# Welcome to ESE 356 Digital Systems Specification and Modeling

Fall Semester 2020

## Project Discussion 2-1

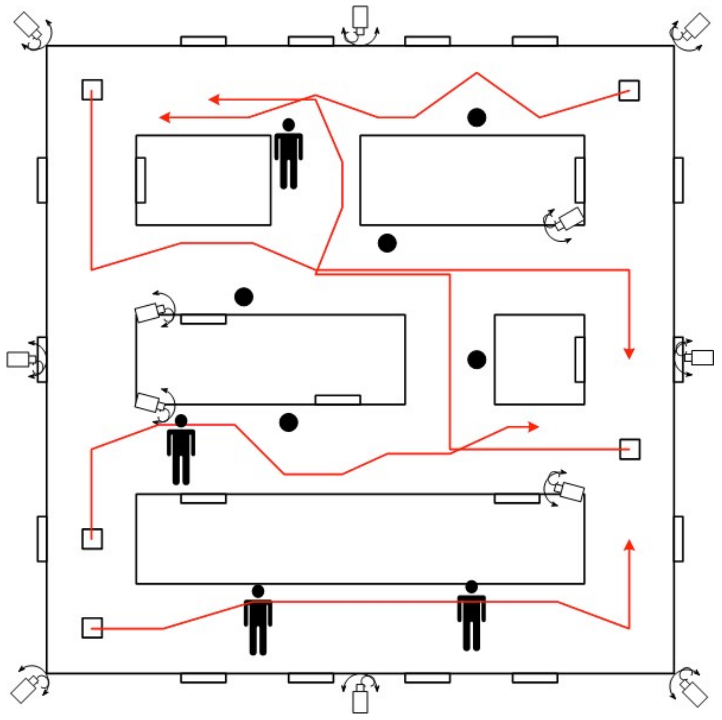
1. Project Overview and Phase 1 Description
2. Map Usage and Grid Based Navigation
3. Robot, Server Modeling and Communication
4. Processing Module for Sensor Processing
5. Obstacle Handling and Fixed Speed Navigation (Phase 1)
6. Simulation and Verification
7. Submission Requirement
8. Grading

# Project Overview

This project is to design a navigation and synchronization system for multiple robots.

Three key modules with message passing communication scheme are needed for the system

Robot Module
Server Module
Processing Module



## Project Overview

Multiple robot navigation control

- Modeling abstraction (as high as possible)
- Separate functional modeling of sensors from the physical robot
- Confined area map model and path sequence
- Speed control based on graph construction
- Grid-based synchronization

# Path and Map Representation

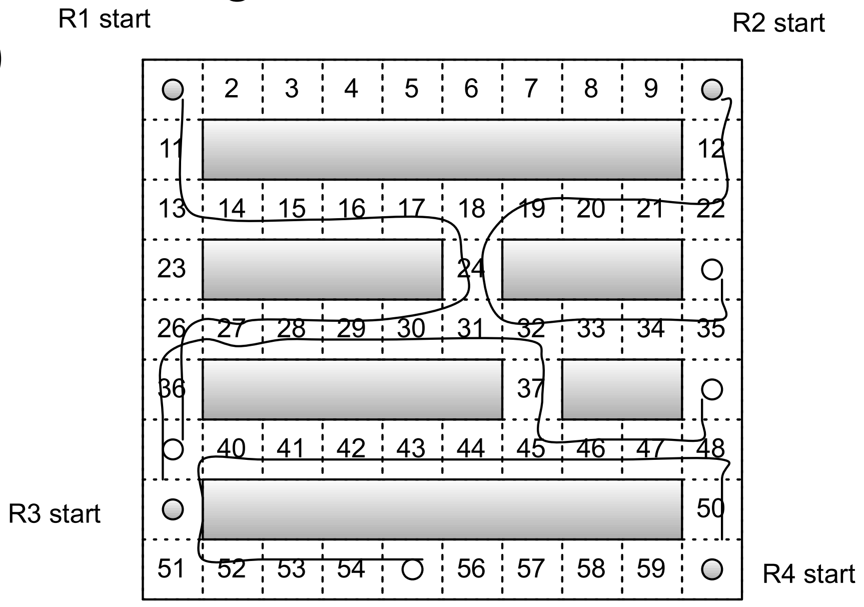
Path of a robot is specified by a sequence of grids.

