



BIONETS

WP 1.3 – PERFORMANCE ANALYSIS AND EVALUATION

D1.3.3 BIONETS Performance Analysis

Reference:	BIONETS/BUTE/wp1.3/1.0
Category:	Deliverable
Editor:	Sándor Szabó (BUTE)
Authors:	Vilmos Simon, Sándor Szabó, László Bacsárdi, Márton Bérces, Endre Varga (BUTE), Giannis Koukoutsidis (NKUA), Francesco De Pellegrini (CN), Juhani Latvakoski, Tomi Hautakoski (VTT)
Verification:	Daniele Miorandi (CN), Heiko Pfeffer (TUB), Iacopo Carreras (CN), Juanjo Aparicio (TechIdeas)
Date:	Dec 12, 2008
Status:	Final
Availability:	Public

EXECUTIVE SUMMARY

In this deliverable the new, SerWorks aligned architecture of the BIONETS Simulation Platform is introduced, together with the simulator integration activities spanning across WPs and SPs. The reshaped BIONETS Simulation Platform gives us the opportunity to verify and test the properties of the BIONETS architecture on a wide scale, by simulating a huge number of T-nodes and U-nodes, as it is expected in real-life scenarios. The Simulation Platform was redesigned in a tight cooperation with the SerWorks and the WP5 prototyping task force, sharing the expertise between the different implementation activities as a joint work between WP1.1 (Requirements and Architectural Principles), WP1.3 (Simulation and Performance Analysis) and WP5 (Prototyping and Validation). This cooperation activity is in consonance with the recommendations of the reviewers to create a Simulation Platform capable of testing the whole BIONETS architecture. This new structure is completely aligned with these principles, providing the possibility to simulate the overall BIONETS architecture. To achieve the integration of the different level components, the Simulation Platform was designed in a way to accommodate SerWorks, the unified service-oriented architecture, where the adaptation and the evolution of the network protocols is driven by the running services. In the presented version we provide implementation for the Networking Framework functions of SerWorks, integrating the results in this area from previous deliverables, and the architecture for higher layer services. The Networking Framework API was specified in detail, completely defining the lowest level framework of SerWorks. This new API allows dynamic composition of new networking services using already existing components. The primitives of these basic components were not defined in the first version of the SerWorks specification, so in order to build simulations and prototypes the first set of the API for these components were specified. A scalable and high-performance simulation of the T-Nodes were developed which allows flexible trade-off between performance and accuracy using two different T-Node models (Explicit and Virtual T-Nodes). We also provided a standardized simulation scenario by identifying the features of BIONETS that can and should be simulated in an integrated way. In the appendices the concrete examples of the integration of the various networking solutions are presented (introduced in earlier deliverables) like the Situated Adaptive Forwarding (SAF) and the Cooperative Content Retrieval (CCR). The unique features of the SAF dissemination protocol required unique modifications of the Simulation Platform. Also for the integration of CCR scheme the Simulation Platform had to be extended with new functionalities for supporting the features of the CCR scheme, like the advanced T-node subsystem and a new mobility model that uses a graph representation of the street grid on which the U-nodes move. This document is meant to set the foundation of the cooperation between the Simulation, Prototyping and SerWorks Task Forces. One of the outcomes of this cooperation will be the adaptation of the Simulation Platform to support significantly the prototyping of the BIONETS system, by using a hybrid hardware-in-the-loop simulation feature, where the prototypes use the data produced by the simulated T-nodes, enabling a large-scale T-node system which is not feasible in a prototyped network.

DOCUMENT HISTORY

Version History

Version	Status	Date	Author(s)
0.1	Draft	13 Oct 2008	Vilmos Simon (BUTE)
0.2	Draft	8 Nov 2008	Vilmos Simon (BUTE); Endre Varga (BUTE)
0.3	Draft	13 Nov 2008	Vilmos Simon (BUTE)
0.4	Draft	1 Dec 2008	Vimos Simon (BUTE); Endre Varga (BUTE)
1.0	Final	12 Dec 2008	Vilmos Simon (BUTE); Endre Varga (BUTE)

Summary of Changes

Version	Section(s)	Synopsis of Change
0.1		Outline
0.2		First contributions
0.3		Integration of contributed sections
0.4		Corrections according to internal review
1.0		Final assembled version

SUMMARY

Contents

1	Motivation and relevance	6
2	Architecture and Components of the Simulation Platform	7
2.1	Networking Framework primitives	9
2.2	Networking Framework modules	10
2.3	T-Nodes simulation architecture	10
2.4	Interface for prototype—simulation integration (hardware-in-the-loop simulations)	11
2.5	Mediators and support for dynamic composition	13
3	BIONETS Networking Framework API specification	14
3.1	Opportunistic Communications	14
3.2	Information Filtering	15
3.3	Information Gathering	15
3.4	Data Dissemination	15
3.5	Naming System	16
3.6	Secure Communication (as described in D1.1.3/3.1.3)	16
4	Simulation Scenario Draft	17
4.1	Goals of the simulation	17
4.2	Abstract entities of the Simulation	17
4.3	Description of the simulation scenario	18
5	Conclusions and Next Steps	20
	APPENDIX	21
A	Map of theoretical models of BIONETS	21
A.1	Dense Scenario	21
A.2	Sparse Scenario	21
A.3	From analysis to design: the problem of control	23
B	Integration of the Situated Adaptive Forwarding	24
C	Integration of the Cooperative Content Retrieval scheme	25
C.1	Introduction and Objective	25
C.2	General Simulation Scenario	25
C.3	Implementation Details	26
C.4	Integration of the CCR scheme	28
D	BIONETS Simulation Platform Documentation	30
	BIBLIOGRAPHY	31

1 Motivation and relevance

In the process of engineering an autonomic computing and communication system like BIONETS, through the introduction of self-evolving autonomic services, a key issue is the anticipatory evaluation and testing of the emerging solutions and ideas. The complexity and the disappearing nature of the network environment in the BIONETS system architecture demand a profound validation and testing of the system properties, like cooperation of the heterogeneous network components, performance and scalability aspects. The verification of the performance and the suitability of the designed system can be done by prototypes for small number of network components (such as the U-nodes), but for large scale validation a unified simulator can give us deeper insight.

