

Interconnecting robots using IoT for multi-agent patrolling

April 14, 2021

Abstract: *Patrolling tasks can be encountered in a variety of real-world domains, ranging from computer network administration and surveillance to computer wargame simulations. It is a complex multi-agent task, which usually requires agents to coordinate their decision-making in order to achieve optimal performance of the group as a whole. In this paper, we present an in-depth discussion of multi-agent patrolling task issues, and an empirical evaluation of possible solutions. We have used conscientious reactive architecture with IoT framework of multi-agent systems.*

ADD

1 Introduction

Patrolling is a key part of any security protocol as it can both prevent as well as detect the acts of crime. It has been already established ([13]) that the presence of security personnel in a particular area/locality deters criminals or other bad actors from carrying out violence. In addition, patrolling also helps in identifying suspects by observing unusual behaviours. This task is inherently multi-agent as, traditionally, *Policemen* (either in groups or individually) visit different places in a defined schedule under the jurisdiction of their local unit. Given the advancements in the fields of autonomous driving and computer vision, today we can explore the possibility of automating the patrol process and eliminate humans. Various Multi Robot Patrolling Problems (detailed in Section 2) have been developed to automate the patrolling process.

The Multi Robot Patrolling Problem (MRPP) deals with planning of traversal strategies for multiple autonomous vehicles patrolling a given environment. A patrol strategy is said to be effective if the vehicles visit different locations in the environment as often as possible. For the first time, the multi-agent patrolling using an IoT (Internet of Things) methodology is investigated through this work. More specifically, we propose a decentralized network of interconnected IoT [15] devices that would support coordination among patrolling vehicles. While the concept of IoT is new, its core is interconnection and extracting useful information from this interconnection.

Majority of the relevant literature develops the MRPP problem from an agents' point of view. In particular, various possibilities of communication and capabilities between the agents (vehicles) have been explored. While this is an important direction, a decentralized MRPP strategy for patrolling an outdoor campus-like environment cannot assume full communication between the vehicles distributed all around the campus. This is because of the extend of patrol spread over several square kilometres of area. In order to address this drawback, in this work we suggest an alternate approach by setting up a network of IoT devices. The devices in this network communicate locally amongst themselves and also with the patrolling vehicles in their vicinity, thus maintaining the benefits of decentralized approach. While IoT device network supports coordination among agents, it must facilitate high-level decision making for uniform patrolling by multiple agents. Furthermore, the use of IoT devices introduces dependency on information exchange on these devices. Therefore, an elaborate study is needed to evaluate the patrolling performance under device failures.

In this work, we investigate the use of IoT technique for providing a solution for decentralized MRPP and summarize our contributions as follows:

- A novel technique for Multi Robot Patrolling using IoT devices for a decentralised decision making and robust solution against device failures.
- A local, reactive, decentralized algorithm (MRPP-IoT) to minimize Graph Idleness (defined in Section 3) with IoT devices at Junctions as the decision making units.
- Performance study of MRPP-IoT algorithm based on simulations carried out on SUMO (a traffic simulator). Comparison of the MRPP-IoT algorithm with two state of the art strategies - Conscientious Reactive as well as Reactive with Flags [8] demonstrate the possibility of decentralised solution for MRPP that maintains the uniformity of patrolling even during device failures.

The rest of the paper is organized as follows - Section 2 contains related works from the literature, Section 3 describes the Problem Formulation. In Section 4 we detail the proposed MRPP-IoT algorithm. The simulation analysis and results are presented in Section 5 and we conclude in Section 6.

2 Related Works

The investigation into MRPP started in the early 2000s, due to a confluence of factors like, increase in computational power, increasing maturity of the Multi-Agent domain, and others. MRPP can also be seen as a natural extension of mapping and other coverage tasks, where the objective is to visit (cover) all the locations (area) in the environment. In MRPP, the task of visiting all the locations is to be done repetitively preferably not in the same sequence.

Different heuristics [8] under varied problem setups (for example, level of communication between the agents) has been developed using the term *Idleness*.

Though the work in [8] is pioneering in many aspects, the empirical validation of the strategies are done under ideal conditions. In [3], the area is discretized into grids, and the agents negotiate to divide the grid cells amongst themselves. Each grid cell has a priority associated with it specified using probability of an event occurring at that location. Two strategies [6] based on dynamic task assignment have been developed, the second of which is an auction based mechanism.

The objective of a generalized partitioning strategy [2] is to arrive at a minimum number of agents which can patrol the given environment constrained by minimum frequency of visits. Each partition is assigned one or more agents. Though the partitioning is possible in polynomial time, the strategy is valid only for Outer planar Graphs. A graph partitioning based algorithm [11] partitioned the given environment into similar sized sub-graphs which are then allotted per agent. However, the partitioning method is not flexible for online applications. In [12], a Bayes' rule based strategy uses the conditional probability for traversing any particular edge depended on the idleness of the node, the number of visits to the node as well as how frequently that edge was traversed in the recent past. In [16], the problem is formulated as an MDP and Q-learning is used to obtain a strategy which reduced the idleness.

