetime of the network shown in 11.4

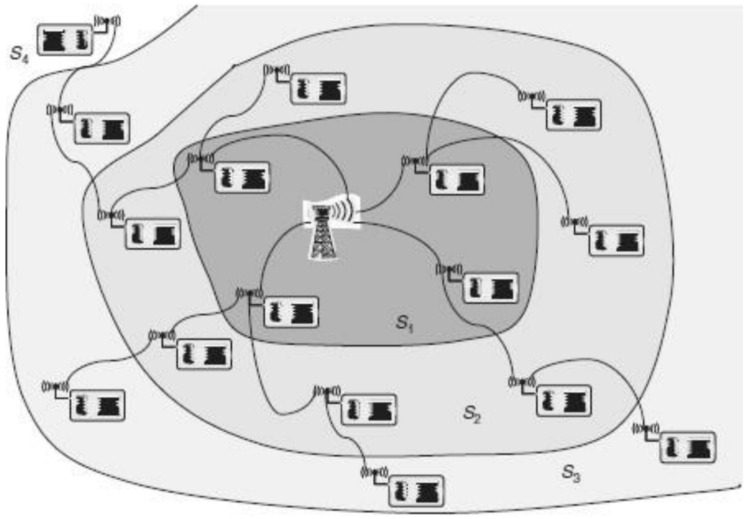


Figure 11.4 Spheres and balls as used by Alonso et al. [19]

Multipath unicast routing***

- Multiple paths between a given source/destination pair
  - *Energy consumption across multiple path is therefore an option worthwhile exploring.*
  - *Fault-tolerance: multiple paths provide redundancy in that they can serve as "hot standbys" to quickly switch to when a node or a link on a primary path fails*

- **Sequential Assignment Routing (SAR):**
  - Problem: *computing such k-disjoint paths requires about k times more overhead than a single-path routing protocol.*
  - SAR: *require paths different neighbors of the sink. constructing trees outward from each sink neighbor; in the end, most nodes will then be part of several such trees*

- **Constructing energy-efficient secondary paths:**
  - Concern:*The energy efficiency of these secondary paths compared to the optimal primary path*
  - For disjoint paths:
    - Primary path: *via its best neighbor toward the data source neighbor.*
    - This alternate path: *forwarded toward the best neighbor that is not already on the primary path.*
  - For braided paths:
    - Require to leave out some nodes of the primary path but are free to use other nodes on the primary path. See figure 11.***5***

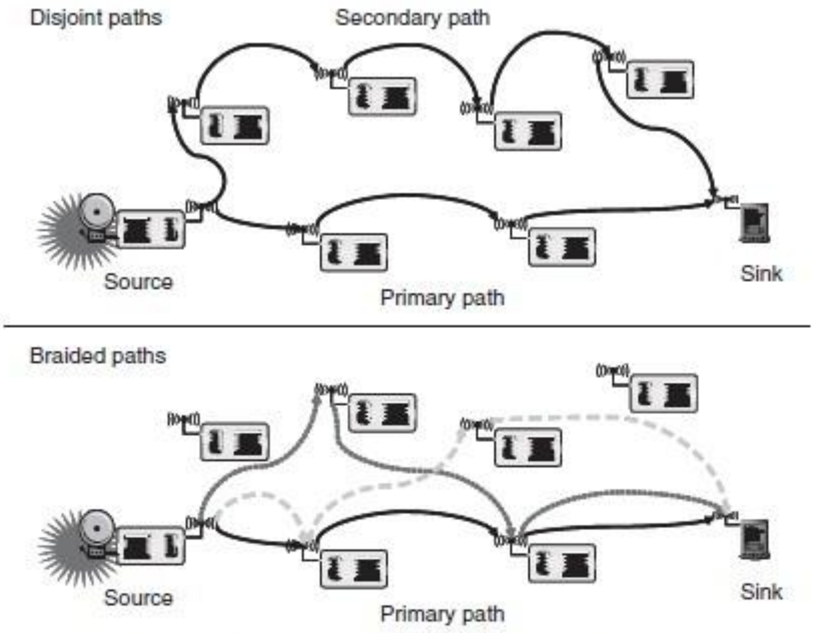


Figure 11.5 Disjoint and braided paths around a primary path

- **Simultaneous transmissions over multiple paths**
  - There is some delay in detecting the need to use a secondary path.
  - The idea:
    - assume node-disjoint paths
    - Send several copies of a given packet over these different paths to the destination.
  - This trades off resource consumption against packet error rates.

