AnyBody: a Self-organization Protocol for Body Area Networks

Thomas Watteyne*, Isabelle Augé-Blum
ARES INRIA / CITI F-69621
INSA Lyon, France
firstname.lastname@insa-lyon.fr

Mischa Dohler, Dominique Barthel
France Telecom R&D
Grenoble, France
firstname.lastname@orange-ftgroup.com

ABSTRACT

Self-organization for wireless multi-hop systems can be divided into two categories: proactive cluster-based solutions and reactive on-demand solutions. Whereas the former have been studied for ad-hoc networks, the latter seem more adapted to low-energy low-traffic wireless sensor networks. We show that, despite the relative high cost to build and maintain a topology, a cluster-based approach is particularly suited for Body Area Networks. We present AnyBody, a self-organization protocol in which sensors attached to a person are grouped into clusters.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Wireless Communication

1. INTRODUCTION

Wireless Sensor Networks (WSNs) [1] are composed of nodes capable of doing three complementary tasks: measuring a physical value, processing that value and communicating over a wireless channel. Whereas in this paper we look at WSNs from a computer network point of view, this field of Research is highly interdisciplinary, ranging from Microelectronics to Biology and Sociology. WSNs are foreseen to have very diverse applications, such as forest fire detection, smart homes or Health monitoring. This diversity directly reflects in the solutions which are proposed, and which differ radically depending of the final application.

Body Area Networks (BANs) can be seen as a subclass of WSNs, with specific constraints. Note that in this paper, we will consider that the nodes of the BAN communicate wirelessly. A wireless sensor is constrained in terms of available memory, bandwidth, computational power and embedded energy. Depending on the application of BANs, additional constraints may be added.

*Thomas Watteyne is primarily affiliated to France Telecom R&D.

BODYNETS 2007, June 11-13, Florence, Italy

DOI 10.4108/bodynets.2007.186

As we assume no fixed infrastructure is available (such as a base station), some sort of functional structure needs to be brought into the network. This can be done by adding a self-organizing function to the communication protocols running on our BAN[13]. Once established, this structure can facilitate finding a multi-hop path between source and destination (*i.e.* routing), reducing the load of the network by performing data aggregation, and reducing the energy consumption by turning off redundant nodes.

We consider a group of patients staying at a hospital. On a given floor of the building, sensors are deployed. These can be medical sensors (coupled with a medical sensing device), and/or sensors gathering information about the patients' environment (temperature, noise level, ...). All sensors have the same communication facilities, and periodically report some information. This information needs to flow in a multi-hop mode toward a special gathering node which we call sink node. The sink node acts as a gateway node between the hospital's (wired) computer network, and the BAN. Once the wireless network is running, the medical crew is alerted when a patient needs urgent medical assistant, and can monitor the environment of the patient. The purpose of our self-organization protocol is to create a functional structure inside the sensor network to enable this information flow.

This paper is organized as follows. In Section 2, we present related work. This includes flat routing, and self-organization and hierarchical routing protocols. In Section 3, we describe AnyBody. Five steps are required to set up a functional structure inside the BAN, and to set up the routing paths. A realistic example is given to describe the protocol. In Section 4, we present results obtained by extensive simulation. The characteristics of the built structure is presented, as well as the cost of building it. This paper is concluded, and future work is presented in Section 5.

2. RELATED WORK

It is not clear whether it is better to adopt a flat or hierarchical structure, especially for routing. Flat routing protocols run on a network of undifferentiated sensors. Examples are AODV[11] and DSR[6], where the sink node floods the whole network with a request. The nodes which have the requested information use the path the request came from for returning the answer to the sink. Geographic routing protocols also run on a flat topology. Assuming each node knows its position and the sink's, the information is sent to the neighbor node closest to the sink node. This simple protocol has

been improved to guarantee delivery of the message[4, 7].

Whereas guaranteed delivery can be achieved, the assumption that nodes are position-aware is rather strong. Having each node equipped with a GPS module can turn to be too expensive. What's more, additional functionalities such as data aggregation and node scheduling (*i.e.* turning off redundant nodes to spare their batteries) is not easy on a flat topology. Self-organization protocols are therefore studied. Nodes are grouped together to form clusters, and each one of these clusters is lead by a clusterhead[5, 17, 2]. Hierarchical routing protocols can then run on this structure in order to setup paths toward the sink node. It is important to differentiate clustering and routing protocols running on these clusters.

