POSE: Prediction-Based Opportunistic Sensing for Energy Efficiency in Sensor Networks Using Distributed Supervisors

James Z. Hare, Shalabh Gupta, *Member, IEEE*, and Thomas A. Wettergren, *Senior Member, IEEE*

***Abstract*—This paper presents a distributed supervisory control algorithm that enables opportunistic sensing for energy-efficient target tracking in a sensor network. The algorithm called Prediction-based Opportunistic Sensing (POSE), is a distributed node-level energy management approach for minimizing energy usage. Distributed sensor nodes in the POSE network self-adapt to target trajectories by enabling high power consuming devices when they predict that a target is arriving in their coverage area, while enabling low power consuming devices when the target is absent. Each node has a Probabilistic Finite State Automaton which acts as a supervisor to dynamically control its various sensing and communication devices based on target's predicted position. The POSE algorithm is validated by extensive Monte Carlo simulations and compared with random scheduling schemes. The results show that the POSE algorithm provides significant energy savings while also improving track estimation via fusion-driven state initialization.**

***Index Terms*—Distributed fusion, distributed sensor networks, multimodal network control, network intelligence, opportunistic sensing, prediction-based control, sensor scheduling.**

NOMENCLATURE

A	Probabilistic finite state automaton (PFSA) alphabet.
C	Number of associated tracks.
E^{s_i}	Energy consumed by sensor node s_i.
E_{net}	Energy consumed by the network.
$\mathbf{F}$	Target state transition matrix.
f	False alarm rate.
$\mathbf{H}$	Measurement matrix.
$\tilde{\mathbf{I}}$	Identity matrix.
$\hat{\mathbf{I}}^{s_i}_{\mathbf{x}}, \hat{\mathbf{I}}^{s_i}_{\mathbf{\Sigma}}, \hat{\mathbf{I}}^{s_i}_{\mathbf{W}}$	Received information ensembles.
M	Four-tuple PFSA.
N	Number of sensor nodes.
$\mathcal{N}^{s_i}$	Neighborhood of sensor node i.
$\mathcal{N}^{s_i}_R$	Set of neighbors with packet reception.
$\mathcal{N}^{s_i}_T$	Set of trustworthy neighbors.
$P^{s_i}_d(k)$	Sensor detection model.
$P^{s_i}_r$	Packet reception model.
P_{fa}	False alarm probability model.
$P^{s_i}_{\text{HPS}}$	Probability that a target is present within sensor node s_i's coverage area.
P_m	Probability of missed detection.
P_{sleep}	Probability of sleep in a random network.
p	PFSA state transition probabilities.
$\mathbf{Q}$	Process noise covariance.
R_c	Sensor node communication radius.
$R_{s,\text{HPS}}, R_{s,\text{LPS}}$	Sensing radius of high power sensing (HPS) and low power sensing (LPS) devices.
$\mathbf{R}$	Measurement noise covariance.
$\mathcal{S}$	Set of sensor nodes.
$\mathbf{S}^{s_i,c}$	Innovation covariance of estimate c.
s_i	Sensor node i.
T	Cycle time.
$\mathcal{T}$	Set of Targets.
$\hat{\mathbf{T}}^{s_i}_{\mathbf{x}}, \hat{\mathbf{T}}^{s_i}_{\mathbf{\Sigma}}, \hat{\mathbf{T}}^{s_i}_{\mathbf{W}}$	Trustworthy sets of information received.
$\mathbf{u}^{\tau_l}, \mathbf{u}^{s_i}$	Positions of target l and sensor s_i.
$\mathbf{v}^{s_i,c}$	Innovation of estimate c.
$\hat{\mathbf{W}}^{s_i}$	Filter gain matrix
$\mathbf{w}(k)$	Measurement noise.
$\mathbf{x}(k)$	Target state at time k.
$\hat{\mathbf{x}}^{s_i}$	Estimated target state.
$\mathbf{z}(k)$	Measurement vector at time k.
α	Probability of detection.
β	Probability of receiving packets.
$\mathbf{\Gamma}$	Target noise gain matrix.
δ	PFSA state transition function.
ζ	Designed weight for the $P^{s_i}_{\text{HPS}}$ function.
Θ	Set of PFSA states.
λ	Probability of staying in the *Sleeping* state.
$\Lambda^{s_i,c}$	Probability of target c in coverage area.
ξ	Threshold of trustworthy estimates.
ρ	Network density.
σ^2_v	Process noise variance.
σ^2_x, σ^2_y	Position measurement variance.
$\hat{\mathbf{\Sigma}}^{s_i}$	Estimated target covariance.
$\tilde{\mathbf{\Sigma}}^{s_i,c}$	Cross-covariance matrix between sensor.
τ_l	Target l.
$\upsilon(k)$	Model process noise at time k.
Ω	Region of interest (ROI).
ϱ^2	Track association measure.

Manuscript received August 15, 2016; revised December 26, 2016, March 22, 2017, and June 6, 2017; accepted July 2, 2017. This paper was recommended by Associate Editor R. Selmic. *(Corresponding author: Shalabh Gupta.)*

J. Z. Hare and S. Gupta are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269 USA (e-mail: shalabh.gupta@uconn.edu).

T. A. Wettergren is with Naval Undersea Warfare Center, Newport, RI 02841 USA.

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2017.2727981

I. Introduction

RAPID developments in sensing, communication, and data processing technologies have led to the emergence of distributed sensor networks (DSNs) which are used in a wide range of applications. For example, DSN are deployed for *intelligence, surveillance,* and *reconnaissance* operations [1], target tracking [2]–[4], environmental monitoring [5], personal health monitoring [6], structural health monitoring [7], [8], complex system diagnostics [9], border surveillance [10], [11], traffic monitoring [12], and many others. Furthermore, these sensor networks have evolved to consist of advanced sensor nodes, where each node is equipped with multiple heterogeneous sensing devices, a communication device, a data processing unit (DPU), and a memory unit [13]. However, a major limitation that still persists in sensor networks is that of the limited availability of energy resources at each sensor node. If a sensor node is depleted of its energy reserves, it fails to operate and collect data and thus leads to coverage gaps and missed detections. Often it is difficult and time consuming to charge or replace the failed nodes due to the large size of the ROI. This can lead to poor performance of the network and in the worst case scenario mission suspension.

Therefore, it is important that these sensor nodes utilize their different devices in an energy-efficient manner to improve the network lifetime. A simple method is random scheduling which turns the devices on and off randomly [14], thus preserving energy, while other methods perform scheduling according to a clock. In contrast, there are yet other methods that optimize the activity of the network based on the local spatio-temporally varying demand [15]. These methods fall under the category of *Opportunistic Sensing* where the devices are enabled and disabled dynamically for efficient utilization of energy resources [16]. To perform opportunistic sensing in a pro-active manner, it is desired that the network has predictive intelligence to allow the sensor nodes to prepare for an event (e.g., target arrival) in advance.

For opportunistic sensing, information must be communicated between sensor nodes; however, sharing information consumes energy [17], thus making it difficult to balance communication requirements and energy minimization. Specifically, in the multitarget tracking application, it is desired to track the targets with high power sensing devices opportunistically, i.e., to turn on the high-power sensing devices only on the nodes which are located in the region(s) where the target(s) are currently present or approaching. This requires accurate predictions of the targets' paths via distributed fusion of information received from multiple nodes detecting the targets. However, the information received may be associated to different targets [18], corrupted by noise, or provide false beliefs of targets present in the environment [19]. These issues can drastically increase the tracking error if data validation, association, and filtering are not incorporated into the DSN.

This paper addresses these issues by presenting a distributed supervisory control algorithm called *Prediction-based Opportunistic Sensing* (POSE). This algorithm manages the power consumption of each sensor node in a distributed fashion to enable energy-efficient target tracking while minimizing tracking error and missed detection rates. Each node consists of multiple devices including high power sensors, low power sensors, a DPU and a communication device. Each node also contains an embedded supervisor, i.e., a PFSA, which enables and disables the devices on the node in an opportunistic manner to conserve energy.

The states of the PFSA represent the different activities of the node which correspond to enabling/disabling its various sensing and communication devices [11], [20]. The state transition probabilities of the PFSA are dynamically updated based on the information received from neighboring nodes or from direct observation of a target using the local sensor suite. These probabilities control the switching of the PFSA states to facilitate opportunistic sensing based on the predicted target(s) trajectories. For example, the PFSA of a sensor node enables its high power consuming devices when a target is present within its coverage region, while disabling them and entering into a low power consuming cycle when it is not present.

The main contribution of this paper is the development of the POSE algorithm that is built upon distributed supervisors (i.e., PFSA) that enable scheduling of multimodal sensor nodes consisting of several sensing devices that consume different amounts of energy. The POSE algorithm is validated by Monte Carlo simulations for multiple scenarios on a MATLAB platform and compared with random scheduling schemes. The results show that the POSE algorithm provides significant energy savings and low missed detection rates while improving tracking accuracy via fusion-driven state initialization.

II. Related Work

Energy-efficient target tracking has been studied using various approaches throughout the DSN literature. To begin with, the sensor nodes must be deployed in a manner that enables full coverage of the ROI [21], [22]. This can be achieved by optimally placing sensor nodes to maximize coverage [23], [24]; however, this becomes impractical for large scale networks, in complex geographical regions, and due to the deployment time. To overcome this limitation, the k-coverage problem was studied [25] to ensure that every point within the ROI is observable by at least k sensor nodes at a given time. This assumes that the sensor nodes are deployed according to a random distribution, e.g., uniform or Poisson [2]. Once the sensor nodes are deployed, energy-efficient scheduling schemes are employed. In target tracking problems, the highest priority is accurate state estimation of the target. Therefore, adaptive sampling methods [26], [27] were developed to adjust the sampling interval to ensure accurate track estimation where the trace of the covariance matrix is used to represent the estimation accuracy.

Typical sensor scheduling approaches found in the literature are based on three types of network architectures: 1) centralized; 2) hybrid; and 3) distributed. Centralized networks require each sensor node to communicate their observations to a central computing station which calculates and transmits the optimal scheduling decisions to each sensor node while estimating the target trajectory [28], [29]. Some of these methods employ sensor selection techniques to minimize the number of sensors used for data collection [29], [30]. The centralized architectures can guarantee that the scheduling protocol

will minimize the energy consumption of the network; however, these architectures are not scalable and thus very difficult to implement on large scale networks due to multiple factors such as communication limitations, transmission delays, and data processing requirements.

Hybrid network architectures were proposed to overcome the limitations of centralized architectures by identifying a few sensor nodes throughout the ROI, to act as central computers, called cluster heads (CHs) [31], [32]. Each CH forms an independent cluster which consists of multiple sensor nodes within its communication neighborhood. These sensor nodes send all the measurement data to their CH for analysis to generate state estimates for target tracking, sensor scheduling, and even for new CH selection [15], [27], [33]. The CH performs sensor scheduling by means of an optimization algorithm which selects the optimal subset of sensors within the cluster to track the target. This requires the knowledge of the positions of all the sensors within the cluster as well as their respective energy levels. Since the information measured outside of the cluster is unobservable, scheduling decisions in hybrid networks are only locally optimal.

The CH-based methods allow for controlled energy consumption throughout the network to maximize its lifetime [34]. These methods eliminate the bandwidth limitation found in centralized networks by limiting a cluster's size to allow for single-hop communications. However, data transmission between the sensor nodes and the CH occur frequently [35], thus consuming energy. Furthermore, if a CH were to fail, due to defects or energy depletion, the entire cluster would stop receiving scheduling decisions, rendering a large gap in the coverage area of the overall network [36]. This requires implementation of dynamic CH election schemes [37].

