Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Diploma Thesis

Distributed routing in networks and its application

Bc. Jan Cicvárek

Supervisor: Kristian Hengster-Movric, Ph.D

Study Programme: Open Informatics

Field of Study: Cyber Security

May 23, 2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Cicvárek** Jméno: **Jan** Osobní číslo: **406368**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Kybernetická bezpečnost**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Distribuované směrování v sítích a jeho použití

Název diplomové práce anglicky:

Distributed routing in networks and its application

Pokyny pro vypracování:

1. Decentralized networking - State-of-the-art with introduction to Kademlia algorithm, [1]. Investigate probabilistic data structures with special focus on the Skip list, [2].
2. Study cooperative consensus protocols for application to fully distributed networks, [6].
3. Propose a fully distributed network with dynamic node addition and removal.
4. Develop numerical simulations of the proposed distributed network. Systematically test the network properties and performance.
5. Investigate a specific application of the proposed distributed network architecture to StarCraft II player simulation, [4].

Seznam doporučené literatury:

[1] Kademlia: A peer-to-peer information system based on the XOR metric, P. Maymounkov D. Mazieres, Proceedings of the International Workshop on Peer-to-peer systems, New York University, 2002.
[2] Skip lists: a probabilistic alternative to balanced trees, W. Pugh, Communications of the ACM, 33.6, 1990.
[3] An Improved Kademlia Routing Algorithm for P2P Network, L. Guangmin, Proceedings of the International Conference of the New Trends in Information and Service Science, 2009.
[4] StarCraft II: A New Challenge for Reinforcement Learning, O. Vinyals, K. Calderone et al. Preprint Arxive 2017
[5] On two types of deviation from the matching law: Bias and undermatching, W.M. Baum, Journal of Experimental analysis of Behaviour, 21(1), 1974
[6] Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches, F. L. Lewis, H. Zhang, K. Hengster-Movric, A. Das, Springer, 2013.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Kristian Hengster-Movric, Ph.D., katedra řídicí techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **25.02.2019** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **19.02.2021**

Kristian Hengster-Movric, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání	Podpis studenta

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Cicvárek** Jméno: **Jan** Osobní číslo: **406368**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Kybernetická bezpečnost**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Distribuované směrování v sítích a jeho použití

Název diplomové práce anglicky:

Distributed routing in networks and its application

Pokyny pro vypracování:

1. Decentralized networking - State-of-the-art with introduction to Kademlia algorithm, [1]. Investigate probabilistic data structures with special focus on the Skip list, [2].
2. Study cooperative consensus protocols for application to fully distributed networks, [6].
3. Propose a fully distributed network with dynamic node addition and removal.
4. Develop numerical simulations of the proposed distributed network. Systematically test the network properties and performance.
5. Investigate a specific application of the proposed distributed network architecture to StarCraft II player simulation, [4].

Seznam doporučené literatury:

[1] Kademlia: A peer-to-peer information system based on the XOR metric, P. Maymounkov D. Mazieres, Proceedings of the International Workshop on Peer-to-peer systems, New York University, 2002.
[2] Skip lists: a probabilistic alternative to balanced trees, W. Pugh, Communications of the ACM, 33.6, 1990.
[3] An Improved Kademlia Routing Algorithm for P2P Network, L. Guangmin, Proceedings of the International Conference of the New Trends in Information and Service Science, 2009.
[4] StarCraft II: A New Challenge for Reinforcement Learning, O. Vinyals, K. Calderone et al. Preprint Arxive 2017
[5] On two types of deviation from the matching law: Bias and undermatching, W.M. Baum, Journal of Experimental analysis of Behaviour, 21(1), 1974
[6] Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches, F. L. Lewis, H. Zhang, K. Hengster-Movric, A. Das, Springer, 2013.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Kristian Hengster-Movric, Ph.D., katedra řídicí techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **25.02.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **19.02.2021**

Kristian Hengster-Movric, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Abstrakt

Tato diplomová práce popisuje decentralizovaný systém jménem NodeSkipper určený pro kteroukoli spojitou neorientovanou síť. Uzly v této síti mohou posílat nebo vyhledávat jiné uzly nebo vyvolat proces "consensus", kdy se celá síť shodne na hodnotě zvolené veličiny tak, aby byl výsledek ovlivněn každým uzlem a byl pro všechny uzly stejný. NodeSkipper je inspirovaný datovou strukturou Skip List, která díky náhodnosti své struktury, která se přes postupné přidávání a ubírání uzlů přibližuje zvolenému pravděpodobnostnímu rozdělení, nabízí velmi všestranný výkon a vysokou robustnost.

Protokol NodeSkipper pracuje nejlépe pro sítě s efektem malého světa, který se vyskytuje ve skutečných sítích přirozeně. Díky tomuto efektu roste průměr sítě pouze logaritmicky vzhledem k množství uzlů. V takové síti je NodeSkipper schopný doručit zprávu nebo hledat uzel v logaritmickém čase. Díky své necentralizovanosti a absenci konkrétní struktury funguje velmi dobře s velkými sítěmi, kde jsou nové uzly nepředvídatelně přidávány a odebírány a přímá spojení navazována a ztrácena, jako například vozidla v silniční dopravě, doručovací roboti, stroje v továrně, bezpečnostní systémy pro velká území, počítače spolupracující na výpočetně náročné úloze nebo roboti účastnící se boje.

Protože tento systém nemá žádné uzly s vyšší důležitostí, je odolný vůči cíleným útokům a vzhledem k tomu, že funguje na kterémkoli spojitém grafu, je odolný vůči náhodným útokům a selháním. Díky schopnosti dojít ke shodě může dobře koordinovat své prostředky.

Abstract

This thesis describes a decentralized system that can work over any connected undirected network called NodeSkipper. Each node in this system can send a message to another node, look-up any node or request the system to reach consensus, which means that every node in the system will agree on a quantity of interest in a manner where each node influences the result. The system is designed after the Skip List data structure, which uses randomized structure that over successive entries and removals converges towards its probability distribution, while providing great all-rounded performance and robustness.

The NodeSkipper protocol works best over networks with small-world effect, which occurs naturally on real networks. This effect manifests itself by network diameter scales logarithmically with the number of nodes. On such network, NodeSkipper can deliver messages and look-up nodes in logarithmical time as well. Thanks to its decentralized nature and no rigid structure, it works well with large networks where new nodes are unpredictably added and removed and direct connections gained and lost, such as cars on the road, delivery robots, machines in a manufacturing plant, large scale security system, computers working together on computationally demanding task or battle units in armed conflict.

Because this system does not have any nodes of special importance, it is resistant to targeted attacks. Because it works as long as the graph is connected, it is resistant to random attacks and failures. Thanks to its ability to reach network wide consensus, it can coordinate its efforts.

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Author statement for undergraduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date…… ……

…….
signature

Acknowledgements

I would like to express my sincere gratitude to my advisor Kristian Hengster-Movric, Ph.D., for his enthusiasm, words of support, wealth of knowledge, willingness to share that knowledge with me, invaluable input and observations and help with the many problems and challenges, that I was mulling over on my own, but figured out after a short conversation with him and our study officer Renata Fialová for her support and care that she dedicates to all of her students.

Distributed routing in networks and its application

Bc. Jan Cicvarek
Czech Technical University
Department of Control Engineering
Hradec Kralove
Czech republic
Open Informatics - Cyber Security
Cicvarek.Jan@gmail.com

May 23, 2019

Contents

List of Figures

List of Tables

1 Introduction

Network is one of the ways in which a system can be represented, it can help to highlight the parts that the system is composed of, their interactions, relations or dependencies. A network is composed of nodes and edges. Edges represent a connection between nodes, they often have a label or a state description. Nodes are the subsystems that make a whole, much like edges, they can have a state or a label, but can also feature internal logic or even some level of autonomy, [2].
When we are studying a complex system, it might not be feasible to look at everything at once. Dividing a problem into sub-problems or just simple states can make a problem easier to understand and help us with finding the solution, [3]. This is not something that is done only in computers or only by humans. Many animals seem to have developed such a method of analysis long before humans even existed, [5].
Nevertheless, a problem that is described and understood still needs to be solved and networks can help with that as well. Once again, humans were not the first to discover that, as shown time and time again, *alone, we can do little, but together, we can do so much*, although we might have been the first to make a proverb out of it. Plants can communicate in a cascade to warn others of threats, [1], animals, such as fish and birds can pass information, distribute it through their group, reach a consensus and act on it in unison, [18], with some mammals even implementing advanced decision making, [21].
In computational theory, networking has been important from the very start. In fact the Analytical Engine, the first proposed general purpose computer designed by Charles Babbage, [22], could be represented as an automaton, since it can be emulated on a computer with random access memory (i.e. modern computer) and for each program P of n steps that would run on such computer, a Turing machine that produces the same result for each instance of program P can be constructed, [28]. Although a network can be used to represent a computer that does not mean that it is a good idea and it is indeed seldom used, with notable exception of small subsystems of computer logic, such as branch prediction, [30].

1.1 Early communication networks

Where network representation makes much more sense are network of individual computers, namely the Internet, the latest iteration in the effort to communicate over large distance, but not the first. The attempts to communicate in cases where voice and gestures do not reach are probably as old as the first civilizations, with the likely use of smoke signals. First well documented and far spanning com-

munication systems, apart from simply dispatching a messenger, can be found in 4th century BC in Roman Republic and China, during the Warring States period. These systems include signal fires, pot drainage systems or drum beats, [24]. Even though these systems might seem crude and impractical by today's standards, they must have seemed sufficient at the time, since little progress can be observed over the next 2000 years. The invention of a telescope in 1608 prompted this to finally change with optical semaphore signalization being proposed by Robert Hook in 1684, [24]. It prompted the creation of wide spanning communication networks all over Europe. Even though the transfer speed was as low as 2 words per minute and bad weather could disrupt communication, it was the first time a real time conversation could be carried out over vast distances. Even though it was mainly used for military purposes, it showed the potential of communication less constrained by distance.

It would not be until 1816, when the system that would give the public its chance to enjoy near real time communication over long distances took form. Many inventors worked on similar concepts, but it was Francis Ronalds who was first able to present a working prototype of electrical telegraph, [24]. The first glimpse of the things to come appeared with teleprinters. They were machines intended to transfer text to one another. One of the early examples is the Morse telegraph from 1837 by Samuel Morse and Alfred Vail, [24]. The devices in this circuit switched network could only connect to one other device for a defined period of time and transmit messages during this period. This means that the topology of the network would change to accommodate a new connection, [2]. This network became truly global in 1858, when the *Atlantic Telegraph Company* successfully laid a cable across Atlantic ocean, connecting Europe to the North America. While this connection failed in the same year, lasting success for *Atlantic Telegraph Company* came in 1866, when they have managed to lay a new cable as well as rescue and finish one of previous attempts. Another breakthrough came in 1886 by Heinrich Rudolf Hertz, when he demonstrated wireless transmitting of telegraph messages, [24]. This system was refined by Guglielmo Marconi in 1899 and adopted for naval use. It was one of the first wide broadcasts, with information for ships and their crews.

The next big step forward was the invention of a telephone in 1876 by Alexander Graham Bell and its implementation a year later. Apart from allowing its subscribers to communicate by spoken words, not just text, it provided the users with much greater bandwidth, something that Telex messaging network took advantage in 1933, [24]. Telex enabled anyone with access to a telephone line and a modi-

fied telephone switch to connect to anyone with the same set-up. This network has already supported multiplexing, allowing up to 25 parallel connections on a single line and automatic routing.
Next important development was the creation of modulation and demodulation device, *modem* for short, in 1949, [24]. This device allowed computers to communicate over the network directly, human operator was no longer required. This provided all the pieces needed for creation of Bell 101 modem, allowing computers to connect to each other over telephone lines.
Now that it was possible for average consumer to connect to another computer, it was important to also provide a reason to do so and a way to do it without expert knowledge. This is where commercial networks come in. The ones focusing on corporate clientele, such as CompuServe, focused mainly on services such as digital office, distributed computing, shared data access and advanced communication, which were big drives motivating the rapid adoption of computer networking. Other networks were targeting mainly the non-professional customers, for example Minitel, focused on providing access to news, games or connecting and finding other users, [12]. They offered their own software and often specialized hardware for their clients that can be used to interpret the data coming from the telephone line and provided an infrastructure that user can connect to.

1.2 Backbones and inter-networking

Although there was an abundance of networks for both businesses and individuals to connect to, there was no universal product. A Tymnet subscriber could not send an email to a Telenet subscriber, even thou both systems supported the service, [20], since the networks could not link to one another. This started to be a problem, since it was not feasible to switch computers from two networks to a new one every time an expansion was needed or to have a separate network for each subset of computers that needed to be networked together. This was especially infeasible for the networks that, unlike those using modems and telephone lines, used dedicated connections, like Cyclades and NPL in Europe, [13] or ARPAnet, Satnet or Prnet in the United states. As a result the OSI and TCP/IP, 1.2.1, layered protocols were proposed, [16], allowing for data to be transferred between computers of different networks.
While TCP/IP was a project inside the ARPA, OSI was heavily influenced by the existing standards, such as IBM's SNA and Digital Equipment Corporation's DECnet, who introduced the layering concept in answer to the still growing networks and the need to be inter-connected without the need to affect the functionality of the original local networks, [6].

Initially, the OSI model has seen some success and was initially considered to

be the future of inter-networking, in great part thanks to DEC and IBM, whose SNA was the most dominant architecture, being its big proponents. Nevertheless, the TCP/IP has also seen substantial adoption, mostly in the United States of America, since its intercontinental demonstration in 1977, thanks to the support from National Science Foundation (NSF), which provided significant investment to the infrastructure of many networks that were using it and the quickly growing vendor, CISCO.

Since the protocol battle lasted, many hardware and software vendors added support for both OSI and TCP/IP, [6]. While both systems aimed for wide compatibility, the DECnet and SNA were still rather restrictive, wanting its customers to remain fully within their ecosystems. When Data Link Switching allowed the much larger SNA install base to run over IP network and when the networks supported by NSF, which were originally strictly for non-commercial use, with most users being government employees and university students, lifted these restrictions in 1991, which allowed commercial networks to connect to the existing infrastructure using the TCP/IP model, turning them into *Internet service providers* (ISP), the TCP/IP protocol has gained a lot of momentum. This has eventually resulted in its dominance over the OSI model, with TCP and IP protocols still widely used to this day.

ARPAnet, mentioned above, was one of the oldest network systems, started in 1969, [24], it was strictly non-commercial network. It has undergone a critical transformation in 1985, when parts of it got re-purposed as NSFNET. This new network allowed much greater volume of data to be transferred compared to the ARPAnet connections, [9]. This has brought about a major change, one that is the defining feature of the Internet we are using today. For the last 2400 years, we have been building new connections in a network based on a need of connecting two points. If two places needed to communicate, we would build a connection between them. Some existing infrastructure might be used, but mostly if it was already between the two points. These connections were quite similar, with similar bandwidths, reliability or priority. But now, there were tiers of connections. The *backbone* of the network, the first tear, was now NSFNET. With this new structure, when a new connection was required, we would seek to connect it to the backbone, even if the goal is to connect it to just one or a few of specific remote nodes.

