Predictive Routing for Wireless Networks: Robotics-Based Test and Evaluation Platform

Abolfazl Razi, Chaoju Wang, Fahad Almaraghi, Qiyuan Huang, Yuting Zhang, Hanxiao Lu, Arnau Rovira-Sugranes
School of Informatics, Computing and Cyber Security (SICCS)
Northern Arizona University, Flagstaff, AZ 86011

***Abstract*—Emerging Internet of Things (IoT) provides connectivity to a wide range of mobile nodes including indoor wireless users, pedestrian, ground robotics, vehicles, and flying objects. Such decentralized network require rethinking user-centric communication protocols which accommodate extremely dynamic environments of autonomous nodes. The authors recently proposed a predictive routing algorithm, which enables a delay-optimal communication through incorporating network topology prediction into the Dijkstra's shortest path algorithm. In this work, we extend the proposed solution to jointly optimize the end-to-end latency and total transmission power. Further, we develop a ground robotics platform in order to study the utility of the proposed algorithm in real-world applications. The simulation results which verified by the test platform, confirm the superiority of the proposed algorithm compared to the conventional shortest path algorithms by improving the delay and power consumption by a factor of 10% to 15%.**

I. Introduction

Internet of Things is an emerging technology to provide connectivity for a wide range of mobile nodes to realize smart cities [1]. The idea is to integrate furnish everyday objects with embedded control units and integrate them into a connected network with ubiquitous access [2]. This large-scale network encircles a wide range of heterogeneous nodes with different levels of autonomy, mobility, storage and data communication requirements, which demands for more efficient Machine to Machine (M2M) communications [3].

In this works, we focus on developing optimized communication protocols appropriate for large-scale networks which include freely-moving nodes with probabilistic but predictable motion trajectories. Adaptive communication is a long-lasting paradigm studies from many different perspectives. The objective of a typical adaptive communication approach is to tune communication parameters or make smarter decisions at different layers of the communication protocol (e.g. packet forwarding in network layer) such that a desired performance metric (e.g. data throughput, transmission delay, age of information, network connectivity, etc.) is optimized [4]–[6].

Majority of the previously reported network optimization frameworks take into account the current network situation in the optimization process. This is a careless ignorance of predictive information that is currently available using advanced machine learning algorithms. For instance, selecting a communication link which maximized the instantaneous performance metric at the current time but is subject to link loss during the transmission session is extremely inefficient [7]. This issue becomes more challenging in flying ad-hoc networks (FANET), where network parameters constantly change due to dynamic network topology [8].

Recently, the idea of incorporating predicted network topology into communication protocols is introduced in order to optimize the end-to-end delay in Unmanned Aerial Vehicles (UAV) networks [9], [10]. However, these works either rely on GPS information, or consider only delay optimization. In this work, we plan to extend this idea to jointly optimize the end-to-end delay and the total transmission power, since both influenced by the network topology. Further, to demonstrate the applicability of the proposed algorithm in real-world applications, we develop a test and evaluation platform based on autonomously controlled ground robotics. We choose ground robots for their motion stability and predictability [11]. However, the idea is general and applicable to other mobile objects such as drones, small satellites and airplanes [12].

The rest of this paper is organized as follows. In section II, the system model and the utilized mobility model are presented. Section III elaborates on the proposed predictive routing algorithm. In section IV, the test and evaluation platform is presented. Finally, simulation results are provided in section V, followed by concluding remarks in section VI.

II. Network Mobility Model

Consider a network of freely moving objects that communicate with high bit rate through a queued communication platform. An example is a network of UAVs that exchange video information for border patrolling. The purpose of a

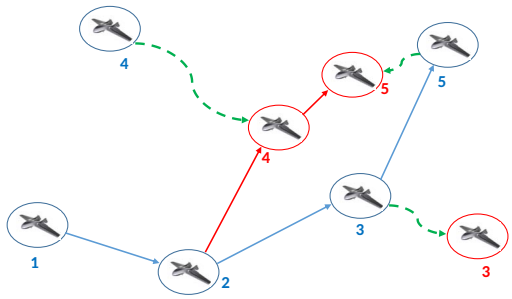


Fig. 1: Utility of predictive routing in finding optimal paths for dynamic networks.

978-1-5386-4649-6/18/$31.00

routing algorithm is to find an end-to-end communication path from the source to the destination such that a desired performance metric is optimized. In a queue-less systems with light traffic regime, the network topology does not significantly change during a short transmission session. However, in a queued system with heavy traffic regime, network topology can change significantly, while a data packet is waiting in a transmission buffer in the intermediate nodes. Thereby, the optimal path, if found by the source node based on the initial network topology using a typical shortest path algorithm, may not remain optimal. An illustrative example is shown in Fig. 1, where the blue and red circles, respectively show the original and the updated positions of the nodes (after motions shown by dashed green arrows). A conventional algorithm would determine (1-2-3-5) as the optimal path from source node 1 to destination 5 (represented by blue arrows) based on the original positions (blue circles), whereas the proposed *predictive routing* algorithm selects the path (1-2-4-5) (represented by red arrows) taking into account predicted network topology change, while the packet is waiting in the transmission buffer of node 2.

