Ad-hoc On-demand Distance Vector Routing Protocol

AODV

AODV

- The Ad hoc On-Demand Distance Vector (AODV) routing protocol is intended for use by **mobile nodes** in an ad hoc network.
- AODV allows mobile nodes to obtain routes quickly for **new destinations**, and does not require nodes to maintain routes to destinations that are **not in active communication**

Message types defined

- Route Requests (RREQs)
- Route Replies (RREPs)
- and Route Errors (RERRs)

RREQ

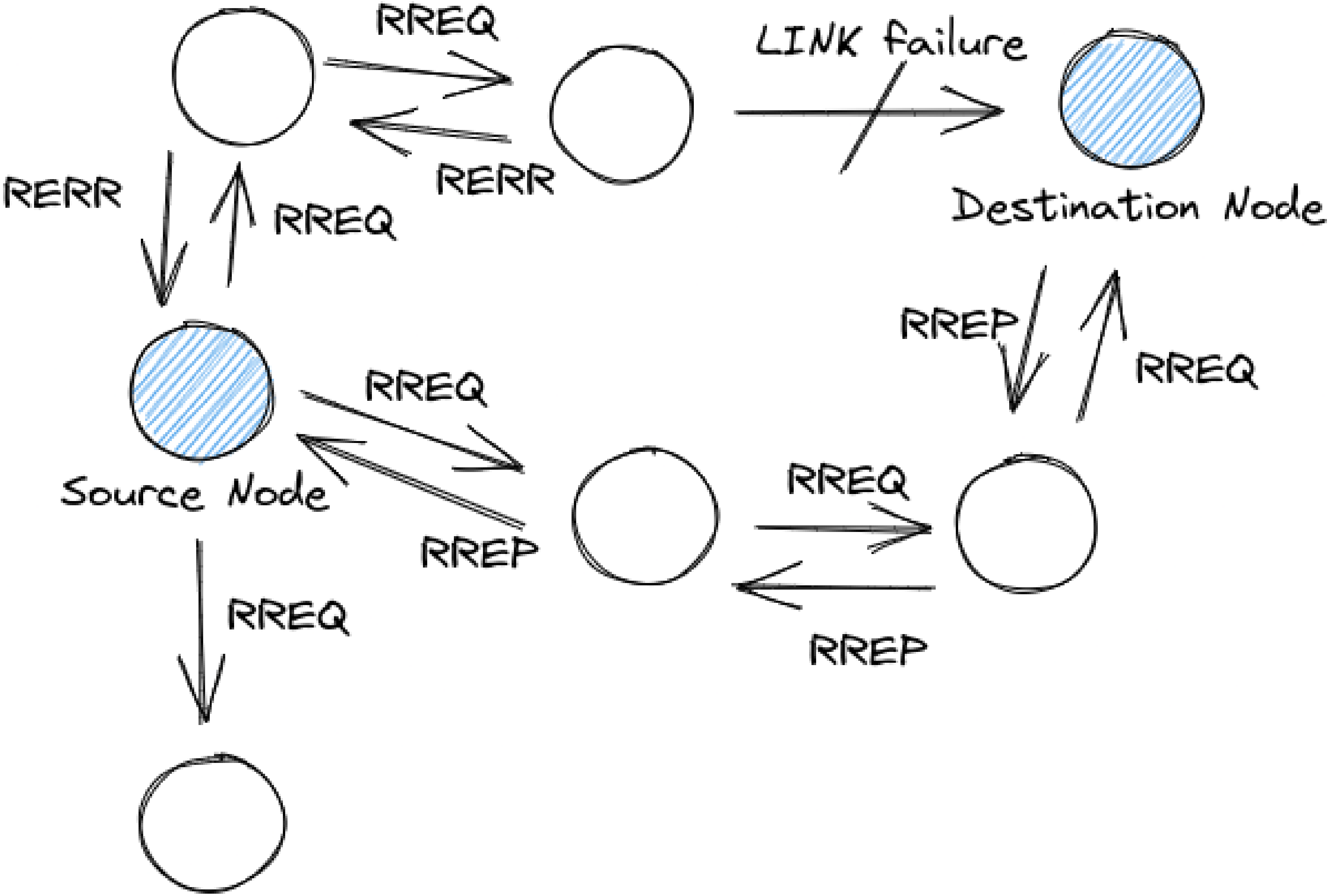
- When a **route to a new destination is nee**ded, the node broadcasts a RREQ to find a route to the destination.

RREP

- A route can be determined when the RREQ reaches either the **destination itself**, or an **intermediate node** with a 'fresh enough' route to the destination.
- A **'fresh enough**' route is a valid route entry for the destination whose associated sequence number is at least as great as that contained in the RREQ.
- The route is made available by **unicasting a RREP** back to the origination of the RREQ.

