FACULTY OF FUNDAMENTAL PROBLEMS OF TECHNOLOGY
WROCŁAW UNIVERSITY OF TECHNOLOGY

ESTIMATING DIAMETER IN AD-HOC NETWORKS

BARTOSZ CHODOROWSKI

M.Sc. thesis written
under the supervision of
prof. dr hab. Jacek Cichoń

WROCŁAW 2012

Abstract

The goal of this paper is to analyse simple algorithms that estimate the diameter of a connection graph in ad-hoc networks. In particular, time and energy complexity are considered. We also measure how the algorithm is robust by analysing the number of points of failure. Selection of an algorithm is strongly associated with a trade-off between time needed, energy consumed and reliability.

In this work two approaches have been defined and studied. We have also performed a number of computer simulations that can help make a decision which algorithm to use in a specific application.

The main contribution of this work is the introduction of **Breath First Spanning Tree Building with 2 Phases** algorithm. The simulations on random geometric graphs of size from 100 to 1000 have shown that the average relative error of this method varies from 2% to 3% with very low energy consumption. We also proved that time complexity of the method is $O(D)$ (where D is the diameter).

Contents

1 Introduction

Distributed computing, along with wireless ad-hoc networks, have been very popular topics in recent years due to a need of mobile, robust communication. Usage of infrastructure based networks (like GSM or WiMAX) is impossible in some scenarios — such as military communication on the battlefield. Such systems demand an extremely high mobility of every component as well as an elimination of single points of failure (system still has to work even though some of a key components fail).

Ad-hoc networks are the solution here. In contrast to infrastructure based communication, ad-hoc networks consist only of equal nodes without given hierarchy. There is no base station or access point, so every node has to be aware of a situation in the network and perform routing operations for the others.

Performance of many known algorithms used in ad-hoc networks depends on a diameter of a connection graph. When nodes possess an information about the diameter, they can alter their algorithms to improve performance and lower energy costs. Therefore, it may be useful to quickly estimate the diameter before running more complex communication algorithms.

1.1 The Problem and Related Work

Exact computation of the diameter, especially in distributed systems like ad-hoc networks, is not a trivial task. Fortunately, a good estimation of this value is sufficient in many cases.

Off-line estimation of the graph diameter is quite well studied [1,7], whereas there is little number of methods that work in ad-hoc networks.

Knowing that building breadth first spanning trees in distributed systems is relatively easy, which we shall see later, we can examine depth of that tree and use it as an estimator of the diameter. The relative error of such approximation is 50% in the worst case. Using certain tricks we can increase accuracy and reliability of this method.

The second known alternative is Extrema Propagation [4]. This probabilistic technique was adjusted so it can estimate the diameter [3]. This technique happens to be quick and robust.

1.2 Disposition

In section 2 the most important basics are discussed. In particular, essential elements of graph theory, distributed computing and ad-hoc networks are covered. Section 3 includes description and analysis of four methods with particular regard to pointing out their the advantages and disadvantages. Section 4 presents how computer simulations, used in this paper, have been designed and implemented. Section 5 comprises obtained experimental results. Summary can be found in the section 6.

2 Background

This section reminds the basic definitions and concepts connected to ad-hoc networks. They are essential to perform an analysis of algorithms in the section 3.

2.1 Basic Concepts

Elements of graph theory, computer networking, distributed computing and ad-hoc networks have been summarised and presented in this subsection, so one can remind them and familiarize himself with definitions which are used later in the work.

2.1.1 Graphs

We start with a classic definition of a graph and its basic properties, which are consistent with (or directly taken from) [11]. As in the aforementioned book, we use word 'family' as a synonym for 'multiset' — a generalization of set in which elements can occur more than once.

Definition 1 (Graph, undirected graph). *A graph $G = (V,E)$ consists of a non-empty finite set V of elements called **vertices**, and a finite family E of unordered pairs of (not necessarily distinct) elements of V called **edges**. We call V the **vertex set** and E the **edge family** of G.*

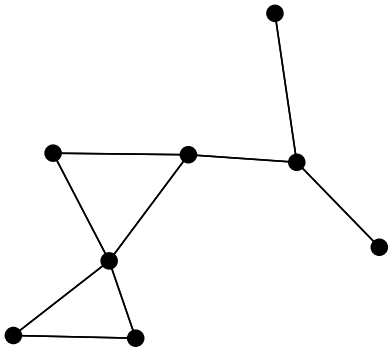

Figure 1: Undirected graph

Definition 2 (Simple graph). *G is a simple graph if the arcs of G are all distinct, and if there are no 'loops' (edges of the form (v,v)).*

Definition 3 (Directed graph, digraph). *A directed graph, or digraph, $G = (V,A)$ consists of a non-empty finite set V of elements called **vertices**, and a finite family A of ordered pairs of elements of V called **arcs**. We call V the **vertex set** and A the **arc family** of G.*

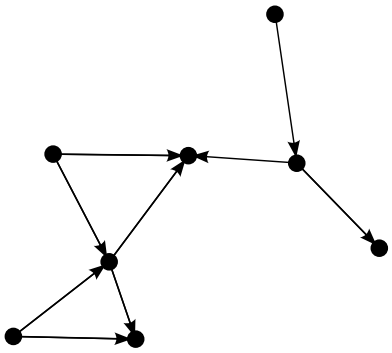

Figure 2: Directed graph

Definition 4 (Outgoing arcs). *We define a set $\overrightarrow{A_u}$ of outgoing arcs of a vertex $u \in V$ in the following way:*

$$\overrightarrow{A_u} = \{a \in A : (\exists v \in V)(a = (u,v))\}$$

Definition 5 (Incoming arcs). *We define a set $\overleftarrow{A_u}$ of incoming arcs of a vertex $u \in V$ in the following way:*

$$\overleftarrow{A_u} = \{a \in A : (\exists v \in V)(a = (v,u))\}$$

Definition 6 (Simple digraph). *G is a simple digraph if the arcs of G are all distinct, and if there are no 'loops' (arcs of the form (v,v)).*

Definition 7 (Symmetric digraph). *A directed graph G is called a symmetric if, for every arc in G, the corresponding inverted arc also belongs to G.*

Please note the following fact for it will be used in the simulator application to justify modeling graphs with a special case of digraphs.

Fact 1. *A symmetric simple directed graph $G = (V,A)$ is equivalent to a simple undirected graph $G' = (V,E)$, where pairs of inverse arcs in A correspond one-to-one with the edges in E; thus the number of edges in G' is $|E| = |A|/2$. [10].*

From now on to the end of subsection about graphs, we give the definitions for directed graphs only, because definitions for undirected graphs are intuitive and can be obtained similarly.

Definition 8 (Walk). *A walk is a finite sequence of arcs of the form $(v_0,v_1),(v_1,v_2),\ldots,(v_{m-1},v_m)$. We sometimes write this sequence as $v_0 \to v_2 \to \cdots \to v_m$, and speak of a walk from v_0 to v_m. The number of arcs in a walk is called its **length**.*

Definition 9 (Trail). *A walk in which all the arcs are distinct is a trail.*

Definition 10 (Path). *A trail in which all vertices are distinct (except, possibly, $v_0 = v_m$), then the trail is a path.*

Definition 11 (Connectivity). *A digraph is connected if it cannot be expressed as the union of two digraphs, defined in the obvious way.*

Definition 12 (Strong connectivity). *A digraph is strongly connected if, for any its two vertices v and w, there is a path from v to w.*

2.1.2 Diameter

The diameter is the key concept in this work.

Definition 13 (Shorthest path). *Let $d(u,v)$ denote a length of the shortest path between u and v.*

Definition 14 (Diameter). *The diameter D of a digraph $G = (V,A)$ is defined as follows:*

$$D(G) = \max_{u,v \in V} d(u,v)$$

2.1.3 Networks

Graphs are very general structures that have many applications and interpretations. In computer networking, we think about graph as of a representation of a computer network.

We will now give an interpretation we need to a directed graph and turn it into a computer network.

Nodes represent devices who want to participate in the communication, whereas arcs denote direct communication channels. In other words, device represented by a node u can transmit data directly to a device represented by a node v only when there exists an arc (u,v).

In our model we assume, that each arc (u,v) contains two buffers for data that is transmitted from u to v.

Definition 15 (Network). *Pair $N = (V,A)$ is called a network, where V and A are the same as in definition 3. Elements of V are also called **devices** or **processors** and elements of A are sometimes called **communication channels**. For each $a \in A$, we define $Ibuf_a, Obuf_a \in M \cup \{null\}$, where M is a set of all possible messages that devices can send through a communication channel.*

2.1.4 Distributed Computing

In this subsection we define how messages are being transmitted through the network. The ideas and definitions contained here are consistent with the majority of works on similar subjects, in particular, model given in this work slightly generalizes the model given in [2].

Informally we can say, that each device in the network runs a program (performs an algorithm). It is aware of its neighbourhood, that is, it knows of its all incoming and outgoing arcs. The algorithm is being performed in **steps**. In each step, the device can receive data through its incoming arcs, perform necessary computation and send data through its outgoing arcs.

We will rephrase the previous paragraph in a formal way. Messages transmission takes place as follows: a device $u \in V$, when performing its algorithm, can write only to those $Ibuf_a$ for which $a \in \overrightarrow{A_u}$ and read from those $Obuf_a$ for which $a \in \overleftarrow{A_u}$. Let us assume that a is an outgoing arc of u and an incoming arc of v. Node u can send data to v by changing value of $Ibuf_a$ — it places data in the communication channel's buffer. A **delivery** occurs, when arc a moves data from $Ibuf_a$ to $Obuf_a$, simultaneously setting $Ibuf_a$ to $null$. Then v can receive the message by reading $Obuf_a$ and setting it back to $null$.

2.1.5 Synchronous and Asynchronous Communication

Definition 16 (Synchronous and asynchronous communication). *If the execution of devices' algorithm steps and delivery of messages is done in an order, such that we can divide it into two phases that follow each other:*

1. *all devices perform their algorithm steps,*

2. *all arcs deliver messages from input to output buffers,*

then we say, that communication is synchronous. Otherwise, if the order is arbitrary, then we call such communication asynchronous.

In general, computer networks that exist in reality are more similar to asynchronous model of communication. Nevertheless, synchronous model is often used in algorithms design and analysis due to the convenience.

2.1.6 Ad-Hoc Network

"The word 'ad-hoc' is from Latin and means 'for this (only)'. In the case of computer networks, the ad-hoc networks mean wireless network without infrastructure, they can be called spontaneous networks.

One way to understand ad-hoc networks is by comparing them with infrastructure based wireless networks, such as cellular network and WLAN. In the infrastructure based wireless networks a node can only send a packet to a destination node only via access point (in cellular network like GSM, it is called base station). The access point establishes an network area and only the nodes in this area can use access point's services. There are some unknown events, which cause access point's malfunction. The nodes lose their network and are quasi not working. It is the biggest infrastructure's disadvantage. [...]

The wireless ad-hoc networks only consist of nodes equipped with transceiver. The network are created to be independent from an infrastructure. Therefore, the nodes must be able to arrange their own networks. Keep in mind, that a node can now communicate only with other nodes in its transmission range. In the infrastructure based wireless network, the nodes can communicate with a node which is located in another network area, by transmitting data to destination access point and this access point relay the data to the desired node." [6]

2.1.7 Wireless Sensor Network

"Ad-hoc networks are used in **Wireless sensor networks** (WSNs) consisting of small programmable devices equipped with radio-enabled sensing capabilities and have been applied in information gathering ranging from the environment temperature, radiation, the presence of fire, seismic vibrations, and more. WSNs compared with wired networks provide many advantages in deployment, cost and size. Wireless technology enables users to set up a network quickly, more it enables them to set up a network where it is inconvenient or impossible to wire cables. Moreover, common WSNs can consist of up to several hundreds of those small devices." [5]

2.1.8 Random Geometric Graphs

The goal of this work is to examine some algorithms that operate on graphs in a distributed environment. In order to familiarize with them, we will also predict how they behave on special case graphs, such as linear graph, grid and clique. We will also try to assess, possibly using computer simulations, how those algorithms work in random networks.

To build a Monte-Carlo simulation, we need a method of obtaining random instances of the problem. In our case, we need to know how to create random graphs (networks) on which the algorithms would run. We chose random geometric graphs model, because it maps the reality fairly well as it comes to wireless networks. Theoretical properties of the model is well studied in the monograph [8].

Definition 17 (Geometric Graph). *Let $||\cdot||$ be some norm on $\mathbb{R}^d$, and let p be some positive parameter. Given a finite set $\mathcal{X} \in \mathbb{R}^d$, we denote by $G(\mathcal{X};p)$ the undirected graph with vertex set $\mathcal{X}$ and with undirected edges containing all those pairs $\{X,Y\}$ with $||Y-X|| \leqslant p$. We shall call this a* ***geometric graph****.*

Definition 18 (Random Geometric Graph). *Let f be some specified probability density function on $\mathbb{R}^d$, and let $X_1, X_2, \ldots$ be independent and identically distributed d-dimensional variables with*

common density f. Let $\mathcal{X}_n = \{X_1, X_2, \ldots, X_n\}$. Our main subject is the graph $G(\mathcal{X}_n; p)$, which we shall call a ***random geometric graph****.*

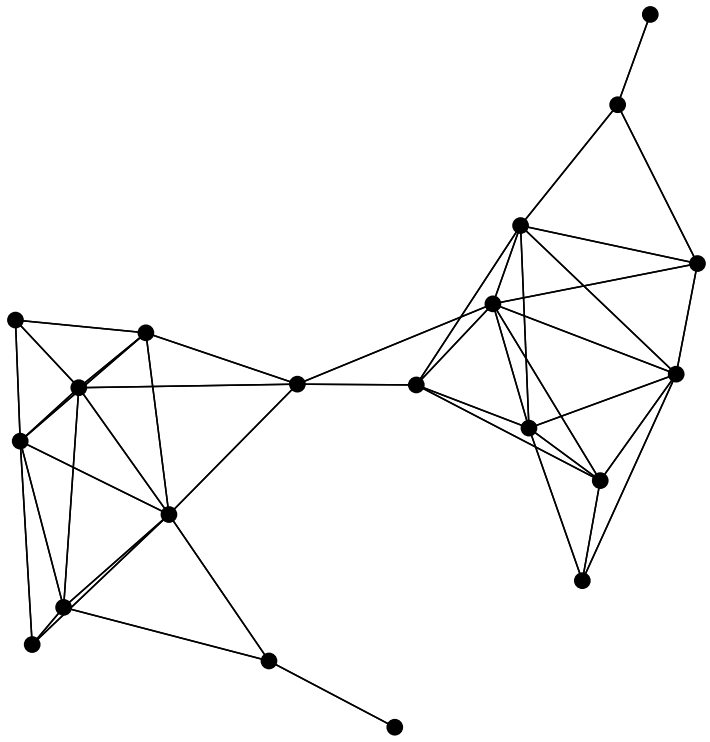

Figure 3: Example of a random geometric graph on a unit square with $n = 20$, $p = 0.3$

The reason why such model has been chosen is the fact, that wireless ad-hoc networks work in a similar way. Devices can communicate directly only to those, which are not too far away. More sophisticated models could have been adopted, such as Bluetooth topology [9], nevertheless, we chose random geometric graphs for the sake of simplicity and clarity.

