Mobile Computing for IEEE 802 Wireless Networks

Assignment 04:
Practical experience with free-space optical networks

Important

- This is assignment 4:
  Practical experience with free-space optical networks
- Submit your results via e-mail: **schmist@inf.ethz.ch**
- Submission deadline: **Monday, 2014-Nov-24, 11:59pm**

- Start early enough with this assignment.
- This assignment can be done in groups of 2 or 3 (not more).
- Do not flash the Arduino's firmware with a different firmware, you will not be able to complete the assignment anymore.

Motivation

- Learn about free-space optical communication.
- Understand LED-to-LED communication.
  - Why and how can an LED be used as a receiver?
  - What are the limitations of LEDs as a sender/receiver?
    - What are the communication ranges (distances, angles)?
    - What is the performance (throughput, delay, robustness)?
- Learn how to conduct experiments with real hardware.
  - How to setup a testbed.
  - How to build tools/scripts to support measurements/evaluation.
  - How to evaluate and present the results.
  - How to apply statistics to evaluation results.

Step 1: Setup

- Every student receives 1 Arduino Uno, 1 USB cable and 1 red LED. (Please return everything after finishing with the assignment.)
- The LED can be used as transmitter and receiver.
  (Connect the cathode to A0 and the anode to A1, the LED should be enabled if the Arduino is powered through USB.)
- Build groups of two or three students. (To be able to run communication experiments.)
- The Arduino's microcontroller is preloaded with the necessary firmware to run LED-to-LED communication experiments. (Please do not flash it with a different firmware.)

Step 2: Theory

- Read through the Visible Light Communication lecture slides and if needed additional material:
  - P. Dietz, W. Yerazunis, and D. Leigh, “Very Low-Cost Sensing and Communication Using Bidirectional LEDs”
  - S. Schmid, G. Corbellini, S. Mangold, and T. Gross, “An LED-to-LED Visible Light Communication system with software-based synchronization”
  - S. Schmid, G. Corbellini, S. Mangold, and T. Gross, “LED-to-LED Visible Light Communication Networks”
- Answer the following questions with a few sentences:
  - How can an LED be used as a receiver?
  - Why is synchronization necessary and how is it achieved?
  - What is the benefit from using an LED as a receiver instead of a photodiode?

Step 3: Arduino / PC communication

- The serial to USB interface can be used to send commands and data to the Arduino and receive from it. The Arduino should be recognized on Linux and OSX as a serial com port. On Windows you have to follow these instructions to install the driver: http://arduino.cc/en/Guide/Windows#toc4
  You will then find a new serial device available on your machine with the following port name:
  - Windows: COMXX (check the device manager for the port number)
  - Linux | OSX: /dev/ttyUSBXX | /dev/tty.usbserial-XX (use ls to find the name of the port)
- Write a tool in the script/programming language of your choice to communicate with the Arduino. Use and extend this tool or create multiple scripts to run the experiments described in the later steps.
- Use 115200 as baudrate, other serial settings are default values, for most libraries you do not have to configure more.

Step 3: List of available serial libraries

- Matlab
  http://www.mathworks.ch/ch/help/instrument/serial-port-instrument-communication.html
- Java with RXTX or jSSC
  http://rxtx.qbang.org/wiki/index.php/Main_Page
  https://code.google.com/p/java-simple-serial-connector/
- Python with pySerial
  http://pyserial.sourceforge.net/
- Node.js with node-serialport
  https://github.com/voodootikigod/node-serialport

- These are just a few examples, you are free to use whatever you want as long as it works!

Step 3: Serial communication protocol (I)

- Commands (1 byte)
  - 'r' : resets the device
  - 'p' : **returns** the name of the firmware (vlc4_mobicom)
  - 'a' : **returns** the device's address in hex (e.g. AB for 0xAB)
  - 'd' : disables communication
  - 'e' : enables communication
  - 'c' : configures the VLC MAC parameters, followed by the configuration string terminated by '\0': "RETRANS DIFS CWMIN CWMAX\0"
    RETRANS: number of retransmission (1 byte)
    DIFS: inter-frame space [ms] (2 byte)
    CWMIN: contention window min [ms] (2 byte, power of 2)
    CWMAX: contention window max [ms] (2 byte, power of 2)
    returns the configuration string separated by commas
  - 'm' : sends a message, followed by the destination address (as 1 byte) and the message as byte stream terminated with '\0', **returns** "STATUS,DEST"
    STATUS: 0: packet dropped, 1: packet queued
    DEST: destination address in hex

Step 3: Serial communication protocol (II)

- Command return values (serial input from the device):
  - JSON (employ a JSON library for the language you use) object as string terminated by '\0'.
  - Contains up to three elements:
    - "c": the command it returns
    - "d": return value
    - "s": statistics
  - Example 1: writing 'a' to the device with address 0xB1 will return {"c":97,"d":"B1"}, 97 is the integer value of 'a' .
  - Example 2: writing 'p' to the device will return {"c":112,"d":"vlc4_mobicomp"}.
- Serial input from the device without preceding commands:
  - c=1: message received, d: payload, s: receive statistics
  - c=2: message sent, but still in queue, waiting for ACK/possible retransmissions, s: transmission statistics
  - c=3: message done, queue ready for new message
  - Example: receiving a message looks as follows: {"c":1,"d":"message","s":"statistics"}