There have been many works in the sub-domain of *adversarial* patrolling [1], [17]. The objective in adversarial patrolling is to make sure that the adversary (intruder), cannot attack (intrude) the given environment. The perimeter setting is considered and probabilistic guarantees in terms of detecting the intrusion have been obtained. In [1], realistic assumptions are made in terms of evolution of events (intrusion), non-deterministic nature of observability by the sensors, etc. A few works (notably [9] and [10]) have also focused on setting up their problem in realistic scenario, such as considering localization and sensing issues, non-uniform performance of different agents etc. In [4], the objective is to find the minimum number of agents which can be successful in intrusion detection. The environment is described as a grid with different *penetration time* for some *target* cells. A probabilistic guarantee is achieved that the *target* cells can be reached before its *penetration time* by at least one of the robots. However, the task of patrolling is not an object of interest.

The existing works under the ambit of MRPP have varied both in scope as well as the extent to which they address the task. A few axes along which the state-of-the-art techniques can be distinguished are as follows -

- Implementation - Online ([8, 6, 12]) vs Offline ([5], [11], [1])
- Environment Description - Perimeter ([9], [1]), Area ([3]), Graph ([5], [6])
- Problem Setting - Adversarial ([1], [17]) vs Non-adversarial ([2], [4])
- Coordination - Centralized ([11], [1]) vs Decentralized ([6], [3])
- Approach - Graph Theoretic ([5], [2]), Heuristic ([8]), Task Assignment ([6], [10]), Negotiation Based ([3]), Learning Based ([12], [16])
- Results - Deterministic ([11], [5]) vs Probabilistic ([1], [4])

- Bounds and Limits - Theoretical ([5], [11], [1]) vs Statistical ([12], [6], [16])

In the last decades, many IoT applications are emerging out, and it is indeed the future of communication that has transformed Things (Objects) of the real world into smarter devices. There are multi-agent scenarios that are being explored to take advantage of the IoT framework. The similarities in dealing with the interconnection of multiple agents or devices enable such an opportunity [Dahling'2021'Springer'AAMAS]. Multiple UAVs are explored as edge IoT devices [Sied'2021'IEEE'IoT] for restoring network under emergency situations. Edge IoT concept is further used for offloading computations [Shen'2019'ACM'Sensor'Network] from the data intensive centralized computations. Investigation of MRPP from IoT gives an opportunity to explore decentralized and secured implementation of local (range-based) communication protocol. The information exchange about visited along with visiting areas, and accordingly, decision making for each patrolling agent are the key for effective patrolling. The IoT devices will make it easier, secured, deployable and fault-tolerant solution in real scenarios. We can use the same IoT network for smart city monitoring [14] or managing any other natural or man-made catastrophe that warrants action to save lives and to protect property, public health, and safety. To the best of our knowledge, no work has analysed the MRPP from IoT perspective and provided meaningful information on system performance due to faulty devices.

The decentralized approach for solving MRPP has a possibility of deployment over large area with secured communication links between IoT Devices. The direct implementation of existing patrolling strategies under decentralized methodology is to be explored and a detailed comparative analysis of these direct implementations in IoT framework is fruitful to quantify system performance for effective patrolling.

3 Problem Formulation

We represent the patrolling environment using a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where each junction is represented by a node $v \in \mathcal{V}$ and each road segment from junction v to junction w by an edge $(v, w) \in \mathcal{E}$ ¹. For a node v , we denote the in-coming neighbours by $\mathcal{N}_{in}(v) = \{w | (w, v) \in \mathcal{E}\}$ and the out-going neighbours by $\mathcal{N}_{out}(v) = \{w | (v, w) \in \mathcal{E}\}$.

We use the time interval between consecutive visits to the nodes by patrolling vehicles as an evaluation criteria. For this purpose, we use the notion of *Idleness* (introduced in [8]) at time t defined as follows:

- The time elapsed since the last visit to a node $v \in \mathcal{V}$ by any patrolling vehicle is referred to as its *Instantaneous Idleness* (or simply, *Idleness*) and is denoted by $I_v(t)$.

¹We assume, for simplicity, that there exists only one road segment going from junction v to junction w

- The *Node Average Idleness* (or simply, *Node Idleness*) $\bar{I}_v(t)$ of a node $v \in \mathcal{V}$ is given by,

$$\bar{I}_v(t) = \frac{1}{t} \int_0^t I_v(\tau) d\tau$$

- The *Graph Instantaneous Idleness* $I_G(t)$ is given by,

$$I_G(t) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} I_v(t)$$

- The *Graph Average Idleness* (or simply, *Graph Idleness*) $\bar{I}_G(t)$ is given by,

$$\bar{I}_G(t) = \frac{1}{t} \int_0^t I_G(\tau) d\tau$$

Now, the patrolling objective is stated as follows: *For a given environment (represented by $\mathcal{G}(\mathcal{V}, \mathcal{E})$) with N patrolling agents, minimize the Graph Average Idleness $\bar{I}_G(t)$ over the duration of patrol.*

We consider two sets of IoT devices in their minimal hardware requirement form for enabling on e-to-one communication for security -

1. **Junction Device:** These are devices placed at each node v of \mathcal{G} representing junctions in the road network. Each device has a minimal memory, processing and communication units. The memory unit stores the Idleness values of the neighbouring nodes. The on-board processor is able to carry out primitive operations like changing the values stored in memory and performing comparisons between different values. The communication unit has both transmitter and receiver to communicate with neighbouring junctions and the agent visiting the node v .
2. **Agent Device:** Each patrolling agent (vehicle) is equipped with a receiver unit which can receive instructions from the Junction devices. On reaching a particular junction, this device requests the corresponding Junction Device for the next edge to traverse. This next edge information is taken as an input by the vehicle's navigation system.

