MASTERARBEIT

Titel der Masterarbeit

"Distributed Matrix Multiplication on a Mobile Sensor Network"

Verfasser

Edhem Brakmic, Bakk.techn.

angestrebter akademischer Grad

Diplom Ingenieur (Dipl. -Ing.)

Wien, 2012

Studienkennzahl lt. Studienblatt:	066 940
Studienrichtung lt. Studienblatt:	Scientific Computing
Betreuer:	Univ.-Ass. Privatdoz. Dr. Wilfried Gansterer

Acknowledgments

Alhamdulillah, I completed this thesis with support and assistance from benevolent people around me. First of all, I would like to thank my supervisor Dr. Wilfried Gansterer for his professional support and guidance. His patience and advices were invaluable for this thesis. Further, I also give my thanks to my colleagues from TU Wien for their proofreadings and feedbacks. I expand my thanks to all my friends for their help and moral support they gave me during these months. Last but not least, I would like to thank my entire family for their support. Particularly my wife Zehra for her love and understanding.

Abstract

One of the most frequent operations in the field of image processing is the matrix multiplication. Parallel algorithms and concepts are well known and developed, but in case of decentralized systems there is still a lack of new approaches. In this thesis we proposed different approaches of performing a matrix multiplication on a decentralized mobile sensor network. The aggregation process over the nodes is based on the Push Sum algorithm. We introduced four strategies for the matrix multiplication on a mobile sensor network. Results have shown that it is relatively hard for a random node distribution and constant mobility to achieve high accuracy. Therefore, we considered different deterministic topologies with high immobility periods, particularly in the transient phase of the aggregation process. Under certain circumstances, we reduced the message loss and achieved nearly half of double precision.

Keywords: matrix multiplication, network, sensors, nodes, forwarding, mobility, subnet, push-sum, gossip algorithms, wireless

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations and Symbols

WSN	Wireless Sensor Network
PSA	Push Sum Algorithm

1 Introduction

1.1 Background

Advances in wireless communication offer a large set of opportunities. Sensing and actuation technology can be embedded into our physical environment at an unprecedented large scale and fine granularity [1]. Further, dense sensor networks provide highly accurate information from our physical environment. Measured data have to be processed and distributed over the network. Additionally a base station collects the data and sends it to an external host machine for further processing. Items covered with sensor nodes can communicate with the environment. Connecting these items to the present IT infrastructure raises new challenges. A world-wide connected *network of things* is foreseen as the future of internet [2]. Such architectures demand new approaches and algorithms to give answers to the future challenges of distributed systems.

Wireless sensor networks solve problems not suited for conventional IT infrastructures. The communication of Wireless Sensor Networks (WSNs) is based on gossiping protocols. A gossiping protocol is a communication protocol inspired by the social phenomena of gossiping. Gossiping protocols have to be used in case of frequent node failures and message losses.
Gossiping and peer-to-peer networks rely on the assumption of neighborhood communication. These kinds of networks satisfy the following conditions:

- fault-tolerance is provided
- reliable communication is not assumed
- convergence should be achieved in finite rounds
- network self-stabilization

Matrix multiplications are some of the most common arithmetic operations used in several scientific applications. They are most of all used in the fields of graphics, robotics, signal processing and digital control. Matrix multiplications are usually performed on high performance computers with several cores and an immense of shared memory. Parallel con-

cepts for these kinds of problems are widely known and well developed. However, in the case of ad-hoc networks without appropriate IT infrastructure and limited resources, there is still a lack of new approaches and algorithms. In the more recent past camera networks have become widely pervasive in many applications in our environment. Public institutions, shopping centers, public places are equipped with a large scale of cameras for surveillance and safety reasons. Such technologies enabled police officials to share visual information in real time over the internet. At the same time, a lot of problems of processing and storing recorded data has arose from this development. Security, bandwidth and storage concerns led to the necessity of developing algorithms for distributed camera architectures. Usually data retrieved from these networks is analyzed and processed manually. Thus, this is a extremely costly task [3]. Distributed camera sensor networks perform these tasks before sending the raw data to host machine. In such records a large noise level can exists and, in order to eliminate the noise from the picture a filter could be set (Figure 1). In fact, a common color filter represents a matrix multiplication between a 2D image matrix and a 2D filter matrix.

Figure 1: Noise filtering

Similarly, robot sensor networks in rescue missions, collects images of the operational environment. In such areas there is no appropriate network infrastructure to collect the data and send it to a server machine. Also, there is not enough time to process the images on the server. On the other hand, matrix multiplication is also essential in grid computing. Extremely large problem sizes could be distributed over a network of high performance com-

puters. These networks could take advantage from the fault tolerant and self-organizing properties of a distributed matrix multiplication approach.

1.2 Problem description

The objective of this study is to implement a distributed matrix multiplication algorithm. The first step is to find a gossiping algorithm which is appropriate for the matrix multiplication. Therefore, the Push Sum algorithm (PSA) is chosen and adapted to the matrix multiplication. The idea is to distribute the elements of two matrices A and B over a gossiping network and let them communicate with each other. The main challenge of this idea is to let the nodes communicate, but only in the same range (subnet). This means that the first subnet consists of the product of the first row of matrix A and the first column of matrix B. The complexity of mobile sensor networks also opens the question of routing. Nodes move, lose old and establish new connections, but the neighborhood route table must be updated. Similarly, there is the question of the message overhead. The huge bulk of messages would perhaps lead to a faster convergence, however, it needs more energy and the possibility of node failure is much higher. All of these problems will be considered and we will try to give an overview of various matrix multiplications and the possible unstable states of the Push-Sum algorithm (PSA).

1.3 Related work

Papers related to the problem of sensor networks and matrix operations seem to be rare. Therefore, we designed entirely new approaches based on the ideas of [4], [5] and [20].

The authors of [4] discussed the challenges of designing algorithms for sensor networks. They categorized them, upon how their in-network aggregation is performed into centralized, tree and gossip based aggregation. Further, they introduced a hybrid-protocol, which combines the advantages of tree and gossip-based protocols. In such a protocol, nodes are partitioned into groups, where the groups communicate with each other over cluster heads. The results showed that a hybrid protocol outperform tree and gossip-based protocols. Additionally, the authors of [5] have also proposed a similar hybrid-protocol for large scale networks with a high fault rate. They also assumed that nodes are grouped, and a clusterhead for group to group communication is present. In paper [20], a deterministic hot-potato

protocol was investigated. For such a protocol, every node holds the information of *good directions,* where the probability to reach a destination node is high. The authors of [20] considered a statical network and the *good directions* are calculated before the sending exactly one time. However, for mobile networks, the calculation of *good directions* is rather expensive. Every time, when the nodes are moving, they has to recalculate the new *good directions.* Thus, we decided to use a random hot-potato protocol for the forwarding of messages.

Moreover, the authors of [22] explained the effect of mobility in relation to the convergence speed and accuracy. They stated that mobility is a challenging behavior, in order to achieve a high accuracy, but on the other hand, mobility has a positive effect on the convergence speed. All these patterns led us to the idea of mobility and forwarding messages between subnet groups.

1.4 Project organization

This master thesis is a continuation of the lecture “Praktikum Scientific Computing”, in which we have simulated and designed a Push Sum algorithm in OmNet++. The implementation provides the feature to categorize the nodes in subnets and provides principles to let them communicate over the whole network. In addition, we held two seminars on this topic. The master thesis is based on the implementation from the previous lecture and upgraded by the MiXiM framework. The scientific method of this research is a simulation under limited conditions. A sample of representative scenarios have been determined. The first section will present the gossiping technique and the push sum algorithm. What follows is a discussion of simulation conditions and limitations. We will then go on to demonstrate the initial matrix distribution and different consensus approaches. The final selection will evaluate the simulation results and give a conclusion.

2 Gossiping and the Push Sum algorithm

2.1 Gossiping

Figure 2: Social behavior of insects

The gossiping phenomena is widely pervasive in nature. Information broadcast in a gossiping based manner is simple and very effective. Viral infections spread with enormous speed, following simple rules of peer-to-peer communication. Ants are a good example of self-organizing networks. On their food search ants mark the routes with pheromones. The ants algorithm works as follows [6]:

- A food source is recognized
- In every unit T of time the ant pushes the news from its position X to position Y randomly chosen in its neighborhood
- First choosing a destination then updating pheromones

This form of communication we also recognize in the waggle dance of bees and other social insects (Figure 2). Here, we can recognize a basic pattern of gossiping communication.
The Gossiping protocol is a communication protocol between two nodes (sensor, cell phone, peer etc.) in a decentralized manner. Every node performs the same GP and chooses at a time unit R a random neighbor (peer) for communication. This time unit R is called *round*. Each node measure, sends (pushes) and updates its value in every round. After a certain number of rounds the information is spread around the whole network. Every node tries to obtain the global information; sending/receiving messages from the local neighborhood.

Each node can send messages independent (asynchronous) or simultaneous with all other nodes in the network (synchronous). For asynchronous implementations the message must have a round counter to determine the round for each node. Gossiping protocols have three types of message exchange:

- Push (Each node pushes news in its neighborhood)
- Pull (Each node sends messages and asks its neighborhood if there is some news)
- Push-Pull (Hybrid algorithms for better optimization)

Other important characteristics of gossiping algorithms are convergence time, number of rounds and time synchronization. These characteristics are variable and depend on the actual developed algorithm. They have been investigated especially in relation to computation complexity of the gossiping algorithm (for instance aggregation algorithms). Gossip-based protocols can be classified into two different classes [7]:

- anti-entropy protocols
- rumor-mongering protocols

Anti-entropy protocols have the characteristics to detect and correct inconsistency in a sensor network in a certain number of gossip rounds. [8]. Every node chooses another node for message exchange. Both nodes exchange their values and search for any difference between the two values [**9**]. Anti-entropy protocols are highly reliable, but on the other hand very expensive. In every round nodes have to communicate in both directions and compare the whole values. Therefore, anti-entropy protocols are designed to solve problems with small value sizes. In case of huge matrices as values, anti-entropy protocols produce a large overhead. Rumor-mongering protocols distribute the information until all nodes are informed, with a very high probability. This assumption does not ensure that all nodes are infected with the new information. In rumor-mongering protocols nodes are sending messages only in one direction, sending a rumor in every round to a randomly chosen neighbor. This received message becomes the next “hot rumor” which is sent to the next neighbor. Mostly, such algorithms stop after a node has recognized that a large number of neighbors have updated their values. Rumor rounds can be more frequent than anti-entropy rounds because they need fewer resources on each side. Algorithms that spread information in a natural

manner, following simple rules to spread information and having only a local view, are called epidemic algorithms [9]. Further, there is the classification related to the epidemic algorithms in simple and complex classes. Simple epidemics have two states for the nodes: susceptive and infective (SI model). A node is infective if it has the information and starts to spread it. A susceptive node waits for the information and as soon as it receives the information it becomes infective. Complex epidemics have three states. The third state is the removed or recovered state. Nodes in the removed state cannot be infected. For reason of infection rate control, some nodes are set to the removed state [7].

2.2 Push Sum Algorithm

The PSA is a very natural gossip-based algorithm. It belongs to the group of aggregation algorithms. The goal of the PSA is to compute sums, averages, minima or maxima in a decentralized manner. The PSA also fits more in the group of rumor-mongering algorithms because of its aggregating nature. We assume that each node *i* holds in each round *n* a value $x_i(n)$ and a weight $w_i(n)$.

Let *p* be a randomly chosen communication partner from the neighborhood. The PSA performs the following steps[10]:

Algorithm 1: Classic Push Sum

i. Initial step set all pairs { $x_i(0)$, $w_i(0)$ } where *i* represents the node number.
ii. Choose randomly a communication partner *p* from the neighborhood *H*.
iii. Send { $\frac{1}{2}x_i(n)$, $\frac{1}{2}w_i(n)$ } to *p* and *i*(yourself).
iv. Calculate the new pair { $x_i(n+1)$, $w_i(n+1)$ } with

$x_i(n+1)=\frac{1}{2}x_i(n)+\sum_{\forall r\in H}x_r$ and $w_i(n+1)=\frac{1}{2}w_i(n)+\sum_{\forall r\in H}w_r$ where *r* represents a neighbor sending to *i* and *n* the round number. For the next round we start from ii. again.

In each round *n*, nodes send their pair { $\frac{1}{2}x_i(n)$, $\frac{1}{2}w_i(n)$ } to itself and a randomly chosen neighbor. Further, the nodes collect all received messages and summarize it to the new pair { $x_i(n+1)$, $w_i(n+1)$ }. In consideration to the initial value we can calculate the sum or average of the PSA values by setting $x_i(0)$ and $w_i(0)$ in the step i. of Algorithm 1. The average will be calculated for the initial $x_i(0)$ random and $w_i(0)$ =1. If we initialize $x_i(0)$ random and a weight $w_1(0)$ = 1 and all other with 0, the PSA performs the summation of $x_i(0)$ in the network. The summation can easily be modified to a node counter by setting $x_i(0)$ = 1. The estimate of the aggregation function is $\frac{x_i(n)}{w_i(n)}$ for each node in every round. The estimation accuracy of the PSA improves as far as the nodes adjust their values.

The sum of all values in the network must be constant (*mass conversation*) along the time as stated in [10]. This property is a prerequisite for the correctness of the PSA. In case of node failures and *mass* (message) *loss* this prerequisite is not satisfied any more. The author of [10] recommends a *mass* recover strategy by sending undeliverable messages back to the source. Unfortunately, the author only offers the approach how to solve this problem, but no implementation. This approach opens various questions for the network stability and routing.

The error in average calculation drops within ε in $O(\log n + \log\frac{1}{\varepsilon} + \log\frac{1}{\delta})$ rounds where 1-δ is the probability that the estimate of $\frac{x_i(n)}{w_i(n)}$ conforms within every node in the network [10]. Thus, this convergence criteria is valid only for static networks. The convergence criteria of mobile networks also depend on area size, mobility speed and nature of movements.