Therefore our goal was to provide a holistic tool for assessing the performance of the architectural and algorithmic solutions developed within the project. This was done by designing the BIONETS Simulation Platform, which gives us the opportunity to verify and test the properties of the BIONETS architecture on a wide scale, by simulating a large number of T-nodes and U-nodes, as it is expected in real-life scenarios. An important requirement to be fulfilled by the simulator is to be capable of simulating the overall BIONETS architecture, which means not only the different networking solutions, but also some of the service- and security-level functionalities of the project. For accomplishing this goal the BIONETS Simulation Platform should incorporate all the substantial elements and algorithmic solutions of the BIONETS system, by the means of software module integration and benchmarking.

To achieve the integration of the different level components, the Simulation Platform was designed to accommodate SerWorks [1], a unified service-oriented architecture, where the adaptation and configuration of the network protocols is driven by the running services. By defining the interfaces and its networking primitives between the network and service framework (the Networking Framework part of SerWorks), the networking functionality implemented in the simulator can be tailored to the specific service needs.

Another aspect of the integration activity is the incorporation of the simulation modules implemented for testing different scenarios into the unified Simulation Platform. These modules (e.g. Situated Adaptive Forwarding, Cooperative Content Retrieval) were decomposed into the common networking primitives (which define the lowest level framework of SerWorks) therefore the diverse networking modules can be simulated together defining general simulation scenarios.

Furthermore the Simulation Platform could also support significantly the prototyping of the BIONETS system, by using a hybrid hardware-in-the-loop simulation feature, where the prototypes use the data produced by the simulated T-nodes, enabling a large-scale T-node system which is not feasible in a prototyped network.

This document is organized as follows: in Sec. 2 the new architecture of the BIONETS Simulation Platform is introduced, Sec. 3 describes the Networking Framework primitives for accommodating SerWorks in the Simulation Platform, Sec. 4 concludes the deliverable with a standardized simulation scenario, identifying features of BIONETS that can and should be simulated in an integrated way. App. A depicts a map of BIONETS theoretical models, App. B and C describes the integration of the various simulation modules (Situated Adaptive Forwarding, Cooperative Content Retrieval) into the Simulation Platform, while App. D presents the documentation of the BIONETS Simulation Platform together with a get started manual which helps the installation and usage of simulation package.

Concern	Implementation status
Networking Framework primitives	implemented
Opportunistic Communications	implemented
Data Dissemination	implemented
Naming System	implemented
Information Filtering	implemented
Information Gathering	implemented
P2P Cloud	not implemented
Interoperability with IP	not implemented
Security	partially implemented
T-Node modeling	implemented
Dynamic composition support	partially implemented
Hardware-in-the-loop simulation	in progress (with WP5)

Table 1: Status of implementation

2 Architecture and Components of the Simulation Platform

The most important achievement presented in this deliverable is the harmonization of the simulator architecture with the SerWorks specification [1]. In this section we discuss the achievements and implementation details of the Simulation Platform following the SerWorks terminology. SerWorks is a unified representation and specification of the BIONETS architecture including services and networking. Its most important features are modularity, reusability and late binding between components which allows flexible composition of components in all levels of the architecture. SerWorks itself consists of three different frameworks that encapsulate different aspects of the BIONETS technology. The highest level framework is the Service Framework, which defines the whole service lifecycle including discovery, migration and evolution. Just below the Service Framework sits the Interaction Framework which provides different inter-service communication models inside a node.

The focus of this deliverable is on the lowest level framework, the Networking Framework which provides networking level communication and control facilities. Just like the Service Framework, the Networking Framework allows dynamic composition of new networking services using existing components. The amount of flexibility is however more constrained at the networking level. In the Networking Framework eight basic components are defined (see Figure 2). Every node in BIONETS *must* implement a subset of these components. The set of mandatory basic components depend on the type of the node as defined in the SerWorks specification [1]. Another constraint is that these components can not be distributed between nodes, nor can be migrated to other nodes — unlike Service Cells in the Service Framework.

The validation of the Service Framework with simulation has fundamental limitations. The important difference from the Networking Framework is the vast amount of possible services that can be implemented compared to the limited domain and flexibility of networking protocols. Also performance measurements of networking solutions are more natural, as the relevant quality metrics are well known and understood. Services on the other hand are directly interfacing with humans which makes evaluation by automated means impossible in general. Given this, we could still imagine specific services that have numerically expressible and calculable quality measures. Such services could be evaluated using the Simulation Platform by using custom extensions. For testing services that have no such quality metrics it is better to use prototypes with real human testers. To help the implementation of controlled experiments, the Simulator Task Force in cooperation with WP5 started work on integrating prototypes into simulations. This will allow human interaction without sacrificing the insight given by simulations.

The integration of the SerWorks architecture was continuously synchronized with WP5 prototyping efforts to ensure the convergence and consistency of implementations. One of the results of this cooperation is the detailed specification of Networking Framework primitives detailed in Section 3.

Figure 1 shows an overview of the simulator and the scope of the simulation efforts and Table 1 summarizes the current status of implementation. The following sections discuss the components shown in the figure, detailing their implementation and simulation strategies.

