CS4222/CS5422

PROGRAMMING ASSIGNMENT #4

AY 2020-2021 / SEM 2

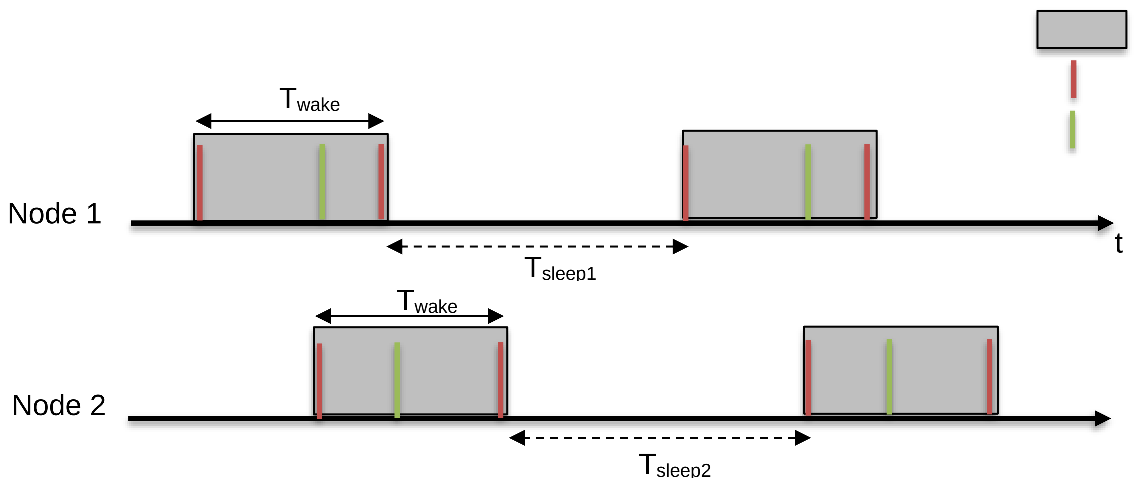
DUE: March 29 (Monday) 23:59

1. ***This homework is to be completed by each project group.***
2. ***Total marks: 40.***
3. ***This assignment carries 10% weightage to your final grade.***
4. ***There is a 10% penalty per-day for late submission.***
5. ***For any clarification on this assignment, email Ebram Kamal William (ebramkw@comp.nus.edu.sg) or the lecturer (chanmc@comp.nus.edu.sg)***

1. Overview

In this homework, you will learn more about neighbor's discovery. Also, you will study different settings and how it affects the power consumption and the efficiency of the neighbor's discovery.

You are provided with code that implement a basic "birthday protocol" whereby each node randomly wakeup to transmit and listen for transmissions from nearby devices. The operations are illustrated below.



The same code is run on all nodes. A node wakes up for a period of T_{wake}. When a node wakes up, it sends two packets: one at the beginning of the wakeup time frame and the other one at the end before going to sleep.

After the first transmission, the node will keep its radio ON to listen for possible receptions. The figure above shows the settings where node 1 and node 2 follow their own timers. The interval between two wakeup slots (T_{sleep1} and T_{sleep2} in figure above) is chosen randomly from a uniform distribution with the same mean T_{sleep}.

Changing T_{wake} and T_{sleep} will affect the latency a node takes to discover their neighbors and the (radio) energy consumption.

The duty cycle of the radio is given as T_{wake} / (T_{wake} + T_{sleep})

2. Programs and Parameters Settings

You are given a C program “nbr_discovery.c” which implements the basic logic described in the section above. The parameters which you can change in the program are

WAKE_TIME (T_{wake}): The default value is (RTIMER_SECOND / 10) or 100ms.

The maximum length of a single sleep interval is limited by the duration in which the RTIMER count wraparound. Hence, instead of a single sleep, the total sleep interval between two wakeup intervals is the product of the duration of a single sleep (SLEEP_SLOT) and the number of sleep cycles (SLEEP_CYCLE).

SLEEP_SLOT: The default value is the same as WAKE_TIME.

SLEEP_CYCLE: The average number of sleep cycles. The default value is 9.

Duty cycle is given as WAKE_TIME / (WAKE_TIME + SLEEP_CYCLE*SLEEP_SLOT)

In this assignment, you will run experiments in two different environments. The first one is on CC2650 SensorTags and the second environment is on COOJA simulator using sky mote.

Running on CC2650 allows you to evaluate the behavior on an actual platform, while running on Cooja allows you to perform power measurements of your code execution in a simulation environment.

3. Running on CC2650

Compile and execute the neighbor discovery code by doing the following:

- Place the given folder named "***nbr_discovery***" under contiki/examples/.
- Compile the nbr_discovery program using command "`make TARGET=srf06-cc26xx BOARD=sensortag/cc2650 nbr_discovery.bin CPU_FAMILY=cc26xx`" in the directory "***contiki/examples/nbr_discovery***".
- Use uniflash program to burn the binary file to the SensorTag.
- Observe the output of the program through the USB serial port.