Here, the main idea is to utilize prediction algorithms and incorporate the predicted network topologies into the optimal path selection algorithm. The network is composed of N nodes uniformly distributed in a two-dimensional squared grid. Therefore, the initial node positions $\vec{l_i}(0)$=$[x_i(0), y_i(0)] \sim \mathcal{U}(-L, L)$ follow a Uniform distribution within the predefined range $(-L, L)$. Edges between nodes represent bi-way communication links in terms of a contact graph, where existence of each link following an i.i.d. Bernoulli distribution with sparsity level s, i.e. $Pr(e_{ij} = 1) = 1 - Pr(e_{ij} = 0) = s$. The edge metrics $w_{ij}(t)$ are based on the previously set up contact graph and the distances between the nodes $d_{ij}(t)$, where $w_{ij}(t) = w_{ji}(t)$. Initial velocities $\vec{v}_i(0) = [v_i^{(x)}(0)\ v_i^{(y)}(0)]$ and acceleration vectors $\vec{a}_i = [a_i^{(x)}\ a_i^{(y)}]$ are randomly generated using uniform distributions within a predefined range.

The motion trajectories can be obtained in terms of a discrete state transition model as follows:

$$\begin{cases} \vec{s}_i(k+1) & = A\vec{s}_i(k) + B\vec{a}_i(k) + \vec{w}_i(k), \\ \vec{o}_i(k) & = \gamma_i(k) C\vec{s}_i(k) + \vec{z}_i(k), \end{cases} \tag{1}$$

where we have:

$$\vec{s}_i(k) = \left[x_i(k)\ y_i(k)\ v_i^{(x)}(k)\ v_i^{(y)}(k)\right]^T, \quad \text{(state vector)}$$

$$\vec{a}_i(k) = \left[a_i^{(x)}(k)\ a_i^{(y)}(k)\right]^T, \quad \text{(input vector)}$$

$$\vec{o}_i(k) = \left[o_i^{(x)}(k)\ o_i^{(y)}(k)\right]^T, \quad \text{(observation vector)}$$

$$A = \begin{bmatrix} \vec{I}_{2\times 2} & dt\vec{I}_{2\times 2} \\ \vec{0}_{2\times 2} & \vec{I}_{2\times 2} \end{bmatrix}, B = \begin{bmatrix} \vec{0}_{2\times 2} & \vec{I}_{2\times 2} \end{bmatrix}, C = \begin{bmatrix} \vec{I}_{2\times 2} & \vec{0}_{2\times 2} \end{bmatrix}^T$$

$$\vec{w}_i(k) \sim \mathcal{N}(\vec{0}, \vec{R}_i), \quad \vec{z}_i(k) \sim \mathcal{N}(\vec{0}, \vec{Q}_i). \tag{2}$$

Here, k represents the discrete time point $k.dt$ for an arbitrarily chosen time step dt. Also, $\vec{w}_i(k)$ and $\vec{z}_i(k)$ are zero mean Gaussian distributed random vectors of covariances R_i and Q_i that respectively represent the system and observation noise terms. Finally, $\gamma_i(k)$ is a Bernoulli distributed random variable $\Pr(\gamma_i(k) = 1) = \lambda$ to capture object tracking success. The optimal prediction of locations when the measurement is absent ($\gamma_i(k) = 0$) for a known input vector $\vec{a}_i$ can be obtained using Kalman filtering with intermittent observation [13].

From motion trajectories we can determine actual node locations $\vec{l_i}(t) = [x_i(t)\ y_i(t)]$ and we assume that the location estimates and predictions $\hat{\vec{l}}_i(t) = [\hat{x}_i(t)\ \hat{y}_i(t)]$ for all neighbor nodes are available to each node n_i using a proper tracking system, where we include the prediction error term $\vec{e}_i(t) = \hat{\vec{l}}_i(t) - \vec{l_i}(t)$. Here, we take a realistic consideration that the prediction certainty declines over time. In other words, we have $e_i(t) \sim \mathcal{N}(0, \sigma^2(t)\mathbf{I}_{2\times 2})$, where the prediction noise variance σ^2 increases over time as

$$\sigma^2(t) = \sigma_0^2 + \alpha t, \tag{3}$$

where α is a predefined discount factor to capture the rise in localization uncertainty. Lastly, to model queuing delay, we assign a random waiting time to each node, $w_i(t) \sim \mathcal{U}(0, W)$. Likewise, each node has an estimate of other node's waiting time as $\hat{w}_i(t) = w_i(t) + e_{wi}(t)$, where $e_{wi}(t) \sim \mathcal{N}(\mu_w, \sigma_w^2)$ is the error measurement for the waiting time. The optimal path from the source to destination is determined based on the predicted locations and waiting times, whereas the actual end-to-end objective function calculations are made based on the actual locations and waiting times.

