Nano Communication Networks 1 (2010) 138–156

Contents lists available at ScienceDirect

Nano Communication Networks

journal homepage: www.elsevier.com/locate/nanocomnet

NanoNS: A nanoscale network simulator framework for molecular communications

Ertan Gul, Baris Atakan, Ozgur B. Akan *

Next-generation Wireless Communications Laboratory, Department of Electrical and Electronics Engineering, Koc University, 34450, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 30 July 2010
Accepted 5 August 2010
Available online 22 August 2010

Keywords:
Molecular communication
Molecular diffusion
Nanonetworks
Nanonetwork simulation tool
NS-2

ABSTRACT

A number of nanomachines that cooperatively communicate and share molecular information in order to achieve specific tasks is envisioned as a nanonetwork. Due to the size and capabilities of nanomachines, the traditional communication paradigms cannot be used for nanonetworks in which network nodes may be composed of just several atoms or molecules and scale on the orders of few nanometers. Instead, molecular communication is a promising solution approach for the nanoscale communication paradigm. However, molecular communication must be thoroughly investigated to realize nanoscale communication and nanonetworks for many envisioned applications such as nanoscale body area networks, and nanoscale molecular computers. In this paper, a simulation framework (NanoNS) for molecular nanonetworks is presented. The objective of the framework is to provide a simulation tool in order to create a better understanding of nanonetworks and facilitate the development of new communication techniques and the validation of theoretical results. The NanoNS framework is built on top of core components of a widely used network simulator (ns-2). It incorporates the simulation modules for various nanoscale communication paradigms based on a diffusive molecular communication channel. The details of NanoNS are discussed and some functional scenarios are defined to validate NanoNS. In addition to this, the numerical analyses of these functional scenarios and their experimental results are presented. The validation of NanoNS is shown via comparative evaluation of these experimental and numerical results.

1. Introduction

Nanotechnology enables the practical realization of nanoscale devices commonly referred to as nanomachines, the size of which ranges from 1 to 100 nm. The fabrication techniques in nanotechnology can be categorized into two main groups, i.e., dry and wet techniques. Dry techniques are mostly used for the fabrication of carbon nanotubes and nanowires whereas wet techniques are employed to fabricate nanoscale biological systems that already exist in natural aqueous media.

The realization of such biological nanomachines has been always worked by a large set of biochemical methods for many years. For example, biochemists already tackle with the production of functional cell components such as ribosomes that can be programmed for the synthesis of beneficial proteins. Doubtlessly, a network of communicating nanomachines can also be programmed to share nanoscale information over a nanonetwork so as to fulfill more complex tasks such as collaborative drug delivery, health monitoring, and biological or chemical attack detection [5].

In the literature, there are currently four main nanoscale communication techniques, i.e., nanomechanical, acoustic, molecular, and electromagnetic communication that may be used for the realization of nanonetworks [12]. In acoustic communication, information is carried via an acoustic signal. In nanomechanical communication, information flows through the mechanical connection of nanoscale devices. In electromagnetic communication, information is

* Corresponding author.
E-mail address: akan@ku.edu.tr (O.B. Akan).

1878-7789/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.nancom.2010.08.003

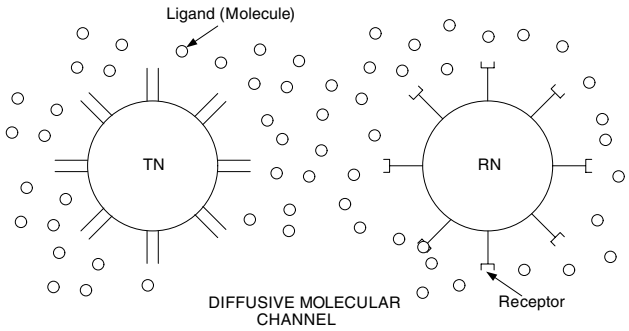

Fig. 1. Diffusive molecular communication model [2].

encoded into electromagnetic waves [4]. Molecular communication provides nanoscale communication among nanomachines using molecules as communication carriers [1].

Due to lack of the acoustic, nanomechanical, and eletromagnetic communication hardware that are sufficiently small and biologically compatible, molecular communication is the most promising and biologically compatible communication paradigm in order to enable frontier nanonetworks. Molecular communication can be viewed as an interdisciplinary research area that incorporates the domain of nanotechnology, biotechnology, and communication theory. Thus, the performance of molecular communication systems is affected by the physical laws in these domains. This brings many crucial research challenges, that must be efficiently addressed, into traditional communication systems.

Molecular communication systems can be categorized into three main groups according to their propagation mechanisms, i.e., diffusive, motor-based and gap junction-based molecular communication. Molecular communication through gap junctions, inspired from inter-cellular communication in nature, is presented in [23]. The system, in which calcium signaling is used as a communication carrier, is based on the diffusion of information molecules through gap junctions connecting the cytoplasm of two adjacent cells. The permeability and selectivity of gap junctions vary according to the constructive proteins of the gap junction [10]. In [22], the detailed design of a molecular channel with gap junctions is proposed. Signal switching, filtering and aggregation functionalities are controlled by adjusting the permeability and selectivity of gap junctions.

Molecular motor-based communication is inspired from intra-cell communication of real biological cells. Message molecules are transported by molecular motors and protein filaments that naturally exist in biomolecular linear systems [27]. There are two propagation types in molecular motor-based communication systems. In [21,20], carrier molecules are inserted into a vesicle embedded with channel proteins to transport the molecules. Vesicles, which encapsulate the information molecules, behave like communication interfaces between sender and receiver. They protect carrier molecules from environmental conditions and maintain compatibility between information molecules and environment.

In addition to gap junction-based and motor-based communications, molecules may also freely propagate in an aqueous medium to allow the diffusive molecular communication among nanomachines. Diffusive molecular communication is one of the fundamental communication mechanisms that is used by biological systems. This model is based on passive transport, which means that no external energy is required during transportation. In this model, the transmitter nanomachine releases molecules into an aqueous medium, then molecules propagate in the medium, and the receiver nanomachine captures some of these molecules that chemically react with its receptors as in Fig. 1. A simplified version of the ligand-receptor mechanism is analyzed and the channel capacity of the diffusive biochemical channel is estimated in [29]. A physical molecular channel is modeled and analyzed in [25]. Moreover, an information theoretical approach for molecular communication is presented in [2], where the molecular communication channel capacity between two nanomachines is analyzed. Molecular multiple access, broadcast and relay channel models for molecular communication are presented, and their capacity expressions are derived in [3].

In the current literature, the methodologies and tools used to analyze molecular communication and investigate its performance are extremely limited. In [17], unicast communication on microtubule topology is investigated. The probability of successful transmission is compared for the cases of diffusive and motor-based propagation. However, the topology of the environment is composed of only one single sender, one single receiver and the microtubule, which renders the nanonetwork topology oversimplified and unrealistic. In [16], delay characteristics of different microtubule topologies are analyzed. In [18], three types of molecular communication propagation approaches are defined. In a diffusive propagation system, microtubules are not used to connect sender and receiver. In a motor-based propagation system, sender and receiver are connected and an information molecule is directly transferred from the sender to the receiver nanomachine. In the hybrid propagation system, information molecules are released, then information molecules may propagate to the receiver by diffusing in the environment or binding microtubule which has an unstable behavior and moves through the bound microtubule. The information rates of propagation models are compared. Additionally, the noise analysis of these models is investigated in [19].

Despite promising progress performed in molecular communication, there is currently no molecular communication simulator which provides a practical and beneficial simulation suite to be used in the development of

Table 1
Traditional vs. molecular communication [15].

	Traditional communication	Molecular communication
Communication carrier	Electromagnetic waves	Molecules
Signal type	Electronic and optical (electromagnetic)	Chemical
Propagation speed	Light (3×10^8 m/s)	Extremely slow
Medium conditions	Almost immune	Affect communication
Noise	Electromagnetic fields and signals	Particles and molecules in medium
Encoded information	Voice, text, and video	Phenomena, chemical states, or processes
Other features	Accurate communication and high energy consumption	Stochastic communication and low energy consumption

network protocols for nanonetworks. Thus, it is essential to develop a simulator which allows the investigation of different nanonetwork topologies and molecular communication scenarios. The purpose of this paper is to introduce a molecular communication simulator, NanoNS, developed based on the open source network simulator (ns-2) [28] to yield a better understanding of nanonetworks and simplify the development of new communication techniques and validation of theoretical results. The NanoNS framework is comprised of simulation modules for molecular communication models based on a diffusive molecular communication channel. The structure of NanoNS is explained and numerical analyses of some functional scenarios are given. Moreover, the numerical analyses and the experimental results of these functional scenarios are compared in order to validate the NanoNS framework.

The rest of this paper is organized as follows. In Section 2, molecular communication models are explained including the details used in simulator design and employed algorithms. Design and implementation details of diffusive molecular communication simulator are described in Section 3. A numerical analysis of molecular communication is introduced in Section 4. The validation and performance evaluation of the NanoNS framework are presented in Section 5. Finally, concluding remarks are given in Section 6.

2. Diffusive molecular communication channel model

In this section, the characteristics of molecular communication are first briefly presented. Then, the diffusive communication model used in the simulator design is introduced.

Molecular communication significantly differs from traditional wireless communication. Some features may give the opportunity to develop more complex and efficient communication networks in the future, whereas some of them may limit the feasibility of molecular communication. Table 1 summarizes the major differences between traditional communication and molecular communication.

2.1. Diffusive communication

In nature, diffusive molecular communication is usually modeled based on a ligand-receptor binding mechanism. A sender biological entity releases ligand molecules to the medium. The released molecules diffuse in the environment and some of them bind to the receptor of another biological entity, i.e., receiver. The binding event is a chemical reaction between ligand and receptor molecules and allows the receiver entity to capture the ligand molecules. After the receiver entity captures the molecules from the molecular channel, it decodes the captured ligand molecules to fire an action potential. The diffusive molecular communication model is depicted in Fig. 1.

2.1.1. Diffusion process

The principles of molecular diffusion can be mostly investigated by means of the Brownian motion which is the random movement of the particles in gas or aqueous medium and a consequence of the constant thermal motion of atoms, molecules, and particles. Its source is the collision of molecules with the atoms, molecules, and particles of the medium.

Fick's Second Law states that the time rate of change in molecule concentration is proportional to the curvature of concentration and to the diffusion coefficient. In order to formulate the displacement of a single molecule, A, the concentration parameter is replaced with the probability density function (pdf) of molecule A's displacement in Fick's Second Law

$$\frac{\mathrm{d}p_A(r,t)}{\mathrm{d}t} = D_A \bigtriangledown^2 p_A(r,t) \tag{1}$$

where $p_A(r,t)$ is the pdf of molecule A's displacement, D_A is the diffusion coefficient, r is the displacement of molecule A, t is time. The solution of (1), given in [9,6], presents the pdf of molecule A's displacement in one direction, which has a Gaussian form as follows

$$G_S(\Delta x) \equiv \frac{1}{\sigma_A\sqrt{2\pi}} \mathrm{e}^{\left(-\frac{\Delta x^2}{2\sigma_A^2}\right)} \tag{2}$$

$$\sigma_A \equiv \sqrt{2D_A\Delta t} \tag{3}$$

where $G_S(\Delta x)$ is a normalized Gaussian with zero mean and standard deviation σ, which is equal to step length of molecule A. The displacements of each direction (x, y, z) are independent occurrences. Hence, the pdf of molecule A's total displacement is estimated as follows

$$p_A(r+\Delta r, t+\Delta t) = G_{S_A}(\Delta x) G_{S_A}(\Delta y) G_{S_A}(\Delta z). \tag{4}$$

(3) formulates the step length of molecule A in one dimension. The total displacement of molecule A is defined as follows

$$\Delta r = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}. \tag{5}$$

Therefore, the total displacement of molecule A is equal to $\sqrt{6D_A\Delta t}$.