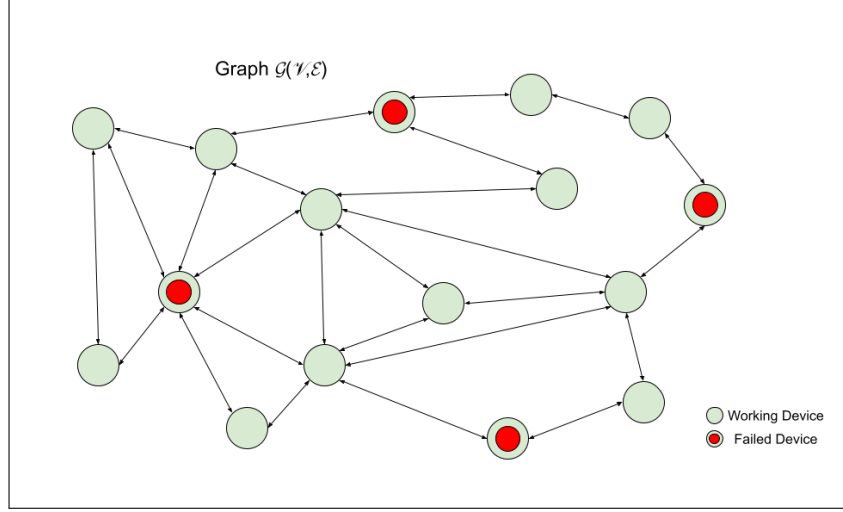


Figure 1: An illustration of directed graph representing an environment with Junction Devices placed at every nodes. Failed Junction devices are marked in red. Arrows show direction of traversal

In the suggested novel decentralized setting (more details in Section 4) wherein the decision making units are the stationary Junction Devices placed at the nodes. For each node $v \in \mathcal{V}$, the Junction Device at v maintains the Idleness of node v and nodes in $\mathcal{N}_{out}(v)$ based on agents' visits to these nodes. The information exchange between nodes is as follows - The Junction Device placed at node v receives data from $\mathcal{N}_{out}(v)$ and sends data to $\mathcal{N}_{in}(v)$. Such that each Junction Device obtains information about all of its out-going neighbours. This is relevant as the patrolling agents visiting a node can go only to one of its out-going neighbours.

A failed Junction Device implies that it is not able to communicate either with those placed at neighbouring nodes or with the agents in its vicinity. Such failures can occur due to power failures or malfunctioning of the devices. It is important to accommodate for such failures in an algorithm which will be applied in realistic settings. Furthermore, analysing the effect of a few failed devices gives the possibility to reduce the number of IoT devices in the environment, thereby avoiding redundancy of IoT devices, if communication limits permit. In our analysis, we also consider situations where the Junction Devices fail (refer to Figure 1).

ADD

4 MRPP-IoT Algorithm

In this section we explain the proposed patrolling strategy - *MRPP-IoT Algorithm*. It is a decentralized strategy with the IoT devices placed at nodes in the

environment as well as on the patrolling vehicles. These devices communicate amongst each other locally (recall from Section 3). Algorithms 1 and 2 correspond to parts of the algorithm running on Agent Devices and Junction Devices respectively. Figure 2 describes the communication protocol between these devices when a patrolling vehicle arrives at a node. The scenario is as follows - patrolling agent (say) a arrives at node v after traversing the edge (u, v) . The in-coming neighbours of v are $\mathcal{N}_{in}(v) = \{u, w\}$ whereas its out-going neighbours are $\mathcal{N}_{out} = \{u, w, x\}$. Hence, the agent cannot come to node v directly after visiting node x . When the patrolling agent a visits the node v , it requests (refer to Algorithm 1) the corresponding Junction Device for the next node amongst the outgoing neighbours $\mathcal{N}_{out}(v)$ (refer to Algorithm 2). First, the Agent Device a broadcasts the message *NodeVisit*, to which the Junction Device at v (if active) responds with *NextNode*. The Junction Device at v also broadcasts another message *NeighbourVisit* which is received by its incoming neighbours (nodes u and w in Figure 2).

Figure 2 describes the communication protocol between these devices when a patrolling vehicle arrives at a node. When a patrol agent (say) a visits a node (say) v , it requests (refer to Algorithm 1) the corresponding Junction Device for the next node amongst the outgoing neighbours $\mathcal{N}_{out}(v)$ (refer to Algorithm 2). First, the Agent Device a broadcasts the message *NodeVisit*, to which the Junction Device at v (if active) responds with *NextNode*. The Junction Device at v also broadcasts another message *NeighbourVisit* which is received by its incoming neighbours (nodes u and w in Figure 2).