Step 3: Serial communication protocol (III)

- Receiving / transmitting statistics format:
  - "MODE,TYPE,SRC->DEST,SIZE,SEQ,CW,CWSIZE,TIME"
    - MODE: S: sent, R: received
    - TYPE: D: data, A: ACK, DR: data retransmission
    - SRC: source address (hex)
    - DEST: destination address (hex)
    - SIZE: frame size
    - SEQ: frame sequence number
    - CW: randomly chosen contention window value
    - CWSIZE: current contention window size
    - TIME: time in milliseconds since the last device reset
  - Example: "S,D,AB->CD,5,123,9,16,3459" or "R,DR,CD->AB,5,124,0,0,4356"
  - There is no contention parameter statistic for ACKs and received frames and the missing values are therefore replaced by zeros.

Step 3: Serial communication protocol (IV)

- Serial communication example using Python

```
s = serial.Serial('COM4',115200,timeout=1) #opens a serial port (resets the device!)
time.sleep(2) #give the device some time to startup everything (2 seconds)

#write to the device
s.write('c') #write 'c' to start configuring the device
s.write("3 10 4 16\0") #write the configuration string
time.sleep(0.1) #wait for settings to be applied (or wait for the return command)
s.write('m') #initiate message transmission
s.write('\xAB') #write destination address as a byte value
s.write("hello world!\0") #write message terminated with '\0'

#read from the device (best done in a separate thread)
while True:
  message = ""
  while True: #while not terminated
    try:
      byte = s.read(1) #read one byte (blocks until data available or timeout reached)
      if byte=='\0': #break if termination character reached
        break
      message = message + byte #concatinate the message
    except serial.serialutil.SerialException:
      continue #on timeout try to read again
    except KeyboardInterrupt:
      sys.exit() #on ctrl-c terminate program
  print message #print the message (JSON string)
```

Step 4: Chat application

- To familiarize yourself with the devices and to try out your serial communication tools, write a small chat application to send messages between machines using the Arduino VLC devices as sender and transmitter.
- Send the messages to the broadcast address (0xFF) so that they can be received by multiple devices (if you have a network of three devices)
- The application does not need to have a GUI, CLI is fine.
- Provide some screenshots and the application source code.

Step 5: Distance measurement

- Use one device as a sender and one as a receiver, connected to two different machines or the same machine.
- Packets sent to a device's address are acknowledged and if no ACK is received repeated until the defined number of retransmission is reached.
- Packets sent to the broadcast address (0xFF) are received from all devices and not acknowledged or retransmitted.
- At the sender, generate saturation traffic: send a new message immediately after you receive the message done command (c=3) over serial.
- Try to point the LEDs directly at each other (the beams ar tight) to communicate over longer distances.

Step 5: Distance measurement experiments

- Run the same experiment for a packet payload size of 1, 10, 50, 100, 150, 200 byte (200 byte is the maximum possible payload).
- Start with a very small distance (LEDs almost touching) and gradually increase the distance to find the maximal communication distance. Use at least 5 additional intermediate steps. (6 [packet size] x 7 [distances] experiments.
- Run every experiment long enough to collect enough data to provide also statistical infromation for your evaluation (standard deviation, confidence interval, see the document on the course website).
- Evaluate data throughput and packet delay. As delay use the roundtrip delay (the reception of the ACK) which you can evaluate on the same device with its local time.
- Provide data logs, graphs and describe your findings with a few sentences.

Step 6: MAC parameters

- RETRANS: Retry sending a frame for at least this number additional times.
- DIFS: Wait at least for DIFS before transmitting the next packet.
- CWMIN/CWMAX: Contention window minimum/maximum, on an ACK timeout, the current contention window is doubled, starting from CWMIN upto CWMAX. The backoff time is computed as a random number between 0 and the current contention window.
- Run experiments at a reasonable distance for different DIFS and CWMIN values. Groups with 3 devices (3 students) can try to cause collisions by using small CWMIN values.
- Evaluate throughput and delay and also provide statistical information for your measurements.
- Provide data logs, graphs and describe your findings with a few sentences.

Expected results

- Please submit the following results:
  - Name and Legi number.
  - Step 2: Written answers to the theory questions.
  - Step 3: Source code of your communication scripts/tools.
  - Step 4: Screenshots and source code of the chat application.
  - Step 5 / Step 6:
    - Experiment script/tools
    - Experiment logs
    - Throughput and delay plots with statistical information
    - Interpretation

- Send your report to: **schmist@inf.ethz.ch**

Assignment 04:
Practical experience with free-space optical networks
– END OF ASSIGNMENT –