2.1.9 Complexity of Algorithms

For every execution of an algorithm, number of sent messages E and number of time slots used T is measured. Since in the physical world some amount of energy is needed to transmit a message, we can treat E as the measure of energy that the algorithm used. T, on the other hand, tells us how quickly the algorithm comes up with an approximation of the diameter. It depends on the application, which of those values are more important. Usually, a good compromise between those two is the optimal solution.

Those values depend on the algorithm, connection graph and starting condition, e.g. which device triggers the algorithm. Knowing how the algorithm works, we can give their asymptotic behaviour using big-O-notation. For deterministic topologies (that is, not random) those values are usually easy to calculate explicitly.

2.2 Conventions

A pseudocode is used to describe distributed algorithms. Formal language is avoided for the convenience of the reader. It is assumed that a device can perform some initialization code at the very beginning. It can also execute a piece of code on every time step. However, the most important actions are usually performed as a result of receiving a message. Therefore, pseudocode also defines `OnReceive(`M, p`)` function, where M is the received message and p is an index of the node from which the message came (we denote the set of devices as $V = \{v_1, v_2, \ldots, v_n\}$). Actions like sending a message to particular neighbour or broadcasting a message to all the neighbours are defined naturally using the model given a few paragraphs earlier.

In the rest of this paper, we will consider only symmetric graphs, that is, we assume that each communication channel is bidirectional. Moreover, we shall use undirected graphs instead of directed on behalf of the fact 1. When we talk about the number of edges, we mean the number of bidirectional communication channels.

We consider only connected graphs.

If the graph, which is referred to, is known from the context, we will simply write D instead of $D(G)$. If variables n, m, c, η occur without being strictly defined, they by default mean respectively the number of nodes, the number of edges, the number of children of given node in the spanning tree, the number of all neighbours of given node.

The devices are aware of their neighbourhood, so in the pseudocode we assume that every node has an array `neighbours[`$1 \ldots \eta$`]` of references to its neighbours. We also allow the devices to use `indexOf(element, array)` function, that returns an index of given `element` in given `array`.

2.3 Basic Algorithms

Here, we define two most basic algorithms which later will be used as components of more complex programs.

2.3.1 Flooding

One can imagine a scenario where one device wants to pass some information to all other nodes in the network. All it needs to do is broadcast the information to all its neighbours. When other node receives a message for the first time, it resends it to all its neighbours except the one it received the message from. If a node receives a message for second or more time, it ignores it. Algorithm 1 contains a pseudocode for this process.

```
Input: i — index of the current processor
Input: r — index of the processor that triggers the algorithm
Input: M — the message that is to be broadcast
1  OnInitialize()
2      if i = r then
3          done := true;
4          send M to all neighbours;
5      else
6          done := false;
7  OnReceive (N, s)
8      if done then
9          return;
10     else
11         send N to all neighbours but v_s;
12         done := true;
```

Algorithm 1: Flooding algorithm

Let us now consider the complexities T and E for this algorithm. In order to do that, we need the following theorem:

Theorem 1. *In the execution of the basic flooding algorithm, every processor at distance t from the processor i receives the message M in round t.*

Proof. It is easily proved by induction. In the first round ($t = 1$) every node adjacent to v_r receives the message. Therefore, the basis of the induction is correct.

Now let us assume that at time t every device that is not farther than t from v_r already received the message M.

In the round $t+1$ devices, whose distance from v_r is equal $t+1$, receive the message M. Thus all devices not farther than $t+1$ from v_r have received the message by the time $t+1$. □

Let $d = \max_{v \in V} d(v_r, v)$. By theorem 1, after d rounds every node will have `done` variable set to `true`. The algorithm ends, so $T = d$. As $d \leqslant D \leqslant 2d$, we can write $\frac{1}{2}D \leqslant T \leqslant D$.

Message M is transmitted once or twice through every edge — it depends on a topology of a network. Then $m \leqslant E \leqslant 2m$.

Using asymptotic notation, $T = O(D)$, $E = O(m)$.

2.3.2 Broadcast

Let us assume that there exists a rooted spanning tree on a network, that is, at the beginning every node has a constant `parent` filled with a reference to its parent, or equal **`null`** when it is the root itself. It also has a constant `c` — the number of children, and an array `children[1 ... c]` of references to its children.

With this additional information, the flooding algorithm can be performed a little bit more effectively when it is triggered by the root. Instead of sending the message to all neighbours, we need to send it only to the children. The modification is called **broadcast** and it is included in the algorithm 2.

Input: i — index of the current processor
Input: r — index of the processor that triggers the algorithm (the root)
Input: M — the message that is to be broadcast
Input: `parent` — the reference to the parent
Input: c — number of children
Input: `children[1 ... c]` — array of references to the children

```
1  OnInitialize()
2      if i = r then
3          done := true;
4          send M to every neighbour in children;
5      else
6          done := false;

7  OnReceive (N, s)
8      if done then
9          return;
10     else
11         send M to every neighbour in children;
12         done := true;
```

Algorithm 2: Broadcast algorithm

Even though the time complexity is the same as in flooding, knowledge about the spanning tree comes down to savings of the energy complexity. The number of sent messages is equal to the number of edges in the spanning tree, therefore $E = n - 1 = O(n)$, which is obviously better than $O(m)$, since in the worse case $m = O(n^2)$.

2.3.3 Convergecast

When a spanning tree is given at the start, we can consider a different scenario. Suppose, that each node has some value the network wants to aggregate somehow, e.g. each device in the sensor network measures the temperature and the network would like to know what is the maximal temperature of the system.

In **convergecast** algorithm every leaf of the given spanning tree starts by sending its value to its parent. When a node receives a message, it stores the information about that event in a buffer of length equal to number of its children. Next, if the message was received from all the children then the node aggregates the data stored in the buffer (calculates the maximum), sends the result to its parent and terminates. As a result, the root obtains the final result (e.g. the maximum of all measurements in the sensor network). Collecting minimum or sum instead of maximum is just as easy.

Input: i — index of the current processor
Input: r — index of the processor that triggers the algorithm (the root)
Input: M — value which the node has
Input: `parent` — the reference to the parent
Input: c — number of children
Input: `children[1 ... c]` — array of references to the children

```
1 OnInitialize()
2     if c = 0 then
3         send M to parent;

4 OnReceive(N, s)
5     buf[indexOf(v_s, children)] := N; if buf is full then
6         M := max(buf[1],buf[2],...,buf[c],M);
7         send M to parent;
```

Algorithm 3: Convergecast algorithm

You can easily see that complexities E and T of the convergecast is the same as in the broadcast.

3 Problem Analysis

This section covers the description and brief analysis of algorithms that estimate the diameter in distributed systems. They can be divided into two main groups — the first is based on spanning tree building, the second uses *Extrema Propagation* method.

3.1 Building Spanning Tree

Broadcast and convergecast algorithms, mentioned in the previous section, assume that a spanning tree is given at the start. Nevertheless, there is a simple and effective way to build such tree using modified flooding algorithm with message M_1. Every node remembers the first neighbour who contacts them and treats it as a parent. Every other message M_1 is ignored, but the fact of receiving message is remembered by the receiver. When a node realizes that it is a leaf in the tree (all neighbours sent M_1), it initializes convergecast algorithm with message M_2 along with a counter with value 1, which is incremented by every node during the convergecast algorithm. When root receives M_2 messages from all neighbours, the maximum received counter yields a depth of constructed spanning tree.

Nodes do not need to remember which of their neighbours are its children.

Algorithm 4 describes the whole process.

Input: i — index of the current processor
Input: r — index of the processor that triggers the algorithm
Input: η — number of neighbours

```
1  OnInitialize()
2      if i = r then
3          send M_1 to all neighbours;
4      parent := null;
5      done := false;
6      received[1 ... η] := false;
7      depth := 0;
8  OnReceive (N, s)
9      if done then
10         return;
11     if N = M_1 then
12         if parent = null then
13             parent = v_s;
14             send M_1 to all neighbours but v_s;
15             received[indexOf(v_s, neighbours)] := true;
16         else
17             received[indexOf(v_s, neighbours)] := true;
18     else if N = (M_2, out counter) then
19         received[indexOf(v_s, neighbours)] := true;
20         depth := max(depth, counter);
21     if received[j] = true for every j ∈ {1, ··· η} then
22         if parent ≠ null then
23             send (M_2, depth + 1) to parent;
24         else
25             depth is the final result;
26         done := true;
```

Algorithm 4: Building Spanning Tree algorithm

3.1.1 Analysis

It is assumed that algorithm 4 is triggered by a single node. After successful process, the result (estimation of the diameter) is known in that particular node.

The biggest disadvantage of that algorithm is the fact, that every device is a single point of failure — for every node there exists a round on which a failure of the node causes the whole algorithm to stale. This is unacceptable property for an algorithm that is supposed to work in ad-hoc environment. Nevertheless, it is a basis for other algorithms presented in this section.

Theorem 2. *Assuming synchronous communication, spanning Tree built by algorithm 4 is a Breadth First Spanning Tree rooted at v_r.*

The proof follows from lemma 2.8 from the book [2].

Let d be the result of running algorithm 4. Using the theorem 2, we conclude that $d \leqslant D \leqslant 2d$. Thus, the relative error can not exceed 50%.

If we only look at M_1 messages, communicates are passed just like in the flooding algorithm. M_2 messages flow exactly like in convergecast algorithm, but the leaves usually don not send M_2

in the same round. Thus time complexity of algorithm 4 is not worse than complexities reached by flooding and convergecast algorithms added together. Energy complexity is an exact sum of complexities reached by flooding and convergecast algorithms. Therefore $T = O(D)$, $E = O(m)$.

Linear Graph

Let $T^r_{Lin(n)}$ be the number of time slots that algorithm 4 uses when starting from node v_r and running on the linear graph (see figure 4) of size n. Similarly, let $E^r_{Lin(n)}$ denote number of messages sent in that execution of the algorithm.

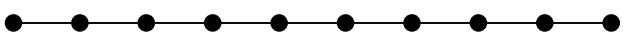

Figure 4: Linear graph with $n = 10$

Number of time slots used is equal to doubled depth of created spanning tree. Therefore

$$T^r_{Lin(n)} = \begin{cases} 2(n-r) & \text{if } r \leqslant \lceil \frac{n}{2} \rceil \\ 2(r-1) & \text{if } r > \lceil \frac{n}{2} \rceil. \end{cases}$$

In graphs with no cycles (i.e. in trees) number of sent messages is equal to doubled number of edges, as a result

$$E^r_{Lin(n)} = 2(n-1).$$

Two Dimensional Grid

Let $T^{(p,q)}_{Grid(x,y)}$ and $E^{(p,q)}_{Grid(x,y)}$ be defined analogously, but relate to different network topology — two dimensional grid (see figure 5) of size $x \times y$. Note that in this case we exceptionally index nodes with a pair of integers. $v_{(p,q)}$ starts the algorithm, $p \in \{1,2,\ldots,x\}$, $q \in \{1,2,\ldots,y\}$.

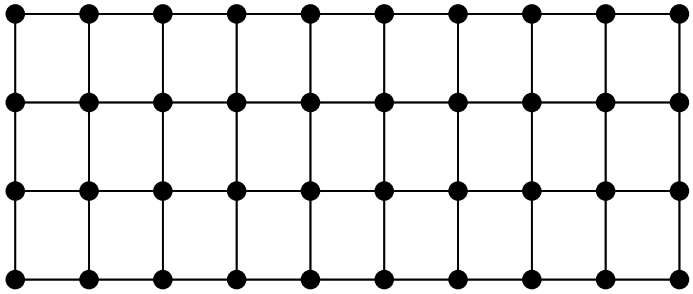

Figure 5: 2D graph with $x = 10$, $y = 4$

Using the same method as in linear graph case, we claim that

$$T^{(p,q)}_{Grid(x,y)} = \begin{cases} 2(x-p+y-q) & \text{if } p \leqslant \lceil \frac{x}{2} \rceil \wedge q \leqslant \lceil \frac{y}{2} \rceil \\ 2(p-1+y-q) & \text{if } p > \lceil \frac{x}{2} \rceil \wedge q \leqslant \lceil \frac{y}{2} \rceil \\ 2(x-p+q-1) & \text{if } p \leqslant \lceil \frac{x}{2} \rceil \wedge q > \lceil \frac{y}{2} \rceil \\ 2(p-1+q-1) & \text{if } p > \lceil \frac{x}{2} \rceil \wedge q > \lceil \frac{y}{2} \rceil. \end{cases}$$

Energy used is also easy to conclude. M_1 message is sent through every edge once or twice (M_1 goes twice through exactly $(x-1)(y-1)$ edges). M_2, on the other hand, goes once through every node in the obtained spanning tree, which is increases E by the number of nodes decreased by one.

$$E^{(p,q)}_{Grid(x,y)} = x(y-1)+y(x-1)+(x-1)(y-1)+xy-1.$$

3.2 Building Spanning Tree with Acknowledgements

Algorithm described in the previous section is a valid solution only when we assume that network does not change during the execution (no nodes fail, leave or join the network). It can not be used in real life ad-hoc networks then.

Building Spanning Tree method can be easily modified so it is more robust. We try to fix the problem of infinite awaiting for the response which occurs when the device we wait for crashes. We propose the following rule — every node has to send acknowledgements (M_3 messages) to its parent in time slots that are powers of two $(1,2,4,8,16,32,\dots)$ from the time M_1 was received. Please note, that every node knows exactly when its neighbours have to acknowledge their existence. When a node notices that some neighbour did not send M_3 when it was supposed to do so, it ignores the neighbour.

The aforementioned changes has been translated into the pseudocode and included in the algorithm 5.

