Internet of Things

IoT Challenges and Solutions, Messaging Protool, ZigBee

Antony Franklin
Networked Wireless Systems Lab
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad

IoT Challenges & Solutions

IoT Challenges & Solutions

- Identification (IPv6, URIs)
- Discovery (Physical Web, mDNS, DNS-SD)
- Communication / Transport (WiFi, Bluetooth, ZigBee)
- Connectivity (6LowPAN, RPL)
- Data Protocols (MQTT, CoAP, AMQP, Websocket, Node)
- Device Management (TR-069, OMA-DM)
- Semantic (JSON-LD, Web of Thing Model)
- Multi-layer Frameworks (Alljoyn, IoTivity, Weave, Homekit)
- Aitificial Intelligence (IoT → IoIT)

Addressing

- ‘Addressing’ IoT devices depends on the network structure (Type of IoT service network)
  - Non dedicated network (“Things” connect to users’ network)
  - Dedicated network connected to the Internet
  - IoT dedicated closed network (Unique Local Address (ULA) of IPv6)
  - Non-IP network with IP gateway, etc. (Unique Local Address (ULA) of IPv6)

Addressing

- EPC → Electronic Product Code (used in product tags)
- Distributed addressing (Used in Zigbee Network → tree like structure in addressing)
- URIs – Use DNS to resolve the names to the IP address.
  - www.iith.ac.in/Auditorium/Light/Temp_sensor1/

Discovery

- One differentiation of IoT with Internet is the Discovery
- New services / devices are hosted very frequently
- Applications need to identify these services automatically
- mDNS - Multicast Domain Name System (mDNS)
  - Service discovery protocol to resolve host names to IP addresses in a local network without using any unicast DNS server.
  - No additional infrastructure or DNS server in the network
  - Uses IP multicast UDP packets through which a node in the local network enquires the names of all other nodes.
- uPnP – universal Plug and Play
- SDSP – Simple Discovery Service Protocol

Communication

- Last mile connectivity for the IoT devices
- WiFi, Bluetooth / BLE, ZigBee, zWave, NB-IoT, LoraWAN, etc.
- Selection of technology depends on the application requirement
    - Low Cost
    - Low power
    - Low data rate
    - Low to long coverage range

Connectivity

- Connecting devices over low power devices
- 6LowPAN – IPv6 over Low Power Personal Area Network
- RPL – IPV6 Routing Protocol for Low Power and Lossy Network

Data Protocols

- Traditional Application Layer Protocols (HTTP) have very high overhead for the IoT devices
  - HTTP works over TCP
  - ASCII header (large size)
  - Pull based One on One (Request Response)
  - Larger code base
- More efficient mechanism are developed in recent times
  - MQTT (**Message Queue Telemetry Transport)**
  - CoAP (Constraint Application Protocol)
  - AMQP (Advanced Message Queueing Protocol)
  - Websocket

Device Management

- Auto-configuration
- Software / Firmware image management
- Software module management
- Status and performance managements
- Diagnostics
- TR-069, OMA-DM (Open Mobile Alliance - Device Management)
    - Provisioning, Device Configuration, Software Upgrades, Fault Management

Semantic

- To enable interoperability for sensors and sensing systems
- Can assist in managing, querying, and combining sensors and observation of data
- Allowing users to operate at abstraction levels above the technical details of format and integration, instead of working with domain concepts and restrictions on quality
- Machine-interpretable semantics allows autonomous or semi-autonomous agents to assist in collecting, processing, reasoning about, and acting on sensors and their observations
- Linked Sensor Data may serve as a means to interlink sensor data with external sources on the Web.

Multi-layer Frameworks

- Extensible and robust architecture that works for smart and thin devices
- Alljoyn framework from Linux Foundation
- IoTivity from Open Interconnect Consortium
- Major functions
    - Device Discovery
    - Data Transmission
    - Device Management
    - Data Management

Artificial Intelligence

- Internet of Things → Internet of Intelligent Things
- AI is the key to unlock the IoT potential
- Simple connect and control → Automated intelligent control

    - Venture capital funding of AI-focused IoT start-ups is growing fast.

    - Acquisitions of AI-focused IoT start-ups are on the rise.

    - Vendors of IoT platforms—Amazon, GE, IBM, Microsoft, Oracle —integrating AI capabilities

    - Large organizations across industries are already leveraging or exploring the power of AI with IoT to deliver new offerings and operate more efficiently

    - Gartner predicts that by 2022, more than 80 percent of enterprise IoT projects will include an AI component, up from only 10 percent as of 2019

Messaging Protocol - MQTT

MQTT

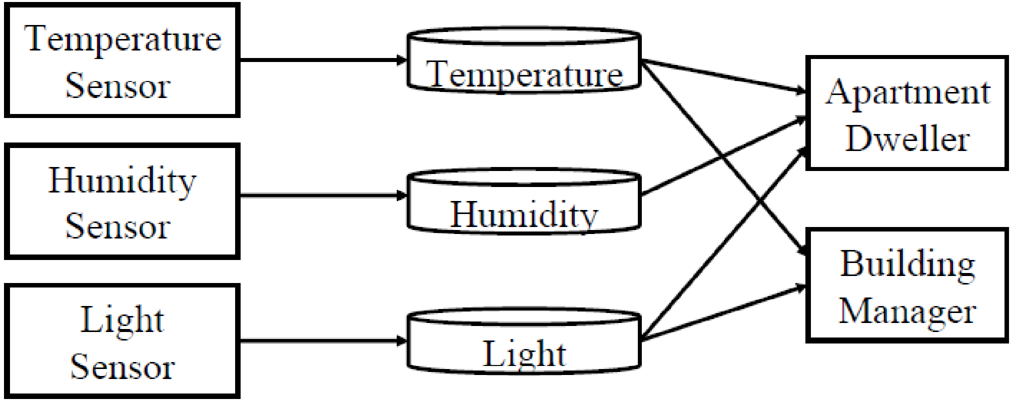
- **Message Queue Telemetry Transport** (ISO standard (ISO/IEC PRF 20922))
- Lightweight messaging protocol for M2M communication
- Telemetry = Tele-Metering = Remote measurements
- Invented and sponsored by IBM (Now Open source, Open Source libraries available)
- MQ originated from “message queueing (MQ)” architecture (used by IBM for service oriented networks). But there is no queueing in MQTT
- Telemetry data goes from devices to a server or broker
- Lightweight = Low network bandwidth and small code footprint

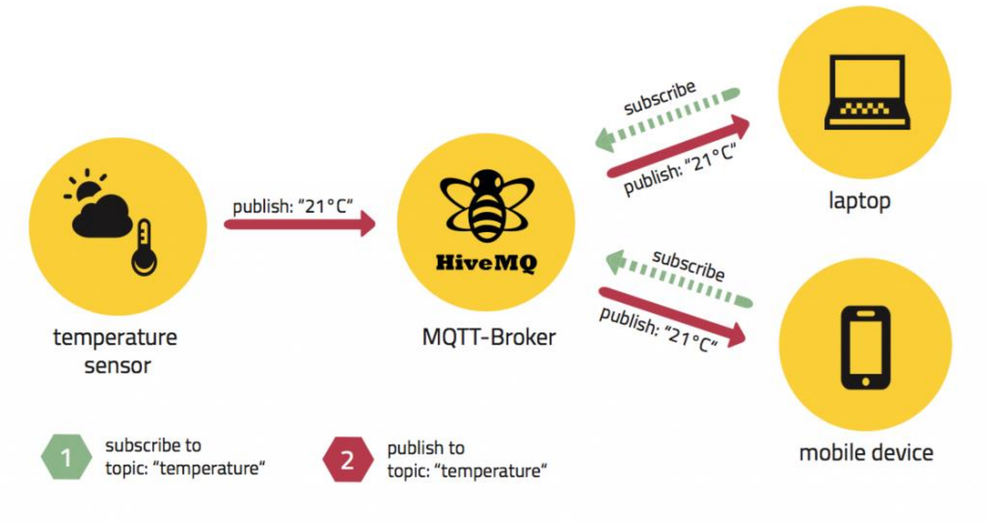
MQTT

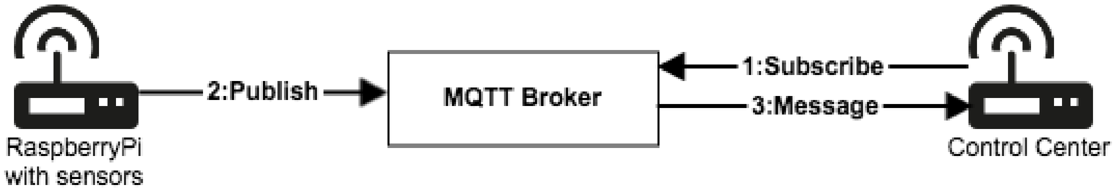
- Designed to provide connectivity (mostly embedded) between applications and middlewares on one side and networks and communications on the other side
- A message broker controls the publish-subscribe messaging pattern
- A topic to which a client is subscribed is updated in the form of messages and distributed by the message broker
- Designed for
  - Remote connections
  - Limited bandwidth
  - Small-code footprint

MQTT Components

- Publishers (Lightweight sensors, Source of data)
- Subscribers (Remote Applications, Interested in the sensor data)
- Brokers (Connects publishers and subscribers)
- Classify sensor data into topics




temperature sensor
publish: "21°C"
HiveMQ
MQTT-Broker
subscribe
publish: "21°C"
laptop
subscribe
publish: "21°C"
mobile device
1
subscribe to topic: "temperature"
2
publish to topic: "temperature"


RaspberryPi with sensors
2:Publish
MQTT Broker
1:Subscribe
3:Message
Control Center

MQTT Communication

- Publish/subscribe is event-driven (enables messages to be pushed to clients)
- The central communication point is the MQTT broker (in charge of dispatching all messages between the senders and the rightful receivers)
- Client include a topic to the message while publishing to the broker - it is the routing information for the broker
- Clients want to receive messages subscribe to the topics
  - Broker delivers all messages with the matching topic to the client
  - The clients don’t have to know each other (communicate over the topic)
- This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers

MQTT Communication

- Quality of Service Levels (Three levels)
  - 0 = At most once (Best effort, No Ack) → Fire and Forget
  - 1 = At least once (Acked, retransmitted if ack not received)
  - 2 = Exactly once [Request to send (Publish), Clear-to-send (Pubrec), message (Pubrel), ack (Pubcomp)]
- Retained Messages: Server keeps messages even after sending it to all subscribers (New subscribers get the retained messages)

