HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

Anke Küstner

A Simulation Framework for Connecting In-Body Nano Communication with Out-of-Body Devices

Bachelor Thesis in Computer Science

17 June 2020

Please cite as:
Anke Küstner, “A Simulation Framework for Connecting In-Body Nano Communication with Out-of-Body Devices,” Bachelor Thesis (Bachelorarbeit), Heinz Nixdorf Institute, Paderborn University, Germany, June 2020.

Distributed Embedded Systems (CCS Labs)
Heinz Nixdorf Institute, Paderborn University, Germany

Fürstenallee 11 · 33102 Paderborn · Germany

http://www.ccs-labs.org/

A Simulation Framework for Connecting In-Body Nano Communication with Out-of-Body Devices

Bachelor Thesis in Computer Science

vorgelegt von

Anke Küstner

geb. am 20. April 1996
in Wolfhagen

angefertigt in der Fachgruppe

Distributed Embedded Systems
(CCS Labs)

Heinz Nixdorf Institut
Universität Paderborn

Betreuer:	**Lukas Stratmann**
Gutachter:	**Falko Dressler**
	Holger Karl
Abgabe der Arbeit:	**17. Juni 2020**

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde.
Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Declaration

I declare that the work is entirely my own and was produced with no assistance from third parties.
I certify that the work has not been submitted in the same or any similar form for assessment to any other examining body and all references, direct and indirect, are indicated as such and have been cited accordingly.

(Anke Küstner)
Paderborn, 9 December 2020

Abstract

Since our mankind will always be facing struggle against diseases and illnesses, it is crucial to continuously research on and develop new approaches and tools for medical treatments that enable us new insights and possibilities to cope with these medical issues. This thesis presents a promising tool, namely the connection between *in-body nanonetworks* and *out-of-body devices*, which give us insight into our body. These networks are able to do measurements within the body or eliminate detected problems, e.g., morbid cells. This thesis focuses on the controllability of nanobots from outside the body which is tested by building a simulation framework, because experiments in this field cannot be simply carried out on a human or animal body due to ethical reasons. A simulation framework was developed, which connects an out-of-body device to a gateway via a *Low-Rate Wireless Personal Area Network (LR-WPAN)* and provides the connection between a gateway and nanobots or only among nanobots with a *proximity approach*. In addition the simulation framework *BloodVoyagerS (BVS)* was used to simulate the cardiovascular system of a human body. In the evaluation trends were found on latency, hop count and the amount of blood vessels in which medical issues can be reliably detected.

Kurzfassung

Heutzutage kann eine umfassende Anzahl an Erkrankungen mit medizinischen Behandlungen adressiert werden. Dennoch gibt es viele Krankheiten, die bisher noch nicht ausreichend oder sogar gar nicht behandelt werden können. Aufgrund dessen werden immer wieder neue medizinische Werkzeuge entwickelt wie zum Beispiel Netzwerke bestehend aus Nanorobotern, die man im Körper einsetzen kann um Messungen vornehmen zu können oder sogar an entdeckten Problemen zu arbeiten. Diese Arbeit befasst sich damit, wie solche Netzwerke von außen durch eine Schnittstelle am Körper kontrolliert werden können. Aus ethischen Gründen ist es nicht vertretbar Experimente an Lebewesen auszuführen, daher müssen solche Experimente simuliert werden, um Ergebnisse zu erhalten.

Im Zuge dieser Arbeit ist ein *Simulations-Framework* entwickelt worden, das eine erste Annährung an ein solches Kommunikationsnetzwerk darstellt. Während die Verbindung zwischen außer-körperlichen Geräten und einem Gateway durch eine *LR-WPAN* Verbindung implementiert wurde, wurden die Verbindung zwischen einem Gateway und den Nanorobotern und die Verbindung zwischen Nanorobotern selber durch einen abstrakteren Ansatz simuliert, der es ihnen nur erlaubt die Nachricht untereinander auszutauschen, wenn sie in der Nähe voneinander sind. Zusätzlich wurde *BVS* verwendet um das kardiovaskuläre System eines Menschens zu simulieren. In einer ersten Evaluierung des Systems konnten Trends für die Latenz, die Anzahl der Zwischenstationen einer Nachricht und die Anzahl der Blutgefäße, in denen medzinische Auffälligkeiten zuverlässig gefunden werden können, entdeckt werden.

Contents

Abstract **iv**

Kurzfassung **v**

1 Introduction **1**

2 Fundamentals **4**
2.1 Wireless Body Area Networks 4
2.2 Nanobots and Nanonetworks 7
2.3 Overview of ns-3 9
2.3.1 Packets, Headers, and Trailers 9
2.3.2 The Tracing System 10
2.4 Implementation of IEEE 802.15.4-2006 in ns-3 12
2.5 BloodVoyagerS 14

3 Implementation **18**
3.1 LR-WPAN connecting Laptop with Gateway 19
3.2 Gateway Functionality 21
3.3 Proximity-Approach 22
3.4 Configuration 25

4 Evaluation **26**
4.1 Parameter & Metrics 26
4.2 Results 29
4.2.1 Latency 29
4.2.2 Fraction of Vessels 34
4.2.3 Hop Count 36

5 Conclusion **38**
5.1 Future Work 39

Bibliography **44**

Chapter 1

Introduction

Improving medical treatments is a research field that will continuously face new challenges. Today our health-care system is able to treat many diseases and illnesses well enough so that patients have a chance of recovery or they are at least able to better live with restrictions. However, there are situations such as the COVID-19[1] pandemic, that our health-care systems cannot handle due to lack of vaccines. Furthermore, there are diseases such as cancer and cardiovascular diseases, which remain two of the leading causes of death in the last two decades[2] and cannot be treated sufficiently effective, or other diseases which cannot be treated at all with the current treatment options. Additionally, some medical treatments also destroy healthy cells within the body and can therefore lead to a weakened immune system. Also, it makes a patient more vulnerable to other diseases, which in the worst case can cause a patient's death.

If computer science is included in the research for new treatments, completely new possibilities arise to help a patient at all or develop much more precise medical treatments. This field of research can be split up into out-of-body technologies and in-body technologies. The former are currently already in use in the form of sensors [1], which assist in monitoring the patient's body values without keeping them stationary [2]. These sensors can also be connected as a Body Area Network (BAN) and forward data to a physician or nursing staff who can react better and faster to changes in the monitored values. In-body technologies are the part of research in which completely new challenges arise. It focuses on how to develop nanodevices that can be used inside the body [3] and prevent the material used from being rejected by the body. These nanoscale technologies should enable to get a deeper understanding of the human body or even to achieve higher mobility for patients. Akyildiz et al. [3] suggest using nanobots since they have the potential to

[1] https://www.who.int/health-topics/coronavirus#tab=tab_1

[2] https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

form networks with the ability to make measurements inside the body, send collected information to the outside [4] or maybe even work on detected problems. Hence, there exist mainly concepts on nanonetworks and in-body communication such as it is presented in Dressler and Akan [5].

If the aforementioned two research fields of in-body technologies and out-of-body technologies were combined in one system, this could build a powerful tool for physicians to be able to precisely detect where a morbid cell or a virus is located in the human body and react to the detected problem [6]. Due to ethical reasons, the testing of nanodevices in a human or animal body cannot simply be carried out since these procedures can have a negative influence on the health of a living being [7]. Resulting from this, the goal of this thesis is to build up a simulation framework which combines the physical environment, a human body, nanodevices and an out-of-body communication network. Nanodevices, which form a nanonetwork by being able to communicate with each other, enable the possibility to send collected information to the outside of the body. On the other hand, the out-of-body communication network provides the ability to access the nanodevices from the outside of the body by sending commands from an out-of-body device also referred to as smart device.

The physical environment can be simulated by the framework BloodVoyagerS designed by Geyer et al. [7]. In BloodVoyagerS nanobots for in-body communication already do exist. However, at the moment these nanobots are just unconnected nodes. To be able to access theses nodes from a laptop a physician can use a gateway as it is proposed in Dressler and Fischer [4] and Galluccio et al. [8]. Through the gateway, a physician can also gather data from the inside of the human body by the nanobots.

The proposed system architecture for this thesis is modeled in accordance to Dressler and Fischer [4] and Santagati et al. [9]. For the design of the out-of-body communication part, which will be the connection between a laptop and a gateway, the ns-3 implementation on Low-Rate Wireless Personal Area Network (LR-WPAN) devices can be used [10]. The gateway should be able to forward information on tasks from a laptop to the nanobots by using ultrasonic communication as proposed by Santagati et al. [9]. For controlling the desired group of nanobots Function Centric Networking (FCN) [6] will be used.

If a nanobot receives a message from another nanobot, the message should simply be forwarded to the gateway. If it receives a message from the gateway, it should check its ID, function, and location and compare them to its own data to decide whether to just forward the message to all other nanobots or to additionally perform a task because the nanobot itself was one of the addressed ones.

If the simulation system considers all important details of the human body which can affect communication between nanonetworks as well as the effects nanodevices

can have on a human body, the system might be tested and evaluated carefully for also being able to perform experiments on a human body.

While writing this bachelor thesis we also submitted a short paper to ACM NanoCom 2020 on the basis of this thesis: A. Kuestner et al., "A Simulation Framework for Connecting In-Body Nano Communication with Out-of-Body Devices," in *7th ACM International Conference on Nanoscale Computing and Communication (ACM NanoCom 2020)*, under review, College Park, MD, Sep. 2020 [11].

Chapter 2

Fundamentals

In this chapter, I will first give an introduction to the different communication domains of the framework, followed by some important ns-3 modules, which I have used for the implementation of my simulation framework. Finally, I explain in a brief overview the the actual implementation status of BloodVoyagerS (BVS).

2.1 Wireless Body Area Networks

Wireless Body Area Networks (WBANs) are an important part of research regarding patient monitoring and remote medical treatments. As an example, this kind of network could consist of sensors attached to the human body to gather data and send it to a device with more computing power to process the received information. Through processing the data, human body values can be evaluated and a physician can be alerted or an actuator also attached to the body can adjust the medication injected into the human body. In this scenario, it becomes clear that enabling safe communication between the devices is important. However, there are critical aspects such as low-power consumption, security, and privacy which need to be considered in this section. Additionally, I will give an overview of possibly useful radio technologies for WBANs concerning the implementation of the communication link between laptop and gateway and explain in Section 2.4 why I chose IEEE 802.15.4 as underlying radio technology.

There are several surveys that give an overview on the requirements of WBANs and technologies which can be used to meet the aforementioned requirements, for example [2], [1], [12], [13], [14] and [15]. Most of the surveys split up WBANs into three tiers, the first one is intra-BAN communication, the second is inter-BAN communication and the third is beyond-BAN communication. I will refer to the explanation by Chen et al. [1] for these three terms. Beyond-BAN communication is in the scope of this thesis negligible, as it describes how the gathered data of

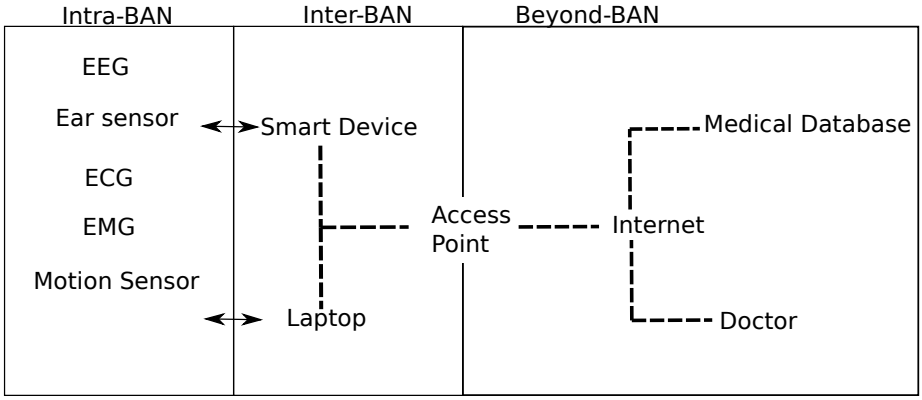


Figure 2.1 – This figure is designed based on the first figure in [1]. It is a simplified version that shows that sensors are able to communicate with devices from the Inter-BAN part and how further communication of the gathered data could look like. The sensors are located on a human body.

the WBAN can be forwarded to a physician or a database through for example the internet. The same holds for inter-BAN communication because in the scope of this thesis it is not important to be able to send gathered data to an access point or something similar. The important part is intra-BAN communication as it focuses on the communication between sensors and the communication between sensors and a personal device such as a smartphone or laptop. As a gateway is directly located on the body most of the characteristics and constraints for sensors regarding communication also hold for the gateway. Sensors and gateway must consist of a power supply, a processor, a memory unit, and a transmitter [2][16].

