



Budapest University of Technology and Economics  
Faculty of Electrical Engineering and Informatics  
Department of Networked Systems and Services

# Location based data dissemination algorithms in self-organized mobile networks

## Helyfüggő adatterjesztő algoritmusok mobil önszerveződő hálózatokban

THESIS

*Author*

András Kőkuti

*Supervisor*

Vilmos Dr. Simon

16. December. 2013

# Contents

|                                                                      |           |
|----------------------------------------------------------------------|-----------|
| <b>Abstract in Hungarian</b>                                         | <b>3</b>  |
| <b>Abstract</b>                                                      | <b>5</b>  |
| <b>Introduction</b>                                                  | <b>6</b>  |
| <b>1. Multihop broadcast protocols</b>                               | <b>8</b>  |
| 1.1. Location based protocols . . . . .                              | 10        |
| 1.2. The 3-way handshaking . . . . .                                 | 12        |
| 1.3. The Distance Based Handshake Gossiping (DBHG) . . . . .         | 15        |
| 1.4. The Valency Based Handshake Gossiping (VBHG) . . . . .          | 16        |
| 1.5. The Average Valency Based Handshake Gossiping (AVBHG) . . . . . | 17        |
| 1.6. The Adaptive Handshake Gossiping (AHG) . . . . .                | 18        |
| <b>2. Spatial computing</b>                                          | <b>19</b> |
| 2.1. Overview of spatial computing . . . . .                         | 19        |
| 2.2. Mobile ad hoc and sensor networks . . . . .                     | 21        |
| 2.3. Swarm and modular robotics . . . . .                            | 22        |
| 2.4. Parallel and reconfigurable computing . . . . .                 | 23        |
| 2.5. Amorphous computing . . . . .                                   | 24        |
| 2.6. The proposed spatial computing DSL . . . . .                    | 25        |
| <b>3. The novel protocol</b>                                         | <b>26</b> |
| 3.1. Motivation . . . . .                                            | 26        |
| 3.2. The Direction Based Handshake Gossiping . . . . .               | 28        |
| 3.3. Performance analysis . . . . .                                  | 31        |
| 3.3.1. The simulator . . . . .                                       | 31        |
| 3.3.2. The performance indicators . . . . .                          | 32        |
| 3.3.3. Results . . . . .                                             | 33        |
| <b>4. The novel solution in spatial computing environments</b>       | <b>39</b> |
| 4.1. Overview . . . . .                                              | 39        |
| 4.2. The dissemination plugin . . . . .                              | 40        |
| 4.3. The Proto code . . . . .                                        | 42        |
| 4.4. Performance analysis . . . . .                                  | 44        |

|                                                                        |           |
|------------------------------------------------------------------------|-----------|
| 4.4.1. The simulator settings and the performance indicators . . . . . | 44        |
| 4.4.2. Results . . . . .                                               | 45        |
| <b>5. Conclusion</b>                                                   | <b>48</b> |
| <b>Acknowledgement</b>                                                 | <b>50</b> |
| <b>List of figures</b>                                                 | <b>51</b> |
| <b>List of tables</b>                                                  | <b>52</b> |
| <b>References</b>                                                      | <b>56</b> |

# Abstract in Hungarian

A hagyományos távközlési rendszerek mellett egyre inkább felértékelődik az infrastruktúra nélküli, peer-to-peer kommunikációt lehetővé tevő elosztott hálózatok szerepe, közöttük jelentős szerepet tölt be a mobil önszerveződő hálózatok családja. Önszerveződő hálózati környezetben, ahol nem áll rendelkezésre központi infrastruktúra az átviendő csomagok tárolására és továbbküldésére, nagyon fontos szempont, hogy a többes ugrásos szórt adású (multi-hop broadcast az angol nyelvű szakirodalomban) adatterjesztő algoritmusok milyen hatékonysággal szórják szét az információt a hálózatban. Ezen adatterjesztő algoritmusok komoly felhasználást nyerhetnek olyan helyzetekben, mint például vészhelyzetek esetén (pl.: tűz üt ki az épületben, földrengés ...), információbegyűjtés és -terítés szenzorhálózatokban (pl.: rajkövetés), forgalom irányításban és még számos más területen. Annak érdekében, hogy minimalizáljuk a csomópontok energiafelhasználását (amely nagyon szűkös erőforrás egy mobil önszerveződő hálózatban), olyan lokális információkra kell hagyatkoznunk, mint a szomszédos csomópontok száma, aktuális helyzete, általuk vett üzenetek halmaza vagy szomszédaik száma. Amennyiben ezen lokális információk rendelkezésre állnak, az eszközök helyinformációi alapján kitüntetett irányok menti adást is végezhetünk (üzenet-továbbadási valószínűsésekkel megvalósítva), amely képes drasztikus mértékben visszaszorítani a hálózatban forgalmazott felesleges üzenetek számát.

Fontos felhasználási területet nyerhetnek az ún. spatial computing kommunikációban, amelynek lényege, hogy a teret megfeleltethetjük számítási egységeknek, s így számos probléma (például csoportosulás, csoportkövetés ...) sokkal egyszerűbben kezelhetővé válik. Az eddig megoldások primitív adatterjesztéssel rendelkeztek (vak elárasztás), melynek eredményeképp rendkívül pazarló megoldások születtek. Ezeket a kommunikációs primitíveket lecserélve hatékonyabb kommunikáció és ennek eredményeképp hatékonyabb működés valósulhat meg.

Ezen megfontolások alapján egy olyan iránymenti adatterjesztő protokollt fejlesztettem ki, amely iránymenti adást valósít meg a fentiekben ismertetett rendkívül dinamikus környezetben. A protokoll tesztelésre került az általam fejlesztett mobil önszerveződő hálózati szimulátorban és ezen szimulátor segítségével a szakirodalomban publikált három másik algoritmussal (Distance Adaptive Dissemination, a Ni et al protokoll és a General Probabilistic Broadcast Algorithm) került összehasonlításra, különböző paraméterbeállítások mellett. Az eredményekből látható, hogy a legfontosabb teljesítményhatékonysági mértékeket tekintve számos olyan eset van, amely során hatékonyabban működik az általam fejlesztett protokoll, és gyorsabb lefedettséget is biztosít.

A protokoll teljesítmény validációja után egy még realisztikusabb környezetben is implementálásra került, még hozzá az MIT által fejlesztett Proto Spatial Computing rendszerben. A Proto-ban végzett méréseim alátámasztják, hogy lecserélve a jelenlegi primitív információterjesztési megoldást, nem csak egy sokkal effektívebb megoldást kaphatunk, hanem egy olyan algoritmust, amely képes spatial computing környezetben is hatékonyan működni.

# Abstract

In line with traditional communication systems, more and more attention is given to autonomous, self-organized networks with no central infrastructure, based on peer-to-peer communication. Designing multihop broadcast protocols for these networks is a complex problem as the task of these protocols is to disseminate messages in a network effectively while avoiding unnecessary use of resources. The vast majority of these protocols (as those used in the present day Internet) do not use spatial information of the nodes to optimize the bandwidth and channel usage. By increasing the awareness of the nodes and equipping them with their physical location, we can achieve a higher level of autonomous functioning, better performance, and higher level communication primitives, like transmitting in a given direction.

Spatial computing implementations and languages do consider the physical location properties, however the communication protocols in these implementations are far from optimal, as they utilize oversimplified communication solutions.

A novel communication protocol has been designed based on the spatial properties of the system, called the Direction Based Handshake Gossiping, which was implemented in my self-organizing network simulator. For performance comparison three other location based data dissemination protocols have been picked from the literature (Distance Adaptive Dissemination, the General Probabilistic Broadcast Algorithm, and Ni et al's location-based scheme). The simulation results show that the novel solution over-performs the other three protocols in terms of control overhead and number of duplications, which is crucial in self-organized mobile networks, where radio bandwidth and energy are usually scarce resources.

To prove the availability of the novel algorithm, it has been implemented in a more realistic simulator, namely the Proto spatial computing simulator which was developed by MIT. The results reveal that this protocol is not only a more effective solution compared to the current data dissemination realization in Proto, but it can deliver proper results in various spatial computing applications.

# Introduction

Today more and more appreciated are networks with no infrastructure, which are self-organized and where the communication is based on a peer-to-peer model. A major representative of these networks is the mobile ad hoc network (MANET [1]), which is a self-configuring network, consisting of mobile devices, that can communicate with each other via a radio interface. The distributed and self-configuring nature of these networks, combined with their ease and flexible deployment, make MANETs appealing for a wide range of application scenarios including, e.g., emergency situations, sensor networks for environmental monitoring [2], vehicular ad hoc networks [3], and many others [4, 5].

The common denominator behind all these application scenarios is the fully distributed nature of the network infrastructure supporting them, together with the support of nodes mobility. In particular, this last characteristic is reflected in a network topology that can change over time, depending on the density and mobility of nodes.

Many mobile self-organized network applications rely on the existence of a broadcast medium for the dissemination of control information. A first implementation of this communication primitive can be flooding: if a node receives a message, it immediately repeats it, if the message has been not received by him earlier. However, this implementation is far from optimal, and collisions in the medium can lead to serious congestion and loss of packets. To improve the primitive implementation, many efficient broadcast techniques were designed, that take into account some information about the nodes' surroundings (local information), instead of blindly repeating every packet.

These algorithms differ in their assumptions about the environment (like the assumption of a connected or disconnected network) and in the information available for decision (i.e. the availability of location information via GPS or other means). The central problem of broadcast algorithms is to decide when and who should retransmit messages. Nodes have to forward packets so the message reaches every part of the network, however the performance relies heavily on the set of nodes that do this. When nodes decide whether to retransmit or not, they actually decide whether they are part of the forwarding set (the set of nodes retransmitting the message) or not. Too many retransmissions cause collisions and waste the network bandwidth, but choosing the smallest forwarding set is not easy because a global view of the network is not available, and local information gets obsolete very quickly if the velocity of the nodes are high. Also, it is not preferred to minimize the size of the forwarding set, as it would degrade the robustness of the communication primitive (the message might not reach every node in the network).

Only a few of the existing multihop broadcast solutions utilize the up to date physical location and distance information of the nodes from the environment, which can provide a more reliable picture on the current state of the local topology (which in our case changes dynamically). By obtaining additional spatial information, the nodes can make a better guess whether they should be part of the forwarding set or not.

Traditional models of computation have abstracted out physical locations in space, like the present day Internet, however, as the number of elements in mobile self-organized networks scales up, computations must be distributed in space and therefore location information impacts the performance and feasibility of the computation. Furthermore as we embed computing in the physical world, position serves as an input to computation and also as a key part of the desired result of the computation. All of the domains that have been mentioned so far share a property: a close relationship between the computation and the arrangement of the computing devices in space. In self-organized systems the difficulty of moving information between any two devices is strongly dependent on the distance between them, and the functional goals of the system are generally defined in terms of the system's spatial structure.

Across many different communities, researchers have built domain specific languages (DSLs) that use spatial abstractions to simplify the problem of programming aggregates, taking a wide variety of different approaches, often blending together generally applicable spatial concepts with specific requirements of a target sub-domain or application, and have met with varying degrees of success[6]. However, these spatial computing DSLs or other implementations are not really taking into account the communication aspects of self-organized networks, they apply simple heuristics for information dissemination and topology discovery. As the majority of these implementations are tested only in simulation frameworks, they are not coping with the real life radio channel challenges: broadcast storms[7], intra and inter-channel interference, duplications of messages are not considered.

In my thesis, I would like to give an overview of the existing location based multihop protocols, together with a survey and classification of the spatial computing DSLs, pointing out that novel, location based communication protocols are needed to optimize the networking load if using them in a real environment. A novel communication protocol is proposed (called Direction Based Handshake Gossiping), leveraging available spatial information. This novel protocol was compared with location based multihop broadcast protocols from the scientific literature, namely the Distance Adaptive Dissemination, the General Probabilistic Broadcast Algorithm, and Ni et al's location-based scheme, by observing the number of the duplications and coverage time. After a deep analysis of the protocols in the self-organizing network simulator, my algorithm was implemented in the Proto spatial computing system as well, to show that this solution is a fine choice for implementing a new communication primitive (information dissemination) in the Proto spatial computing system.

# 1. Chapter

## Multihop broadcast protocols

The inexperienced first implementation was the blind flooding[8] algorithm. The operation of BF algorithm is very simple, once a node receives a message, it immediately rebroadcasts it. We might think this is a good solution for data distribution, because it is very simplistic, and covers all the nodes (assuming that the network is connected), but it was realized very soon, that this is far from optimal. The effect of frequent packet repetition may result in collisions in the medium and can lead to serious congestions, such as broadcast storms[7].

The Counter Based Method, originally introduced in [9] is one of the first controlled broadcast methods. It is based on a simple observation, that if a duplicate of a packet is received, then the probability of reaching any new node is low. To exploit this idea, the nodes do not immediately transmit on the receipt of a packet, but instead they wait for a random amount of time, which is called the Random Assessment Delay (RAD). If a duplicate is received during the RAD, a counter is increased. If the counter reaches a threshold before the RAD expires, the node cancels the transmission. The original method has different adaptive versions[10], which try to adapt the length of the RAD and the threshold of the duplicate counter to the current network conditions.

Another broadcast method is the Gossiping algorithm [9], where every node broadcasts the received message with a predefined probability. The optimal probability can be calculated off-line for a static network, or can be learned adaptively. Some of these adaptive versions are covered in [11]. While the Counter Based method is a fine example of a simple heuristic based deterministic algorithm, Gossiping is an example of the simple heuristic based stochastic methods.

