WILLEM: a Wireless InteLLigent Evacuation Method

W.H. van Willigen R.M. Neef A. van Lieburg
TNO Defence, Security and Safety, The Hague, The Netherlands
willem@few.vu.nl, {martijn.neef, anthonie.vanlieburg}@tno.nl

Martijn C. Schut
VU University Amsterdam, The Netherlands
mc.schut@few.vu.nl

Abstract

In this paper we present WILLEM*, a system for dynamic evacuation routing in buildings, using a wireless sensor network. Dynamic evacuation routing is the process of dynamically determining the fastest routes to the exits. The routes may be changed in case a fire occurs somewhere. We also present an algorithm for detecting congestions in corridors during evacuation, and a means of providing the people in those congestions an alternative route towards the exit. Each phase of the method is descibed extensively: the deployment of the wireless sensor network, the automatic topology learning of the network and the actual evacuation routing methods. We have built a simulation framework in which all types of evacuation routing can be simulated. The results of our experiments were surprising in the sense that dynamic evacuation routing turned out not to be faster than static evacuation routing in every setup; however, we did find out why this is the case. We also performed some experiments on a real wireless sensor network, in order to find out if our automatic configuration method could work in real life. The results are promising. We also present an algorithm for mapping the learned topology of the wireless sensor network upon a virtual map. This way, the network topology can be visualised – which is an important feature for emergency services.*

1 Introduction

Quick evacuation is a critical aspect of large-scale incidents. The main goal of evacuation is to guide people away from an affected region into safe surroundings, and to minimise the time and risk it takes to get there. Despite the fact that most buildings have predefined evacuation plans and procedures, the evacuation itself is very much a dynamic process. Determining of the optimal evacuation route is a complicated matter due to the amount of dynamic parameters involved. The best route for anyone involved greatly depends on the specifics of the incident, the actual state of the building, the condition and location of the evacuees and so forth. For instance, standard evacuation routes may be blocked by smoke or fires, and corridors may be too narrow for the number of evacuees. Present day evacuation plans usually consist of light signs (arrows) that point to the nearest exit. A major drawback of these plans is that they are *static*: they are predefined, and the routes to the exits do not change for whatever reason. These static evacuation plans potentially lead to very dangerous situations, often involving or leading to panic of the evacuees [5, 4].

This paper presents two methods for *dynamic evacuation routing* using a wireless sensor network: one method that will lead everyone to the nearest exits (and dynamically adjusts exit signs according to the current situation); and a method that builds further on this method by also detecting congestions in corridors and is able to provide alternative routes to an exit.

The main goals of this research are: 1) to describe a method for creating a dynamic evacuation routing system in buildings using wireless sensor networks; 2) to compare this method to static evacuation routing; and 3) to develop and test an extension to the dynamics evacuation algorithm, taking into account detection of corridor congestions.

We incorporate these methods in an evacuation system called WILLEM: a Wireless InteLLigent Evacuation Method. This system consists of wireless sensor nodes distributed throughout a building and *tags* that are worn by the people in the building. The nodes are able to observe and react to nearby tags. The unique features of WILLEM are that 1) the nodes autonomously learn the network topology of the wireless sensor network by means of monitoring the tags that are passing by, 2) it is able to perform dynamic evacuation routing (with and without congestion detection during evacuation), and 3) visualisation of the network on a

map of the building (for example, on a handheld device) to be used by the emergency services during an incident. We later describe all the steps that are necessary to succesfully install and use such a system, from the deployment phase to the actual evacuation phase.

This paper is structured as follows. In Section 2, we describe related work and outline the relation of it to the work described in this paper. In Sections 3–6, we present the WILLEM system in different phases: the installation phase, the used methods for evacuation routing, and the visualisation of the network to be used by emergency services. Finally, Section 7 concludes and presents some pointers for future research.

2 Background

Wireless Sensor Networks (WSN) – The literature on WSN development shows that the most important design choices to be made concerns energy efficiency and the choice of an information routing algorithm [2].