You need to run the program on at least 2 devices to perform the experiment. Consider two devices A and B.

3.1 Task 1

1. Using the default settings, observe and record how long the devices take to discover each other. Pick one of the devices as A and plot the cumulative distribution of the intervals between packet receptions on device A hearing from device B.
2. Reset device B and observe how long it takes for device A to hear from device B after device B reboots. You may need to modify the given code to observe this duration. Perform the experiments at least 10 times and plot the cumulative distribution.
3. Try out different settings and discuss your observations.

4. Running on COOJA

4.1 COOJA Simulator Setup

You will use the Cooja simulator in this part. Cooja is a wireless sensor networks simulator which comes with Contiki OS.

Running natively on Ubuntu

You can find it under "***contiki/tools/cooja***" and you will be able to run it using the command "***ant run***". You may need to setup JAVA environment for it to work.

<u>Running in VM</u>

If you are using VM/Ubuntu, performing the following:

- sudo add-apt-repository ppa:openjdk-r/ppa
- sudo apt-get update
- sudo apt-get install openjdk-8-jdk openjdk-8-jre ant libncurses5-dev

To be able to make for sky mote target, use the following command:

- sudo apt-get install binutils-msp430 gcc-msp430 msp430-libc msp430mcu mspdebug

4.2 Compile to run Sky Mote Simulation

You are given a Cooja simulator configuration file called "***cooja_nbr_discovery.csc***" which has the network setup and linked to the binary file of the nbr_discovery program. Compile the nbr_discovery program using command "***make TARGET=sky nbr_discovery.upload***" in the directory "***contiki/examples/nbr_discovery***".

4.3 Running COOJA

Run Cooja (using the command "***ant run" in the ~/contiki/tools/cooja directory***) and open the simulator file: "***File->Open_simulation***" and choose the "**nbr_discovery/*cooja*_nbr_discovery.*csc***" file.

You can save the output from the GUI, "***Mote output"*** window ➔ "***File->save to file***".

➢ If there is an error running Cooja, there may be an issue with the java environment. Run "sudo update-alternatives --config java" and choose to use "java-8-*".

4.4 Program Output

The given program logs the following information:

- Tx/Rx: logs whenever it sends or receives with the sequence number and the timestamp.
- Power trace every minute: Powertrace tracks the duration of activities of a node being in each power state. In other words, the outputs show the fraction of time that a node remains in a particular power state. The output is as the following:
  Sample output:

```
2408 P 193.250 17 11309 578542 0 2574 0 2574 689 32078 0 144 0 144
(radio 0.43% / 0.43% tx 0.00% / 0.00% listen 0.43% / 0.43%)
```

Output Description

(1)	2408	clock time
(2)	193.250	rime address
(3)	17	sequence number
(4)	11309	accumulated CPU energy consumption
(5)	578542	accumulated Low Power Mode energy consumption
(6)	0	accumulated transmission energy consumption
(7)	2574	accumulated listen energy consumption
(8)	0	accumulated idle transmission energy consumption
(9)	2574	accumulated idle listen energy consumption
(10)	689	CPU energy consumption for this cycle
(11)	32078	LPM energy consumption for this cycle
(12)	0	transmission energy consumption for this cycle
(13)	144	listen energy consumption for this cycle
(14)	0	idle transmission energy consumption for this cycle
(15)	144	Idle listen energy consumption for this cycle.

Between the brackets is the percentage of time spent on each state of the radio:

(1)	Radio 0.43% / 0.43%	cumulative / percentage of the last cycle.
(2)	tx 0.00% / 0.00%	cumulative / percentage of the last cycle
(3)	listen 0.43% / 0.43%)	cumulative / percentage of the last cycle

The radio value is the sum of the second and third (tx and listen) values.

You can edit the program to vary the output information to suit your needs.

4.5 <u>Task 2 (Power Measurement)</u>

Show the powertrace log when WAKE_TIME is set to 50ms, SLEEP_CYCLE to 4 and SLEEP_SLOT to 200ms.

4.6 <u>Task 3 (Deterministic Discovery)</u>

Modify the code so that two-way discovery (A receives B AND B receives from A) can be completed in a deterministic manner within 10 seconds. You should choose settings so that the radio power consumption is "minimized".

In your submission, you must include the following:

- the algorithm you have implemented
- the parameters chosen
- the maximum two-way latency observed
- the radio duty cycle achieved

5 Submission

Your submission consists of a single zip file (GroupNumber-hw4.zip) containing the code you have written for Task 3 and a single pdf file with the name GroupNumber-hw4.pdf to the LumiNUS folder assignment4-submission with the following:

a) Solution to Task 1
b) Solution to Task 2
c) Solution to Task 3

6 Grading

The grading weightage are as follow:

1) 5% on following the submission instructions.
2) Task 1 – 20%
3) Task 2 – 15%
4) Task 3 – 60%