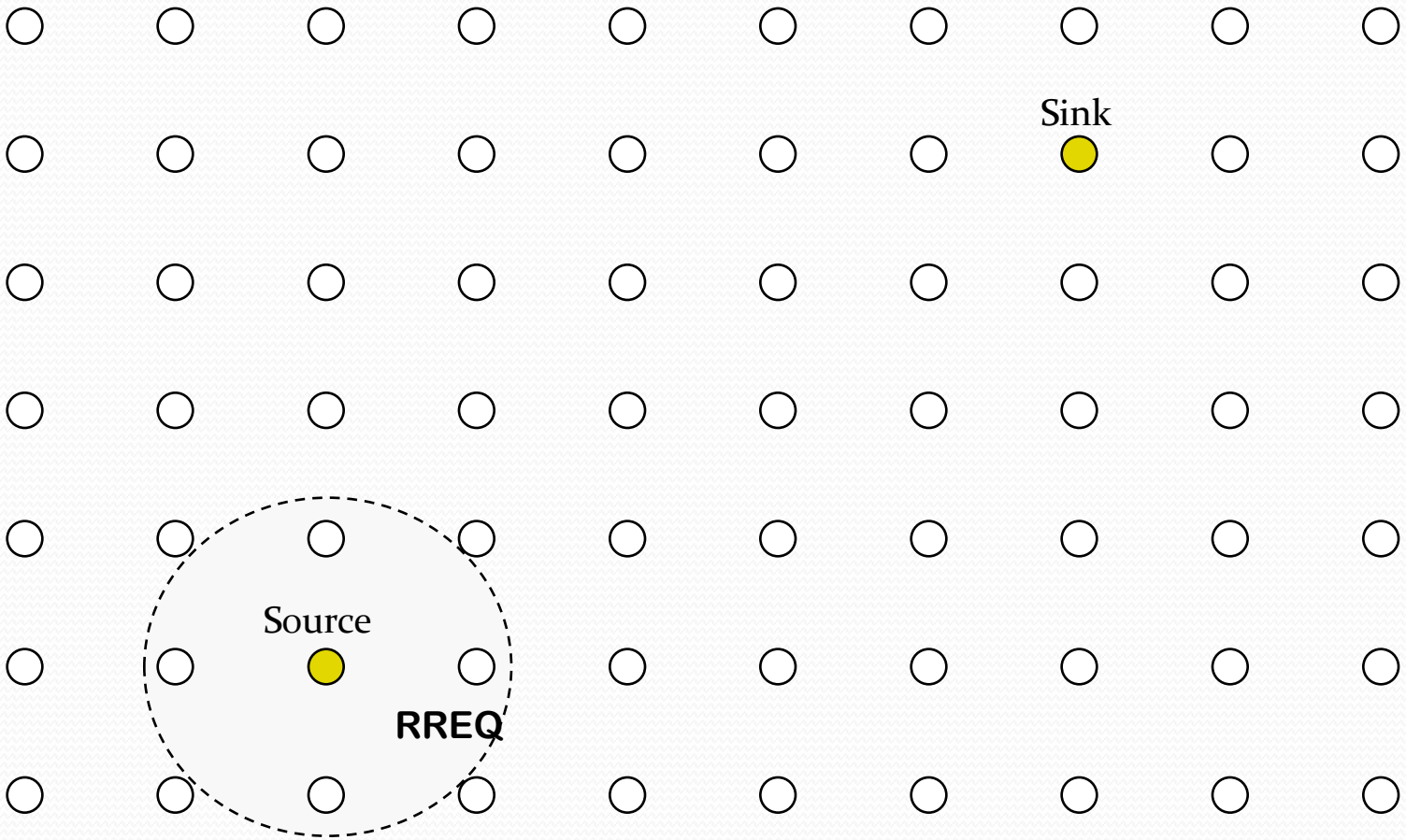
Ad-hoc On-demand Distance Vector (AODV)

Junseok Kim
Wireless Networking Lab (WINLAB)
Konkuk University, South Korea
http://usn.konkuk.ac.kr/~jskim

