MolComML: The Molecular Communication Markup Language

Eduard Alarcon[1], Raul G. Cid-Fuentes[1], Alan Davy[4], Luca Felicetti[2], Mauro Femminella[2], Pietro Lio'[3], Gianluca Reali[2], and Josep Solé-Pareta[1]
[1]Universitat Politecnica de Catalunya, Spain, [2]University of Perugia - CNIT RU, Italy
[3]University of Cambridge, UK, [4]Waterford Institute of Technology, Ireland

ABSTRACT

Given the high multi-disciplinarity of Molecular Communications (MolCom), researchers often face significant difficulties to understand each other. This impairment not only affect researchers with different backgrounds, but it also affects the different software tools. This paper motivates the development of the Molecular Communication Markup Language (MolComML). MolComML is proposed as an XML-based format to represent the considered elements, interactions, configuration and results of the experiments and simulations in the field of MolCom. MolComML is designed with the objective of converging all fields of research within MolCom to help the exchange of information. We overview its main functionality and define its basic composing elements.

Categories and Subject Descriptors

I.6.3 [**Simulation and Modeling**]: Simulation Languages

Keywords

Molecular communications, markup language, simulation, standardization

1. INTRODUCTION

Molecular Communication (MolCom) is a disruptive field of research that has been rapidly growing in the recent years [3]. Given its high multidisciplinary, bridging the disparate islands of knowledge stems as a pending challenge.

This issue does not only appear in the communication among researchers, but it also interferes with the interaction between software and simulation tools, thus imparing reproducibility of the research outcomes. As an example, consider the existing simulation tools, which have been used to simulate diffusion-based channels. Among others, BiNS [10], N3Sim [16] or Ns-3-based [5] utilize different configuration and output files, making cross-validation and reproducibility often unfeasible, especially for complex communication architectures [11, 7].

NANOCOM'16, September 28-30, 2016, New York, NY, USA
© 2016 ACM. ISBN 978-1-4503-4061-8/16/09...$15.00
DOI: http://dx.doi.org/10.1145/2967446.2967460

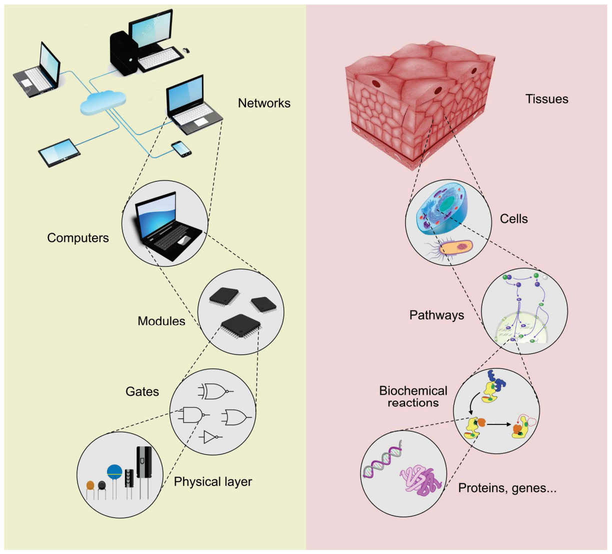

Figure 1: Top-bottom comparison between a computer network and a biological system.

This paper introduces the MolCom Markup Language (MolComML) as an XML-based language that promises to reunite both numerical analysis and experimental synthesis by ensuring a flexible markup language [1]. The development of such tool will allow cross-validation of experiments with the theoretical results, as well as to reduce significant researcher time in interfacing different software tools. As sketched in Fig. 1, the target of this activity can be compared with that of XML-based languages, which allow specifying network configuration in an unified way (e.g. NETCONF [9]).

We present the main objectives, elements and functions of the proposed language, as well as its structure. The definition of the MolComML is guided through a simple example, which aids the overall understanding of the language.

The structure of this paper is as follows: Section 2 overviews the background and the related work. Section 3 defines the molecular communication markup language and overviews its properties, elements, and functionality. Finally, Section 4 concludes our work.

2. BACKGROUND AND RELATED WORK

In this section, we overview existing specification languages for other disciplines, as well as existing simulators and experimental test-beds.

