nanoNS3: A Network Simulator for Bacterial Nanonetworks based on Molecular Communication ☆

Yubing Jian [a], Bhuvana Krishnaswamy [a], Caitlin M. Austin [b]
A. Ozan Bicen [a], Arash Einolghozati [a], Jorge E. Perdomo [b], Sagar C. Patel [b]
Faramarz Fekri [a] Ian F. Akyildiz [a], Craig R. Forest [b] and Raghupathy Sivakumar [a]

[a] *School of Electrical and Computer Engineering*
[b] *George W. Woodruff School of Mechanical Engineering*
Georgia Institute of Technology, Atlanta, GA, USA

Abstract

We present *nanoNS3*, a network simulator for modeling Bacterial Molecular Communication (BMC) networks. *nanoNS3* is built atop the Network Simulator 3 (ns-3). *nanoNS3* is designed to achieve the following goals: 1) accurately and realistically model the real world BMC, 2) maintain high computational efficiency, 3) allow newly designed protocols to be implemented easily. *nanoNS3* incorporates the channel, physical (PHY) and medium access control (MAC) layers of the network stack. The simulator has models that accurately represent receiver response, microfluidic channel loss, transfer rate and error analysis, modulation, and amplitude addressing designed specifically for BMC networks. We outline the design and architecture of *nanoNS3*, and then validate the aforementioned features through simulation and experimental results.

Keywords: BMC Simulator, ns-3, Diffusion-based Molecular Communication, Experimental-based Simulator

1. Introduction

Molecular communication is an emerging field of communication between nodes using chemical molecules. It is a multidisciplinary field with concepts from biology, chemistry, information theory and communication used in tandem to develop molecular communication systems. The communication between nodes can, in turn, trigger the development of sophisticated practical applications that require cooperation. The medium for the molecular communication and hence the transceivers differ based on the applications, environment, signals to be sensed, etc. In recent years, bacteria have emerged as a promising candidate for molecular communication nodes or transceivers for biological applications. Engineered bacteria is used in toxicology to detect metals[1] and arsenic pollution[2]. In this work, we focus on molecular communication with bacteria as transmitter and receiver nodes.

There exist many works focusing on the theoretical analysis of BMC. [3] analyzes theoretical limits of information rate and [4, 5, 6] propose mathematical models for the channel and transceiver of BMC. Protocols and algorithms that are designed to improve the throughput performance of BMC have been studied in [7, 8]. BMC is a super-slow communication mechanism [8] as it takes 10x to 100x of minutes per signal for the receiver response. Thus, using experimental analysis to validate the performance of each state-of-the-art algorithm is extremely time-consuming. Also, exactly replicating the experimental setup for different algorithms is difficult. Thus, building a computer-based BMC simulator to analyze the performance of different BMC algorithms is an important problem. In this work, we focus on building a network simulator atop ns-3, so that different algorithms can be implemented in the simulator and the performance of those algorithms can be analyzed and compared with each other. The easy-to-use layered approach of ns-3 has resulted in it becoming one of the most popular network simulators.

There have been other attempts to build molecular communication simulators [9, 10, 11, 12, 13, 14, 15, 16]. Accurate modeling of receiver response is a key factor in the simulator. Existing simulators use a simplified approximation of the receiver response thus affecting the accuracy of the simulation. A detailed analysis of related work is presented in Section 2. The major contributions of this work are thus the following:

1) An accurate bacterial receiver response module is built in *nanoNS3*. The bacterial receiver response model is validated using experimental results.

2) A microfluidic channel loss model is implemented in *nanoNS3* with user-defined geometries and properties.

3) A bit-level communication with On-Off Keying (OOK) modulation scheme is implemented in *nanoNS3*.

4) In *nanoNS3*, new attributes are added to a new node class in ns-3 to define a bacterial transmitter, a bacterial receiver

☆This work was supported in part by the National Science Foundation under grant CNS-1110947.

and channel parameters.

5) An amplitude addressing mechanism is built in *nanoNS3*.

6) A channel capacity and an error rate analysis models are built in *nanoNS3*.

Based on the current features in *nanoNS3*, it is easy to extend the functionality of *nanoNS3* to incorporate other related BMC research. The current version of *nanoNS3* is available to be downloaded at: `http://gnan.ece.gatech.edu/ns-allinone-3.24.zip`[1].

The rest of the paper is organized as follows. In Section 2, we discuss challenges in building a network simulator for BMC networks and review existing molecular communication simulators. In Section 3, we describe the architecture of *nanoNS3* and in Section 4, we explain briefly the protocols implemented in *nanoNS3*. Finally, in Section 5, we present performance results for *nanoNS3*, and in Section 6 we present some conclusions.

2. Background and Related Work

Experimental analysis and verification of BMC networks are time-consuming [8]. Fig. 3b in Section 5 presents the response of a receiver bacteria to an input rectangular pulse of AHL signal with concentration 15μM and a pulse width of 50 minutes. It can be observed from Fig. 3b that it takes the receiver bacteria few hundred minutes to generate a GFP response and ready to receive next signal. Due to such high processing time and transmission delays, experimental verification of different algorithms is extremely time inefficient. Also, since the receiver nodes are live bacteria, it is difficult to replicate parameters across different experiments. Due to the complexity of experimental setup and the time involved, it is difficult to vary different parameters like channel characteristics or characteristics of the transmitted signal. A computer-based BMC simulator is thus necessary to analyze the performance of the existing or state-of-the-art algorithms of BMC. The objective of *nanoNS3* is thus to simulate a BMC network with genetically engineered bacterial transceivers in a microfluidic environment. Based on the extensible property of *nanoNS3*, customized BMC related features can also be implemented in *nanoNS3*.

Implementing a molecular communication simulator with bacteria as transceivers has the following challenges.

1) Due to the dynamic nature of the system, algorithms developed for BMC networks differ from traditional communication algorithms significantly. Implementing state-of-the-art protocols is important and is not a trivial extension of the traditional communication modules. Transceivers in a BMC network have high processing time and transmission delays, so BMC networks are hence called super-slow networks [8]. Protocols to improve efficiency of such slow networks differ from traditional communication protocols in terms of frame structure, dimensions used, and algorithms.

2) The response of a bacterial receiver to chemical molecules involves multiple processes [6] and is non-linear. Accurate modeling of the receiver response is important to simulate a molecular communication network with bacterial transceivers. The model in use must be specific to the transceiver considered. In this work, we focus on the bacterial communication network and hence an accurate modeling of the response of receiver bacteria to an input chemical must be used in the simulator.

