Intel Serv Robotics (2009) 2:81–93
DOI 10.1007/s11370-009-0035-x

ORIGINAL RESEARCH PAPER

Towards load-balanced de-congested multi-robotic agent traffic control by coordinated control at intersections

D. V. Karthikeya Viswanath · K. Madhava Krishna

Received: 20 January 2008 / Accepted: 19 February 2009 / Published online: 15 March 2009

Abstract This paper presents a methodology for the coordination of multiple robotic agents moving from one location to another in an environment embedded with a network of agents, placed at strategic locations such as intersections. These intersection agents, communicate with robotic agents and also with each other to route robots in a way as to minimize the congestion, thus resulting in the continuous flow of robot traffic. A robot's path to its destination is computed by the network (in this paper, 'Network' refers to the collection of 'Network agents' operating at the intersections) in terms of the next waypoints to reach. The intersection agents are capable of identifying robots in their proximity based on signal strength. An intersection agent controls the flow of agent traffic around it with the help of the data it collects from the messages received from the robots and other surrounding intersection agents. The congestion of traffic is reduced using a two-layered hierarchical strategy. The primary layer operates at the intersection to reduce the time delay of robots crossing them. The secondary layer maintains coordination between intersection agents and routes traffic such that delay is reduced through effective load balancing. The objective at the primary level, to reduce congestion at the intersection, is achieved through assigning priorities to pathways leading to the intersection based on the robot traffic density. At the secondary level, the load balancing of robots over multiple intersections is achieved through coordination between intersection agents by communication of robot densities in different pathways. Extensive comparisons show the performance gain of the current method over existing ones. Theoretical analysis apart from simulation show the advantages of load-balanced traffic flow over uncoordinated allotment of robotic agents to pathways. Transferring the burden of coordination to the network releases more computational power for the robots to engage in critical assistive activities.

Keywords Wireless network · Distributed computation · Multi-agents · Multiple robots · Robot traffic control

D. V. K. Viswanath (✉) · K. Madhava Krishna
International Institute of Information Technology, Hyderabad, India
e-mail: viswanath@research.iiit.ac.in

1 Introduction

Network-mediated robot navigation has become popular [1–3] in recent years from different viewpoints. First, the network acts as a computing medium thereby reducing the computational payload on-board the robot. In a manner akin to swarm robotics, where each of the individual entity has limited intelligence but the group in itself behaves as a sufficiently intelligent system, the network allows the robots to be possessed with minimal decision-making capabilities but the network plus the robot behaves as a system of enhanced intelligence. Second, the network provides for fault tolerance capabilities for if the on-board sensors fail or misbehave the robotic agent can look up to the network for information about the environment. Third, the network supplements the computational capacity of the robot. Efficiently designed sensor fusion algorithms can agglomerate intelligence gathered through on-board as well as off-board resources to come up with robust decisions. The essential novelty of this work is that, among the survey of papers on a similar theme, the authors have not come across the one that provides for multi-robotic traffic control in a world mediated by a network. While single robot navigation mediated by a network is well studied [1,2], there has been little in the area of multi-robotic navigation. The performance gain of this method over existing methods of traffic coordination is also reported.

In this paper, we describe the problem of coordinating multiple robots by network agents, placed at critical locations such as intersections and T-junctions, such that the robots are guided to their destinations in the fastest way. The paper is divided into two parts. The first part deals with the primary layer; where the robot movements are computed such that the time delay/congestion at the intersection is reduced. The flow of traffic is coordinated by assigning priorities to pathways[1] based on the robot density and its rate of change. In this method the robotic agent hailing from the pathway having the highest agent density and lowest rate of change is allotted the highest priority and the paths of agents with lower priority are attuned to accommodate the paths of the robot with higher priority. We compared our method with the popular method of reservation [5,6] which has been dovetailed to the current situation. In this method, a robotic agent requests for a space–time allocation at the intersection. The intersection agent allows passage for the robot till the point of no conflict in the path of the vehicle through the intersection. Extensive comparisons between the two methods have been done and the performance gain of the proposed method is well illustrated in the graphs and tables shown at the end of the first part of the paper.

The second part of paper deals with the secondary layer where routing of a robot from its initial position to its destination in the best available path is done. Individually each robot's path in the environment is distributedly computed by the network as a sequence of waypoints to the goal, each successive waypoint one hop less than the previous. Here, the waypoints are the intersections and a single hop-distance is the distance between two intersections. In the proposed method, intersection agents coordinate to propagate the robot traffic density information to their neighbors so that at each intersection, the agent guides a robot towards the next waypoint which promises minimum time delay. Simulations have shown, by this method uniform traffic load is maintained at all intersections and no particular intersection is overwhelmed with robots while others are free of them. Theoretical analysis apart from simulation show the advantages of load-balanced traffic flow over uncoordinated allotment of robotic agents to pathways.

In the area of multi-agent traffic control, the work reported in [4,5] is relevant here. However, the difference being in the current method coordination at intersections is achieved by considering the density and rate of change of density along the incoming pathways to an intersection while [4,5] relies largely on a system of reservation of grids at an intersection based on a first come first served like policy. In [6–8], a mechanism for coordination between various intersection agents through an evolutionary agent paradigm was presented. The chief advantage of this method over purely multi-agent-based traffic controllers is that the intersection agents have data that can model more accurately the density and rate of change of it along a pathway. This is because robotic agents interact with intersection agents at a more basic and active level and hence the obtained data can be used more profitably for reducing time spent at intersections when compared with methods such as [4,5] as the simulation section reveals. Moreover, in this method of traffic control, the network takes care of the entire routing of the robots with little involvement of the robots themselves.

The paper is organized as follows. Section 2 deals with the formulation of the problem with all the assumptions clearly stated. Section 3 details the proposed strategy at primary layer. All the comparisons, results and graphs related to the primary layer are listed at the end of Sect. 3. Section 4 describes the secondary layer of the strategy along with the simulation results.

2 Problem formulation

Given: A planar world embedded with a network of intersections agents. Robot agents crisscross this world. The map of the environment is unknown to the robots.

Objective: Guiding the robots to their respective destinations. During the process of navigation, they often cross intersections. The objective is to have the intersection agents coordinate the traffic such that the sum over the time spent by each robotic agent to reach its destination is reduced.

Assumptions:

(a) Number of robotic agents is not fixed and they can be introduced in any pathway till such time there is no place to spawn any further due lo lack of space or congestion in that pathway.
(b) The intersection agents are capable of localising robots in their proximity based on the strength of signal received from the robot.
(c) The motion of robot agents is modeled as integer multiple of the resolution of a cell for every time sample. The cell distance is such that the agents can modify their kinematics to move by that distance or multiple of it between any two time samples. The time interval between successive samples is the same throughout.
(d) Each intersection agent is programmed to store the IDs and directions of its surrounding intersection agents that are one hop count away from it, and the maximum capacity in terms of robots of each of the pathways that lead to it.
(e) Each intersection agent is programmed to store the estimated freeway capacity (the number of robots that could be accommodated), length of each pathway that lead into it.