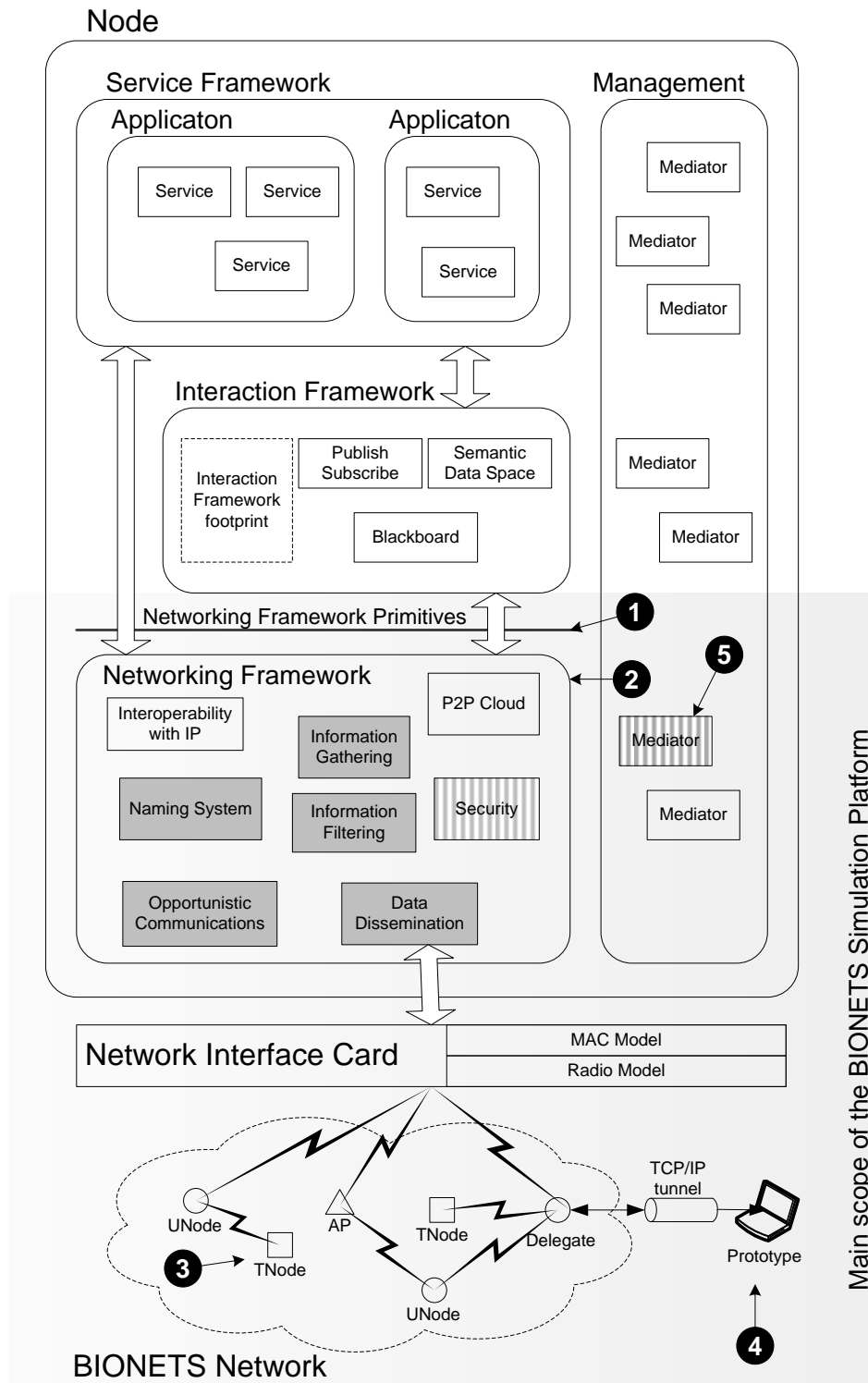


Figure 1: Overview of the BIONETS Simulation Platform. Dark gray indicates the implemented entities and striped gray indicates partial implementation. The annotations show the subsections that discuss the given entities in detail.

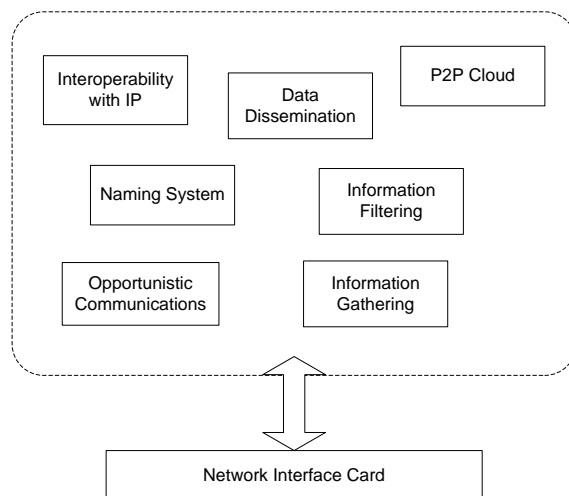


Figure 2: Basic components of the Networking Framework

2.1 Networking Framework primitives

The basic components of the Networking Framework (Figure 2) and their services provide a common language for the implementation of BIONETS networking services. The “vocabulary” of this language consists of the *primitive functions* (elements of the components public API). By using, combining and wrapping these functions a large number of networking solutions and protocols could be assembled. The primitives of these basic components were not defined in the 1.0 version of the SerWorks specification, but were left for later iterations. In order to build simulations and prototypes these interfaces had to be specified. As a joint work between WP1.3 (Simulation and Performance Analysis) and WP5 (Prototyping and Validation) a first set of the APIs for these components were defined. The result of this work is detailed in Section 3.

In itself, the implementation of the primitives of the Networking Framework are not specified, only their semantics as interface contracts. This allows some flexibility on the underlying implementations, and makes it possible to assemble more and more complex services by using some basic building blocks. One example is implementing security. Security does not have an own module in this model. Instead, the security subsystem can *wrap* primitives like `connectTo()`, to provide value-added services, like an authenticated connection, or encrypted channel. This is needed because security is a *cross-cutting concern* meaning that it can not be cleanly decomposed from the rest of the system in both the design and implementation. While all the security related functions could be seen as a module, they are in fact spread throughout the system, deeply integrated with other functions.

To be able to implement such a system, at the lowest level all modules must use the same set of standard protocol messages so that the other nodes in the network could interpret them. This means that there must be an implementation of a service that can not be decomposed anymore. These “atomic” services are basic protocols that are themselves very simple, but allow the assembly of more sophisticated protocols.

The primitives of SerWorks are mostly blocking calls, meaning that the caller entity is suspended by the operating system of the device and placed in a wait queue until the IO operation finishes. One fundamental limitation of discrete event simulations is that they are not very suited for modeling this kind of blocking IO. The problem is, that in order to implement blocking calls, context switches and preempting has to be handled. While some of this functionality is supported by simulation libraries (like OMNeT++), these thread simulation solutions are not scalable and not very stable. For this reason the Simulation Platform uses asynchronous APIs instead of the blocking model. This means that the API defined in the Networking Framework is translated to an asynchronous model, by splitting blocking calls to an invocation part, and a callback part that is used to return the result of the invocation to the caller. The request function — unlike its blocking counterpart — returns immediately to the caller, giving back an identifier called *handle*. The caller could now proceed with his work without waiting. After the request is fulfilled, the IO system calls the callback provided by the caller. The callback is then invoked with the result of invocation, and with the handle of the original request. This handle could be used by the caller to identify which request the result corresponds to.

