Paper Sensor Network Intruder detection

Abstract:

We will show that it is possible to detect an intruder based on changing RSSI values of deployed sensor nodes inside a room. Furthermore the presented method is capable to differ between moving and standing inside the given room.

Introduction:

In this paper we want to describe a way to observe a given space by measuring the connectivity of deployed sensor nodes. We will present a set of functions to convert the given signals into a KNN cluster that can be used to differ between 3 possible states:

The Room is empty, someone is moving through the room or standing still.

For our measurements we used the RSSI values that are generated between each Sensor node (in this paper we will referee to this as link) and distributed to a base station that will then give the information to the PC where it is processed.

There are some difficulties when using the RSSI value of Sensor nodes; the result is often not stable and the behaviour is often hard to determine as the signal strength does not always (but sometimes) match expectations.

For the Data cleansing we will use an algorithm called Alpha-Trimmed Mean Filter that will eliminate some of the noise and will the cluster the values.

We use the K-nearest-neighbour algorithm to cluster the RSSI values of each link based on different training data (empty, walking, standing); This allows us to get a live view of the current state for each link and based on that results in a statement about the whole situation in the room.

Materials and Methods

In this paper we used 9 Telos-B sensor node running Tiny-OS to gather RSSI values. 8 of the sensors do communicate between each other and distribute there RSSI values to the 9th ; this sensor node does not gather RSSI values but will provide the collected data to the pc.

The whole process contains the following steps:

I. Collection data on the sensor node

II. Cleaning each link data with Alpha-Trimmed-Mean-Filter

III. Clustering each link data with KNN

IV. Combining each link data to one resulting state

I.Collection Data

The basic code idea is to call a certain function each time a given amount of time has passed ( in our example we used <Hier bitte eine Zahl> milliseconds)

Integer \_RSSIValues[Amount of Nodes];

…

receivedMessage(message\* m)

{

\_RSSIValues[m->nodeId] = getRSSI(m);

}

…

timeFired()

{

message\* m();

m->setPaketData(\_RSSIValues);

broadcast(m);

}

As a result we will receive an array for each sensor node containing the current RSSI value from that node to each other node at index = otherNodeID. (The RSSI value to oneself is always 0 as it is never set).

II. Alpha-Trimmed Mean Filter

The received array will be spitted into link information containing 4 values:  
sourceNodeId, targetNodeId, RSSIValue and timestamp. Each link is the given to it’s own Alpha-Trimmed Mean Filter (where sourceNodeId->targetNodeId ⬄ targetNodeId->sourceNodeId).

The basic idea of the ATMF is to select the newest n values of the data and sort this data in a nondecending order. Using a given alpha value you can then determine to amount of points to be trimmed at each side using the formula: (alpha\*n)/2. After removing this amount of points you take the average of the remaining.

This value will then be given to the clustering algorithm.

III. Clustering

As mentioned we uses the KNN algorithm to cluster the given values to determine the correct state.

Basically this is achieved by collecting training data for 3 possible situation pattern ( standing, walking, empty) and divide all processed data into packets sized p. For each packet we will determine the mean and variance and put the results in an 2 dimensional graph with x being the mean value and y being the variance. We do receive a graph as shown: <Bild hier>

After the system has been trained with the data we can start the observation of the room; Incoming link information use there mean and variance based on the ATMF to be temporarily placed inside the graph. We do then determine the nearest data Points and retrieves there type (walking, standing, empty). This classification is then passed to the combining link phase.

IV. Combining link information

<Not yet done>

Results

Discussion

Summary

Acknowledgements

References