Fault-Tolerant Prediction-Based Scheme for Target Tracking Application

Oualid Demigha\textsuperscript{1,2}, Nadjib Badache\textsuperscript{2}, Mohamed Aissani\textsuperscript{1,2,3}, and Abdelhamid Mellouk\textsuperscript{3}

\textsuperscript{1}Laboratory of Research in Artificial Intelligence, Polytechnic School (EMP)
P.O. Box 17, Bordj-El-Bahri 16111, Algiers, Algeria
\{wdemigha, maissani\}@gmail.com

\textsuperscript{2}Computer Science Department, University of Science and Technology (USTHB)
P.O. Box 32 EMP, EL-Alia 16111, Algiers, Algeria
\{badache@mail.cersit.dz\}

\textsuperscript{3}Laboratory of Images, Signals and Intelligent Systems (LiSSi)
120-122, Paul Armandot 94400, Vitry-sur-Seine, France
\{aissani, mellouk\}@univ-paris12.fr

\textbf{Abstract}—Fault-Tolerance is an important function in target tracking application using wireless sensor networks. We propose in this paper, an efficient fault-tolerant approach for target tracking that prevents the loss of the target. Instead of using a single prediction mechanism, our approach uses a multi-level incremental prediction technique that adjusts the prediction precision of the target movement. The responsible node of target detection uses multiple historical information pieces to calculate multi-level predictions which have different precision levels according to the number of information pieces used. Thanks to our parametric prediction model, our approach increases the prediction success rate and decreases the target loss frequency compared to basic approaches that use simple prediction models.

\textbf{Keywords}— Wireless Sensor Network, Target Tracking, Linear Prediction Mechanism, Fault-Tolerance, Simulation Analysis.

\section{I. INTRODUCTION}

Traditional target tracking systems use powerful sensor nodes to detect and to locate a target within very high sensing range. These battery-free nodes can perform the tracking operation using advanced signal processing techniques, but they form an error-prone network that is not scalable.

Target tracking using Wireless Sensor Networks (WSN) is introduced for the first time in [1]. Technology advances in integrating sensing and communication capacities in a single node allow the deployment of a large number of sensor nodes in the surveillance field, which can form a double- or a triple-sized network. The advantages of such systems are: (1) Better sensor reading quality thanks to the proximity of nodes to the detected target, (2) Fast deployment of the network via wireless technology, and (3) Better reliability due to data redundancy. However, this low-cost approach brings up new challenges: (1) necessity of collaborative communication and processing between nodes, (2) impossibility of using advanced signal-processing techniques due to limited computing resources of the sensor, and (3) impossibility of keeping nodes on all the time due to limited energy resources.

Therefore, if we consider the loss of the target as a system failure then fault-tolerance can be viewed as a compromise between target tracking efficiency and energy efficiency. Indeed, in this case, the prediction failure may affect the tracking system performance.

A target tracking process using WSN consists of initialization, surveillance and tracking phases. In the initialization phase, fault-tolerance can be integrated in the deployment strategy and the clock synchronization between nodes. The deployment strategy aims at decreasing the number of deployed nodes and maintaining the network coverage and connectivity. The synchronization allows the implementation of efficient high-performance algorithms that require strong synchronization hypothesis. At the same time, target tracking requires time-stamping of successive positions of the target to trace its trajectory.

The goal of fault-tolerant surveillance in a target tracking process is to provide a given Quality-of-Surveillance (QoSv) [2] or a given Quality-of-Monitoring (QoM) [3] while achieving energy efficiency. Finally, the tracking process handles the network dynamics by organizing logically the network into clusters, trees or agents.

In order to handle errors and faults in a tracking process, we propose in this paper a fault-tolerant approach that is adaptive to the target movement. It aims at minimizing the number of awaken nodes and enhancing the prediction mechanism reliability via incremental predictions. For simplicity, faults such as node crash and message loss are not considered here.

The rest of the paper is organized as follows. In section II, we present related works that deal with the target-tracking problem. In section III, we describe the system model and the mechanisms of the incremental prediction-based scheme that we propose to track targets using WSN. In section IV, we present some simulation results.
II. RELATED WORKS

To achieve energy efficiency in a target-tracking scheme, sensor nodes should be waked-up according to the target trajectory in order to turn their sensing units off most of the time. So, prediction-based schemes are more energy efficient than are cluster-based, tree-based and agent-based schemes. The most important prediction-based schemes proposed in the literature are summarized in this section.