2.1.2. Reaction process

There are two types of reaction modeling approaches: a deterministic approach and a stochastic approach. A deterministic process assumes that the molecular reactions are continuous and completely predictable processes and modeled by deterministic differential reaction-rate equations. However, a reaction system is not a continuous process as the number of molecules changes discretely, and it is not possible to predict the exact molecular population levels without knowing and tracking the exact position and velocity of the molecules.

Here, in order to devise a realistic channel model in molecular communication, we follow a stochastic method commonly known as Gillespie's method and introduced for describing the stochastic simulation of coupled chemical reactions in a well-mixed medium [13].

In Gillespie's method, N kinds of molecules which are able to react with each other through M reaction channels are assumed to be uniformly mixed in a fixed volume V. These M reactions are formulated as follows:

$$\left.\begin{array}{c} R_1: S_1 + S_2 \rightarrow C_1 \\ \vdots \\ R_M: S_{N-1} + S_N \rightarrow C_M \end{array}\right\} M \text{ Reaction channels} \tag{6}$$

where $S_i, i \in \{1, \ldots, M\}$ are the molecules that react with each other, $C_i, i \in \{1, \ldots, M\}$ is the output of the R_ith reaction channel.

In order to determine which reaction occurs next and when it occurs, $P(\tau, \mu)$, is defined as the probability that given the state $X_i(t)$ $(i = 1, \ldots, N)$, which denotes the number of molecules of the species, at the time t, the next reaction in the volume will occur in the infinitesimal time interval $(t + \tau; t + \tau + \mathrm{d}\tau)$ and will be an R_μ. The reaction probability of R_μ is the case that no reaction will occur during τ after reaction μ has occurred. Hence, $P(\tau, \mu)$ equals the product of $P_0(\tau)$ and the propensity value of R_μ, a_μ, which gives the probability of occurrence of reaction μ in $(t, t + \tau)$ [13], i.e.,

$$P(\tau, \mu)\mathrm{d}\tau = P_0(\tau)a_\mu \mathrm{d}\tau. \tag{7}$$

The propensity function has a specific mathematical form given as

$$a_\mu = h_\mu c_\mu \tag{8}$$

where c_μ is the rate constant which depends on the radius of the molecules, their average velocities and individual masses. h_μ is the quantity of reaction input combinations. For instance, h_μ value of reaction "$R_\mu: S_1 + S_2 \rightarrow$ anything" equals X_1X_2.

After applying derivations in [13], $P_0(\tau)$ is obtained as

$$P_0(\tau) = \mathrm{e}^{\left(-\sum_{\mu=1}^{M} a_\mu \tau\right)}. \tag{9}$$

Substituting (9) into (7) yields

$$P(\tau_R, \mu) = \begin{cases} a_\mu \mathrm{e}^{(-a_0\tau_R)}, & 0 \le \tau_R < \infty, \\ & \mu = 1, \ldots, M, \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where $a_\mu = h_\mu c_\mu$ and $a_0 = \sum_{\upsilon=1}^{M} a_\upsilon$. As shown in (10), $P(\tau_R, \mu)$ is equal to the multiplication of two separate functions, $f(\mu)$ and $g(\tau)$. τ and μ can be obtained using

$$\tau = \frac{1}{a_0} \ln \frac{1}{r_1} \tag{11}$$

$$\sum_{\upsilon=1}^{\mu-1} a_\upsilon < r_2 a_0 \le \sum_{\upsilon=1}^{\mu} a_\upsilon \tag{12}$$

by assigning two random variables, r_1, r_2, in interval [0, 1] [13].

2.1.3. Reaction-diffusion model

There are several stochastic approaches to model biochemical activities. We consider a mesoscopic level archetype, which treats molecules discretely, however, it does not track positions in a compartment or within a subvolume, to model a molecular channel of the simulator. Since our channel model needs both reaction and diffusion capabilities, we model our channel to develop with a reaction-diffusion paradigm.

In this model, diffusion and reaction are distinct events. The basis of diffusion is the multiparticle lattice gas automata (LGA) algorithm [26]. In the multiparticle LGA algorithm, the medium is divided into lattice sides. Consequently, the inhomogeneity of the system is reduced to the lattice volume. Each lattice site holds a discrete number of uniformly distributed particles. Molecules perform random walks on the lattices and are distributed to 6 neighbor lattices randomly. If a small number of molecules exist in the lattice, molecules move individually to neighbor lattices. If the number of molecules is larger than 60 [26], molecules move in bulk to a lattice according to a Gaussian distribution. In the algorithm, the exact position of a molecule is not necessary, however, only the lattice position of the molecule is required. Thus, lattice coordinate system is utilized in the simulator.

Normally, every species has a particular diffusion coefficient. The diffusion time step, τ_{D_S}, of each species is calculated as follows

$$\tau_{D_S} = \frac{1}{2d} \frac{\lambda^2}{D_S} \tag{13}$$

where D_S is diffusion coefficient of the species, λ is the length of each lattice, d is the dimension of the simulation medium.

As indicated, a reaction process is distinct from a diffusion process. Reaction events occur between diffusion events. It is assumed that chemical reactions are local events. Therefore, the Stochastic Simulation Algorithm (SSA) is applied to perform reaction events in each lattice side. As mentioned in Section 2.1.2, SSA can only be applied to well-mixed volumes. Hence, the length of the lattice sides should not be long in order to keep homogenization of lattice volume.[1] In NanoNS, the lattice length is a user-defined input parameter. If the length of lattice

[1] The analysis of the optimal lattice size is beyond the scope of this work. An analysis of optimal lattice size is presented in [26].

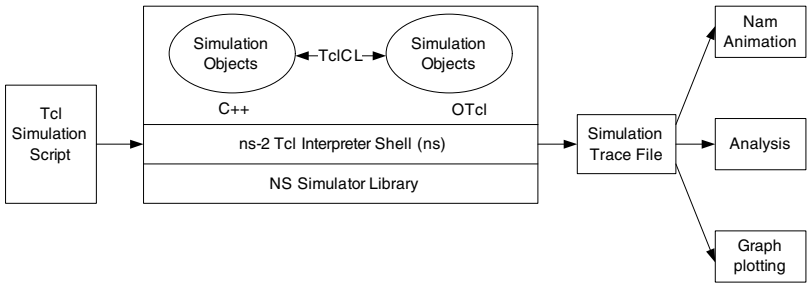

Fig. 2. ns-2 simulation architecture.

size is not assigned, the simulator uses a default lattice size, which is 100 nm.

The diffusion and reaction events in the molecular channel are entirely independent from the communication of nodes. Even though there is no communication between nanonodes, the reaction-diffusion events are performed in the background. In fact, a molecular channel has individual commands to start and stop channel simulation as explained in Section 3. As expected, this independency is unilateral in the simulator. The communication between nodes completely depends on reaction-diffusion events. The propagation of information molecules and reception of the carrier molecules are the crops of reaction-diffusion events. Therefore, nodes cannot communicate with each other unless reaction-diffusion events are performed by the molecular channel.

3. Simulation of diffusive molecular communication

In this section, the details of NanoNS, a diffusive molecular communication simulation tool developed on top of ns-2, are introduced. The NanoNS components follow the diffusive molecular communication model given in Section 2.1. A brief introduction of ns-2 is first given. Then, an overview of a diffusive molecular communication model is explained. Finally, the implementation details of diffusive molecular communication components in NanoNS are presented.

3.1. Overview of Network Simulator (NS-2)

NS-2 is an open source discrete event-driven network simulator which provides the simulation of several networking layers, e.g., transport, routing, and multicast, protocols over wired and wireless networks [28]. Moreover, it provides a development environment to model promising networks which are different from traditional communications as [14]. Due to these features of ns-2, it is chosen as our simulation environment.

NS-2 is an object-oriented simulator which is written in C++ and an object-Tcl (OTcl) interpreter. C++ is generally utilized in complicated protocol implementations due to its fast execution. On the other hand, OTcl is used for simulation configuration since it can be changed very quickly. OTcl interpreter, which is the user interface of ns-2 as shown in Fig. 2, provides the scheduling of the simulation, setting the network's topology, configuring network parameters and plumping the components of the network [11]. In NanoNS, the *ns-mol.tcl* library has been developed since we designed a new node structure called NanoNode, which is plumped in *ns-mol.tcl*, besides, new network components, parameters and methods for molecular communication are defined in this file.

3.2. Overview of diffusive molecular communication model

Diffusive molecular communication components of NanoNS provide a communication mechanism for nanomachines to communicate over a short distance (nm-m) via the propagation of molecules in an aqueous medium. Its main components consist of sender nanomachines, receiver nanomachines, and carrier molecules.

We design `NanoNode` class in order to simulate nanomachines. The sender nanonode sends the molecules to the receiver nanonodes by releasing carrier molecules into the environment. We define a `Molecule` class that has the features of carrier molecules such as proteins, ions or DNA. Data is encoded into Molecule objects. To manage the diffusion of molecules in the environment, a `Diffusion` class is created. We choose the diffusion model proposed in [26]. In this study, the environment is assumed to be composed of a virtual lattice structure. Thus, we design the `Lattice` class for lattice sides, and reallocate every object according to lattices. We use the coordinate system, `MolPosition`, the unit element of which is a lattice. Owing to diffusion, molecules walk randomly from one lattice side to another. We designed the `Diffusion` and `Randomizer` classes in order to accomplish this requirement.

When carrier molecules reach the receiver nanonode, the receiver nanomachine detects and receives the carrier molecules by binding them. In order to construct an apprehend mechanism, reaction channel objects that define chemical reactions and a reaction object which is able to apply SSA are required. Therefore, we design the `Reaction` and `ReactionChannel` classes.

A class that provides a shared medium for nanonode and manages the propagation and a binding mechanism is also required for the simulator. Hence, we designed the diffusive molecular channel class, `MolChannelDiffusion`. We propose the `MolecularEnvironment` class to keep all data about molecular communication channel such as lattice sides, reaction channels and species. The `MolChannelDiffusion` class accesses the entire environment data through a pointer attribute of `Molecular Environment` class.

3.3. Implementation of diffusive molecular communication

In this section, we focus on some of the key components in the implementation and take a detailed look at each component of diffusive molecular communication system in NanoNS. The structures of the nanonode and user interface commands are built up in the OTcl programming language while almost all of the functionalities are implemented in C++.

3.3.1. NanoNode: constituent of molecular communication

The basic structure of molecular communication is NanoNode in ns-2. The `NanoNode` class is derived from the `Node` class. The `Node` class is a standalone class in OTcl [11]. Almost all the components of the `Node` class are TclObject instances. NanoNode is a split object which has some additional functionalities like releasing-binding carrier molecules on a molecular channel that simulate an aqueous environment and *volume occupation* on environment. A Nanonode is created in the lattice space by a volume occupation facility. NanoNode creation is performed at the beginning of the simulation when the simulation time is zero. NanoNodes are assumed to be immobilized. All functions of the `NanoNode` class are implemented in C++ except the construction of the `NanoNode` itself.

