Warehouse Simulation

Hardik Bhati IIT2019013
Garvit Suri IIT2019057

under the guidance of

Dr. Rahul Kala

IIT Delhi
Indian Institute of Technology Delhi

IHFC

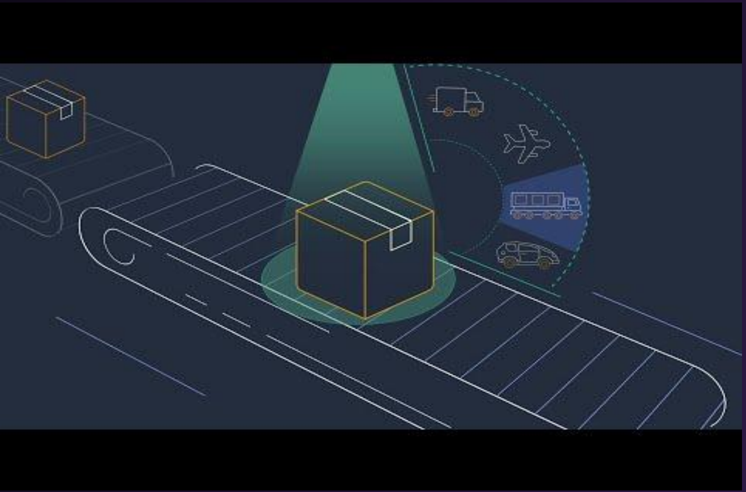
सत्यमेव जयते
Department of Science and Technology (DST)
DST

IIIT Allahabad

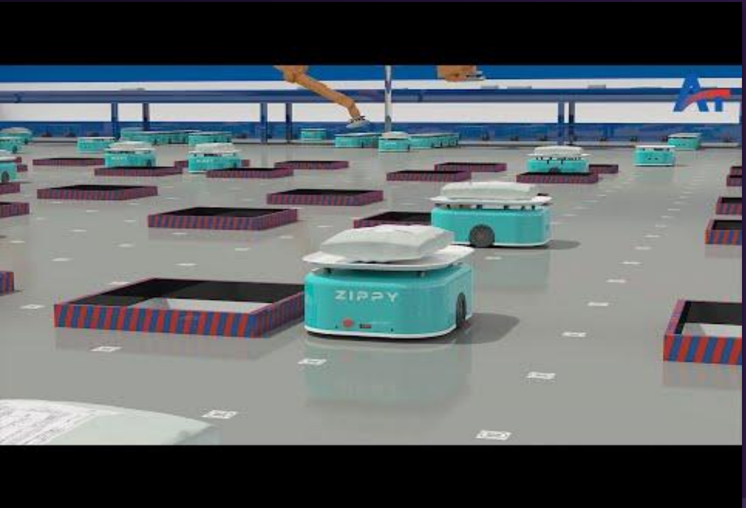
CIR

Simulation of Warehouse Robots

Our main aim is to simulate robots in a way such that they are able to successfully simulate a real working environment in a warehouse.

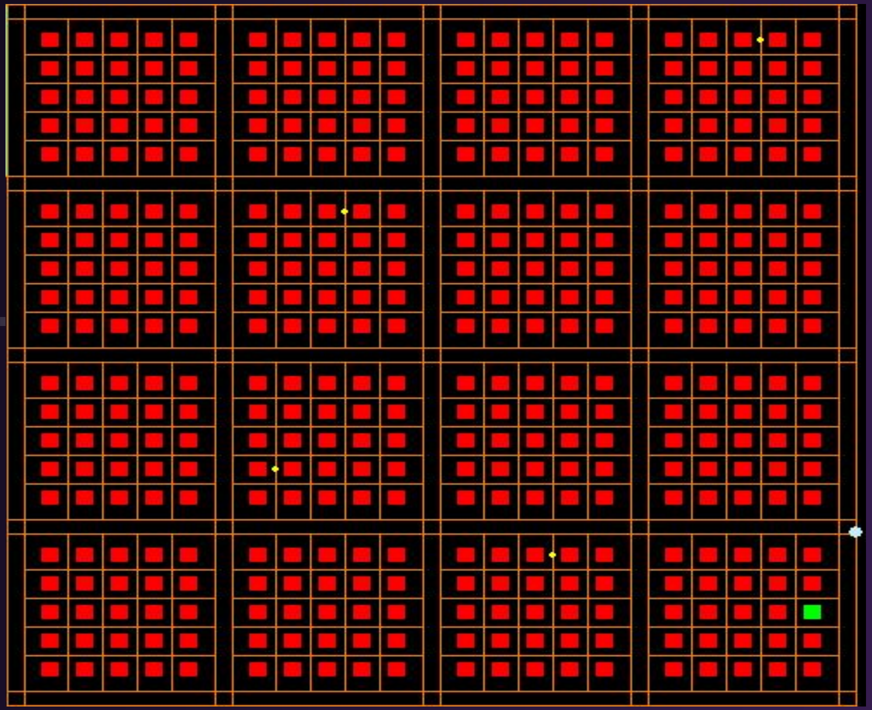


- **Ordering Mechanism (Ref - Amazon Fulfillment Center)**



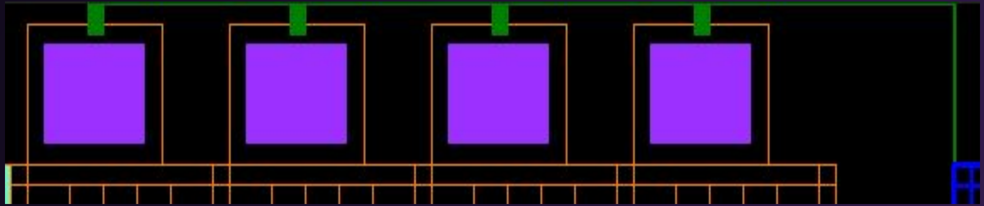
- **Sorting Mechanism (Ref - Addverb Technologies)**