RERR

- Nodes monitor the link status of next hops in active routes.
- When a **link break** in an active route is detected, a RERR message is used to notify other nodes that the loss of that link has occurred.


RREQ
LINK failure
RERR
RERR
RREQ
Destination Node
RREP
RREQ
RREQ
Source Node
RREQ
RREP
RREQ
RREP

AODV uses the following fields with each route table entry:

- Destination IP Address
- Destination Sequence Number
- Network Interface
- Hop Count (number of hops needed to reach destination)
- Next Hop
- Lifetime (expiration or deletion time of the route)

and some flags and other fields for link failures.

RREQ

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |  FLAGS  |     Reserved        |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            RREQ ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Destination Sequence Number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Originator IP Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Originator Sequence Number                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Type 1
- **Destination only flag** indicates only the destination may respond to this RREQ
- **U Unknown** sequence number; indicates the destination sequence number is unknown
- **Hop Count** The number of hops from the Originator IP Address to the node handling the request.
- **Destination IP Address** The IP address of the destination for which a route is desired.
- **Destination Sequence Number** The latest sequence number received in the past by the originator for any route towards the destination.
- **Originator IP Address** The IP address of the node which originated the Route Request.
- **Originator Sequence Number** The current sequence number to be used in the route entry pointing towards the originator of the route request.

Generating Route Requests

- A node disseminates a RREQ when it determines that **it needs a route** to a destination and does not have one available.
- This can happen if the destination is previously **unknown to the node**, or if a previously valid route to the destination **expires or is marked as invalid**.
- **The Destination Sequence Number** field in the RREQ message is the last known destination sequence number for this destination and is copied from the Destination Sequence Number field in the **routing table.**
- If no sequence number is known, the unknown sequence number flag MUST be set.

Generating Route Requests

- **The Originator Sequence Number** in the RREQ message is the node's **own sequence number**, which is incremented prior to insertion in a RREQ.
- The RREQ ID field is **incremented by one from the last RREQ ID** used by the current node. Each node maintains only one RREQ ID.
- **The Hop Count field is set to zero**

Processing and Forwarding Route Requests

- When a node receives a RREQ, it first **creates** or **updates** a route to the **previous hop** without a valid sequence number.
- Checks to determine whether it has received a RREQ with the same Originator IP Address and RREQ ID.
- If such a RREQ has been received, the node silently discards the newly received RREQ.

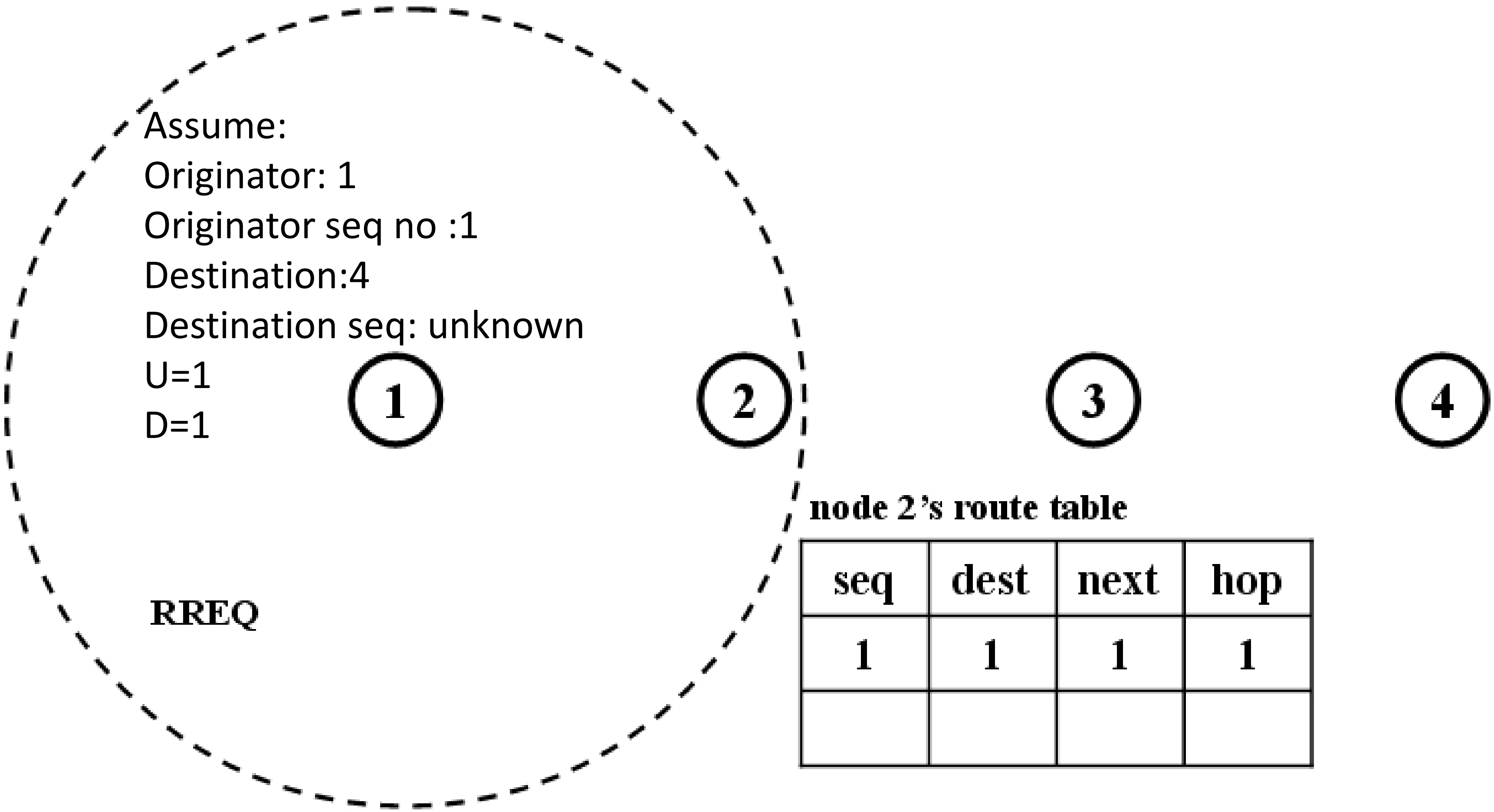
Actions taken for RREQs that are not discarded:

