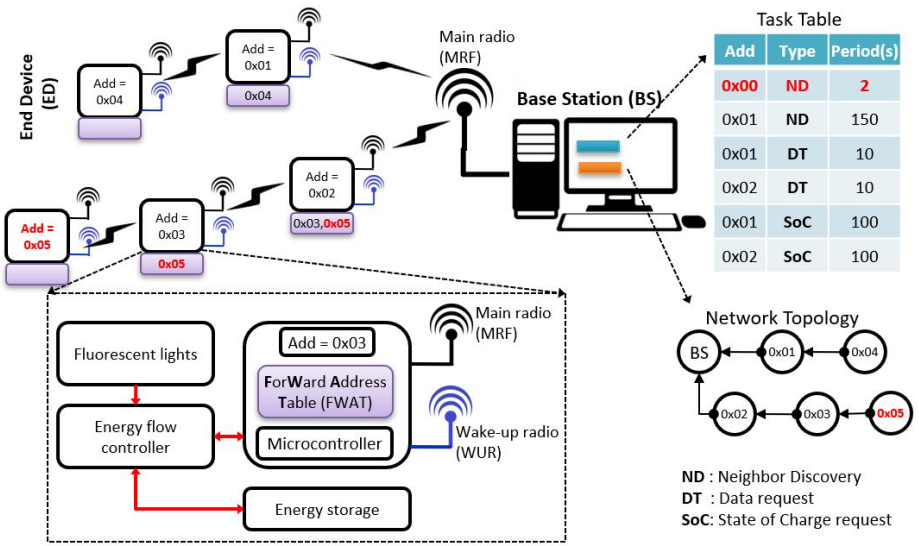
Wireless Communication in Internet of Things (IoT)

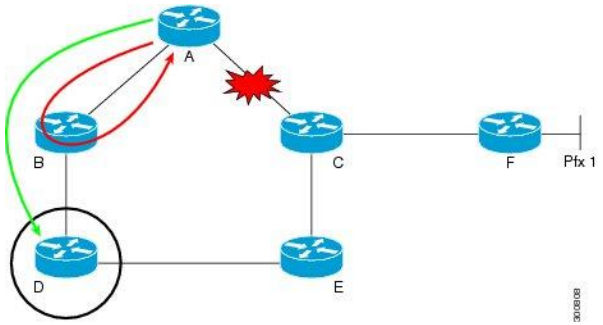


Task Table

Add	Type	Period(s)
0x00	ND	2
0x01	ND	150
0x01	DT	10
0x02	DT	10
0x01	SoC	100
0x02	SoC	100

Network Topology

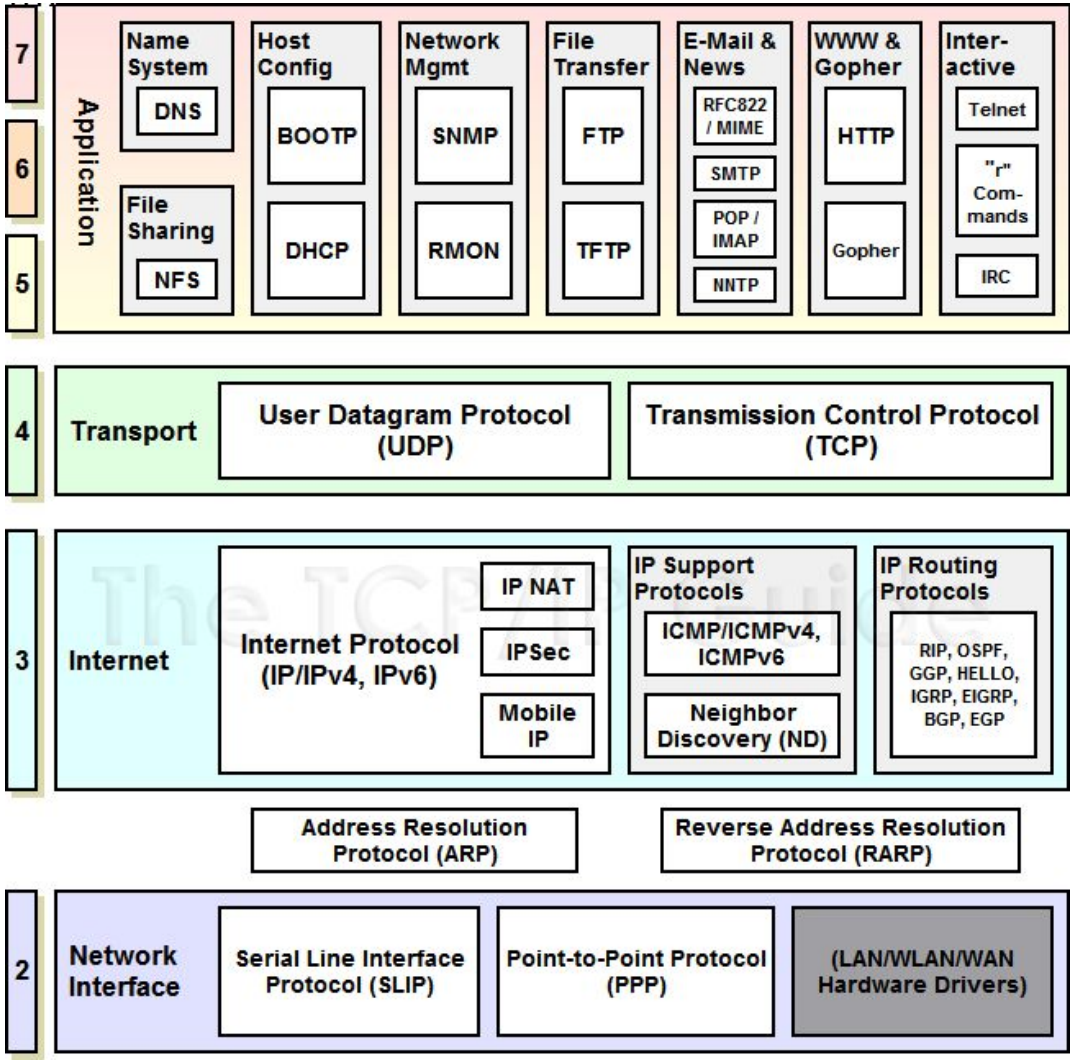
ND : Neighbor Discovery
DT : Data request
SoC: State of Charge request