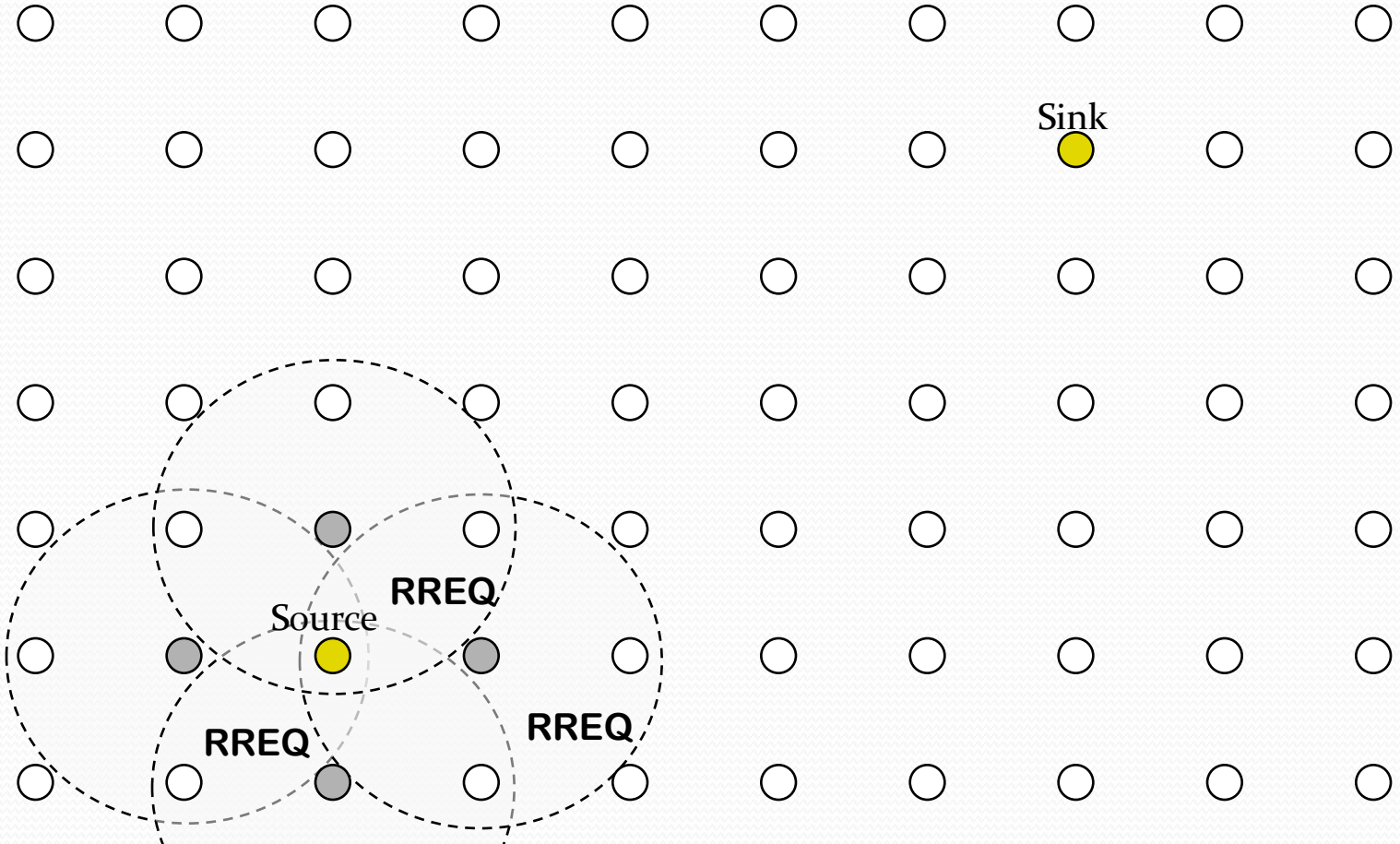
AODV - route establishment

- Broadcast *R*oute *Req*uest (RREQ) packet to find where the sink node is



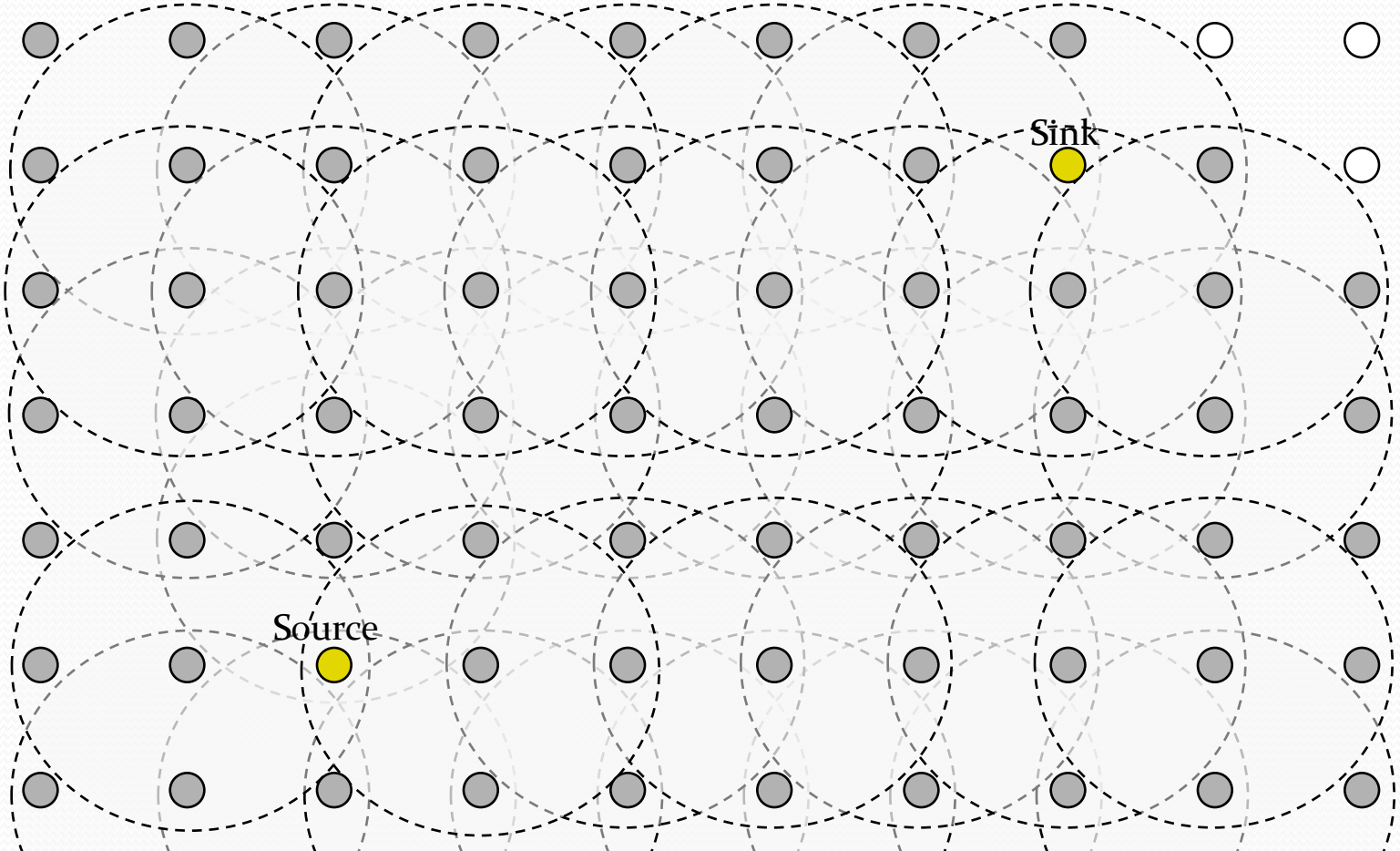
Route establishment (Cont.)