[1] A pathway is a segment connecting two intersections.

Assumption (a) is often used in agent community [5,6]. It serves as a yardstick for evaluating the control mechanism. It is a welcome assumption more than anything.

Assumption (b) is routinely used in sensor network community [1,2] to detect the event, if the robot has come close enough to a sensor mote to send the next action from the mote.

Assumption (c) is used to reduce the search space over the possible velocities of robots. It is once again a common theme in several discrete time optimization problems that involve discretizing a large state space in both multi-robotic [9] and single robotic planning setting [10]. Moreover, it provides an easy way to test collision by looking for space–time overlays in cells without compromising the original philosophy of the coordination algorithm. Path discretization for collision checking is not uncommon either [9,10].

Assumption (d) comes into picture, when a robot is to be directed by the intersection agent to its next waypoint and when the intersection agent should calculate the density of robots coming its way.

Assumption (e) is used in the load-balancing algorithm.

3 Primary layer: reducing congestion at an intersection

3.1 Motivation

Consider vehicles approaching an intersection with maximum speeds and without respite/continuously. Clearly, such a situation would lead to congestion at the intersections (Fig. 1a, b) thus curbing the free flow of robots. It is an undesirable situation, as the congestion at the intersection would become a bottleneck for robot movement. The bottleneck at a single intersection could spread to other intersections as more and more robots keep getting delayed causing significant loss in efficiency and productivity of the system. If the number of robots approaching the intersection is large then there is a possibility of the worst case scenario arising in which the robots block the paths of each other and get trapped in a deadlock and hence there would not be any movement in the traffic at all. Clearly, the intersection agent needs to employ a strategy by which robot movement at the intersection is coordinated in an efficient way maximizing the throughput of the intersection.

3.2 The methodology

3.2.1 Priority ordering

The intersection agent receives messages about the robots which are approaching it from other intersection agents which have been the previous waypoints of their route. The intersection agent thus knows the density of robots approaching it in each of the pathways that lead to it. The intersection agent maintains the list of robots corresponding to a pathway; the list is updated every time a new robot comes its way. The intersection agent also calculates the rate of change in densities from the list of robots it has. Having aggregated info from all the pathways, it assigns priorities to them. First the pathways are clustered based on the density values as high-density and low-density clusters. Among the clusters with high density, the pathways are prioritised on increasing order of rate of change of density with the pathway having the slowest rate of change of density getting the highest priority. This process is repeated for clusters classified as low-density clusters. Thus, among all the pathways connecting the intersection, the pathway with highest density and lowest rate of change of it gets the top most priority, since this is a situation corresponding to congestion. Within a pathway the agents are prioritised based on their closeness to the intersection. The first n_a number of them are assigned priorities. All the agents in a lower priority pathway have priorities lower than those in a higher priority pathway.

The path of the agent with highest priority is left as such for the next T time samples if it does not collide with those already crossing the intersection. If within those T time samples, a collision is detected, say, at xth ($x < T$) time sample from the current time then path reservation is made only till the next $x - 1$ time samples and then the path of the robot from $x+1$ till T is tried to be reserved (pertaining to the same collision restrictions). Similarly, one by one, the paths of all robots are calculated in the order of their priorities. When the robot traverses its reserved path then the intersection agent

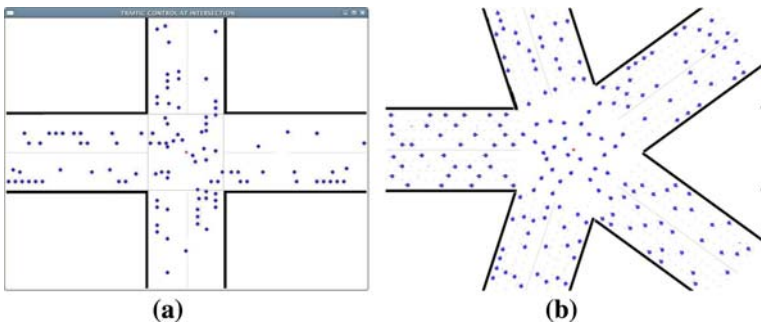

Fig. 1 Multiples robots approaching an intersection continuously leading to congestion

re-calculates the path of the robots with the current set of priorities. This process, of calculating priorities and assigning them to the robots, is repeated every τ samples, $\tau > T$ by the intersection agent. Computed path for the next T samples is transmitted to the robots by the network agent. This kind of prioritised multi-robot path planning could be efficiently implemented using search algorithms for finding a solution in the joint search space [9,11]. Algorithm 1, invoked every τ samples, details how priorities are assigned. Algorithm 2, invoked every T time samples of the robot's movement, gives the sequence of steps to calculate the path of robots.

Algorithm 1: Priority Assignment

1. Let $d_1, d_2 \ldots d_m$ be the set of robots approaching in the m pathways leading to the intersection respectively currently, at time $t = t_1$.
2. Let $dp_1, dp_2 \ldots dp_m$ be the set of robots in the pathways at time $t = t_1 - \tau$.
3. Let β be the parameter used for segregating high density pathways from low density pathways.
4. Repeat for every τ time samples.
5. Calculate the rate of change of density, rd_i = cardinality of $(d_i \cap dp_i)$, for $i = 1, \ldots m$.
6. Group all pathways that have robot densities higher than β and group the rest.
7. Assign Priorities to each of the pathways in the high density group based on their corresponding rate of change of density, rd_i. The pathways with the lowest rate of change of density gets the highest priority. Repeat the process for the low density group such that the priorities of the low density group are lower than those of the higher density group.
8. Assign priorities to robots according to the priority of the pathways in which they are. Further, the robots of the same pathway are prioritized based on their distance from the intersection. The nearer a robot is to the intersection, the higher its priority is.

Algorithm 2: Path Calculation

1. Let $a_1, a_2, \ldots, a_n$ be the set of robots that are approaching the intersection from all the pathways.
2. Let $p(a_i)$ be the priority of the robot a_i.
3. Sort $[a_i]_{i=1}^{n}$ such that $p(a_i) > p(a_{i+1})$
4. for $i = 1$ to n do
   Calculate the path of a_i and reserve the space–time, till next T samples, at the intersection to a_i wherever there are no collisions.

3.2.2 Reservation

The priority ordering method has been compared with the popular reservation method, in which priorities are assigned based on a first come first served basis. The intersection agent receives requests for space reservation. The robot whose request is first received is allotted the highest priority. The robot whose request is received next becomes the second highest and so on. Whenever a request is received, the intersection agent computes the path of that robot and sees if it is collision free. If no collisions are detected it grants the request, else it computes the path till the cell just ahead of collision. The intersection agent thus computes collision free paths based on this order and guides agents through the intersection.

