Author
Arpan Govindraju (a.govindaraju@student.tudelft.nl)
Title
TODO TITLE
MSc presentation
5th July 2016

Graduation Committee

TODO GRADUATION COMMITTEE	Delft University of Technology
TODO GRADUATION COMMITTEE	Delft University of Technology

Abstract

TODO ABSTRACT

Preface

Sesnor localization is one of the

I would like to thank my supervisor Ashish Pandharipande(Philips Research), David(Philips Research) , Venkatesh Prasad(TU Delft) for all the support and guidance provided during the course of this 9 months. I would also like to thank my parents for all the support and love they have provided me all these years. A special thanks to all the people of FO 52 past and the present who made the stay in Eindhoven so much more pleasant.

Arpan Goindaraju

Delft, The Netherlands
5th July 2016

Contents

Chapter 1

Introduction

The European Union (EU) has pledged to cut the consumption of primary energy by 20% by the year 2020. It is estimated that buildings consume 40% of the energy produced[1]. This has resulted in an increase in the demand to reduce the energy consumption of buildings. To reduce the consumption of energy, building automation systems (BAS) are being widely employed. BAS are computer-based systems that help to manage, control and monitor building technical services (HVAC, lighting etc.) and the energy consumption of devices used by the building. BAS deploy huge amount of sensors, which provide inputs to perform efficient control of various services like HVAC and lighting. BAS brings with it various benefits,at the same time, offers numerous challenges too.One of the primary challenges is generating and updating the locations of the installed sensors. For performing effective control, information about the location is highly important. As the size and distribution of the deployed sensors are high, it is highly cumbersome and error prone to manually maintain the meta-data about the sensor placement. Also as building evolve and change managing this spatial information becomes a cumbersome process. Hence to ease the process of sensor location verification an automatic process to locate the sensors in a building is required.

[1] according to value published at https://ec.europa.eu/energy/en/topics/energy-efficiency/buildings

Chapter 2

Literature Review

2.1 Sensors Localization in Buildings

Various approaches have been taken to automate the process of determining the sensors location in buildings using various data analytics and signal processing tools. Hong et al.[5] apply empirical mode decomposition to 15 sensors in 5 rooms to cluster sensors which belong to the same room by analyzing the correlation coefficients of the intrinsic mode functions. They characterize the correlation coefficient distribution of sensors in the same room and different rooms and show that there exists a correlation boundary analogous to the physical boundary and can be discovered empirically. In [1] Akinci et al. propose a feature: energy content in HVAC delivered air, which can be derived from HVAC system sensors which could lead to identification of the space in which the sensors are located . They combine sensor measurements and building characteristics(floor area) for the identification of the space in which the sensors are present.
Lu et al.[6] describe a method to generate representative floor plans for a house. Their method clusters sensors to room and assigns connectivity based on simultaneous firing of the sensors placed on the door and window jambs. The algorithm gives a small set of possible maps from which the user has to choose the right map. The authors were able to calculate the floor map of 3 out of the 4 houses they evaluated. There method requires special placement of the sensors . Ellis et al.[3] proposed an algorithm to compute the room connectivity using PIR and light sensor data. They compute room connectivity based on the artificial light spill over between rooms and occupancy detection due to movements between two rooms. They calculate the transition matrix for light sensor and occupancy sensor. Fuse both the data together to compute the connectivity graph. Here the authors have considered a situation where there is only one PIR and light sensor per room. Müller et al. define sensor topology as a graph with directed and weighted edges. All pairs of consecutive sensors triggers are are interpreted as as a

user walking from the former to the latter sensing region indicated in the sensor graph by a edge from the former to the latter. Every time a consecutive edge triggers are observed the weight of the edge between the sensors are incremented. They define a method to filter out erroneous edges based on the relative frequency of use of all the outgoings edges per node.

2.2 Subgraph Isomorphism

In our work we reduce the problem of mappings the sensors to its location in the grid to graph monomorphism problem. Graph monomorphism is widely used in pattern recognition for comparing the graph representing an object to a model graph, or prototype. Various algorithms have been developed so far to solve the problem of graph monomorphism. In our work here we make use of the VF2[2] algorithm to solve for monomorphism.