- Flood RREQ packets over the network



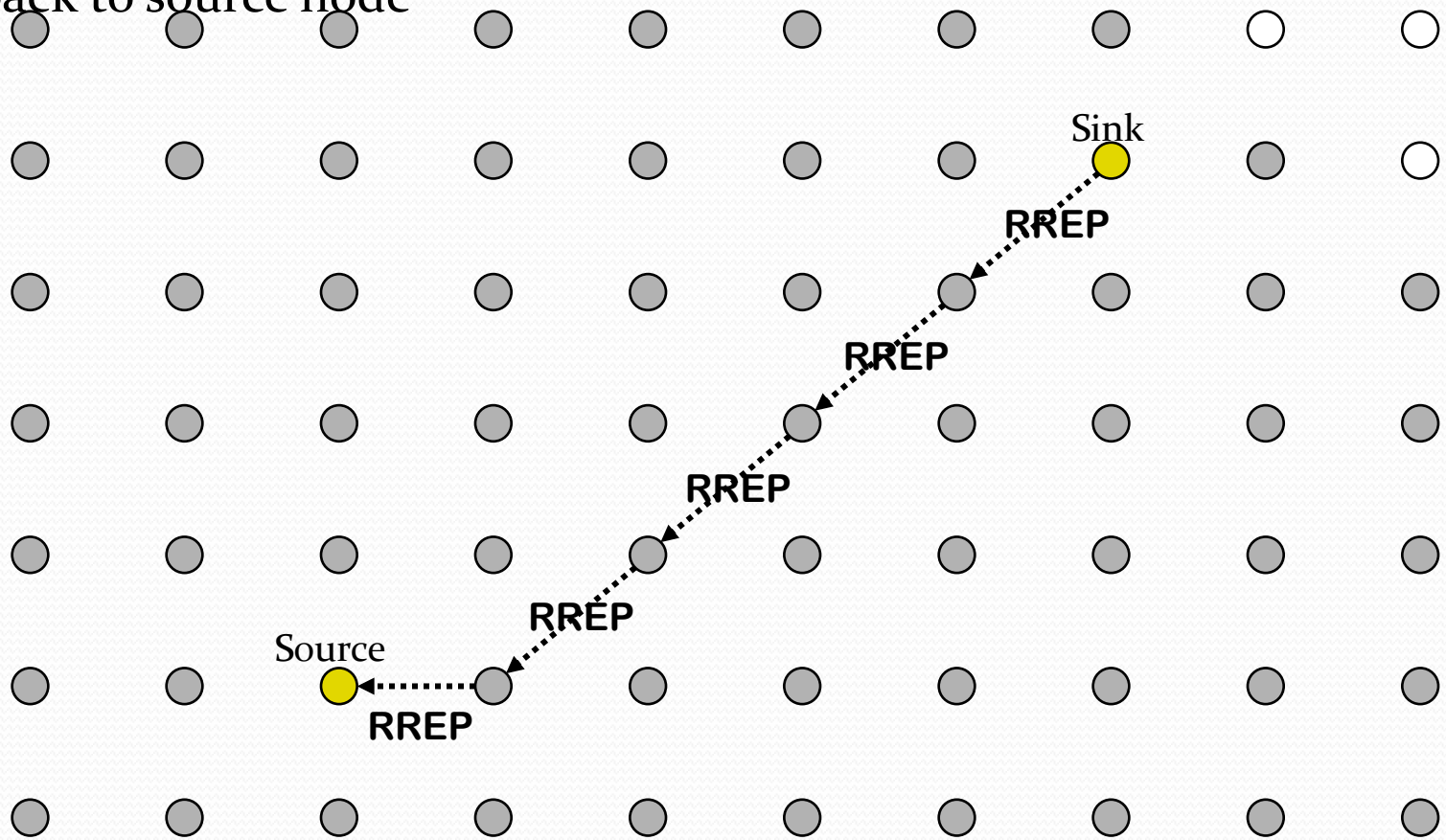
Route establishment (Cont.)