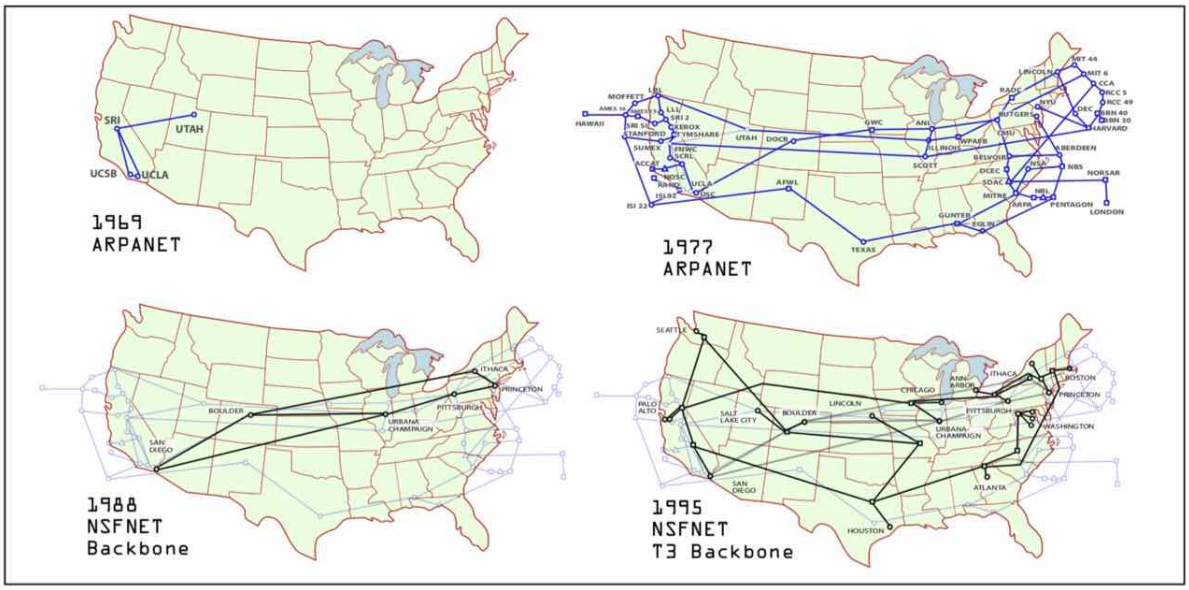

Figure 1: Networks with backbone structure, [36]

1.2.1 TCP/IP model

Both the TCP/IP and OSI models introduce layers that serve to specify the function of various protocols and how they should interact and pass data to one another, [19]. While these models are not strict, not adhering to them could negatively effect compatibility and should be only done for a good reason, such as a considerable simplification or an increase in efficiency.

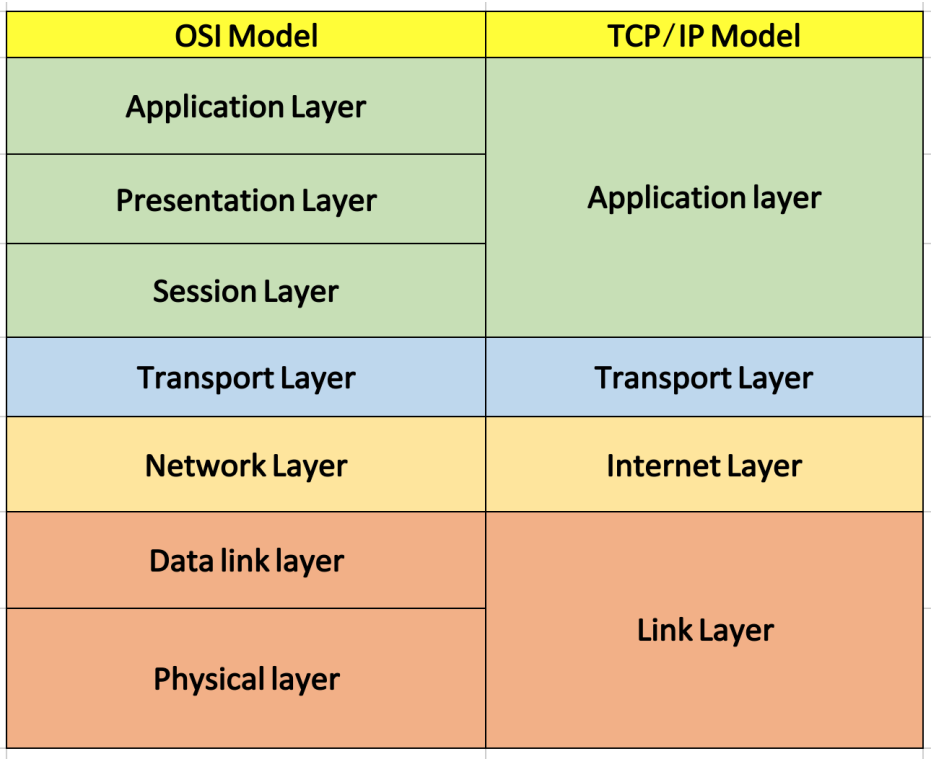

Figure 2: Main networking layer models, [38]

The TCP/IP model has 4 layers, they, along with they most agreed mapping to the ISO model can be seen on figure 2. The function of individual layers is, [17]:

- **Link layer**
  A protocol in a link layer provides connection between two hosts that allows them to communicate. It assumes the other host is directly reachable without the need for any interconnecting node in the middle. For this reason, a protocol in the link layer usually operates on local networks(LAN), although it can be used over remote hosts which lack direct connection. In this case, there is usually a tunnelling protocol in use which hides all the intermediate elements from the link layer protocol, so that the two host can be treated as if they had a direct connection.

- **Internet layer**
  The internet layer delivers a *packet*, which is formatted piece of data that is

carried over network, from its source to the destination node. It introduces an addressing that can be used to define the nodes and can carry the packet over the boundaries of local networks, connecting multiple networks through gateways. Internet layer protocol has to find a *route* between the endpoints (i.e. routing).

- **Transport layer**
  Transport layer protocol enables a communication between two nodes. It facilitates the sending of an entire messages that consist of one or more packets. It is responsible for segmenting data in the message at the source node and reassembling the message and the destination node. It enables a protocol in the application layer to transfer data to another node.

- **Application layer**
  The application layer ensures that a message can be sent to another node and that this message has a format that is understood by other nodes on the network, allowing it to be read. It provides and interface that allows the communication over the network. It specifies the availability of the communication, establishes agreement on strategy for achieving data integrity and error recovery. It is what an application, such as web browser of email client, uses to connect to other machines.

1.3 Centralized and distributed approach to networking

While the centralized approach to networking certainly managed to move the Internet into the mainstream and allow humans and businesses from all around the world, and even the Earth's orbit, to communicate in real time for relatively low costs in a very short period of time, it also brought many problems and drawbacks. While it is clear that enthusiast driven decentralized approach to networking would not be able to spread that fast and the inter-continental communication is still feasible mostly for large corporations, the economically accessible centralized solution has mostly removed the motivation to develop such a system. This left us with something of a monopoly, not just businesses, even though oligopoly just 11 major backbone providers for the whole word, is not as much as many would have hoped, but rather monopoly of just one approach with no other option anywhere in sight. The Internet is important part of our lives, our economies, our decision processes, our relationships, yet we have little control over it.
In the current and only system, a backbone provider can chose to *blackhole* a range of addresses, [7] and the average user is left little to know chance that such a site even existed. The routing of our requests is controlled by a few entities with

unknown or unclear intentions, which inevitably leads to exploitation, [8]. The solution? Just a patch and continue on, heading unchanged, even when whole nations lose the Internet connection[14], a different approach is hardly even discussed.
Now, this approach is seeping into the content of the Internet as well. Google dominates the search market in all but four countries of the world and owns two of the most visited websites, about half of the e-commerce done in the united states is done through Amazon, half of the domain names have been registered through Godaddy, 60% of content management services, used largely by news sites and opinion driven sites is provided by WordPress. The largest social media site, Facebook, has 2.3 billion monthly users, [39]. For many people, Internet has turned into a gateway for accessing a very limited selection of large portals, whose content is closely monitored and adjusted based on the profiles they have created by providing them with wealth of data. Sadly, if 90% of websites vanished overnight, many might not even notice.
Thanks to the Internet's centralized nature, governments have many options of blocking access to content they deem inappropriate, [42], [43], [44], [45], [46]. Recent developments in European union even present threat to possibility of legal existence for small internet platforms that allow for user input, [47], possibly forcing them to shut down or have a large company filter, view and control their traffic. While there is little chance that a decentralized network that could compete with the Internet would form in near future, there are still many options to stay less restricted. Virtual private networks are more affordable then ever, TOR provides great options for obfuscating traffic and avoiding DNS blocking, DHT and blockchain technologies provide distributed networks over the centralized one, that are difficult to disrupt. And even though they get less traffic, smaller platforms do exist, as well as decentralized variants to each of the mainstream website that do not need any server to let its users connect.

2 State of the art

This project is not made in vacuum, it is based on previous research and existing ideas. It is not the first distributed routing protocol, just a different take on it.

2.1 The Skip list data structure

The skip list is the basis for the proposed NodeSkipper protocol, whose name is intentionally similar. Skip list is a probabilistic data structure that arguably offers all the benefits and none of the drawbacks of all the other data structures. It is easy to implement, works for random or structured input, has insertion, deletion and search for an element in logarithmic time, can retrieve both the highest and lowest

element in constant time, so it can be used very much like a *heap*, can scale up and down easily, generates very little overhead for pointers, needs no pre-allocation for the data stored and can easily be implemented for multi-threaded workloads, [10]. The structure of skip list is very similar to linked list. Each element points to the both the element that precedes it and the one that follows it, these are called pointers on level 0. On top of this, some elements also have links of higher level, where the probability of having pointers up to a level L is $(p)^L$, with p being a parameter defined by the user. Both the remove and insert operations rely on the search operation and then apply changes to the pointers, meaning that as long as the search operation is fast, all the basic operations are. As the name suggests, the search operation relies on skipping some of the elements in the list, since it starts on the highest level available and only goes to a lover level once the value that it searches for is exceeded.

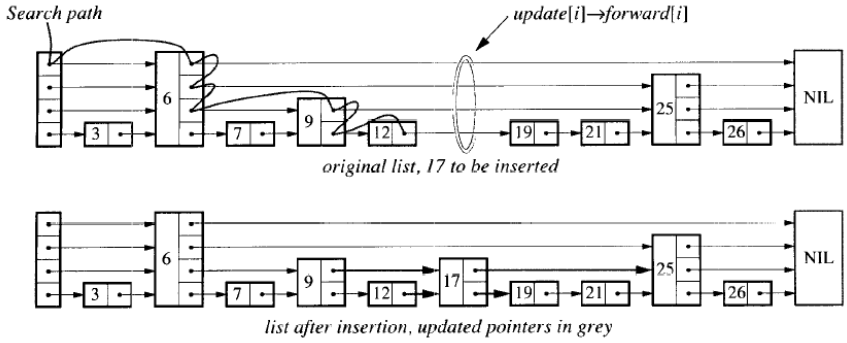

Figure 3: The structure and search pattern in skip list, [10]

As seen in figure, (3), skip list also has a header block at the beginning and an optional sentinel block the end as well. These blocks have level equal to the maximal level in the whole skip list and each level links to the first or last element

of that level, respectively.

```
initialization;
for each search level do
    while forward[level] ≤ search key do
        step to next element on level
        update forwards(element.forward)
        update backwards(element.backward)
    end
end
return element
```

Algorithm 1: Abstraction of the search operation

It can be seen in the pseudocode of the search algorithm, see (1), that it holds lists of all to closest nodes of each level in both the backwards and forward direction. It either returns the searched for element or the one that directly precedes it, if searched for key is not in skip list.

```
initialization;
search(key)

if value exists then
    change element
    return
end
while random <p do
    count increase
end
create new element
for count levels do
    element.forward[level] = forwards[level]
    forwards[level].backward = new element
    element.backward[level] = backwards[level]
    backwards[level].forward = new element
end
```

Algorithm 2: Abstraction of the insert operation

Since the probability of reaching each new level decreases geometrically, we will rarely see high level pointers and thus the creation of new element has near

constant speed once the position if found with search method.

```
initialization;
search(key)

if value does not exist then
    return False
end
for each element.level do
    if forwards[level] = header & backwards[level] = sentinel then
        forwards[level].backward = null
        backwards[level].forward = null
    end
    else
        forwards[level].backward = element.backward[level]
        backwards[level].forward = element.forward[level]
    end
end
delete element
```

Algorithm 3: Abstraction of the delete operation

2.2 Consensus protocols for fully distributed networks

Solving problems in a distributed way with autonomous agents, no matter if it is a data analysis simulation on single computer, physical network of network attached computers or robots interacting with the world around them, provides many advantages, but also some disadvantages. One of the greatest challenges is control of such a *multi-agent system*. Where for a single algorithm, we would simply code how it should react to specific inputs, what are its goal and how to reach it. In multi-agent system, such problems are categorized under cooperative control and are generally solved by consensus protocols, [31].

A consensus protocol implements a distributed control policy that enables the agents to reach agreement on quantities of interest represented by an internal state of each agent. This protocol is distributed in a sense that for each agent, it relies only on information held by that agent or its neighbours (i.e. local information), [23]. Based on the nature of the connection between agents, they adjust their internal state based on a differential equation, if the connection is continuous, with little interruption and sufficient bandwidth. If the connection does not meet these qualities and the communication relies on discrete packets delivered among agents,

difference equation is used, [33].
In general, a consensus algorithm ensures that each node's value in a network is driven towards the value of its neighbours, as can be seen in this common continuous consensus algorithm

$$\dot{x}_i(t) = \sum_{j \in N} (x_j(t) a_{i,j}) - x_i(t) d_i$$

Or in matrix form (see 3.3 for the full transition)

$$\dot{x}(t) = -Lx(t)$$

Where $x(0)$ is vector of internal states of the nodes, $a_{i,j}$ is an element of adjacency matrix A and d_i is an element of in-degree matrix D. This strategy is sufficient to ensure that if strong connectivity exists, a common value will be reached, [26]. The initial vector $x(0)$ has to be provided and will be used to determine the resulting common value, which will be between x_min and x_max, in this case, the common value will be a linear combination of the initial vector $x(0)$, [33]. Internal logic of each node might also adjust its $x_i(t)$, generally in case it needs to reflect on a change of external factors.
In case continuous communication cannot be guaranteed, but nodes are still able to communicate at a discrete instants of time, a difference equation is used, [33]. One simple algorithm, similar to the continuous algorithm presented above, is

$$x_i[k+1] = \frac{x_i[k] + \sum_{j=1}^{n} a_{i,j}[k] x_j[k]}{1 + d_i}, \qquad i = 1, ..., n \tag{1}$$

Or in matrix form

$$x[k+1] = (I - D)^{-1}(I + A)x[k]$$

Where $x[0]$ is vector of internal states of the nodes, $a_{i,j}$ is an element of adjacency matrix A, D is an in-degree matrix and d_i an element of in-degree matrix used to prevent the equation to diverge from weighted average when the adjacency and in-degree matrices are not normalized. This algorithm drives the shared value towards the weighted average of the initial vector and it can be shown, using Gershgorin's disc theorem that if the directed graph has a spanning tree or is strongly connected, or if the undirected graph is connected, the $\mid x_i[k] - x_j[k] \mid \to 0$ as $k \to \infty$ for all i and j, or in words, the the difference between the values of each pair of node converges to zero, generating consensus across the connected nodes. If the adjacency and in-degree matrices are not row stochastic, the resulting consensus might be higher than the highest initial value, if the row sums exceed 1 or below the initial value, if the opposite is true [4]. If both matrices are stochastic, global average will be reached.

2.3 Kademlia, a popular decentralized routing and lookup system

Kademlia is a decentralized protocol for routing queries and node lookup. The two things needed when two nodes in a graph want to communicate. It can achieve this without any global authority or enforced network structure and the complexity of a query or a lookup completion is $O(\log n)$, where n is the number of nodes in a graph, [11].

While Kademlia is not the first communication protocol for decentralized systems, it is one of the most popular. Thanks to its many advantages, few drawbacks and well rounded performance, it was Kademlia which moved algorithms based on distributed hash tables (DHT) into the mainstream, being utilized by individual users in torrent clients, chatting and VoiP programs or multi-player games, same as big companies such as Facebook and Twitter, [29]. Instead of a rigid structure dependant on a central server that needs to handle the clients, which can be easily overloaded or targeted in an attack, peer to peer systems scale with the amount of clients, do not have a single point of failure, are able to spread the load among individual users and the more advanced ones, such as Kademlia, dynamically adjust the route based on network utilization. The Kademlia algorithm assigns a 160bit key to each client, each of these hashes has a corresponding value that can be looked up on the network. The position of the nodes within Kademlia networks corresponds to nodes in binary search tree, where each bit of the address determines whether it is necessary to go to the left or right node on that level.