3) The simulator should have high computational efficiency in simulating large BMC networks and long packet sizes.

In this context, we have developed *nanoNS3* that addresses the challenges identified in building a BMC network simulator. *nanoNS3* is developed and validated based on experiments performed using genetically engineered bacteria in microfluidic environment. *nanoNS3* is built on top of ns-3, which is a discrete event based simulator. A discrete event based simulator is best suited for simulating processes with long delays. It tracks the change in state of events and not the absolute time. Simulating absolute time is expensive for super-slow networks like BMC and therefore, event-based simulations are faster and scalable. Later in this section, we present simulation time complexity of time-based simulator that takes very long time to simulate a molecular communication system.

BMC networks are super-slow networks with very long transmission delays. Thus, using ns-3 helps *nanoNS3* to be time efficient. ns-3 borrows concepts from [17] focusing on building a scalable network simulator, so ns-3 is also equipped with good scalability performance. Therefore, we choose to develop our BMC network simulator on top of ns-3 allowing us to address one of the challenges viz., simulation time efficiency and simulating large networks. Some of the advantages of using ns-3 are: 1) maintaining good computational efficiency for large networks with high payloads, 2) open source and ease of extensibility which enable users to implement state-of-the-art algorithms based on users' demands, 3) with supporting tools from ns-3 that can be utilized directly (e.g. ns-3 logging and tracing systems).

Fig. 1 presents the time to run a simulation for an input signal with pulse duration as x-axis when using molecular communication simulators NanoNS, N3Sim and *nanoNS3*[2]. Y-axis, plotted in logarithmic scale, plots the time for a simulation to run in an *OPTIPLEX 9020* desktop computer running Intel(R) Core(TM) i7-4770 CPU at 3.40GHz and 24GB RAM. The background processes are shut down and we use *time* command of Linux to calculate the CPU time to run the simulation. We explain in details for some of the existing simulators later in the section. We implemented OOK in N3Sim and NanoNS. N3Sim is a time-based simulation framework developed to simulate diffusion based molecular communication networks. We

[1]The coding style of *nanoNS3* follows ns-3 coding style, e.g., the code layout of *nanoNS3* follows the GNU coding standard layout for C and extends it to C++, and name encoding in *nanoNS3* follows the CamelCase convention.

[2]Details of simulators comparison with *nanoNS3* is presented in Table 1.

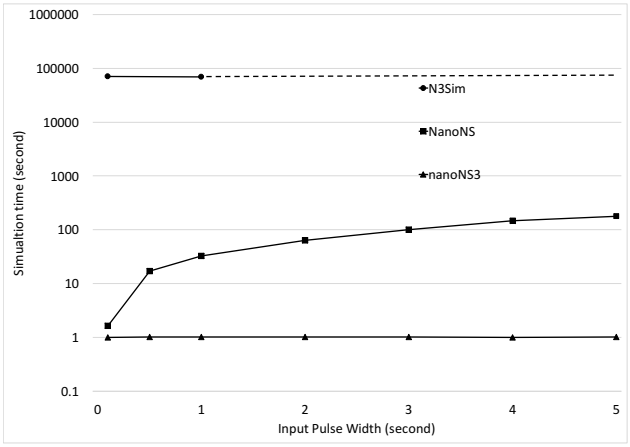


Figure 1: Simulation Time Comparison

use the default setting of the simulator for amplitude, position of the transmitter and receiver, the radius of an emitter and receiver, and diffusion coefficient. We change the pulse width of the input signal from 100 milliseconds to 5 seconds. N3Sim simulations for a pulse width 1 second takes close to 20 hours to finish the simulation. Therefore, we extrapolate the results to 5 second. We repeat the simulation for NanoNS to compare the overall simulation time when individual molecules are simulated. NanoNS is a molecular communication simulator built on top of ns-2, an event based network simulator. We set the amplitude, radius of transmitter and receiver and distance between transmitter and receiver and diffusion coefficient to be the same as that used in our aforementioned N3Sim simulations. We vary the pulse width of the input signal and observe the simulation time for increasing pulse width. It can be observed from Fig. 1 that NanoNS is faster than N3Sim. This is because, NanoNS is an event based simulator and does not wait for absolute time to perform simulations. Both NanoNS and N3Sim assume pulse widths of few nanoseconds to milliseconds in their simulations.

As presented in Fig. 3b, a bacterial receiver requires a pulse width of 50 minutes to generate a response. Therefore, a molecular communication network simulator must be able to simulate long pulse widths accurately without increasing simulation time significantly. Simulating each individual molecule in the channel to improve the accuracy is not a scalable method for super slow networks.

Existing molecular communication network simulators focus on the physical transmission and reception of molecules by simulating the interaction between and propagation of each individual molecule, leading to time inefficient simulation. *nanoNS3* simulates bit-level transmission and reception, instead of molecular level transmission and reception, which leads to higher computational efficiency. *nanoNS3* is able to simulate how transmitted bits are modulated to pulses at the transmitter side, and how the concentration of molecules is propagated and attenuated through a microfluidic channel. At the receiver side, the biological response of how N-Acyl homoserine lactone (AHL) diffuses through the biofilm material and subsequently across the bacteria membrane is simulated (Physical Layer). Then, the receiver can identify the ID of the transmitter based on the received concentration (MAC Layer). Afterward, the receiver can demodulate the received concentration to recover the transmitted information.

There are several existing works focusing on simulating Molecular Communication (MC) [9, 10, 11, 12, 13, 14, 15, 16]. Table 1 lists the properties of some of the aforementioned simulators focusing on the MC network. These approaches validate their respective simulators using numerical analysis or purely simplified theoretical models. Thus, the simulators are not verified against real-life behaviors. We discuss in detail some of the MC network simulators. NanoNS is built on top of Network Simulator 2 (ns-2), and it provides various nanoscale communication paradigms based on a diffusive MC channel [9]. This work only presents the details of the channel layer, and it simulates the diffusion and reception process using a single equation, which may not be accurate in the practical situation. This work simulates MC using molecules based approach, which is time-consuming as the molecule scales (for practical cases, the size of molecules is immense). Also, this work is based on ns-2 which is computationally inefficient with regards to memory usage and CPU utilization. Currently, ns-2 is not actively maintained, and the most recent version of ns-2 was released in 2011. dMCS developed in [15] proposes a simulation framework for the general case of diffusion-based MC, and it is developed using a customized simulator. Using customized simulator is likely to lose the advantages of dedicated network simulators like ns-3 (e.g. scalability and computational efficiency). Again, [15] is also modeling MC network using molecules based approach, which incurs large time complexity as the number of molecules scales and no higher layer protocol is implemented in this work.