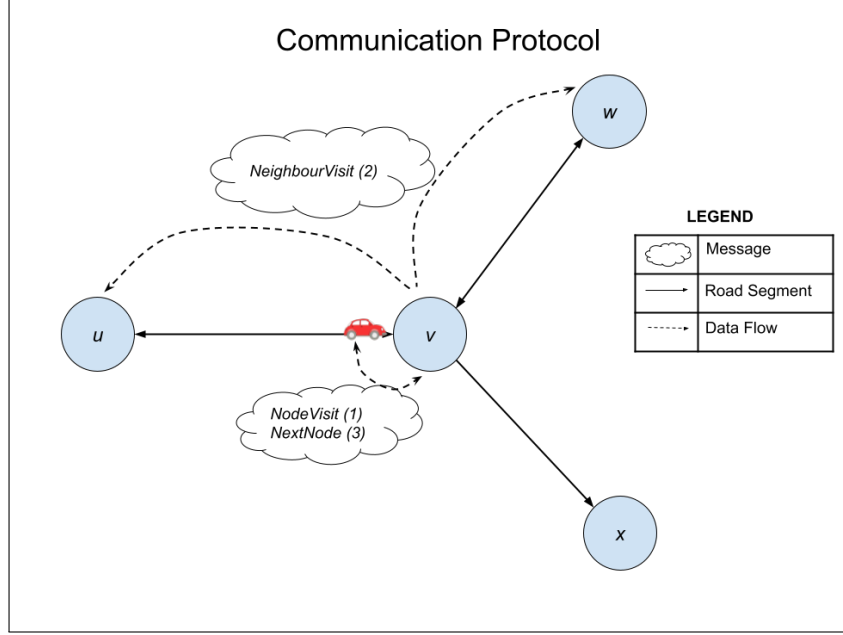


Figure 2: Communication protocols when a patrol agent a visits node v . The numbers in parenthesis besides the messages represent the order in which each of them is broadcast.

In Algorithm 1, we describe the functioning of Agent Devices placed on patrolling vehicles. At the start of the patrol, we assume that the vehicles are placed at the junctions. Consider an agent a at junction v . The Agent Device in a broadcasts a message (Line 2) $NodeVisit(a, v, t)$ and waits for instructions (Line 3) from the Junction Device at v . Here, t denotes a 's time of arrival at v . The Junction Device responds (Line 5) with $NextNode(a, v, u)$ where $u \in \mathcal{N}_{out}(v)$ is the out-going neighbour that a visits next (Line 6). We also have a fail-safe (Lines 8 - 11) if the Junction Device is malfunctioning. If the agent doesn't receive $NextNode$ within a reasonable time (that is, if *timeout*), the Agent Device selects one of the out-going neighbours at random (Line 9) and the vehicle moves to that neighbour.

Algorithm 1 Agent Device(a)

```
1: upon event reach Node( $v$ ) do
2:   send NodeVisit( $a, v, t$ )
3:   wait for NextNode
4: upon event receive NextNode( $a, v, u$ ) do
5:   traverse ( $v, u$ )
6: upon event timeout do
7:    $w \leftarrow \text{sample } (\mathcal{N}_{out}(v))$ 
8:   traverse ( $v, w$ )
```

In Algorithm 2, we describe the functioning of Junction Devices placed at the junctions in the environment. Consider the device placed at junction v . The internal clock is set to $t_c = 0$ (Line 2). We initialize a $|\mathcal{N}_{out}(v)| + 1$ sized vector I with zeroes (Lines 3-4). Each entry $I(w)$ gives the Idleness of node $w \in \mathcal{N}_{out}(v) \cup \{v\}$.

As described before, when agent a reaches junction v , it broadcasts *NodeVisit*(a, v, t). When the Junction Device at v receives this message (Lines 5-14), it responds with an out-going neighbour (say) $u \in \mathcal{N}_{out}(v)$ that the agent is supposed to visit next. First, the Junction Device updates the vector I appropriately (Lines 6-8). Since, $(t - t_c)$ duration has elapsed, each entry in I is incremented by $(t - t_c)$. Then, the entry $I(v)$ is set to zero as agent a has visited node v . It, then, determines the out-going neighbour (say) v_{max} with highest Idleness value² and sends out *Next Node*(a, v, v_{max}) (Lines 10-11). Finally, the Idleness value of node v_{max} is set to zero (Line 12). This ensures that no two agents are assigned the same out-going neighbour consecutively. Finally, it broadcasts a message (Line 13) *NeighbourVisit*(a, v, t) which is received by its in-coming neighbours.

When an out-going neighbour u is visited, Junction Device at v receives a message *NeighbourVisit*(a, u, t) signifying that agent a visited node u at time t (Lines 15-21). The Junction Device then updates its I vector (Lines 16-19) to account for this visit.

²if there are multiple nodes with equal Idleness values, one is selected at random

Algorithm 2 Junction Device(v)

```
1: procedure INITIALIZE
2:    $t_c \leftarrow 0$  ▷ Current Time
3:   for  $w \in \mathcal{N}_{out}(v) \cup \{v\}$  do
4:      $I(w) \leftarrow 0$ 
5:   upon event receive NodeVisit( $a, v, t$ ) do
6:     for  $w \in \mathcal{N}_{out}(v) \cup \{v\}$  do
7:        $I(w) \leftarrow I(w) + (t - t_c)$ 
8:      $I(v) \leftarrow 0$ 
9:      $t_c \leftarrow t$ 
10:     $v_{max} \leftarrow \max_{w \in \mathcal{N}_{out}(v)} I(w)$ 
11:    send NextNode( $a, v, v_{max}$ )
12:     $I(v_{max}) \leftarrow 0$ 
13:    send NeighbourVisit( $a, v, t$ )
14:  upon event receive NeighbourVisit( $a, u, t$ ) do
15:    for  $w \in \mathcal{N}_{out}(v) \cup \{v\}$  do
16:       $I(w) \leftarrow I(w) + (t - t_c)$ 
17:    if  $u \in \mathcal{N}_{out}(v)$  then
18:       $I(u) \leftarrow 0$ 
19:     $t_c \leftarrow t$ 
```

5 Performance Study using Simulation Results

5.1 Simulation Setup

5.1.1 Simulator

We investigate the performance of the MRPP-IoT algorithm based on simulations carried out on ‘Simulation of Urban MObility’(SUMO) [7], a realistic traffic simulator. It is an open source, highly portable, microscopic and continuous traffic simulation package designed to handle large networks. We assume empty road scenarios with patrolling agents as the only vehicles traversing the road network. The Junction Devices are mimicked by software programs running concurrently and communicating with each other via Robot Operating System (ROS) [quigley2009ros]. The communication between these software programs and the agents in SUMO is done via Traffic Control Interface (TraCI), provided in SUMO.