2.1 Molecular Communication Standard

The model defined by the IEEE 1906.1 Working Group [5] stands as the first nanoscale and molecular communication standard and constitutes a recommended practice for the definition of a general framework for the nanoscale communications. This standard proposes an architecture based on the following blocks: NetDevice, Communication Interface, Medium, Motion, Field, Specificity, and Perturbation.

2.2 Specification languages

As for mark-up languages used in biology, one of the most important related language is SBML [15], that is nowadays the standard for representing computational models in system biology. It allows the communication and storing of computational models of biological processes and its success is due to the possibility to represent different classes of biological phenomena, such as cell signaling pathways, regulatory networks, and many others. Its main purpose is to ease the model sharing among different software environments.

The CellML language is an open standard based on the XML markup language [8]. The purpose of CellML is to store and exchange computer-based mathematical models. CellML allows scientists to share models even when they use different model-building software. It also enables them to reuse components from one model to another one, thus accelerating model building. CellML includes information about model structure, mathematics and metadata.

NeuroML is an XML-based model description language, which provides a powerful common data format for defining and exchanging models of neurons and neuronal networks [19]. The structure and behavior of ion channel, synapse, cell, and network model descriptions are based on underlying definitions provided in LEMS, a domain-independent language for expressing hierarchical mathematical models of physical entities. It includes two Application Programming Interfaces (APIs) written in Python to simplify the process of developing and modifying models expressed in NeuroML and LEMS.

2.3 MolCom Simulators and Experimentation

In this section, we list some of the main simulators that have been developed to characterize the broad set of MolCom systems:

- **BiNS - Biological and Nano-Scale communication simulator** is a simulation multi-threaded package for MolCom systems developed at the University of Perugia [10]. Its customizable design provides a set of tools for creating objects and for modeling the behavior of biological entities, including the collision handling and the modeling of diffusion-drift propagation into both constrained and open space environments.

- **N3Sim** is a simulation framework for diffusion-based molecular communications, which allows the evaluation of molecular networks with several transmitters and receivers [16]. The diffusion of particles through the medium is modeled as Brownian motion, taking into account particle inertia and collisions among particles. It implements a three-layer architecture, which are the user interface layer, the data layer and the domain layer. The simulation parameters are determined by means of a text configuration file.

- **COMSOL Multiphysics** is a commercial multipurpose platform designed for simulating physics-based problems through a unified workflow for electrical, mechanical, fluid, and chemical applications [6]. It implements finite element analysis, for different physics and engineering applications. An example of the use of COMSOL Multiphysics for simulating a MolCom drug delivery system is presented in [6].

- **NS-2 and NS-3 Based Simulators**: NS-2 and NS-3 are discrete-event network simulators, which were not originally developed for MolCom. These simulators are organized in different software libraries that can work together. Its flexible structure has allowed implementing some basic elements of MolCom. Among others, NanoNS[14] is an NS-2 based simulator for diffusive molecular communication in aqueous mediums, with constant thermal motion of molecules. In addition, a NS-3 based a simulation tool has been developed in the framework of the IEEE P1906.1 working group [5]. User programs can be written in C++ or Python programming languages.

- **BNSim** is a multithread java simulator for bacteria networks [21]. These networks interconnect engineered bacteria that communicate at nanoscales. BNSim integrates three simulation methods: (i) the Gillespie stochastic simulation algorithm; (ii) stochastic differential equations, used to model large-scale chemical system with a controlled level of approximation; (iii) a hybrid algorithm which integrates the above methods.

- **NCSim - Bacteria Nanonetworks** is a comprehensive simulation framework for molecular communications, utilizing flagellated bacteria for information delivery [4]. Its major focus is on different message encoding techniques. It can simulate several simultaneous links between the nanomachines. NCSim incorporates the stochastic model for bacteria mobility, and the plasmid/chromosome transfer between bacteria through the conjugation process. NCSim consists of three modules: (i) physical (PHY) layer of bacterial nanonetworks; (ii) scenarios generator and simulation monitor; and (iii) plotting tool, intended to post-process raw simulation data and to generate plots.