In [10], N3Sim is developed based on the diffusion propagation channel to model MC networks. Similar to [15], [10] is built on a customized simulator and those network layers higher than PHY is absent in this work. N3Sim allows us to configure the network using a configuration file on the front end making it easy to use the simulator. N3Sim does not follow layered architecture in simulating the network. It focuses primarily on the physical layer diffusive channel. Therefore, we cannot use N3Sim to simulate, compare and analyze the performance of MC network for upper layer algorithms. Nano-Sim developed in [11] is also built on top of ns-3, and it provides functions to model Electromagnetic (EM) wave based nanonetworks. Similar to our work, Nano-Sim utilizes the framework and advantages of ns-3 to build EM-based nano simulator. The transmission/reception scheme in Nano-Sim is orthogonal to our work in this paper. Thus, it is feasible to combine *nanoNS3* with Nano-Sim, since they are both implemented atop ns-3. Other than aforementioned MC simulators, [14] proposes a simulation framework that is adaptable to any kind of nano bearer and the simulator is also validated using experimental analysis in [18], but it is developed using a customized simulator. Thus, it is likely to lose the advantages of dedicated network simulators. To the best of our knowledge, *nanoNS3* is the first BMC network simulator validated using experimental analysis that achieves a demodulation accuracy greater than **92.5%**.

Table 1: Simulators Comparison

Features/Simulator	NanoNS	N3Sim	Nano-Sim	nanoNS3
Physical Layer/Channel Model	Diffusive channel, Gillespie model for stochastic reaction process	Baraff's algorithm simulating collision	Spectrum channel model	Bacterial receiver model and channel loss model
Protocols Implemented	Molecular node emitting molecules	Emitter types to generate molecules pulse trains	Selective and Random routing, TransparentMAC, SmartMAC, Time spread OOK	OOK, Source addressing, Error analysis and transfer rate analysis model, ns-3 application layer
Validation	Numerical analysis of models used	N/A	N/A	Experimental evaluation
Compatibility of Network Simulator	ns-2	N/A	ns-3	ns-3
Programming language used	Tcl, C++	Java	C++	C++
Modularity of Network Architecture	Higher layer protocols of ns-2 can be integrated with the physical layer	Does not have provision to implement higher layer protocols	Higher layer protocols of ns-3	Higher layer protocols of ns-3

3. Network Architecture

nanoNS3 is developed atop ns-3 [19]. ns-3 is a discrete event, open source and widely used network simulator for internet systems, targeted primarily for research and educational use (ns-3 is developed in C++ and python). ns-3 is developed based on modules, and each individual module represents a protocol (e.g. AODV), a technology (e.g. WiFi) or an attribute of networks (e.g. mobility). It enables the easy and convenient upgrade of source code and triggers the ease of extensibility in ns-3 by this modular implementation method. ns-3 is actively maintained and it is free software and licensed under GNU GPLv2 license. ns-3 has the best overall performance compared with other popular network simulators [20]. E.g. ns-3 has the least memory usage for large-scale network simulations compared with ns-2, OMNeT++, JiST and SimPy. Implementing *nanoNS3* in ns-3 has the following major advantages: 1) open sourced availability and ease of implementation for new algorithms, 2) high computational efficiency for large-scale networks, and 3) supporting tools from ns-3 can be utilized directly (e.g. ns-3 logging and tracing systems).

3.1. nanoNS3 Network Architecture

The high-level structure of *nanoNS3* is shown in Fig. 2. The name of seven important classes with the structure of the corresponding network layers are given in Fig. 2. The functionality of each class is discussed briefly below:

- *NanoNetDevice*: It is similar to the Network Interface Card (NIC), and it can support different nano communication technologies (e.g. diffusive or EM wave based nano communication schemes) and corresponding protocols (e.g. amplitude addressing).
- *NanoNode*: It can be regarded as the physical device, and different *NanoNetDevices* can be integrated with *NanoNode* to provide corresponding communication technologies and protocols to enable *NanoNode* to communicate with each other.
- *PacketSocket*: This class is a simple and original ns-3 application class, which does not use IP addresses. It is used to set up user defined applications for nano communications by controlling application-related parameters, e.g., packet arriving interval, the number of maximal transmission packets and packet size.
- *NanoRouting*: This class manages message forwarding by each *NanoNode*.
- *NanoMAC*: This class manages channel access of different *NanoNodes*, and it also manages MAC layer addressing mechanism.
- *NanoPHY*: This class is used to simulate the process of transmitters and receivers to transmit and receive the nano signals. The corresponding functionality of this class includes modulation, demodulation, error analysis and transfer rate analysis, and receiver response.
- *NanoChannel*: This class is used to set up channel conditions, and then the channel loss can be calculated to simulate how the transmitted signals are propagated and attenuated in the corresponding microfluidic channel.

Specifically, packet client, packet server, and packet address are three classes related to packet socket class. The packet

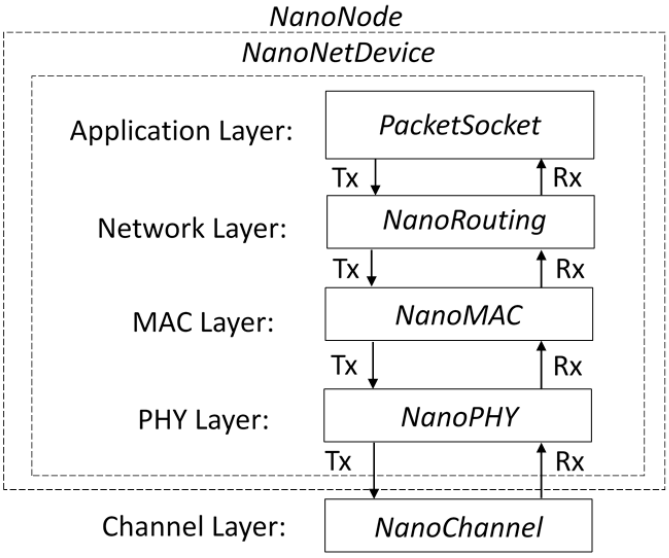


Figure 2: nanoNS3 Architecture

client class defines how packets are transmitted from Tx application layer, and the packet server class defines how packets are received from Rx application layer. The packet address class defines how addresses of different nodes are set up. We integrate the ns-3 original application class into *nanoNS3*, in order to make it more convenient to integrate other original ns-3 classes with *nanoNS3*, e.g. transport layer protocol. In ns-3, transport layer protocol is integrated with socket, so we integrate packet socket into *nanoNS3* to make it possible for transport layer extension. The parameters for each aforementioned class can be customized by users. Protocols implemented in *NanoMAC*, *NanoPHY*, and *NanoChannel* will be discussed in Section 4.