Ghamari et al. [12], Latré et al. [2] and Cao et al. [14] state some important requirements for WBANs that mainly focus on intra-BAN communication. The low-power consumption of devices is arguably more important than in other networks as overheating of devices could lead to injury or to a rise in temperature [2] which we want to avoid at all cost, because of the application on the human body. Moreover, high power consumption would lead to a short lifetime of the network, as power is mostly supplied by batteries which are not that easy to exchange [2] [12] [14]. An often proposed solution to this could be energy harvesting, which Ghamari et al. [12] explained it as deriving energy from the surrounding environment and transforming it into electrical energy. In [17] a first draft of an energy harvesting system was given with flexible solar panels as the front side of wearable sensors, which could supply the sensors with energy for up to 15 h if the person who wears it stay outside two times a day for 30 to 60 mins. This is a good option when used as an additional power supply but not as the only energy source.

Another aspect which is crucial to meet is the transmission reliability as data that does not arrive in time or that does not arrive at all, could lead in the worst

case to serious injuries or even to death [12]. Providing reliability is mainly the task of the physical layer and the Medium Access Control (MAC) layer. For example, it must be considered how waves propagate in the surrounding of a human body. In addition to reliability, latency in communication must also be handled, which demands a trade-off between the two of them as the need for latency is based on the same reasons as reliability. Khan and Pathan [15] gave an overview of energy-efficient routing protocols to support MAC and Physical Layer (PHY) by providing a good trade-off with a not negligible side effect on latency and reliability. Further requirements are security and privacy in WBAN. Health-related data of a patient should just be available to the patient and their physician. An adversary must not have a chance to change recorded data such that an important change in the body values gets lost [13]. There are other things to consider, but these are the most important to provide stable communication between gateway and laptop.

Khan and Pathan [15], Ghamari et al. [12] and Chen et al. [1] give an overview on the most common radio technologies used for WBAN and some protocols with great potential. Additionally to these surveys, the aforementioned surveys state ZigBee in combination with IEEE 802.15.4, Bluetooth (IEEE 802.15.1), and UWB (IEEE 802.15.6) as the most promising approaches for this kind of network. All of them are short-range wireless technologies and enable communication between small and mobile devices. Today Bluetooth is mainly used for transmission of audio and data streams [14]. The devices using Bluetooth, form a so-called piconet which is a short-range network [18]. Within this piconet, a master device exists which handles the synchronization of all other (worker) devices. Bluetooth operates in the 2.4 GHz ISM frequency band. The Bluetooth Low Energy Standard can provide a 1 Mbps data rate [12]. The standard IEEE 802.15.4 for Wireless Personal Areas has a possible range that exceeds the 2 m needed for WBANs. In addition to the standard which focuses on the PHY and MAC sublayer, ZigBee can add security, network, and application layers [12]. The main purpose of these two protocols is to provide a communication link with devices that need low power for a long battery lifetime [12]. Similar to Bluetooth the devices are split up into masters named Full Function Devices and workers named Reduced Function Devices except that in a ZigBee network more than one Full Function Device can exist. Two physical bands can be used for communication in this standard, the 2.4 GHz band with data rates up to 250 kbps and the 868/916 MHz with data rates of 20/40 kbps. The use of Bluetooth as well as ZigBee is considered for communication between two end-devices [15]. The standard IEEE 802.15.6 [19] which is explicitly modeled for the use in, on and around the body, uses a frequency allocation of 3.1-10.6 GHz. It can cope with multipath fading, offers a large bandwidth for communication between devices, and also supports low power consumption [15]. There are some other technologies reviewed in the surveys of Khan and Pathan [15], Ghamari et al. [12]

and Chen et al. [1], but as Bluetooth, ZigBee and IEEE 802.15.6 were considered technologies for my implementation and also most frequently mentioned in research papers I chose to focus on the explanation of these three.

2.2 Nanobots and Nanonetworks

In the following I will introduce the research field of nanobots and nanonetworks, whereas in Section 2.5 I will explain how nanobots were designed for the simulation framework BloodVoyagerS.

As nanobots and nanonetworks are relatively new and a very diverse research field I will review some definitions on these two terms. Additionally, I will explain the most important aspects and challenges for designing suitable devices for nano inner-body communication.

Büther et al. [20] gives an overview of the terminology and state the difference between nanosensors, nanomachines, nanorobots, nanonodes, and the term nanodevice under which all of them can be summed up. Nanodevices are often described by being very small devices that have a specifically defined function which they should perform in a specific environment. The most interesting term for this thesis is nanorobot since it can be every possible combination of nanosensors, nanomachines, and nanonodes. Like nanosensors they can detect their environment, a nanomachine is defined to be a nanodevice with a predefined task such that they can be an actuator in their environment and a nanorobot has the additional possibility to be reprogrammable such that the predefined task can be changed during its lifetime. To establish a terminology I will use the terms nanorobot and nanodevice interchangeably in the following description of the type of device which is interesting in the context of this thesis.

An important characteristic is the size, as every source is presupposing it for classifying a device as a nanodevice. The scientific literature on the size of the nanobots is in disagreement but most of them are in a range from “tens of nanometers" [21] up to a “few hundreds of nanometers" [22].

Other important characteristics are information processing, power supply, communication, memory, actuating and sensing, locomotion, and internal clocks [20]. Since the goal of this thesis is to implement a simulation framework connecting out-of-body and in-body networks with each other, the most important aspects are the basis of the design of the devices, their movement, and how they can communicate with each other.

Regarding the design of the nanodevices, the literature has two different approaches. In the first place electrical devices which are also called man-made devices[23] require the adaptation of “the construction principle of electronic de-

vices" [20] to the size of a few nanometers. One of the most important findings in this research field were Carbon Nanotubes [23]. They are working within a resonance frequency range of 50 MHz to 5 GHz which means if a radio wave is reaching a nanotube, the nanotube is going to vibrate. Only if this vibration is equal to the resonance frequency, the nanobot can receive a signal through the nanotube [24]. They do not only support communication between nanobots, but they also can be used for sensing their environment [20].

In the second place nanodevices based on a biological design are claimed in both Akyildiz, Jornet, and Pierobon [23] and Stelzner et al. [22]. It was found that cells can also provide a basis for building blocks of nanodevices [22][23]. As stated by Akyildiz et al. [3] cells can use natural components to fulfill the requirements a nanobot needs for being able to communicate with other nanodevices and interact with its environment, e.g. not only transceiver but also control units, memory units and more. Going into detail, e.g., DNA is an interesting natural component for designing bio-nanodevices, as information can be encoded in it. Additionally, it can also be used for building circuit boards at a suitable size for in-body communication [23].

Nanobots are just able to perform small tasks on their own, therefore a nanonetwork is needed to use many nanobots for performing tasks that exceed the sphere of action of a single nanobot [20]. To build these nanonetworks it is important to have a connection among nanobots for communication between them. For inner-body communication among nanobots there are mainly two paradigms proposed by Büther et al. [20], by Stelzner et al. [22] and in some publications by Akyildiz, Brunetti, and Blázquez [25][23][3]. They are called molecular communication and electromagnetic communication. Dressler and Akan [5] consider molecular communication the most promising communication scheme. Molecular communication describes mainly the “transmission and reception of information encoded in molecules" [26] supported by Akyildiz, Brunetti, and Blázquez [25] Suda et al. [27]. On the other hand, electromagnetic communication is building its basis on antennas using for example Carbon Nanotubes to propagate waves. The terahertz range is the only frequency range which enables the nanobots to work efficiently [26].

Another important issue is how nanodevices can be addressed. Due to their mostly short communication range, there are several thousands of nanobots needed to build a suitable communication network in the body [7]. According to Stelzner, Dressler, and Fischer [6],[4], addressing can be done by FCN as IP-addressing won't be feasible for these amounts of nanobots. FCN is focusing on addressing nanobots by their function and location in the body instead of addressing every single nanobot by hand [6].

2.3 Overview of ns-3

In this subsection, I will explain some basic implementations of ns-3 that will be used for my simulation framework. ns-3 is an open-source discrete-event network simulator written in C++ and Python. It is recommended to use ns-3 under Linux and users have to work with the command line, C++, and Python development tools. Unlike other network simulators, this one has no graphical user interface. ns-3 is using the *Waf build system* written in Python to build the ns-3 source code, run, or debug simulations. There are several different ways to download and build ns-3 describe in the ns-3 tutorial.[5]. While C++ is needed to write simulation scripts and modules, they are executed from the command line through using Waf while Python resolves dependencies between specific scripts and ns3 modules. The command for running a script is:

```
./waf --run <name-of-script>
```

ns-3 is modular so that a high reusability of the modules and other C++ libraries is guaranteed by including the modules or libraries at the beginning of the script. At the time of this thesis, most of the existing modules focus on the simulation of Internet Protocols, but other use cases can be simulated as well. For developing ns-3 modules some key abstractions named `Node`, `Channel`, and `Net-Device` are important which are contained in the ns-3 *network module*. A `Node` is the basis for every simulation since it is used to represent devices, but it just represents the device itself and no functionality. Through adding a `Net-Device` to a `Node`, functionality for communicating with other `Nodes` is added to the `Node`. The communication between `Net-Devices` is handled by a `Channel` to which both `Net-Devices` must be connected. It is possible to use multiple `Channels` via multiple `Net-Devices`. The use of packets, headers, trailers, and the tracing system is explained below, as these are essential built-in features for adding content to a packet, removing it, and tracking packet receipt.

2.3.1 Packets, Headers, and Trailers

The implementation of packets has a few guidelines described in the ns-3 documentation.[4] For example, they have been implemented in such a way that it is avoided to change the simulation core if new types of packet headers or trailers are introduced. In addition, attention is paid to efficient memory management, and the possibility

[5] `http://web.archive.org/web/20200601163459/https://www.nsnam.org/docs/release/3.30/tutorial/ns-3-tutorial.pdf`, chapter 1-3

[4] `http://web.archive.org/web/20200325105541/https://www.nsnam.org/docs/release/3.30/models/ns-3-model-library.pdf`, chapter 23

is given to simulate actual application data or dummy data. For the developed simulation framework it is only of interest how the byte buffer works and how the memory management is handled.

The byte buffer stores the serialized content of the headers and trailers that are added to the packet. The intention behind this is to organize packets in a way as close as possible to real network packets. A further interesting aspect is that the memory management of packets is organized automatically. It is "modeled by a virtual buffer of zero-filled bytes for which memory is never allocated unless explicitly requested by the user."[4] If some kind of information should be added or removed from the byte buffer the usage of the `Header` and `Trailer` classes becomes important. This includes layer information as well as application data. Headers and trailers are added to the initially created packet as buffer data.
A newly implemented header or trailer has to derive from the abstract base class `ns3::Header` and implement its four private virtual methods. These methods are called:

- `Print()`
- `GetSerializedSize()`
- `Serialize(Buffer::Iterator start)`
- `Deserialize(Buffer::Iterator start)`

The first one is used to organize what is necessary to be printed to the output stream. The other ones provide functionalities for the serialization and deserialization of data.

To implement an ns-3 header or trailer, a new ns-3 module can be generated in which the implementation of a C++ header file and a C++ source code file has to be stored or they are both added to the ns-3 module `Applications`. To use modules, it is sufficient to include them as a C++ header file in the simulation script. In these simulation scripts the transmission of a packet can be scheduled as well through a method call: `Simulator::ScheduleWithContext()`. I use headers to define potential application data at a smart device and add it to a packet, which then can be transmitted to a gateway. At the gateway, a second header is used to add a sequence number, a tag to decide on the communication direction, and a hop count to the packet.

2.3.2 The Tracing System

The ns-3 tracing system is important to understand as it helps to generate precise output from the implemented simulation. Additionally, it examines the inner functionalities of the simulations since it provides the ability to "discover which significant

events are happening inside the simulation and under which conditions" [28]. There is a simple way to get output from a simulation by using the ns-3 log components. Output can simply be printed on the command line during the simulations. But the log component implementation has one big drawback: it outputs a huge amount of data, as there are pre-defined components[5]. These pre-defined components are used in all ns-3 modules, by using an existing module you will get the predefined output of its log component even though it might not be relevant in your context. Consequently, the data must be post-processed to extract the information needed from the simulation. On the other hand, ns-3 has also pre-implemented abilities to trace the packet exchange such as redirecting output to a .pcap-file or tcpdump [28].