- **HLA Simulator**. In [2], the authors introduce a simulator design focusing on scalability, and adopting the high level architecture (HLA) model, which is standardized under IEEE 1516. It is used to design a distributed simulation tool for molecular communications, so that different scalability options can be used to include additional processing power to reduce the execution time. This model allows designing large systems, which could be difficult to do otherwise.

As for experimentation, significant examples for molecular communications are those illustrated in [12, 18]. In the former, real in-vitro communications between platelets and endothelial cells by means of CD40L molecules are tested. In the latter, the authors have combined cells with nanotechnology for an integrated molecular processing network. Specifically, engineered cell populations are triggered by a quorum sensing signal molecule to express surface-displayed fusions consisting of a fluorescent marker and an affinity peptide. The latter provides means for attaching magnetic nanoparticles in order to fluorescently activate subpopulations for coalescence into colour-indexed output. The resultant nano-guided cell network assesses quorum sensing activity and conveys molecular information as a 'bio-litmus' in a manner read by simple optical means.

3. MOLCOMML

A new MolCom Markup Language (MolComML) is necessary for bridging the areas of interest for molecular communications. Accordingly, this needs to be flexible and generic, such that it can be understood by *any software and simulation platforms* for molecular communications. It also needs to be capable of providing a large level of detail by providing all governing equations, parameter values and necessary conditions, such that it can entirely describe a MolCom environment in either a simulation or experimental test-bed.

In addition, due to the large growing rate of MolCom-related research and its high multi-disciplinarity, MolComML needs to be easily extensible to integrate the most recent and on-going advancements, compatible with existing languages of neighboring disciplines (e.g., SBML [15] for systems biology and NeuroML [19] for neurology). Finally, MolComML should be compliant with the IEEE P1906.1 recommended practice for nanoscale and molecular communication.

Its basic structure consists of several blocks reflecting the main components of a general molecular communication case. Each block is composed of a set of required parameters and a set of custom parameters that could be defined each time according to the molecular communication needs.

3.1 Objectives

The main objectives of the MolComML format are listed in what follows:

- Represent the many different classes of molecular communication scenarios, in all levels of abstraction.
- Enable the use of multiple software tools without having to rewrite models to conform to different file formats.
- Ensure the survival of models beyond the lifetime of the software used to create them.
- Use a single language both to analyze the considered scenarios through software tools, as well as to synthesize actual experiments. This ensures repeatability and cross-validation.
- Enable models to be shared and published in a form that any researcher can use even by making use of different software environments.
- Enable future expandability of the markup language. Due to the rapid knowledge growth in this field, this language needs to be constantly updated. MolComML will be versioned to structure the integration of novel definitions and models.

3.2 Elements and Functionality

This section overviews the composing elements of MolComML, as well as their basic functionality. Fig. 3 illustrates the general UML diagram showing the interconnections and dependencies among different blocks. The different units and blocks are defined according to the specifications of MolComML. As shown, the network elements stand as the central element of the diagram.

3.2.1 Network Elements

Network elements are considered as the building blocks for users. They are created and interconnected to define the simulation or experimental set-up. Each element may be defined by different levels of abstraction. For instance, it can be an entirely conceptual entity; it can have some real physical interpretation, or both. Each network element has a set of standard attributes that could be extended by the introduction of custom ones. The most important attributes describe the shape and size of the element, its mass and time to live properties, the accepted and transmitted signals and, finally, the motion rules, if such element is equipped by autonomous propulsion system.

The main network elements are as follows:

- Transmitter: These elements are in charge of encoding information in the form of a molecular communication. They need to include all the relevant parameters. Among others, rate of creation of molecules, rate of emission, and the molecule release mechanism.
- Receiver: These elements are in charge of decoding the information by detecting the induced fluctuations in the molecular channel. They need to consider multiple configurations and types of receivers, such as absorbing receiver and receiver with absorbing receptors
- Signal: The transmitted signal carries the information towards the receiver. This can be based on DNA, proteins, ions, and others.

Derivations of these elements, as well as other elements (active or passive) in the communication channel that may affect the communication can also be part of this list.

3.2.2 Communication Interface

Network Elements can be connected with other elements by using the Communication Interface element, that describes the external interfaces of each network element. Such interfaces have several properties that describe also the type of transmitted and received signals, their affinity and the direction of communication. Again, a subset of custom parameters can be defined in order to describe more in details the properties of each interface, as described in what follows.