- The hop count value in the **RREQ is incremented by one**. the hop count is **copied to routing table** of receiving node from the Hop Count in the RREQ message;
- Then the **node searches for a reverse route** to the Originator IP Address.
- If need be, the route is **created, or updated** using the Originator Sequence Number from the RREQ in its routing table.
- the Originator **Sequence Number** from the RREQ is compared to the corresponding **destination sequence number** in the route table entry and copied if greater than the existing value there
- the **next hop in the routing table becomes the node from which the RREQ was received**
- Whenever a RREQ message is received, **the Lifetime of the reverse route entry** for the **Originator IP** address is updated.

Generating Route Replies

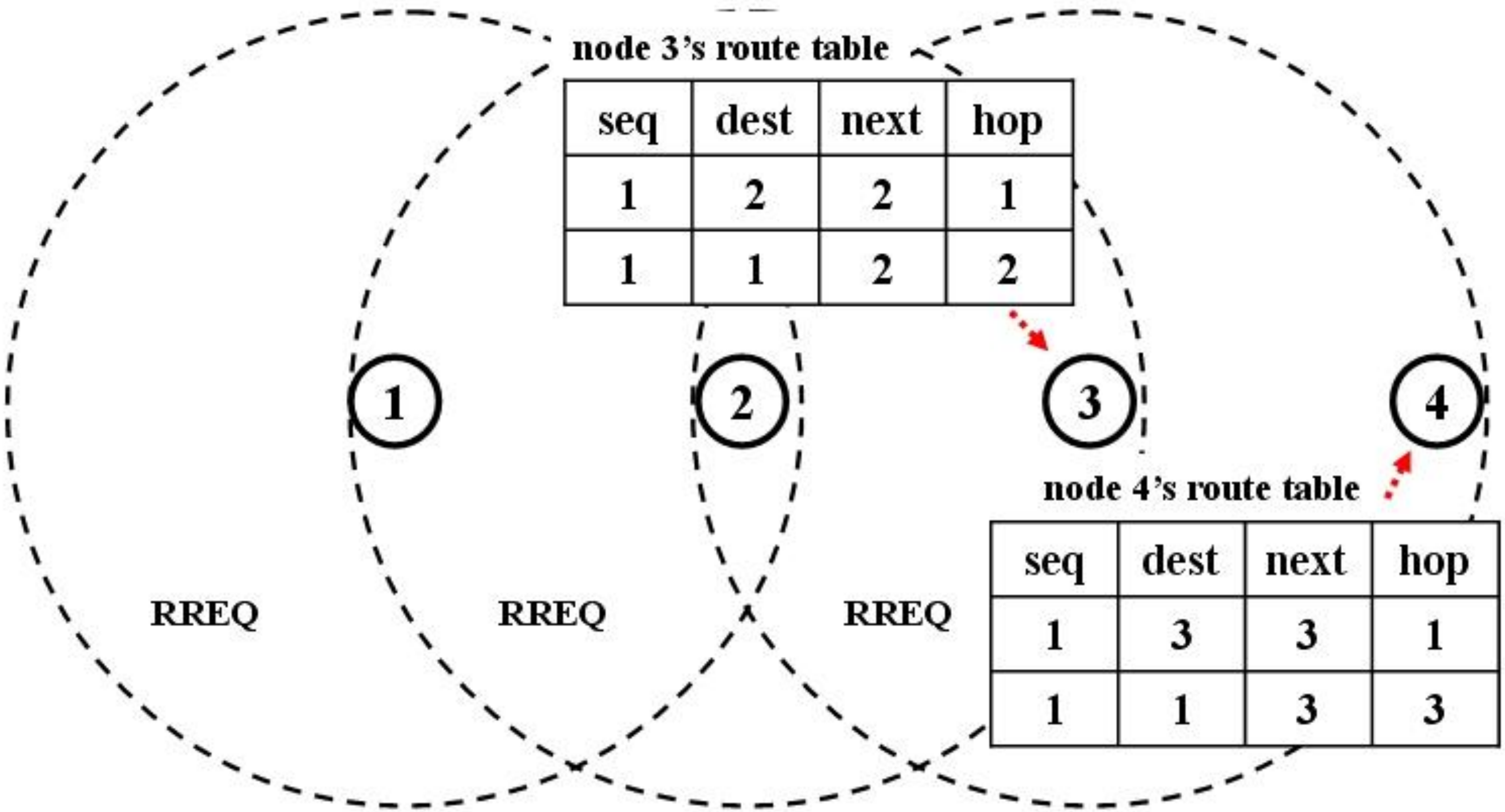
- A node generates a RREP if either:
- (i) it is itself **the destination**, or
- (ii) it has an active route to the destination, the destination sequence number in the node's existing route table entry for the destination is valid and **greater than or equal to the Destination Sequence Number** of the RREQ and the "destination only" ('D') flag is NOT set.