Networks separated to disjoint partitions pose a new problem, that the previously mentioned protocols can not handle. Hypergossiping [12] is specifically designed for partitioned networks, where nodes are mobile, and partitions join and split from time to time. It is an advanced version of the Gossiping algorithm, extended by neighbor information and partition join detection. The algorithm uses a simple adaptive gossiping strategy for in-partition forwarding, but rebroadcasts some of the packets if it detects a join with another partition. The join detection is based on the simple heuristic, that the nodes in the same partition received the same messages recently. Every node maintains a list, called LBR (Last Broadcasts Received), of the recently broadcast messages. They send HELLO mes-

sages periodically, to indicate their presence. When a new node is detected, one of the nodes include its LBR in the next HELLO message. When the other node receives this LBR, it compares with its own LBR. If the overlap between the LBR of two nodes is smaller than a threshold, then the node is considered coming from another partition, so a new message is sent, called BR (Broadcasts Received), which contains the list of messages that the node already received. From this the other node knows that a partition join happened, and rebroadcasts all the messages that were not inside the other nodes BR. After this rebroadcast, dissemination proceeds using adaptive gossiping.

A fine example of a self-pruning algorithm is the Scalable Broadcast Algorithm (SBA) [13]. It requires 2-hop neighbor information and the last sender ID in the broadcast packet. When a node  $v$  receives a broadcast packet from a node  $u$ , it excludes the neighbors of  $u$ ,  $N(u)$  from the set of its own neighbors  $N(v)$ . The resulting set  $B = N(v) - N(u)$  is the set of the potentially interested nodes. If  $|B| > 0$  then the node will start a Random Assessment Delay (RAD). The maximum RAD is calculated by the  $\frac{d_v}{d_{max}} * T_{max}$  formula, where  $d_v = |N(v)|$  and  $d_{max}$  is the degree of the node with the largest degree in  $N(v)$ , and  $T_{max}$  controls the length of the RAD. Nodes choose the time of transmission according to a uniform random variate from this interval. This ensures that nodes with a higher degree often broadcast packets before nodes with fewer neighbors.

A quite different approach from the algorithms discussed so far is the IOBIO algorithm[14]. It is a variation of the SPIN [15] dissemination protocol. It uses a simple 3-stage handshake to discover neighbors that are interested in one of the carried messages. The goal of the protocol is to reduce the unnecessary load of neighboring nodes by duplicate or unneeded data ("spamming"). There are three IOBIO message types that are used by the protocol. The ADV (Advertisement) messages are sent periodically, and they contain the list of messages that the sending node has. Neighbor nodes indicate their interest in the advertised messages by sending a REQ (Request) packet. In response to the REQ, the originator node sends the required DATA packets. The transmission of a REQ after an ADV is not done immediately, but after a random delay. During this delay, the nodes listen to each other, and they only request packets that were not requested before.

### 1.1. Location based protocols

Location based multihop protocols use spatial information to make their decision about message broadcasting, in addition to the classical neighborhood and topology information. In most of the cases this means that the device should have a Global Positioning System (GPS) or another positioning technique to acquire this information. These methods use HELLO messages, just like the neighbor information algorithms do, but they collect the location of the neighbors as well. There are other algorithms that need to know only the distance to their neighbors, like the Distance Adaptive Dissemination (DAD)[16] algorithm, which may be measured by signal strength. In this case, the use of HELLO messages may not be necessary, as it uses distance information instead of exact positions. The authors propose a scheme that chooses forwarding nodes according to their distance, using the signal strength as a measure for distance. The goal of the algorithm is to try and get the outermost neighbors of a node to rebroadcast, thus minimizing overlap of transmission ranges. It uses 1-hop neighbor information and records signal levels from the neighbor nodes. The authors propose two variants called DAD-NUM and DAD-PER. DAD-NUM chooses a signal strength  $S_{thres}$  so that there are  $k$  number of neighbors that have transmitted with a signal strength lower than  $S_{thres}$ . On arrival of a new packet, the node checks if the signal strength is greater of  $S_{thres}$  or not. If it is smaller then the node rebroadcasts. DAD-PER is very similar, but instead of finding the  $k$  farthest nodes it chooses  $p$  percent of them.

Another example for location based protocols is the Optimized Flooding Protocol (OFP) which is a deterministic dissemination algorithm, that uses a geometric approach instead of the usual graph theory solutions. The algorithm tries to cover the 2D space efficiently with  $R$  radius circles. This paper does not detail the algorithm here, mostly because covering the simulation area with circles is not an appropriate approximation in urban and in-building environments. Details can be found in [17].

Stojmenovic's method[18] also uses position information, it is a variant of Wu and Li's algorithm[19]. There are two important improvements over the original algorithm: it uses 1-hop information coupled with position information to implement the marking process and rules 1 and 2 introduced in Wu and Li's algorithm. The other difference is that it also implements a random backoff scheme, similar to the Scalable Broadcast Algorithm (SBA). The nodes do not broadcast immediately, but rather wait for a random amount of time. If a node  $v$  hears a transmission during this interval from a node  $u$  then he removes  $N(u)$ , the neighbors of  $u$ , from its own neighbor set  $N(v)$ .

Two other protocols are introduced by Ni et al[9]. The first one tries to use the relative distance between hosts to make the decision. Suppose host  $H$  heard a broadcast message from  $S$  for the first time. If the distance, say  $d$ , between  $H$  and  $S$  is very small, there is little additional coverage  $H$ 's rebroadcast can provide. If  $d$  is larger, the additional coverage will be larger. In the extreme case, if  $d = 0$ , the additional coverage is 0 too. If  $H$  has heard the same message several times, let  $d_{min}$  be the distance to the nearest host from which the same message is heard. If  $d_{min}$  is smaller than some distance threshold  $D$ , the rebroadcast transmission of  $H$  is canceled. The other one uses positioning devices such as GPS (Global

Positioning System) receivers, to estimate the additional coverage more precisely. Let a host's location be  $(0, 0)$ , and suppose the host has received the same broadcast message from  $k$  hosts located at  $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$  positions. So the additional area that can be covered by the host's rebroadcast can be calculated. Let  $AC((x_1, y_1), (x_2, y_2), \dots, (x_k, y_k))$  denote the additional coverage divided by  $\pi r^2$ , where  $r$  is the central node's radio range. This value should be compared to a predefined coverage threshold  $A$  ( $0 < A < 0.61$ ) to determine whether the receiving host should rebroadcast or not.

Min-Te Sun et al.[20] identified two primary design issues regarding information dissemination protocols, namely the defer time generation and the redundant message classification, which they used to devise a broadcast protocol that enjoys flooding's high reachability and the bandwidth efficiency of the non-flooding schemes'. Let us consider the issue of setting defer times. Nodes with a larger defer time are scheduled to retransmit a message later than those with a smaller defer time. Unless a node decides on redundant/non-redundant retransmission regardless of other nodes' retransmissions, the node with a larger defer time is more likely to find its retransmission redundant than the node with a smaller defer time. Since the purpose of a retransmission is to forward the message to more nodes, it seems plausible to let a node cover new areas by retransmitting the message earlier than the node covering less of the still uncovered area. Thus, instead of randomly choosing a defer time, they have proposed the Distance-based Defer Time Scheme. When node  $S$  receives a broadcast message from node  $N$ , it sets the defer time to a value inversely proportional to a power of  $||\vec{SN}||$ . That is,  $defer\_time = Max\_Defer\_Time * (R^\epsilon - ||\vec{SN}||^\epsilon)/R^\epsilon$ . The second issue is how to identify a redundant retransmission. Since broadcast service requires high reachability, a retransmission should not be easily discarded unless the coverage area is known to be completely covered. To achieve the maximal reachability and fast computation, they propose the Angle-based Scheme to identify redundant retransmissions. They calculate two values:  $\alpha$  and  $\beta$  form the intersection of covered circles. So if during its defer period node  $S$  receives the retransmissions of a same message from a number of neighbors with cover ranges  $[\alpha_1, \beta_1], \dots, [\alpha_k, \beta_k]$  and if  $\cup_i [\alpha_i, \beta_i] = [0, 360]$ , then  $S$  will not further retransmit the message.

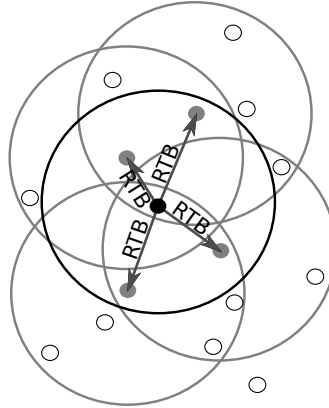
The Border Retransmission Based Probabilistic Flooding (BRBPF) was presented in [21]. They have observed that the distance between two nodes with a full duplex communication can be evaluated by comparing their neighbor lists. When two nodes  $src$  and  $dist$  can contact each other, the union of their communication areas ( $Z_{src}$  and  $Z_{dest}$ ) can be partitioned in three zones:

- $Z_a = Z_{src} \cap \overline{Z_{dest}}$ : the communication area covered only by  $src$ ,
- $Z_b = \overline{Z_{src}} \cap Z_{dest}$ : the communication area covered only by  $dest$ ,
- $Z_c = Z_{src} \cap Z_{dest}$ : the communication area covered by both  $src$  and  $dest$ .

They define a ratio:  $\mu = \frac{N_b}{N_a + N_c}$ , where  $N_i$  denotes the number of neighbors in zone  $Z_i$ . Based on this parameter the  $dest$  determinates its own probability of sending  $p$  with the next formula:  $p = \frac{A-\alpha}{M^\alpha} \mu^\alpha + \alpha$ , where  $\alpha$ ,  $A$  and  $M$  are fine-tuning parameters.

## 1.2. The 3-way handshaking

There is a group of adaptive multihop broadcast protocols: the Distance Based Handshake Gossiping, the Valency Based Handshake Gossiping, the Average Valency Based Handshake Gossiping and the Adaptive Handshake Gossiping [22], which all assign message forwarding probabilities to the nodes, calculated from the network parameters, such as the degrees of nodes or distance of the nodes from each other, by exploiting the available spatial information in the network. The protocols differ from each other only in the methods of calculating the message forwarding probabilities. It is assumed that the devices have the knowledge of their actual geographical positions with the help of GPS transceivers or through other positioning techniques (a realistic assumption considering today's smart phones and notebooks). All three protocols consists of the same three phases. The first phase(Figure 1.1) is called RTB [23] (Ready-To-Broadcast), which is similar to the RTS phase in CSMA/CA [24]. The device, which has a data message to disseminate, broadcasts an RTB signal message in its radio range. It places its own unique identifier (for example MAC address) in the packet and in the case of multi-message communication (where mobile nodes disseminate different kind of data messages) the message type and identifier.

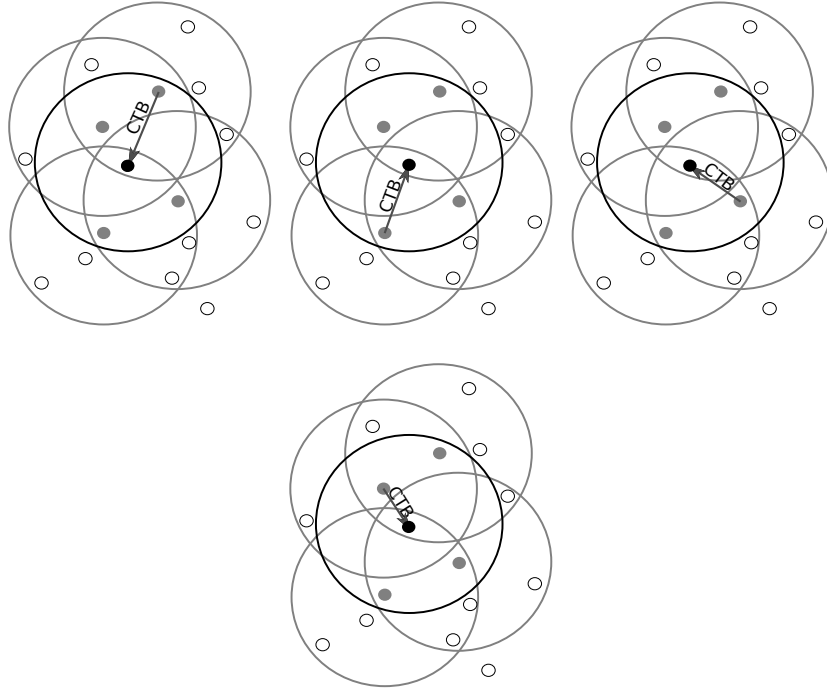


**1.1. figure.** *The first phase*

In Figure 1.1 the black point marks the sender node, the grays are the nodes in the senders range, while the whites depict the other devices in vicinity. The transmitting device, marked with black colour, is entering the first phase by disseminating an RTB packet. The neighboring nodes (gray colour) in its radio range receive the message due to the radio broadcast.

In the next phase all of the nodes, which received the RTB message, start a random timer from the  $[0; t_{CTB}]$  range. This timer ensures that the CTB (Clear-To-Broadcast) messages in reply to the RTBs do not cause broadcast storms, and this way avoiding collisions. This message includes the CTB sender's and the "recipient"'s (which sent the RTB packet in the previous phase) unique identifier. With the returned identifiers, the central (marked with black colour in Figure 1.1) node can decide, if it is the recipient of the packet or another node at a 2-hop distance. The message also contains the geographical coordinates of the CTB sender (by using the GPS transceiver or other positioning technique), and the

number of its neighbors, namely its degree as well. To detect their neighbors, the nodes periodically distribute HELLO signal messages. All devices transmit a HELLO message at appropriate intervals for monitoring their own environment. If HELLO is received, the node updates the list of its neighbors. So the received HELLO messages help to determine the actual degree of the given node. In addition to these, CTB messages also include the identifiers of the messages (getting it from the RTB packet), which are needed by the CTB sender node.