NanoNodes are also built up in the lattice space. The shape of NanoNodes is considered as spheres. Consider the function that calculates the distance between the membrane lattice (x_m, y_m, z_m) and the central lattice (x_c, y_c, z_c):

$$D(x_m, y_m, z_m) = \left[(x_c - x_m)^2 + (y_c - y_m)^2 + (z_c - z_m)^2\right]^{1/2}. \quad (14)$$

We restrict the size of the membrane lattice as follows

$$D(x_m, y_m, z_m) \leq R < D(x_m, y_m, z_m \pm 1) \quad (15)$$

where (x_m, y_m, z_m) is the coordinate of the membrane lattice, R is the radius of the nanomachine defined by the user. The lattices between the *membrane* lattice and the central lattice are *cytosol* lattices.

There are some differences between the structures of the NanoNode and the Node in ns-2. Like all ns-2 nodes, the NanoNode has an *entry* point that does not designate to regular classifiers different from the other nodes in ns-2. A molecular routing agent, which is not in the scope of this paper, is assumed to check the direction of MolecularData and orients it in the current implementation of the NanoNS. If incoming MolecularData is destined to this node, it is directed to *Src/Sink Agent*. Additionally, we cancel the port classifier, *demux_*, which exists in the base *Node* class, since there is no equivalent mechanism in current molecular communication models. The structure of a NanoNode object is shown in Fig. 3.

As mentioned before, an entry point is connected to the routing agent. When the routing agent receives a MolecularData, it checks the direction of it. If the direction field indicates down, it sends molecules to link the layer. Otherwise, the routing agent transfers MolecularData to the sink. In the `NanoNode` class, there are pointers which show additional objects. Each NanoNode object contains a

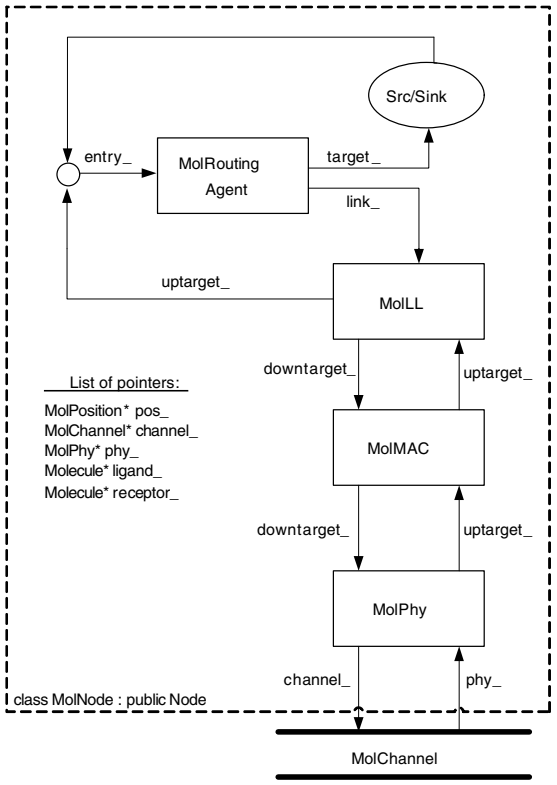

Fig. 3. Schematic of a NanoNode.

position object that points to the center of itself, a ligand and receptor molecule pointers. It also keeps the radius of the nanonode as a static variable and the number of receptor molecules of the nanonode. Likewise, there are some pointers which point to the network components in NanoNode. Each NanoNode contains a pointer to channel (channel_) and its physical layer (phy_). Besides, the NanoNode contains a static integer variable, *type_*, to keep the type of the molecular communication system. The type of communication channel system is distributed from this variable to the other classes in the NanoNS.

3.3.2. MolecularData: Packet unit of molecular communication

`Packet` class instances are the fundamental exchange units between objects in the ns-2 simulation [11]. The `MolecularData` class is derived from the `Packet` class. The `MolecularData` class, which requires to be inherited from the `Event` class, passes through the network structures and provides the communication between network components. Its building structure is MolecularDataUnit objects. Information is encoded into the number of molecules. A MolecularData object contains MolecularDataUnit objects in order to keep the number of molecules. In NanoNS, information is encoded in terms of the molecule concentration as given in [2]. Therefore, individual molecules do not carry any information.

We assume that ligand molecules in the simulator are spheres. Thus, according to *Stokes–Einstein* formula, [24]

the diffusion coefficient of a ligand molecule is given as follows

$$D = \frac{kT}{6\pi r\eta} \tag{16}$$

where k is Boltzmann constant, T is the temperature of the medium, r is the radius of the sphere and η is the viscosity of the aqueous medium.

3.3.3. Diffusion: Propagation system

The `Diffusion` class inherits from a `TclObject` class since the type of diffusion object is determined by the user via the *diffusion* parameter in the node configuration interface. Currently, only one type of Diffusion model is implemented; nevertheless, new diffusion models can be implemented and joined into the simulator.

The `Diffusion` class is also derived from the `Handler` class. When an event is ready, the handle method of the Handler derived class, `Diffusion`, is called. In the handle function of the `Diffusion` class, a diffusion event is triggered. The `Diffusion` class has an interface only with the `MolChannelDiffusion` class. The minimum interface with other components brings modularity to the diffusion object.

The basis of these diffusion processes is the multiparticle LGA algorithm [26]. In this algorithm, the medium is divided into lattices. Each lattice site holds a discrete number of uniformly distributed particles. As a result of diffusion, molecules perform a random walk on the lattices. Molecules are distributed to neighbor lattices randomly. The exact position of a molecule is not necessary, only the lattice position of the molecule is needed. Every species has a particular diffusion coefficient. The diffusion time of each species is calculated with the lattice length and diffusion coefficient. If the number of molecules existing in the lattice is less than the boundary value defined by the user [26], the molecules move individually to a neighboring lattice. If the number of molecules is larger than the boundary value, molecules are moved in bulk to a lattice according to Gaussian distribution.

As mentioned before, *membrane* lattices act like a wall. Molecules coming from outside the nanomachine cannot enter *cytosol* lattices. To simulate *membrane* lattices like a wall, reflective boundary conditions are used. If molecules try to diffuse into *cytosol* lattices, these molecules reflect from the *cytosol* lattices and stay in their local lattice.

Every lattice has two different attributes in order to hold the type and number of the molecules. One of them keeps the current state, which keeps the actual molecule number in the lattice; while the other one keeps a temporary state, which keeps the molecule numbers in a diffusion state. During the diffusion process, the molecules in the current state are distributed to temporary states of neighboring lattice sides. At the end of the diffusion process, the temporary state is transferred to the current state and reset.

3.3.4. Reaction: Capture mechanism

The `Reaction` class is derived from the `TclObject` and `Handler` classes like a Diffusion object. Three types of reaction models, i.e., *NoReaction*, *Berg*, *Gillespie*, can be selected via the Tcl script. In the NoReaction option, a ligand molecule is captured whenever it collides with a receiver nanomachine. The Berg scheme implements the reaction model given in [7]. The Gillespie selection realizes SSA as described in Section 2.1.2. Some adaptations and assumptions added to SSA are as follows:

- We have a limited time interval to apply the algorithm,
- If a reaction occurs in a *vitro* type lattice, the number of receptor and ligand molecules decreases,
- If a reaction occurs in a *membrane* type lattice, the number of receptor molecules does not change; only number of ligand molecule decreases,
- There cannot be a chemical reaction inside a *cytosol* lattice.

The identity element of the reaction mechanism is the reaction channel. The `ReactionChannel` class presents the chemical reaction that is defined by the user via the *set-reaction* method of a molecular channel. In the simulator, only the second order chemical reactions can be defined. The types of molecules and diffusion coefficient rates define a reaction channel. The class diagram of a reaction model is depicted in Fig. 4.

3.3.5. MolChannelDiffusion: Challenge of molecular communication

The challenging issue of a molecular communication simulator is the modeling of a molecular channel. The `MolChannelDiffusion` class is derived from the `Channel` class. `MolChannelDiffusion` accesses to environmental data through a pointer that points to the `MolecularEnvironment` class. The `Molecular Environment` class keeps all the data about the molecular channel like a repository of simulator. The simulator environment contains nanomachines, species, reaction channels, and lattices.

Lattice objects are the members of the `Molecular Environment` class which keeps the entire lattice medium. There are three kinds of lattices for diffusive molecular communication. These lattice types are *cytosol, membrane, vitro*. As mentioned before, nanonodes are constructed with *membrane* and *cytosol* lattices. The lattices in which the propagation of molecules occurs are expressed as *vitro* type lattices. *Vitro* type lattices behave like the outsides of nanonodes. Molecules cannot diffuse into *cytosol* lattices from *membrane* and *vitro* lattices. The lattice side is supposed to keep the molecules in. The lattice class fulfills this requirement with two maps, which keep the molecule type and the number of molecules. One of them holds the current state, the other keeps the temporary state of the lattice.

In addition to keeping the lattices and lattice spaces, the `MolecularEnvironment` class retains species and reaction channels in vectors. Other classes can reach and use these data at any time of the simulation. It also keeps environmental attributes, which affect the result of the simulation, i.e., viscosity, temperature and lattice side length.

In diffusive molecular communication, the molecular channel keeps all nanonodes in the simulator. The diffusive molecular channel is associated with diffusion and reaction

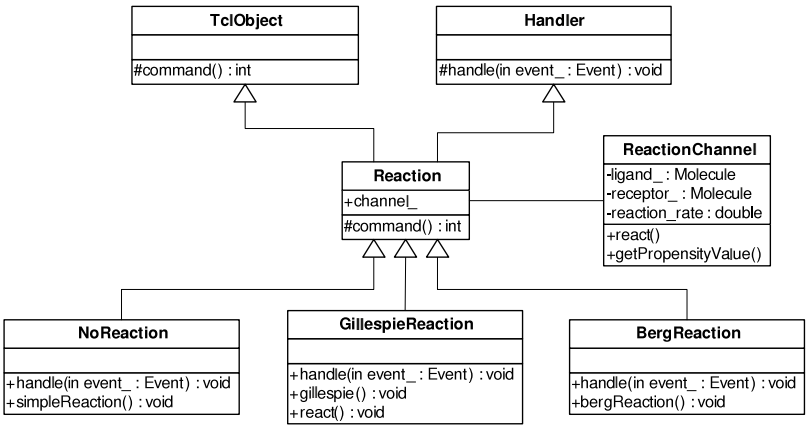

Fig. 4. Class diagram of reaction model.

classes and "has a" relationship with the molecular environment class as depicted in Fig. 5. In order to obtain modularity in the simulator, the interface between components is kept at a minimum as much as possible.

The molecular channel also provides a shared medium for all nanonodes to communicate with each other. The diffusion and reaction events in the molecular channel are entirely independent from the communication of nodes. Although they are independent from the communication of nodes, communication between nodes is affected by these events. Molecules, which can be expressed as information, are transported and captured by these events.

3.3.6. Molecular trace support

There are several ways to collect trace data on a simulation. Generally, trace data is either displayed directly during execution of the simulation, or more commonly, stored in a file to be post-processed and analyzed. In the current ns-2 simulator, there is a `Trace` class to monitor capabilities. A trace records each individual event as it arrives, departs, or is dropped at a link or queue. Trace objects are configured into a simulation as nodes in the network's topology. They are inherited from the `Connector` class. Thus, they can be easily plugged into a network stack.