- The RREP is **unicast** to the next hop toward the originator of the RREQ, as indicated by the route table entry for that originator.
- As the **RREP is forwarded back towards the node which originated the RREQ message,** the Hop Count field is incremented by one at each hop.
- Thus, when **the RREP reaches the originator, the Hop Count represents the distance**, in hops, of the destination from the originator.



node 2's route table

seq	dest	next	hop
1	1	1	1



node 3's route table

seq	dest	next	hop
1	2	2	1
1	1	2	2

node 4's route table

seq	dest	next	hop
1	3	3	1
1	1	3	3

node 4's route table

seq	dest	next	hop
1	3	3	1
1	1	3	3

node 2's route table

seq	dest	next	hop
1	1	1	1
1	3	3	1
1	4	3	2

node 1's route table

seq	dest	next	hop
1	2	2	1
1	4	2	3

node 2's route table

seq	dest	next	hop
1	1	1	1
1	3	3	1
1	4	3	2

node 1's route table

seq	dest	next	hop
1	2	2	1
1	4	2	3

Distributed address mechanism of zigbee

- A device is said **to join a network successfully** if it can **obtain a network address** from the coordinator or a router.
- Before forming a network, the coordinator determines the **maximum number of children of a router** (Cm), the **maximum number of child routers of a router** (Rm), **and the depth of the network** (Lm).
- Note that a child of a router can be a router or an end device, so Cm >= Rm.
- ZigBee specifies a **distributed address assignment using** parameters Cm, Rm, and Lm to calculate nodes' network addresses.

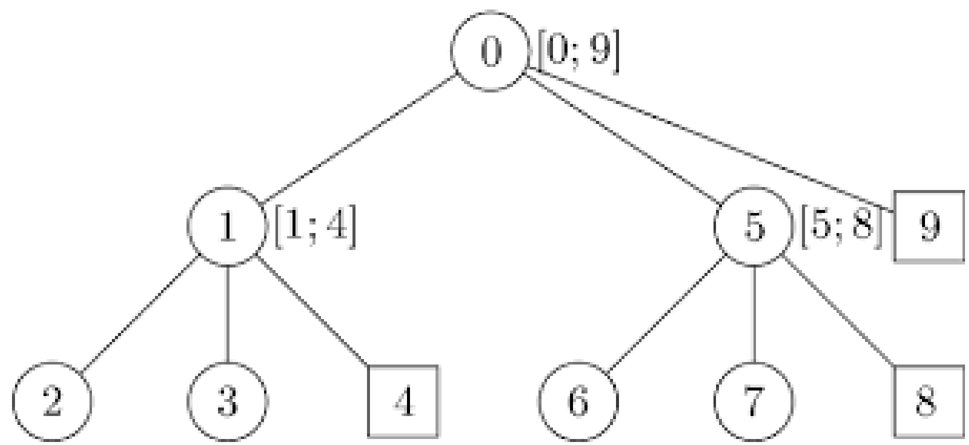
- The **allowed number of end devices** accepted by a router device is calculated as:
- MaxEndDevices = MaxChildren - MaxRouters = Cm - Rm
- In the sequence, when **a router device successfully joins** a network, its **parent device allocates a block of address** for its use, which means the joined router device becomes a potential parent device.
- Each joined router device can accept a certain number of children devices whose number cannot exceed Cm.
- The joining of the new router device is considered to extend the depth of the network, and the depth should not be greater than Lm
- If an end device successfully joins a network, it will be allocated a **network address by its parent devic**e.
- The joined **end devic**e does **not have** the capability to **accept new children devices**.