DCE
DEPT. OF COMPUTER ENGINEERING

IoTs based TCP/IP Architecture

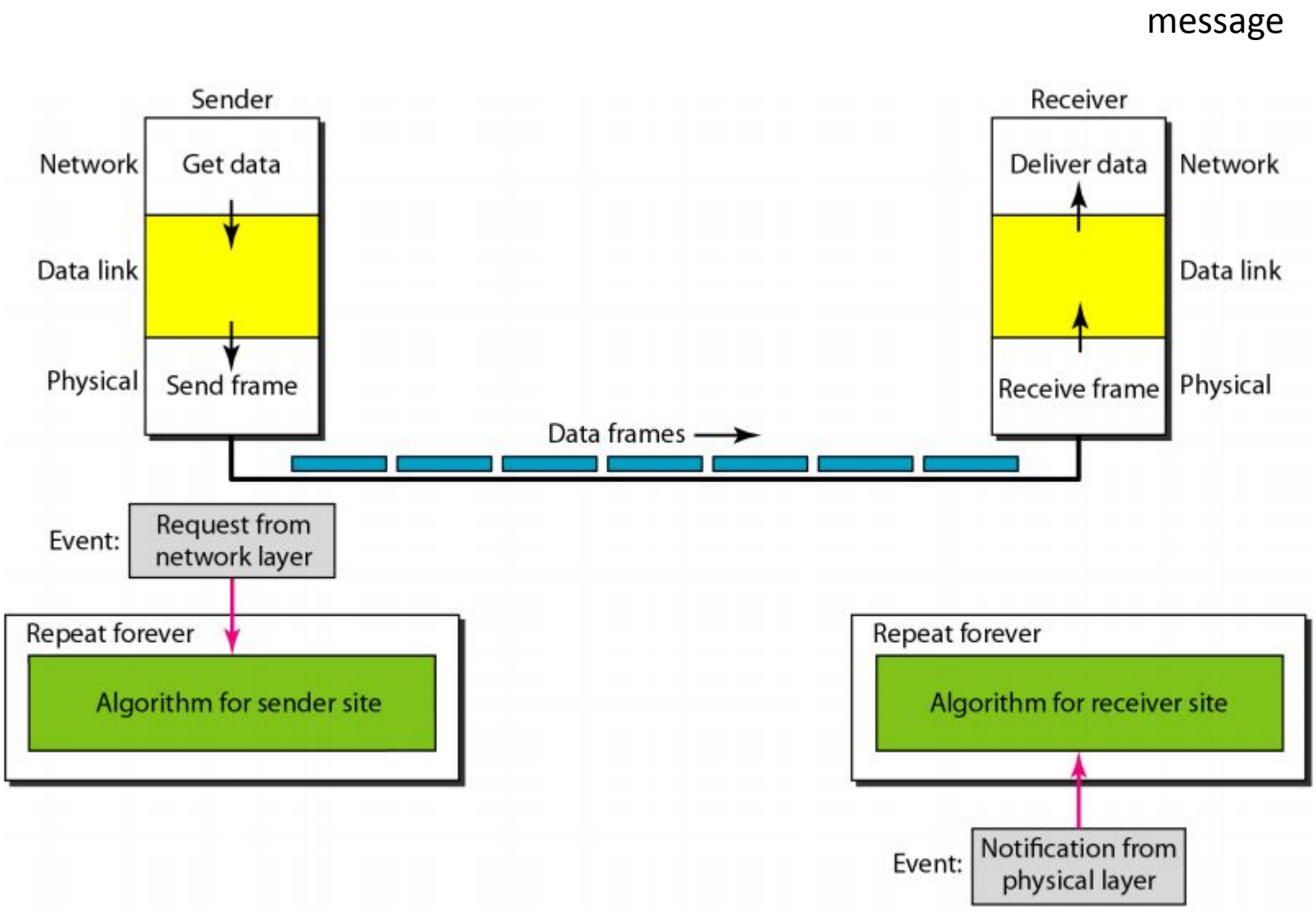
- Application layer
- Transport layer
- Internet layer
- Network access layer
- Physical layer



