МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Кіберзахист об’єктів критичної інфраструктури

Лабораторний практикум №4

Використання симулятора Cooja для моделювання атак на об’єкти критичної інфраструктури

Перевірив:
Войцеховський А. В.

Виконав:
студент І курсу
групи ФБ-41мп
Сахній Н. Р.

Київ 2024

Мета роботи: Ознайомитись із можливостями симулятора Cooja для моделювання атаки на емульовану мережу безпровідних пристроїв.

Завдання до виконання:

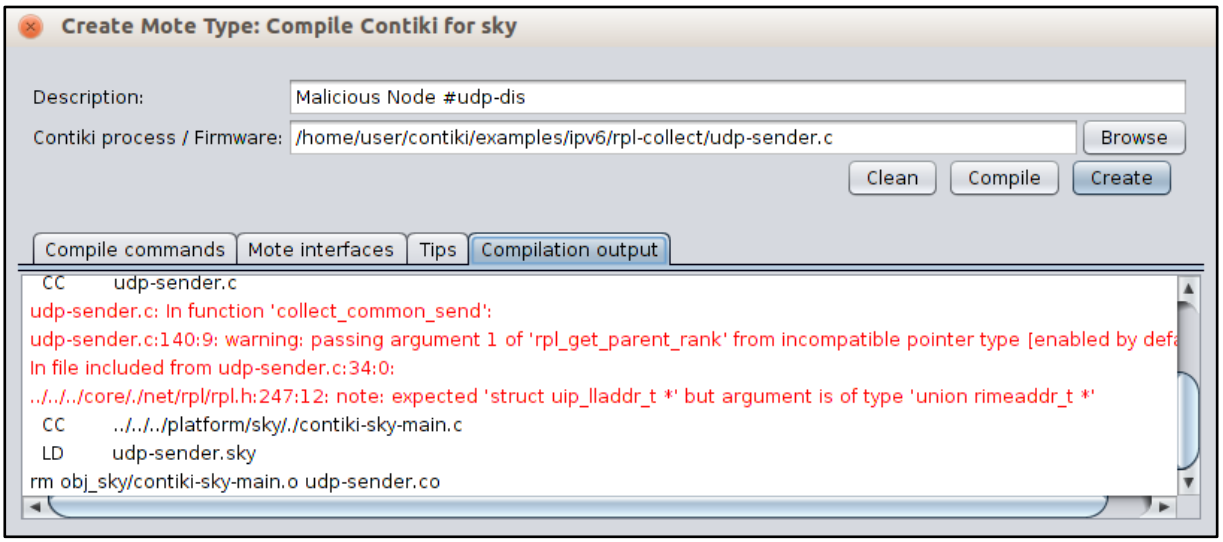
Варіант №5. Дано **5** вузлів типу UDP-sender **та 1** UDP-sink.

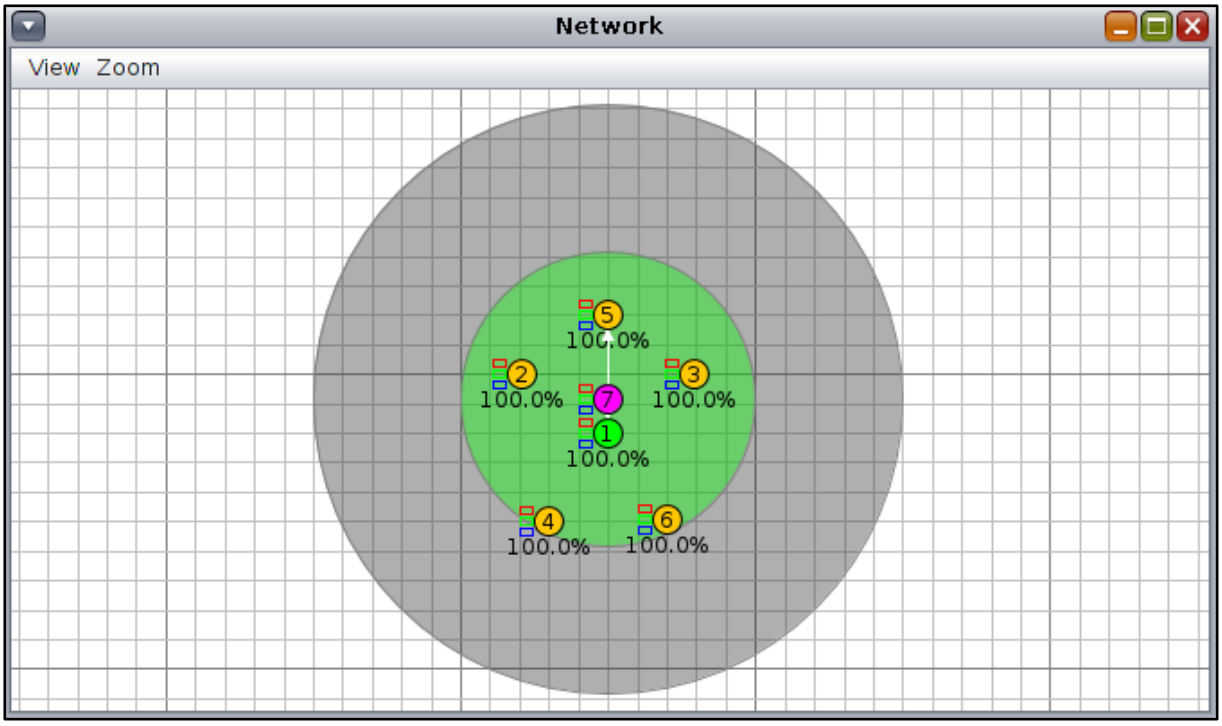
1. Зробити копію папки `rpl` до змін (`/contiki/core/net/rpl`).