3 Routing

Unlike conventional networks, WSNs still have no standard routing protocol. The routing strategy of WSN depends on several factors, such as power capacity, network topology, observed environment, mobility etc. Conventional wireless networks use state or distance vector routing protocols for the routing link. But for mobile ad-hoc networks these approaches are not appropriate. They periodically produce a large bulk of control messages and for large networks frequent message exchange is needed [11]. Routing in WSNs is especially difficult if frequent and fast mobility is imposed. For gossip-based algorithms a local view of the neighborhood is obligated. In case of static networks, routing is achieved in a first initial round, but for mobile approaches the local view must constantly be updated. Nodes move, lose old and establish new connections. If the nodes move fast, the update periods will be short and the message overhead increases. Thus for networks with slow motion the update periods and message overhead remain feasible. According to [12] all WSN routing protocols can be classified in:

- flat-based
- hierarchical-based
- location-based

In flat-based routing, all nodes have the same roles and abilities, they strive to perform some a couple of sensing task. In such protocols, every node floods its neighborhood. Flooding produces implosion, caused by receiving duplicated messages. Overlaps occur if two nodes are sensing the same area and flooding the same neighbor. Flooding also leads to resource blindness, consuming large amounts of power [13]. Hierarchical-based or cluster-based routing is a two layered routing protocol. It is assumed that nodes with various capabilities and amount of power exist in the network. Nodes are typically organized into clusters where every cluster has a cluster-head node. Data transition, in such networks, can be *intra-cluster* (member nodes of a cluster communicate with each other) or *inter-cluster* (cluster-heads communication) [14]. Member nodes are only allowed to communicate over their cluster-heads with other cluster nodes. This assumption saves energy and reduces the routing tables of member and cluster-head nodes. Location-based routing depends on the posi-

tion information of the nodes. This approach attempts to send messages to its destination, rather than flooding the whole network. Each node calculates the distance between itself and the destination node, by acquiring position information about position. The position information can be retrieved by the estimation of signal strengths or via GPS signal [12].
In case of a gossip-based algorithm there is no need for a global view of the network. Every node is communicating only with its neighbor nodes. But a local view of the neighborhood is for some gossip-based algorithms essential. Gossiping nodes would have to perform some kind of *local routing*. Every node would have a routing vector with all the addresses of its neighbors. If a node performs a push sum algorithm, it would send its message in every round to a randomly chosen neighbor. Hence, how does the node know its neighborhood? If we assume a static network, the nodes have to explore its neighborhoods only in a first initial round or would be provided by a global administrator. However, if we have a mobile sensor network, nodes have to keep track about its neighbors constantly. Such nodes would have a permanent changing neighborhood. This fact opens a large topic of routing in mobile sensor networks. We solve the local routing problem with a kind of flat-based protocol. Our first approach was to send periodically controlled messages and wait for response. In fact, if we choose small intervals between the control messages, the nodes could move faster. Unfortunately, in such an approach, each node would be too busy with sampling the neighborhood that there would not be enough resources to perform an aggregation algorithm. Therefore, we assume slow moving nodes, where the sampling period is rather large. After a couple of initial experiments we recognized that for slow motion there is no need for control messages. In the first round all nodes would send a message to the environment with no recipient. Each node (which is in the transmit range of the sending node) would then collect all messages from its neighborhood and save the consignee address to its routing vector. For the next round each node would pick a certain neighbor from its routing vector. In such a routing protocol nodes acquire information about its neighborhoods from the previous round. This kind of routing is learning by the past behavior of the network. Benefits of such an approach are avoiding message overhead, simplicity and faster convergence (same number of rounds but faster in real time).

4 Push Sum adapted for the matrix multiplication

The matrix multiplication is defined as a row(i)-by-column(j) multiplication of two matrices A and B. Each row of matrix A is multiplied with the first column of matrix B to an entry of the product matrix. The matrix multiplication is in general not commutative, therefore $AB \neq BA$.

In order to compute the product matrix $P = AxB$ where *A* and *B* are *dense matrices* of *N*x*N* dimension, we have to perform $O(N^3)$ operations. For each element P_{ik} we have to calculate:

$$p_{ik} = \sum_{j=1}^{N} a_{ij} b_{jk} \tag{5}$$

or

$$p_{ik} = a_{i1} b_{1k} + a_{i2} b_{2k} + a_{i3} b_{3k} \ldots a_{ij} b_{jk} \tag{6}$$

where p_{ik} represents the elements of the product matrix. For the push-sum algorithm we perform a summation over the values of a node network, by setting the weight of the first node to 1, and the rest to 0. The sum approximation is represented by the estimation of $\frac{x_i}{w_i}$ on each node. If the pairs $a_{ij} b_{jk}$ are the values x_i on each node, then these pairs can be exchanged in a gossip-based manner. The estimation $\frac{(a_{ij} b_{jk})}{w_i}$ is the approximation of one element of the product matrix at each round. We introduced three different initialization and message exchange strategies. The first two initialization strategies build a network with a different number of sub-networks, namely:

- Element distribution
- Block-matrix distribution

These initialization strategies use different message exchange approaches to achieve convergence. We classify them as:

- Message forwarding
- Node mobility
- Forwarding and mobility

Likewise, we have also analyzed a sub-network free approach based on a column/row distribution. In the following sections we describe these initialization and message exchange strategies in detail.

4.1 Node initialization

We have introduced three strategies in relation to memory capabilities of the nodes, whose initialization matrices are uniformly distributed across the network. Each node in the network stores the same amount of information. Furthermore, the amount of message exchange/passing in the network is inverse proportional to the load of information distributed across the nodes. This means that a large amount of information decreases the number of messages in the network, while increases the bandwidth for one single message exchange. The first two approaches perform data aggregation in the sphere of distributed sensor network clusters or sub-networks. The authors of [15][16][17] propose several clustering techniques for WSNs, which mostly assume static networks [18][19] with certain distribution. Communication in such networks can be intra-cluster, where all nodes communicate within the cluster, or inter-cluster where cluster-heads communicate with each other. In [4], a hybrid protocol for inter-cluster (group) gossiping was introduced. In this so called hybrid protocol, nodes are grouped into clusters. Every cluster has a cluster-head which identifies the cluster, and the cluster-heads collect information from the clusters and forward it to the base station (sink). In [5], a similar approach was presented. The intra-cluster aggregation is still gossip-based but the inter-cluster aggregation, represented by the cluster-heads, is now tree based. Thus, if the nodes are mobile and permanently change their positions, clustering becomes more complex. In order to let the members of one cluster communicate with each other, despite of the fact that they are out of range of one another, multiple hop communication is required. However, for our proposed method, cluster-heads are unnecessary due to

the fact that we disregard the problem of data collecting to a base station. It also has to be mentioned that all nodes within a subnet have the same capabilities and resources.

4.1.1 Element distribution

As mentioned in the previous section, we distribute one element (a_{ij}) of matrix *A* and one element (b_{jk}) of matrix *B* on every node. The product of these two elements ($a_{ij}b_{jk}$) represents the value of the push-sum algorithm. For the first element of the NxN product matrix(aggregation result) N values and N nodes are required. In Figure 3, the general idea is depicted. The product $a_{11}b_{11}$ is stored on the first node, where $a_{12}b_{21}$ is stored on the second and so on. As we can see in Figure 3, the values $a_{11}b_{11}$, $a_{12}b_{21}$ and $a_{13}b_{31}$ build a sub-network (gray sub-network). The aggregation for the first element of the product matrix, is performed over the members of the gray sub-network.

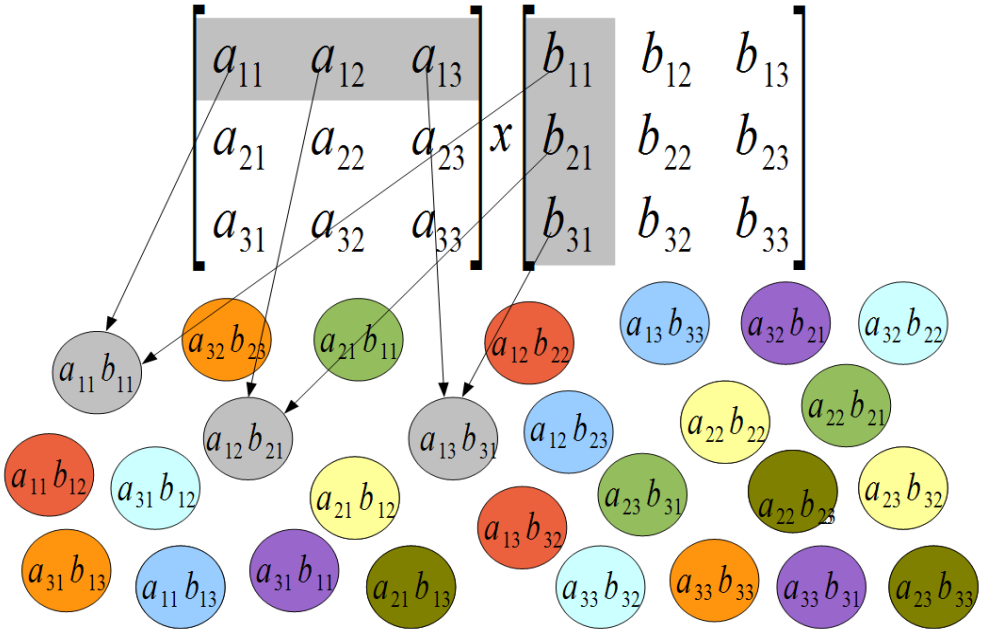


Figure 3: Element distribution of two 3x3 matrices over the whole network and the subnet grouping

Figure 3 demonstrates the initial distribution of two 3x3 matrices. For this example we have 9 sub-networks and 27 nodes in the network. For a *N*x*N* matrix we need N^3 nodes, separated into N^2 subnets. For the first node (In case of subnet 1 node $a_{11}b_{11}$) of each subnet, the weight is set to 1 and the rest to 0.

4.1.2 Block-matrix distribution

For the block-matrix distribution we utilized the same pattern as for the element distribution. Hence, instead of two elements we distributed a block of the matrices A and B (Figure 4) on each node.

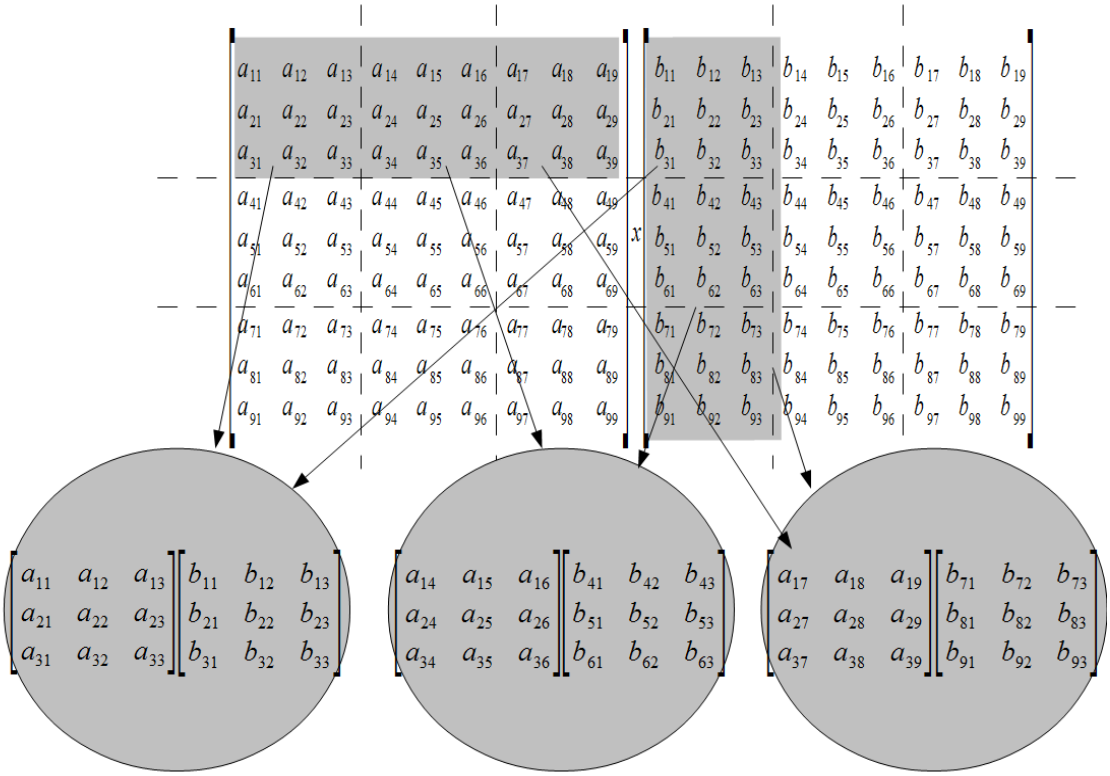


Figure 4: Block matrix distribution over the first sub-network

For this strategy we had to perform a matrix multiplication of two sub-matrices on every node. In Figure 4, the first left 3x3 matrices are multiplied and stored on the first node. The value of the PSA on every node turns into a matrix, whereas the weight remains a number. With this approach we unbound the matrix size from the network size. The consequence of such a distribution is of course the large message size. Every node has to send a matrix as a message instead of a number.

4.1.3 Column/row distribution

Finally, we designed a third method, where one column of matrix *A* and one row of matrix *B* are distributed on each node. Every node builds an initial matrix with this informa-

tion, which represents the value of the PSA. The initial matrix is built by calculating the outer product of one column of A($C_a^{'}$ is C_a transposed) C_a and one row of B($R_b^{'}$ is R_b transposed) R_b :

$$C_a^{'} \times R_b^{'} = C_a^{'} R_b^{'} = \begin{bmatrix} a_{11} \\ a_{21} \\ \cdot \\ \cdot \\ a_{n1} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & . & . & . & b_{1n} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1n} \\ a_{21}b_{11} & a_{21}b_{12} & \cdots & a_{21}b_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}b_{11} & a_{n1}b_{12} & \cdots & a_{n1}b_{1n} \end{bmatrix} \quad (7)$$

This approach builds an approximation matrix(right matrix in expression 7) for the PSA. Sub-networks are avoided for this solution since all nodes are in the same network range and communicate with each other. All nodes store different approximation matrices, and by performing the PSA they converge to the product of matrix A and B. This leads to a considerable reduction of the number of nodes. Thus, for an $N \times N$ matrix multiplication only N nodes are required. In Table 1 a comparison of the value and weight for all distribution strategies is given.

Strategy	Value [x_i]	Weight [w_i]
Element distribution	Number	Number
Block-matrix distribution	Matrix	Number
Column/Row distribution	Matrix	Vector

Table 1: Parameters of the PSA for different distribution strategies

4.2 Subnet convergence strategies

Introducing sub-networks creates new challenges for message exchange. Nodes are grouped in subnets and have to communicate only inside of their subnet. If a node wants to send a message to a subnet member beyond its transmission range, it has to build bridges over nodes with different subnets or to move towards the neighborhood of the destination node. All the following message exchange strategies are built on the random neighbor assumption. Each node picks a random neighbor for communicating in every round. In addition, all nodes move randomly and hope to find a subnet member for communication.

4.2.1 Message forwarding

We have introduced a forwarding strategy for static networks with a certain number of sub-networks. All member nodes of one subnet strive to achieve a consensus. Our solution for this problem is an adapted *hot-potato* protocol [20]. The hot-potato protocol forwards messages from a source to a sink (destination) with multiple hops. All nodes try to forward the message to the first possible opportunity in the neighborhood. For such an approach only a local network view is necessary. Our subnet distribution assumes multiple message forwarding and multiple sinks in the network. In every round each node sends a message to itself and to a randomly chosen neighbor. The receiver proves if its subnet number is equal to the source subnet number. If it is different, it picks and forwards the message to a randomly chosen node from the neighborhood (except the node of its origin). Additionally, every message poses a hop counter. If the hop counter achieves a pre-defined limit, the forwarding stops immediately. If the network is split or the message is pending between two nodes, we would get a message overhead. In the worst case scenario, the message might never find a subnet member. Because of this problem we set a hop counter. The message is counting how often it is forwarded, and after a certain number of hops the message is deleted. The hop number depends on the time span between two rounds. For large time spans it is possible to have more time to hunt for a subnet member, but for small time spans it is likely to have only a few hops. If the hop count is small, the probability of a consensus in the subnet is also rather small. Algorithm 2 shows the steps of initialization and forwarding messages. In iii. and iv. the general idea of forwarding is described. Nodes forward a message until the condition $hopCount < hopLimit$ is true. If the *hopLimit* is achieved the message has to be deleted.