III. Predictive Routing Algorithms

The predictive algorithm is a modification of the well-known conventional Dijkstra's shortest path algorithm [14]. The algorithm is efficient and is the fundamental core of most shortest path algorithms. However, for scenarios where the edge metrics are time-varying, the conventional version is not appropriate, especially for queued communications.

In our network model, we use both conventional and predictive routing, respectively using initial and predicted location information to find the shortest path from the source node to the destination. The optimization goal is find the optimal path $\mathcal{P}_{\text{opt}}$, which maximizes the function in (4), where $d(\mathcal{P})$ and $p(\mathcal{P})$ represent the end-to-end delay and the total transmission power for path $\mathcal{P}$. Then, a desired importance factor γ is used to balance between the two objective functions.

$$f(\mathcal{P}) = \gamma d(\mathcal{P}) + (1-\gamma)p(\mathcal{P}) \tag{4}$$

The proposed predictive routing algorithm starts from the source node at time 0 and finds the next intermediate node in the path by including the predicted locations and estimating when the packet would reach each of the possible intermediate nodes. On this basis, it select the best intermediate node according to the lowest objective function and updates the topology at the time the packet is ready for transmission in the intermediate node. This process repeats for the intermediate node, excluding previously visited nodes, until we reach the destination. Consequently, we obtain the path which minimizes the multi-objective function. The following is the edge update procedure.

In a general formulation, we can consider edge weights as $w_{ij}(t) = f(d_{ij}(t)) + g_{ij}(t)$, where $f(d_{ij}(t))$ mimics the distance-related terms as can be the propagation delay and $g_{ij}(t)$ represents other terms, for example the waiting delay. In this paper, the goal is to minimize the transmission delay and power, represented by $f()$ in (4). To optimize the power, noting that transmission power is proportional to distance squared, we use $d_{ij}^2(t)$ as the surrogate of power in selecting the optimal path. To optimize the end-to-end delay, we consider an edge metrics $w_{ij}(t)$ that represents the delays associated with the edge e_{ij} accounting for time delays for a packet when it reaches node i denoted by t_i until the epoch it is delivered to node j, denoted by t_j. This time is composed by the waiting time in transit buffer of node i at time t_i, denoted by $w_i(t_i)$ and the actual propagation time. The propagation time for a packet that leaves node n_i at time $t_i + w_i$ and reaches the mobile node j, denoted by $p_{ij}(t_i+w_i)$ is calculated by solving the following equations:

$$\begin{cases} d_{ij} = |\hat{l}_j(t_i + \hat{w}_i + p_{ij}(t_i + \hat{w}_i)) - l_i(t_i + \hat{w}_i)|_2, \\ p_{ij}(t_i + \hat{w}_i) = d_{ij}/c, \end{cases} \tag{5}$$

where the first equations characterizes the predicted distance between node n_i (when the packet leaves this node at time $t_i + w_i$) and node n_j (when the packet reaches this node at time $p_{ij}(t_i + w_i)$). The second equations relates this distance to the propagation time $p_{ij}(t_i + w_i)$ using the wave propagation phenomenon with the speed of light c. The operator $|\vec{x}|_2 = \sqrt{x_1^2 + x_2^2}$ is the second norm of vector $\vec{x} = [x_1 x_2]$. Note that (5) may require numerical methods to solve the nonlinear equation, since $\hat{l}_j(t)$ is the solution of (1) and in general is not linear. However, noting the fact that propagation time is negligible compared to waiting times, we can use the convenience of $p_{ij} = 0$ in (5), thereby the end-to-end delay is the accumulated waiting times through the path.

Once, we solve (5) with or without the simplifying assumption, the edge metrics are determined as

$$w_{ij}(t_i) = w_i + p_{ij}(t_i + w_i), \quad t_j = t_i + w_{ij}. \tag{6}$$

The total transmission delay of a K-hop path $\mathcal{P} = (n_{i_1}, n_{i_2}, \ldots n_{i_K})$ is calculated as:

$$d(\mathcal{P}) = \sum_{k=1}^{K-1} w_{i_k i_{k+1}}(t_{i_k}) = \sum_{k=1}^{K-1} w_{i_k} + p_{i_k i_{k+1}}(t_{i_k} + w_{i_k}) \tag{7}$$

Likewise, the power can be represented by

$$p(\mathcal{P}) = \sum_{k=1}^{K-1} d_{i_k i_{k+1}}^2 \tag{8}$$

The following is the summary of the proposed algorithm.