Chapter 3

Methodology

This chapter explains the method developed to locate the sensors in a sensor grid, using the information about the locations of the vertices of the grid and data obtained from the sensors. Before we explain the method we introduce the definitions of the terms that are used to explain the method.

Definition 3.0.1. Neighboring sensors: Two sensors are said to be neighbors if they have overlapping field of view.

Definition 3.0.2. Grid Adjacency Matrix: We represent the sensor grid in the form of an adjacency matrix. Two vertices of the grid i and j are adjacent if the distance between them ($d_{i,j}$) is less than twice the radius(r) of the sensors field of view.

$$GAM_{i,j} = \begin{cases} 1, & \text{if } d_{i,j} < \ 2\text{r } \forall \ i \neq j \\ 0, & \text{otherwise} \end{cases}$$

From the definitions we can say that neighboring sensors will always be placed on neighboring vertices.

Consider a m $\times$ n grid as shown in the figure 3.1. There are $N = (m \times n)!$ ways in which the sensors can be uniquely placed on the grid. Out of this N ways we have to find out the actual arrangement of the sensors in the grid. Along with the information of the vertex locations we also have the data from the sensors available. With the help of sensor data we need to determine the sensor location.
As neighboring sensors have an overlapping field of view they observe the same events and thus are highly correlated. On the raw signals of the PIR data stream we use a sliding window with 50% overlap and calculate the energy using the equation 3.1

$$E_s = \sum_{n=0}^{k} |x(n)|^2 \tag{3.1}$$

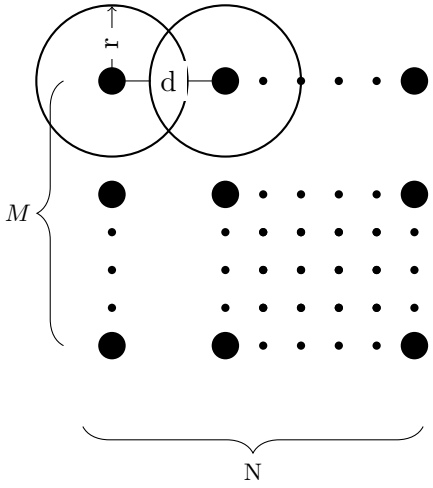


Figure 3.1: A M × N grid having a sensor with field of view r and distance between the nodes being d.

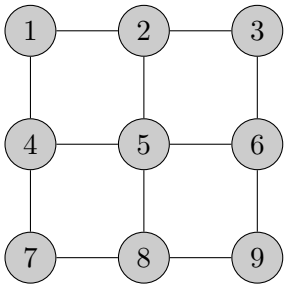


Figure 3.2: Correct arrangement

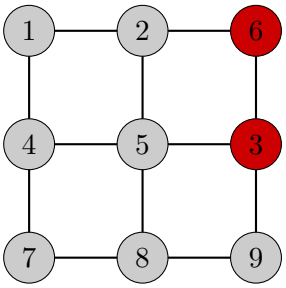


Figure 3.3: Incorrect arrangement

With the newly computed energy data stream, compute the cross correlation between all the sensors using equation 3.2

$$r(x,y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{3.2}$$

X and Y are sensor data stream for sensor X and Y.
$\overline{X}$ and $\overline{Y}$ is the mean value of X and Y respectively.
n is the number of samples.

3.1 Grid Correlation Sum

Using the cross correlation values between the sensors, we compute correlation matrix R between the n sensor nodes as shown in the equation 3.3

.

$$R = \begin{bmatrix} r(1,1) & r(1,2) & \dots & r(1,n) \\ r(2,1) & r(2,2) & \dots & r(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ r(n,1) & r(n,2) & \dots & r(n,n) \end{bmatrix} \tag{3.3}$$

r(α,β) representing correlation value between the sensor α and β.