Trace files used for molecular communication have the same structure as conventional ns-2 tracing. The class `MolTrace` derives from the class `Trace`. Molecular trace objects are incorporated into network stack if they are enabled in the Tcl script. Although the structure trace mechanism is similar with conventional ns-2 tracing, their formats are totally different, due to the communication carrier and channel structure incompatibility. Hence, the information to be traced is changed. The format of the molecular trace is given in Table 2.

There are three types of molecular traces. These are the transmitter (Trace/Mol/Trans), the receiver (Trace/Mol/Recv) and error traces (Trace/Mol/Error). Transmitter ones are utilized to trace sent molecules. The type of this trace is *t*. Since the receiver of the molecular trace is not known, the receiver node id, receptor molecule and position of receiver nanonode fields are set to "−1", "?" and "−999, −999, −999", respectively. Receiver traces are utilized to trace accepted molecules. The type of this trace is *r*. When molecules are received, MolecularDataUnit instances are constructed and then MolecularData is created by them. If the number of possible transmitter nodes of a received molecule is more than one, the sender node id and position of sender nanonode fields are set to "−1" and "−999, −999, −999", respectively. Error traces are utilized to trace molecules which are corrupted by the error model. The type of this trace is *e*. The error traced molecules are dropped due to errors defined by the error model.

3.3.7. Molecular error model

The `MolErrorModel` class inherits from the `Error Model` class. These error models, causing molecules to be corrupted according to various probability distributions, are simple and do not necessarily correspond to any experience on an actual molecular channel. Each nanonode can insert a given statistical error model either over outgoing or incoming molecular channels. More specifically, the instantiated error model is inserted between mac and phy modules for an incoming link and between the link layer and mac modules for the outgoing link as depicted in Fig. 6. For the outgoing link, the error module would be pointed by the *downtarget_* of the above link layer module while for the incoming link, it would be linked by the *up-target* pointer of the below phy module. Thus, the *target_* of the error module points to the mac layer in both cases.

In the molecular error model, every molecule is checked according to the rate and distribution of a random variable. If no error is obtained from the error model for a molecule, the molecule continues its way. Otherwise, it is directed to a NULL agent. If the tracing facility is enabled, information about error molecules is written into the trace file.

The molecular error model can be applied to molecular data in the molecule or the MolecularData level. This implies that error models can check erroneous cases for every molecule or MolecularData. This capability is determined by the user with the *unit* command. If the Molecule option is chosen as the unit of the error model, each molecule is inspected whether it is corrupted or not. Otherwise, each MolecularData is examined.

The reason of placing error models, i.e., incoming and outgoing, over two different locations is that the outgoing error model causes all the receivers to receive the packet

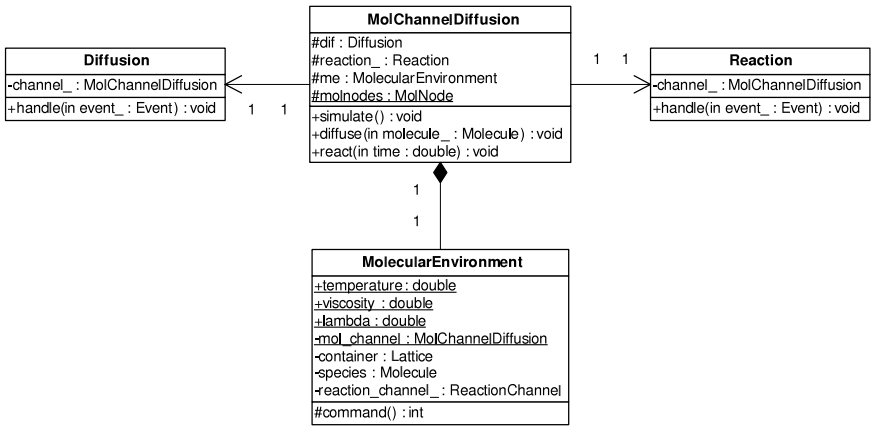

Fig. 5. Class diagram of diffusive molecular channel.

Table 2
The format of the molecular trace.

1	Time	2	*s*	3	*d*	4	*lig*	5	*rec*	6	*sp*	7	*dp*	8	*a*	9	*cp*	10
Description of fields																		
[1]	Type of the trace																	
[2]	Time of the trace																	
[3]	No of the sender NanoNode																	
[4]	No of destination NanoNode																	
[5]	Type of the ligand molecule																	
[6]	Type of the receptor molecule																	
[7]	Molecular position of sender NanoNode																	
[8]	Molecular position of destination NanoNode																	
[9]	Number of captured molecules																	
[10]	Molecular position of the lattice where the ligand molecule is captured																	

suffering the same degree of errors, since the erroneous situation is determined before phy layer releases the molecules to the molecular channel. On the other hand, the incoming error module lets each receiver to get the molecules corrupted with different degree of error, since the error is computed independently in each error module.

3.4. Diffusive molecular communication links

In this section, we describe how MolecularData moves up and down the stack, and the key points to note at each layer. Here, the composite structure of diffusive molecular communication links is considered as a *network stack*. Fig. 6 provides a detailed look at how molecular communication links are constructed.

When MolecularData departs from a NanoNode, it passes to the class `MolLL`. If an outgoing error model is defined for NanoNode, MolecularData passes to the molecular error model. After the error model is applied to MolecularData, the *recv()* method of the `MolMAC` class is called. Unless an outgoing error model is declared, the MolLL object sends MolecularData to the MolMAC object. A basic static allocation-based (e.g., time-division multiple access) molecular MAC protocol in order to serve as a basis for the future MAC protocol implementations is devised and incorporated in the MolMAC layer. Next, the packet is sent to MolPhy.

The `MolPhy` class inherits from the `Phy` class. As usual, MolPhy provides a connection between MolMAC and the molecular channel. In addition to this, MolPhy is in charge of releasing molecules to the molecular channel and capturing the molecules from the molecular channel. When MolPhy receives a MolecularData instance, it checks the direction of MolecularData. If its direction is down, it releases the molecules by calling the *sendMolecules()* of the molecular channel and calls the *recv()* method of the molecular channel. In *sendMolecules()* of the diffusive molecular channel, molecules are put into the *membrane* lattices of the nanonode. The release of molecules is the beginning of its own time slot.

The outgoing molecules are finally received by MolChannelDiffusion. As described earlier, diffusion and reaction processes are independent from the sending and receiving cycle. It can be supposed that the received molecules are inserted into a reaction-diffusion cycle. When a molecule is captured, MolChannel creates MolecularData using the information of the captured molecule. The direction of MolecularData is set to up and this MolecularData is sent to physical layer of the receiver nanonode.

When the *recv()* method of MolPhy is called, MolPhy checks the direction of MolecularData. If the direction is up, it fills the *data* field of MolecularData with the type and number of captured molecules in that interval. Then, MolPhy sends MolecularData to MolMAC if no incoming error model is declared. Otherwise, MolecularData is sent to the molecular error model object. If the MolecularData arrives safely to MolMAC, it next passes to the MolLL object. Finally, the packet moves to the entry of the NanoNode.

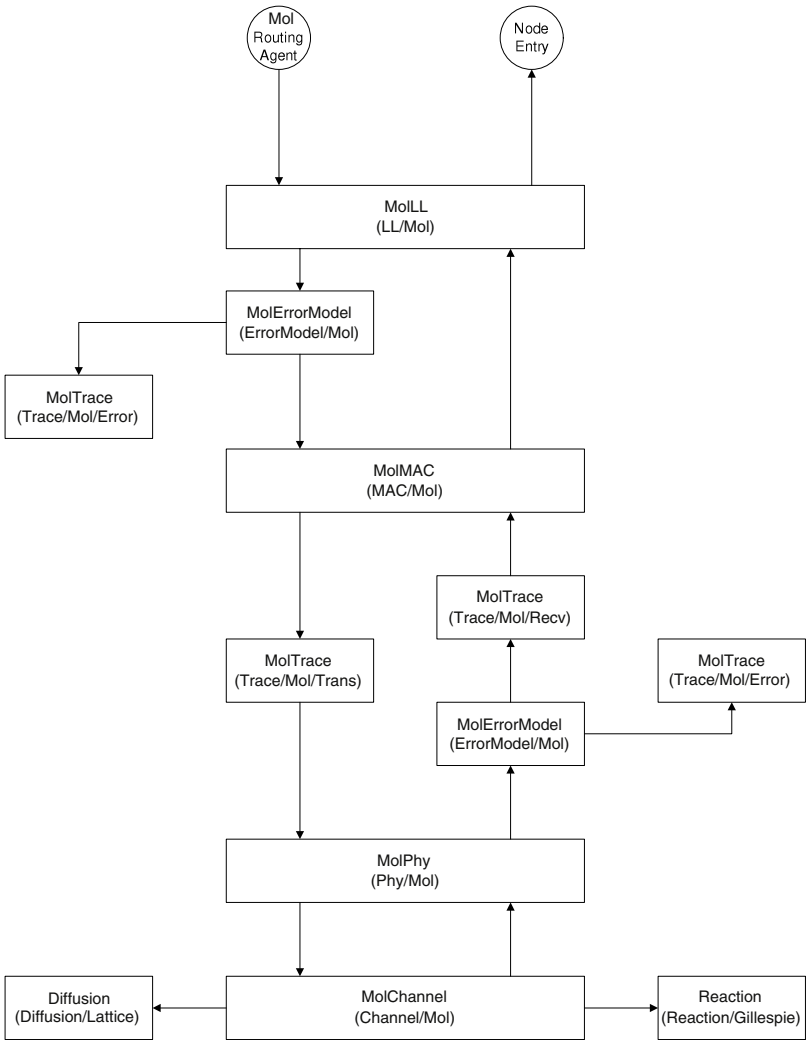

Fig. 6. Detailed look at network interface stack.

4. Numerical analysis of molecular communication

In this section, we present the numerical analysis of molecular communication. We define some functional scenarios to facilitate the validation of NanoNS. We make numerical analyses and formulate these scenarios. In the next section, we demonstrate the validation of NanoNS by comparing the results of numerical analyses and NanoNS.

4.1. The description of scenarios

We consider two types of scenarios that are called *Diffusion-Scenario* and *Gillespie-Scenario* to analyze, specific functionalities of molecular communication and then validate the NanoNS framework.

The aim of Diffusion-Scenario is to provide a feasible context to analyze diffusion functionality. In the Diffusion-Scenario, we cover the burst mode, in which the source emits the information molecules instantaneously into a stationary medium. A point source exists in the center of a sphere plate. There is a receiver nanomachine the center of which is on the surface of the outer sphere. When an information molecule contacts with the receiver nanomachine, it is captured. Otherwise, if the molecule touches the outer shelter sphere, it is destroyed.

Owing to the fact that an information molecule is captured when it collides with the receiver nanomachine's surface, the surface of the system should be taken into account in the analysis. However, as the concentration function is only valid for volume calculations, it is assumed that there is a shelter surrounding the surfaces. The thickness of the shelter and nanomachine, denoted as d, is infinitesimal.

In the Gillespie-Scenario, we reconfigure our medium to realize SSA. The ligand and receptor molecules are released in a well-stirred volume, e.g., the lattice, at the beginning of the simulation. When a reaction occurs between ligand and receptor molecules, the number of receptor molecules remains the same, whereas the number of ligand molecules decreases and an information molecule is obtained.