LEACH[5] is perhaps the simplest clustering protocol. Depending on a pre-defined probability, nodes elect themselves as clusterhead. Other nodes then join the closest clusterhead. As we will see in Section 4, the number of clusters grows linearly with the number of nodes, which may not be desirable. What's more, as clusterheads are placed randomly, some non-clusterhead nodes may have no clusterhead at communication range. As a result, they are disconnected from the network although physically there exists a multihop path to the sink node.

Clustering algorithms have been proposed to alleviate the aforementioned problems. Mitton et al.[9] defines the *density* of a node as the ratio between the number of links within the node's 2-hop neighbors and the number of 2-hop neighbors. Nodes who have the local highest density are elected clusterhead. The authors show that the obtained cluster-based structure can be used for example for efficient broadcasting[8]. This assumes 2-hop neighborhood knowledge, which implies sending several hello packets per node.

The previous two works focus on clustering only. In [15], Tholeyre et al. present a complete self-organization protocol, coupling a routing and a clustering protocol. The resulting protocol starts by electing nodes as dominators, which form a Connected Dominated Set. This means that all nodes are either dominators (thus part of the Connected Dominating Set), or neighbors of a dominator. Only after this step, some dominators are elected clusterheads, and other nodes join them to form clusters. The main difference with AnyBody (to be explained in detail in Section 3) is that our protocol starts by building the clusters before interconnecting them, which is done the other way in Tholeyre et al.'s proposition. We argue that using our approach, we can really understand how the clusters are built. We have chosen to use Mitton et al.'s clustering algorithm in AnyBody. This can easily be interchanged to meet specific needs. This flexibility is harder to achieve when using Tholeyre et al.'s proposition.

3. THE ANYBODY PROTOCOL

3.1 Overview

AnyBody runs in five steps, which will be detailed in the next subsection. First, a node discovers which other nodes it can directly communicate with – its 1-hop neighbors – by exchanging hello messages (step 1). Then, based on this information, nodes are grouped into clusters, with one clusterhead for each cluster (step 2 and 3). Clusters are then interconnected (step 4), and routing paths are set up toward the sink node (step 5).

3.2 Setting up the cluster-based structure

Please refer to Fig. 1, which illustrates all five steps. We assume all nodes start functioning at the same time. When a tie occurs, it is broken randomly (*e.g.* in step 3, if a node has two neighbor with the same density, it sends to either one of them).

1. **Neighbor discovery.** Each node sends out a $\texttt{hello}_1$ message in which it puts its unique identifier. During a given time frame, it waits for $\texttt{hello}_1$ messages from its neighbors. This time frame is used to reduce the number of collisions between $\texttt{hello}_1$ message. Its duration must be optimized depending on the MAC protocol. After this step, all nodes know what neighbors they have. In Fig. 1, we draw an edge between neighbor nodes, leading to the *connectivity graph*. Furthermore, nodes relay the $\texttt{hello}_1$ message they have received from their direct neighbors by sending a $\texttt{hello}_2$ during a second time frame. Upon hearing those $\texttt{hello}_2$ messages, each node has a complete knowledge of its 2-hop neighborhood (*what are my 1-hop and 2-hop neighbors, and which of those nodes can communicate together ?*)

2. **Density calculation.** Based on the node's knowledge of its 2-hop neighborhood, it calculates its density as defined in [9]. The density is the ratio between the number of links and the number of nodes within the 2-hop neighborhood. Each node then sends out a $\texttt{hello}_3$ message containing its density, and receives the density of its neighbors. This density calculation serves in step 3 to elect a clusterhead and form clusters.

3. **Contacting clusterhead.** Once each node has received these $\texttt{hello}_3$ messages from all its neighbors, it sends its list of 1-hop neighbors to its neighbor of highest density with a `join` message. No message is sent if the node has a higher density than all its neighbors. The `join` messages are relayed by the receiving nodes until it reaches a node with highest local density. Note that this process results is electing local nodes with highest density as clusterheads, and not the node with network wide highest density.