Using correlation matrix(R) and grid adjacency matrix(GAM) we define a parameter called **Grid Correlation Sum (GCS)** as given in the equation 3.4. As the matrices are symmetric along the diagonals we only consider the product of the upper triangle of the matrices.

$$\text{GCS} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{R}(\phi(i), \phi(j)) \times \text{ GAM(i,j)} \tag{3.4}$$

i,j represent i^{th} and j^{th} vertex on the grid.
$\phi(i)$ represents the sensor on vertex i.
R(a,b): correlation coefficient between the sensor a and b.
Grid Adjacency Matrix: Adjacency matrix of the grid as definition.

We use GCS parameter to identify the correct arrangement out of all the possible arrangement. If we compute GCS for all the possible arrangements, the one with the maximum correlation sum will represent the actual arrangement of the sensors on the grid. If two non neighboring sensors are kept on neighboring vertex or vice versa then the correlation value between those two sensors will be low and thus decreasing the overall sum of the correlation over the grid. For example Consider the arrangement of the sensor grid in the figure 3.2. It consists of 3 $\times$ 3 grid. GCS for the the grid is :

$$GCS_1 = \text{r(1,2)+ r(1,4)+...r(3,2)+...r(5,6)+...r(8,9)}$$

Now consider the arrangement shown in figure 3.3, where the position of the sensor 3 and 6 are interchanged

$$GCS_2 = \text{ r(1,2)+r(1,4)+...r(6,2)+...r(3,5)+...r(8,9)}$$

GCS_1 will be greater than GCS_2 as r(2,3) > r(2,6) and r(5,6)>r(5,3) as sensors 2,6 and 3,5 are non neighboring sensor nodes.
When the number of sensors is low we can compute the correct mapping by checking the sum of correlation value for the grid for all the possible N ways. As the number of sensors increase the possible mappings to be checked also increases. We need to find a way to decrease the possible mappings.

3.2 Maximum spanning Tree

Having established a method to determine the true arrangement of the sensor nodes on the grid from all the possible **N** arrangements, the next step is to prune the possible arrangements.
From the definitions of neighboring sensors and neighboring vertices we can say that two neighboring sensors will always be placed on the neighboring vertices. From the grid adjacency matrix we can determine the neighboring vertices of a vertex. If we can determine the neighboring sensor nodes then we can prune the number of possible arrangements by using the constraints that only neighboring sensors can occupy the neighboring vertices.

Though the correlation value between two neighboring nodes are high compared to the correlation value between the non neighboring nodes, finding a threshold to differentiate between the neighboring and non neighboring nodes when the arrangement is not known is non trivial.
Although we might not be able to identify all the neighboring nodes even if we are able to find a minimum of one neighboring node per node we will be able to reduce the number of mappings that needs to be checked. To achieve this we calculate the maximum spanning tree for the correlation matrix **R**.

If we consider R as an adjacency matrix with each sensor representing a vertex and the correlation value between the sensors representing the edge weight between them. Then the maximum spanning tree for such a graph will connect all the vertices together such that the total weight for the edges in the tree is maximum. The important property of the maximum spanning tree is that

- Covers all the sensor nodes
- It connects a node to another node with which it has the maximum correlation coefficient value.

Computing the maximum spanning tree has an edge between 2 nodes which have high correlation matrix compared with the other nodes. Therefore if an edge exists between two nodes in the maximum spanning tree then we can say that those two nodes are neighboring nodes.
To compute the maximum spanning tree we use Prim's algorithm [8].Originally Prim's algorithm is designed to compute the minimum spanning tree for a graph. In order to compute the maximum spanning tree the correlation matrix is negated and given as the input to the algorithm. The minimum spanning tree obtained from negated weights is the maximum spanning tree for the original weights.This maximum spanning tree represents one of the spanning tree for the grid as can be seen from the figure 3.4.