There are two types of devices in our wireless sensor network: the nodes (that are placed in the corridors of the building) and the tags (that each person[1] carries). In our wireless sensor network, we do not have much of the energy efficiency constraints as other wireless sensor networks. The reason for this is that our network is deployed inside a building, and each sensor is clearly visible and reachable. This means that batteries can be changed easily when necessary. Also, our sensor nodes only have to be active during the learning phase of the network and during evacuation. The tags have no real energy efficiency issues, since they contain rechargeable batteries that can be reloaded when needed. A person's tag could be placed in the charger for example when he or she leaves the building. The size and robustness of nodes is fortunately also not a big issue in our system. They do not need to be extremely small and they stay put in the same position after careful deployment.The size and robustness of the tags is a somewhat more important issue: they need to be carried by the people in the building. A belt tag is possible, but something smaller like a RFID tag in a card is ideal, since users will find this less of a burden. Robustness is important; the tags are carried by people and are inherently prone to fall.

Our routing algorithm is rather straightforward. We do not need complex aggregation methods or other energy-saving protocols; the information to be iterated over the network is the gradient value that the exits send out periodically in case of an emergency. The nodes then relay this signal to its other neighbours upon receipt. They also need to be able pick up signals from near tags and reply to them. The tags have a quite simple communication scheme: they need to be able to send out signals to the nearest node and process the reply from the nodes.

Evacuation Routing – Concerning evacuation routing, we briefly look at other simulation studies for this and specifically at earlier use of WSN technology to facilitate such routing.

With respect to the movement of people, existing research contains more complex behavioural models than ours. Our behavioural model is inspired by panic models by Helbing [6, 5] and Schreckenberg [7], but it is a simplification. We have chosen for a simplification since we considered it sufficient to show different outcomes of our dynamic evacuation routing methods. Concerning evacuation simulation, there has been much research using virtual reality technology for evacuation simulation, such as Exodus [1]. We did not consider this necessary for our research. In our research, it is more important to show that the method itself works, rather than to show that the method works using an elaborate 3D environment. Also, we do not address human factors aspects of evacuation processes, such as the physical appearance of the evacuation signs or the user experience of dynamic routing advice.

In Barnes et al [3], a method is proposed for dynamic evacuation routing using wireless sensor networks. There is an important difference between that method and our proposed method. The big difference lies in the core architecture of the wireless sensor network. In the method by Barnes et al, a lot of information about the building needs to be preprogrammed into the system. This is not necessary in our system, since our system has automatic configuration.

Our approach to evacuation routing is inspired by the work by Lieburg et al [8], who also used a simple gradient method to solve the routing problem.

3 The WILLEM Method

WILLEM is a multi-component system for evacuation routing. The system is designed to be an out-of-the-box, easy to deploy, fully distributed, robust evacuation routing system. This system consists of two different phases – *installation*, which concerns the installation of the sensors, the initial deployment (including self-organising configuration of the communication network), and the working of the sensors while there is no emergency situation; and *evacuation*, which includes the dynamic routing of evacuees by means of sensor signals (arrows indicating the best direction) and information provision to the emergency services (mapping out the network and current situation on a mobile device). In the following Sections, we explain each phase in detail. Each explanation contains a description of what happens in that particular phase of WILLEM and an experimental evaluation of that phase.

[1] Or a proportion of the people in the building

4 Phase A: Installation

The first step of the system is the deployment and self-organising configuration of the wireless sensors. Smoke sensors that also contain a means of pointing to possible escape directions are deployed inside a building. When deploying the sensors, the following three rules should be obeyed. Firstly, each corner of the building (by corner we mean a location where three or more corridors come together) should contain a sensor. Secondly, each exit of the building should have a sensor above it. These sensors at the exit also have to know that they are above an exit (this could be accomplished for example by a hardware switch).

Finally, the nodes should not be too far away from their direct neighbours. In order to comply with this rule, the hardware specifications should be checked. However, exit routing signs are usually located close to each other as well.