MQTT Communication

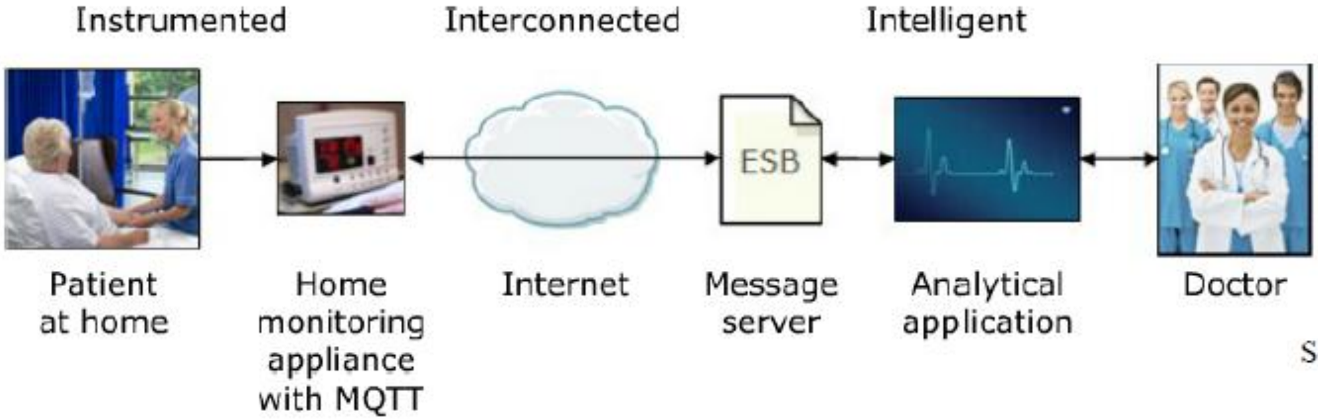
- Clean Sessions and Durable Connections: At connection setup:
  - Clean session flag → all subscriptions are removed on disconnect
  - Otherwise subscriptions remain in effect after disconnection
  - Subsequent messages with high QoS are stored for delivery after reconnection
- Wills: At connection a client can inform that it has a will or a message that should be published if unexpected disconnection
  - Alarm if the client loses connection
- Periodic keep alive messages → If a client is still alive

MQTT Topics

- A topic is a simple string that can have more hierarchy levels, separated by a /
- A sample topic for sending temperature data of the living room could be **house/living-room/temperature**
- The client can subscribe to the exact topic or it can use a wildcard
- The subscription to house/+/temperature would result in all messages sent to the previously mentioned topic house/livingroom/temperature, as well as any topic with an arbitrary value in the place of living room, such as house/kitchen/temperature
- The + is a single level wildcard and only allows arbitrary values for one hierarchy
- If more than one level needs to be subscribed, such as, the entire sub-tree, there is also a multilevel wildcard (#)
- For example house/# is subscribing to all topics beginning with house

MQTT Application Example

- ❖ Home pacemaker monitoring solution
  - ➢ Sensors on patient
  - ➢ Collected by a monitoring equipment in home (broker) using MQTT
- ❖ Subscribed by a computer in the hospital
- ❖ Alerts the doctor if anything is out-of-order



Source: Lampkin 2012

Popular MQTT Applications

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

SMQTT

- Secure MQTT is an extension of MQTT which uses encryption based on lightweight attribute based encryption
- Broadcast encryption feature → message is encrypted and delivered to multiple other nodes, which is quite common in IoT applications
- The mechanism consists of setup, encryption, publish and decryption
  - In the setup phase, the subscribers and publishers register themselves to the broker and get a master secret key (developer's choice of key generation algorithm)
  - When the data is published, it is encrypted and published by the broker which sends it to the subscribers, which is finally decrypted at the subscriber end having the same master secret key
- The key generation and encryption algorithms are not standardized

MQTT Vs HTTP

	MQTT	HTTP
Design	Data Centric	Document Centric
Pattern	Publisch / Subscribe	Request / Response
Complexity	Simple	More Complex
Message Size	Small Binary with 2B Header	Large ASCII
Libraries	30kB in C, 100kB in Java	Large
Data Distribution	1 to zero, one, or n	1 to 1 only

Constrained Application Protocol

CoAP - Constrained Application Protocol

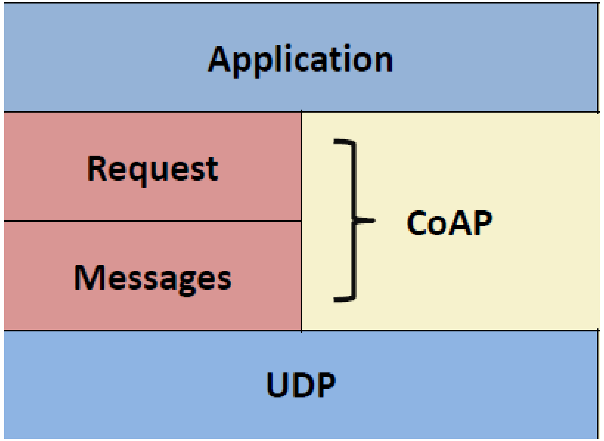
- Web transfer protocol for use with constrained nodes and networks
- Designed for Machine to Machine (M2M) applications such as smart energy and building automation
- Based on Request-Response model between end-points
- Client-Server interaction is asynchronous over a datagram oriented transport protocol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

CoAP - Constrained Application Protocol

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

CoAP

- Built over UDP, instead of TCP and has a light mechanism to provide reliability
- CoAP architecture is divided into two main sub-layers
  - Messaging - for reliability and duplication of messages
  - Request/response - for communication
- CoAP has four messaging modes:
  - Confirmable
  - Non-confirmable
  - Piggyback
  - Separate

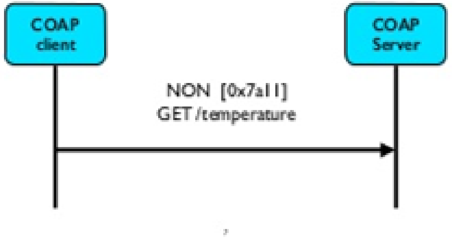


CoAP

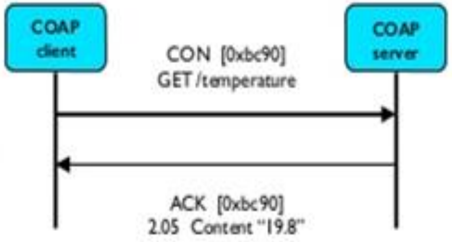
- Confirmable and Non-confirmable → reliable transmission vs unreliable transmissions
- Piggyback and Separate modes → for request/response
- Piggyback → used for client/server direct communication where the server sends its response directly after receiving the message, i.e., within the acknowledgment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

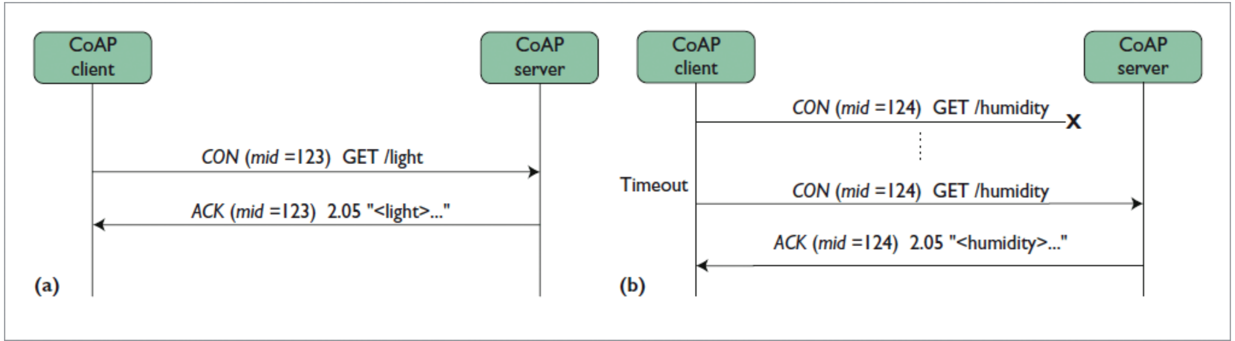
CoAP Request-Response Model

Non Confirmed Request



Confirmed Request




CoAP client
CoAP server
CON (mid =123) GET /light
ACK (mid =123) 2.05 "<light>..."
(a)
CoAP client
CoAP server
CON (mid =124) GET /humidity
X
Timeout
CON (mid =124) GET /humidity
ACK (mid =124) 2.05 "<humidity>..."
(b)

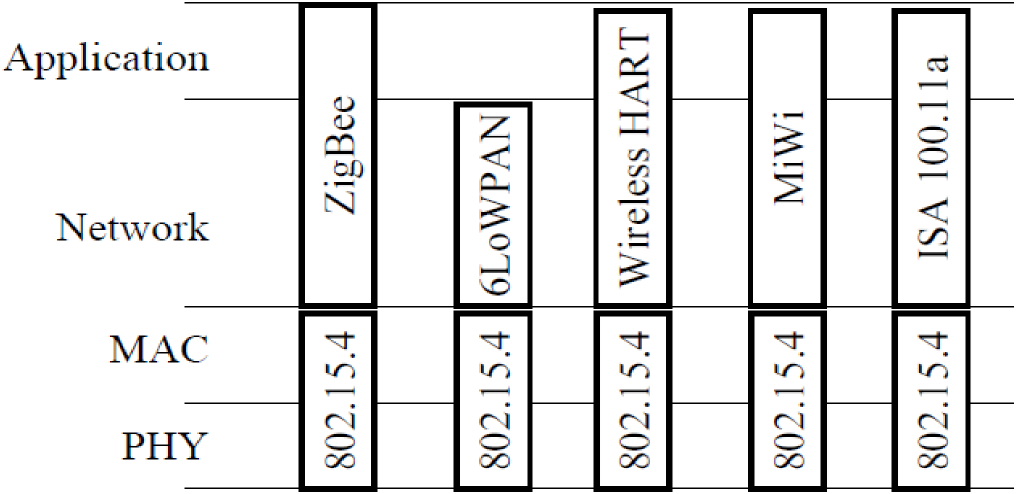
CoAP - Features

- Reduced overheads and parsing complexity
- URL and content-type support
- Support for the discovery of resources provided by known CoAP services
- Simple subscription for a resource, and resulting push notifications
- Simple caching based on maximum message age

IEEE 802.15.4 MAC/PHY

IEEE 802.15.4

- Used by several IoT protocols
- ZigBee, 6LowPAN, Wireless HART, MiWi, and ISA 100.11a