Medium Access Control (MAC) Protocol

- **Protocol** is the combination of **framing**, **flow control**, and **error control** to achieve the delivery of data from one node to another
- The protocols are normally **implemented in software** by using one of the common programming languages
- Flow control: Refers to a set of procedures used to **restrict the amount of data that the sender can send** before waiting for acknowledgment
- Error control: is both **error detection and error correction**

Simplest Protocol Design

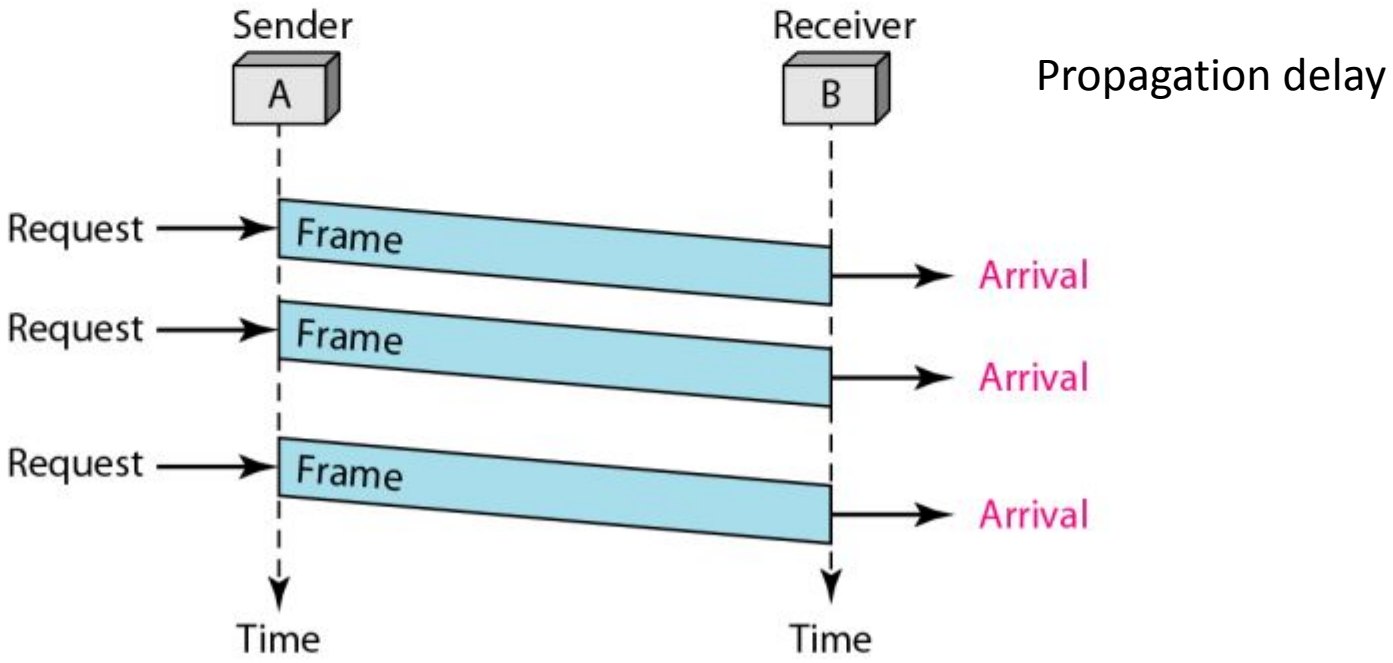


Implementation

```
while(true){
    WaitForEvent();
    if(Event(RequestToSend)){
        GetData();
        MakeFrame();
        SendFrame();
    }
}
```

```
while(true){
    WaitForEvent();
    if(Event(ArrivalNotification)){
        ReceiveFrame();
        ExtractData();
        DeliverData();
    }
}
```

Example



- The sender **sends a sequence of frames** without even thinking about the receiver
- There is no error handler
- There is no synchronization (the receiver processing time is slower than the transmission speed)

MAC Protocol in IoTs

- Synchronous protocols
  - The stream of data to be transferred is encoded as fluctuating voltage levels in one wire (the 'DATA'), and a periodic pulse of voltage on a separate wire (called the **"CLOCK"**) which tells the receiver that the current DATA bit is **available** at this moment in time