Algorithm 2: Message Forwarding and Push Sum

i. Initial step set $x_i(0)=a_{ij}b_{jk}$. If *i* is first subnet member then $w_i(0)=1$ else $w_i(0)=0$. Set $s_i=subnetID$, where subnet members have the same *subnetID* .

ii. Chooses randomly a communication partner *p* from the neighborhood *H*.

iii. Set $hopCount=0$. Send { $\frac{1}{2}x_i(n)$, $\frac{1}{2}w_i(n)$, s_i , *hopCount* } to *p* and *i* (yourself).

iv. If $s_i \neq s_r$ and $hopCount < hopLimit$ choose a new communication partner p and forward { $\frac{1}{2}x_r$, $\frac{1}{2}w_r$, s_r, *hopCount+1* } to p. If $s_i=s_r$ calculate the new pair{ $x_i(n+1)$, $w_i(n+1)$ } with $w_i(n+1)=\frac{1}{2}w_i(n)+\sum_{\forall r\in H} w_r$ where *r* represents a neighbor sending to *i* and *n* the round number . For the next round we start from ii. again.

In case of the block matrix distribution, the same pattern is followed. Hence, the value $a_{ij}b_{jk}$ is a matrix instead of a number.

4.2.2 Node mobility

Our second approach deals with the fact that the nodes are moving and that the member nodes of one subnet potentially move towards each other in finite time. Each node moves and sends messages expecting to address the message to a subnet member node. Unfortunately, the drawback of such an approach is obvious; the nodes are sending large numbers of messages to themselves and to non subnet members which reject the message in order to perform the push sum. Every rejection falsifies the consensus of its sub-network, and as a result, the absorption of self messages with no awareness of delivery success leads to an irreversible *mass loss*. In fact, for mobile sensor networks with random mobility, there is yet no guarantee of convergence speed [21]. Nevertheless, in [22] is shown that for a small number of mobile nodes, a positive effect on the convergence speed is plausible. The authors of [22] analyzed the influence of different mobility models on gossip-based algorithms. The area of mobility is limited and nodes move, as in our simulation in a limited

2D space. These models are rather unrealistic but they have also stated that a *random walk,* under certain circumstances, can show the same results as for instance a bidirectional mobility. Namely, mobility that is limited to two directions can yield to a major benefit in convergence speed.

Algorithm 3: Mobility and Push Sum

i. Initial step set $x_i(0)=a_{ij}b_{jk}$. If *i* is first subnet member then $w_i(0)=1$ else $w_i(0)=0$. Set $s_i=subnetID$, where subnet members have the same *subnetID* .

ii. Chooses randomly a communication partner *p* from the neighborhood *H*.

iii. Send { $\frac{1}{2}x_i(n)$, $\frac{1}{2}w_i(n)$, s_i } to *p* and *i*(yourself).

iv. If $s_i \neq s_r$ then delete message. If $s_i=s_r$ calculate the new pair

{ $x_i(n+1)$, $w_i(n+1)$ } with $x_i(n+1)=\frac{1}{2}x_i(n)+\sum_{\forall r\in H}x_r$ and

$w_i(n+1)=\frac{1}{2}w_i(n)+\sum_{\forall r\in H}w_r$ where *r* represents a neighbor sending to i and n the round number . For the next round we start from ii. again.

Algorithm 3 is relatively similar to Algorithm 2 except that we no more have the *hopCount* parameter for forwarding messages. If a node receives a message with a different *subnet*, it will be deleted immediately.

4.2.3 Forwarding and mobility

The combination of the previous strategies is expected to unify the benefits of both, the forwarding and mobility approach. Since this approach has a higher message deliver reliability than the forwarding strategy, it provides a smaller message loss and an improved accuracy. Additionally, nodes are moving and changing their positions, also leading to a fair message distribution over the sub-networks.

4.3 Column/Row strategy

As we have seen in the previous section, it is rather challenging to perform a correct push sum aggregation over a sub-network if the member nodes are far away from each other. This happens especially if the nodes are mobile and large message loss is produced. To avoid the partitioning of the network in sub-networks and overcome this large message loss, we designed a sub-network free strategy. The idea behind this strategy is to distribute more information of the $N \times N$ matrices and find an approximate of the resulting matrix. In order to perform the push sum, we have to send some elements of this approximation matrix from one node to the other. The first approach is to send a randomly chosen element, which as a matter of fact, would lead to a large number of rounds for the convergence of large networks. Therefore, we consider one randomly selected row as the PSA value, meaning that we also have a row of N entries as the weight.

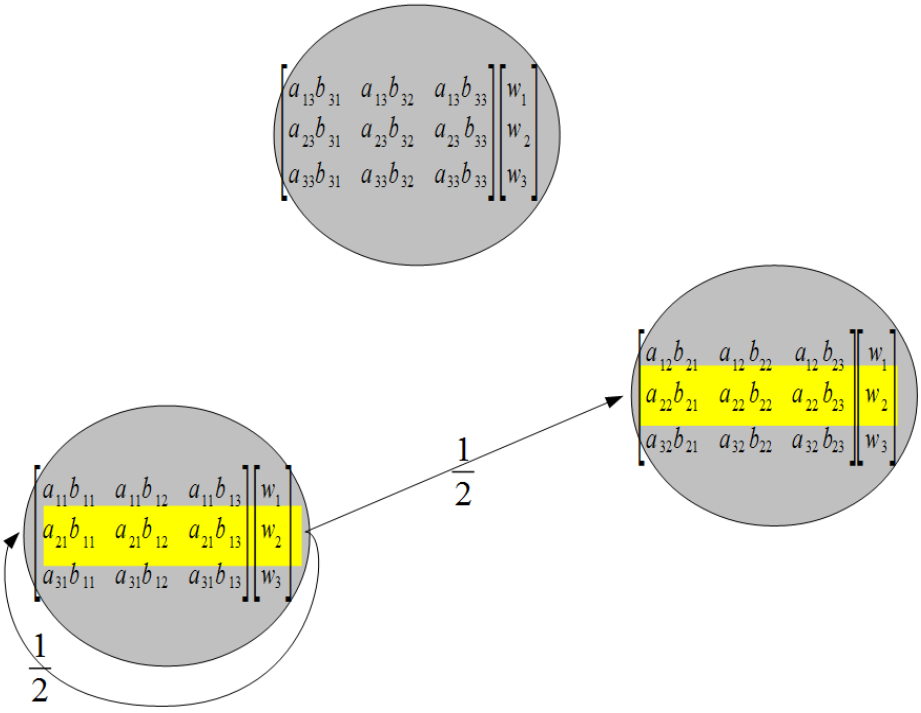


Figure 5: Message exchange within the Column/Row strategy of a 3x3 matrix

In Figure 5 we have a matrix A and B, whereas their columns/rows are distributed and transformed into the approximation matrix as depicted in expression *(7)*. Every node has its ap-

proximation matrix and a weight vector. According to Algorithm 4, each node selects a random row and the appropriate weight for the message transmission in every round of the PSA. The node in the bottom left corner of Figure 5 selects the second row and sends a PSA message with the half of $[a_{21}b_{11} \;\; a_{21}b_{12} \;\; a_{21}b_{13}]$ and w_2 to itself and its neighbor. As the PSA goes on all nodes store the product matrix of *A* and *B* at the end of the simulation. The cost to pay for this strategy is the large message size. In fact, instead of sending one number for the value and the weight, we have to send two vectors as value and weight. The following algorithm describes the column/row approach:

Algorithm 4: Column/Row strategy

i. Initial step built from the column of *A* and row of *B* the approximation matrix X_i , if *i*=0 set $w_0=1$ else $w_i=0$.

ii. Choose randomly a row number *k* from X_i and the *k* appropriate entry from w_i .

iii. Choose randomly a communication partner *p* from the neighborhood *H*.

iv. Send { $\frac{1}{2}X_i^k(n)$, $\frac{1}{2}w_i^k(n)$, *k*} to *p* and *i* (yourself).

v. Calculate the new pair{ $X_i^k(n+1)$, $w_i^k(n+1)$ } with

$$X_i^k(n+1)=\frac{1}{2}X_i^k+\sum_{\forall r\in H}X_r^k \quad \text{and} \quad w_i^k(n+1)=\frac{1}{2}w_i^k+\sum_{\forall r\in H}w_r^k$$

where *r* represents a neighbor sending to *i* and *n* the round number . For the next round we start from ii. again.

5 Simulation Environment

All simulations are performed in OmNet++ version 4.2. OmNet++ is an extensible and component-based C++ framework for network simulation. One module description consists of:

- Interface description (.NED file)
- Behavior description (C++ class)

Components are written in C++ and connected to the high level language NED. Components describe the behavior of a module whereby the NED file describes the module interfaces and network connections. Modules, gates and links can be created statically at the beginning of the simulation, or dynamically at the run-time. There is also an Eclipse IDE for OmNet++ with a simulation GUI. For the simulation of mobility we used the MiXiM framework.

5.1 Simulation area

First of all, we assume that the simulation area is a 2D space. It is also limited and quadratic. Nodes cannot move outside the simulation area; in other words nodes reflect on the area borders. Convergence speed and mobility are highly influenced by the size of the area and the network size. For large networks the simulation area must be adapted to provide enough space for node mobility, and to limit the visibility of each other. Thus, the simulation area is proportional to the network size. Figure 6 shows a network of three nodes in a square network space with the side length *l*. All nodes have the same transmission range *r*. Node *n2* and *n3* see each other and communicate with each other, whereas *n1* is out of the transmission range of *n2* and *n3*.

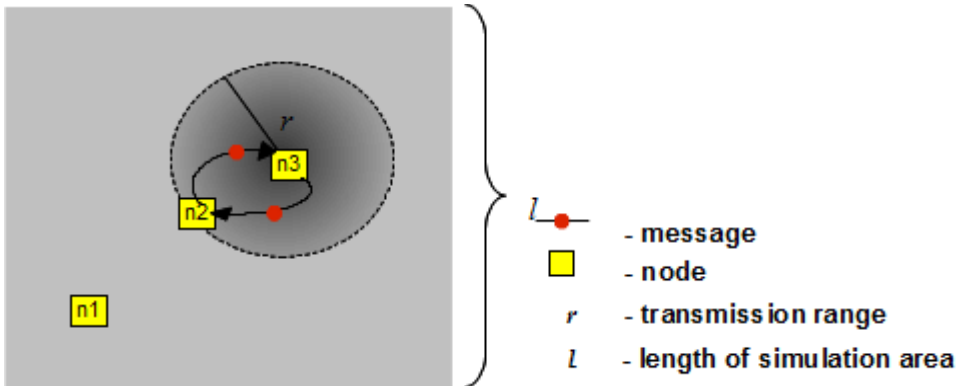


Figure 6: Simulation area

We have determined the size of the area in order to avoid a fully connected network (see Figure 6). This means that if we have N nodes and a transmission range *r*, the side length of the simulation area has to be:

$$l = r \cdot N \qquad (1)$$

Thus, the size of our simulation area is always:

$$P = (r \cdot N)^2 \qquad (2)$$

OmNet++ provides two configuration files; omnetpp.ini and config.xml. In these files we can set parameters, such as number of nodes, physical characteristics, signal strength, mobility speed etc.

5.2 Message sending and network topology

The network topology for our simulation is built by random. The transmission medium is air. If one node broadcasts, all nodes within the transmission range receive the message. Such a behavior is described in Figure 7. Node one is sending a message and all nodes within the transmission range receive the message.

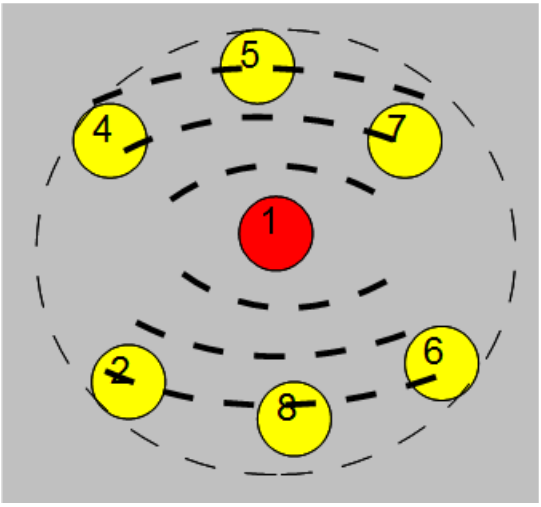


Figure 7: Illustration of a wireless transmission

There is no limitation for the bandwidth and number of connections per node. Nodes have the ability to send a message to the neighborhood, to send a message to itself and to receive messages. Nodes are equal and there is no base station in the network. The problem of collecting data from the network to a base station is not regarded. Connection between nodes is usual in both directions. Lost messages cannot be recovered. Unfortunately, nodes have no mechanism to recognize *mass losses*. If one node has no connection, it stays in its idle state. We sophisticated the stop criteria for our simulation. The simulation stops if all nodes (in the same subnet) store the same estimation $\frac{x_i}{w_i}$ in the order of 10^{-2} for random topologies, thus for deterministic topologies this criteria is set up to 10^{-16}. Once the nodes have reached a common estimate, the simulation finishes. If we set the precision to be of the order 10^{-2} the simulation will end sooner, but the price to pay is the estimation accuracy. In this case nodes adjust their values only to the second decimal place and the simulation stops afterwards. This assumption is not feasible for real world scenarios. In real world scenarios nodes have only their local estimations and no global view of the estimations in the network, the number of connections and the message size is limited. But for our prototypical analysis of the distributed matrix multiplication, it is sufficient.

5.3 MiXiM mobility features

Mobility in WSNs is either an incidental side effect or a desirable feature. Mobility can be produced by environment influences such as wind or water. Senses or nodes can also posses the ability to move, carried by some subjects or have automotive capabilities. Thus, we can categorize mobility in WSNs in active (desirable) or passive (influenced by some external power) . Mobility can be attached to certain nodes or to the whole network. Between the movements there can be a long immobility period or the nodes can move constantly [5].

In our simulation we used the MiXiM framework. The MiXiM framework is an Omnet++ extension, created for simulating fixed or wireless mobile networks. MiXiM is a merger of several simulation frameworks. The precedents are:

- ChSim by Universitaet Paderborn
- Mac Simulator by Technische Universitaet Delft
- Mobility Framework by Technische Universitaet Berlin, Telecommunication Networks Group
- Positif Framework by Technische Universitaet Delft

MiXiM offers a lot of features for modeling radio wave propagation, mobility and interference estimation. Mobility speed in MiXiM is defined as meter per second. Further, mobility speed in the MiXiM framework can be constant or variable. However, in our examples we considered constant speed mobility. Immobility periods are defined by the mobility update interval. A mobile node moves along a straight line for a certain period of time before it makes a turn. This moving period is a random number, normally distributed with an average of 5 seconds and standard deviation of 0.1 second.
When it makes a turn, the new direction (angle) in which it will move is a normally distributed random number with an average equal to the previous direction and standard deviation of 30 degrees. When it hits a wall, it reflects off the wall at the same angle.

6 Implementation study

The MiXiM framework provides a mechanism to simulate a fully layered network infrastructure. However, for our propose, only the application layer is in the sphere of our interest. We implemented our adapted PSA on the application layer and used only the address space from the underlying layer for the uniform message exchange.

6.1 Structure

Our implementation consists of 4 main methods on every node:

- `initialize(int stage)`
- `sendBroadcast()`
- `selfMessage()`
- `handleLowerMsg(cMessage* msg)`

In the `initialize` method, the nodes initialize the matrix value and weight. Further, `sendBroadcast` creates a messages, chooses a randomly chosen neighbor address and sets all parameters. Simultaneously, the node calls `selfMessage()` and sends the same message to itself. When a message is received, the `handleLowerMsg` method is called. In this method, the node decides if the message is in the same subnet or should be forwarded. The message has to be deleted if the maximal hop count is achieved.