By using the ns-3 tracing system a programmer can generate exactly the output of interest[5]. This system consists of trace sources, trace sinks, and hooks between them. A trace source is an entity that tracks the occurrence of an event happening during a simulation. It signals one or more trace sinks through the use of a function [28], that a specific event has happened and it can additionally provide access to information on the event, e.g., it can provide a pointer to a traced packet or the position of a network node[5]. The trace sink uses the information provided by the trace source to invoke other functions by using the given information as a reaction on the event just happened, or to just output the information or even to get statistical information about the event [28]. A trace source internally holds a list of all trace sinks and generates a point-to-multipoint link according to this list[5]. This is also called a callback mechanism which builds the hook between trace source and trace sink [29]. A trace sink itself has no actual use, it becomes useful when it adds itself as a callback to the list of a specific trace source. If the trace sink is not assigned to a trace source, it won't be able to output information on the simulation. This means the output of the simulation will be more precise on the events one wants to observe during the simulation. Additionally, to not print uninteresting output, using a trace source produces only a "very small execution overhead"[5].

For using the tracing system no new module is needed and callbacks can be implemented in an existing module or a simulation script depending on what events should be observed. The callbacks are currently mainly important for the LR-WPAN connection between the smart device and the gateway. A detailed description of how to implement callbacks can be found in the ns-3 tutorial[5]. They can be used to trace when a packet was received at either of these two devices. If the gateway is the receiving device, the callback is also used to invoke the transmission procedure to the nanobots.

[5] `http://web.archive.org/web/20200601163459/https://www.nsnam.org/docs/release/3.30/tutorial/ns-3-tutorial.pdf`, chapter 7

2.4 Implementation of IEEE 802.15.4-2006 in ns-3

The connection between laptop and gateway is implemented as a Wireless Personal Area Network (WPAN) and relies on the IEEE 802.15.4 standard [10] for LR-WPANs. Since the ns-3 module `LR-WPAN`[6] uses the standard version published in 2006 [10] as basis for its implementation I will refer to it. The goal of this section is to give an overview of the standard and how it is implemented in ns-3 to provide a basic understanding of how the laptop-gateway connection works. I chose this standard for the implementation of this connection because it focuses on low power consumption and it is approved for the communication between two end-devices. Additionally, in ns-3 exists an implementation for this standard while there is no implementation for Bluetooth and IEEE 802.15.6.

The basic idea of WPANs is to build up connections with short distances between the sender and receiver. These distances up to 10 m operate at a low rate such that they are power-efficient and inexpensive but also reliable. By using a low rate only little power is needed, hence devices with no battery or just limited battery consumption can be used. This is important because the gateway will be located on the body and thus must not overheat. Also important for low power consumption is the simplicity and flexibility that this standard provides. Further information on this can be found in the standard [10].

Moreover by using an overview on the PHY and the MAC sublayer [30] and the IEEE 802.15.4 standard [10] a description on an LR-WPAN data transfer is given.

In Figure 2.2[6] you can see the MAC sublayer and the PHY. Initially a packet is given to the MAC layer through an `McpsDataRequest()` at the MAC Common Part Sublayer Service Access Point (MCPS-SAP). You can access a Service Access Point (SAP) by one of four service primitives, one of which is the request. The MCPS-SAP is the MAC sublayer data service and enables the transmission and reception of MAC protocol data units across the PHY data service [10], the MAC header and trailer are added to the packet and it is enqueued for Clear Channel Assessment (CCA). By enqueuing the packet, the MAC Layer Management Entity Service Access Point (MLME-SAP) is sending a `PlmeCcaRequest()` to the Physical Layer Management Entity Service Access Point (PLME-SAP). By receiving a confirmation we know if the channel is accessible. Afterward the MCPS-SAP is sending another request to the PLME-SAP to be able to set the transmission state. Only now we are able to send the `PdDataRequest()`. By this request, we can access the PHY through the PHY Data Service Access Point (PD-SAP). The PD-SAP is the PHY data service, which enables the transmission and reception of PHY protocol data units across the physical medium [10]. At the other device the PHY protocol data units are sent to the

[6] `http://web.archive.org/web/20200325105541/https://www.nsnam.org/docs/release/3.30/models/ns-3-model-library.pdf`, chapter 18

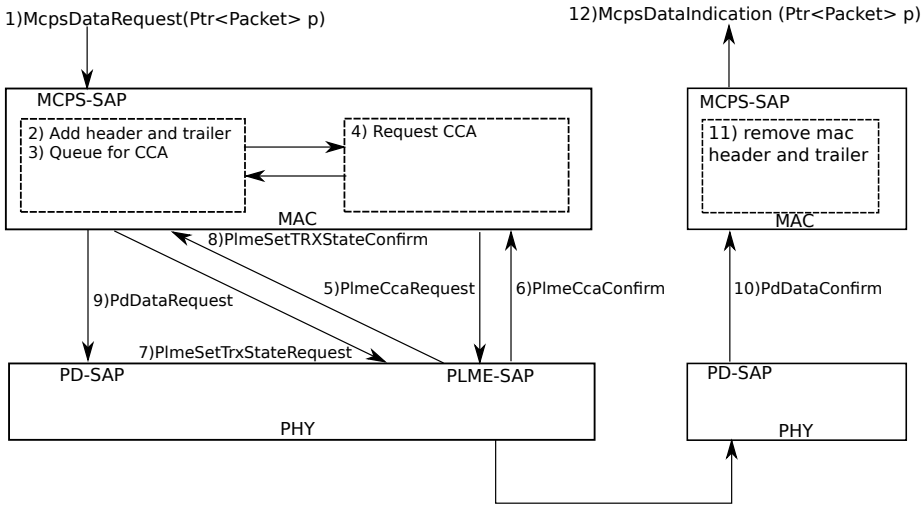


Figure 2.2 – This figure, consisting of PHY and MAC layer, illustrates a simple data transfer between two end-to-end connected LR-WPAN devices. It is designed based on the model from the lr-wpan model library.

MCPS-SAP through a `PdDataConfirm()`-method. At the MAC layer, the header and trailer are removed and they are sent through an `McpsDataIndication()` primitive to a higher layer.

This is the description of a data transfer in ad hoc mode, which is at the moment the only mode supported by the implementation of ns-3 [30]. The features which should be supported on the MAC layer are beacon management, channel access, Guaranteed Time Slots (GTS) management, frame validation, acknowledged frame delivery, as well as association and disassociation as described in section 7 of the IEEE 802.15.4 standard [10] and in Rege and Pecorella [30]. The current implementation of this layer is lacking in support for coordinators, association, disassociation, and beacon management for wake-up functionality [30]. On the PHY the features stated in the standard [10] are activation and deactivation of the radio transceiver, transmission, and reception of packets over the wireless channel, as well as performing additional tasks that may be required by higher layers. The current state of the PHY in ns-3 models the PHY service specifications, the PHY protocol data unit, and personal area network attributes [30]. Additionally, the standard [10] foresees three possible unlicensed bands as communication channels with different modulation schemes. But at present, the 2.4 GHz channel with O-QPSK modulation is the only channel supported in simulation[6].

The LR-WPAN module can be used by including the whole module into the simulation script. A node has to be defined which is equipped with a `LrWpanNetDevice`. To this device a `SpectrumChannel` has to be added and a `Mac16Address` or a

`Mac64Address` for being able to communicate with other `LrWpanNetDevices`. Finally, the position of the node needs to be set for a realistic communication scenario.

2.5 BloodVoyagerS

The simulation framework BloodVoyagerS is mainly consisting of two parts, the physical environment and the movement of the nanodevices within the physical environment namely the circulatory system of a human body [7]. It is implemented in C++ for the “discrete-event network simulator ns-3" [7] as an ns-3 module[3]. The module automatically outputs a .csv-file which tracks the position of every nanobot per time step.

The nanodevices are currently just nodes with fewer characteristics than mentioned in the previous chapter but have a length and a width of 100 nm and additionally, they can be transported through the circulatory system. In Figure 2.3, Figure 2.4 and Figure 2.5 you can see the nanobots which move through the circulatory system during different time steps. They are colored red if they move through an artery and blue if they move through a vein. Additionally, one can see that the nanobots move through all major vessels. This leads to the physical environment of the nanobots.

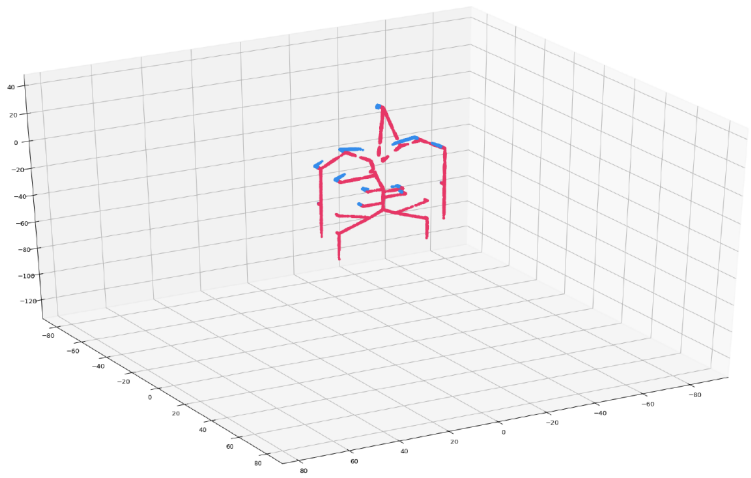


Figure 2.3 – Screenshot from a video plot of BloodVoyagerS

[3] `http://web.archive.org/web/20200325105541/https://www.nsnam.org/docs/release/3.30/models/ns-3-model-library.pdf`, chapter 1

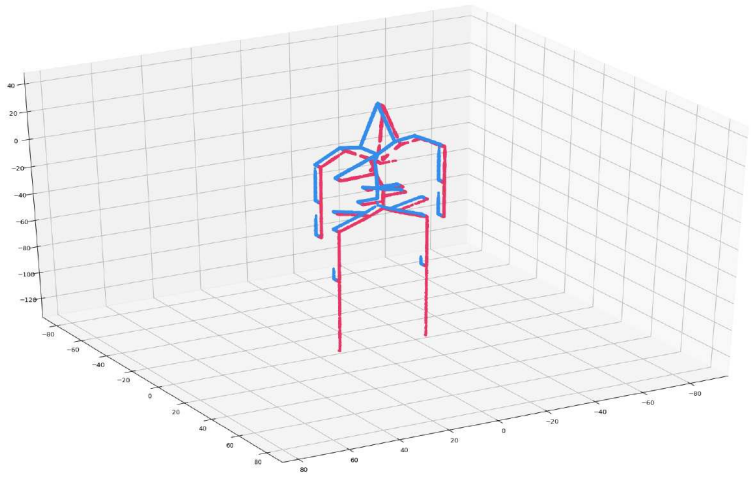


Figure 2.4 – Screenshot from a video plot of BloodVoyagerS

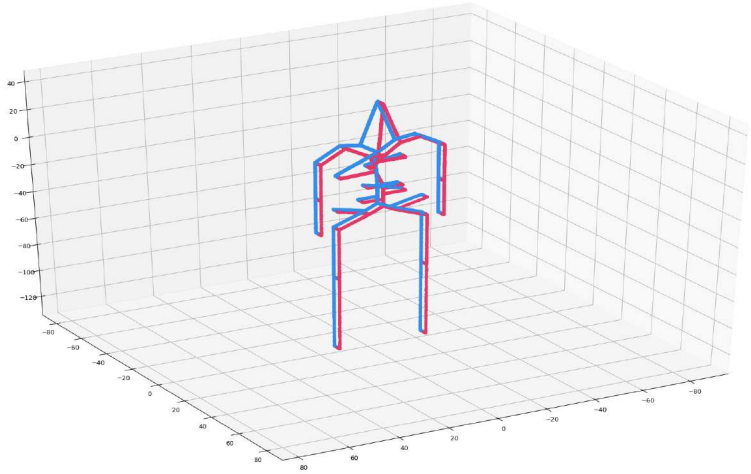


Figure 2.5 – Screenshots from a video plot of BloodVoyagerS

While researching other human body simulations like, e.g., SimVascular [31], Geyer et al. [7] set up their requirements for a simulation framework of the physical environment for medical scenarios. The physical environment is currently in a more prototypical state and does not yet meet all of them but in the following, I will give an overview of these requirements and what the current state of implementation is like. The first of them is a “complete human circulatory system" [7]. The current state of the art for this requirement is an existing circulatory system, but only the major organs, arteries, and veins are implemented at the moment. Additionally, through using anatomical drawings, the positions of the organs are close to the positions in a

real human body. They used a coordinate system where only the z-coordinate is still more abstracted as can be seen in Figure 2.3, Figure 2.4 and Figure 2.5. Secondly, there is a requirement of a “spatial model of the cardiovascular system" [7] for being able to know the position of a nanobot as accurately as possible. This one is met by being able “to track the position of each nanobot in the bloodstream" [7]. The third requirement is that the medium, blood, which is flowing through the circulatory system has to be represented somehow in the simulation framework. This is a requirement that is not integrated into the current state of the art because for this a molecular database is needed. A molecular database provides information on the molecular conditions within a bloodstream. Last, of all, there is a requirement of simulating that the circulatory system can “push floating particles around with its flow" [7]. In the simulation framework, the nanobots are moved by the circulatory system but more in an abstract way than one describing the realistic movement of particles. This is done by simulating a constant velocity for each type of vessel which is of course not the case in a human body. In conclusion, BVS is a raw model of the circulatory system but an appropriate approach to be useful for simulating the physical environment in my simulation framework.