4.2. Numerical analysis of diffusion mechanism

Suppose that Q molecules are released instantaneously from the origin of a Cartesian coordinate system at $t = 0$. The spatial density of concentration in space through time is given as [8]:

$$U(x, y, z, t) = \frac{Q}{(4\pi Dt)^{3/2}} e^{-r^2/4Dt} \tag{17}$$

where $r^2 = x^2 + y^2 + z^2$ and D is the diffusion constant in (16).

In (17), the instant concentration of a volume unit is calculated. However, we need to integrate the concentration of molecules over time in order to evaluate the captured molecules. The integration of concentration over time is given as

$$\begin{aligned} U(r, t) &= \int_0^t \frac{Q}{(4\pi D\tau)^{3/2}} e^{-\frac{r^2}{(4D\tau)}} d\tau \\ &= -\frac{Q}{4D\pi r} erf\left(\frac{r}{\sqrt{4D\tau}}\right)\Big|_0^t \\ &= \frac{Q}{4D\pi r} erfc\left(\frac{r}{\sqrt{4Dt}}\right) \end{aligned} \tag{18}$$

in which $erfc(x)$ is a complementary error function. If propagation of the molecules is observed for a long time, the value inside the $erfc()$ function approaches 0. Hence, the density function of captured molecules approaches the limit

$$U(r) = \frac{Q}{4D\pi r}. \tag{19}$$

In order to calculate the volume of a thin shelter, consider two spheres. The radius of first sphere is $a + d$, and the radius of other sphere which is inside of the first one is a. The subtraction of spheres' volumes gives the volume of the shelter. Since our source is a point source and the variable of (17) is r, the volume of the sphere has to be expressed in r. In order to calculate the volume of the sphere in terms of r, the sphere is divided into small flat slices, the radius of which is $\sqrt{a^2 - r^2}$. Then, the slices are summed up as follows

$$V = \int_{-a}^{a} \pi(a^2 - r^2) dr \tag{20}$$

which estimates the volume of sphere by taking the center of it as a reference point. However, our reference point is R away from the sphere's center. Hence, (20) should be re-expressed as

$$V = \int_{R-a}^{R+a} \pi(a^2 - (r - R)^2) dr. \tag{21}$$

The total amount of molecules existing in the sphere during time t is estimated as follows

$$C(r, t) = \int_y^x \frac{Q}{4D\pi r} erfc\left(\frac{r}{\sqrt{4Dt}}\right) \pi(a^2 - (r - R)^2) dr \tag{22}$$

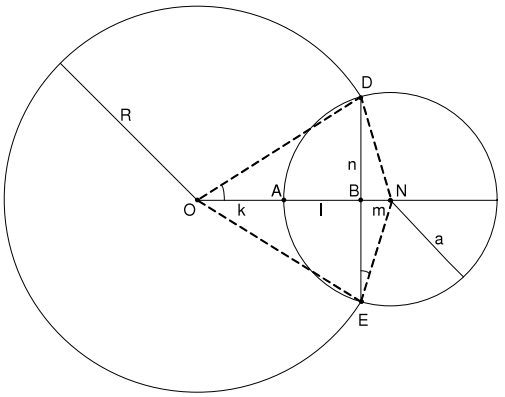

Fig. 7. The representation parameters of the shelter sphere and the nanomachine used.

where the limits of (22), $[x, y]$, are determined according to the visible sides of the nanomachine by a point source.

As projected in Fig. 7, a point source can only view the (DE) surface of a nanomachine. In order to determine the limits of (22), we need to find l. As seen in Fig. 7, $(\widehat{DON}) = 2 \times (\widehat{DEN}) = \alpha$. Hence,

$$\tan(\widehat{DON}) = \frac{2\tan(\widehat{DEN})}{1 - \tan^2(\widehat{DEN})} \Rightarrow \frac{2m}{n^2 - m^2} = \frac{1}{R - m} \tag{23}$$

which yields

$$m = \frac{a^2}{R} \Rightarrow l = a\left(1 - \frac{a}{R}\right). \tag{24}$$

To find the molecules captured by a thin shelter, the ones captured by a bigger sphere should be subtracted from the ones captured by the smaller sphere. Moreover, only the visible side of sphere should be considered. Hence, the total number of molecules captured by the shelter is equal to

$$\begin{aligned} C(r, t) = \frac{Q}{4D}\Bigg(&\int_{R-a-d}^{R-a^2/R} erfc\left(\frac{r}{\sqrt{4Dt}}\right) \\ &\times \frac{((a + d)^2 - (r - R)^2)}{r} dr \\ &- \int_{R-a}^{R-a^2/R} erfc\left(\frac{r}{\sqrt{4Dt}}\right) \frac{((a)^2 - (r - R)^2)}{r} dr\Bigg). \end{aligned} \tag{25}$$

The value calculated from (25) is not the final result. It should be normalized with the whole space which is a summation of the (25) and the surrounding sphere. The molecules captured by the surrounding sphere are given by

$$S(r, t) = \beta \int_{R-d}^{R} \frac{Q}{4D\pi r} erfc\left(\frac{r}{\sqrt{4Dt}}\right) 4\pi r^2 dr \tag{26}$$

where β, i.e., the ratio between the visible area from the point source and the entire area of the surrounding sphere, is

$$\beta = 1 - \frac{a}{2R}. \tag{27}$$

After a long time period, the concentration of molecules over time approaches a limit value. The limit values should be used in the calculation of a normalization factor. As a result, the normalization factor, NF, is given as

$$NF = \lim_{t \to t_{sat}} \frac{1}{C(r,t) + S(r,t)} \tag{28}$$

in which t_{sat} is the saturation time in which the concentration of molecules approaches the limit value.

The multiplication of (25) and (28) provides the probability of crashing a molecule into the nanomachine with time:

$$P_{Col}(t) = \frac{C(r,t)}{\lim\limits_{t \to t_{sat}} (C(r,t) + S(r,t))}. \tag{29}$$

The multiplication of $P_{Col}(t)$ with the total number of carrier molecules, Q, gives the number of collided molecules, $M(t)$, over time, i.e.,

$$M(t) = P_{Col}(t)Q. \tag{30}$$

To calculate the total number of collided molecules, we need to consider that a long period of time passes, i.e., $t \to \infty$. Hence, the probability of collision of a molecule after a long period of time is

$$P^*_{Col} = \frac{C(r,\infty)}{C(r,\infty) + S(r,\infty)}. \tag{31}$$

As in (19), when $t \to \infty$, the *erfc* function approaches 1. Hence, the solution of P^*_{Col} is as follows (32) is given in Box I. The total number of collided molecules is given as (33) in Box II. Although the diffusion coefficient, D, is one of the terms that effects the evaluation of the analytical diffusion mechanism as in (17), it has no impact on the total number of collided molecules. The numerical results obtained in this section are compared with experimental data in Section 5.1.

4.3. Numerical analysis of the Gillespie model

The goal of this section is to introduce a theoretical investigation of SSA implementation of our reaction system in the NanoNS. The chemical deterministic description of the system is given by the following ordinary differential equation

$$\frac{\partial C_L(t)}{\partial t} = kC_L(t)C_R \tag{34}$$

where $C_L(t)$ is the concentration of ligand molecules and C_R is the concentration of receptor molecules. The concentration of ligand molecules changes with time, whereas the number of receptor molecules stays constant.

In (34), the variables are in terms of concentration. However, the input of the analysis is in terms of the number of ligand and receptor molecules. Therefore, we need to map the concentration values into the number of molecules. The relation between the concentration and the number of molecules can be expressed as

$$C = \frac{n}{NV} \tag{35}$$

where n is the number of molecules, N is the Avogadro constant and V is the volume in units of *liter*. Hence, the solution of (34) is given as

$$n_L(t) = n_L(0)\mathrm{e}^{-\frac{kn_R t}{NV}} \tag{36}$$

where $n_L(0)$ is the initial number of ligand molecules.

We can figure out the characteristics of the captured molecules number, $n_M(t)$, from $n_L(t)$. Since the ligand molecules are transformed into captured molecules, we can state that the number of captured molecules is the complement of the ligand molecules. The summation of ligand and captured molecules always equals the initial number of ligand molecules. Hence, the number of captured molecules can be expressed as follows

$$n_M(t) = n_L(0)\left(1 - \mathrm{e}^{-\frac{kn_R t}{NV}}\right). \tag{37}$$

According to (37), the number of captured molecules increases with the reaction rate, the number of ligands and receptors goes up, whereas it decreases as the volume increases. (37) is used in the validation of SSA in Section 5.4.

5. Validation of the NanoNS framework

In this section, we present the validation of the NanoNS by comparing the outputs of the NanoNS and the results of our analytical investigations. We also make a performance evaluation of the NanoNS by using the utilization ratio, captured molecules over total molecules, of the NanoNS and analytical investigation results.

5.1. Validation of diffusion functionality

In this section, we validate and evaluate the performance of the diffusion functionality through captured molecules, the diffusion coefficient and the scalability of NanoNS.

5.1.1. Validation by captured molecules

In this section, we present a validation of the simulator according to the number of captured molecules by the NanoNS with time. We analyze the behavior of captured molecules with respect to the ratio between the nanomachine's radius and the shelter's radius, a/R, and the number of released molecules.

First, we analyze the behavior of captured molecules with respect to the ratio, a/R. We fix the simulation parameters except a and change a every 0.1 steps in the analysis. The parameters used in the simulation are given in Table 3. The simulator results and numerical analyses for ratios 0.4 and 0.5 are shown in Fig. 8.

As in Fig. 8, although numerical analyses and NanoNS outputs exhibit a similar behavior for each a/R ratio, they do not exactly match. In fact, this mismatch results from the assumptions and constraints in the numerical analysis. The significant point of the behavior of the

$$P^*_{Col} = 1 - \frac{4R\left(1-\frac{a}{2R}\right)}{(R+a)+4R\left(1-\frac{a}{2R}\right)+2a\ln\left(\frac{R-\frac{a^2}{2R}}{R-a-d}\right)+\frac{(a^2-R^2)}{d}\ln\left(\frac{R-a}{R-a-d}\right)} \tag{32}$$

Box I.

$$M^*_{Col} = Q\left(1 - \frac{4R\left(1-\frac{a}{2R}\right)}{(R+a)+4R\left(1-\frac{a}{2R}\right)+2a\ln\left(\frac{R-\frac{a^2}{2R}}{R-a-d}\right)+\frac{(a^2-R^2)}{d}\ln\left(\frac{R-a}{R-a-d}\right)}\right) \tag{33}$$

Box II.

Table 3
Simulation parameters to validate diffusion capability by captured molecules with respect to a/R.

Symbol (unit)	Definition	Quantity
Q	Number of molecules	1000
R (μm)	Radius of shelter	1
a (μm)	Radius of nanomachine	0.x
D (μm^2/s)	Diffusion coefficient	2.197
s (μm)	Integral step	0.1

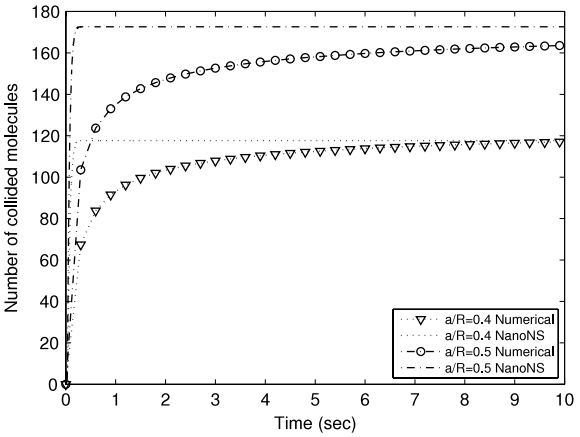

Fig. 8. Number of collided molecules for $a/R = 0.4$, $a/R = 0.5$.