Assume lower left as the origin (0, 0)

Each grid is 2m by 2m in dimension.

Grid position is represented by

(l,l) and (u.r).

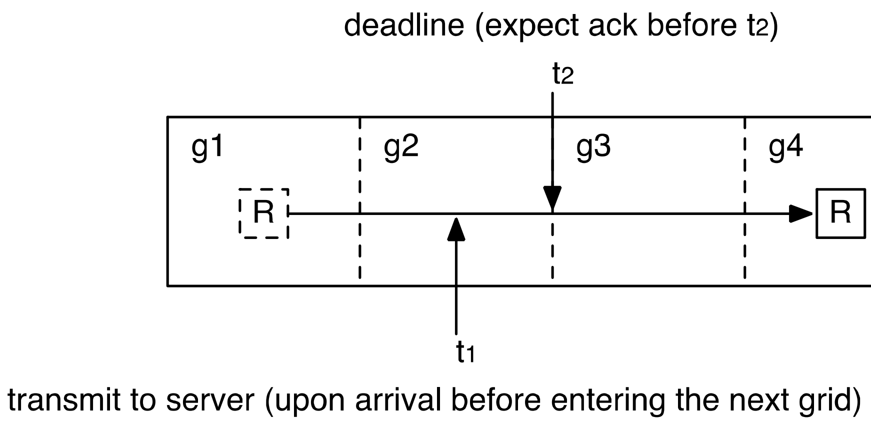


| GRID | LLX | LLY | URX | URY | N | S | W | E |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 16 | 2 | 18 | -1 | 11 | -1 | 2 |
| … | | | | | | | | |
| 17 | | | | | | | | |

# Grid Based Navigation

Navigation Principle:
- Only one robot is allowed within a grid.
- Obstacles can be in the same grid with the robot.

Navigate in right angle at the corner (arrive at the center of the corner grid then change the direction)


deadline (expect ack before t2)
t2
g1
g2
g3
g4
g5
R
R
t1
transmit to server (upon arrival before entering the next grid)

The robot stops when obstacle is within the proximity sensor range (3m)

## Main Model Overview

Three main models:
- Robots
- Processing
- Server

Robot modeling
- Separation of communication hardware from functional model of various sensors and hardware
- Proximity sensor, GPS sensors, motor mechanism operations are modeled in processing module
- Communication handshake and synchronization operations are handled by robot module

## Main Model Overview

Processing modeling

- Models the functionality of the robot (sensor interactions with the surrounding determination of the location, and operation of the motors)
- Obstacle modeling
- Interaction between multiple robot modules and a processing module
- Map information data structures

Server modeling

- Data structures for maintain the system status
- Dynamic status handling
- Multiple robot modules interaction
- Map information data structures

## Processing Module

Proximity sensor modeling - checking the distance between robots and all obstacles. Only the obstacles in front of robot are considered.

GPS and Robot motion modeling - maintains robot path information

Based on the path information, update positions of the robots using the map information.

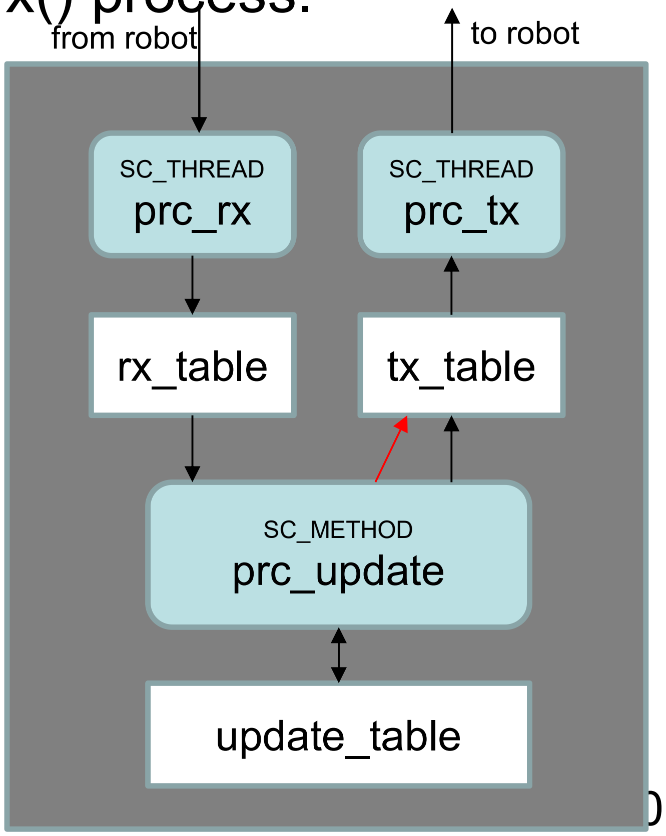
- Every clock,
    - x_current = x_prev + (speed)(clock period) or
    - y_current = y_prev + (speed)(clock period). Not both.
    - obstacle positions are also updated