Distributed network architectures have also been proposed where each sensor node dynamically controls its own tasks based on the information measured by its own sensor suite or perceived from its neighboring nodes [38]. The methods reported here deal with partial information broadcasting [39], adapting sensor frequency [40], and scheduling binary sensor nodes [41] for accurate target tracking with energy minimization. Unlike most hybrid approaches, the distributed networks tend to be reactive in the sense that they only communicate information when an event has occurred, thus allowing for savings on transmission costs. Also, distributed architectures do not require sensor or CH selection, thus allowing for less computation requirements. Similar to hybrid networks, the scheduling decisions here are only locally optimal due to limited observation region; however, these architectures are scalable, provide fault tolerant attributes by construction, and eliminate the bandwidth limitations of centralized networks.

The POSE algorithm developed in this paper utilizes distributed supervisors that rely on target's state predictions to probabilistically control the multimodal sensor nodes for energy minimization. This algorithm provides significant energy savings for network lifetime extension while ensuring low missed detection rates and tracking errors.

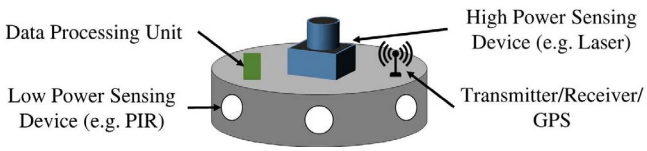

Fig. 1. Example of a sensor node.

III. Problem Formulation

The main objective of the POSE algorithm is to utilize energy resources opportunistically, i.e., only when needed, for a target tracking application. Specifically, the objective is to track targets with minimal energy consumption by enabling the high power consuming devices on any node only when a target is present or predicted to be within its coverage area. Otherwise, the high power consuming devices are disabled, but the node stays aware in low-power consuming states.

Let $\Omega \in \mathbb{R}^2$ be the ROI and let $\mathcal{T} = \{\tau_1, \tau_2, \ldots \tau_L\}$ be the set of L targets which are traveling through Ω. Let the position of a target $\tau_\ell \in \mathcal{T}$ at time step k be denoted as $\mathbf{u}^{\tau_\ell}(k) = (x^{\tau_\ell}, y^{\tau_\ell})(k) \in \Omega$. Now let $\mathcal{S} = \{s_1, s_2, \ldots s_N\}$ be the set of N sensor nodes which are randomly deployed throughout Ω to track the targets.

A. Description of Sensor Node

Each sensor node $s_i \in \mathcal{S}$, located at $\mathbf{u}^{s_i} = (x^{s_i}, y^{s_i}) \in \Omega$, contains a sensor suite of heterogeneous sensing devices, a DPU, a communication device (transmitter/receiver), and a GPS device, as shown in Fig. 1. The sensor suite includes several low power sensing (LPS) devices [e.g., passive infrared (PIR) or ultrasonic sensors] for target detection, and a high power sensing (HPS) device (e.g., a laser or a camera) for further target interrogation (e.g., accurate position measurements or high-resolution images).

The energy consumed [41] by a node s_i is given as

$$E^{s_i} = e_{\text{DPU}} \cdot t_{\text{DPU}} + e_{\text{LPS}} \cdot t_{\text{LPS}} + e_{\text{HPS}} \cdot t_{\text{HPS}} + e_{\text{RX}} \cdot t_{\text{RX}} + e_{\text{TX}} \cdot t_{\text{TX}} + e_{\text{clock}} \cdot t_{\text{clock}} \quad (1)$$

where e_j denotes the energy consumed per unit time by a certain device "j" including DPU, LPS, HPS, receiver (RX), transmitter (TX), and clock; and t_j denotes the total time duration for which the device is on. The total energy consumed by the network is computed as

$$E_{\text{net}} = \sum_{i=1}^{N} E^{s_i}. \quad (2)$$

Definition 1 (Neighborhood): The neighborhood of a sensor node $s_i \in \mathcal{S}$ is defined as

$$\mathcal{N}^{s_i} \triangleq \left\{ s_j \in \{\mathcal{S} \setminus s_i\} : \left\| \mathbf{u}^{s_j} - \mathbf{u}^{s_i} \right\| \leq R_c \right\} \quad (3)$$

where R_c is the communication radius of the node.

B. Target Motion and Measurement Models

The motion of a target, τ_ℓ, is modeled using the *discrete white noise acceleration* model [42] as follows:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{\Gamma}\boldsymbol{\upsilon}(k) \quad (4)$$

where $\mathbf{x}(k) \in \mathbb{R}^n$ is the target state at time k, $\mathbf{F}$ is the state transition matrix, $\mathbf{\Gamma}$ is the noise gain matrix, the process noise $\boldsymbol{\upsilon}_k$ is zero-mean white Gaussian sequence and the covariance of the process noise multiplied by the gain is $E[\mathbf{\Gamma}\boldsymbol{\upsilon}(k)\boldsymbol{\upsilon}(k)'\mathbf{\Gamma}'] = \mathbf{\Gamma}\sigma_\upsilon^2\mathbf{\Gamma}' = \mathbf{Q}$, where σ_υ^2 is the process noise variance of the position estimates. The elements of $\mathbf{x}(k)$, $\mathbf{F}$, and $\mathbf{Q}$ depend on the state estimation algorithm. In this paper, three different state estimation algorithms are used: 1) the Kalman filter (KF); 2) the extended KF (EKF); and 3) the interacting multiple models (IMMs) filter.

For the KF algorithm, a linear target motion model is used and the target state is defined as $\mathbf{x}(k) \triangleq [x(k), \dot{x}(k), y(k), \dot{y}(k)]'$. This model assumes that the target is traveling in a straight line and the state transition matrix and process noise covariance are given as [42]

$$\mathbf{F} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{4}T^4 & \frac{1}{2}T^3 & 0 & 0 \\ \frac{1}{2}T^3 & T^2 & 0 & 0 \\ 0 & 0 & \frac{1}{4}T^4 & \frac{1}{2}T^3 \\ 0 & 0 & \frac{1}{2}T^3 & T^2 \end{bmatrix} \sigma_\upsilon^2 \tag{6}$$

where T is the cycle time between state transitions.

For the EKF algorithm, a nonlinear motion model is used and the target state is augmented with the turning rate, $\phi(k)$; thus $\mathbf{x}(k) \triangleq [x(k), \dot{x}(k), y(k), \dot{y}(k), \phi(k)]'$. This model assumes that the target is traveling with a constant turning rate to accommodate for curved trajectories. The state transition matrix and process noise covariance are given as [42]

$$\mathbf{F} = \begin{bmatrix} 1 & \frac{\sin(\phi(k)T)}{\phi(k)} & 0 & -\frac{1-\cos(\phi(k)T)}{\phi(k)} & 0 \\ 0 & \cos(\phi(k)T) & 0 & -\sin(\phi(k)T) & 0 \\ 0 & \frac{1-\cos(\phi(k)T)}{\phi(k)} & 1 & \frac{\sin(\phi(k)T)}{\phi(k)} & 0 \\ 0 & \sin(\phi(k)T) & 0 & \cos(\phi(k)T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$\mathbf{Q} = \begin{bmatrix} \frac{\sigma_\upsilon^2}{4}T^4 & \frac{\sigma_\upsilon^2}{2}T^3 & 0 & 0 & 0 \\ \frac{\sigma_\upsilon^2}{2}T^3 & \sigma_\upsilon^2 T^2 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sigma_\upsilon^2}{4}T^4 & \frac{\sigma_\upsilon^2}{2}T^3 & 0 \\ 0 & 0 & \frac{\sigma_\upsilon^2}{2}T^3 & \sigma_\upsilon^2 T^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\phi^2 T^2 \end{bmatrix} \tag{8}$$

where σ_ϕ^2 is the process noise variance of the turning rate.

For the IMM state estimator, the KF and the EKF estimators are run in parallel and combined to utilize both linear and nonlinear motion models described above. Details of the IMM state estimator are presented in [42].

The measurement model is given as

$$\mathbf{z}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{w}(k) \tag{9}$$

where $\mathbf{z}(k) \in \mathbb{R}^2$ is the measurement vector of the target at time k, $\mathbf{H}$ is the measurement matrix, and $\mathbf{w}(k)$ is the zero-mean white Gaussian noise such that $E[\mathbf{w}(k)\mathbf{w}(k)'] = \mathbf{R}$.

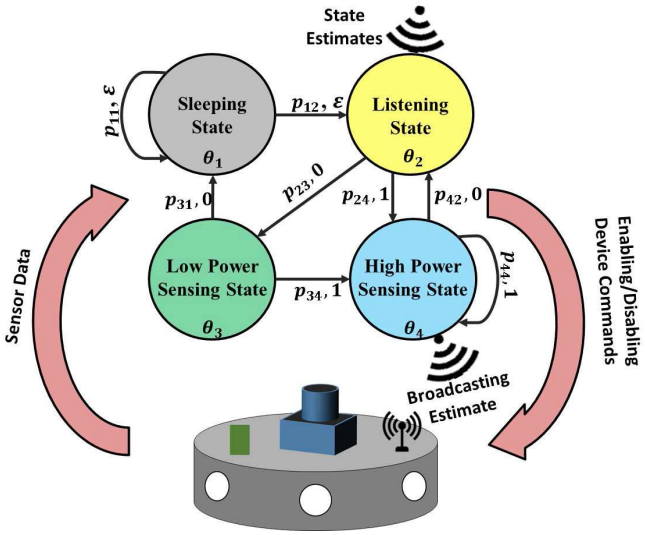

Fig. 2. PFSA based distributed supervisor.

The measurement matrix is assumed to extract the position estimates of the state and is given as

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{10}$$

When the nonlinear model is implemented in EKF, the $\mathbf{H}$ matrix is augmented with zeros to form a 2×5 matrix. The covariance matrix $\mathbf{R}$ of the measurement noise is given as

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{11}$$

where σ_x^2 and σ_y^2 are the variances of the x and y positions, respectively. When the nonlinear model is implemented, $\mathbf{R}$ is augmented with zeros to form a 3×3 matrix.

Note: Detailed derivations of the above models are provided in [42] and are beyond the scope of this paper.

C. Distributed Supervisors for Opportunistic Sensing

The scheduling of each sensor node is controlled by a distributed PFSA-based supervisor, as shown in Fig. 2. First a definition of PFSA is presented below.

Definition 2 (PFSA): A PFSA [43] is defined as a four-tuple $M = \langle \Theta, A, \delta, P \rangle$, where:

1) Θ is a finite set of states;
2) A is a finite alphabet;
3) $\delta \subseteq \Theta \times A \times \Theta$ is a set of state transitions;
4) $p : \Theta \times \Theta \rightarrow [0, 1]$ are the state transition probabilities which form a stochastic matrix $P \equiv [p_{i,j}]$, where $p_{i,j} \equiv p(\theta_i, \theta_j)\ \forall \theta_i, \theta_j \in \Theta$, such that

$$\sum_{\theta' \in \Theta} p(\theta, \theta') = 1, \ \forall \theta \in \Theta.$$

The state set Θ consists of four states: 1) *Sleeping* (θ_1); 2) *Listening* (θ_2); 3) LPS (θ_3); and 4) HPS (θ_4), as shown in Fig. 2. These states have the following attributes.

1) *Sleeping (θ_1):* Disables all devices on the node to consume minimal energy.