4. Protocols Implemented

nanoNS3 implements some of the basic protocols to simulate bacterial molecular communication networks. The important **5** models implemented in *nanoNS3* are: 1) Receiver response model, 2) Channel loss model, 3) OOK model, 4) Amplitude addressing model and 5) Transfer rate and error analysis model.

4.1. Receiver Response Model

As discussed in Section 2, accurate modeling of receiver response is important to simulate BMC networks. *nanoNS3* focuses on simulating BMC networks and hence modeling of the response of bacteria to molecular signal is described in this section. The bacterial receiver implemented in this simulator is a population of genetically engineered *E. coli* bacteria that generates a Green Fluorescent Protein (GFP) on receiving AHL molecules. The transceivers are located in a microfluidic device connected by microfluidic pathways. The transmitter transmits molecules that are transported to the receiver through microfluidic pathways and the receiver emits green fluorescence. The relative fluorescence of the receiver bacteria indicates the signal received. [6] reviews the existing bacterial receiver models in a microfluidic environment and proposes a new model. [6] validates the model using experiments. In *nanoNS3*, we implement the following model proposed in [6].

$$\frac{dAHL_i}{dt} = k_c(AHL_e - AHL_i) - k_1 AHL_i{}^2 LuxR^2 + k_1 C_1 \quad (1)$$

$$\frac{dC_1}{dt} = k_1 AHL_i{}^2 LuxR^2 - k_1 C_1 - k_2 C_2 P_{Lux} \quad (2)$$

$$\frac{dC_2}{dt} = k_2 C_1 P_{Lux} - k_2 C_2 - k_{tr} C_2 \quad (3)$$

$$\frac{dLuxR}{dt} = k_{Lc} - 2C_1 - 2C_2 \quad (4)$$

$$\frac{dGFP_i}{dt} = k_{tr} C_2 - k_{Gm} GFP_i - k_{Gd} GFP_i \quad (5)$$

$$\frac{dGFP_m}{dt} = k_{Gm} G + i - k_{Gd} GFP_m \quad (6)$$

AHL_e and AHL_i are the external and internal concentrations of molecules at the receiver. $LuxR, C_1, C_2$, and GFP_i represent internal parameters of the receiver bacteria. GFP_m represents the concentration of GFP which in turn represents the relative fluorescence at the receiver. Vector $k = [k_c, k_1, k_2, k_{Lc}, k_{tr}, k_{Gd}, k_{Gm}]$ represents rate constants of the processes in the receiver. The equations and corresponding derivation are explained in detail in [6].

The above equations are used to model the channel and receiver response in *nanoNS3*. The transmitter module generates bits and those bits are input to the modulator. Modulator implemented in *nanoNS3* is OOK and hence bits are mapped to rectangular pulses. The rectangular pulses are input to these equations to simulate the GFP response of the receiver. A numerical inverse of these equations is used to sample and quantize the received signal. The output of the sample and quantize modules is then fed to the demodulator to process. This model gives a bit level response at the receiver.

4.2. Channel Loss Model

Microfluidic channel loss model is an important component for BMC simulators, which provides insights for how the concentration of signals are attenuated while propagating in microfluidic channel. [5] provides a comprehensive coverage of the microfluidic channels with fluid flow for diffusion-based Flow-induced Molecular Communication (FMC). In FMC, the fluid is flowing through a microfluidic channel and it serves as a communication channel to connect patches of molecular transmitter and receiver, such as bacterial habitat. In [5], an analytic study of the propagation of the molecules in the form of the impulse response is performed incorporating the physical system parameters. The goal of the propagation loss model is to determine the channel loss effects caused on the molecular signal with respect to the distance, fluid flow parameters (pressure drop, flow velocity, microfluidic channel geometry and fluid type), and type of the molecule (diffusion constant). In [5], channel loss models for the basic microfluidic channel shapes (straight and turning) and cross-sections (rectangular, square,

elliptical, circular) are developed incorporating the characteristics of the fluid flow and mass transport in the microfluidic channels. In *nanoNS3*, we implement the models presented in [5]. In the interest of brevity, we will illustrate rectangular cross-section microfluidic channel loss model in this section, and evaluate the specific microfluidic channel loss model in Section 5. The governing set of equations for the channel loss in a rectangular microfluidic channel are given as [5].

$$G_{rect} = \frac{h^3 w}{12\mu l} * (1 - 0.63\frac{h}{w}) \tag{7}$$

$$u_{rect} = G_{rect} * \triangle p \tag{8}$$

$$\tau_{rect} = l/u_{rect} \tag{9}$$

$$TF_{rect} = e^{-(k^2 D + jku_{rect}) * \tau_{rect}} \tag{10}$$

where G_{rect}, u_{rect} are the hydraulic conductance of the microfluidic channel and area-averaged flow rate, respectively. G_{rect} is a function of channel cross-section shape, dimensions, and viscosity of the fluid (μ). u_{rect} is a function of pressure drop ($\triangle$p) and G_{rect}. τ_{rect} and TF_{rect} are the delay and attenuation of channel, respectively. h and w represent the channel height and width. l and D are the lengths of the straight channel and Taylor dispersion-adjusted diffusion constant. The detailed explanations for all microfluidic channel loss models are given in [5].

4.3. On-Off Keying (OOK) Model

Modulation is the process of varying the properties of a signal to convey the *information*. Modulation determines the rate of transmission. ns-3 is a packet-level simulator and hence allows users to change modulation by changing the transmission rate. In *nanoNS3*, we implement bit level simulation and hence implement a module for modulation that maps each input bit to a signal to be transmitted. OOK is one of the simplest modulation techniques and majority of works on BMC assume OOK as the modulation technique. OOK transmits a rectangular pulse of amplitude/concentration A for a duration of T_1 time units to send bit 1 and no signal for T_2 time units to send bit 0. OOK module in *nanoNS3* generates a rectangular pulse of a given amplitude and a given duration. Then, the generated rectangular pulse is fed to the channel model.