3.2.3 Properties and parameters

Defining a list of custom attributes and properties is a key aspect to define the studied problem. These are identified by the <*param*> tag. This approach allows a high customization capability, extending the predefined attributes or introducing completely new ones. The format is quite simple, one has just to define a set of new attributes after the <*param*> tag. This approach allows specifying both properties and attributes of each element described in the MolComML file.

3.2.4 Compartment Elements

The compartments are intended as a kind of well-stirred container of a particular type and finite size where species (e.g. chemical substances) may be located. A model may contain multiple compartments even of the same compartment type, and they can also be located inside each another, hierarchically. Connections between different compartments are handled by Gates, that define the rules for crossing. Note that each network element in a model must be located within a compartment.

3.2.5 Gates

It is possible to define a list of Interconnection gates between a couple of adjacent compartments. Each gate is identified by a unique name, a position, size, shape and orientation, in order to create a sort of passing hole on the surface of both compartments.

3.2.6 Interaction Rules

A set of rules need to be defined. They restrict or specify the operation of the network elements and their connections. The collision behavior is a typical example: upon impact, molecules can join, merge, or absorb one another or they can bounce away from each other. Each rule could be a global rule valid everywhere or it could be more specific, describing only a part of the communication environment or being only valid for a subset of network elements. In general, rules are described by mathematical expressions imported from an external model (e.g. MathML [20]). It is also possible to initialize constants and variables of the imported equations.

3.2.7 Communication Channels

The transmitted signals are transferred to receivers through communication channels. There exist several channel types. Junction-based, diffusion, and diffusion-with-drift are examples of existing channel types. A channel element has to be defined and connected to each compartment placed in the communication environment. This ensures the description, with a high degree of accuracy, of the local environmental conditions, by means of specific mathematical rules defined in one or more external MathML files. It is also possible to initialize a set of the parameters used in the imported equations. Each channel definition could be shared between two or more Compartments. The association channel-compartment is defined in the compartment section.

3.2.8 Network Topology

Each Network Element described in section 3.2.1 defines only the main properties of such element. The Network Topology section is used to place the required elements in the proper Compartment. The Topology of the molecular communication network is described by defining the position and orientation of each element and also the communication protocol at the basis of the end-to-end communication. The initialization of any node parameters are declared here.

3.2.9 Protocol Stack

Taking inspiration from a previous work [17], this section defines the protocol stack that could be used for the communication needs. It is possible to define any number of protocol stacks and each one could have a custom structure composed of different depth and identification name. Each layer could map the well known protocol stacks of the traditional telecommunication field or define completely new layers. The layers are defined by a set of rule and each one is composed of two signals, the first is for the forward communication and the last is for the backward communication. For each signal, the type of carrier that will be transmitted

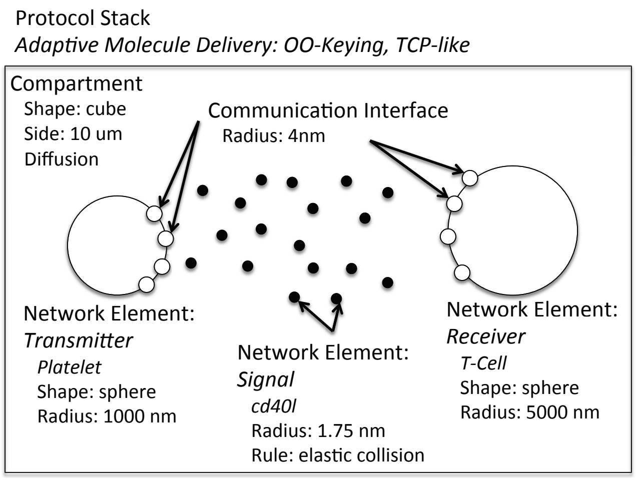

Figure 2: Graphical description of the MolComML to illustrate the framework in [13].

(defined previously in the Network Elements section) and the modulation type are defined. It also allows the definition of a set of custom parameters in the standard format of <*param name=”” value=”” and unit=””* /> in order to initialize all the required parameters for that layer. As shown in Fig. 4, in the Link layer the format of the transmitted messages is specified, along with the algorithms that manage the communication, such as the synchronization and the rate control algorithms for each signal type defined above.