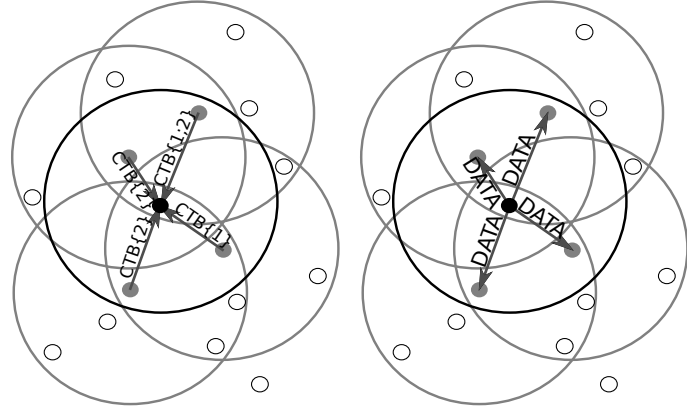


**1.2. figure.** *The second phase*

First, as shown in Figure 1.2, the node in the (a) subfigure gets the opportunity to broadcast the CTB message, because its timer expired earlier than the other nodes which also started a timer upon receiving the RTB message. Then the nodes, shown in (b), (c) and (d) subfigure will transmit one after other, determined by their timers. Meanwhile the central node is waiting for a  $t_{DATA}$  time, which is calculated as the sum of the  $t_{CTB}$  and the maximum propagation time.

The last phase is responsible for the data message dissemination, since the first two phases use only signal messages. The central node starts the data message transmission (the node which sent out previously the RTB packet) after its  $t_{DATA}$  timer expired. First, it summarizes, based on its 1-hop neighbor list, which nodes are currently within its radio range, and to these IDs (which can be found in the reply CTB messages) assigns the degrees and the coordinates. After it calculates the retransmission probability for each adjacent device. From the CTB messages the central node can determine which data messages need to be disseminated, by calculating the union of the requested data messages.

Figure 1.3 shows an example of how is the central node determines which data messages should be disseminated. The adjacent devices, marked by gray colour, put the requested



(a) An example for requesting the data messages (b) The DATA message dissemination

**1.3. figure.** *The third phase*

data message IDs in the CTB packets (in braces). Based on these identifiers, the central node calculates which messages (here, the messages with ID 1 and 2), need to be broadcasted. Thus, all parameters are known (messages, probabilities) in order to complete the information dissemination process.

The DATA message's header contains the sender (same as RTB sender) unique identifier and the calculated message forwarding probabilities, the payload is comprised of the data message. The sender's identifier is necessary as there may be other 2-hop distance nodes, which are in the same transmission phase. Therefore the common neighboring nodes can distinguish which node's DATA packet have been received. From the rest of the header every adjacent node get its message forwarding probability, as it is assigned to the nodes ID. As it has been mentioned, the protocols only differ in their methods of the probability calculation, so in the next subsections the concrete protocols will be described.

### 1.3. The Distance Based Handshake Gossiping (DBHG)

For the Distance Based Handshake Gossiping protocol the degree of the nodes is not needed, so that can be omitted from the CTB message. Moreover, the periodically sent topology discovery signal messages (HELLO) are unnecessary as well. For calculating the probabilities only the geographical coordinates are needed, which can be found in the CTB message. The calculation is based on the nodes distances, hence the name, which is defined using the following formula.

$$d_i = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2} \quad (1.1)$$

$$d_i = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2 + (z_0 - z_i)^2} \quad (1.2)$$

The  $x_0, y_0, z_0$  parameters are the central node's, while  $x_i, y_i, z_i$  are the  $i$ . adjacent device's coordinates. The central node selects the largest distance, i.e. the highest value, it will be  $d_{max}$ .

$$d_{max} = \max_{\forall i} \{d_i\} \quad (1.3)$$

The protocol, based on these parameters, assigns the following message forwarding probability to the  $i$ . neighboring node:

$$p_i = \frac{d_i}{d_{max}} \quad (1.4)$$

This means, being further from the central node, the message forwarding probability will get higher for the given adjacent node. This way the duplication caused by closer nodes can be avoided, moreover the area covered by more distant nodes is more likely to contain previously not covered nodes [25]. This means that the furthest nodes' (which is the farthest from the central node) assigned probability will be 1, so it will initiate the retransmission with 100%. Thus, in the set of the neighboring nodes, there will be at least one node (the farthest), which will continue the information dissemination.

#### 1.4. The Valency Based Handshake Gossiping (VBHG)

A shortcoming of the previous protocol is that it does not consider the degree of the nodes when assigning the message forwarding probabilities to the neighboring nodes. It may happen, that the farthest adjacent device has no other neighbors, than the initial transmitter, nevertheless it has got 100% forwarding probability because of its position. Furthermore the protocol still does not „penalize” the nearby nodes enough. In order to avoid unnecessary duplication of messages, the nearby nodes with a high degree need to get a low chance for the retransmission, while the nodes further away should have a higher probability. The Valency Based Handshake Gossiping is trying to achieve this goal. Of course, in this case it is necessary to distribute periodically the topology discovery signal messages, and some additional parameters are needed. The  $d_i$ ,  $d_{max}$  variables are identical to those used in the DBHG protocol. The  $d_{average}$  indicates the average value of the distances, calculated as an arithmetic mean of the central node-neighbors distances

$$d_{average} = \frac{1}{n} * \sum_{i=1}^n d_i \quad (1.5)$$

where  $n$  is the number of adjacent nodes, while  $f_i$  is the  $i$ . neighboring node's degree. The neighbors can be distinguished by the unique identifier in the CTB message. Similar to  $d_{max}$  an  $f_{max}$  is needed for VBHG, to determinate the probabilities.

$$f_{max} = \max_{\forall i} \{f_i\} \quad (1.6)$$

The probability is calculated in the following way for the VBHG protocol:

$$p_i = \begin{cases} \frac{d_i}{d_{max}} - \frac{d_i}{d_{max}} * \frac{f_i}{f_{max}}, & \text{if } d_i < d_{average} \\ \frac{d_i}{d_{max}} + \frac{d_{max}-d_i}{d_{max}} * \frac{f_i}{f_{max}}, & \text{if } d_i \geq d_{average} \end{cases} \quad (1.7)$$

As equation 1.7 shows the calculation is comprised from two disjoint cases. The upper equation is used, when the adjacent device is closer to the transmitter than the average distance. In this case, the goal is to assign a lower retransmission probability, because this node's radio range is already largely covered by the central node, so the rebroadcast would cause many unnecessary duplications. On the other hand, when the device is far from the transmitter, more likely it covers new, still not covered nodes, the lower equation is utilized for this case.

### 1.5. The Average Valency Based Handshake Gossiping (AVBHG)

The Average Valency Based Handshake Gossiping is based on the experience provided by the two previous versions. It uses one more parameter, the  $f_{average}$  indicates the average valency of the network:

$$f_{average} = \frac{1}{n} * \sum_{i=1}^n f_i \quad (1.8)$$

where  $n$  is the number of the neighbors for the given node. Consequently, the probability is determined by:

$$p_i = \begin{cases} \frac{d_i}{d_{max}} + \frac{d_{max}-d_i}{d_{max}} * \frac{f_i}{f_{max}} & \text{if } f_i \geq f_{average} \\ & \text{and } d_i \geq d_{average} \\ \frac{d_i}{d_{max}} - \frac{d_{max}-d_i}{d_{max}} * \frac{f_i}{f_{max}} & \text{if } f_i \geq f_{average} \\ & \text{and } d_i < d_{average} \\ \frac{d_i}{d_{max}} & \text{else} \end{cases} \quad (1.9)$$

As it can be seen, the probability calculation is a combination of the previous two protocol probability assignments, therefore those nodes get lower probabilities for retransmission which are closer to the central node than the average distance, and at the same time they have a high valency value.

### 1.6. The Adaptive Handshake Gossiping (AHG)

All three versions of the protocols should be installed on the mobile device, as they differ only in the way of calculating the forwarding probability, therefore, no additional memory or processing cost is introduced. Thus, the nodes can adaptively change the function of the probability assignment, depending on which parameter(s) needed to be optimized. If the system requires a rapid dissemination, the device would select the DBHG version. However, if it is of high cost to deliver a message, then they should switch to VBHG, while AVBHG being a trade-off between the two aspects.

Based on these an adaptive handshake gossiping protocol was designed, consisting of the DBHG and the VBHG protocols. If the device senses, that it is in a dense environment (this information can be obtained from the node's own local database, for example its valency value), it can change to the VBHG version, as this version causes the smallest number of duplications. Furthermore it performs nearly as fast as the DBHG in a dense environment. In a sparse environment, the DBHG protocol should be used, because of its benefits. Thus after the second phase, when the CTB messages are received, the transmitting node should adapt the function of the probability assignment (utilizing DBHG or VBHG) depending on its own local parameters.

$$p_i = \begin{cases} \text{Equation (1.4)}, & \text{if } f_{transmitter} < f_{average} \\ \text{Equation (1.7)}, & \text{if } f_{transmitter} \geq f_{average} \end{cases} \quad (1.10)$$

where  $f_{transmitter}$  is the valency value of the currently transmitting node, while the  $f_{average}$  is the obtained valency parameter, which indicates whether the node is in a dense or in a sparse environment. It can be calculated by the average of the local environments from time to time that the nodes store in their own local database.

## 2. Chapter

# Spatial computing

### 2.1. Overview of spatial computing

|                   | Type        | Platform                | Layer      |
|-------------------|-------------|-------------------------|------------|
| Regiment          | Imperative  | Wireless sensor network | AD         |
| TOTA              | Imperative  | Wireless sensor network | AD, SM     |
| TinyDB            | Declarative | Wireless sensor network | SC         |
| Proto             | Functional  | Any                     | SC, AD     |
| Erlang            | Functional  | Parallel hardware       | SC, AD, SM |
| Cellular automata | Declarative | Simulation              | SC, AD     |
| Graph Grammars    | Functional  | Robot swarms            | SC, AD     |

**2.1. table.** *Categorization of spatial computing DSLs*

In this section I will give a high level overview of the spatial computing approaches, starting with the idea of spatial computing and then enumerating some of the most influential proposed solutions in the field. A short introduction of the environments will be given in which the described system is designed to work; along with the main goal of the approach. As we will see there are at least two different levels of abstraction below spatial computing (SC) serving as building blocks to the next level[6].

As technology develops and devices become cheaper and more widespread, there will be an increasing number of networked mobile devices around us. These networks have two properties in common:

1. the difficulty of moving information between any two devices is strongly dependent on the distance between them, and
2. the "functional goals" of the system are generally defined in terms of the system's spatial structure.

However, if we want to use them for a specific purpose we face quite a challenge: we cannot program the aggregate behavior of these devices efficiently. What we have instead are means to program the devices one by one. Getting them to produce the wanted global behavior becomes a daunting task quite easily. One of the goals of spatial computing is to help the system designer with this global to local transformation. The ideal spatial computing

system allows the programmer to specify some global behavior in terms of geometric fields (or whatever spatial structure) and the system compiles the given specification to local code that can be run on the devices. The local code produced mostly runs on virtual machines because there needs to be a layer between the individual devices and the general bytecode generated from the global specification. These virtual machines help maintain the spatial abstraction and free the programmer to think about communication issues or scenarios where certain devices vanish or new devices appear in the network.

Functionality of the virtual machines mentioned in the previous paragraph can be divided into two distinct items: abstract device (AD) and system management (SM). The abstract device functionality hides differences between the devices in the network by providing a unified interface for every device to interact with any other device in the network. We can imagine it as providing a common protocol. The other aspect of virtual machines are the system management role, which concerns itself with interfacing with the platform: actuators, sensors, battery and thread management.

Let's explore a few of the many different research areas that can benefit from the spatial computing approach and review some of the most influential technologies/languages in those fields. The following enumeration won't describe all related fields and technologies, rather I would like to introduce some of the ideas that can be approached from a spatial computing viewpoint.

## 2.2. Mobile ad hoc and sensor networks

The first big area to cover are the wireless sensor networks. These networks consist of small, resource constrained devices that either have the ability to move or the whole network is fixed after deployment. This mobility aspect makes it much harder to maintain an abstraction over the dynamically changing network and we have to be more careful not to overuse certain resources of the nodes. Battery power is always the most scarce resource in this kind of network and there are usually two energy intensive tasks: communication and moving of the nodes. A basic example from the field is Regiment[26]. Regiment is a spatiotemporal macroprogramming language, meaning that programs written in it make use of not just the nodes' locations in physical space and topological relationships, but also time-varying sensor data as well. A Regiment program mainly manipulates signals, which are functions that map a time to a value. Signals can carry a number of different types of values ranging from primitive types (integer, float, etc.) to tuples of primitive types. The other important concept in Regiment is the region, which is a collection of signals. Regions can be formed based on spatial or topological information including neighborhoods with a certain hop count or geometric distances from the originating node.

An interesting group of domain specific languages (DSL) for wireless sensor networks are the database-like DSLs. These languages treat the sensor network as a database and help the user with data aggregation. They work on a higher abstraction level, where the low level networking and routing problems are hidden from the user completely. An example for this group is TinyDB[27], which allows the users to issue SQL-like queries to the sensor network and uses the high level description of the query to optimize it. The user usually interacts with a node in the network, TinyDB builds a routing tree from this node to all the nodes in the network and maintains it continuously. When a sensor sends some data to the root, it can be aggregated with other sensors' data while routed back to the root. By lifting the aggregation and routing tasks from the application developer it becomes much easier to write TinyDB requests, while the requests can be optimized in the network. The spatial aspect of the system is that the user can specify the area and the time of interest and the device level code is compiled from the high level description the user provides.

TOTA[28] on the other hand does not have the same notion of space as Regiment or TinyDB. TOTA is a middleware infrastructure to provide applications with contextual information. The main components are tuples, which can be injected in the network and propagate through nodes in a predefined pattern. This pattern is called the propagation rule and defines how the tuple evolves depending on other tuples it interacts with on nodes and how far and in which direction the tuple should propagate in the network. Tuples provide uncoupled interactions for applications and form a spatially distributed data structure. TOTA automatically handles the dynamic nature of the network, for example if a new neighbor becomes visible to a node, it will propagate suitable tuples (according to their propagation rule) to the new node. Similarly if a node disappears, tuples are evolved accordingly.