# Processing Module Structure

Three processes: prc_tx(), prc_rx(), prc_update() processes and two tables

rx_table : Activity status modified by prc_rx() process.
When rx_table is modified by the prc_rx ()process,
prc_update() process is notified.

tx_table: Transmit status modified by prc_update process.
When prc_update() process has Information to send to Server module (via Robot module), prc_update() process modifies tx_table and notifies prc_tx() process



## Processing Module

In prc_update() process, there are four main loops

- For each **valid** robot, update the position: use the map information and path sequence
- For **all** obstacles, update positions: use the map information and predefined path sequence
- For each **valid** robot, check the distance between each robot and all other obstacles.
- For each **valid** robot, check the distance from the boundary

Validity of the status of robots are updated if any change is made from rx_table.

## Processing Module

rx_table: 3 columns

- Column 1: Robot Index – all robots are listed
- Column 2: New Status (Revised by Server)
- Column 3: Modified (set to 1 is revised by prc_rx(), after prc_update(), reset to 0)

tx_table: 3 columns

- Column 1: Robot Index – all robots are listed
- Column 2: New Status (Revised by Processing)
- Column 3: Modified (set to 1 is revised by prc_update(), after prc_tx(), reset to 0)

# Processing Module

Update database structure:

| Robot Index | Navigation Status | Current Grid | Next Grid | Next Node | Speed | Server Status |
|---|---|---|---|---|---|---|
| 0 | STOP | 23 | 2 | 1 | 0 | Stop by the  Obstacles |
| 1 | IDLE | 45 | N/A | 2 | 0 | No Service |
| 2 | MOVING | 4 | 2 | 3 | 1 | Stop at the Boundary |
| 4 | STOP | 3 | 4 | 3 | 0 | Stop by the Server |

## Robot Module

There are 5 processes:

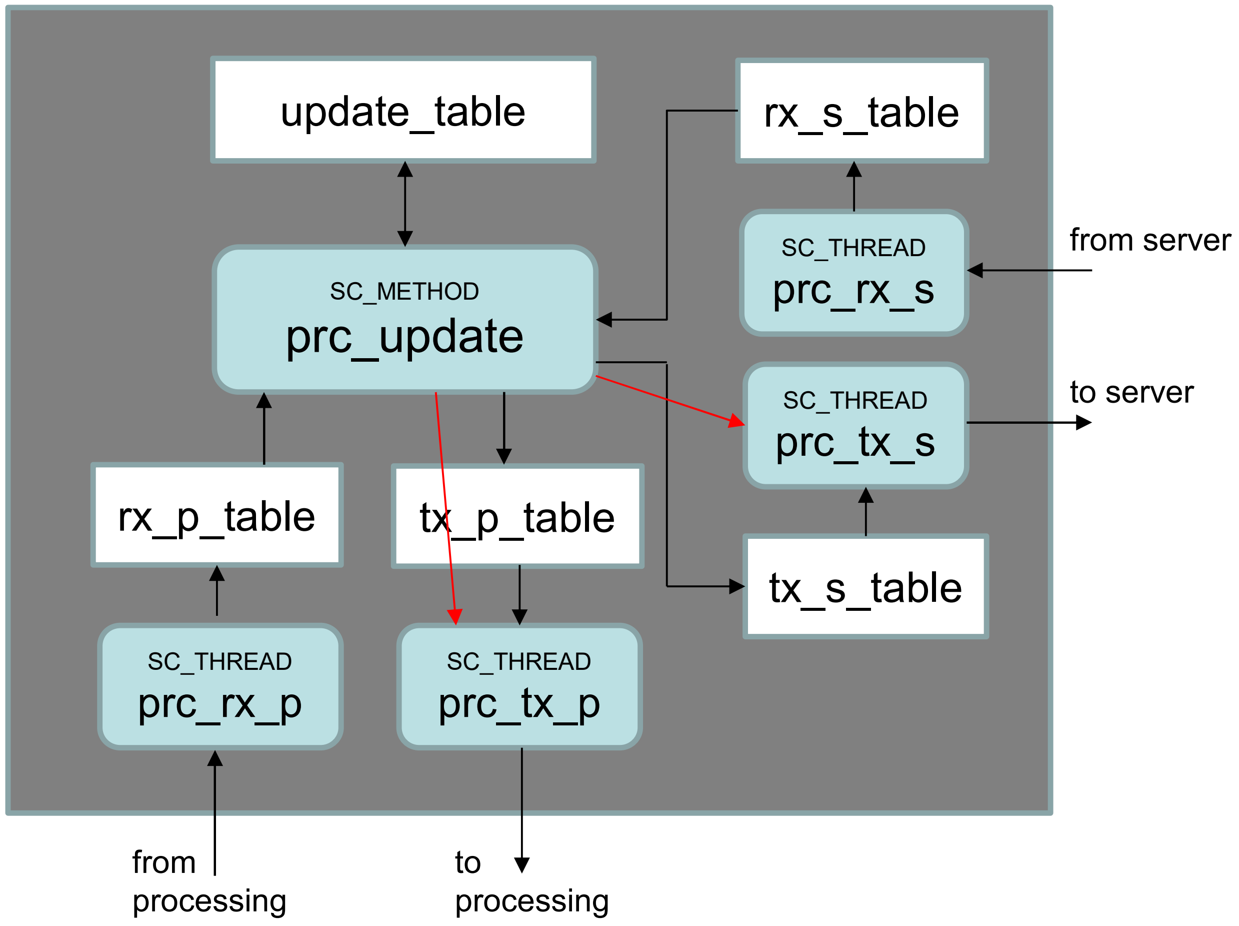
- prc_update(),
- prc_tx_s(),
- prc_rx_s() (Server side)
- prc_tx_p(),
- prc_rx_p() (Processing side)

4 tables:

- rx_s_table and tx_s_table: (Server side)
- rx_p_table and tx_p_table: (Processing side)

## Robot Module

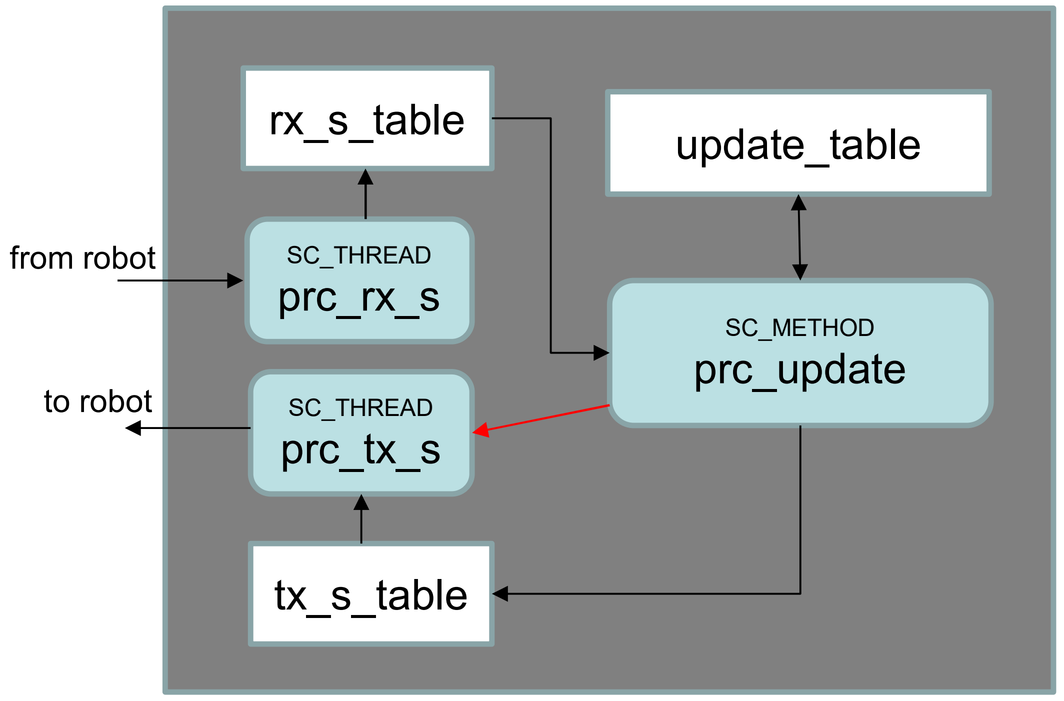
Robot module structure:



# Server Module

Three processes:
- prc_tx(),
- prc_rx(),
- prc_update().