Input: as in algorithm 4

```
1  OnInitialize()
2      if i = r then
3          send M_1 to all neighbours;
4          awaitingCounter := 0;
5      else
6          awaitingCounter := −1;
7      parent := null;
8      done := false;
9      received[1 ... η] := false;
10     depth := 0;
11 BeforeReceiving()
12     ackReceived[1 ... η] := false;
13 OnReceive (N, s)
14     if done then
15         return;
16     if N = M_1 then
17         if parent = null then
18             parent = v_s;
19             send M_1 to all neighbours but v_s;
20             received[indexOf(v_s, neighbours)] := true;
21             awaitingCounter = 0;
22         else
23             received[indexOf(v_s, neighbours)] := true;
24     else if N = (M_2, out counter) then
25         received[indexOf(v_s, neighbours)] := true;
26         ackReceived[indexOf(v_s, neighbours)] := true;
27         depth := max(depth, counter);
28     else if N = M_3 then
29         ackReceived[indexOf(v_s, neighbours)] := true;
30 AfterReceiving()
31     if done then
32         return;
33     if awaitingCounter − 1 is a power of 2 then
34         for j ∈ {1 ··· η} do
35             if ackReceived[j] = false then
36                 received[j] := true;
37     if awaitingCounter ⩾ 0 then
38         awaitingCounter += 1;
39     if received[j] = true for every j ∈ {1, ··· η} then
40         if parent ≠ null then
41             send (M_2, depth + 1) to parent;
42         else
43             depth is the final result;
44         done := true;
45     else if awaitingCounter is a power of 2 then
46         send M_3 to parent;
```

Algorithm 5: Building Spanning Tree with Acknowledgements algorithm

3.2.1 Analysis

In this case, just like it was in the algorithm 4, exactly one node triggers the algorithm and the same node receives the final result.

The most important improvement is the fact that this solution has only one single point of failure — the node that initializes the algorithm, i.e. the root of created spanning tree. If it fails, the whole algorithm fails. What is important, corruption of any other device does not stale the whole algorithm.

Acknowledgements change nothing as it comes to the time complexity of the algorithm, therefore $T = O(D)$.

Let us consider how introducing acknowledgements affect the energy consumption. We shall prove, that it grows slowly comparing to the previous solution.

Theorem 3. *Energy consumption of algorithm 5 is* $E = O(m + n\log_2 D)$.

Proof. Let v be an arbitrary node where algorithm 5 forms a rooted spanning tree T. Let d be the maximum distance in the tree T between v and a leaf which is a descendant of v in T. Time slots between receiving M_1 and returning M_2 by v shall be $2d$. During that time, v sends $\lceil \log_2(2d) \rceil$ M_3 messages. Since $d < D$ and number of nodes is n, we conclude that number of M_3 messages send throughout whole algorithm is $O(n\log_2 D)$. Number of messages M_1 and M_2 in the system is $O(m)$, as it was proved earlier. □

Linear Graph

Since the time of execution is the same as in algorithm 4, we can write

$$T^r_{Lin(n)} = \begin{cases} 2(n-r) & \text{if } r \leqslant \lceil \frac{n}{2} \rceil \\ 2(r-1) & \text{if } r > \lceil \frac{n}{2} \rceil. \end{cases}$$

To analyze number of acknowledgements, we need to introduce auxiliary functions. Let

$$f(d) = \begin{cases} 3d - 1 + \sum_{k=0}^{\lfloor \log_2 d \rfloor} (d - 2^k) & \text{if } d \neq 0 \\ 0 & \text{if } d = 0 \end{cases}$$

and

$$g(d) = f(d) - 2d.$$

Basing on how the algorithm works, we can observe, that

$$E^1_{Lin(n)} = 3(n-1) - 1 + \sum_{k=0}^{\lfloor \log_2(n-1) \rfloor} ((n-1) - 2^k) = f(n-1).$$

When we take a look at the situation when v_r starts, we can divide sent messages into those 'on the left' and 'on the right' hand side of v_r. The size of each of those messages groups can be expressed with f. Finally,

$$E^r_{Lin(n)} = f(n-r) + f(r-1).$$

Two Dimensional Grid

Identically as in algorithm 4, we can write

$$T^{(p,q)}_{Grid(x,y)} = \begin{cases} 2(x-p+y-q) & \text{if } p \leqslant \lceil \frac{x}{2} \rceil \wedge q \leqslant \lceil \frac{y}{2} \rceil \\ 2(p-1+y-q) & \text{if } p > \lceil \frac{x}{2} \rceil \wedge q \leqslant \lceil \frac{y}{2} \rceil \\ 2(x-p+q-1) & \text{if } p \leqslant \lceil \frac{x}{2} \rceil \wedge q > \lceil \frac{y}{2} \rceil \\ 2(p-1+q-1) & \text{if } p > \lceil \frac{x}{2} \rceil \wedge q > \lceil \frac{y}{2} \rceil. \end{cases}$$

Now let us assume, without loss of generality, that $x \geqslant y$.

We would like to calculate how many messages go through the system. Unfortunately, that is more complex since the number of acknowledgements depends on the 'shape' of created spanning tree. The 'shape' is, on the other hand, dependant on the order in which every node receives messages. Because of the fact that we do not know the order, we can not explicitly compute E as we did it in the earlier sections.

Nevertheless, we can consider the worst case. Let us focus on the situation where a node in the corner starts. The maximum number of messages is sent when the tree has the longest possible branches, like in figure 6.

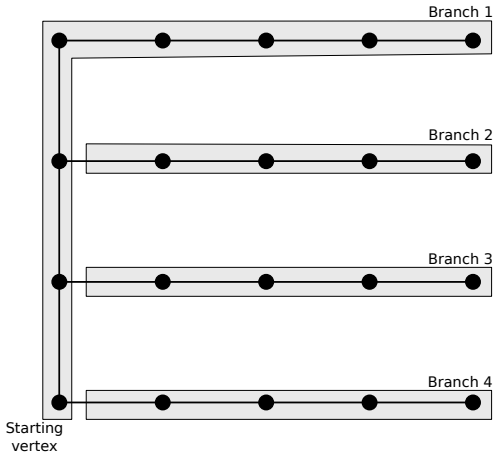

Figure 6: Worst possible spanning tree for grid

$f(d)$, intuitively, was constructed in a way, that it gives the total number of messages sent in one branch of depth d. $g(d)$, on the other hand, gives the number of acknowledgements in that process. Using the knowledge how the algorithm 5 works, analyzing the above figure and observations about function $g(d)$, we can conclude that

$$E^{(1,1)}_{Grid(x,y)} \leqslant x(y-1)+y(x-1)+(x-1)(y-1)+xy-1+g(x+y-2)+(y-1)g(x).$$

Achieved equation is confirmed by the computer simulations.

It is also clear that, for the worst case, $E^{(p,q)}_{Grid(x,y)} \leqslant E^{(1,1)}_{Grid(x,y)}$ for every $p \in \{1,\ldots,x\}$, $q \in \{1,\ldots,y\}$.

3.3 Two Phase Spanning Tree Building

Two previous solutions base on the fact, that we can estimate diameter with the relative error of at least 50% by measuring the depth of breadth first spanning tree created by any node.

Now we try to use the fact, that the topology of ad-hoc networks is generally similar to geometric graphs — there is a physical limit for transmission range, so devices neighbour only those, which are geographically close. It means, when we create breadth first spanning tree from a random node v_r, the farthest leave v_s reached should be 'somewhere around the edge' of the graph. Then we request that v_s builds its own spanning tree. Its depth should be more close to the diameter.

We assume that every device has its own identification number, which is unique among all participants. During convergecast we will store the ID of the node with maximal depth found so far. In that case, the root obtains the ID of the leave with maximal depth. It triggers the flooding algorithm with M_4 message concatenated with aforementioned ID. When the leaf notices, that given ID is the same as its own ID, it triggers building spanning tree one more time. When result is obtained, the node floods it with message M_5 so that every other device can know it.

Algorithm 6 describes the process in much more concise form.

1 v_r triggers algorithm 4, where identification number `id` of the farthest found leaf is being sent along with M_2 and `counter`;
2 v_r triggers the flooding algorithm with M_4 and `id` of the node v_s, who has the maximum deepness found;
3 when v_s receives M_4, it triggers algorithm 4 once again;
4 v_s uses the flooding algorithm with M_5 and result obtained in point 3.

Algorithm 6: Two Phase Spanning Tree Building algorithm

Instead of ordinary building tree algorithm, method with acknowledgements could be used as well. To avoid complications and keep things simple, we shall proceed with the analysis of version without acknowledgements.

3.3.1 Analysis

As it was before, one device has to trigger the whole algorithm. However, at the end the result is known to every node in the network.

Overall time complexity comprises of two executions of algorithm 4, and two executions of algorithm 1. It still is $O(D)$, but hidden constants are significantly greater than before.

In similar way we can assess the energy complexity — we sum up complexities of all steps. It turns out, that asymptotically we do not leave $O(m)$.

If we used variant with acknowledgements, we would get $E = O(m + n\log_2 D)$.

Linear Graph

Using the same methods as in previous sections about linear graphs,

$$T^r_{Lin(n)} = \begin{cases} 3(n-r)+3(n-1) & \text{if } r \leqslant \lceil \frac{n}{2} \rceil \\ 3(r-1)+3(n-1) & \text{if } r > \lceil \frac{n}{2} \rceil, \end{cases}$$

$$E^r_{Lin(n)} = 6(n-1).$$

Two Dimensional Grid

Since we assume we do not use acknowledgements, T and E is also easy to calculate for the two dimensional grid.

Note that T is increased by one because of the behaviour at the end — the last node receives two M_5 messages in the same time slot. It responds immediately after receiving the first one and broadcasts it, prolonging the whole process by one round. E, on the other hand, is decreased by one in an unintuitive way. The reason for that is the fact that v_s (using notation from the pseudocode of algorithm 6) receives M_4 from one of its two neighbours and does not broadcast M_4 to the second.

Eventually,

$$T^{(p,q)}_{Grid(x,y)} = \begin{cases} 3(x-p+y-q)+3(x-1+y-1)+1 & \text{if } p \leqslant \lceil \frac{x}{2} \rceil \wedge q \leqslant \lceil \frac{y}{2} \rceil \\ 3(p-1+y-q)+3(x-1+y-1)+1 & \text{if } p > \lceil \frac{x}{2} \rceil \wedge q \leqslant \lceil \frac{y}{2} \rceil \\ 3(x-p+q-1)+3(x-1+y-1)+1 & \text{if } p \leqslant \lceil \frac{x}{2} \rceil \wedge q > \lceil \frac{y}{2} \rceil \\ 3(p-1+q-1)+3(x-1+y-1)+1 & \text{if } p > \lceil \frac{x}{2} \rceil \wedge q > \lceil \frac{y}{2} \rceil, \end{cases}$$

$$E^{(p,q)}_{Grid(x,y)} = 4(x(y-1)+y(x-1)+(x-1)(y-1))+2(xy-1)-1.$$

3.4 Extrema Propagation

All the previous methods rely on the fact that construction of breadth first spanning tree is easy in distributed environment and gives some approximation of the diameter. **Extrema Propagation** is an algorithm that was invented in order to allow quick and robust distributed estimation of the sum of positive real number, as well as the size of the network [4]. A little enhancement makes this technique able to estimate the diameter in a small amount of time [3].

> "Basically, each node generates a vector of K exponentially distributed random numbers (`Minimums` vector in the rest of this document) and sends it to its neighbours. Each node aggregates and resends vectors from neighbours using the pointwise minimum — an idempotent operation. After convergence (determined by each node, after a predefined number of rounds have passed without changing the `Minimums` vector) the resulting vector — with the K global minimums — can be used to estimate the number of nodes in the network.
>
> [...] For each entry in the `Minimums` vector of the Extrema technique, one adds a corresponding entry in a new vector `Hops`. The `Hops` vector is initialized at every node with zeroes. Every time that a node updates the `Minimums` vector because it has received a smaller value from one of its neighbours, it also updates the corresponding entry in the `Hops` vector with the value received from that neighbour. The `Hops` vector is also updated when a node receives a smaller entry in the `Hops` vector for an equal value in the `Minimums` vector (meaning that a shorter route to that minimum was found). A copy of the `Hops` vector, with values increased by 1, is sent along with the `Minimums` vector, to every neighbour, in each round.
>
> After convergence, in each node $\max(\texttt{Hops})$ is an estimate of its eccentricity and the network diameter that can be determined across the network by aggregating a maximum of these eccentricities. In particular, if one of the minimums was generated in a periphery node the final diameter estimate will be exact." [3]

Algorithm 7 shows a pseudocode of Extrema Propagation technique, in which every node estimates its eccentricity.

Input: K — length of Minimums and Hops tables
Input: τ — number of rounds node waits for new data

```
1  OnInitialize()
2      Minimums[1 ... K] := randomExp(1);
3      Hops[1 ... K] := 0;
4      noNews := 0;
5      done := false;
6  BeforeReceiving()
7      if done then
8          return;
9      oldMinimums[1 ... K] := Minimums[1 ... K];
10 OnReceive ((MinimumsReceived, HopsReceived), s)
11     if done then
12         return;
13     for k ∈ {1, ..., K} do
14         if MinimumsReceived[k] < Minimums[k] then
15             Minimums[k] := MinimumsReceived[k];
16             Hops[k] := HopsReceived[k];
17         else if MinimumsReceived[k] = Minimums[k] ∧ HopsReceived[k] < Hops[k] then
18             Hops[k] := HopsReceived[k];
19 AfterReceiving()
20     if done then
21         return;
22     if oldMinimums = Minimums then
23         noNews++;
24     else
25         noNews := 0;
26     send (Minimums, Hops) to all neighbours;
27     if noNews ⩾ τ then
28         result := max(Hops);
29         done := true;
```

Algorithm 7: Extrema Propagation algorithm

3.4.1 Analysis

Extrema Propagation assumes, that every node starts the algorithm in the same time slot. When the process ends, every node has its own eccentricity — value that can be used to estimate the diameter. Since the greatest eccentricity is the best estimation, it would be good idea to aggregate maximum of these values after Extrema Propagation finishes (we treat aggregation as a technique separate and independent to Extrema Propagation and therefore it is not described in this paper).

The substantial property of the technique is the fact, that it has no single points of failure.

The algorithm needs relatively a small number of time frames — in the worst case the node has to wait D rounds to receive *Minimums* messages plus τ rounds of awaiting for more changes. Therefore, $T \leqslant D+\tau = O(D+\tau)$.