- Flood RREQ packets over the network



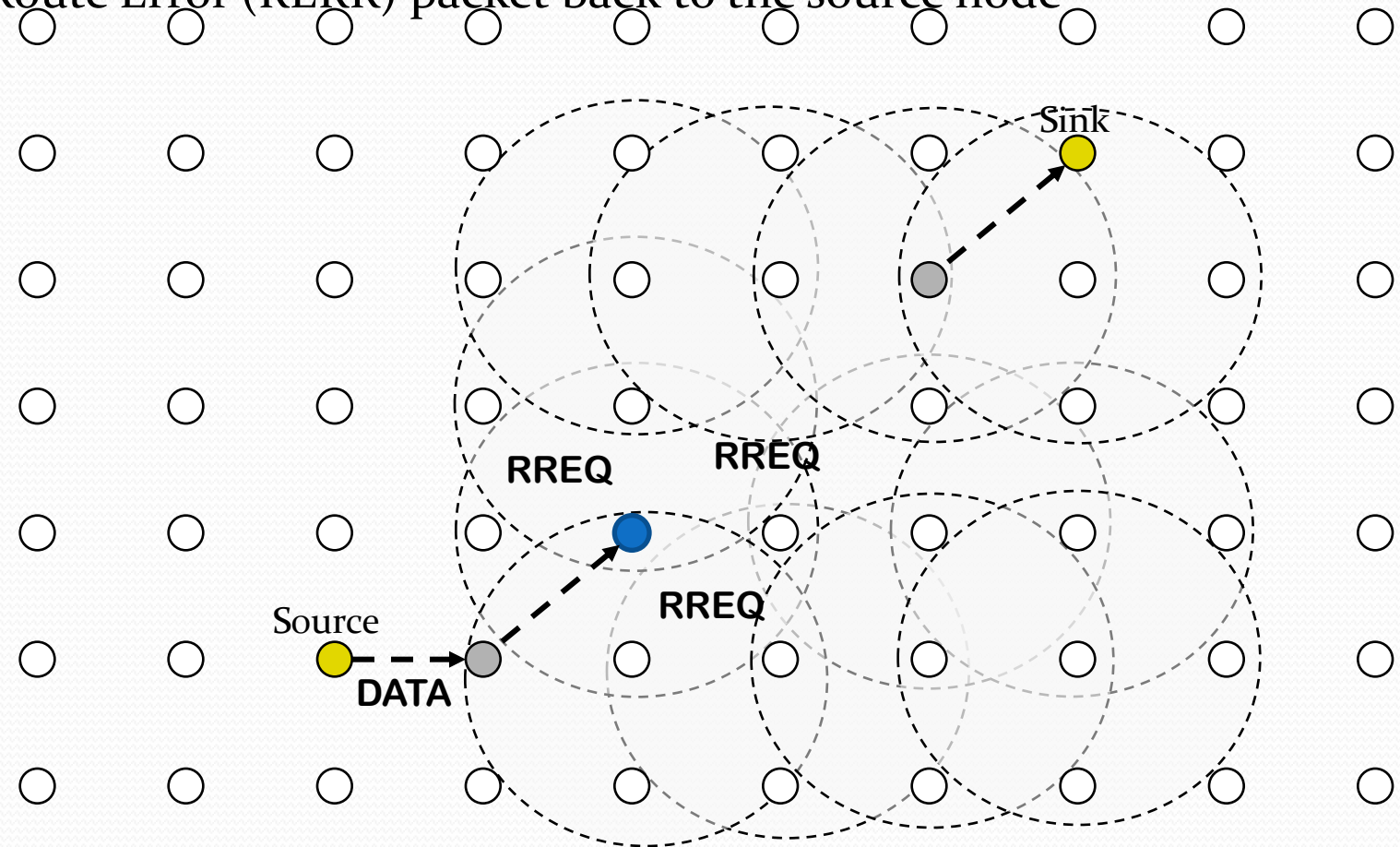
Route establishment (Cont.)

- When sink node received RREQ, it forwards *R*oute *Rep*ly (RREP) packet back to source node



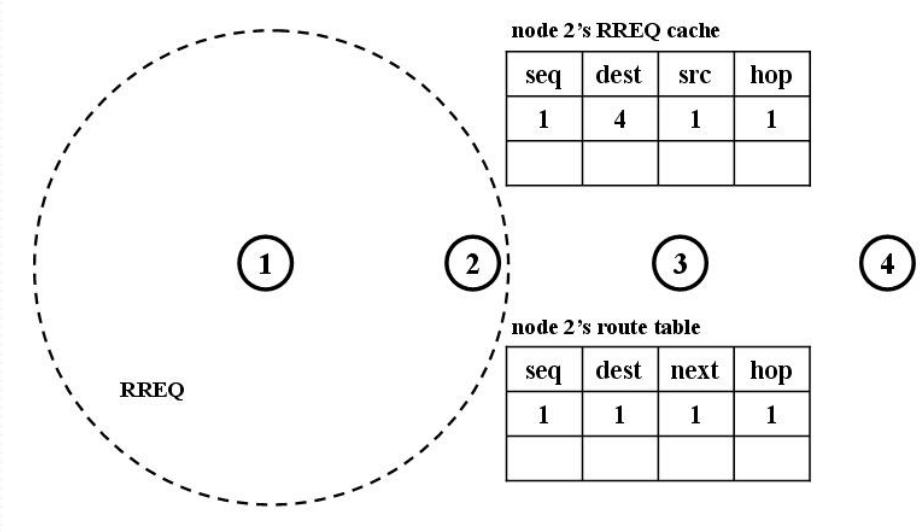
Route Recovery

- When a node detects a route break, it floods RREQ again or forwards Route Error (RERR) packet back to the source node



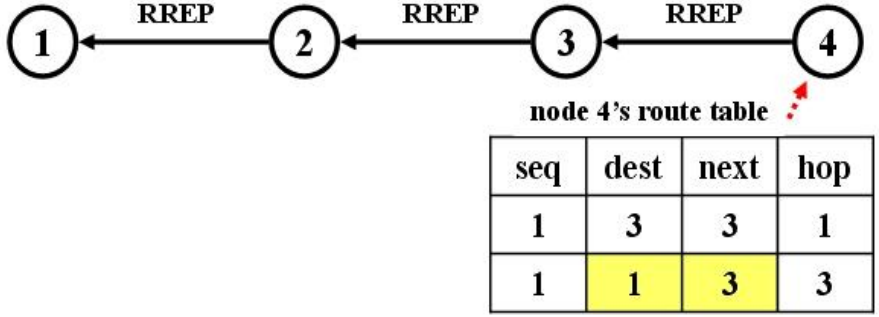
Route Establish (details)