In the protocol PaM [4], the network lifetime can be extended using selective activation of tracking nodes. Given the predicted position of the target, sensor nodes around it, called Sensor Coordinators (SCs), will be activated to capture it and to activate other nodes that will become themselves SCs, and so on. The PaM protocol uses a mesh process to recover failures and errors. Sung-Min Lee et al. [5] propose another energy efficient approach, called Avoid-Correct, that handles location errors. This technique considers real situations where the target suddenly changes its direction or there is an event detection failure or an error message transmission failure. The error handling is performed in two phases: error avoiding and error correction. The error avoiding phase requires subdividing the node set into four subsets according to the current and the future awaken zones. The error correction phase starts when an active node in the future awaken zone receives a control message. Consequently, an error message is transmitted to the current node and a new larger future awaken zone is created. In DPR [6], authors propose a technique where both the base station and the sensor nodes predict the motion of the target based on historical information, while the PES scheme [7] aims at minimizing the number of nodes involved in the tracking process and at turning-off the other nodes. The protocol DPT [8] is based on prediction and clustering. It has three components: target description, node selection and error recovery. PBA [9] is slightly different from the other schemes as it uses the target motion regularity to minimize energy consumption.

Each above-mentioned fault-tolerance handling scheme follows two steps: prevention-detection and recovery. In the prevention-detection step, PaM uses target speed update to avoid target loss whereas the Avoid-Correct scheme is based on the error-free future awaken zone. On the other hand, PES and DPR use heuristics to determine which nodes should be waken-up to track the target, while DPT increases the sensing range and asks for help from neighbor nodes. Finally, PBA uses the profile concept to prevent the loss of the target. In the recovery step, incremental range and flooding are widely used by these approaches.

III. FAULT-TOLERANT PREDICTION-BASED SCHEME

In our fault-tolerant scheme, we suppose that the network is organized into clusters and every node knows its location. Every cluster is controlled by a cluster-head that knows the location of its members. All nodes begin at the sleep state. They turn themselves on upon reception of a request.

Note that a prediction-based scheme can be combined with cluster-based or tree-based schemes. The system performance depends on the precision of the prediction mechanism because of the energy consumption and the communication overhead induced by prediction failures. Thus, a compromise should be done between prediction precision and energy consumption.

Indeed, a complex prediction mechanism will provide precise positions, so that less prediction failures will occur, but it will require intensive processing on the sensor node, which will reduce the node battery and therefore the network lifetime. On the other hand, if the prediction mechanism is simple then predictions will be less precise but will induce more energy consumption and communication overhead. Nonetheless, a lightweight prediction mechanism is more suitable for a sensor node with scarce processing resources.

From this point of view, fault-tolerance aims at realizing this compromise by using a lightweight prediction mechanism and recovering the prediction failures when the target is lost. When a fast target is detected for the first time, the problem consists at estimating the target trajectory and preventing its loss. We do not consider here the problems of clustering, routing and information collecting. We use lightweight methods for trajectory estimation and prediction calculation that are suitable for sensor nodes. Our method is flexible and always provides accurate predictions because the complexity of calculations is adapted to that of the target motion. We use two mechanisms: Target Tracking and Prediction.

A. Target Tracking Mechanism

Figure 1 shows the data flow diagram of our target-tracking mechanism. All nodes, except the one that keeps the current target state, are in the sleep state to conserve energy. This contrasts with methods that wakeup all nodes or a subset of nodes to detect the target. The tracking process starts when a cluster-head receives a wakeup message via a low-energy channel indicating the possible presence of a target in the zone covered by its cluster. So, all processing tasks are executed on the cluster heads. The members execute only sensing tasks then transmit detection messages to their heads. The target tracking mechanism uses two kinds of timers. The first, called Tracking Timer, is alarmed when the current cluster head broadcasts a wakeup message to its members.

The second timer, called Recovery Timer, is alarmed when the current head declares the loss of the target. Thus, when the tracking timer has expired, the current head checks the reception of one or more detection messages. In this case, it can conclude that the previous prediction has succeeded and the tracking process continues with the estimation of the target position and the generation of multi-level predictions. Then, the current head sends a wakeup message to the next predicted head.

