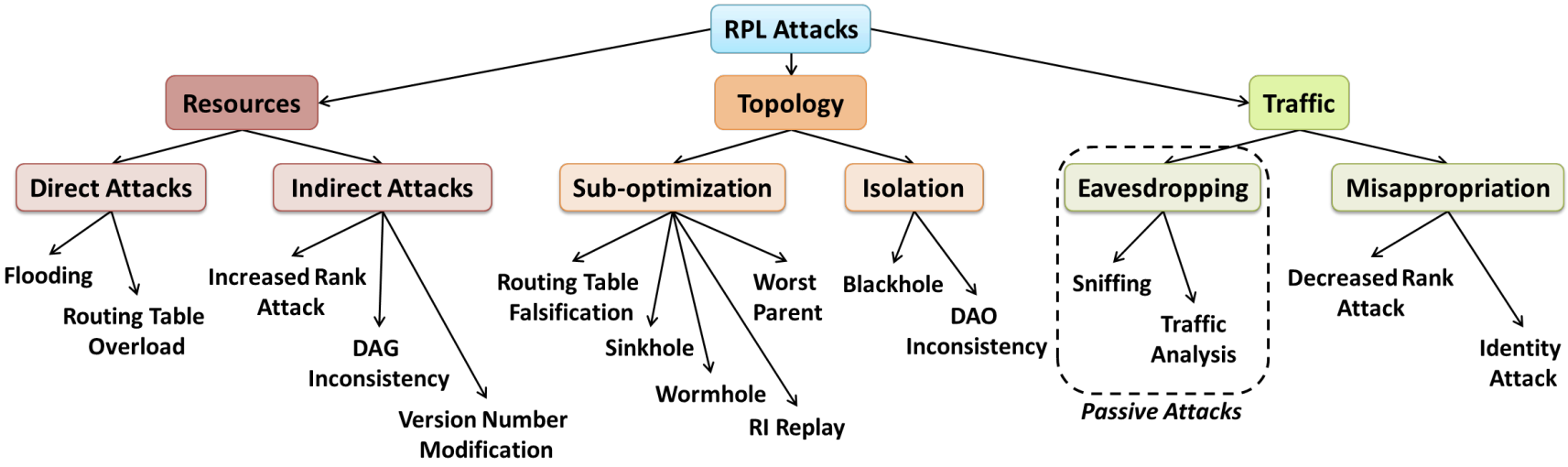
Decreased Rank Attack Tutorial

Traffic Attacks

- Traffic attacks are one category of security attacks on the RPL protocol, as shown below
  - Their purpose is to introduce malicious nodes that do not disturb the network, e.g, for information leakage by traffic eavesdropping or impersonating legitimate nodes



Source: https://hal.inria.fr/hal-01207859/document

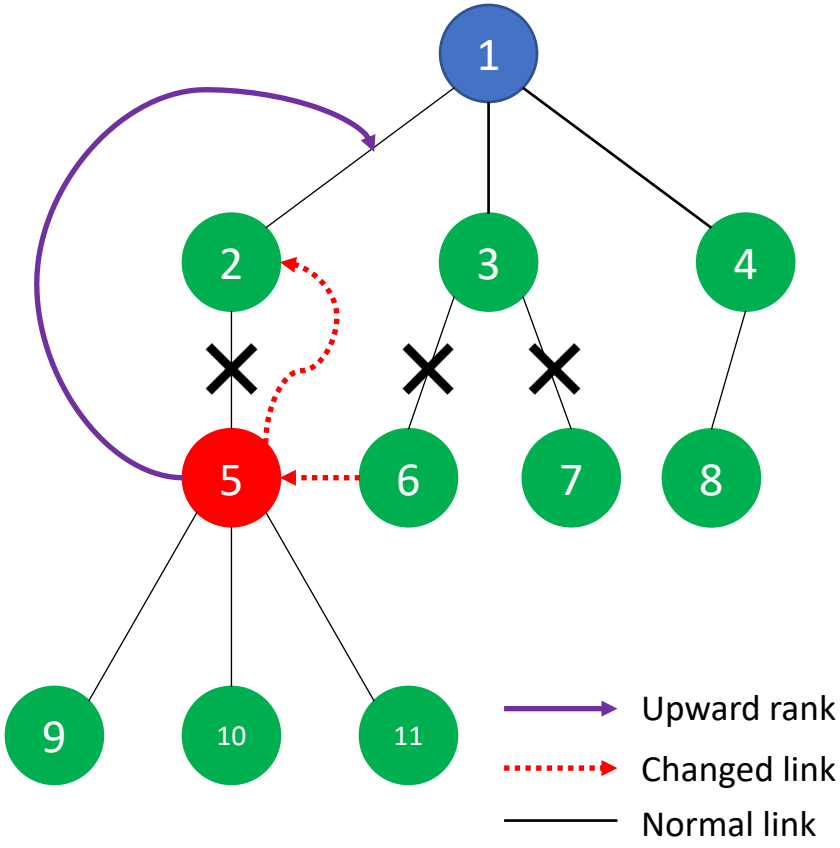
Traffic Attacks (cont.)