Module	Primitives	Used algorithms
Opportunistic Communications	Connection.send(), connectTo()	SAF
	doDiscovery(), getNeighbors(), getActiveQueries(), alreadyHasData()	MMSBA (HELLO mechanism), MIOBIO (ADV mechanism), SAF (HELLO mechanism)
	registerNeighborEventListener()	MMSBA (HELLO mechanism), MIOBIO (ADV mechanism), SAF (HELLO mechanism)
Information Filtering	registerFilter(), unregisterFilter()	Filtering algorithms
Information Gathering	read(), registerFilteredListener	Information gathering methods
Data Dissemination	getData()	MIOBIO (REQ mechanism)
	searchData()	MIOBIO(REQ mechanism), MMSBA broadcast
	pushData()	MMSBA broadcast
	advertise()	MIOBIO (ADV mechanism), MMSBA (HELLO mechanism)
Naming System	getAddressByName(), getIdentitiesByName(), getNameByAddress()	MMSBA (HELLO mechanism)

Table 2: SerWorks primitives and their mapping to existing algorithms. N/A indicates where mapping is not applicable. (the parametrization of the primitives is omitted in this table — for details refer to Section 3)

2.2 Networking Framework modules

The previous section discussed the API design of the Networking Framework primitives from a high level perspective. In this section the actual implementation of the primitives are discussed. The current set of Networking Framework primitives were mapped to the algorithms that were developed in previous deliverables ([2]) to provide a reference implementation of networking services. This pairing of primitives and protocols is shown in Table 2. However this correspondence is not one-to-one, as some primitives are mapped only to a subset of features of an existing protocol (e.g. HELLO messages and neighbor discovery). Also this mapping is not necessarily disjoint because some features could be implemented by different algorithms. It is the responsibility of the corresponding Mediator [1] to choose the most appropriate implementation for the desired service.

2.3 T-Nodes simulation architecture

T-Nodes are fundamental in the BIONETS architecture and it is one of the most important goals of the simulation to thoroughly model T-Node operation. While T-Nodes themselves are really simple, their number is huge in a network. This imposes serious scalability problems on the Simulation Platform, making large-scale simulations unfeasible. In order to handle these massive amount of entities the BIONETS Simulation Platform offers two different (but not mutually exclusive) approaches to simulate T-Nodes. Explicit T-Nodes are completely simulated entities without simplifying assumptions that can be used to test T-Node—U-Node communication and are appropriate for smaller scale (≈ 100 T-Nodes) simulations. Virtual T-Nodes on the other hand are lightweight entities that allow large scale deployment (≈ 10000 T-Nodes) by sacrificing some of the simulation accuracy. This is achieved by the use of a global traffic generator that acts as the collection of several T-Nodes. U-Nodes can not distinguish between Virtual or Explicit T-Nodes — they are totally transparent.

Both Virtual and Explicit T-Nodes are stored in a spatial index based on an R*-Tree (which is an improved R-Tree) [3, 4] implementation [5]. This significantly reduces the otherwise $O(n^2)$ time complexity of discovering nodes in a given radio range. This index is populated at the beginning of the simulation with the

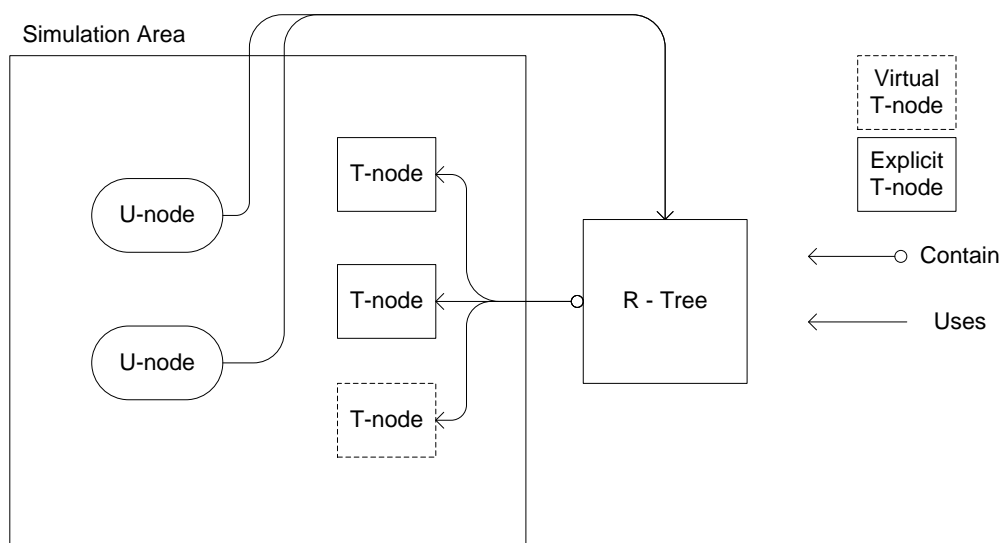


Figure 3: T-Nodes stored in the spatial index

	Virtual T-Node	Explicit T-Node
Traffic	generated by a global traffic generator	individually generated
Internal state	not maintained	individually maintained

Table 3: Comparison of virtual and explicit T-Nodes

coordinates of the T-Nodes that are given in input files. Whenever a transmission is initiated by a U-Node, it queries the spatial index to get a list of the neighboring T-Nodes. This list could include both Virtual and Explicit T-Nodes.

The communication between the T-Nodes and U-Nodes are handled differently depending on the type of the given T-Node. Explicit T-Nodes — as the name suggests — are simulated explicitly without simplification. The whole internal state of the T-Node is maintained during the whole simulation and every T-Node is modeled individually.

Virtual T-Nodes are simple entities having only an ID and an (x, y) position which is read from a configuration file, or it is chosen randomly. The behavior of Virtual T-Nodes are modeled by a traffic generator that is global in the simulator. This means that Virtual T-Nodes are not modeled individually. If a U-Node has for example 3 Virtual T-Nodes in neighborhood, and tries to communicate with them, it will instead communicate with the traffic generator automatically. The traffic generator takes the number of T-Nodes and their distance into account to generate realistic traffic for the U-Node.