The main disadvantage of the method is its large energy complexity. Since through every edge

two messages can be transmitted (each one in the opposite direction), and whole process can take up to $D+\tau$ time slots, we assess $E \leqslant 2m(D+\tau) = O(m(D+\tau))$.

4 Computer Simulations: Project and Implementation

We used computer simulations in order to compare two methods of estimating the diameter on random geometric graphs on a unit square, where nodes are located with uniform probability distribution. Simulations were run with number of nodes $n = 100, 200, \ldots, 1000$ and threshold p chosen in a way, so that likelihood of obtaining connected graph is satisfactory. If the drawn graph was found not to be connected, it was rejected and drawn again.

The first method to be examined is modified algorithm 6 — acknowledgements has been added just like in the algorithm 5. From now on, we will call this algorithm *Breadth First Spanning Tree Building Algorithm with 2 Phases and Acknowledgements*, or simply *BFST Building* algorithm.

The second method is Extrema Propagation method where the result is aggregated throughout the network after the actual Extrema Propagation code quits. What is important, 15 combinations of the algorithm parameters has been examined separately — for each pair of parameters $K = 10, 50, 100, 200, 300$ and $\tau = 2, 4, 6$, so that we can also see how these parameters affect the performance and accuracy of the Extrema Propagation method.

Such methods have been chosen because they hold similar properties and they can be compared — at the end of execution, every node should know the result of the estimation. The difference between them is the fact that for the first algorithm the leader has to be chosen, whereas, in the second method every node has to know exactly when to begin the whole process to start at the same time. Nevertheless, the difference does not stop us from comparing those methods.

For chosen parameters n, p and algorithm versions, 5000 connected networks has been randomly created. The algorithm was examined, diameter estimation d, number of messages sent E and number of time slots used T were obtained. Next, the real diameter D has been computed off-line using Dijkstra's algorithm, which was run for every node. Relative error $\delta = |\frac{d}{D}|$ was computed. Average values of δ, T, and E, along with their variances, were calculated.

C++ application has been written to simulate the model described in this work and obtain the data. Next, the data was processed using *Perl*, *Bash* scripts and *R-Project* software. Everything can be run in a *GNU/Linux* operating system. Computed means and variances are included in the tables in the section 5. This section also includes plots that show chosen, the most interesting, pieces of data.

Figure 7 presents inheritance hierarchy of the application. Classes that represent vertices are associated with specific algorithms (`CBFSTreeBuilder`, `CBFSTreeBuilderAck`, `CBFSTreeBuilder2Phase`, `CBFSTreeBuilder2PhaseAck`, `CExtremaPropagation`). Various classes which inherit from `CNetwork` represent different network topologies (`CLinearNetwork`, `CGrid2DNetwork`, `CRandomGeoNetwork`).

4.1 Executables and Usage

To compile the application, use `make` program.

The whole application consists of three executables, namely

- `simulator_interactive` — interactively asks, via standard input and output, for method to be used, network topology, and all the parameters. Then, performs one simulation (which starts from a random vertex) and prints results to standard output,
- `simulator_all_rgg_bfst` — for given n and p performs 5000 simulations of *BFSTB2PhasesAck* algorithm and saves the results in files,

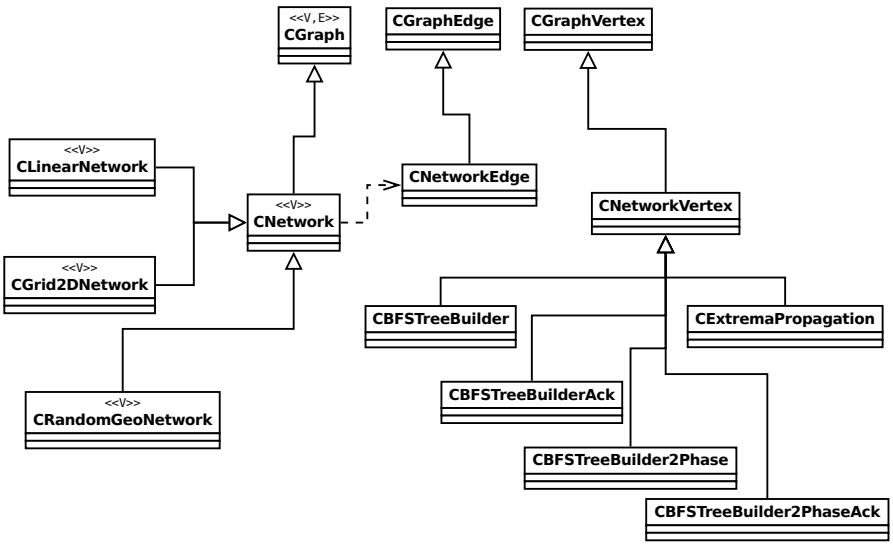

Figure 7: Inheritance hierarchy of the simulator

- `simulator_all_rgg_ep` — for given n and p performs 5000 simulations of *Extrema Propagaion* algorithm and saves the results in files.

The best way to test the simulator is to run `simulator_interactive`. Its usage is intuitive. Listing 1 shows the output of the example session. The result is written to the standard output. `n` is the number of nodes, `m` is the number of edges, `r` is an index of starting vertex (from interval $[1,\ldots,m]$), `E` is the number of transmitted messages, `T` is the number of time slots used, `d` is the estimation of the diameter, `D` is the real diameter.

In order to obtain more detailed information about execution of algorithms (information about every message that is being sent), some changes in `Makefile` have to be done before the compilation — `# debug` section has to be uncommented.

```
Which method do you want to use?
  0 - Breadth First Spanning Tree Building
  1 - Breadth First Spanning Tree Building with Acknowledgements
  2 - Breadth First Spanning Tree Building with 2 Phases
  3 - Breadth First Spanning Tree Building with with 2 Phases and Acks.
  4 - Extrema Propagation
3
Which network topology do you want to use?
  0 - Linear Network
  1 - 2D Grid
  2 - Random Geometric Graph
2
Enter n
100
Enter p
```

```
0.2
Enter rounds limit
100
n = 100
m = 529
r = 34
E = 4437
T = 51
d = 8
D = 8
```

Listing 1: Example usage of `simulator_interactive` program

5 Experimental Results

Performed experiments gave an answer to the question: how accurate those methods are, how much energy and time do they use?

5.1 Relative Error

As it comes to the accuracy of estimation, it is measured by the average relative error $\bar{\delta}$. Values of $\bar{\delta}$ and experimental variance of all performed simulations are shown in the table 1.

Figure 8 shows that with greater n, the relative error for Extrema Propagation method grows slightly, whereas BFST Building method seems not to loose its accuracy. For Extrema Propagation, choice of K, the length of `Maximums` and `Hops` arrays, is crucial.

Selection of τ is less important for the accuracy. The smaller K is, the more significant τ is, as it can be seen in the figures 9 and 10.

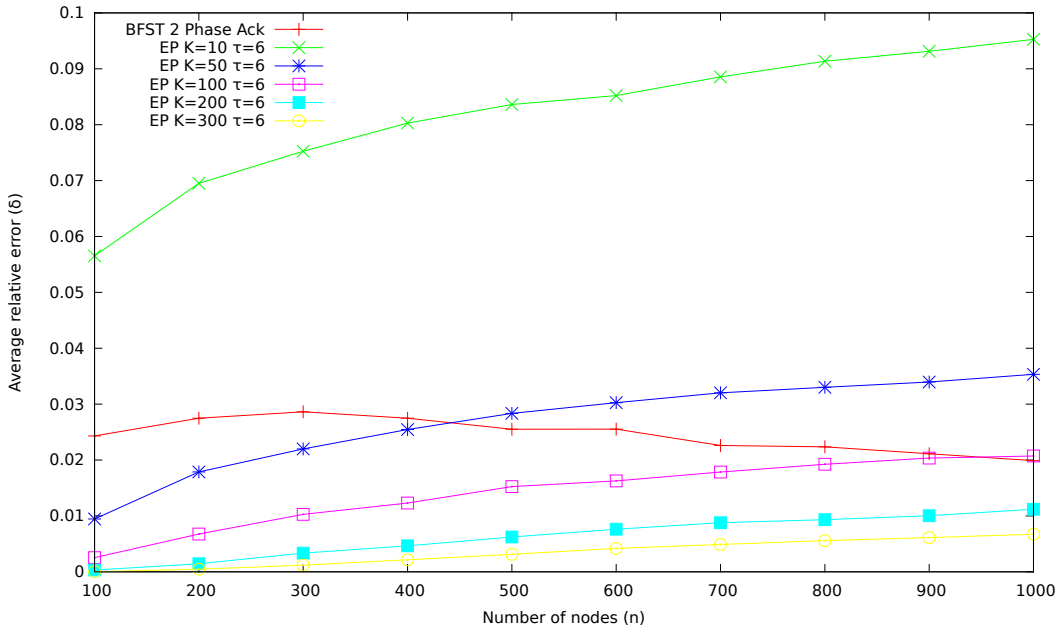

Figure 8: Average relative error of BFST Building and Extrema Propagation

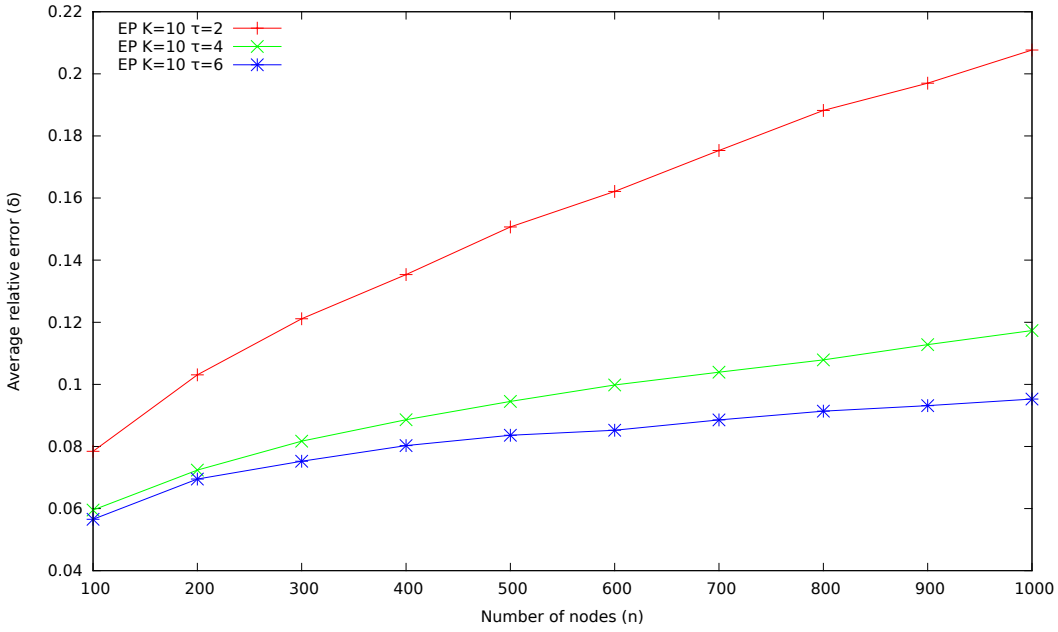

Figure 9: Average relative error of Extrema Propagation method with various τ and $K = 10$

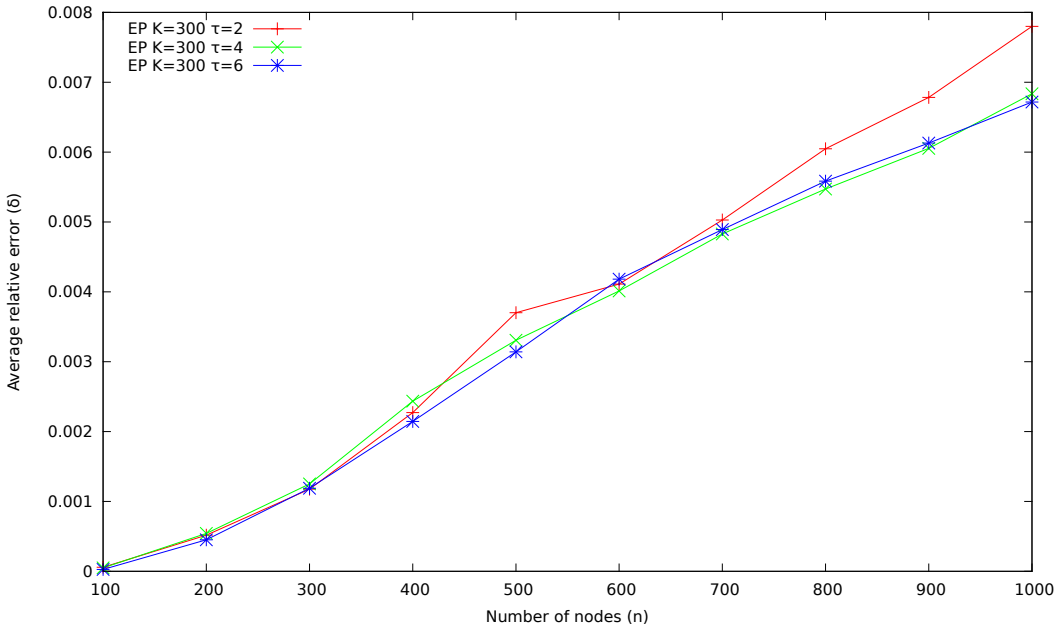

Figure 10: Average relative error of Extrema Propagation method with various τ and $K = 300$

Table 1: Experimental results: average relative error of the estimation with the variance