5.1.2 Graph Layouts

The software program *netedit*, supplied with SUMO, is used to create the test layouts. Three representative graphs as shown in Figure 3 are used for analysing the results. Graph A is a symmetric graph with 25 nodes and equal edge lengths. Graph B also has 25 nodes but varying edge lengths with some uniformity.

Graph C is asymmetric with 28 nodes and random edge lengths. Each edge on every graph is bidirectional, that is, for any pair of connected nodes, the agents can traverse in either direction.

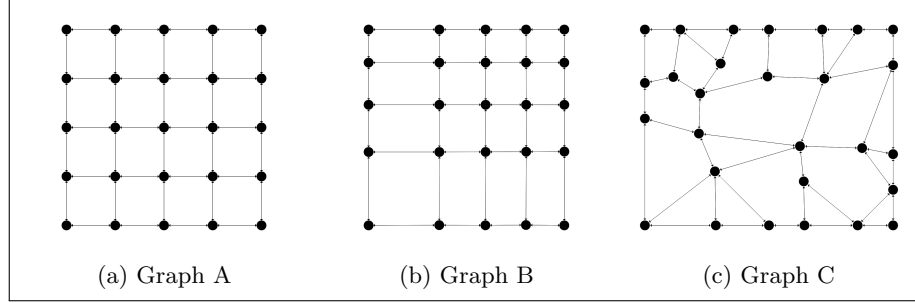


Figure 3: Different layouts used for simulations

5.1.3 Patrolling Strategies for Comparison

As base cases, we chose two strategies from [8] - *Conscientious Reactive* (CR) and *Reactive with Flags* (RwF). When at a node, agents under Conscientious Reactive strategy choose the neighbour with the highest ‘perceived’ Idleness value. As agents do not communicate with each other, they do not know the ‘true’ Idleness values of the nodes. Whereas, agents under Reactive with Flags can communicate with each other and hence have access to ‘true’ Idleness values of the nodes. CR and RwF act as perfect benchmark strategies for our work since we assume - (i) no communication amongst agents (directly), (ii) reactive decision making (that is, paths are planned one edge at a time).

In order to compare these strategies under the problem settings with failed Junction Devices, we modify them such that the agents get no information about nodes corresponding to the failed Junction Devices. We refer to the modified strategies, henceforth, as - *CR-Junction* and *RwF-Junction* respectively.

5.1.4 Simulation Parameters

We generate a total of 540 simulations with each simulation running for 30000s (CPU Time). For every graph layout in Figure 3, we run 60 simulations under each patrolling strategy (MRPP-IoT, CR-Junction, RwF-Junction). The number of patrolling agents in each run as well as the failed Junction Devices was fixed throughout the duration of that simulation. For every combination of graph layout and strategy, we had 15 runs each with 1, 3, 5 and 7 agents respectively. These 15 runs were further divided into 5 bins of 3 repetitions each with each bin corresponding to one of 0, 7, 12, 18 and 25 failed Junction Devices. In summary we had, $3(\text{Repetitions}) \times 5(\text{Device Failures}) \times 4(\text{Agents}) \times 3(\text{Strategies}) \times 3(\text{Graphs}) = 540$ simulation runs.

At the beginning of each simulation, each agent is spawned randomly at the nodes and we assume Idleness value of each node is zero, as if they have

just been visited. The patrol agents can attain a maximum velocity of 10 m/s, acceleration of 1 m/s², and deceleration of 5 m/s². Graph Idleness value is computed as Average of Instantaneous Graph Idleness over 30000s.

5.2 Result Analysis

5.2.1 Graph Idleness under MRPP-IoT Algorithm

Figure 4 illustrates the performance of MRPP-IoT algorithm under all the parameter settings. On each graph layout, the Graph Idleness value increased with increase in number of failed Junction Devices irrespective of the number of patrolling agents in the simulation. We also observed that the amount of increase in Graph Idleness reduced under higher number of agents. The Junction Device failure impacts highly for single agent scenarios (with Graph Idleness value increasing by a factor of $1.5\times$ to $2\times$ for 5 device failures). For 3 agents or more, the performance decays very gradually as the number of device failures increase. For Graphs A and B (refer to 3), 25 failed devices implies that the agents execute random sampling at each and every node. For simulations with 7 agents, the Graph Idleness value under the random sampling strategy is approximately $2\times$ that under MRPP-IoT algorithm executed at all nodes (that is, with 0 failed Devices).