Application

Network

MAC

PHY

ZigBee

6LoWPAN

Wireless HART

MiWi

ISA 100.11a

802.15.4

802.15.4

802.15.4

802.15.4

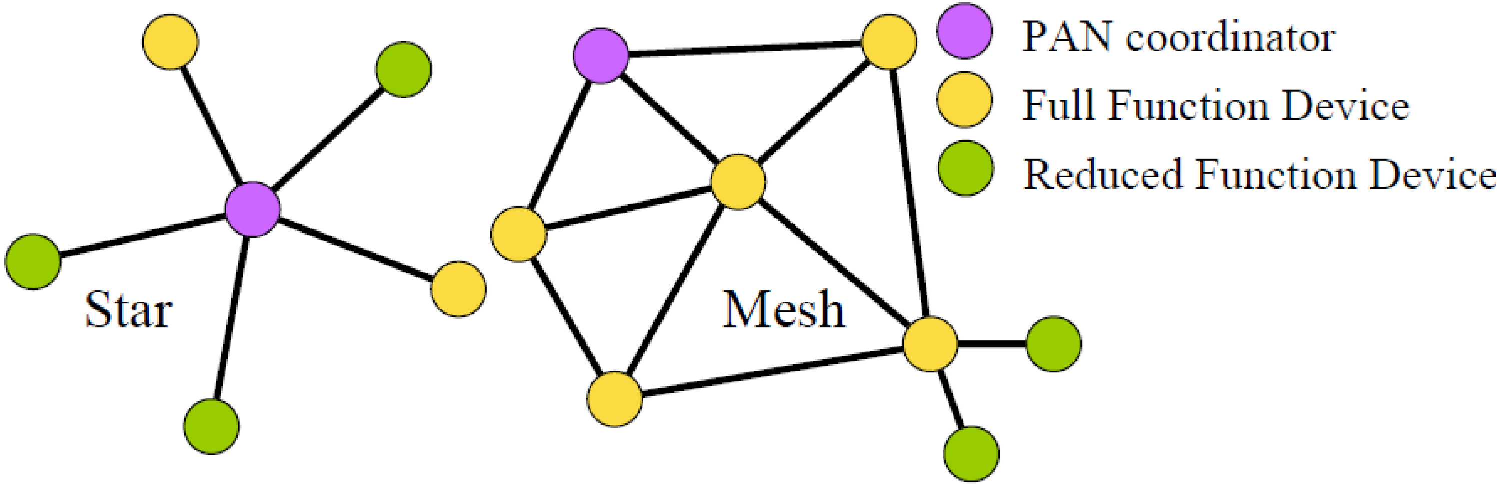
802.15.4

IEEE 802.15.4 Overview

- Low Rate Wireless Personal Area Network (LR-WPAN)
- 2.4 GHz (most common). 16 5-MHz channels
- 250 kbps PHY → 50 kbps application data rate
- Peak data rate depends upon symbol rate → multilevel 4b/symbol
- Similar to 802.11: Direct Sequence Spread Spectrum, CSMA/CA, Backoff, Beacon, Coordinator (similar to Access point)
- Lower rate, short distance → Lower power → Low energy
- Each node has a 64-bit Extended Unique ID (EUI-64)
  - 48bit MAC address + 16 bit
- No segmentation/reassembly. Max MAC frame size is 127 bytes with a payload of 77+ bytes.

IEEE 802.15.4 Topologies

- ❖ Star and Peer to Peer
- ❖ Two types of devices
  - ➢ Full Function device (FFD)
  - ➢ Reduced Function device (RFD)

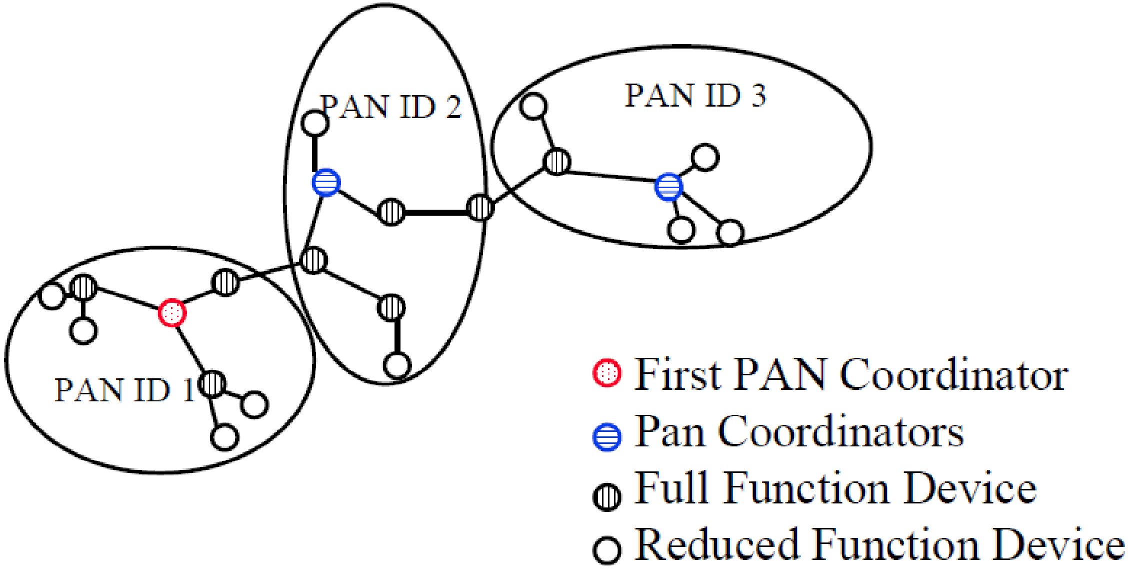


Coordinator

- FFDs can become coordinator and can also route messages to other nodes
- RFDs cannot become coordinator and can only be a leaf
- FFD that starts a PAN becomes the coordinator
- In star topology, all communication is to/from the coordinator
- In P2P topology, FFDs can communicate directly also
- Each PAN has a PAN ID and is called a cluster
- Nodes join a cluster by sending association request to the coordinator. Coordinator assigns a 16-bit short address to the device. Devices can use either the short address or EUI-64 address.

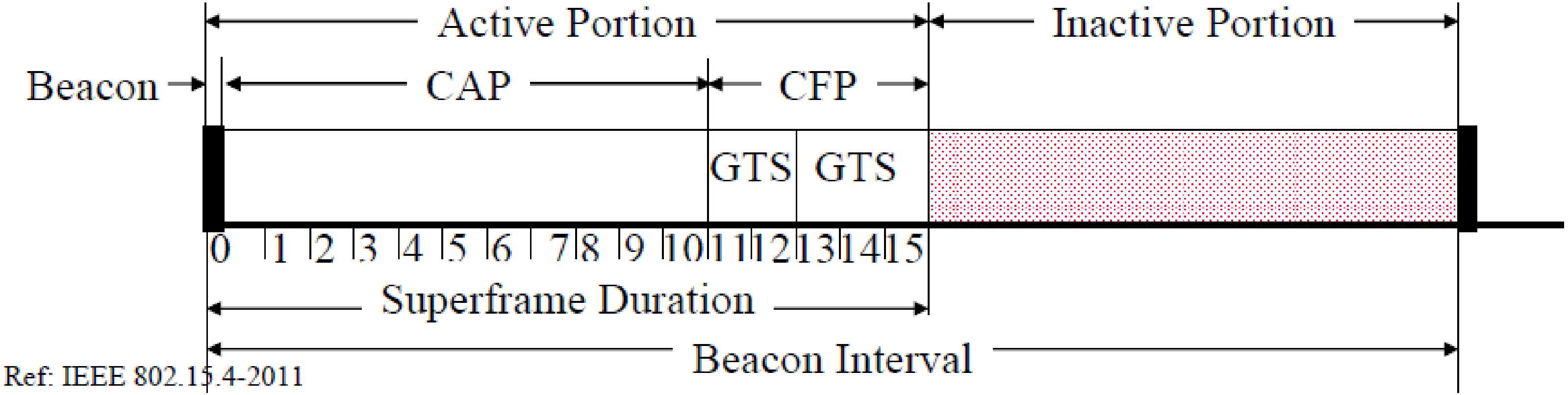
Cluster Tree Network

- A coordinator can ask another FFD to become a coordinator for a subset of nodes. Tree → No loops



IEEE 802.15.4 MAC

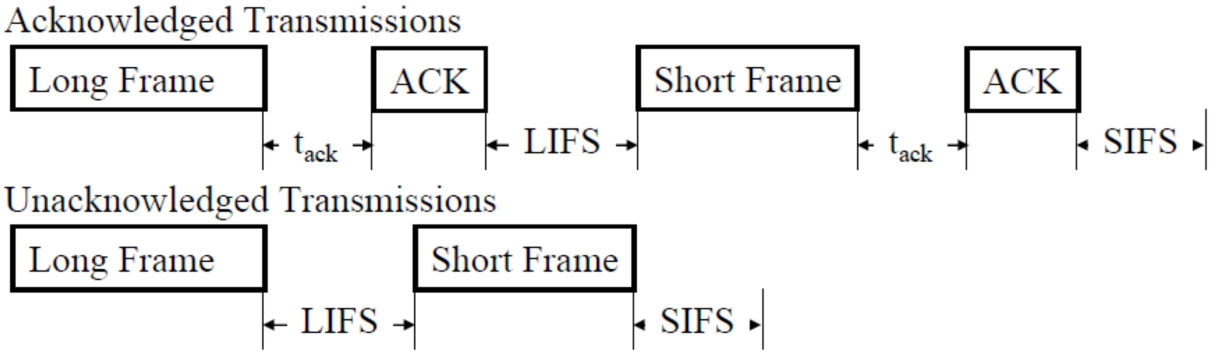
- Beacon-Enabled CSMA/CA
- Coordinator sends out beacons periodically
- Part of the beacon interval is inactive → Everyone sleeps
- Active interval consists of 16 slots
- Guaranteed Transmission Services (GTS): For real-time services. Periodic reserved slots.



Ref: IEEE 802.15.4-2011

IEEE 802.15.4 MAC

- Beaconless Operation: Unslotted CSMA
  - If coordinator does not send beacons, there are no slots
- Acknowledgements if requested by the sender.
- Short inter-frame spacing (SIFS) if previous transmission is shorter than a specified duration
- Otherwise, Long inter-frame spacing (LIFS)



802.15.4 CSMA/CA

- Wait until the channel is free.
- Wait a random back-off period If the channel is still free, then transmit.
- If the channel is busy, backoff again.
- Backoff exponent limited to 0-2 in battery life extension mode.
- Acknowledgement and Beacons are sent without CSMA-CA.