   After step 3, nodes are grouped into clusters. Clusterheads are the nodes with highest density inside the cluster. They know which nodes are attached to them, including each one of its members' neighbors list. We call *intra-cluster gradient* the paths followed by the `join` messages. Each node which has relayed a message needs to remember to which node it has sent it. The intra-cluster gradient will be used for all intra-cluster communication. It is represented in Fig. 1 by the green arrows.

4. **Setting up the backbone.** So far, we have set up independent clusters; now they need to be interconnected. Each clusterhead identifies the gateway nodes (GW) of its cluster. To do so, it goes through its list of cluster members, and pinpoints those who have a neighbor not inside the cluster. It uses the intra-cluster

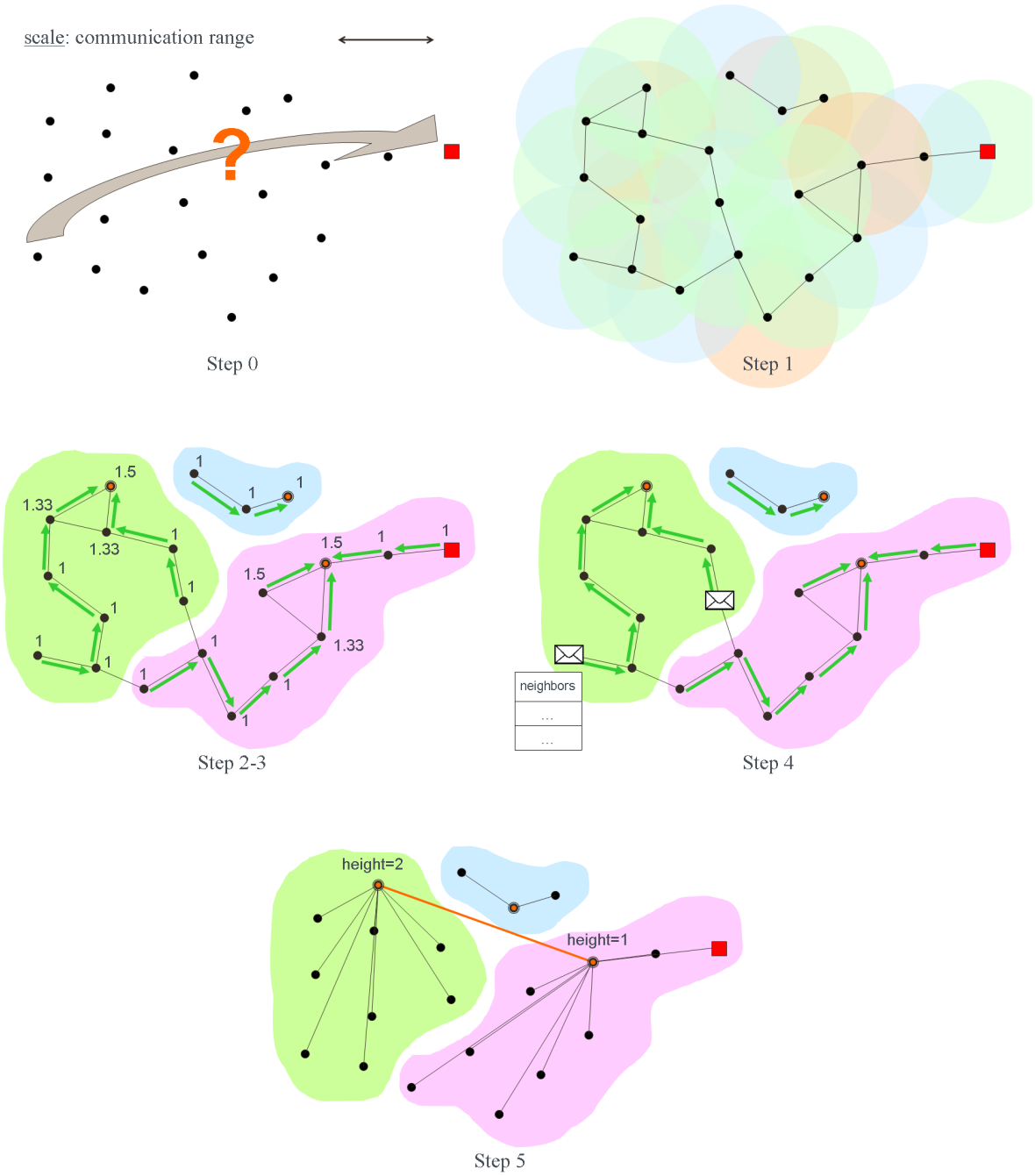

Figure 1: The five steps of the AnyBody self-organizing protocol. The black dots represents the nodes. The sink node is represented by a red square. Communication ranges are represented by colored circles centered on the nodes (step 1). An edge interconnects nodes which are able to communicate (step 2). The intra-cluster gradient paths are represented by green arrows (step 2-3). The density is written next to each node in step 2-3. The envelopes represent the `join` messages containing the nodes' neighbors. In step 5, the edges represent the virtual links, black for intra-cluster links (the intra-cluster gradients), and orange for the inter-cluster links (inter-cluster gradients).

gradient paths to send them a `gw_inform` message, informing them they have been elected *GW* nodes. There are always two gateway nodes facing each other, each one of a different cluster. The job of a *GW* node is the following: each time it receives a message from the gateway node of the other cluster, it sends it to its clusterhead using the intra-cluster gradient paths. If it receives a message from its clusterhead, it sends it to gateway node it is connected to.

At the end of this step, clusters are formed, and clusterheads are virtually connected to each other through GW nodes. This virtual connection is represented in Fig. 1, step 5, by the orange edge. All these virtual connection between clusterheads will form what we call a *virtual backbone* (a term also used in [14]). All inter-cluster communication will be done on this virtual backbone.

