Tracking and Predicting Moving targets in Hierarchical Sensor Networks

Zhibo Wang
23rd June, 2008
Outline

- Introduction
- Related work
- Hierarchical Prediction Strategy
- Simulation
- Conclusion
- Experiments
- Future work
What is target tracking?

- Detect the presence of an object and determine its path in an area of interest
- Target tracking is an important application of Wireless Sensor Network (WSN)
Applications

- Target tracking has a wide variety of applications
  - Battlefield surveillance and military action
  - Transportation monitoring and management
  - Rare animal localization and management
  - Etc.
Limitations in WSN

- Wireless sensor network has its unique features, such as,
  - Limited power
  - Limited computation capacity and process capability
  - Constrained communication bandwidth

- Large-scale networks frequently suffer from packet losses, communication delays, energy limitation, etc.

- *So how to coordinate a large-scale network to efficiently track moving targets while conserve network resources is a great challenge*
Contribution

- Propose an energy-efficient Hierarchical Prediction Strategy (HPS) for target predicting and tracking in sensor network
  - Cluster-based hierarchical architecture
  - Dynamic sensor selected mechanism
  - Recursive Least-Square (RLS) prediction algorithm
  - Failure recovery mechanism

- Simulation shows this strategy achieve a nice tradeoff between tracking quality and energy consumption
Outline

- Introduction
- Related work
- Hierarchical Prediction Strategy
- Simulation
- Conclusion
- Experiments
- Future work
Related work

- **Geographical information-based target tracking**
  - Takes a traditional approach to target tracking: it dynamically divides the region of interest based on the target’s velocity and tracks multiple targets simultaneously by classifying them and associating each with a particular track.

- **Information utility-based target tracking**
  - Attempts to select the next sensor node that most likely results in “the greatest benefit at the lowest cost” for estimation.
Outline

- Introduction
- Related work
- Hierarchical Prediction Strategy
- Simulation
- Conclusion
- Experiments
- Future work
Network architecture

Cluster-based hierarchical network

*Global distributed with local centralized*
Assumptions

- All normal nodes (NNs) are able to communicate with their cluster head (CHs) after initial deployment, but they don’t communicate with each other.
- The cluster head can send orders to normal nodes and different cluster heads can talk with each other.
- The cluster head can know the location of nodes through an initial setup process.
- The cluster head has stronger computation power and infinite energy resource, so we don’t consider cluster head’s energy consumption.
- All normal nodes have an identical sensing range.
Initialization

- Normal nodes are deployed uniformly while cluster heads are deployed randomly.
- The locations of NNs and CHs are assumed globally known.
- Cluster are formed according to the locations of NNs and CHs.
- Every cluster assigns an identical transmission range to its NNs.
Cluster formed

- Employ voronoi diagram

Let $L_i$ be the location of node $i$ and $L_{H_i}$ be the location of CH $i$, respectively.

NN $i$ belongs to CH $k$

$$k = \min_k \| L_{H_k} - L_i \|$$

Then every CH broadcast a message to tell nodes who belong to it.

So every nodes know which cluster it belongs to.
In order to save energy and active every node in one cluster, we assign different transmission radius for clusters.

Transmission radius $R_t$ equals to the diameter of circumcircle of the local convex hull.
Working mode

- Cluster head
  - Idle
  - Active

- Normal node
  - Sleeping
  - Waiting
  - Tracking
Node detection model

- Binary detection model

\[
R_i = \begin{cases} 
1, & \|L_i - L_T\| \leq R_s, \\
0, & \|L_i - L_T\| > R_s. 
\end{cases}
\]

**Advantages:**

- Minimal assumption about the sensing capability
- Minimal communication requirements
- Save bandwidth and energy
Energy consumption model

- Three basic energy consumption types—**sensing**, **transmitting** and **receiving**
- Here we just consider the transmitting energy consumption while ignore sensing and receiving energy, because they are highly application dependent (not reasonable)
- Assume transmitting energy consumption is a function of distance $E = E_k \cdot d^2$ (while $E_k$ denotes the energy consumption that one node transmit one message over a distance of one meter)
Performance metrics

- **Tracking error (Q)**
  - Euclidean distance: \( q(t) = d(L_a(t), L_e(t)) \)
  - At period \( T \):
    \[
    Q = \int_0^T q(t) \, dt
    \]

- **Energy consumption**
  - All the consumption equals to the sum of every node consumption during all the tracking process:
    \[
    E(1) = E_k \times d_1^2 \times n_1 + E_k \times d_2^2 \times n_2 + E_k \times d_m^2 \times n_m
    \]
    \[
    E = E(1) + E(2) + \ldots + E(n)
    \]
Location algorithm

- Centroid localization

Node location:

\[ L_{i,j}(x_{i,j}, y_{i,j}), j = 1, \ldots, k. \]

Estimated target location:

\[ L_e(t) = \left( \frac{\sum_{j=1}^{l} x_{i,j}}{l}, \frac{\sum_{j=1}^{l} y_{i,j}}{l} \right) \]

- sleeping node
- Active node sensing target
- Active node can not sensing target
Prediction algorithm