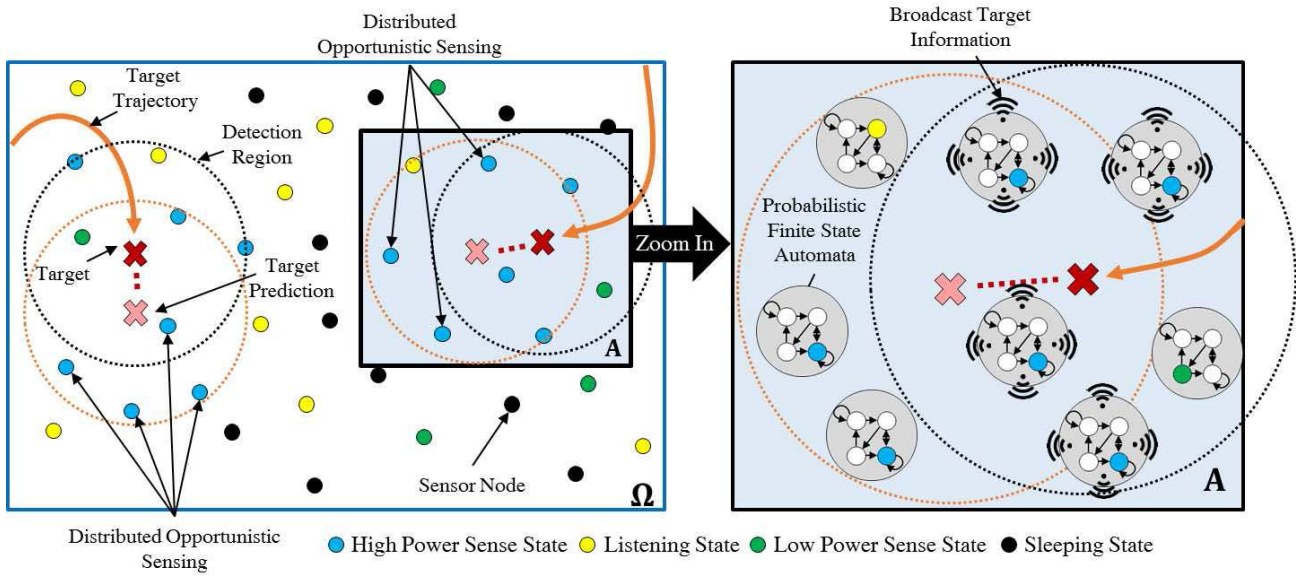

Fig. 3. Illustration of the functioning of the POSE algorithm.

TABLE I
DEVICES ENABLED(✓)/DISABLED(✗) AT PFSA STATES

States \ Devices	DPU	LPS	HPS	Transmitter	Receiver
Sleeping (θ_1)	✓	✗	✗	✗	✗
Listening (θ_2)	✓	✗	✗	✗	✓
LPS (θ_3)	✓	✓	✗	✗	✗
HPS (θ_4)	✓	✗	✓	✓	✗

2) *Listening (θ_2):* Disables all sensing devices but collects information from the neighbors, fuses them, and predicts if a target will arrive in its own HPS coverage area.
3) *LPS (θ_3):* Detects the target using a low power sensor for conserving energy.
4) *HPS (θ_4):* Detects the target using a high power sensor, estimates its current state, predicts its next position for tracking and broadcasts it to the neighbors.

Table I shows the various devices that are enabled or disabled on different PFSA states.

The alphabet is defined as $A = \{\epsilon, 0, 1\}$, where ϵ is the null symbol which is emitted when no information is available, 0 indicates no target detection, and 1 indicates target detection. A symbol is emitted at each state transition based on the information observed/received at the current state as shown in Fig. 2. Thus a symbol sequence is generated which keeps track of the node's target detection history.

The state transition probabilities of the PFSA are dynamically updated based on the observed/received data to facilitate opportunistic sensing. The PFSA at each sensor node is designed to enable the HPS state only when it detects that a target is present or predicts that it is arriving within the node's HPS coverage area. When the target is outside its coverage area, the node conserves energy by cycling in the low power consuming states, i.e., *Sleeping, Listening*, and LPS. In this cycle, the target may still be detected by the LPS device or predicted in the *Listening* state. In the LPS state the node detects a target using simple LPS devices. In the *Listening* state the node does not perform any sensing itself but it collects information from its neighbors about the target's presence and its state in their coverage areas. It then uses this information to fuse and predict the arrival of the target in its own coverage area. The node then transitions from either the LPS or the *Listening* state to the HPS state depending on the target detection or its predicted arrival, respectively.

Fig. 3 illustrates the POSE algorithm in a multitarget tracking scenario with N randomly deployed sensor nodes and two targets traversing through the region Ω. As seen in the figure, the sensor nodes form spatio-temporal clusters of HPS nodes in a distributed manner, within the regions where the two targets are detected and predicted to be, while the sensor nodes outside of these regions conserve energy by staying in *Sleeping*, *Listening*, and LPS states. The HPS node clusters broadcast the target state information to alarm all the *Listening* nodes in the neighborhood. The *Listening* nodes then again use this information received from multiple HPS nodes to predict the next position of the target, and accordingly transition to the HPS state during the next time step. In this manner, HPS clusters are formed around the target for tracking accuracy, while the remaining network conserves the energy. The zoomed in region in Fig. 3 shows the corresponding PFSAs in action.

IV. POSE ALGORITHM

This section presents the POSE algorithm which uses a distributed PFSA-based supervisor at each node $s_i \in \mathcal{S}$ for scheduling. The following sections describe the details of the individual states of the PFSA with the corresponding subalgorithms, and the computation of the state transition probabilities which govern the switching of states.

A. Sleeping State

The *Sleeping* state θ_1 is designed to minimize energy consumption by disabling all devices on the node when no target

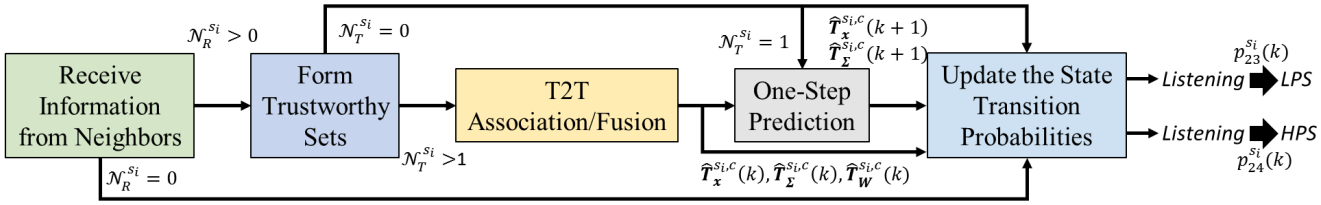

Fig. 4. Process diagram for the *Listening* state.

is detected nearby. In the *Sleeping* state, the sensor node either stays in the *Sleeping* state with a probability $p_{11}^{s_i}(k) = \lambda$ or makes a transition to the *Listening* state with a probability $p_{12}^{s_i}(k) = 1 - \lambda$, where λ is a design parameter that governs the looping of the node in the *Sleeping* state. The value of λ is chosen to be 0.5 based on its effect on network characteristics, which are presented in Section V-A. At the *Sleeping* state θ_1, the following state transitions are possible: $\delta_{\theta_1} = \{(\theta_1, \epsilon, \theta_1), (\theta_1, \epsilon, \theta_2)\}$, where ϵ denotes no observation of the target.

B. Low Power Sensing State

The LPS state θ_3 is designed to conserve energy while enabling target detection using the LPS device. Only the LPS device and the DPU are enabled at this state while all communication devices are disabled. From the LPS state the node can transition to either the HPS state θ_4 if a target is detected or to the *Sleeping* state θ_1 if no target is detected according to the following probabilities:

$$\begin{aligned} p_{34}^{s_i}(k) &= P_d^{s_i}(k) \text{ and} \\ p_{31}^{s_i}(k) &= 1 - P_d^{s_i}(k) \end{aligned} \tag{12}$$

where $P_d^{s_i}(k)$ is the detection model. This paper, like other works [15], uses a simple detection model which assumes that the target is detected if it falls within the sensing radius. A more realistic model can be incorporated into $P_d^{s_i}$ to represent the real sensor suite and will be studied in our future work. The model used here is equal to the detection probability $\alpha = 0.95$ if a target is present within the sensing radius, while it is equal to the probability of a false alarm [44] $P_{\text{fa}} = 1 - e^{-fT}$ if a target is absent, where $f = 10^{-3}\ s^{-1}$ is the false alarm rate during a T second scan. Thus, at the LPS state θ_3 the following state transitions are possible: $\delta_{\theta_3} = \{(\theta_3, 0, \theta_1), (\theta_3, 1, \theta_4)\}$, where the symbol 0 or 1 is emitted for target absence or detection, respectively.

C. Listening State

The *Listening* state θ_2 is designed to conserve energy while enabling target prediction without sensing itself. Only the DPU and the Receiver are enabled at this state while all sensing devices are disabled. Here the sensor node listens to its neighbors to receive if there is any target information available. The information received must then be: 1) validated; 2) associated; and 3) fused together to estimate and initialize the state of the target. Then, a one-step prediction of the target state is computed to predict the target's path. If a target is predicted to arrive in the next time step within the node's HPS coverage area, then the state transition probabilities of this state are updated to favor the transition of the node's state to the HPS state for further interrogation. The associated steps are shown in Fig. 4 and described below.

1) Information Received From Neighbors: Consider a neighboring sensor node $s_j \in \mathcal{N}^{s_i}$ and consider that a target is present within its HPS sensing radius $R_{s,\text{HPS}}$. If this node is in the HPS state at time k, then it observes the target and generates estimates of the target state $\hat{\mathbf{x}}^{s_j}(k|k)$, its covariance $\hat{\mathbf{\Sigma}}^{s_j}(k|k)$, and the filter gain matrix $\hat{\mathbf{W}}^{s_j}(k)$ (for details see Section IV-D). Then, it broadcasts the information packet $(\hat{\mathbf{x}}^{s_j}(k|k), \hat{\mathbf{\Sigma}}^{s_j}(k|k), \hat{\mathbf{W}}^{s_j}(k))$ to its neighbors (assuming a single-hop network) which are in the *Listening* state within the communication radius, R_c. Since node s_i is in the *Listening* state, it receives these packets with a probability $P_r^{s_i} = \beta$, where $\beta = 0.95$. A realistic communication model based on channel properties will be studied in future work. Let $\mathcal{N}_R^{s_i} \subseteq \mathcal{N}^{s_i}$ be the set of neighbors in the HPS state from which packets are successfully received. Then the following information ensembles are constructed:

$$\begin{aligned} \hat{\mathbf{I}}_{\mathbf{x}}^{s_i}(k) &= \left\{\hat{\mathbf{x}}^{s_j}(k|k), \quad \forall s_j \in \mathcal{N}_R^{s_i}\right\} \\ \hat{\mathbf{I}}_{\mathbf{\Sigma}}^{s_i}(k) &= \left\{\hat{\mathbf{\Sigma}}^{s_j}(k|k), \quad \forall s_j \in \mathcal{N}_R^{s_i}\right\} \\ \hat{\mathbf{I}}_{\mathbf{W}}^{s_i}(k) &= \left\{\hat{\mathbf{W}}^{s_j}(k), \quad \forall s_j \in \mathcal{N}_R^{s_i}\right\}. \end{aligned} \tag{13}$$

2) Formation of Trustworthy Sets: Since some of the received estimates might be poor due to noise and other factors, it is important to ensure that they are reliable before using them further. Therefore, the received information ensembles $\hat{\mathbf{I}}_{\mathbf{x}}^{s_i}(k), \hat{\mathbf{I}}_{\mathbf{\Sigma}}^{s_i}(k)$, and $\hat{\mathbf{I}}_{\mathbf{W}}^{s_i}(k)$, are filtered to form sets of trustworthy estimates. This allows the node to determine reliable state estimates, thus reducing false alarms.