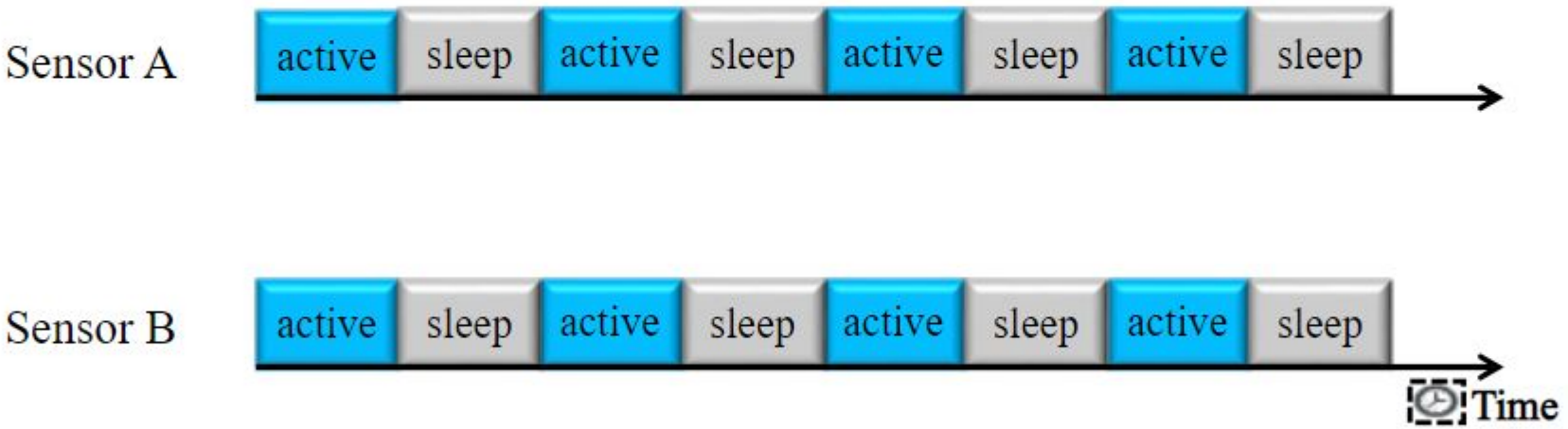
- Asynchronous protocols
  - Data is transmitted at a **random time.** Normally, a start and stop conditions are used to begin a **"rendez-vous"** to initiate a comunication

Advantage and Disadvantage

	Advantage	Disadvantage
Asynchronous transmission	• Simple, doesn't require synchronization of both communication sides • Cheap, asynchronous requires less hardware • Suitable for low data rate applications	Large relative overhead, a high proportion of the transmitted bits are uniquely for control purposes and thus carry no useful information
Synchronous transmission	Lower overhead and thus, greater throughput	• Slightly more complex • Hardware is more expensive

Synchronous Protocols: S-MAC

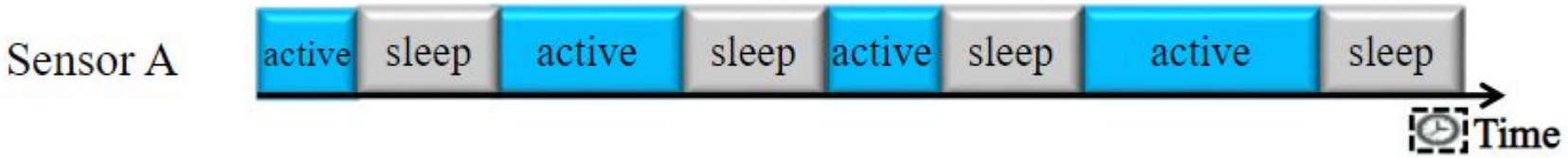
- Communication are synchronized in time with two state:
  - Active: Carrier sensing, Request To Send, Clear To Send and **Sync Packet**
  - Sleep: Low power mode for energy reservation



- Drawback: Energy wasted for active period:
  - When there is no packet to send?????