The configuration of the system is done automatically, based on the movement of persons in the building. Each person in the building should be equipped with a tag. This tag could be a RFID tag in an entrance card or a belt tag. The tag is able to communicate with the wireless sensor nodes. Using this communication, the nodes are able to determine which sensor nodes are their direct neighbours. The nodes do not only learn which neighbours they have, but also measure the distance to this neighbour. This distance measure is also used in the evacuation routing algorithm.

The signal that the tags send out contains two variables: The ID of a node x that the tag has seen last (if the tag has seen a node before), and a time variable t that denotes how long ago it was that the tag a saw that node x. When a tag a is directly underneath a node y, that node y will pick up the signal and send an acknowledgement back to the tag a. The tag a resets the time variable t and replaces the last seen node ID by the ID of the node y. When a node y receives a signal, it updates its neighbour info. If the node ID x contained in the signal is not yet in the neighbour-list of y, it is added. Also, the time variable t is stored. The variable t is stored as a distance measure between x and y.

During the learning phase, both the tags and nodes play important roles. A tag increases a counter each timestep representing the distance from the last seen node and sends a signal at fixed intervals. If a tag receives an acknowledgement from a node, then it sets the last-seen-node to this node and initialises the distance from it to 0. A node stores a list of its neighbours including the mean distances between itself and these nodes, and the number of observations based on which this mean distances was calculated. A node then acts if it receives a signal, and consequently updates all these three values. Details including pseudocode algorithms can be found in [9].

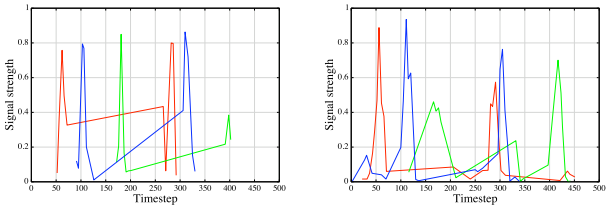

Figure 1. Results of topology learning for nodes 128 (left) and 160 (right). Each colored line represents one of the three tags. The y-axis shows the received signal-strength by the respective node.

4.1 Experiments

We have deployed a very simple wireless sensor network, consisting of nodes and tags, in an actual building. The nodes in the network are able to detect nearby tags, and log the results. We had three tags in this experiment. Each of these tags was carried by a person, who took a certain route through the building. The tests involved a network with 27 nodes in two parallel (horizontal) corridors connected by three more (vertical) corridors.

Figure 1 shows two examplary result charts, each for one single node. Each point in a chart is a received signal by the corresponding node. From all the node charts taken together (27 in total), we should be able to see if a topology could be derived. The current status is that some charts provide very promising results: if we follow one single tag along a couple of nodes, we see some promising results. However, at some points, we also have false positives (a node observing a tag that is not actually there)[2] and false negatives (a node missing a tag that is actually there)[3].

In general, we consider the overall results as promising, and by means of slight modifications of the real-world settings (e.g., better wireless sensor technology, post-processing of the results) it is feasable to effectively implement the above presented topology learning method.

4.2 Summary

In summary we can say that we obtain perfect results of the topology learning method in simulation, given that the simulation runs sufficiently long for each sensor to be reached by someone coming from each direct neighbour. In reality however, the sensors need to deal with interference, signal failure and other problems. The major problem is to

[2] solutions to this problem include preprocessing of the data

[3] adding more nodes could solve this problem

detect if a tag is located near a node - we have to finetune the signal strength of the used tags.

5 Phase B.1: Evacuation - Routing

After the network has been installed (and the topology of it has been learned as described above), it can be used for evacuation routing in emergency situations. In this Section, we describe two different methods of *dynamic* evacuation routing, with the main difference between them being that one includes *congestion detection*, i.e., re-routes if there are too many people in a corridor, while the other does not. In addition, a benchmark *static* evacuation routing method was implemented, in which all nodes use a simple gradient descent method to determine the nearest exit. The difference between static and dynamic routing, is that static routing does not allow for re-routing (in case of a hazard or congestion), while dynamic does.