The set of trustworthy neighbors $\mathcal{N}_T^{s_i} \subseteq \mathcal{N}_R^{s_i}$ is obtained by evaluating the sum of the position error as follows:

$$\mathcal{N}_T^{s_i} = \left\{s_j \in \mathcal{N}_R^{s_i} : \text{Trace}\left(\mathbf{H}\hat{\mathbf{\Sigma}}^{s_j}(k|k)\mathbf{H}'\right) \leq \xi\right\} \tag{14}$$

where $\mathbf{H}$ is the measurement matrix and ξ is the maximum tolerance of the estimate. In this paper, $\zeta = (\sigma_x + \sigma_y)/2$ was chosen. Next we define the trustworthy sets.

Definition 3 (Trustworthy Sets): Define $\hat{\mathbf{T}}_{\mathbf{x}}^{s_i}(k) \subseteq \hat{\mathbf{I}}_{\mathbf{x}}^{s_i}(k)$, $\hat{\mathbf{T}}_{\mathbf{\Sigma}}^{s_i}(k) \subseteq \hat{\mathbf{I}}_{\mathbf{\Sigma}}^{s_i}(k)$, and $\hat{\mathbf{T}}_{\mathbf{W}}^{s_i}(k) \subseteq \hat{\mathbf{I}}_{\mathbf{W}}^{s_i}(k)$ to be the trustworthy sets of state estimates, covariance estimates, and filter gain estimates, respectively.

These trustworthy sets are constructed as follows:

$$\begin{aligned}\hat{\mathbf{T}}_{\mathbf{x}}^{s_i}(k) &= \left\{\hat{\mathbf{x}}^{s_j}(k|k), \quad \forall s_j \in \mathcal{N}_T^{s_i}\right\}\\ \hat{\mathbf{T}}_{\mathbf{\Sigma}}^{s_i}(k) &= \left\{\hat{\mathbf{\Sigma}}^{s_j}(k|k), \quad \forall s_j \in \mathcal{N}_T^{s_i}\right\}\\ \hat{\mathbf{T}}_{\mathbf{W}}^{s_i}(k) &= \left\{\hat{\mathbf{W}}^{s_j}(k), \quad \forall s_j \in \mathcal{N}_T^{s_i}\right\}. \end{aligned} \tag{15}$$

Remark 1: The trustworthy sets capture reliable information which enables: 1) elimination of faulty state estimates; 2) enhanced fusion of the state estimates; 3) reduced computational complexity; and 4) reduced false alarms.

3) Track-to-Track (T2T) Association: Since there could be multiple targets in the ROI which might be passing close to each other, the trustworthy sets $\hat{\mathbf{T}}_{\mathbf{x}}^{s_i}(k)$, $\hat{\mathbf{T}}_{\mathbf{\Sigma}}^{s_i}(k)$, and $\hat{\mathbf{T}}_{\mathbf{W}}^{s_i}(k)$ formed in the previous section, may contain information related to different targets. Therefore, the estimates in trustworthy sets must be associated to the correct targets and used to identify the number of targets for improving the state vector fusion. Since each sensor node in the network uses the same process noise in the motion model [45], each estimate received is correlated and the traditional Mahalanobis distance metric cannot be used. To overcome this, Bar-Shalom [45] proposed a track association measure using the metric

$$\begin{aligned}\varrho_{jh}^2 = \left(\hat{\mathbf{x}}^{s_j}(k|k) - \hat{\mathbf{x}}^{s_h}(k|k)\right)'\Big[\hat{\mathbf{\Sigma}}^{s_j}(k|k) + \hat{\mathbf{\Sigma}}^{s_h}(k|k)\\ - \hat{\mathbf{\Sigma}}^{s_j,s_h}(k|k) - \hat{\mathbf{\Sigma}}^{s_h,s_j}(k|k)\Big]^{-1} \times \left(\hat{\mathbf{x}}^{s_j}(k|k) - \hat{\mathbf{x}}^{s_h}(k|k)\right)\end{aligned} \tag{16}$$

where ϱ_{jh}^2 is the association metric between sensors s_j and s_h which belong to the set of trustworthy neighbors $\mathcal{N}_T^{s_i}$; and $\left(\hat{\mathbf{x}}^{s_j}(k|k),\ \hat{\mathbf{\Sigma}}^{s_j}(k|k)\right)$ and $\left(\hat{\mathbf{x}}^{s_h}(k|k),\ \hat{\mathbf{\Sigma}}^{s_h}(k|k)\right)$ are the state and covariance estimates for a target at time k, for sensors s_j and s_h, respectively. The cross covariance term $\hat{\mathbf{\Sigma}}^{s_j,s_h}(k|k)$ in (16), between sensor s_j and s_h is computed as follows:

$$\begin{aligned}&\hat{\mathbf{\Sigma}}^{s_j,s_h}(k|k)\\ &= \Big[\mathbf{I} - \hat{\mathbf{W}}^{s_j}(k)\cdot\mathbf{H}\Big]\mathbf{F}\cdot\hat{\mathbf{\Sigma}}^{s_j,s_h}(k-1|k-1)\cdot\mathbf{F}'\Big[\mathbf{I} - \hat{\mathbf{W}}^{s_h}(k)\cdot\mathbf{H}\Big]'\\ &+ \Big[\mathbf{I} - \hat{\mathbf{W}}^{s_j}(k)\cdot\mathbf{H}\Big]\cdot\mathbf{Q}\cdot\Big[\mathbf{I} - \hat{\mathbf{W}}^{s_h}(k)\cdot\mathbf{H}\Big]'\end{aligned} \tag{17}$$

where $\mathbf{I}$ is an identity matrix, and $\hat{\mathbf{\Sigma}}^{s_i,s_j}(k-1|k-1)$ is the previous cross covariance between sensors s_j and s_h.

To implement *T2T Association*, the sensor node s_i forms groups by associating the estimates received from its trustworthy neighbors using the χ^2-Test [45]. This is done in a manner such that any two sensors, say s_j and s_h, within a group, must satisfy $\varrho_{jh}^2 \leq \chi_m^2(\gamma)$, where m is the number of measurements and $\gamma = 0.05$ is the level of significance.

Thus each sensor from the set of trustworthy neighbors is assigned to one of the groups $\mathcal{N}_T^{s_i,c}$, $c = 1, 2, \ldots, C$, such that

$$\bigcup_{c=1}^{C} \mathcal{N}_T^{s_i,c} = \mathcal{N}_T^{s_i} \tag{18}$$

and the intersection between any two groups is empty. Note that C may not be necessarily equal to the total number of targets L in the ROI. This is because not all targets may have been observed by the same group of sensors.

The corresponding estimates in trustworthy sets $\hat{\mathbf{T}}_{\mathbf{x}}^{s_i}(k)$, $\hat{\mathbf{T}}_{\mathbf{\Sigma}}^{s_i}(k)$, and $\hat{\mathbf{T}}_{\mathbf{W}}^{s_i}(k)$ are grouped as follows:

$$\begin{aligned}\hat{\mathbf{T}}_{\mathbf{x}}^{s_i,c}(k) &= \left\{\hat{\mathbf{x}}^{s_j}(k|k),\ \forall s_j \in \mathcal{N}_T^{s_i,c}\right\},\ \forall c = 1,\ldots,C\\ \hat{\mathbf{T}}_{\mathbf{\Sigma}}^{s_i,c}(k) &= \left\{\hat{\mathbf{\Sigma}}^{s_j}(k|k),\ \forall s_j \in \mathcal{N}_T^{s_i,c}\right\},\ \forall c = 1,\ldots,C\\ \hat{\mathbf{T}}_{\mathbf{W}}^{s_i,c}(k) &= \left\{\hat{\mathbf{W}}^{s_j}(k),\ \forall s_j \in \mathcal{N}_T^{s_i,c}\right\},\ \forall c = 1,\ldots,C.\end{aligned} \tag{19}$$

This procedure is an important step for each node since it needs to develop a reliable state estimate corresponding to every target that is believed to be in its neighborhood. If the node were to bypass association, the fused state estimate could be inaccurate leading to poor control strategies.

4) Track-to-Track Fusion: Once the information received from neighbors is filtered and associated to form C trustworthy groups, $\hat{\mathbf{T}}_{\mathbf{x}}^{s_i,c}(k)$, $\hat{\mathbf{T}}_{\mathbf{\Sigma}}^{s_i,c}(k)$, and $\hat{\mathbf{T}}_{\mathbf{W}}^{s_i,c}(k)$, $\forall c = 1, 2, \ldots, C$, the next step is fusion of information within each group separately to generate a total of C target state estimates. This paper utilizes the *T2T Fusion* algorithm [46] for this purpose which combines multiple state estimates into a single estimate.

For fusion, the following procedure is performed. For each group $c = 1, 2, \ldots, C$, the cross covariance matrix $\widetilde{\mathbf{\Sigma}}^{s_i,c}(k)$ between all $s_j, s_h \in \mathcal{N}_T^{s_i,c}$ is constructed as follows:

$$\widetilde{\mathbf{\Sigma}}^{s_i,c}(k) = \begin{bmatrix} \vdots & \vdots & \vdots \\ \cdots & \hat{\mathbf{\Sigma}}^{s_j,s_h}(k|k) & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{20}$$

where $\hat{\mathbf{\Sigma}}^{s_j,s_h}(k|k)$ is given as

$$\hat{\mathbf{\Sigma}}^{s_j,s_h}(k|k) = \begin{cases} \hat{\mathbf{\Sigma}}^{s_j}(k|k) & \text{if } s_j = s_h \\ \hat{\mathbf{\Sigma}}^{s_j,s_h}(k|k) & \text{otherwise.} \end{cases} \tag{21}$$

Then, the node s_i computes the fused covariance matrix and state vector from its trustworthy associated groups as follows:

$$\hat{\mathbf{\Sigma}}^{s_i,c}(k|k) = \left(\widetilde{\mathbf{I}}\left(\widetilde{\mathbf{\Sigma}}^{s_i,c}(k)\right)^{-1}\widetilde{\mathbf{I}}'\right)^{-1} \tag{22}$$

$$\hat{\mathbf{x}}^{s_i,c}(k|k) = \hat{\mathbf{\Sigma}}^{s_i,c}(k|k)\widetilde{\mathbf{I}}\left(\widetilde{\mathbf{\Sigma}}^{s_i,c}(k)\right)^{-1}\begin{bmatrix} \vdots \\ \hat{\mathbf{x}}^{s_j}(k|k) \\ \vdots \end{bmatrix} \tag{23}$$

where $\widetilde{\mathbf{I}} = \left[\mathbf{I}\ldots\mathbf{I}\right]$ consists of $|\mathcal{N}_T^{s_i,c}|$ $n \times n$ identity matrices and $\hat{\mathbf{x}}^{s_j}(k|k) \in \hat{\mathbf{T}}_{\mathbf{x}}^{s_i,c}(k)$. In this manner single estimates are produced for each associated group in the trustworthy set.

5) One-Step Prediction: The next step in the *Listening* state algorithm is to predict the states of all expected targets during the next time step. This allows the sensor node to construct a belief of each expected target's trajectory and update the state transition probabilities of the PFSA for opportunistic sensing. Thus for each expected target $c = 1, 2, \ldots, C$, the one-step predictions of its state and covariance are computed.