	$n = 100$ $p = 0.16248$	$n = 200$ $p = 0.11959$	$n = 300$ $p = 0.09982$	$n = 400$ $p = 0.08776$	$n = 500$ $p = 0.07940$	$n = 600$ $p = 0.07314$	$n = 700$ $p = 0.06823$	$n = 800$ $p = 0.06424$	$n = 900$ $p = 0.06091$	$n = 1000$ $p = 0.05807$
BFST building 2 Phases, ACK	$\delta \approx 0.02431$ $\sigma^2 \approx 0.00262$	$\delta \approx 0.02749$ $\sigma^2 \approx 0.00230$	$\delta \approx 0.02863$ $\sigma^2 \approx 0.00204$	$\delta \approx 0.02748$ $\sigma^2 \approx 0.00176$	$\delta \approx 0.02551$ $\sigma^2 \approx 0.00152$	$\delta \approx 0.02553$ $\sigma^2 \approx 0.00146$	$\delta \approx 0.02261$ $\sigma^2 \approx 0.00120$	$\delta \approx 0.02235$ $\sigma^2 \approx 0.00114$	$\delta \approx 0.02112$ $\sigma^2 \approx 0.00103$	$\delta \approx 0.01989$ $\sigma^2 \approx 0.00087$
Extrema Prop. $K = 10 \quad \tau = 2$	$\delta \approx 0.07843$ $\sigma^2 \approx 0.00549$	$\delta \approx 0.10307$ $\sigma^2 \approx 0.00457$	$\delta \approx 0.12114$ $\sigma^2 \approx 0.00402$	$\delta \approx 0.13536$ $\sigma^2 \approx 0.00395$	$\delta \approx 0.15071$ $\sigma^2 \approx 0.00384$	$\delta \approx 0.16213$ $\sigma^2 \approx 0.00388$	$\delta \approx 0.17532$ $\sigma^2 \approx 0.00382$	$\delta \approx 0.18821$ $\sigma^2 \approx 0.00386$	$\delta \approx 0.19699$ $\sigma^2 \approx 0.00378$	$\delta \approx 0.20767$ $\sigma^2 \approx 0.00394$
Extrema Prop. $K = 10 \quad \tau = 4$	$\delta \approx 0.05951$ $\sigma^2 \approx 0.00359$	$\delta \approx 0.07237$ $\sigma^2 \approx 0.00324$	$\delta \approx 0.08170$ $\sigma^2 \approx 0.00317$	$\delta \approx 0.08862$ $\sigma^2 \approx 0.00298$	$\delta \approx 0.09451$ $\sigma^2 \approx 0.00287$	$\delta \approx 0.09980$ $\sigma^2 \approx 0.00289$	$\delta \approx 0.10392$ $\sigma^2 \approx 0.00274$	$\delta \approx 0.10787$ $\sigma^2 \approx 0.00281$	$\delta \approx 0.11279$ $\sigma^2 \approx 0.00279$	$\delta \approx 0.11734$ $\sigma^2 \approx 0.00278$
Extrema Prop. $K = 10 \quad \tau = 6$	$\delta \approx 0.05653$ $\sigma^2 \approx 0.00337$	$\delta \approx 0.06950$ $\sigma^2 \approx 0.00317$	$\delta \approx 0.07525$ $\sigma^2 \approx 0.00298$	$\delta \approx 0.08029$ $\sigma^2 \approx 0.00298$	$\delta \approx 0.08360$ $\sigma^2 \approx 0.00285$	$\delta \approx 0.08522$ $\sigma^2 \approx 0.00268$	$\delta \approx 0.08856$ $\sigma^2 \approx 0.00258$	$\delta \approx 0.09136$ $\sigma^2 \approx 0.00255$	$\delta \approx 0.09314$ $\sigma^2 \approx 0.00249$	$\delta \approx 0.09526$ $\sigma^2 \approx 0.00254$
Extrema Prop. $K = 50 \quad \tau = 2$	$\delta \approx 0.00984$ $\sigma^2 \approx 0.00069$	$\delta \approx 0.02041$ $\sigma^2 \approx 0.00104$	$\delta \approx 0.02679$ $\sigma^2 \approx 0.00113$	$\delta \approx 0.03317$ $\sigma^2 \approx 0.00120$	$\delta \approx 0.03885$ $\sigma^2 \approx 0.00122$	$\delta \approx 0.04263$ $\sigma^2 \approx 0.00126$	$\delta \approx 0.04584$ $\sigma^2 \approx 0.00124$	$\delta \approx 0.05019$ $\sigma^2 \approx 0.00128$	$\delta \approx 0.05270$ $\sigma^2 \approx 0.00129$	$\delta \approx 0.05603$ $\sigma^2 \approx 0.00124$
Extrema Prop. $K = 50 \quad \tau = 4$	$\delta \approx 0.00904$ $\sigma^2 \approx 0.00066$	$\delta \approx 0.01762$ $\sigma^2 \approx 0.00091$	$\delta \approx 0.02325$ $\sigma^2 \approx 0.00095$	$\delta \approx 0.02573$ $\sigma^2 \approx 0.00093$	$\delta \approx 0.02946$ $\sigma^2 \approx 0.00091$	$\delta \approx 0.03176$ $\sigma^2 \approx 0.00098$	$\delta \approx 0.03305$ $\sigma^2 \approx 0.00094$	$\delta \approx 0.03454$ $\sigma^2 \approx 0.00091$	$\delta \approx 0.03574$ $\sigma^2 \approx 0.00089$	$\delta \approx 0.03671$ $\sigma^2 \approx 0.00089$
Extrema Prop. $K = 50 \quad \tau = 6$	$\delta \approx 0.00946$ $\sigma^2 \approx 0.00068$	$\delta \approx 0.01788$ $\sigma^2 \approx 0.00089$	$\delta \approx 0.02200$ $\sigma^2 \approx 0.00094$	$\delta \approx 0.02546$ $\sigma^2 \approx 0.00092$	$\delta \approx 0.02835$ $\sigma^2 \approx 0.00093$	$\delta \approx 0.03026$ $\sigma^2 \approx 0.00094$	$\delta \approx 0.03203$ $\sigma^2 \approx 0.00090$	$\delta \approx 0.03302$ $\sigma^2 \approx 0.00088$	$\delta \approx 0.03397$ $\sigma^2 \approx 0.00087$	$\delta \approx 0.03534$ $\sigma^2 \approx 0.00088$
Extrema Prop. $K = 100 \quad \tau = 2$	$\delta \approx 0.00232$ $\sigma^2 \approx 0.00017$	$\delta \approx 0.00716$ $\sigma^2 \approx 0.00040$	$\delta \approx 0.01124$ $\sigma^2 \approx 0.00052$	$\delta \approx 0.01510$ $\sigma^2 \approx 0.00061$	$\delta \approx 0.01853$ $\sigma^2 \approx 0.00063$	$\delta \approx 0.02061$ $\sigma^2 \approx 0.00067$	$\delta \approx 0.02256$ $\sigma^2 \approx 0.00066$	$\delta \approx 0.02498$ $\sigma^2 \approx 0.00070$	$\delta \approx 0.02588$ $\sigma^2 \approx 0.00067$	$\delta \approx 0.02767$ $\sigma^2 \approx 0.00069$
Extrema Prop. $K = 100 \quad \tau = 4$	$\delta \approx 0.00248$ $\sigma^2 \approx 0.00019$	$\delta \approx 0.00653$ $\sigma^2 \approx 0.00036$	$\delta \approx 0.01054$ $\sigma^2 \approx 0.00047$	$\delta \approx 0.01221$ $\sigma^2 \approx 0.00048$	$\delta \approx 0.01580$ $\sigma^2 \approx 0.00054$	$\delta \approx 0.01688$ $\sigma^2 \approx 0.00051$	$\delta \approx 0.01840$ $\sigma^2 \approx 0.00053$	$\delta \approx 0.01951$ $\sigma^2 \approx 0.00053$	$\delta \approx 0.02028$ $\sigma^2 \approx 0.00051$	$\delta \approx 0.02147$ $\sigma^2 \approx 0.00051$
Extrema Prop. $K = 100 \quad \tau = 6$	$\delta \approx 0.00253$ $\sigma^2 \approx 0.00019$	$\delta \approx 0.00676$ $\sigma^2 \approx 0.00037$	$\delta \approx 0.01028$ $\sigma^2 \approx 0.00046$	$\delta \approx 0.01231$ $\sigma^2 \approx 0.00048$	$\delta \approx 0.01524$ $\sigma^2 \approx 0.00051$	$\delta \approx 0.01626$ $\sigma^2 \approx 0.00050$	$\delta \approx 0.01786$ $\sigma^2 \approx 0.00052$	$\delta \approx 0.01923$ $\sigma^2 \approx 0.00051$	$\delta \approx 0.02036$ $\sigma^2 \approx 0.00051$	$\delta \approx 0.02072$ $\sigma^2 \approx 0.00050$
Extrema Prop. $K = 200 \quad \tau = 2$	$\delta \approx 0.00021$ $\sigma^2 \approx 0.00002$	$\delta \approx 0.00181$ $\sigma^2 \approx 0.00011$	$\delta \approx 0.00349$ $\sigma^2 \approx 0.00017$	$\delta \approx 0.00532$ $\sigma^2 \approx 0.00022$	$\delta \approx 0.00704$ $\sigma^2 \approx 0.00026$	$\delta \approx 0.00861$ $\sigma^2 \approx 0.00029$	$\delta \approx 0.00965$ $\sigma^2 \approx 0.00030$	$\delta \approx 0.01087$ $\sigma^2 \approx 0.00032$	$\delta \approx 0.01222$ $\sigma^2 \approx 0.00033$	$\delta \approx 0.01292$ $\sigma^2 \approx 0.00034$
Extrema Prop. $K = 200 \quad \tau = 4$	$\delta \approx 0.00020$ $\sigma^2 \approx 0.00001$	$\delta \approx 0.00133$ $\sigma^2 \approx 0.00008$	$\delta \approx 0.00333$ $\sigma^2 \approx 0.00016$	$\delta \approx 0.00471$ $\sigma^2 \approx 0.00020$	$\delta \approx 0.00621$ $\sigma^2 \approx 0.00023$	$\delta \approx 0.00745$ $\sigma^2 \approx 0.00025$	$\delta \approx 0.00873$ $\sigma^2 \approx 0.00027$	$\delta \approx 0.01004$ $\sigma^2 \approx 0.00029$	$\delta \approx 0.01005$ $\sigma^2 \approx 0.00027$	$\delta \approx 0.01110$ $\sigma^2 \approx 0.00028$
Extrema Prop. $K = 200 \quad \tau = 6$	$\delta \approx 0.00031$ $\sigma^2 \approx 0.00002$	$\delta \approx 0.00145$ $\sigma^2 \approx 0.00008$	$\delta \approx 0.00334$ $\sigma^2 \approx 0.00016$	$\delta \approx 0.00465$ $\sigma^2 \approx 0.00020$	$\delta \approx 0.00624$ $\sigma^2 \approx 0.00023$	$\delta \approx 0.00762$ $\sigma^2 \approx 0.00026$	$\delta \approx 0.00879$ $\sigma^2 \approx 0.00028$	$\delta \approx 0.00933$ $\sigma^2 \approx 0.00027$	$\delta \approx 0.01003$ $\sigma^2 \approx 0.00027$	$\delta \approx 0.01119$ $\sigma^2 \approx 0.00028$
Extrema Prop. $K = 300 \quad \tau = 2$	$\delta \approx 0.00006$ $\sigma^2 \approx 0.00000$	$\delta \approx 0.00052$ $\sigma^2 \approx 0.00003$	$\delta \approx 0.00118$ $\sigma^2 \approx 0.00006$	$\delta \approx 0.00227$ $\sigma^2 \approx 0.00010$	$\delta \approx 0.00370$ $\sigma^2 \approx 0.00015$	$\delta \approx 0.00411$ $\sigma^2 \approx 0.00015$	$\delta \approx 0.00503$ $\sigma^2 \approx 0.00017$	$\delta \approx 0.00605$ $\sigma^2 \approx 0.00019$	$\delta \approx 0.00678$ $\sigma^2 \approx 0.00020$	$\delta \approx 0.00780$ $\sigma^2 \approx 0.00022$
Extrema Prop. $K = 300 \quad \tau = 4$	$\delta \approx 0.00005$ $\sigma^2 \approx 0.00000$	$\delta \approx 0.00054$ $\sigma^2 \approx 0.00003$	$\delta \approx 0.00125$ $\sigma^2 \approx 0.00006$	$\delta \approx 0.00243$ $\sigma^2 \approx 0.00010$	$\delta \approx 0.00331$ $\sigma^2 \approx 0.00013$	$\delta \approx 0.00401$ $\sigma^2 \approx 0.00015$	$\delta \approx 0.00483$ $\sigma^2 \approx 0.00016$	$\delta \approx 0.00547$ $\sigma^2 \approx 0.00017$	$\delta \approx 0.00605$ $\sigma^2 \approx 0.00017$	$\delta \approx 0.00684$ $\sigma^2 \approx 0.00018$
Extrema Prop. $K = 300 \quad \tau = 6$	$\delta \approx 0.00003$ $\sigma^2 \approx 0.00000$	$\delta \approx 0.00045$ $\sigma^2 \approx 0.00003$	$\delta \approx 0.00119$ $\sigma^2 \approx 0.00006$	$\delta \approx 0.00215$ $\sigma^2 \approx 0.00010$	$\delta \approx 0.00314$ $\sigma^2 \approx 0.00013$	$\delta \approx 0.00418$ $\sigma^2 \approx 0.00015$	$\delta \approx 0.00489$ $\sigma^2 \approx 0.00016$	$\delta \approx 0.00558$ $\sigma^2 \approx 0.00017$	$\delta \approx 0.00613$ $\sigma^2 \approx 0.00018$	$\delta \approx 0.00672$ $\sigma^2 \approx 0.00018$

5.2 Time Complexity

Average amount of time $\bar{T}$ used by the algorithms, measured in the number of time slots, along with the variance, is included in the table 2.

Figure 11 shows how much time BFST Building method needs comparing to Extrema Propagation, more closely, to its most ($K = 10$, $\tau = 2$) and least ($K = 50$, $\tau = 6$) time-consuming parameters that were chosen.

The optimal value of K, as it comes to time usage, depends on the values of T and n. Smaller K ends the algorithm more quickly when τ is small and n is big. Unfortunately, as we have noticed in the precious subsection, the accuracy decreases significantly then.