- **Randomly choosing** one of **several paths**
  - Each node maintains an energy cost estimate for each of its neighbors.
  - The next hop is randomly chosen proportional to the energy consumption of the path over this neighbor.
- **Trade-off analysis for multicast routing:**
  - Supporting such multiple paths in a network implies a trade-off between robustness and energy efficiency.
  - This tradeoff is analyzed by Krishnamachari et al. who compare the robustness gained by multiple paths with those owing to simply increasing transmission power. Result: Single path with a larger transmission power dominates.

Geographic routing***

- Geographical routing uses location information to formulate an efficient route search toward the destination.
- Geographical routing is very suitable to sensor networks, where data aggregation is a useful technique to minimize the number of transmissions toward the base station by eliminating redundancy among packets from the different sources. Geographic routing addresses these

two issues:

1. Routing packets successfully given any topology

2. Acquiring location information of nodes reflecting the given topology.

- **Two types of Geographical routing:**
  1. ***Geo-casting*****:** sending data to arbitrary nodes in a given region.
  2. ***Position-based routing***: Use position information to aid in routing. In particular in combination with a location service.

Basics of position-based routing

- ***Some simple forwarding strategies:***

1. ***"Most forward within range r" strategy***: In a simple greedy forwarding approach, the packet is forwarded to that neighbor that is located closest to the destination.

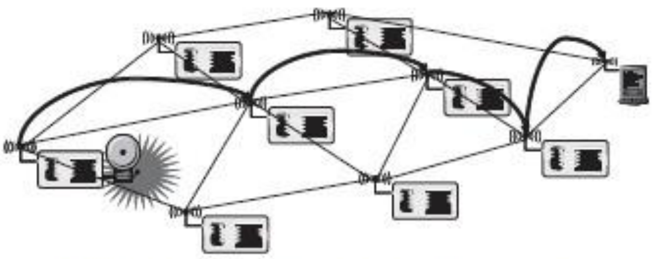


Figure 11.11 Simple greedy geographic forwarding

Figure 11.11 illustrates this scheme and immediately shows one principal shortcoming: in general, not able to find the shortest possible path (in hop count). This trade-off between simplified routing scheme and reduced efficiency is, in general, unavoidable.

2. ***Nearest node with (any) forward progress****:* Idea: Minimize transmission power.
3. ***Directional routing:*** Choose next hop that is angularly closest to destination. Choose next hop that is closest to the connecting line to destination. Problem: Might result in loops!
4. ***The problem of dead ends:*** Simple strategies might send a packet into a dead end. Figure 11.12 illustrates how an obstacle that blocks the direct path between source S and destination D interrupts communication even though S and D are actually connected by the network.

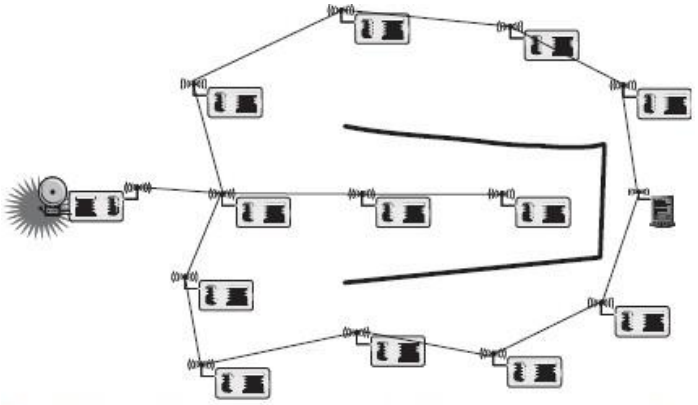


Figure 11.12 Simple greedy geographic forwarding fails in presence of obstacles

5. **Restricted flooding:** Restricted flooding is quite suited to compensate for mobility of the destination.

- Assume: the destination moves at a given speed v and the distance between transmitting node and destination is known, then a source forwards to some of or all of the nodes that are closer to the destination than itself. It is called *geographically restricted flooding.*

6. **Right-hand rule to recover greedy routing – GPSR:** Basic idea to get out of a dead end: Put right hand to the wall, follow the wall.
   - o Does not work if on some inner wall: will walk in circles
   - o Need some additional rules to detect such circles: *Use Geometric Perimeter State Routing (GPSR):* It forwards a packet as long as possible using greedy forwarding with the "most forward" rule.
   - Earlier versions: Compass Routing II, face-2 routing
   - Use greedy, "most forward" routing as long as possible
   - If no progress possible: Switch to "face" routing
     - *Face*: largest possible region of the plane that is not cut by any edge of the graph; can be exterior or interior.
     - Send packet around the face using right-hand rule
     - Use position where face was entered and destination position to determine when face can be left again, switch back to greedy routing
   - Requires: planar graph! (topology control can ensure that)