4.4. Transfer Rate and Error Analysis Model

Based on the framework of *nanoNS3*, a theoretical analysis model [3] is implemented in *nanoNS3*, which is designed to estimate the theoretical limits of the information transfer rate and corresponding error probabilities of bacterial molecular communication based on the uncertainty of communications, i.e., transmission, propagation and reception. In [3], this work considers a molecular communication setting in which the diffusion channel inputs and outputs are OOK with the binary set {0, 1}. The effects of uncertainty in the production of molecules, channel parameters and reception process on the overall noise of the communication are considered. This model can be used to study the theoretical limits of the information transfer rate in terms of the number of bacteria per node, noise level and maximum molecule production levels. Also, it can be utilized to analyze the achievable rates and the error probabilities of M-ary schemes.[3] In *nanoNS3*, we implement the models presented in [3]. In the interest of brevity, we will not present the description of the transmission, propagation and reception model and corresponding formula proofing.

4.5. Amplitude Addressing Model

The models mentioned above (receiver response, channel loss, transfer rate and error analysis and OOK) model the channel and physical layer of BMC networks. A MAC protocol is required in a network with multiple sources to achieve fairness and reduce collisions. MAC protocols used in wired or wireless networks as implemented in ns-3 increase the delay and decrease the throughput in a super-slow network like the BMC network. [7] considers a multiple sources and single receiver topology and proposes a MAC protocol to improve the BMC network throughput. Multiple sources transmitting to a single receiver is a typical sensing network scenario. [7] proposes a local addressing mechanism which implicitly performs MAC, and it is implemented in *nanoNS3*. [7] analyzes the advantages and disadvantages of various addressing mechanisms and proposes *Amplitude Addressing* for BMC networks. *Amplitude Addressing* assigns distinct amplitudes to each user in a BMC network and each user uses the assigned amplitude with OOK to transmit information. The receiver receives the sum of the transmitted amplitudes which are then resolved to identify the individual amplitude thus solving addressing and MAC in the local network. We further implement two different receivers viz., load-aware and load-unaware. Load-aware receiver has an estimate of input load of the transmitter. Therefore, the decoding table (decoding table is used to decode the transmitter ID based on the received signal) is a function of the input load of the transmitter. On the other hand, load-unaware receiver assumes that the input is uniformly distributed for the transmitter and the decoding table is fixed. We present the results for both the receivers in the next section. A global address is required to map local address to the source. *nanoNS3* does not have a global addressing module, but MAC address of ns-3 nodes can be used as a global address module in simulations

5. Results

In this section, we present the evaluation results of *nanoNS3* using both simulation-based and experimental-based validation for the **5** models mentioned in Section 4. *nanoNS3* provides two examples of scenario setup: 1) single Tx and single Rx, and 2) multiple Txs and single Rx.

5.1. Methodology

In this section, we validate *nanoNS3* with protocols mentioned in Section 4. For the receiver response model, we validate the simulation results of *nanoNS3* with results from MATLAB analytic model used in [6] and experiments. For amplitude

[3] Although *nanoNS3* only supports OOK, it can be utilized to study the theoretical limits of transfer rate and error probability for M-ary schemes.

addressing model, we validate *nanoNS3* with python simulator used in [7]. For channel loss model, we validate *nanoNS3* with MATLAB analytical model used in [5]. For transfer rate and error analysis model, we validate *nanoNS3* with MATLAB analytical model used in [3]. Unless otherwise mentioned, the transceivers used are bacteria and the carrier signal is a molecular signal. To validate the performance of the aforementioned models in *nanoNS3*, the following scenarios are used:

- Single Tx and single Rx scenario: It is a single link scenario where one transmitter sends signals to one receiver. The default transmitted molecular concentration is set as 15 μM.

- Multiple Txs and single Rx scenario[4]: Multiple transmitters send signals to a single receiver in this scenario. The amplitude assigned to each transmitter is based on the mechanisms shown in [7], and two examples will be given in Section 5.5.

5.2. Receiver Response

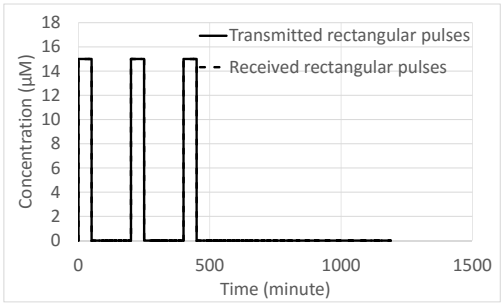
We implement the receiver response model derived in [6]. The set of differential equations presented in [6] defines the GFP response of the receiver bacteria to a given concentration of molecules. We built an *Inverse model* of the receiver response at the receiver to estimate the molecules received from the receiver GFP response.

We validate *nanoNS3* receiver response in two steps. First, we verify the numerical inverse response using simulations. A transmitter sends information using OOK, i.e. rectangular pulses of a fixed amplitude and a fixed duration for bit 1 and no signal for bit 0. The rectangular pulses are input to *Forward response* generating receiver GFP response which is then fed to *Inverse response* derived numerically that estimates the signal transmitted based on the GFP response. Second, we compare the forward receiver response obtained from simulator with the response from experiments. We also input the receiver response from experiments to the *Inverse model* and compare the estimated signal with the actual transmitted signal.

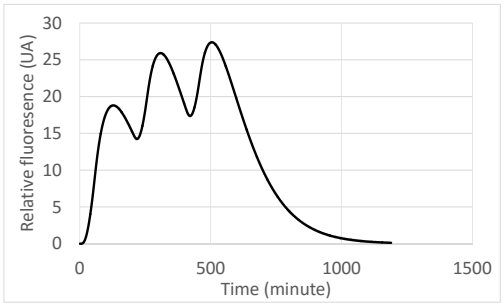
5.2.1. Receiver response : simulation validation

In this section, we validate the inverse receiver response model using OOK. Fig. 3a and Fig. 3b present the transmitted and received rectangular pulses for the input bits and forward receiver response, respectively. The corresponding simulation parameters are shown in Table 2. As we can see from Fig. 3a, the transmitted rectangular pulses exactly match the received rectangular pulses. This result illustrates that the receiver can recover the transmitted pulses in *nanoNS3*. The corresponding forward receiver response for the transmitted rectangular pulses is given in Fig. 3b.

[4]Only amplitude addressing model is validated using the multiple Txs and single Rx scenario.