6.2 Message

The messages contain following content:

- `int destAddr`
- `int srcAddr`
- `double mvalue[MATRIXSIZE*MATRIXSIZE]`
- `double weight`
- `int subnet`
- `int hopCount`
- `int round|optional`

Messages in the MiXiM framework are restricted to scalar and 1 dimensional array values. Therefore, we always had to ensure that the value matrix is converted to a 1 dimensional array with the size of MATRIXSIZE* MATRIXSIZE. Every node in the MiXiM framework has a unique node number from 0 to *N-1* (*N* is the network size). If one node is sending a message the srcAddr parameter will be set to the node number. The destAddr is a node number from the neighborhood vector. The neighborhood vector contains node numbers from all neighbors of the sending node (see Chapter 3 Routing).

7 Simulation Study

7.1 Correctness and Accuracy

In order to evaluate the correctness of the computed matrix, we compare the matrix norm of the true and computed value. The absolute error is defined as follows:

$$Absolute\,error = \|A_{calculated} - A_{true}\|_1$$

where

$$\|A\|_1 = max_{1 \leq j \leq n} \sum_{i=1}^{m} |a_{ij}| \qquad (3)$$

Further, the relative error in our calculations is:

$$Relative\ \ error\ \ \% = \frac{\|A_{calculated} - A_{true}\|_1}{\|A_{true}\|_1} * 100 \qquad (4)$$

$A_{calculated}$ is the resulting matrix distributed over the subnets. Each member of a subnet stores the same sub-matrix of $A_{calculated}$ at the end of the simulation. In the further simulation study, we have chosen one member node from every subnet, and collected their PSA estimations, which represents a sub matrix of the product matrix $A_{calculated}$. From these sub-matrices we build $A_{calculated}$ and calculate the relative error as described in (4). However, in case of the Column/Row strategy, all nodes hold the same matrix (estimation) at the end of the simulation. In order to find the $A_{calculated}$ only the estimation of one node from the network is necessary.

In further simulations we have distributed two randomly generated matrices over the network. The elements of these matrices are numbers between 0 and 1 from the set $\mathbb{R}$. These matrices are stored in a flat file and initialized before the simulation starts.

7.2 Mobility

In Tables 2, 3 and 4, we compared the relative error for several node speed values and for different mobility update intervals. By mobility update interval, we refer to the interval of time for which a node is immobile. In the simulations below, we considered a PSA performing an average aggregation over all distributed values over a network. Our aim was to show that the PSA is hardly vulnerable to message loss and link failures produced by node mobility. As we can see from the below tables, the relative error increases with arising node speed and network size. Nevertheless, for larger mobility update intervals, the relative error decreases. In fact for large mobility update intervals the nodes can perform a lot of rounds, between the two mobility moments with a very small message loss.

Speed [mps]	Mob.UpdateInterval [s]	Relative error %	Networksize
0.5	10	20.25	8
2	10	30.87	8
4	10	45.21	8
8	10	63	8
16	10	106.2	8
32	10	110.5	8
0.5	50	31.25	8
2	50	25.3	8
4	50	39.41	8
8	50	45.89	8
16	50	83.4	8
32	50	110.53	8
0.5	100	10	8
2	100	13.1	8
4	100	15.9	8
8	100	16.01	8
16	100	40.98	8
32	100	77.4	8
0.5	200	6	8
2	200	12.25	8
4	200	10.2	8
8	200	13.45	8
16	200	17.47	8
32	200	41.2	8

Table 2: Different speed and mobility update intervals for network size 8 - PSA averaging

Speed [mps]	Mob. UpdateInterval [s]	Relative error %	Networksize
0.5	10	14	27
2	10	11.48	27
4	10	15.11	27
8	10	17.62	27
16	10	63.25	27
32	10	132.5	27
0.5	50	12.5	27
2	50	13.67	27
4	50	14.39	27
8	50	17.76	27
16	50	54.9	27
32	50	102.1	27
0.5	100	8.51	27
2	100	11.23	27
4	100	12.45	27
8	100	15.2	27
16	100	39.12	27
32	100	83.23	27
0.5	200	7.48	27
2	200	9.23	27
4	200	11.5	27
8	200	14.11	27
16	200	20.9	27
32	200	75.4	27

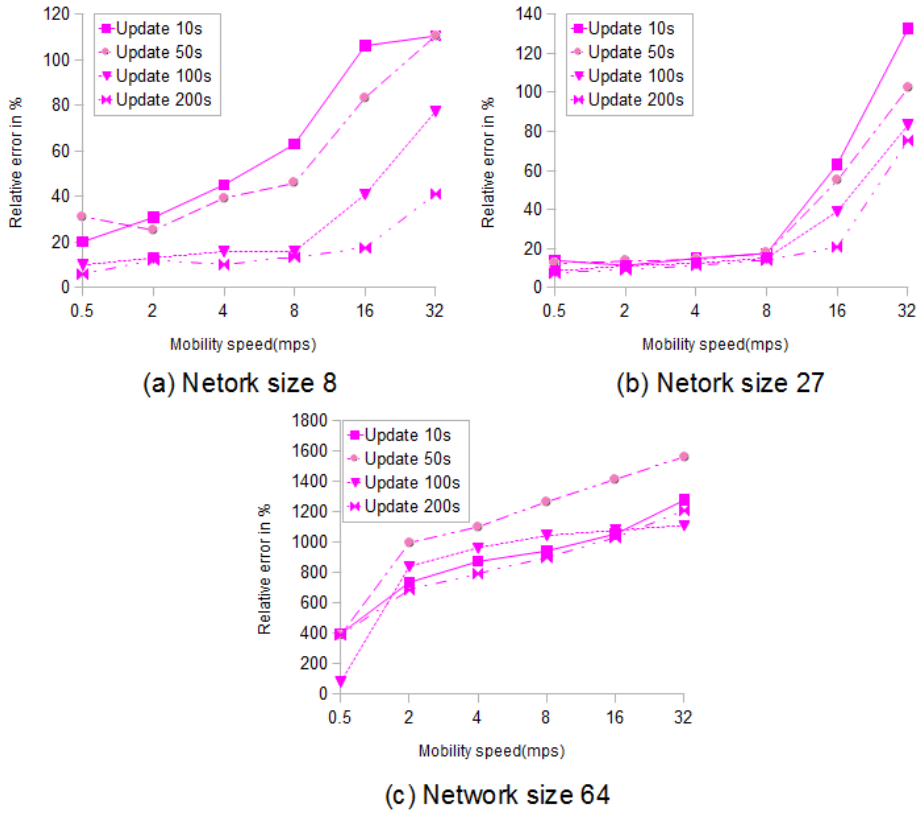
Table 3: Different speed and mobility update intervals for network size 27- PSA averaging

As we can see from Table 2 and 3, the relative error for 8 and 27 nodes occurs approximately in the same order. But intuitively, the relative error has to be larger for larger networks. The relative error in Table 4 is extremely large in comparison to Table 2 and 3. At this moment we have no explanation for this kind of behavior. This behavior has to be analyzed in-depth.

Speed[mps]	Mob. UpdateInterval [s]	Relative error %	Networksize
0.5	10	400.06	64
2	10	737.56	64
4	10	875.06	64
8	10	943.81	64
16	10	1056	64
32	10	1278.4	64
0.5	50	391.68	64
2	50	997.93	64
4	50	1102.4	64
8	50	1267.6	64
16	50	1416.2	64
32	50	1562.78	64
0.5	100	81.81	64
2	100	843.36	64
4	100	965	64
8	100	1045.4	64
16	100	1078.7	64
32	100	1112.3	64
0.5	200	390.93	64
2	200	689.25	64
4	200	794.8	64
8	200	901.4	64
16	200	1032.32	64
32	200	1210.2	64

Table 4: Different speed and mobility update intervals for network size 64- PSA averaging

In Table 4 we analyzed the behavior of the relative error for 64 nodes. The relative error is exploding, namely at the end of the simulation we have an altogether different product matrix for the matrix multiplication. Such a trend supports our assumption that the relative error is growing proportionally to the network size.



a) This figure shows the influence of mobility update interval and speed on a network with 8 nodes. After 8 [mps] a abrupt loss of accuracy is noticeable.
b) As in a) after 8 [mps] the relative error is growing faster.
c) From this figure we can conclude that mobility has a hazardous effect on large networks. The relative error, in this figure is at the 0.5 [mps] already about 100-400%. The result of this matrix is completely different from the true result.

Figure 8: Error for various kinds of speed and mobility update interval - PSA averaging

After 6 [mps] the accuracy decreases abruptly for all network sizes. In fact, permanent and fast movement of the nodes has a large influence on the correctness of the PSA. This kind of behavior can in particular be observed in Figure 8b. Therefore, it is highly complicated to hold track of the local neighborhood, if the nodes are moving permanently and fast.

7.3 Random subnet distribution and network topology

7.3.1 Forwarding

In the following simulation, a static network with random node distribution and topology is presented. Subnet member nodes are mostly connected virtually over different

non member nodes. Nodes can have 0 up to *N*-1 connections where *N* is the number nodes in the network. The message queue is set up to 100 messages for every node.

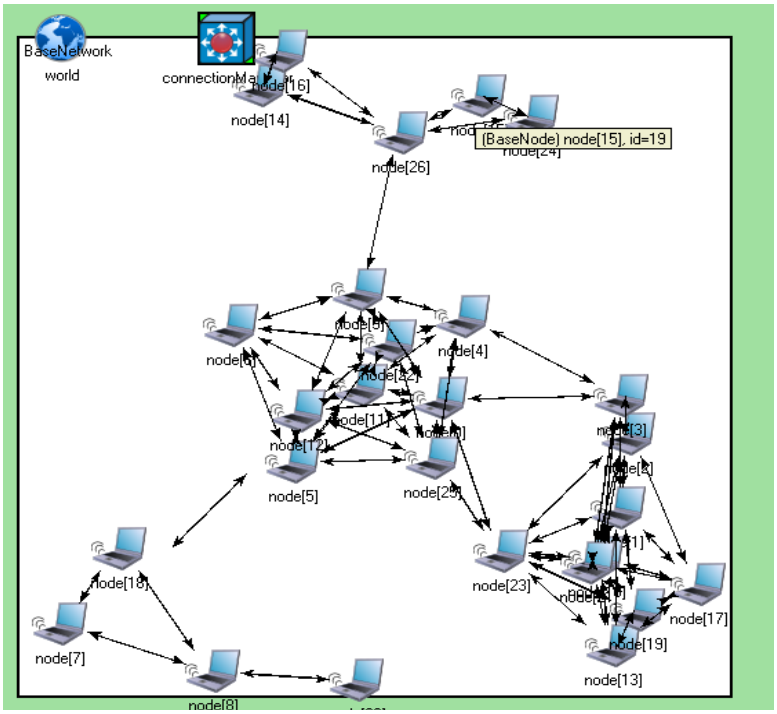


Figure 9: Random node distribution and topology

Figure 9 represents a random initial topology. In such a network, all nodes have a different number of connections. Some parts of the network are connected only over one connection with the other nodes in the network . In the case of mobility, the network could also fall apart in two groups but should join again because of the limited area size, after several numbers of cycles. In consideration of the *hot-potato* protocol, a large number of connections might be an advantage. Nodes would have a higher probability to be connected over few hops with the rest of its subnet members. On the other hand, a large number of connections makes it really hard for the hot-potato protocol to hunt a member node within the hop count limitation. In Table 5 we show the forwarding approach with different problem sizes. The column “sent messages” or “received messages” is the sum of all sent or received messages of all nodes in the network. We introduce the message loss ratio, which represents the proportion of sent and received messages (sent messages/received messages rounded to the closest tens). Meaning that for instance, if three messages are sent, the number of lost messages is three times the message loss ratio (3 x message loss ratio).

Matrix size	Rounds until stop	$\|A_{calculated} - A_{true}\|_1$ /(relative error %)	Sent messages	Received messages	Message loss ratio	Nodes	Hop Count
60x60	18	5.4(4.5%)	128	85	2	8	20
120x120	18	12(5%)	128	85	2	8	20
300x300	18	34(5.16%)	128	84	2	8	20
600x600	18	58(5.16%)	128	84	2	8	20
900x900	18	80(5.22%)	128	84	2	8	20
1200x1200	18	98(5.54%)	128	83	2	8	20
60x60	146	21.5(7.77%)	3876	1201	3	27	40
120x120	147	23(8.51%)	3810	1169	3	27	40
300x300	147	87(9.66%)	3801	1156	3	27	40
600x600	147	270(10%)	3746	1121	3	27	40
900x900	148	379(10.33%)	3678	1079	3	27	40
1200x1200	148	492(13.66%)	3510	989	4	27	40
60x60	951	49.5(20.62%)	55867	8408	7	64	100
120x120	951	105(21.87%)	55811	8367	7	64	100
300x300	952	307(25.62%)	55798	8360	7	64	100
600x600	952	619(25.79%)	55790	8358	7	64	100
900x900	952	933(25.91%)	55796	8360	7	64	100
1200x1200	953	1320(27.5%)	55750	8330	7	64	100

Table 5: Forwarding approach for different hop counts, matrix and network sizes

In Table 5 we can recognize a relatively similar relative error within the same network size. The message loss ratio is also small in comparison with the other approaches. For a rising network size we have also increased the number of maximal hops of forwarded messages (Hop Count). Figure 10 shows the number of received messages for each node. It demonstrates that the consumed message ($consumed \neq forwarded$) distribution over 27 nodes varies highly. Such a message distribution is a direct cause of the underlying network topology. Nodes with less connections have a smaller probability to acquire messages from the network. This behavior influence of course the final convergences value of a subnet.

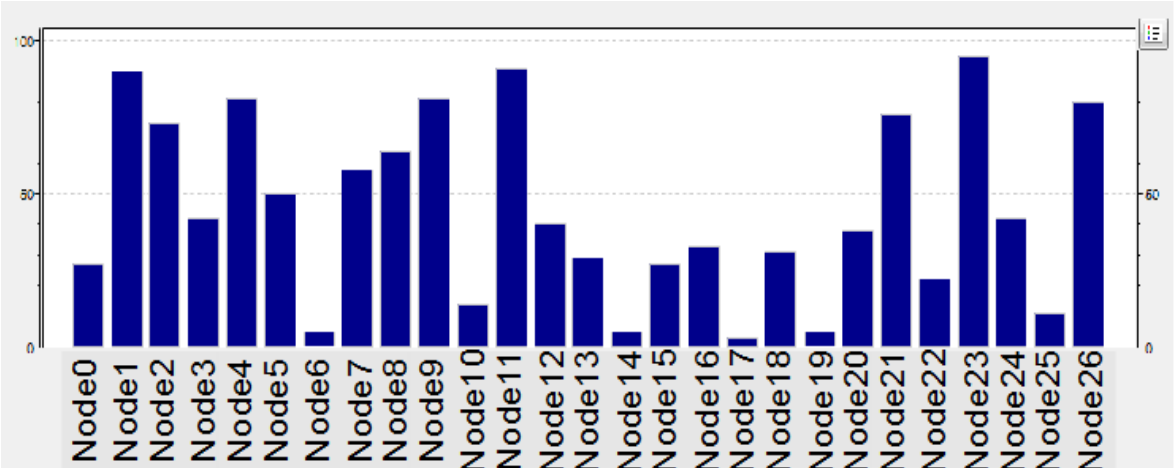


Figure 10 : Mobility approach message consumption of each node (Network size 27)

7.3.2 Mobility

The mobility approach is based on the assumption that nodes are constantly moving and subnet members will find each other in finite time. If a node receives a message it compares the message with its subnet address. If the subnet address matches the subnet address in the message header, it will consume the message. Otherwise it will reject and delete it. Thus, due to the fact that nodes are sending messages without acknowledgment from the other side a message loss is produced. For instance, node 1 is in the transmission range of node 4. Node 1 is in subnet 1 and sends a message to node 4, which is in subnet 2, thus node 4 will delete this message (message loss) because their subnet numbers are different. After a certain period, all nodes move again and the topology looks completely different. If node 1 is now in the transmission range of node 2 (which is also in subnet 1), node 1 might choose node 2 for communication. If node 2 receives the message, it will consume the message and calculate the new pair of x_i and w_i .