### 2.3. Swarm and modular robotics

Swarm robotics combine simple individual robots to be able to achieve a global goal that would be impossible for the individuals to reach alone. Search and rescue or environmental monitoring are two use cases usually associated with swarm robotics. It is usually desirable for the constituent robots to position themselves relative to the others in the physical space. These use cases can be handled efficiently with Proto[29] by calculating vector fields over the swarm and move the individual robots according to them.

Another kind of swarms are made of robots with limited positioning capabilities (only a gradient field is present or the robot can measure environmental phenomena). Graph Grammars[30] for example can be used to achieve the desired end configuration (assembly) of these robots[31]. The initial configuration can be an edgeless graph with the nodes representing the robots. From this configuration through successive application of transformation rules the desired end configuration can be reached, connections between robots can be made or removed.

Modular robotics is another interesting field for spatial computing as here robots are assembled from individual modules to the desired configuration or shape[32]. The problem essentially is the same as in other fields: there is a global specification of a desired end state and would like the individual modules to act cooperatively to reach that state. Motion generation of such assemblies can be solved as a spatial computing problem with Proto[33]. We know the relative positions of the modules and we can manipulate their actuators. The nature of the problem is a bit different than in the swarm robotics use case where the control of individual robots is needed, while here an assembly is the main focus and to essentially transform this assembly if needed.

## 2.4. Parallel and reconfigurable computing

High speed computation demands fast and efficient communication between the system components. Communication latencies between these computing devices (processors, GPUs) are strongly dependent on the physical distance between them. It is usually the job of the language to parallelize and efficiently distribute the computation to the available computing devices. One of the topological programming languages is Erlang[34], among others like MPI[35] or OpenMP[36]. The computation can be viewed as communicating processes, where communication is realized with asynchronous message passing.

Another approach is the Scalable Modeling System[37], which is a layered set of software and directives built on top of low level communications libraries such as MPI. The user can insert SMS directives into the supplied Fortran code and the system does the parallelization of the computation automatically: loop translations, array redeclarations, inter-process communication.

Cellular languages are also part of this group, examples include ALPACA[38], CELLANG[39], NetLogo[40]. They all describe some spatial structure of the cells forming the cellular automata and rules for evolution of those cells.

## 2.5. Amorphous computing

Another interesting example comes from the amorphous programming community. Amorphous computing imagines space as an amorphous medium[41], which has a computing device at every point. Information flows with a bounded velocity and a point in space can access the past states of neighboring points. This interesting medium can be approximated with more or less accuracy by discrete computing devices, which represent a part of the original manifold. Users can define their programs using a continuous space abstraction and with physical units, which has a number of advantages. For example physical units can help an application adapt to changes in the network topology: it does not matter if the number of devices doubles in an area or certain devices cease to exist. The virtual machines running on the nodes can maintain the continuous space abstraction and our programs (with more or less accuracy) can continue to operate regardless of the actual configuration of the network they are running on. The continuous space abstraction also means that the programmer no longer thinks on the level of individual devices, rather describes the desired behavior using fields, which can be selected using a number of different ways: physical distance, predicate based on some sensor value or node density.

## 2.6. The proposed spatial computing DSL

My choice of spatial programming system became Proto[33], because it provides the most general approach and it is very well suited to solve problems in a number of different domains: wireless mobile ad hoc and sensor networks, robotic swarms or biological systems. Proto is a high level functional programming language, which is used to specify the desired global behavior of the system. From this description the compiler generates device level code and runs the same code on every device in the network. The virtual machine run by the nodes maintains the spatial abstraction in the network, making it easy to handle new, disappearing or mobile nodes. There are four kinds of operators in the proto language and all of them operates on fields (in practice a set of nodes in the network): pointwise operators like addition, restriction operators that can manipulate the domain of the program, neighborhood operators for communication and feedback operators to establish state and specify it's evolution through time. Being a functional programming language it is really easy to extend the Proto system. One can create her own primitives and store them in a plugin.

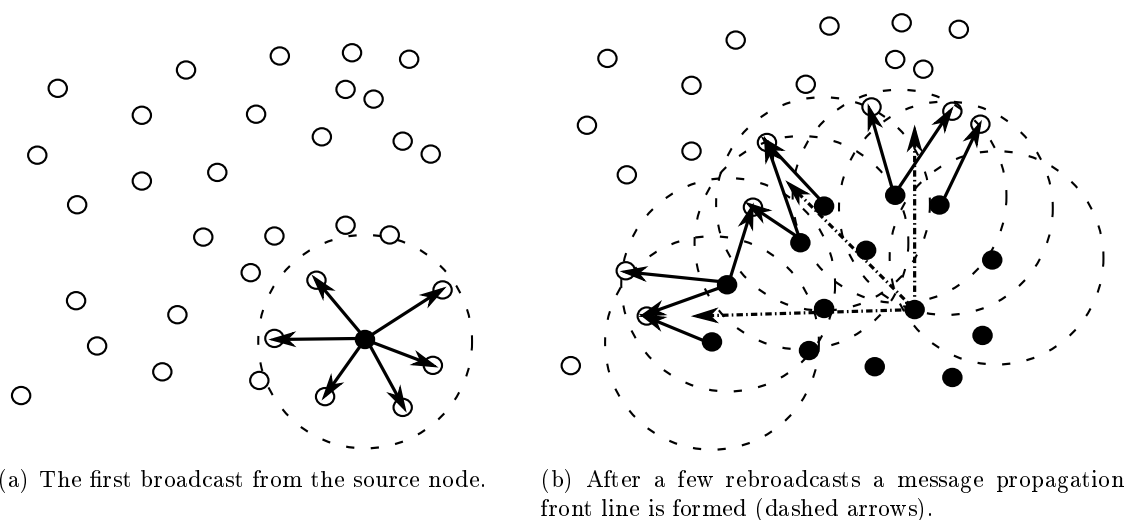
Communication in Proto is essentially a data dissemination problem. Right now no serious analysis has been done whether it can be optimized or how faulty nodes influence the accuracy of the approximation of continuous space in the system. The communication in Proto can be made more efficient by implementing the data dissemination algorithm, namely the Direction Based Handshake Gossiping, which will be introduced in Section 3.2 of my thesis.

## 3. Chapter

# The novel protocol

### 3.1. Motivation

Huge number of multihop broadcast protocols (introduced in Chapter 1) can be found in scientific literature, all targeting the goal to communicate effectively in this specific environment. Almost all of them are using signal messages (since it is the only way to get some information about the neighbor nodes) to reduce the enormous number of unnecessarily sent messages, to avoid packet collisions or just to improve the quality of the service in the network. However, it can be realized that the protocols' performance depends heavily on the gathered local information (which can be obtained from the signal messages, unfortunately producing more overhead).



**3.1. figure.** *The data dissemination process*

The current communication protocols for autonomous mobile networks do not enable the mobile nodes to choose their direction of message propagation by influencing the direction of the transmission using local interactions. The novel protocol presented in my thesis is based on the simple assumption, that all the messages circulating among the mobile nodes are originated from a given source node. In this scenario it would be desirable for the source

node to be able to manipulate the direction of the message propagation in the network, if needed (shown in Figure 3.1(a)).

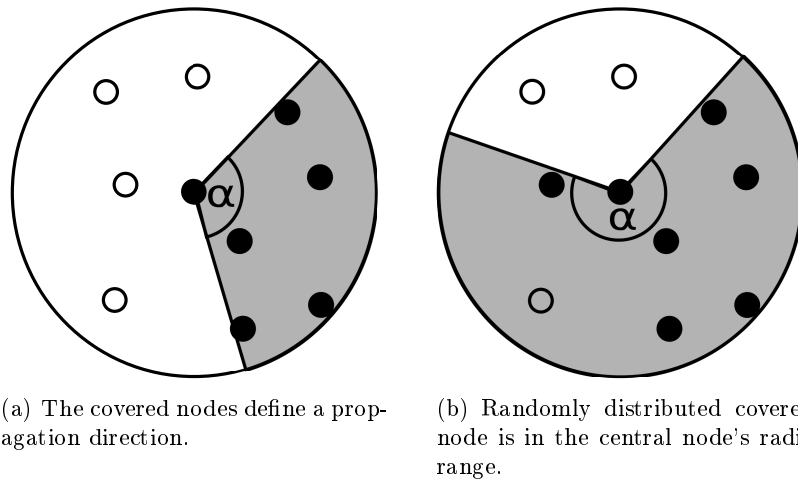
Thus the novel protocol's goal is to maintain the desired direction of the data dissemination, by using probability assignments to limit the nodes retransmitting the message. In a mobile self-organized environment there is no clear winner for information dissemination, forcing us to find a trade-off between efficiency (like number of the duplications) and simplicity (like spamming the messages all the time).

### 3.2. The Direction Based Handshake Gossiping

The contribution is to enable the rebroadcasting nodes (which become the new source nodes) to propagate their messages in a chosen direction, by giving a higher retransmission probability to nodes in that direction (see Figure 3.1(b)). With this kind of assignment of retransmission probabilities, the protocol is able to exclude many unnecessary duplications and force the messages to follow a given direction from the sender nodes, which can be very useful in various application cases (like if we want to cover only portions of the network or to control group of nodes moving in a given direction). Also, making nodes aware of their physical location is crucial to enable the network to implement more advanced communication primitives, such as transmitting a message in a given direction. This way the local interactions (probability assignments) will contribute to a global behavior of the network.

The Direction Based Handshake Gossiping protocol consists of two steps: the first one is the direction calculation, while the second one performs the probability assignment based on the first step's result. In the first step the algorithm tries to determine whether the covered nodes define a clear transmission direction, or they just want to uniformly distribute the messages in the network. In the latter case, the novel solution will use the probability assignment of an earlier developed protocol, the Valency Based Handshake Gossiping, while in the first scenario the direction vector will need to be calculated.

The initial sender node is given by its coordinates, together with its radio range, already covering a group of nodes in its 1-hop neighborhood. In the first step the goal is to calculate the smallest circular sector, containing all previously covered nodes (which do not need to receive the message again, as it will be a duplication). As the circular sector area is determined by its angle and the radius, and the latter is a constant value, only the angle will need to be calculated.



**3.2. figure.** *Node distributions taken into account by the adaptive mechanism in my solution.*

Firstly a reference vector should be defined, with which we can calculate the angle to

the covered nodes. Let this vector be the base vector  $\vec{i}(1,0)$ , therefore, the simplest way to calculate these angles will be the scalar product. Let  $v_i$  be the vector to the  $i^{th}$  covered node, then

$$\alpha_i = \begin{cases} \arccos(\frac{v_i \cdot i}{|v_i| |i|}) & , \text{when } y_i \geq 0 \\ 2\pi - \arccos(\frac{v_i \cdot i}{|v_i| |i|}) & , \text{else} \end{cases} \quad (3.1)$$

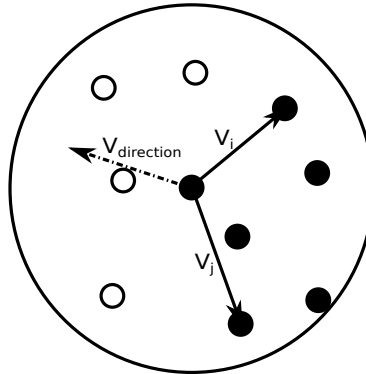
With this addition, the angle values will be in the  $[0; 360]$  domain, and not in  $[0; 180]$ , as this is essential to a proper calculation. Now the required angle can be determined with the help of the following statement.

**The sought circle sector will be given by adjacent vectors enclosing the smallest angle.**

The above stated finding can be proved by the use of two steps. The first one states that the sought circle sector is determined by two adjacent vectors. This step can be trivially justified: Two vectors divide the whole circle in two circle sectors and these vectors are adjacent if and only if the two sectors contain only covered or uncovered nodes. The second step says that from the set of these adjacent pairs, the desired solution would be the one, where the angle of the circle sector, where the already covered nodes are, is the smallest. The angles can be calculated by the differences of the vector's angle (with some caution, because the sequence plays a major role in this operation), which are defined previously.

After the first step,  $\alpha$  and the adjacent vectors are known, and based on these data we can proceed with the protocol's next step. My novel solution uses an adaptive mode of operation, therefore, if  $\alpha > \alpha_{threshold}$  then it uses the probability assignment of the Valency Based Handshake Gossiping protocol. In this case there is no distinctive transmission direction, therefore, it is better to use an existing broadcast protocol.

If  $\alpha \leq \alpha_{threshold}$ , there is an emphasised direction of transmission. This means that by proper selection of  $\alpha_{threshold}$ , one can tune the message propagation directions in the network. The direction vector can be determined from the circle sector in the following way:



**3.3. figure.** Calculation of the  $v_{direction}$  vector.

$$v'_{direction} = \begin{cases} \frac{-v_i}{|v_i|} - \frac{v_j}{|v_j|}, & \text{if } \alpha \leq \pi 180 \\ \frac{v_i}{|v_i|} + \frac{v_j}{|v_j|}, & \text{else} \end{cases} \quad (3.2)$$

$$v_{direction} = \frac{v'_{direction}}{|v'_{direction}|} \quad (3.3)$$

By utilizing this direction vector, the probability calculation is simple, the retransmission probability ( $p_i$ ) for a given uncovered mobile node will be the scalar product of  $v_{direction}$  and the vector pointing to the uncovered node divided by the length of the radio range (since it guaranties that the length of the vector will be less or equal to 1). Let  $v'_i$  point to node  $i$ , which is currently not covered, then

$$p_i = \max(v'_i * v_{direction}, 0) \quad (3.4)$$

If this equation is examined more thoroughly, it can be noticed that the variable depends also on the length of the vectors:

$$v'_i * v_{direction} = |v'_i| * |v_{direction}| * \cos(\theta) = |v'_i| * \cos(\theta) \quad (3.5)$$

So the probability assignment of my protocol is based not only on the transmission direction, but also on the distance of the nodes (like in DBHG).