Two tables:
- rx_table,
- tx_table:



## Server Module

Update database structure:

| Robot Index | Navigation Status | Current Grid | Next Grid | Next Node | Speed | Server Status |
|---|---|---|---|---|---|---|
| 0 | STOP | 23 | 2 | 1 | 0 | Stop by the  Obstacles |
| 1 | IDLE | 45 | N/A | 2 | 0 | No Service |
| 2 | MOVING | 4 | 2 | 3 | 1 | Stop at the Boundary |
| 4 | STOP | 3 | 4 | 3 | 0 | Stop by the Server |

# Basic Message Passing Mechanism

Status Value from Robot (Processing) to Server

0 STOPPED (stopped due to obstacles)
1 MOVING (started after obstacle clearance)
2 CROSSING (just before the boundary crossing)
3 STOPPED (stopped at the boundary due to no Ack from Server)

Status Value from Server to Robot (Processing)
4 OK (ok to cross)
5 STOP (stop, do not cross)
6 START (resume from previous stop)
7 SPEED (sending speed value via FIFO) – Phase 2
8 PATH (sending path sequence via FIFO) – Phase 2

## Server Update Process

Respond to boundary crossing request. Decision is made based on the table. If robot 1 send boundary crossing request, server send STOP and change the status to STOP

| R Index | Current | Next | … | Status |
|---|---|---|---|---|
| 1 | 1 | 2 | | MOVING |
| 2 | 2 | 3 | | MOVING |
| 3 | 4 | 5 | | IDLE |

If the robot 2 crossed, the server notifies the robot 1 to resume.

If robot sends STOPPED status, server updates the table

If robot sends STARTED status, server updates the table

## Inter-Module Interaction

RX modules are connected to TX modules

Out Connection (TX to destination RX process)

- sc_out<bool> flag,
- sc_out<sc_uint<16> > data;
- sc_in<bool> ack;

In Connection (Destination TX Process to RX)

- sc_in<bool> flag,
- sc_in<sc_uint<16> > data;
- sc_out<bool> ack;

## Communication Processes: TX

```
// for prc_tx()
while(true) {
    wait(); // wait for signal/event from prc_update
    flag = 0
    while(tx_counter != 0) {
        flag = 1; // creating posedge
        data = status; // information from tx_table
        wait(); wait for ack form the receiving process
        flag = 0;
        tx_counter--; // repeat until no outstanding request
    }
}
```

## Communication Processes: RX

```
// for prc_rx()
while(true) {
    ack = 0;
    wait(); // wait for request (flag.pos()) from sending process
    status = data; // update rx_table
    ack = 1;  // creating posedge
    rx_counter++;  // this counter will be checked by prc_update()
}
```

## Verification and Simulation

- Simulation with 2 robots and 2 obstacles in 2-dimensional maps described in the Project Description.
- Grid size is 2m by 2m.
- Robots starts at the opposite corners of the map.
- Obstacles move (back-and-forth) with 4 m/s speed.
- Robots move with 2m/s speed.
- The path for the robots are parameterized.

## Submission Requirement

- Source codes for robot, processing, server, top main and necessary test-bench codes
- Database structures for submodules
- Pseudo code description for handshaking machanism
- Verification/Simulation results
- Summary report (1-2 pages)

The report grading will be based on

1. Clarity of the report, 2. Completeness of the results.

Electronic submission per group

## Grading

1. Source Codes (4)
    1. Processing
    2. Robots
    3. Server
    4. Main and Testbench
2. Pseudo Codes/Usage of Data Structure (4)
    1. Data Structure for Processing, Robots, Server
    2. Handshake Mechanisms R-P, R-S
3. Correct Map Description/Implementation (2)
4. Verification of Operation (5)

Total: 15 Points