Matrix size	Rounds until stop	$\|A_{calculated} - A_{true}\|_1$ /(relative error %)	Sent messages	Received messages	Message loss ratio	Nodes
60x60	200	21(17.5%)	1478	96	15	8
120x120	201	33(13.75%)	1493	94	16	8
300x300	202	35(5.83%)	1494	97	15	8
600x600	203	39(3.25%)	1501	102	14	8
900x900	201	250(13.88%)	1446	92	16	8
1200x1200	198	150(6.25%)	1469	94	16	8
60x60	950	75.4(27.92%)	25084	766	33	27
120x120	958	163(30.29%)	25296	773	33	27
300x300	963	312(34.66%)	25368	782	32	27
600x600	966	500(27.77%)	25394	789	32	27
900x900	966	735(27.22%)	25334	779	32	27
1200x1200	973	1010(28.05%)	25311	743	34	27
60x60	1205	938.25(390.93%)	70187	1222	57	64
120x120	1211	1968(410%)	70293	1245	58	64
300x300	1198	5034(419.5%)	70195	1234	57	64
600x600	1203	10122(421.75%)	70302	1256	56	64
900x900	1220	15500(430.55%)	70336	1301	54	64
1200x1200	1226	20533(427.77%)	70358	1423	49	64

Table 6: Mobility approach for different hop counts, matrix and network sizes

Table 6 demonstrates the relation of message loss and accuracy for the mobility approach. It turned out that with the mobility approach it is very hard to achieve a good precision. Additionally, the message loss is relatively high (message loss ratio 58). The probability to send

a message to a subnet non member node, for this approach, is rather high. On the other hand, constant mobility is useful in order to achieve a fair message distribution for all nodes.

7.3.3 Forwarding and Mobility

The combination approach should unify the benefits of the forwarding approach as well as the mobiltiy approach. The forwarding strategy is statical and nodes with less connections are treated unfair. The mobility strategy distributes uniformly the number of consumed messages over the network but produces also a high message loss.

Matrix size	Rounds until stop	$\|A_{calculated} - A_{true}\|_1$ /(relative error %)	Sent messages	Received messages	Message loss ratio	Nodes
60x60	60	2.4(2%)	432	100	4	8
120x120	60	2.4(1%)	437	100	4	8
300x300	61	20(3.33%)	440	101	4	8
600x600	61	24(2%)	444	103	4	8
900x900	61	30(1.66%)	448	107	4	8
1200x1200	63	48(2%)	407	114	4	8
60x60	160	17(6.29%)	4349	464	9	27
120x120	173	24(4.44%)	5142	541	10	27
300x300	175	100(11.11%)	5517	580	10	27
600x600	181	210(11.66%)	5534	585	9	27
900x900	185	315(11.66%)	5545	587	9	27
1200x1200	190	458(12.72%)	5552	591	9	27
60x60	572	80.5(33.56%)	33224	1437	23	64
120x120	580	161.3(33.6%)	33256	1442	23	64
300x300	583	437(36.42%)	33301	1456	23	64
600x600	591	934(38.91%)	33336	1459	23	64
900x900	592	1432(39.77%)	33378	1483	23	64
1200x1200	589	1934(40.29%)	33365	1471	23	64

Table 7: Forwarding and mobility approach for different hop counts, matrix and network sizes

Table 7 shows, in comparison with Table 6. a faster convergence for all nodes. The ratio of lost messages is also smaller for the combination approach. But it has to be mentioned that the message loss ratio is not so good as in Table 5. However, we achieved a smaller error for small networks with the combination approach. This is the result of a better message distribution over the nodes because of the mobility in the network. That is to say, in case of the forwarding approach some nodes or subnets have an unfair static position (with no or a

small number of connections to the other nodes) in the network, thus they receive less messages than other nodes. So, few very accurate subnets with a very high message consumption cannot correct the error made on some remotely separated member nodes. The combination strategy distributes the messages uniformly over the nodes. The probability that all nodes have a high message consumption is rather high, if the nodes are moving with constant speed. Figure 11 demonstrates the consumed message distribution of the combination strategy. A consumed message distribution is the total number of received and processed messages (without the forwarded messages) of each node.

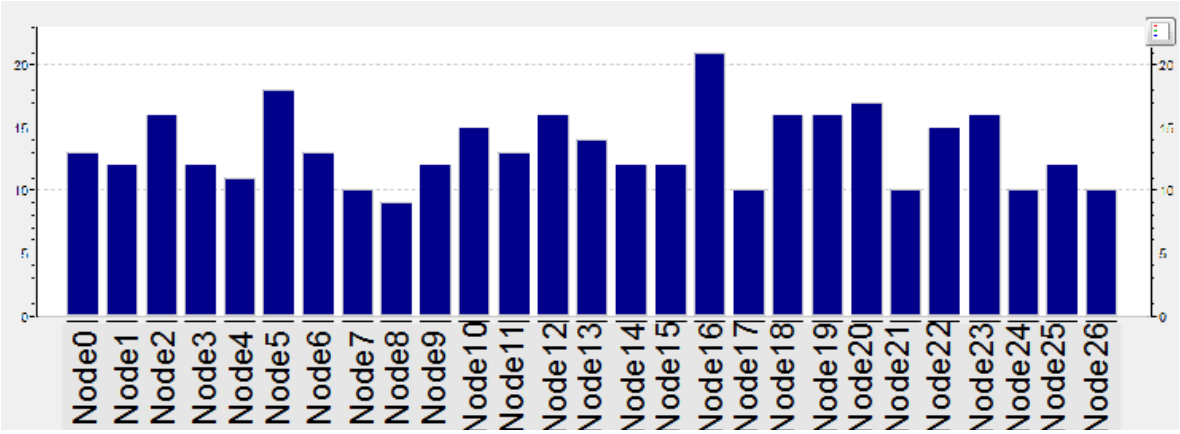
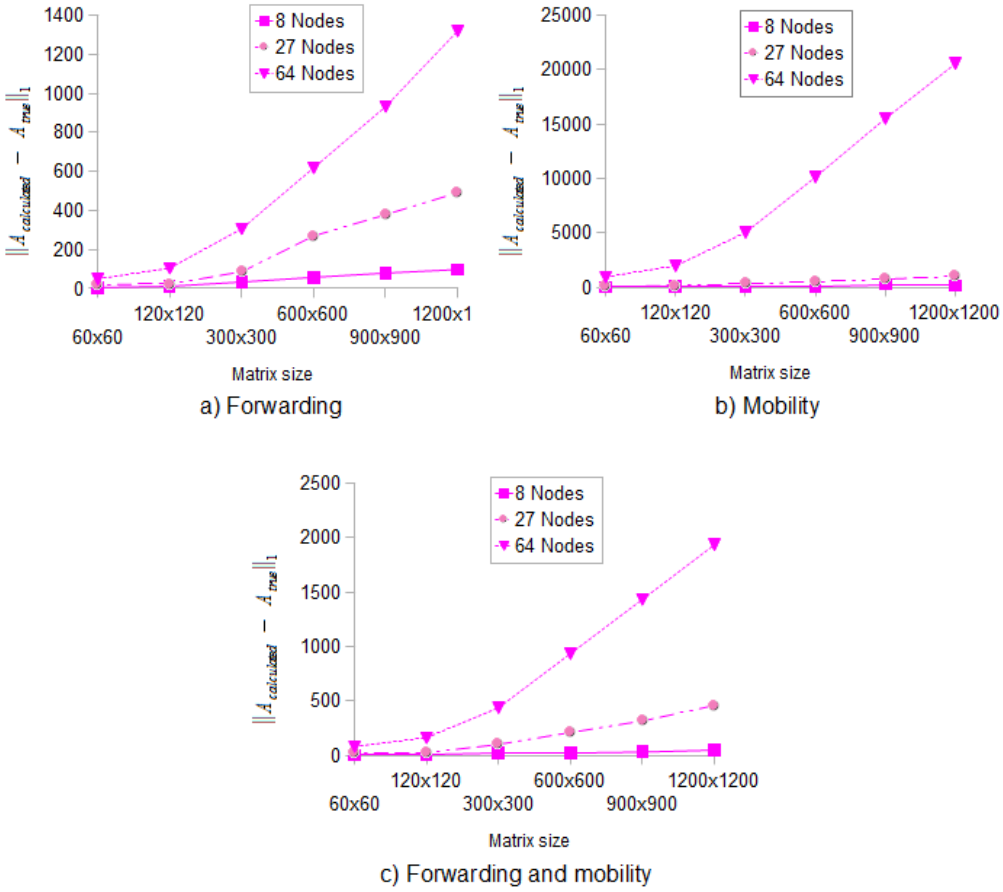


Figure 11 : Combination approach message consumption of each node (Network size 27)

The following Figure 12 shows the comparison between the accuracy and problem sizes for the three aggregation strategies. As we can see, the accuracy is relatively the same for a 50x50 and a 1200x1200 matrix. But the accuracy is decreasing steadily if the matrix is distributed on a larger number of nodes. The mobility approach particularly shows a huge decline of accuracy for a network of 64 nodes. In fact, under certain circumstances, the relative error is about 400%, which is a completely different matrix than the true resulting matrix.



Mobility update interval: 200 [s]	Mobility speed: 0.5 [mps]

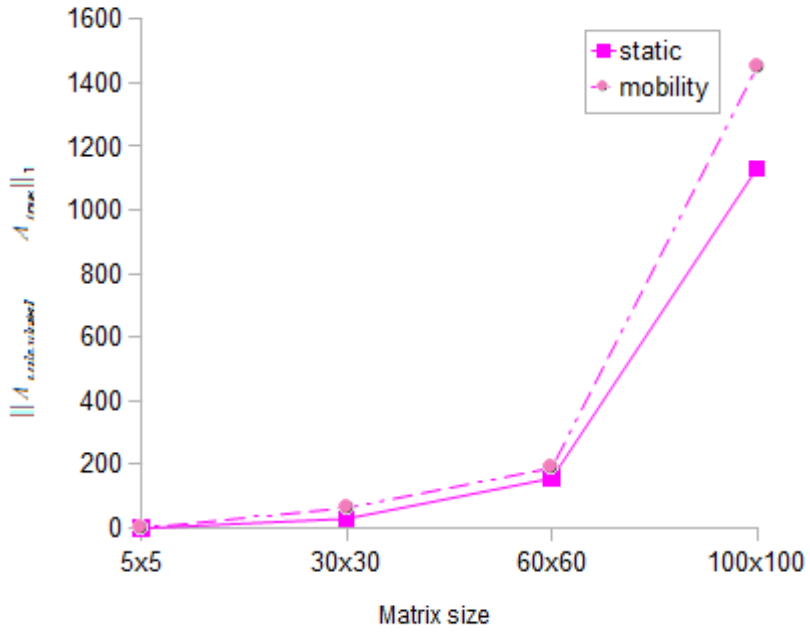
a) In this figure we have analyzed the influence of the problem size on accuracy of the forwarding approach. As it is depicted in the figure, the absolute error is growing proportionally with the matrix size. The relative error is approximately the same for all problem sizes for a certain network size (Table 5). We can also recognize that with rising network size, the error is in a different order.
b) Here we can observe a huge loss of accuracy for all network sizes. Nevertheless, the absolute and relative errors are behaving similarly to the rising problem sizes as in a). The huge inaccuracy is influenced by the frequent message loss in the simulation.
c) This figure shows the combination of the forwarding and the mobility approach. In fact, for small networks it outperforms both previous approaches.

Figure 12 : Accuracy comparison with different matrix and network sizes between the matrix multiplication approaches

Compared to the forwarding or mobility approach, the combination of both approaches delivers the best results for a small number of nodes. It combines the benefits of the forwarding and mobility approach. All nodes have a fair message consumption (approximately the same) because of the mobility, whereas the message forwarding avoids such a large message loss as in the mobility approach.

7.3.4 Column/Row strategy

The Column/Row approach is an altogether different strategy. In order to avoid subnets we distribute more information on one node. Every node holds a column from the first matrix and a vector from the second one. Figure 13 shows the absolute error for this approach. We executed all simulations with a static and mobile assumption. We have chosen smaller problem sizes for this strategy, in order to perform the aggregation in reasonable time. Thus, if the matrix size is larger than 100x100, the simulation runs up to half an hour.



Mobility update interval: 200 [s] Mobility speed: 0.5 [mps]

Figure 13 : Comparison of the accuracy between a static and mobile Column/Row strategy

As we can see, such an assumption also suffers from the message loss. But, here we have in fact another hazardous influence. The classical PSA sends a value and a weight in each round, thus, in the column/row approach, every row represents an independent value. Let us assume that two nodes perform the message exchange in a certain round. First, both nodes randomly pick a row and the appropriate weight from the weight vector. Then they send half of all the entries in the row and half of the appropriate weight. Thus, the first node probably picks a different row than the second node. A loss of synchronization is produced due to the fact that we are not simultaneously sending the same value in a certain round. For small matrices the probability that two nodes pick the same value is much higher and that ex-

plains the high accuracy for small matrices. But for large matrices this randomness influences the convergences speed as well as the accuracy. The nodes adjust their values in each round until they store the same value, but this value can be different from the true value. So, the benefit of this strategy is also that all nodes store, at the end of the simulation the same value $A_{calculated}$. But this result is under certain circumstances (especially for large problem sizes) far away from the real product matrix A_{true} . In other words, the error in the calculated matrix is large.

Matrix size	Rounds until stop	$\|A_{calculated} - A_{true}\|_1$ /(relative error %)	Sent messages	Received messages	Message loss ratio	Nodes
5x5	171	0(0%)	680	610	1	5
30x30	1133	30(3.33%)	33960	27079	1	30
60x60	9345	157(4.36%)	487456	389552	1	60
100x100	12342	1131(11.31%)	845999	634566	2	100

Table 8: Static network column/row strategy for different matrix and network sizes

Matrix size	Rounds until stop	$\|A_{calculated} - A_{true}\|_1$ /(relative error %)	Sent messages	Received messages	Message loss ratio	Nodes
5x5	220	0(0%)	968	821	1	5
30x30	1523	64(7.11%)	44607	27793	2	30
60x60	7290	189(5.25%)	301940	135191	2	60
100x100	13971	1450(14.5%)	1271334	832433	2	100

Table 9: Mobile network column/row strategy for different matrix and network sizes

Table 9 shows the increase of message loss and decrease of accuracy, if the network is mobile. The mobility speed in all previous simulations was set to 0.5 [mps] and a mobility update interval of 200 [s]. In comparison with Table 8, Table 9 has already a larger message loss ratio, which also results in a higher error in the resulting matrix. All in all, in case of very small matrices the error in the resulting matrix is small. So we note that the relative error for this strategy is proportional to the matrix size. The column/row approach is of course more accurate but has the disadvantage of performing a double random selection. Namely, first a random row from the approximation matrix has to be selected, and second the communication partner from the neighborhood. Such a behavior highly influences the convergence speed. Especially in case of large matrices the simulation slows down.