- Traffic attacks can be divided into two classes
  - Eavesdropping attacks, which deploy malicious nodes that perform eavesdropping activities, such as sniffing and analyzing network traffic
  - Misappropriation attacks, in which the identity of a legitimate node is usurped or its performance is overclaimed
    - These attacks are not directly so damaging to the RPL network, but they are often used as a first step for other attacks, such as resource or topology attacks
- An example of misappropriation attack is the decreased rank attack, that we shall address next

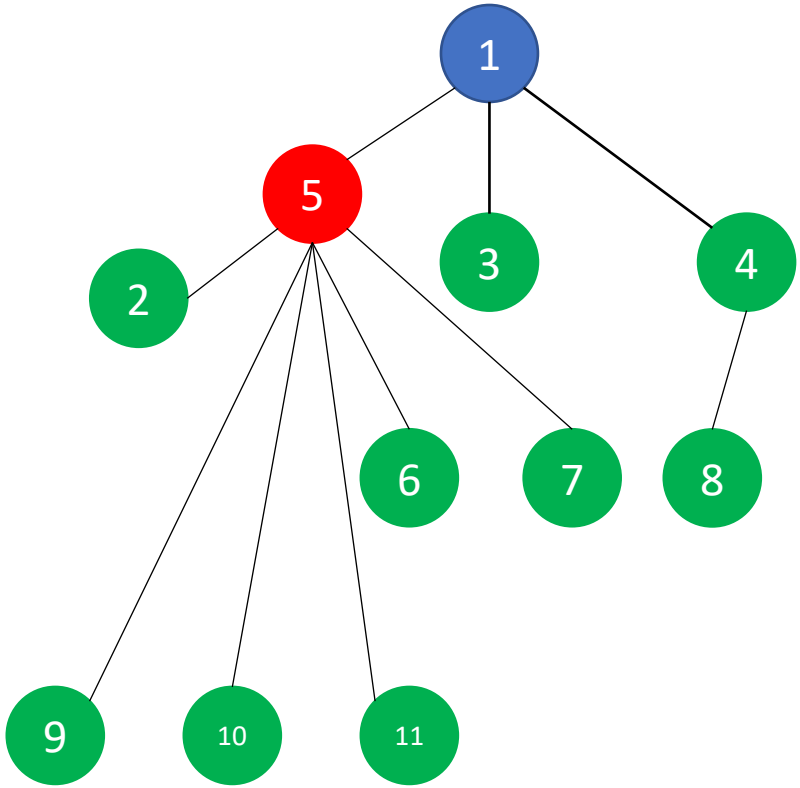
Decreased Rank Attack

- As learned before, in a DODAG graph, the lower a node rank is, the closer the node is to the topology root, and the more traffic the node has to manage
- If a malicious node illegitimately advertises a lower rank value (decreased rank attack), it overclaims its performance
  - As a result, many nodes connect to the DODAG graph via the malicious node
- The decreased rank attack can be a forerunner for sinkhole, blackhole or eavesdropping attacks

Decreased Rank Attack (cont.)