IEEE 802.15.4e Enhancements

- Low latency deterministic operation: pre-assigned slots
- Channel adaptation: Different channels used by different nodes for contention free period
- Time slotted channel hopping: Higher layers coordinate the slot allocation along with its frequency. Good for harsh industrial environments.
- Each device can select its listening channel
- Transmitter and receiver coordinate their cycles (very low duty cycle)
- Transmit only when requested by receiver

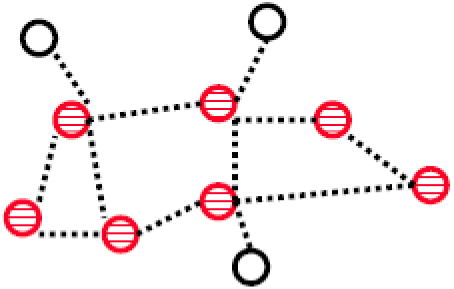
ZigBee

ZigBee Overview

- Industrial monitoring and control applications requiring small amounts of data, turned off most of the time (<1% duty cycle), e.g., wireless light switches, meter reading, patient monitoring
- Ultra-low power, low-data rate, multi-year battery life
- Power management to ensure low power consumption
- Less Complex. 32kB protocol stack vs 250kB for Bluetooth
- Range: 1 to 100 m, up to 65000 nodes.
- Tri-Band:
  - 16 Channels at 250 kbps in 2.4GHz ISM
  - 10 Channels at 40 kb/s in 915 MHz ISM band
  - One Channel at 20 kb/s in European 868 MHz band

ZigBee Overview

- IEEE 802.15.4 MAC and PHY. Higher layer and interoperability by ZigBee Alliance
- Up to 254 devices or 64516 simpler nodes
- Named after zigzag dance of the honeybees - Direction of the dance indicates the location of food
- Multi-hop ad-hoc mesh network
  - Multi-Hop Routing: message to non-adjacent nodes
  - Ad-hoc Topology: Nodes discover each other
  - Mesh Routing: End-nodes help route messages for others
  - Mesh Topology: Loops possible



Features

- Stochastic addressing: A device is assigned a random address and announced. Mechanism for address conflict resolution. Parents don’t need to maintain assigned address table.
- Link Management: Each node maintains quality of links to neighbors. Link quality is used as link cost in routing.
- Frequency Agility: Nodes experience interference report to channel manager (e.g., trust center), which then selects another channel
- Multicast
- Many-to-One Routing: To concentrator
- Asymmetric Link: Each node has different transmit power and sensitivity. Paths may be asymmetric.
- Fragmentation and Reassembly

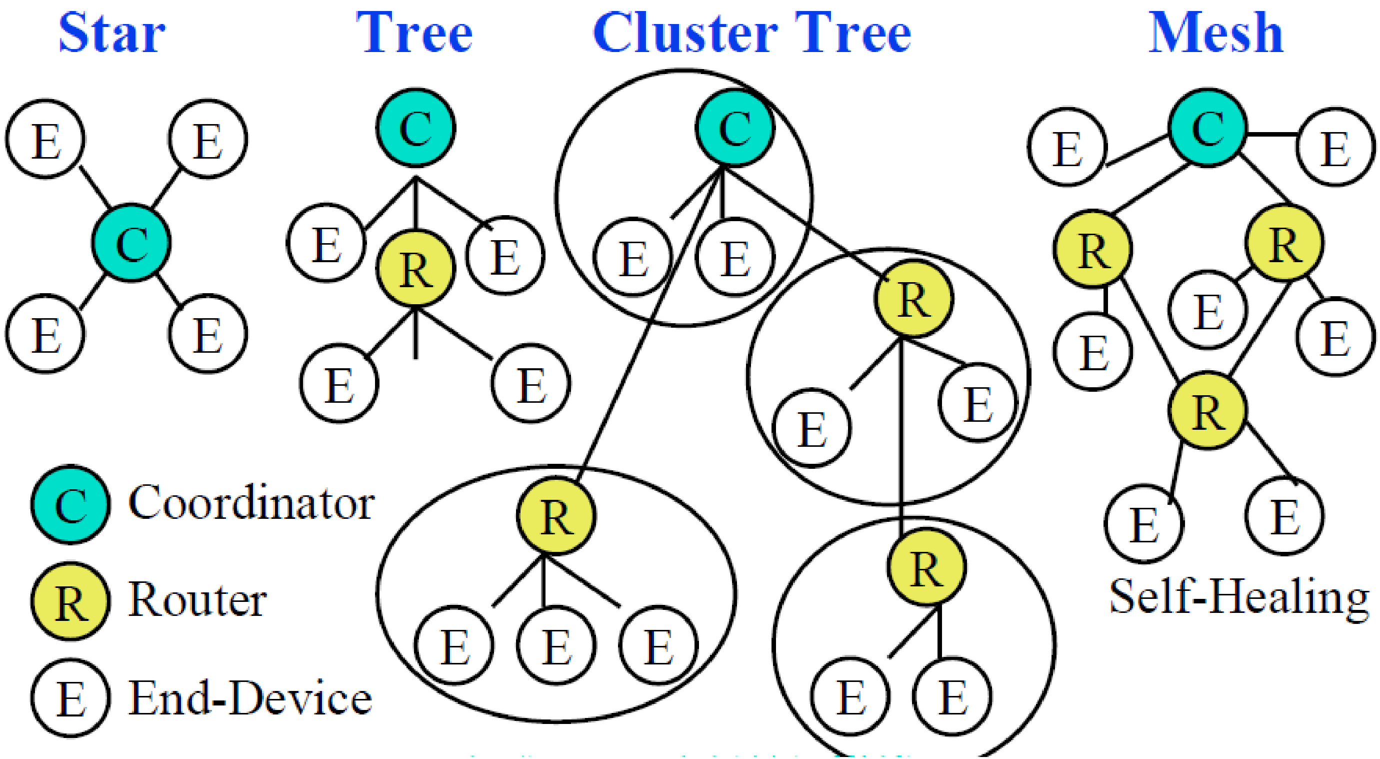
Features

- Power Management: Routers and Coordinators use main power. End Devices use batteries.
- Security: Standard and High End-Devices get new security key when they wake up.

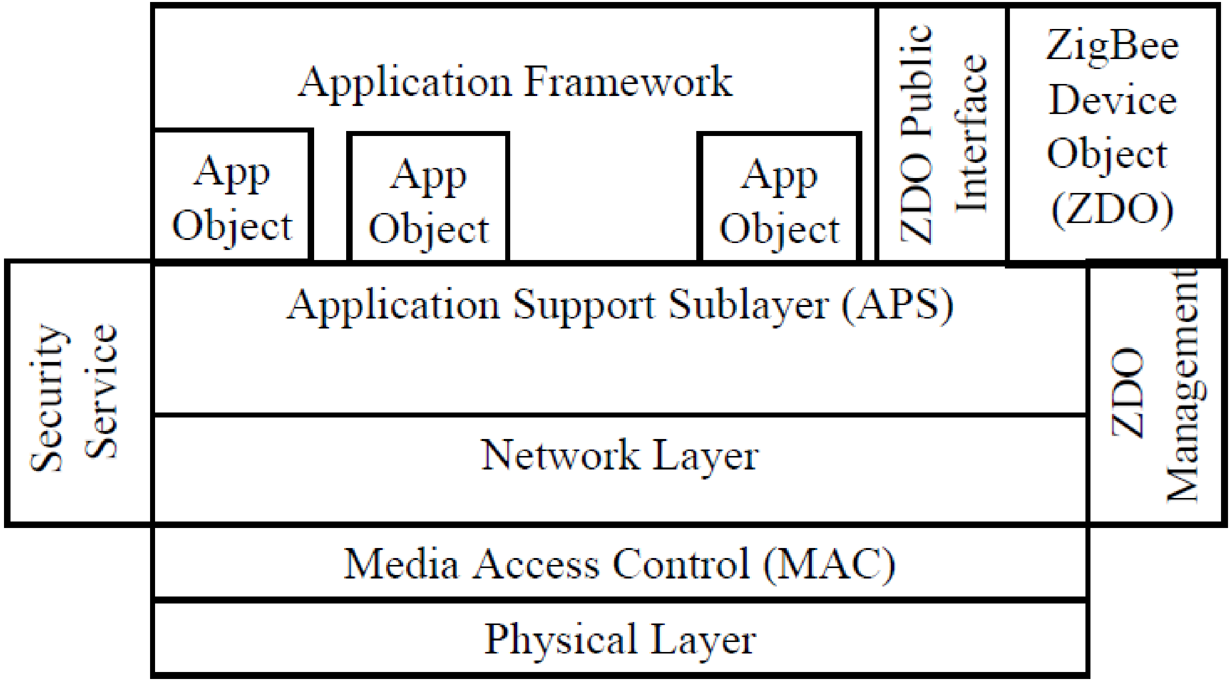
ZigBee Device Types

- Coordinator: Selects channel, starts the network, assigns short addresses to other nodes, transfers packets to/from other nodes
- Router: Transfers packets to/from other nodes
- Full-Function Device: Capable of being coordinator or router
- Reduced-Function Device: Not capable of being a coordinator or a router (Leaf node)
- ZigBee Trust Center (ZTC): Provides security keys and authentication
- ZigBee Gateway: Connects to other networks, e.g., WiFi, 3G/4G

ZigBee Topologies



ZigBee Protocol Architecture



ZigBee Protocol Architecture

- Application Objects: E.g., Remote control application. Also referred to as End-Point (EP).
- End-Node: End device.
- Each node can have up to 250 application objects.
- ZigBee Device Object (ZDO): Control and management of application objects. Initializes coordinator, security service, device and service discovery
- Application Support Layer (APS): Serves application objects.
- Network Layer: Route Discovery, neighbor discovery
- ZDO Management
- Security Service

ZigBee Application Layer

- Application layer consists of application objects (aka end points) and ZigBee device objects (ZDOs)
- 256 End Point Addresses:
  - 240 application objects: Address EP1 through EP240
  - ZDO is EP0
  - End Points 241-254 are reserved
  - EP255 is broadcast
- Each End Point has one application profile, e.g., light on/off profile
- ZigBee forum has defined a number of profiles. Users can develop other profiles
- Attributes: Each profile requires a number of data items. Each data item is called an “attribute” and is assigned an 16-bit “attribute ID” by ZigBee forum

ZigBee Application Layer

- Clusters: A collection of attributes and commands on them. Each cluster is represented by a 16-bit ID. Commands could be read/write requests or read/write responses
- Cluster Library: A collection of clusters. ZigBee forum has defined a number of cluster libraries (e.g., on/off, level control, alarms, etc.)
- Binding: Process of establishing a logical relationship (parent, child, ..)
- ZDO:
  - Uses device and service discovery commands to discover details about other devices.
  - Uses binding commands to bind and unbind end points.
  - Uses network management commands for network discovery, route discovery, link quality indication, join/leave requests