RPL (Routing Protocol for Low-Power and Lossy Networks) – це протокол маршрутизації для малопотужних мереж, зокрема безпровідних мереж з низьким енергоспоживанням, як правило, чутливий до втрати пакетів:

```
user@instant-contiki:~/contiki/core/net$ cp -R rpl rpl_backup
user@instant-contiki:~/contiki/core/net$ ls
dhcpc.c       packetbuf.h      resolv.h      tcpdump.c     uip_arp.c        uip-fw.c      uip-neighbor.c
dhcpc.h       packetqueue.c    rime          tcpdump.h     uip_arp.h        uip-fw-drv.c  uip-neighbor.h
hc.c          packetqueue.h    rime.h        tcpip.c       uip.c            uip-fw-drv.h  uipopt.h
hc.h          psock.c          rpl           tcpip.h       uip-debug.c      uip-fw.h      uip-over-mesh.c
mac           psock.h          rpl_backup    uaodv.c       uip-debug.h      uip.h         uip-over-mesh.h
Makefile.uip  queuebuf.c       sicslowpan.c  uaodv-def.h   uip-ds6.c        uip-icmp6.c   uip-packetqueue.c
nbr-table.c   queuebuf.h       sicslowpan.h  uaodv.h       uip-ds6.h        uip-icmp6.h   uip-packetqueue.h
nbr-table.h   rawpacket.h      simple-udp.c  uaodv-rt.c    uip-ds6-nbr.c    uiplib.c      uip-split.c
netstack.c    rawpacket-udp.c  simple-udp.h  uaodv-rt.h    uip-ds6-nbr.h    uiplib.h      uip-split.h
netstack.h    rawpacket-udp.h  slipdev.c     uip6.c        uip-ds6-route.c  uip-nd6.c     uip-udp-packet.c
packetbuf.c   resolv.c         slipdev.h     uip_arp.h     uip-ds6-route.h  uip-nd6.h     uip-udp-packet.h
user@instant-contiki:~/contiki/core/net$
```

2. Додатково створити шкідливий вузол (псевдодатчик) в мережі.





3. Змінити файли `rpl_private.h` та `rpl_timers.c` (попередній крок)

Нижче показаний фрагмент коду із відповідними змінами у файлі `rpl_private.h`. Зміна значень здійснюється для того, щоби імітувати діяльність зловмисного вузла, який постійно надсилає DIS-повідомлення. Існують дві директиви, що визначають інтервал DIS-повідомлень та таймери затримки запуску. Обидва значення будуть встановлені до 0, у нормальному стані це ненульові параметри, а натомість 60 та 5 секунд:

```
 GNU nano 2.2.6                File: rpl-private.h                     Modified


/* DIS related */
#define RPL_DIS_SEND                    1
#ifdef  RPL_DIS_INTERVAL_CONF
#define RPL_DIS_INTERVAL                RPL_DIS_INTERVAL_CONF
#else
#define RPL_DIS_INTERVAL                0 /* Was 60 */
#endif
#define RPL_DIS_START_DELAY             0 /* Was 5 */
/*---------------------------------------------------------------------*/
```