Initial network topology



Final network topology

Decreased Rank Attack Simulation

Decreased Rank Attack Simulation

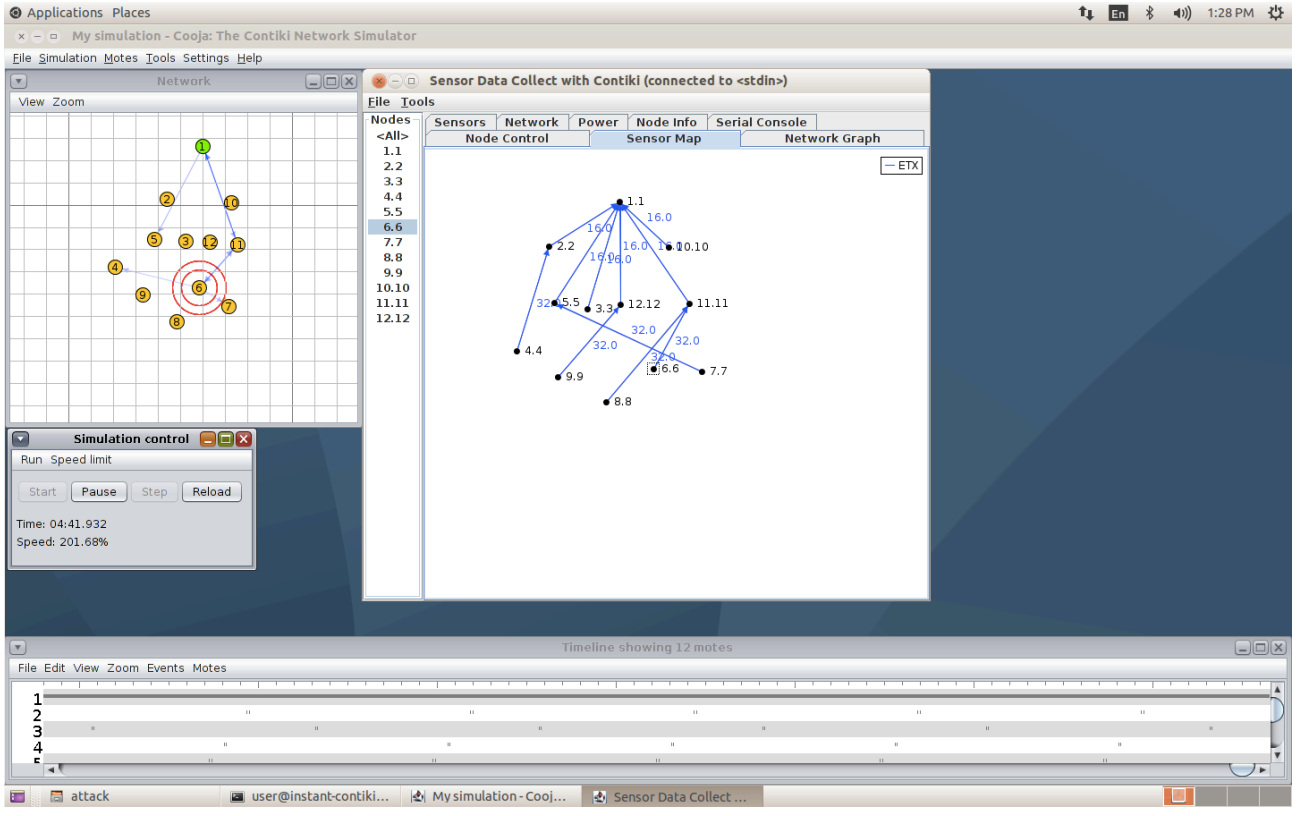
- Open the desired simulation in Cooja by selecting the corresponding scenario via the IoTrain-Sim interface
  - We recommend that you first select the “Reference Scenario Simulation” entry to view the reference scenario
- Alternatively, the simulations can be opened manually as follows
  - In Cooja, select the menu File > Open simulation > Browse…
  - Go to the folder “iotrain-sim/database/security_training/rank_attack/simulation/”
  - Select “rank_attack-reference.csc” for the reference scenario, and click “Open”

Decreased Rank Attack Simulation (cont.)