The binary tree in this form is a representation of set of consecutive one dimensional values. When the tree is iterated through with a depth-first algorithm, the original structure is recovered in ascending order. Each node has links to several other nodes. Apart from its parent, it also has the link to every sub-tree it encounter along the way to the root. Therefore, it has the amount of links that is equal to its level. While it is true that each node's address is the same length, 160bits, not every node holds links to 160 other nodes. If there is any sub tree with only one node, the node is placed at the root of this sub-tree, since there is no reason to continue with the searching, once this sub-tree is located, since it is already clear that the search is for that node or a non-existent node.

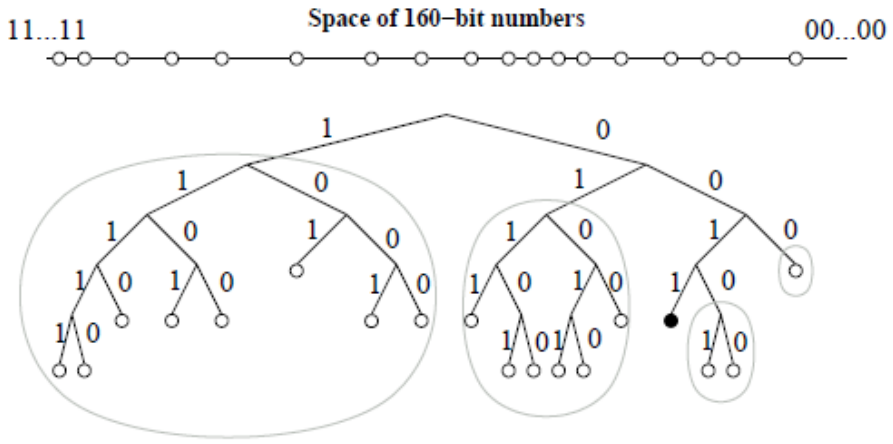

Figure 4: Kademlia tree, the circled sub-trees are the ones black node has connection to, [11]

The Kademlia system also defines a closeness metric. It is defined as XOR of the two addresses in their binary representation, the result is a 160bit number that has one for each digit where the addresses were different and 0 for each place where they were the same. The higher the leftmost *one* in the result is, the more are the two addresses different, more importantly, this number also tells each node, in which sub-tree to look for the other node. If the first one in in the highest digit, the other node is in the other half of the tree, if it is the second digit, it is the tree that splits one level below root, and so on. Once the correct sub-tree is located, the node sends a message to the node it is connected to in that tree. Since that node must have the digit mentioned above, and all the preceding once same as the target node, it is closer, based on the XOR metric. From here, the process repeats, with each step getting closer to the target, finally, when the result of XOR metric is 0, the node has been found. Because the amount of levels in a tree grows logarithmically with respect to the number of nodes, the average number of look-up steps needed to reach a node is grows at the same rate as the amount of levels.

3 The NodeSkipper protocol

NodeSkipper is a decentralized routing protocol that allows nodes that form a connected undirected graph to exchange messages with any other node, look-up any other node, look up the first node with higher and lower address as an unoccupied

address, join the network if it is not part of it and has direct connection to at least one node that is, form new direct connections, release direct connections and leave the network. It is resistant to directed attacks on its structure, thanks to having no structure and is resistant to attack on any random node, save for bridges, as all graphs become disconnected when losing a bridge node, since it only requires the graph to be connected, other properties only serve to improve its performance.
When a node is joining the network, a process described here, 3.6, in more detail, it will make a request towards nodes that it directly connects to and that are already part of the network. These nodes will request the system to reach consensus on the address of the new node. This value will be so that it is similar to the nodes close to the new one, as is described and tested here, 3.3. Once the address is determined, the direct neighbours of the new node will look-up this address, provided it does not exist, the call will return the closest preceding and succeeding node to that address, since when the process reaches one of these nodes, it would return its address and address of its closest neighbour in the direction of the look-up. In the extreme case where the found node is the largest or smallest, only its value will be returned. These addresses will be passed to the new node, which will store them as its first remote connections, with the direct neighbours being the direct connections, these two sets can overlap. At this point, the node is added to the system and it will start building its remote connections. By default, it will look up addresses based on equation:

$$diff = \frac{abs(adr1 - adr2)}{2} * 4^n \quad n = 1, 2, 3, ... \tag{2}$$

where $adr1$ and $adr2$ are the remote connections of the new node and $diff$ is difference of address from own address in each direction. Value must not exceed the lower extreme 0 and upper extreme $2^{128} - 1$, apart from that, both the multiplication factor and divisor in equation, 2, can be adjusted by each node, based on computational capability and memory size, or just preference. Look-ups used for building connections are marked and nodes they pass through build a dictionary that allows them to pass the subsequent request to that same destination quickly, without having to search the closest match. This approach is inspired by the *Multiprotocol Label Switching* (MPLS), one of the most successful networking protocols of this millennium, as it was implemented into most Internet's major autonomous systems, something most innovative and advantageous novel approaches fail to do, as the industry is often resisting change.
It is not only the new nodes that are building list of remote connections. Any existing nodes needs to maintain its list, if a new closest node appear, it needs to add it to the list, in favour of the previous closest node in that direction. If the node experiences disproportionate amount of lookups into some range of addresses, it

could also choose to add a remote connection there.
Apart from remote connection building, there is one other special marked look-up, path optimization. When a node has spare computational cycles, it can request the next steps in its MPLS inspired directory to tell it its next step. It can then try to route to that node through another node and saving it as the next step should the process be faster. This will also serve as a check that the next node is alive. Another check into node availability needs to be done towards the remote connections. This approach is inspired by the great shortest paths algorithm by Robert W. Floyd, [27].

```
initialization;
l(i,j) = a(i,j) p(i,j) = i for k = 1:n do
    for i = 1:n do
        for j = 1:n do
            if l(i,j)¿l(i,k) + l(k,j) then
                l(i,j)=l(i,k) + l(k,j) p(i,j) = p(k,j)
            end
        end
    end
end
```

Algorithm 4: Floyd's shortest paths algorithm

While this algorithm's complexity is $O(n^3$, it will tell, just based on the adjacency matrix A, represented by its elements a(i,j) the length of every shortest path in the graph as l(i,j), all the shortest loops as l(i,i), but most importantly, the p(i,j) tells us which node to go to next, if we are in node i and want to travel along the shortest path to the node j. Since the decentralized system does have n computational units at its disposal and could complete the task in roughly $O(n^2$, which is feasible, it would have to be changed each time something in the system changes. Instead, applying these rules locally and randomly, it will not guarantee a shortest path, but will give similar result on average.
With the application of the above, we can guarantee that a look-up will find the requested node or closest nodes on both sides and that a message intended for existing node will be delivered. This is acutaly guaranteed by the fact that node can only be added when it knows its nearest neighbours on either side means, that each valid request can be passed to a node that is closer to the target than the preceding node. In this sense, NodeSkipper uses the XOR metric, just as well as Kademlia, if only not as directly. Going by one node is of course the worse scenario and would form quite an oxymoron with a name like NodeSkipper. The experience from Skip List shows that with the list of remote connections, the number of nodes to query is likely to be of a logarithmic relation to the number of nodes, not linear, again much like Kademila in this regard. But while Kademila is primarily aimet at *power-law*

networks and indeed, NodeSkipper can work on those networks as well, it is designed mainly for the naturally occurring *small world effect* networks, where it can form paths that are approaching optimum thanks to its path adjustment and consensus driven address assignment.
The ability to reach a system-wide consensus, described in, 3.3, is not only important for the assignment of node addresses, it provides the whole system with a mechanism that allows it to cooperate more efficiently. Along with the ability to dynamically add and remove both nodes and edges, it makes NodeSkipper protocol a great fit for real world systems, such as cars on a road, delivery robots, machines in a manufacturing plant, large scale security system, computers working together on computationally demanding task or battle units in armed conflict.
Additionally, this section also discusses the node manager, that takes care of single computer multithreaded uses of the protocol, 3.5, the multithreading used in this project is also described, 3.2, along with the utility module, 3.1, used to work with the NodeSkipper packets, 3.4, also described in this section.

3.1 Utility module

The *util.py* module provides agents with base conversion and hashing functionality needed for building NodeSkipper packages.
Its most important components are:

- **str2byteArr**
  Creates UTF-8 encoding from given string.
- **byteArr2str**
  Uses bytearray with UTF-8 encoding to generate a string.
- **hex2byteArr**
  Accepts a string with hexadecimal number and converts it into *bytearray*. If the number of digits is odd, it prepends a 0. Throws ValueError for invalid hexadecimal number.
  It encodes each two digits as one Byte, to reduce space used.
- **byteArr2hex**
  Converts each Byte into two digits of a hexadecimal number. These pairs are ordered in the same way they were in the *bytearray* and returned as a string.
- **int2byteArr**
  Converts an integer into a *bytearray*, by expressing it as a base 256 number and storing each digit as an *item* in the bytearray.

- **byteArr2int**
  Treats *bytearray* as a base 256 number and converts it into a base 10 integer.

3.2 Multithreading

The language used for the practical part of this thesis, Python, is based on the von Neumann model, where processor executes a finite list of commands that act on data stored in *random access memory*, [41]. Such a list of commands is often called a *thread*. Most modern computers have more than one processing units, or *cores*, which allow for multiple instruction to be computed at once, but this is not possible with just one thread, [40]. While this inefficiency could be ignored in the testing phase of the *NodeSkipper* protocol, this model is intended to be used as a groundwork for multi-agent system playing the game *Starcraft2*, a real time strategy and it is necessary to make sure that all the computational resources can be utilized and that all the threads get sufficient computational time, since each thread will represent one agent and each agent one unit or structure, where inactive thread will result in effective loss of that building or structure.
The Python language supports multithreading, the mode used in this project is shared-memory multithreading, [41], where all the threads share the same sector of computer memory and use python synchronization methods in order to access these resources without conflict.
Another instance of multithreading occurs when the NodeSkipper protocol is implemented, even in the testing model. The NodeSkipper protocol can only work with nodes that are able to reach *consensus*, which means that multiple agents with their own internal logic need to execute computations based on input from other agents that result in all of the agents arriving at the same value that reflects the initial state of participating agents, [23]. This mode of multithreading operation is often called the message-passing model, [41].

3.2.1 Ensuring that each thread gets sufficient computation time

In multithreaded workloads, it is possible to significantly decrease the computation efficiency if incorrect amount of computation time is allocated to a thread, [32]. In worst case scenario, this could result in a *deadlock*, [34], halting the computation and breaking the whole program. Even in the less extreme scenarios, the effects of inefficient load balancing can lead to unpredictable behaviour and unresponsiveness of the program.
To test the that all the threads are getting similar amount of compute time, a test scenario, where multiple threads were given similar workload, has been set up, taking data from shared synchronized data structure. They are all started in a span

of several milliseconds and would compare the deviation in run time to the average run time. The resolution is in seconds in hopes of making the averaging trivial, since It is only necessary to confirm that they all get roughly the same computation time. This testing was done on CPU: Intel Core2 Quad Q9550 2.8GHz, GPU: ATI HD 3650, RAM: Corsair CM3X1024-1066C7 2GB (4 sticks), Python v3.7.0:1bf9cc5093.

```
initialization;
while queue not empty & not enough work done do
    for number of tasks * number of threads do
        create random workload
    end
    create lock for number of threads do
        create new thread
    end
    for each thread do
        start thread
    end
    for each thread do
        join thread
    end
end
```

Algorithm 5: Abstraction of the thread manager

```
initialization;
while queue not empty & not enough work done do
    initialize lock
     pop workload from queue
     release lock
     if queue empty then
        Terminate unsuccessfully
    end
    count += 1
     calculate library hashing function
     sleep(0.01s)

end
```

Algorithm 6: Abstraction of the testing thread

Table 1: Testing, tasks = 5000, threads = 100, tries = 100

Start (last - first) [s]	Fastest runtime [s]	Slowest runtime[s]	Finish (last - first) [s]
0	50	50	0

3.3 Using network consensus to assign an address to a node

The NodeSkipper protocol assumes that each node is connected to a both the node whose address directly precedes its address and the node whose address directly follows it. Random disruptions of this principle should not cause significant damage to connectivity, but if the nodes were able to chose their address, targeted attack would be possible, with chance of temporally impacting reachability of the target node. Furthermore, it is desirable to assign a node with IP address that respects its neighbourhood, for improved performance.
To test the IP address assignment a graph with structure similar to well initialized NodeSkipper graph has been created. Its nodes are connected to nodes with closest IP address and the probability that a node is connected to another node decreases logarithmically with delta of their addresses. The edge weights simulate ping and are related to square root of delta of their addresses.

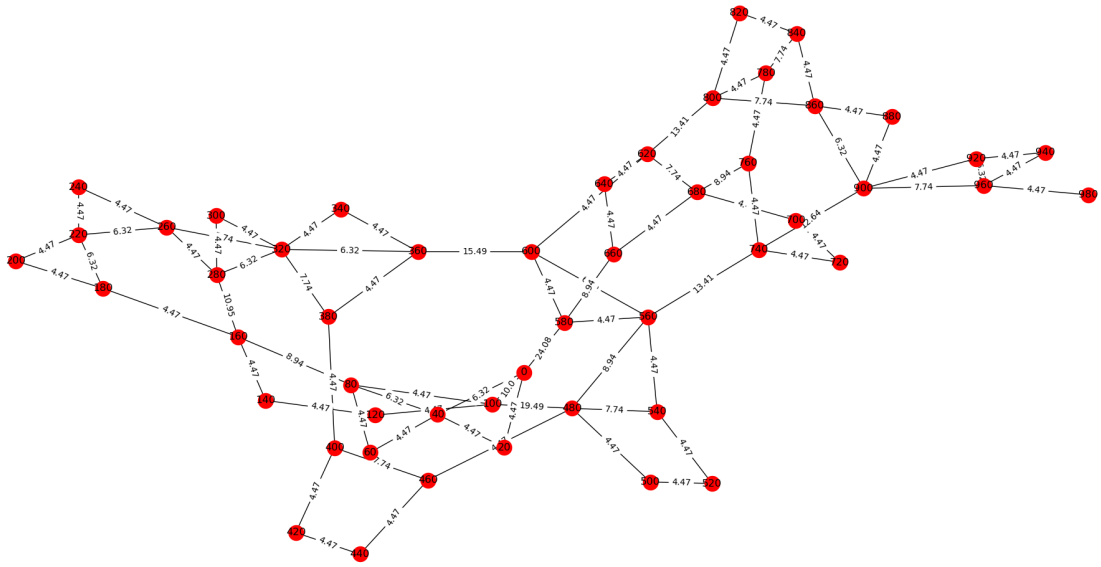

Figure 5: Example of test network for IP address assignment, seed=45

To reach consensus on this simulated network, see 2.2, the global dynamics given by $\dot{x} = -Lx$ were chosen for this task. The motivation for using this model being its simplicity, an ease of implementation and good option for analysing the convergence to the final value, since if the IP address assigned is not desirable, it is beneficial to have the opportunity to track down where the result started to go wrong.

For this reason, using matrix operations was not chosen and the dynamics were implemented by an algorithm that iterates over all the nodes and applies given formula, which means it is necessary to understand the individual steps behind the matrix form formula. This is also important, because in the live version, Laplacian matrix of the whole graph will not be available and this computation will be carried out by individual nodes that will only see their immediate neighbourhood.

$$\dot{x}(t) = -Lx(t)$$

$$\dot{x}(t) = (A - D)x(t)$$

$$\dot{x}(t) = Ax(t) - Dx(t)$$

$$\dot{x}_i(t) = \sum_{j \in N} (x_j(t)a_{i,j}) - x_i(t)d_i \tag{3}$$

Where each node has an integer as its state description, once consensus is reached, this integer will be used as the IP address for the new node, x represents the vector of state descriptions of all the nodes in the network in current iteration, $\dot{x}$ represents the difference in state of each node in the next iteration, N is set of all the nodes, A represents the adjacency matrix, $a_{i,j}$ represents one element of the adjacency matrix, D represents the degree matrix, d_i represents one diagonal element of the degree matrix and L represents the Laplacian matrix.
Because the graph has weighted edges, the computation is more difficult than subtracting the cardinality times the value of each x_i from sum of x_j belonging to its neighbours. Also the weights shown in figure, 5, will not be used, since they do not reflect where the new node is connected and will thus not return a value that helps to keep the structure of the graph. Instead, a simulated value that attempts to replicate the amount of ping between the nodes that connect to the new node and each x_j will be used.
This will be done by depth-first search algorithm, starting at the node or nodes connected to the new node that will assign each search node the value of simulated ping as a sum of ping of the node that added it to the search queue and the weight of the edge between them. To make nodes with higher ping less influential, the multiplicative inverse of their simulated ping will be used as their weight. The nodes originally connected to the new node will have edge weight of 4, those connected probabilistically based on distance, will have the weight based on square root of their IP difference.