To include BVS to ns-3, the `BVS`-folder simply needs to be copied to the `src`-folder of ns-3, because BVS a ns-3 module. If the ns-3 source code is built again, the `BVS`-module can be used like every other ns-3 module. By calling
`Bloodcircuit:BeginnSimulation()`
in the simulation script, BVS is started. The start-call of the BVS simulator needs input parameters that define the simulation duration, the number of nanobots one wants to simulate, and the blood vessel in which the nanobots should be injected at the beginning of the simulation. Otherwise, if only the BVS simulation is needed, it can be started on the command line using a pre-defined simulation script which is included in the `BVS`-folder:

```
./waf --run "start-blood-voyager-s"
   or
./waf --run "start-blood-voyager-s --simulationDuration=<n>
--numOfNanobots=<i> --injectionVessel=<j>"
```

In the first version, the default values of BVS will be used in the simulation run. In the second version the number of nanobots, the simulation duration and the vessel in which the nanobots should be injected can be defined. The injection vessels and their IDs can be found in the appendix of the BVS-Github-Repository[10]. In the simulation output file, namely the .csv-file, information can be found on the position of every nanobot at any time step of the simulation. The column order is the nanobot

[10] `https://github.com/ReginеWendt/blood-voyager-s`

ID, the x-Coordinate, the y-Coordinate, the z-Coordinate, the simulation time in nanoseconds, the vessel ID in which the nanobot is, and the stream ID in which the nanobot is.

On top of that, Wendt, formerly known as Geyer, Deter, and Fischer [32] developed a visualization tool[11] to process the resulting .csv-files from BVS. This tool creates a 3-dimensional animation of the nanobot movement throughout different time steps. The body is depicted as if a human being is standing in front of the viewer and looking at them, which means that what is on the right side for the user, is showing the left part of the body. The simulated person is facing into positive Z direction. Throughout the evaluation of this thesis, this tool becomes necessary to interpret the results more precisely.

[11] `https://github.com/RegineWendt/BVS-Vis`

Chapter 3

Implementation

Within this chapter I will explain the implementation of a connection between a smart device and a gateway, the gateway functionality provided by the implemented framework and the proximity communication links.

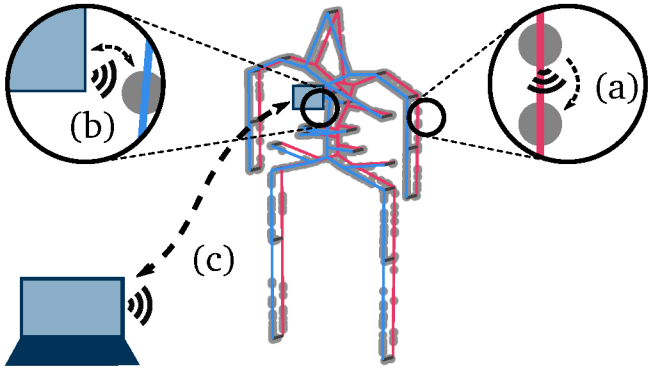


Figure 3.1 – This model is taken from our submitted paper [11]. Communication paths between: (a) different nanobots; (b) nanobots and gateway, e.g., via ultrasonic communication; and (c) gateway and smart device via an IEEE 802.15.4 network.

In Figure 3.1 the structure of the framework is visualized consisting of the proximity implementation for abstract communication among nanobots, the proximity implementation between gateway and nanobots, and the Laptop-Gateway-Connection. I will also give an overview on the functionality at the gateway, which is able to process information received by a laptop such that it can be used to address the nanobots. Inversely, data from the inside of the body can be sent to a laptop via the same connections.

3.1 LR-WPAN connecting Laptop with Gateway

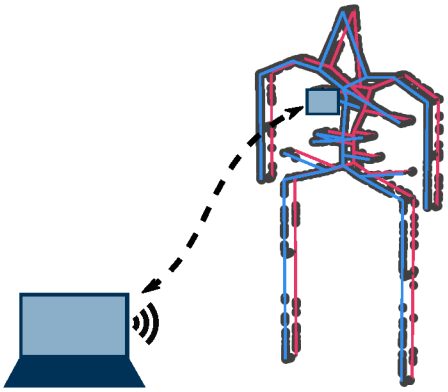


Figure 3.2 – Connection between smart device and gateway via IEEE 802.15.4 network

A laptop can provide an application for a doctor such that the doctor is able to control the movement and tasks of the nanobots on the inside of the body. This gives them the ability to treat problems on the inside of the body more precisely and thus can enhance therapies and medical treatments. But a laptop or smart device is not able to communicate with the nanobots per Wi-Fi because most of the electromagnetic waves would be attenuated by the body mainly consisting of water. As the gateway will be able to communicate with the nanobots by using ultrasound communication, the gateway can also be equipped with a radio frequency transceiver such that it can also communicate by using LR-WPANs.

In Chapter 2 I already described the existing IEEE 802.15.4 standard and its implementation in ns-3. Building upon this explanation I implemented a point-to-point link between a laptop and the gateway which is shown in Figure 3.2. I have initialized the PHY and MAC sublayer for both devices and also set their position, so that it is as realistic as possible for the aforementioned scenario. BVS introduced a coordinate system with the point of origin located in the left half of the heart. The position of the gateway is (0,0,2) directly above the left half of the heart on the chest and the position of the laptop is 5 m apart from the gateway on the z-axis. Afterwards, one is able to send packets over this link.

As I already described packets and their implementation in Chapter 2 I will now describe how I used them and how I implemented the application layer.

A network layer is not needed at the moment, because only an end-to-end data transfer between two devices is used for the framework. It has to be considered later if also sensors located on the body are used.

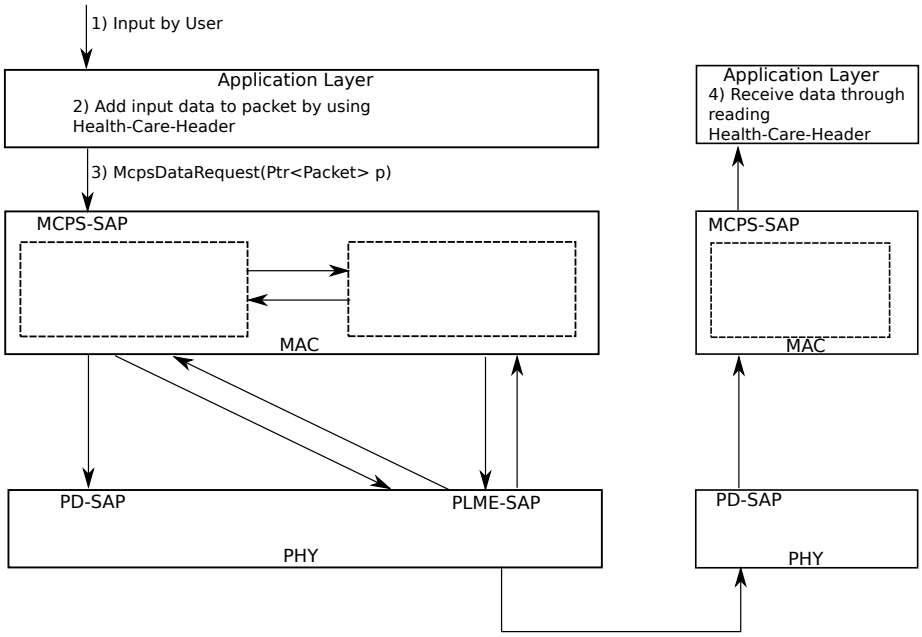


Figure 3.3 – This figure shows the application layer added to the lr-wpan data transfer. It is designed based on the model from the lr-wpan documentation.

As you can only add data to a packet by using a header class, I implemented the application layer(see Figure 3.3 [6]) on a laptop and the gateway as a class called `HealthCareHeader`. The goal is to provide options for the doctor to set a range of nanobots they want to address, a function by which they want to address or the kind of addressing, e.g., unicast. Sometimes they want to address the area where the nanobots currently are located in the body and want to be able to give them a task to accomplish. To be able to set these input variables one can use the constructor, or if only some of these variables should be set, the setter methods for each variable(see '1)' in Figure 3.3). Since the `HealthCareHeader` has to inherit from the `Header` class of ns-3, functions provided by the base class have to be overwritten.

The `Print()` method is important for tracing the data sent through this connection. The `Serialize()` method defines the order in which the data should be added to the packet; In this case these are all the variables the doctor can use to address or give the nanobots a task to proceed. The `GetSerializedSize()` method returns the amount of bytes needed to represent the variables. It is needed to store a header in the byte buffer of a packet by adding it with the ns-3 method `Packet::AddHeader()`[4](see '2)' in Figure 3.3). The `Deserialize()`-method reads the data from the packet in the same order it was added to the packet. By adding such a header to the packet, the data which is needed to be processed at the gateway can be sent from a laptop to a gateway over the LR-WPAN link.

To start the LR-WPAN transmission procedure the method `McpsDataRequest()` (see '3)' in Figure 3.3) has to be invoked on the packet. At a receiver the data can either be extracted or further processed(see '4)' in Figure 3.3).

Since this is a simulation which needs input parameters I make use of another function of ns-3 that is able to read command line values which are passed at program start. By starting the simulation through the command line, default values are already set for the parameters as mentioned in Table 4.1.

parameter	description	default value
startRangeNanobotId	lowest value of nanobot id range	0
endRangeNanobotId	highest value of nanobot id range	100
nanobotFunction	function ID a nanobot has to provide for the task	0
bodyArea	ID of target vessel	2
task	task nanobots should accomplish	2

Table 3.1 – Simulation Input Parameter – Addressing Nanobots

3.2 Gateway Functionality

In contrast to the smart device a gateway has some additional functionalities to be able to also exchange packets with nanobots. So besides the LR-WPAN netdevice provided by ns-3 there are some special methods which make it possible to copy a message to a nanobot. How the message is copied is treated in Section 3.3. I will now only explain what happens at the gateway when a message is received. If the packet was received through the LR-WPAN link, there is a functionality provided by the LR-WPAN module, called `McpsDataIndicationCallback`. This relies on the ns-3 tracing system explained in Section 2.3. By receiving this packet through the MAC sublayer, a function at the gateway is triggered that takes the packet and adds a second header to the packet, called `GatewayToNanobotHeader`. This header adds a sequence number, a tag and a hop count to the packet before setting a flag, which shows the nanobots that a packet is ready to be copied by them at the gateway. The sequence number is implemented as a counter which is increasing with every time a new `GatewayToNanobotHeader` is added to a packet. This is not done by using the constructor, as copying the message would also cause the sequence number to increase. There is a special function `SetSeqNr()` that has to be called to set the sequence number. The tag is just set to true or false depending on whether the packet is addressed to nanobots. At the gateway the hop count is always set to zero.

The gateway can also receive messages from nanobots. The description on how this is implemented can be found in Section 3.3. Upon receiving a packet from

nanobots, the gateway first checks the `GatewayToNanobotHeader` and removes it. It just uses the sequence number for examination, because the tag will already be evaluated by nanobots. If the sequence number did not increase, the message will not be sent to the smart device.

3.3 Proximity-Approach

The crucial in-body part of the framework is highly abstracted for now. Instead of implementing molecular or terahertz communication like suggested in most of the research publications on nanonetworks and nanocommunication, I assumed that there is a possibility how nanobots are able to exchange information among each other. This assumption includes that if a nanobot is in the communication range of another nanobot, it is able to simply copy the packet which is provided by the other nanobot. There currently is no channel model, physical layer or medium access control.