ZigBee Application Profiles

- Smart Energy: Electrical, Gas, Water Meter reading
- Commercial Building Automation: Smoke Detectors, lights, …
- Home Automation: Remote control lighting, heating, doors, …
- Personal, Home, and Hospital Care (PHHC): Monitor blood pressure, heart rate, …
- Telecom Applications: Mobile phones
- Remote Control for Consumer Electronics: In collaboration with Radio Frequency for Consumer Electronics (RF4CE) alliance
- Industrial Process Monitoring and Control: temperature, pressure, position (RFID), …
- Many others

ZigBee Address Assignment

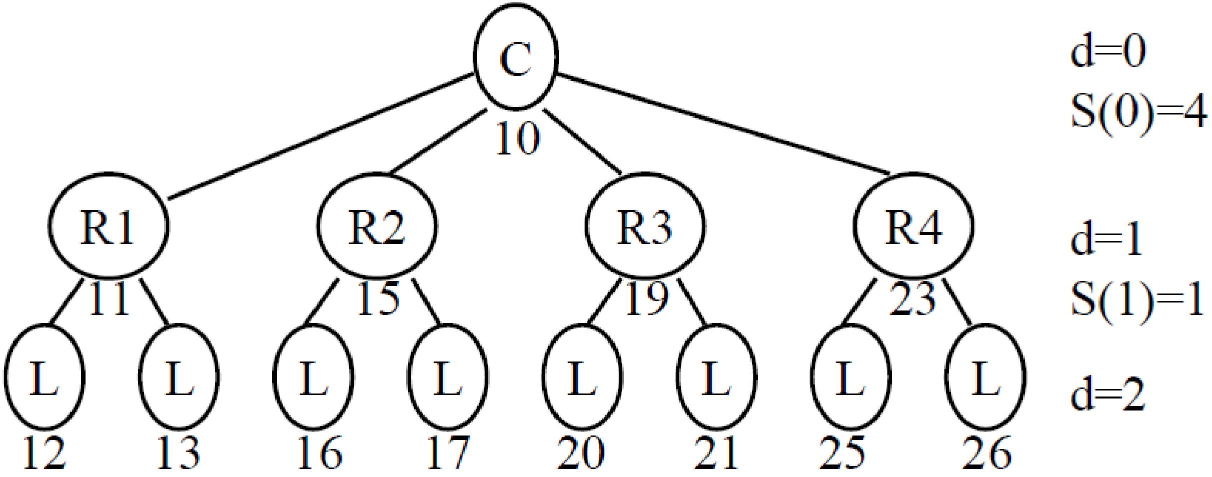
- Each node gets a unique 16-bit address
- Two Schemes: Distributed and Stochastic
- Distributed Scheme: Good for tree structure
  - Each child is allocated a sub-range of addresses.
  - Limit on maximum depth (L)
  - Maximum No. of children per parent (C) and Maximum number of routers (R)
  - Address of the nth child is - Parent+(n-1)S(d)

$$S(d) = \begin{cases} 1 + C(L - d) & \text{if } R = 1 \\ \frac{CR^{L-d-1} - 1 - C + R}{R - 1} & \text{if } R > 1 \end{cases}$$

Distributed Scheme Example

- Max depth L=2, Routers R=4, Children C=3
- Coordinator: d=0. Skip

$$S(0) = \frac{CR^{L-d-1} - 1 - C + R}{R - 1} = \frac{3 \times 4^{2-0-1} - 1 - 3 + 4}{4 - 1} = 4$$



Distributed Scheme Example

- Assume the address of coordinator is 10 (decimal)
  - Address of R1 = 10+1 = 11
  - Address of R2 = 10+1+S(0) = 11+4=15
  - Address of R3 = 10+1+2*S(0) = 11+8 = 19
  - Address of R3 = 10+1+3*S(0) = 11+12 =23
- Routers R1-R4 compute S(1):

$$S(1) = \frac{CR^{L-d-1} - 1 - C + R}{R - 1} = \frac{3 \times 4^{2-1-1} - 1 - 3 + 4}{4 - 1} = 1$$

- Children of R1 are assigned 12 and 13
- Children of R2 are assigned 16 and 17

Stochastic Address Assignment

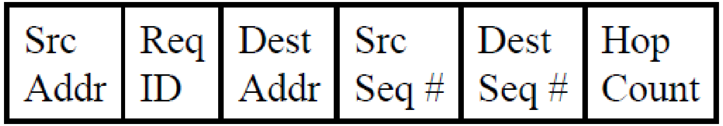
- Parent draws a 16 bit random number between 0 and 216-1 and assigns it to a new child. A new number is drawn if the result is all-zero (null) or all-one (broadcast). So the assigned address is between 1 and 216-2.
- Parent then advertises the number to the network
- If another node has that address an address conflict message is returned and the parent draws another number and repeats
- There is no need to pre-limit # of children or depth

ZigBee Routing

- Ad-Hoc On-Demand Distance Vector (AODV)
- Dynamic Source Routing (DSR)
- Tree Hierarchical Routing
- Many-to-one routing

AODV

- Ad-hoc On-demand Distance Vector Routing
- On-demand → Reactive → Construct a route when needed
- Routing Table: Path is not stored. Only next hop
    - Entry = <destination, next node, "sequence #" (timestamp)>
- Route Discovery: Flood a route request (RREQ) to all neighbors. Neighbors broadcast to their neighbors

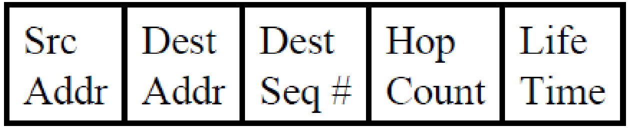


Src Addr	Req ID	Dest Addr	Src Seq #	Dest Seq #	Hop Count

- Request ID is the RREQ serial number. Used to discard duplicates.
- Source sequence # is a clock counter incremented when RREQ is sent.
- Destination sequence # is the most recent sequence from the destination that the source has seen. Zero if unknown.

AODV

- ❖ Intermediate nodes can reply to RREQ only if they have a route to destination with higher destination sequence #
- ❖ Route reply (RREP) comes back “unicast” on the reverse path

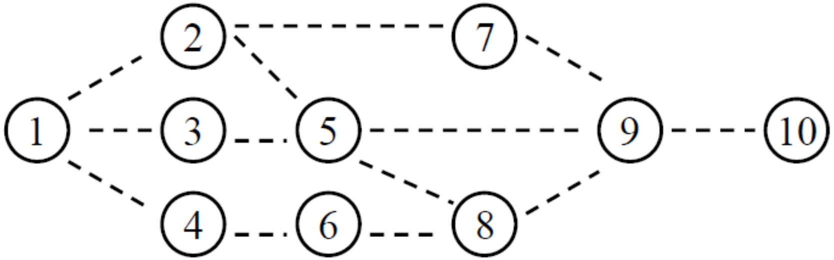


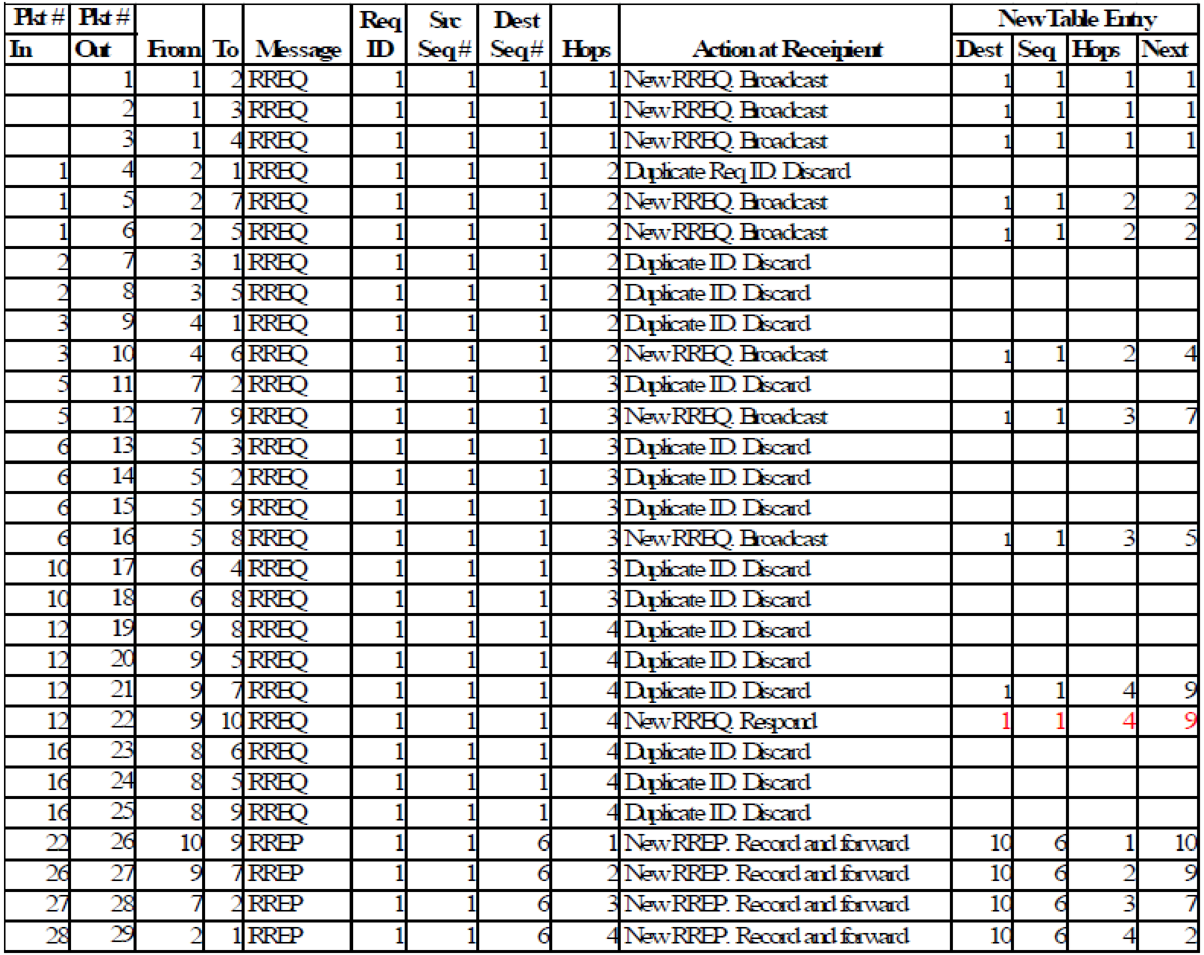
Src Addr	Dest Addr	Dest Seq #	Hop Count	Life Time

