NETWORKED EMBEDDED SOFTWARE
LAB

POLITECNICO
MILANO 1863

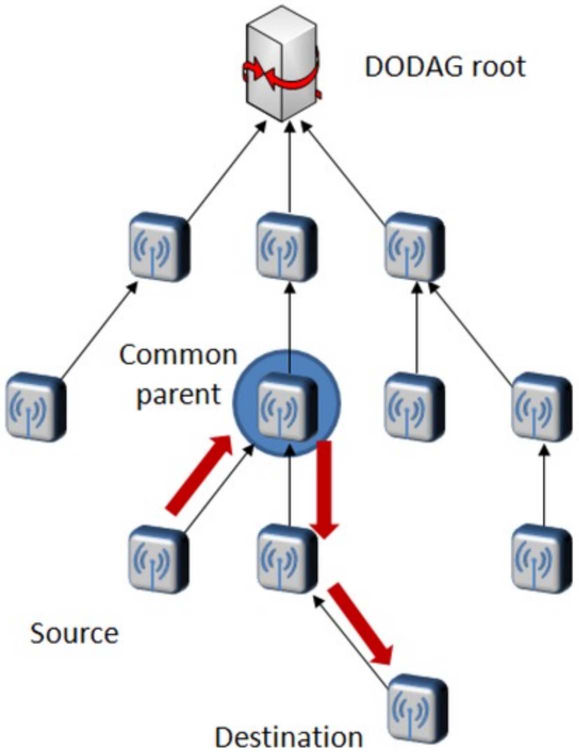
RISE

Contiki-NG

RPL

Luca Mottola
luca.mottola@polimi.it

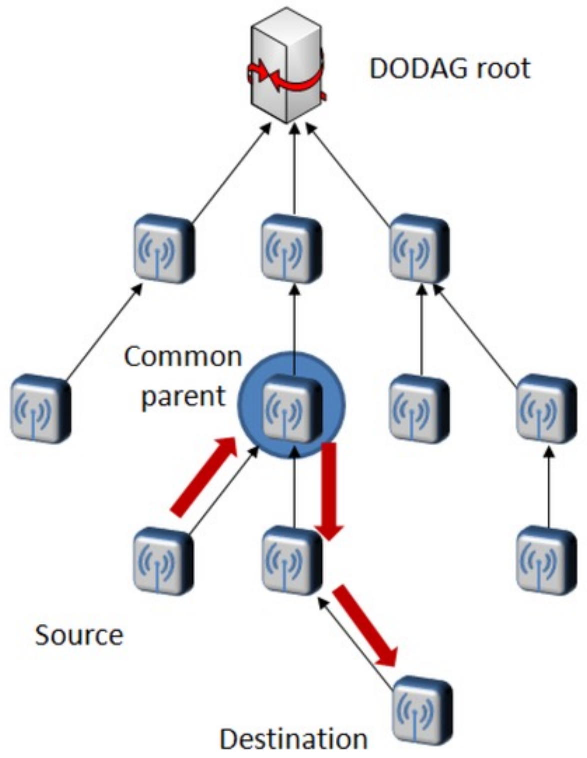
Routing: RPL (1)