Within the framework a function call is used to additionally start the BVS framework which I described in Section 2.5. Since the BVS framework already includes nanobots, most of the nanobot communication is implemented in the nanobot class provided by BVS and causes the frameworks to merge, so they cannot be used independently.

In BVS a movement routine is scheduled in every simulation second for every nanobot. Additionally to that, I scheduled a message routine in every simulation second. This message routine relies on the positions of the nanobots, so the positions are saved in a map provided by the programming language C++. A map is a combination of key value pairs: the key is the blood vessel a nanobot is currently in and value is a pointer to the specific nanobot. I extended BVS to update the map every time a nanobot move. By using this map all nanobots can be found within the message routine. For every nanobot in the direct vicinity it is checked if its new received packet is newer than the last received packet. If this is the case it also disassembles the packet and tries to copy its message to every other nanobot in its vicinity.

If a nanobot is in the left heart chamber, which can be checked through the map on nanobot position information, it can check whether the gateway has received a packet by the smart device. Every nanobot with the a position in the left heart vessel can then copy the packet and set a flag at the gateway, so the gateway knows that at least one nanobot received the packet. After the message routine is completed in the simulation second, the flag can be reset if it was set. Additionally, the flag showing that the gateway has received a packet, can be reset. After the nanobot has copied the message, it modifies the hop of the packet by adding one

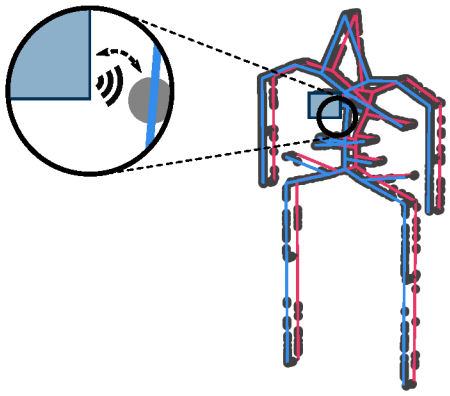


Figure 3.4 – The connection between gateway and nanobots. By using a proximity approach nanobots can only exchange information with a gateway when they are located in the left half of the heart.

hop. Nanobots copy the message from the gateway by using the in ns-3 built-in method `ns3::Packet::Copy()`. This method "returns [a] packet which behaves like an independent copy of the original packet, even though they both share the same datasets internally"[7].

Afterwards Nanobots can check if the newly received packet is newer than the one they received before. If they haven't received a packet before, it is the newest packet and they can perform disassembling or copying to other nanobots on the packet. The same applies to the case when the sequence number of the new packet is newer than the one of the packet received before. If both sequence numbers are equal, or the sequence number of the new packet is even less, it is not new and the bot will not react to it. The sequence number only shows the actuality of a packet.

The reaction to a new packet by a nanobot is to find out whether it is addressed to nanobots or not. If so, it gets disassembled. The first part of disassembling the packet is removing the header which was added at the gateway and includes the sequence number and tag which shows whether the packet was for the nanobots or the gateway. Afterwards we get the `HealthCareHeader` from the packet and check if the nanobot ID, function ID and body area of the nanobot fits the ones claimed in the packet. If so, they can decide which task they have to accomplish depending on the task claimed in the `HealthCareHeader`. These tasks are currently implemented as three different `RandomTasks()`. Within the random task the nanobot is forced to wait for a defined amount of time which is different for each task. After the execution of a task the nanobot creates a new packet, which is meant to be an acknowledgment

[7] `http://web.archive.org/web/20200607231516/https://www.nsnam.org/docs/release/3.30/doxygen/classns3_1_1_packet.html#afb38be706cfc761bc1c0591f595fc1b7`

packet that should be sent out of the body. This packet is set as the new packet of the nanobot. It will be handled as a new packet in the next simulation step. Therefore, the methods before explained will be applied to it and it will get spread in the human body.

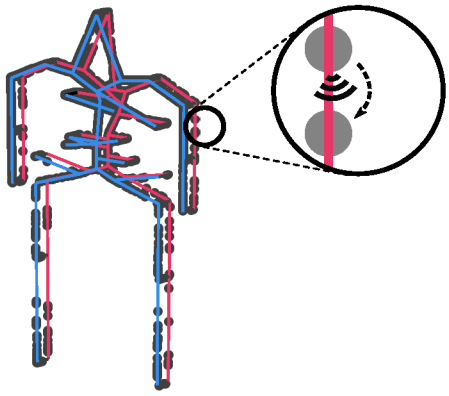


Figure 3.5 – The connection between nanobots. By using a proximity approach they can only exchange information among each other while being in the communication range of each other, which is 1 cm for every nanobot.

The most important and not uncomplex task is how to handle packet copying between nanobots. Sending a packet is not dependent on whether a new packet was received in this simulation second. It is dependent on whether there is a received packet from rounds before and if it was sent before. If it was already sent to another nanobot in a simulation second before, the packet will not be sent again to prevent the network from being flooded. But if there is a packet and it was not sent until now, a nanobot starts searching for other nanobots in a range of 1 cm on every of the three axes. If there is one that hasn't received a packet until now or the sequence number of the other bot is lower than its own, it can copy the packet to the other bot and can tag the packet as sent. The tag is being reset during the comparison of the new packet and the last one received. Again, the hop count of the packet in the header increases by one at the nanobot to which the packet was copied to.

If nanobots created their acknowledgment packet it will be sent around in the body by the nanobots. When the packet is received by a nanobot, which currently is in the left heart chamber and thus close to the gateway, the packet can be copied to the gateway. The method in which it is checked whether a nanobot is in the left heart chamber is extended by an additional examination, which checks the `GatewayToNanbotHeader` of the last received packet of a nanobot. It checks whether the tag for nanobots is set or not. If not, the gateway can copy the packet to itself, otherwise nanobots will keep the previously mentioned handling of the packets. This

additional examination is implemented such that a nanobot can first copy its packet to the gateway and afterwards copy a message from the gateway. Consequently none of both packets can get lost while the exchange of packets between gateway and nanobots.

3.4 Configuration

The before described `HealthCareHeader` and `GatewayToNanobotHeader` can be found in the module `Health-Care-System`. Additionally to the headers, a `laptop`-class and a `gateway`-class are implemented within this module. Both of them are implemented as ns-3 `Nodes` with a `LrWpanNetDevice` attached to them as explained in Section 2.4. Both, `laptop` and `gateway`-class are using callbacks on the MAC sublayer level, to trace the packet transmission and reception. Within the callback, the laptop outputs the size of a received packet, whereas the callback of the gateway triggers a method to set a flag for the nanobots as explained in Section 3.3. Furthermore, the `gateway` includes a method with which a message can be copied from the nanobots. To avoid that both classes have to be initialized separately, there is a helper class called `laptop-gateway-connection` which initializes both through a constructor call of this class. Within this class the `channel` including `PropagationLossModel` and `PropagationDelayModel` is defined for the LR-WPAN connection. The constructor call provides a position, the communication channel, and a MAC address for both laptop and gateway. In a simulation script, only a constructor call of the `laptop-gateway-connection`-class, the addition of headers to packets, the scheduling of packets, and the function call `Bloodcircuit::BeginnSimulation()` is needed to start the simulation. This makes it relatively easy to use the two modules `Health-Care-System` and `BVS`.

Chapter 4

Evaluation

Since the simulation framework is a completely new development except the usage of BVS and no system has been developed that is able to execute the simulation scenario in reality yet, there are no exact values against which I can evaluate the system. In this chapter, the current parameters of the simulation system are therefore evaluated to obtain first results, which may lead to important conclusions and new research ideas and strategies. Reliability in medical application context is one of the most important aspects, e.g., if latency or packet loss is to high the system won't have a use for physicians. Hence, one of the goals is to find out what effects the distribution of nanobots in the human body has on the reliability of the system, since Geyer et al. [7] mentioned that nanobots are not uniformly spread in the whole body. Apart from that, the objective is to give a comprehensive parameter study, e.g., how many hops are needed on average to reach a nanobot.

4.1 Parameter & Metrics

The scenario I simulated was used to mainly determine the latency of the created system, since this is one of the crucial aspects. This is done by simulating the transmission of a packet by a smart device. The packet gets received by the gateway, forwarding the packet to the nanobots. In this scenario nanobots cannot create acknowledgment packets or other packets, they can only exchange the packets they receive from outside the body. The goal is to find out if messages from outside the body can be received by all nanobots and if so, how long does it take. In the following the amount of nanobots used will also be described by using the word population, where 1000 bots are a small population and 6500 bots are a large population. Two positions are considered for locating the gateway on the body, the first one is on the chest above the left half of the heart and the second one is on the left forearm, since it is more likely to integrate the gateway functionality into a smart watch.

Furthermore it is important to be able to detect and eliminate problems within all vessels in the body. Since nanobots have to be in the vessel of interest during reception of a packet, the amount of vessels that can be reached during the first reception of a message is important.

The distribution of nanobots is a parameter which should be considered during the evaluation, but it was not influenced by the simulation scenarios. All scenarios ran with the same random seed for nanobot movement. By varying the time stamps at which the packet gets sent during the simulation, different distributions of nanobots are considered. The 7 minute time stamp was chosen because of Geyer et al. [7] mentioning: “the nanobots achieve a dynamic equilibrium after about 7 min". To explain the impacts of the distribution of nanobot on my results I will also rely on screenshots of the visulization tool BloodVoyagerS-Visualizer (BVS-VIS) implemented by Wendt, Deter, and Fischer [32] formerly known as Geyer.

In Table 4.1 an overview on the simulation parameter can be found. Afterwards the metrics for evaluating the system are described. The first three parameter of the Table 4.1 which were explained in Section 2.5, provide the opportunity to set the amount of nanobots, the simulation duration and the vessel in which the nanobots should be injected. The last two parameters are resulting from the introduced simulation framework and describe the scheduling of a packet transmission by the smart device and the position of the gateway. When a message was scheduled at 1 min the simulation duration was set to 300 seconds, when it was scheduled at 4 mins the duration was 600 seconds and when the message was scheduled at 7 mins the duration was 800 seconds. For each of the remaining combinations a simulation run was done.

parameter	description	default value
simDuration	simulation duration in seconds	[300, 600, 800]
numOfNanobots (population)	amount of nanobots	[small (1000), large (6500)]
injectionVessel	ID of injection vessel [1-94]	1
time stamps	simulation time stamp at which a packet is scheduled in minutes	[1 min, 4 min, 7 min]
positionGateway	position of the gateway	[left half of heart, left forearm]

Table 4.1 – Simulation parameters

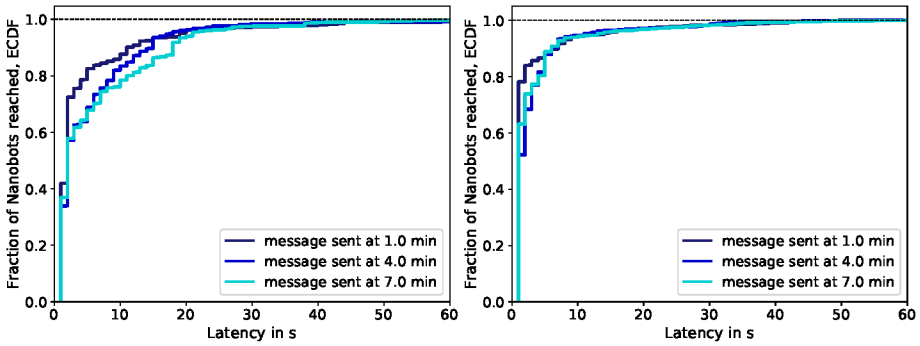
Latency describes how long it takes a packet to reach a fraction of nanobots. It should be noted that the latencies do not correspond to real values but can only represent trends since nanobots can only execute their routines once per simulated second. The routines are scheduled once per second since Geyer et al. [7] scheduled their movement routine in the same way. As before mentioned latency is a crucial part of medical applications. A system has no use for physicians if the bots are not controllable on a real-time basis. In the worst-case latencies can be decisive when it comes to life-threatening situations.

Hop Count explains how many intermediate nodes were used until the packet was received. By evaluating this metric, we can define a potential Time to Live for packets, so the network won't get flooded by packets. Besides hops will become relevant in later cases, when a processing delay is considered at each nanobot for every packet.

Fraction of Vessels is a metric that addresses the need to reach all blood vessels in a human body to provide a system that can eliminate problems in any part of the body.