### 3.3. Performance analysis

#### 3.3.1. The simulator

For measuring the performance of the protocols a self-organizing network simulator was implemented in C++, in which the protocols can operate under similar conditions as in a real MANET. The adjustable parameters of the simulator are introduced in Table 3.1

| Parameter name     | Description             |
|--------------------|-------------------------|
| N[pc]              | Number of the nodes     |
| R[e]               | Transmission range      |
| R'[e]              | Movement range          |
| A[e <sup>2</sup> ] | Simulation area         |
| Pid                | Simulated protocol's ID |
| t[pc]              | Number of tests         |

**3.1. table.** *The parameter list of the network simulator*

A modified version of the random waypoint mobility model[42] was used, where the waypoint should be chosen in the area of nodes' movement range. The simulator uses the previously introduced 3-way handshaking mechanism to avoid packet collisions and broadcast storms.

The simulator's default settings could be seen below in Table 3.2.

| Parameter name     | Value                 |
|--------------------|-----------------------|
| N[pc]              | 500,600,800           |
| R[e]               | 0,2e                  |
| R'[e]              | 0,1e                  |
| A[e <sup>2</sup> ] | 4e <sup>2</sup>       |
| Pid                | DAD, GEN, Ni's, DiBHG |
| t[pc]              | 100                   |

**3.2. table.** *Default settings of the network simulator*

When there is only one type of data message to be disseminated in the system, we could assume that all of the network devices need to receive it (e.g. in an emergency case). It can be seen from this table that simulations were run for 500, 600 and 800 nodes. The results were calculated based on averaging 100 tests to deal with variations from test to test.

### 3.3.2. The performance indicators

One of the most important performance indicators of data dissemination in a mobile ad hoc network is the **number of duplications**, as it effects the resource usage and efficiency of the whole network (speed of the data dissemination, energy consumption, coverage). The number of duplications specify how many unnecessarily received messages exist in the system (the nodes already own them, so resources are consumed to "spam" other nodes). It can be calculated as a difference between the requested and the received messages in the system:

$$dupl = \sum_{i=1}^N i_{message\_received} - \sum_{i=1}^N i_{message\_requested} \quad (3.6)$$

A high number of duplications in the system would mean inefficient utilization of the resources, so optimally this metric would be 0, meaning that all devices have received only the messages requested by them.

Another important performance metric is the **coverage**, the percentage of requested messages received by the nodes. Let  $message\_received'$ , be the number of messages received without duplications, then

$$coverage = \frac{\sum_{i=1}^N i_{message\_received'}}{\sum_{i=1}^N i_{message\_requested}} * 100\% \quad (3.7)$$

The third performance indicator is the **number of sent messages**. It shows how many broadcasts occurred in the network. It can be calculated as the sum of the number of broadcast transmissions of the nodes.

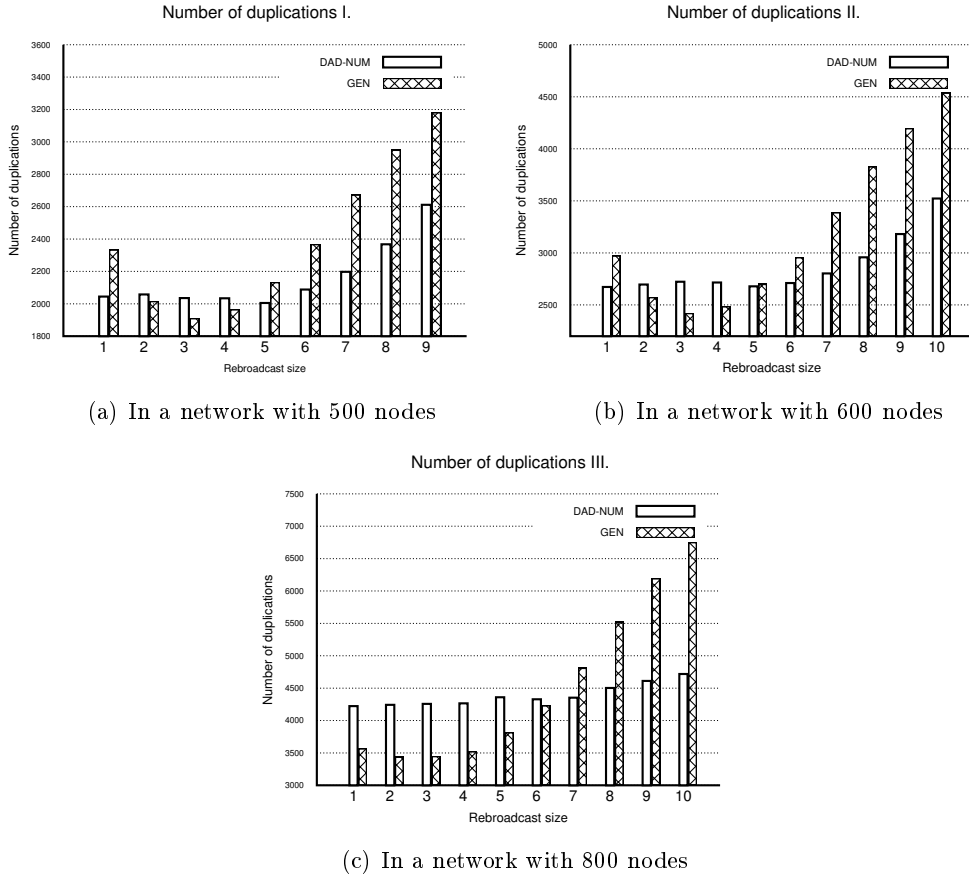
The fourth performance metric is the **coverage time**, which is the time needed for a given protocol to achieve a fixed level of coverage of the nodes. An idealistic protocol would achieve a high coverage in the shortest time possible, however, as it can be seen that the performance indicators are contradicting each other, so there will always be a trade-off between them. The „goodness” of a data dissemination protocol is always dependent on the application. Different protocols prioritize the performance metrics differently. For example, the blind flood protocol delivers the best coverage in the shortest time, but it is an extremely wasteful solution regarding resource usage, collisions and number of duplications in the system.

The protocols performance was evaluated by the help of the above described parameters fixing the coverage level at 95%. Thus, the simulation will stop when the specified protocol reaches 95% coverage.

### 3.3.3. Results

My novel protocol, the Direction Based Handshake Gossiping (DiBHG) was compared to the Distance Adaptive Dissemination (DAD) [16], the General Probabilistic Broadcast algorithm [9], and to Ni et al.'s algorithm, which were all described in Section 1.1. (The previous 4 protocols, the DBHG, VBHG, AVBHG and the AHG, have been already presented in [22]) These protocols try to optimize the resource usage in the system not only by using only locally available information about their neighbors, but also by utilizing available spatial information. Performance measurements of the described protocols were published in several papers, proving that they disseminate information in self-organized mobile networks very efficiently, overperforming most of the available data dissemination algorithms, that can be found in the literature.

All three reference protocols in my mobile self-organized network simulator, and the values of the  $k$  parameter (the target rebroadcast size) have been determined as well, for which the Distance Adaptive Dissemination (DAD-NUM) and the General Probabilistic Broadcast Protocol (GEN) performs the best, regarding the number of duplications, by simulations. Choosing the optimal  $k$  value for further performance evaluations ensures a fair comparison between the tested protocols. The number of duplications for the best  $k$  values were measured for all three mobile node densities and summarized in Table 3.3.



**3.4. figure.** Duplication overhead of the DAD-NUM and GEN protocols

Figure 3.4 shows that for all three different node density scenarios, the GEN protocol over-performed the DAD-NUM solution regarding the number of duplications only for small  $k$  values. By increasing the  $k$  parameter, the DAD-NUM always gives a better result than the GEN protocol in terms of number of duplications. The best  $k$  parameter values for all three scenarios have been selected and summarized in Table 3.3.

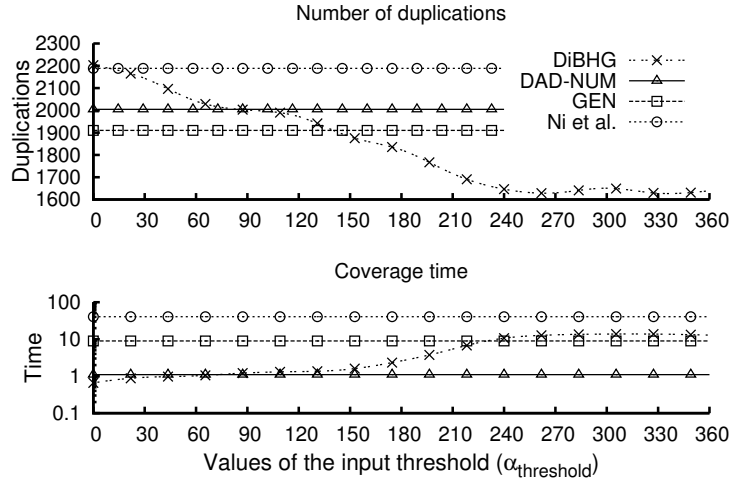
| Number of nodes | Protocols | Rebroadcast sizes | Number of duplications |
|-----------------|-----------|-------------------|------------------------|
| 500             | DAD-NUM   | 5                 | 2004.71                |
|                 | GEN       | 3                 | 1910.29                |
| 600             | DAD-NUM   | 1                 | 2672.82                |
|                 | GEN       | 3                 | 2420.84                |
| 800             | DAD-NUM   | 1                 | 4222.48                |
|                 | GEN       | 2                 | 3438.40                |

**3.3. table.** *The selected best  $k$  values for DAD-NUM and GEN protocols.*

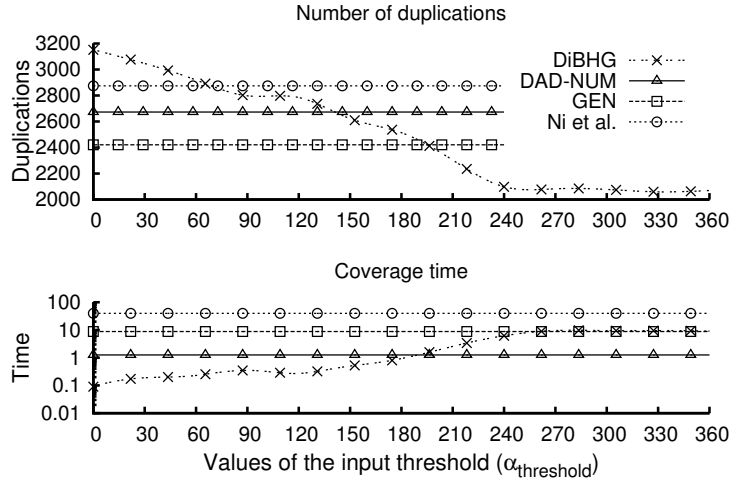
Ni et al’s algorithm does not require any additional input parameters (like the  $k$  parameter for DAD-NUM and GEN), therefore, it can be compared more easily with the Direction Based Handshake Gossiping.

In the DiBHG protocol, the input parameter is the angle,  $\alpha_{threshold}$ , with which we can adjust the propagation direction. As described earlier in Section 3.2, if this parameter is set to a smaller value, the probability assignment of the Valency Based Handshake Gossiping protocol will be used in the majority of the cases instead of the direction based mechanism. Otherwise, the direction based solution will be the dominant one.

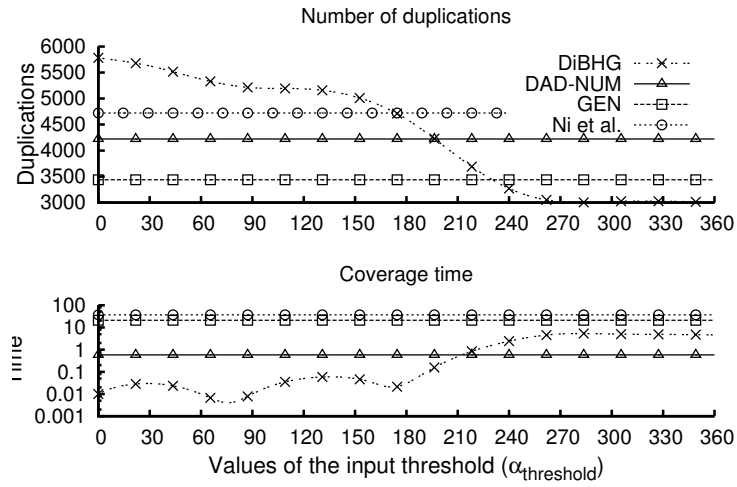
The two performance metrics deemed most important, are the number of duplications and coverage time. The two metrics contradict each other, as the solutions can reduce the coverage time only by the cost of having more duplications, and vice versa. For example, the blind flood has the shortest coverage time, as every node rebroadcasts every received message without any backoff time, but this way the number of duplications will explode. Therefore, a compromise and trade-off should be found between the two.