3.3 Localization errors

Localization of the robots by an intersection agent is essential to the proposed methodology. However, at high speeds, localization is a challenge and hence not accurate. To overcome the risks of inaccurate localization, path reservation for each robot at each step can be made for a larger space than the robot's size. The localized position of each robot is represented as a Gaussian probability distribution, whose parameters depend on how accurately the intersection agent can localize. The values of these parameters can be set from trial runs and also updated as the system is in use. The obtained distribution determines how large a space should be reserved for each robot. This method will lead to under-utilization of the intersection area but would be successful in avoiding collisions due to localization errors. Localization at regular intervals, during the robots' movement in the intersection, makes sure large errors do not creep into the system.

3.4 Lossy communication

In a realistic scenario, messages that an intersection manager sends to the robots could be delayed or lost forever. As discussed in Sect. 3.2.1, each robot gets its path for the next few instants at regular intervals. We define a simple protocol that a robot and the intersection agent have to follow in the wake of a delayed or lost 'path' message. The robots are equipped with a reactive navigation mechanism such as in [12–14] which would make sure they avoid collisions even when not being guided by the intersection agent. The details of the scheme are not elaborated further for brevity of space. A robot which does not receive its path from the IA,[2] sends a request to the IA for a path and starts moving in the direction, in which it has to turn at the intersection, on its own. The intersection agent after receiving the request, relocalises the robot's position, calculates its path and sends the same to the robot. This method, detailed in Algorithms 3 and 4, makes sure that lossy communication does not drive the system to a halt. Robots that are forced to move on their

[2] Intersection agent.

own obstruct other robots, causing the intersection agent to process more requests resulting in delayed traffic and hence lesser throughput.

Algorithm 3: Protocol that a robot follows when it doesn't receive the path it has to traverse for the next T instants

1. Send a request message to the IA, $rq\_msg = (RID =$ robot's ID)
2. Travel in the direction of the ultimate destination (left, right or straight)
3. If detect any collisions, stop and let other robots pass
4. Repeat till a new path message, $rply\_msg = (PATH, CT)$, $CT =$ current time, is received from the IA

Algorithm 4: Protocol that an IA has to follow when it receives an $rq\_msg = (RID)$

1. Re-localise the robot with ID $= RID$.
2. Calculate the robot's path for the next T instants and broadcast the message, $rply\_msg = (PATH =$ path for next T instants, $CT =$ current time)

3.5 Simulation environment

3.5.1 The simulator

We have developed a simulator which can simulate the communication between robots and intersection agents. Using the assumptions mentioned in the earlier sections the simulator was built to model the discrete motion of the robots. The robots move with different velocities. In the simulator, the intersection area can accommodate up to 40 robot agents at a time. This number is dependent upon the size of the intersection and the size of the robots. These parameters could be accurately modeled in our simulator. For deriving the results shown we have set the size of the intersection and robots such that the number of robots which can be in an intersection is 40. However, it does not mean that 40 robots could always move in a continuous manner at the intersection, it can be done only with a proper traffic coordination policy.

When a robot's path is blocked it has to be halted and the robots that are dependent on the current robot's movement are halted as well. So the obvious metric for evaluating the two policies is the average number of robots stopped over a period of time. The more the number of robots stopped the lesser is the efficiency of the policy. Statistics were collected after a simulation of 10,000 time samples.

3.5.2 Multi-lane model

In addition to the single lane 4-way intersection simulator, we also developed multi-lane models as shown in Fig. 1b. In the multi-lane model, each pathway consists of multiple lanes with each lane assigned to those robots which will turn in the same direction at the intersection they are approaching. Hence, if the intersection is connected by four pathways, as in Fig 1a, then there would be three lanes in each pathway. If the intersection is connected by five pathways, as in Fig 1b, there would be four lanes in each of the pathway connecting the intersection. The total number of robots that can be accommodated in the intersection is increased significantly. The algorithm is modified to assign priorities to each lane of a pathway instead of the entire pathway.

3.6 Simulation results

Table 1 shows the statistics for the two policies for different look-aheads. The 'Avg' column in this table shows the

Table 1 Results obtained for different look-ahead time samples T and for different number of robots approaching the intersection

(a) Robots	(b) $T = 1$				(c) $T = 3$				(d) $T = 5$			
	Reservation		Priority		Reservation		Priority		Reservation		Priority	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
10	0	1	0	1	0	1	0	1	0	1	0	1
20	0.4	4	0.4	4	0.3	4	0.2	4	0.26	4	0.18	3
30	8.3	12	5.1	8	6.2	10	2	6	5.3	8	1.76	5
40	15.7	20	7.5	16	10.8	15	3	10	9.5	13	2.5	8
50	30.1	40	15.2	25	20.8	32	8	20	15.2	28	5.9	15
60	DL	DL	30.6	40	35	40	15.3	35	24.8	35	12	29
70	DL	DL	DL	DL	DL	DL	25	50	39.2	52	18	40
80	DL	DL	DL	DL	DL	DL	40	70	DL	DL	35	60

The Avg column shows the average number of robots that were halted in their path to accommodate robots of higher priority. The Max column shows the highest number of robots that were halted in a single time sample
DL deadlock

average number of robots that were halted in their path per sample to accommodate robots of higher priority when averaged over 10,000 samples. The 'Max' column shows the highest number of robots that were halted in a single time sample over 10,000 samples. Column 1 shows the numbers of robots approaching the intersection from the various pathways (here four) for which the 'Avg' and 'Max' values are computed. It is evident from the statistics that the priority ordering policy fares better than the reservation policy of traffic control. The simulation tests have shown that the robot density based priority ordering policy minimizes the congestion at the intersection reducing considerably the possibility of a deadlock.

Figure 2 is a comparison graph which plots the data collected for priority ordering policy. Comparison is done between the average number of robots halted when the look-ahead time is 1, 3 and 5, respectively. Clearly the system fares better, i.e. congestion is less when motion planning is done for a higher look-ahead time. Figure 3 compares the data collected for the two control policies for the same look-ahead time. It is evident that the priority ordering policy shows a much better performance than the reservation policy. We compared the average time taken to cross the intersection by a robotic agent when the two policies are employed. Figure 4 shows the comparison graph for the same. Because of better management of congestion, the priority ordering policy routes robots across the intersection faster. Figure 5 is the comparison graph for the multi-lane 5-way intersection model.