For the KF estimator, these are obtained as

$$\begin{aligned}\hat{\mathbf{x}}^{s_i,c}(k+1|k) &= \mathbf{F}\hat{\mathbf{x}}^{s_i,c}(k|k)\\ \hat{\mathbf{\Sigma}}^{s_i,c}(k+1|k) &= \mathbf{F}\hat{\mathbf{\Sigma}}^{s_i,c}(k|k)\mathbf{F}' + \mathbf{Q}\end{aligned} \tag{24}$$

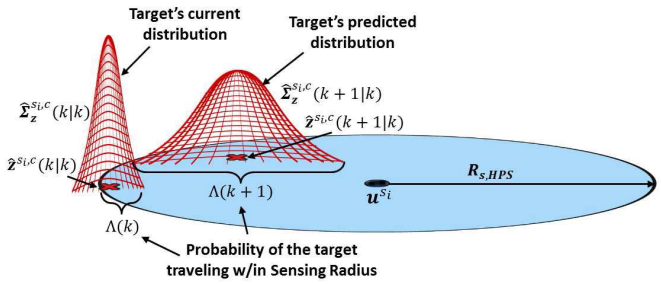

Fig. 5. Computation of the transition probability, $P_{\text{HPS}}^{s_i}(k)$.

while for the EKF estimator they are obtained as

$$\begin{aligned}\hat{\mathbf{x}}^{s_i,c}(k+1|k) &= \mathbf{F}(k)\hat{\mathbf{x}}^{s_i,c}(k|k)\\ \hat{\mathbf{\Sigma}}^{s_i,c}(k+1|k) &= \mathbf{f}_x(k)\hat{\mathbf{\Sigma}}^{s_i,c}(k|k)\mathbf{f}_x(k)' + \mathbf{Q}\end{aligned} \tag{25}$$

where the corresponding $\mathbf{F}$ and $\mathbf{Q}$ are used for the KF and the EKF, respectively; and $\mathbf{f}_x(k) = [\bigtriangledown_x \mathbf{F}(k)']'|_{x=\hat{\mathbf{x}}^{s_i,c}(k|k)}$. For the IMM, the model that produces the maximum likelihood, $\Delta_{0j}^{s_i,c}$ (described in the supplementary material), is used to compute the one-step prediction as described above.

These predictions made by the sensor node s_i in the *Listening* state are then used to determine whether or not to enable its HPS device, as described below.

6) Updating the State Transition Probabilities: Once the one-step predictions are obtained for the C tracks, the sensor node s_i computes the probability of a target traveling through its HPS coverage region at time k, which is given as

$$\Lambda^{s_i,c}(k) = \int\int_D N\Big(\hat{\mathbf{z}}^{s_i,c}(k|k), \hat{\mathbf{\Sigma}}_{\mathbf{z}}^{s_i,c}(k|k)\Big)dxdy \tag{26}$$

where $D = \{x, y|(x-x^{s_i})^2 + (y-y^{s_i})^2 \leq R_{s,\text{HPS}}\}$, $\hat{\mathbf{z}}^{s_i,c}(k|k) = \mathbf{H}\hat{\mathbf{x}}^{s_i,c}(k|k)$ and $\hat{\mathbf{\Sigma}}_{\mathbf{z}}^{s_i,c}(k|k) = \mathbf{H}\hat{\mathbf{\Sigma}}^{s_i,c}(k|k)\mathbf{H}'$. Next it computes the probability that a target is present within its sensing radius (or will travel there at time $k+1$) as follows:

$$P_{\text{HPS}}^{s_i}(k) = \max_c\big(\zeta\,\Lambda^{s_i,c}(k) + (1-\zeta)\Lambda^{s_i,c}(k+1)\big) \tag{27}$$

where the weight $\zeta \in [0,1]$. Preferably more weight is given to the predicted position of the target, s.t. $\zeta < 0.5$, to allow the sensor node to prepare and activate its HPS device opportunistically for energy-efficient target tracking. A visual illustration of this is shown in Fig. 5.

Finally, the PFSA on node s_i which is in the *Listening* state utilizes $P_{\text{HPS}}^{s_i}(k)$ to update its state transition probabilities. From the *Listening* state, it may transition to either the LPS or HPS states with the following probabilities:

$$\begin{aligned}p_{23}^{s_i}(k) &= 1 - P_{\text{HPS}}^{s_i}(k) \text{ and}\\ p_{24}^{s_i}(k) &= P_{\text{HPS}}^{s_i}(k).\end{aligned} \tag{28}$$

Thus, at the *Listening* state θ_2 the following state transitions are possible: $\delta_{\theta_2} = \{(\theta_2, 0, \theta_3), (\theta_2, 1, \theta_4)\}$, where the symbol 0 or 1 is emitted for target's predicted absence or detection within the HPS sensing radius, respectively.

D. High Power Sensing State

The HPS state θ_4 is designed to estimate target's state and track it when it is within its sensing radius $R_{s,\text{HPS}}$. Here the HPS device, DPU, and transmitter are enabled and all other devices are disabled. At this state, the node broadcasts the target's state $\hat{\mathbf{x}}^{s_i}(k|k)$, covariance $\hat{\mathbf{\Sigma}}^{s_i}(k|k)$, and filter gain matrix $\hat{\mathbf{W}}^{s_i}(k)$ to the other nodes within its neighborhood. Since this state consumes the most energy, it should be enabled only when the target's current or predicted position is within its sensing radius. Fig. 6 shows the steps of the HPS state.

The node can come to the HPS state from either the LPS or the *Listening* state. If it has come from the LPS state, then it must have detected a target, but has not initialized the target's state and covariance, thus requiring state initialization. In contrast if it has come from the *Listening* state, then it must have fused received estimates and concluded that a target is traveling or will be traveling through its sensing radius during the next time step. Here it has already initialized estimates of C target tracks; therefore, it uses each of these C tracks to estimate the targets' next state using C parallel estimators. The track that produces the maximum likelihood is used throughout the rest of the estimation process. First we describe the state initialization process when it comes from the LPS state.

1) State Initialization: In this paper, state initialization is achieved using the *two-point differencing* method [42] which is described in the supplementary material.

2) State Estimation: Next the sensor node performs state estimation to continue tracking the target. As discussed earlier, the three different state estimation algorithms considered in this paper include the KF, the EKF, and the IMM filter [42].

Now consider that it has come from the *Listening* state, so it already has C track estimates $\hat{\mathbf{x}}^{s_i,c}(k-1|k-1)$, $\hat{\mathbf{\Sigma}}^{s_i,c}(k-1|k-1)$, and $\mathbf{W}^{s_i,c}(k)$, $c = 1, 2, \ldots, C$. Then using the KF or the EKF algorithm, the sensor node s_i first detects the target and receives a new measurement $\mathbf{z}^{s_i}(k)$. Then, for each existing track the following updates are computed:

$$\mathbf{W}^{s_i,c}(k) = \hat{\mathbf{\Sigma}}^{s_i,c}(k|k-1)\mathbf{H}'\big(\mathbf{S}^{s_i,c}(k)\big)^{-1} \tag{29}$$

$$\hat{\mathbf{x}}^{s_i,c}(k|k) = \hat{\mathbf{x}}^{s_i,c}(k|k-1) + \mathbf{W}^{s_i,c}(k)\mathbf{v}^{s_i,c}(k) \tag{30}$$

$$\hat{\mathbf{\Sigma}}^{s_i,c}(k|k) = \hat{\mathbf{\Sigma}}^{s_i,c}(k|k-1) - \mathbf{W}^{s_i,c}(k)\mathbf{v}^{s_i,c}(k)\mathbf{W}^{s_i,c}(k)' \tag{31}$$

where $\mathbf{v}^{s_i,c}(k)$ and $\mathbf{S}^{s_i,c}(k)$ are the innovation and innovation covariance (see the supplementary material). Next, the sensor node must associate the measurement $\mathbf{z}^{s_i}(k)$ with the target $c \in C$. This is done by selecting the target track $\hat{c}$ that maximizes the likelihood of the estimator as follows:

$$\acute{c} = \underset{c}{\text{argmax}}\left\{|2\pi\mathbf{S}^{s_i,c}(k)|^{\frac{1}{2}}\exp\left\{\frac{-\mathbf{v}^{s_i,c}(k)'(\mathbf{S}^{s_i,c}(k))^{-1}\mathbf{v}^{s_i,c}(k)}{2}\right\}\right\}. \tag{32}$$

Then the updated estimates produced by the sensor node are $\hat{\mathbf{x}}^{s_i,\acute{c}}(k|k)$, $\hat{\mathbf{\Sigma}}^{s_i,\acute{c}}(k|k)$, and $\hat{\mathbf{W}}^{s_i,\acute{c}}(k)$. These estimates are then broadcasted to the neighbors.

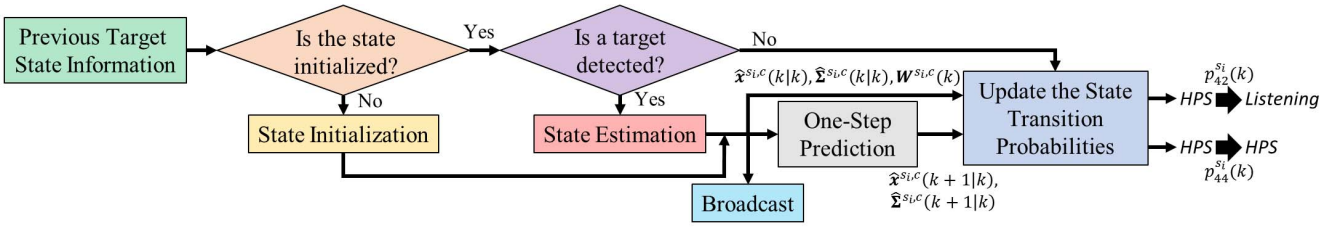

Fig. 6. Process diagram of the HPS state.

When using the IMM filter, both KF and EKF estimators are run in parallel. The IMM gives priority to the model that closely matches the target's trajectory, i.e., uses the EKF when the target is turning or the KF when the target is traveling in a straight line. Similar to the KF and EKF described above, each $\hat{\mathbf{x}}^{s_i,c}(k-1|k-1)$ and $\hat{\mathbf{\Sigma}}^{s_i,c}(k-1|k-1)$ are passed through the IMM and the resulting estimates are associated with the measurement $\mathbf{z}^{s_i}(k)$. The IMM filter computes the state estimate, covariance, and filter gain as follows:

$$\hat{\mathbf{x}}^{s_i,c}(k|k) = \sum_{j=1}^{2} \hat{\mathbf{x}}_{0j}^{s_i,c}(k|k)\mu_j^{s_i,c}(k) \tag{33}$$

$$\hat{\mathbf{\Sigma}}^{s_i,c}(k|k) = \sum_{j=1}^{2} \mu_j^{s_i,c}(k)\left\{\hat{\mathbf{\Sigma}}_{0j}^{s_i,c}(k|k) + \left[\hat{\mathbf{x}}_{0j}^{s_i,c}(k|k) - \hat{\mathbf{x}}^{s_i,c}(k|k)\right] \times \left[\hat{\mathbf{x}}_{0j}^{s_i,c}(k|k) - \hat{\mathbf{x}}^{s_i,c}(k|k)\right]'\right\} \tag{34}$$

$$\mathbf{W}^{s_i,c}(k) = \hat{\mathbf{\Sigma}}^{s_i,c}(k|k)\mathbf{H}'\mathbf{R}^{-1} \tag{35}$$

where $\hat{\mathbf{x}}_{0j}^{s_i,c}(k|k)$ is the mixed updated state estimate, $\mu_j^{s_i,c}(k)$ is mode probability, and $\hat{\mathbf{\Sigma}}_{0j}^{s_i,c}(k|k)$ is the mixed updated covariance estimate for model j, $j = 1, 2$; i.e., KF or EKF. Details are provided in the supplementary material.

After all C state and covariance estimates are updated, the track $\acute{c}$ is selected as

$$\acute{c} = \underset{c}{\operatorname{argmax}} \sum_{j=1}^{2} \Delta_{0j}^{s_i,c}(k) \tag{36}$$

where $\Delta_{0j}^{s_i,c}(k)$ is the likelihood function corresponding to model j which is described in the supplementary material.