(a) Tx/Rx pulses comparison



(b) Forward receiver response

Figure 3: Simulation Validation

Table 2: Simulation Parameters for Receiver Response

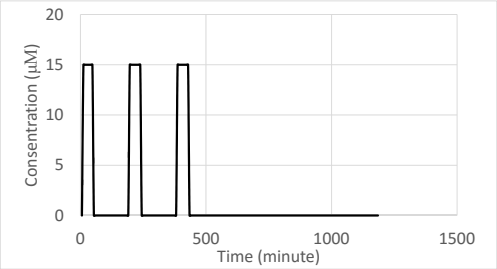
Parameters	Default Settings
Tx sequence	1000100010
Modulation	OOK
Tx pulses amplitude	15 μM
Tx pulses width	50min
$threshld$	7.5μM
k_c	254/60
k_{Gm}	1.8/60
k_{Gd}	39/60
k_{tr}	1334/60
k_{Lc}	1200/60
k_1	20/60
k_2	200/60

5.2.2. Receiver response : experimental validation

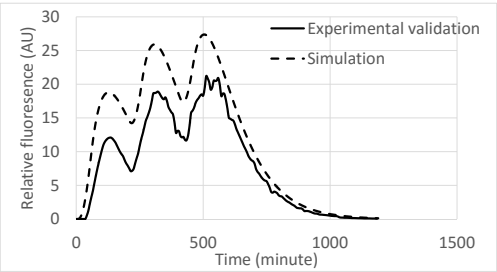
In this section, we validate the receiver response model using experimental results. Experimental setup used to obtain these results are the same as explained in [6]. We compare the receiver response obtained from experiments and simulations. We also verify the inverse of receiver response. The receiver response from experiments is input to the inverse model and we compare the estimated transmitted signal with the actual transmitted signal.

Forward receiver response validation

Fig. 4a presents the transmitted rectangular pulses. Fig. 4b shows the simulation results and experimental results of the forward receiver response. It can be observed that the simulation and experimental results have the similar trends (peaks of each pulse in experimental results can be exactly captured by simula-

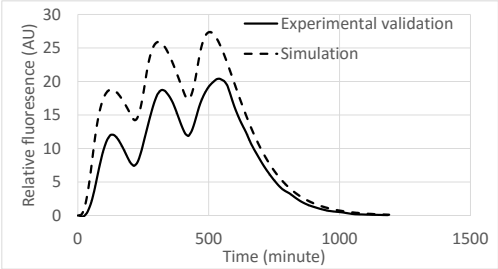


(a) Experimental Tx pulses

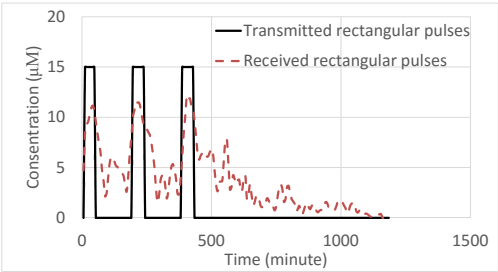


(b) Forward receiver response

Figure 4: Experimental Validation



(a) Processed experimental results



(b) Processed experimental results validation

Figure 5: Experimental Validation

tion). Four different Tx sequences with 10 bits in each sequence are set as Tx sequence in the experimental apparatus, and each of them shows the similar trend as the presented results.

Inverse receiver response validation

Based on the experimental results of forward receiver response, we validated the inverse of the receiver response. From Fig. 4b, it is clear that experimental results of forward receiver response are not as smooth as the simulation results of forward receiver response. In order to get rid of noises and system errors, we use Loess smooth function (with 10% span) in MATLAB to smooth the experimental results. The corresponding experimental results are shown in Fig. 5a. Then, we input those smoothed experimental results to the inverse of receiver response model in *nanoNS3*, and we succeeded to recover the transmitted bits at the receiver side. From Fig. 5b, it can be observed that the peaks of transmitted and received pulses match with each other. In order to demodulate the received pulses, we calculate the average concentration for the pulse duration of each bit, where average concentration for bit i is represented as Ave_i. We set a threshold to determine the bit level, where the *threshold* is set as half of transmitted concentration. Received bit is determined by the following equation:

$$\text{Received bit} = \begin{cases} 1 & \text{if } Ave_i >= threshold \\ 0 & \text{if } Ave_i < threshold \end{cases}$$

Utilizing this method, we can achieve **100%** of demodulation rate for the presented case of experimental results. The average demodulation rate of four sets of experimental results achieves **92.5%**.

To conclude, *nanoNS3* provides an experimental-based receiver response model:

> *For the receiver response model, the simulation results of* nanoNS3 *match the simulation results of MATLAB analytic model. The implemented model enables higher layer protocols in* nanoNS3 *to achieve high accuracy of their simulation performance.*

5.3. Channel Loss

In this section, we compare the channel loss model in single Tx and single Rx scenario. We compare the results obtained by *nanoNS3* with respect to the MATLAB using the described analytic channel loss model in Section 4.2. The objective is to show that the numerical results of *nanoNS3* match the numerical results from MATLAB for the analytic model channel loss model. The details of the default parameter settings are shown in Table 3.

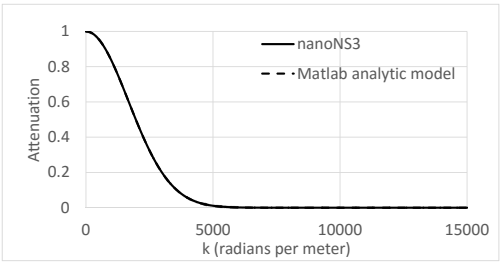
Fig. 6a and 6b illustrate how the impulse response attenuation of straight and turning channel varies with frequency (radians per meter), respectively. We observe that the channel loss module implemented in *nanoNS3* provides the exactly same results with the analytic model evaluation in MATLAB for both Fig. 6a and 6b.

To conclude, *nanoNS3* provides a microfluidic channel loss model:

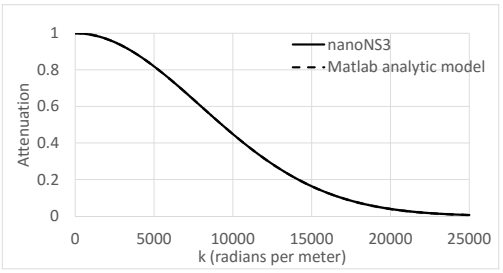
> *The simulation results of* nanoNS3 *match the simulation results of MATLAB analytic model. More channel properties and other channel loss model can be easily implemented in* nanoNS3 *based on the implemented channel loss model.*

Table 3: Simulation Parameters for Channel Loss

Parameters	Default Settings
Channel shape	rectangular
Turning angle	30 degree
viscosity	10^{-3}Pa*s
D	$10 * 10^{-10} m^2/s$
$\triangle p$	500 pa
l	10mm
h	$6\mu m$
w	$25\mu m$



(a) Straight rectangular channel



(b) Turning rectangular channel

Figure 6: Channel loss validation

Table 4: Simulation Parameters for Capacity and Error Analysis

Parameters	Default Settings
Number of bacterial	100
Number of ligand receptors	50
Square of combined noise variance	0.1
M-ary scheme	16

5.4. Transfer Rate and Error Analysis

In this section, we compare the transfer rate and error analysis model in single Tx and single Rx scenario. We compare the results obtained by *nanoNS3* with respect to the MATLAB analytic model used in [3]. The objective is to show that the numerical results of *nanoNS3* match the numerical results from MATLAB analytic model. The details of the default parameter settings are shown in Table 4.