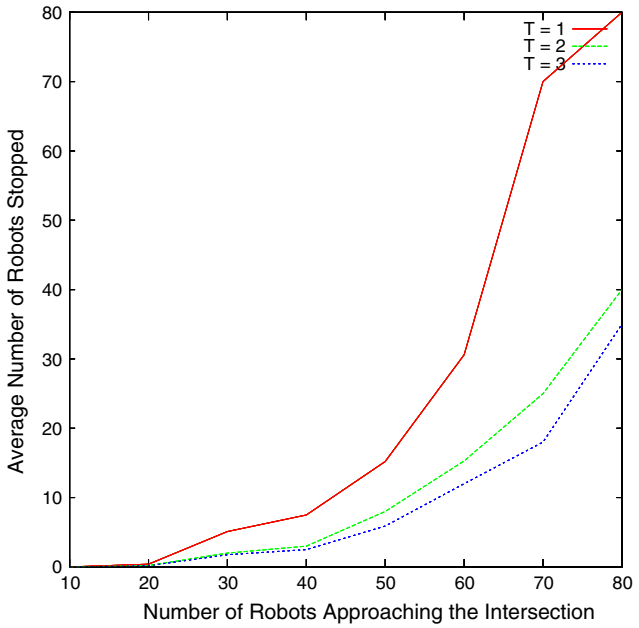

Fig. 2 Graph showing the average number of robots stopped at the intersection for different look-ahead time samples

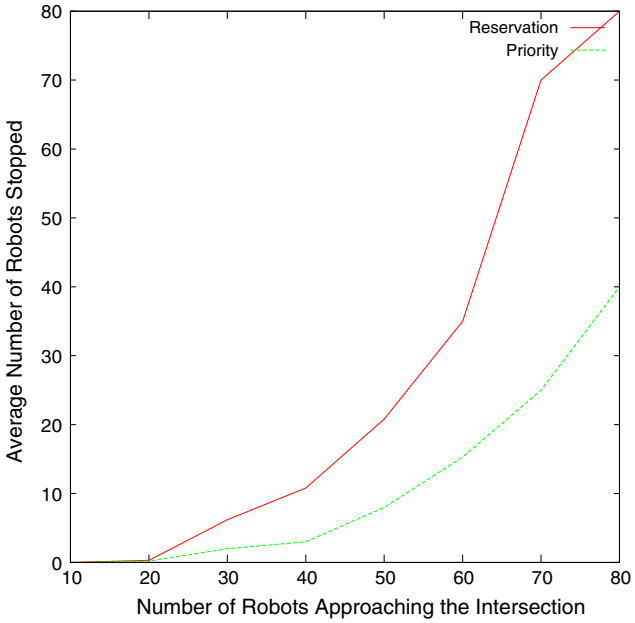

Fig. 3 Graph illustrating the efficiency of priority ordering based policy over the reservation based policy ($T = 3$)

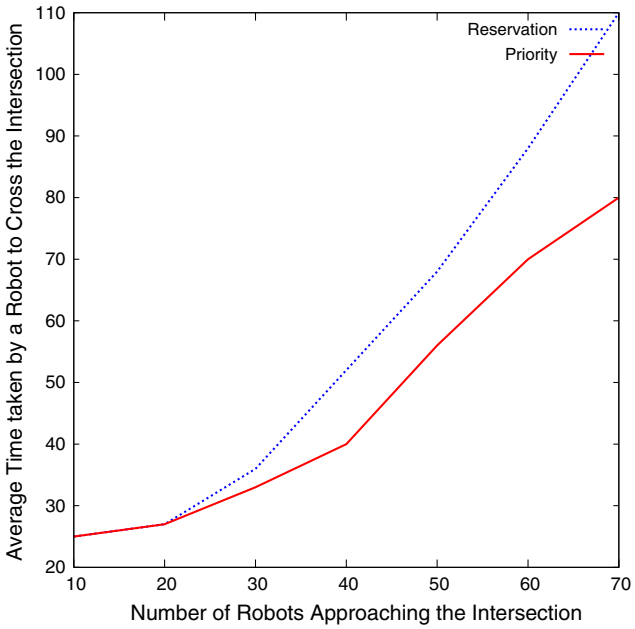

Fig. 4 Comparison of the average time taken by a robot in a 4-way intersection model

All the results discussed till now have been taken assuming ideal conditions with no errors in localization and no loss in communication. Table 2a tracks the decrease in efficiency of the system with increasing error in the localization of the robots. The path of a robot is stored as the area of the intersection that it will occupy at each time instant and hence variance in localization represents the extra area that would be deemed occupied in addition to what it truly occupies. This

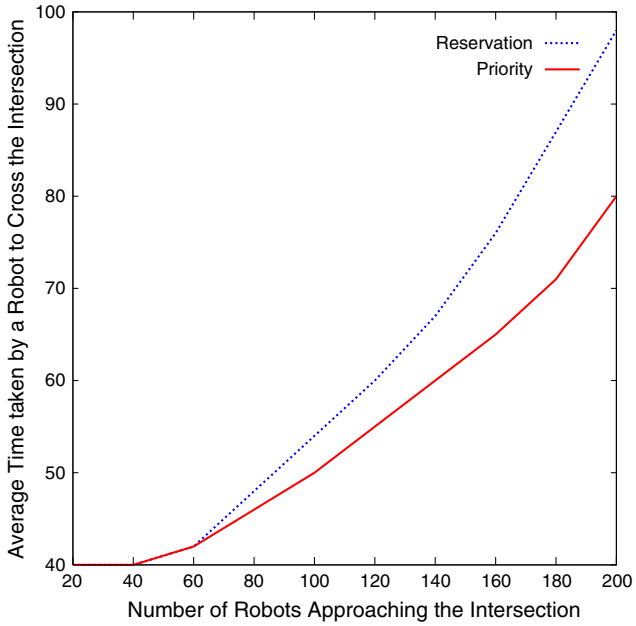

Fig. 5 Comparison of the average time taken by a robot in a 5-way intersection model

error-in localization is distributed around the robot, beyond the area currently occupied by the robot, and not just in one direction so that a realistic situation is simulated. In the simulator, the area occupied by a robot translates to the number of pixels that the simulated robot occupies. A variance of σ is then an excess of σ pixels in all directions beyond the area actually occupied by the robot. The variance column of Table 2a reflects the same. The size of a robot in our simulator is around 10 pixels. Table 2b shows the throughput of the system with increasing communication loss. The table tracks the average time taken by robots to cross the intersection as the percentage of dropped messages increases. The results were obtained over multiple simulation runs with the intersection accommodating 50 to 60 robots at all time.

As the graphs and tables indicate, our methodology works increasingly better with increasing traffic. This is because in our algorithm we incorporate the current traffic situation for assigning priorities and thus the algorithm makes sure that the robots causing congestion at the intersection are moved faster than the rest.

3.6.1 Emergency robots

Our algorithm assigns priorities to robots purely based on the current traffic movement irrespective of the type of robots that are being guided. A situation may arise when a particular robot needs to cross the intersection urgently. When such emergency robots are approaching an intersection, the intersection agent assigns highest priority to them irrespective of the priority of the pathway in which they are traveling. During our simulations, over multiple runs, we found the reduction in delay of an emergency robot to be approximately 45%, i.e. emergency robots cross the intersection at nearly half the average time taken by the rest of the robots to cross an intersection.