4.2 Results

4.2.1 Latency



(a) 1000 bots **(b)** 6500 bots

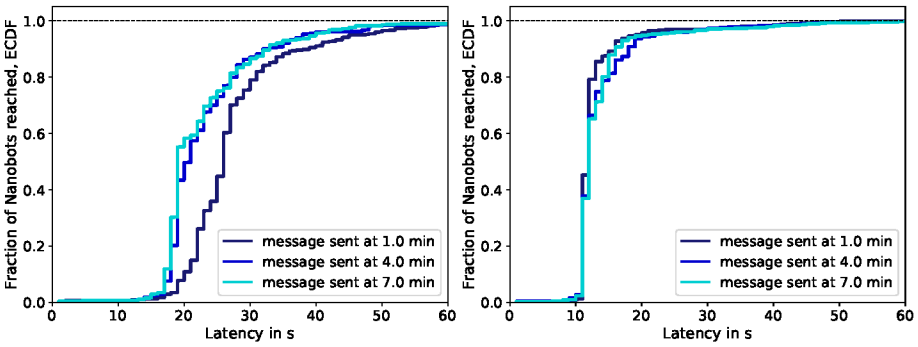
Figure 4.1 – Latency of packet reception at nanobots with variation in amount of nanobots and transmission timestamp. The nanobots can try to send their packet until they have copied their packet to at least one other nanobot

In a first evaluation, I focused on different combinations of nanobot population and time stamps the message was sent by the smart device. The gateway was positioned on the left chest and nanobots try to copy the received message until they copied it to at least one other nanobot. Nanobots get injected into the body at the beginning of the simulation, i.e., at the times when messages are sent, all nanobots are already in the bloodstream. In Figure 4.1 a first trend can be observed.

In every case the first nanobot receives the message after one second, independent of the chosen nanobot population and when the message was sent. At first sight, one second seems to be high latency, but the message routine of the bots is scheduled to be executed once per simulation second and therefore one second is the minimum value that can be achieved.

If we use a large population the fraction of nanobots that received the message is increasing faster to 90 percent than by using a small population. Whereas the timing of a message seems to have no predictable but still a considerable influence on the latency metric. In comparison, the 90 percent mark is reached after 6 seconds if the message is sent at 4 or 7 mins or after 7 seconds if the message is sent at 1 min for 6500 bots. But it takes 11 seconds for a message sent at 1 min, 15 seconds for a message sent at 4 mins and 18 for a message sent at 7 mins during the usage of the small population. Even in the 6500 bot case, the latency is considerably high, but still, this is due to the fact, that nanobots copy their message once per second. These

values should be observed when replacing the abstract proximity message exchange with other communications channels, e.g., ultrasound and molecular communication. Another important result noticeable in Figure 4.1 is the so-called heavy tail until the whole nanobot population received the packet. While the small population has latency values of 145 seconds, 117 seconds, and 65 seconds for 7, 4, and 1 minute, the values for the large population are 63, 63, and 100 seconds until the whole number of nanobots received the packet. Since the latency is crucial in life-threatening situations and physicians need a system that reacts on a real-time basis, these values are exceptionally bad.



(a) 1000 bots **(b)** 6500 bots

Figure 4.2 – Latency of packet reception at nanobots when setting the position of the gateway on the left forearm with variation in amount of nanobots and transmission timestamp

In a second scenario, only the position of the gateway changed to the left forearm. It is evident that the results discussed in the first scenario are further supported by this second scenario.

The first bots receive the message again after one second regardless of the combination of the number of bots and when the message was sent. This can also be traced back to the aforementioned reason that bots can only send once per second. In the 6500 bot case the 90 percent mark is reached after 16 seconds for 1 and 7 minutes, and after 18 seconds for 4 minutes. Whereas in Figure 4.2a the increase until 90 percent of all bots received the message takes 39 seconds for one minute and 33 seconds for 4 and 7 minutes. The latencies for this scenario are even worse than for the scenario before and are far from being a system that works on a real-time basis.

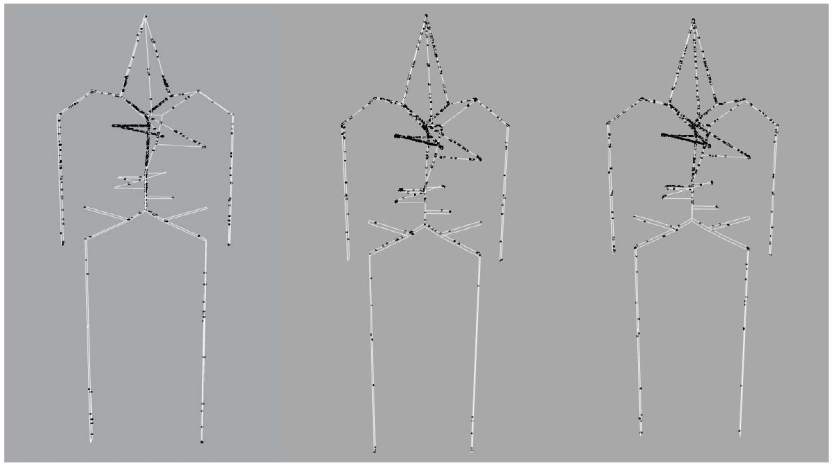
Again, in both cases the heavy tail until the whole nanobot population received the packet is conspicuous. The latencies for the packet to be received by all bots increases to 73 second for messages sent at 1 and 4 minutes and 72 seconds for the message sent at 7 minutes during the use of the large population. In contrast,

the increase during the use of the small population is significantly higher with 152 seconds for 1 minute, 192 seconds for 4 minutes, and 200 seconds for 7 minutes. These are values that would not be sustainable in such a system.

A comparison between the two positions of the gateway also shows that the position of the gateway is crucial for latency values. The strong increase in nanobots receiving the message is delayed in the case of a smartwatch gateway and 6500 bots by 11 seconds. For the small population, the delays vary more but in the best case, the strong increase is delayed by 17 seconds in comparison to a gateway located on the chest.

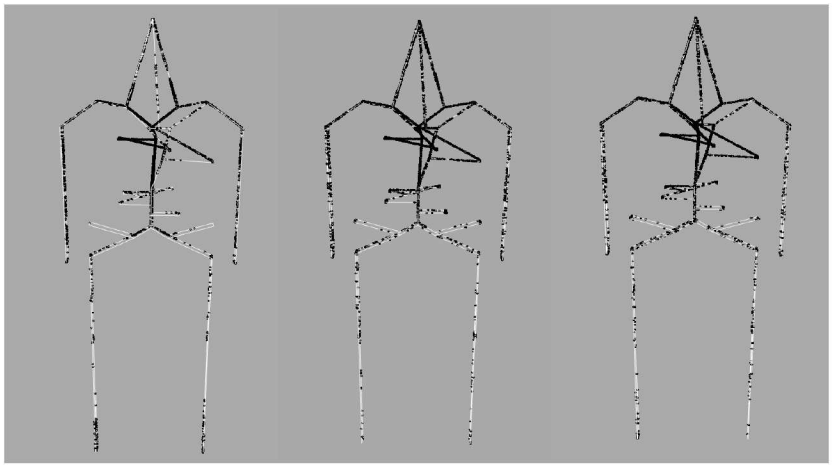
To get a better understanding of where the heavy tail for both gateway positions and the high latency comes from, the distribution of nanobots is an interesting aspect to examine. By looking at the screenshots of BVS-VIS in Figure 4.3 and Figure 4.4, the distribution of the nanobots in the body can be seen at the time the packets were sent. The previously observed behavior in the plots gets explained in more detail by this. In both cases, most of the nanobots are in the upper body. It explains why the packet first spreads within a few seconds between the bots in both cases and afterward it takes a relatively long time to reach the remaining bots. If there are just a few bots or even none in one vessel the packet forwarding is interrupted because the communication range of 1 cm will not suffice. In this case, a nanobot has to first travel into the communication range of other nanobots which already received the packet. This explains the large latencies until the full population of nanobots received the message. In the 6500-nanobot case, more than half of all bots are in the upper body at each of the different time stamps, which explains the fast increase of the fraction of bots which received the packet in Figure 4.1b. During the comparison of the different gateway positions, the strong increase in Figure 4.1 and the relatively flat increase in Figure 4.2 for the packet sent at 1 minute stands out. Since the movement of the nanobots does not differ between the two different scenarios due to the same random seed, in both cases most bots are located in the upper body and not as distributed as in the 7 minutes version. In the case of a gateway positioned on the chest most bots are nearby and can directly copy the message or receive the message by another bot.

To explain the high latency for a gateway positioned on the left forearm, a few screenshots with focus on the left forearm were made (see Figure 4.5 and Figure 4.6). The fact that the nanobots which copied the message from the gateway, need to first move back into the upper body where most of the bots are during the whole simulation, has a significant influence on the latency. With a small population, there are only a few bots at all in the left arm independent of the time stamp. By using a large population, nanobots are mainly in the veins after 60 simulation seconds and later relatively well distributed. The bots in the left arm have to transport the packet into the upper body where it can be more extensively spread among the devices. By



(a) 1 min (b) 4 min (c) 7 min

Figure 4.3 – Distribution of 1000 nanobots at different timestamps – Screenshot taken from BVS-VIS



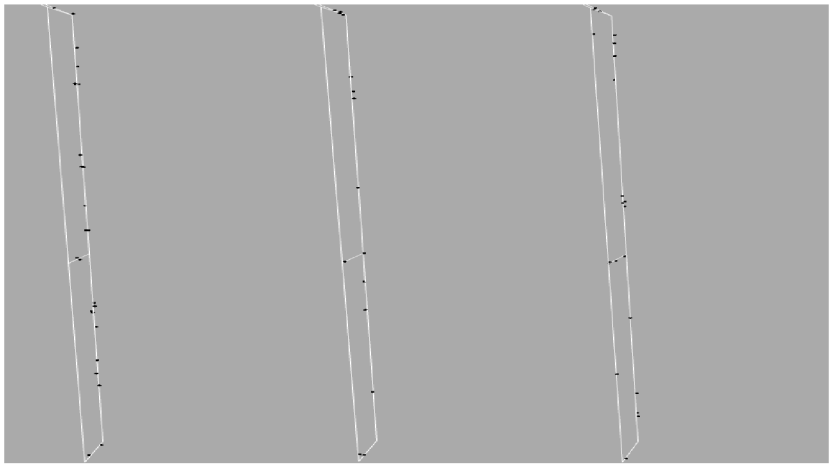
(a) 1 min (b) 4 min (c) 7 min

Figure 4.4 – Distribution of 6500 nanobots at different timestamps – Screenshot taken from BVS-VIS

comparing the usage of 1000 and 6500 nanobots, significantly fewer are present in the left arm. If there are fewer bots in the communication range of each other it takes more time to bring the message into the upper body. This in turn leads to an increase in latency.

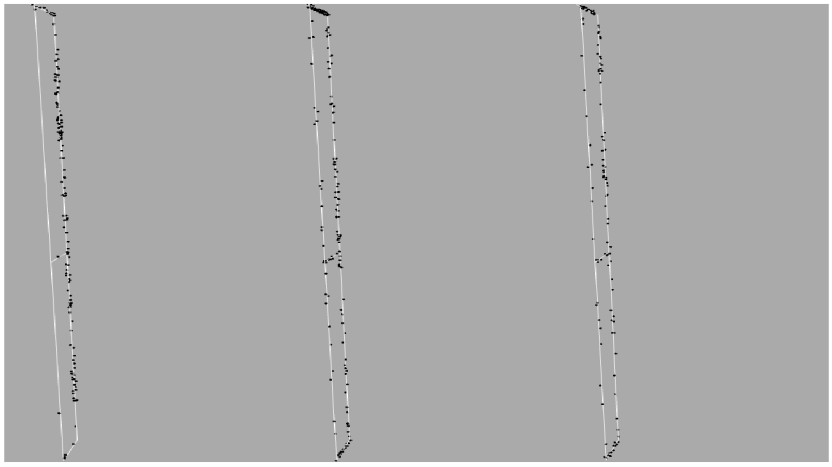
However, a result that can be drawn from this, is that the gateway should rather be placed in a location that is close to many bots so that the message spreads faster. Furthermore, the latency has to be considered when further implementing the system.

It is of crucial importance to reduce the latency values due to the medical field of application.



(a) 1 min **(b)** 4 min **(c)** 7 min

Figure 4.5 – Distribution of 1000 nanobots at different timestamps – Screenshot taken from BVS-VIS – Left Arm



(a) 1 min **(b)** 4 min **(c)** 7 min

Figure 4.6 – Distribution of 6500 nanobots at different timestamps – Screenshot taken from BVS-VIS – Left Arm

4.2.2 Fraction of Vessels

Since the screenshots are just a snapshot of one explicit time stamp another interesting information needed is to find out where nanobots are located when they first receive the packet. In this framework it is assumed that nanobots can only execute a task if they are at the position where the task must be executed at the time of an incoming packet.