7.4 Deterministic subnet distribution and network topology

7.4.1 Static network

In this section we describe a static network with predefined subnet distribution and topology. Simulations show that loss of messages due to link failures does not influence the consensus agreement of the nodes. In the models presented so far, the nodes hold the same result at the end of the simulation. However, we have observed that random topologies influence the accuracy of the computed result matrix due to link failures. In order to tackle these problems we consider from now on a deterministic distribution. The sub-networks are separated and only subnet member nodes are connected with each other. In case of no mobility, each subnet performs a PSA without interference from the non member nodes. Figure 14 describes a static network with network size 27 and 9 subnets where each subnet has 3 members. The matrix size distributed on this network is 150x150, therefore every node holds a block matrix of a 50x50 matrix as the PSA value.

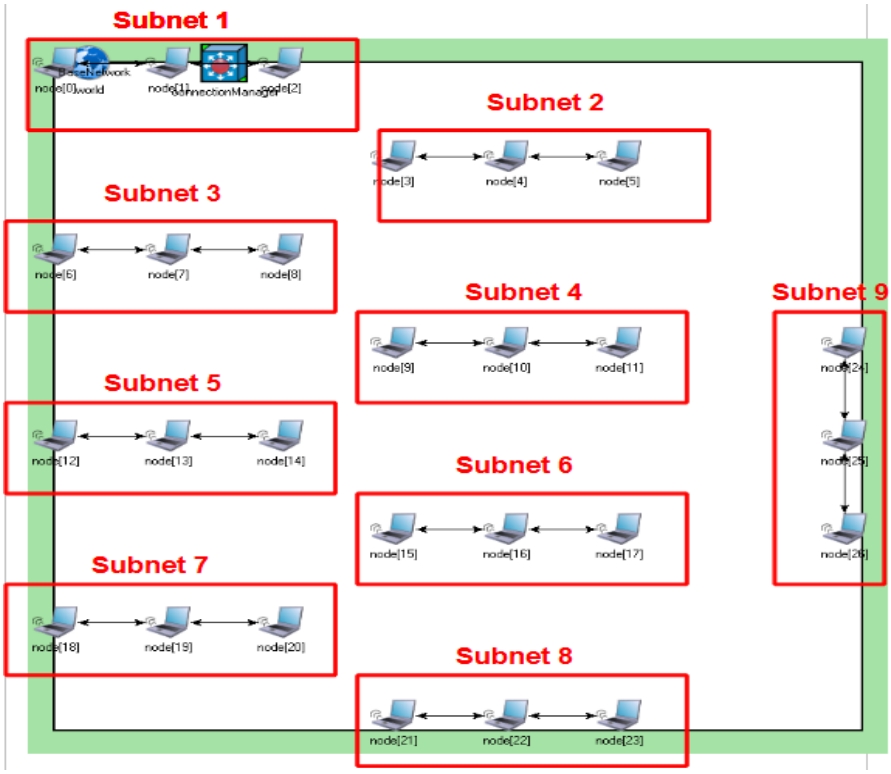


Figure 14: Initial distribution of a static network with size 27 nodes, no connections between non member nodes

After 48 rounds the PSA has achieved double precision. The improvement curve, as we can recognize from the Figure 15, is almost linear.

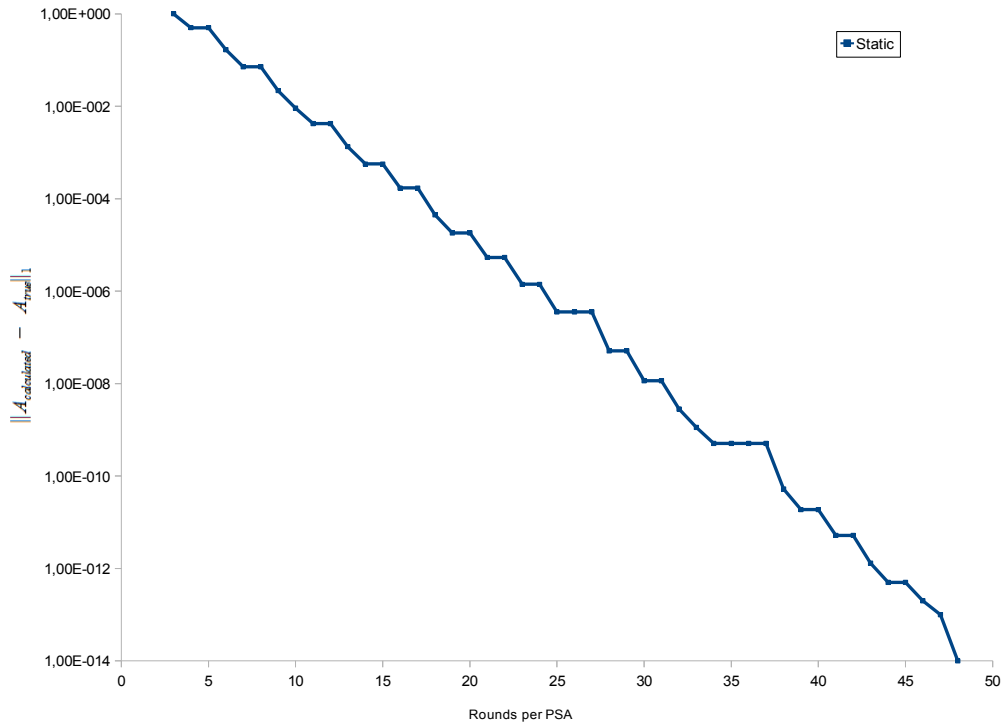


Figure 15: Accuracy of a static network with size 27 nodes, no dependability between non member nodes (matrix size 150x150)

This simulation also showed that the local PSA (on each subnet) works properly. Generally, such a behavior is the consequence of a very high message delivery rate. Additionally, there is no interference between the subnets, which means that every time a node chooses a neighbor, it is for sure a subnet member node. In other words, all subnets perform a classic PSA without forwarding and mobility, where the value of the PSA is a matrix. At the 48th round, the total number of messages that are sent is 1323 and the number of correctly delivered messages is 1283, thus this is a small message loss of 3.02%.

7.4.2 Forwarding

Let us assume that the network is static and all nodes are connected to this network but they are still distributed in such a manner that subnet members build connected groups. The border nodes of such groups could also have connections to other non member nodes. The network topology of such a network is represented in Figure 16. Nodes still forward messages to random nodes expecting to find a node with the same subnet number. Meaning that subnet nodes build virtual connections between two nodes separated by a certain num-

ber of non member nodes. If this were not the case, the forwarding becomes a possible source of message loss in a network. In such a network, the border nodes of the subnet groups could incidentally choose a non member node. In addition, they have always at least one subnet member as neighbor. So, there is definitely a probability that a message sent outside the group, finds the route back back to the group, but there are no guarantees.

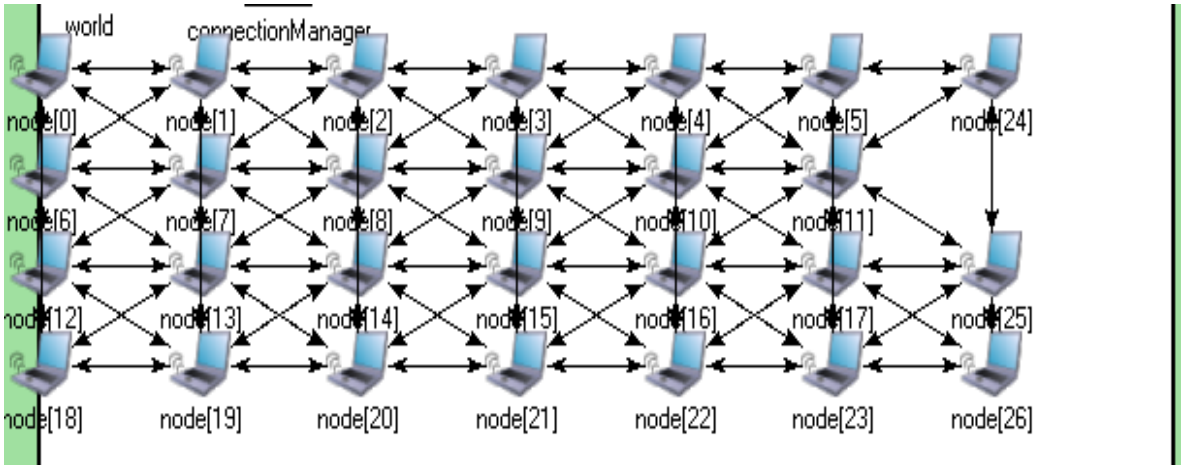


Figure 16: Initial distribution of a fully connected network 27 nodes with high dependability between subnets

Because of the hot-potato protocol, the nodes attempt to forward the messages as fast as possible, sending the messages to a randomly chosen neighbor. The hop count in Figure 17 is the maximal number of forwardings of one message (see chapter 4.2.1).

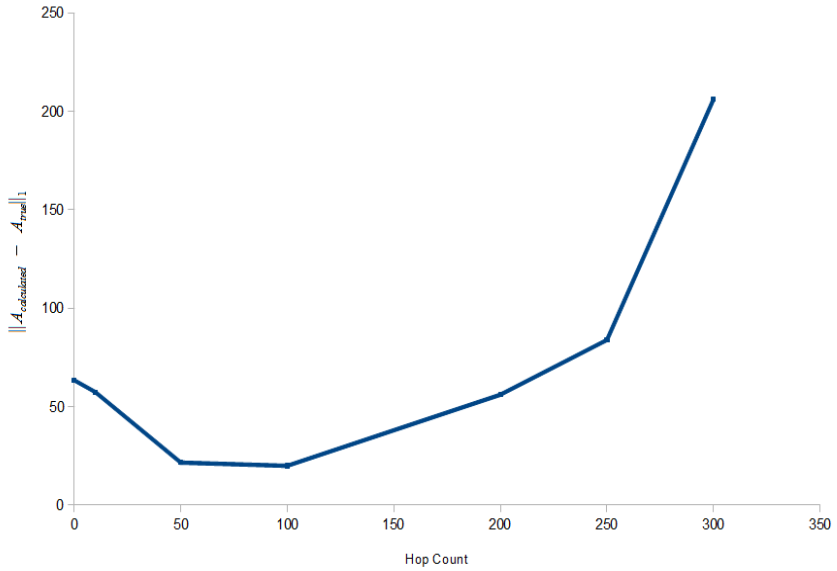


Figure 17: Relation between hop count and accuracy of a network with high dependability between subnets (matrix size 150x150)

The absolute error increases abruptly after setting the hop counter limitation higher than 100 hops. The reason for this is in fact the limited number of message in the message queue of each node. In Figure 17, the queue limit is set to 100. If the message queue is full, the following messages will be rejected, which explains the behavior in Figure 17. It turns out that nodes reject messages when the message queue is full. Such a behavior increases the message loss and influences the accuracy. Nodes have no mechanism to recognize if one node is busy and no longer able to consume or forward messages.

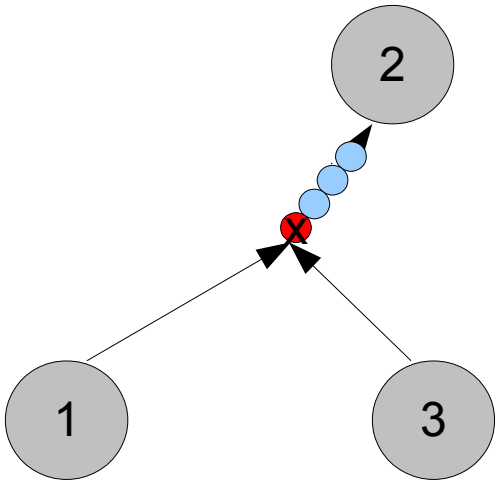


Figure 18: Illustration of a limited message queue

In Figure 18 we can see 3 nodes; node 1 and node 3 are sending messages to node 2. Hence, the message queue of node 2 is full, thus all following messages will be rejected. This let us to the idea to perform the simulation with different queue sizes and compare the results. Figure 19 describes this behavior.

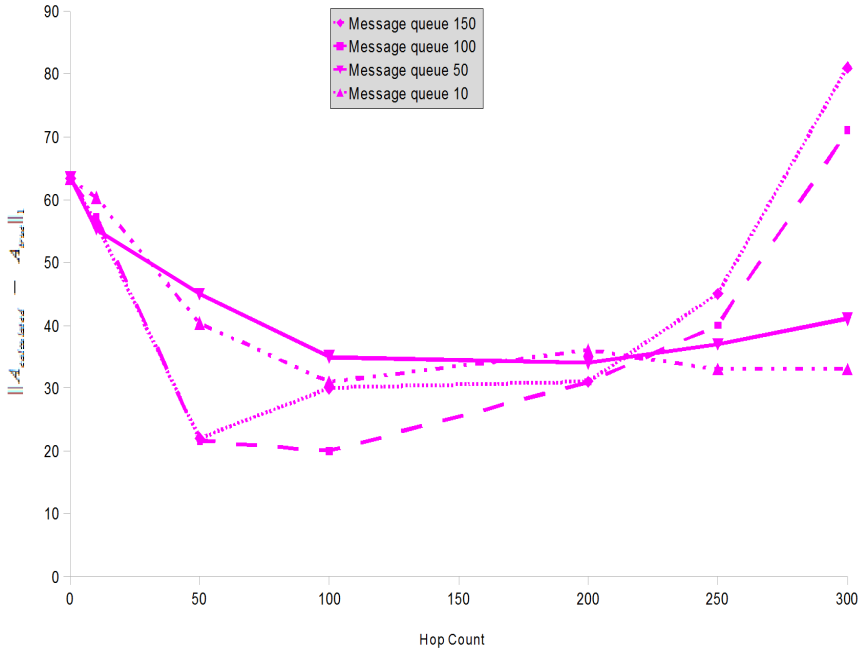


Figure 19: Relation between accuracy and hop count for different message queue sizes (matrix size 150x150)

It is noticeable that after a hop count of 200 messages, the accuracy is worsening abruptly. The accuracy is also irregular after a hop count of 200 messages. Namely, larger message queues have no more influence to the accuracy. We have no explanation for such a behavior, but our assumptions are that the MiXiM protocol cannot handle such a high message traffic on the topology in Figure 16.

The forwarding approach causes loss of accuracy of the computed matrix. Such an inaccuracy is a product of the very high message loss. There is no strategy to overcome such a behavior if we use the hot-potato protocol for forwarding messages. Nevertheless, the benefit of the hot-potato protocol lies in its simplicity and efficient resource utilization.

Additionally, we simulated a network with maximal 2 connections for each node and where all subnet members are neighbors as well. Figure 20 represents a network with a low number of connections. The nodes in the middle of the network are interfaces between subnets. Only these 9 nodes produce message losses sending messages outside of its subnet. Such a network topology certainly reduces the forwarding, but on the other hand difficults that the forwarded messages (produced by the nodes in the middle of the network) find the route back to the subnet.