3.6.2 Statistical significance

We used the Student's t distribution to verify the statistical significance of our results. We compared the two policies by using the mean time taken by a robot to cross the intersection. The Research hypothesis is, 'The time taken by a robot to cross the intersection when reservation based policy is used is more than the time taken when priority based policy is employed' and thus the null hypothesis H_0: 'There is no difference between the two policies'. Table 3 shows the results used for testing the statistical significance. H_0 was rejected at a 0.001 level of significance. Thus, we conclude the advantage gained by using priority ordering is significant.

4 Secondary layer: coordination for load-balanced robot traffic

4.1 Motivation

A robot can be guided to its destination by intersection agents using the hop-count distance method [1]. The algorithm in

Table 2 System performance in non-ideal conditions

(a) Localization error			(b) Lossy communication		
Variance (no. pixels)	Mean time to cross		Messages lost (%)	Mean time to cross	
	Reservation	Priority		Reservation	Priority
0	100	80	4	102	80
4	104	83	8	110	85
9	114	90	12	124	96
16	135	107	16	142	110
25	160	133	20	165	128

Table 3 Statistical significance

	Reservation	Priority
No. of robots observed	1000	1000
Mean time to cross the intersection	100	80
SD	14.66	11.4
Variance	215	130

$t = 34.07, df = 1998, p < 0.001$

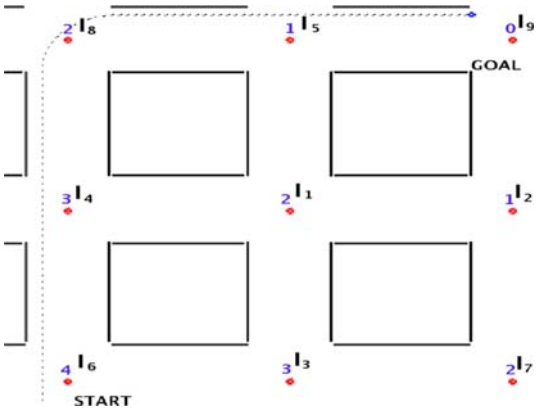

Fig. 6 The path of a single robot guided from its starting position to its goal by the network. The *dots* represent the deployed intersection agents. The *dashed-line* represents robot's path. The *numbers* displayed on top of the intersection agents are the hop-count distances of themselves from the GOAL of the robot

[1] is modified to suit the current problem. Each IA[3] is made to store its hop-count distance from all the other IAs. Packets, sent from an IA are received by other IAs that are within one hop away from the sender. The sender packet consists of the IA's ID and hop count. The receiver IA updates its hop count if the hop count of the received packet is less than its current hop value corresponding to the sender agent and only then does it broadcast it to others after incrementing the updated hop-count by 1. Thus each IA stores its hop count distance from the sender. This routine is implemented by all intersection agents. The robot when wanting to reach a destination sends a query message to the network, the intersection agent which receives this query message guides the robot in the direction of a surrounding network agent closest to the robot's destination. This process repeats till the robot reaches its destination. This can be thought as akin to the method used in motion planning to compute a global potential field that has a minimum at the goal, so that a robot can reach its goal by doing gradient descent [15]. Figure 6 shows the path of a single robot from its starting position, START to its destination, GOAL. Algorithm 5 gives the formal description of the hop-count distance method. From Fig. 6, it is evident that there are multiple paths from START to GOAL of equal lengths. If all the robots, that have the same initial positions and destinations, are routed through the same set of intersections, a mismatch in traffic load would arise with some intersections handling a large number of robots while others have significantly lesser number of robots to cater to. Ideally, all the paths of equal lengths must be exploited so that there is uniform load of traffic over all the intersections. Clearly, there is a need of a mechanism for routing the robots through the best available intersections.

Algorithm 5: Hop-Count Distance Calculation

1. Let IA be the intersection agent initiating the hop-count distance method.
2. IA broadcasts a message $msg = (ID = IA, hop(IA) = 0)$
3. for all intersection agents IA_i initialize hop count to $hop_{iIA} = \infty$
4. for all intersection agents IA_i do
5. for all received messages $m = (m_{id}, hops)$ do
6. if $hops + 1 < hop_{iIA}$ then
7. Broadcast $msg = (m_i, hop_{iIA} = hop_{iIA} + 1)$.

4.2 The methodology: pathway capacity estimation

We propose a method in which the intersection agents exchange information, about the estimated number of robots that could be accommodated, in addition to those already present in that pathway, for the next time window. The time window could be set to any number of time samples depending upon the length of the pathways. The exchanged information is used by the intersection agents for making the robot routing decisions. An intersection agent estimates the remaining capacity of a pathway based on three parameters—maximum capacity of the pathway, number of robots routed from that particular pathway in the previous time window and the current density of robots in that pathway. If in the current time window $\dot{n}$ robots have been cleared to move from the pathway then there is maximum likelihood that in the next time window also the number of robots cleared would be $\dot{n}$. With this assumption, we can predict the number of robots that can be accommodated in the pathway in the next time window.

Let N_1 be the set of robots present in the pathway at time t_1 and $c(N_1)$ be the cardinality of the set N_1

Let N_2 be the set of robots present in the path at time $t_1 + \Delta t$, where Δt is the duration of the time window

Let $\dot{n}$ be the number of robots routed from the pathway during the previous time window, from time t_1 to $t_1 + \Delta t$

Let n_c be the maximum capacity of the pathway

Let n_f be the number of robots that can be accommodated in the pathway in addition to those already present

$$\dot{n} = c(N_1) - c(N_1 \cap N_2) \quad (1)$$

$$n_f = n_c - [c(N_2) - \dot{n}] \quad (2)$$

[3] Intersection agent.

This number n_f is calculated by the intersection agent for all the pathways that lead to it. At the end of each time window, the intersection sends messages containing n_f to all its neighbors. Each intersection agent, which receives this message, stores this number and the ID of the sender. When a robot has to be guided towards its next waypoint, the intersection agent has to choose between two or more intersections which are at equal hop-count distance from the robot's goal. At such situation, the intersection agent chooses a neighbor of it which sends the maximum value of n_f, i.e. the intersection agent guides the robot towards the pathway which can accommodate more number of robots.When an intersection agent IA_1 sends a robot towards another intersection agent IA_2, that has sent the maximum value of n_f, IA_1 accordingly updates its data by decrementing the n_f corresponding to IA_2 by 1. This ensures equal distribution of robots.

If suppose each n_f of all the neighbors, which are at equal hop-count distance from the GOAL of a robot that has to be guided, is 0 which means no robots could be accommodated then the current intersection agent should re-initiate the hop-count distance algorithm by removing the intersection agents, that have n_f as 0, from the picture. Thus, the robot is re-routed along another path, albeit a longer one.

4.2.1 Analysis

The objective of load balancing is to have traffic channeled in pathways such that it is in proportion to the clearing/outgoing rates of those pathways provided the agents through those pathways reach target with same amount of distance. Through the following analysis we prove the utility of the load-balancing mechanism (Fig. 7).