2.4 Interface for prototype—simulation integration (hardware-in-the-loop simulations)

One problem of simulations is that they are necessarily simplified compared to a prototype. Another problem is that the characteristics of the simulation platform (e.g. discrete event simulation) constrains the implementation of the architecture, which may be different from the real-world implementation (e.g. blocking calls vs. asynchronous calls). With prototypes — because of their nature — more accurate scenarios could be implemented. The problem with prototypes however that the evaluation of larger scale tests (≈ 100 nodes) is not possible in practice. To overcome this limitation, with cooperation of WP5, a possible hybrid mode is planned which makes it possible to connect the prototypes directly to the Simulation Platform. This will make possible for prototypes to communicate with simulated entities and vice versa. This is illustrated on Figure 5. Because increasing the number of simulated nodes is cheap, the prototypes could be evaluated in larger scale scenarios, including large number of T-Nodes. This is called hardware-in-the-loop simulation and it is especially appropriate for testing prototypes in a mixed environment where prototypes and

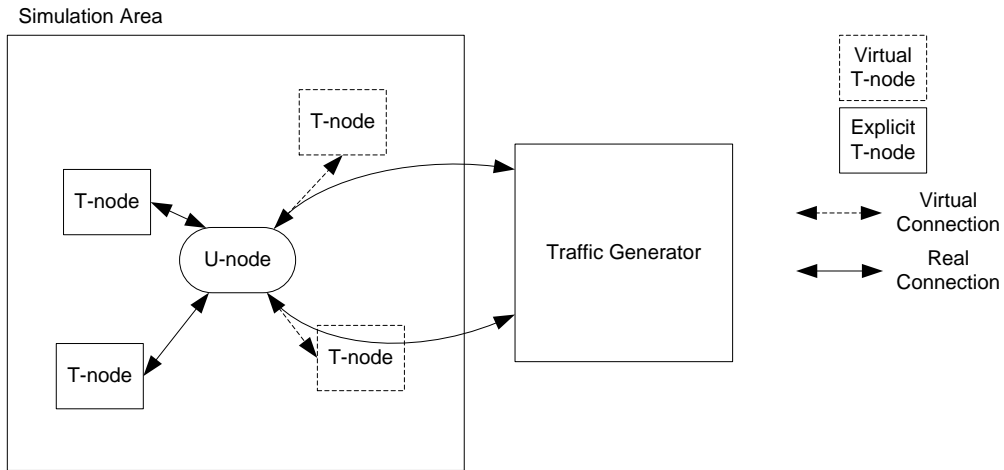


Figure 4: Implementation of Virtual T-Nodes. Communication between U-Node and Virtual T-Node is rerouted to the traffic generator.

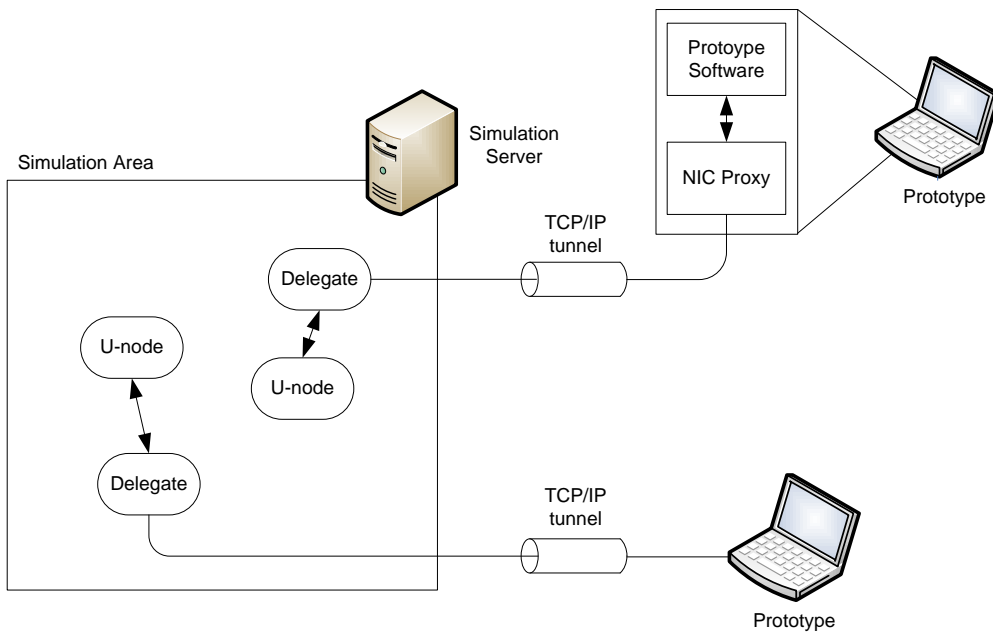


Figure 5: Connecting prototypes to the simulator

simulated U-Nodes can communicate without knowing that the other entity is virtual or real.

2.5 Mediators and support for dynamic composition

One of the innovative aspect of SerWorks is the possibility of dynamic composition of networking services. While at the networking level this flexibility is somewhat limited compared to the flexibility of the services, it still poses several challenges to implementation and simulation. To tackle this complexity, the Simulation Platform defines 3 compliance levels that represents increasing complexity from partial support to full implementation:

1. The networking level functionality is assembled from the eight predefined module by the user of the simulator before running the simulation. Different nodes may use different user specified configurations, but the configuration is fixed during the simulation run.
2. The user specifies several static configurations in an input file. During the simulation, the nodes may switch from one predefined configuration to the other, or use them parallely, but can not create new configurations.
3. Fully dynamic operation. Every node is able to programmatically create new configurations during simulation and switch to any of them at any time.

Compliance levels will be implemented step-by-step, adding new features incrementally. This will allow continuous testing of the BIONETS Networking Framework without waiting for every feature to be available. This also minimizes the impact if some functionality turns out to be not feasible to implement with the underlying technology. At this point, the Simulation Platform has compliance level 1, which already allows a great deal of experimentation.

3 BIONETS Networking Framework API specification

The BIONETS Networking Framework (BNetFW) is a dynamically programmable network stack that can be tailored to the specific needs of different services. The BNetFW consists of a fixed set of functions instead of following the fully dynamic nature of service composition in BIONETS. These hardwired functions – the *primitives* – are the backbone of the networking system in BIONETS.

The primitives are grouped into classes to encapsulate the implementation details, therefore acting as black boxes, so they could be safely replaced by other modules implementing the same interface. The combination of these building boxes are done by the programming pattern Inversion of Control (IoC). This means that no class should directly instantiate another class in the Networking Framework, but instead provide a method that accepts an instance of the needed class as parameter, so that an external configuration entity (sometimes called IoC container) could provide it. This reduces the coupling between modules.

Consider a module called `ExampleModule` that needs the service provided by another module `ExampleServiceModule`. The naive implementation could be that `ExampleModule` somewhere in his code calls `new ExampleServiceModule()`. This is bad however, as the type of the used service is hardwired after the `ExampleModule` has been compiled and therefore there is no way to change it during runtime. A better approach is that `ExampleModule` has a method called `setExampleService(ExampleService service)`. The `ExampleServiceModule` implements the `ServiceModule` interface, therefore it could be provided to the `ExampleModule` through the "setExampleModule(...)" method. The gain is however that any type of service could be injected in the `ExampleModule` provided that it implements the `ExampleService` interface. A nice example could be a `SecuredExampleServiceModule` which is a security enhanced implementation of the original service.