(a) In a network with 500 nodes



(b) In a network with 600 nodes

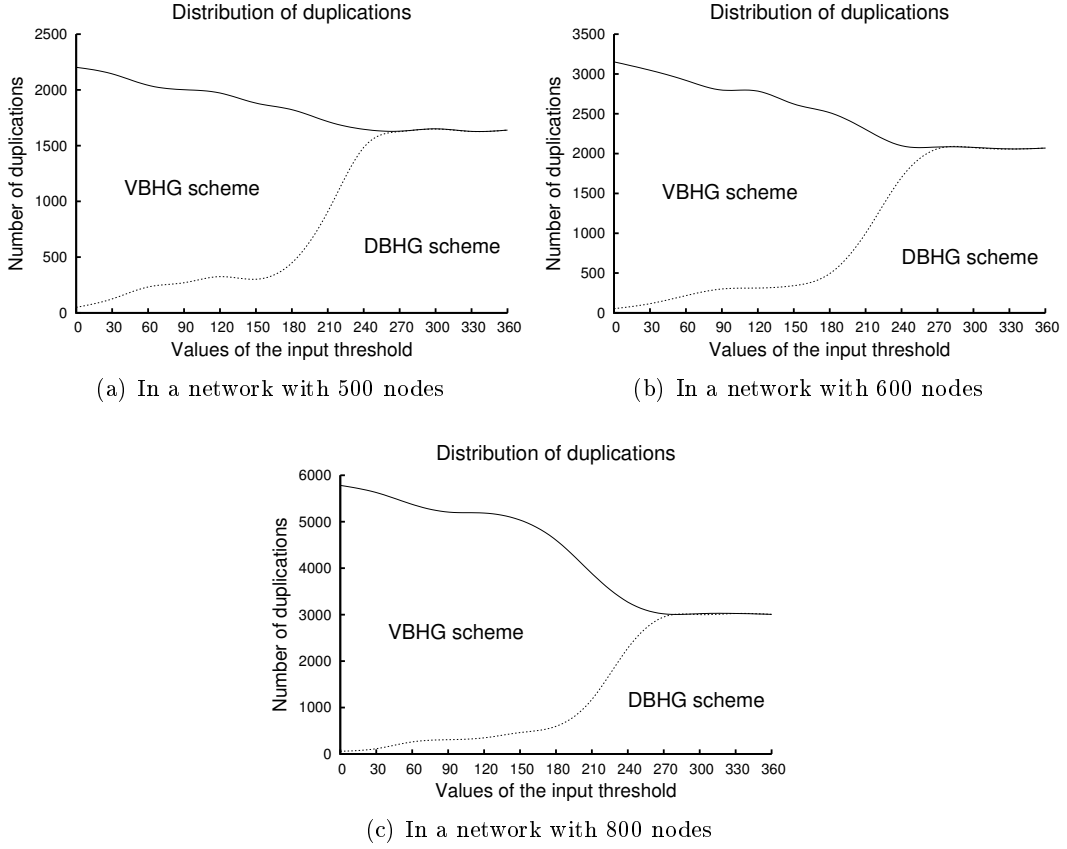


(c) In a network with 800 nodes

3.5. figure. Results for the 4 protocols in different scenarios

When testing the novel solution, we could focus on these two metrics to verify if it can hold a reasonable trade-off compared to the existing protocols. The results for number of duplicated messages and coverage times for 500 nodes can be seen in Figure 3.5(a) for the Direction Based Handshake Gossiping, in the function of the  $\alpha_{threshold}$  input parameter. For easier comparison, the number of duplications and coverage times for the other three protocols are placed in the same figure, measuring them with the previously determined  $k$  parameters (listed in Table 3.3). As expected, for small  $\alpha_{threshold}$  values the number of duplications for the Direction Based Handshake Gossiping is high, as a small  $\alpha_{threshold}$  value means that only a small fraction of the neighboring nodes are covered, which results in a broad angle of data dissemination. In the majority of these cases, (where the  $\alpha_{threshold}$  is small), the retransmission probability assignment of the Valency Based Handshake Gossiping is utilized, without a dominant dissemination direction. As it can be seen, for even small values of the  $\alpha_{threshold}$ , when running simulations with 500 mobile nodes, the DiBHG overperforms the Ni et al. algorithm in terms of number of duplications, and it is performing better than the DAD-NUM protocol above  $\alpha_{threshold} = 80$ , and better than the GEN protocol starting from  $\alpha_{threshold} = 140$ . The figure shows that the Direction Based Handshake Gossiping can overperform the other three protocols in terms of number of duplications, if the input parameter is chosen wisely. However, coverage time should also be considered. It can be seen, that at the point it overperforms the DAD-NUM protocol in terms of duplications, it will be less effective in terms of coverage time. That means, that the DAD-NUM will cover the nodes faster, but will cause more duplicates. When compared to the GEN protocol, it is less effective in terms of coverage time from around  $\alpha_{threshold} = 230$ , while in terms of duplications it is already better from  $\alpha_{threshold} = 140$ , which means that there is a wide domain of the input parameter for which the novel protocol performs better than the GEN for both properties. When compared with the Ni et al. algorithm, it over-performs it for all of the values of the  $\alpha_{threshold}$  parameter, either in number of duplications, or coverage time. The Direction Based Handshake Gossiping is even more convincing for scenarios where 600, or 800 mobile nodes are simulated. If the  $\alpha_{threshold}$  is tuned well, it can significantly reduce the number of duplications, and in the meantime keep the coverage time at the same level.

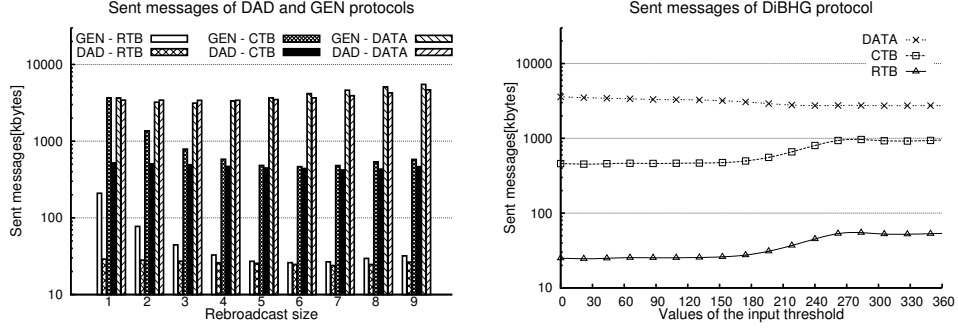
As stated before, my solution is an adaptive scheme, therefore if  $\alpha > \alpha_{threshold}$ , it uses the probability assignment determined by the Valency Based Handshake Gossiping protocol. Otherwise, if there is an emphasised direction of the transmission, the Direction Based Handshake Gossiping scheme is utilized. It is interesting to check, how the selection of the  $\alpha_{threshold}$  input parameter affects the retransmission probability assignment. Figure 3.6 shows us the distribution of the two schemes, when observing the number of duplications. As it can be seen for all three mobile node densities, when setting small  $\alpha_{threshold}$  values (under 150) the Valency Based Handshake Gossiping scheme prevails, which will cause plenty of duplications, as there is no distinguished direction of the transmission. This is the range of the  $\alpha$  parameter, where my solution is not performing so well in comparison to the other three reference protocols (Figure 3.5). By increasing the value of the input parameter (which means we define a clear direction of the transmission), the probability



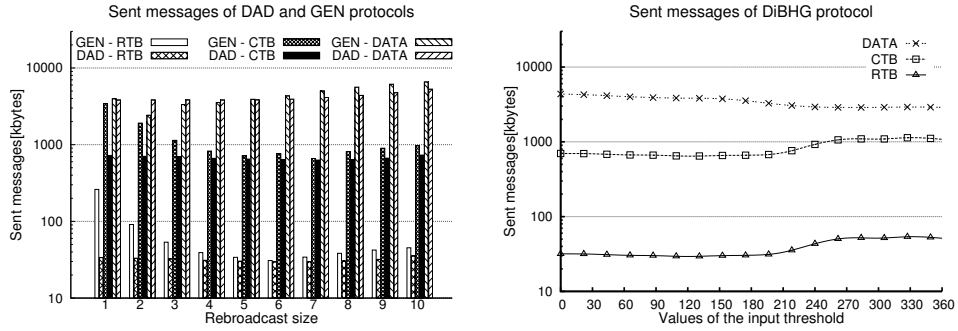
**3.6. figure.** Duplication distribution of the VBHG and DiBHG probability assignments.

assignment of the Direction Based Handshake Gossiping will be dominant, and after reaching  $\alpha_{threshold} \approx 270$ , the adaptive scheme utilizes solely the Direction Based Handshake Gossiping retransmission probability assignment. From this point on, the number of duplications drop drastically, this is the period when the new solution significantly outperforms the reference protocols.

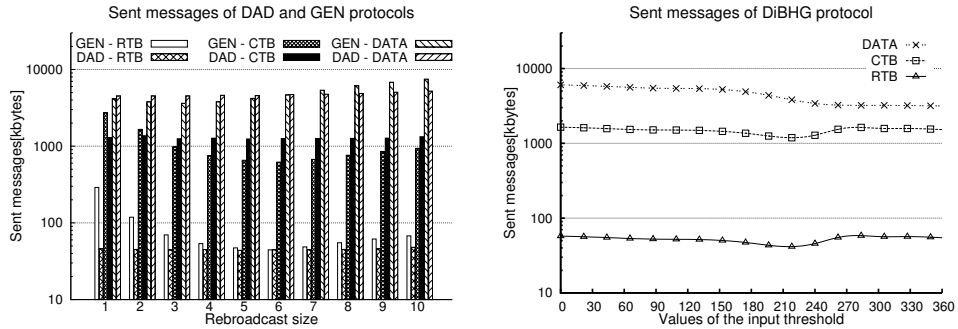
Another interesting (and often overlooked) performance characteristic to investigate is the amount of control overhead produced to achieve these results. It is important to note, that all of the measured protocols are equipped with the 3-way handshaking transmission mechanism, as without this mechanism the reference protocols would produce many more duplications, therefore, a fair comparison would not be possible. The 3-way handshake uses RTB (Ready To Broadcast) and CTB (Clear to Broadcast) control messages to coordinate transmissions locally, and to avoid broadcast storms and unnecessary message transmissions. The concrete algorithm of the 3-way handshake along with measurement results are described in previous chapter(Sec...). As depicted in Figure 3.7, all measured protocols produce around the same amount of RTB and CTB control messages, CTB being the dominant one, compared to the amount of RTBs produced (the source node sends only one RTB message, but all neighboring nodes send back CTB messages). More interesting is the ratio of the data messages and the control messages, also referred to as control overhead. When observing the DiBHG for the small values of  $\alpha_{threshold}$  the overall amount



(a) In a network with 500 nodes



(b) In a network with 600 nodes



(c) In a network with 800 nodes

**3.7. figure.** *Distribution of data and control messages of the measured protocols.*

of control messages (RTB+CTB) is less than 5% compared to the data messages. As the  $\alpha_{threshold}$  increases, the ratio of the control overhead and the data traffic grows (even up to 10% in some cases), however, in these cases the amount of the data messages will decrease, which means the algorithm need less sent messages to obtain the same coverage. In these simulations the size of the data messages were fixed to 50 kbytes (which can be transferred in many packets), while the control messages were less then 100 bytes. We are well aware that the 3-way handshake mechanism is only effective, when using it for the propagation of data messages, which are in average larger than 20-30 kbytes. If the average data message is less than this size, the 3-way handshake mechanism should not be used, because of the inefficient control overhead it would produce.

## 4. Chapter

# The novel solution in spatial computing environments

### 4.1. Overview

As detailed in the previous chapter, numerous implementations and solutions can be found in scientific literature which aim to implement the spatial computing approaches. From these the Proto spatial computing system (originally developed at MIT) was chosen, since on it is a general solution with a clear structure and a general functional programming language. Proto is very well suited to applications running in mobile ad-hoc and wireless sensor networks.

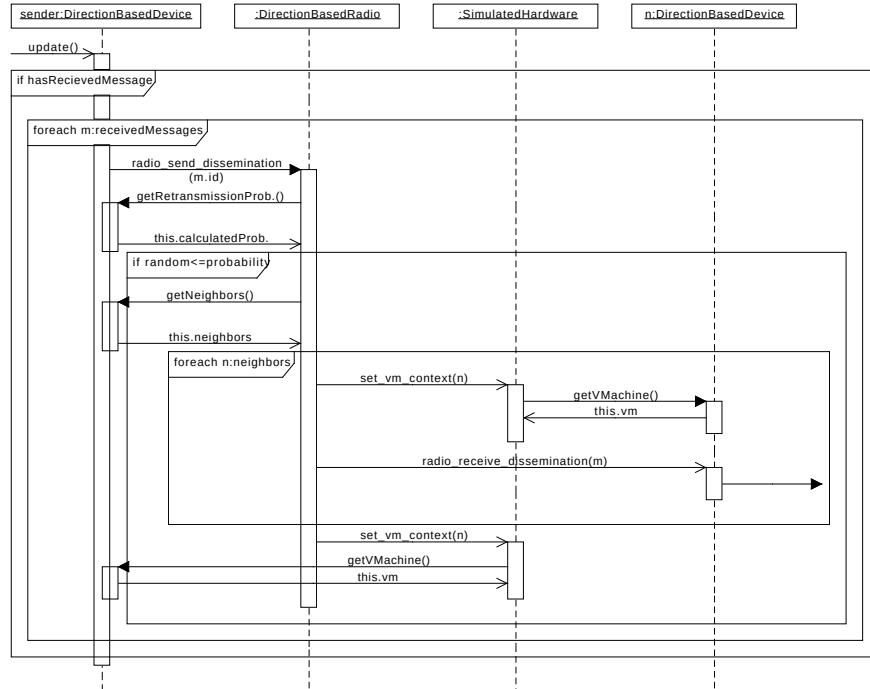
In the current Proto system there is no sophisticated multihop information dissemination implementation. Proto nodes share their state periodically, which means that they aggregate the states of their neighbors, do their part of the computation, and in the end broadcast their state to their neighbors. This method can be used to disseminate information in the network, but it is not effective. Disseminated information in our terminology is static information, therefore the state sharing approach of Proto (while great for continuous computation and variable data) is essentially a blind flood algorithm from this viewpoint. It is crucial to improve the system by extending it with a new primitive, being able to disseminate static information in the system effectively.