We observe that the objective function in Algorithm 1 is the combination of the end-to-end delay and power along. Also, note that the predicted node locations $\hat{l}_j(t)$ and estimated waiting times $\hat{w}_i(t)$ are used to find the optimal intermediate nodes in the path, while actual node locations $l_j(t)$ and node delays $w_i(t)$ are used to calculate the actual delay and power metrics. In the case of linear motions, we have $B = 0$ in (1) and the state transition equations are reduced to $\vec{l}_i(t+1) = \vec{l}_i(t) + dt\vec{v}_i$+noise, which further simplifies the calculations by linearizing (5).

Algorithm 1: Optimal shortest path algorithm

Data: N: number of nodes, n_s: source, n_d: destination, $\mathcal{G} = (V, E_{ij}(t))$, $\vec{l}_i(t)$, $\hat{l}_i(t)$, $w_i(t)$, $\hat{w}_i(t)$ c
Result: $\mathcal{P}$: optimal path, $obj(\mathcal{P})$: path objective function, $d(\mathcal{P})$: path delay, $p(\mathcal{P})$: path power
(initialization):
$\{Visited\} \leftarrow \{n_s\}$; $t_{n_s} = 0$;
$\{Unvisited\} \leftarrow V \setminus \{n_s\}$; $t_n = \infty \forall n \in \{Unvisited\}$;
$n_i \leftarrow n_s$;
(finding optimal path:)
while $n_d \notin \{Visited\}$ **do**
  for $n_j \in \{Unvisited\}$ **do**
    if $e_{ij} = 1$ **then**
      update $w_{ij}(t_i)$ using (6);
      update d_{ij} using (5);
      update obj_{ij} using $w_{ij}(t_i)$ and d_{ij} in (4);
      $t_j = t_i + w_{ij}(t_i)$;
      $d_{sj} = d_{si} + d_{ij}$;
      $obj_{sj} = obj_{si} + obj_{ij}$;
    $\{Visited\} \leftarrow \{Visited\} \cup \{n_i\}$;
    $\{Unvisited\} \leftarrow V \setminus \{n_i\}$;
    update next node: $n_i \leftarrow \underset{j \in \{Unvisited\}}{\text{argmin}} \{obj_{sj}\}$;
    Previous(n_j)=n_i;
(form the optimal path:)
$\mathcal{P} = \{n_d\}$; $n = n_d$;
while $n \neq n_s$ **do**
  $n \leftarrow$ Previous(n);
  $\mathcal{P} \leftarrow n \cup \mathcal{P}$
(calculate delay:)
t=0; dist=0; obj=0;
for $n \in \mathcal{P}$ **do**
  $i = n$; $j =$ next(n);
  calculate d_{ij} using (5) by substituting $\hat{l}_j(t)$ with $l_j(t)$ and $\hat{w}_i(t)$ with $w_i(t)$;
  update $w_{ij}(t)$ using (5) and (6);
  update obj_{ij} using $w_{ij}(t_i)$ and d_{ij} in (4);
  $t \leftarrow t + w_{ij}(t)$;
  $dist \leftarrow d + d_{ij}$;
  $obj \leftarrow obj + obj_{ij}$;
$d(\mathcal{P}) = t$;
$p(\mathcal{P}) = p(\mathcal{P}) + d_{ij}^2$;
$f(\mathcal{P}) = \gamma d(\mathcal{P}) + (1-\gamma)p(\mathcal{P})$;

IV. Test and Evaluation Platform Design

In order to test the practicality of the developed Predictive Routing algorithm in practical Dynamic Networks, we develop a test and evaluation platform using ground robots. The main

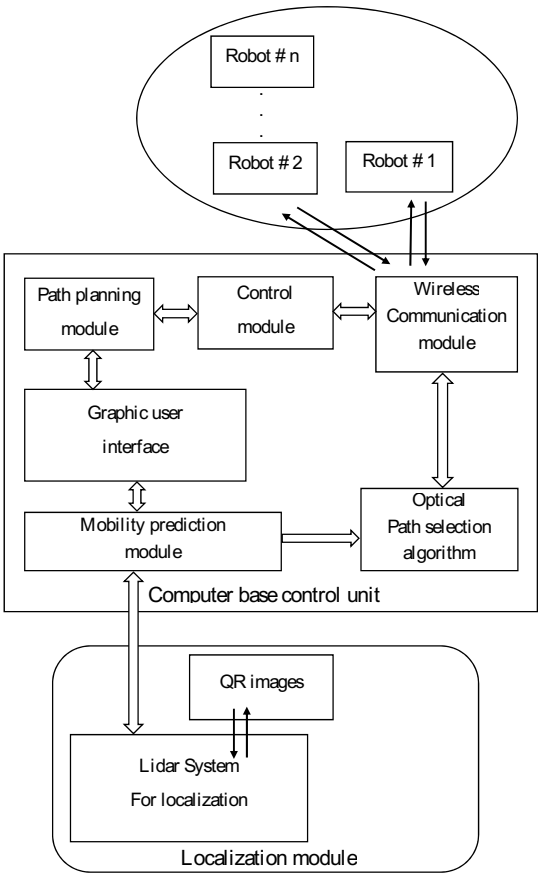


Fig. 2: Block diagram of the test and evaluation platform. Green color represents the external modules.

goal is to assess the superiority of the proposed algorithm in finding optimal paths in real scenarios compared to conventional shortest path algorithms. In order to validate the credibility of the obtained simulation results, we compare the actual end-to-end distances for an end-to-end communication extracted from the captured video against the numerical values obtained by processing the planned motion trajectories. Fig. 2 illustrates the block diagram of the proposed system.