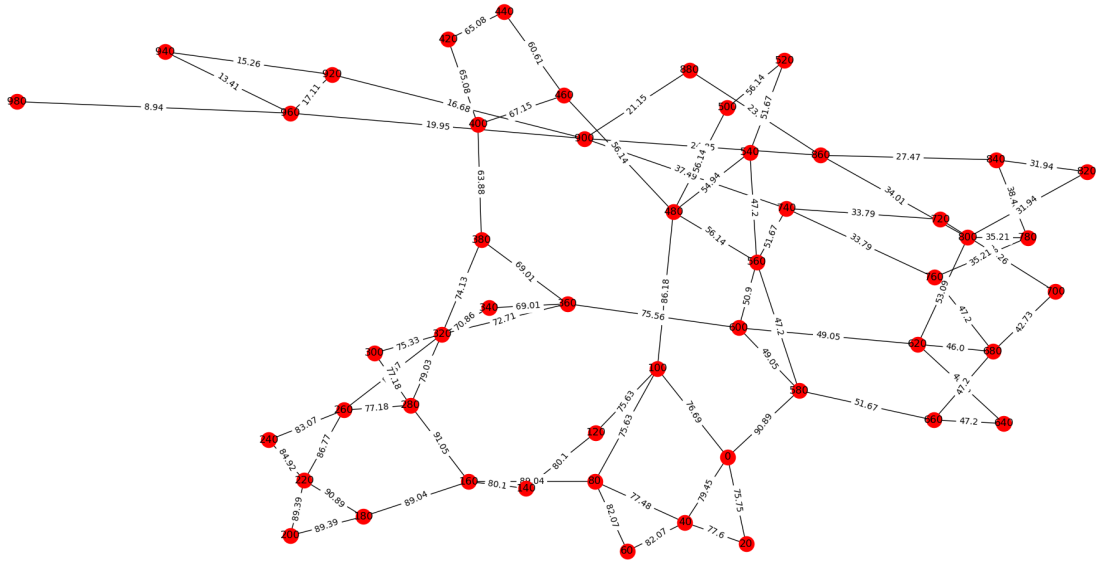

Figure 6: Example of test network for IP address assignment with simulated ping, seed=45

Since the nature of the communication is discrete and could experience time-outs and de-synchronization caused by outside factors during deployment, a difference equation would be better suited for this task, see 2.2. Since this is a weighted graph, the equation needs be in a form that would not grow beyond largest initial value.

$$x_i[k+1] = x_i[k] + \sum_{j \in N, a_{i,j} > 0} ((x_j[k] - x_i[k])(1/a_{i,j})) \tag{4}$$

Where $x[k]$ is vector of states of all nodes in the iteration k and rest of the variables have the same meaning as the ones in 3. The equation 4, is applied to each node for as long as the value is changing.

```
initialization;
for 0:1000:20 do
  | add node to a graph G
end
for each node in G do
  | add edge to node with first smaller and larger address
end
select all nodes in G
 for n times do
  | remove half the nodes in selection
  |  add edge to node with first smaller and larger address within selection
end
```

Algorithm 7: Graph construction and consensus testing

To evaluate the result of applying the equation 4, 100 random nodes were added. The result of a consensus algorithm should be value that is close to the value of the nodes that are directly connected to the new node. In some test cases, the new node is connected to only one other node, in others, it is connected to several direct neighbours. The *expected result* is the simple average of the addresses of the nodes the new node is directly connected to. While it is a simple method of obtaining an estimate of the result, it should be sufficient, since NodeSkipper works with any address distribution and if the addresses correspond to their location within the graph, it only speeds up the query and lookup operations, although not asymptotically. It is more important of having a method that would have the nodes agree on a some common quantity, having an idea what the result could be and have a way of influencing this result. Since NodeSkipper is intended for many diverse uses it might need facilitate decisions such as, whether to manufacture item A or B, in what ratio to purchase commodities C and D, is node E malicious, should a group of units attack enemy worker line or go defend from an army with a trajectory towards their base, and so on.

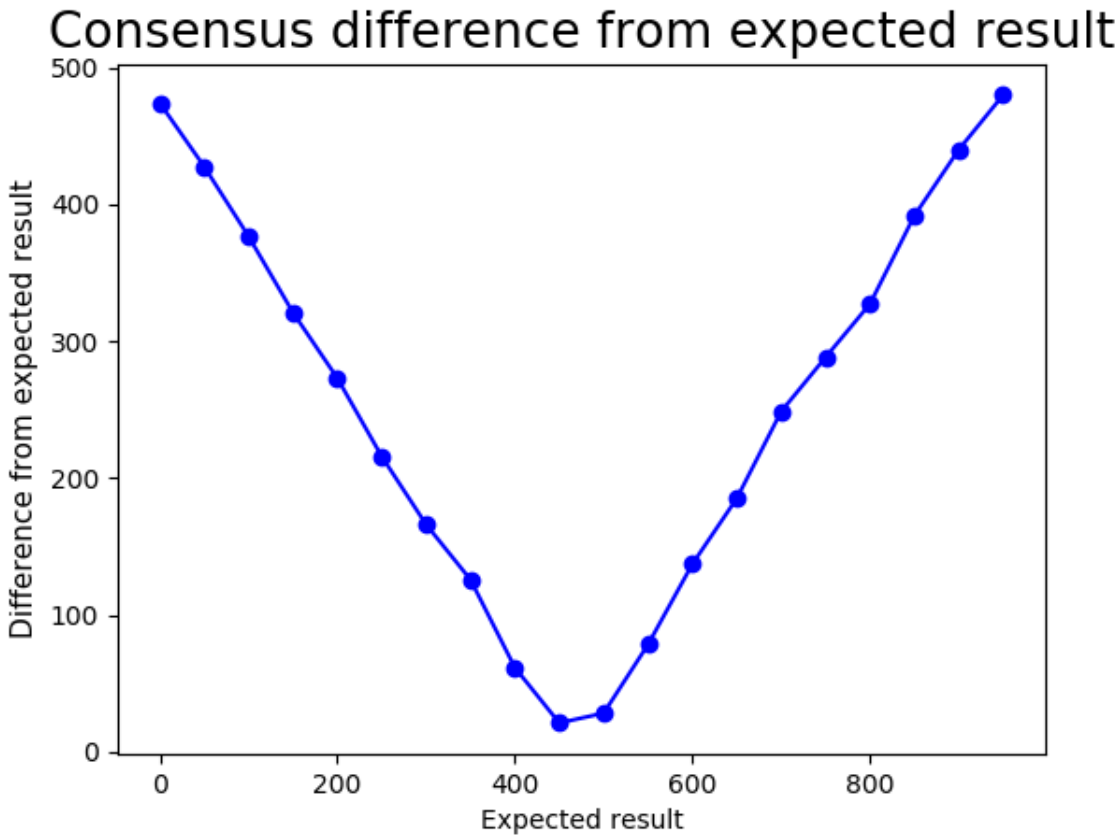

Figure 7: Reaching consensus with simple difference equation described in 4

The result showed in 7 shows little to no reaction of the strategy to the place where the new node was added. It returns results around the average value and the difference increases as the distance from graph average increases. While this result of the consensus algorithm could be used in *NodeSkipper* algorithm, as it does provide agreement on the new IP address and does not allow only small subset of nodes to pick a desired address, but the structure of the network would not reflect the position of the nodes and distribution of the nodes across the address space would not be uniform.
To mitigate this, a vector representing ping to each node from the new node is

introduced into the equation.

$$x_i[k+1] = x_i[k] + (\sum_{j \in N, a_{i,j} > 0} ((x_j[k] - x_i[k])(1/a_{i,j}))) * ping_i/k, \quad k > 0 \quad (5)$$

The higher the ping, the quicker is the node to change its value. The result is that the nodes farther from the new node will have less influence on the resulting number. The constant k is used to adjust the size of step in each iteration.

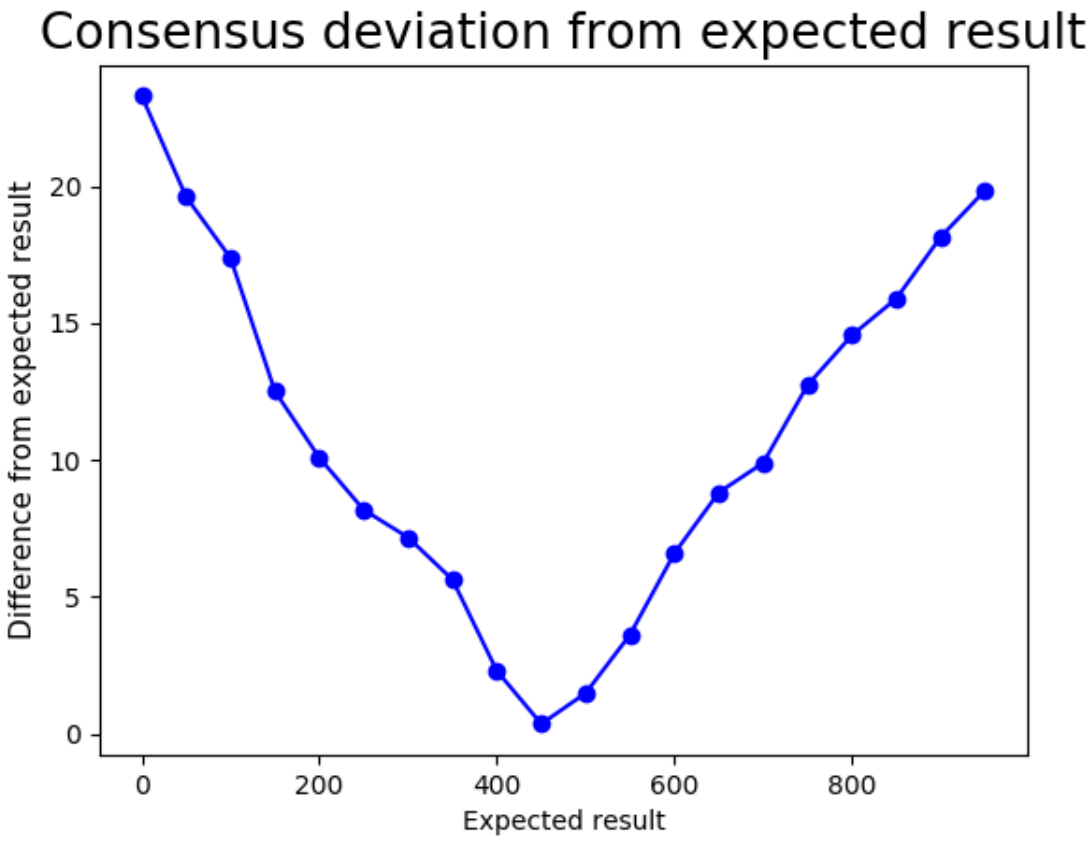

Figure 8: Reaching consensus with difference equation and added vector of ping described in 5

Since the values in matrix A reflect time needed to pass information between two nodes, it seems to be providing the same information as the ping vector, yet did

not have the desired effect. The result is still closest to the expected value around the global average, but the effect is much diminished. To simplify the equation and decrease computational complexity, we can reduce it with a signum function.

$$x_i[k+1] = x_i[k] + (\sum_{j \in N, a_{i,j} > 0} ((x_j[k] - x_i[k])(1/sgn(a_{i,j})))) * ping_i/k, \quad k > 0$$

$$x_i[k+1] = x_i[k] + (\sum_{j \in N, a_{i,j} > 0} ((x_j[k] - x_i[k])(1/1))) * ping_i/k, \quad k > 0 \quad (6)$$

$$x_i[k+1] = x_i[k] + (\sum_{j \in N, a_{i,j} > 0} (x_j[k] - x_i[k])) * ping_i/k, \quad k > 0 \quad (7)$$

Since the result of the $sgn(a_{i,j})$ would always be 1 because of the sum conditions, it was replaced by that number in equation 6 and subsequently removed completely in equation 7, the whole equation was simplified and the weights in adjacency matrix disregarded.

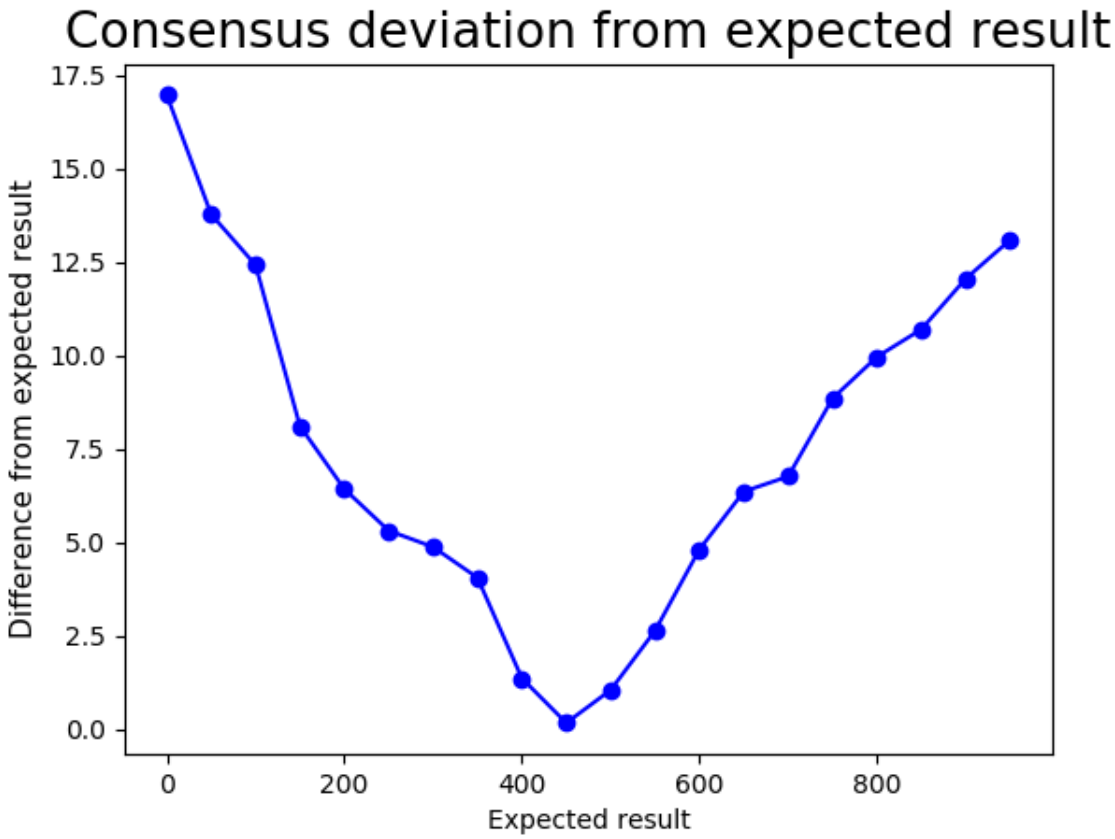

Figure 9: Simplified algorithm without weighted adjacency matrix described in 7

The result looks much the same as the one reached with weighted adjacency matrix, 5, although there was no need to reconstruct the matrix or use it to divide the sum.
To further improve the result, we could use quadratic or even cubic form of the ping to make the effect stronger.

$$x_i[k+1] = x_i[k] + (\sum_{j \in N, a_{i,j} > 0} (x_j[k] - x_i[k])) * ping_i^2/k, \quad k > 0 \qquad (8)$$

