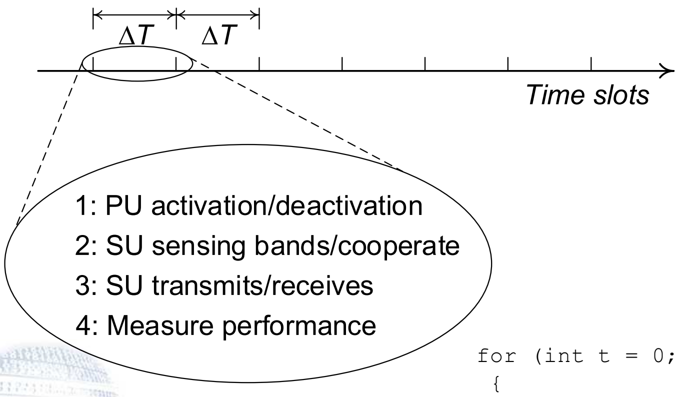
Project I: Simulation Framework for CRNs v1.0

Dr. Mohammed Hawa
Electrical Engineering Department
The University of Jordan

Project I

- Build a simulation framework to test cognitive radio networks (including cooperative sensing).
- The framework allows SUs to perform sensing, exchange sensing information, perform spectrum allocation, then measure the resulting performance.
- Use C/C++ and MS Visual Studio 2017.
- Include a probabilistic method implementation and results (see below).
- LATER: Your proposed algorithm: can be sensing, spectrum allocation, energy harvesting, etc.
- Show plots for the various performance parameters.

Assume Slotted System




```
for (int t = 0; t < SLOTS; t++)
 {
 .
 .
 .
 }
```


Simulation Parameters

- Number of secondary users (SUs) $N = 10$
- Number of spectrum bands $M = 100$
- One primary user per band.
- So, number of primary users (PUs) $= M$
- Number of time slots $T = 20{,}000$
- Demand $\hat{S}_n = 5 - 15$
- So, Load $L = 50$– 150%
- Later we can vary the load or change the parameters.

Spectrum Sensing

- PU is activated/deactivated in its band randomly with probability 0.1/0.9, or in a deterministic way at t_{active}.
- Sensing is not perfect. Assume:
- Probability of **false alarm:**

$$P_{FA} = \Pr[\Gamma > \gamma \mid \mathcal{H}_0] = 0.1$$

- Probability of **misdetection:**

$$P_{MD} = \Pr[\Gamma \leq \gamma \mid \mathcal{H}_1] = 0.1$$

Spectrum Allocation

- Staring point: probabilistic spectrum assignment.
- Each SU[n] selects $|\hat{S}_n|$ randomly (with equal probability) of the M available spectrum bands for the current time slot.
- No dependence on sensing (no intelligence).
- Sensing here is monitoring (detecting collisions that happen after transmitting data).
- There is no coordination with other SUs.
- SU[n] can only hope it was lucky that no one else chose the same bands it has selected (causing a collision).

Using C++ Classes

```
class Band_Details
{
public:
        ...

        bool PUState; // true means PU is ON
                      // false means PU is OFF
                      // in current time slot

        double ProbON; // ProbOFF = 1 - ProbON

        std::vector <int> Occupants;
        // IDs of the SUs who are using this band

        ...
};
```


Abstracting to Organize Thoughts

```
class SecondaryUser
{
public:
        ...

        double Pfa; // probability of false alarm
        double Pmd; // probability of misdetection

        std::vector <unsigned int> BandBeingSensed;

        std::vector <unsigned int> SuccessfulBandsPerSlot;

        ...
};
```

Suggestion

- Use `std::vector`.
- Easier memory management than arrays or two-dimensional arrays. For example:
- `vector <Band_Details > PU;`
- `vector <SecondaryUser > SU;`
- Each SU records it allocated bands:
- `SU[0] = {1, 25,` **`5`**`, 88, 17}`
- `SU[3] = {40,` **`5`**`, 65, 14, 79}`

Flipping Coins

```
rand()
// integer value between [0, RAND_MAX]

double RandomNumber =
       double(rand()) / double(RAND_MAX);
       // uniform value between [0.0, 1.0]

if (RandomNumber <= 0.7)
  do something; // probability 0.7
else
  do something else; // probability 0.3
```


Performance Parameters

- SU sensing results: false alarm count $f_n(t)$ and misdetection count $d_n(t)$
- SU successful transmission: $s_n(t)$
- SU contention band count: $c_n(t)$
- PU interference ratio: ψ_m
- Utilization: $U_m(t)$
- Throughput: $Y_m(t)$
- Need to write results to CSV file then read the data in MATLAB for plotting a graph (MATLAB `plot`).

CSV file from C++ to MATLAB

```
1,1,1,0,0,0,8,8,8,8,8,8,8,0,0,0,0,0,0,0,0,0,0
1,1,0,1,0,1,1,0,1,0,1,0,0,1,8,8,8,8,8,8,8,0,0
1,2,1,1,0,6,0,1,0,1,7,1,1,1,0,0,1,1,1,1,1,1,2
```

```
>> A = csvread('test.csv');
>> A

A =

  Columns 1 through 12

     1     1     1     0     0     0     8     8     8     8     8     8
     1     1     0     1     0     1     1     0     1     0     1     0
     1     2     1     1     0     6     0     1     0     1     7     1

  Columns 13 through 23

     8     0     0     0     0     0     0     0     0     0     0
     0     1     8     8     8     8     8     8     8     0     0
     1     1     0     0     1     1     1     1     1     1     2
```