The system consists of three main modules including computer-based control unit (CBCU), localization unit (LU), and 5 ground robots. The ground robots are three-wheel car platform equipped with two electric motors, a LiDAR based localization module, and a control and wireless communication module based on *Raspberry pi 3 Model B*.

The control unit is developed in MATLAB environment with a graphic user interface (GUI) which includes the following modules: i) path planing module (PPU) in order to program the robots motion trajectories according to the probability distributions presented in section II, ii) command and control unit (CCU) in order to convert the planned motion trajectories as well as the communication parameters into a sequence of control commands, iii) a bi-way communication module in order to send the control commands to the ground robots and receive the relevant measurements from the robots, iv) topology prediction module in order to predict network topology based on the localization information, and v) a predictive routing algorithm in order to find the optimal path from the source to the destination, as detailed in section III. Finally, we note that the localization module includes two different approaches based on QR imaging and LiDAR system, as detailed in sections IV-B and IV-C.

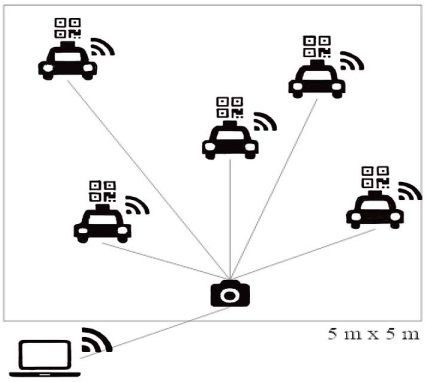
A. Communication Protocol

The information exchange occurs between the CBCU and the robots through WiFi connectivity and using TCP/IP protocol. We select TCP/IP over WiFi (IEEE 802.11 family) connectivity as opposed to other candidates such as ZigBee protocol (based on IEEE 802.15) for implementation convenience and also its capability of exchanging high data rates. In order to simulate the waiting times at intermediate nodes, we program the nodes to hold data-packets for a pre-programmed time before forwarding to the next node. In order to facilitate information exchange among the robots and CBCU, we propose to use the following template for data packets. Note that TCP/IP includes built-in per-link routing, framing and integrity check, but we include relevant fields in our packet format to make it independent from the underlying communication protocol.

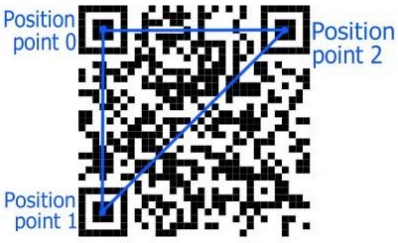
TABLE I: Communication Protocol: Unified Packet Format

Field	Values: Options
Start Flag	Fixed value: 01111110
Source Id	Unique source node ID
Destination Id	Unique destination node ID
Command	Options: Localization Info, Control command, Data Packet, ...
Length	Length of the payload data
Payload Data	Measurement information, motion trajectory information, ...
CheckSum	Module-256 addition
End Flag	Fixed value: 01111110

The *Source* and *Destination* fields include uniques IDs of the source and destination nodes. We assign ID=1,2,...,5 to the 5 robots, ID=0 to CBCU, and ID=6 to localization module. *Start* and *Stop Flags* with constant patterns are used to mark the beginning and end of the packet. The fields *Length* and *CheckSum* are used for additional integrity check. The *Source* and *Destination* fields define the two ends of a per-link communications, and it is changed per link based on the optimal path determined by the predictive routing algorithm. This information is programmed in terms of routing tables in robots at the beginning of a transmission session. The *Command* filed defines the type of the packet including i) path planning command, ii) delay programming, ii) routing update command, iii) *datapacket*, and iv) localization measurement. The Payload data has a variable length and its content depends on the command. For instance, the payload data includes the velocity and initial directions of robots for a *path planning* command. Details of commands and payload data are omitted here for the sake of brevity.



(a)



(b)

Fig. 3: Image-based identification/localization based on QR codes. (a): the localization module, (b): sample QR code.