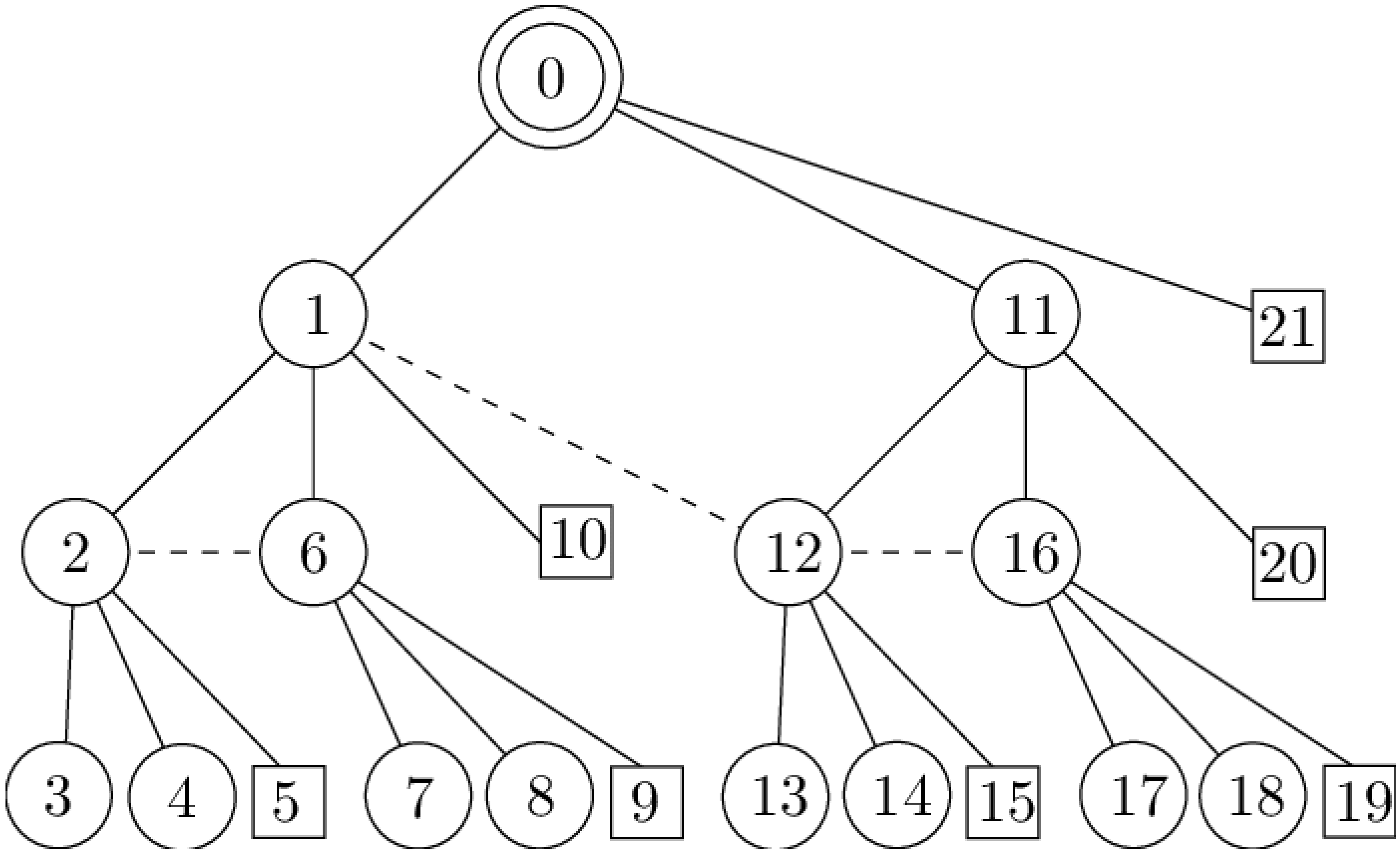
- Cskip is the method for calculating the total number of possible descendants that exist down any branch in the network. It is defined as follows:

$$Cskip(d) = \begin{cases} 1 + C_m(L_m - d - 1), & if \quad R_m = 1 \\ \frac{1 + Cm - Rm - Cm * Rm^{Lm-d-1}}{1 - Rm}, & otherwise \end{cases}$$

Then the distributed address assignment can be executed accordingly with $Cskip(d)$. For each parent device with address A_{parent} of depth d, the network addresses A_k for its kth router-capable child and A_n for its nth end device child are defined as follows.