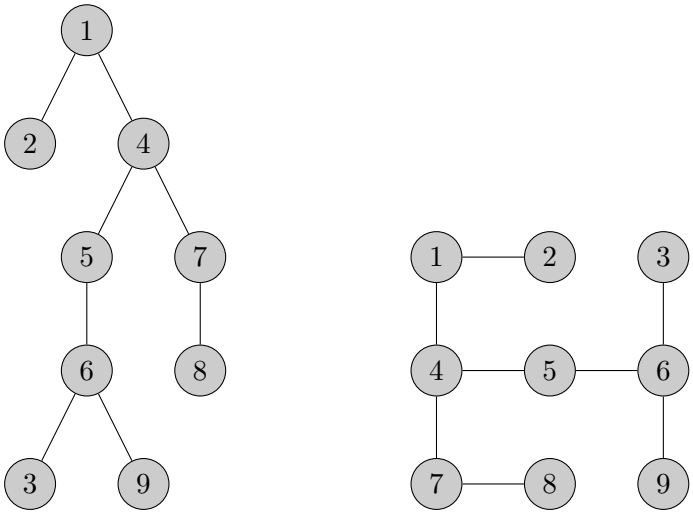


Figure 3.4: A maximum spanning tree incident on the grid

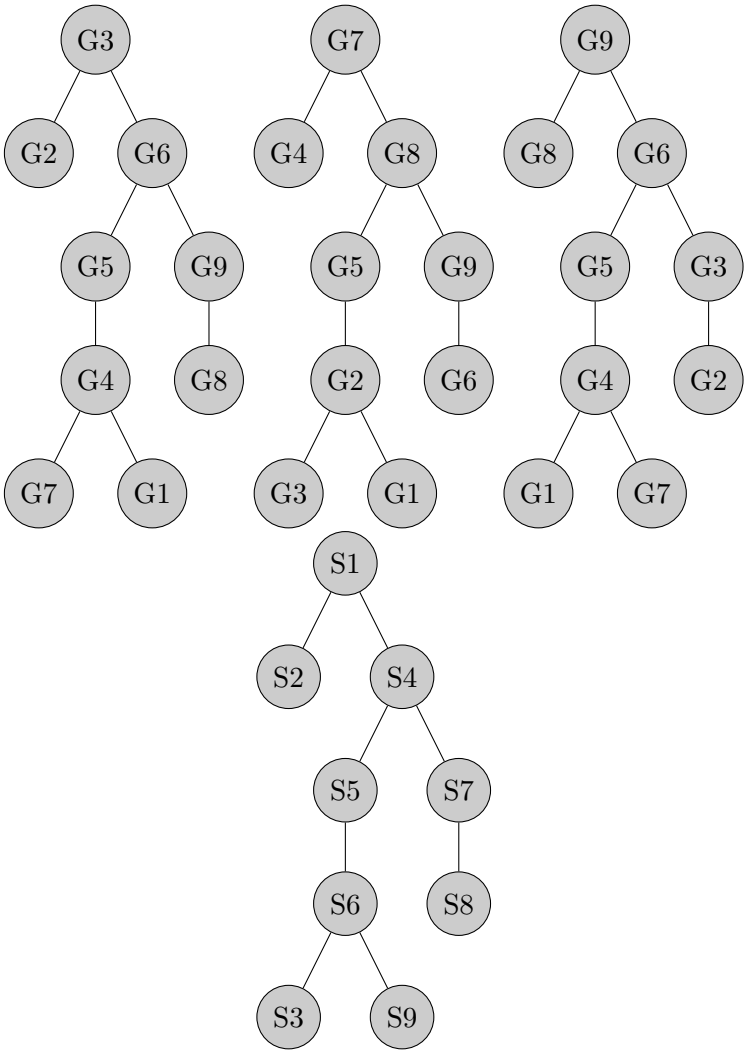
3.3 Graph Monomorphism

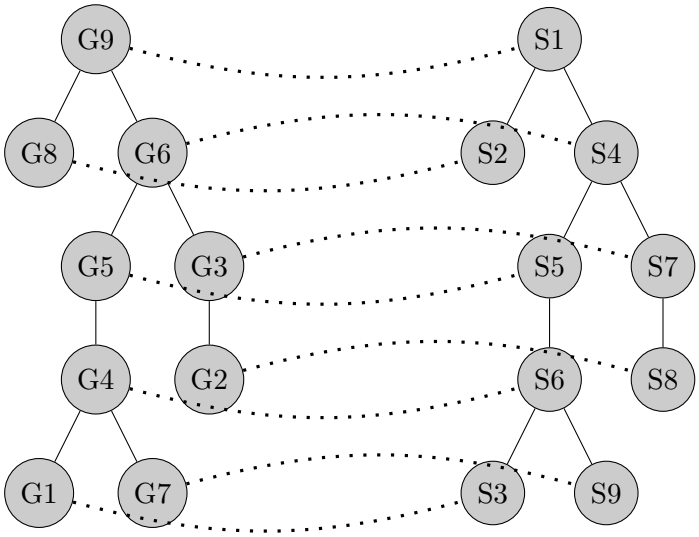
In the previous section we obtain a maximum spanning tree for the correlation matrix and saw how this represents a spanning tree for the grid. In this section we describe how to obtain the sensor location on the grid using the maximum spanning tree. We can classify the problem of obtaining the mapping between a spanning tree of a graph G to itself as a problem of graph Monomorphism.