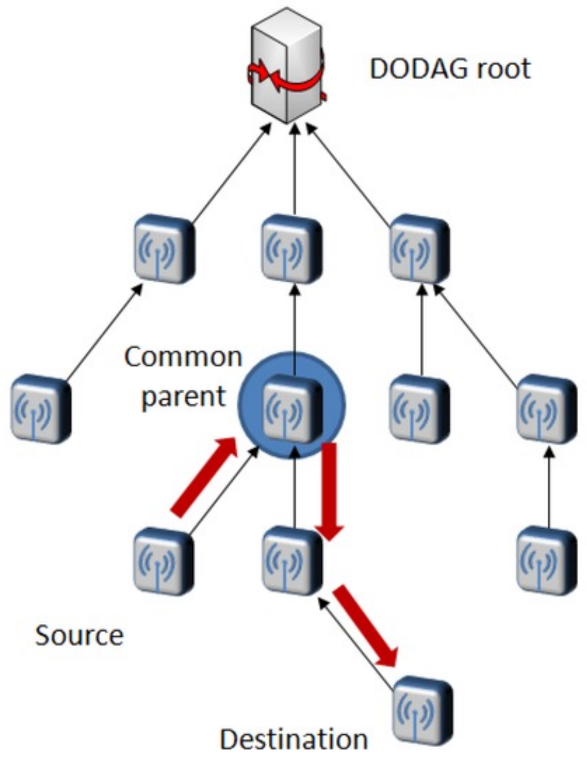
- Creates a tree-shaped topology rooted at the node with direct Internet access
  - Destination-Oriented Directed Acyclic Graph (**DODAG**)
  - Naturally supports **many-to-one** communication
- Packets called DODAG Information Objects (**DIO**) disseminated downwards from the root
  - Carry information on what **objective function** to use to select the parent in the tree
  - Every node has a **preferred parent** and might have multiple backup parents

Routing: RPL (2)

- Objective functions in Contiki-NG
  - **OF0**: looks for a ”good-enough” potential parent and a back-up one;
    - Simple but
    - ...might be inefficient and yield unstable routes
  - **MRHOF**: looks for the minimum rank among potential parents and applies histerisis to avoid flipping between parents;
    - More efficient provided good connectivity



Routing: RPL (3)



- To support one-to-many and one-to-one communication, every node advertises itself upwards with Destination Advertisement Objects (**DAO**)
- In **storing mode**, every intermediate node builds local routes for downstream nodes
  - Allows routes to “shortcut” paths at the first common parent
  - Might be heavy on memory
- In **non-storing** mode, only the root stores downstream routes
  - Point-to-point routes necessarily stretch up to the route
  - The root typically has memory

UDP over RPL: API

```
struct simple_udp_connection {
  struct simple_udp_connection *next;
  uip_ipaddr_t remote_addr;
  uint16_t remote_port, local_port;
  simple_udp_callback receive_callback;
  struct uip_udp_conn *udp_conn;
  struct process *client_process;
};


typedef void (* simple_udp_callback)
  (struct simple_udp_connection *c,
   const uip_ipaddr_t *source_addr,
   uint16_t source_port,
   const uip_ipaddr_t *dest_addr,
   uint16_t dest_port,
   const uint8_t *data, uint16_t datalen);
```

Represents a UDP connection

Executed for incoming packets

Receive callback signature

UDP over RPL: API

Initializes UDP layer

```
void simple_udp_init(void);
```

Setup a UDP connection

```
int simple_udp_register(struct simple_udp_connection *c,
                        uint16_t local_port,
                        uip_ipaddr_t *remote_addr,
                        uint16_t remote_port,
                        simple_udp_callback receive_callback);


int simple_udp_sendto(struct simple_udp_connection *c,
                      const void *data, uint16_t datalen,
                      const uip_ipaddr_t *to);
```

Send a packet to a specific destination, if the UDP connection is setup with **`NULL`** as **`remote-addr`**

UDP over RPL: Example

```
simple_udp_register(&udp_conn, UDP_CLIENT_PORT, NULL,
                    UDP_SERVER_PORT, udp_rx_callback);

etimer_set(&periodic_timer, random_rand() % SEND_INTERVAL);
while(1) {
  PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&periodic_timer));

  if(NETSTACK_ROUTING.node_is_reachable()) {
    rpl_dag_t *dag = NETSTACK_ROUTING.get_root_ipaddr(&dest_ipaddr);
    if(dag != NULL) { /* Only a sanity check. */
      LOG_INFO("Sending request %u to ", count);
      LOG_INFO_6ADDR(&dest_ipaddr);
      LOG_INFO_("\n");
      simple_udp_sendto(&udp_conn, &count, sizeof(count), &dest_ipaddr);
      count++;
    }
  } else {
    LOG_INFO("Not reachable yet\n");
  }
```

No remote address

Send to the DODAG root

Send the value of count