$$\begin{aligned} A_k &= A_{parent} + 1 + Cskip(d) \cdot (k-1) \\ A_n &= A_{parent} + Cskip(d) \cdot Rm + n. \end{aligned} \tag{1}$$




0
1
11
21
2
6
10
12
16
20
3
4
5
7
8
9
13
14
15
17
18
19

$$\text{Cskip}(d) = \begin{cases} \dfrac{1 + Cm \times (Lm - d - 1)}{Lm - d - 1} & ; \; Rm = 1 \\[2ex] \dfrac{1 + Cm - Rm - Cm \times Rm}{1 - Rm} & ; \; o/w. \end{cases}$$

here in example,

$Cm = 3, \quad Rm = 2, \; Lm = 3.$

$$\text{Cskip}(0) = \frac{1 + 3 - 2 - 3 \times 2^{(3-0-1)}}{1 - 2}$$

$$= \frac{1+3-2-12}{-1} = 10$$

$$Cskip(1) = \frac{1+3-2-3\times 2^{(3-1-1)}}{1-2} = 4$$

$$Cskip(2) = \frac{1+3-2-3\times 2^{3-2-1}}{1-2} = 1$$

Address allocation :—

childrensLevel 0 :

$$R_1 = A_{parent} + 1 + Cskip(d).(k-1)$$

$$= 0+1+10.(1-1) = 1$$

$$R_2 = 0+1+10(2-1) = 11$$

~~R3 = 0+1+10(3-1) = 21~~

$$E_1 = A_{parent} + Cskip(d) \cdot Rm + n$$

$$= 0 + 10\times 2 + 1$$

$$= 21$$

children of R1 (level 1).

$R_{11} = 1 + 1 + 4 \times 0 = 2$

$R_{12} = 1 + 1 + 4 \times 1 = 6$

$E_{11} = 1 + \overset{6}{\underset{4}{\cancel{\ }}} \times 2 + 1 = 10$

chidren of R_2 (level 1).

$R_{21} = 11 + 1 + 0 = 12$

$R_{212} = 11 + 1 + 4 = \cancel{18}\ 16.$

$E_{211} = 11 + 8 + 1 = 20.$

children of R_{11}.