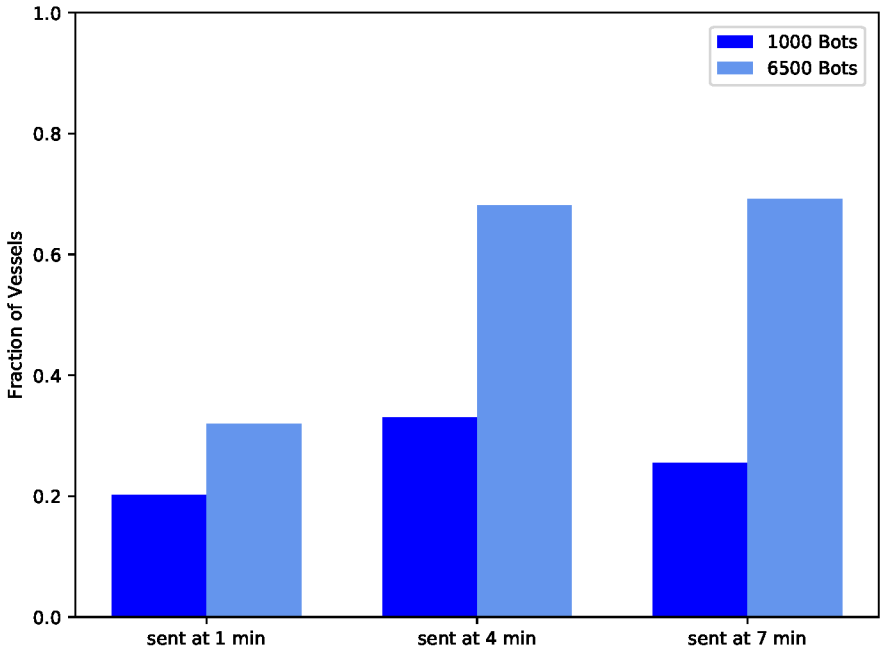


Figure 4.7 – Amount of vessels in which bots first received the message in percentage

Once more the bot population has a noteworthy influence on the fraction of vessels that can be reached by the system. In no simulation scenario, was it possible to reach the entire amount of blood vessels. Anyhow by using the large population the number of vessels that can be reached was once under 50 percent with 30 out of 94 vessels, when the message was sent at one minute. When sending the message at 4 and 7 minutes, 64 and 65 vessels out of 94 can be reached. The small population does not even attain half of all vessels independent of where the gateway was located or when a message was sent. In the best case, 31 vessels were attained, when the packet was transmitted at 4 minutes.

The position of the gateway seems to have an influence on the fraction of vessels reached, but cannot be explained yet and needs further investigation. Figure 4.8

shows that the amount of vessels that can be reached does not vary as much as in Figure 4.7. This is a desirable effect since the goal is to build a reliable system.

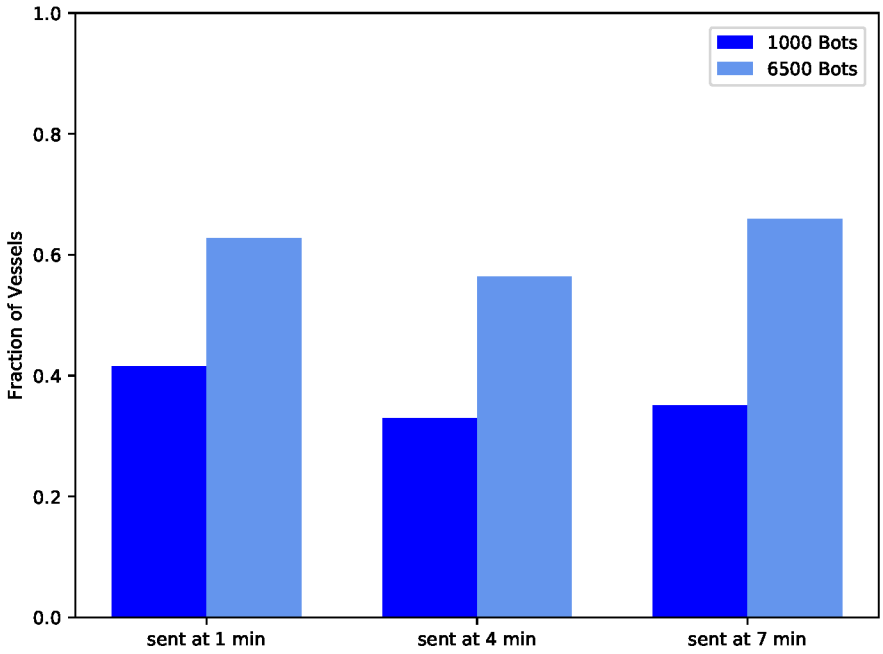


Figure 4.8 – Fraction of vessels in which bots first received the message in percentage – Gateway at left forearm

First of all, the influence of when a message was sent is not yet explainable, which causes that the values presented for the attainable vessels can't be considered as representable. Nevertheless, by using this metric the importance of the bot population used and their distribution is again highlighted. On the other hand these results need further investigations on the effect of the gateway position as well as the amount of nanobots. It would be interesting to find out whether there are gateway positions that enable to access nanobots in a greater fraction of vessels from the outside or if using more than one gateway has an influence on this metric. During the usage of a small amount of bots there won't be nanobots at all in some vessels (see Figure 4.3), what can cause problems in providing a reliable system. So another important question for both latency and the fraction of vessels reached by a transmission is the amount of bots needed to provide reliability or the ability to access nanobots in all vessels.

4.2.3 Hop Count

The distribution of nanobots and therefore also the amount of nanobots have a decisive influence on the hop count metric. By using more nanobots the hop count increases, which will in later simulations with processing overhead at nanobots cause higher latencies and therefore increase the heavy tail of the system. A large population results in a larger distribution which in turn will result in a higher hop count. While in the 1000 bot case for a gateway positioned on the left chest, a mean value of 125 hops was observed when sending the message at 1 minute, the hop count for the large population was 395. In the scenarios when the messages were sent at 4 and 7 minutes the hop count the difference in the mean value is 115 hops and 101 hop more during the usage of the large population. The high hop count emerges from more nanobots being in the communication range of each other. Therefore, the communication chains among the bots do not break off as quickly as with the small population.

Further, the position of the gateway has a strong influence on the mean amount of hop counts. While the average hop count for 1000 bots is mostly around 55, with 47, 59 and 64, the large population causes hop counts of 466, 384 and 356 on average for the gateway positioned on the left forearm. The cause for the smaller standard deviation (see Figure 4.10) may be caused by the small amount of bots first receiving the message. Therefore at the beginning only few bots will have few hops and when the message attains the location of the upper body, the amount of nanobots receiving the message will increase. If the gateway is positioned on the left chest in the beginning of the simulation most bots will receive the message with few hops and the bots in the heavy tail will have relatively high hop counts.

Again these values are not representative since the influence of the message time stamps can't be determined. A high hop count can lead to a higher error rate, which is at the moment not considered in this simulation system but will be relevant when using terahertz communication or molecular communication. Especially at the last bots receiving the message, the probability that the packet got corrupted is higher if the hop count is extremely high. At this point, a first trade-off arises between latency and error rate, which must be considered when further implementing the framework and choosing a nanobot population. Additionally, a high hop count is resulting in high energy consumption because of the processing overhead, which is not a desirable effect for nanobots since. The relation of average hops and the number of Nanobots suggest that the average hop count is not that high. But if the energy of a bot is used for sending messages around the human body, the bot won't have enough energy to accomplish any tasks, and the message is sent around without any use.

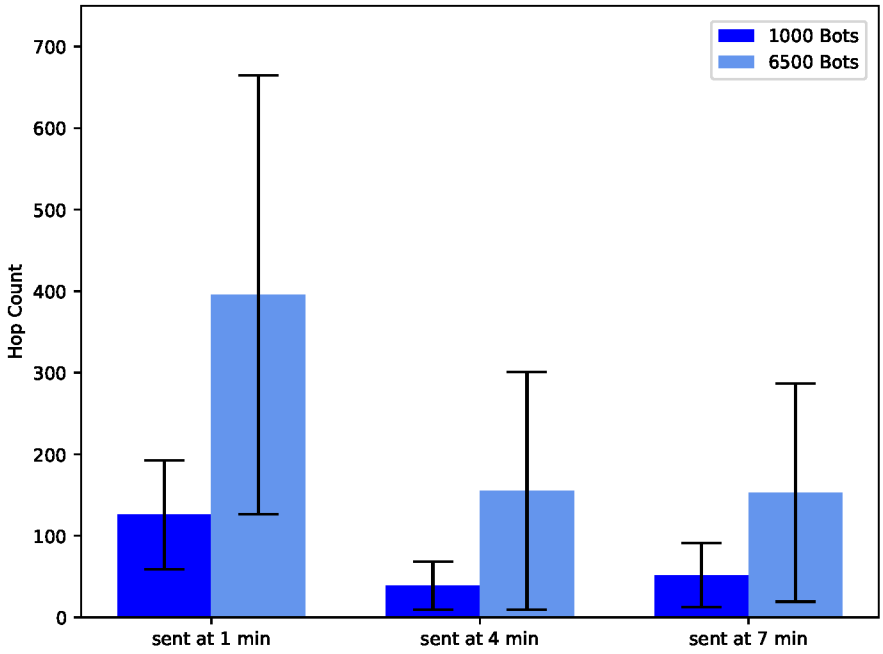


Figure 4.9 – Hop Count – Gateway at heart

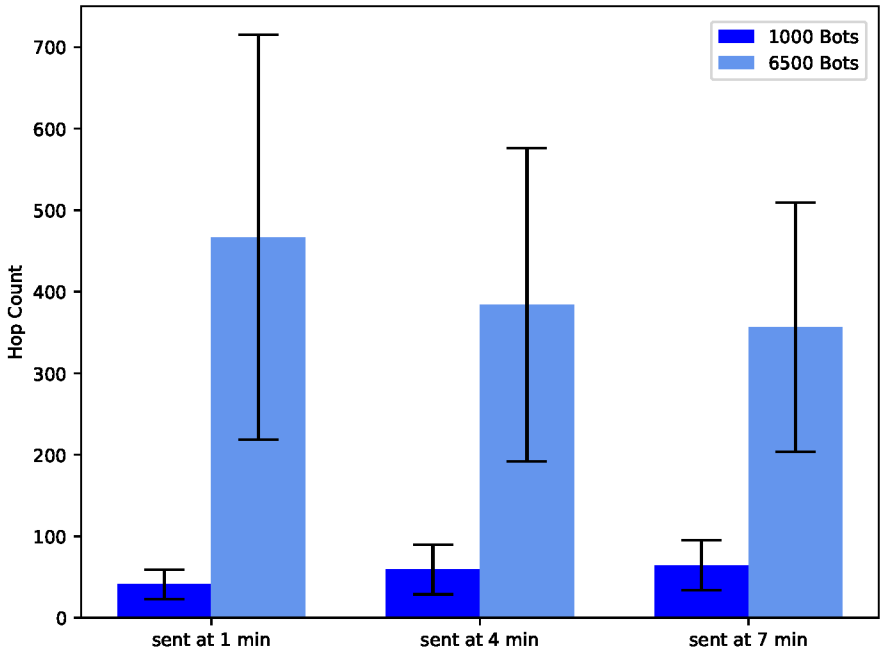


Figure 4.10 – Hop Count – Gateway at left forearm

Chapter 5

Conclusion

In the course of my thesis, I developed a simulation system, which is a first attempt to link in-body communication with out-of-body communication. For this purpose, I used a LR-WPAN connection as a communication link between a laptop and a gateway. I implemented a proximity approach providing the link between a gateway and nanobots as well as between nanobots themselves. To ensure the reliability of the system and to find weaknesses of the system I ran different simulation scenarios varying the gateway positions, the number of nanobots, and the timing of a message.