- Simulation and data collection procedure
  1. In the CollectView window, click on the “Start Collect” button, then click on the “Send command to nodes” button
  2. In the Simulation control window of Cooja, click on the “Start” button to begin the simulation
  3. Wait for at least two minutes of simulation time
  4. Back in the CollectView window, go to the Sensor map tab and see the network topology for the scenario
- Follow the same procedure to perform the attack simulation and compare the results
  - The attack scenario can be opened via the menu “Decreased Rank Attack Simulation” in IoTrain-Sim, or directly in Cooja via the file “rank_attack-simulation.csc”
  - You may need to wait for more than five minutes of simulation time to get statistics for all the nodes

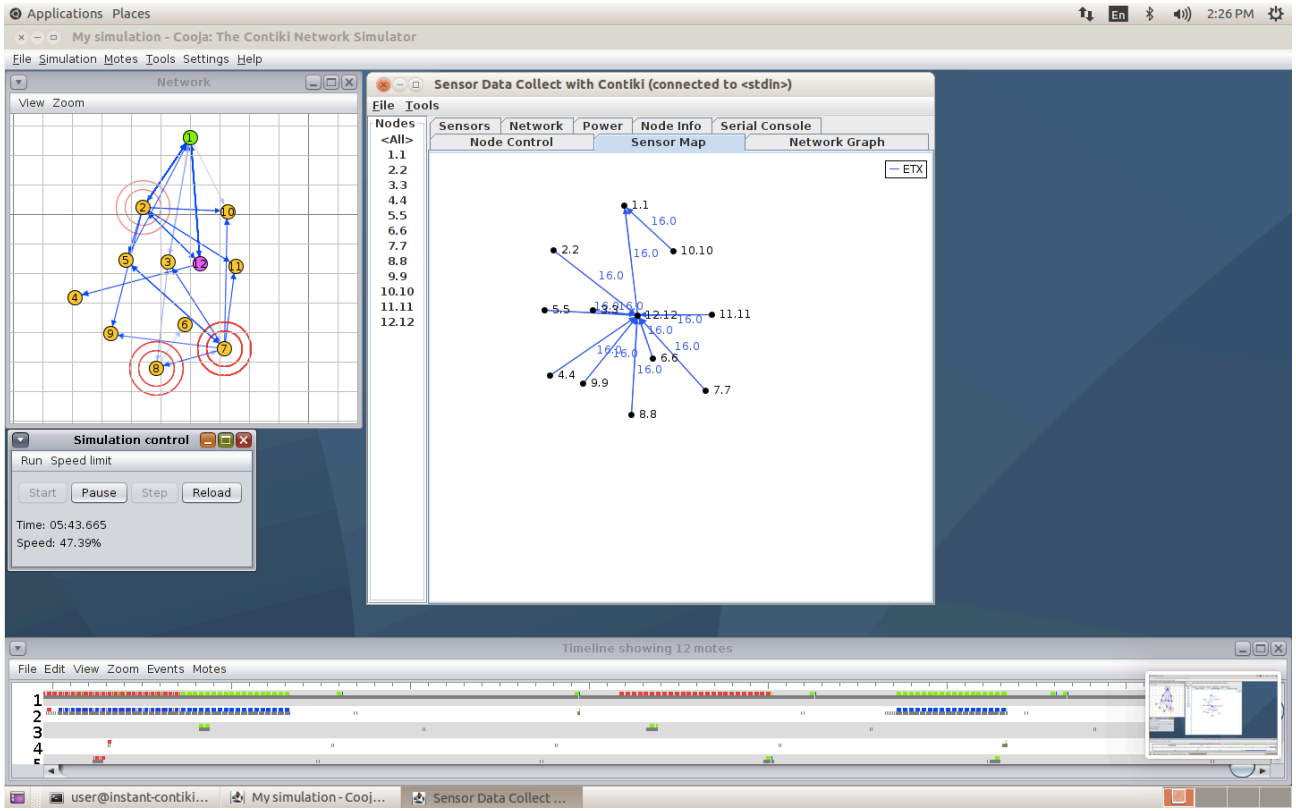
Reference Scenario and Results

- Node 1 (green color) is a SINK node that acts as a border router; the other nodes are sender nodes that act as normal sensors
- Notice that all the nodes are in the range of node 12, but nodes 4 through 9 are not in the range of node 1



Attack Simulation and Results

- Node 1 and the nodes in yellow color have the same roles as before
- Node 12 became a malicious node that is performing a decreased rank attack



Discussion

- Reference scenario
  - All the nodes are included in the network topology shown in Sensor map panel
    - Notice that nodes can be moved around with the mouse in the Sensor map panel to make the topology easier to see
  - The nodes can automatically determine and select the best path to the root node
- Attack simulation
  - As the malicious node 12 advertises a lower rank value, nearly all legitimate nodes connect via it to the network
    - Given the new topology, any other subsequent attacks from node 12 would have a major impact on the network

Decreased Rank Attack Implementation

Implementation Overview

- To implement the decreased rank attack, some changes are necessary to the normal source code for the RPL implementation in Contiki
- The files to be modified are located in the directory “contiki/core/net/rpl/”
  - rpl-private.h, which contains private declarations for the Contiki RPL implementation, such as the default values for the ICMP control messages and timers associated to them, modes of operation, DAG routing tables, etc.