  - ➢ Destination Sequence # is from Destination’s counter
  - ➢ Lifetime indicates how long the route is valid
  - ➢ Intermediate nodes record node from both RREP and RREQ if it has a lower cost path → the reverse path
  - ➢ Backward route to Destination is recorded if sequence number is higher or if sequence number is same and hops are lower
  - ➢ Old entries are timed out
  - ➢ AODV supports only symmetric links

AODV

- Node 1 broadcasts RREQ to 2, 3, 4:
- "Any one has a route to 10 fresher than 1. This is my broadcast #1"
- Node 2 broadcasts RREQ to 1, 5, 7
- Node 3 broadcasts RREQ to 1, 5
- Node 4 broadcasts RREQ to 1, 6

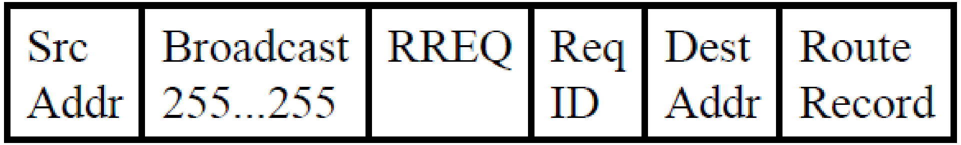




Pkt #	Pkt #				Req	Src	Dest			New Table Entry			
In	Out	From	To	Message	ID	Seq#	Seq#	Hops	Action at Receipient	Dest	Seq	Hops	Next
	1	1	2	RREQ	1	1	1	1	New RREQ. Broadcast	1	1	1	1
	2	1	3	RREQ	1	1	1	1	New RREQ. Broadcast	1	1	1	1
	3	1	4	RREQ	1	1	1	1	New RREQ. Broadcast	1	1	1	1
1	4	2	1	RREQ	1	1	1	2	Duplicate Req ID. Discard				
1	5	2	7	RREQ	1	1	1	2	New RREQ. Broadcast	1	1	2	2
1	6	2	5	RREQ	1	1	1	2	New RREQ. Broadcast	1	1	2	2
2	7	3	1	RREQ	1	1	1	2	Duplicate ID. Discard				
2	8	3	5	RREQ	1	1	1	2	Duplicate ID. Discard				
3	9	4	1	RREQ	1	1	1	2	Duplicate ID. Discard				
3	10	4	6	RREQ	1	1	1	2	New RREQ. Broadcast	1	1	2	4
5	11	7	2	RREQ	1	1	1	3	Duplicate ID. Discard				
5	12	7	9	RREQ	1	1	1	3	New RREQ. Broadcast	1	1	3	7
6	13	5	3	RREQ	1	1	1	3	Duplicate ID. Discard				
6	14	5	2	RREQ	1	1	1	3	Duplicate ID. Discard				
6	15	5	9	RREQ	1	1	1	3	Duplicate ID. Discard				
6	16	5	8	RREQ	1	1	1	3	New RREQ. Broadcast	1	1	3	5
10	17	6	4	RREQ	1	1	1	3	Duplicate ID. Discard				
10	18	6	8	RREQ	1	1	1	3	Duplicate ID. Discard				
12	19	9	8	RREQ	1	1	1	4	Duplicate ID. Discard				
12	20	9	5	RREQ	1	1	1	4	Duplicate ID. Discard				
12	21	9	7	RREQ	1	1	1	4	Duplicate ID. Discard	1	1	4	9
12	22	9	10	RREQ	1	1	1	4	New RREQ. Respond	1	1	4	9
16	23	8	6	RREQ	1	1	1	4	Duplicate ID. Discard				
16	24	8	5	RREQ	1	1	1	4	Duplicate ID. Discard				
16	25	8	9	RREQ	1	1	1	4	Duplicate ID. Discard				
22	26	10	9	RREP	1	1	6	1	New RREP. Record and forward	10	6	1	10
26	27	9	7	RREP	1	1	6	2	New RREP. Record and forward	10	6	2	9
27	28	7	2	RREP	1	1	6	3	New RREP. Record and forward	10	6	3	7
28	29	2	1	RREP	1	1	6	4	New RREP. Record and forward	10	6	4	2

Dynamic Source Routing (DSR)

- On-Demand (reactive) routing using "Source Route“
- Source Route = List of routers along the path in the packet.
- Routing database: Complete route to recent destinations
- Each entry has an expiration period and is timed out
- If a route is not available, send "route request" to all neighbors



Src Addr	Broadcast 255...255	RREQ	Req ID	Dest Addr	Route Record

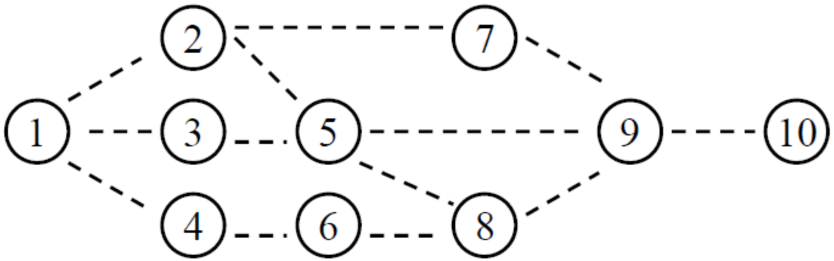
- Each neighbor adds itself to the route in the request and forward to all its neighbors (only first receipt). Does not change source address.
- If a node knows the route it appends the rest of the route and returns the “route reply (RREP)”
- RREP goes back along the recorded path
- All nodes record paths in RREP and RREQ. Multiple routes cached.

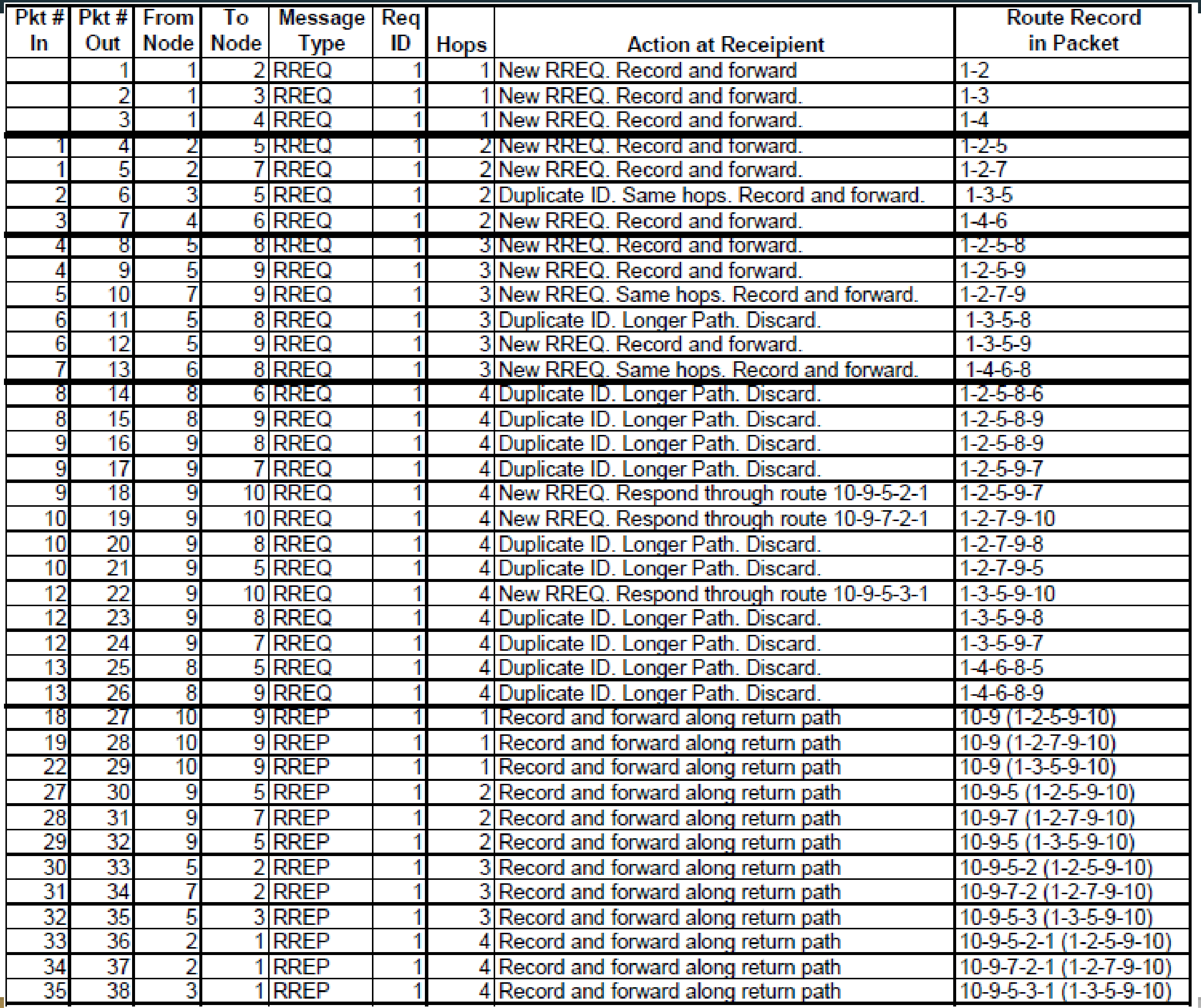
Dynamic Source Routing (DSR)

- Node 1 sends RREQ to 2, 3, 4:

  "Any one has a route to 10”

- Nodes 2 send RREQ to 5, 7. Note: RREQ not sent to 1.
- Node 3 sends RREQ to 5
- Node 4 sends RREQ to 6