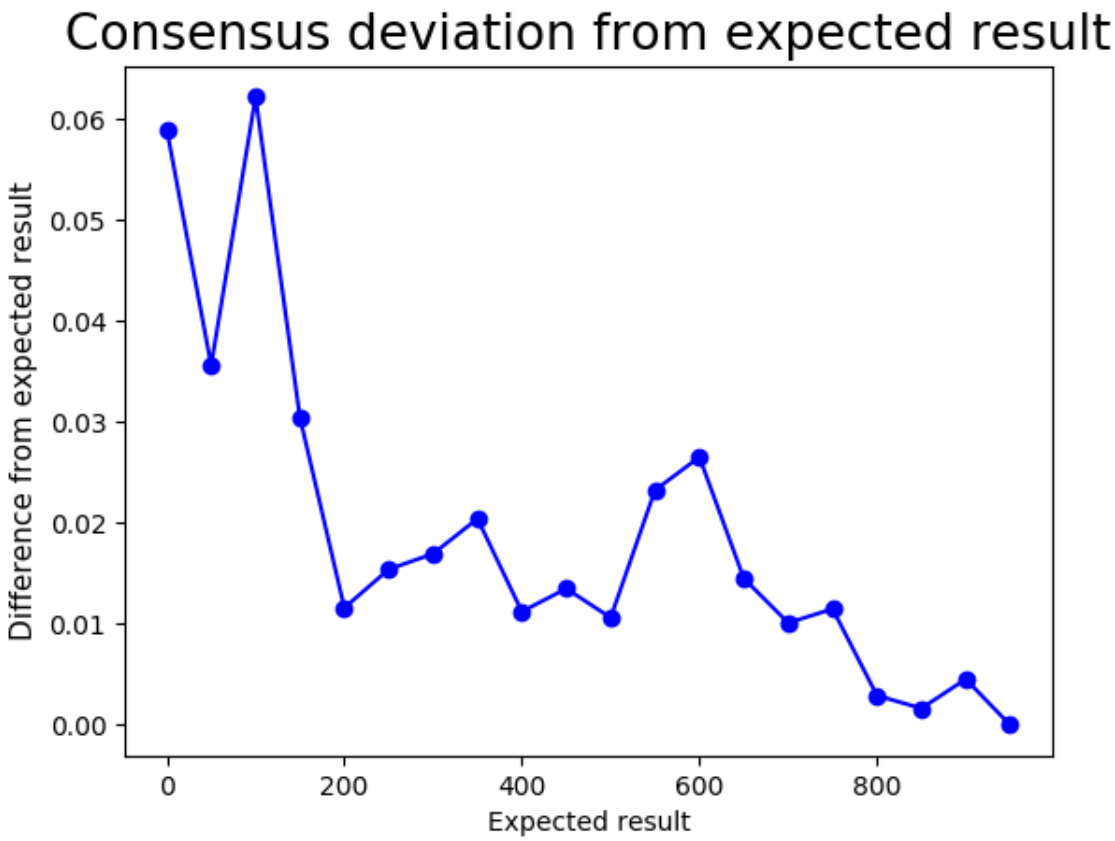

Figure 10: Algorithm without weighted adjacency matrix and cubic values of ping described in 8

This result copies the expected value almost perfectly and shows no apparent correlation between address average and consensus result, although there seems to be higher deviation from expected result when lower number are expected, an effect which reoccurs when using this algorithm when connecting a new node to several different nodes at once, as seen in, 13. The result could possibly be too close to the expected value, since it could possibly enable an attack that would leverage the foreknowledge of consensus result.

It is possible that new node is physically connected to more than one node when joining the network. For example when a wireless device in range of several other

devices is added. In such a case, it is desirable to have the address of the new device close be roughly average of the addresses of its direct neighbours. The equation remains unchanged, the only difference is that the ping vector holds distance to closest direct neighbour, not the one specific direct neighbour.

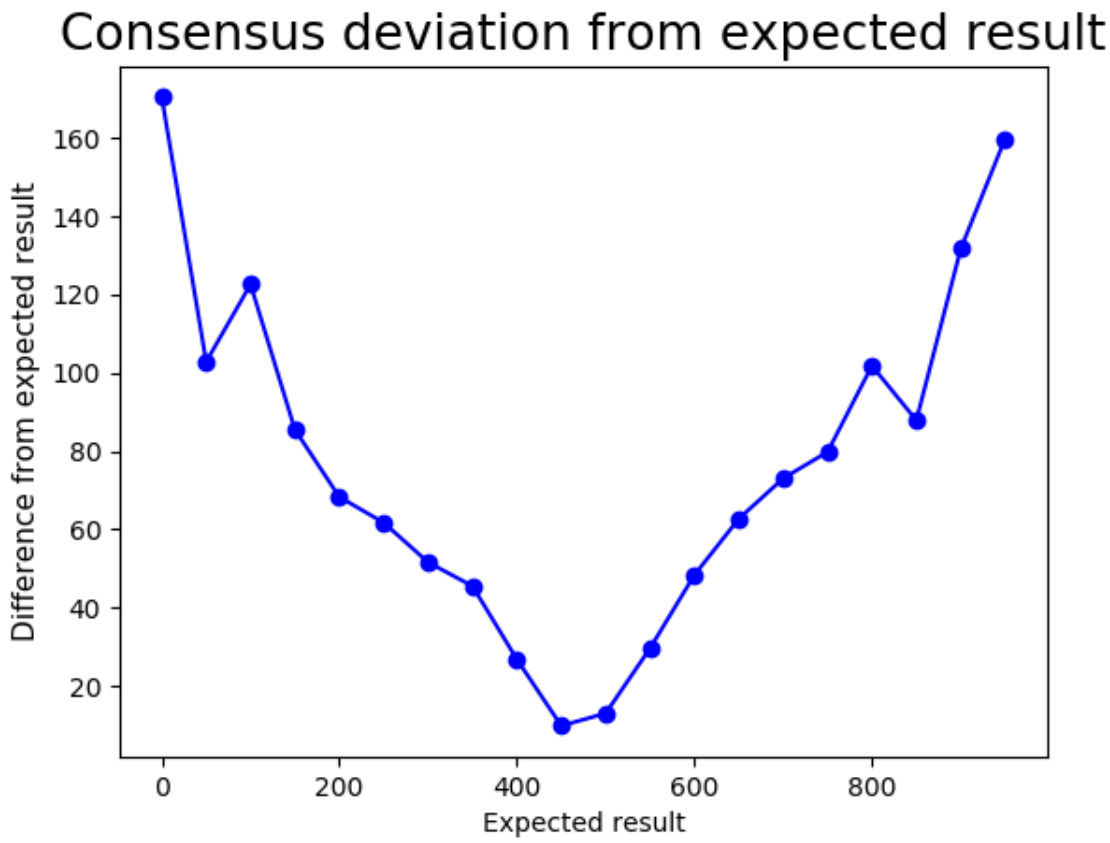

Figure 11: Algorithm without weighted adjacency matrix, and multiple direct neighbours for added node, described in 7

In figure 11, an increase in accuracy as expected value drops from 100 to 40 and rises from 800 to 860 can be observed. Random dip is not impossible, considering the low sample size of 100 nodes, but the symmetry is very strange and unexpected.

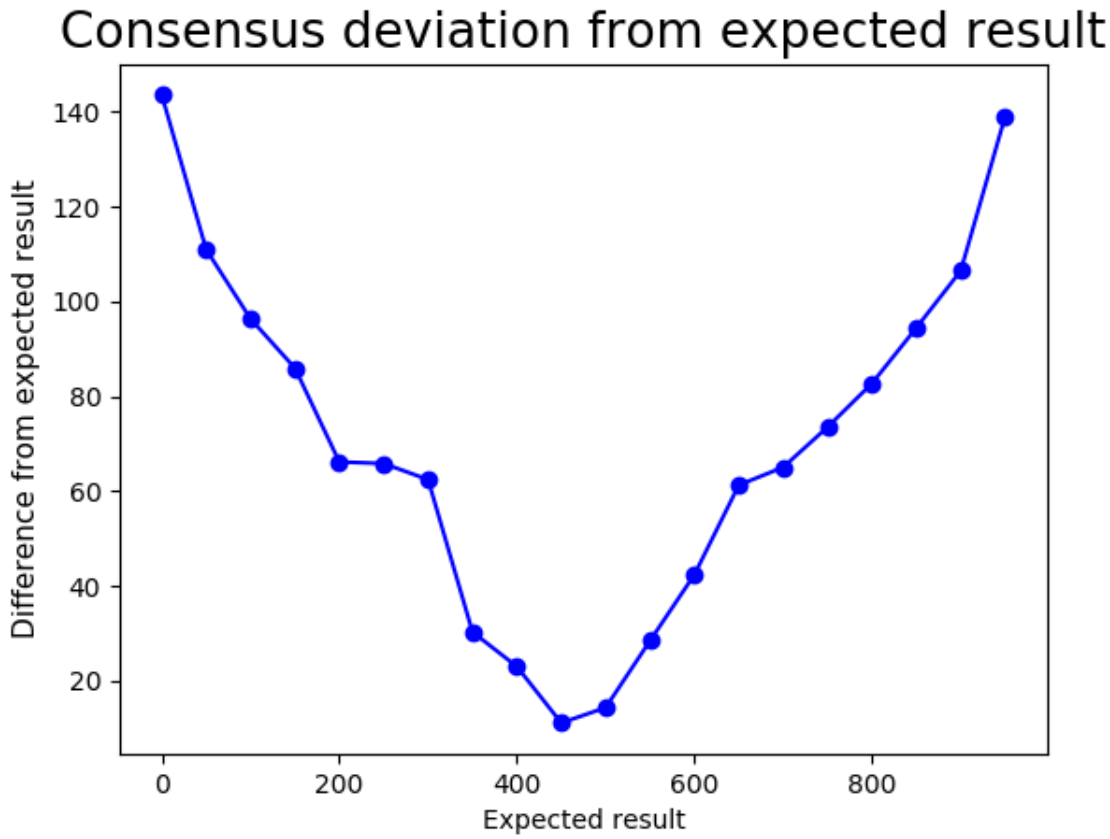

Figure 12: Algorithm without weighted adjacency matrix, and multiple direct neighbours for added node, described in 7, retested

When the seed of 45 was removed and the test was conducted with different random values, the phenomenon did not reoccur, as shown in figure 12. It is unknown why the result in figure 11 had the symmetric inconsistency. Both the results show noticeably worse adherence to expected value then when single direct neighbour was present, yet the highest accuracy around global average and steady and its linear increase dependant on distance from this average, remains. The inaccuracy is still much smaller than that shown in algorithm 4 and the resulting address does reflect the immediate neighbourhood of the new node.

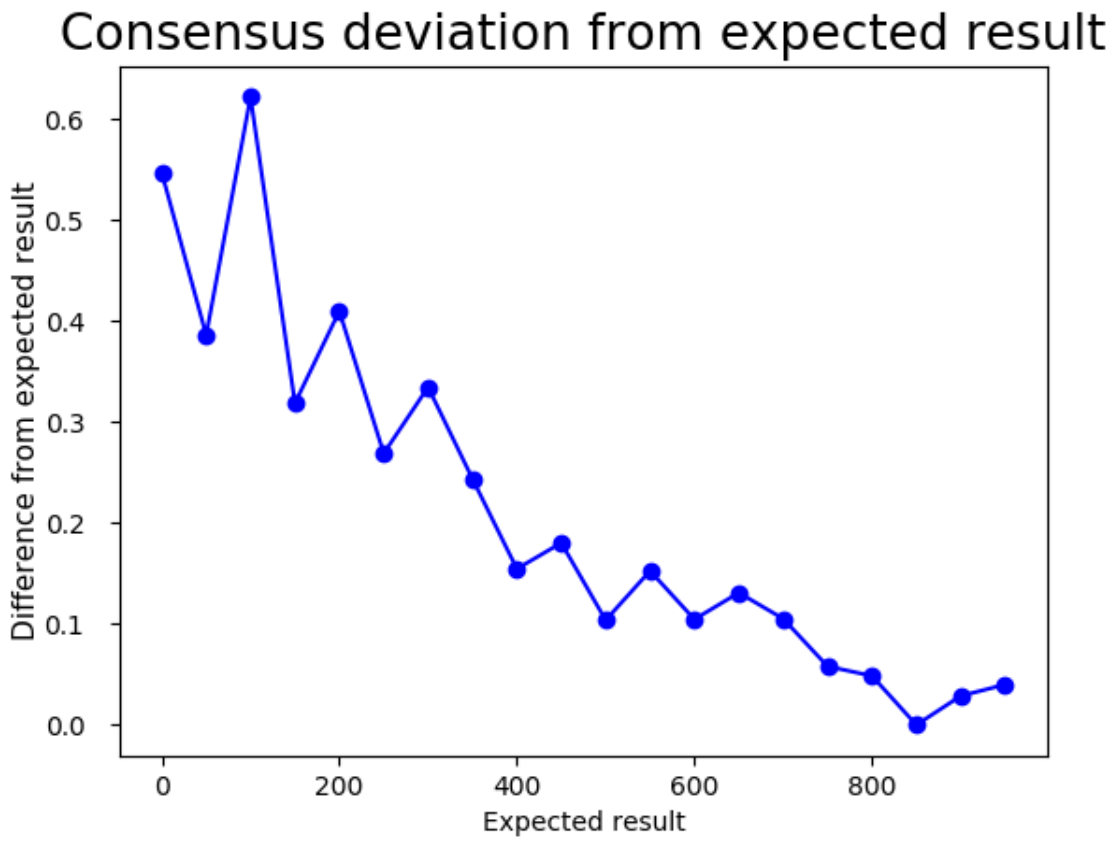

Figure 13: Algorithm without weighted adjacency matrix and cubic values of ping with multiple direct neighbours described in 8

When applying the equation described in 8, the consensus is farther from expected value when the expected value is smaller. The same behaviour can be observed in the case of adding a node connected to only one neighbour when using the same algorithm (see 10). This could be caused by other nodes who are close to the direct neighbours of the new nodes, who also have small ping, skewing the result, since with small values, the same difference in value is of greater proportion (5 is only smaller by 15 than 20, but is only 75% of its value, while 135 is smaller, also by 15, than 150, but is 90% of its value). The same effect would then be amplified by the method used to calculate the expected value, a simple average of direct neighbour addresses. It could also be the case that squared ping values make a step

too big and destabilize the whole system, it does not approach infinity, since maximal value of a node is kept at 1000 and since not every step is over-approximated the overall consensus does not remain at this maximum value.
Because the adjacency matrix used is not normalized, it is necessary to check for these over-approximations. The general equation, 1, avoids this problem by averaging the local neighbourhood in each iteration, thus returning value that is not higher that highest local value and not lower than lowest local value. To leverage the vector ping with this equation, we would need to normalize it as well, which is problematic, since maximal ping is not an information known globally and overestimating the ping would slow down the process. As a solution, it is possible to guess the maximal ping and check that local extremes were not exceeded should the ping be higher than our estimate.

$$x_i[k+1] = min(max(x_i[k] + (\frac{x_i[k] + \sum_{j \in N, a_{i,j}>0} x_j[k]}{1+d_i} - x_i[k]) * (\frac{ping_i}{norm}),$$

$$\min_{j \in N} x_j[k]), \max_{j \in N} x_j[k]), \quad i = 1, ..., n,\ a_{i,i} = 0,\ norm > 0,\ \frac{ping_i}{norm} < 1,\ d_i \geq 0 \tag{9}$$