3.2.10 Event Scheduler

The Event Scheduler tag allows the definition of both the initial state for each Network Element and specific events that cause a state transition on a target node, upon the occurrence of an event or at scheduled times.

3.2.11 Unit Definition

Each numerical attribute defined in the MolComML files is associated with a unit of measure declared between the listOfUnits tags. It is possible to define the most common units, and their multiples, in the International System. The definition of custom units is based on the combination of the previous ones, by setting their values, scales and exponents through this format: $customUnit = (value*10^{\,scale})^{\,exponent}$, inside the <*listOfUnitDefinitions*> tags. The name chosen for each custom unit can be used in the other sections of the file, i.e. for the definition of the numerical value of the considered attribute.

3.2.12 External Sources

MolCom systems is expected to interact with elements modeled in neighboring disciplines. In order to allow interoperability, it is necessary to translate the configuration parameters and results from the other languages, such as SBML or SBOL, providing a flexible adaptation layer that will help integrating and extending the usability of MolComML. For this reason, in the MolComML files it is possible to define the elements to be imported from each external source. For each one, the list of imported elements is defined by specifying the coupling between the original name in the external source with the name used in the MolComML file.

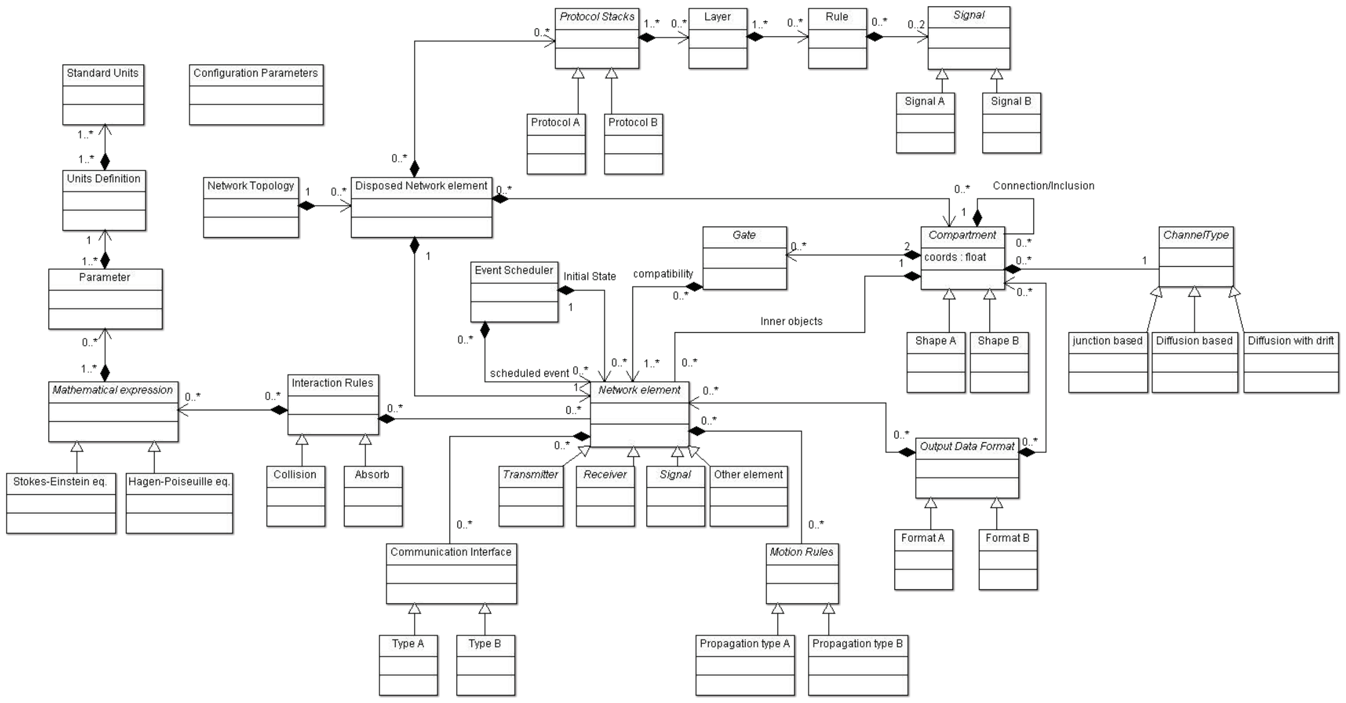

Figure 3: General UML diagram of MolComML elements.