Therefore, the goal of this chapter is to present the implementation of my location based information dissemination protocol (the Direction Based Handshake Gossiping) in the Proto spatial computing system, and verify the performance of the implemented code by comparing it to the initial Proto solution. The Proto nodes and environment essentially form a wireless sensor network, therefore, it is not needed to do the detailed performance evaluation as in the previous chapters. Proto has a layered architecture and is easily extensible with new language primitives. Since there is no information dissemination language primitive in the system, like the directional dissemination, therefore, first we need to design a layer above the proto kernel, which realizes the transmission mechanism of the DiBHG protocol. This layer will contain the dissemination method that the simulator can reach from the high level proto code.

## 4.2. The dissemination plugin

The Proto system has a layered architecture, that makes it easy to extend the system's functionality. Given this architecture the developer can extend one part of the system without having to modify other parts of it. For each implemented layer there is a device wrapper class in the Proto system, which enables the developer to store information and add new functionality to the device without modifying existing pieces of the code.

I have implemented the so called "DirectionbasedPlugin" layer, which can be switched on-off before running a Proto application. There are two main classes in this plug-in, the DirectionBasedDevice and the DirectionBasedRadio. The first one, as it was mentioned, is a wrapper class for nodes in the simulation. The DirectionBasedDevice class contains the implementation specific inner state of a node such as the node's retransmission probability, identifiers of the received messages, the number of duplications etc. Thus, a communicating node in a spatial computing environment can be defined as an instance of the DirectionBasedDevice class. The DirectionBasedRadio class is based on the existing communication layer implementation, "UnitDiscRadio", which provides a simple wireless communication solution. At any given time, the context of the Proto simulator is on one device in the system. All functions executed manipulate that exact device. Therefore the broadcast of a message consists of changing the context to the receiver device, calling the global *receive\_message* method, then changing the context to the next receiver, and so on. In the end the context is restored to the sender node and the broadcast ends. Because of this, there are two dissemination methods in the plugin, one for handling the sending logic, and one for receiving part. This behavior is depicted in Figure 4.1.



4.1. figure. Context switching in the Proto system

The Proto simulator is a discrete event simulator, and the system provides certain callbacks to the nodes for different events (before and after them as well). One of these callbacks is called "update", and it is called at a fixed interval. I have used this callback to time the broadcasts from the nodes. There can be more sophisticated implementations as well, using random backoff or random assessment delays, but for a proof of concept approach this is suitable. As it can be seen on the flowchart, the global context of the simulator is changed for each event executed in the discrete event simulator. This global context contains the machine, which "executes" the event. In the "update" callback it is first checked if there are received messages in this node not forwarded yet. If there are, a retransmission probability should be allocated for them according to the chosen implementation (Blind Flood or DiBHG in our case). Once the retransmission probability is available, the message is sent with the defined probability to each neighbor in its radio range. For this to happen, its neighbors are enumerated (collected via state change communication mentioned before) and for each of them the simulator context is changed and the receive method is called.

### 4.3. The Proto code

In the previous section the Proto extension layer was introduced, therefore, the Proto code description follows, which triggers the dissemination protocol. With the help of the programming language we can specify the desired global behavior of the system. From this description the compiler generates a per device code and runs the same code on every device in the network as it was described in 2.6.

There are lot of primitives already implemented in the core language, which can be used in my code as well. First, we would like to start the dissemination from a source which is selected by a user. It can be done by using the *sense* operator:

```
(if (sense 1) (disseminate 180) 0)
```

As it can be seen, the proto language is a lisp-like language, which means that the programmer need to enclose the operators in brackets and it uses Polish notation. In this short code there are 2 operators and a logical statement. The first one is (*if* < *expr* > < *whenitistrue* > < *else* >). So now we can understand that if (*sense* 1) expression returns with *true* the (*disseminate* 180) expression will be run else nothing happens (since there is a zero, which means nothing). In the test expression of the *if* statement we use the *sense* operator. There are three LEDs on each device, which can be turned on-off in the simulator (by clicking on it and pressing either the 't' or 'r' or 'u' buttons), and each LED represents an integer value. These LEDs are sensors from the point of view of the programming language as one can test, whether they are on or off. Thus, the *sense* operator can compare the state of the LED to an input parameter and if these two are equal it returns *true*, else it returns *false* (e.g. the devices with blue LED can be gathered by (*sense* 2)). The last operator is my dissemination language primitive linked to the previously described implementation. This operator needs only one input parameter, which is  $\alpha_{threshold}$ , the required input parameter for the DiBHG protocol. Now the main part of the Proto code is already completed thus a dissemination can be started by selecting a device and pressing the 't' button in the simulator. By running this Proto code, we could observe that the nodes are static, since there are no instructions implementing mobility. So the code should be completed with the mobility implementation or this version can be used for modeling static sensor networks. Therefore, the complete version of the code is shown below.

```
(def disseminate_func (angle)
  (let ((mobile (once (< (rnd 0 1) 0.5))))
    (mov (* (if mobile 0.01 0) (dither))))
  (if (sense 1) (disseminate angle) 0)
)
```

There are some currently unreviewed keywords in this version. In the first line there is an expression beginning with (*def*). This shows how to define a function with the name *disseminate\_func* and one argument, *angle*. In this case a single argument should be given, since the DiBHG protocol requires only the  $\alpha_{threshold}$  (thus the angle threshold parameter is given directly to the *disseminate* primitive). The next two lines are responsible for moving the nodes according to a simple random waypoint mobility model. The

(*rnd*) is a function that generates a random number between its two arguments. In this case, it picks a random number between 0 and 1, then the next is the (< comparator that compares the first number to the second and returns with a boolean value. The (*once*) primitive computes the expression after it only once and then remembers it. The last one in this line is (*let*), which allows for local variable bindings exactly like in LISP. On the first line (after the definition) a random variable (*mobile*) is calculated (only once) on each device. If there are infinite devices in the simulation, it can be said that half the devices have true values and the other half have false values in the *mobile* parameter (because of the 0.5 value). In the next line there are two previously undescribed core primitives (and a multiplication). (*Dither*) creates a random velocity vector and it gets multiplied with 0.01 if the node is mobile and with 0 if not. *Mov* simply moves the device (it is an actuator) in the specified direction with the specified speed (which depends on the size of the given vector).

In this section a Proto function was introduced, demonstrating the ease of the Proto programming language, as it takes only few lines to make an appropriate mobile self-organized network scenario. Moreover the properties of the network such as moving speed and the number of the moving devices are tunable as well. Thus, this simulation environment can be used by invoking the defined function like this: "(*dissemination\_func angle*)".

## 4.4. Performance analysis

### 4.4.1. The simulator settings and the performance indicators

To measure the performance of the implemented DiBHG in Proto, first of all the input parameters (e.g. number of nodes, movement velocity ...) should be declared. I use the same parameters as in my self-organizing network simulator, because the aim of this section is to verify the previously described characteristics of the DiBHG protocol's Proto implementation. Let the number of the nodes be 500, 600 and 800 and the transmission range for the radio is 20 (since the simulation area size is 100x100). Because all of the nodes were moving in the previous simulations, the 0.5 value (from the Proto code) should be changed to 1.0. Finally, two different dissemination protocols will be compared in the Proto simulator: Blind Flood and the DiBHG algorithms. The settings of the Proto simulator could be seen below in Table 4.1.

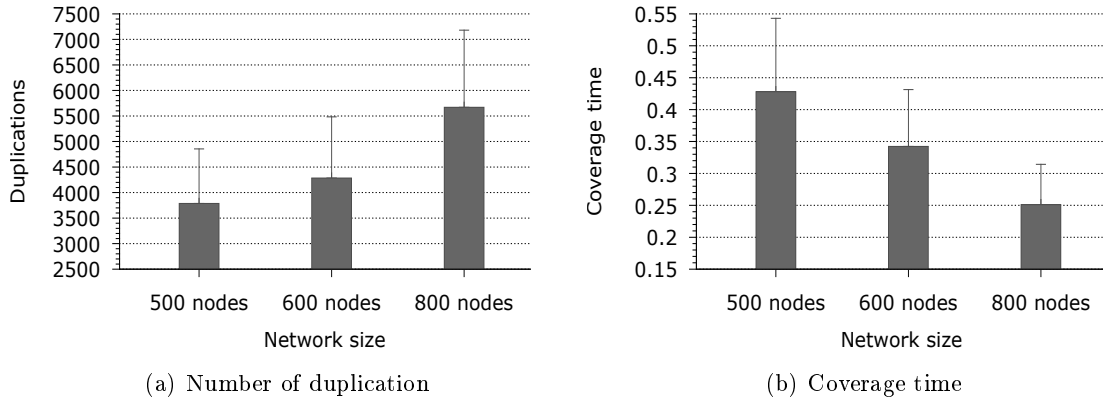
| Parameter name | Description             | Value       |
|----------------|-------------------------|-------------|
| -n[pc]         | Number of the devices   | 500,600,800 |
| -r[e]          | Transmission range      | 20          |
| Pid            | Simulated protocol's ID | BF, DiBHG   |
| t[pc]          | Number of tests         | 100         |

**4.1. table.** *The Proto simulator parameter list*

The performance indicators are the same as they have been presented in Section 3.3.1. Simulation will be terminated, when the specified protocol reaches 95% coverage of the nodes. The threshold of 95% has been chosen, since there could be few nodes which moved far from others, so the results would be significantly distorted by them.

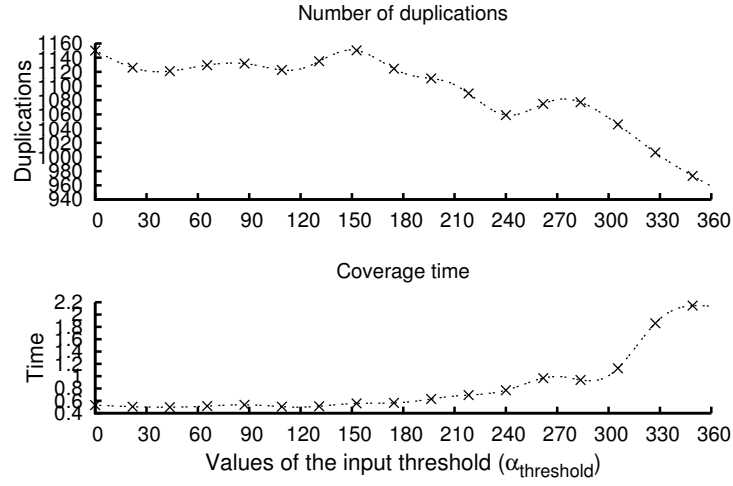
#### 4.4.2. Results

The two indicators which deemed most important in this simulation are the number of duplications and the coverage time (same as in the previous chapter). First, the performance of the Blind Flood protocol will be examined as a reference or base line result. After that, the DiBHG algorithm will be reviewed to verify that the performance characteristics match those presented in the detailed performance evaluation.

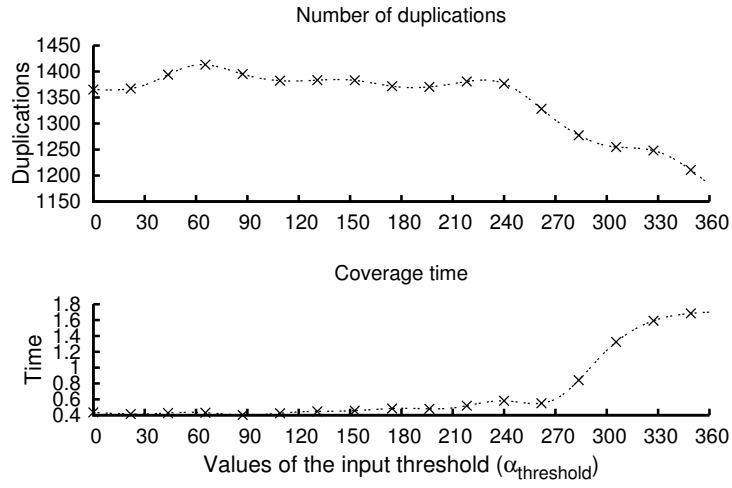


**4.2. figure.** *The Blind Flood protocol's performance in the Proto simulator*

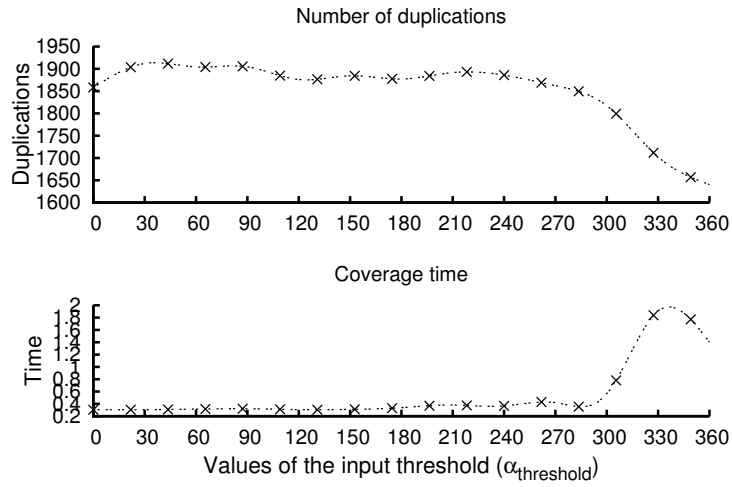
The results for Blind Flood are depicted in Figure 4.2. It can be seen, that on one hand the Blind Flood is extremely wasteful, since it produces over 3000 duplications in each scenario (in a network with 800 nodes it could be up to 5750). On the other hand, however, it gives a really fast coverage. These results are quite intuitive, as Blind Flood rebroadcasts every received message immediately, therefore, wasting resources, but also chooses the shortest path to each node in the network, in the process it tries all the paths. If we compare the two subfigures, it could be noticed that the two indicators are contradicting to each other. If there are more nodes in the network (on the same simulation area), there will be more connections between them and there is a greater chance, that a broadcast will generate more duplications, and for the same reason the messages can go via multiple paths, which enables for faster dissemination. From the results it can be concluded that for using this spatial computing environment in a real life scenario, the communication algorithm should be changed, since it is extremely wasteful.