Although the values are far from being realistic since the models are quite abstract, preliminary trends of the system and possible trade-offs can be found that have to be considered for further implementing this simulation framework. Through this initial evaluation, it can be observed that the various parameters all have an effect on the latency of packets. While the number of nanobots and the position of the gateway has explainable effects on the latency, the timing of the message also has a strong not yet explainable effect. Independent of the nanobot population, more nanobots are present in the upper body than in any other body region. This causes a message to spread faster in this part of the body than in others. Consequently, medical problems like morbid cells in vessels located in the upper body can be addressed better than in other body regions. The variation in the number of bots has a significant influence on the latency of packet reception at the nanobots. This effect is especially pronounced if the position of the gateway is set to a more convenient position for the patient like the forearm. Further investigations are needed to find out, if the massive amount of bots in the upper body is due to the circulatory system or if it is a characteristic of the implementation. By getting an average hop count a first trade-off that has to be considered was found. A high amount of nanobots is able to reduce latencies but goes along with a higher hop rate and therefore a potentially higher error rate.

5.1 Future Work

Still, some important metrics for evaluating the reliability of the system are missing for example the Round Trip Time (RTT) if sending acknowledgments by nanobots is enabled or even the packet loss of acknowledgment packets since nanobots will not have large storage capacities and many acknowledgments will be sent at the same point in time. These are metrics that need to be carefully evaluated in future work. Although some effects have found an explication in this bachelor thesis, others remain open and more investigation is required. This is particularly relevant for finding patterns and classifying the relevance of the different timings of messages.

In addition, a huge part of future work will be the replacement of the proximity communication links. Molecular communication and terahertz communication are considered to provide reliable communication channels for nanobots within a human body. It was originally planned to replace the proximity link between gateway and nanobots by an ultrasonic channel, due to a lack of data and time it will be part of future work. Nevertheless, Santagati et al. [9] is stating how the influence of signal-to-interference-plus-noise-ratio can be calculated for an ultrasonic communication channel and will be important to consider if the implementation of the framework is in a less abstracted state.

List of Abbreviations

BAN	Body Area Network
BVS	BloodVoyagerS
BVS-VIS	BloodVoyagerS-Visualizer
CCA	Clear Channel Assessment
FCN	Function Centric Networking
GTS	Guaranteed Time Slots
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Medium Access Control
MCPS-SAP	MAC Common Part Sublayer Service Access Point
MLME-SAP	MAC Layer Management Entity Service Access Point
PD-SAP	PHY Data Service Access Point
PHY	Physical Layer
PLME-SAP	Physical Layer Management Entity Service Access Point
RTT	Round Trip Time
SAP	Service Access Point
WBAN	Wireless Body Area Network
WPAN	Wireless Personal Area Network

List of Figures

2.1 This figure is designed based on the first figure in [1]. It is a simplified version that shows that sensors are able to communicate with devices from the Inter-BAN part and how further communication of the gathered data could look like. The sensors are located on a human body. 5
2.2 This figure, consisting of PHY and MAC layer, illustrates a simple data transfer between two end-to-end connected LR-WPAN devices. It is designed based on the model from the lr-wpan model library. 13
2.3 Screenshot from a video plot of BloodVoyagerS 14
2.4 Screenshot from a video plot of BloodVoyagerS 15
2.5 Screenshots from a video plot of BloodVoyagerS 15

3.1 This model is taken from our submitted paper [11]. Communication paths between: (a) different nanobots; (b) nanobots and gateway, e.g., via ultrasonic communication; and (c) gateway and smart device via an IEEE 802.15.4 network. 18
3.2 Connection between smart device and gateway via IEEE 802.15.4 network 19
3.3 This figure shows the application layer added to the lr-wpan data transfer. It is designed based on the model from the lr-wpan documentation. 20
3.4 The connection between gateway and nanobots. By using a proximity approach nanobots can only exchange information with a gateway when they are located in the left half of the heart. 23
3.5 The connection between nanobots. By using a proximity approach they can only exchange information among each other while being in the communication range of each other, which is 1 cm for every nanobot. 24

4.1 Latency of packet reception at nanobots with variation in amount of nanobots and transmission timestamp. The nanobots can try to send their packet until they have copied their packet to at least one other nanobot 29
4.2 Latency of packet reception at nanobots when setting the position of the gateway on the left forearm with variation in amount of nanobots and transmission timestamp 30
4.3 Distribution of 1000 nanobots at different timestamps – Screenshot taken from BVS-VIS 32
4.4 Distribution of 6500 nanobots at different timestamps – Screenshot taken from BVS-VIS 32
4.5 Distribution of 1000 nanobots at different timestamps – Screenshot taken from BVS-VIS – Left Arm 33
4.6 Distribution of 6500 nanobots at different timestamps – Screenshot taken from BVS-VIS – Left Arm 33
4.7 Amount of vessels in which bots first received the message in percentage 34
4.8 Fraction of vessels in which bots first received the message in percentage – Gateway at left forearm 35
4.9 Hop Count – Gateway at heart 37
4.10 Hop Count – Gateway at left forearm 37

List of Tables

3.1 Simulation Input Parameter – Addressing Nanobots 21

4.1 Simulation parameters 27

Bibliography

[1] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. Leung, "Body Area Networks: A Survey," *ACM/Springer Mobile Networks and Applications (MONET)*, vol. 16, no. 2, pp. 171–193, Apr. 2011. DOI: 10.1007/s11036-010-0260-8.

[2] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester, "A Survey on Wireless Body Area Networks," *ACM/Springer Wireless Networks (WINET)*, vol. 17, no. 1, pp. 1–18, Jan. 2011. DOI: 10.1007/s11276-010-0252-4.

[3] I. F. Akyildiz, M. Pierobon, S. Balasubramaniam, and Y. Koucheryavy, "The Internet of Bio-Nano Things," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 32–40, Mar. 2015. DOI: 10.1109/MCOM.2015.7060516.

[4] F. Dressler and S. Fischer, "Connecting In-Body Nano Communication with Body Area Networks: Challenges and Opportunities of the Internet of Nano Things," *Elsevier Nano Communication Networks*, vol. 6, pp. 29–38, Jun. 2015. DOI: 10.1016/j.nancom.2015.01.006.

[5] F. Dressler and O. B. Akan, "Bio-inspired Networking: From Theory to Practice," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 176–183, Nov. 2010. DOI: 10.1109/MCOM.2010.5621985.

[6] M. Stelzner, F. Dressler, and S. Fischer, "Function Centric Networking: an Approach for Addressing in In-Body Nano Networks," in *3rd ACM International Conference on Nanoscale Computing and Communication (NANOCOM 2016)*, New York City, NY: Association for Computing Machinery, Sep. 2016. DOI: 10.1145/2967446.2967479.

[7] R. Geyer, M. Stelzner, F. Büther, and S. Ebers, "BloodVoyagerS: Simulation of the Work Environment of Medical Nanobots," in *5th ACM International Conference on Nanoscale Computing and Communication (NANOCOM 2018)*, Reykjavík, Iceland: ACM, Sep. 2018, 5:1–5:6. DOI: 10.1145/3233188.3233196.

[8] L. Galluccio, T. Melodia, S. Palazzo, and G. E. Santagati, "Challenges and Implications of Using Ultrasonic Communications in Intra-body Area Networks," in *9th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2012)*, Courmayeur, Italy: IEEE, Jan. 2012, pp. 182–189. DOI: 10.1109/WONS.2012.6152227.

[9] G. Santagati, T. Melodia, L. Galluccio, and S. Palazzo, "Medium Access Control and Rate Adaptation for Ultrasonic Intrabody Sensor Networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 4, pp. 1121–1134, Aug. 2015. DOI: 10.1109/TNET.2014.2316675.

[10] "802.15.4-2006 - IEEE Standard for Information technology-Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)," Institute of Electrical and Electronics Engineers, Std 802.15.4, Sep. 2006.

[11] A. Kuestner, L. Stratmann, R. Wendt, S. Fischer, and F. Dressler, "A Simulation Framework for Connecting In-Body Nano Communication with Out-of-Body Devices," in *7th ACM International Conference on Nanoscale Computing and Communication (ACM NanoCom 2020)*, under review, College Park, MD, Sep. 2020.

[12] M. Ghamari, B. Janko, R. S. Sherratt, W. Harwin, R. Piechockic, and C. Soltanpur, "A survey on wireless body area networks for ehealthcare systems in residential environments," *Sensors*, vol. 16, no. 6, p. 831, 2016.

[13] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless body area networks: A survey," *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1658–1686, 2014.

[14] H. Cao, V. Leung, C. Chow, and H. Chan, "Enabling technologies for wireless body area networks: A survery and outlook," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 84–93, 2009.

[15] R. A. Khan and A.-S. K. Pathan, "The state-of-the-art wireless body area sensor networks: A survey," *International Journal of Distributed Sensor Networks*, vol. 14, no. 4, 2018.

[16] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine (COMMAG)*, vol. 40, no. 8, pp. 102–114, Aug. 2002. DOI: 10.1109/MCOM.2002.1024422.

[17] T. Wu, F. Wu, J.-M. Redoute, and M. R. Yuce, "An autonomous wireless body area network implementation towards IoT connected healthcare applications," *Ieee Access*, vol. 5, pp. 11 413–11 422, 2017.

[18] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN)," Institute of Electrical and Electronics Engineers, std 802.15.1, Jun. 2005.

[19] "ISO/IEC/IEEE International Standard - Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15-6: Wireless body area network," Institute of Electrical and Electronics Engineers, Std 802.15.6-2017, Mar. 2018. DOI: 10.1109/IEEESTD.2018.8323448.

[20] F. Büther, F.-L. Lau, M. Stelzner, and S. Ebers, "A Formal Definition for Nanorobots and Nanonetworks," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, Springer, 2017, pp. 214–226.

[21] M. L. Etheridge, S. A. Campbell, A. G. Erdman, C. L. Haynes, S. M. Wolf, and J. McCullough, "The big picture on small medicine: the state of nanomedicine products approved for use or in clinical trials," *Nanomedicine: nanotechnology, biology, and medicine*, vol. 9, no. 1, p. 1, 2013.

[22] M. Stelzner, F.-L. Lau, K. Freundt, F. Buether, M. L. Nguyen, C. Stamme, and S. Ebers, "Precise Detection and Treatment of Human Diseases Based on Nano Networking," in *11th International Conference on Body Area Networks (BODYNETS 2016)*, Turin, Italy: EAI, Dec. 2016.

[23] I. F. Akyildiz, J. M. Jornet, and M. Pierobon, "Nanonetworks: a new frontier in communications," *Communications of the ACM*, vol. 54, no. 11, pp. 84–89, Nov. 2011. DOI: 10.1145/2018396.2018417.

[24] B. Atakan and O. B. Akan, "Carbon nanotube-based nanoscale ad hoc networks," *IEEE Communications Magazine (COMMAG)*, vol. 48, no. 6, pp. 129–135, Jun. 2010. DOI: 10.1109/MCOM.2010.5473874.

[25] I. F. Akyildiz, F. Brunetti, and C. Blázquez, "Nanonetworks: A New Communication Paradigm," *Elsevier Computer Networks (COMNET)*, vol. 52, pp. 2260–2279, 2008. DOI: 10.1016/j.comnet.2008.04.001.

[26] I. F. Akyildiz and J. M. Jornet, "The Internet of Nano-Things," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 58–63, Dec. 2010. DOI: 10.1109/MWC.2010.5675779.

[27] T. Suda, M. Moore, T. Nakano, R. Egashira, A. Enomoto, S. Hiyama, and Y. Moritani, "Exploratory research on molecular communication between nanomachines," in *Genetic and Evolutionary Computation Conference (GECCO), Late Breaking Papers*, vol. 25, 2005, p. 29.

[28] G. Carneiro, P. Fortuna, and M. Ricardo, "FlowMonitor: a network monitoring framework for the network simulator 3 (NS-3)," in *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, ser. VALUETOOLS '09, Brussels, Belgium, Oct. 2009, p. 1. DOI: 10.4108/ICST.VALUETOOLS2009.7493.

[29] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[30] V. Rege and T. Pecorella, "A Realistic MAC and Energy Model for 802.15.4," in *4th Workshop on ns-3 (WNS3 2016)*, E. Gamess, B. Swenson, H. Tazaki, and T. R. Henderson, Eds., Seattle, WA: Association for Computing Machinery, Jun. 2016, pp. 79–84. DOI: 10.1145/2915371.2915379.

[31] A. Updegrove, N. M. Wilson, J. Merkow, H. Lan, A. L. Marsden, and S. C. Shadden, "SimVascular: an open source pipeline for cardiovascular simulation," *Annals of biomedical engineering*, vol. 45, no. 3, pp. 525–541, 2017.

[32] R. Wendt, C. Deter, and S. Fischer, "BVS-Vis: A Web-based Visualizer for BloodVoyagerS," in *7th ACM International Conference on Nanoscale Computing and Communication (ACM NanoCom 2020)*, under review, College Park, MD, Sep. 2020.