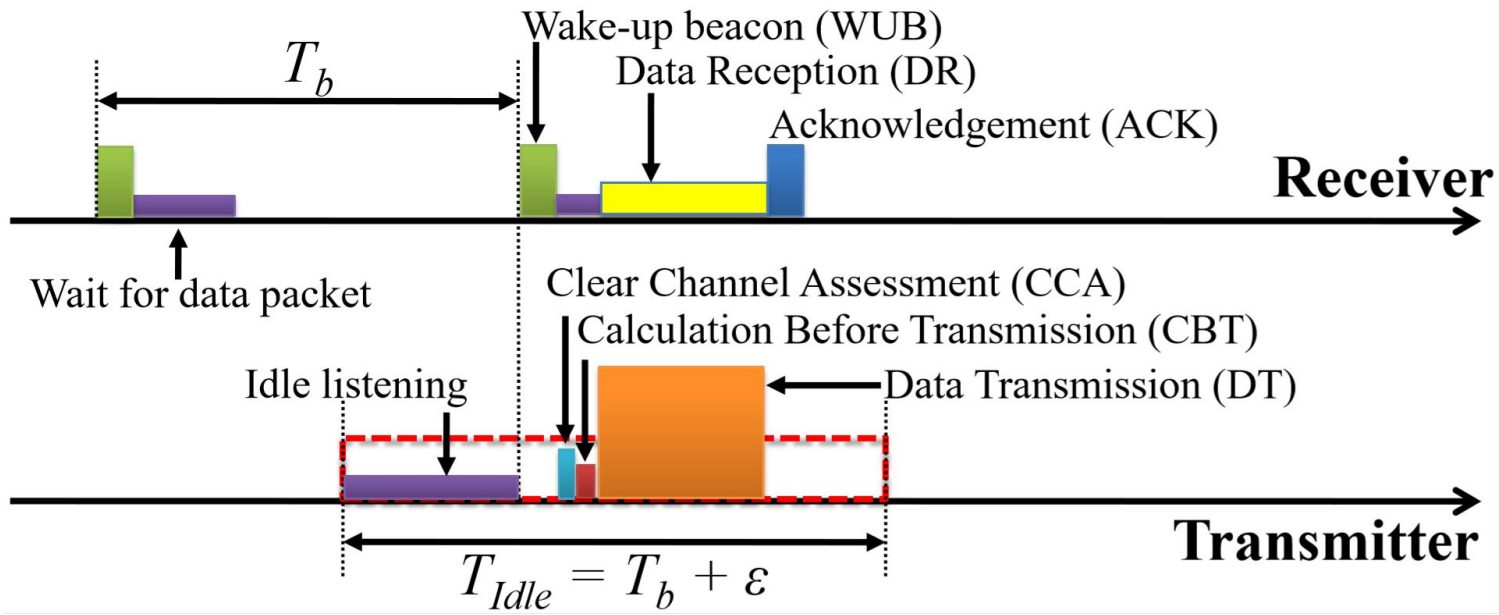
Synchronous Protocols: T-MAC

- An extended version from S-MAC:
  - Active period is adapted: if there is no RTS or CTS after a period, the node go to sleep mode



- Drawback: Energy wasted due to SYNC packet
  - High impact on low data rate networks

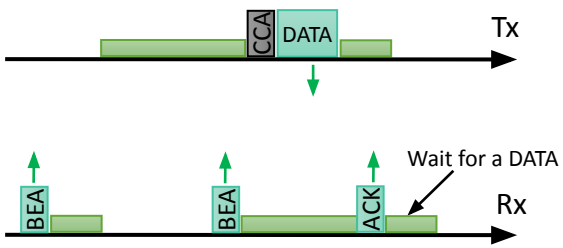
Asynchronous Protocols

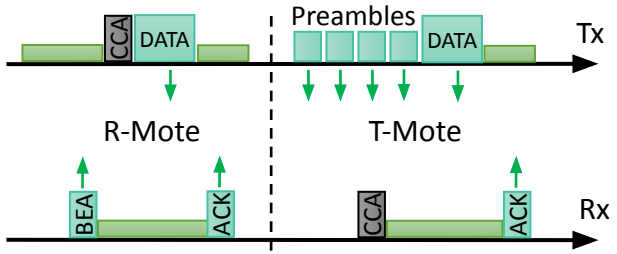


- **RICER (Receiver Initiated Cycled Receiver):**
  - Receiver sends a BEACON
  - Transmitter waits for a BEACON, before sending its DATA
  - ACK is used to confirm a communication