(a) In a network with 500 nodes



(b) In a network with 600 nodes



(c) In a network with 800 nodes

4.3. figure. The DiBHG protocol's performance in the Proto simulator

The results for the DiBHG algorithm can be seen above in Figure 4.3. The same observations can be made as in Section 3.3.3. First it can be said that DiBHG outperforms Blind Flood, because it generates up to 4000 duplications less (when comparing the 800 nodes scenarios). However, the shape of the curves are not exactly identical (because the results were generated by two different simulators, based on different approaches), but it can be concluded, that for small  $\alpha_{threshold}$  values, the number of duplications is higher then for a bigger threshold. The same trade-off between efficiency and speed can be identified as in section 3.3.3. While the number of duplications can be reduced by increasing the threshold (increasing the ratio between the DiBHG and VBHG probability assignment), the coverage time increases. The proper threshold should be chosen according to the application scenario (speed vs. resources).

In this chapter I have described the implementation of my DiBHG protocol in the Proto spatial computing system. A new communication primitive was developed, along with a language primitive to trigger the dissemination process. These two components are packed in a Proto plugin, which can be included when starting a Proto application (either in the simulator or on real hardware). No changes were made to core Proto source code, which ensures compatibility with other plugins and makes it easier to make this functionality available in future Proto versions as only the plugin's code needs to be updated. From the simulation results it can be observed, that the implementation of the DiBHG protocol was successful, the same performance characteristics can be identified in as in the measurements carried out in 3.3.3. With the presented protocol, we can achieve a more energy efficient functioning of the system when trying to disseminate static information in the network. Implementing the DiBHG protocol in Proto and making it available in an easy to use Proto plugin takes it one step further, to enable Proto become a real life spatial computing system and move it away from "simulator only" use.

## 5. Chapter

# Conclusion

The clear majority of the data dissemination protocols for self-organized networks do not use spatial information of the network nodes to optimize the bandwidth and channel usage. Even the communication protocols which utilize spatial properties of the network are not capable of enabling the mobile nodes to contribute with their local message retransmissions to a global message propagation direction, this way empowering the mobile nodes to be aware of the global targets. A novel communication protocol has been designed for autonomous mobile systems, the Direction Based Handshake Gossiping, which was implemented in my self-organizing network simulator, together with three other location based data dissemination protocols from the literature (Distance Adaptive Dissemination, the General Probabilistic Broadcast Algorithm, and Ni et al's location-based scheme) and in the Proto spatial computing simulator as well. The novel solution enables the nodes to propagate their messages in a chosen direction, by giving a higher retransmission probability to nodes in that direction. With this kind of retransmission probability assignment, it is possible to avoid many unnecessary duplications and attempt to follow a given direction from the sender nodes. This can be a very useful trait in various application cases, for example covering only portions of the network or controlling a group of nodes moving in a given direction.

The simulation results show that the Direction Based Handshake Gossiping can overperform the other three protocols in terms of the number of duplications, if the input parameter is chosen correctly. If this  $\alpha_{threshold}$  is tuned well, it can significantly reduce the number of duplications, while keeping coverage time the same, and even improving it in some cases. Another important performance indicator we measured is the control overhead, the ratio of data and control messages. When observing the DiBHG for the small values of  $\alpha_{threshold}$  the overall amount of control messages (RTB and CTB) is less than 5% of the data messages. As  $\alpha_{threshold}$  increases, the control overhead also increases up to 10% in some cases, while the amount of data messages decrease, which means that our protocol can reach the same level of coverage with less data messages sent.

The novel protocol has been implemented in the Proto spatial computing simulator, developed initially by MIT, as well to show that the current dissemination primitive should be replaced with the presented novel protocol. Since the Blind Flood protocol could make

the whole environment unsuited for a real life application as it wastes too much energy (which can be a scarce resource in a mobile self-organized network). It was demonstrated that the DiBHG protocol can work under different applications (such as spatial computing approaches), because it performs nearly the same characteristics.

It can be concluded that the adaptive version of the DiBHG (combined with the probability assignment of the Valency Based Handshake Gossiping) can be an effective data dissemination scheme for mobile self-organized networks and for spatial computing environments as well. Making the nodes aware of their physical locations I was able to implement a directed transmission communication primitive, which can be very useful in a variety of use cases.

# Acknowledgement

## **In English**

First I would like to thank my supervisor (Vilmos Dr. Simon) for the patient guidance and advice he has provided throughout my works (which includes many articles, paper on Scientific Students' Associations Conference and this thesis as well) as well as for all the extra work which is caused by that this thesis was written in English, and that I spent the last academic semester in London.

I would also like to thank Wiandt Bernát, who helped revising the paper. He gave me a lot of useful comments and remarks which were helped to raise the standard of the paper. His help was essential for the proper completion of this thesis.

---

## **In Hungarian**

Szeretném elsősorban megköszönni konzulensem (Dr. Simon Vilmos) segítségét. Egyrésztől végig a közös munka során adott segítségeket, tanácsokat (melynek során számos cikk, TDK dolgozat és ez a diplomamunka is elkészült) továbbá mind azt a többletmunkát amelyet azzal okoztam, hogy angolul készült el a dolgozat, illetve hogy az utolsó félévemet Londonban töltöttem.

Továbbá szeretném megköszönni Wiandt Bernát kutatótársam munkáját is, aki a dolgozat lektorálásában segített. Nagyon sok hasznos észrevételeket, megjegyzéseket tett, amelyek segítettek a dolgozat színvonalának emelését. Segítsége elengedhetetlen volt a diplomamunka megfelelő elkészítéséhez.

# List of figures

|                                                                                                 |    |
|-------------------------------------------------------------------------------------------------|----|
| 1.1. The first phase . . . . .                                                                  | 12 |
| 1.2. The second phase . . . . .                                                                 | 13 |
| 1.3. The third phase . . . . .                                                                  | 14 |
| 3.1. The data dissemination process . . . . .                                                   | 26 |
| 3.2. Node distributions taken into account by the adaptive mechanism in my<br>solution. . . . . | 28 |
| 3.3. Calculation of the $v_{direction}$ vector. . . . .                                         | 29 |
| 3.4. Duplication overhead of the DAD-NUM and GEN protocols . . . . .                            | 33 |
| 3.5. Results for the 4 protocols in different scenarios . . . . .                               | 35 |
| 3.6. Duplication distribution of the VBHG and DiBHG probability assignments.                    | 37 |
| 3.7. Distribution of data and control messages of the measured protocols. . . . .               | 38 |
| 4.1. Context switching in the Proto system . . . . .                                            | 40 |
| 4.2. The Blind Flood protocol's performance in the Proto simulator . . . . .                    | 45 |
| 4.3. The DiBHG protocol's performance in the Proto simulator . . . . .                          | 46 |

# List of tables

|                                                                        |    |
|------------------------------------------------------------------------|----|
| 2.1. Categorization of spatial computing DSLs . . . . .                | 19 |
| 3.1. The parameter list of the network simulator . . . . .             | 31 |
| 3.2. Default settings of the network simulator . . . . .               | 31 |
| 3.3. The selected best k values for DAD-NUM and GEN protocols. . . . . | 34 |
| 4.1. The Proto simulator parameter list . . . . .                      | 44 |

# References

- [1] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic. *Mobile ad hoc networking*. Wiley-IEEE press, 2004.
- [2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *ACM Sigplan Notices*, volume 37, pages 96–107. ACM, 2002.
- [3] J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proc. ieee infocom*, volume 6, pages 1–11. Barcelona, Spain, 2006.
- [4] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *Communications Magazine, IEEE*, 41(6):128–136, 2003.
- [5] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM, 2003.
- [6] J. Beal, S. Dulman, K. Usbeck, M. Viroli, and N. Correll. Organizing the aggregate: Languages for spatial computing. *arXiv preprint arXiv:1202.5509*, 2012.
- [7] O.K. Tonguz, N. Wisitpongphan, J.S. Parikh, F. Bai, P. Mudalige, and V.K. Sadekar. On the broadcast storm problem in ad hoc wireless networks. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–11. IEEE, 2006.
- [8] A. Al Hanbali, M. Ibrahim, V. Simon, E. Varga, and I. Carreras. A survey of message diffusion protocols in mobile ad hoc networks. In *Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools*, page 82. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
- [9] S.Y. Ni, Y.C. Tseng, Y.S. Chen, and J.P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162. ACM, 1999.

- [10] Y.C. Tseng, S.Y. Ni, and E.Y. Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *Computers, IEEE Transactions on*, 52(5):545–557, 2003.
- [11] J. Cartigny, D. Simplot, and J. Carle. Stochastic flooding broadcast protocols in mobile wireless networks. *LIFL Univ. Lille*, 1, 2002.
- [12] A. Khelil, P.J. Marrón, C. Becker, and K. Rothermel. Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. *Ad Hoc Networks*, 5(5):531–546, 2007.
- [13] W. Peng and X.C. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 129–130. IEEE Press, 2000.
- [14] E. Varga, T. Csvarics, L. Bacsardi, M. Berces, V. Simon, and S. Szabo. Novel information dissemination solutions in biologically inspired networks. In *Telecommunications, 2007. ConTel 2007. 9th International Conference on*, pages 279–284. IEEE, 2007.
- [15] W.R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185. ACM, 1999.
- [16] X. Chen, M. Faloutsos, and S. Krishnamurthy. Distance adaptive (dad) broadcasting for ad hoc networks. In *MILCOM 2002. Proceedings*, volume 2, pages 879–883. IEEE, 2002.
- [17] V.K. Paruchuri, A. Durresi, D.S. Dash, and R. Jain. Optimal flooding protocol for routing in ad-hoc networks. In *IEEE Wireless Communications and Networking Conference*, 2003.
- [18] I. Stojmenovic and X. Lin. Applications and algorithms-loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1023–1032, 2001.
- [19] F. Dai and J. Wu. Performance analysis of broadcast protocols in ad hoc networks based on self-pruning. *Parallel and Distributed Systems, IEEE Transactions on*, 15(11):1027–1040, 2004.
- [20] M.T. Sun, W. Feng, and T.H. Lai. Location aided broadcast in wireless ad hoc networks. In *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE*, volume 5, pages 2842–2846. IEEE, 2001.
- [21] J. Cartigny and D. Simplot. Border node retransmission based probabilistic broadcast protocols in ad-hoc networks. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2003.

- [22] A. Kokuti and V. Simon. Location based data dissemination in mobile ad hoc networks. In *Telecommunications and Signal Processing (TSP), 2012 35th International Conference on*, pages 57–61. IEEE, 2012.
- [23] G. Korkmaz, E. Ekici, and F. Ozguner. Black-burst-based multihop broadcast protocols for vehicular networks. *Vehicular Technology, IEEE Transactions on*, 56(5):3159–3167, 2007.
- [24] K. Xu, M. Gerla, and S. Bae. How effective is the ieee 802.11 rts/cts handshake in ad hoc networks. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 1, pages 72–76. IEEE, 2002.
- [25] J. Kim, Q. Zhang, and D.P. Agrawal. Probabilistic broadcasting based on coverage area and neighbor confirmation in mobile ad hoc networks. In *Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004. IEEE*, pages 96–101. IEEE, 2004.
- [26] R. Newton, G. Morrisett, and M. Welsh. The regiment macroprogramming system. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 489–498. IEEE, 2007.
- [27] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.
- [28] M. Mamei, F. Zambonelli, and L. Leonardi. Tuples on the air: A middleware for context-aware computing in dynamic networks. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 342–347. IEEE, 2003.
- [29] J. Bachrach, J. Beal, and J. McLurkin. Composable continuous-space programs for robotic swarms. *Neural computing & applications*, 19(6):825–847, 2010.
- [30] I. Litovsky, Y. Métivier, and W. Zielonka. The power and the limitations of local computations on graphs. In *Graph-theoretic Concepts in Computer Science*, pages 333–345. Springer, 1993.
- [31] E. Klavins, R. Ghrist, and D. Lipsky. A grammatical approach to self-organizing robotic systems. *Automatic Control, IEEE Transactions on*, 51(6):949–962, 2006.
- [32] M. Yim, W.M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G.S. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *Robotics & Automation Magazine, IEEE*, 14(1):43–52, 2007.
- [33] J. Beal and J. Bachrach. Infrastructure for engineered emergence on sensor/actuator networks. *Intelligent Systems, IEEE*, 21(2):10–19, 2006.

- [34] J. Armstrong, R. Virding, C. Wikström, and M. Williams. *Concurrent programming in ERLANG*, volume 2. Prentice Hall, 1996.
- [35] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: portable parallel programming with the message passing interface*, volume 1. MIT press, 1999.
- [36] B. Chapman, G. Jost, and R. Van Der Pas. *Using OpenMP: portable shared memory parallel programming*, volume 10. MIT press, 2007.
- [37] M. Govett, L. Hart, T. Henderson, J. Middlecoff, and D. Schaffer. The scalable modeling system: directive-based code parallelization for distributed and shared memory computers. *Parallel Computing*, 29(8):995–1020, 2003.
- [38] C. Pressey. The alpaca meta-language. <http://catseye.tc/projects/alpaca/>, 2012.
- [39] J.D. Eckart. Cellang: Language reference manual. *Radford University, Radford, VA*, 1995.
- [40] U. Wilensky. Netlogo. <http://ccl.northwestern.edu/netlogo/>, 2011.
- [41] J. Beal. Programming an amorphous computational medium. *Unconventional Programming Paradigms*, pages 97–97, 2005.
- [42] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1312–1321. IEEE, 2003.