5. **Setting up the routing paths.** Now that we have interconnected clusters, we can set up the routing paths. As we have a many-to-one communication scheme (one being the sink node), we set up a gradient. The sink node starts by sending out a `gradient_setup(1)` message to its clusterhead. This one then increases the counter and sends a `gradient_setup(2)` message on all the backbone links it is connected to. All clusterheads receiving this message increase the counter, and resend a `gradient_setup(3)` message on the backbone links they're connected to. This process goes on until all clusterhead have sent one `gradient_setup` message. All clusterheads set their *height* as the smallest counter contained in a `gradient_setup` message they have received. They also remember who they received this message from, and call this other clusterhead *next hop*.

   At the end of this step, we have formed clusters, have interconnected them, and have set routing path from each cluster to the sink node. We call *inter-cluster gradient* this path. They follow the clusterheads with decreasing heights. Note that all links of the inter-cluster gradient are part of the virtual backbone.

3.3 Using the structure for routing

We have described how AnyBody sets up a functional structure. Let's see how this structure is used for routing on the practical example depicted in Fig. 2. In this figure, we have three clusters called A, B and C. For practical reasons, we have labeled each node with the cluster it belongs to, and a unique number inside that cluster (order is of no importance).

After step 1, the connectivity graph is revealed, and it turns out nodes $C.1$, $C.2$ and $C.3$ are disconnected from the sink. They will never be able to send any message. After step 3 of the AnyBody protocol, nodes $A.9$, $B.7$ and $C.3$ have been elected clusterhead of clusters A, B and C, respectively. Moreover, the intra-cluster gradient path (depicted with blue dotted arrows on Fig. 2) are identified. In step 4, nodes $A.2$, $B.1$, $A.8$ and $B.2$ are informed they are gateway nodes. In step 5, the sink sets up the inter-cluster gradient path depicted as plain orange arrows. Note that these paths converge to the clusterhead of the cluster to which the sink node belongs.

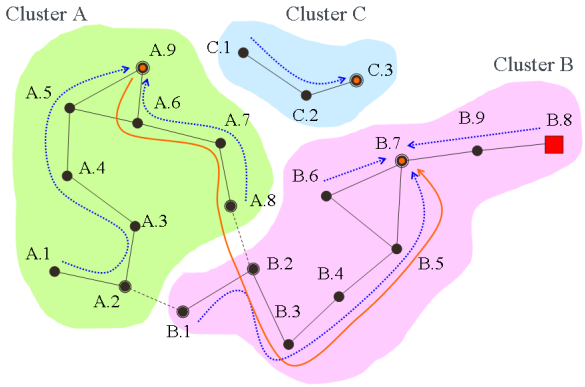

Figure 2: Routing in a cluster-based hierarchical structure. The dotted blue and plain orange arrows represent the intra- and inter-gradient paths, respectively.