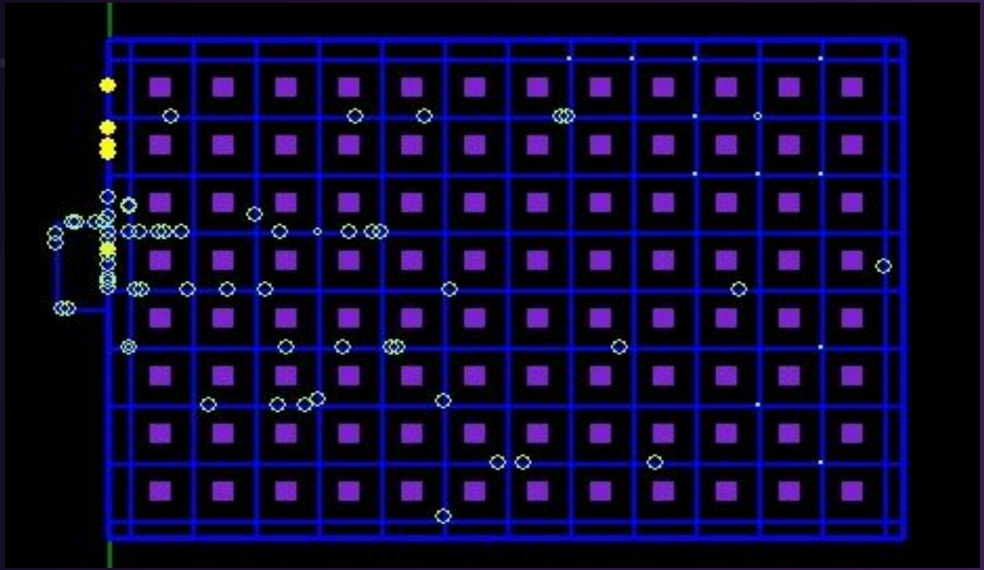
SIMULATOR OUTLINE



- ITEM STORAGE AREA OF WAREHOUSE

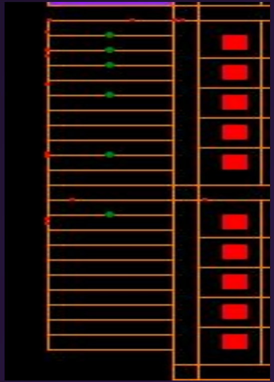
SIMULATOR OUTLINE



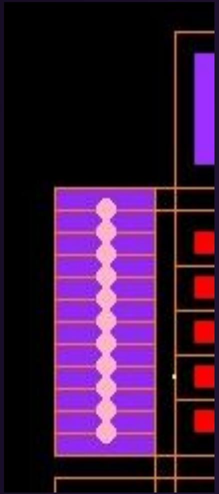


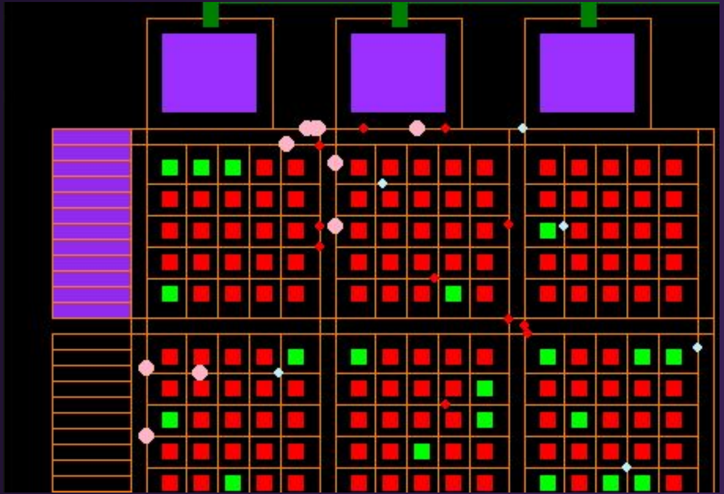
- HUMAN COUNTER
- SORTING ZONE

SIMULATOR OUTLINE



- CHARGING ZONE





- TRUCK BOTS

APPROXIMATE COMPARISON

OUR SIMULATOR	MICRO FULFILLMENT CENTRE
10 pixels	1 m
10 timesteps	1 second
Speed→ 1 pix/timestep → 1 m/s	Speed→ 1.5 m/s
Battery Life→ 6.6 min	Battery Life→8 Hours
Charging Time → 2000 timesteps→ 3.3 min	Charging Time → 30 mins
Workshop Size(Variable) → 500 pixel*500 pixel =2500 m^2 = 25000 sq. ft.	Workshop Size(Variable) → 2000 to 50000 sq. ft.

CONGESTION MANAGEMENT

- Scheme 1 (Without Congestion Management)
- Scheme 2 (With Congestion Management)

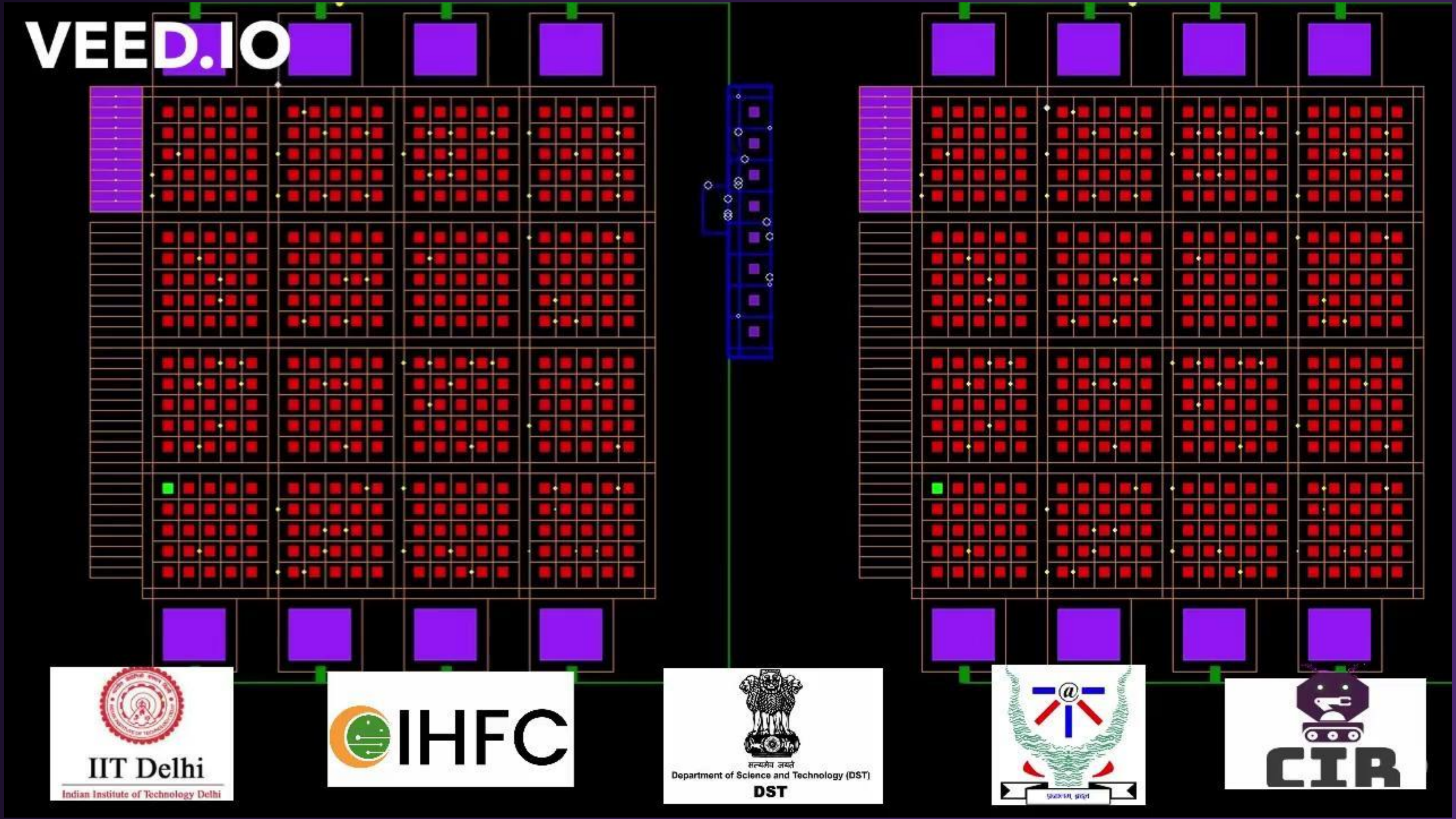
Scheme 1 (Without Congestion Management)