- When AODV agent received a data message from its upper layer,
  - it first looks the route table to find the next hop information
  - If there is no route, AODV broadcasts RREQ
- For example shown below,
  - when node 2 received RREQ,
  - it stores information (sequence, destination address, etc) into RREQ cache for not re-broadcasting same RREQ
  - node 2 also inserts information of previous node and source node into the route table



Route Establish (details)

- When node 4 received RREQ, a reverse route (4→1) is established
  - Then, node 4 forwards RREP back to node 1
  - Node 4 sends RREP to node 3 since the next hop node for node 1 is node 3

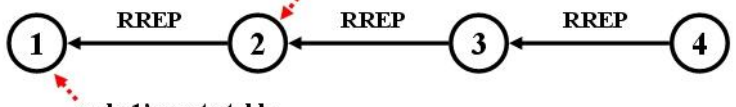


seq	dest	next	hop
1	3	3	1
1	1	3	3

  - When node 1 received RREP, a forward route (1→4) is established

node 2's route table

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

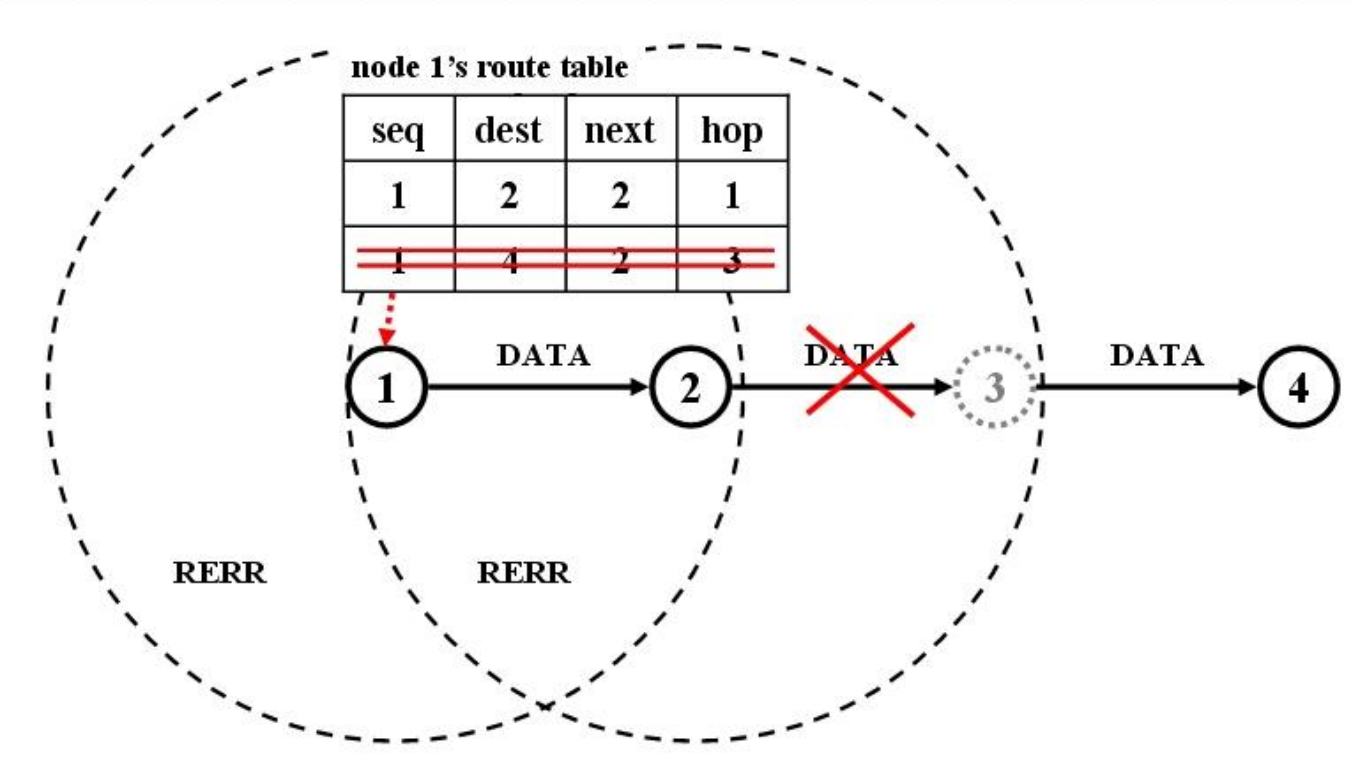


node 1's route table

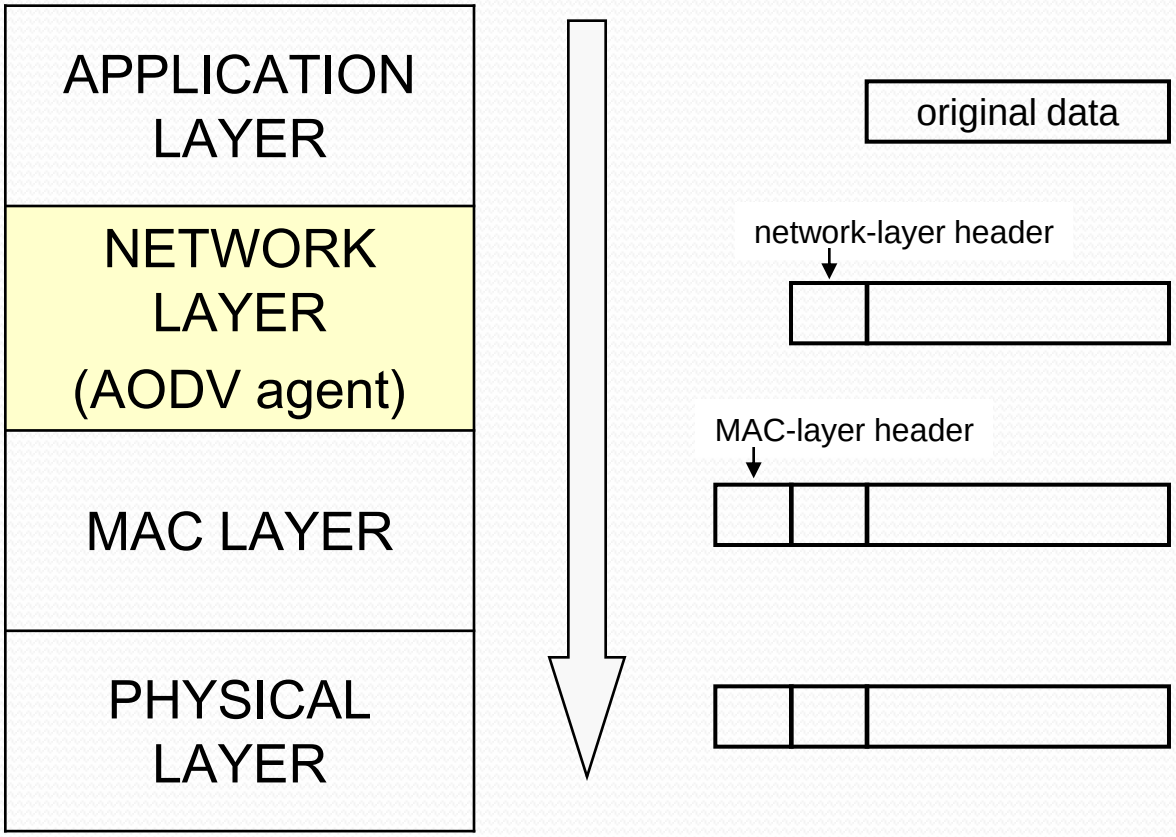
seq	dest	next	hop
1	2	2	1
1	4	2	3

Route Recovery (details)

- When node 2 continuously fails to transmit packets to node 3,
  - it sends a route error (RERR) packet to node 1
  - When a node received RERR, it delete information of node 3 in route table
  - After node 1 received RERR, it restarts RREQ flooding