B. QR-Code Based Identification and Localization

In this project, we use Quick Response (QR) code for both identification and localization purposes. QR codes are extensions of bar-codes into matrix format and can be used for image-based identification. Here, we attach a unique QR code printout on the upper side of each robot, representing its unique ID. A central camera pointing downwards is used 3 meters above the field level to locate the robots as shown in Fig. 3a. We use the *Zxing: zebra crossing* package developed in Python programming language to identify and locate the objects labeled with QR codes. This package provides accurate readings for three corner points of QR codes (Fig. 3b). The dimensions of the coverage area is fixed ($5m \times 5m$), and the exact position of the camera are known. Even without these information, relative distances between labeled objects can be easily found with high accuracy by scaling the distances in the captured images (video frames). If the actual distance between the QR points 0 and 1 in a QR label is d_{01} and the program output is Point 0: (x_0, y_0) and Point 1:(x_1, y_1), the distance between any arbitrary readings (x_i, y_i) and (x_j, y_j) is:

$$d_{ij} = d_{01}\sqrt{\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{(x_0 - x_1)^2 + (y_0 - y_1)^2}} \tag{9}$$

The QR readings can also be used to determine the orientation of each labeled object with respect to a reference direction. For instance if Points 0 and 2 are aligned with the reference direction (horizontal direction in Fig .3b), the relative angle θ is calculated as follows:

$$\theta = \tan^{-1}[(y_2 - y_0)/(x_2 - x_0)] \tag{10}$$

The obtained information is shared with other robots and the CBCU for predicting future locations and executing predictive routing algorithm. Note that only one central camera is used to reduce the implementation cost, but a distributed version can be implemented if one camera is deployed by each robot.

C. LiDAR-based Localization Method

The QR-based imaging provides accurate localization for both centralized and distributed methods. However, it requires high-resolution camera and image processing which increases the implementation cost. To implement low-cost solution, we plan to use laser based localization, as depicted in Fig. 4. In this approach, a reference rod is located in the center of the field. Also, a controllable turret with LiDAR transmitter and detector of type *Benewake TFMINI Micro LiDAR Module* mounted on the robot's body. The LiDAR module rotates a full circle using an embedded servo motor in order to detect and locate the reference rod. The strength of the reflected light as well as the angular phase, which maximizes the signal strength provides an accurate estimate of the robots location. Each robot share its location information with other robots as well as the CBCU in order to realize predictive communications. Note that similar to the QR-based localization, this approach is centralized, but it is easily extend-able to decentralized method if the LiDAR transceiver keep track of all surrounding objects. This approach can be integrated with QR-based imaging to realize a fully distributed joint identification and localization method. The accuracy of utilized LiDAR transceiver is the range of few millimeters within the localization 30cm to 12m.

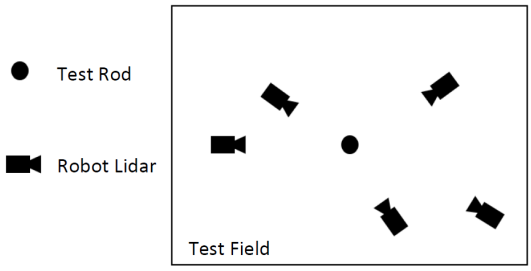


Fig. 4: LiDAR localization based on wireless networks.

Finally, Fig. 5 shows different building blocks of each robot.

V. Experimental Results

In this section, we provide simulations results as well as the practical tests in order to verify the optimality of the proposed predictive routing algorithm as well as the utility of the developed test and evaluation platform.

A. Experimental Setup

The objective of the practical test is to program robots such that they follow a pre-planned motion paths. The motion

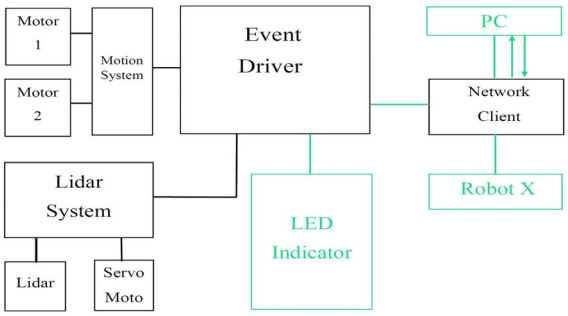


Fig. 5: Block diagram of the designed Robotic vehicle.

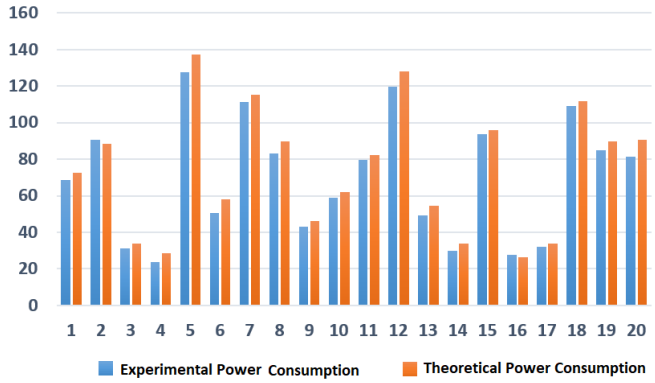


Fig. 6: Comparison of expected transmission power with the experimental values.