Додавання **while**-циклу із i<20 дозволяє повторно виконувати функцію dis_output() до 20 разів, поки умови циклу залишаються виконаними. Це

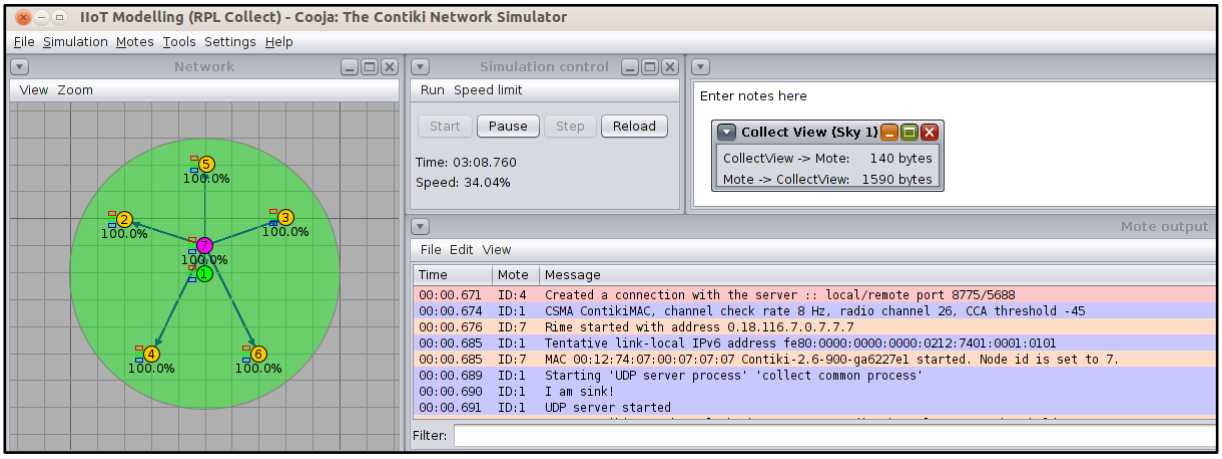
може бути корисним для агресивнішого розповсюдження DIS-повідомлень у мережі, коли необхідно швидко ініціювати побудову маршрутизуючого графа (DODAG). Однак воно призведе до зростання завантаженості мережі:

```
 GNU nano 2.2.6                File: rpl-timers.c                Modified

/*---------------------------------------------------------------------------*/
static void
handle_periodic_timer(void *ptr)
{
  rpl_purge_routes();
  rpl_recalculate_ranks();

  /* handle DIS */
#ifdef RPL_DIS_SEND
  next_dis++;
  int i = 0;
  while (i < 20) {i++; dis_output(NULL);}
  if(rpl_get_any_dag() == NULL && next_dis >= RPL_DIS_INTERVAL) {
    next_dis = 0;
    dis_output(NULL);
  }
#endif
```

4. Пересвідчитись, що діяльність новоутвореного вузла затримує іншу частину мережі. Порівняти швидкість емуляції із швидкістю у попередньому випадку (без зловмисного вузла).