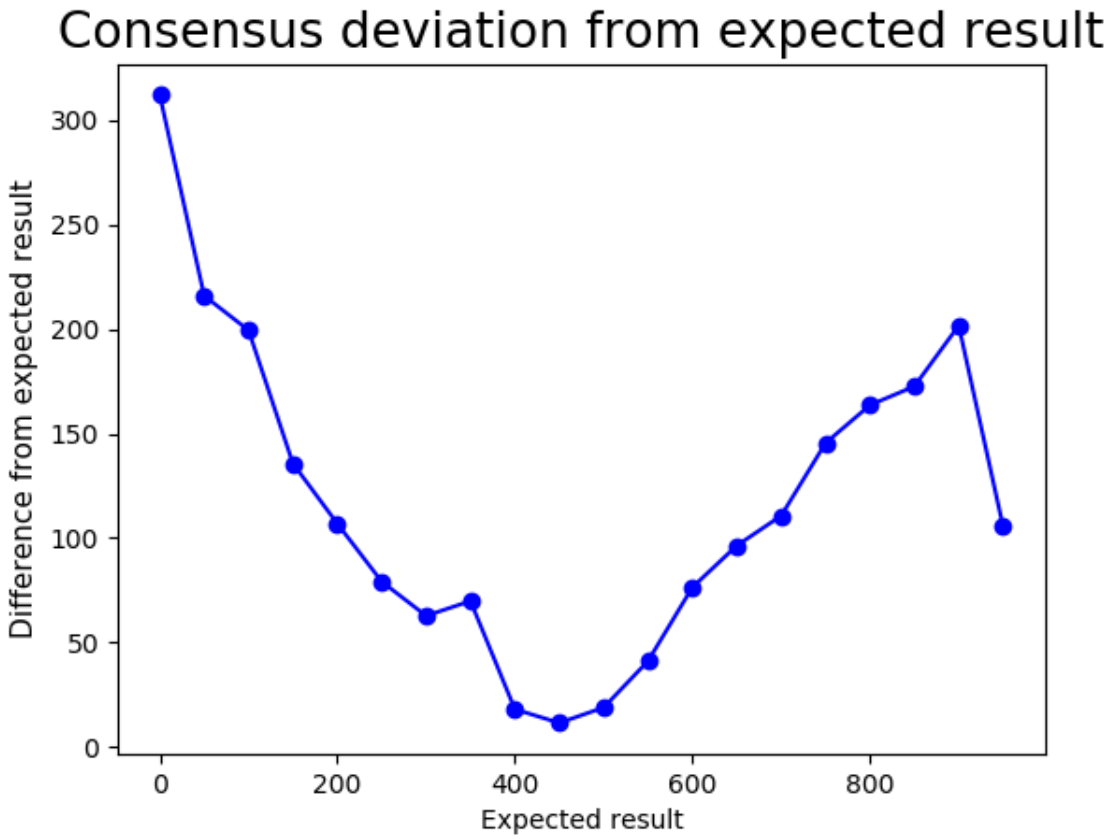

Figure 14: Local averages with linear ping 9

When the equation, 9, is applied, the influence of global average is quite significant, as can be seen in figure, 14. The vector of pings in its linear form does not slow down the nodes close to the origin enough. The ping can be more prominent if it is raised to a higher power, but since it has to be normalized in order to keep it from exceeding the largest original value, it will turn it into even smaller number, which slows down the whole consensus algorithm.

$$x_i[k+1] = min(max(x_i[k] + (\frac{x_i[k] + \sum_{j \in N, a_{i,j}>0} x_j[k]}{1+d_i} - x_i[k]) * (\frac{ping_i}{norm})^2,$$

$$\min_{j \in N} x_j[k]), \max_{j \in N} x_j[k]), \quad i = 1, ..., n,\ a_{i,i} = 0,\ norm > 0,\ \frac{ping_i}{norm} < 1,\ d_i \geq 0 \tag{10}$$

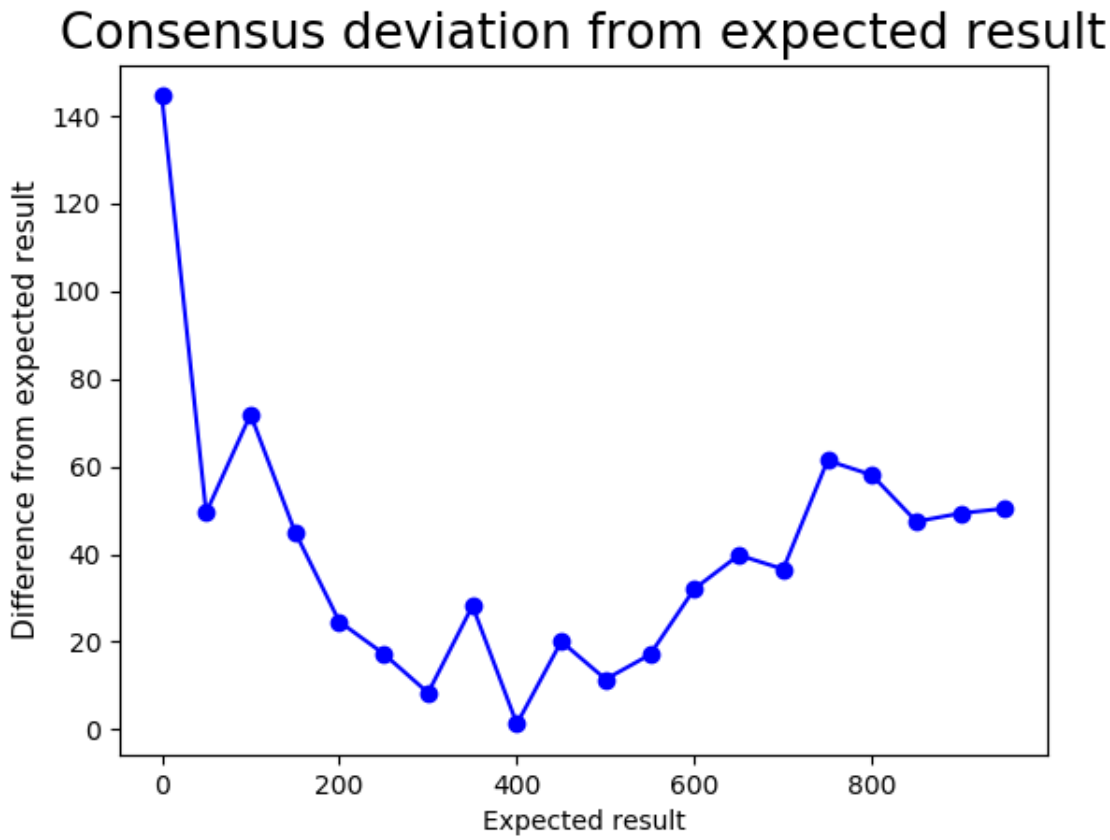

Figure 15: Local averages with quadratic ping 10

The result achieved with equation, 10, and shown in figure, 15, is indeed much improved, but the computation time has increased as well. The fail-safe that was set up is not being used, because all the results are being normalized. But since there

is control ensuring the results do not exceed local minima, it is possible to decrease the maximum ping estimate, even below what is known to be correct value.

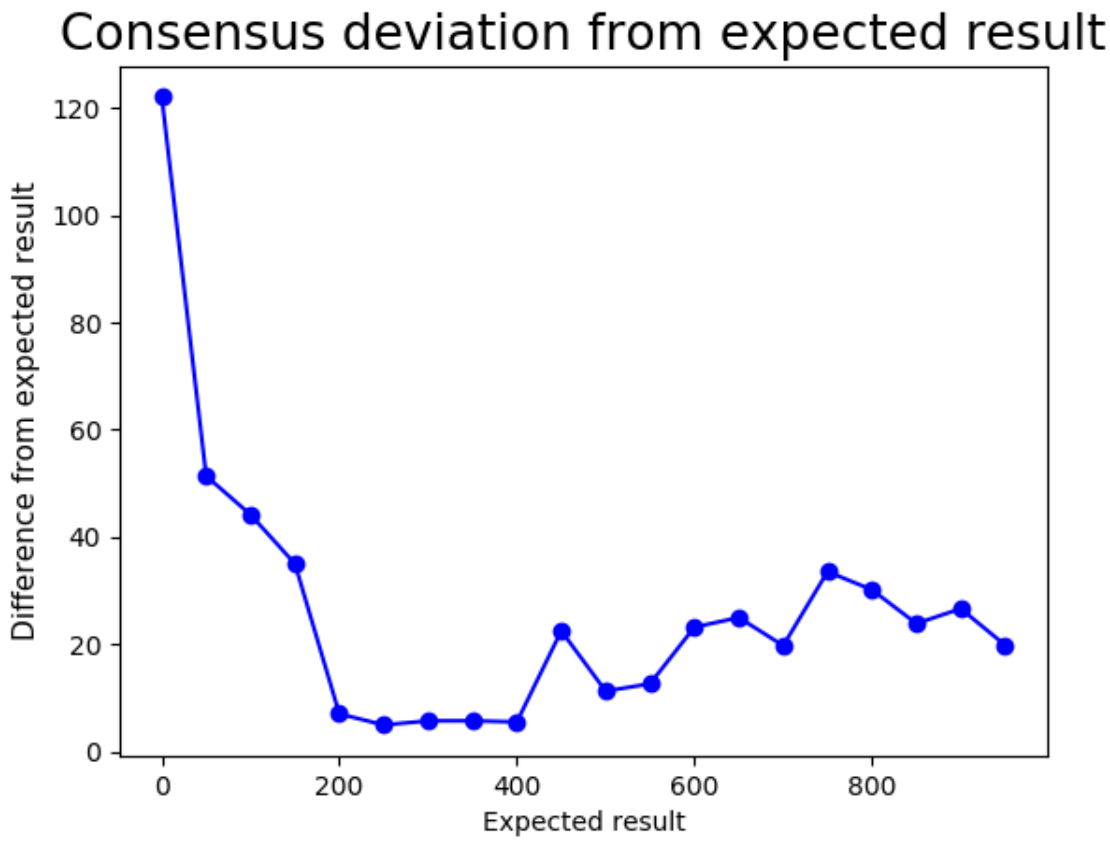

Figure 16: Local averages with quadratic ping exceeding normalized values 10

And indeed, this approach produces similar result in fraction of the time. Interestingly, even though the not fully normalized ping multiplier got as large as 4, in no iteration did the local consensus exceed local extremes.

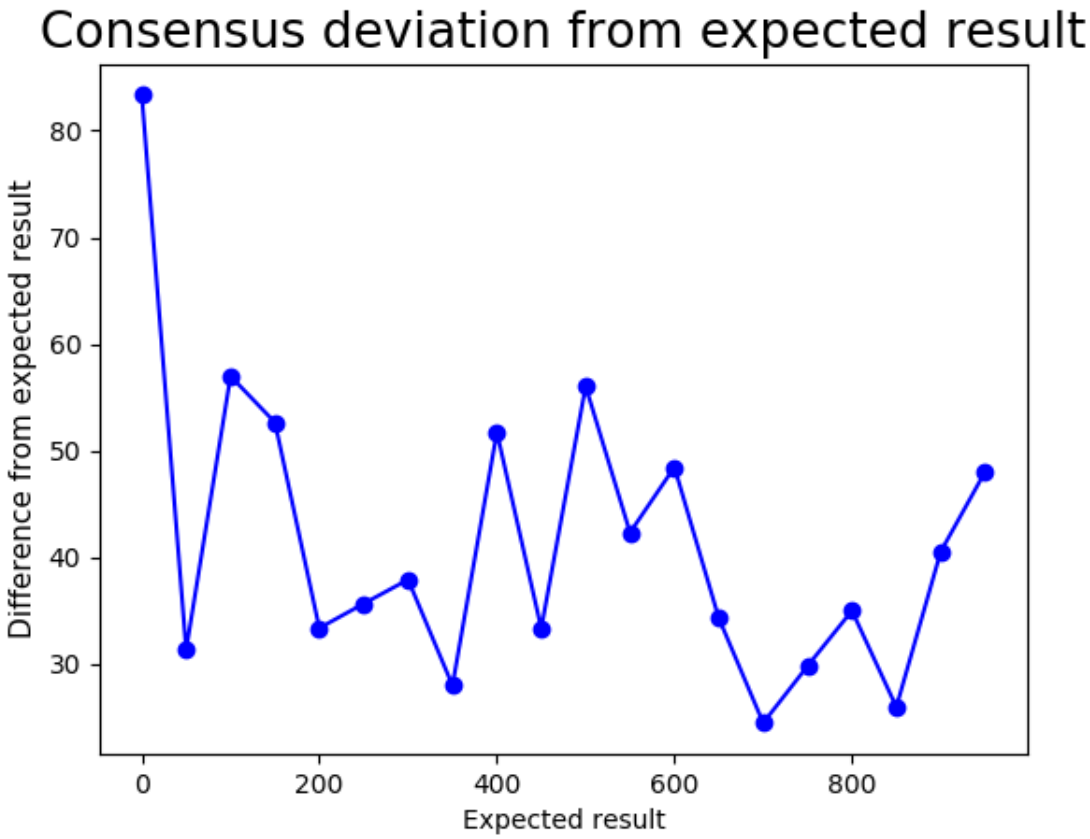

Figure 17: Local averages with quadratic ping exceeding normalized values and multiple neighbours to the new node 10

When multiple nodes are being added, the result is even better and again with no need for the fail-safe to be engaged. The strategy given by the equation, 10, with very optimistic maximal ping estimate is therefore the most consistent, best performing in the tests conducted and is so far the best candidate to be used in the NodeSkipper protocol, since the approaches attempting to make larger steps, using linear regression to estimate future values, are way too volatile. In future, it might be worth investigating whether proportional integral derivative (PID) control concept might provide better and safer estimates.
It is of course possible to reach consensus even with higher orders of the ping vector, such as

$$x_i[k+1] = min(max(x_i[k] + (\frac{x_i[k] + \sum_{j \in N, a_{i,j}>0} x_j[k]}{1+d_i} - x_i[k]) * (\frac{ping_i}{norm})^3,$$

$$\min_{j \in N} x_j[k]), \max_{j \in N} x_j[k]), \quad i = 1, ..., n,\ a_{i,i} = 0,\ norm > 0,\ \frac{ping_i}{norm} < 1,\ d_i \geq 0 \tag{11}$$

The problem is that with this approach, once the value is not properly normalized, it can grow quite large, even if it is just it is cubed, $2.5^2 = 6.25$, but $2.5^3 = 16.625$. This results in the max and min function to be used quite often, but when nodes are just oscillating between maximal and minimal values in their neighbourhoods, reaching consensus could prove challenging. A simulation that was using strategy identical to the one producing figure, 16, except the formula raised the vector of pings to third power, not second, as can be seen here, 11, did not reach consensus on a single IP address after 10 000 000 cycles.
To be sure that the result is converging to some common value, the vector needs be normalized to begin with, resulting in very small steps and very long procedure. Regardless, such a system is still guaranteed to reach consensus, [25].

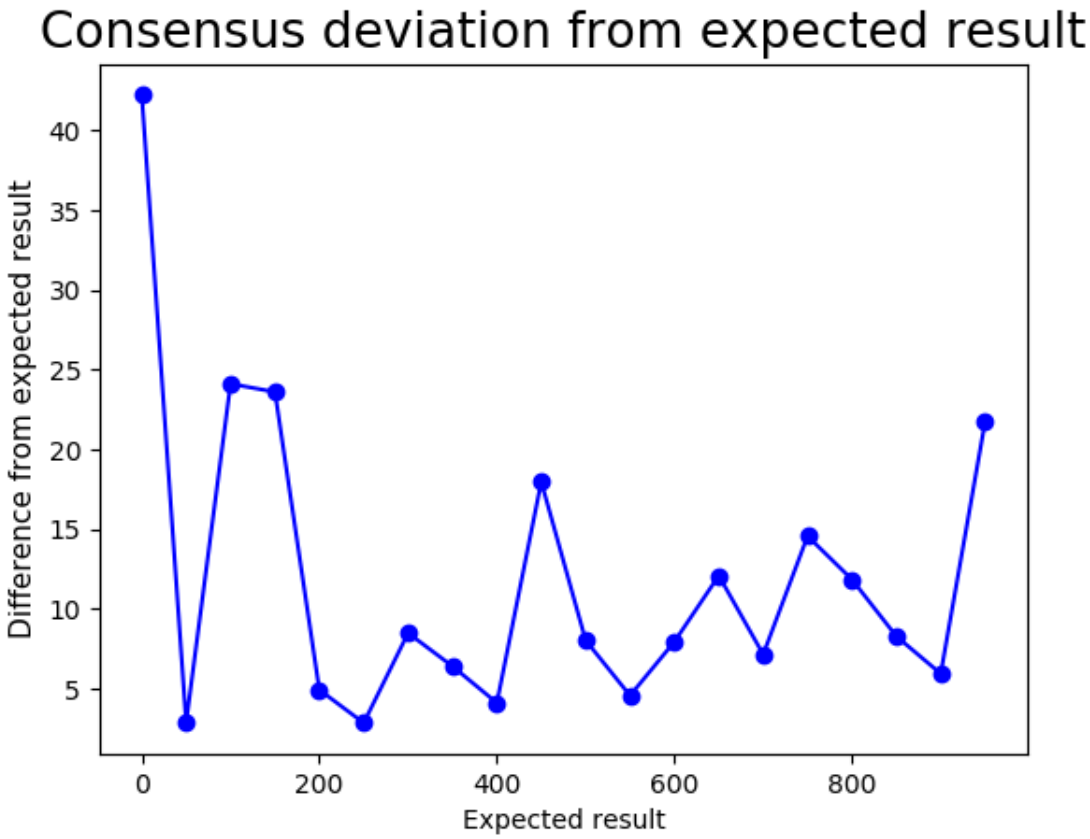

Figure 18: Local averages with cubic ping 10

This approach was already unacceptably slow with squared values, raising the power had the expected effect and increased the cycles needed to a point where even testing on consumer hardware is difficult and time consuming. The results, on the other hand, are very good. An approach that would produce results like those demonstrated by the cubic equation, 11, but one that also converges as quickly as 10 would be beneficial.