trajectory of each node is determined by its initial position, orientation, and velocity. These parameters are randomly generated by the CBCU in the initialization phase using probability distributions discussed in section II and sent as a set of control commands to robots using the wireless communication module. Also, waiting time for each robot is randomly generated and programmed. The robots are programmed to send packets based on the determined optimal path all the way to the final destination. Each node once receives a packet, holds it according to its programmed waiting time and relays it to the next node. Transmit, receive and waiting modes are indicated by three LEDs for easy visualization. This framework can be used to verify the optimality of the developed predictive routing algorithm based on the collected localization information in real-world applications. In order to confirm the accurate operation of different steps (path planning, robot programming, routing, and waiting time enforcement), we develop a simple scenario where two robots exchange a data packet 5 times back and forth under different scenarios, and we compare the experimental values of the link lengths during each transmission phase (obtained from the captured video) with mathematically calculated values (based on the programmed parameters). The results are shown in Fig. 6. The transmission power in this figure is calculated as $\sum_{j=1}^{|\mathcal{C}|} d_{i_j i_{j+1}}$, where $\mathcal{C} = \{i_1, i_2, \ldots, i_{|\mathcal{C}|}\}$ is the selected path which includes the ordered set of $|\mathcal{C}|$ nodes. The error between the analytically derived and practically obtained values for total power consumption across 20 scenarios is less than 1%, which ensures the accuracy of the subsequent simulation results.

B. Simulation Results

For testing the performance of the predictive routing protocol, the path planning module first generates random network topologies by defining initial positions $\vec{l_i}(t)$, initial velocities, and the acceleration profile for each nodes based on distributions presented in section II. The robots are programmed with path-planning parameters along with the randomly generated waiting time.

Different simulation scenarios include investigating the impact of different network parameters including i) the number of nodes, ii) the average node velocity and iii) the average waiting time. In particular, we are interested to see the improvement of the delay and power consumption performance for the proposed method compared to the Dijkstra's shortest path algorithm. The idea is to identify the optimal path using the predicted locations and estimated waiting times and quantify the objective function with the actual node motions and waiting times.

Fig. 7 shows the effect of the number of nodes in the efficiency of the method proposed. It is observable that as number of nodes in the network increases, the optimal algorithm exhibits a higher performance improvement in terms of delay and power utilization compared to the conventional algorithm. The reason is that, more decision making steps are involved in a larger network, and thereby there is larger margin between the two algorithms. Further, the network topology change is more extreme for larger networks, simply because it takes a longer time for a packet to reach the destination. For a network of 50 nodes, the performance of the optimal method shows about 10% improvement. For a larger network, this improvement is expected to rise.

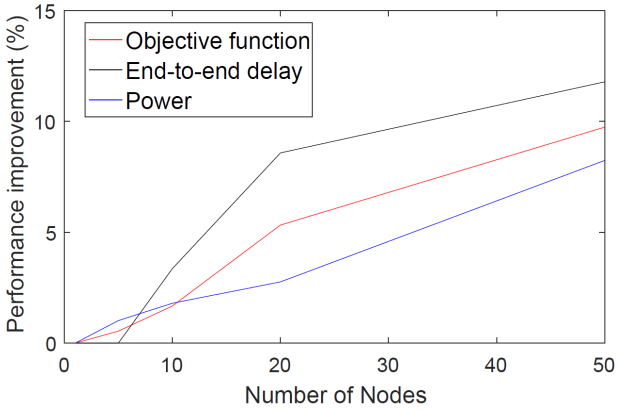


Fig. 7: The performance improvement (in percentage) by using predictive routing compared to conventional shortest path method. Objective function, the end-to-end delay and the power consumption show improvement consistently.

Next, the objective function for both methods regarding the average waiting time has been evaluated. Simulation results, as shown in Fig. 8, suggest higher performance gain for the optimal algorithm when the average waiting times are longer. As we increase the waiting time for each node, topology changes are more severe and we obtain higher gains by predicting network topology. Therefore, substantial benefits can be obtained for queued networks with heavy traffic mode.

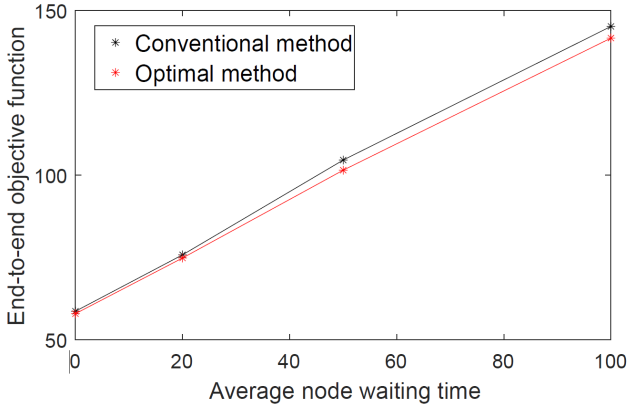


Fig. 8: Objective function for the proposed and the conventional Dijkstra's shortest path algorithms based on the average node waiting time.