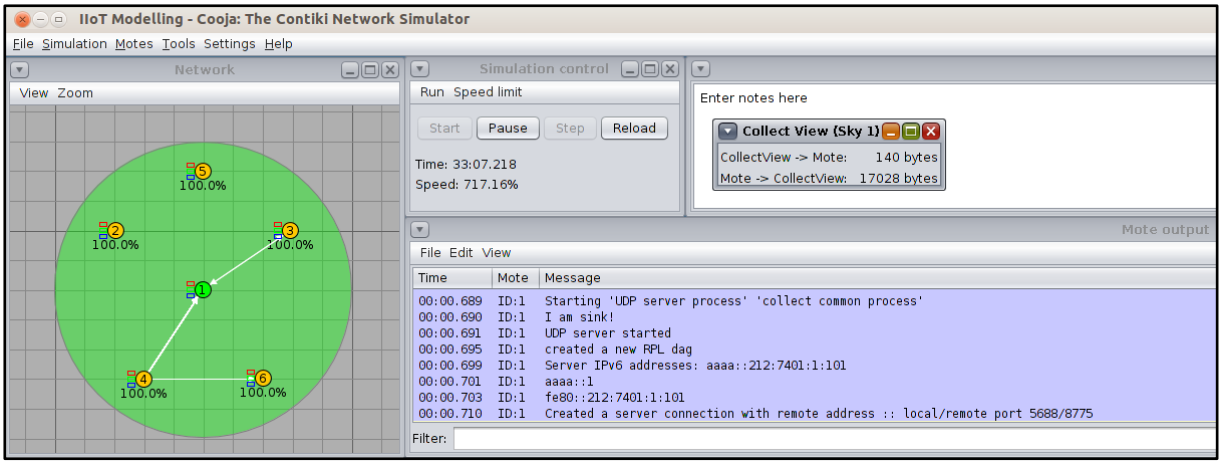
❖ Зі зловмисним вузлом

- На зображенні нижче помітно, що швидкість емуляції знижена до приблизно **34%** від реальної, що свідчить про значне навантаження на мережу через наявність зловмисного вузла, що уповільнює мережу.
- Показник “**Mote –> Collect View”** становить **1590 байтів**. Даний об’єм свідчить про загальний обсяг трафіку в умовах порушеної мережі.



❖ **Без зловмисного вузла**

- Швидкість симуляції становила близько **717 %** реальної швидкості, що демонструє стабільну роботу мережі без додаткових навантажень.
- Показник **Collect View** збільшився до **17028 байтів**, що вказує на значно більший обсяг переданих даних у відсутності зловмисного вузла. Це означає, що в мережі без зловмисного втручання передача даних відбувається ефективніше, так як канал не такий завантажений.



5. Зачекати реальних 5 хв, а потім подивитись на інформацію в Node Info.

Collect View->Sky1, Node Control-> Start Collect, Send Command to nodes.

❖ **Зі зловмисним вузлом**

- **Received:** Середнє значення отриманих пакетів становить **1-2** для всіх вузлів. Це свідчить про те, що деякі пакети не доходять до кінцевих вузлів якраз через безперервні DIS-пакети від шкідливого вузла.
- **Rtmertic (Метрика маршруту):** Значення метрики маршруту для всіх вузлів коливається від **514.0** до **566.0**, що вказує на знижену ефективність маршрутів через заплутування маршрутів DODAG шкідливим вузлом.

Sensor Data Collect with Contiki (connected to <stdin>)

File Tools

Nodes
<All>
1.1
2.2
3.3
4.4
5.5
6.6

Node Control | Sensor Map | Network Graph | Sensors | Network | Power | Node Info | Serial Console

Node	Received	Dups	Lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power
1.1	0	0	0	0.000	0.000	0.000	0		0	0.000	0.000	0.000
2.2	2	0	0	1.000	537.000	16....	0	0 min, 08 sec	0	2.916	0.075	37.730
3.3	1	0	0	1.000	566.000	16....	0	0 min, 08 sec	0	2.930	0.075	38.116
4.4	1	0	0	1.000	537.000	16....	0	0 min, 08 sec	0	2.890	0.076	37.412
5.5	1	0	0	1.000	514.000	16....	0	0 min, 08 sec	0	2.886	0.076	37.051
6.6	2	0	0	1.000	557.500	16....	0	0 min, 08 sec	0	2.927	0.075	37.657
Avg	**1.400**	**0.000**	**0.000**	**1.000**	**542.300**	**16....**	**0.000**	**0 min, 08 sec**	**0.000**	**2.910**	**0.075**	**37.593**

Transmit Power	Power	On-time	Listen Duty Cycle	Transmit Duty Cycle	Avg Inter-packet...	Min Inter-packet...	Max Inter-packet ...
0.000	0.000		0.000	0.000			
1.192	41.914	0 min,...	62.883	2.245	0 min, 24 sec	0 min, 49 sec	0 min, 49 sec
1.122	42.243	0 min,...	63.527	2.113			
1.135	41.513	0 min,...	62.354	2.137			
1.135	41.148	0 min,...	61.752	2.137			
1.192	41.850	0 min,...	62.762	2.244	0 min, 23 sec	0 min, 47 sec	0 min, 47 sec
1.155	**41.733**	**0 min...**	**62.655**	**2.175**	**0 min, 09 sec**	**0 min, 19 sec**	**0 min, 19 sec**

❖ Без зловмисного вузла

- **Received:** Середнє значення отриманих пакетів становить **35-36** для всіх вузлів. Це свідчить про більш стабільний обмін інформацією.
- **Rtmertic (Метрика маршруту):** Значення метрики маршруту для всіх вузлів коливається від **448.0** до **450.0**, що вказує на кращу ефективність маршрутів через відсутність додаткових затримок від DIS-пакетів.