Since the cubic formula causes the nodes with high ping change values very quickly, while slowing down everything close to the new node, unfortunately it slows them so much that it is infeasible to wait for the result in the real time system, it could be used in a fixed amount of steps at the start of the algorithm. That way the asymptotic complexity is not affected, it serves just as linear complexity pre-computation.

Once these steps are completed, quadratic ping formula, 10, can be used, for the fast performance. The far away nodes would have their values adjusted, while the close nodes would not have changed much.

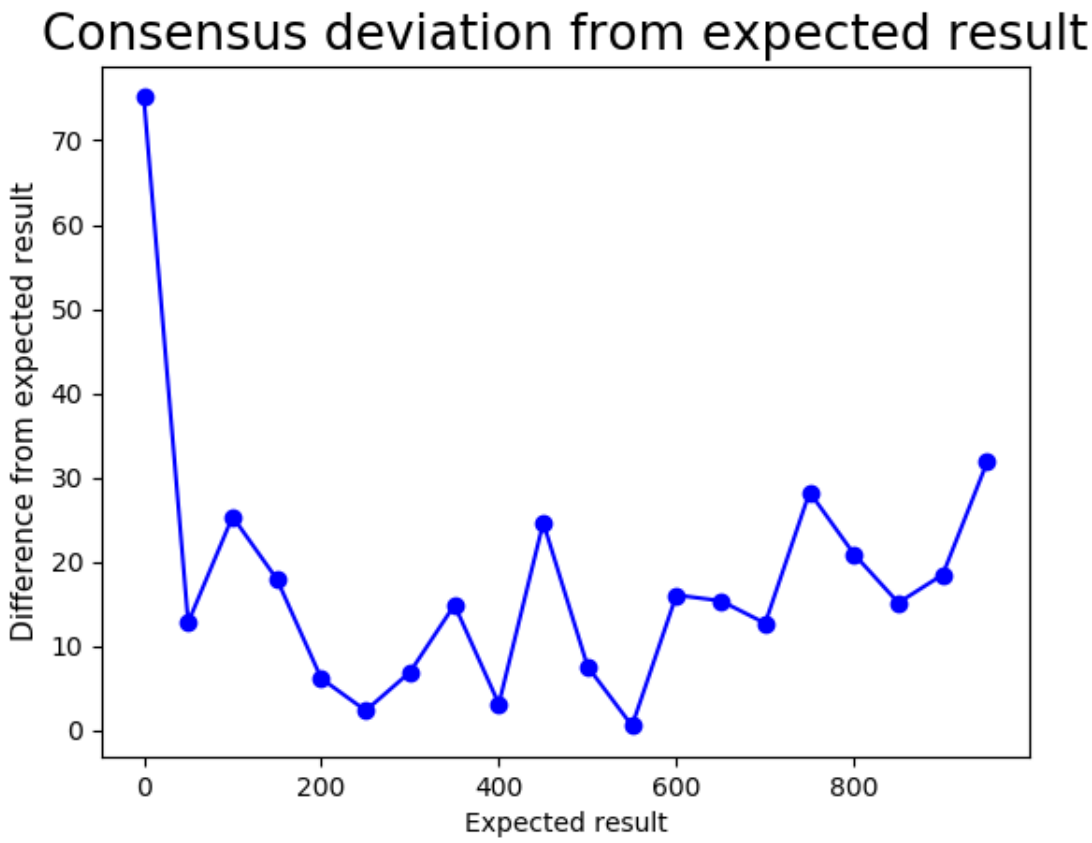

Figure 19: Local averages with cubic and square ping

And the result confirms those assumptions, while the value is not as close as with purely cubic ping, 18, it works better than purely square ping, 17, for a similar computational time.

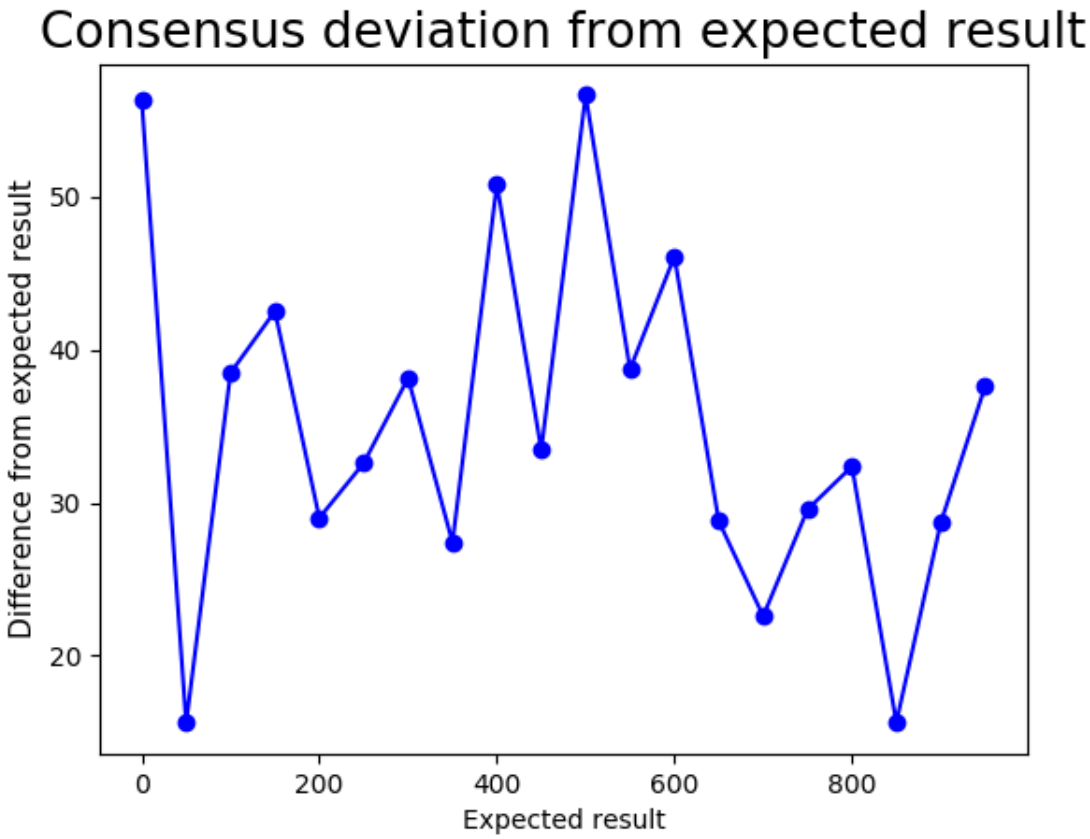

Figure 20: Local averages with cubic and square ping with multiple neighbours to the new node

In the figure, 20, it can be seen that even if the new node connects to multiple nodes, the result are still excellent.

From all the approaches in this chapter, the combination of local average formula multiplied by vector of ping raised to the third power, 11, and switching to a quadratic version of this formula, 10, after constant, 500 in this case, amount of steps, provides the best results. It guarantees convergence, does not return results based on distance from global average, returns values close to those of the neighbouring nodes and produces the result quickly.
In real case scenario, not all nodes will be at the same amount of cycles, like it was

in this testing environment. In that instance, it is possible for a node that switched to the quadratic algorithm to propagate this value along with its value in current iteration, but while there might be no apparent immediate risks connected to this approach, experience shows that when one entity could influence how computation on a difference entity is done, and even change the mode of operation at will, it often exposes serious security risks, (e.g. *downgrade attacks*), [15]. The safer option is for each node to switch once it does the set amount of cycles. This would cause different nodes to be using different equations at the same time, but such a situation will occur for only limited period of time and since only scale of the state change is affected, it would not prevent consensus from being formed, since the whole transition from k to $k + 1$ can still be represented by a single normalized matrix multiplied by a vector of states.

3.4 Format of the NodeSkipper packet

For compatibility reasons, the NodeSkipper packet is represented as an array of bytes with a specific format that allows for unambiguous encoding and decoding.

- **Protocol version**
  1 Byte - The count starts at 0, included for compatibility reasons.
- **Packet length**
  10 Bytes - The value is for the whole packet, the parts with fixed length are included as well.
- **Source address**
  16 Bytes - The address of packets original sender.
- **Destination address**
  16 Bytes - The address of the node for which the packet is intended for.
- **Next hop**
  16 Bytes - The address that previous node evaluated as most likely to be able to locate destination node in least amount of steps.
- **Packet data**
  Its length is *(Packet length - 75)*, it contains the data being sent.
- **Hash check**
  32 bytes - It contains a SHA-256 hash of the whole packet, excluding the *Hash check* section.

3.5 The node manager

The goal of node manager is to allow the individual nodes operate, from their point of view, as independent units that can pass messages to a few other independent units. Its secondary use is for benchmarking the performance of the whole system. It monitors the speed of node addition, message sending and node removal. It is the only part of the program that knows the addresses of all the nodes. However, it does not provide this information to anyone other component.

```
initialization;
set up list of message queues;
set up list of locks for synchronization;
set up map of IDs and addresses;
pick random address within range;
create new thread;
for 1:number of nodes do
    pick random number of direct neighbours;
    create new thread with IDs and addresses of direct neighbours;
end
for 0:number of nodes do
    start stopwatch;
    start thread;
    while no job done announcement do
        wait 1ms;
    end
    record stopwatch;
end
for 0:5 do
    for 0:10 do
        start stopwatch;
        send message between 2 random nodes;
        sum record time;
    end
    save(sum/10);
    for 0:10 do
        start stopwatch;
        remove random node;
        record stopwatch;
    end
end
```

Algorithm 8: Operation of the node manager

3.6 The NodeSkipper node

Any NodeSkipper node must be able to join the system, leave the system, accept another nodes in the system, look-up nodes within the system, send messages to other nodes, pass look-up and message requests, answer relevant look-up and next step requests and accept messages.

```
initialization;
look-up own IP;
wait of answer;
add nodes to remote connections;
diff = ((right address - left address)/2)*4;
while address-diff¿0 do
    look-up address-diff;
    wait of answer;
    add closer node to remote connections;
    diff = diff*4;
end
diff = ((right address - left address)/2)*4;
while address-diff¡(2**128)-1 do
    look-up address-diff;
    wait of answer;
    add closer node to remote connections;
    diff = diff*4;
end
```

Algorithm 9: Operation of the basic NodeSkipper node

4 Numerical testing

The tests in this section were done on CPU: Intel Core2 Quad Q9550 2.8GHz, GPU: ATI HD 3650, RAM: Corsair CM3X1024-1066C7 2GB (4 sticks), Python v3.7.0:1bf9cc5093

It is important for a NodeSkipper system to be able to add nodes quickly, especially since if two nodes want to join the system and both have an address that shares at least one closest neighbour, the second node to be announced has to wait for an answer until the addition process of the first node is complete.

The following test was done by node manager, (see 3.5). Nodes were added from 0 to 100 ten times, the time intervals were then averaged.

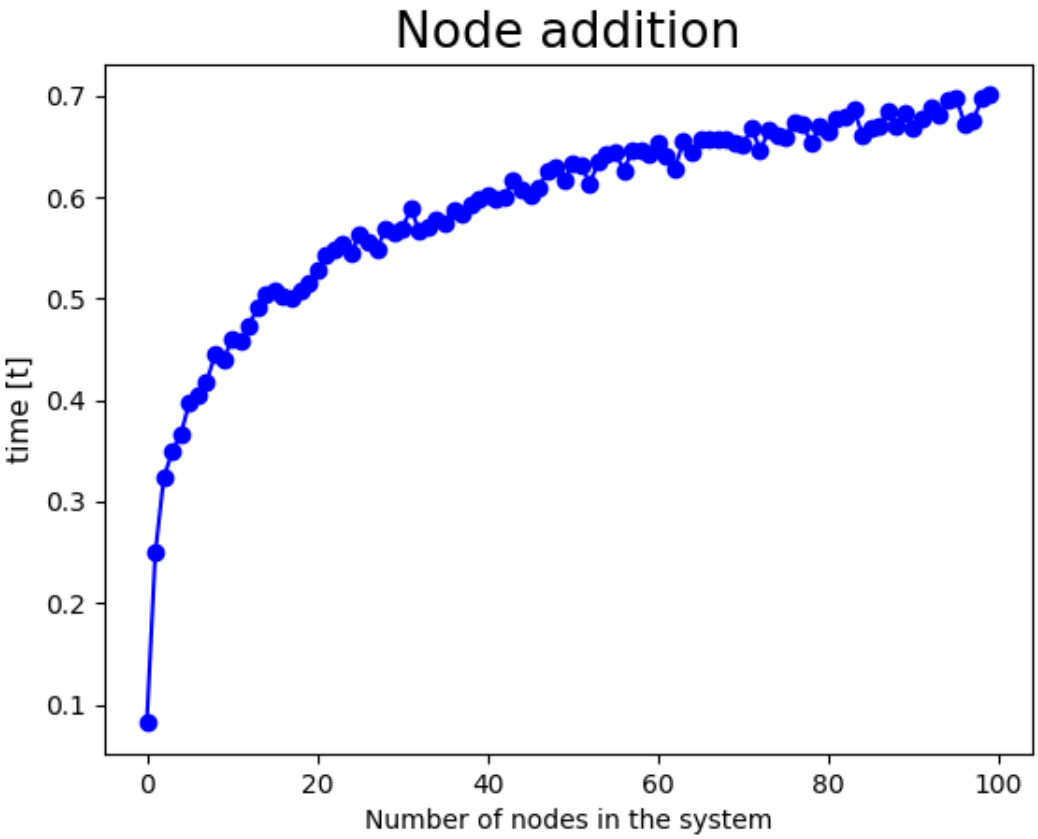

Figure 21: Time needed to add a node to the system

The time needed to add the first few nodes is very low, since there is not much network that needs to be build, but while it does grow fast, it is clearly slowing its growth and no result exceeded 1 second.
Node removal is quite simple, since the exiting node only needs to send messages to its list of contacts. Once they are sent, it can end its own thread. 50 nodes from 100 node graph were removed 10 times and the results averaged.