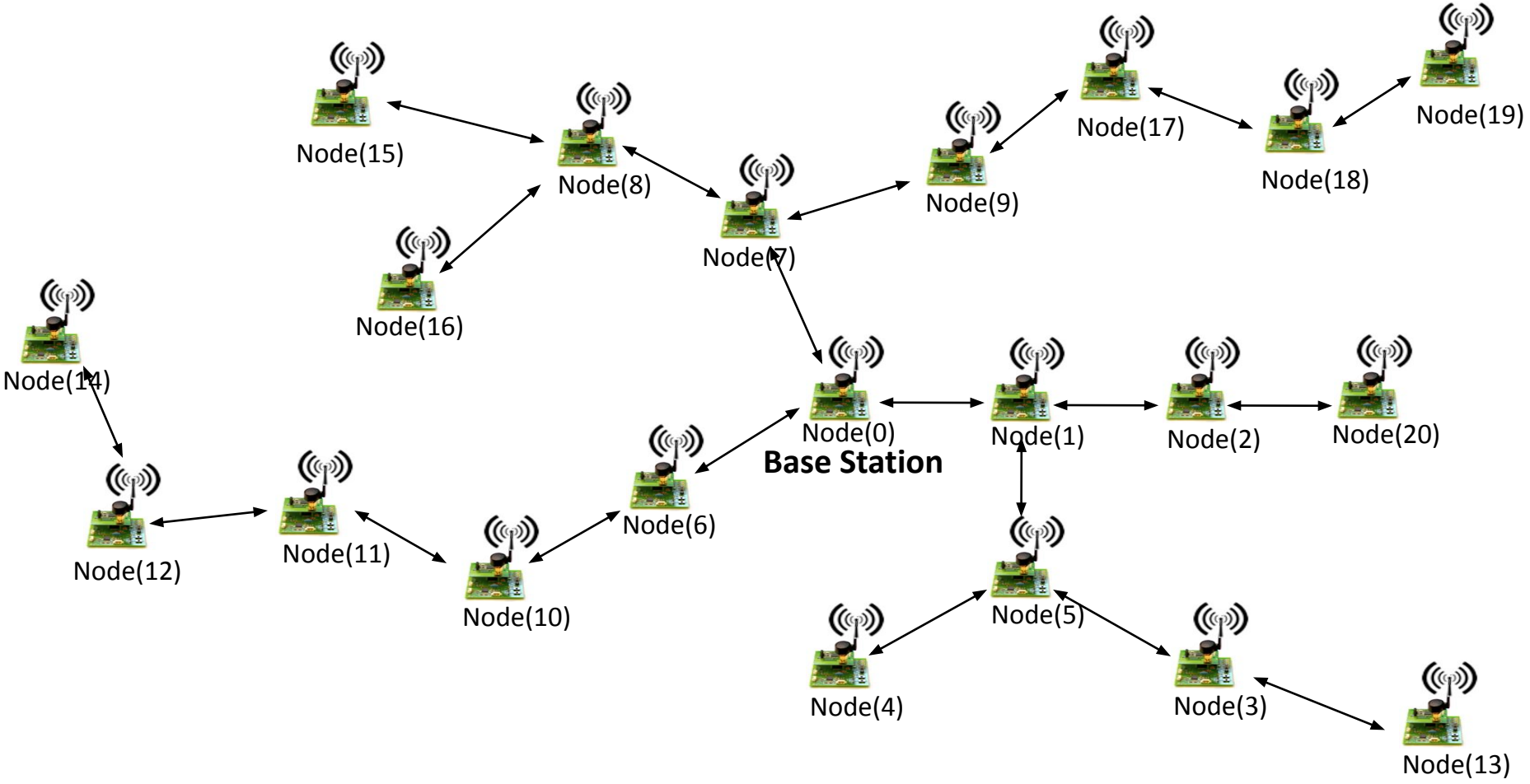
Asynchronous Protocols

- Extended version of RICER:
  - RICER3, RICER3b, RICE5
  - ODMAC: ACK plays the role of a new BEACON
  - SymMAC: RICER + TICER