The primitives are grouped according to the classification presented in the Serworks Architecture 1.0 document.

3.1 Opportunistic Communications

3.1.1 Pairwise Communications Channel Unit

- `Connection connectTo(Address otherUNode)` – Connects to a U-Node with a given `Address` and returns with a `Connection` object. Connections are *soft* as they do not need to be closed explicitly but they are closed automatically after a given time of inactivity, or on loss of communication (no ACK packets in given time).
- `Status Connection.send(List<Data> payload)` – Sends out data on the established `Connection` and returns a `Status` object that indicates if the transmission was successful. May contain back a vector indicating which `Data` were transmitted successfully

3.1.2 Neighbor Discovery Unit

- `List<NeighborInfo> doDiscovery()` – Initiates a one-shot discovery of neighboring nodes. The function returns with a list of `NeighborInfo` elements that contain the following information. The Neighbor Discovery Unit may cache the neighborhood information between requests to reduce channel load.
 - `List<Address> getNeighbors()` – returns the neighbors of the given node (2-hop neighborhood)
 - `List<Query> getActiveQueries()` – returns a list of queries that the given node is interested. Neighboring nodes should automatically relay information that satisfies these `Query` items. Active queries are not mandatory to publish and any node could decide not to respond to active queries.
 - `List<DataId> alreadyHasData()` – with filling this list, a node can indicate that it is not interested in the `Data` corresponding to the listed `DataId` items. This is useful for reducing the channel load when implementing flood like services, for example searches. This list is optional in the packets, and a node could decide to fill it up with only recent entries omitting older ones to reduce the size of the packet.
- `void registerNeighborEventListener(NeighborEventListener listener, int period)` – this method registers an event listener that receives updates through the callback functions `neighborDetected(`

`Address node`) and `neighborLeft(Address node)` that are defined in the `NeighborEventListener` interface. The Neighbor Discovery Unit will refresh the neighborhood table periodically where the period is given by the `period` parameter in milliseconds. If multiple listeners are registered, the lowest period will be used. The period is not guaranteed however, as the Neighbor Discovery Unit may jitter the discovery messages to reduce collisions on the channel.

3.2 Information Filtering

- `void register(Filter filter, Policy policy)` – Registers a filter that has a callback method `Vote filterData(Data data)` which is called periodically for each `Data` item and returns a vote that could have the values `remove`, `keep`, or `indifferent`, indicating that the message should be removed or not. `Policy` could take three values: `consent`, `democratic`, `veto`.
 1. `consent` means that each of the filters must agree on the removal of the message from the pool
 2. `democratic` means that the majority of the filters should agree on the removal of message
 3. `veto` means that if any filter nominates a `Data` item for removal it is then removed immediately

If multiple filters with different policies indicate that a `Data` item should be removed then the **strongest policy of the filters that voted (i.e. not indifferent)** is applied. This method with the different policies allows to resolve potential contradictions in filters. The recommended default policy should be `democratic`. In case of excessive storage use, the Information Filtering module should switch to `veto` policy regardless of the actual custom policy of the filters. * `void setFilterPeriod(int period)` – sets the period of filter invocations in milliseconds

- `void unregisterFilter(Filter filter)` – Removes the given filter from the invocation chain

3.3 Information Gathering

This module is responsible for gathering environmental data from T-Nodes. It leverages the U-to-T-Node interface present in all U-Nodes. It can be used for both pulling data from T-Nodes (sending requests for a specific type of data) as well as for receiving data pushed by T-Nodes (applying an appropriate filter on the incoming type of message, according to the current set of interests). T-Node push is not implemented in the case of *Virtual T-Nodes* in the simulation. This is a modeling trade off to achieve better scalability. However, push is implemented by *Explicit T-Nodes*. The Information Gathering module is characterized by the following primitives:

- `List<Message> read()` – This primitive is invoked for proactively performing a search of data coming from T-Nodes in close proximity. It returns an array containing the messages received from all the T-Nodes in the communication range.
- `void registerFilteredListener(Filter filter, TNodeMessageListener listener)` – As data originating from T-Nodes can be potentially filtered, this method sets the filtering policy that is applied by the Information Gathering module. A `Filter` contains the list of `Interests` the user (or service) registered to. The `Filter` is set per listener, so the interested module will be notified only on the arrival of relevant messages. It is necessary to have different `Filters` for different clients, because multiple services may want use different configurations of the Information Gathering module.

3.4 Data Dissemination

This module is responsible for ensuring the dissemination of data system-wide. The data dissemination in BIONETS relies on the movement of U-nodes and on a peer-to-peer data exchange. The U-nodes, after having read the information provided by the T-nodes, undertake the task of information dissemination. Information dissemination refers to sharing the whole or part of the information with other U-nodes in the area, which may be interested in the information that is spread. The data dissemination module includes a set of policy that defines the behavior of the node when getting in contact with another U-Node. Policies include also measures related to the reputation index of the node getting within communication range. Policies have to be engineered in such a way to ensure a proper trade off between efficiency (efficient usage of resources available in the system) and utility (perceived by users as their ability to access information of interest).

- `List<Data> getData(Query query, Address UNode)` – This primitive is invoked in order to retrieve data from a single node. In particular, when two nodes meet, they exchange a Query which contains a high level description of the information each node is looking for. Such Query is intercepted by the Data Dissemination module, and is used for retrieving data currently stored in the internal memory of the receiving node. Such data retrieval process is driven by a specific forwarding scheme, and determine the data that is sent back to the encountered node.
- `List<Data> searchData(Query query, SearchListener listener)` – disseminates the given Query in a connectivity island exploiting the broadcast medium. On reception of answer the registered listener is invoked.
- `void pushData(Data data)` – disseminates the given Data in a connectivity island exploiting the broadcast medium.
- `void advertise(Advertisement adv, RequestListener listener)` – the node advertises a list of available Data. The request messages from surrounding U-Nodes are collected and are given to the listener in one pass. The listener is presented with a `ResponseChannel` object which encapsulates the algorithm that is used to send out requested Data. An implementation of the `ResponseChannel` may use pairwise connections to send the Data to the nodes one-by-one, or may exploit the broadcast channel to minimize channel load when responding to multiple nodes.