- **Only Shortest Path Matters.**
- **No Replanning after evaluating the Master Plan.**

Scheme 2 (With Congestion Management)

- **Shortest Path and Heat Value of Path both Matters.**
- **Do Re-planning if Congestion around the agent is high.**

CONGESTION MANAGEMENT Demo



INDUCED CONGESTION

Description:
We vary the number of Agents and to induce congestion every order was assigned to go to a single human counter with Number of Orders=100.

Number of Agents	4	10	25	50	75	100
Without Congestion	98501	38750	17878	13670	15070	16237
With Congestion	99338	41773	17887	13304	14469	15637

TEST -2 (NORMAL SCENARIO)

Description:
We vary the number of Agents and this time the human counters were allotted to each order in a random fashion with Number of Orders equal to 100.

Number of Agents	4	10	25	50	75	100
Without Congestion	38062	16238	7418	4862	4636	5752
With Congestion	38492	16320	7355	4890	4646	5823
When being more sensitive to Congestion	38095	16392	7485	4955	4632	5849

INTERSECTION MANAGEMENT

- Scheme 1 (Normal Intersection Management)
- Scheme 2.1 (Partially Randomized Intersection Management with Epsilon = 0.5)
- Scheme 2.2 (Partially Randomized Intersection Management with Epsilon = 0.75)
- Scheme 3 (Fully Randomized Intersection Management)

Scheme 1 (Normal Intersection Management)

- Prefer the lane in which there are more number of cars waiting.
- More Deterministic
- There is a chance of starvation in this.

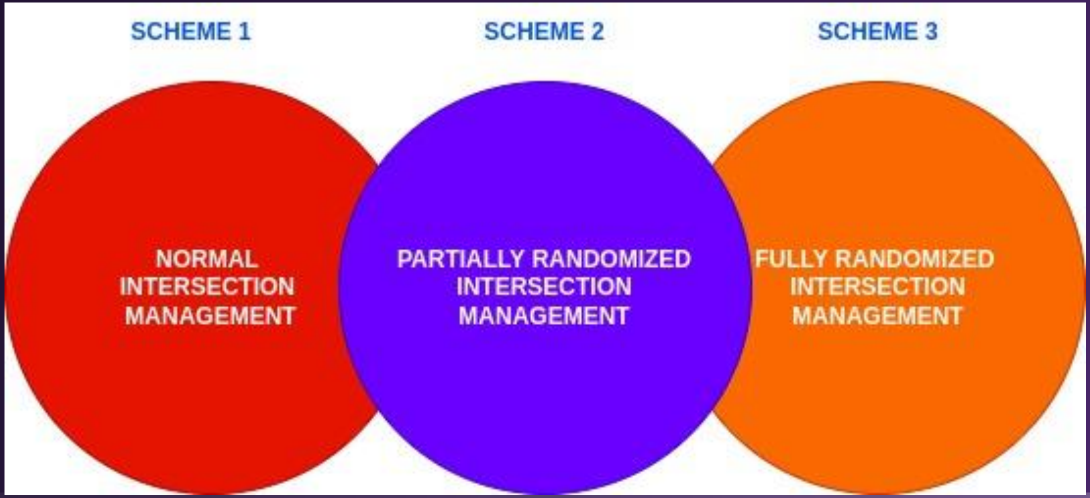
Scheme 2 (Partially Randomized Intersection Management)

- Introduced a Probability Factor (ε) with which we did the Random Choosing and otherwise follow the Scheme 1.
- Flavor of Randomization expected to improve Results.

Scheme 3 (Fully Randomized Intersection Management)

- Just for testing the result of Randomization to maximum extent, we made the value of Probability Factor (ε) to be 1. So, it always behaves in a random way.