Let I_1 be an intersection monitored by agent IA_1.

Let I_1 be connected by pathways to $I_2, I_3, \ldots, I_m$. Hence there are m pathways coming in and out of I_1. They are denoted as $p_{i1}, \ldots, p_{im}$ for incoming and $p_{o1}, \ldots, p_{om}$ for outgoing.

Let the incoming and outgoing rates be $r_{i1}, \ldots, r_{im}$ & $r_{o1}, \ldots, r_{om}$. Without loss of generality let $r_{o1} < \cdots < r_{om}$.

Let the net rate at which robotic agents enter I_1 be r_i & the outgoing rate be r_o. $r_i = (r_{i1} + \cdots + r_{im})$ & $r_o = (r_{o1} + \cdots + r_{om})$

Lemma 1 *If the net incoming rate exceeds outgoing rate, $r_i > r_o$, then there will be congestion in at least the slowest outgoing pathway if the trend continues for n_{f1}/n_{r1} time when agents are sent out sequentially into the pathways. Where n_{fi} is the current number of free slots in ith pathway and n_{ri} = the rate of accumulation of robots in ith pathway*

Proof r_{oi} is the outgoing rate on the ith pathway and r_{ii} is the incoming rate into the ith pathway.

$r_{ii} = r_i/m = r_{i1}, r_{i2}, \ldots, r_{im}$

Now, $r_o = \Sigma r_{oi}$
$\Rightarrow r_o > r_{o1}$
$\Rightarrow r_o/m = \Sigma(r_{oi}/m) > r_{o1}$
$\Rightarrow r_i/m > r_o/m > r_{o1}$
$\Rightarrow r_i > m \times r_{o1}$
$\therefore$ In the slowest outgoing pathway, the net increase in robots per unit time or the rate of accumulation of robots, $n_{ri} = r_{ii} - r_{oi}$
$\Rightarrow$ the time taken for ith pathway to fill up is n_{fi}/n_{ri}.
Hence, in the slowest pathway congestion occurs in n_{f1}/n_{r1} time.

In load balancing, the objective is to have the net rate at which a pathway gets filled or cleared equal or balanced across pathways. Hence pathways with fast clearing time have faster rates at which agents enter than pathways with slower clearing time.

Mathematically, this translates for any two outgoing p_{oi}, p_{oj} as $r_{ii} - r_{oi} = r_{ij} - r_{oj}$

Lemma 2 *With input rates > output rates, the time for any pathway to congest with load balancing is always longer than the time for the slowest clearing pathway when agents are added sequentially.*

Proof Consider the case where $r_{ii} - r_{oi} = r_{ij} - r_{oj} = r_{i1} - r_{o1}$
$t_i = 1/r_i$ is the time interval between arrival of two agents in incoming pathway.

Let n_{rib} be the rate of accumulation of robots in a pathway when using the load-balancing schema and n_{ris} be its counterpart when using the sequential allotment.

Now, $r_{i1} < r_i/m$

Hence, the rate at which robots enter the slowest outgoing pathway is slower than in sequential case.

$\therefore$ the net rate of accumulation of robots in slowest pathway, $(n_{r1b} = r_{i1} - r_{o1})$ is less than in the sequential allotment, i.e. $n_{r1b} < n_{r1s}$.

Hence the time taken to congest the pathway $n_{f1}/n_{r1b} = n_{fib}/n_{rib} < n_{f1}/n_{r1s}$.

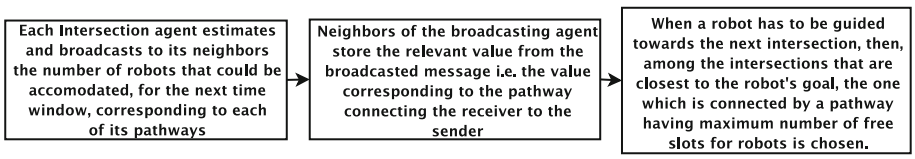

Fig. 7 A brief overview of the load-balancing methodology

Lemma 3 *The onset of congestion is always delayed by adapting load balancing than a naiver sequential allotment procedure.*

Proof Follows from Lemmas 1 and 2.

Lemma 4 *When the net input rate at an intersection is lesser than the net output/clearing rate, load balancing ensures that there will be no congestion on any of the pathway.*

Proof Consider the case where $r_{ii} - r_{oi} = r_{i1} - r_{o1} = n_{rib}$ or n_{rb}

$\therefore \Sigma_{j=1}^{m}(r_{ij} - r_{oj}) = m \times n_{rb}$

But $\Sigma_{j=1}^{m} r_{ij} = r_i < \Sigma_{j=1}^{m} r_{oj} = r_o$

$\therefore m \times n_{rb} < 0$ and hence the net rate of accumulation in any pathway in load-balanced scheme, $n_{rb} < 0$, since m is a positive integer.

Thus, there is no accumulation of traffic in any of the pathways and hence no congestion either.

Lemma 5 *Whereas even if incoming rate is less than outgoing rate, sequential allotment can still result in congestion in the slowest pathway if $t_{o1} > mt_i$ i.e. $r_{o1} < r_i/m$*

Proof Follows from Lemma 1.

The outcome of the above lemmas is to prove that load balancing is better than a naive sequential allotment procedure of outgoing traffic. It guarantees delayed onset of congestion than sequential allotment when incoming traffic is greater than outgoing. It guarantees no congestion when incoming traffic is lesser than outgoing, which can never be guaranteed with the sequential procedure.

The sequential allotment can be seen as the deterministic counterpart of random allotment. The advantages of load balancing over random allotment can be shown on similar lines by invoking probability calculus.In this paper the outgoing traffic is load-balanced between pathways that have the same hop-count distance to the goal and not over all pathways. Nonetheless the above result are equally valid for such a scenario as well.

4.3 Simulation environment

The proposed method has been simulated for a planar environment which has 20 intersection agents. Figure 8 shows a snapshot of the simulator in action. The robots are being introduced continuously and are being routed to their respective destinations. The following section shows the results obtained. We also simulated the proposed algorithm in a non-symmetric environment as shown in Fig. 9. The algorithm was modified to take into account the difference in lengths of pathways, the number of free slots (n_f) is normalized by dividing it with the length of the corresponding pathway. Also, auxiliary agents are introduced at places that are in the middle of long pathways, even though there are no intersections at those places. These auxiliary agents help in maintaining the hop-count distance metric as a true reflection of the actual distance between intersections. Figure 9 has two such agents.