Fig. 7 illustrates how the channel capacity varies with the maximum concentration of molecules. As M-ary scheme is utilized, Fig. 8a and 8b illustrate how the channel capacity and corresponding error rate varies with the maximum concentration of molecules. We observe that the theoretical limits for both capacity and error rate generated from *nanoNS3* match the results with the MATLAB analytic model from Fig. 7, Fig. 8a and 8b.

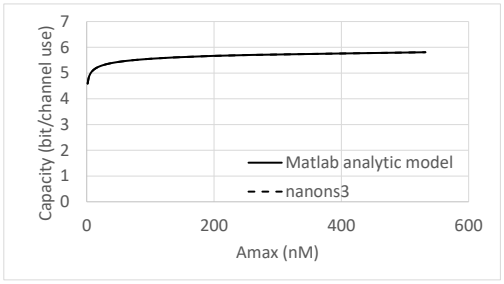
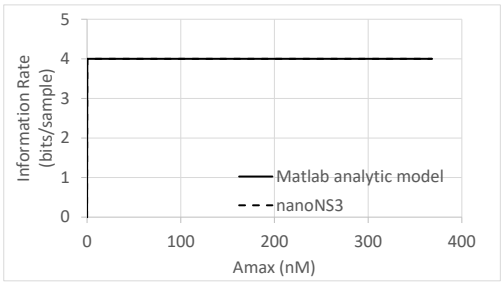
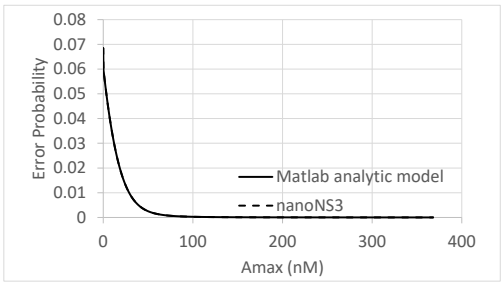


Figure 7: Transfer Rate Analysis



(a) Transfer Rate Analysis



(b) Error Rate Analysis

Figure 8: Transfer Rate and Error Rate Analysis for Corresponding Modulation

To conclude, *nanoNS3* provides a theoretical analysis model:

> *The simulation results of* nanoNS3 *match the simulation results of the MATLAB analytic model. Based on the implemented theoretical analysis model, transfer rate and error probability with corresponding modulations can be estimated.*

5.5. Amplitude Addressing

In this section, we validate the amplitude addressing mechanism in multiple Txs and single Rx scenario. We compare the performance of *nanoNS3* versus the custom built python simulator used in [7]. The objective is to show that the simulation results of *nanoNS3* match the simulation results from the aforementioned python simulator. The details of default pa-

Table 5: Simulation Parameters for Amplitude Addressing

Parameters	Default Settings
Number of Tx bits	100
Amplitude assignment mechanism	Integer/Binary
Tx pulses width	50mins
Tx pulses interval	20mins
p_t	0.5
Max amplitude	$15\mu M$
Number of Tx users	5

rameter settings are shown in Table 5. Two examples of amplitude assignment mechanism will be briefly introduced. For integer amplitude assignment mechanism, each user is assigned with a unique amplitude based on the node ID. E.g. amplitudes {1,2,3} are assigned to transmitters with ID {1,2,3} with one-to-one correspondence. For binary amplitude assignment mechanism, amplitudes {1,2,4,8} are assigned to transmitters with ID {1,2,3,4} with one-to-one correspondence.

Fig. 9a plots the demodulation accuracy for load unaware addressing at the receiver for varying input load for integer amplitude assignment. As we can see from Fig. 9a, the performance of *nanoNS3* is very close to that of the custom built python simulator used in [7]. Fig. 9b plots the decoding accuracy at the receiver for varying input load using binary amplitude assignment. It can be observed that the simulation results of *nanoNS3* and the python simulator are very close to each other.

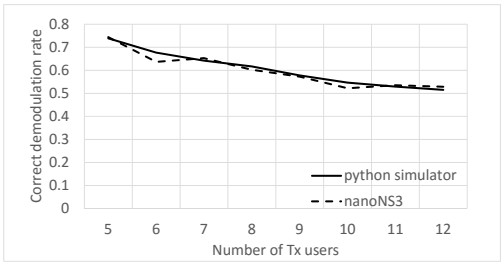
We also compare the performance of load aware and load unaware receiver. Fig. 10 shows the correct demodulation rate using load aware and load unaware receiver with binary amplitude assignment. It can be observed that when the receiver is aware of the load of the transmitter, correct demodulation rate increases compared with load unaware receiver. The amplitude sequences used for evaluation are described in detail in [7].

To conclude, *nanoNS3* provides an amplitude addressing mechanism:

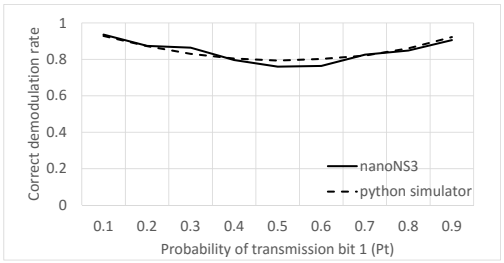
> *The simulation results of* nanoNS3 *match the simulation results of the python simulator. Based on the implemented addressing model, the performance of higher layer protocols can be explored (e.g. routing mechanisms).*

5.6. Simulator time complexity comparison

In Section 2, we presented the simulation time complexity of three simulators NanoNS, N3Sim, and *nanoNS3* for increasing simulation input pulse width. We observed that a time-based simulator like N3Sim that simulates molecular interaction is not suitable for long simulations. The simulation time complexity is in the order of hours for input signal duration of few milliseconds and is not suitable for long simulations. NanoNS improves simulation time by using an event-based simulator like ns-2 and simplifying molecular interactions. Using simple equations to represent molecular interactions can affect the accuracy of response of a receiver to an input signal. Simulation time complexity of NanoNS increases exponentially with input pulse width making it unsuitable for longer packet sizes.