VENN DIAGRAM OF ALL SCHEMES



TESTING INTERSECTION MANAGEMENT

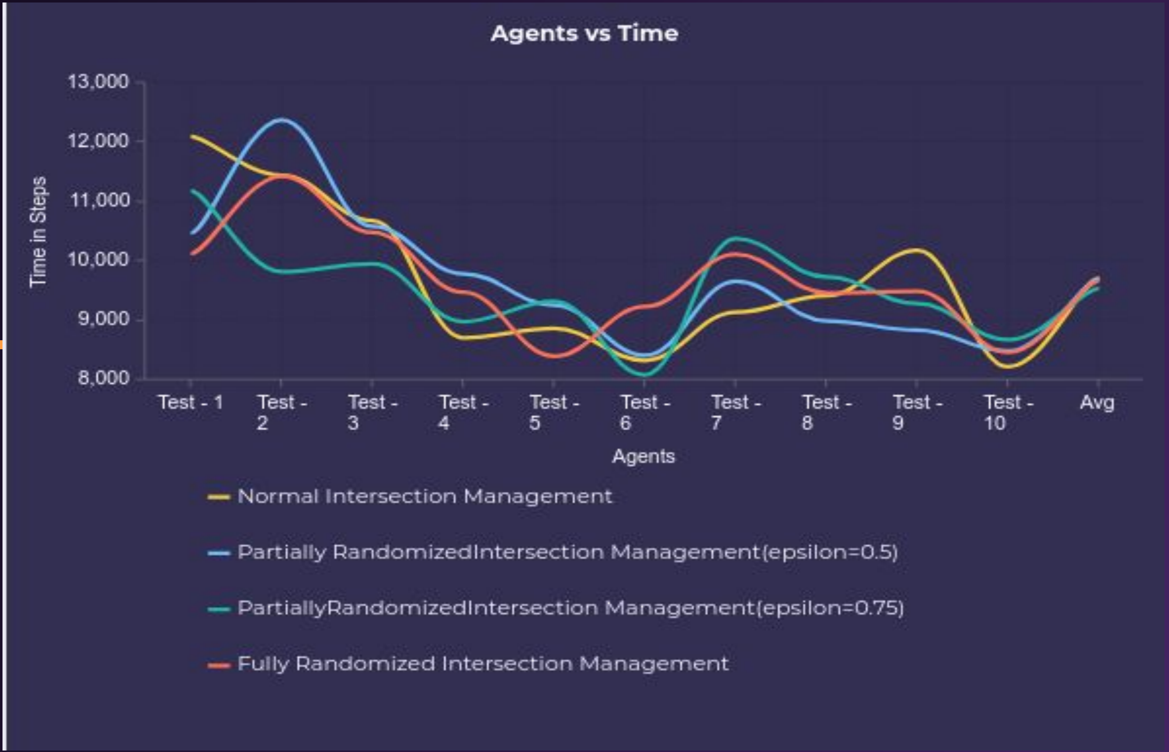
Test Description

- Number of Agents = 300
- Number of Items in Warehouse = 300
- Number of Test Cases = 10
- Total Number of Orders for Testing = 300*10 =3000 Orders

RESULTS

Scheme	Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	Test - 7	Test - 8	Test - 9	Test - 10
Normal Intersection Management	12098	11449	10683	8712	8871	8330	9142	9424	10186	8222
Partially Randomized Intersection Management (epsilon=0.5)	10477	12383	10585	9790	9258	8416	9664	8994	8841	8486
Partially Randomized Intersection Management (epsilon=0.75)	11184	9821	9956	8982	9324	8087	10381	9735	9285	8677
Fully Randomized Intersection Management	10129	11432	10488	9478	8398	9235	10116	9468	9497	8463

RESULTS (Cont.)



Scheme	Average Steps
Normal Intersection Management	9712
Partially Randomized Intersection Management (epsilon=0.5)	9689
Partially Randomized Intersection Management (epsilon=0.75)	9543
Fully Randomized Intersection Management	9670

Item Placement Strategies

Scheme 1 (Random Fashion)

- Randomly Distributed Items

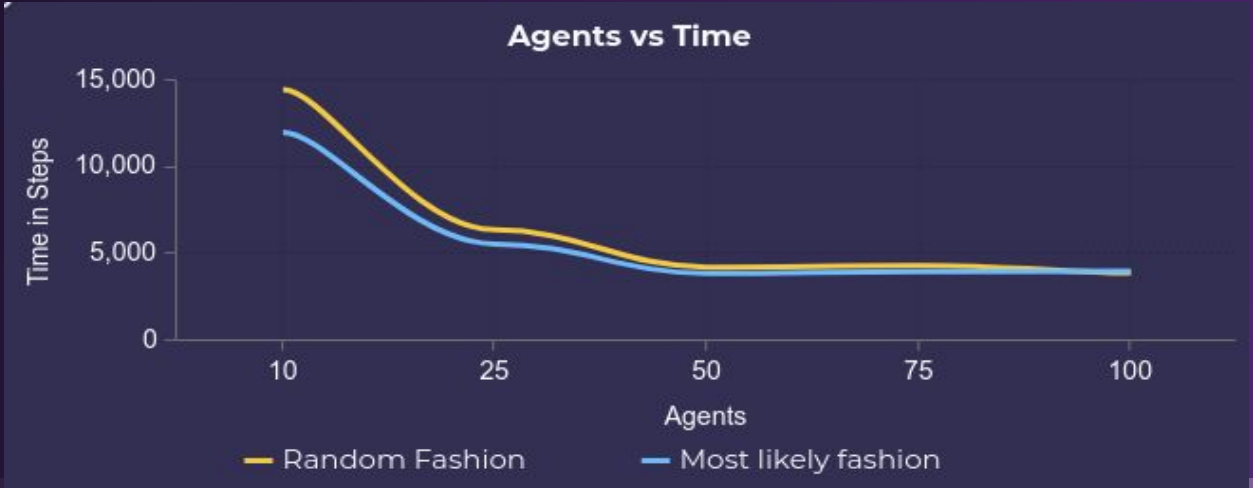
Scheme 2 (Most Likely Fashion)

- Items most likely to be ordered are near to the human counter.
- Intuitively reduces timesteps but in actual there is a tradeoff between congestion and timesteps.

Item Placement Strategy

Number of Items = 100
Items Ratio: 9:1

Number of Agents	10	25	50	75	100
Random Fashion	14485	6401	4258	4354	3918
Most likely Fashion	12035	5592	3891	3993	4014