NanoNS and numerical analysis is the number of captured molecules in a steady state. In Fig. 8, it is observed that the results of the NanoNS reach the saturation time faster than the numerical analysis results. As explained in forthcoming sections, the curves and the saturation times of the simulations do not affect the total number of collided molecules and depend on the speed of the carrier molecules, D. The reason of this mismatch is the lattice structure of the environment. Moreover, the NanoNS is implemented in a Cartesian coordinate system as described in Section 3.3.1. In Cartesian coordinate system, there always exists a discretization error while constructing a sphere, i.e., NanoNode. However, we do not use a Cartesian coordinate system in our numerical analysis. The validation of the NanoNS framework based on the comparison of numerical and experimental results is given in Section 5.5.

Table 4
Simulation parameters to validate diffusion capability by captured molecules with respect to Q.

	Q	a (μm)	R (μm)	D (μm^2/s)
Trial 1	100	0.5	1	2.197
Trial 2	200	0.5	1	2.197
Trial 3	500	0.5	1	2.197
Trial 4	1 000	0.5	1	2.197
Trial 5	2 000	0.5	1	2.197
Trial 6	5 000	0.5	1	2.197
Trial 7	10 000	0.5	1	2.197
Trial 8	20 000	0.5	1	2.197

We also analyze the attitude of captured molecules with respect to the number of released molecules, Q. In the analysis, only Q is changed while the other simulation parameters are fixed as given in Table 4.

In Fig. 9(a), although the NanoNS results quickly reach the saturation time, it is observed that the simulator results and numerical analyses of various Q's show a similar behavior. Besides, the normalized manners of simulation results and numerical analysis, which are normalized with Q, are presented in Fig. 9(b). Here, the normalized simulation results for various Q show a similar behavior.

In addition to these, we present an exhaustive performance evaluation of captured molecules according to Q in Table 5. Here, the error ratio is estimated as follows

$$ER = \frac{|ECM - OCM|}{RM} \tag{38}$$

where ECM is the expected captured molecules according to numerical analyses, OCM is the captured molecules obtained from NanoNS, RM is the total number of released molecules. As observed in Table 5, although the number of difference molecules increases with the amount of released molecule, the error ratios of trials do not depend on the amount of released molecules. The error ratios of trials range from 0.07% to 1.35%. This low error level, which is an acceptable value, shows that the amount of released molecules does not affect the error ratio of NanoNS.

5.2. Validation of diffusion coefficient

In this section, we validate the diffusion functionality of the NanoNS framework with respect to the diffusion coefficient, D, which is the parameter that affects the speed of the diffusion. D is expressed in terms of temperature,

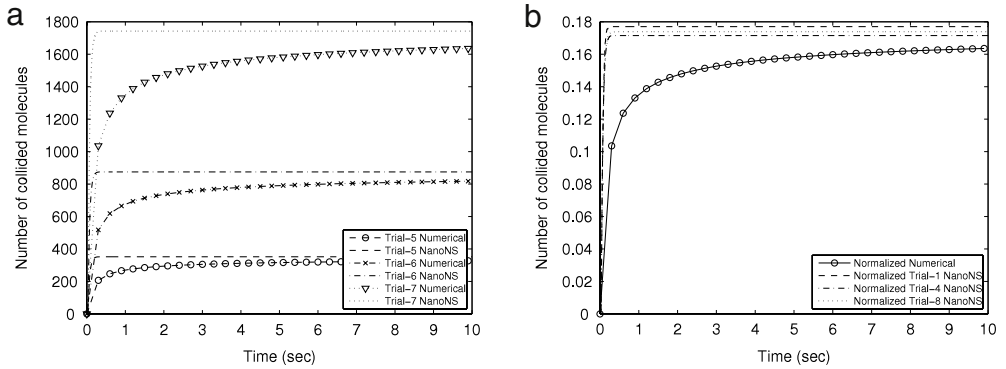

Fig. 9. (a) Behavior of collided molecules for $Q = 2000, 5000, 10\,000$. (b) Normalized number of collided molecules for $Q = 100, 1000, 20\,000$..

Table 5
Error analysis of diffusion capability by captured molecules with respect to Q.

	Obtained captured molecules	Expected captured molecules	Difference molecules	Error ratio (%)
Trial 1	17.7	16.4	1.34	1.35
Trial 2	32.9	32.7	0.14	0.07
Trial 3	82.3	81.8	0.57	0.12
Trial 4	171.5	163.5	7.95	0.8
Trial 5	351.9	327.1	24.76	1.24
Trial 6	874.9	817.7	57.13	1.14
Trial 7	1742.8	1635.4	107.32	1.07
Trial 8	3477.8	3270.9	206.94	1.03

Table 6
Simulation parameters to analysis impacts of temperature, viscosity of the medium and the radius of the carrier molecules on simulation results.

		Q	a (μm)	R (μm)	T (K)	η (J/K)	r_M (nm)	D (μm^2/s)
a	Trial 1	1000	0.3	1	150	0.001	100	1.098
	Trial 2	1000	0.3	1	300	0.001	100	2.197
	Trial 3	1000	0.3	1	600	0.001	100	4.394
	Trial 4	1000	0.3	1	900	0.001	100	6.591
b	Trial 5	1000	0.4	1	300	0.0005	100	4.394
	Trial 6	1000	0.4	1	300	0.001	100	2.197
	Trial 7	1000	0.4	1	300	0.002	100	1.098
	Trial 8	1000	0.4	1	300	0.003	100	0.732
c	Trial 9	1000	0.6	1	300	0.001	50	4.394
	Trial 10	1000	0.6	1	300	0.001	100	2.197
	Trial 11	1000	0.6	1	300	0.001	200	1.098
	Trial 12	1000	0.6	1	300	0.001	300	0.732

viscosity and the radius of the molecule as in (16). First of all, we analyze the effects of temperature, viscosity and the radius of the molecule on simulation results. Then, we analyze the behavior of D on NanoNS. Moreover, we measure the performance of NanoNS with respect to the diffusion coefficient.

If we consider our nanomachine as a unit volume, we can use (18) in our analysis. In (18), the saturation time, t_{sat}, can be expressed in terms of D as follows

$$t_{sat} = \frac{C_1}{D(erfc'(C_2 D))^2} \tag{39}$$

in which $erfc'$ is the inverse function of $erfc$, and C_1, C_2 are constant values. From (39), it is observed that while D increases, t_{sat} decreases and vice versa.

We categorize the trials in order to inspect the effects of temperature, viscosity and the radius of the molecule on D individually. We analyze temperature, viscosity and the radius of molecule in categories a, b and c, respectively. The simulation parameters and categorization of these are given in Table 6.

In category a, we examine the impact of temperature on simulation results. We fix all simulation parameters except temperature. In Fig. 10(a), while the temperature increases, which means an increase of D, the saturation time of trials decreases as derived in (39). We examine the influence of the viscosity of the medium on simulation results in category b. We fix all simulation parameters except viscosity. In Fig. 10(b), while viscosity increases, which means a decrease of D, the saturation time of trials increases. In category c, we inspect the effect of the radius of the carrier molecule on simulation results. We fix all simulation parameters except the radius of the information molecule. In Fig. 10(c), while the radius of

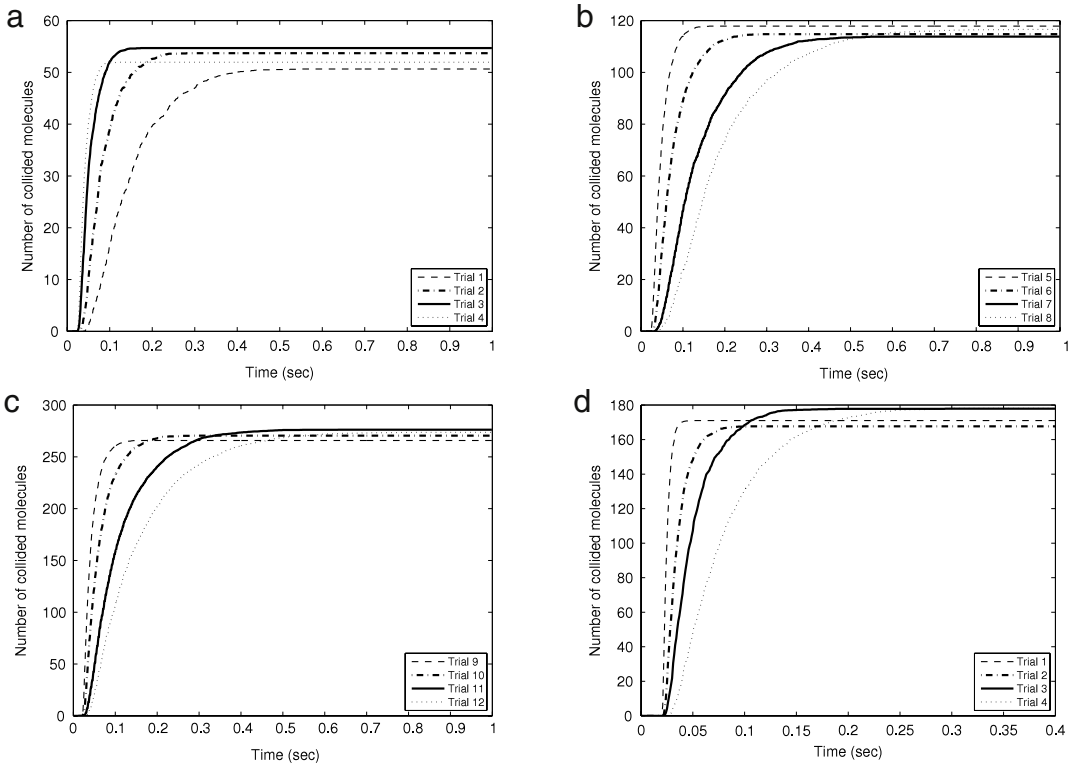

Fig. 10. (a) Impact of the temperature on simulation results. (b) Impact of the viscosity of the medium on simulation results. (c) Impact of the radius of the carrier molecule on simulation results. (d) Number of collided molecules for different diffusion coefficients.

Table 7
Simulation parameters to validate the diffusion capability by the diffusion coefficient.

Symbol (unit)	Trial 1	Trial 2	Trial 3	Trial 4
Q	1000	1000	1000	1000
R (μm)	1	1	1	1
a (μm)	0.5	0.5	0.5	0.5
s (μm)	0.1	0.1	0.1	0.1
T (K)	300	100	100	100
η (J/K)	0.001	0.001	0.002	0.002
r_M (nm)	10	10	10	20
D (μm^2/s)	21.973	7.324	3.662	1.831

carrier molecule increases, which means a decline of D, the saturation time of trials increases. In the rest of this section, we examine the impact of D on NanoNS. To achieve our purpose, the simulator is run with the parameters given in Table 7.

Fig. 10(d) shows the number of collided molecules with time behavior of all trials given in Table 7. The diffusion coefficient has no impact on the total number of collided molecules according to (32). If we check the total number of collided molecules of trials on Fig. 10(d), we see that they are in the range 170–180. The width of the range, 10, is an acceptable value in order to infer that the diffusion coefficient does not affect the total number of collided molecules.