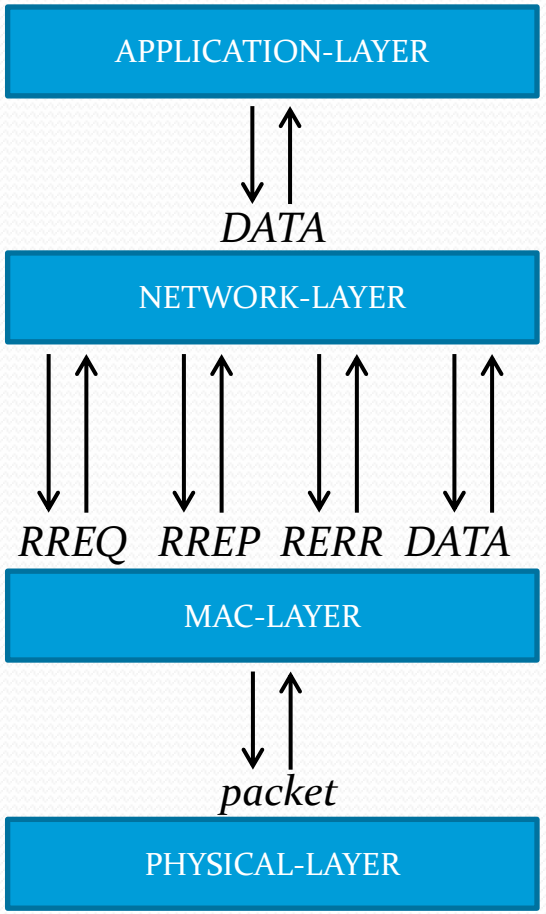


Packet Encapsulation

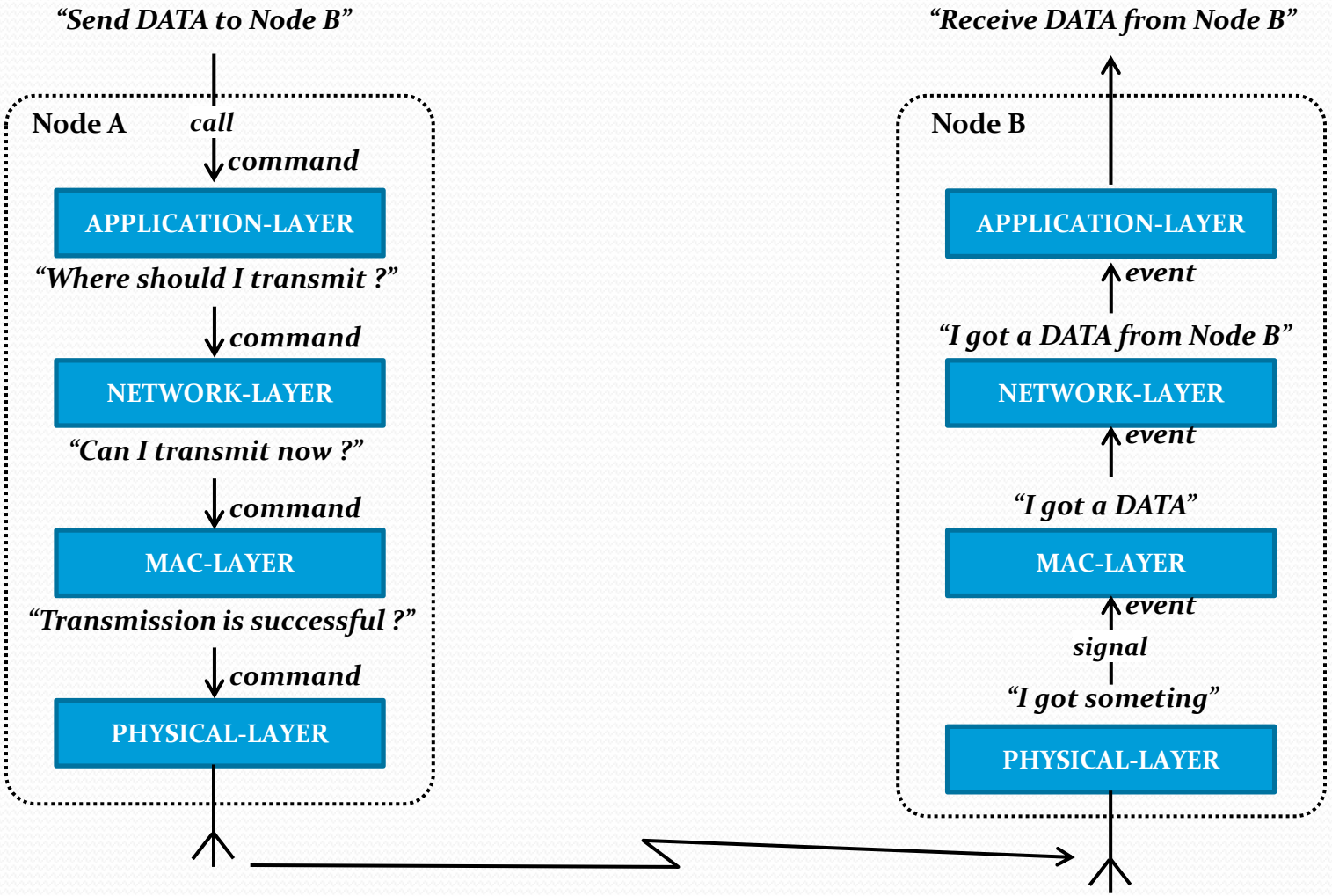


TinyOS Implementation

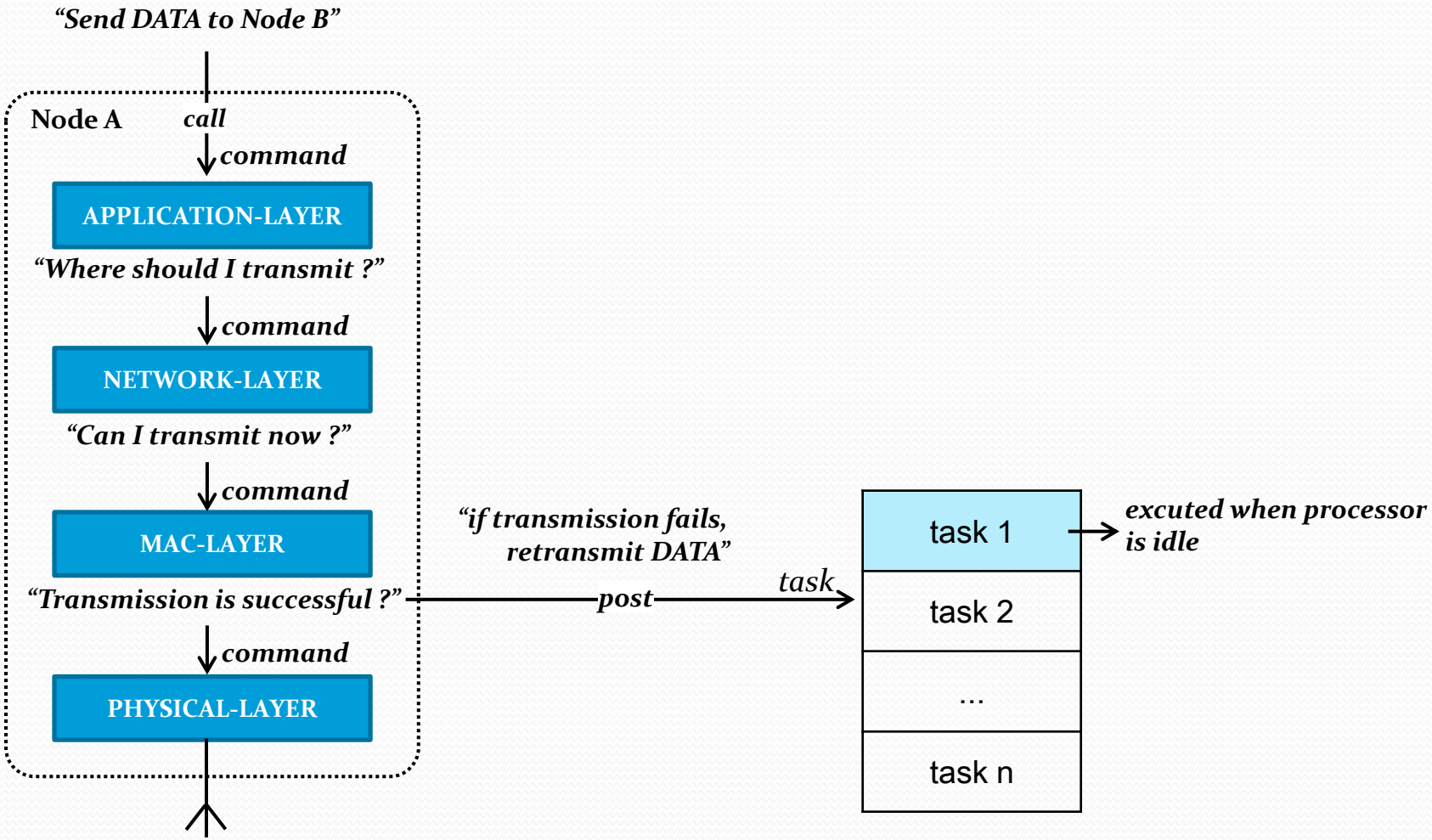
```
module AODV_M {
  provides {
    interface SplitControl;
    interface AMSend[am_id_t id];
    interface Receive[uint8_t id];
  }

  uses {
    interface SplitControl as AMControl;
    interface Timer<TMilli> as Timer;
    interface Leds;
    interface AMPacket;
    interface Packet;
    interface AMSend as SendRREQ;
    interface AMSend as SendRREP;
    interface AMSend as SendRERR;
    interface Receive as ReceiveRREQ;
    interface Receive as ReceiveRREP;
    interface Receive as ReceiveRERR;
    interface AMSend as SubSend;
    interface Receive as SubReceive;
    interface PacketAcknowledgements;
  }
}
```