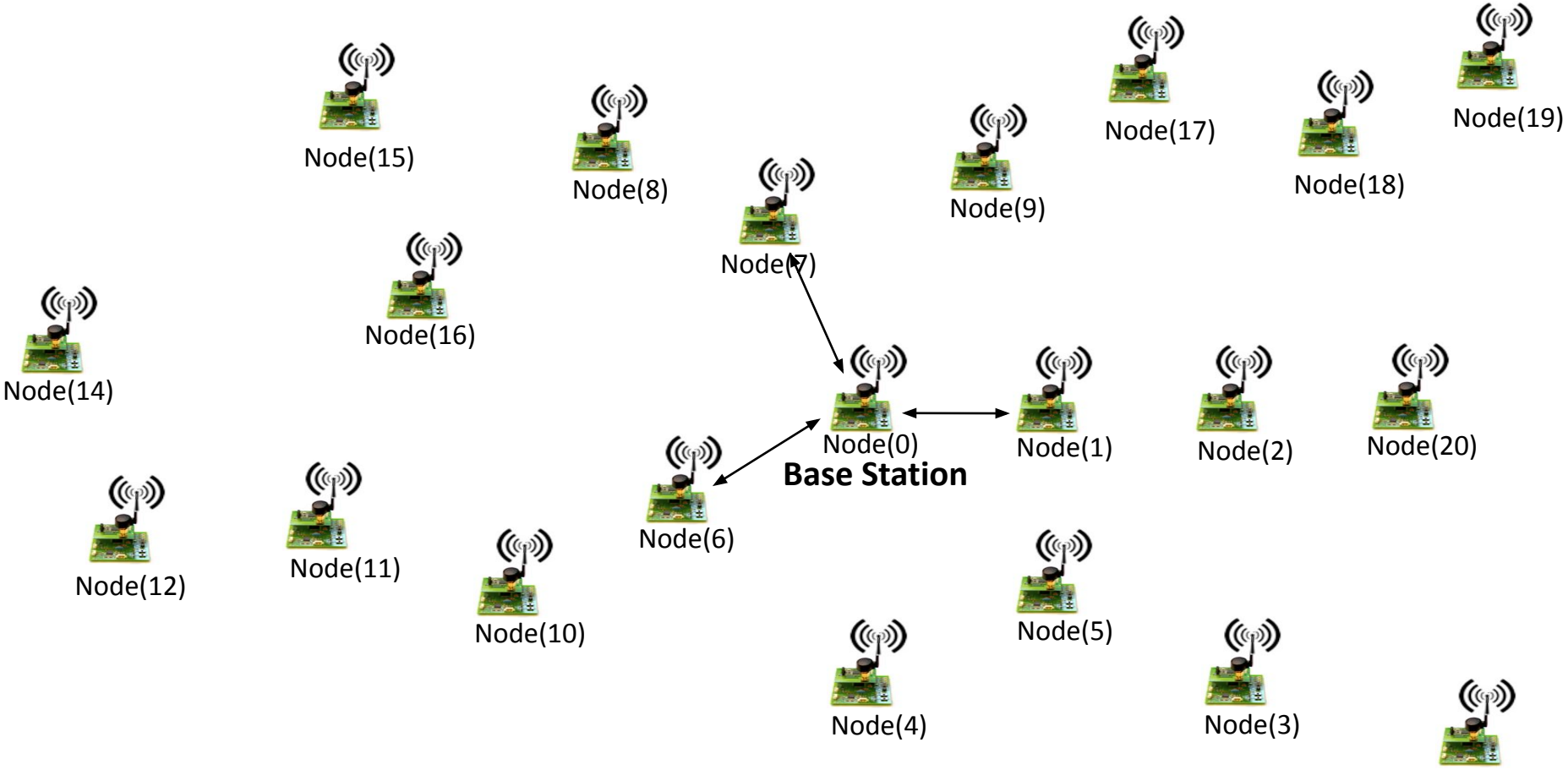


Routing Protocol in IoTs



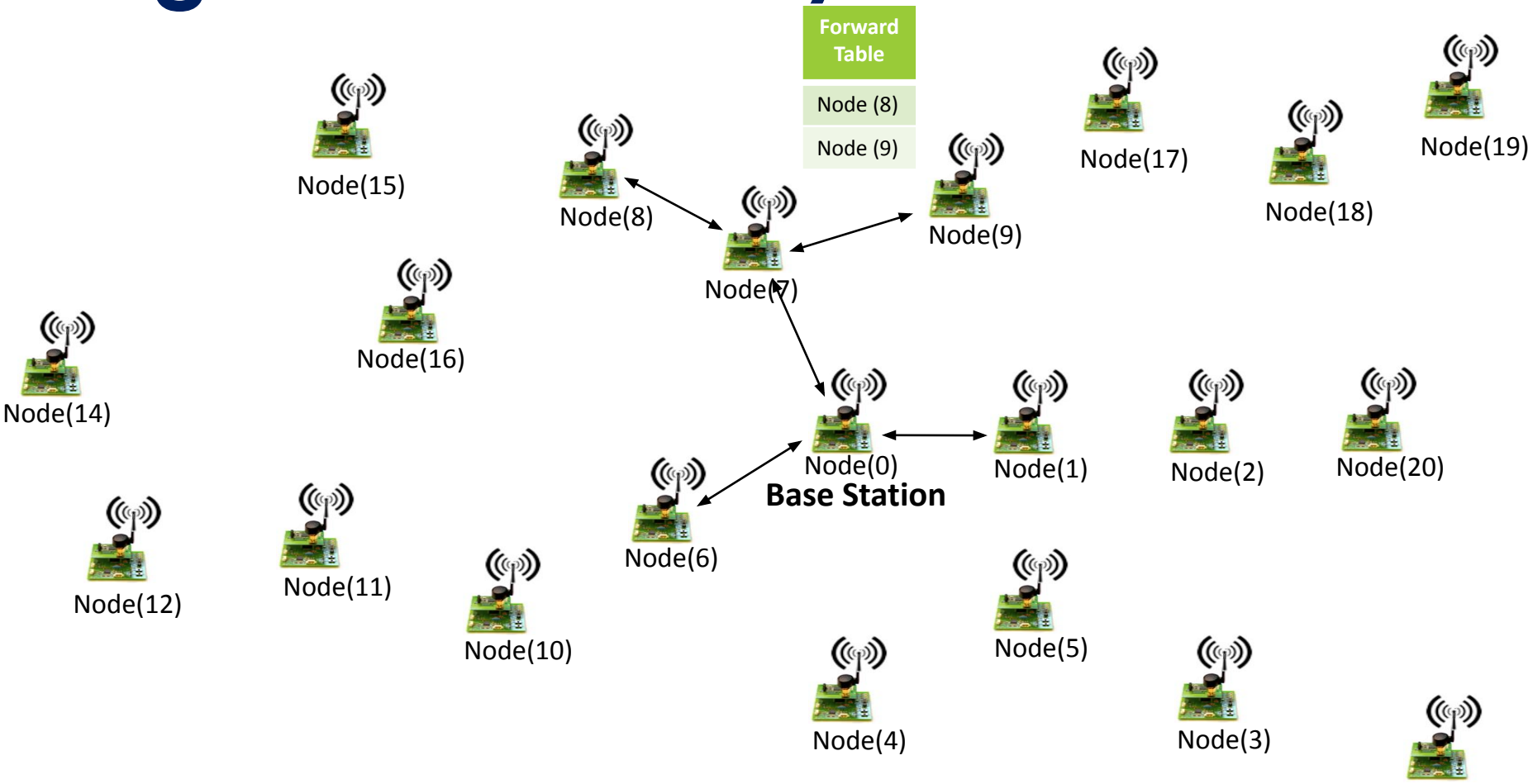
- **Energy efficient routing** for sending a packet???