If the current head does not receive any detection messages from its members, it checks the wakeup message to look for more prediction levels to be explored. In this case, the head wakes-up the next predicted head and the tracking process continues again. Otherwise, if there is no more prediction
level to be explored by the current head then the node triggers the **Recovery Procedure**.

![Data flow diagram](image)

The recovery procedure consists of computing a Recapture Range based on the maximum and the last recorded target speeds. A recovery message is broadcasted to all heads within the calculated recovery zone and a recovery timer is alarmed. The heads, within the recovery zone, that receive the recovery messageakeup their members and alarm their tracking timers. When the tracking timer expires the head checks if the target is detected. Then, it sends an Acknowledgment message to the current head in order to prevent it from computing a new recapture range and re-broadcasting a new recovery message. The detecting head becomes the current one and the tracking process resumes from it. On the other hand, if the current head does not receive any acknowledgement message before the expiration of the recovery timer then it will compute a larger wakeup zone and will broadcast a new recovery message to the heads within this zone. If there is more than one head that detect the target in the recovery procedure and, in order to prevent creating multiple paths for the same target, the current head that receives the first acknowledgment message sends a Notification message to the other detecting heads to stop their tracking processes.

**B. Prediction Mechanism**

The second component of our approach is the prediction mechanism. It computes the number of prediction levels according to the target motion after a learning phase where nodes record the number of failures. Each prediction failure is added to the total number of failures, which is then compared to a constant threshold FT (Fail Threshold). If it exceeds this threshold, the maximum number of prediction levels is incremented. Likewise, each successful prediction is added to the total number of successful predictions, which is compared to another constant threshold ST (Success Threshold). If it exceeds this threshold, the maximum number of prediction levels is decremented.

Figure 2 shows an example of successful prediction where the maximum number of prediction levels is three: the first two predictions are incorrect but the third one is successful because it is the most accurate. Therefore, nodes can deduce that the motion complexity is high enough to generate three prediction levels.

To generate the different prediction levels, we cannot use complex prediction algorithms because they consume a large amount of processing resources. Instead, we use a simple linear prediction algorithm based on the fact that the target trajectory is linear between two close points. So, speed and direction estimation can be calculated by using the following geometric formulas:

\[
\begin{align*}
\theta &= \cos^{-1} \frac{x_i - x_{i-1}}{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}} \\
\end{align*}
\]

Thus, the predicted position coordinates can be obtained as follows:

\[
\begin{align*}
x_{i+1} &= x_i + vt \cos \theta \\
y_{i+1} &= y_i + vt \\
\end{align*}
\]

where \((x_i, y_i), (x_{i-1}, y_{i-1})\) and \((x_{i+1}, y_{i+1})\) are the current, the previous and the next position coordinates, respectively.

This is the first level of the incremental prediction that can be applied for a simple motion. For more complicated random motions, we can generalize expressions (1) and (2) to integrate more historical information pieces and to generate more accurate predictions as follows:

\[
\begin{align*}
x_{i+1} &= x_i + vt \cos \theta + \frac{1}{2} at^2 \cos \theta' \\
y_{i+1} &= y_i + vt \sin \theta + \frac{1}{2} at^2 \sin \theta' \\
\end{align*}
\]

Where:

\[
\begin{align*}
\theta' &= \cos^{-1} \frac{x_i - x_{i-2}}{\sqrt{(x_i - x_{i-2})^2 + (y_i - y_{i-2})^2}} \\
\end{align*}
\]

\[
\begin{align*}
a &= \frac{v_i - v}{t_i - t_{i-2}} \quad \text{(the instant acceleration)} \\
v_i &= \frac{\sqrt{(x_i - x_{i-2})^2 + (y_i - y_{i-2})^2}}{t_i - t_{i-2}} \\
\end{align*}
\]

A more sophisticated prediction can be also obtained via the following equations:

\[
\begin{align*}
x_{i+1} &= x_i + vt \cos \theta + \frac{1}{2} at^2 \cos \theta' + \frac{1}{6} mt^3 \cos \theta'' \\
y_{i+1} &= y_i + vt \sin \theta + \frac{1}{2} at^2 \sin \theta' + \frac{1}{6} mt^3 \sin \theta'' \\
\end{align*}
\]
node 0 starts the tracking process by activating its sensing information pieces in the wakeup messages. The tracking timer is set to 30 ms and the recovery timer is set to 7 x (tracking timer) x (re-capture range).