3.5 Naming System

BIONETS relies on names for identifying communicating devices. *Names* in BIONETS are intentional and are defined as a set of pairs `<attribute,value>`. Names are location-independent identifiers, i.e., they have global spatial scope and do not change as the node moves in the system. Names have limited temporal scope, i.e., they might change over time. All nodes in BIONETS have a name. Names are not unique. Names can be used for taking decisions concerning information/data forwarding, consistently with the data-centric nature of BIONETS networks. Nodes which are subject to trust and reputation systems (i.e., all U-Nodes and some classes of T-Nodes) possess a unique static identifier called *identity*. The identity of a node has global scope in space and time. It represents a fingerprint of the node, and it is expressed as a numerical value. We assume that node identities are hardwired in the nodes by the manufacturer. Identities are not used for taking forwarding decisions, and are not exposed to the network framework. Identities are exposed to trust and reputation services only. BIONETS encompasses also the use of identifiers with local scope in both space and time, which are termed *addresses*. An address is constituted by a numerical value, which can be associated to U-Nodes and to some classes of T-Nodes. Addresses are unique within a two-hop neighborhood. Addresses are generated locally according to a random procedure, coupled with mechanisms for resolving collisions.

Primitives that are provided by the Naming System:

- `List<Address> getAddressByName(Name name)` – gives back the Address-es of U-Nodes in a 1-hop range that has the given name. Returns a List because names are not necessary unique.
- `List<Identity> getIdentitiesByName(Name)` – returns the identities of U-Nodes with a given name in 1-hop range.
- `Name getNameByAddress(Address address)` – resolves the Name of the U-Node with the given Address. Address-es are unique in 2-hop range therefore the primitive returns with a single Name instance. In case of transient Address collision the return value is unspecified and may differ in different implementations.

3.6 Secure Communication (as described in D1.1.3/3.1.3)

Secure Communications. This module is responsible to ensure secure communications between nodes on the network layer. On T-node level we identify different security classes which depend on the resources available to perform security relevant operations. Security modules to ensure secure communication on network level for U- and AP-nodes are mandatory. They provide functionalities required for authentication, confidentiality, and integrity of the network traffic and ensure the secure operation of, for example, the service life-cycle and the SDS.

The security modules may be implemented as Proxy or Decorator classes wrapping the original unsecured classes and providing security services transparently. One example could be a `TrustedNeighborDiscovery` that wraps the standard `NeighborDiscovery` and hides the neighbors that are not trusted from the user of the service.

4 Simulation Scenario Draft

The aim of this document is to integrate simulation efforts into a standardized simulation scenario. It should identify features of BIONETS that can and should be simulated in an integrated way.

4.1 Goals of the simulation

There are many possible goals for a simulation of a complex system. During early phases of development usually parts are simulated separately, and their behavior is analyzed typically by changing several parameters and observing the effects.

In later phases simulation of the whole system can be used to test the integration of the components. In these simulations the analysis approach is different from that of the isolated scenarios. Because the parameters of the system consist of the parameters of the components the parameter space becomes unmanageable, making parameter sweeps unfeasible. If we also want to allow heterogeneity in these values, the total simulation becomes impossible. However the focus of an integrated scenario is to show that the components work effectively together and they operate close to the "working point" that was analyzed before. Therefore it is very important to carefully harmonize simulation efforts so that they

- analyze components in a standardized way, sweeping through interesting parameters and identifying a "working point"
- validate the expected features of the BIONETS systems, in terms of "level of autonomy". This can be done in 2 steps:
 - qualitative (can the system change its internal configuration to adapt to unpredictable working conditions?)
 - quantitative (how "well" can the system change its configuration, in other words, how good is it at converging to the changing optimal operating point?)

4.2 Abstract entities of the Simulation

This section identifies the high level, abstract simulation entities of BIONETS. The division between services and networking is a logical one and they will converge to SerWorks.

4.2.1 Environment

Environment is everything that is outside the designer's control. This includes both low level and high level aspects. Typical low level elements of the environments are the mobility pattern of users, technical details of user devices (channel bit rate, battery life, transmission power). High level elements are the applications that will be executed on the devices. BIONETS sits between these two environment aspects, providing an adapter layer between them, translating user needs into low level device primitives and vice versa.

4.2.2 Special simulation needs for Environment entities

The most important issue of the environment is that it is essentially unknown. This means that a simulated environment must be designed very carefully to account for future unpredictability.

One possible method to deal with this issue is to specify a wide range of simulation scenarios and test the system in each of them. This can be achieved for example by sweeping through parameters. As we discussed before, this becomes unmanageable because of parameter space explosion. Another approach can be to design a heterogeneous scenario, where different elements with different parameters are present at the same time. A heterogeneous scenario is more realistic but it makes evaluation of the results harder.

4.2.3 Services

Service entities handle user requests and try to execute them using Networking entities when it is appropriate. Simulation of service entities is a complex task, because they operate close to the High Level Environment, which includes human interaction. Performance of services is not easy to measure mostly because it is subjective. This kind of "performance" can be tested by humans using the service and completing a survey. Specific services can be tested more formally, where appropriate performance measures are available.

4.3 Description of the simulation scenario

Simulated area The simulation area should be large enough to allow modeling different device densities and different local mobility behavior. However it should be small enough to be feasible to simulate the whole scenario. *Recommended area size: 1500x1500m*

Number of U-Nodes The number of nodes should be large enough to verify scalability and stability promises of the system. *Recommended U-Node number: 500-1000*

Number of T-Nodes T-Nodes can be simulated explicitly by placing separate simulation entities in the area or implicitly by using traffic generators. The first allows a stricter verification; the latter allows larger scale simulations. *Recommended T-Node number: should be about 10 times the number of U-Nodes 5000-10000 (Only if virtual T-Node are used. Explicit T-Node simulation are less scalable, only 50-100 explicit T-Node can be used.)*

Mobility Mobility is very difficult to simulate in a realistic manner. It is shown that several algorithms perform quite differently in different mobility environments. Therefore it is essential to use a heterogeneous mobility environment, which is more realistic than using a single model.