Let's assume node $A.1$ wants to send a message to the sink. It starts by sending its message to its clusterhead $A.9$, following the intra-cluster gradient path $A.1$-$A.2$-$A.3$-$A.4$-$A.5$-$A.9$. The clusterhead receives the message, and sends it along the inter-cluster gradient path $A.9$-$A.6$-$A.7$-$A.8$-$B.2$-$B.3$-$B.4$-$B.5$-$B.7$. The message was sent from cluster A through cluster B by the gateway nodes $A.8$ and $B.2$. The clusterhead $B.7$ finally redirects the message to the sink node, using the reverse intra-cluster gradient path of cluster B, following the path $B.7$-$B.9$-$B.8$.

More generally, using AnyBody, all message will follow the path *sending node - its clusterhead - other clusterheads - sink's clusterhead - sink*.

3.4 Other possible uses of the structure

So far, we have focused the description of AnyBody on routing. Setting up intra- and inter-cluster gradients lets any node send a message to the sink in a localized and distributed manner. Yet, the AnyBody structure offers other possible uses, some of which are (but are not limited to):

- Redundant sensors could be turned off in order to reduce the energy consumption. In Fig. 2), a sensor like $A.1$ could be turned off without affecting the global functioning. Other nodes such as $A.9$ could turn off their sensing device, but should let their communication device remain on, as it is necessary for relaying other nodes' messages.
- Before sending messages onto the inter-cluster gradient paths, clusterheads could buffer it for some time. In case a second message reaches the clusterhead while the first one is still buffered, these message could be combined (either appended one to each other, or averaged out, summed up, depending on the nature of the data). This data aggregation technique would reduce the number of messages sent on the inter-cluster gradient paths.

- Heterogeneous networks are composed of sensors with different possibilities. Some nodes may have more battery, higher available bandwidth, more memory etc. It would be interesting to give those more capable nodes a more demanding role such as being a clusterhead. Those super-nodes could for example increase their density so as to get elected clusterhead.

Whereas the exact implementation of these other possible uses of the AnyBody structure is considered future work, the adaptation is rather simple, and the possibilities endless.

4. PERFORMANCES

We have implemented AnyBody onto the Georgia Tech Sensor Network Simulator (GTSNetS)[10], an event-driven object-oriented simulation tool specifically designed for Wireless Sensor Networks (including Body Area Networks). In this Section, we will detail the models and parameters used, and present simulation results on both the characteristics of the built structure and the cost of building it.

4.1 Modeling the environment

We consider a 1000×1000 area, possibly the floor of the hospital the sensor network is deployed on. In that area, nodes are positioned uniformly (*i.e.* each node chooses a x and y axis coordinate in [0...1000], uniformly). Among these sensors, one is randomly chosen to be the sink.

As we want to measure only the performance of our self-organization protocol, we want to free ourselves from lower layer considerations. We chose to use the IEEE 802.11 MAC protocol[16], which is the default MAC protocol of our simulator. Nevertheless, the chosen MAC layer has little or no impact on the simulation results we gathered because we only log the activity of our self-organizing protocol. For example, when counting the number of exchanged messages, we don't count specific IEEE 802.11 layer 2 messages.

Similarly, we use the Unit Graph Model for propagation, which is a rather simplified model. Each node has a communication radius. Nodes separated by a distance smaller than that radius can communicate. This implies link bidirectionality.

In order to evaluate the impact of growing number of neighbors, simulations were performed for a communication radius of 150 and 250.

4.2 Cluster characteristics

In this subsection, we study the characteristics of the self-organized structure build by AnyBody.

We first focus on the mean number of clusters. For this, we increase the size of the network, by increasing the number of nodes from 100 to 1000. We plot results for AnyBody and LEACH[5] for a range of 150 (Fig. 3) and 250 (Fig. 4). Note that a clusterhead election probability of 5% is used for LEACH.