Lastly, we test the performance gain with respect to the average node velocity. For more dynamic networks, where the node velocities are higher, the optimal method improves compared to the conventional, as shown in Fig. 9. The reason is that for higher node velocities the network topology is evolving rapidly and consequently, the proposed algorithm shows a better performance. For this reason, the proposed algorithm is well suited to FANET with fast-flying objects.

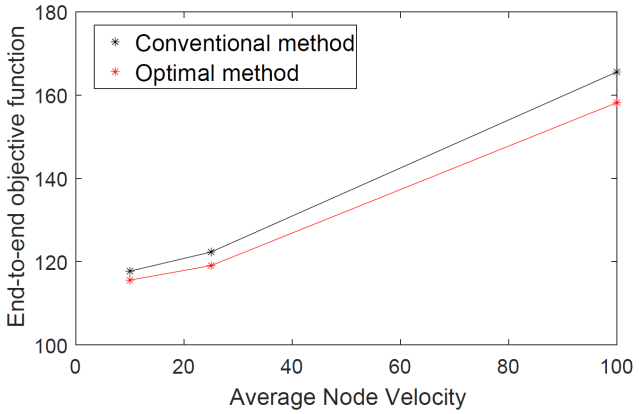


Fig. 9: Objective function for the proposed and the conventional Dijkstra's shortest path algorithms based on the average node velocity.

VI. Conclusions

In this paper, an optimal routing method based on Dijkstra's shortest path algorithm is proposed for UAV networks by incorporating predicted network topology into the path selection criterion. Using a multi-objective function, we were able to jointly optimize the end-to-end delay and the transmission power. In order to mimic the real-world prediction uncertainty, we let the prediction error rise over time. We conducted preliminary tests on ground robots to showcase the optimality and applicability of the proposed method in real-world application. Simulation results conform an improvement of about 10% for moderate network sizes. The performance gain increases for larger networks, larger average waiting time and higher node velocities.

This method can be viewed as a primary step towards developing predictive communications and we envision that much larger gains can be obtained by incorporating predictive network topology into different layers of communication protocols.

References

[1] E. Theodoridis, G. Mylonas, and I. Chatzigiannakis, "Developing an iot smart city framework," in *Information, intelligence, systems and applications (iisa), 2013 fourth international conference on*. IEEE, 2013, pp. 1–6.

[2] S.-H. Yang, "Internet of things," in *Wireless Sensor Networks*. Springer, 2014, pp. 247–261.

[3] S. Andreev, O. Galinina, A. Pyattaev, M. Gerasimenko, T. Tirronen, J. Torsner, J. Sachs, M. Dohler, and Y. Koucheryavy, "Understanding the iot connectivity landscape: a contemporary m2m radio technology roadmap," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 32–40, 2015.

[4] V. Asghari and S. Aissa, "Adaptive rate and power transmission in spectrum-sharing systems," *IEEE Transactions on Wireless Communications*, vol. 9, no. 10, pp. 3272–3280, 2010.

[5] A. Razi, A. Valehi, and E. Bentley, "Delay minimization by adaptive framing policy in cognitive sensor networks," in *Wireless Communications and Networking Conference (WCNC), 2017 IEEE*. IEEE, 2017, pp. 1–6.

[6] M. Hammoudeh and R. Newman, "Adaptive routing in wireless sensor networks: Qos optimization for enhanced application performance," *Information Fusion*, vol. 22, pp. 3–15, 2015.

[7] Y. He, W. Xu, and X. Lin, "A stable routing protocol for highway mobility over vehicular ad-hoc networks," in *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*. IEEE, 2015, pp. 1–5.

[8] O. K. Sahingoz, "Networking models in flying ad-hoc networks (fanets): Concepts and challenges," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, p. 513, 2014.

[9] A. Rovira-Sugranes and A. Razi, "Predictive routing for dynamic uav networks," in *IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, Oct 2017.

[10] S. Rosati, K. Kruzelecki, L. Traynard, and B. Rimoldi, "Speed-aware routing for uav ad-hoc networks," in *Globecom Workshops (GC Wkshps), 2013 IEEE*. IEEE, 2013, pp. 1367–1373.

[11] E. F. Flushing, L. M. Gambardella, and G. A. Di Caro, "On using mobile robotic relays for adaptive communication in search and rescue missions," in *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 370–377.

[12] A. Razi, F. Afghah, and J. Chakaresk, "Optimal measurement policy for predicting uav network topology," in *Asilomar Conference on Signals, Systems, and Computers*, Oct 2017.

[13] J. Rohde, J. E. Stellet, H. Mielenz, and J. M. Zllner, "Localization accuracy estimation with application to perception design," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 4777–4783.

[14] H. Ortega-Arranz, D. R. Llanos, and A. Gonzalez-Escribano, *The Shortest-Path Problem:Analysis and Comparison of Methods*. Morgan & Claypool, 2014.