The transmission range is set to 1.5 m and the sensing range to 0.75 m. The simulation time varies between 100 and 200 s. The tracking timer is set to 30 ms and the recovery timer is set to 7 x (tracking timer) x (re-capture range).

Table I. Performance Evaluation Scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Scenario type</th>
<th>Size (Grid)</th>
<th>Maximum speed (m/s)</th>
<th>Target mobility parameters' update frequency</th>
<th>Speed and direction standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Medium</td>
<td>7x7</td>
<td>20</td>
<td>3</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>Complex</td>
<td>7x7</td>
<td>30</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>Complex</td>
<td>11x11</td>
<td>30</td>
<td>3</td>
<td>1.0</td>
</tr>
<tr>
<td>4</td>
<td>Simple</td>
<td>11x11</td>
<td>20</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>Medium</td>
<td>11x11</td>
<td>20</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>Complex</td>
<td>11x11</td>
<td>20</td>
<td>3</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Figure 2. Example of a successful prediction.

Re-capture range calculation is based on the following formulas:

\[ a = \frac{V_{\text{max}} - V_{\text{last}}}{\Delta t} \] (16)

\[ R = \frac{V_{\text{max}}^2 - V_{\text{last}}^2}{2a} \] (17)

Where, \( V_{\text{max}} \) and \( V_{\text{last}} \) are the maximum and the last recorded target speed and \( a \) is the target acceleration between the moment of the target loss and the actual instant.

IV. SIMULATION RESULTS

To demonstrate the improvement of the target tracking system performance, we implemented our scheme on the TinyOS 1.x system [10] and we compared it in TOSSIM [11] simulator with two variants, namely, Simple Prediction and Complex Prediction which use only one prediction level and based on formulas (5)-(6) and (10)-(11), respectively. Note that the simulated target follows the Gauss-Markov mobility model [12]. In the following sub-sections, we describe the simulation setup and some performance parameters analysis of our approach.

A. Simulation Setup

The simulated scenarios are characterized by the following parameters: the network size, the maximum target speed, the update frequency of the target mobility parameters, the standard deviation of the speed and the direction of the target as shown in Table I. The network topology is a grid in which node 0 starts the tracking process by activating its sensing interface and waking-up its cluster members. If there is no node within the zone covered by the node zero’s cluster that detects the target, the recovery procedure is triggered.

The success rate is the mean ratio between the number of successful predictions and the total number of predictions. Likewise, the target loss frequency is the average ratio between the number of successful predictions and the total simulation time. The re-capture time is the average period between the target loss instant and re-capture instant.

Figure 3 (a) shows an improvement of the success rate of the incremental prediction approach compared to the simple and the complex prediction variants, except in scenario 1. The simple prediction variant outperforms the complex one in scenario 4 because it does not require more historical information pieces to predict the next position. Indeed, communication and processing overhead generated by the complex prediction variant do not improve the success rate of predictions because it indicates far positions from the real target position, which, in this case, is closer to the estimated position. However, incremental prediction outperforms the two variants because, in this case, the number of prediction failures sent in every wakeup message does not increase. So, the simple predictions still persist from the beginning of the tracking process.

In complex scenarios 2, 3 and 6, the complex prediction variant outperforms the simple one thanks to the complex predictions generated in this case. So, the simple variant does not handle the variation of the target speed, which is very high in this case. On the other hand, the incremental prediction protocol is quickly adaptive to the target motion. Indeed, the tracking process starts by failing simple predictions then it increases the number of failure predictions in the historical information pieces in the wakeup messages.

B. Performance Analysis

We define three performance parameters, namely Success Rate, Target Loss Frequency and Re-capture Time.

The success rate is the mean ratio between the number of successful predictions and the total number of predictions. Likewise, the target loss frequency is the average ratio between the number of successful predictions and the total simulation time. The re-capture time is the average period between the target loss instant and re-capture instant.
So, the waked-up nodes will not explore only one prediction level but 2 or 3 levels, which helps at finding the target, given that the predictions of level $i$ are more accurate than those of level $i-1$.

