Introduction

Target tracking in sensor networks is the problem where at least one target exists in a monitored by sensors environment, while the sensor network users want to localize that target. Depending on the scenario, a typical application could have the users mobilized while localizing the target, or at least one mobile agent localizing the target and sending reports back to the sensor users who again might be mobile.

Rationale

Most algorithms use the sensor network to actively participate in the tracking process, sensing the target and propagating information regarding its position. The trade of, in this case, is a high message and energy consumption overhead. There are also existing issues concerning the scalability of those solutions and how they can be mapped into low density sensor networks. In addition there are also issues of inverse tracking, in the case of a mobile center/user that the agent has to report to.

Algorithm

This scenario will propose the target tracking protocol discussed in [1], [2], [3] also know and as THTP (Trace Handling Tracking Protocol). The basic concept is analogous the way a lion (mobile agent), tracks an antelope (target) through scent (traces) that are spread throughout the environment (sensor network). The result is a lightweight protocol that operates with low message overhead, low power consumption while scaling well in:

- Low network densities,
- Various moving target patterns
- High or low duty cycle nodes, where communication between them suffers from low to high delays.

The scent spreading or trace propagation throughout the sensor network works as follows. As soon as the sensor nodes, representing the environment, detect the presence of a target in their vicinity, they initiate a trace of maximal intensity which value will diminish (linearly) with time. The trace itself is a collection of dynamic or static attributes.

Attributes	**Possible Values**
Type	Initial or spread.
Start_time	Time at which the trace was initialized.
Start_intensity	Initial intensity of the trace.
Path	A list of 1 to 3 parent traces
Max_intensity	300
TargetID	Used for the tracking of multiple targets

A trace is always propagated to two chosen neighboring nodes. The choice for the first node is deterministic and for the second node randomized. The deterministic approach wants the first chosen node to be the neighbor that is further away from the repulsion point. The repulsion point is actually the grandparent of the node in the trace propagation process. If the path is too short then the repulsion point can be the parent or even the node itself from the path attribute. Each of these two selected nodes will choose again another pair of neighbors to spread the trace, and so on. In order to control the trace spreading and minimize the number of messages sent, before a node spreads a trace to its neighbors, it queries them, so that it only spreads the trace when the neighbor holds a trace of lower intensity. In each trace that is spread, the intensity is updated with a spread penalty. When the sensor network user wants to track a target, he initiates a tracking agent who firstly executes a random walk within the communication graph, marking nodes that have been visited. Once a node holding a trace is found the agent greedily moves towards the node that holds the highest intensity trace. When the node reaches the trace that holds the type "initial", it will report back to the user. When traversing this binary tree, in the case of reaching a local minimum node, a backing mechanism kicks in that marks the local minimum node as a "bad" node. This node will not be visited again (until

the "bad" marker is removed from a fresher trace). Next the agent moves greedily to the node with highest intensity. When the target is found, the localization info of the target must reach the network user/base station. This is fairly easy if the destination has limited mobility, since the tracking agent can remember where it was initialized by the network user. More complicated could be the fact that the base station is highly mobile. In this case the tracking protocol addresses this as an inverse tracking scenario where the base station is treated as the target.

Experiments

The above algorithm was tested through simulated experiments with the following default values:

Semantic	**Symbol**	**Default value**
Number of nodes	*n*	300 sensors
Communication radius	d_{trx}	100 meters
Sensing radius	d_{dtx}	25 meters
Target speed	*detection_radius*	6 km/h
Message propagation frequency	*freq*	1 MSG/second
Network density	*dens*	$10(100^2 \cdot 3.14)$* nodes per m^2

To create different scenarios individual parameters as the network density, target speed, detection radius, message propagation frequency and running time where varied accordingly. The respective results show that the algorithm scales well in low density networks. Also when the message frequency is below the threshold of 20 seconds, the target is localized successfully in a timely manner.

Challenges – Future work

Parameter-centric
From the above experiments, we can observe that the search space of possible permutation of variable parameters, mutations or even the addition of other external parameters that could deviate, challenge or strain the existing results, is large.
Environment-centric
One important aspect that could be explored is the addition of extreme environmental conditions that could alter the static environment such as introducing new obstacles in the network or adding some sort of limited mobility (randomized or patterned) or even failure ratios to the nodes.
Target-centric
In real life scenarios there are existing targets that are required to be tracked and also follow certain mobility patterns. (An antelope is expected to move to a pool of water, moves at a specific speed range and follows a specific biological schedule). One challenge could be the introduction of mobility prediction, or even adaptive mobility prediction based on the type of target. Instead of one hop to the neighbors of a node, the tracking agent can perform double or triple hops base on prediction data. This will also introduce a faster tree traversal but increased backtracking in case of an error.
Data-centric
In the case of low message frequencies, the ratio of the number of nodes selected for trace propagation to the actual number of traces propagated is low. That means the tree that the tracking agent will be traversing is not expanding fast enough. A possible optimization would be to increase the level of the tree (instead of 2 neighbors selected per node there can be a selection of 3 or 4 based on the value of the trace propagation frequency). This could introduce the same exploitation of environmental resources (nodes) on the same amount of time.
Agent-centric
Exploring the scenarios where two or more tracking agents can exchange information through the network while tracking the same or different targets. For instance agent01 and agent 02 can follow the same target and tags from

one agent in a node can inform the other about fruitless searches in the tree. Or in the same fashion agent01 can get information about target01 from traces that agent02 happened to leave in the agent01 tree.

[1] O. Powell. Passive and lighweight target tracking for sensor networks.In Invited paper in the ProSense adjunct workshop proceedings of DCOSS 08, 2008.
[2] Andrei Marculescu, Sotiris E. Nikoletseas, Olivier Powell, José D. P. Rolim: Efficient Tracking of Moving Targets by Passively Handling Traces in Sensor Networks. GLOBECOM 2008: 271-276
[3] A. Marculescu, S. Nikoletseas, O. Powell and J. Rolim, Lightweight target Tracking Using Passive Traces in Sensor Networks, (2008) Computing Research Repository (CoRR) of Distributed, Parallel and Cluster Computing, ref. cs.DC/08032219.