As LEACH uses a clusterhead election probability equal for all nodes, we obtain an linearly increasing number of clusterheads (*i.e.* clusters) with the number of nodes. This phe-

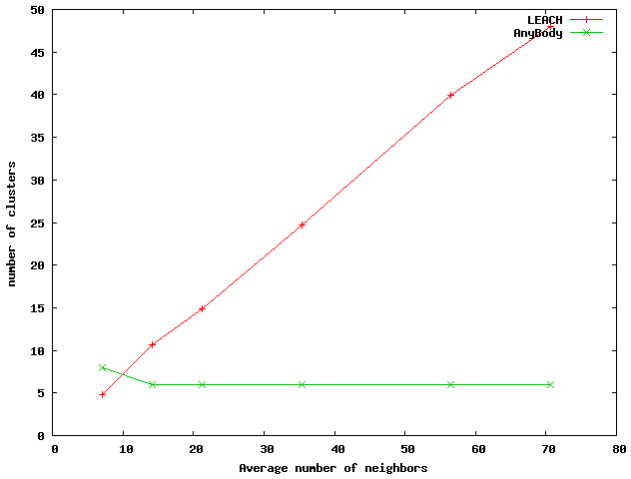

Figure 3: Mean number of clusters with a communication range of 150. A clusterhead election probability of 5% is used for LEACH.

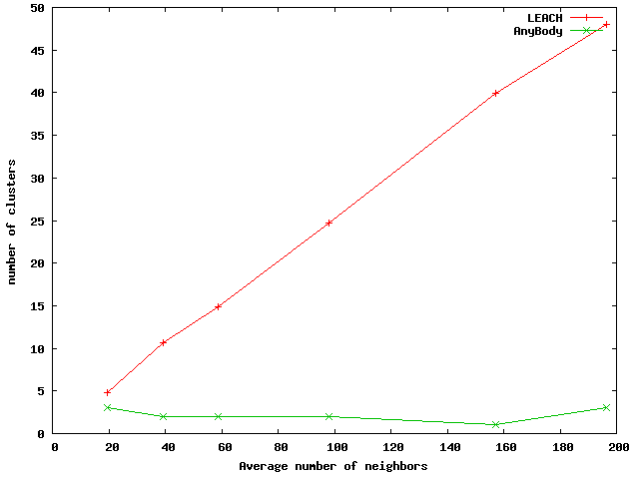

Figure 4: Mean number of clusters with a communication range of 250. A clusterhead election probability of 5% is used for LEACH.

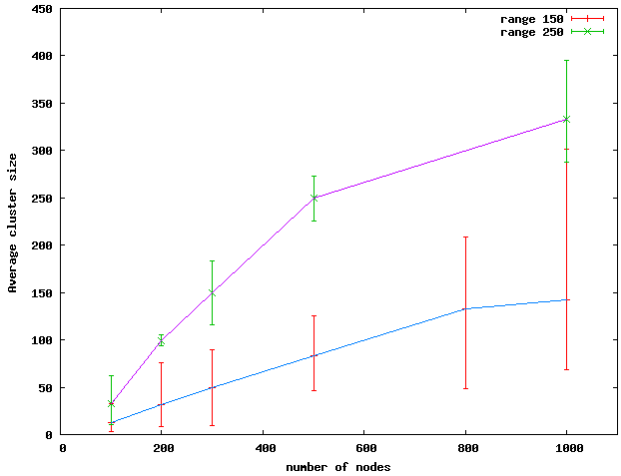

Figure 5: The mean cluster size in number of nodes. It is higher for 250 range because there are less (but bigger) clusters than for the 150 range. The vertical error bars represent the ranges between the min and max values obtained around the –here interconnected– average values.