Let us consider a base graph G(M,A) and a pattern graph H(N,B) with vertex set M,N and edge set A and B respectively. A problem of finding all the Monomorphisms of the pattern graph into the base graph is defined as Monomorphism problem.

Two graphs $G = (V_G, E_G)$ and H $=(V_H, E_H)$ are monomorphic if and only if there exists an injective (node) mapping $\phi V_G \rightarrow V_H$: for which $\forall v, w \in V_G : (v, w) \in E_G \Rightarrow (\phi((v), \phi(w)) \in E_H$.
holds true.

Monomorphism is often confused with subgraph isomorphism. Monomorphism is weaker kind of subgraph Monomorphism.
Two graphs $G = (V_G, E_G)$ and H $=(V_H, E_H)$ are sub graph isomorphic if and only if there exists an injective (node) mapping $\phi(V_G) \rightarrow V_H$: for which $\forall v, w \in V_G : (v, w) \in E_G \Leftrightarrow (\phi((v), \phi(w)) \in E_H$.
holds true. The relationship between edges of the graphs are equivalence for subgraph isomorphism, and for Monomorphism relationship is an implication. It is well known that Graph Monomorphism is a well NP-Complete problem [4]. As discussed in section **??** there have been several algorithms that have been proposed to solve the problem of Monomorphism. In our work we make use of the VF2 algorithm.


G3
G2
G6
G5
G9
G4
G8
G7
G1
G7
G4
G8
G5
G9
G2
G6
G3
G1
G9
G8
G6
G5
G3
G4
G2
G1
G7
S1
S2
S4
S5
S7
S6
S8
S3
S9



3.3.1 VF2

In this section we give a brief description about VF2 algorithm.
Pseudo-code
Consider the graphs shown in figure **??** representing

VF2 algorithm is an algorithm to solve subgraph isomorphism problem. VF2 is a depth search first algorithm. It uses recursive backtracking technique. A process of matching the grid graph G to its spanning tree consists of determining a mapping M which associate nodes of $Grid(G)$ with nodes of the Tree(T) and vice versa, with some constraints. Mapping is expressed as a set of pairs (n,m) with (n $\in$ G and m $\in$ T). In VF2 algorithm the process of finding a mapping is represented as a State Space Representation (SSR). Each state s of the matching process can be associated to a partial mapping solution M(s), which contains only a subset of M. A transition from current state(s) to the next state(s') represents the addition of a mapping (n,m) to the state s.

VF2 algorithm introduces a set of rules which helps to prune the number of possible SSR that needs to be checked before obtaining the a valid mapping. Figure 3.5 gives a high-lvel description of the VF2 algorithm. There are 2 important functionalities in the algorithm one is the generation of possible mappings and the other is the checking of the validity of the mapping.

3.3.2 Computation of candidate pair set P(s)

This section explains the method to compute the candidate pair set P(S) for an undirected graph G and T. Set $T_1(s)$ and $T_2(s)$ are defined for Graph G and T respectively, representing the nodes which are neighbors of the set of nodes including in the partial mapping state(s). Set P(s) will be made of

Procedure Match(s)
INPUT: an intermediate state s; the initial state s_0 has $M(s_0) = \emptyset$
OUTPUT: the mappings between two graphs¿ **Match(s)**
IF M(s) covers all the nodes of G_2 **THEN**
OUTPUT M(s)
ELSE
Compute the set P(s) of the pairs of candidates for inclusion in M(s)
FOREACH (n,m) ∈ P(s)
IF F(s,n,m) **THEN**
Compute the state s' obtained by adding (n,m) to M(s)
CALL Match(s')
ENDIF
ENDFOREACH
Restore data structure
ENDIF

Figure 3.5: Pseudo code for VF2 algorithm

all the the node pairs(n,m) with n∈ G and m ∈ T. If either of T_1 or T_2 set is empty then the set P(s) will the pair of nodes not contained in either G(s) or T(s).