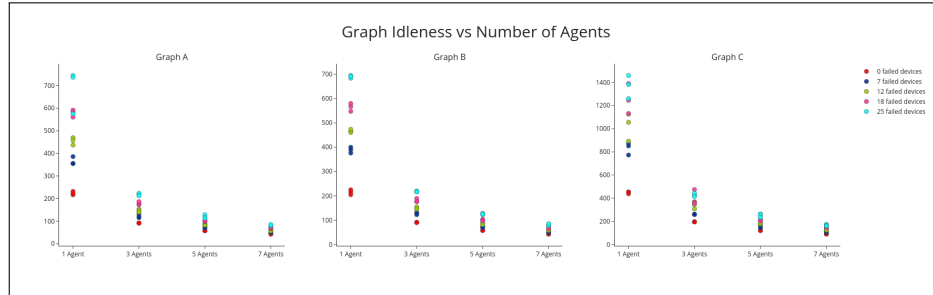


Figure 4: The plots of Graph Idleness values (in seconds) of each simulation runs under MRPP-IoT algorithm with different number of agents as well as different combinations of failed Junction Devices.

5.2.2 Temporal Analysis of Instantaneous Graph Idleness

Figure 5 contains plots of Instantaneous Graph Idleness values for a set of simulation runs on Graph C (refer to 3). We observed that even though the Instantaneous Graph Idleness varied over short periods of time, in the long run the values remained bounded in a particular range. In particular, with 0 Junction Device failures, the Instantaneous Graph Idleness varied in a cyclic manner implying that the agents' paths converged to cyclic patterns. As failure of Junction Device leads to random sampling at the corresponding node (refer to Algorithm 1),

the agents' path cannot converge to cyclic patterns and hence the Instantaneous Graph Idleness value is chaotic with time.

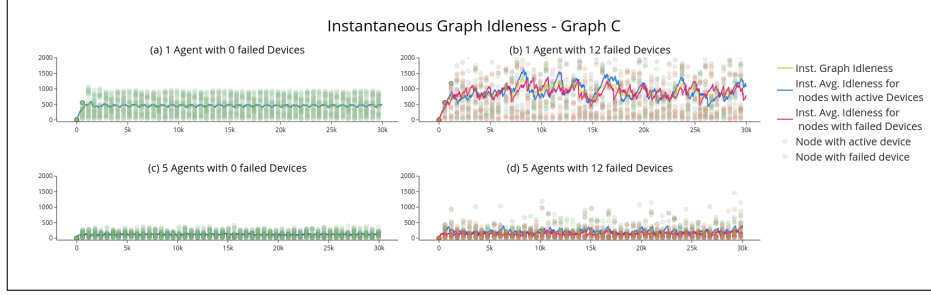


Figure 5: The plots of Instantaneous Graph Idleness values (in seconds) for simulation runs on Graph C under MRPP-IoT algorithm with (a) 1 Agent and 0 failed Devices, (b) 1 Agent and 12 failed Devices, (c) 5 Agents and 0 failed Devices, and (d) 5 Agents and 12 failed Devices.

5.2.3 Node Idleness under MRPP-IoT Algorithm

In Figure 6³, we depict the Node Idleness values observed for simulation runs on Graph C (refer to 3). The Node Idleness values of those in the center was consistently lower than those on the boundaries of graph, irrespective of number of agents as well as the combination of failed Junction Devices. The same was observed for each and every simulation run irrespective of the graph layout, the number of patrolling agents or the combination of failed Junction Devices.

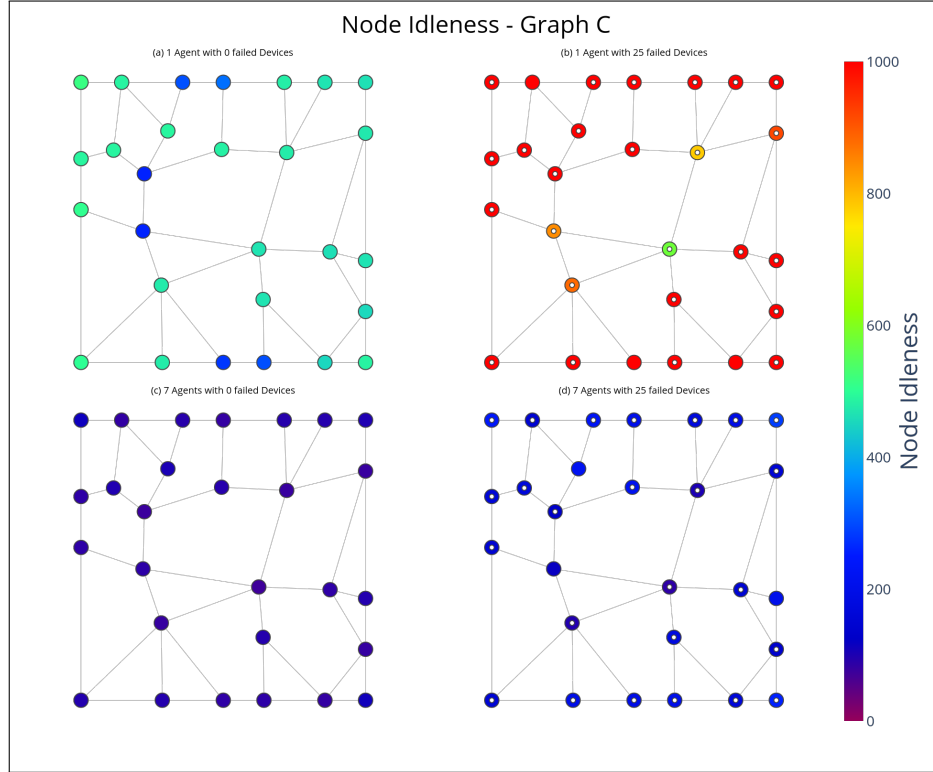


Figure 6: The plots of Node Idleness values (in seconds) for simulation runs on Graph C under MRPP-IoT algorithm with (a) 1 Agent and 0 failed Devices, (b) 1 Agent and 25 failed Devices, (c) 7 Agents and 0 failed Devices, and (d) 7 Agents and 25 failed Devices.