In (39), if we neglect the impact of D on $erfc'$, it turns into a constant value. Based on this assumption, it is seen that there is an inverse proportional relation between t_{sat} and D as given in Fig. 11.

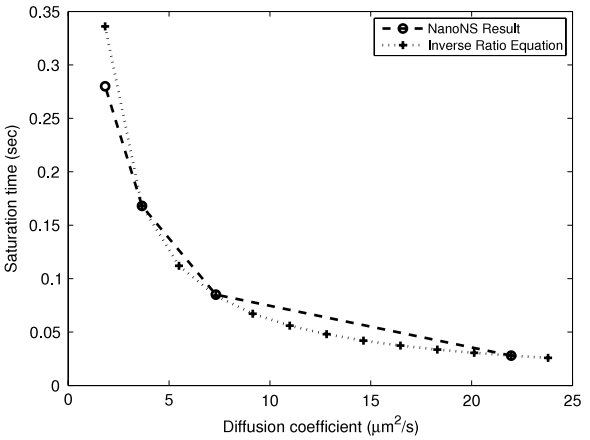

Fig. 11. The inverse proportional relation between t_{sat} and D.

5.3. Validation of NanoNS scalability

We evaluate the performance of the simulator with respect to the ratio of the radius of the nanomachine and the radius of the shelter. We also examine the scalability of the NanoNS with respect to a/R in the validation and performance evaluation. We make our scaling analysis for a random ratio, i.e., 0.4. We run the simulator with the parameters given in Table 8.

Fig. 12(a) shows the number of captured molecules with time for all trials given in Table 8. Since the lattice size of

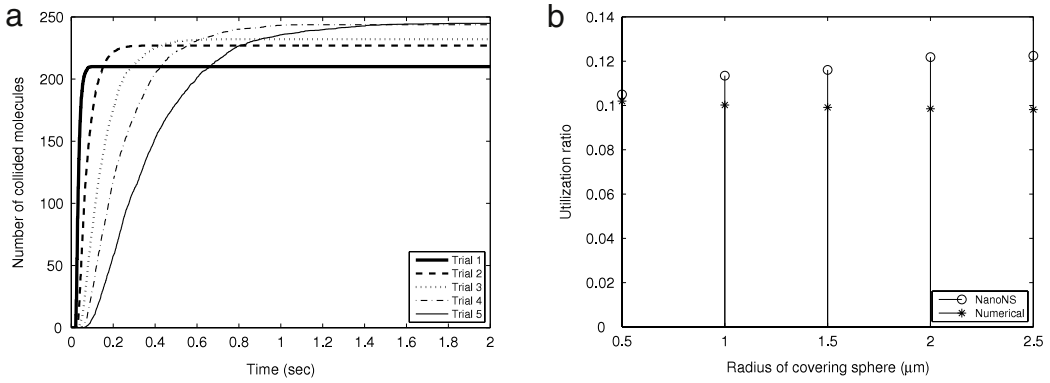

Fig. 12. (a) Number of collided molecules for different scales. (b) Utilization ratio for different scales.

Table 8
Simulation parameters to validate the diffusion capability by simulator scale.

Symbol (unit)	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
Q	2000	2000	2000	2000	2000
R (μm)	0.5	1	1.5	2.0	2.5
a (μm)	0.2	0.4	0.6	0.8	1
D (μm^2/s)	2.197	2.197	2.197	2.197	2.197
λ (nm)	100	100	100	100	100

the simulation is fixed, the distance between the source and the nanomachine is proportional to the number of lattices. If we consider our nanomachine as a unit volume, we can use (18) in our analysis. In (18), the saturation time, t_{sat}, can be given in terms of r as follows

$$t_{sat} = \frac{r^2}{C_1(erfc'(C_2 r))^2} \tag{40}$$

in which $erfc'$ is the inverse $erfc$ function and C_1, C_2 are constant values. It is seen that t_{sat} is proportional to r^2. Hence, the following relationship exists between the t_{sat} trials

$$R_1 < R_2 < R_3 < R_4 < R_5 \Rightarrow t_{SAT1} < t_{SAT2} < t_{SAT3} < t_{SAT4} < t_{SAT5}. \tag{41}$$

It is observed in Fig. 12(a) that the relationship among t_{sat} of simulator results is consistent with (41). If the effect of scaling is analyzed considering (32), all parameters except $\frac{(a^2-R^2)}{d} \ln\left(\frac{R-a}{R-a-d}\right)$ are proportional to the scaling factor, whereas $\frac{(a^2-R^2)}{d} \ln\left(\frac{R-a}{R-a-d}\right)$ is proportional to k^2. Since $(a^2 - R^2)$ is always negative, the number of collided molecules decreases when k increases as depicted in Fig. 12(b). Here, it is observed that when the radius of the covering sphere is 0.5 μm, the utilization ratio is almost the same for the NanoNS and the numerical analysis, whereas the ratio of the difference between them over the utilization ratio of the numerical analysis, error ratio, increases up to 20% as the radius goes up to 2.5 μm. Although the error ratio is approximately 20% for a large scaling factor, the ratio of the difference over the total utilization ratio is approximately 2% which is an acceptable figure.

5.4. Validation of Gillespie model

In this section, we validate and evaluate the performance of the SSA implementation of NanoNS. We make a numerical analysis of the SSA according to Gillespie-Scenario, in which ligand and receptor molecules are in a lattice side, in Section 4.3. In the validation of the SSA implementation, we make our trials according to the parameters given in Table 9.

As derived in (37), the number of captured molecules depends on initial number of ligand molecules, receptor molecules, reaction rate, volume of well-stirred medium and time. We check the impact of each of these parameters in Fig. 13. In each figure, the response of the NanoNS and the numerical analysis are plotted in order to validate each of these parameters one by one.

The impact of the reaction rate can be observed in Fig. 13(a). Numerical result and the NanoNS output show almost the same behavior. Trial 1, in which a deviation exists between the NanoNS and numerical results, has the largest reaction rate. Fig. 13(b) shows the behavior of the NanoNS and the numerical analysis for a varying number

Table 9
Simulation parameters to validate SSA.

	# of Ligand molecules	# of receptor molecules	Reaction rate (s/μm^2)	Volume side (nm)
Trial 1	40	20	0.01	100
Trial 2	40	20	0.001	100
Trial 3	40	20	0.005	100
Trial 4	80	10	0.005	100
Trial 5	80	20	0.005	200
Trial 6	80	20	0.005	100
Trial 7	80	20	0.005	50

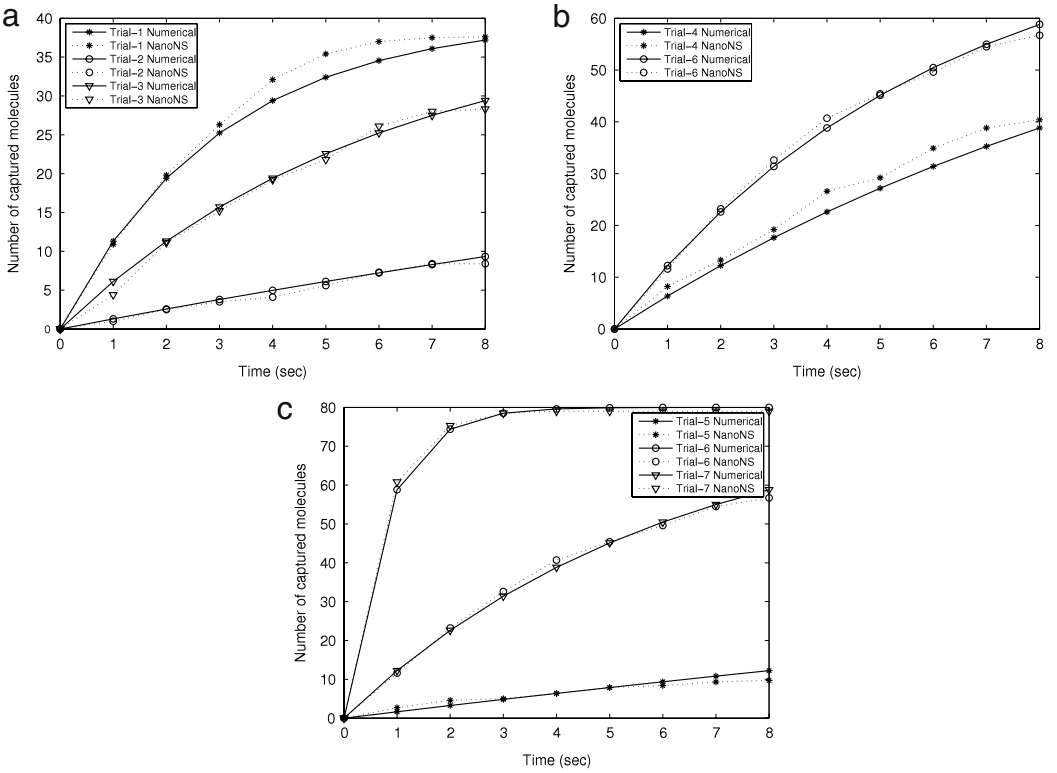

Fig. 13. (a) The captured molecules for varying reaction rates. (b) The captured molecules for varying number of receptors. (c) The captured molecules for varying volumes.

of receptor molecules. Numerical results and the NanoNS outputs show almost the same. The trials that have a lower number of receptor molecules show more deviation from the numerical analysis. Fig. 13(c) gives the behavior of the NanoNS for varying volumes. Here, the NanoNS exhibits similar responses with the numerical results for alternating volumes.

5.5. Performance evaluation of NanoNS

In this section, we make performance evaluation according to the utilization ratio of NanoNS. The utilization ratio of a trial, *UR*, can be achieved as follows

$$UR = \frac{\#\ of\ captured\ molecules}{Total\ \#\ of\ Molecules}. \tag{42}$$

The utilization of the simulator for different a/R ratios is given in Fig. 14(a). Here, captured molecules are plotted for two sources which are obtained from simulator and numerical analysis. It is observed that the results show similar patterns. However, they do not exactly match. We analyze this discrepancy in an error model. In this error model, we divide the difference in the number of captured molecules between the NanoNS and numerical analysis results by numerical analysis result as follows

$$ER = \frac{|SR - AR|}{AR} \tag{43}$$

in which *ER* is the error ratio, *SR* is the number of captured molecules by the NanoNS, *AR* is the number of captured molecules with respect to analysis result. Fig. 14(b) shows the *ER* value of the NanoNS with varying a/R ratios.

We can find the error ratio of the NanoNS by taking the average of the error ratios of the radius given in Fig. 14(b). Here, it is observed that the error ratio decreases and becomes stable for a/R values larger than 0.5. Thus, it is reasonable to select a/R values in the simulations.

The average error ratio of the NanoNS is 14.9% according to the error ratio values given in Fig. 14(b). However, note that there is a peak value on $a = 0.2$ for the numerical analysis result. The reason for this peak value is that the captured molecule number is quite small for $a = 0.2$. Although the difference between *SR* and *AR* is quite small with respect to total number of receptor molecules, a larger error ratio is observed for $a = 0.2$. Thus, it is rational to ignore $a = 0.2$ while taking the average. If we ignore $a = 0.2$, the error ratio of the NanoNS becomes 8.3%.