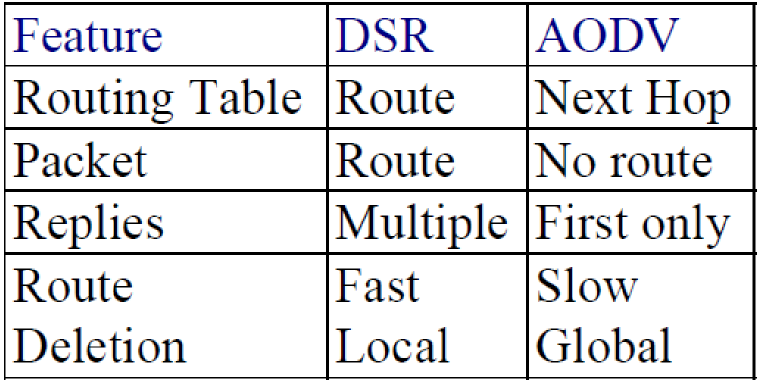
Pkt # In	Pkt # Out	From Node	To Node	Message Type	Req ID	Hops	Action at Receipient	Route Record in Packet
	1	1	2	RREQ	1	1	New RREQ. Record and forward	1-2
	2	1	3	RREQ	1	1	New RREQ. Record and forward.	1-3
	3	1	4	RREQ	1	1	New RREQ. Record and forward.	1-4
1	4	2	5	RREQ	1	2	New RREQ. Record and forward.	1-2-5
1	5	2	7	RREQ	1	2	New RREQ. Record and forward.	1-2-7
2	6	3	5	RREQ	1	2	Duplicate ID. Same hops. Record and forward.	1-3-5
3	7	4	6	RREQ	1	2	New RREQ. Record and forward.	1-4-6
4	8	5	8	RREQ	1	3	New RREQ. Record and forward.	1-2-5-8
4	9	5	9	RREQ	1	3	New RREQ. Record and forward.	1-2-5-9
5	10	7	9	RREQ	1	3	New RREQ. Same hops. Record and forward.	1-2-7-9
6	11	5	8	RREQ	1	3	Duplicate ID. Longer Path. Discard.	1-3-5-8
6	12	5	9	RREQ	1	3	New RREQ. Record and forward.	1-3-5-9
7	13	6	8	RREQ	1	3	New RREQ. Same hops. Record and forward.	1-4-6-8
8	14	8	6	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-2-5-8-6
8	15	8	9	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-2-5-8-9
9	16	9	8	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-2-5-8-9
9	17	9	7	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-2-5-9-7
9	18	9	10	RREQ	1	4	New RREQ. Respond through route 10-9-5-2-1	1-2-5-9-7
10	19	9	10	RREQ	1	4	New RREQ. Respond through route 10-9-7-2-1	1-2-7-9-10
10	20	9	8	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-2-7-9-8
10	21	9	5	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-2-7-9-5
12	22	9	10	RREQ	1	4	New RREQ. Respond through route 10-9-5-3-1	1-3-5-9-10
12	23	9	8	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-3-5-9-8
12	24	9	7	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-3-5-9-7
13	25	8	5	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-4-6-8-5
13	26	8	9	RREQ	1	4	Duplicate ID. Longer Path. Discard.	1-4-6-8-9
18	27	10	9	RREP	1	1	Record and forward along return path	10-9 (1-2-5-9-10)
19	28	10	9	RREP	1	1	Record and forward along return path	10-9 (1-2-7-9-10)
22	29	10	9	RREP	1	1	Record and forward along return path	10-9 (1-3-5-9-10)
27	30	9	5	RREP	1	2	Record and forward along return path	10-9-5 (1-2-5-9-10)
28	31	9	7	RREP	1	2	Record and forward along return path	10-9-7 (1-2-7-9-10)
29	32	9	5	RREP	1	2	Record and forward along return path	10-9-5 (1-3-5-9-10)
30	33	5	2	RREP	1	3	Record and forward along return path	10-9-5-2 (1-2-5-9-10)
31	34	7	2	RREP	1	3	Record and forward along return path	10-9-7-2 (1-2-7-9-10)
32	35	5	3	RREP	1	3	Record and forward along return path	10-9-5-3 (1-3-5-9-10)
33	36	2	1	RREP	1	4	Record and forward along return path	10-9-5-2-1 (1-2-5-9-10)
34	37	2	1	RREP	1	4	Record and forward along return path	10-9-7-2-1 (1-2-7-9-10)
35	38	3	1	RREP	1	4	Record and forward along return path	10-9-5-3-1 (1-3-5-9-10)

Route Maintenance in DSR

- If a transmission fails, route error (RERR) is sent to the source. It contains hosts at both ends of the link.
- Intermediate nodes remove or truncate all routes with that link.
- Source may reinitiate the route discovery.
- Caching multiple routes results in a faster recovery but the routes may be stale resulting in cache poisoning at other nodes.
- Not suitable for high-mobility environments.
- Source-route overhead in each packet.

AODV vs. DSR

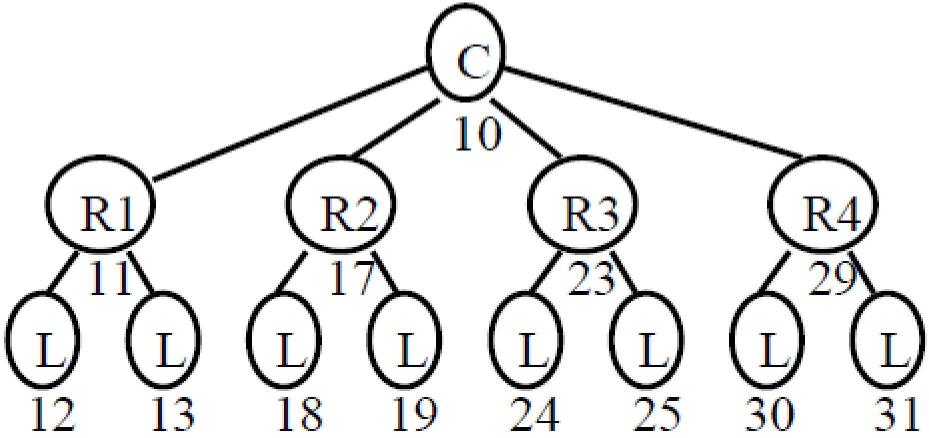
- In DSR a single RREQ can result in routes to several destination
- In DSR RERR messages are sent to the source not broadcast → Many nodes are unaware of failure
- In DSR, route discovery is delayed until all cached entries have been tried → Not good for high mobility



Feature	DSR	AODV
Routing Table	Route	Next Hop
Packet	Route	No route
Replies	Multiple	First only
Route Deletion	Fast Local	Slow Global

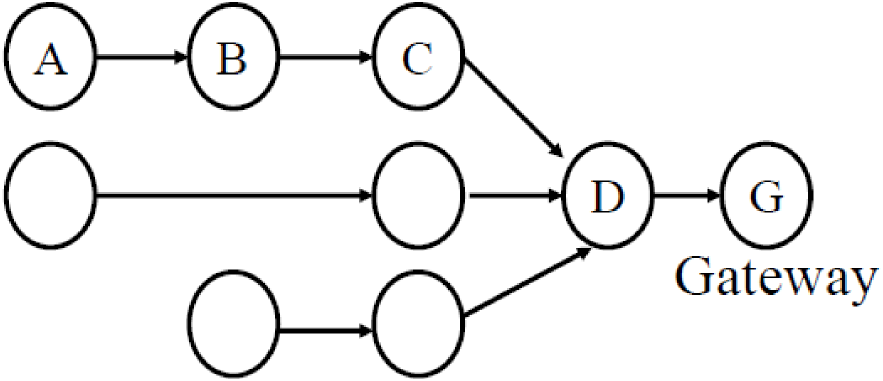
Tree Hierarchical Routing

- All leaf nodes send the packet to their parent
- Each parent checks the address to see if it is in its subrange.
  - If yes, it sends to the appropriate child.
  - If not, it sends to its parent
- Example: A12 to A30. A12 → R1 → Coordinator → R4 → A30



Many-to-One Routing

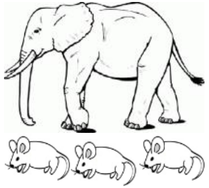
- Used for sensor data collection. All data goes to a concentrator or a gateway
- Gateway has a large memory and can hold complete routes to all nodes
- But each node only remembers the next hop towards gateway



6LowPAN

6LowPAN

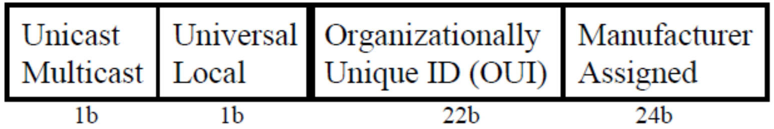
- IPv6 over Low Power Wireless Personal Area Networks
- How to transmit IPv6 datagrams (elephants) over low power IoT devices (mice)?



- Issues:
  - IPv6 address formation: 128-bit IPv6 from 64-bit EUI64
  - Maximum Transmission Unit (MTU): IPv6 at least 1280 bytes vs. IEEE 802.15.4 standard packet size is 127 bytes
  - Address Resolution: 128b or 16B IPv6 addresses. 802.15.4 devices use 64 bit (no network prefix) or 16 bit addresses
  - Optional mesh routing in datalink layer → Need destination and intermediate addresses.

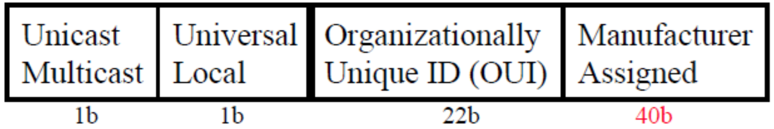
EUI64 Addresses

- Ethernet addresses: 48 bit MAC



Unicast Multicast	Universal Local	Organizationally Unique ID (OUI)	Manufacturer Assigned
1b	1b	22b	24b

- IEEE 802.15.4 Addresses: 64 bit Extended Unique Id (EUI)

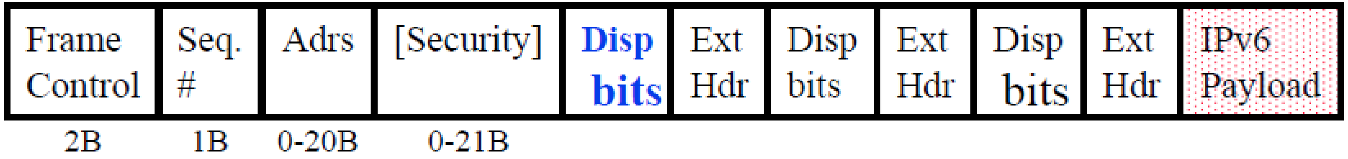


Unicast Multicast	Universal Local	Organizationally Unique ID (OUI)	Manufacturer Assigned
1b	1b	22b	40b

- Local bit was incorrectly assigned. L=1 Local, but all-broadcast address = all 1's is not local.
- IETF RFC4291 changed the meaning so that L=0 Local
- The 2nd bit is now called Universal bit (U-bit) → U-bit formatted EUI64 addresses

6LowPAN Adaptation Layer

- MAC-level retransmissions versus end-to-end:
  - Optional hop-by-hop ack feature of 802.15.4 is used but the max number of retransmissions is kept low (to avoid overlapping L2 and L4 retransmissions)
- Extension Headers: 8b or less Shannon-coded dispatch
- header type
  - 10: Mesh addressing header (2-bit dispatch)
  - 11x00: Destination Processing Fragment header (5-bit)
  - 01010000: Hop-by-hop LowPAN Broadcast header (8-bit)
- IPv6 and UDP header compression