Item Placement Strategy Demo

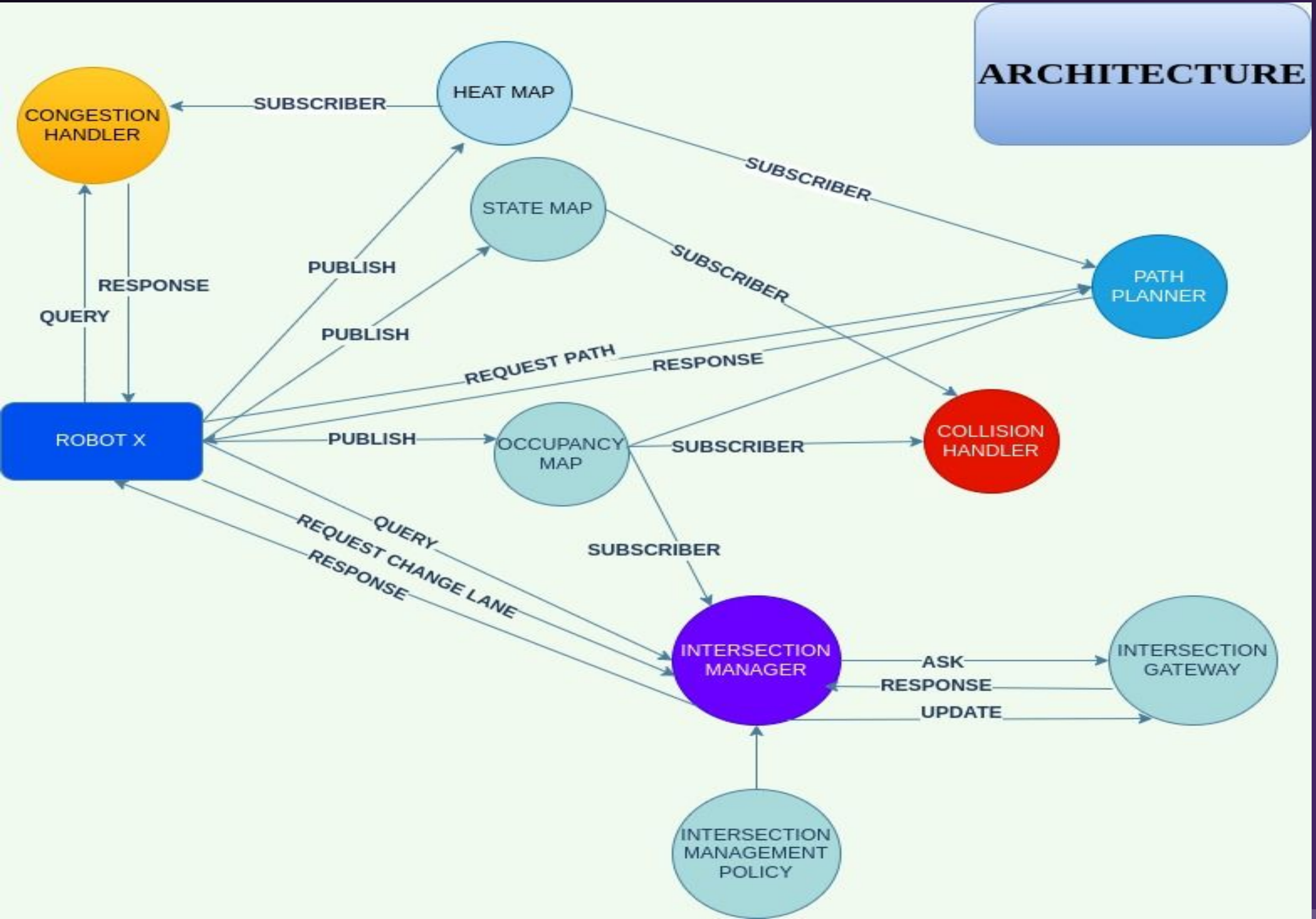
Result Comparison with Actual Warehouse

Real Life→ One day output→ 5000 Orders per Day in 25000 sq ft Semi-Automated mini Fulfillment Centres

Our Simulator→ Average Time for 100 orders = 10000 Timesteps = 1000 sec
Now, in 24 hours → Orders= 24*60*60*100/1000= 8640 Orders Per day in 25000 sq ft Fully-Automated Simulator