3) One Step Prediction and Probability Update: Once the state estimation process is completed, the estimates are broadcasted to the neighbors to alert them of the targets' presence. Then, a one-step prediction is computed as described in Section IV-C5 and the state transition probabilities are updated as in Section IV-C6 and result in the following:

$$\begin{aligned} p_{42}^{s_i}(k) &= 1 - P_{\text{HPS}}^{s_i}(k) \text{ and} \\ p_{44}^{s_i}(k) &= P_{\text{HPS}}^{s_i}(k). \end{aligned} \tag{37}$$

Thus, at the HPS state θ_4 the following state transitions are possible: $\delta_{\theta_4} = \{(\theta_4, 0, \theta_2), (\theta_4, 1, \theta_4)\}$, where the symbol 0 or 1 is emitted for target's predicted absence or detection within the HPS sensing radius, respectively.

E. Characteristics of Random Scheduling Network

For comparison with the POSE algorithm, the characteristics of a network implemented with a random scheduling scheme are obtained using the following result.

The lower bound on sensor density ρ needed to obtain the probability of missed detections P_m is given as

$$\rho \geq -\frac{\ln(P_m)}{\pi R_s^2 \alpha (1 - P_{\text{sleep}})} \tag{38}$$

where P_{sleep} is the sleeping probability for each node and α is the probability of detection for an individual sensor.

The above result is derived as follows. The probability that an individual sensor sampled from the ROI Ω with an area A_Ω detects the target, i.e., the hit probability, is represented by a spatial Poisson process given as [44]

$$p = \Pr\{D_{s_i} = 1 | \text{target in } \Omega\} = 1 - e^{-\alpha\phi} \tag{39}$$

where $D_{s_i} = 1$ if there is a detection and 0 otherwise, and $\phi = (\pi R_s^2 / A_\Omega)$ is the probability that the sampled sensor lies within a distance of sensing radius R_s from the target when sensor nodes are uniformly distributed. Now let $N' = N(1 - P_{\text{sleep}}) = \rho A_\Omega (1 - P_{\text{sleep}})$ be the number of sensors that are sensing the target. Since each sensor is statistically independent and identical, the probability of exactly k sensor detections is given by repeated Bernoulli trials as follows [44]:

$$\Pr\left\{\sum D_i = k\right\} = \binom{N'}{k} p^k (1-p)^{N'-k}. \tag{40}$$

The probability of missed detections P_m, is given as

$$P_m = \Pr\left\{\sum D_i = 0\right\} = \binom{N'}{0} p^0 (1-p)^{N'} = e^{-\alpha\phi N'}. \tag{41}$$

Hence, the result follows by taking ln on both sides.

This result could be used to generate the characteristic plots of P_m versus ρ and P_{sleep}. The results section compares these characteristics with those of the POSE network.

V. Results and Discussion

This section presents the results and discussion of validation of the POSE algorithm on an ROI Ω, which is a 100×100 m^2 environment. First the POSE network characteristics are evaluated and then the algorithm is validated by thorough Monte Carlo simulation runs in MATLAB of different scenarios which consist of multiple targets trespassing through the ROI. The sensor node considered in the simulations consists of a Pandaboard DPU [47], a Xbee TX/RX with a communication radius $R_c = 30$ m [48], a Hokuyo UTM-30LX scanning Laser sensor as the HPS device with a sensing

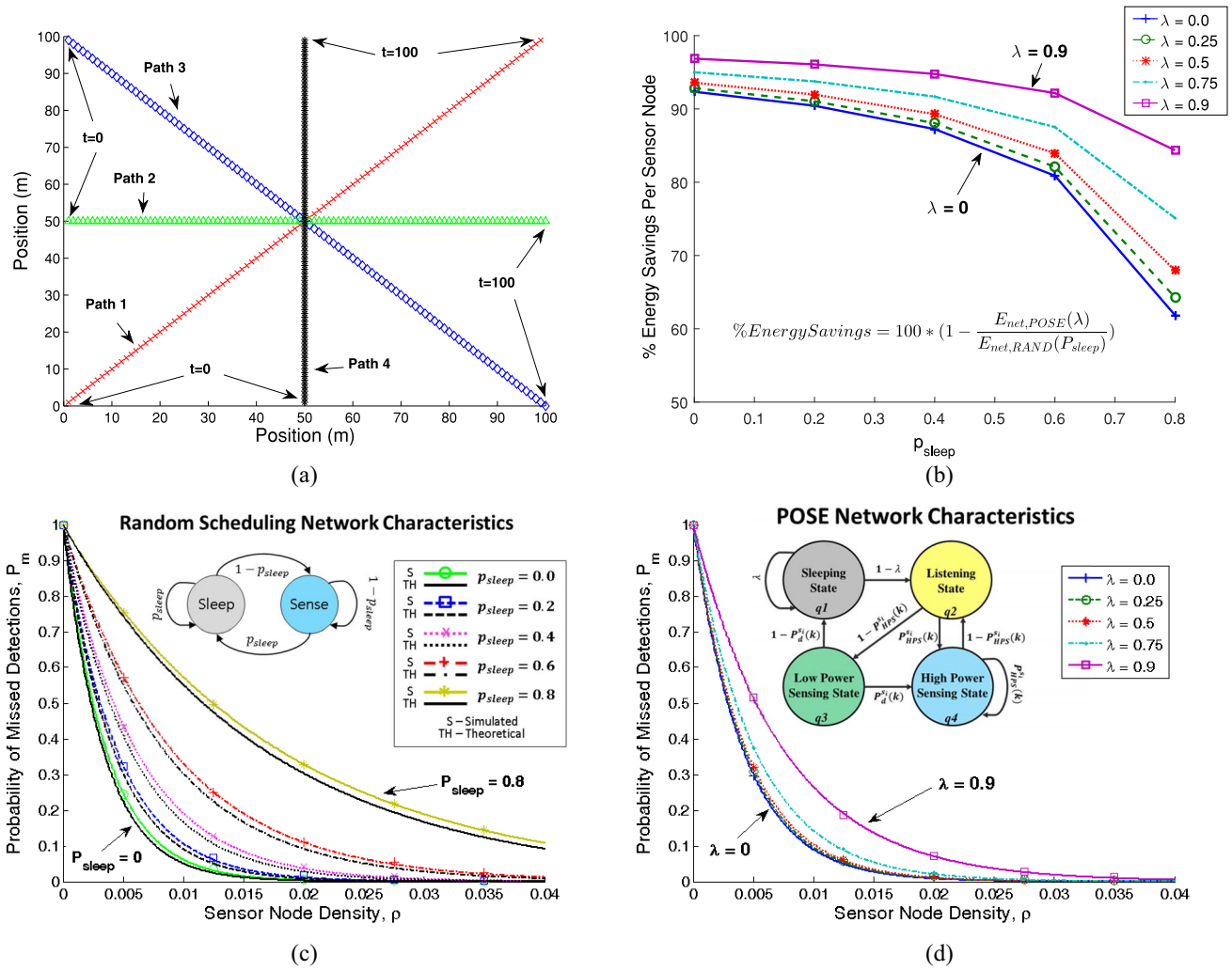

Fig. 7. Comparison of the POSE and random scheduling networks in terms of performance and energy savings. (a) Target trajectories for generating network characteristics. (b) Percentage energy savings of the POSE network. (c) Characteristic curves of the random scheduling network. (d) Characteristic curves of the POSE network.

TABLE II
SENSOR NODE ENERGY MODEL WEIGHTS

e_{DPU}	e_{clock}	e_{LPS}	e_{HPS}	e_{TX}	e_{RX}
1 J/s	0.01 J/s	0.115 J/s	12 J/s	0.3135 J/s	0.165 J/s

radius as $R_{s,\text{HPS}} = 10$ m [49], and six parallax PIR sensors as the LPS device with a sensing radius of $R_{s,\text{LPS}} = 10$ m [50]. The energy consumption of each device is given in Table II.

A. Characteristics of POSE Network Versus Random Scheduling

First, we obtain the characteristics of a network with a random scheduling scheme using the result shown above and validate them by Monte Carlo simulations. The random scheduling scheme has each sensor node randomly switch between sleeping and sensing states with a probability $P_{\text{sleep}} > 0$, while *the all on* scheme [51] has $P_{\text{sleep}} = 0$. For simulation runs, the sensor density ρ was varied and for each ρ the nodes were distributed according to a uniform distribution in Ω. Then, four target trajectories were passed through Ω, as shown in Fig. 7(a). For each target trajectory, 300 Monte Carlo runs were simulated where the sensor node deployment was redistributed during each run. Then P_m was obtained for different P_{sleep} values and plotted. Fig. 7(c) shows the characteristic curves of P_m versus sensor density ρ for different P_{sleep} values. As expected P_m decreases as ρ increases and increases as P_{sleep} increases. Fig. 7(c) also shows that the theoretical and simulated results match closely, thus validating (38).

Subsequently, the POSE network characteristics were obtained in a similar fashion for the same ROI, sensor node densities, and target trajectories [as shown in Fig. 7(a)] as the random scheduling network. Fig. 7(d) shows the characteristic curves of P_m versus sensor density ρ for different λ values, where λ is the loop probability on the *Sleeping* state of the PFSA. As seen from Fig. 7(c) and (d), the characteristic curves for most of the λ values (say $0 \leq \lambda \leq 0.5$) of the POSE network are between $P_{\text{sleep}} = 0$ and $P_{\text{sleep}} = 0.2$; thus are close to the *all on* scheme. However, they do not go below *the all on* scheme which is obvious. The performance of the POSE network approaches the all on scheme as the node density increases. As also seen, the POSE network beats

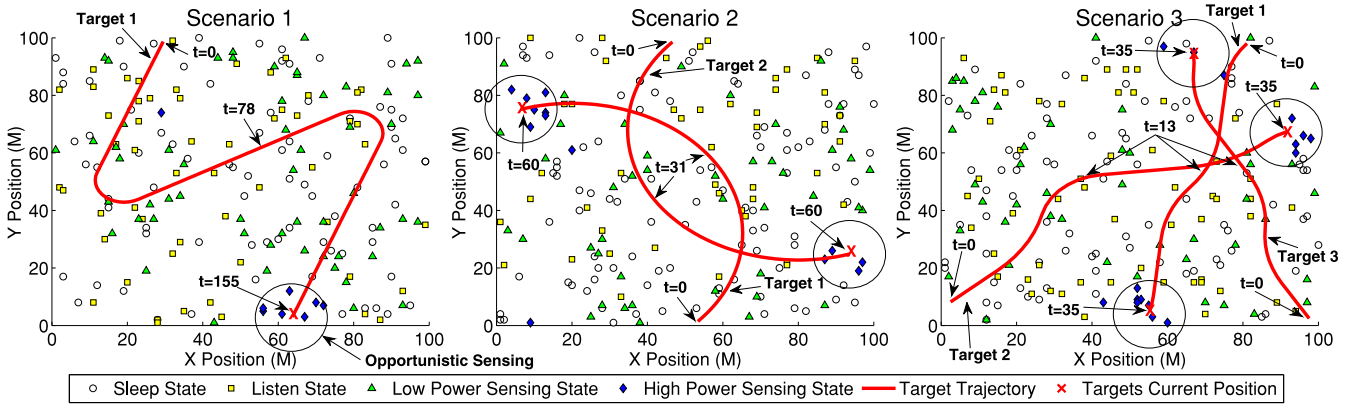

Fig. 8. Simulation scenarios as targets traverse through Ω. All sensor nodes away from the target are in the low power consuming states while the nodes around the targets are in the HPS state allowing for opportunistic sensing.

the performance of most of the random scheduling schemes with $P_{\text{sleep}} > 0.2$.