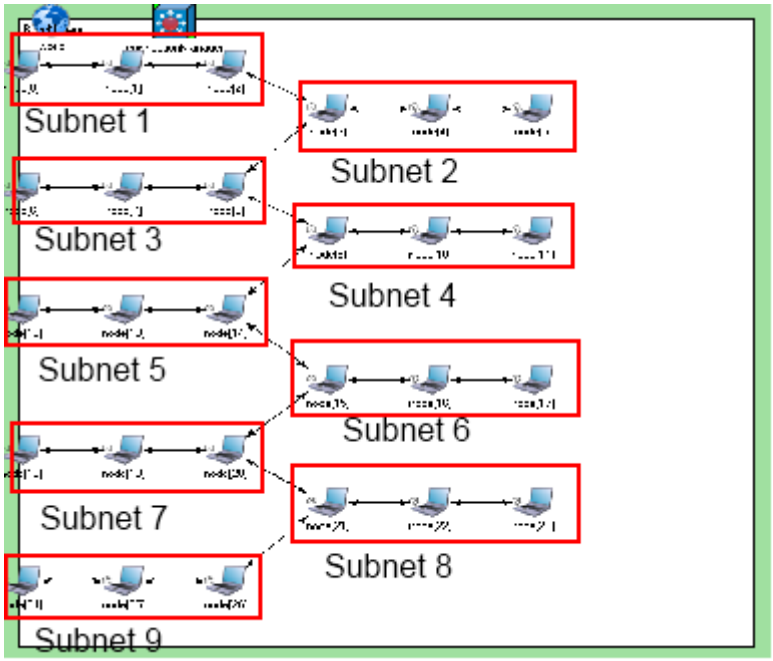


Figure 20: Topology of 27 node with low dependability between subnets

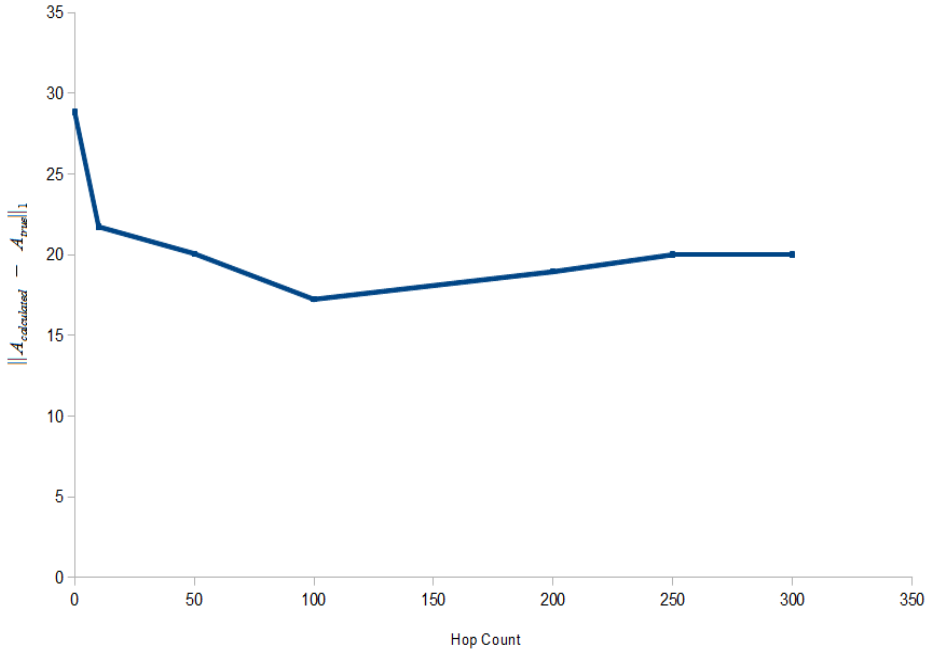


Figure 21: Relation between accuracy and hop count for a topology of low dependability between subnets (matrix size 150x150)

Figure 21 represents the accuracy with different number of hop counts. In comparison with Figure 17, Figure 21 provides a better accuracy for smaller hop counts but for 100 hop

counts it is still not satisfactory. All in all, forwarding with the hot-potato protocol converges every time but to a false value caused by the huge message loss.

7.4.3 Mobility

As we have seen in section 7.1.2, the mobility concern is still a great challenge for a WSN. It is an very difficult task for the routing protocol to keep track of the neighborhood and avoid any message loss, if nodes are moving constantly. Thus, we rather accept the fact that in such a system, there is always a certain message loss. In Figure 22 we analyzed a static network with a typical PSA performing the counting operation of the nodes in the network. Under such circumstances, after 3045 rounds we achieve double precision on every node. But how would the mobility influence such a system?

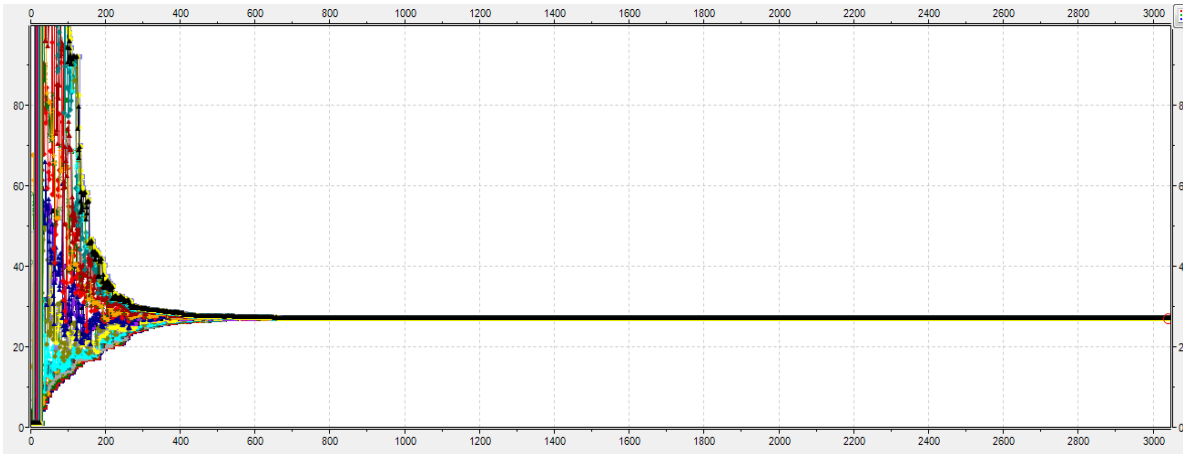
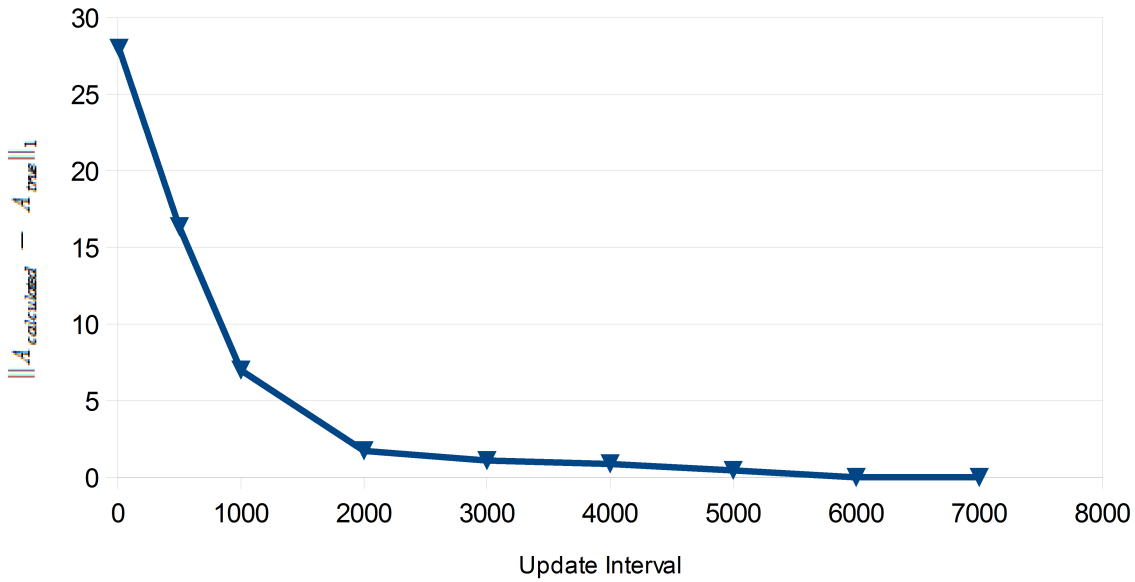


Figure 22: Counting 27 nodes in a static network x axis – number of rounds y axis – estimation value/weight

Figure 23: Counting 27 nodes with mobility x axis – number of rounds y axis – estimation value/weight

In Figure 23 we let the nodes move constantly very slow with 0.1 [mps]. We can still achieve a consensus but the inaccuracy of the estimation is very high; instead of the number 27 the nodes store about 47 at the end of the simulation. Note that this is a consequence of a high message loss (87.18 %). We can also observe that the advantage of mobility in such a network is the convergence speed. Namely, after 710 rounds all nodes in the network agree on a consensus. As a consequence of this, we decided to let the nodes move and then wait for a certain time and perform few rounds. This waiting time is the mobility update interval. In Figure 24 we simulated the PSA behavior on 27 nodes with different mobility update intervals. Finally, we can recognize there is a steady improvement of the accuracy by increasing the mobility update interval.



Mobility update interval: *x* axis Mobility speed: 0.5 [mps]

Figure 24: Relative error by increasing the mobility update interval of a PSA counting 27 nodes

Such a behavior led us to the conclusion that very large mobility update intervals achieve at least half of the double precision. The most important phase of a PSA is the beginning phase (Phase 1), where the differences of the estimation on each node are significantly large (see Figure 25).

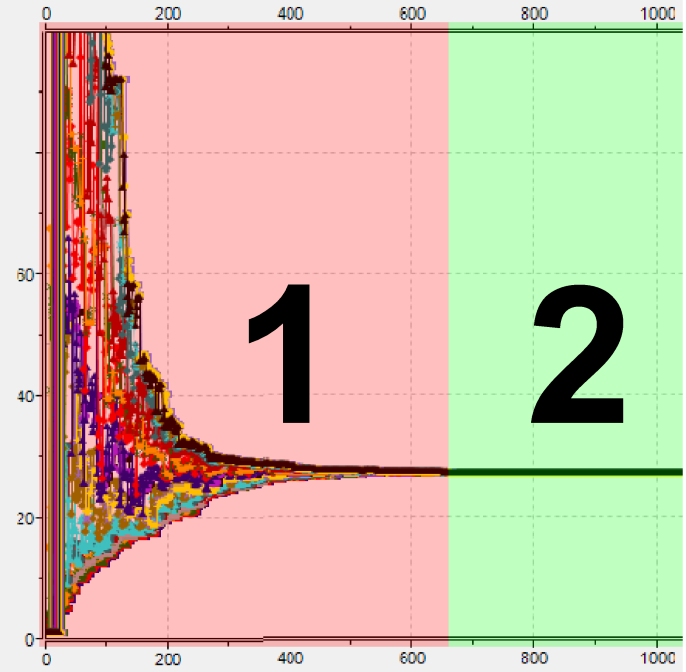


Figure 25: Illustration critical phase vs. robust phase of the PSA

In phase 1, also called the transient phase, message loss and round conflicts should be avoided in order to achieve a high precision. Thus, if the mobility update interval is relatively high it would be possible to avoid a high message loss in phase 1. Such a behavior is demonstrated in Figure 24.

7.4.4 Forwarding and Mobility

Further, we continue to investigate the matrix multiplication. We assume that the nodes are grouped as in Figure 20, they move very slowly and at relatively large time intervals. The initial network structure is in the transient phase as in Figure 20, but falls apart in a random structure at the end of the transient phase. The key idea behind this strategy is to let the network fall apart relatively late, in order to avoid message loss.

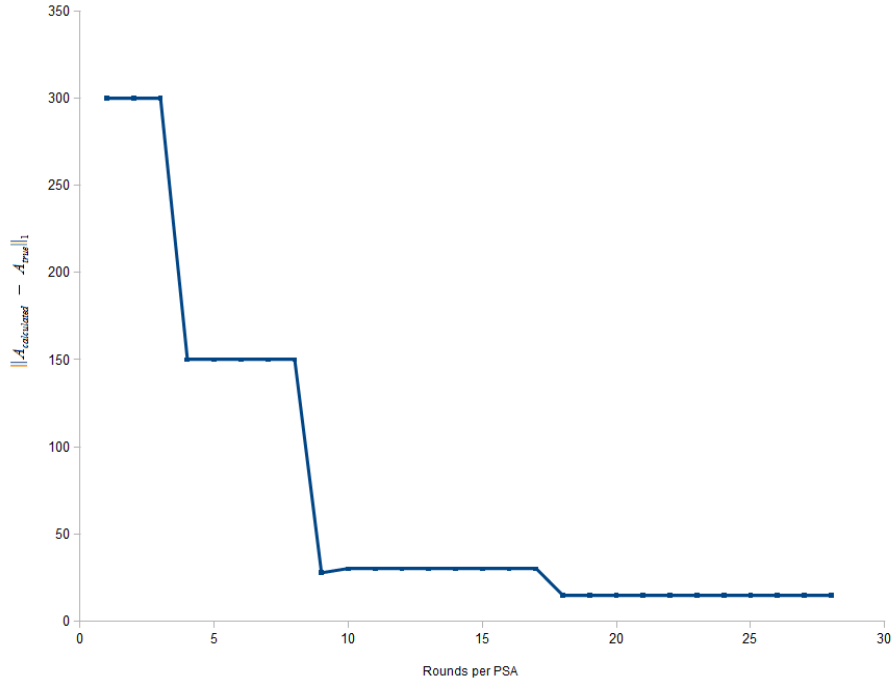
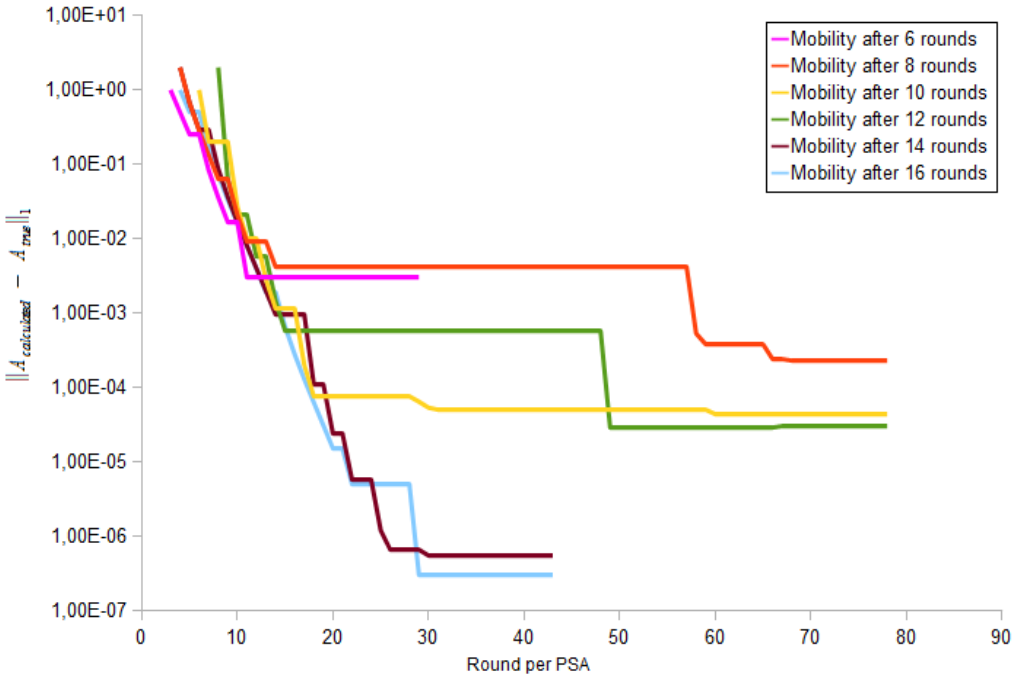


Figure 26: Accuracy for low dependability between subnets of a network with 27 nodes (combination approach with matrix size 150x150)

In Figure 26 convergence speed is fast, but the accuracy is still not satisfactory. Therefore we decide to let the subnets be separated (without connections to other subnets) in the transient phase as in Figure 14 and posteriorly allow them to move after a certain number of rounds.