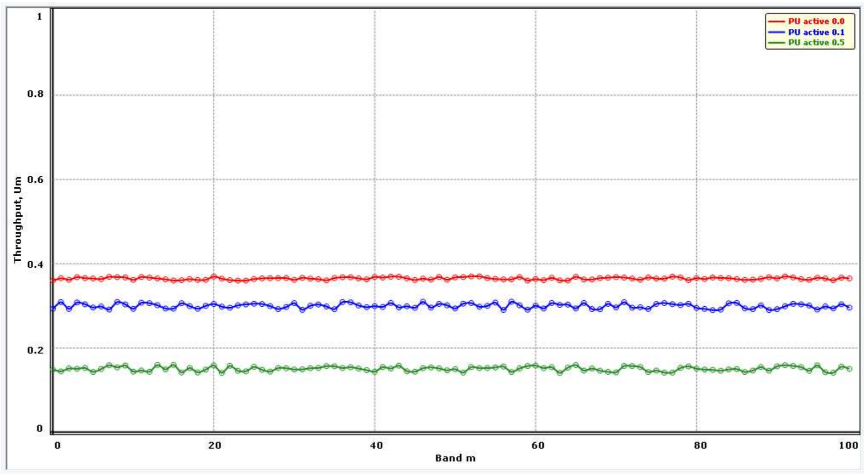

Submission

- Each team will submit the code as a team.
- I will test the code and then ask the team members about the code decisions.
- Team members should meet regularly, and distribute the load amongst themselves.
- No inter-Team cooperation WHATSOEVER!
- Similar solutions with receive zero credit.

Throughput

- $Y_m(t) = 1$ for band m in time slot t if band is successfully utilized by only one SU, and 0 if band is empty, used by many SUs or used by both SU(s) and PU.
- $Y_m = \frac{\sum_{t=1}^{T} Y_m(t)}{T}$
- Throughput Y_m for each band averaged over all time slots Versus band number.
- Three curves: For different PU activation probability = 0.0, 0.15, 0.25
- Use different colors and use a Legend.

Not necessarily actual results…



Utilization

- $U_m(t) = 1$ for band m in time slot t if band is used by one (or more) SUs, or 0 if band is empty or has only a PU.
- $U_m = \frac{\sum_{t=1}^{T} U_m(t)}{T}$
- Utilization U_m for each band averaged over all time slots Versus band number.
- Three curves: For different PU activation probability = 0.0, 0.15, 0.25
- Use different colors and use a Legend.

SU Successful Transmission

- $s_n(t)$ = number of bands successfully acquired by only SU n during time slot t without collisions from other SUs or PUs.
- $S_n = \frac{\sum_{t=1}^{T} s_n(t)}{T}$
- Successfully acquired bands S_n for each SU averaged over all time slots Versus SU number.
- Three curves: For different PU activation probability = 0.0, 0.15, 0.25
- Use different colors and use a Legend.

SU Collisions

- $c_n(t)$ = number of bands acquired by SU n during time slot t that suffer collision from SUs or PUs.
- $C_n = \frac{\sum_{t=1}^{T} c_n(t)}{T}$
- Collision bands C_n for each SU averaged over all time slots Versus SU number.
- Three curves: For different PU activation probability = 0.0, 0.15, 0.25
- Use different colors and use a Legend.

PU Interference Ratio

- Interference Ratio ψ_m for each PU (band) Versus PU (band) number.
- Three curves: For different PU activation probability = 0.0, 0.15, 0.25
- Use different colors and use a Legend.
- Say you have a total of 20,000 time slots, during which PU m is active for 8,000 slots. If PU m suffers interference (from one or more SUs) during 100 slots, then ψ_m= 100/8000 = 0.0125

Successful versus Time

- Successfully acquired bands $s_4(t)$ for SU 4 Versus time slots.
- One curve: Deterministic PU activation. Exactly in the middle of the simulation PU 0 to 49 become active. PU 50 to 99 stay inactive for the whole simulation.
- $s_n(t)$ = number of bands successfully acquired by only SU n during time slot t without collisions from other SUs or PUs.

Cooperative Sensing

- Earlier figures should be done using cooperative sensing: majority rule.
- Show results of $f_n = \frac{\sum_{t=1}^{T} f_n(t)}{T}$ and $d_n = \frac{\sum_{t=1}^{T} d_n(t)}{T}$
- Three curves: PU activation probability = 0.0, 0.15, 0.25
- Results for local sensing and cooperative sensing on the same graph.
- $f_n(t)$ = number of bands seen by SU n during time slot t with active PU, while in reality the PU was inactive.
- $d_n(t)$ = number of bands seen by SU n during time slot t with inactive PU, while in reality the PU was active.

Changing Load

- Earlier figures should be done using $\hat{S}_n = 10$
- Finally draw: Successfully acquired bands S (averaged over all time slots and all SUs) Versus $\hat{S}_n = 5, 8, 10, 12, 15$.
- $S = \frac{\sum_{t=1}^{T} \sum_{n=1}^{N} s_n(t)}{T \times N}$
- Three curves: For different PU activation probability = 0.0, 0.15, 0.25
- Use different colors and use a Legend.

Project I Submission

- Submit your source code (including project, cpp, h, … files) on a CD or flash drive.
- One project per team: on Thursday Nov 1, 2018.
- The code must compile on VS2017 without errors.
- The program should perform: PU activation, unreliable sensing, sensing cooperation, SU allocation, and performance measurement.
- Running the program should result in the above mentioned figures.