The ability to do evacuation routing is the core aspect of the system: in case of a calamity, each node points to the nearest exit. The proposed dynamic methods periodically update the network status. This means that if during evacuation a fire breaks out at another location, the routes to the exits are updated accordingly.

The simplest (benchmark) static method, called method (0), works using a method of *gradient descent*. The exits in the network iterate a signal over the network. Each node relays this signal to its neighbours, and add some strength to the signal. The further this signal is from the exit it was originally sent from, the stronger it gets. Each node then points to the weakest signal that it received. This way, each node automatically points to the nearest exit. Since this is the static method, this only needs to be done once, and the routes will never change afterwards.

The two dynamic methods build further on this benchmark method: method (I) periodically updates the network status (instead of only once, as in the benchmark); method (II) extends this further by means of updating the network status based on information obtained from *congestion detection*.

After the topology and the directions have been determined in the static method, the nodes do not change signs anymore (as in a 'traditional' evactuation system). This means that in an emergency situation, people are possible directed towards hazardous locations. To prevent this, the dynamic routing method (I) lets nodes periodically execute the gradient descent method – also during an emergency situation. A consequence of this is that nodes do not direct evacuees to hazardous locations anymore. This works as follows. When a node detects a hazardous situation, e.g., smoke, it stops relaying the received signals from neighbouring nodes. This means that its neighbours no longer receive signals from the source of the calamity: the neighbours will not point to that node, since they will only point to neighbours of which they have received a signal. (A small change that needs to be made to the gradient descent algorithm is that the nodes should not wait anymore for every neighbour to send a signal; instead a node processes each incoming signal separately.)

The most advanced method (II) is also able to check, during evacuation, if certain routes become very busy. If this is the case, our system is able to provide an alternative route to the people in or behind the congestion. To accomplish this, nodes now (additionally) store the x most recent observations for each neighbour, where an observation consists of the time that it took for a person to come here from a certain neighbour. The mean value of these observations is stored and updated with each new observation. During the configuration phase of the network, each node has learned a distance value for each neighbour. The current distance value is compared to the learned value, and as soon as the mean value from the last observations gets above a certain threshold (e.g. five times larger than the learned value), the node concludes that there is a congestion between that neighbour and itself. As soon as a node detects a congestion, it will not only show the shortest way to the exit, but also the second shortest way to the exit. Our model uses a second gradient to activate this behaviour. This second gradient behaves almost the same as the existing gradient. The exits send out this gradient and the receiving nodes relay this signal. The difference is that the nodes will not only block the signal when the alarm is set off, but also when a congestion is detected at that node. The arrow of the second gradient is only shown when it points to a different neighbour than the frst gradient.

5.1 Experiments

We have built a simulation environment in which we can compare the different methods of evacuation routing. We created a virtual building with three exits and many corridors with variable capacities to run our experiments in. We let a fire start at one exits in the building, after which the evacuation begins. We then measure how much time it takes to completely clear the building.

We conducted experiments with 400, 800, 1200 and 1600 persons, who were randomly distributed throughout the building. We tested three different initial configurations of start locations of the persons (thus not all experiments had independent random starting point distributions). This resulted in an experimental design with 3 (number of methods) $\times$ 3 (number of start-location distributions) $\times$ 4 (number of persons) = 36 different settings; 5 independent runs were conducted per setting. For method (II), two additional independent variables were introduced: *observation-buffer* $\in \{10, 30, 60, 90\}$ (indicating the number of observations