For each pairing, also the importing rule is specified; it could be either a comprehensive or a partial import. For the latter case, it is necessary to define each parameter that has to been imported, by specifying its identification name on the external source.

3.2.13 Output

The output data format is fixed for all simulations and numerical results. The defined output scheme allows it to specify which elements have to be exported. In more detail, it is possible to define the list of Network Elements, of the Compartments and of their attributes, by specifying also the time interval for their monitoring. The definition of the monitored attributes is completely custom, so you can define a rule for each attribute of interest, through the well known rule <*param*> tag, introduced above. If the software performs additional post-processing steps applied on the raw numerical results of simulation steps, these have to be described in detail. This includes the identification of data to process, the order in which changes were applied, and also the nature of changes.

4. CASE EXAMPLE

We utilize MolComML as a basis to describe the framework presented in [13]. This consists on a point-to-point communication link between a Platelet and a T-Cell, using CD40L as communication molecules. This is illustrated in Fig. 2. In order to be able to use a MolComML file, each compliant simulator needs a parser module, able to extract the information form MolComML and translate it into its specific configuration file. A further step is to design an input module, able to read directly the information contained in the MolComML file. We are implementing this input module in BiNS2 [10] as proof-of-concept; the relevant source code will be available on the simulator web site and on the CIRCLE code repository (http://gitlab.fet-circle.eu).

The translation into actual MolComML language enables the exchange of configuration files among simulation platforms. We show a part of the XML code in Fig. 4. As shown, the different elements and parameters are clustered by type.

5. CONCLUSIONS

This paper has laid the foundations of the Molecular Communication Markup Language (MolComML). It is proposed as an XML-based format aimed at universalizing the system description in software, simulation tools and experimental test-beds, targeting both mutual understanding among the disperse islands of knowledge in this highly multidisciplinary research field and to guarantee reproducibility of research achievements. A proof-of-concept prototype is being implemented in the BiNS2 simulator.

6. ACKNOWLEDGMENTS

This work is supported by the EU project H2020 FET Open CIRCLE (Coordinating European Research on Molecular Communications) No. 665564 and by the MolML project funded by University of Perugia.

7. REFERENCES

[1] F. Achard et al. XML, bioinformatics and data integration. *Bioinformatics*, 17(2):115–125, Feb 2001.
[2] A. Akkaya et al. HLA based architecture for molecular communication simulation. *Simulation Modelling Practice and Theory*, 42:163 – 177, 2014.
[3] I. F. Akyildiz et al. Nanonetworks: A new communication paradigm. *Comput. Netw.*,

```xml
<?xml version="1.0" encoding="UTF-8"?>
<molcomml version="1.0">
  <model name="Platelet_TCell_Communication" description="...">
    <listOfConfigurationParameters>
      <param name="T" value="310" unit="K" description="temp" />
    </listOfConfigurationParameters>
    <listOfUnits>
      <unit name="nm" scale="-9" />
    </listOfUnits>

    <listOfNetworkElements>
      <NetworkElement name="platelet" type="transmitter/receiver">
        <size name="radius" unit="nm" value="1000" />
        <signal type="cd40l" direction="out" />
        <signal type="trombin" direction="in" />
        <motion type="none" />
      </NetworkElement>
    </listOfNetworkElements>

    <listOfCommunicationInterface>
      <CommunicationInterface name="platR" type="element">
        <status type="enabled" />
        <direction input="no" output="yes" />
        <signal type="cd40l" direction="out" time="4" unit="s"
         affinity="none" />
        </CommunicationInterface>
    </listOfCommunicationInterface>

    <listOfProtocolStacks>
      <protocolStack name="Adaptive_Mol_Delivery" maxLevel="5">
        <layer level="1" type="Physical">
          <rule>
            <signal type="cd40l" direction="forward"></signal>
            <signal type="IL-4" direction="feedback"></signal>
          </rule>
        </layer>
        <layer level="2" type="Link">
          <rule>
            <signal type="cd40l" direction="forward">
              <controlMessage name="start" format="1011" />
              <synchronization name="syncAlgorithm" />
            </signal>
            <signal type="IL-4" direction="feedback">
              <dataMessage name="upload" format="burst"
               payload="none"/>
              <rateControl name="TCP_Like_Algorithm" />
            </signal>
          </rule>
        </layer>
     </protocolStack>
    </listOfProtocolStacks>

  <listOfCompartments>
    <externalChannel name="diffusion">
      <Compartment name="box" parent="none">
            <shape type="cube" />
        <gate name="gate1" />
        <channel name="diffusion" />
      </Compartment>
    </externalChannel>
  </listOfCompartments>

  <listOfChannels>
    <channel name="diffusion" >
      <mathRule name="diffusion" path="./mathEquations">
      </mathRule>
        </channel>
  </listOfChannels>

  <networkTopology>
    <disposedNetworkElement name="tx1" type="platelet" >
      <compartment name="tube" />
      <position name="coords" x="100" y="200" z="100" unit="um" />
      <protocolStack type="Adaptive_Mol_Delivery" />
    </disposedNetworkElement>
  </networkTopology>
  </model>
</molcomml>
```