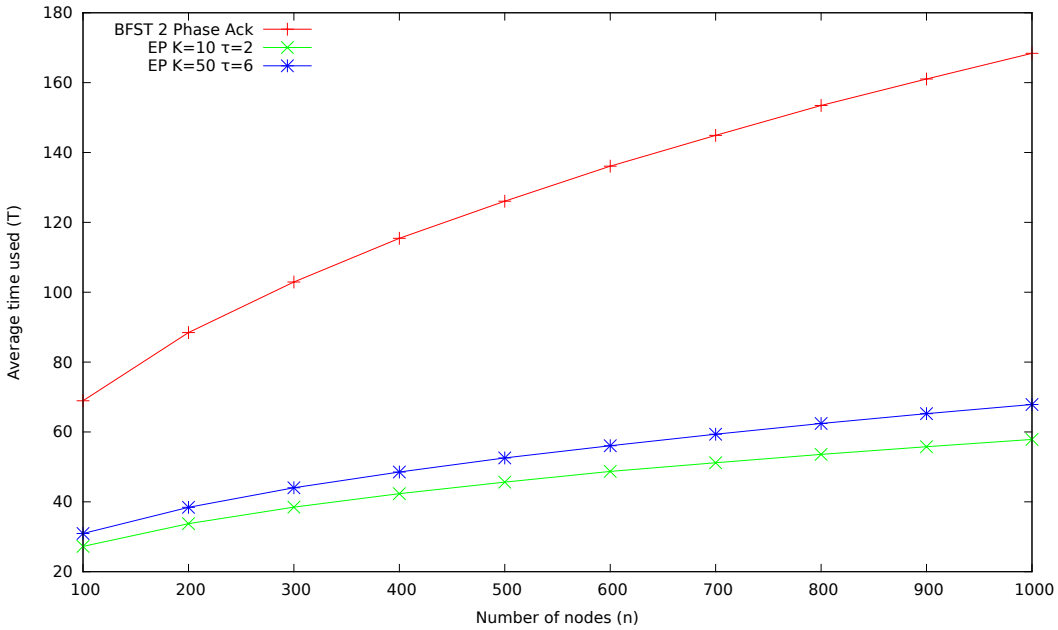

Figure 11: Average time slots usage of BFST Building and Extrema Propagation

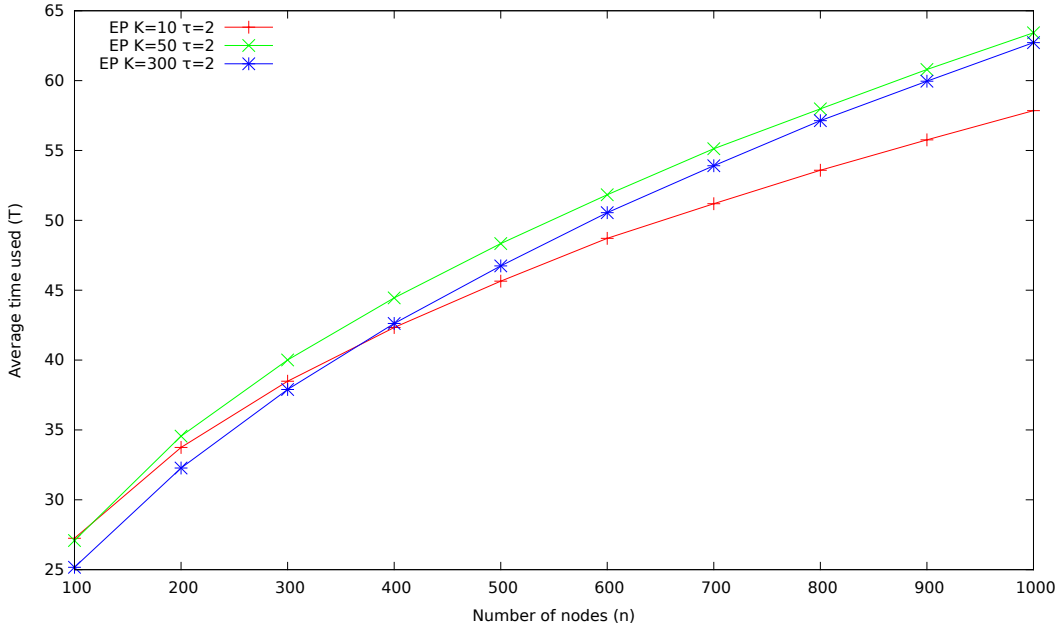

Figure 12: Average time slots usage of Extrema Propagation for $\tau = 2$ and various values of K

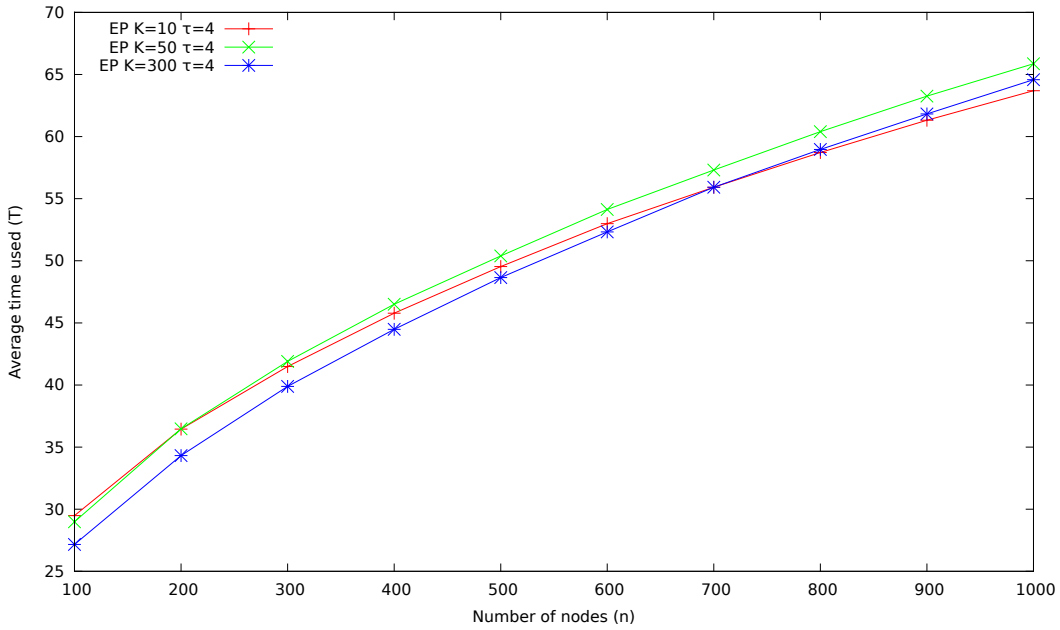

Figure 13: Average time slots usage of Extrema Propagation for $\tau = 4$ and various values of K

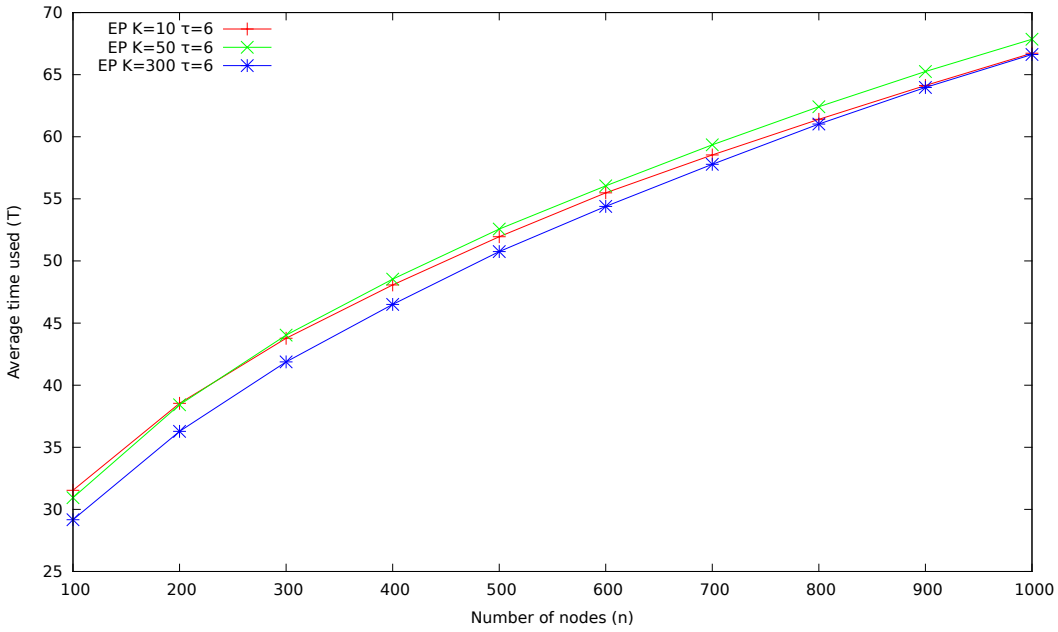

Figure 14: Average time slots usage of Extrema Propagation for $\tau = 6$ and various values of K

Table 2: Experimental results: average number of time slots used with the variance