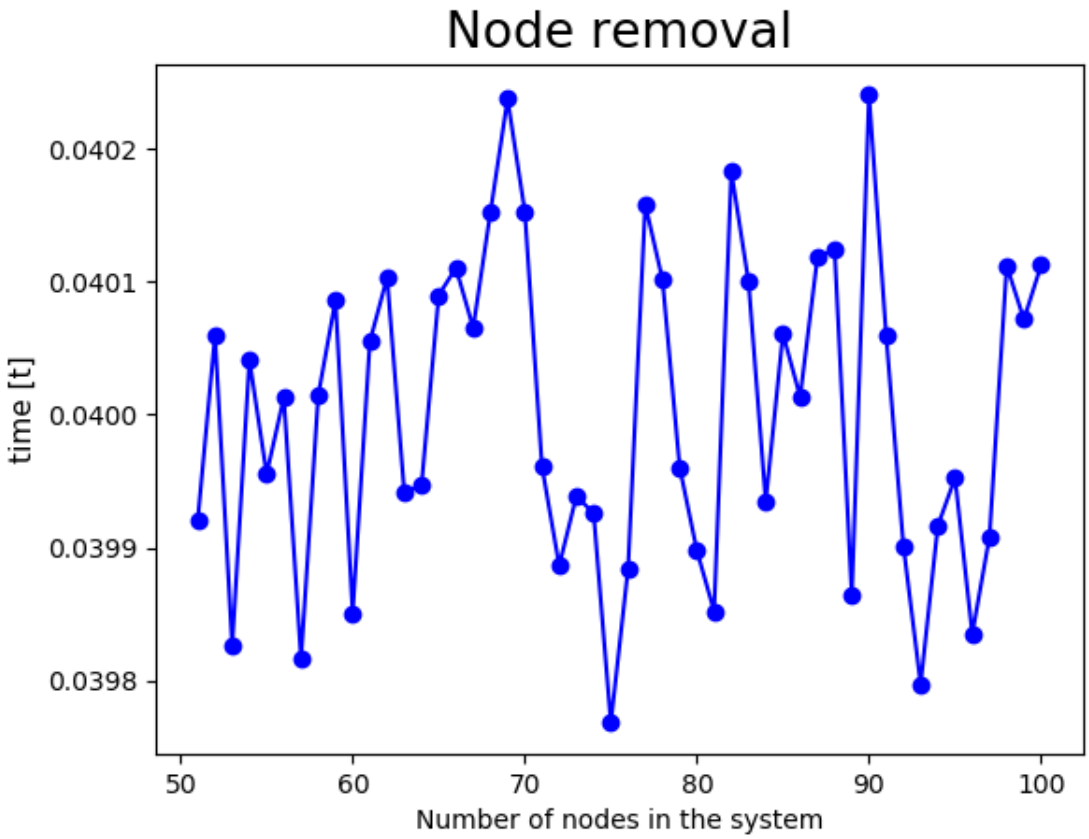

Figure 22: Time needed to remove a node from the system

While the disproportionality of the graph might make it seem very inconsistent, the differences in time needed are very slow and there does not seem to be any observable growth of time needed for the time needed to remove the node and the overall time needed is very small.
The last test concerns sending messages. This functionality is crucial, as it is used in the other processes. 100 messages were sent at the start and then 4 more times upon 10 nodes being removed. The time was measured from a time node A sent a message to a time where node B received the message and then averaged.

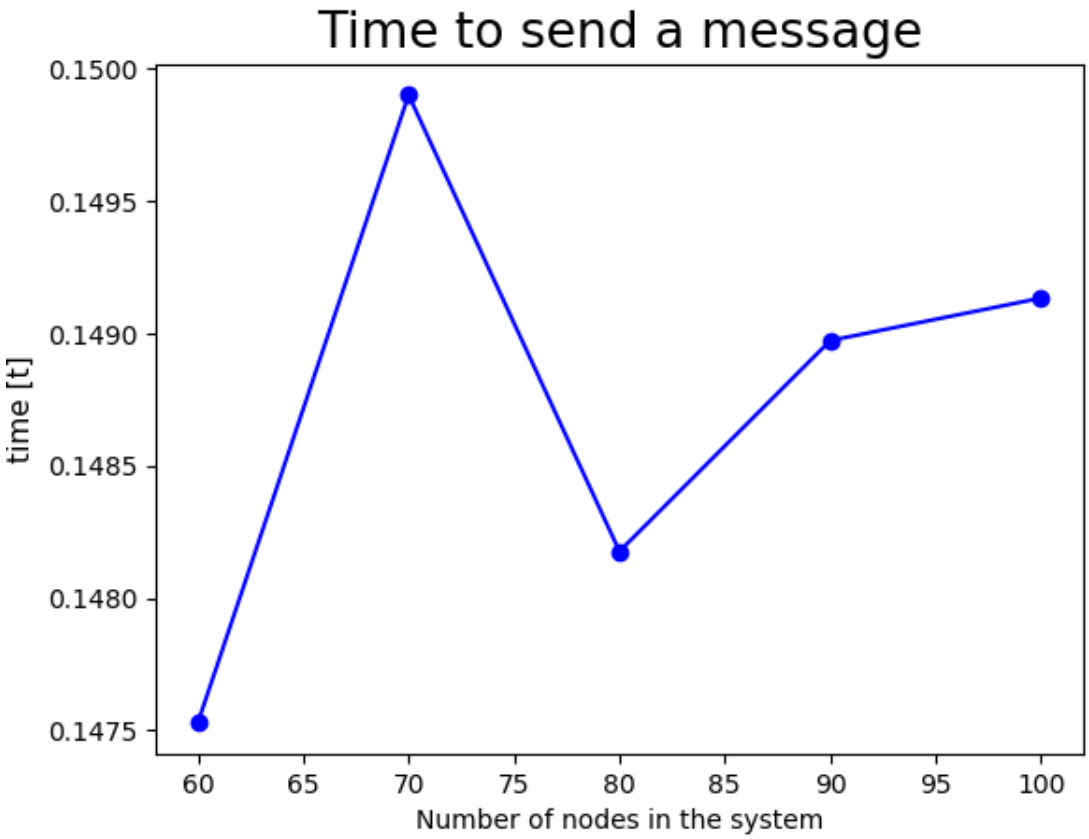

Figure 23: Time needed to send a message within the system

Once again, there is very little variation and the messages were delivered, on average, in a very short time.

5 Investigating the use of NodeSkipper in creating a Starcraft 2 player simulation

Creating a fully featured agent able to play a Starcraft 2 match is a challenging tasks, as Starcraft 2 is one of the most complex human games, which state-space too large to be searched by current technology. The task could be simplified using the *divide & conquer* approach. Instead of implementing the whole strategy, it is

possible to only implement logic for individual components, the army units, the workers and the structures.

5.1 Worker

The motivation of a worker is to collect resources. Possible challenges are enemy army units trying to kill it, and lack of resource fields with close drop-off points.

```
initialization;
while is alive do
    search for unsaturated resource field with drop-off point;
    while field exists & no enemy army unit nearby do
        collect resources;
    end
    if enemy nearby then
        move towards and past the closest defensive unit or structure;
    end
    elsif no unsaturated safe resource field found
     evoke consensus algorithm on claiming a new resource collecting
     base;
end
```

Algorithm 10: Worker internal logic

Should many workers have problems find unsaturated resource fields, the consensus will change towards claiming a new base. Army units having trouble fighting the enemy might skew consensus the other way, since in their motivation another base makes protection harder.

5.2 Army unit

The army unit wants to maximize its value by obtaining upgrades, increasing its development (i.e. threat posing to enemy, usually achieved by positioning) and causing damage to the enemy. Army unit is willing to be killed if it causes more damage than what is its cost, given economical situation of opponent is comparable.

```
initialization;
while is alive do
    evoke consensus algorithm on purchasing upgrade for its type of unit
      if one is not in progress;
    search for a place to attack; if strong enough to attack then
        attack;
    end
    else
        evoke consensus on attacking the target;
    end
    if not enough resources to attack then
        evoke consensus on building additional units;
    end
end
```

Algorithm 11: Army unit internal logic

If the army is strong enough, it would never have problem finding a target to attack, unless the game was won. Should many units have this problem, it is likely more units are needed.

5.3 Structures

The structures provide vision of the battlefield, open technology paths and enable resources to be collected. Their motivation is not being killed, if no buildings remain, the game is lost.

```
initialization;
for Each tech path do
    if Tech path is open then
        evoke consensus on this technology;
    end
    if enemy sighted then
        evoke consensus on attacking the target;
    end
    if not enough resources to attack then
        evoke consensus on building additional units;
    end
end
```

Algorithm 12: Structure internal logic

If the army encounters enemy units susceptible to some technology, they will support it in consensus process. Its reaction to enemy units is stronger, since army

unit could distract itself with easier to attack target. If structure does not find support for its defence, it immediately evokes consensus on building more army.

6 Conclusion

The NodeSkipper showed that it can deliver the same benefits as existing decentralized routing protocols, while also providing less reliance on the network structure and providing an extra functionality in its ability to reach consensus, which has the potential to improve the performance of many instances of its implementation.

References

[1] Andrea Pitzschke, Adam Schikora, Heribert Hirt, MAPK cascade signalling networks in plant defence, *Current opinion in plant biology 12, no. 4*, 2009

[2] M. E. J. Newman, Networks: An Introduction, *Oxford University Press*, 2010

[3] Kostadin Kratchanov, Emilia Golemanova, Tzanko Golemanov, Control Network Programming Illustrated: Solving Problems with Inherent Graph-Like Representation, *Seventh IEEE/ACIS International Conference on Computer and Information Science*, 2008

[4] Guy Latouche, V. Ramaswami, V. G. Kulkarni, Introduction to matrix analytic methods in stochastic modeling, *Journal of Applied Mathematics and Stochastic Analysis 12, no. 4*, 1999

[5] Kevin B. Clark, Insight and analysis problem solving in microbes to machines. *Progress in Biophysics and Molecular Biology 119, no. 2*, 2015

[6] Roshan L. Sharma, A guide to data communications: interconnecting LANs. *IEEE Spectrum 28, no. 8*, 1991

[7] Ernesto Van der Sar, Internet Backbone Cogent Blocks Cloudflare's New 'Pirate Site' IP-addresses, https://torrentfreak.com/internet-backbone-cogent-blocks-cloudflares-new-pirate-site-ip-addresses-170322/ Accessed 2017-03-23

[8] Dan Goodin, Strange snafu misroutes domestic US Internet traffic through China Telecom, https://arstechnica.com/information-technology/2018/11/strange-snafu-misroutes-domestic-us-internet-traffic-through-china-telecom/ Accessed 2018-11-18

[9] Kimberly C. Claffy, George C. Polyzos, H-W. Braun, Traffic characteristics of the T1 NSFNET backbone, *IEEE INFOCOM'93 The Conference on Computer Communications, Proceedings, pp. 885-892*, 1993

[10] W. Pugh, Skip lists: a probabilistic alternative to balanced trees, *Communications of the ACM, 33.6*, 1990

[11] Petar Maymounkov, David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric, *In International Workshop on Peer-to-Peer Systems, pp. 53-65. Springer, Berlin, Heidelberg*, 2002

[12] Jeffrey A. Hart, The teletel/minitel system in France, *Telematics and Informatics 5, no. 1*, 1988

[13] John M. McQuillan, David C. Walden, The ARPA network design decisions, *Computer Networks 1, no. 5*, 1977

[14] Jon Brodkin, Undersea cable damage wipes out most Internet access in Tonga islands, https://arstechnica.com/information-technology/2019/01/undersea-cable-damage-wipes-out-most-internet-access-in-tonga-islands/ Accessed 2018-02-03

[15] Karthikeyan Bhargavan, Christina Brzuska, Cédric Fournet, Matthew Green, Markulf Kohlweiss, Santiago Zanella-Béguelin, Downgrade resilience in key-exchange protocols, *In 2016 IEEE Symposium on Security and Privacy (SP), pp. 506-525. IEEE*, 2016

[16] Vinton G. Cerf, The day the Internet age began, *Nature 461, no. 7268*, 2009

[17] Matthew G. Naugle, Illustrated TCP/IP: a graphic guide to the protocol suite, *John Wiley & Sons, Inc.*, 1998

[18] R. Eftimie, G. De Vries, M. A. Lewis, Complex spatial group patterns result from different animal communication mechanisms, *Proceedings of the National Academy of Sciences 104, no. 17*, 2007

[19] R. Bush, D. Meyer, et. al. Some Internet Architectural Guidelines and Philosophy *The Internet Society, RFC 3439* 2002

[20] Connie M. Schardt, Electronic mail service: applications in the Pacific Northwest region, *Bulletin of the Medical Library Association 71, no. 4*, 1983

[21] Jean-Louis Deneubourg, Simon Goss, Collective patterns and decision-making, *Ethology Ecology & Evolution 1, no. 4*, 1989

[22] The Babbage Engine: The Engines, *Computer History Museum*, Accessed 2019-03-02

[23] Frank L. Lewis, Hongwei Zhang, Kristian Hengster-Movric, Abhijit Das, Cooperative control of multi-agent systems: optimal and adaptive design approaches, *Springer Science & Business Media*, 2013

[24] Christopher H. Sterling, Military Communications: From Ancient Times to the 21st Century, *Abc-clio*, 2008

[25] Ali Jadbabaie, Jie Lin, A. Stephen Morse, Coordination of groups of mobile autonomous agents using nearest neighbor rules, *Departmental Papers (ESE)*, 2003

[26] Reza Olfati-Saber, Richard M. Murray, Consensus problems in networks of agents with switching topology and time-delays, *IEEE Transactions on automatic control 49, no. 9 1520-1533*, 2004

[27] Robert W. Floyd, Algorithm 97: shortest path, *Communications of the ACM 5, no. 6*, 1962

[28] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, Introduction to Automata Theory, Languages, and Computation (3rd ed.), *Pearson*, 2013

[29] Ernesto Van der Sar, Facebook Uses BitTorrent, and They Love It, https://torrentfreak.com/facebook-uses-bittorrent-and-they-love-it-100625/ Accessed 2018-06-15

[30] Adrian Florea, Lucian N. Vintan, Ioan Z. Mihu, Understanding and Predicting Indirect Branch Behavior, *Studies in Informatics and Control 13, no. 1*, 2004

[31] Guoguang Wen, Distributed cooperative control for multi-agent systems, *Ecole Centrale de Lille*, 2012

[32] Ronald Nick Kalla, Minh Michelle Quy Pham, Balaram Sinharoy, John Wesley Ward III, Method and apparatus for selecting an instruction thread for processing in a multi-thread processor, *U.S. Patent 7,360,062*, issued April 15, 2008

[33] Wei Ren, Randal W. Beard, and Ella M. Atkins, Information consensus in multivehicle cooperative control, *IEEE Control systems magazine 27, no. 2*, 2007

[34] Richard C. Holt, Some deadlock properties of computer systems, *In Proceedings of the third ACM symposium on Operating systems principles, pp. 64-71. ACM*, 1971

[35] Derek B. Kingston, Wei Ren, Randal W. Beard, Consensus algorithms are input-to-state stable, *In Proceedings of the 2005, American Control Conference, pp. 1686-1690. IEEE*, 2005

[36] https://www.gustorotondo.it/storia-di-internet/ Accessed 2019-03-03

[37] O. Vinyals, K. Calderone et al., StarCraft II: A New Challenge for Reinforcement Learning, *Preprint Arxive*, 2017

[38] http://www.routexp.com/2017/04/osi-model-vs-tcpip-model.html Accessed 2019-03-17

[39] https://hostingfacts.com/internet-facts-stats/ Accessed 2019-05-17

[40] Michael J. Beckerle, Overview of the START (* T) multithreaded computer, *In Digest of Papers. Compcon Spring, pp. 148-156. IEEE*, 1993

[41] Konrad Hinsen, Parallel scripting with Python, *Computing in Science & Engineering 9, no. 6*, 2007

[42] https://torrentfreak.com/south-korea-expands-site-blocking-efforts-with-sni-eavesdropping-190214/ Accessed 2019-05-17

[43] https://torrentfreak.com/movie-tv-show-companies-want-subtitle-sites-blocked-down-under-180802/ Accessed 2019-05-17

[44] https://arstechnica.com/tech-policy/2018/03/20-porn-unblocking-fee-could-hit-internet-users-if-state-bill-becomes-law/ Accessed 2019-05-17

[45] https://torrentfreak.com/12564-sites-preemptively-blocked-to-protect-indias-most-expensive-movie-181129/ Accessed 2019-05-17

[46] https://torrentfreak.com/japan-plans-to-criminalize-pirate-link-sites-up-to-five-years-in-jail-for-operators-181015/ Accessed 2019-05-17

[47] Ulrich Kelber, Germany's Federal Commissioner for Data Protection and Freedom of Information, Reform des Urheberrechts birgt auch datenschutzrechtliche Risiken, https://www.bfdi.bund.de/DE/Infothek/Pressemitteilungen/2019/10_Uploadfilter.html Approved translation: http://www.fosspatents.com/2019/02/germanys-federal-data-protection.html#translation Accessed 2019-03-01

Appendices

Enclosed CD
There is an enclosed CD-ROM with every print of this thesis.
The contents of the CD-ROM are:
PDF version of this thesis
Python project with implementations of the NodeSkipper protocol used to benchmark it
Graphs and figures, some of which were included in the thesis