Figure 4: MolComML Configuration file of the presented framework in [13].

52(12):2260–2279, Aug. 2008.

[4] S. Balasubramaniam et al. Multi-hop conjugation based bacteria nanonetworks. *IEEE Transactions on NanoBioscience*, 12(1):47–59, March 2013.

[5] S. F. Bush et al. Defining communication at the bottom. *IEEE Trans. on Molecular, Biological and Multi-Scale Comm.*, 1(1):90–96, March 2015.

[6] Y. Chahibi et al. A molecular communication system model for particulate drug delivery systems. *IEEE Transactions on Biomedical Engineering*, 60(12):3468–3483, Dec 2013.

[7] U. Chude-Okonkwo et al. Molecular communication model for targeted drug delivery in multiple disease sites with diversely expressed enzymes. *IEEE Trans. on NanoBioscience*, 15(3):230–245, April 2016.

[8] A. Cuellar et al. The CellML 1.1 Specification. *J Integr Bioinform*, 12(2):259, 2015.

[9] R. Enns et al. Network Configuration Protocol (NETCONF). RFC 6241, June 2011.

[10] L. Felicetti et al. A simulation tool for nanoscale biological networks. *Nano Communication Networks*, 3(1):2 – 18, 2012.

[11] L. Felicetti et al. Simulating an in vitro experiment on nanoscale communications by using bins2. *Nano Communication Networks*, 4(4):172 – 180, 2013.

[12] L. Felicetti et al. Modeling CD40-based molecular communications in blood vessels. *IEEE Transactions on NanoBioscience*, 13(3):230–243, 2014.

[13] L. Felicetti et al. Tcp-like molecular communications. *IEEE Journal on Selected Areas in Communications*, 32(12):2354–2367, Dec 2014.

[14] E. Gul et al. Nanons: A nanoscale network simulator framework for molecular communications. *Nano Communication Networks*, 1(2):138 – 156, 2010.

[15] M. Hucka et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, Mar 2003.

[16] I. Llatser et al. Exploring the physical channel of diffusion-based molecular communication by simulation. In *IEEE GLOBECOM 2011*, Dec 2011.

[17] T. Nakano et al. Molecular communication among biological nanomachines: A layered architecture and research issues. *IEEE Transactions on NanoBioscience*, 13(3), 2014.

[18] J. L. Terrell et al. Nano-guided cell networks as conveyors of molecular communication. *Nat Commun*, 6:8500, 2015.

[19] M. Vella et al. libNeuroML and PyLEMS: using Python to combine procedural and declarative modeling approaches in computational neuroscience. *Front Neuroinform*, 8:38, 2014.

[20] W3C. W3c's math home page. *Available via the World Wide Web at http://www.w3.org/Math/*, 2000.

[21] G. Wei et al. Efficient modeling and simulation of bacteria-based nanonetworks with BNSim. *IEEE Journal on Selected Areas in Communications*, 31(12):868–878, December 2013.