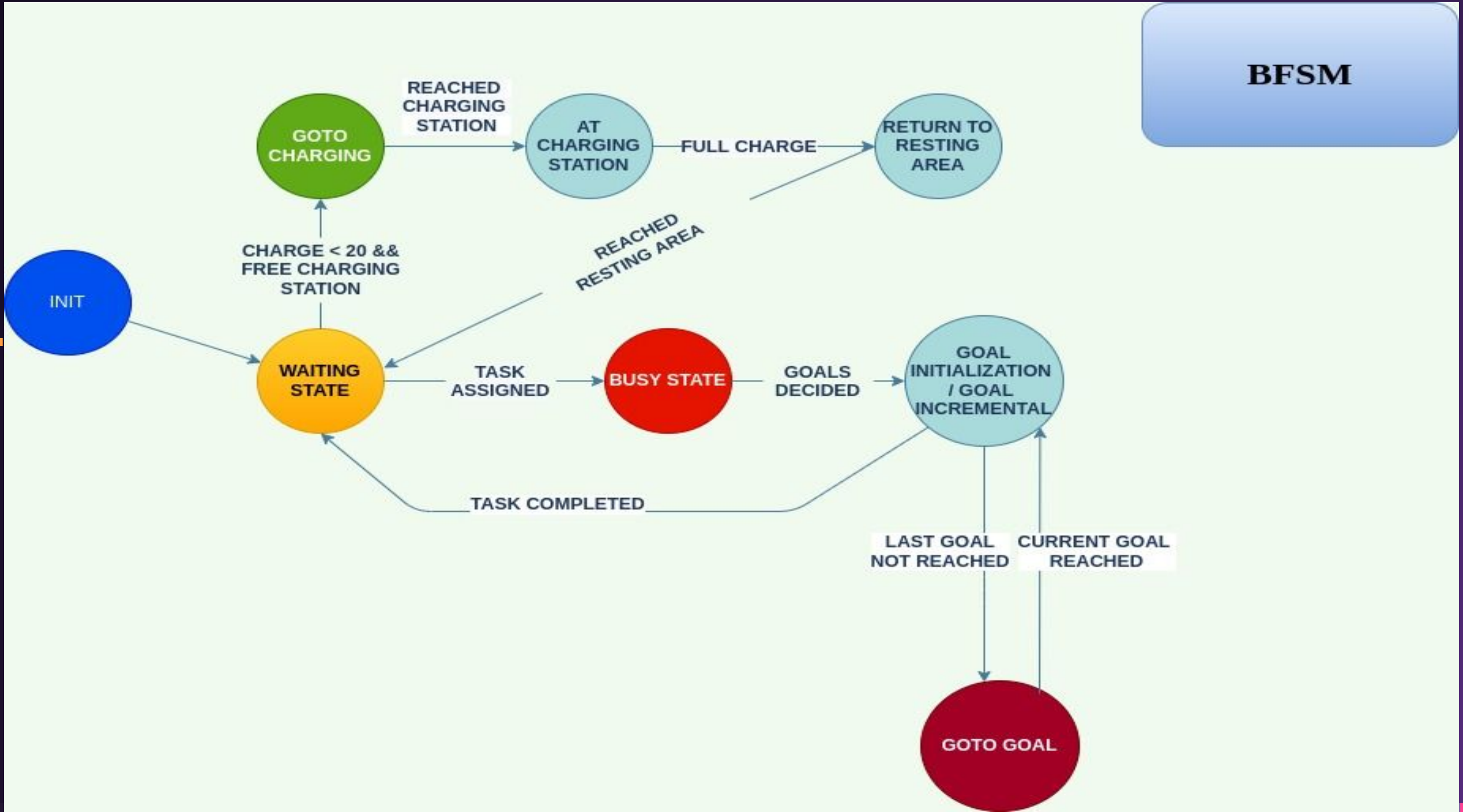
ARCHITECTURE

OVERALL ARCHITECTURE



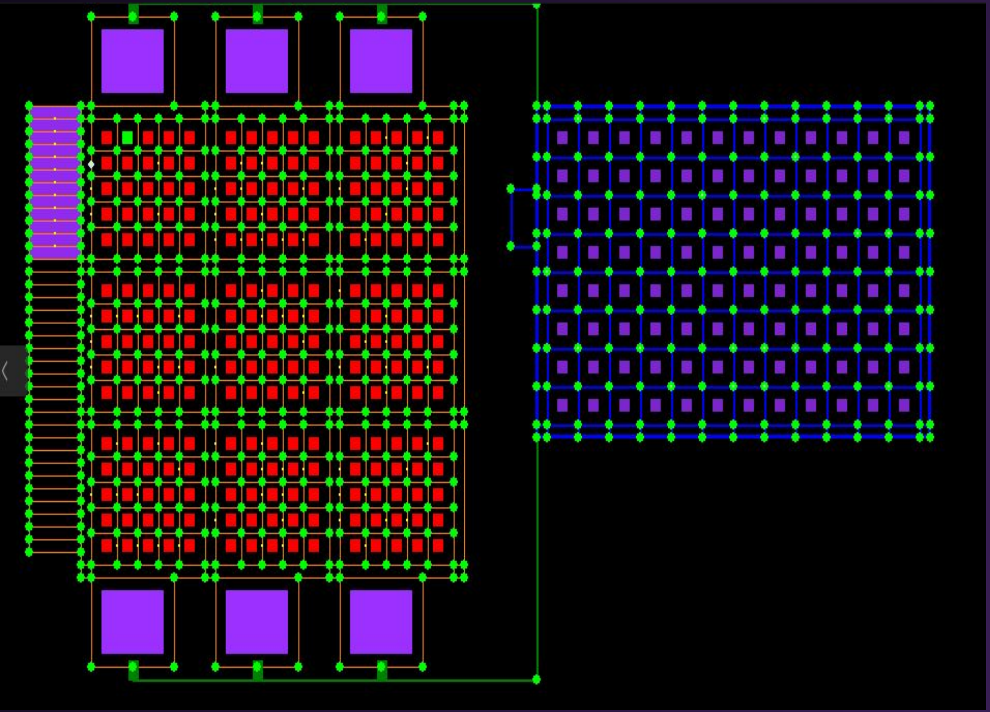
- PUBLISHER-SUBSCRIBER MODEL
- QUERY-RESPONSE SERVICE

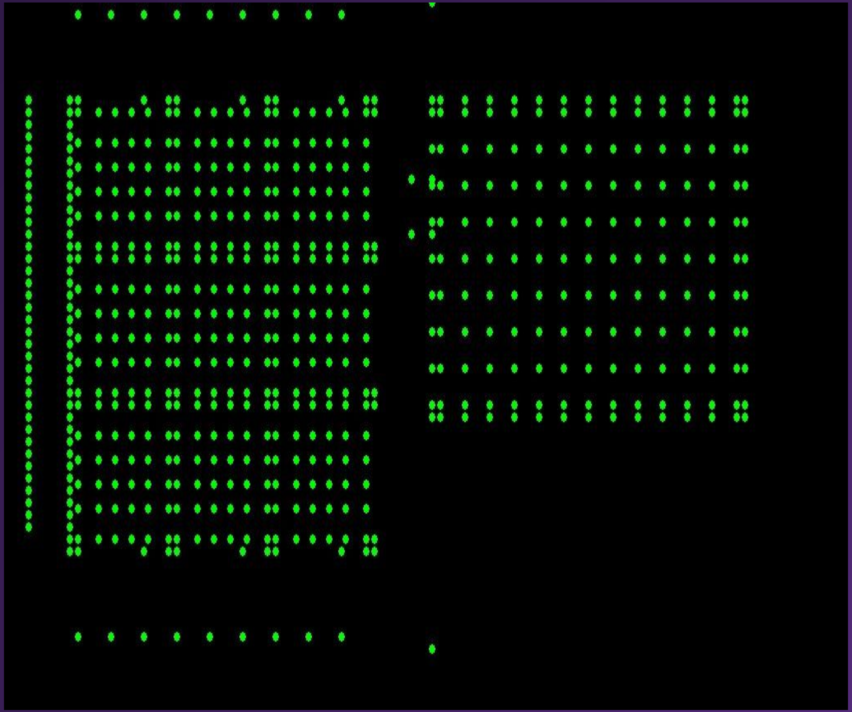
BFSM MODEL


BFSM
GOTO CHARGING
REACHED CHARGING STATION
AT CHARGING STATION
FULL CHARGE
RETURN TO RESTING AREA
REACHED RESTING AREA
CHARGE < 20 && FREE CHARGING STATION
INIT
WAITING STATE
TASK ASSIGNED
BUSY STATE
GOALS DECIDED
GOAL INITIALIZATION / GOAL INCREMENTAL
TASK COMPLETED
LAST GOAL NOT REACHED
CURRENT GOAL REACHED
GOTO GOAL

PATH PLANNING

GRAPH ABSTRACTION





3 PHASES OF PATH PLANNING

1) **Accessibility (Access Point) -** From the point where we are starting the journey, it is not necessary that the source is in our Roadmap. So in our Grid, we need to find the nearest Point to the Source which is in the Roadmap and that point would be the nearest intersection to that Point. So, using BFS we do the task of finding the access point for our journey.

2) **Connectivity -** On reaching the Access Point, we apply A* Algorithm using heuristic as a linear combination of Heat Values and Manhattan Distance from source to Goal. Now we will get a Path but in the Roadmap domain so we need to convert the Path to Workspace Connectivity Graph to move the agent. After this Phase, we will either reach the Goal or a point in the Roadmap Graph that is nearest to Goal.