(a) Integer Tx sequence



(b) Binary Tx sequence

Figure 9: Amplitude Addressing Validation

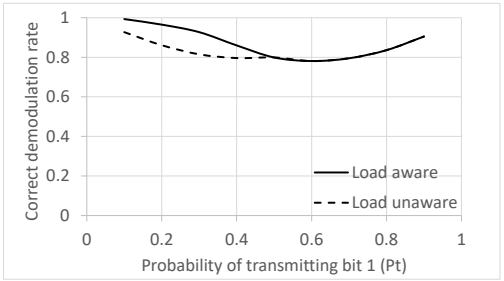


Figure 10: Load unaware vs. load aware receiver

nanoNS3 uses experimentally verified physical layer models to simulate the response of a receiver to an input signal without compromising on the accuracy of the receiver. *nanoNS3* is implemented on top of ns-3, an event based, time efficient simulator. We compare the simulation time performance of other existing simulators with that of *nanoNS3*. As expected, *nanoNS3* has a constant time complexity for increasing pulse width as it uses a bit-level model to simulate bit-level response of receiver bacteria. Constant simulation time complexity makes *nanoNS3* suitable for simulating large MC networks with long pulse widths and long packet sizes.

6. Conclusions

In this paper, we describe *nanoNS3*, a new network simulator built atop ns-3 for BMC networks. The choice of the ns-3 simulator allows high computational efficiency for large-scale networks and easy implementation of new algorithms. An accurate model of the bacterial receiver response to chemical signals

modulated using OOK is implemented. A microfluidic channel loss model that incorporates the physical system parameters is implemented in *nanoNS3*. A source addressing protocol that implicitly solves MAC issues is also implemented in *nanoNS3*. *nanoNS3* thus focuses on the channel, PHY and MAC layers of the network protocol stack. Due to the lack of upper layer protocols in BMC network, *nanoNS3* only provides an original ns-3 application layer model for upper layers. By making use of the layered architecture of ns-3, it is possible to use existing IP, transport and application layer protocols in ns-3 to test the performance of BMC networks. For future work, we plan to improve the completeness of *nanoNS3* by implementing new features such as channel propagation delay model. Moreover, we will explore merging *nanoNS3* with other MC simulators that are also implemented atop ns-3, in order to move closer to the vision of a general purpose MC simulator.

References

[1] T. Charrier, C. Chapeau, L. Bendria, P. Picart, P. Daniel, G. Thouand, A multi-channel bioluminescent bacterial biosensor for the on-line detection of metals and toxicity. part ii: technical development and proof of concept of the biosensor, Analytical and Bioanalytical Chemistry 400 (4) (2011) 1061–1070.

[2] J. Stocker, D. Balluch, M. Gsell, H. Harms, J. Feliciano, S. Daunert, K. A. Malik, J. R. van der Meer, Development of a set of simple bacterial biosensors for quantitative and rapid measurements of arsenite and arsenate in potable water, Environmental Science & Technology 37 (20) (2003) 4743–4750.

[3] A. Einolghozati, M. Sardari, F. Fekri, Design and analysis of wireless communication systems using diffusion-based molecular communication among bacteria, IEEE Transactions on Wireless Communications 12 (12) (2013) 6096–6105.

[4] M. Pierobon, I. F. Akyildiz, A statistical–physical model of interference in diffusion-based molecular nanonetworks, IEEE Transactions on Communications 62 (6) (2014) 2085–2095.

[5] A. O. Bicen, I. F. Akyildiz, System-theoretic analysis and least-squares design of microfluidic channels for flow-induced molecular communication, IEEE Transactions on Signal Processing 61 (20) (2013) 5000–5013.

[6] C. M. Austin, W. Stoy, P. Su, M. C. Harber, J. P. Bardill, B. K. Hammer, C. R. Forest, Modeling and validation of autoinducer-mediated bacterial gene expression in microfluidic environments, Biomicrofluidics 8 (3) (2014) art. no. 034116.

[7] B. Krishnaswamy, R. Sivakumar, Source addressing and medium access control in bacterial communication networks, 2nd ACM Annual International Conference on Nanoscale Computing and Communication (2015) 1–6.

[8] B. Krishnaswamy, C. M. Austin, J. P. Bardill, D. Russakow, G. L. Holst, B. K. Hammer, C. R. Forest, R. Sivakumar, Time-elapse communication: Bacterial communication on a microfluidic chip, IEEE Transactions on Communications 61 (12) (2013) 5139–5151.

[9] E. Gul, B. Atakan, O. B. Akan, Nanons: A nanoscale network simulator framework for molecular communications, Nano Communication Networks 1 (2) (2010) 138–156.

[10] I. Llatser, D. Demiray, A. Cabellos-Aparicio, D. T. Altilar, E. Alarcón, N3sim: Simulation framework for diffusion-based molecular communication nanonetworks, Simulation Modelling Practice and Theory 42 (2014) 210–222.

[11] G. Piro, L. A. Grieco, G. Boggia, P. Camarda, Nano-sim: simulating electromagnetic-based nanonetworks in the network simulator 3, 6th International ICST Conference on Simulation Tools and Techniques (2013) 203–210.

[12] Calcomsim: https://sites.google.com/site/calcomsimulator/.

[13] Comsol-multiphysics: https://www.comsol.com/comsol-multiphysics.

[14] L. Felicetti, M. Femminella, G. Reali, A simulation tool for nanoscale biological networks, Nano Communication Networks 3 (1) (2012) 2–18.

[15] A. Akkaya, T. Tugcu, dmcs: distributed molecular communication simulator, in: 8th ICST International Conference on Body Area Networks, 2013, pp. 468–471.

[16] Y. Jian, B. Krishnaswamy, C. M. Austin, A. O. Bicen, J. E. Perdomo, S. C. Patel, I. F. Akyildiz, C. R. Forest, R. Sivakumar, nanons3: Simulating bacterial molecular communication based nanonetworks in network simulator 3, 3rd ACM International Conference on Nanoscale Computing and Communication (2016) 17–23.

[17] R. M. Fujimoto, K. Perumalla, A. Park, H. Wu, M. H. Ammar, G. F. Riley, Large-scale network simulation: how big? how fast?, 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (2003) 116–123.

[18] L. Felicetti, M. Femminella, G. Reali, P. Gresele, M. Malvestiti, J. N. Daigle, Modeling cd40-based molecular communications in blood vessels, IEEE Transactions on Nanobioscience 13 (3) (2014) 230–243.

[19] Network simulator 3 : https://www.nsnam.org/.

[20] E. Weingartner, H. Vom Lehn, K. Wehrle, A performance comparison of recent network simulators, IEEE International Conference on Communications (2009) 1–5.