While achieving a performance close to that of *the all on* scheme, the POSE network results in significant energy savings, since only a small fraction of HPS nodes are enabled during each time step around the target. Let $E_{\text{net},*}$ denote the total energy consumption of a sensor network provided in (2), where the $*$ represents POSE and random scheduling, respectively. Then the percent energy savings of the POSE network compared to the random scheduling network is defined as $\%\text{EnergySavings} = 1 - (E_{\text{net,POSE}}(\lambda)/E_{\text{net,RAND}}(P_{\text{sleep}}))$ where the total energy consumption of each scheduling algorithm were computed for a given λ and P_{sleep} value, respectively. Fig. 7(b) highlights the significant energy savings which increase as λ increases. Thus the following tradeoff exists: for energy savings the value of λ should be large, while for reducing the P_m, the value of λ should be small. Based the results in plots of Fig. 7(b) and (d), a value of $\lambda = 0.5$ is selected and the sensor node density of $\rho = 0.02$ is chosen for further validation studies to achieve a $P_m < 0.02$.

To further validate the POSE network performance, the probability of missed detection was simulated for various values of α to show the effects of the detection model. The results obtained are included in the supplementary material.

B. POSE Network Validation

The tracking performance of the POSE algorithm is validated on three scenarios that consist of different targets traveling through Ω, as shown in Fig. 8. For each scenario, 300 Monte Carlo runs are simulated with a sensor node density $\rho = 0.02$. For each scenario, the sampling time is assumed to be $T = 1$ s and the state estimation algorithms KF, EKF, and IMM, are implemented. The measurement noise for each model is assumed to be $\sigma_x = \sigma_y = 1$ mm and was taken from the Hokuyo UTM-30LX datasheet. The process noise for the KF and EKF are assumed to be $\sigma_v = 0.03$ m and $\sigma_\phi = 0.2$ degrees/sec, to minimize the estimation error of the targets for accurate predictions. For the IMM, the same values of process noise are implemented for their respective models.

As seen in Fig. 8, scenario 1 has a single target traveling with a velocity of 1.4 m/s though the ROI with an "S" shaped trajectory while scenario 2 has two targets traveling with a velocity of 2 m/s following a curved trajectory. The third scenario incorporates three targets traveling with velocities between 2.8 and 3.2 m/s with crossing trajectories.

The tracking accuracy is compared with *the all on* scheme, to validate that the POSE algorithm obtains the same tracking accuracy while saving energy by switching states. Thus the root mean squared error (RMSE) of the position and velocity of the targets is computed at each time step as

$$\text{RMSE}_{\tau_l}(k) = \sqrt{\frac{1}{300}\sum_{\text{mc}=1}^{300} \text{MSE}_{\text{mc}}^{s_i,\tau_l}(k)}$$

where mc represents the Monte Carlo run, $\text{MSE}_{\text{mc}}^{s_i,\tau_l}(k) = (1/N'')\sum_{j=1}^{N''}(\mathbf{x}_{\text{mc}}^{\tau_l}(k) - \hat{\mathbf{x}}_{\text{mc}}^{s_i,\tau_l}(k))^2$, and N'' is the number of sensors during Monte Carlo run mc that estimated target τ_l.

The RMSE for each target scenario is shown in Fig. 9. These figures show the results of the three state estimation algorithms for each targets position and velocity estimate for POSE and the *all on* network. It can be seen that the POSE algorithm results in a smaller tracking error then the *all on* network for each state estimation algorithm. This can be attributed to the accurate state initialization achieved in the *Listening* state via fusion. Thus, the POSE algorithm is able to track multiple targets traveling through the ROI better than the *all on* network that is always monitoring due to the fusion-driven state initialization of the *Listening* state, while saving tremendous amounts of energy. This result also shows the benefits of each state estimation algorithm for different target trajectories. The POSE-KF produces the smallest tracking error when the targets are traveling in a straight line, while the POSE-EKF is the smallest during the turning conditions. By incorporating both of these models into the IMM, the overall estimation error in the linear and nonlinear trajectories is minimized by switching between these models.

For each scenario, the average energy consumption per sensor node is computed. The energy savings as compared to the random schemes are shown in Fig. 10. It is observed that as the number of targets increases the percentage energy savings decreases since more nodes are enabled in the POSE network around each of those targets.

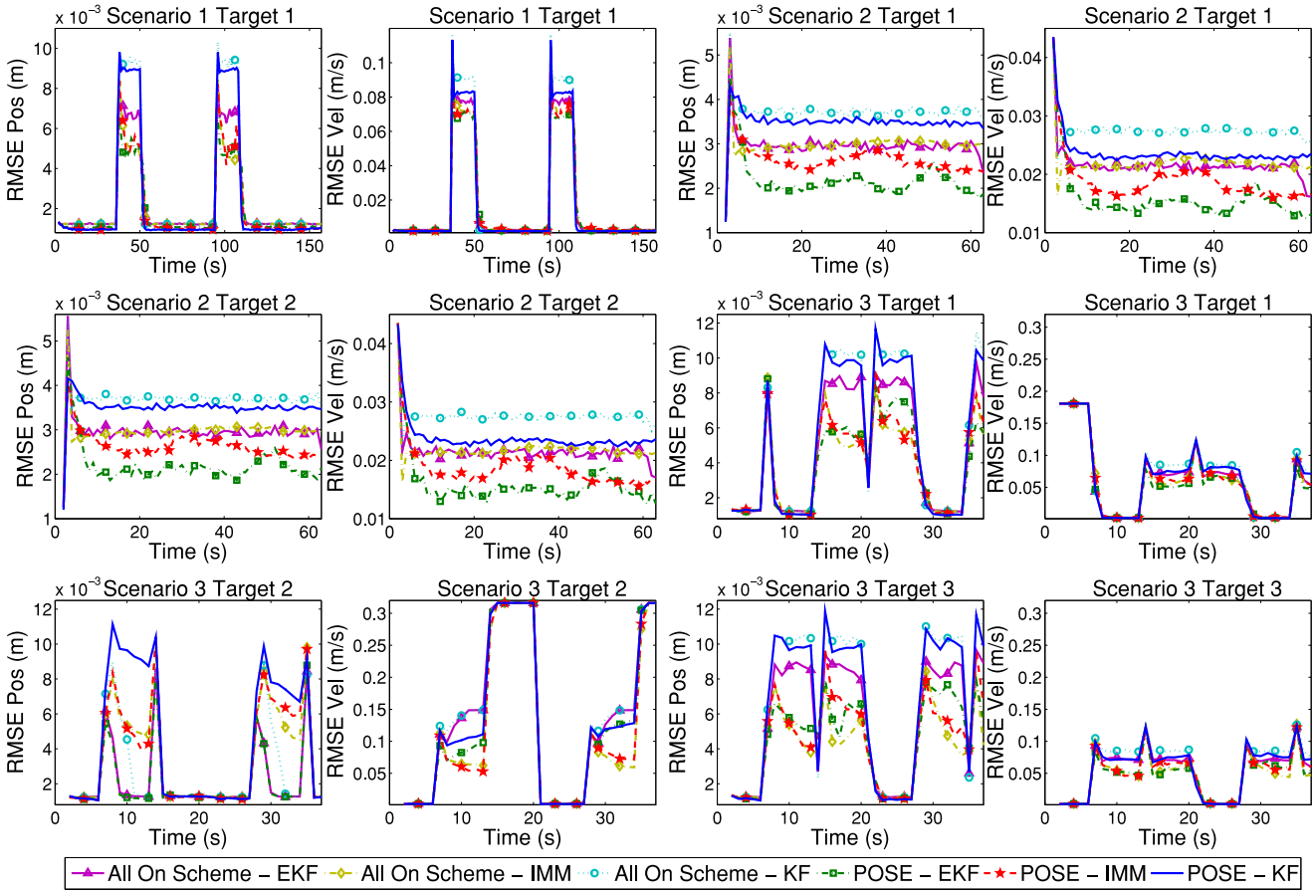

Fig. 9. RMSE results for the POSE and always monitoring scheduling schemes.

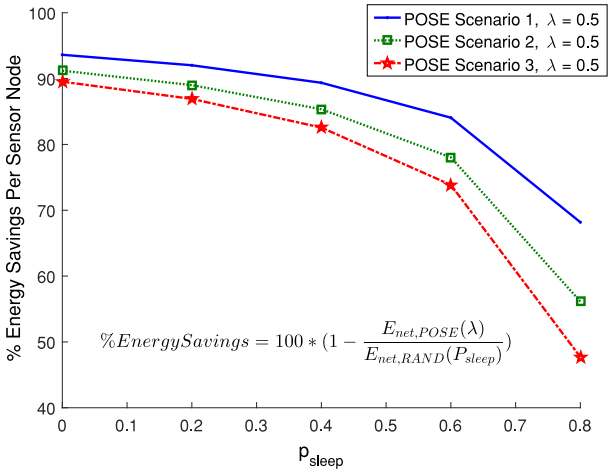

Fig. 10. Percentage energy savings per sensor node.

C. Computational Complexity Analysis

The complexity of the POSE algorithm mainly arises in the HPS and *Listening* states of the PFSA. The HPS states complexity is dominated by the state estimation process. For that process, the primary computation is the inverse of the covariance matrix [52], which is of the order of $O(n^3)$, where n is the number of state variables ($n = 5$ in our examples). Similarly, the *Listening* state's complexity is dominated by the track-to-track fusion step, which is of the order of $O((n \cdot |N_T^{s_i}|)^3)$,

TABLE III
COMPUTATION TIMES IN LISTENING AND HPS STATES

States	KF	EKF	IMM
HPS	$\mu = 0.055ms$	$\mu = 0.078ms$	$\mu = 0.342ms$
Listening	$\mu = 0.683ms$	$\mu = 0.545ms$	$\mu = 1.752ms$

where $|N_T^{s_i}|$ is the number of trustworthy neighbors. Further, the computational complexity was analyzed by measuring the average times taken in each of the above states and the results are shown in Table III. Due to the computational requirements of the HPS device (e.g., laser), the proposed method may require computationally powerful nodes. However, many cheap and powerful DPUs are becoming widely available (e.g., ODROID [53]) making these multimodal sensor nodes very practical. In this paper, the POSE algorithm was simulated in the MATLAB environment on an i5 3.1-GHz CPU computer. Our future work aims to implement this algorithm on an actual sensor node (e.g., with a Pandaboard or an ODROID processor, which have ∼1.2-GHz and ∼1.5-GHz speeds, respectively).

VI. CONCLUSION

This paper presented a novel distributed supervisory control algorithm, called POSE, that facilitates node-level energy management, for minimizing energy usage in target-tracking applications of DSNs. Each sensor node is governed by a supervisor, which is a PFSA, which dynamically adapts its

transition probabilities and switches states to control the communication and sensing devices on the node based on the target's information. Simulation results for various scenarios show that the network can opportunistically adapt to targets' trajectories, minimize missed detection rates, and significantly reduce the energy consumption as compared to random scheduling schemes while ensuring accurate target state estimation.

Future work is envisioned in the following areas. First, alternative methods of data association and fusion would be explored to further improve the computational complexity for more energy savings. Second, the methodology would be extended to 3-D environments for real life applications. Finally, the method would be experimentally validated on a physical sensor network in laboratory and outdoor settings.

REFERENCES

[1] R. A. Best, Jr., *Intelligence, Surveillance, and Reconnaissance (ISR) Programs: Issues for Congress*, document ADA447906, DTIC, Fort Belvoir, VA, USA, 2005.