3.3.3 Feasibility Rules

Feasibility Rules are used to check the consistency of the partial solution s' obtained by adding nodes n,m and prune the search tree. The functionality of the rules are as explained below. Gata graph (G), Querry graph(Q) In vf2 algorithm, those candidates(m,n) are pruned if m is not connected to already matched nodes in G_q (i.e nodes of G_q included in M). Subsequently, the pruning step also removes those node-pairs(m,n) in which n is not connected to the matched nodes in the data Graph G. These pruning rules assume that the query and data graph are connected.

The algorithm also compares the number of neighboring nodes of each n and m that are connected to nodes in M but are not included in M. The number of such nodes in the data graph must be greater than or equal to the number of such nodes in the query graph. finally the number of neighbors nodes of each of n and iq that are not directly connected to nodes in M are compared. The number of such nodes in the data graph must be no smaller than the number of such nodes in the query graph.

3.4 Determination of sensor placement

After we obtain the various mappings. We need to decide which out of the various mappings gives the actual placement of the sensors on the grid. To identify the correct mapping we compute GCS as explained in section 3.1. We calculate the grid correlation sum for all the mappings obtained and the mappings which gives the maximum sum will be the actual placement of the sensors on the grid.

Chapter 4

Test Bed

We setup our testbed in a 8m X 6m office space. Sensors are mounted on the ceiling which is at a height of 3m. The sensors are placed 2.5m apart in the horizontal direction and 2.2m apart in the vertical direction. We used PIR sensor EKMB1101112 from Panasonic. The output of the sensor is binary. With 1 indicating occupancy and 0 if the region is unoccupied. We use eZ430-RF2500 toolkit which consists of MSP430 micro-controller and CC2500 multi channel RF transceiver which is designed for low power wireless applications. The testbed uses simpliciTI a a TI proprietary low-power RF network protocol. To avoid collision we employ CSMA/CA. To decrease the packet loss we enable acknowledgment. An acknowledgment is sent from the AP if the packets are received. If no acknowledgment is received by a node, the node retransmits its data a maximum of 6 times until an acknowledgment is received from the AP. In the figure 4.2 we can see that the there was a decrease in the packet loss per node. The maximum packet loss before the enabling acknowledgment was 23%, after acknowledgment was enabled maximum packet loss for a node was 2%.

We place 8 sensors in the room as shown in the figure 4.2. All the 8 nodes communicate directly to a centrally located access point which is connected to a laptop, which stores the data. We sample the PIR sensor

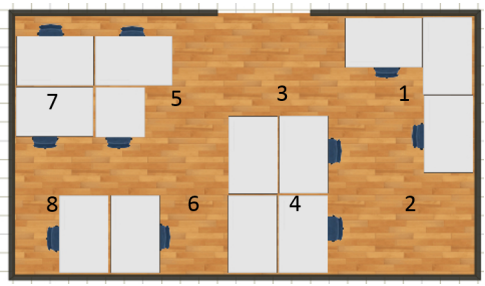
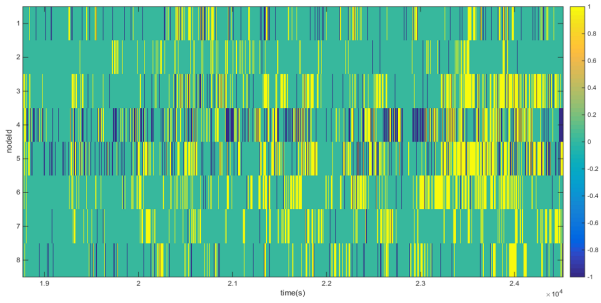
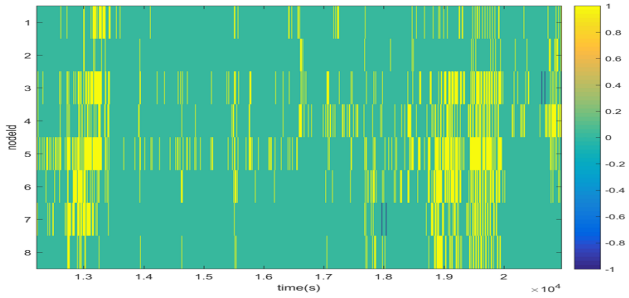