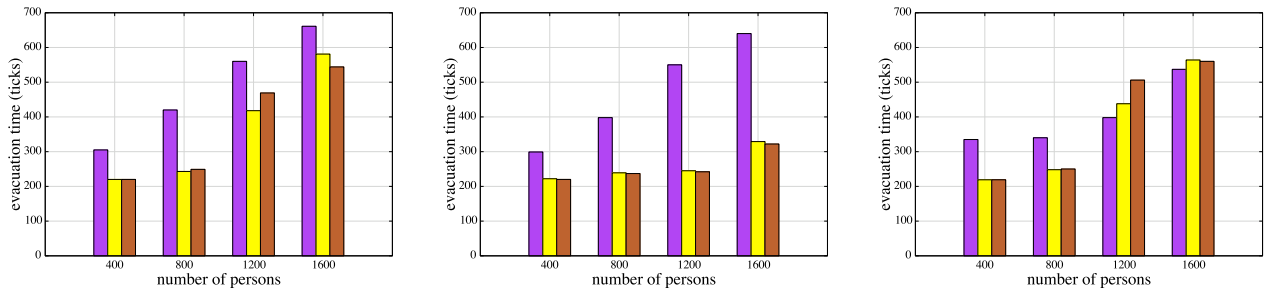

Figure 2. The results of the three distribution sets of persons. The purple bar represents static routing, the yellow bar represents dynamic routing without congestion detection, and the brown bar represents dynamic routing with congestion detection.

that is stored during evacuation), and *threshold* $\in \{3, 6, 9\}$ (states that if the mean of the observations is *treshold* times as large as the learned distance value, the state of the node is switched to 'congested').

The results are in Figure 2. We see for configurations (a) and (b) the evacuation takes the longest with method (0), and the performance of method (II) is slightly better than method (I) when the number of persons is very high. Apart from that, there is not much difference between the two dynamic routing methods. However, for configuration (c), we see that method (0) outperform methods (I) and (II) when there are many people. The reason that it outperforms method (I) is that in (I), the people are directly lead to the best exit, creating a congestion near that exit. Method (0) leads people towards a wrong exit, which means that these people arrive at the correct exit a bit later, so no congestion occurs over there. The static method outperforms method (II) because of another unfortunate chain of events in this configuration: by closer analysis of the runs, we found out that congestions emerge as a *result* of congestion detection. Thus our solution (detection) to a problem (congestion) works exactly opposite in this case: instead of solving, it creates the original problem; hence the worse results.

5.2 Summary

In experimenting with different evacuation methods, we discovered, most importantly, that the distribution of persons in the building, has a major influence of the performance results. We intend to research this finding further in order to better compare the different evacuation methods.

6 Phase B.2: Evacuation - Information

The final phase of the system concerns the visualization of the learned network topology upon a virtual map, which can be used by rescue workers to show the location of the calamity. Our current implementation is rather simple, but user friendly: a user loads a picture of a building into the system; upon which the user draws lines on the places where the corridors are; when this is done, the user can load the neighbour information from the nodes into the system; then, a visualisation of the nodes in the building can be shown. When more solutions are possible (in the case that the building is symmetrical), all options can be seen. The user then can then manually choose the correct mapping.

The core of this method is thus mapping two graphs to each other (the graph of the nodes and the graph of the corridors). The two graphs (the corridor graph and the network graph) are almost the same, with the exception that there can exist more network nodes between two corridor nodes. We solve this mapping problem using a simple depth-first search algorithm with some local knowledge. The method starts by finding the network node(s) (from the learned topology) and the corridor node(s) (from the drawn corridor network) with the highest arity. By definition, the number of network nodes with the highest arity (the node(s) with the most neighbours) is equal to the number of corridor nodes with the highest arity (the position in the building where the most corridors come together). The reason for this is that at each corner of the building, a node must be located. One of these network nodes is then mapped with one of the corridor nodes. Then, the method will find the neighbouring corners of the just mapped nodes. Then, using recursion, the process is started again, using the partially finished mapping. A not yet mapped corner neighbour of an already mapped node is searched for and mapped to a corresponding corridor corner. Additionally, when a neighbouring corner is mapped, the network nodes that lie between these nodes are placed in that corridor. When a solution turns out to be incorrect, the algorithm goes back to an earlier stage and tries again with another mapping using backtracking, until