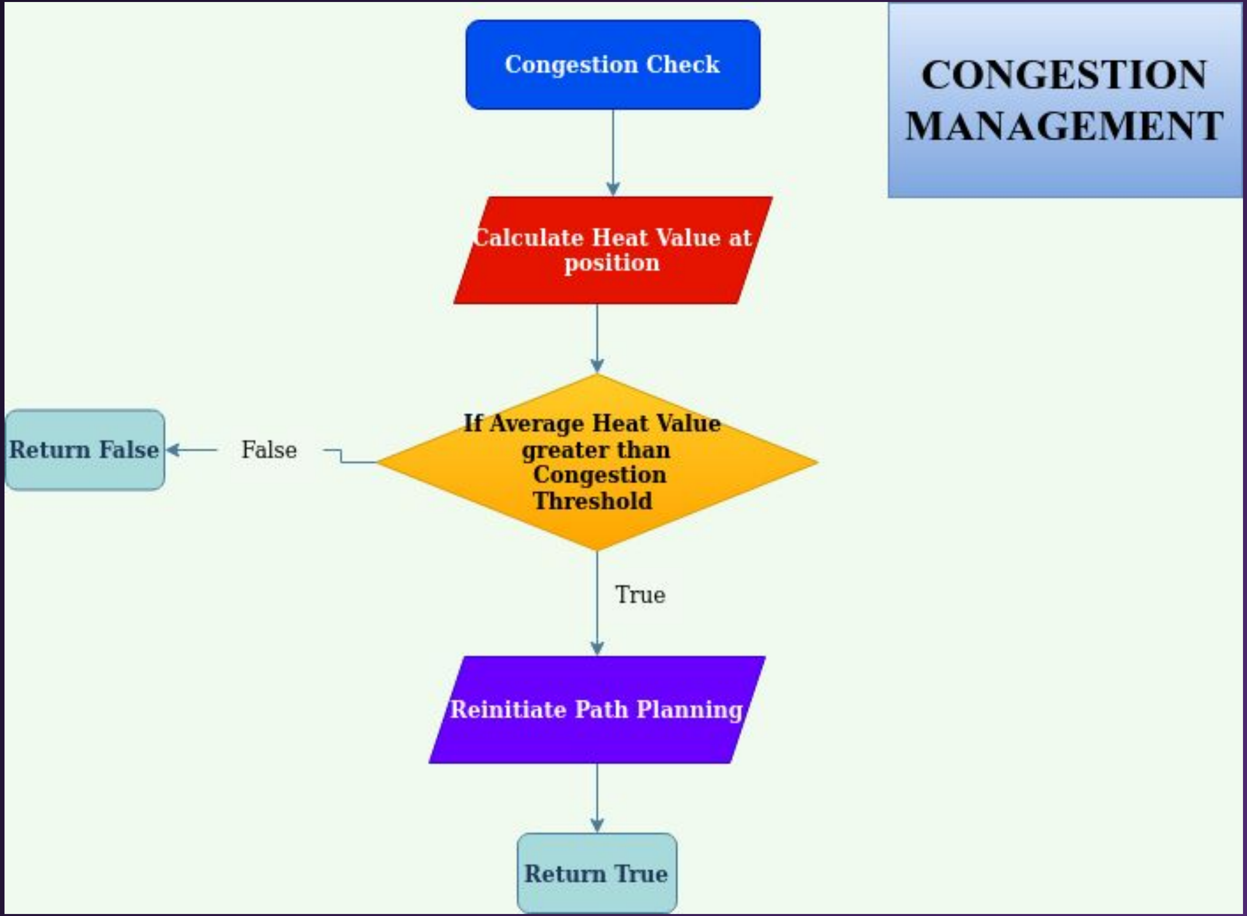
3) **Departability (Depart Point) -** If the Goal we are planning for is not in the Roadmap Graph, then we need to again find the simple straight path to the Goal.

PSEUDO CODE FOR GOTO GOAL CONTROLLER

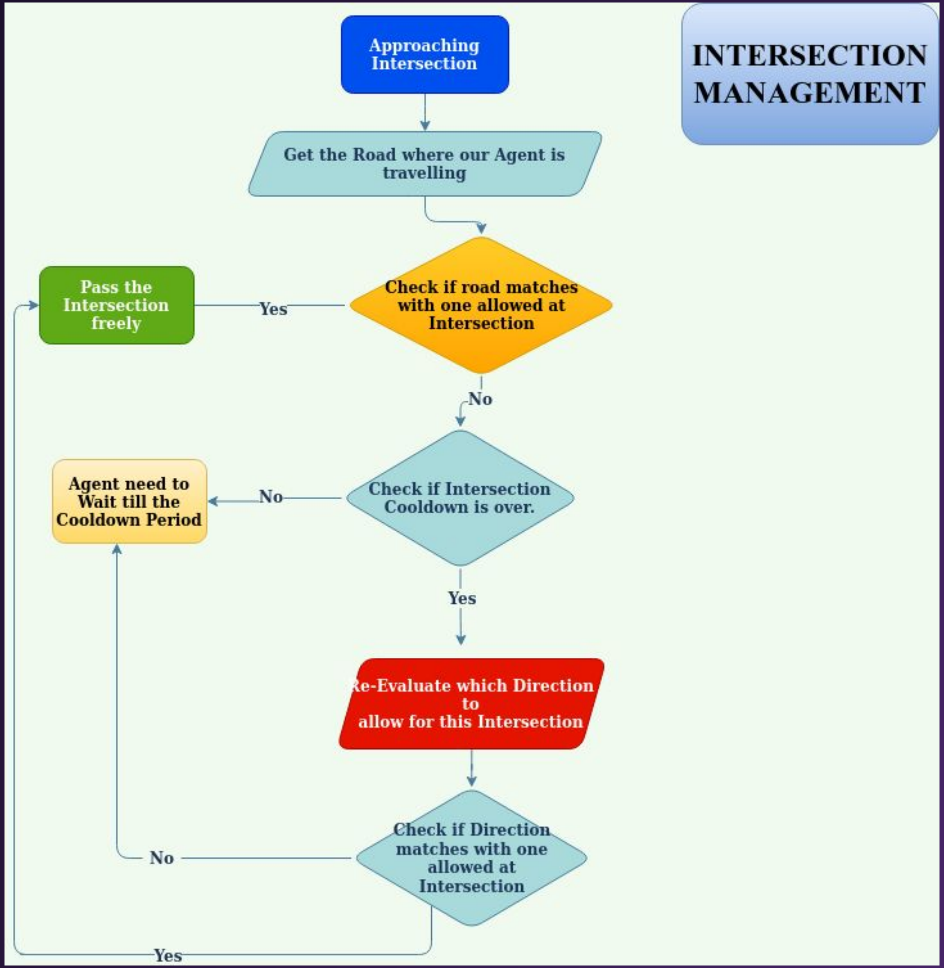
```
τau--> Path to travel on

function Move(Agent):
    flag=1
    if τau is not empty:
        Ghost_Robot<-- Agent
        Ghost_Robot.position=τau[1]
        if Collision_Check(Ghost_Robot.position) is True:
            return (Vx=0,Vy=0)
        if Ghost_Robot is at Intersection:
            if Intersection_Management(request=False,query=True,Ghost_Robot.position) is not
road(Agent.Position):
                Intersection_Management(request=True,query=False,Ghost_Robot.position)
                flag=0
        if flag == 1:
            if Congestion_Management(Agent.position) is True:
                τau=[]
                return (Vx=0,Vy=0)
            else:
                τau.pop_front()
                New = Speed(Ghost_Robot.position,Agent.position)
                Agent<--Ghost_Robot
                return New
        else:
            return (Vx=0,Vy=0)
    else:
        τau=Planning(Agent.position,Agent.goal)
        return (Vx=0,Vy=0)
```

```
function Speed(A,B):
    return Bx-Ax,By-Ay

function Planning(position,goal):
    α=Path from position to Access point
    β=Path from Access point to Depart point
    δ=Path from Depart point to goal
    return α+β+δ

function Collision_Check(position):
    if position is Occupied:  //Current Position is input from Sensor
        return True
    else:
        return False

function Intersection_Management(request,query,Intersection):
    if query==True:
        return road(Intersection)
    else:
        if last_time_updation(Intersection) > threshold:
            Re-Evaluate(Intersection)


function Congestion_Management(position):
    if Heat(position)>Threshold:
        return True
    else:
        return False
```