	$n = 100$ $p = 0.16248$	$n = 200$ $p = 0.11959$	$n = 300$ $p = 0.09982$	$n = 400$ $p = 0.08776$	$n = 500$ $p = 0.07940$	$n = 600$ $p = 0.07314$	$n = 700$ $p = 0.06823$	$n = 800$ $p = 0.06424$	$n = 900$ $p = 0.06091$	$n = 1000$ $p = 0.05807$
BFST building 2 Phases, ACK	$\bar{T} \approx 68.930$ $\sigma^2 \approx 113.116$	$\bar{T} \approx 88.443$ $\sigma^2 \approx 86.670$	$\bar{T} \approx 102.940$ $\sigma^2 \approx 79.249$	$\bar{T} \approx 115.445$ $\sigma^2 \approx 79.595$	$\bar{T} \approx 126.051$ $\sigma^2 \approx 85.360$	$\bar{T} \approx 136.084$ $\sigma^2 \approx 92.006$	$\bar{T} \approx 144.891$ $\sigma^2 \approx 96.471$	$\bar{T} \approx 153.467$ $\sigma^2 \approx 105.700$	$\bar{T} \approx 161.052$ $\sigma^2 \approx 112.572$	$\bar{T} \approx 168.396$ $\sigma^2 \approx 118.551$
Extrema Prop. $K = 10 \quad \tau = 2$	$\bar{T} \approx 27.245$ $\sigma^2 \approx 14.982$	$\bar{T} \approx 33.745$ $\sigma^2 \approx 13.170$	$\bar{T} \approx 38.487$ $\sigma^2 \approx 12.187$	$\bar{T} \approx 42.328$ $\sigma^2 \approx 12.638$	$\bar{T} \approx 45.647$ $\sigma^2 \approx 13.520$	$\bar{T} \approx 48.714$ $\sigma^2 \approx 14.755$	$\bar{T} \approx 51.192$ $\sigma^2 \approx 16.354$	$\bar{T} \approx 53.582$ $\sigma^2 \approx 17.909$	$\bar{T} \approx 55.761$ $\sigma^2 \approx 19.165$	$\bar{T} \approx 57.855$ $\sigma^2 \approx 19.538$
Extrema Prop. $K = 10 \quad \tau = 4$	$\bar{T} \approx 29.488$ $\sigma^2 \approx 18.131$	$\bar{T} \approx 36.452$ $\sigma^2 \approx 15.051$	$\bar{T} \approx 41.487$ $\sigma^2 \approx 13.216$	$\bar{T} \approx 45.779$ $\sigma^2 \approx 12.734$	$\bar{T} \approx 49.536$ $\sigma^2 \approx 13.456$	$\bar{T} \approx 52.992$ $\sigma^2 \approx 14.552$	$\bar{T} \approx 55.924$ $\sigma^2 \approx 15.389$	$\bar{T} \approx 58.718$ $\sigma^2 \approx 16.818$	$\bar{T} \approx 61.314$ $\sigma^2 \approx 17.382$	$\bar{T} \approx 63.692$ $\sigma^2 \approx 18.064$
Extrema Prop. $K = 10 \quad \tau = 6$	$\bar{T} \approx 31.535$ $\sigma^2 \approx 18.429$	$\bar{T} \approx 38.537$ $\sigma^2 \approx 14.857$	$\bar{T} \approx 43.778$ $\sigma^2 \approx 12.872$	$\bar{T} \approx 48.072$ $\sigma^2 \approx 13.048$	$\bar{T} \approx 51.950$ $\sigma^2 \approx 13.331$	$\bar{T} \approx 55.485$ $\sigma^2 \approx 14.115$	$\bar{T} \approx 58.540$ $\sigma^2 \approx 14.459$	$\bar{T} \approx 61.405$ $\sigma^2 \approx 16.437$	$\bar{T} \approx 64.126$ $\sigma^2 \approx 16.383$	$\bar{T} \approx 66.725$ $\sigma^2 \approx 16.956$
Extrema Prop. $K = 50 \quad \tau = 2$	$\bar{T} \approx 27.083$ $\sigma^2 \approx 19.843$	$\bar{T} \approx 34.555$ $\sigma^2 \approx 18.724$	$\bar{T} \approx 40.007$ $\sigma^2 \approx 16.249$	$\bar{T} \approx 44.448$ $\sigma^2 \approx 17.014$	$\bar{T} \approx 48.337$ $\sigma^2 \approx 17.793$	$\bar{T} \approx 51.830$ $\sigma^2 \approx 18.718$	$\bar{T} \approx 55.124$ $\sigma^2 \approx 18.330$	$\bar{T} \approx 57.980$ $\sigma^2 \approx 19.877$	$\bar{T} \approx 60.796$ $\sigma^2 \approx 19.801$	$\bar{T} \approx 63.422$ $\sigma^2 \approx 20.091$
Extrema Prop. $K = 50 \quad \tau = 4$	$\bar{T} \approx 28.984$ $\sigma^2 \approx 18.962$	$\bar{T} \approx 36.473$ $\sigma^2 \approx 17.746$	$\bar{T} \approx 41.896$ $\sigma^2 \approx 16.933$	$\bar{T} \approx 46.504$ $\sigma^2 \approx 17.345$	$\bar{T} \approx 50.394$ $\sigma^2 \approx 18.287$	$\bar{T} \approx 54.128$ $\sigma^2 \approx 19.122$	$\bar{T} \approx 57.309$ $\sigma^2 \approx 19.871$	$\bar{T} \approx 60.401$ $\sigma^2 \approx 21.185$	$\bar{T} \approx 63.259$ $\sigma^2 \approx 21.463$	$\bar{T} \approx 65.863$ $\sigma^2 \approx 22.415$
Extrema Prop. $K = 50 \quad \tau = 6$	$\bar{T} \approx 30.939$ $\sigma^2 \approx 19.252$	$\bar{T} \approx 38.418$ $\sigma^2 \approx 17.374$	$\bar{T} \approx 44.032$ $\sigma^2 \approx 16.608$	$\bar{T} \approx 48.535$ $\sigma^2 \approx 16.757$	$\bar{T} \approx 52.572$ $\sigma^2 \approx 17.389$	$\bar{T} \approx 56.050$ $\sigma^2 \approx 19.524$	$\bar{T} \approx 59.355$ $\sigma^2 \approx 19.884$	$\bar{T} \approx 62.422$ $\sigma^2 \approx 20.892$	$\bar{T} \approx 65.253$ $\sigma^2 \approx 21.800$	$\bar{T} \approx 67.855$ $\sigma^2 \approx 22.514$
Extrema Prop. $K = 100 \quad \tau = 2$	$\bar{T} \approx 26.267$ $\sigma^2 \approx 17.584$	$\bar{T} \approx 33.992$ $\sigma^2 \approx 18.889$	$\bar{T} \approx 39.689$ $\sigma^2 \approx 18.192$	$\bar{T} \approx 44.221$ $\sigma^2 \approx 19.461$	$\bar{T} \approx 48.261$ $\sigma^2 \approx 20.184$	$\bar{T} \approx 51.952$ $\sigma^2 \approx 21.874$	$\bar{T} \approx 55.313$ $\sigma^2 \approx 22.090$	$\bar{T} \approx 58.284$ $\sigma^2 \approx 23.363$	$\bar{T} \approx 61.160$ $\sigma^2 \approx 23.589$	$\bar{T} \approx 63.897$ $\sigma^2 \approx 23.681$
Extrema Prop. $K = 100 \quad \tau = 4$	$\bar{T} \approx 28.256$ $\sigma^2 \approx 17.797$	$\bar{T} \approx 35.898$ $\sigma^2 \approx 18.022$	$\bar{T} \approx 41.403$ $\sigma^2 \approx 18.455$	$\bar{T} \approx 46.131$ $\sigma^2 \approx 19.659$	$\bar{T} \approx 50.167$ $\sigma^2 \approx 21.094$	$\bar{T} \approx 53.875$ $\sigma^2 \approx 22.392$	$\bar{T} \approx 57.232$ $\sigma^2 \approx 22.424$	$\bar{T} \approx 60.199$ $\sigma^2 \approx 25.247$	$\bar{T} \approx 63.014$ $\sigma^2 \approx 26.140$	$\bar{T} \approx 65.606$ $\sigma^2 \approx 27.083$
Extrema Prop. $K = 100 \quad \tau = 6$	$\bar{T} \approx 30.206$ $\sigma^2 \approx 16.645$	$\bar{T} \approx 37.799$ $\sigma^2 \approx 18.282$	$\bar{T} \approx 43.449$ $\sigma^2 \approx 18.350$	$\bar{T} \approx 48.074$ $\sigma^2 \approx 19.180$	$\bar{T} \approx 52.196$ $\sigma^2 \approx 20.203$	$\bar{T} \approx 55.832$ $\sigma^2 \approx 22.483$	$\bar{T} \approx 59.137$ $\sigma^2 \approx 23.377$	$\bar{T} \approx 62.280$ $\sigma^2 \approx 24.571$	$\bar{T} \approx 64.948$ $\sigma^2 \approx 26.200$	$\bar{T} \approx 67.740$ $\sigma^2 \approx 25.975$
Extrema Prop. $K = 200 \quad \tau = 2$	$\bar{T} \approx 25.435$ $\sigma^2 \approx 13.796$	$\bar{T} \approx 32.970$ $\sigma^2 \approx 17.138$	$\bar{T} \approx 38.617$ $\sigma^2 \approx 19.257$	$\bar{T} \approx 43.242$ $\sigma^2 \approx 21.168$	$\bar{T} \approx 47.542$ $\sigma^2 \approx 22.838$	$\bar{T} \approx 51.194$ $\sigma^2 \approx 25.784$	$\bar{T} \approx 54.578$ $\sigma^2 \approx 27.074$	$\bar{T} \approx 57.736$ $\sigma^2 \approx 28.663$	$\bar{T} \approx 60.615$ $\sigma^2 \approx 29.722$	$\bar{T} \approx 63.407$ $\sigma^2 \approx 30.046$
Extrema Prop. $K = 200 \quad \tau = 4$	$\bar{T} \approx 27.458$ $\sigma^2 \approx 13.863$	$\bar{T} \approx 34.938$ $\sigma^2 \approx 16.517$	$\bar{T} \approx 40.564$ $\sigma^2 \approx 18.975$	$\bar{T} \approx 45.286$ $\sigma^2 \approx 20.606$	$\bar{T} \approx 49.489$ $\sigma^2 \approx 22.526$	$\bar{T} \approx 53.172$ $\sigma^2 \approx 24.812$	$\bar{T} \approx 56.523$ $\sigma^2 \approx 26.928$	$\bar{T} \approx 59.603$ $\sigma^2 \approx 28.430$	$\bar{T} \approx 62.459$ $\sigma^2 \approx 29.594$	$\bar{T} \approx 65.125$ $\sigma^2 \approx 31.579$
Extrema Prop. $K = 200 \quad \tau = 6$	$\bar{T} \approx 29.422$ $\sigma^2 \approx 13.757$	$\bar{T} \approx 36.938$ $\sigma^2 \approx 17.173$	$\bar{T} \approx 42.553$ $\sigma^2 \approx 18.687$	$\bar{T} \approx 47.349$ $\sigma^2 \approx 20.625$	$\bar{T} \approx 51.397$ $\sigma^2 \approx 22.533$	$\bar{T} \approx 55.115$ $\sigma^2 \approx 25.561$	$\bar{T} \approx 58.450$ $\sigma^2 \approx 26.293$	$\bar{T} \approx 61.621$ $\sigma^2 \approx 27.877$	$\bar{T} \approx 64.459$ $\sigma^2 \approx 29.524$	$\bar{T} \approx 67.150$ $\sigma^2 \approx 31.226$
Extrema Prop. $K = 300 \quad \tau = 2$	$\bar{T} \approx 25.167$ $\sigma^2 \approx 12.173$	$\bar{T} \approx 32.277$ $\sigma^2 \approx 14.395$	$\bar{T} \approx 37.897$ $\sigma^2 \approx 18.096$	$\bar{T} \approx 42.622$ $\sigma^2 \approx 21.431$	$\bar{T} \approx 46.740$ $\sigma^2 \approx 22.991$	$\bar{T} \approx 50.547$ $\sigma^2 \approx 26.436$	$\bar{T} \approx 53.905$ $\sigma^2 \approx 28.515$	$\bar{T} \approx 57.131$ $\sigma^2 \approx 30.497$	$\bar{T} \approx 59.960$ $\sigma^2 \approx 31.981$	$\bar{T} \approx 62.715$ $\sigma^2 \approx 33.498$
Extrema Prop. $K = 300 \quad \tau = 4$	$\bar{T} \approx 27.166$ $\sigma^2 \approx 12.217$	$\bar{T} \approx 34.319$ $\sigma^2 \approx 15.041$	$\bar{T} \approx 39.883$ $\sigma^2 \approx 17.769$	$\bar{T} \approx 44.484$ $\sigma^2 \approx 20.912$	$\bar{T} \approx 48.649$ $\sigma^2 \approx 23.147$	$\bar{T} \approx 52.324$ $\sigma^2 \approx 26.226$	$\bar{T} \approx 55.924$ $\sigma^2 \approx 28.415$	$\bar{T} \approx 58.967$ $\sigma^2 \approx 30.778$	$\bar{T} \approx 61.825$ $\sigma^2 \approx 31.716$	$\bar{T} \approx 64.581$ $\sigma^2 \approx 34.602$
Extrema Prop. $K = 300 \quad \tau = 6$	$\bar{T} \approx 29.171$ $\sigma^2 \approx 12.399$	$\bar{T} \approx 36.281$ $\sigma^2 \approx 14.811$	$\bar{T} \approx 41.875$ $\sigma^2 \approx 17.713$	$\bar{T} \approx 46.502$ $\sigma^2 \approx 20.762$	$\bar{T} \approx 50.752$ $\sigma^2 \approx 23.088$	$\bar{T} \approx 54.393$ $\sigma^2 \approx 25.965$	$\bar{T} \approx 57.781$ $\sigma^2 \approx 27.876$	$\bar{T} \approx 61.027$ $\sigma^2 \approx 29.925$	$\bar{T} \approx 63.968$ $\sigma^2 \approx 32.142$	$\bar{T} \approx 66.622$ $\sigma^2 \approx 33.040$

5.3 Energy Complexity

Average amount of energy $\bar{E}$ used by the algorithms, measured in the number of message transmissions, along with the variance, is included in the table 3.

The biggest disadvantage of Extrema Propagation algorithm is unveiled in the figure 15. The most ($K = 300$, $\tau = 6$) and least ($K = 10$, $\tau = 2$) energy-consuming versions of EP has been compared to BFST Building method. Tree building approach turns out to be much more energy saving, as it was shown in the theoretical discussion performed in the section 3.

Figure 16 shows the obvious fact, that greater τ causes more message transmissions.

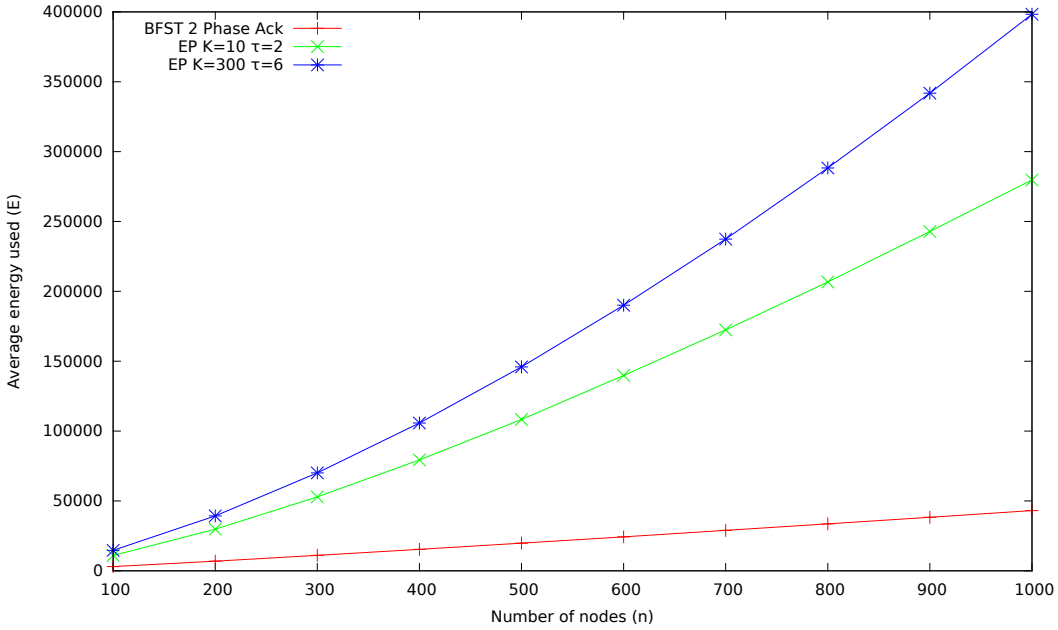

Figure 15: Average number of message transmissions of BFST Building and Extrema Propagation

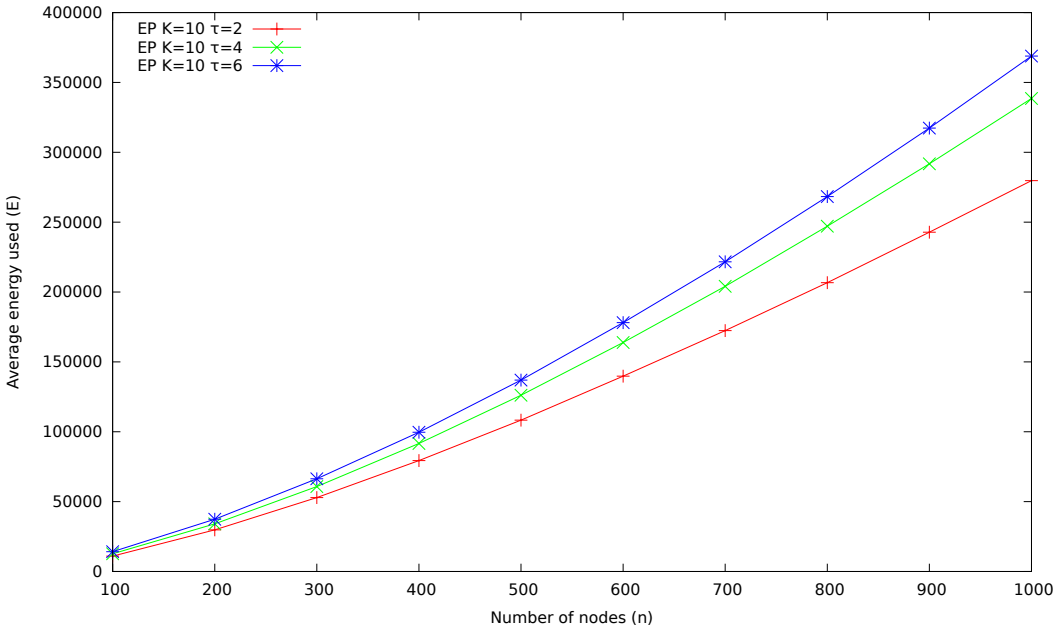

Figure 16: Average number of message transmissions of Extrema Propagation for $K = 10$

Table 3: Experimental results: average number of message transmission with the variance