Figure 4.1: Room layout of the testbed.



(a) Data before ack was enabled



(b) Data after ack was enabled

Figure 4.2: Data from the test bed. - 1 indicates the lost data , 1 indicates occupancy, 0 indicates unoccupied space

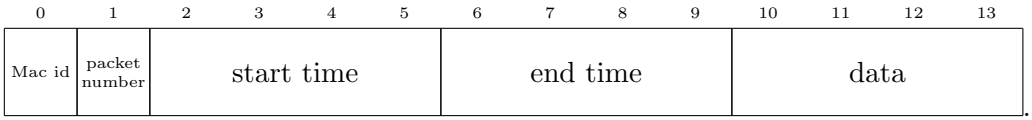


Figure 4.3: Packet Structure

every 100ms. All the nodes have a local clock running on them. With 1 clock tick corresponding to 1ms. All the local clocks are synchronized to one global clock.

Since the data is binary we combine 32 samples and transmit the packet every 3.2s. Data packet is an array of 8 bit. The packet structure is as shown in the figure 4.3

The transmitted packet is an array of 1 byte. Consisting of 14 bytes of payload. The payload structure is as shown in the figure 4.3

- Mac Id - unique identifier for each node.
- Packet number- Keeps track of the packets that are being transmitted from a node. It helps to identify the lost packets as well as repeated packets.

- Start time - The time stamp corresponding to the first PIR sample of the PIR sensor in the packet.
- End time - The time stamp corresponding to the last sample of the PIR sensor in the packet.
- Data - 32 samples of the PIR sensor.

Data was collected for a span of 2 months, under normal working conditions.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

TODO CONCLUSIONS

5.2 Future Work

TODO FUTURE WORK

Bibliography

[1] Burcu Akinci, Mario Berges, and Alejandro Gomez Rivera. *Exploratory Study Towards Streamlining the Identification of Sensor Locations Within a Facility*, chapter 226, pages 1820–1827. doi: 10.1061/9780784413616.226. URL http://ascelibrary.org/doi/abs/10.1061/9780784413616.226.

[2] Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. An improved algorithm for matching large graphs.

[3] Carl Ellis, James Scott, Ionut Constandache, and Mike Hazas. Creating a room connectivity graph of a building from per-room sensor units. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 177–183. ACM, 2012.

[4] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

[5] Dezhi Hong, Jorge Ortiz, Kamin Whitehouse, and David Culler. Towards automatic spatial verification of sensor placement in buildings. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, pages 13:1–13:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2431-1. doi: 10.1145/2528282.2528302. URL http://doi.acm.org/10.1145/2528282.2528302.

[6] Jiakang Lu, Yamina Taskin Shams, and Kamin Whitehouse. Smart blueprints: How simple sensors can collaboratively map out their own locations in the home. *ACM Trans. Sen. Netw.*, 11(1):19:1–19:23, August 2014. ISSN 1550-4859. doi: 10.1145/2629441. URL http://doi.acm.org/10.1145/2629441.

[7] Sebastian Müller, Axel Helmer, Enno-Edzard Steen, Melina Frenken, and Andreas Hein. Automated clustering of home sensor networks to functional re-gions for the deduction of presence information for medical applica-tions. *Wohnen–Pflege–Teilhabe–Besser leben durch Technik* , 2014.

[8] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957. ISSN 1538-7305. doi: 10.1002/j.1538-7305.1957.tb01515.x. URL http://dx.doi.org/10.1002/j.1538-7305.1957.tb01515.x.