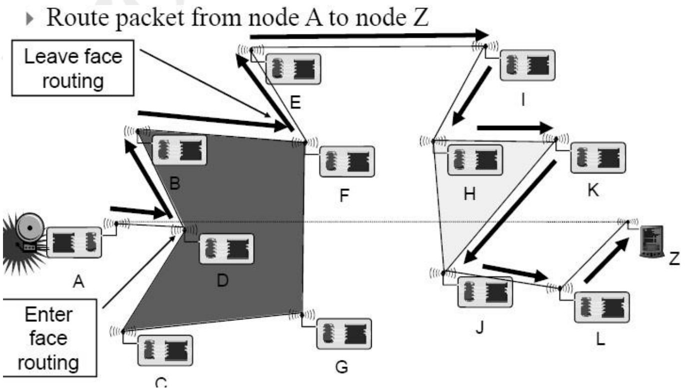


Figure 11.13 Example for GPSR

7. **Performance guarantees of combined greedy/face routing**
   - ▪ Face routing is tasked with routing around obstacles or out of dead ends while greedy routing tries to make quick progress toward the destination.
   - ▪ The first combined greedy/face routing algorithm that is provably worst-case optimal
   - ▪ In order to show the worst-case optimality, quickly switching back to greedy routing could not be used

- the Greedy and (Other Adaptive) Face Routing (GOAFR)+ algorithm that is worst-case optimal and at the same time efficient in the average case
- GOAFR+ algorithm:
  - The algorithm maintains a bounding circle, centered at the destination node, which prevents the face search from needlessly exploring in the wrong direction. A packet maintains two counters, p and q. Counter p contains the number of nodes on the face perimeter that are closer to the destination than is the node where face search started. Counter q counts nodes farther away.

8. **Combination with ID-base routing:** Pure position-based routing in mobile destination node, immediate vicinity can be problematic. Solution: by ID
9. **Randomized forwarding and adaptive node activity – GeRaF :** Here investigate the combination of position-informed, random forwarding and nodes that switch on and off to save energy.
   - *Goal*: Transmit message over multiple hops to destination node; deal with topology constantly changing because of on/off node.
   - *Idea:* Receiver-initiated forwarding
     - Forwarding node S simply broadcasts a packet, without specifying next hop node
     - Some node T will pick it up (ideally, closest to the source) and forward it.
   - *Problem*: How to deal with multiple forwarders?
     - Position-informed randomization: The closer to the destination a forwarding node is, the shorter does it hesitate to forward packet
     - Use several annuli to make problem easier, group nodes according to distance.

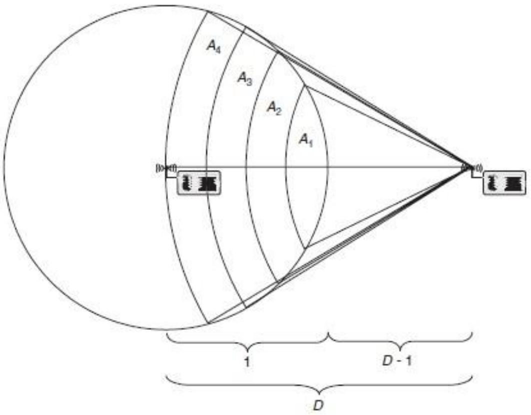


Figure 11.14 Contention regions for selecting the next hop node in GeRaF [941]

10. **Geographic routing without positions – GEM:** Apparent contradiction: geographic, but no position. Use virtual coordinates and preserve enough neighborhood information to be useful in geographic routing. Do not require actual position determination. It has two essential parts:
    1. *Use polar coordinates from a center point*: Assign “virtual angle range” to neighbors of a node.

2. Construct a spanning tree with the center point as the root: Define the radius of a node by the number of hops (in spanning tree)

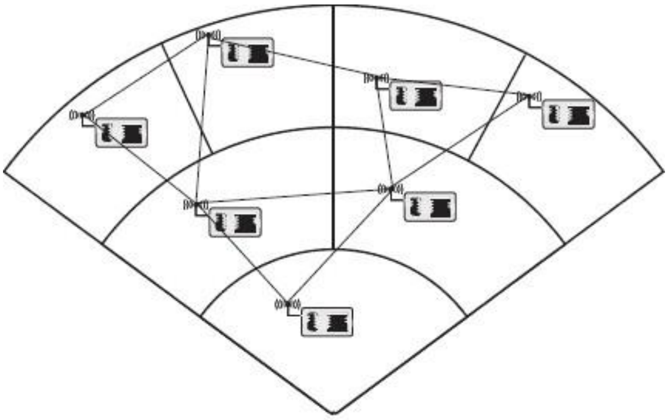


Figure 11.15 Geographic routing with positions [593]

- *Process:*
  - *Choose two nodes in addition to the original root.*
  - *Determine, for each node, the hop count of the shortest path between each of these three nodes(so, total three spanning tree)*
  - *Each node can triangulate its own position in the hop count metric.*

Geocasting

Geocasting: sending data to a subset of nodes that are located in an indicated. Similar to the case of position-based routing, position information of the designated region and the intermediate nodes can be exploited to increase efficiency.