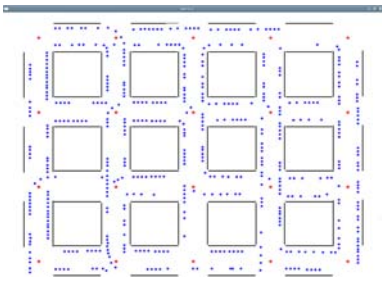

Fig. 8 Multiple robots being guided by the network

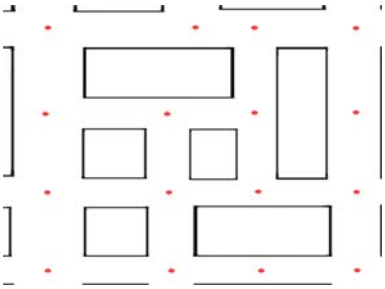

Fig. 9 A non-symmetric map used for simulation

4.4 Simulation results

Three graphs have been plotted to demonstrate the effectiveness of the second layer of the proposed strategy. The data obtained, through the implementation of the proposed strategy of coordination between intersection agents, have been compared with the data obtained when there is no coordination. Figure 10 is a graph plotting the number of robots waiting to be routed at various intersections during the course of simulation. The graph reflects the advantage gained by coordination as the robot traffic load across multiple intersections is uniform compared to the load when there is no coordination. Figure 11 is a graph indicating the amount of congestion that is being caused, in terms of the average number of robots halted at an instant, as the number of robots being introduced increases. Figure 12 compares the average time taken by a robot to reach its destination which is lower in the case of coordination. To obtain the data for this case, a robot trying to reach the farthest intersection has been chosen. Thus, the delay in reaching its destination reflects the entire congestion in the system. Both the comparison graphs were generated from data obtained through the four cases possible—when coordination mechanism is implemented at secondary layer while at the primary layer priority ordering

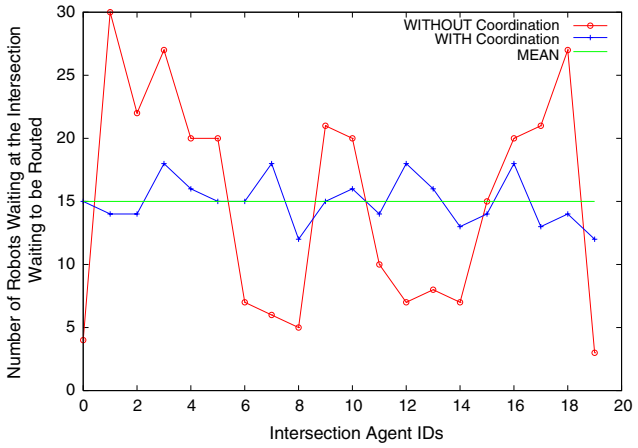

Fig. 10 Graph plotting the robot traffic load at various intersections

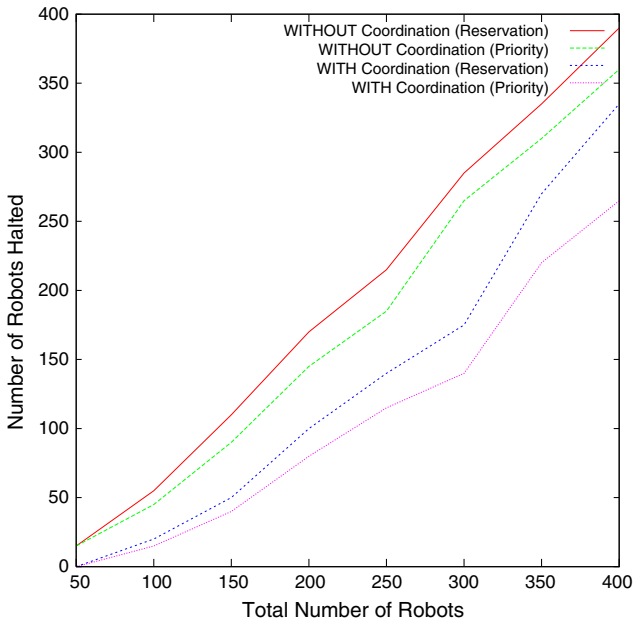

Fig. 11 Graph plotting the average number of robots halted due to congestion

policy or reservation policy is used and the remaining two cases when there is no coordination at the secondary layer. Simulation on a non-symmetric environment yielded similar results with a little degradation in performance which is expected.

4.4.1 Turning overhead

It is to be noted that the cost of turning a robot is more than the cost of sending it in a straight path. In order to balance robot traffic, the coordination mechanism sends robots into pathways that need turning on part of the robot, thus increasing the time taken by a robot to reach its destination This increase in time for some of the robots is a trade-off for

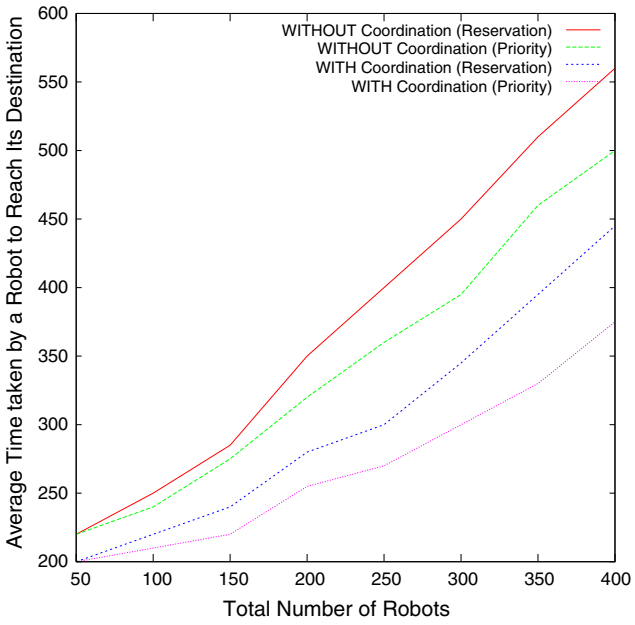

Fig. 12 Graph plotting the average time taken by a robot to reach its destination

the overall decrease in congestion of traffic across multiple intersections. Let t_r be the minimum time taken by a robot to take a right turn, assuming no other robot intercepts its path, and t_s, t_l for going straight and left, respectively. Without any loss of generality it can assumed that $t_l < t_s < t_r$. Consider Fig. 6 which shows nine intersections numbered as $I_1, \ldots, I_9$. A robot starting from I_6 to reach I_9 can be routed through any of the six routes possible with each route having a different number of right and left turns. We have simulated our algorithm for various values of t_r, t_s and t_l and from the results obtained we observed the variance between the time duration taken by robots treading those six different routes is insignificant. Table 4 reflects the same where we show the standard deviation, of time taken to reach destination, among robots taking different routes but reaching the same destination ultimately is very less. A straighter route has a higher probability of having better clearance rate, in which case, the load-balancing mechanism would prefer the straighter route anyway.

Table 4 Columns 2, 3 and 4 show the time taken by robots to complete a turn in the right, straight and left directions respectively.