³In subplot (b), the Node Idleness values at some nodes reached more than 2000 seconds. For clarity and ease of comparison across the 4 plots, we have limited the color range to 1000 seconds

Table 1: Table of Graph Idleness values averaged over 3 simulation runs for each combination of simulation parameters.

# Agents	Graph A			Graph B			Graph C	
	RwF-Junc.	CR-Junc.	MRPP-IoT	RwF-Junc.	CR-Junc.	MRPP-IoT	RwF-Junc.	CR-Junc.
0 Junction Device failures								
1	239	220	223	224	223	215	444	
3	119	120	91	118	120	90	450	
5	78	79	57	78	79	57	201	
7	59	60	42	59	60	42	156	
7 Junction Device failures								
1	9041	10522	365	7747	6931	382	6931	10522
3	4920	6369	122	3800	5314	127	4496	6369
5	4008	7926	72	3890	4532	71	5948	7926
7	5606	2881	51	3211	3048	50	4670	2881
12 Junction Device failures								
1	1701	4780	454	2053	1197	466	2452	4780
3	1427	1551	146	1213	659	148	2014	1551
5	925	709	84	932	753	84	1545	709
7	1489	920	56	1078	1061	58	1076	920
18 Junction Device failures								
1	764	891	578	932	782	564	1732	891
3	431	1272	178	225	256	180	780	1272
5	125	225	100	323	140	100	636	225
7	91	137	68	285	482	67	379	137
25 Junction Device failures								
1	693	584	685	677	697	689	1708	685
3	215	230	218	207	223	217	530	230
5	120	217	119	122	121	125	285	217
7	80	80	81	82	80	80	186	81

5.2.4 Comparison with state of the art strategies

As mentioned previously, we use Conscientious Reactive and Reactive with Flags patrolling strategies as base cases for studying our algorithm’s performance. In Table 1 we have compiled the Graph Idleness values for every combination of simulation parameters we have tested on. Except for the case with 25 Junction Device failures, MRPP-IoT algorithm out performs both Reactive with Flags as well as Conscientious Reactive strategies. In fact, for 7, 12 and 18 Junction Device failures, both RwF-Junction and CR-Junction strategies fail spectacularly in comparison with MRPP-IoT algorithm. This is caused as under MRPP-IoT, the active Junction Devices account for the intent of patrolling vehicles (refer to Algorithm 2). They reset, in their internal memory, the Idleness value of the agent’s next node of visit to zero. This minor modification accounts for significant improvement in performance under cases with Junction Device failures.

For 0 Junction Device failed and more than 1 patrolling agents, MRPP-IoT algorithm performs better than RwF-Junction and CR-Junction consistently. While for the single agent case the three strategies are indistinguishable and hence the difference in values noticed in Table 1 is mainly due to varying initial conditions.

Figure 7 depicts the Node Idleness values for simulation runs on Graph C with 3 patrolling agents. The distribution of Node Idleness values under both RwF-Junction as well CR-Junction is almost identical to that under MRPP-IoT. Specifically, the Idleness of nodes in the center is less compared to those on the boundaries.

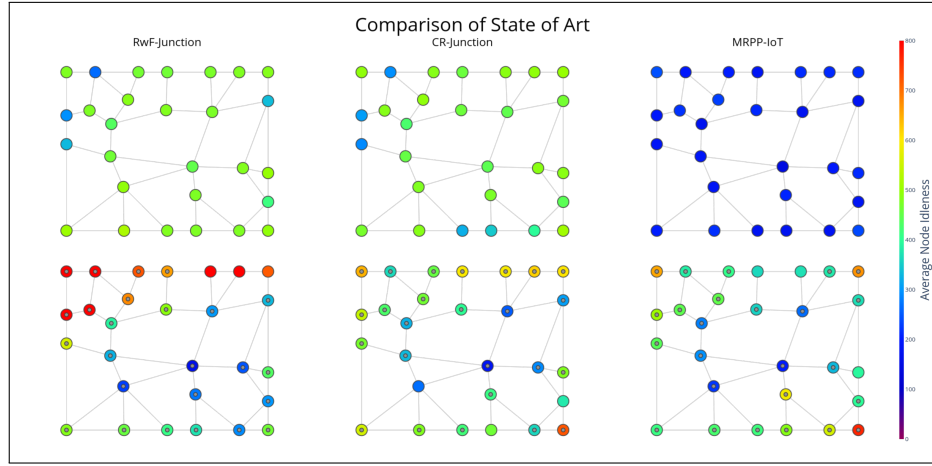


Figure 7: The plots of Node Idleness values (in seconds) for simulation runs on Graph C with 3 agents patrolling under different strategies. Plots in the top row are cases with 0 device failures while those in the bottom row are cases with 25 device failures.