The medium scenarios 1 and 5 show a decrease of the success rate in the incremental prediction protocol because of the static parameters used in the prediction mechanism.

The second performance parameter is the target loss frequency. It represents the system dynamics during the tracking process.

We can observe in Figure 3 (b) that the incremental prediction protocol outperforms the two variants in most of the scenarios namely: 2, 3, 4, 5 and 6. Indeed, in the complex scenarios, if the current prediction fails at finding the target in the incremental prediction protocol, then the node can still try with the next prediction level, which is more accurate than the previous one. The node declares the target lost only if all prediction levels fail.

The scenario in which simple and complex prediction variants outperform the incremental prediction protocol in term of the target loss frequency parameter is the scenario 1 where the motion complexity is medium. This can be explained by the static parameters of the prediction mechanism.

The third performance parameter is the re-capture time. It represents the protocol reactivity (recovery procedure) in presence of target trajectory changes. Figure 3 (c) shows that this parameter is slightly high in the incremental prediction protocol compared to the simple and the complex variants because of the additional time spent in the multi-level explorations. Thus, when a cluster-head receives a wakeup message, it wakes-up all its cluster members then it waits for detection messages. If there is no arrived detection message, the cluster-head tries with the next level and so on. If all the levels fail at finding the target, the recovery procedure is triggered. Consequently, the re-capture time will increase more and more before detecting the target.

C. Parametric Prediction Analysis

As we have seen in sub-section III.B, our approach does not improve performance parameters in medium scenarios 1 and 5 because of the static parameters used in the prediction mechanism. In order to evaluate the impact of prediction parameters on the system performance, we vary the ST and the FT thresholds described in sub-section III.B as follows:

- Case 1: ST=FT=6. The two thresholds are both high. To increase or to decrease the maximum number of prediction levels, the number of successful predictions and the number of failed predictions should both exceed 6.
- Case 2: ST=FT=2. Nodes need not to have a high number of successful or failed predictions to increase or decrease the maximum number of prediction levels.
- Case 3: ST=6 and FT=2. This is the basic version where parameters are fixed via simulation. This version is more sensitive to failures than to success predictions.

Table II shows the different evaluated scenarios. We measure two parameters, namely the success rate and the target loss frequency.

The scenarios 2 and 3 which are complex show the advantage of small success and fail thresholds in complex scenarios. In scenario 3, which is more complex than scenario 2, the basic version outperforms the two other versions thanks to the quick increase of the maximum number of prediction levels compared with its decrease.

However, in scenario 2, the version with ST=6 and FT=6 outperforms the two other versions because it’s a less complex scenario which does not require more reactivity.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Scenario type</th>
<th>size (Grid)</th>
<th>Max. speed (m/s)</th>
<th>Target mobility parameters' update frequency</th>
<th>Speed and direction standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Medium</td>
<td>7x7</td>
<td>20</td>
<td>3</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>Complex</td>
<td>7x7</td>
<td>30</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>More Complex</td>
<td>7x7</td>
<td>30</td>
<td>3</td>
<td>1.0</td>
</tr>
</tbody>
</table>
The target loss frequency results in Figure 6 (b) confirm those obtained for the success rate. The target loss frequency decreases when the fail rate decreases. Consequently, it decreases by the increasing of the success rate.

V. CONCLUSION AND FUTURE WORK

Prediction-based schemes for target tracking allow minimizing energy consumption but suffer from prediction failures, which reduce the system performance. To solve this problem, we have proposed a fault-tolerant prediction-based scheme that is dynamic and adaptive to the target motion complexity. Our approach uses historical information about the target motion and generates multi-level predictions that become more accurate when we add more historical information.

Parametric prediction mechanism allows the tracking protocol at increasing the success rate and decreasing the target loss frequency in various scenarios but slightly increases the re-capture time because of the successive explorations of the different prediction levels.

Thus, we can use, for future work, promiscuous mode at the MAC level protocol to reduce the tracking cycle period: capture-prediction-communication-capture, which can improve the re-capture time parameter.

Finally, energy evaluation using more complex prediction mechanisms can lead to a better understanding of the relationship between fault-tolerance and energy consumption in a target tracking protocol. Also, the dynamic parametric prediction mechanism is necessary in order to handle some variable mobility profiles of mobile targets.

REFERENCES