[2] K. Mukherjee, S. Gupta, A. Ray, and T. A. Wettergren, "Statistical-mechanics-inspired optimization of sensor field configuration for detection of mobile targets," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 3, pp. 783–791, Jun. 2011.

[3] S. Zhu, C. Chen, W. Li, B. Yang, and X. Guan, "Distributed optimal consensus filter for target tracking in heterogeneous sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1963–1976, Dec. 2013.

[4] H. Mahboubi, W. Masoudimansour, A. G. Aghdam, and K. Sayrafian-Pour, "An energy-efficient target-tracking strategy for mobile sensor networks," *IEEE Trans. Cybern.*, vol. 47, no. 2, pp. 511–523, Feb. 2017.

[5] G. Werner-Allen *et al.*, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Comput.*, vol. 10, no. 2, pp. 18–25, Mar./Apr. 2006.

[6] H. Lee, J. S. Choi, and R. Elmasri, "A static evidential network for context reasoning in home-based care," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 6, pp. 1232–1243, Nov. 2010.

[7] S. Gupta, D. S. Singh, and A. Ray, "Statistical pattern analysis of ultrasonic signals for fatigue damage detection in mechanical structures," *NDT E Int.*, vol. 41, no. 7, pp. 491–500, 2008.

[8] H. E. Garcia, W.-C. Lin, S. M. Meerkov, and M. T. Ravichandran, "Resilient monitoring systems: Architecture, design, and application to boiler/turbine plant," *IEEE Trans. Cybern.*, vol. 44, no. 11, pp. 2010–2023, Nov. 2014.

[9] N. Najjar, S. Gupta, J. Hare, S. Kandil, and R. Walthall, "Optimal sensor selection and fusion for heat exchanger fouling diagnosis in aerospace systems," *IEEE Sensors J.*, vol. 16, no. 12, pp. 4866–4881, Jun. 2016.

[10] X. Jin, S. Sarkar, A. Ray, S. Gupta, and T. Damarla, "Target detection and classification using seismic and PIR sensors," *IEEE Sensors J.*, vol. 12, no. 6, pp. 1709–1718, Jun. 2012.

[11] J. Hare, S. Gupta, and J. Wilson, "Decentralized smart sensor scheduling for multiple target tracking for border surveillance," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, 2015, pp. 3265–3270.

[12] T. Semertzidis, K. Dimitropoulos, A. Koutsia, and N. Grammalidis, "Video sensor network for real-time traffic monitoring and surveillance," *IET Intell. Transp. Syst.*, vol. 4, no. 2, pp. 103–112, Jun. 2010.

[13] A. El Kateeb, "Mote design supported with remote hardware modifications capability for wireless sensor network applications," *Int. J. Adv. Smart Sensor Netw. Syst.*, vol. 3, no. 3, pp. 13–21, Jul. 2013.

[14] C. Liu, K. Wu, Y. Xiao, and B. Sun, "Random coverage with guaranteed connectivity: Joint scheduling for wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 6, pp. 562–575, Jun. 2006.

[15] J. Lin, W. Xiao, F. L. Lewis, and L. Xie, "Energy-efficient distributed adaptive multisensor scheduling for target tracking in wireless sensor networks," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 6, pp. 1886–1896, Jun. 2009.

[16] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embedded Comput. Syst.*, vol. 6, no. 4, p. 32, 2007.

[17] M. C. Ranasingha, M. N. Murthi, K. Premaratne, and X. Fan, "Transmission rate allocation in multisensor target tracking over a shared network," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 2, pp. 348–362, Apr. 2009.

[18] J. Liu, M. Chu, and J. E. Reich, "Multitarget tracking in distributed sensor networks," *IEEE Signal Process. Mag.*, vol. 24, no. 3, pp. 36–46, May 2007.

[19] X. Luo, M. Dong, and Y. Huang, "On distributed fault-tolerant detection in wireless sensor networks," *IEEE Trans. Comput.*, vol. 55, no. 1, pp. 58–70, Jan. 2006.

[20] J. Hare, S. Gupta, and J. Song, "Distributed smart sensor scheduling for underwater target tracking," in *Proc. Oceans*, 2014, pp. 1–6.

[21] X. Liu, "A deployment strategy for multiple types of requirements in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2364–2376, Oct. 2015.

[22] C.-P. Chen *et al.*, "A hybrid memetic framework for coverage optimization in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2309–2322, Oct. 2015.

[23] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1448–1453, Dec. 2002.

[24] Y. Song, B. Wang, Z. Shi, K. R. Pattipati, and S. Gupta, "Distributed algorithms for energy-efficient even self-deployment in mobile sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 5, pp. 1035–1047, May 2014.

[25] H. M. Ammari and S. Das, "A study of k-coverage and measures of connectivity in 3D wireless sensor networks," *IEEE Trans. Comput.*, vol. 59, no. 2, pp. 243–257, Feb. 2010.

[26] W. Xiao, S. Zhang, J. Lin, and C. K. Tham, "Energy-efficient adaptive sensor scheduling for target tracking in wireless sensor networks," *J. Control Theory Appl.*, vol. 8, no. 1, pp. 86–92, 2010.

[27] Y. Liu, B. Xu, and L. Feng, "Energy-balanced multiple-sensor collaborative scheduling for maneuvering target tracking in wireless sensor networks," *J. Control Theory Appl.*, vol. 9, no. 1, pp. 58–65, 2011.

[28] Y. Li, L. W. Krakow, E. K. P. Chong, and K. N. Groom, "Approximate stochastic dynamic programming for sensor scheduling to track multiple targets," *Digit. Signal Process.*, vol. 19, no. 6, pp. 978–989, 2009.

[29] E. Masazade, M. Fardad, and P. K. Varshney, "Sparsity-promoting extended Kalman filtering for target tracking in wireless sensor networks," *IEEE Signal Process. Lett.*, vol. 19, no. 12, pp. 845–848, Dec. 2012.

[30] X. Shen and P. K. Varshney, "Sensor selection based on generalized information gain for target tracking in large sensor networks," *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 363–375, Jan. 2014.

[31] Y.-C. Chen and C.-Y. Wen, "Decentralized cooperative TOA/AOA target tracking for hierarchical wireless sensor networks," *Sensors*, vol. 12, no. 11, pp. 15308–15337, 2012.

[32] A. C. Voulkidis and P. G. Cottis, "Optimal node allocation in multiservice WSNs based on correlated strategy," *IEEE Trans. Wireless Commun.*, vol. 15, no. 6, pp. 4196–4205, Jun. 2016.

[33] S. Aeron, V. Saligrama, and D. A. Castanon, "Efficient sensor management policies for distributed target tracking in multihop sensor networks," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2562–2574, Jun. 2008.

[34] O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: Recent developments and deployment challenges," *IEEE Netw.*, vol. 20, no. 3, pp. 20–25, May/Jun. 2006.

[35] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Comput. Commun.*, vol. 30, nos. 14–15, pp. 2826–2841, 2007.

[36] W.-P. Chen, J. C. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 3, pp. 258–271, Jul./Aug. 2004.

[37] O. Younis and S. Fahmy, "HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 4, pp. 366–379, Oct./Dec. 2004.

[38] E. Cayirci, H. Tezcan, Y. Dogan, and V. Coskun, "Wireless sensor networks for underwater surveillance systems," *Ad Hoc Netw.*, vol. 4, no. 4, pp. 431–446, 2006.

[39] J. Li, Q.-S. Jia, X. Guan, and X. Chen, "Tracking a moving object via a sensor network with a partial information broadcasting scheme," *Inf. Sci.*, vol. 181, no. 20, pp. 4733–4753, 2011.

[40] C. Feng, L. Yang, and J. W. Rozenblit, "Adaptive tracking in energy sensitive distributed wireless sensor networks," *Ad Hoc Sensor Wireless Netw.*, vol. 12, nos. 1–2, pp. 55–77, 2011.

[41] J. Chen, K. Cao, K. Li, and Y. Sun, "Distributed sensor activation algorithm for target tracking with binary sensor networks," *Cluster Comput.*, vol. 14, no. 1, pp. 55–64, 2011.
[42] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation With Applications to Tracking and Navigation: Theory Algorithms and Software*. New York, NY, USA: Wiley, 2004.
[43] E. Vidal, F. Thollard, C. De La Higuera, F. Casacuberta, and R. C. Carrasco, "Probabilistic finite-state machines—Part I," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 7, pp. 1013–1025, Jul. 2005.
[44] T. A. Wettergren, "Performance of search via track-before-detect for distributed sensor networks," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 1, pp. 314–325, Jan. 2008.
[45] Y. Bar-Shalom, "On the track-to-track correlation problem," *IEEE Trans. Autom. Control*, vol. AC-26, no. 2, pp. 571–572, Apr. 1981.
[46] Y. Bar-Shalom, X. Tian, and P. K. Willett, *Tracking and Data Fusion: A Handbook of Algorithms*. Storrs, CT, USA: YBS, 2011.
[47] *PandaBoard*. Accessed on Jul. 20, 2017. [Online]. Available: https://www.cs.utexas.edu/∼simon/378/resources/PandaBoardES.pdf
[48] *XBee Transceiver*. Accessed on Jul. 20, 2017. [Online]. Available: https://www.sparkfun.com
[49] *Hokuyo UTM-30lx Laser*. Accessed on Jul. 20, 2017. [Online]. Available: https://www.hokuyo-aut.jp/
[50] *Parallax PIR Sensor*. Accessed on Jul. 20, 2017. [Online]. Available: https://www.parallax.com
[51] S. Pattem, S. Poduri, and B. Krishnamachari, "Energy-quality tradeoffs for target tracking in wireless sensor networks," in *Information Processing in Sensor Networks*. Berlin, Germany: Springer, 2003, pp. 32–46.
[52] F. Daum, "Nonlinear filters: Beyond the Kalman filter," *IEEE Trans. Aerosp. Electron. Syst. Mag.*, vol. 20, no. 8, pp. 57–69, Aug. 2005.
[53] *Odroid*. Accessed on Jul. 20, 2017. [Online]. Available: http://www.hardkernel.com/main/main.php

James Z. Hare received the M.S. degree in electrical engineering from the University of Connecticut, Storrs, CT, USA, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering.

His current research interests include decision making and control in distributed sensor networks, intelligent network control systems, data analysis in complex-networked systems, and fault detection and diagnosis.

Shalabh Gupta (M'07) received the B.E. degree in mechanical engineering from the Indian Institute of Technology Roorkee, Roorkee, India, in 2001, and the M.S. degrees in mechanical engineering and electrical engineering and the Ph.D. degree in mechanical engineering from Pennsylvania State University, University Park, State College, PA, USA, in 2004, 2005, and 2006, respectively.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA. His current research interests include distributed autonomy, cyber-physical systems, robotics, network intelligence, data analytics, information fusion, and fault diagnosis in complex systems.

Dr. Gupta is a member of the American Society of Mechanical Engineers.

Thomas A. Wettergren (SM'06) received the B.S. degree in electrical engineering and the Ph.D. degree in applied mathematics from Rensselaer Polytechnic Institute, Troy, NY, USA.

He is with Naval Undersea Warfare Center, Newport, RI, USA, where he currently serves as the U.S. Navy Senior Technologist for Operational and Information Science. He is also an Adjunct Professor of mechanical engineering with Pennsylvania State University, University Park, State College, PA, USA. His current research interests include development of new analytical and computational methods for the mathematical modeling and control of multiagent undersea sensing systems.

Dr. Wettergren was a recipient of the NAVSEA Scientist of the Year, the Assistant Secretary of the Navy Top Scientist of the Year, and the IEEE-USA Harry Diamond Awards. He is a member of SIAM.