- Targets are maneuvering, sometimes escaping, so how to keep target under tracking all the time and pre-activate nodes to perform tracking and save energy?
- Recursive Least Square (RLS) Prediction algorithm are employed here
- We can predict the “next” position of the target and hand over tracking to nodes best placed to track the target in the “next” position
- So there are only a small subset of them are in active mode at any time
- Prediction algorithm can greatly save energy and reduce communication delays
Adaptive algorithm: function of learn and track, update input vector

Get the optimal linear filter according to the input vector, so the filter is “accurate”

The aim is to minimize the Standard deviation
Rls prediction

Input Signal: $u(n), d(n)$

Parameters: $p = \text{filter order}$

$\lambda = \text{forgetting factor}$

$\delta = \text{value to initialize } P(0)$

Initialization: $w_n = 0$ \hspace{1cm} $(M \times M)$

$P(0) = \delta^{-1}I$ \hspace{1cm} $(M \times M)$

Computation: For $n = 1, 2, \ldots$

$M \times 1 \hspace{1cm} \pi(n) = P(n-1)u(n)$

$M \times 1 \hspace{1cm} k(n) = \frac{\pi(n)}{\lambda + u^H(n)\pi(n)}$

$1 \times 1 \hspace{1cm} \varepsilon(n) = d(n) - w^H(n-1)u(n)$

$M \times 1 \hspace{1cm} w(n) = w(n-1) + k(n)\varepsilon^*(n)$

$M \times M \hspace{1cm} P(n) = \lambda^{-1}P(n-1) - \lambda^{-1}k(n)u^H(n)P(n-1)$

$1 \times 1 \hspace{1cm} u(n+1) = w^H(n-1)u(n)$
Tracking procedure

At Each Sensing Circle $T_{sc}$
1. *Subleaders* calculate local result and send to *leader*
2. *Leader* $CH_l(t)$ calculates $L_e(t)$
3. Predict $L_p(t + 1)$ using RLS
4. Assign new leader $CH_l(t + 1)$ according to $L_p(t + 1)$
5. Deactivate $NN(t - 1)$
6. Activate $NN(t + 1)$, each with a probability of $p$
7. Assign *Subleaders* according to the activated NNs
Overview

Tracking process demonstration
Failure recovery algorithm

- In case of less than 3 NNs report sensing results to CHs, which we consider the target is nearly or already lost. The failure recovery is invoked.
Outline

- Introduction
- Related work
- Hierarchical Prediction Strategy
- Simulation
- Conclusion
- Experiments
- Future work
Simulation environment setup

- 2500 NNs are deployed in a 50m x 50m area
- Each NN is assigned an identical sensing range $R_a = 2m$
- One target is deployed and a moving trajectory is designed

[Diagram showing target trajectory]
Tracking error vs. $R_a/R_s$
Energy consumption vs. $R_a/R_s$

![Graph showing energy consumption vs. $R_a/R_s$. The graph includes different lines for various percentages, with Cluster Number=30, $R_s=2m$, $v=0.25m/s$, Sensing Circle=2s.]
Tracking error vs. Number of clusters

Rs=2m, Ra=1.5Rs, Sensing Circle=2s

Tracking error vs. Number of clusters
Energy consumption vs. Number of clusters

Rs=2m, Ra=1.5Rs, Sensing Circle=2s

Energy consumption vs. Number of clusters
Tracking error vs. Sensing circle

- Cluster Number=30, Rs=2m, Ra=1.5Rs, v=0.25m/s
- Tracking error vs. Sensing circle

Graph showing the relationship between tracking error and sensing circle size with different probabilities (p=40%, p=60%, p=80%, p=100%) for a specified cluster number, radius, and velocity.
Energy consumption vs. Sensing circle
Outline

- Introduction
- Related work
- Hierarchical Prediction Strategy
- Simulation
- Conclusion
- Experiments
- Future work
Contributions

- Our strategy bases on a cluster-based hierarchical network architecture, which can easily organize a large-scale network.
- Compared with homogenous network, it is more steady and scalable, and also with shorter time delay.
- The strategy can predict the target trajectory and selectively activate the most suitable nodes to perform tracking, which greatly save energy and reduce tracking error. So we can easily track the moving target and prolong the network lifetime.
- Failure recovery mechanism is also considered in this strategy.
- Some models are overly simplified and need to be changed or improved
Outline

- Introduction
- Related work
- Hierarchical Prediction Strategy
- Simulation
- Conclusion
- Experiments
- Future work
Experiments

- Associate work with Jianfeng Li
- Successfully tracked a target in the sensor network with one cluster, even the target change sharply at different speed.
- Displayed the dynamic tracking process on PC
- Tracking error is between 20cm and 30cm, and almost at every time, the true position and the estimated position of the target are in the same grid
Outline

- Introduction
- Related work
- Hierarchical Prediction Strategy
- Simulation
- Conclusion
- Experiments
- Future work
Future work

- **Scheme 1**
  - Improve our strategy to make it more reasonable
  - Compare our strategy with other algorithms
  - Explore the relationship of node density, sample time, target speed and tracking error
  - Experiments testify

- **Scheme 2**
  - Find a new idea
  - Multiple targets tracking
  - Control method and analysis will be added