	$n = 100$ $p = 0.16248$	$n = 200$ $p = 0.11959$	$n = 300$ $p = 0.09982$	$n = 400$ $p = 0.08776$	$n = 500$ $p = 0.07940$	$n = 600$ $p = 0.07314$	$n = 700$ $p = 0.06823$	$n = 800$ $p = 0.06424$	$n = 900$ $p = 0.06091$	$n = 1000$ $p = 0.05807$
BFST building 2 Phases, ACK	$\bar{E} \approx 3062$ $\sigma^2 \approx 31156$	$\bar{E} \approx 6924$ $\sigma^2 \approx 71888$	$\bar{E} \approx 11058$ $\sigma^2 \approx 110836$	$\bar{E} \approx 15377$ $\sigma^2 \approx 153044$	$\bar{E} \approx 19787$ $\sigma^2 \approx 191728$	$\bar{E} \approx 24324$ $\sigma^2 \approx 227934$	$\bar{E} \approx 28925$ $\sigma^2 \approx 268488$	$\bar{E} \approx 33610$ $\sigma^2 \approx 322641$	$\bar{E} \approx 38324$ $\sigma^2 \approx 357108$	$\bar{E} \approx 43121$ $\sigma^2 \approx 407812$
Extrema Prop. $K = 10 \quad \tau = 2$	$\bar{E} \approx 11017$ $\sigma^2 \approx 1390509$	$\bar{E} \approx 29764$ $\sigma^2 \approx 5540583$	$\bar{E} \approx 52918$ $\sigma^2 \approx 12831234$	$\bar{E} \approx 79396$ $\sigma^2 \approx 24985001$	$\bar{E} \approx 108277$ $\sigma^2 \approx 44426499$	$\bar{E} \approx 139771$ $\sigma^2 \approx 76639737$	$\bar{E} \approx 172428$ $\sigma^2 \approx 115524238$	$\bar{E} \approx 206714$ $\sigma^2 \approx 174934279$	$\bar{E} \approx 242791$ $\sigma^2 \approx 238095749$	$\bar{E} \approx 279686$ $\sigma^2 \approx 337521804$
Extrema Prop. $K = 10 \quad \tau = 4$	$\bar{E} \approx 12682$ $\sigma^2 \approx 1972228$	$\bar{E} \approx 34094$ $\sigma^2 \approx 7321193$	$\bar{E} \approx 60776$ $\sigma^2 \approx 15459402$	$\bar{E} \approx 91636$ $\sigma^2 \approx 27747309$	$\bar{E} \approx 126110$ $\sigma^2 \approx 44280274$	$\bar{E} \approx 163842$ $\sigma^2 \approx 75592178$	$\bar{E} \approx 204065$ $\sigma^2 \approx 108610478$	$\bar{E} \approx 247138$ $\sigma^2 \approx 159829944$	$\bar{E} \approx 291740$ $\sigma^2 \approx 223216557$	$\bar{E} \approx 338554$ $\sigma^2 \approx 298203865$
Extrema Prop. $K = 10 \quad \tau = 6$	$\bar{E} \approx 14139$ $\sigma^2 \approx 2258278$	$\bar{E} \approx 37415$ $\sigma^2 \approx 7995297$	$\bar{E} \approx 66348$ $\sigma^2 \approx 17221291$	$\bar{E} \approx 99665$ $\sigma^2 \approx 30757626$	$\bar{E} \approx 136971$ $\sigma^2 \approx 48795512$	$\bar{E} \approx 178073$ $\sigma^2 \approx 77536433$	$\bar{E} \approx 221593$ $\sigma^2 \approx 110977404$	$\bar{E} \approx 268281$ $\sigma^2 \approx 162699314$	$\bar{E} \approx 317346$ $\sigma^2 \approx 211252312$	$\bar{E} \approx 368870$ $\sigma^2 \approx 301570288$
Extrema Prop. $K = 50 \quad \tau = 2$	$\bar{E} \approx 11730$ $\sigma^2 \approx 2013180$	$\bar{E} \approx 32297$ $\sigma^2 \approx 6763667$	$\bar{E} \approx 58464$ $\sigma^2 \approx 13099552$	$\bar{E} \approx 88999$ $\sigma^2 \approx 21491953$	$\bar{E} \approx 123424$ $\sigma^2 \approx 32960061$	$\bar{E} \approx 161540$ $\sigma^2 \approx 49919852$	$\bar{E} \approx 202488$ $\sigma^2 \approx 69088821$	$\bar{E} \approx 246281$ $\sigma^2 \approx 98484592$	$\bar{E} \approx 292585$ $\sigma^2 \approx 124543089$	$\bar{E} \approx 341251$ $\sigma^2 \approx 162031363$
Extrema Prop. $K = 50 \quad \tau = 4$	$\bar{E} \approx 13158$ $\sigma^2 \approx 2172483$	$\bar{E} \approx 35599$ $\sigma^2 \approx 7442821$	$\bar{E} \approx 63822$ $\sigma^2 \approx 14332584$	$\bar{E} \approx 96804$ $\sigma^2 \approx 23062202$	$\bar{E} \approx 133804$ $\sigma^2 \approx 35508915$	$\bar{E} \approx 174694$ $\sigma^2 \approx 54881619$	$\bar{E} \approx 218652$ $\sigma^2 \approx 73247935$	$\bar{E} \approx 265968$ $\sigma^2 \approx 100737634$	$\bar{E} \approx 315756$ $\sigma^2 \approx 131510852$	$\bar{E} \approx 368414$ $\sigma^2 \approx 167948443$
Extrema Prop. $K = 50 \quad \tau = 6$	$\bar{E} \approx 14569$ $\sigma^2 \approx 2320009$	$\bar{E} \approx 38800$ $\sigma^2 \approx 7745030$	$\bar{E} \approx 69020$ $\sigma^2 \approx 15047620$	$\bar{E} \approx 103975$ $\sigma^2 \approx 24486994$	$\bar{E} \approx 143138$ $\sigma^2 \approx 36937231$	$\bar{E} \approx 186272$ $\sigma^2 \approx 59227253$	$\bar{E} \approx 232370$ $\sigma^2 \approx 78347860$	$\bar{E} \approx 282051$ $\sigma^2 \approx 108230490$	$\bar{E} \approx 334207$ $\sigma^2 \approx 136471208$	$\bar{E} \approx 389089$ $\sigma^2 \approx 171636689$
Extrema Prop. $K = 100 \quad \tau = 2$	$\bar{E} \approx 11805$ $\sigma^2 \approx 2059032$	$\bar{E} \approx 32665$ $\sigma^2 \approx 7003811$	$\bar{E} \approx 59274$ $\sigma^2 \approx 13022692$	$\bar{E} \approx 90463$ $\sigma^2 \approx 20125603$	$\bar{E} \approx 125763$ $\sigma^2 \approx 31278200$	$\bar{E} \approx 164939$ $\sigma^2 \approx 48535595$	$\bar{E} \approx 207054$ $\sigma^2 \approx 63591382$	$\bar{E} \approx 252375$ $\sigma^2 \approx 87181807$	$\bar{E} \approx 300421$ $\sigma^2 \approx 106056673$	$\bar{E} \approx 351020$ $\sigma^2 \approx 140279319$
Extrema Prop. $K = 100 \quad \tau = 4$	$\bar{E} \approx 13219$ $\sigma^2 \approx 2193508$	$\bar{E} \approx 35898$ $\sigma^2 \approx 7459860$	$\bar{E} \approx 64463$ $\sigma^2 \approx 14317831$	$\bar{E} \approx 97866$ $\sigma^2 \approx 22203165$	$\bar{E} \approx 135326$ $\sigma^2 \approx 34958677$	$\bar{E} \approx 176904$ $\sigma^2 \approx 53959199$	$\bar{E} \approx 221422$ $\sigma^2 \approx 71210873$	$\bar{E} \approx 269457$ $\sigma^2 \approx 95530459$	$\bar{E} \approx 320066$ $\sigma^2 \approx 119882541$	$\bar{E} \approx 373432$ $\sigma^2 \approx 149164252$
Extrema Prop. $K = 100 \quad \tau = 6$	$\bar{E} \approx 14633$ $\sigma^2 \approx 2331899$	$\bar{E} \approx 39099$ $\sigma^2 \approx 7813240$	$\bar{E} \approx 69610$ $\sigma^2 \approx 14960718$	$\bar{E} \approx 105004$ $\sigma^2 \approx 23900402$	$\bar{E} \approx 144590$ $\sigma^2 \approx 35861243$	$\bar{E} \approx 188330$ $\sigma^2 \approx 55823290$	$\bar{E} \approx 235006$ $\sigma^2 \approx 73515954$	$\bar{E} \approx 285208$ $\sigma^2 \approx 101617629$	$\bar{E} \approx 337942$ $\sigma^2 \approx 125037205$	$\bar{E} \approx 393809$ $\sigma^2 \approx 155950007$
Extrema Prop. $K = 200 \quad \tau = 2$	$\bar{E} \approx 11825$ $\sigma^2 \approx 2073688$	$\bar{E} \approx 32817$ $\sigma^2 \approx 7116738$	$\bar{E} \approx 59678$ $\sigma^2 \approx 13362979$	$\bar{E} \approx 91260$ $\sigma^2 \approx 21180118$	$\bar{E} \approx 127078$ $\sigma^2 \approx 32157704$	$\bar{E} \approx 166769$ $\sigma^2 \approx 49523465$	$\bar{E} \approx 209541$ $\sigma^2 \approx 63755586$	$\bar{E} \approx 255728$ $\sigma^2 \approx 88962196$	$\bar{E} \approx 304344$ $\sigma^2 \approx 107675419$	$\bar{E} \approx 355994$ $\sigma^2 \approx 134316847$
Extrema Prop. $K = 200 \quad \tau = 4$	$\bar{E} \approx 13240$ $\sigma^2 \approx 2196176$	$\bar{E} \approx 36039$ $\sigma^2 \approx 7464222$	$\bar{E} \approx 64827$ $\sigma^2 \approx 14341773$	$\bar{E} \approx 98460$ $\sigma^2 \approx 22601531$	$\bar{E} \approx 136390$ $\sigma^2 \approx 34451049$	$\bar{E} \approx 178294$ $\sigma^2 \approx 52489303$	$\bar{E} \approx 223233$ $\sigma^2 \approx 69390397$	$\bar{E} \approx 271641$ $\sigma^2 \approx 93797056$	$\bar{E} \approx 322888$ $\sigma^2 \approx 114343556$	$\bar{E} \approx 376802$ $\sigma^2 \approx 149649193$
Extrema Prop. $K = 200 \quad \tau = 6$	$\bar{E} \approx 14653$ $\sigma^2 \approx 2335601$	$\bar{E} \approx 39243$ $\sigma^2 \approx 7878743$	$\bar{E} \approx 69961$ $\sigma^2 \approx 15063485$	$\bar{E} \approx 105604$ $\sigma^2 \approx 23404144$	$\bar{E} \approx 145586$ $\sigma^2 \approx 35128663$	$\bar{E} \approx 189595$ $\sigma^2 \approx 53936956$	$\bar{E} \approx 236693$ $\sigma^2 \approx 70936862$	$\bar{E} \approx 287471$ $\sigma^2 \approx 96915288$	$\bar{E} \approx 340771$ $\sigma^2 \approx 119403799$	$\bar{E} \approx 396884$ $\sigma^2 \approx 149602314$
Extrema Prop. $K = 300 \quad \tau = 2$	$\bar{E} \approx 11827$ $\sigma^2 \approx 2075013$	$\bar{E} \approx 32854$ $\sigma^2 \approx 7151437$	$\bar{E} \approx 59801$ $\sigma^2 \approx 13666904$	$\bar{E} \approx 91508$ $\sigma^2 \approx 21687776$	$\bar{E} \approx 127455$ $\sigma^2 \approx 32246038$	$\bar{E} \approx 167443$ $\sigma^2 \approx 49034019$	$\bar{E} \approx 210429$ $\sigma^2 \approx 65527155$	$\bar{E} \approx 256872$ $\sigma^2 \approx 89701948$	$\bar{E} \approx 305880$ $\sigma^2 \approx 107745782$	$\bar{E} \approx 357697$ $\sigma^2 \approx 133866728$
Extrema Prop. $K = 300 \quad \tau = 4$	$\bar{E} \approx 13241$ $\sigma^2 \approx 2197900$	$\bar{E} \approx 36061$ $\sigma^2 \approx 7496545$	$\bar{E} \approx 64933$ $\sigma^2 \approx 14366570$	$\bar{E} \approx 98639$ $\sigma^2 \approx 22522137$	$\bar{E} \approx 136712$ $\sigma^2 \approx 34176917$	$\bar{E} \approx 178802$ $\sigma^2 \approx 52545663$	$\bar{E} \approx 223962$ $\sigma^2 \approx 69323741$	$\bar{E} \approx 272692$ $\sigma^2 \approx 94100717$	$\bar{E} \approx 323987$ $\sigma^2 \approx 113477049$	$\bar{E} \approx 378174$ $\sigma^2 \approx 142043376$
Extrema Prop. $K = 300 \quad \tau = 6$	$\bar{E} \approx 14656$ $\sigma^2 \approx 2334144$	$\bar{E} \approx 39270$ $\sigma^2 \approx 7856704$	$\bar{E} \approx 70070$ $\sigma^2 \approx 15101054$	$\bar{E} \approx 105801$ $\sigma^2 \approx 23443353$	$\bar{E} \approx 145933$ $\sigma^2 \approx 35798129$	$\bar{E} \approx 190100$ $\sigma^2 \approx 53633015$	$\bar{E} \approx 237420$ $\sigma^2 \approx 70411588$	$\bar{E} \approx 288330$ $\sigma^2 \approx 97474706$	$\bar{E} \approx 341839$ $\sigma^2 \approx 116997366$	$\bar{E} \approx 398323$ $\sigma^2 \approx 145710819$

6 Conclusions and Future Work

In this work two approaches of diameter estimation have been studied — various versions of *Breadth First Spanning Tree Building* and *Extrema Propagation*. Time and energy needed by each method have been asymptotically computed, and exact values for linear graph and two dimensional grid has been given for most of the cases. The most robust version of BFSTB and Extrema Propagation, with various parameters, have been implemented and tested using the computer simulations.

Both theoretical and simulation results confirm that BFSTB is more time consuming, but in total sends a small number of messages. Extrema Propagation, when good parameters are chosen, is a quick and robust method that can be used for very precise estimation of the diameter.

There is still much research that can be done in the field. Other network topologies, like Bluetooth, can be simulated. More complex model can be chosen to predict more precisely the behavior of the algorithms when multiple failures, quits and joins occur.

References

[1] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). SIAM J. Comput., 28(4):1167–1181, March 1999.

[2] Hagit Attiya and Jennifer Welch. Distributed Computing: Fundamentals, Simulations and Advanced Topics. John Wiley & Sons, 2004.

[3] Carlos Baquero, Paulo Sérgio Almeida, Raquel Menezes, and Paulo Jesus. Extrema propagation: Fast distributed estimation of sums and network sizes. IEEE Trans. Parallel Distrib. Syst., 23(4):668–675, 2012.

[4] Jorge C. S. Cardoso, Carlos Baquero, and Paulo Sérgio Almeida. Probabilistic estimation of network size and diameter. In LADC, pages 33–40, 2009.

[5] Jacek Cichoń, Rafał Kapelko, Jakub Lemiesz, and Marcin Zawada. On alarm protocol in wireless sensor networks.

[6] Martinus Dipobagio. An overview on ad hoc networks.

[7] Michal Parnas and Dana Ron. Testing the diameter of graphs. Random Struct. Algorithms, 20(2):165–183, 2002.

[8] M. Penrose. Random Geometric Graphs. Oxford Studies in Probability. Oxford University Press, 2003.

[9] Alberto Pettarin, Andrea Pietracaprina, and Geppino Pucci. On the expansion and diameter of bluetooth-like topologies. In ESA, pages 528–539, 2009.

[10] Wikipedia. Graph (mathematics) — Wikipedia, the free encyclopedia, 2012. [Online; accessed 13.05.2012].

[11] R.J. Wilson. Introduction to graph theory. Longman, 1996.