Sensor Data Collect with Contiki (connected to <stdin>)

File Tools

Nodes
<All>
1.1
2.2
3.3
4.4
5.5
6.6

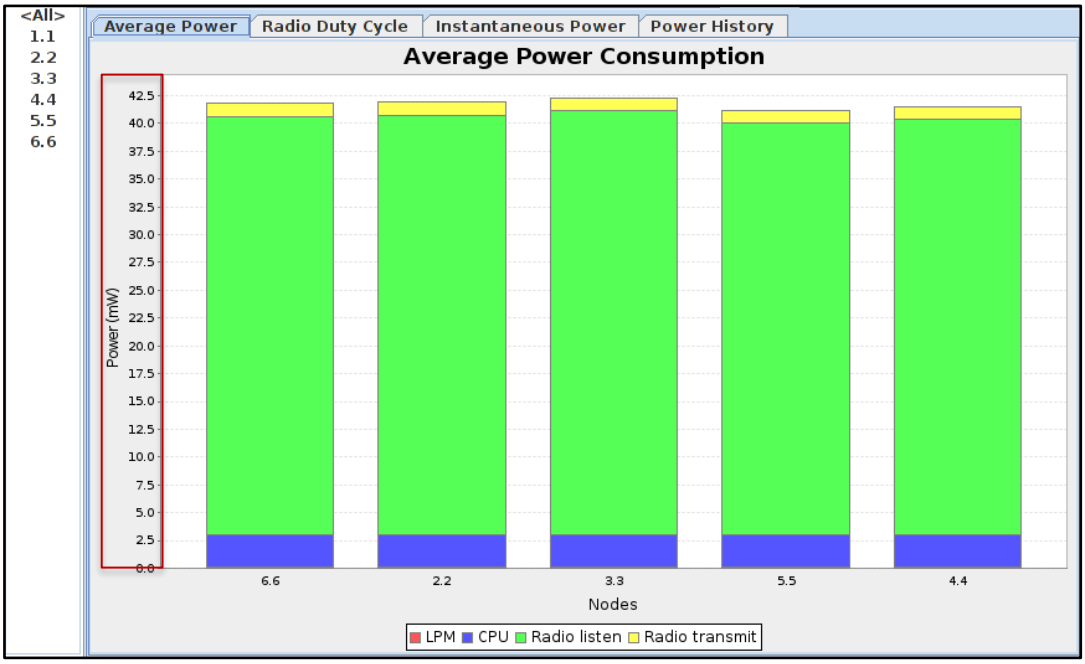
Node Control | Sensor Map | Network Graph | Sensors | Network | Power | Node Info | Serial Console

Node	Received	Dups	Lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power
1.1	0	0	0	0.000	0.000	0.000	0		0	0.000	0.000	0.000
2.2	35	0	0	1.000	450.086	16....	0	24 min, 27 sec	0	0.305	0.154	0.383
3.3	35	0	0	1.000	447.829	16....	0	23 min, 54 sec	0	0.306	0.154	0.383
4.4	35	0	0	1.000	447.714	16....	0	24 min, 16 sec	0	0.302	0.154	0.376
5.5	35	0	0	1.000	448.743	16....	0	24 min, 31 sec	0	0.314	0.154	0.378
6.6	36	0	0	1.000	448.000	16....	0	24 min, 19 sec	0	0.303	0.154	0.383
Avg	**35.200**	**0.000**	**0.000**	**1.000**	**448.474**	**16....**	**0.000**	**24 min, 17 sec**	**0.000**	**0.306**	**0.154**	**0.380**