Mobility update interval: 200 [s] Mobility speed: 0.5 [mps]

Figure 27: Accuracy for different immobility periods in the transient phase of 27 nodes (combination approach with matrix size 150x150)[1]

As we can see in Figure 27 mobility in the transient phase influences the accuracy significantly. If we let the subnets perform a local PSA with a small message loss in the first few rounds we can achieve almost half of double precision. Such a large improvement of the accuracy, in comparison with the previous simulations, is directly the product of avoiding forwarding and mobility in the first few rounds. Therefore, we realize that the classical PSA is rather sensitive to forwarding and mobility due to the high number of lost messages. Note that some simulations in Figure 27 need more rounds to achieve a consensus in all subnets (see Figure 27: red, green and yellow curve). This kind of behavior happens due to the random mobility in the simulations, whereas some subnet members are separated for longer time intervals. Hence, it is comparatively difficult to predict the convergence speed in such a model.

[1] Figure 15, 26 and 27 represent the trend of the absolute error on the first node of each subnet. The first node in the subnet is always the node whose weight equals number one.

7.4.5 Column/Row strategy

We also tested the Column/Row strategy with a deterministic distribution. Additionally, we delayed the mobility of the transient phase in order to achieve a higher precision. The initial network topology is shown in Figure 20. As we can see in Figure 13, the difference between the static and mobile approach of the Column/Row strategy is relatively small. Obviously, the accuracy in such a strategy cannot be more improved than the static approach. Thus, we have no more subnets; every node is able to consume messages from all other nodes in the network. But our intention was to adjust the results up to the accuracy of the static approach, due a small delay in the mobility approach. So, we distributed a 27x27 matrix on 27 nodes and let them move after 10 rounds.

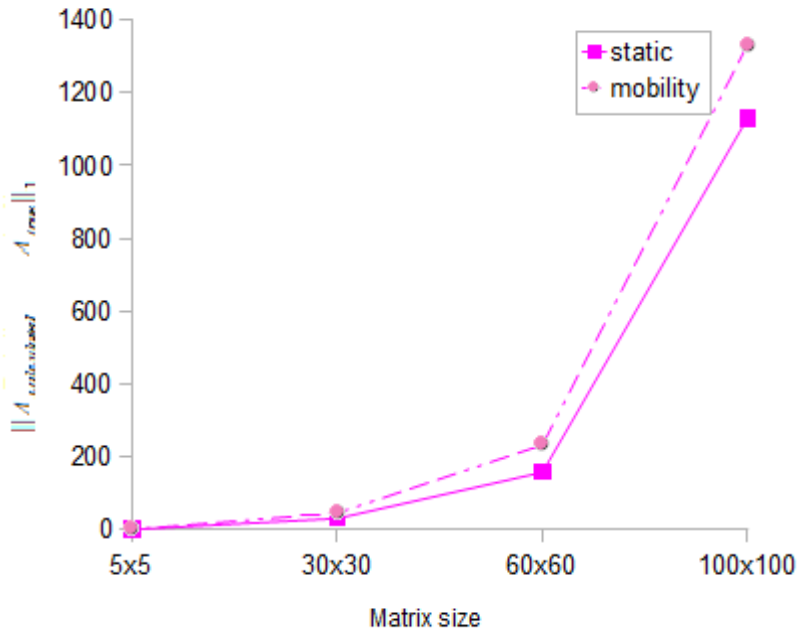


Figure 28 : Accuracy for the static/mobile Column/Row strategy for a deterministic distribution

As we can see in Figure 28, there is a small insignificant improvement in the aggregation in mobile model, if we compare it with the Figure 13. Nevertheless, the simulation is still inaccurate for large matrix sizes. The small improvement here is due to the stable transient phase (reduction of message loss due to immobility in the transient phase).

The accuracy of the static approach is the same as in in Figure 13. In this solution we have less options to improve the accuracy than in the subnets solutions because of the different row selection in each round.

8 Conclusion

The IT environment has changed rapidly in the last few decades. From huge computer blocks in the 70's and 80's, over small desktop PCs in the 90's we managed to arrive into the world of smart phones. So, while the decades before the millennium marked a time of localized computation, the new century brought a lot of new challenges for distributed systems. Especially with the comeback of Apple and the launching of the iPhone, the whole IT industry got a new impetus in the direction of distributed systems. It is to expect that this trend will continue and new distributed technologies will conquer the market. Our contribution in this direction are simple strategies for solving the matrix multiplication distributed on a mobile sensor network. We have analyzed and implemented 3 major approaches with simple considerations of forwarding messages and mobile subnet communication. We also analyzed 2 different approaches of distributing data. The first distribution approach stores a part of the matrix product on each node, where the second approach builds an approximation matrix on every node, and by performing the PSA, it aggregates all approximation matrices to the product matrix. It turned out that the PSA which we used for the aggregation, is extremely sensitive to message loss in the transient phase. As long as the network is static, we can guarantee a low message loss, and we can also define a convergence criteria. Otherwise, if the nodes are mobile, the task of tracking the neighborhood becomes really hard. If a receiving node moves outside the transmission range of a sending node, message loss is produced. It is also to mention that message loss, is not a reliable indicator for accuracy. A network could be highly reliable in the transient phase but extremely hazardous at the end of the simulation and could still achieve a high precision.

In our simulation, we implemented a very simple routing protocol in order to avoid message overheads and save resources on the nodes. Namely, our routing protocol learns from the previous behavior of the network. We are certainly aware of losing messages with this protocol but, on the other hand such an approach makes the simulation more stable. In section 7.1 we described randomly distributed networks. We have shown that it is rather difficult to achieve a high precision under such circumstances. The reason is, as we have described it, the large message loss, particularly in the transient phase. Both the approach of forwarding and mobility has a hazardous effect on the accuracy. Naturally, the forwarding

approach delivers better results in relation to the accuracy than the mobility approach, which is due to the high message loss within the mobility approach. However, the mobility approach does provide a fair message distribution on every node and over the subnets. The combination approach includes properties from both approaches and ensures a trade-off situation. In particular, for small networks (8 or 27 nodes) the combination approach outperforms the individual one. As we mentioned before, the Column/Row strategy is slower than the previous one but the advantage is that no more subnets are necessary within this approach. Every node can communicate with its neighbors without proving that it is in the same subnet. Further, it is important to mention that we are able to achieve a consensus in all simulations (with different errors of the resulting matrix). Also, from the aspect of initial distribution there is a difference between the subnet strategies and the Column/Row strategy. For the subnet strategies block matrices of both matrices have to be distributed on the network, thus in case of the Column/Row strategy one column from the first and one row from the second matrix are sufficient. Note that, consequently the Column/Row approach has different message and network sizes. Namely, for *NxN* matrices in case of the subnet strategies, N^3 nodes are required, whereas the Column/Row approach needs only N nodes. The message size for the subnet strategies is bound to the size of the block matrices but in case of the Column/Row the PSA message contains as value a vector with N elements and one number for the weight. Additionally, in section 7.2, deterministic ways of subnet distribution and network topologies have been investigated. Since random topologies are rather difficult to predict, we grouped the subnets. We simulated scenarios where the subnets are divided into a heap of member nodes (there is no connection to the non member nodes). And we have also analyzed the case of a high and low dependability between the subnets. It turned out that only if the subnets are divided in the very beginning of the aggregation, it is possible to avoid message loss and achieve a high precision in case of the combination strategy (forwarding and mobility). Therefore, we propose an initial idle phase for mobile networks, where the subnets are independent from each other and perform their local push sums. However, forwarding and mobility should not be abolished because of their importance for the convergence. The mobility in particular has a positive influence on the convergence speed. All subnet strategies, in comparison with the Column/Row strategy, show a better performance for deterministic distribution. But on the other hand, every node stores only a part of the resulting matrix, and the network size is always N^3.

9 Further work

With the PSA, we have focused on distributed matrix multiplication approaches. In order to deal with message losses, other aggregation algorithms should be used instead of the PSA for the distributed matrix multiplication . Also a variant of improved PSA, which minimizes or avoids the message loss, could be implemented. Secondly, there are many open questions concerning the routing protocol. It is still a open issue how to build a reliable routing protocol for WSNs. The PSA is not broadcasting messages but rather sending one message to a uniformly chosen neighbor. Therefore, every node has to track its neighborhood. This becomes an extremely hard task in mobile networks. And at the end, the hot-potato protocol could be improved, in order to send messages more reliably to subnet members.

10 References

[1] Karl Aberer. Unleashing the power of wireless networks through information sharing in the sensor internet. In Kay Rmer, Holger Karl, and Friedemann Mattern, editors, *EWSN*, volume 3868 of *Lecture Notes in Computer Science*, pages 3–4. Springer, 2006.

[2] Delphine Christin, Andreas Reinhardt, Parag S Mogre, and Ralf Steinmetz. Wireless sensor networks and the internet of things : Selected challenges. *Structural Health Monitoring*, 5970:31–33, 2009.

[3] Bi Song, Chong Ding, Ahmed Tashrif Kamal, Jay A. Farrell, and Amit K. Roy Chowdhury. Distributed camera networks. *IEEE Signal Process. Mag.*, 28(3):20–31, 2011.

[4] Thanavelu. V. Ashok.A, Harikrishnan.N. Sensor networks and aggregation using hybrid protoc. *Georgian Electronic Scientific Journal*, 1(2):25–35, 2010.

[5] Laukik Chitnis, Alin Dobra, and Sanjay Ranka. Aggregation methods for large-scale sensor networks. *ACM Trans. Sen. Netw.*, 4(2):9:1–9:36, April 2008.

[6] Christophe Guéret, Nicolas Monmarché, and Mohamed Slimane. Self-organizing ant-based information gossiping algorithm for p2p networks. In *IICS*, pages 450–461, 2010.

[7] Aleksandra Krkoleva, Vesna Borozan, and Panayiotis G. Romanos. Implementation of gossip algorithms in power systems. In *ICST E-Energy*, pages 20–28, 2010.

[8] Öznur Özkasap, Mine Çaglar, Emine Sule Yazici, and Selda Küçükçifçi. An analytical framework for self-organizing peer-to-peer anti-entropy algorithms. *Perform. Eval.*, 67(3):141–159, 2010.

[9] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, PODC '87, pages 1–12, New York, NY, USA, 1987. ACM.

[10] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 482–, Washington, DC, USA, 2003. IEEE Computer Society.

[11] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '98, pages 85–97, New York, NY, USA, 1998. ACM.

[12] Jamal N. Al-karaki and Ahmed E. Kamal. Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications*, 11:6–28, 2004.

[13] A. K. Dwivedi and O. P. Vyas. Article:network layer protocols for wireless sensor networks: Existing classifications and design challenges. *International Journal of Computer Applications*, 8(12):30–34, October 2010. Published By Foundation of Computer Science.

[14] Taewook Kang, Jangkyu Yun, Hoseung Lee, Icksoo Lee, Hyunsook Kim, Byunghwa Lee, Byeongjik Lee, and Kijun Han. A clustering method for energy efficient routing in wireless sensor networks. In *Proceedings of the 6th WSEAS International Conference on Electronics, Hardware, Wireless and Optical Communications*, EHAC'07, pages 133–138, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

[15] John C. Whitson Richard A. Burne, Ivan Kadar and Eitan R. Eadan. Self-organizing cooperative ugs network for target tracking.

[16] Alan D. Amis, Ravi Prakash, Thai H.P. Vuong, Dung T. Huynh, Thai H. P, Vuong Dung, and T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *in Proceedings of IEEE INFOCOM*, pages 32–41, 2000.

[17] A. Bruce McDonald and Taieb Znati. A mobility based framework for adaptive clustering in wireless ad-hoc networks. *IEEE Journal on Selected Areas in Communications*, 17:1466–1487, 1999.

[18] O Younis and S Fahmy. Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, 2004.

[19] Ping Ding, Joanne Holliday, and Aslihan Celik. Distributed energy-efficient hierarchical clustering for wireless sensor networks.

[20] Allan Borodin, Yuval Rabani, and Baruch Schieber. Deterministic many-to-many hot potato routing. *IEEE Transactions on Parallel and Distributed Systems*, 8, 1997.

[21] Joanna Geibig and Dirk Bradler. Self-organized aggregation in irregular wireless networks. In *Wireless Days*, pages 1–7. IEEE, 2010.

[22] A.D. Sarwate and A.G. Dimakis. The impact of mobility on gossip algorithms. In *Proceedings of the 28th Annual International Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brazil, April 2009.

Curriculum Vitae

Personal Information-

Name	Edhem Brakmić
Degree	Bakk. techn.
Date of birth	09.07.1986
Place of birth	Zenica (Bosnia and Hercegovina)
E-mail	ebrakmic@gmail.com
School- and professional Education	2009-2012 University of Vienna - Field of study: Scientific Computing 2005-2009 University of Vienna - Field of study: Computer Engineering 2001 – 2005 Gymnasium in Zepce (Bosnia and Herzegovina) 1998 – 2001 Primary school in Zenica (Bosnia and Herzegovina) 1993 – 1997 Primary school in Gehlenbeck (Germany)
Mother tongue	Bosnian (Serbo-Croatian)
Languages	German (excellent written and oral skills) English (good written and oral skills) Swedish (elementary) Turkish (elementary)

Zusammenfassung

Eine der häufigsten Operationen im Bereich der digitalen Bildverarbeitung ist die Matrix Multiplikation. Parallele Algorithmen und Konzepte sind allgemein bekannt und gut entwickelt, aber im Falle von dezentralisierten Systemen gibt es immer noch einen Mangel an neuen Ansätzen. Wir haben in dieser Arbeit verschiedene Ansätze der Matrix Multiplikation auf einem mobilen Sensornetzwerk vorgeschlagen. Der Aggregationsprozess über die Knoten basiert auf dem Push Sum Algorithmus. Wir haben 4 Strategien zur Matrix Multiplikation auf einem mobilen Sensornetzwerk eingeführt. Die Ergebnisse haben gezeigt das es verhältnismäßig schwer ist, für zufällige Topologien und ständige Mobilität eine hohe Genauigkeit zu erreichen. Deshalb haben wir verschiedene deterministische Topologien, mit hohen Perioden der Bewegungslosigkeit in der transienten Phase des Aggregationsprozesses untersucht. Unter bestimmten Bedingungen, konnten wir den Nachrichtenverlust minimieren und fast die Hälfte der Double Precision erreichen.