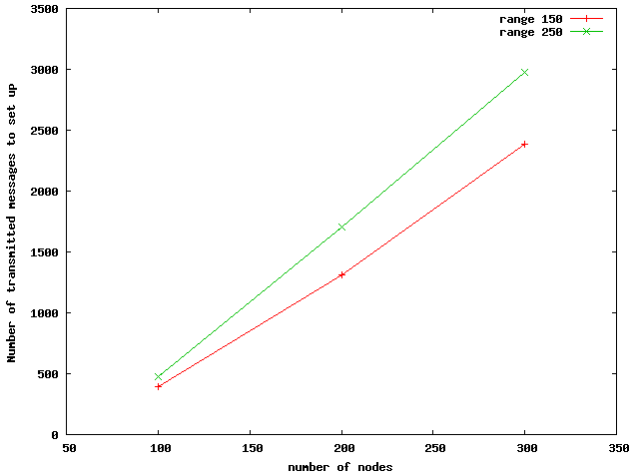

Figure 6: The cost of setting up the AnyBody structure, expressed in number of individual transmissions.

nomenon is independent from the communication range. We argue that having a constant number of clusters is better. Indeed, when nodes are added in the same area, they should join existing clusters. Data aggregation could be more effective (as the number of aggregated flows increases), and scheduling algorithms could be more efficient (with more nodes in a cluster, more are redundant and can be turned off). As can be seen from Fig.-4, AnyBody keeps the number of clusters low, and nearly constant with a growing number of nodes.

By comparing the two figures, it can be seen that AnyBody creates more clusters for a communication range of 150. This is due to the fact that the nodes have a more limited communication area, which translates into clusters having a smaller membership. More clusters are thus created to cover all nodes.

We plot the mean cluster size in Fig. 5. As expected, this cluster size is higher for a range of 250. There are indeed less (but bigger) clusters than with a range of 150.

4.3 Cluster setup cost

In the previous subsection, we have described the characteristics of the structure built by AnyBody. This structure has promising characteristics such as a constant number of clusters when the number of nodes increases. In this subsection, we show how costly it is to set up such a structure.

We represent the energy consumption of the setup phase by the number of sent messages. A more precise model is possible, but necessarily takes into account the underlying layers. As we want our results to be lower layer-independent, we have chosen to count the number of sent messages. Keep in mind that receiving a message can be nearly as costly as sending one[12].

In Fig. 6, we depict the number of sent messages during the setup phase of AnyBody (*i.e.* the five steps described in Section 3). Each node is asked to send three `hello` messages (steps 1–2). It is thus normal to see a near linear increase of the number of sent messages with the number of nodes. In step 3, nodes send `join` messages to their clusterhead. The larger the clusters, the more hops each `join` message will need to travel. Therefore, the number of sent message is higher for a range of 250, as the size of the clusters is larger.

4.4 Run-time cost

Once the structure is built, useful data can be transmitted to the sink node. As sender and receiver may be too far from each other to communicate directly, intermediate node relay that message. Depending on path followed, the number of these relaying nodes can vary. In Fig. 7, we have depicted the average number of individual transmissions (*i.e.* the number of hops) to send a message from source to size. Sending nodes were chosen randomly inside the network. Results are averaged out over 100, 200 and 300 runs for the network of size 100, 200, and 300 respectively.

An important result is that we achieve 100% delivery ratio, which means that each sent message is received. In a flat routing protocol, this is achievable, but with a large hop count[4].

5. CONCLUSION

In this paper, we have presented AnyBody, a complete self-organization and routing protocols for Wireless Body Area Networks. AnyBody starts by building a structure for the network made of clusters, and then use this structure to efficiently send packets from source to sink. The setup phase consists of grouping the nodes into clusters, interconnecting those clusters, and finally identifying the routing paths. Extensive simulation has shown that our structure has interesting characteristics, such as a constant number of clusters when the number of nodes increases.

Having a protocol to maintain the cluster structure of Any-

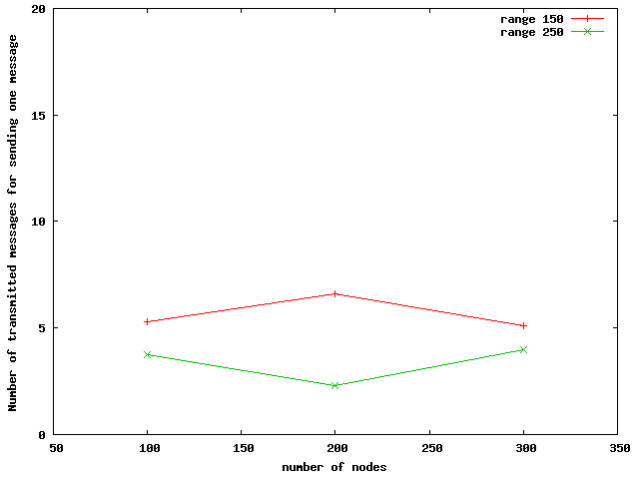

Figure 7: The mean number of individual transmissions to carry a message from source to sink.