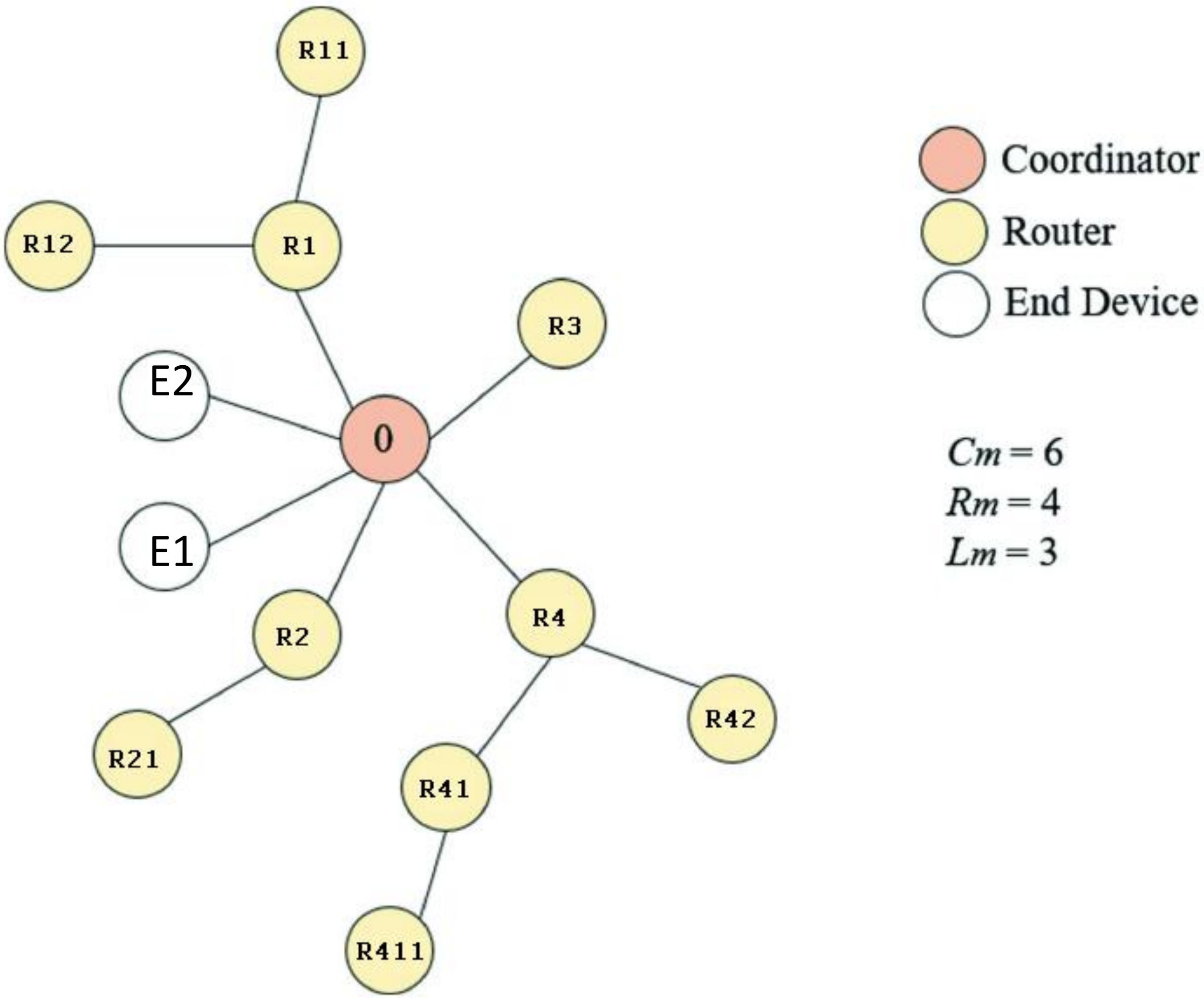
$R_{111} = 2 + 1 + 1(0) = 3$

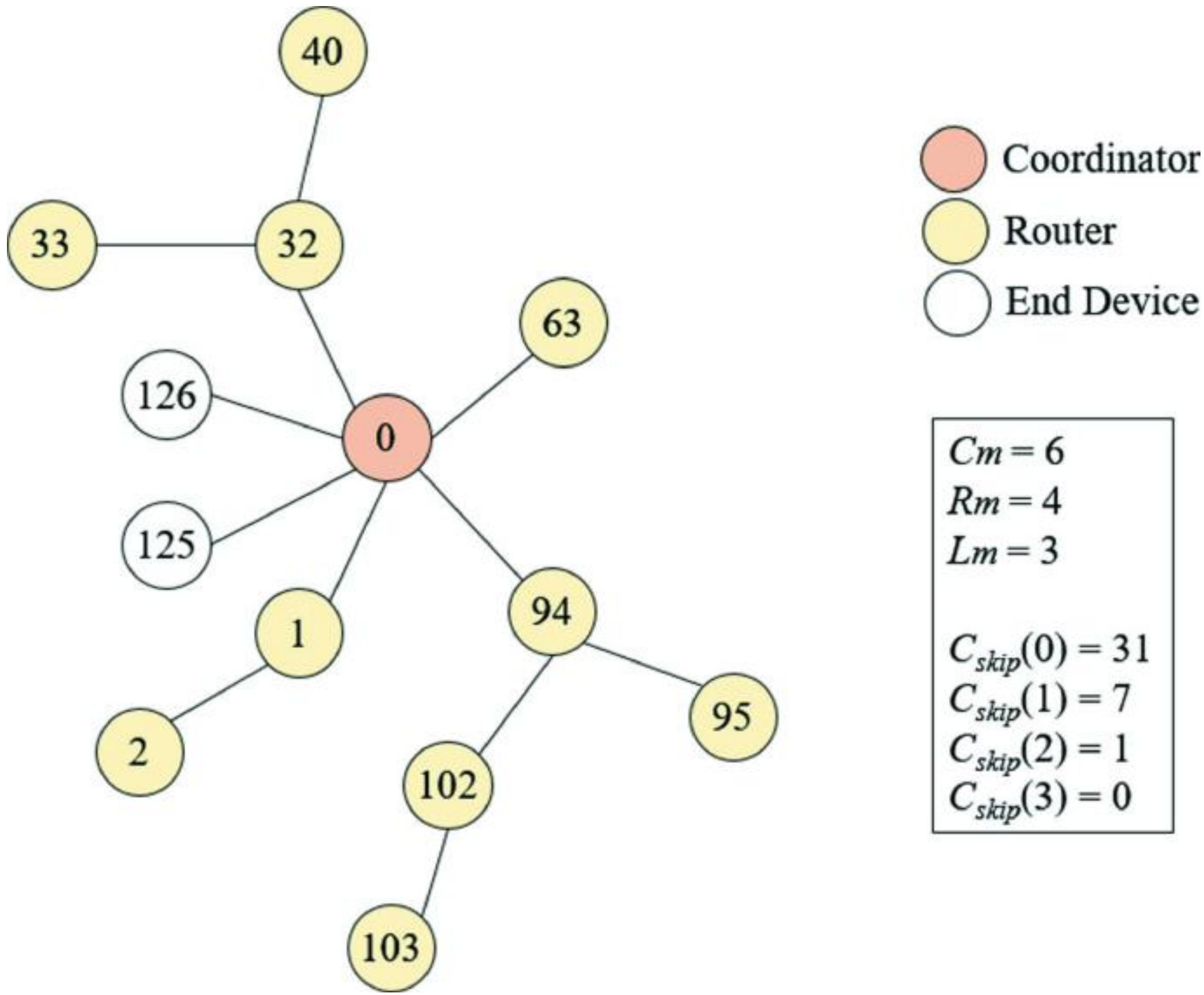
$R_{112} = 2 + 1 + 1 = 4$

$R_{113} = 2 + 2 + 1 = 5$

Find the address of the following devices of the zigbee network. The sequence of the joining is