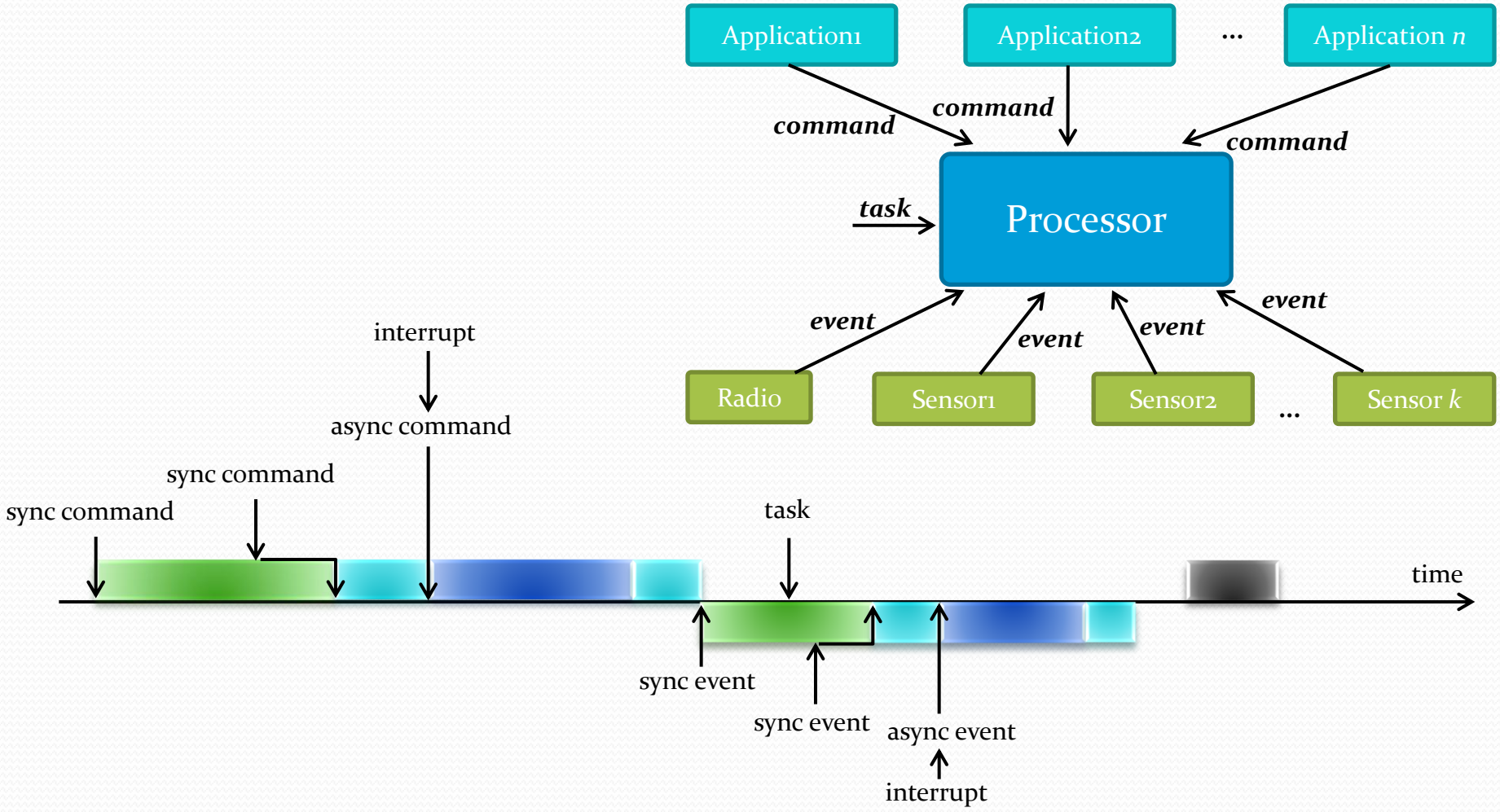
(sync, async) command, event, task



command, event, task (Cont.)



command, event, task (Cont.)



Q and A