Body is considered future work. This could be done periodically (*e.g.* completely rebuilding the structure every 25 hours), or locally on an event-driven basis (*e.g.* when a link breaks). Based on these simulation results, it would be interesting to have a more analytical comparison between the energy consumption of hierarchical and flat routing protocols. This analysis could span over a wide range of both types of protocols. Finally, it would be interesting to merge this analysis on energy consumption with some recent analytical results on network capacity[3] in order to obtain a trade-off.

6. ACKNOWLEDGMENTS

The authors would like to thank Balázs Tirpák, from the Budapest Tech Polytechnical Institution, Hungary, for implementation of AnyBody in GTSNetS, and running simulation during his stay at the CITI Laboratory.

7. REFERENCES

[1] I. F. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: research challenges. In *International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Fort Lauderdale, Florida, USA, December 2004. IEEE.

[2] A. Amis, R. Prakash, T. Vuong, and D. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Annual Joint Conference of the Computer and Communications Societies (INFOCOM)*, number 1, pages 32–41, Tel-Aviv, Israel, March 2000. IEEE.

[3] M. Dohler, T. Watteyne, D. Barthel, F. Valois, and J.-L. Lu. Kumar's, zipf's and other laws: How to structure an optimum large-scale wireless (sensor) network? In *13th European Wireless Conference*, 2007.

[4] H. Frey and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing algorithms in ad hoc and sensor networks. In *Twelfth ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Los Angeles, CA, USA, September 23-29 2006. ACM.

[5] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, 2002.

[6] D. B. Johnson, D. A. Maltz, and J. Broch. *Ad Hoc Networking*, chapter DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks, pages 139–172. Charles E. Perkins, 2001.

[7] B. Karp and H. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Annual International Conference on Mobile Computing and Networking (Mobicom)*, pages 243–254. ACM, August 2000.

[8] N. Mitton, A. Busson, and E. Fleury. Efficient broadcasting in self-organizing sensor networks. *International Journal of Distributed Sensor Networks (IJDSN)*, 2:161–187, 2006.

[9] N. Mitton and E. Fleury. Distributed node location in clustered multi-hop wireless networks. In P. J. Kenjiro Cho, editor, *Technologies for Advanced Heterogeneous Networks: First Asian Internet Engineering Conference (AINTEC'05)*, volume 3837/2005, pages 112–127, Bangkok, Thailand, December 2005.

[10] E. Ould-Ahmed-Vall, G. F. Riley, B. S. Heck, and D. Reddy. Simulation of large-scale sensor networks using gtsnets. In *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 211–218, Atlanta, GA, USA, September 2005. IEEE.

[11] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, USA, February 1999. IEEE.

[12] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)*, Los Angeles, CA, USA, April 2005. IEEE.

[13] C. Prehofer and C. Bettstetter. Self-organization in communication networks: principles and design paradigms. *IEEE Communications Magazine*, 43(7):78–85, July 2005.

[14] F. Theoleyre and F. Valois. About the self-stabilization of a virtual topology for self-organization in ad hoc networks. In *7th International Symposium on Self-Stabilizing Systems (SSS)*, Barcelona, Spain, October 2005.

[15] F. Tholeyre and F. Valois. Virtual structure routing in ad hoc networks. In *IEEE International Conference on Communications (ICC)*, Seoul, Korea, 2005.

[16] S. Xu and T. Saadawi. Does the ieee 802.11 mac protocol work well in multihop ad hoc networks? *IEEE Communications Magazine*, 39(6):130–137, June 2001.

[17] O. Younis, M. Krunz, and S. Ramasubramanian. Node clustering in wireless sensor networks: Recent developments and deployment challenges. *IEEE Network*, 20(3):20–25, May/June 2006.