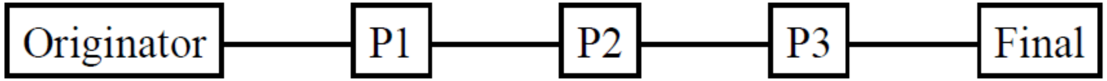
Frame Control	Seq. #	Adrs	[Security]	**Disp bits**	Ext Hdr	Disp bits	Ext Hdr	Disp bits	Ext Hdr	IPv6 Payload
2B	1B	0-20B	0-21B							

IPv6 Address Formation

- Link-Local IPv6 address = FE80::U-bit formatted EUI64
- Example:
  - EUI64 Local Address = 40::1 = 0100 0000::0000 0001
  - U-bit formatted EUI64 = 0::1
  - IPv6 Link-local address = FE80::1 = 1111 1110 1000 0000::1
- IEEE 802.15.4 allows nodes to have 16-bit short addresses and each PAN has a 16-bit PAN ID.
  - 1st bit of Short address and PAN ID is Unicast/Multicast
  - 2nd bit of Short Address and PAN ID is Local/Universal.
  - You can broadcast to all members of a PAN or to all PANs.
- IPv6 Link Local Address = FE80 :: PAN ID : Short Address
  - Use 0 if PAN ID is unknown.
  - 2nd bit of PAN ID should always be zero since it is always local. (2nd most significant = 6th bit from right)

Mesh Addressing Header

- Dispatch = 102 (2 bits) Mesh Addressing Header
- MAC header contains per-hop source and destination
- Original source and destination addresses are saved in Mesh addressing header
- A 4-bit hops-left field is decremented at each hop

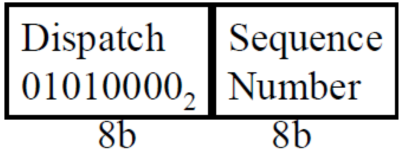

Originator
P1
P2
P3
Final

Dispatch 10	V	F	Hops Left	Originator Address	Final Address
2b	1b	1b	4b	16b/64b	16b/64bit

- V=0 → Originator address is EUI64, V=1 → 16bit
- F=0 → Final address is EUI64, F=1 → 16-bit

6LowPAN Broadcast Header

- For Mesh broadcast/multicast
- A new sequence number is put in every broadcast message by the originator



Dispatch 01010000_2	Sequence Number
8b	8b

6LowPAN Fragment Header

- ❖ Dispatch = 11x00 (5 bits) → Fragment Header
- ❖ Full packet size in the first fragment's fragment header
- ❖ Datagram tag = sequence number
  - ➢ Fragments of the same packet
- ❖ Fragment Offset in multiples of 8 bytes

1st Fragment

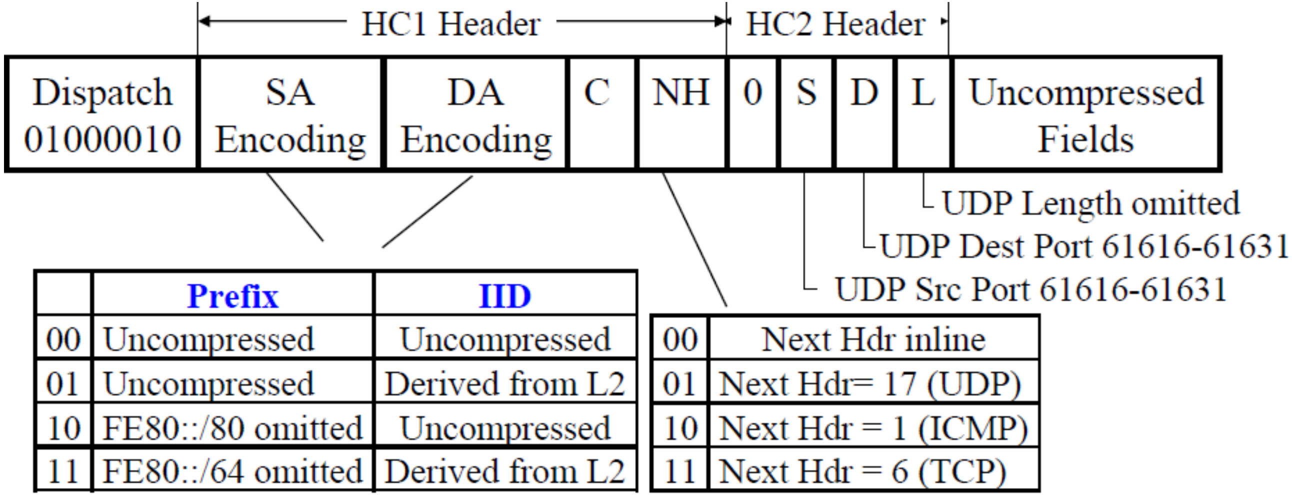
11000	IP Pkt Size	Datagram tag	Payload
5b	11b	16b	

Other Fragments

11100	IP Pkt Size	Datagram tag	Datagram Offset	Payload
5b	11b	16b	8b	

IP+UDP Header Compression: Stateless

- Called HC1-HC2 compression (not recommended)
- IP version field is omitted
- Flow label field is omitted if zero and C=1
- Only 4b UDP ports are sent if between 61616-61631 (F0Bx)
- UDP length field is omitted. IP addresses are compressed.



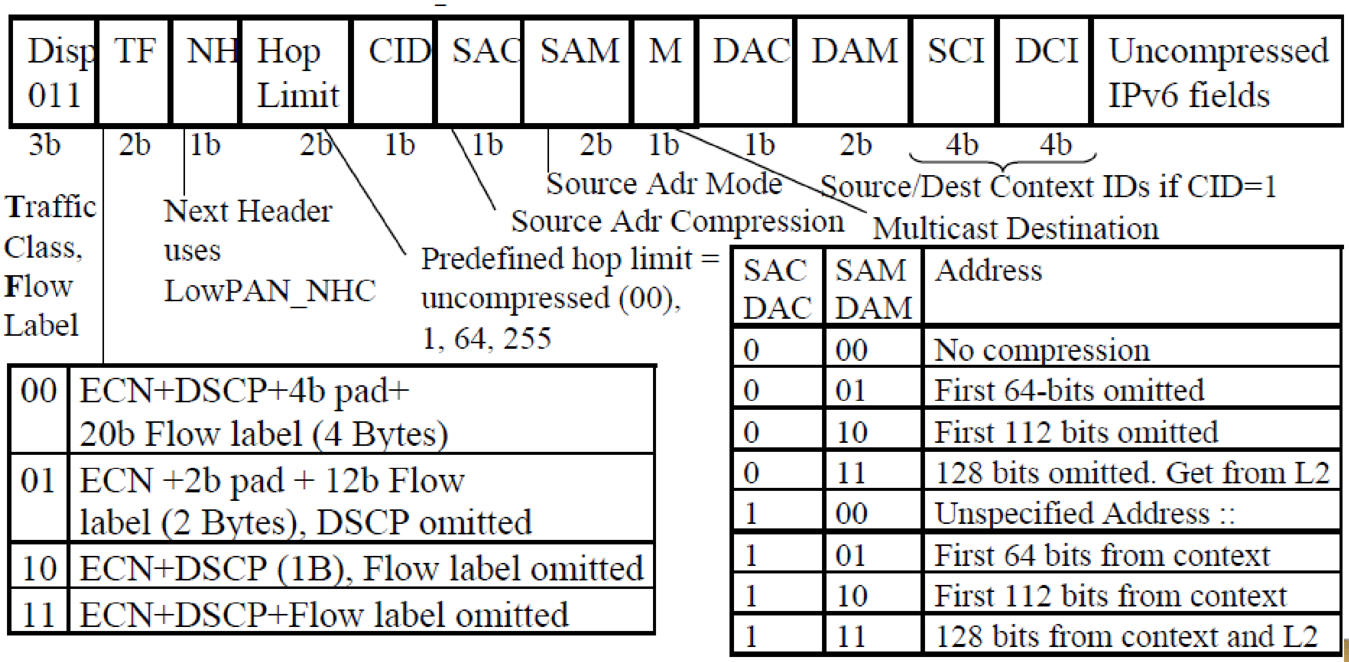
Dispatch 01000010	SA Encoding	DA Encoding	C	NH	0	S	D	L	Uncompressed Fields

	Prefix	IID
00	Uncompressed	Uncompressed
01	Uncompressed	Derived from L2
10	FE80::/80 omitted	Uncompressed
11	FE80::/64 omitted	Derived from L2

00	Next Hdr inline
01	Next Hdr= 17 (UDP)
10	Next Hdr = 1 (ICMP)
11	Next Hdr = 6 (TCP)

Context Based Compression

- HC1 works only with link-local addresses
- Need globally routable IPv6 addresses for outside nodes
- IPHC uses a 3b dispatch code and a 13-bit base header



Disp 011	TF	NH	Hop Limit	CID	SAC	SAM	M	DAC	DAM	SCI	DCI	Uncompressed IPv6 fields
3b	2b	1b	2b	1b	1b	2b	1b	1b	2b	4b	4b	

Traffic Class, Flow Label

Next Header uses LowPAN_NHC

Predefined hop limit = uncompressed (00), 1, 64, 255

Source Adr Mode

Source Adr Compression

Multicast Destination

Source/Dest Context IDs if CID=1

00	ECN+DSCP+4b pad+ 20b Flow label (4 Bytes)
01	ECN +2b pad + 12b Flow label (2 Bytes), DSCP omitted
10	ECN+DSCP (1B), Flow label omitted
11	ECN+DSCP+Flow label omitted

SAC DAC	SAM DAM	Address
0	00	No compression
0	01	First 64-bits omitted
0	10	First 112 bits omitted
0	11	128 bits omitted. Get from L2
1	00	Unspecified Address ::
1	01	First 64 bits from context
1	10	First 112 bits from context
1	11	128 bits from context and L2

Context Based Compression

- If the next header uses LowPAN_NHC
- For IPv6 base extension headers:

1110	IPv6 Ext Hdr ID (EID)	NH	Uncompressed Fields	Next Hdr
4b	3b	1b		

EID	Header
0	IPv6 Hop-by-Hop Options
1	IPv6 Routing
2	IPv6 Fragment
3	IPv6 Destination Options
4	IPv6 Mobility Header
5	Reserved
6	Reserved
7	IPv6 Header

0 = Uncompressed

1 = LowPAN_NHC encoded

LowPAN_NHC UDP Header:

11110	C	P
5b	1b	2b

Checksum omitted

00	All 16-bits in line
01	1st 8-bits of dest port omitted
10	1st 8-bits of src port omitted
11	1st 12-bits of src & dest omitted