6 Conclusion

Through this paper, we present a novel approach to Multi Robot Patrolling Problem which we believe is more practical for city wide patrolling. We set-up a network of Junction Devices placed at every junction in the environment. These Junction Devices can be low-cost modular IoT devices which have minimal storage, basic computing powers and short range communication capabilities. We deviate from the conventional perspective of decentralized strategies with agent based decision making, as seen commonly in literature (refer to Section 2), and suggest the Junction Device based local decision making. The communication between Junction Devices and Agent Devices is assumed to be primitive (one message per decision) (refer to Section 3), our strategy can also be implemented in an existing large scale patrolling setup involving human driven patrol agents by substituting the Junction Devices as decision makers rather than the po-

licemen, thereby utilizing the benefit of both human intelligence gathering capabilities and bias-free objective decision making capabilities of an automated system.

In Section 4, we have explained our proposed patrolling strategy ‘Multi Robot Patrolling Problem - Internet of Things’ (MRPP-IoT) algorithm. This algorithm consists of two parts - (i) Agent Device part and (ii) Junction Device part (refer to Algorithms 1 and 2) running concurrently on patrolling agents and junction devices respectively. The Junction Devices interact with each other locally to obtain *Idleness* information in their neighbourhood and then carry out reactive decision making providing the patrol agents with next node to visit.

We implement MRPP-IoT algorithm on a realistic simulation framework using SUMO and ROS. SUMO provides us with realistic motion simulation of patrolling agents while we use ROS to setup communication between different Junction Devices and Agent Devices. The performance study is presented in Section 5. We use two base case strategies - Conscientious Reactive and Reactive with Flags (RwF) to benchmark our algorithm’s performance. We analyze our algorithm’s decay in performance to Junction Device failures and observe that with increasing number of patrolling agents, the strategy becomes more robust. MRPP-IoT outperforms both CR as well as RwF strategies under full communication as well as cases with failed Junction Devices. In fact, CR as well as RwF strategies fail completely under device failures (refer to Table 1 justifying the need for MRPP-IoT Algorithm).

The current work can be extended along multiple directions. A direct follow-up would be implementation using real-life autonomous robots. We have studied Junction Device failures.

References

- [1] Noa Agmon, Sarit Kraus, and Gal A. Kaminka. “Multi-Robot Adversarial Patrolling: Facing a Full-Knowledge Opponent”. In: *Proceedings of the Journal of Artificial Intelligence Research 42 (2011)*. 2011, pp. 887–916.
- [2] Noa Agmon, Daniel Urieli, and Peter Stone. “Multiagent patrol generalized to complex environmental conditions”. In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. Aug. 2011.
- [3] Mazda Ahmadi and Peter Stone. “A multi-robot system for continuous area sweeping tasks”. In: *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, 2006*. May 2006, pp. 1724–1729.
- [4] Nicola Basilico, Nicola Gatti, and Federico Villa. “Asynchronous multi-robot patrolling against intrusions in arbitrary topologies”. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. July 2010, pp. 1224–1229.
- [5] Y. Chevaleyre. “Theoretical analysis of the multi-agent patrolling problem”. In: *Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004. (IAT 2004)*. Sept. 2004, pp. 302–308.

- [6] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. “Distributed on-line dynamic task assignment for multi-robot patrolling”. In: *Autonomous Robots* 41.6 (2017), pp. 1321–1345.
- [7] Pablo Alvarez Lopez et al. “Microscopic Traffic Simulation using SUMO”. In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. 2018. URL: <https://elib.dlr.de/124092/>.
- [8] Aydano Machado et al. “Multi-agent Patrolling: An Empirical Analysis of Alternative Architectures”. In: *Multi-Agent-Based Simulation II*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 155–170.
- [9] Alessandro Marino et al. “Behavioral control for multi-robot perimeter patrol: A finite state automata approach”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2009, pp. 831–836.
- [10] Charles Pippin, Henrik Christensen, and Lora Weiss. “Performance based task assignment in multi-robot patrolling”. In: *Proceedings of the 28th annual ACM symposium on applied computing*. 2013, pp. 70–76.
- [11] David Portugal and Rui Rocha. “MSP algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning”. In: *Proceedings of the 2010 ACM symposium on applied computing*. 2010, pp. 1271–1276.
- [12] David Portugal and Rui P. Rocha. “Cooperative multi-robot patrol with Bayesian learning”. In: *Autonomous Robots* 40.5 (June 2016), pp. 929–953. ISSN: 1573-7527. DOI: 10.1007/s10514-015-9503-7.
- [13] Jerry H Ratcliffe et al. “The Philadelphia foot patrol experiment: A randomized controlled trial of police patrol effectiveness in violent crime hotspots”. In: *Criminology* 49.3 (2011), pp. 795–831.
- [14] P. Sakhardande, S. Hanagal, and S. Kulkarni. “Design of disaster management system using IoT based interconnected network with smart city monitoring”. In: *2016 International Conference on Internet of Things and Applications (IOTA)*. 2016, pp. 185–190. DOI: 10.1109/IOTA.2016.7562719.
- [15] Amilcare Francesco Santamaria et al. “An IoT Surveillance System Based on a Decentralised Architecture”. In: *Sensors* 19.6 (2019). ISSN: 1424-8220. DOI: 10.3390/s19061469. URL: <https://www.mdpi.com/1424-8220/19/6/1469>.
- [16] Hugo Santana et al. “Multi-agent patrolling with reinforcement learning”. In: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*. 2004, pp. 1122–1129.
- [17] Noga Talmor and Noa Agmon. “On the power and limitations of deception in multi-robot adversarial patrolling”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. 2017.