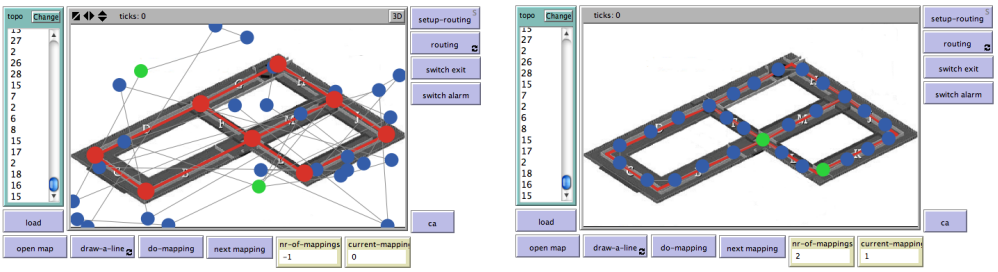

Figure 3. Visualisation of the learned network topology: the topology is first loaded into the system (left) and is then mapped on the drawn corridor graph (right).

all solutions are found. This algorithm is guaranteed to find all possible solutions (as mentioned: multiple solutions can occur in case the building is symmetrical in some way).

In Figure 3, we show the last two steps of this visualisation process (assuming that a map of the building has been loaded and corridors are drawn). The manually drawn corridor graph is shown in red; the network graph contains green (exit) nodes and blue (normal) nodes. The figure on the right in Figure 3 shows the end result: both graphs are mapped on each other.

This mapping method guarantees to find all possible solutions, but it currently is not very fast. Visualising buildings with more than one floor is (visually and computationally) difficult as well. Although much work must still be carried out to improve this method, our formulation of the method can be a good starting point for further research.

7 Conclusions

To conclude, we return to the objectives introduced in Section 1. Regarding the first objective, we have delivered WILLEM: a complete, out-of-the-box method for evacuation routing using wireless sensor network technology. Although practical implementation may be a matter of further research, we believe to have shown that WILLEM is a solid basis for a real-world evacuation system. Concerning the second objective, we have shown by means of simulation that our dynamic routing algorithm outperforms static routing. Finally, with respect to the third objective, we have extended our algorithm such that it can detect corridor congestions and re-route appropriately. We showed that this is extension is simple and elegant modification to the first dynamic algorithm. The experimental tests that we ran with this advanced algorithm showed some surprising results. We observed that in some situations (defined on the starting locations of the people), through an unfortunate chain of events, the algorithm actually performed worse than the static algorithm. The reason for this is that reacting to congestion detection and consequent re-routing can actually lead to more congestions (elsewhere). This is the same effect observed for traffic jams, where cars choose alternative routes because of a traffic jam ahead, leading to a (worse) traffic jam on the alternative route. Because we have not much further data available about how often this happens, this issue is our first step on future work regarding dynamic evacuation routing. Further future work includes a physical implementation and real-life tests.

Acknowledgements The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

References

[1] Exodus, http://fseg.gre.ac.uk/exodus/index.html.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks (Amsterdam, Netherlands: 1999)*, 38(4):393–422, 2002.

[3] M. Barnes, H. Leather, and D. K. Arvind. Emergency evacuation using wireless sensor networks. In *LCN '07: Proceedings of the 32nd IEEE Conference on Local Computer Networks*, pages 851–857, Washington, DC, USA, 2007. IEEE Computer Society.

[4] H. W. Fischer, G. F. Stine, B. L. Stoker, M. L. Trowbridge, and E. M. Drain. Evacuation behaviour: Why do some evacuate, while others do not? *Disaster Prevention and Management*, 4(4):30–36, 1995.

[5] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000 Sep 28.

[6] D. Helbing and B. A. Huberman. Coherent moving states in highway traffic. *Nature*, 396:738–740, 1998.

[7] M. Schreckenberg, A. Schadschneider, K. Nagel, and N. Ito. Discrete stochastic models for traffic flow. *Physical Review E*, 51:2939, 1995.

[8] A. van Lieburg and M. Neef. Bio-inspired evacuation routing support. In *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, 5-6 October 2006, Namur, Belgium*, pages 205–211, October 2006.

[9] W. H. van Willigen. WILLEM: a wireless intelligent evacuation method. Master's thesis, VU University Amsterdam, 2008.