- **Location Based Multicast:** Geocasting by geographically restricted flooding. Define a "forwarding" zone: nodes in this zone will forward the packet to make it reach the destination zone. This zone can be defined in various ways:
  - *Static zone:* smallest rectangle that contains both the source and the entire destination region.
  - *Adaptive zone*: smallest rectangle containing forwarding node and destination zone. Possible dead ends again
  - *Adaptive distances*: packet is forwarded by node u if node u is closer to destination zone's center than predecessor node v (packet has made progress). Packet is always forwarded by nodes within the destination zone itself otherwise.
- **Finding the right direction: Voronoi diagrams and convex hulls**

  Goal: Use that neighbor to forward packet that is closest to destination among all the neighbors. Use Voronoi diagram computed for the set of neighbors of the node currently holding the packet.

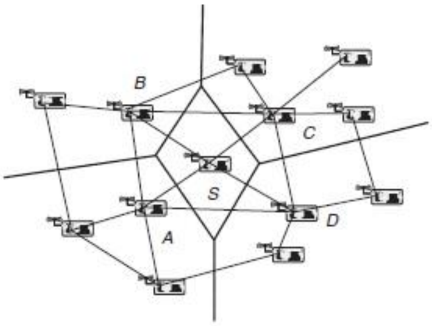


Figure 11.16 Illustration of the Voronoi diagram-based neighbor selection scheme [795] – node S uses the Voronoi cells to decide which neighbor to use for a given destination area

- **Tessellating the plane:**
  - Tessellation: of the plane is a collection of plane figures that fills the plane with no overlaps and no gaps. The first protocol uses a fixed tessellation of the plane into hexagons where each hexagon either has a "manager" in charge of it or is classified as an obstacle to be rooted around.
  - *The second protocol is GeoGRID*: The plane is divided into square grids where each grid has an elected gateway in charge of it. Only those gateway nodes propagate packets among different grids, resulting in a need to control the size of such a grid.
- **Mesh-based geocasting:**
  - *Geocast Adaptive Mesh Environment for Routing (GAMER):* a mesh-based protocol for geocasting. It improves upon other mesh-based geocasting protocols by adapting the density of the created mesh according to the mobility of the nodes in the network.
- **Geocasting using a unicast protocol – GeoTORA:**
  - GeoTORA: All nodes in the destination region act as sinks

    _ Different nodes have different heights above ground.

    _ Destination is the lowest point

    _ No local minimum
- **Trajectory-based forwarding (TBF):** Think in terms of an "agent": Should travel around the network, e.g., collecting measurements

  _ Random forwarding may take a long time

  _ Idea: Provide the agent with a certain trajectory along which to travel Described, e.g., by a simple curve .Forward to node closest to this trajectory

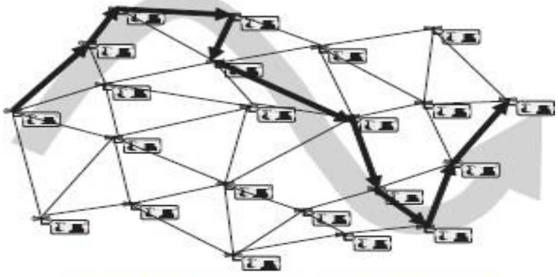


Figure 11.17 Trajectory-based forwarding

Further reading on geographic routing

- ***Impact of localization errors:*** In a real system, it is unrealistic to expect that all nodes know their correct positions.
- ***Location services:*** This service is important for ad hoc or Internet-based geographic information but rarely needed in WSNs. Such "position databases" or "location tables" can be organized centrally or the information can be kept distributed in structures akin to routing tables.
- ***Location-Aided Routing (LAR)***: This protocol uses location information to assist in the flooding phases of standard ad hoc routing protocols. The protocol is similar in many respects to the LBM
- ***Making geocasting energy aware:*** Geographic and Energy Aware Routing (GEAR) is a geocasting scheme that introduces load-splitting among neighbors when forwarding toward the target region, trying to equalize the energy consumption of all nodes.
- ***Geographic routing without geographic coordinates:*** The coordinates used for geographic routing are purely virtual ones and are constructed without actually recurring to the physical location of nodes at all. Another schemes where perimeter nodes do not know their location and show that, even then, virtual coordinates are still useful for geographic routing protocols.

Prepared By:
Prof. Suresha V
Dept. of Electronica and communication Engineering.
Reach me at: suresha.vee@gmail.com
Whatsapp: +91 94485 24399