No.	t_r	t_s	t_l	SD (%)
1	1	0.80	0.40	3
2	1	0.60	0.40	7
3	1	0.40	0.40	8

The values have been normalised with t_s and t_l expressed as fractions of t_r. Column 5 shows the standard deviation between the time duration taken by those robots that travelled to the same destination but in different routes

5 Communication

5.1 Messages passed

Pertaining to the algorithms we described in this paper, there are three types of messages that are passed between the agents—the messages between two intersection agents, messages sent by an intersection agent to a robot and finally the messages sent by a robot to an intersection agent. The following are the list of messages.

5.1.1 Intersection agent → intersection agent

(a) *Incoming robot.* Whenever an intersection agent sends a robot towards one of its neighbors, it follows up by sending a message containing the robot's ID and the ID of the eventual destination of the robot.
(b) *Hop-count calculation.* These messages are passed whenever the hop-count distance algorithm is initiated. Message includes the ID of the intersection agent from whom the distance is being calculated, the ID and the hop-count distance from the GOAL of the neighbor which has sent the current message.
(c) *Free slots.* This message containing n_f is sent by an intersection agent to all its neighbors at periodic intervals.

5.1.2 Intersection agent → robotic agent

(a) *Path.* An intersection agent calculates the path of each robot and sends the same to the respective robot.

5.1.3 Robotic agent → intersection agent

(a) *Destination intimation.* Whenever a robot is about to start it sends a message containing its destination to the nearest intersection agent.
(b) *Emergency.* Before it starts, an emergency robot sends its status to its nearest intersection agent so that it gets highest priority whenever it approaches an intersection.

5.2 Communication complexity

Multiple kinds of messages are exchanged constantly leading to a fairly complex communication. However, the current method succeeds in making the number of messages that a robot sends negligible because the entire computation of paths, directions etc are done by the intersection agents. This means there is less pressure on the robots to have robust communication devices. We can fairly assume that the intersection agents, which are fewer in number, are fitted with communication devices capable of sending thousands of messages in a specified time. The intersection agents exchange messages of two kinds—one, that are independent of the number of robots and the other(free slots calculation, hop-distance calculation), that are dependent on the number of robots approaching them (path calculation, incoming robot intimation). The major contribution to the communication complexity is by the second category as the first category messages are negligible in number. Table 5 gives the average number of messages sent by an intersection agent to route a particular number of robots that are approaching the intersection when the simulation is run with $\tau = 10$ and $T = 5$.

Table 5 No. of messages sent by an intersection agent

No. of robots	No. of messages
10	30
20	62
30	105
40	141
50	208
60	301
70	393
80	589

6 Conclusion

A two-layered hierarchical strategy has been proposed for coordinating robotic agent traffic by a network of intersection agents. Figure 13 shows the overview of the proposed system. The primary layer tackles the challenge of efficiently guiding the robots across an intersection using a method of allotting priorities to pathways depending on the density of robotic agents and rate of change of it in those pathways. This method coordinates traffic flow better and reaches deadlock situations for a far higher number of robots in the pathways over the existing method of reservation adapted to network framework. Robots take significantly less time to cross intersections in the current method consistently over several runs of simulation. Also, the number of robots need to be halted at the intersection is also lesser. The Secondary layer addresses the challenge of routing robots through intersections such that the traffic load is uniform all over. This has been achieved by estimating the number of robots that could be accommodated in a pathway for the next few time samples. Multiple runs over simulations have shown that our method achieves the goal of traffic load balancing. The paper describes a complete system in which multiple robots are guided to their respective destinations while minimizing the congestion. No topological map of the environment is used in the system by the robots

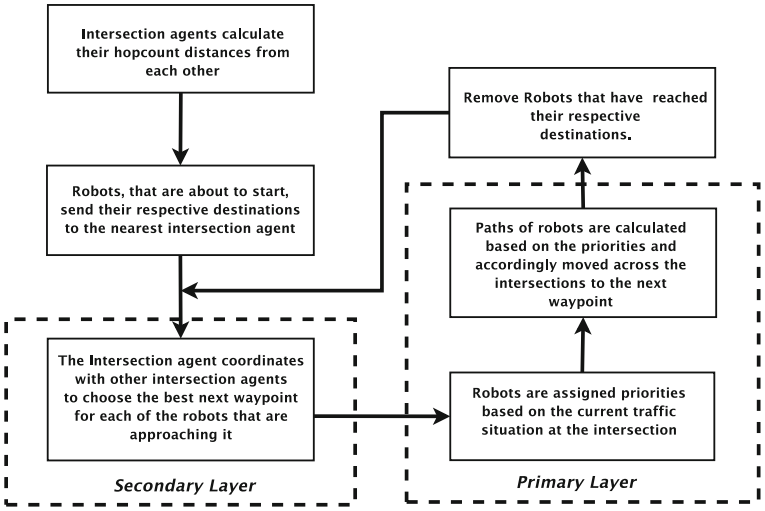

Fig. 13 An overview of the proposed system

to navigate and no intersection agent knows about the global picture of the distribution of other intersection agents.

References

1. Li Q, DeRosa M, Rus D (2003) Distributed algorithms for guiding navigation across a sensor network. In: Proceedings of the 2nd international workshop on information processing in sensor networks
2. Batalin M, Sukhatme GS, Hattig M, Robot M (2004) Navigation using a sensor network. In: Proceedings of the IEEE international conference on robotics and automation
3. Corke P, Peterson R, Rus D (2005) Localization and navigation assisted by cooperating networked sensors and robots. Int J Robot Res 24(9):771–786
4. Dresner K, Stone P (2004) Multiagent traffic management: a reservation-based intersection control mechanism. In: Proceedings of the third international joint conference on autonomous agents and multiagent systems
5. Dresner K, Stone P (2005) Multiagent traffic management: an improved intersection control mechanism. In: Proceedings of the fourth international joint conference on autonomous agents and multiagent systems
6. Bazzan ALC (2005) A distributed approach for coordination of traffic signal agents. Autono Agents Multi-Agent Syst 10(2):131–164
7. Roozemond DA (1999) Using intelligent agents for urban traffic control control systems. In: Proceedings of the international conference on artificial intelligence in transportation systems and science, pp 69–79
8. Trail Research School, Architecture of an Agent Based Urban Intersection Control System, Report (1999)
9. Bennewitz M, Burgard W, Thrun S (2002) Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. Robot Auton Syst 41(2):89–99
10. LaValle S, Kuffner J (2001) Randomized kinodynamic planning. Int J Robot Res 20(5):378–400
11. Sveska P, Overmars M (1995) Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In: Proceedings of the IEEE international conference on robotics and automation
12. Madhava Krishna K, Hexmoor H, Chellappa S (2005) Reactive navigation of multiple moving agents by collaborative resolution of conflicts. J Robot Syst 22(5):249–269
13. Minguez J, Montano L (2004) Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios. IEEE Trans Robot Autom 20(1):45–59
14. Borenstein J, Koren Y (1989) Real-time obstacle avoidance for fast mobile robots. IEEE Trans Syst Man Cybernet 19(5):1179–1187
15. Latombe J-C (1991) Robot motion planning. Kluwer Academic, Dordrecht