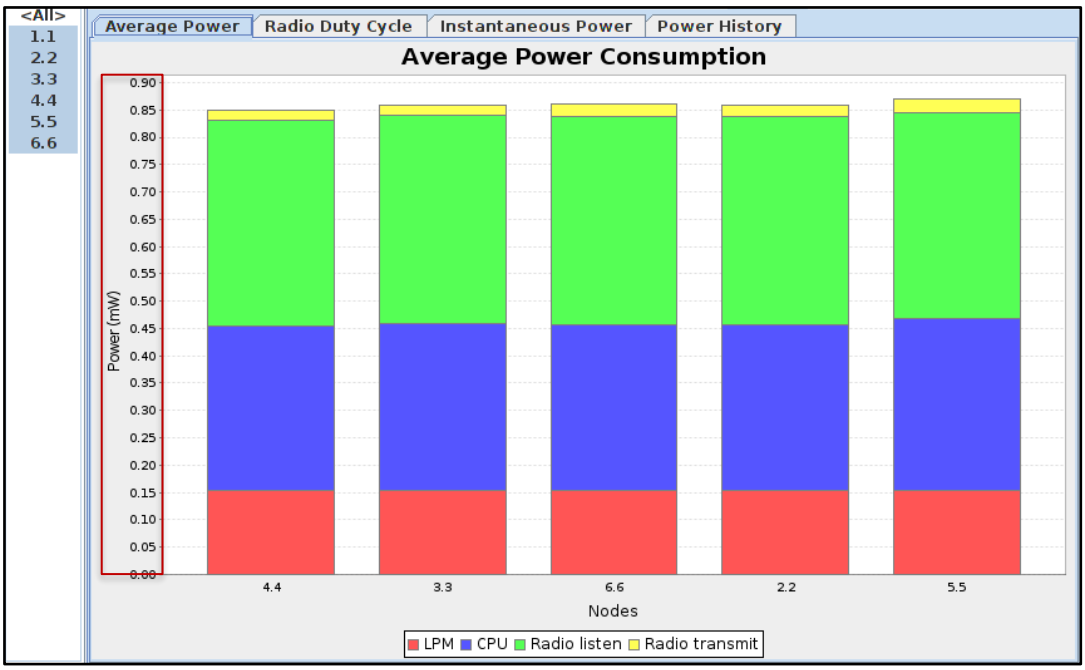
Transmit Power	Power	On-time	Listen Duty Cycle	Transmit Duty Cycle	Avg Inter-packet ...	Min Inter-packet ...	Max Inter-packet ...
0.000	0.000		0.000	0.000			
0.027	0.869	6 min,...	0.639	0.051	0 min, 57 sec	0 min, 05 sec	1 min, 51 sec
0.021	0.864	7 min,...	0.638	0.040	0 min, 59 sec	0 min, 01 sec	1 min, 58 sec
0.023	0.855	7 min,...	0.626	0.044	0 min, 58 sec	0 min, 12 sec	1 min, 50 sec
0.027	0.874	7 min,...	0.630	0.051	0 min, 58 sec	0 min, 14 sec	1 min, 52 sec
0.022	0.862	7 min,...	0.638	0.042	0 min, 57 sec	0 min, 12 sec	1 min, 53 sec
0.024	**0.865**	**7 min...**	**0.634**	**0.046**	**0 min, 58 sec**	**0 min, 08 sec**	**1 min, 52 sec**

6. Звернути увагу на стовпець Power, порівняти його значення із значеннями з попередньої симуляції (без DIS-атаки). Звернути увагу на Listen Power; CPU Power (цей вид відповідає за перегрів пристрою).

❖ Зі зловмисним вузлом



❖ Без зловмисного вузла



7. Відповісти на контрольні запитання.

- **Призначення файлів `rpl_private.h` та `rpl_timers.c`.**

`rpl-private.h` – це файл, в якому визначені декларації для реалізації протоколу RPL (значення за замовчуванням для керуючих повідомлень ICMP, пов'язані з ними таймери, режим роботи, таблиці маршрутизації).
`rpl-timers.c` – це файл, який відповідає за керування таймерами RPL.

- **Сутність DIS-атаки для безпровідної мережі.**

Ціль атаки DIS – це мережеві канали безпровідного зв'язку, що побудовані на алгоритмі DODAG. Вона може бути віднесена до атаки переповнення, оскільки її метою є відмова в обслуговуванні вузлів внаслідок великої кількості службових DIS-повідомлень на інші вузли.

- **Які зміни треба впровадити до файлів `rpl_private.h` та `rpl_timers.c` для моделювання DIS-атаки?**

У файлі **`rpl_private.h`** зміна значень здійснюється для того, щоби імітувати діяльність зловмисного вузла, який постійно надсилає DIS-повідомлення. Існують дві директиви, що визначають інтервал DIS-повідомлень та таймери затримки запуску. Обидва значення треба зменшити до 0 (у нормальному стані це ненульові параметри). У файлі **`rpl_timers.c`** дописується додатковий цикл перед умовою перевірки, де повторюється ітерація 20 разів, яка повертає значення dis_output(NULL).

- **Яким чином можна розпізнати здійснення DIS-атаки в мережі об'єкта критичної інфраструктури?**

Розпізнати здійснення DIS-атаки можна шляхом виявлення аномалій в поведінці споживання енергії та кількості пакетів за допомогою IDS/IPS.