Neighbor Discovery Process



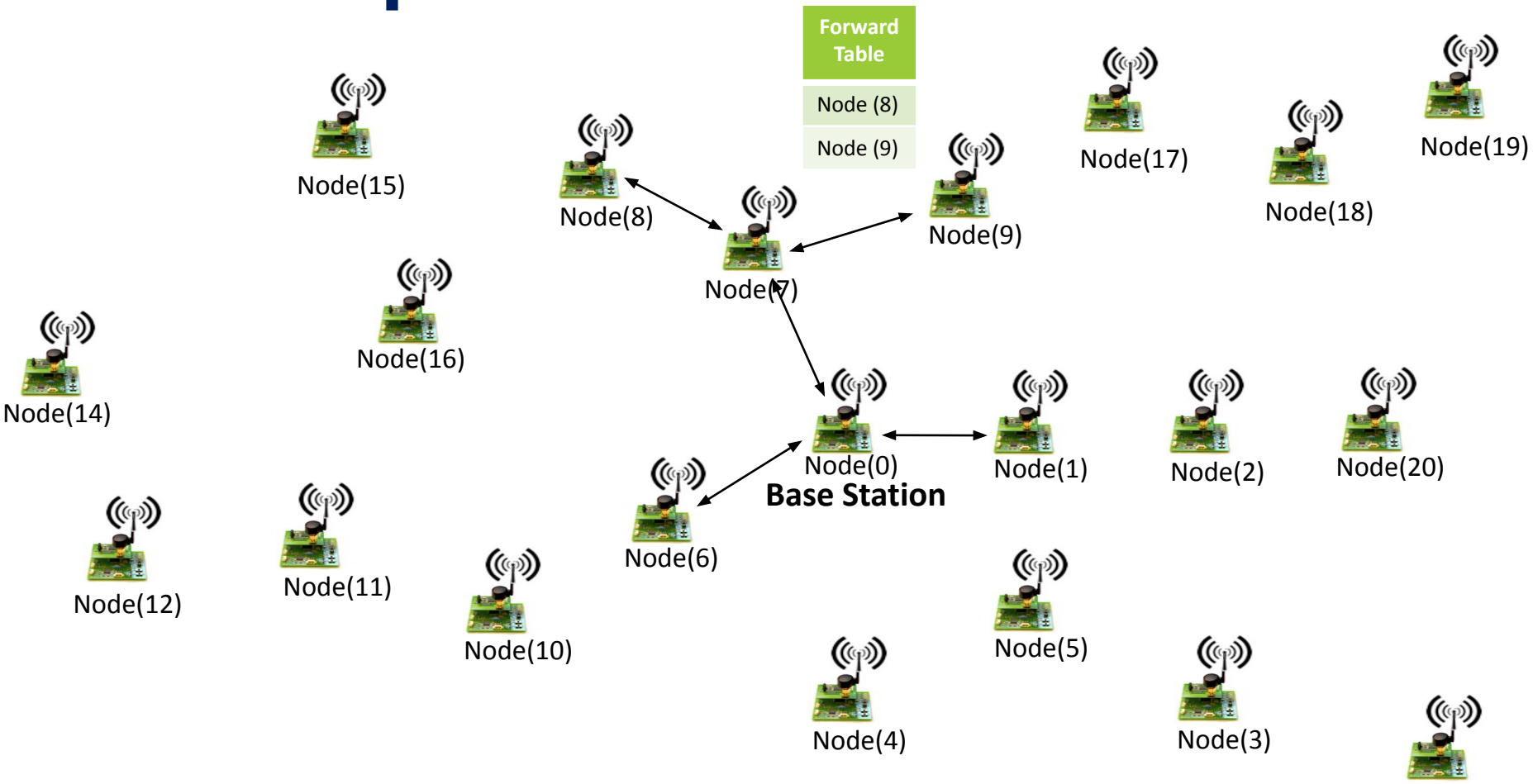
- Base station nodes send neighbor discovery packet:
  - **Node (1) (6) (7)** are discovered

Neighbor Discovery Process



- Node (7) sends neighbor discovery packet:
  - Node (8) and (9) response
  - (8) and (9) are added to the forward table of node (7)

Data Request Process



- Node (0) send data request packet to Node 8.
- Node (7) forwards this packet
- Node (8) response to Node (7) and then, forward to Node (0)