

RPL

Carlo Vallati

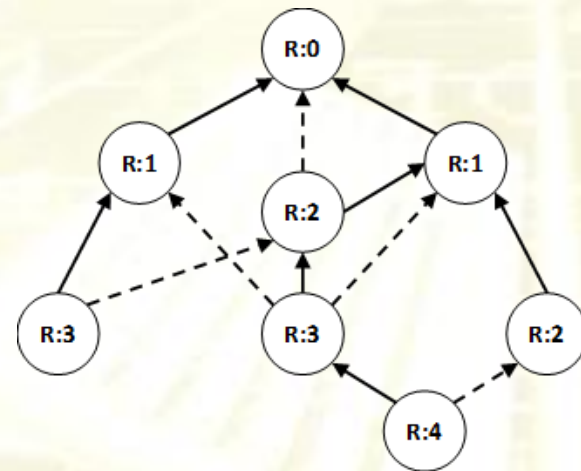
Assistant Professor @ University of Pisa

c.vallati@iet.unipi.it

RPL



- RPL stands for Routing Protocol for Low-power and Lossy Networks
- It is a Layer 3 routing protocol aimed at building a Destination Oriented Directed Acyclic Graph
- Every network must have a root node that originates the RPL DAG
- Each node selects a set of neighbors as parent set
- A Preferred parent is selected for data forwarding



RPL - Messages

- *DODAG Information Object* (DIO) messages: broadcast by every node for upward route formation
- *DODAG Information Solicitation* (DIS) message: sent asynchronously to request routing information
- *DODAG Advertisement Object* (DAO) message: sent from nodes to root to form downward routes

Enable RPL in Contiki

- In the program:
 - `#include "net/rpl/rpl.h"`
- In the Makefile
 - `CFLAGS+= -DUIP_CONF_IPV6_RPL`
- For debugging or stats collection:
 - `CFLAGS+= -DRPL_CONF_STATS=1`

Enable RPL in Contiki

- In the project-conf.h:
#undef UIP_CONF_IPV6_RPL
#define UIP_CONF_IPV6_RPL 1
- RA protocol can be disabled (disabled by default):
#undef UIP_CONF_ND6_SEND_RA
#define UIP_CONF_ND6_SEND_RA 0

Initialize RPL

- In non-root nodes RPL is initialized automatically as the node starts
- In the RPL ROOT node RPL must be initialized explicitly to enable ROOT operations and include the network prefix to be advertized:

```
rpl_dag_t *dag;  
dag = rpl_set_root(RPL_DEFAULT_INSTANCE,  
                  (uip_ip6addr_t *)&ipaddr);  
uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0);  
rpl_set_prefix(dag, &ipaddr, 64);
```

Right after network interface initialization

RPL global repair

- RPL defines a procedure called *global repair*, which is invoked by the root node to reset all the routing information in the network and start over from a clean state
- Root node can trigger RPL global repair calling this function:

```
rpl_repair_root(RPL_DEFAULT_INSTANCE);
```

Do it!!



- Modify the examples *receiver.c* and *unicast-sender.c* from the previous lesson in order enable RPL (the receiver is the ROOT node).
- Modify the code of the root node in order to trigger the global repair procedure when the USR button of the mote is pressed.
- Check through wireshark the RPL messages, compare DIOs before and after the reset

Recall Trickle

- Each node maintains a counter c and a timer t in range $[I/2, I]$ (at start, $I = I_{\min}$)
- When a node receives metadata that is “consistent”, it increments c
- At time t , the node broadcasts a DIO message if $c < K$ (redundancy threshold)
- When the interval I expires
 - I is doubled (up to I_{\max})
 - c is reset to zero
 - t is reset to a new value in the range $[I/2, I]$
- When a node receives a DIO message with metadata that is “inconsistent” I is reset to I_{\min} (also c and t are reset)

Contiki Trickle

- All the RPL configuration parameters are in:
 - `core/net/rpl/rpl-conf.h`
- `K = RPL_CONF_DIO_REDUNDANCY`
- `Imin = RPL_CONF_DIO_INTERVAL_MIN`
 - $2^{RPL_CONF_DIO_INTERVAL_MIN}$
- `Imax = RPL_CONF_DIO_INTERVAL_DOUBLINGS`
 - $2^{(RPL_CONF_DIO_INTERVAL_MIN + RPL_CONF_DIO_INTERVAL_DOUBLINGS)}$

Change trickle parameter

- Trickle parameters can be modified in the project-conf.h file (see core/net/rpl/rpl-conf.h):

```
#undef RPL_CONF_DIO_REDUNDANCY  
#define RPL_CONF_DIO_REDUNDANCY 1
```

```
#undef RPL_CONF_DIO_INTERVAL_MIN  
#define RPL_CONF_DIO_INTERVAL_MIN 3
```

```
#undef RPL_CONF_DIO_INTERVAL_DOUBLINGS  
#define RPL_CONF_DIO_INTERVAL_DOUBLINGS 5
```

Display RPL output

- To obtain an insight on RPL operations the advanced debug can be enabled inside the single files within `core/net/rpl/`
- To investigate Trickle the `DEBUG` macro in `rpl-timers.c` can be set to `DEBUG_PRINT`
- Custom *'printf'* can be added in order to track custom events

Do it!!



- Deploy a network in Cooja with many nodes and change some RPL parameters (e.g. set the redundancy threshold to 1)
- Check simulation logs to observe the behaviour of trickle in each node enabling the log of the file “rpl-timers.c”