  - rpl-timers.c, which is the RPL timer management implementation in Contiki

Changes to rpl-private.h

- The file “rpl-private.h” contains various constant definitions related to the DAG rank calculation
- The decreased rank attack can be implemented by altering some of these constants so as to interfere with the safeguards that are included in the rank computation algorithm

```
#ifndef RPL_CONF_MIN_HOPRANKINC
#define RPL_CONF_MIN_HOPRANKINC     0                                   //added set RPL_CONF_MIN_HOPRANKINC to 0
#define RPL_MIN_HOPRANKINC          256
#else
#define RPL_MIN_HOPRANKINC          RPL_CONF_MIN_HOPRANKINC
#endif
#define RPL_MAX_RANKINC             0                                   //change (7 * RPL_MIN_HOPRANKINC) to 0

#define DAG_RANK(fixpt_rank, instance) \
  ((fixpt_rank) / (instance)->min_hoprankinc)

/* Rank of a virtual root node that coordinates DAG root nodes. */
#define BASE_RANK                       0

/* Rank of a root node. */
#define ROOT_RANK(instance)             (instance)->min_hoprankinc

#define INFINITE_RANK                   256                          //change 0xffff to 256
```

Decreased rank attack implementation via modification of certain algorithm constants

Changes to rpl-timers.c

- The file “rpl-timers.c” contains code that recalculates the node ranks used in RPL
- The implementation of the decreased rank attack also requires to disable this recalculation, so that the effects of the rank decrease are not undone

```
/*---------------------------------------------------------------------------*/
static void
handle_periodic_timer(void *ptr)
{
  rpl_purge_routes();
  //rpl_recalculate_ranks();                    remove this line

  /* handle DIS */
#if RPL_DIS_SEND
  next_dis++;
  if(rpl_get_any_dag() == NULL && next_dis >= RPL_DIS_INTERVAL) {
    next_dis = 0;
    dis_output(NULL);
  }
#endif
  ctimer_reset(&periodic_timer);
}
/*---------------------------------------------------------------------------*/
```

Decreased rank attack implementation via disabling RPL rank recalculation

Exercises

- After making the suggested modifications in a copy of the Contiki source code, compile the files and assign the resulting malicious code to one of the motes in the reference scenario
- We suggest you use node 12 first as malicious one, as in our example, then change the malicious node to see how the simulation results change