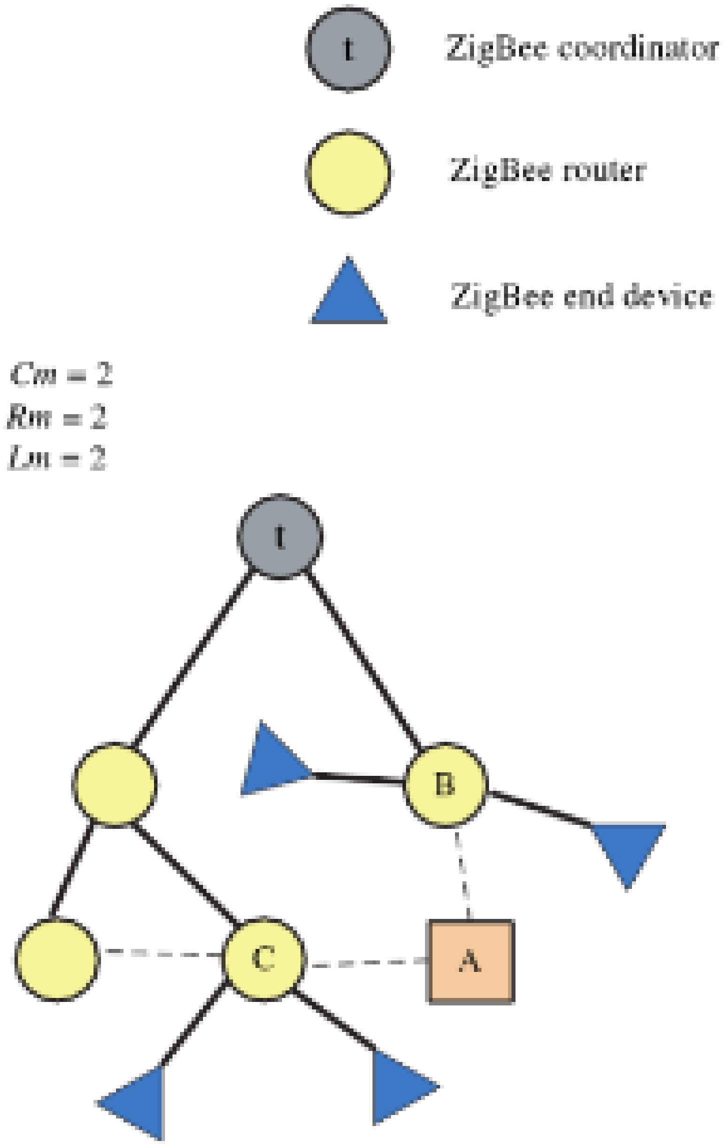
R2->R1->R3->R4->E1->R12->R21->R11->R42->R41->R411->E2




40
33
32
63
126
0
125
1
94
95
2
102
103
Coordinator
Router
End Device
$Cm = 6$
$Rm = 4$
$Lm = 3$
$C_{skip}(0) = 31$
$C_{skip}(1) = 7$
$C_{skip}(2) = 1$
$C_{skip}(3) = 0$

- While these parameters facilitate address assignment, they may sometimes prohibit a node from joining a network.
- **A node is called an orphan node when it can not associate with the network but there are still unused address spaces remaining.**
- This situation is called the **orphan problem.**
- For example, in Fig., the router-capable device A has two potential parents B and C. But, router A cannot associate to router B or C because B and C have reached their maximum capacity of Cm = 2 children. So, A becomes an orphan node.

Orphan problem



REF:

- https://datatracker.ietf.org/doc/html/rfc3561
- Handbook On Sensor Networks. (2010). Singapore: World Scientific Publishing Company.
- Tennina, S., Koubâa, A., Daidone, R., Tovar, E., Jurčík, P., Pereira, N., Severino, R., Hauer, J., Bouroche, M., Dini, G., Alves, M., Tiloca, M. (2013). IEEE 802.15.4 and ZigBee as Enabling Technologies for Low-Power Wireless Systems with Quality-of-Service Constraints. Germany: Springer Berlin Heidelberg.