6. Conclusions

In this paper, owing to the distinctions between traditional and molecular communication, we have designed and implemented our molecular channel model in ns-2.

In this paper, the diffusive molecular channel model is described. We also analyze and model diffusive molecular communication. We choose the reaction-diffusion algorithm, *GMP*, to simulate the diffusive molecular channel of the NanoNS. We represent our classes, their functionalities and relationship with each other in order to build up a diffusive molecular channel. We construct our network

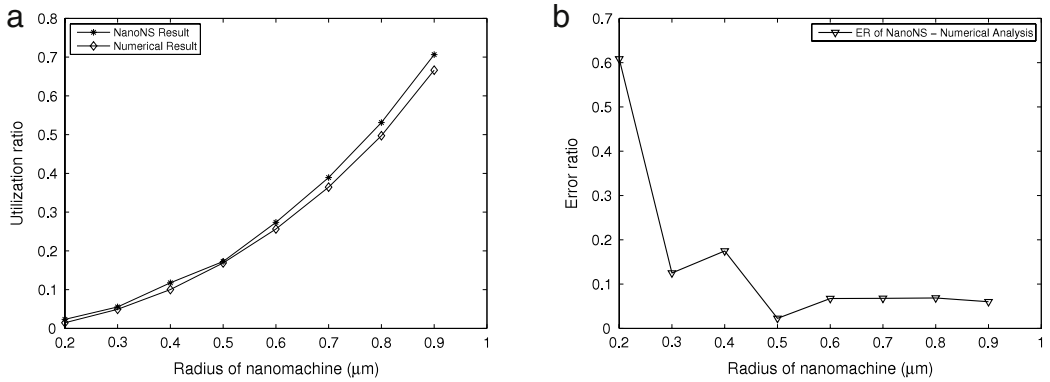

Fig. 14. (a) The utilization ratio of the NanoNS and numerical analysis. (b) NanoNS error ratio.

stack by creating classes which are inherited from the basic structures of ns-2. We merge diffusive channel structures and this network stack. Nevertheless, we do not implement complex MAC, routing or transport layer protocols. We implement a basic MAC protocol as an illustration.

We exhibit a numerical analysis of molecular communication for specific scenarios. In the analysis, we formulate the behavior of captured molecules with time in terms of the radius of the nanomachine, distance from source and time. Additionally, we present performance evaluation and validation for diffusion and reaction capabilities of the NanoNS. The performance evaluation and validation are made in terms of captured molecules, diffusion coefficient, simulation scalability and utilization rate. Additionally, we analyze the error ratio of the NanoNS, *ER*. The average *ER* value of the NanoNS is 8.3%. Nevertheless, there is a drawback in the performance evaluation of the NanoNS scalability. The error ratio is approximately 20% for large scaling factors. Fortunately, the amount of error is almost 2% of the total number of molecules for $a/R = 0.4$.

We have only modeled and implemented a diffusive molecular channel. Thus, elaborate modeling and implementation of motor-based and gap junction-based molecular communications can be considered as future work. Moreover, in the current implementation of NanoNS, nanomachines are fixed objects. Adding mobility functionality to nanomachines is another future work. We believe that NanoNS will provide a practical and beneficial simulation suite to develop network protocols for nanonetworks.

Acknowledgements

This work was supported in part by the Turkish Scientific and Technical Research Council Career Award under grant #109E257 and by the Turkish National Academy of Sciences Distinguished Young Scientist Award Program (TUBA-GEBIP).

References

[1] I.F. Akyildiz, F. Brunetti, C. Blazquez, Nanonetworking: a new communication paradigm, Computer Networks Journal (Elsevier) (2008).

[2] B. Atakan, O.B. Akan, On channel capacity and error compensation in molecular communication, Springer Transaction on Computational System Biology (2008).

[3] B. Atakan, O.B. Akan, On molecular multiple-access, broadcast, and relay channels in nanonetworks, in: International Conference On Bio-Inspired Models of Network, Information and Computing Systems Archive Proceedings of the 3rd International Conference on Bio-Inspired Models of Network, 2008.

[4] B. Atakan, O.B. Akan, Carbon nanotube-based nanoscale ad hoc networks, IEEE Communications Magazine 48 (2010) 129–135.

[5] B. Atakan, O.B. Akan, Deterministic capacity of information flow in molecular nanonetworks, Nano Communication Networks Journal (Elsevier) 1 (2010) 31–42.

[6] H.C. Berg, Random Walks in Biology, 2nd ed., Princeton University Press, Princeton, NJ, 1993.

[7] H.C. Berg, E.M. Purcell, Physics of chemoreception, Biophysical Journal (1977) 193–219.

[8] W.H. Bossert, E.O. Wilson, The analysis of olfactory communication among animals, Journal of Theoretical Biology 5 (3) (1963) 443–469.

[9] J. Crank, The Mathematics of Diffusion, 2nd ed., Oxford University Press, Oxford, 1975.

[10] C. Elfgang, R. Eckert, H. Lichtenberg-Frate, A. Butterweck, O. Traub, R.A. Klein, D.F. Hulser, K. Willecke, Specific permeability and selective formation of gap junction channels in connexin-transfected Hela cells, Journal of Cell Biology 129 (1995) 805–817.

[11] K. Fall, The ns Manual (formerly ns Notes and Documentation), in: The VINT Project, January 2009.

[12] R.A. Freitas, Nanomedicine, Volume I: Basic Capabilities, Landes Biosience, 1999.

[13] D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions, The Journal of Physical Chemistry 81 (25) (1977).

[14] A.F. Harris, M. Zorzi, Modeling the underwater acoustic channel in ns2, in: Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools, October 2007, pp. 22–27.

[15] S. Hiyama, Y. Moritani, T. Suda, R. Egashira, A. Enomoto, M. Moore, T. Nakano, Molecular communication, in: Proc. NSTI Nanotech'05, vol. 3, May 2005, pp. 391–394.

[16] M.J. Moore, A. Enomoto, T. Nakano, Y. Okaie, A. Kayasuga, H. Kojima, H. Sakakibara, K. Oiwa, T. Suda, Molecular communication: simulation of microtubule topology, in: Proc. 2nd Int. Workshop Natural Comput., 2007, pp. 134–144.

[17] M.J. Moore, A. Enomoto, K. Oiwa, T. Suda, Molecular communication: uni-cast communication on a microtubule topology, in: IEEE International Conference on Systems, Man and Cybernetics, SMC, October 2008, pp. 18–23.

[18] M.J. Moore, A. Enomoto, S. Watanabe, K. Oiwa, T. Suda, Simulating molecular motor uni-cast information rate for molecular communication, in: The 43rd IEEE Annual Conference on Information Sciences and Systems, CISS, March 2009.

[19] M. Moore, T. Suda, K. Oiwa, Molecular communication: modeling noise effects on information rate, IEEE Transactions on NanoBioscience 8 (2009) 169–180.

[20] Y. Moritani, S. Hiyama, S.M. Nomura, K. Akiyoshi, T. Suda, A Communication interface using vesicles embedded with channel forming proteins in molecular communication, in: Proc. of the 2nd

ICST International Conference on Bio-Inspired Models of Network, Information and Computing Systems, BIONETICS, December 2007.
[21] Y. Moritani, S. Hiyama, T. Suda, Molecular communication among nanomachines using vesicles, in: Proc. NSTI Nanotech' 06, vol. 2, May 2006, pp. 705–708.
[22] T. Nakano, T. Suda, T. Koujin, T. Haraguchi, Y. Hiraoka, Molecular communication through gap junction channels: system design, experiments and modeling, in: Proc. 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems, BIONETICS 2007, December 2007.
[23] T. Nakano, T. Suda, M. Moore, R. Egashira, A. Enomoto, K. Arima, Molecular communication for nanomachines using intercellular calcium signaling, in: Proc. 5th IEEE Conference on Nanotechnology, 2005.
[24] G. Peskir, On the diffusion coefficient: the Einstein relation and beyond. Stochastic models, Stochastic Models 19 (3) (2003) 383–405.
[25] M. Pierobon, I.F. Akyildiz, A physical channel model for molecular communication in nanonetworks, IEEE Journal on Selected Areas in Communications (JSAC) 28 (2010) 602–611.
[26] J.V. Rodriguez, J.A. Kaandorp, M. Dobrzyński, J.G. Blom, Spatial stochastic modelling of the phosphoenolpyruvatedependent phosphotransferase (PTS) pathway in escherichia coli, Bioinformatics 22 (15) (2006) 1895–1901.
[27] T. Suda, M. Moore, T. Nakano, R. Egashira, A. Enomoto, S. Hiyama, Y. Moritani, Exploratory research in molecular communication between nanomachines, UCI Technical Report, 05–3, March 2005.
[28] The network simulator [Online]. Available: http://www.isi.edu/nsnam/ns/ [accessed 29.07.10].
[29] P.J. Thomas, D.J. Spencer, S.K. Hampton, P. Park, J.P. Zurkus, The diffusion mediated biochemical signal relay channel, in: Proc. 17th Annual Conference on Neural Information Processing Systems, NIPS'03, 2003.

Ertan Gul received the B.S. and M.S. degrees in Electrical and Electronics Engineering from Ege University, Izmir, Turkey and Middle East Technical University (METU), Ankara, Turkey, in 2004 and 2010, respectively. His current research interests are in nano-scale and molecular communications and nanonetworks.

Baris Atakan received the B.Sc. and M.Sc. degrees in electrical and electronics engineering from Ankara University and Middle East Technical University, Ankara, Turkey, in 2000 and 2005, respectively. He is currently a research assistant in the Next-generation Wireless Communication Laboratory and pursuing his Ph.D. degree at the Department of Electrical and Electronics Engineering, Koc University. His current research interests include nanoscale communication, nanonetworks, and biologically-inspired communication protocols for wireless networks.

Ozgur B. Akan received the B.S. and M.S. degrees in Electrical and Electronics Engineering from Bilkent University and Middle East Technical University, Ankara, Turkey, in 1999 and 2001, respectively. He received the Ph.D. degree in Electrical and Computer Engineering from the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, in 2004. He is currently Associate Professor with the Department of Electrical and Electronics Engineering, Koc University, Istanbul, Turkey. His current research interests are in wireless communications, bio-inspired communications, nano-scale and molecular communications, network information theory.

Dr. Akan is an Associate Editor for IEEE Transactions on Vehicular Technology, International Journal of Communication Systems (Wiley), Nano Communication Networks Journal (Elsevier). He served as an Editor for ACM/Springer Wireless Networks (WINET) Journal (2004–2010), Area Editor for AD HOC Networks Journal (Elsevier) (2004–2008), as a Guest Editor for several special issues, as the TPC Co-Chair for the 5th ICST/ACM International Conference on Nano-Networks (Nano-Net 2011), TPC Co-Chair for the 13th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (ACM MSWiM 2010), the General Co-Chair for The Third International Conference on Bio-Inspired Models of Network, Information, and Computing Systems (ICST/IEEE BIONETICS 2008), the European Vice Chair for ICST/ACM Nano-Net 2007, an International Vice Chair for IEEE INFOCOM 2006, and in organizing committees and technical program committees of many other international conferences. He is the Vice President for IEEE Communications Society—Turkey Section. He is an IEEE Senior Member (Communications Society), and a member of ACM. Dr. Akan received the IBM Faculty Award twice in 2010 and 2008, Turkish Academy of Sciences Distinguished Young Scientist Award 2008 (TUBA-GEBIP).