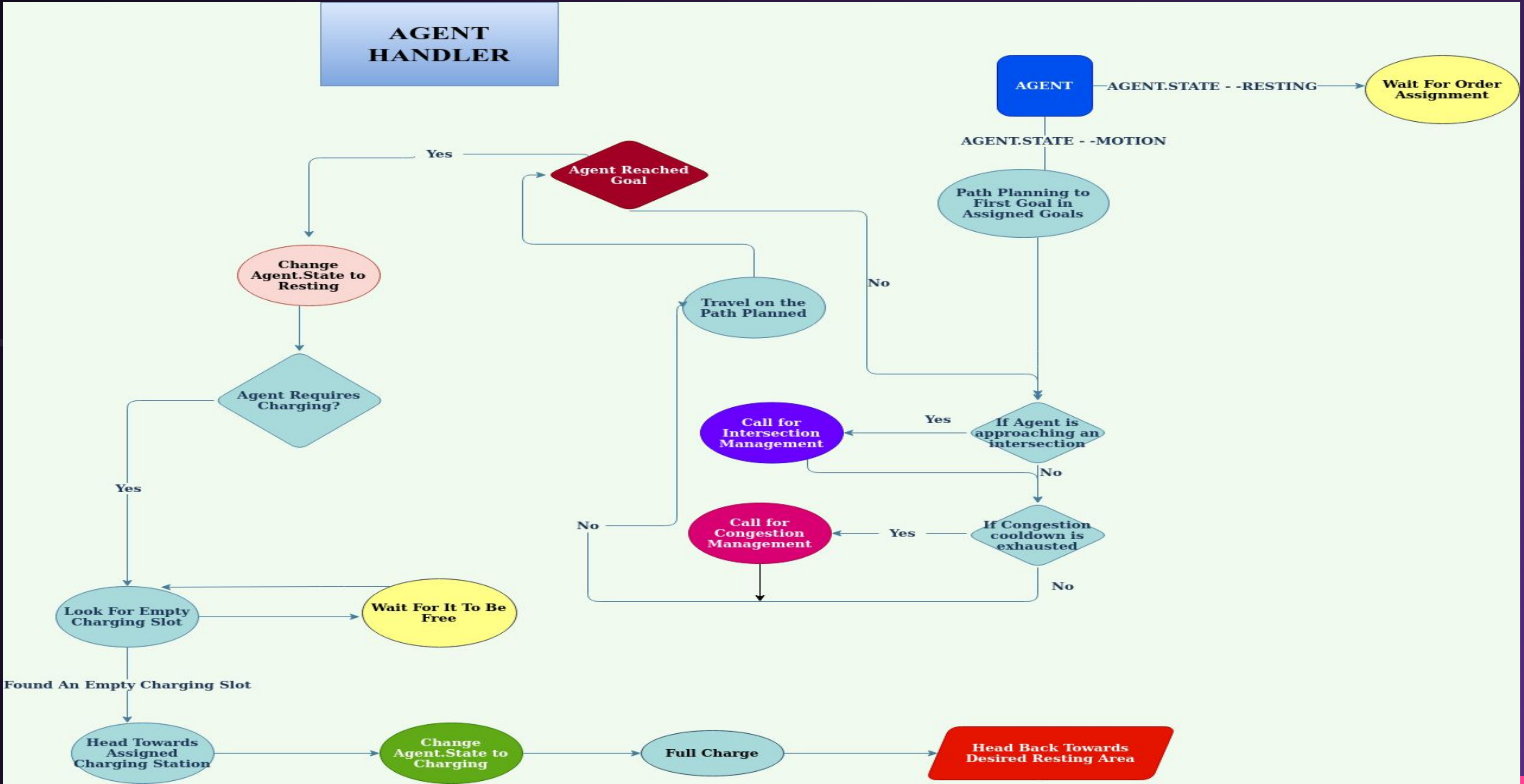
BLOCK DIAGRAM OF CONGESTION MANAGEMENT



BLOCK DIAGRAM OF INTERSECTION MANAGEMENT



BLOCK DIAGRAM OF AGENT HANDLER



CONCLUSION

CONCLUSION

- **Congestion Management:** Moderately Sensitive to Congestion
- **Intersection Management:** Partially Randomized Intersection Management (with Epsilon = 0.75)
- **Item Placement Strategy:** Random Fashion when many agents and Most Likely Item near Counter when less agents

THANK YOU!!