Recommended mobility model:

- Reference Point Group Mobility Model with Dynamic Groups (RPGMMDG) [see D1.3.1]

Recommended parameters for the mobility model::

- Building \approx 20% of U-Nodes
 - 1m/s when moving
 - staying in place for 5 minutes on average
 - moving in groups
 - size of groups: 5 U-Node
- Pedestrians \approx 60% of U-Nodes
 - 1m/s when moving
 - staying in place for 10 seconds on average
 - 20% of them (12% overall) moves in groups
 - size of groups: 5 U-Node
 - 80% of them (48% overall) moves independently
- Vehicles \approx 20% of U-Nodes
 - 15m/s constantly
 - moving along predefined paths

Initial distribution of positions A worst-case assumption can be taken, therefore the initial positions of the U-nodes should start with a uniform distribution (uniform is worst-case as it is the one with the smallest probability of having someone else in range).

The simulated time-interval *Recommended time: 2 simulated hours*

Physical layer properties *Recommended physical layer*

- Frequency: 2.5GHz (ISM band)
- Thermal noise: -120dBm
- Path loss coefficient: 4 (urban areas with fading)
- Raw bit rate: 54Mbit/s
- Transmitter power: 1mW
- Sensitivity to carrier signal: -100dB
- Airframe header length: 16bit
- Radio switch time between SEND and RECEIVE mode (half-duplex operation): 0.1ms

Media Access Control *Recommended media access mode*

- *Simple CSMA media access*
- *Signal level to consider the medium busy: - 90dB*
- *Contention window: 31*
- *Maximum transmission attempts: 7 (because of epidemic style message forwarding, retransmissions should be handled in the upper layer; this must be set to a higher number if no explicit backoff is implemented in the higher layer)*

4.3.1 Detailed simulation output

Measurements for Networking entities Measurements are aggregated during the whole time of the simulation (excluding warm-up):

- Total number of messages sent (completely transmitted on channel)
- Total number of messages successfully delivered (received without error)
- Total number of useful messages delivered (received without error and useful for receiver /not duplicate and on the interest list of user/)
- Total number of messages lost due to collision
- Average delay between source and destination
- Measurements that are collected as a function of time
- Visualization of adaptive networking solutions (like showing the distribution of an optimized parameter, or showing how many nodes use a given version of a dissemination algorithm)

Evaluation of results It is mandatory to use a 95% confidence interval in the process of evaluating the simulation results.

5 Conclusions and Next Steps

In this deliverable the components of the new architecture of the BIONETS Simulation Platform has been presented, which is fully compatible with the SerWorks specification and principles. This was the outcome of a tight cooperation of the Simulation, SerWorks and Prototyping Task Forces. The SerWorks enabled Simulation Platform gives us the opportunity to verify and test the properties of the overall BIONETS architecture on a wide scale, and evaluating not only the different networking solutions, but also some of the service- and security-level functionalities of the project.

While having the advantage of simulating large systems, a simulation has the drawback of simplifying the system properties compared to a prototype. Considering this, one of our future plans, in cooperation with WP5, is to create a hardware-in-the-loop mode, which makes possible to connect prototypes directly to the Simulation Platform. This will enable prototypes to participate in simulated scenarios and to communicate with simulated entities. Because simulations are much cheaper than using real devices, larger scale scenarios become feasible, including large number of T-Nodes.

A Map of theoretical models of BIONETS

In this section we will provide a self-contained description of the theoretical framework developed to model and assess the performance of BIONETS system.

As it will appear clear in the following, it is actually possible to draw a common thread along the past activities in the project. In particular, investigation on the system scalability has driven several efforts of the BIONETS research. Scalability was characterized theoretically from different perspectives. To some extent, this section provides a retrospective look to some of the proposed models.

In particular, the scalability of the system was assessed in two opposite situations, e.g., in case of very dense networks and in the case of very sparse networks. At those ends, in fact, asymptotic analysis is possible, using appropriate mathematical tools. We remark that the simulation results are not a viable solution when testing scalability over scales systems, either due to memory or time limitations. In such cases, the analytical approach is the only viable method to obtain reliable performance evaluation.

Starting from these two opposite sparse and dense scenarios, we can draw a unitary framework, which finally reflects into a general taxonomy.

In the dense network scenario, the density of devices, either U-nodes or T-nodes, is large enough to permit transmissions with no need for delay-tolerant mechanisms. In such cases, in particular, we understand that mobility plays a minor role in the transport of data since connectivity is present: communications can occur for example through standard gossip mechanisms. Note that this scenario is rather conventional compared to the sparse one, but, nevertheless, several novel tools have been identified to deal with massively dense ad-hoc networks. In the sparse scenario, conversely, connectivity is unavailable most of the time, so that delay tolerant mechanisms become relevant. The main branches of the techniques devised so far in the project are reported in Fig. 6. We provide a description of the research topic and results in the following.

A.1 Dense Scenario

- **Routing:** spatial models from *electrostatics* and *electrodynamics* have been used for describing the optimal paths followed by packets routed in dense ad-hoc network: the limit, as the density of the network becomes large, can therefore be described and its performance computed from simpler continuum models. The standard references for such research are [6, 7], where an *optical* analogy is used in order to derive message routing laws that resemble light propagation in a dielectric medium. Other approaches employ models that emerge as limits of *models in road traffic* [8]. Within the project, several results were obtained in such domains [9, 10, 11].
- **Connectivity over Space:** it is known that under finite coverage radii, the density of the nodes in a given area able to guarantee connectivity is ruled by interesting phenomena related to percolation models, such as the emergence of threshold effects. In particular, connectivity properties of a cluster of mobiles occupying a given area can be described using the theory of *continuum percolation*; the standard reference in the field is the book [12]. In particular, 2-D spatial coverage processes have been employed in BIONETS in order to model the properties of connectivity in presence of channel randomness [13].
- **Capacity:** techniques coming from information theory, aided by the use of specific mathematical tools such as random matrix theory, can be applied in order to describe channel capacity. Within the project, a survey on random matrix theory and applications is reported in [8]. Further results on random matrices and their application have been described in [14, 15]; studies in [16] were used to characterize the performance of the T-nodes to U-nodes communications in terms of capacity and spectral efficiency.

A.2 Sparse Scenario

- **In-Network Computing:** some works in literature [17, 18] recently described a novel paradigm that merges the concept of computing and communications in order to achieve lower communication overhead while performing distributed computations over a network. Also, some other works explored the algorithmic behind consensus and gossip, see for example [19, 20]. The in-network paradigm fits both the T-nodes level and the U-nodes level. At the U-nodes plane, in particular, aggregation and population protocols as those reported in [19, 20] can be used leveraging mobility. In the project, research on those techniques has been performed in order to collect data related to the system operations, e.g. the number of nodes running a given service [21].

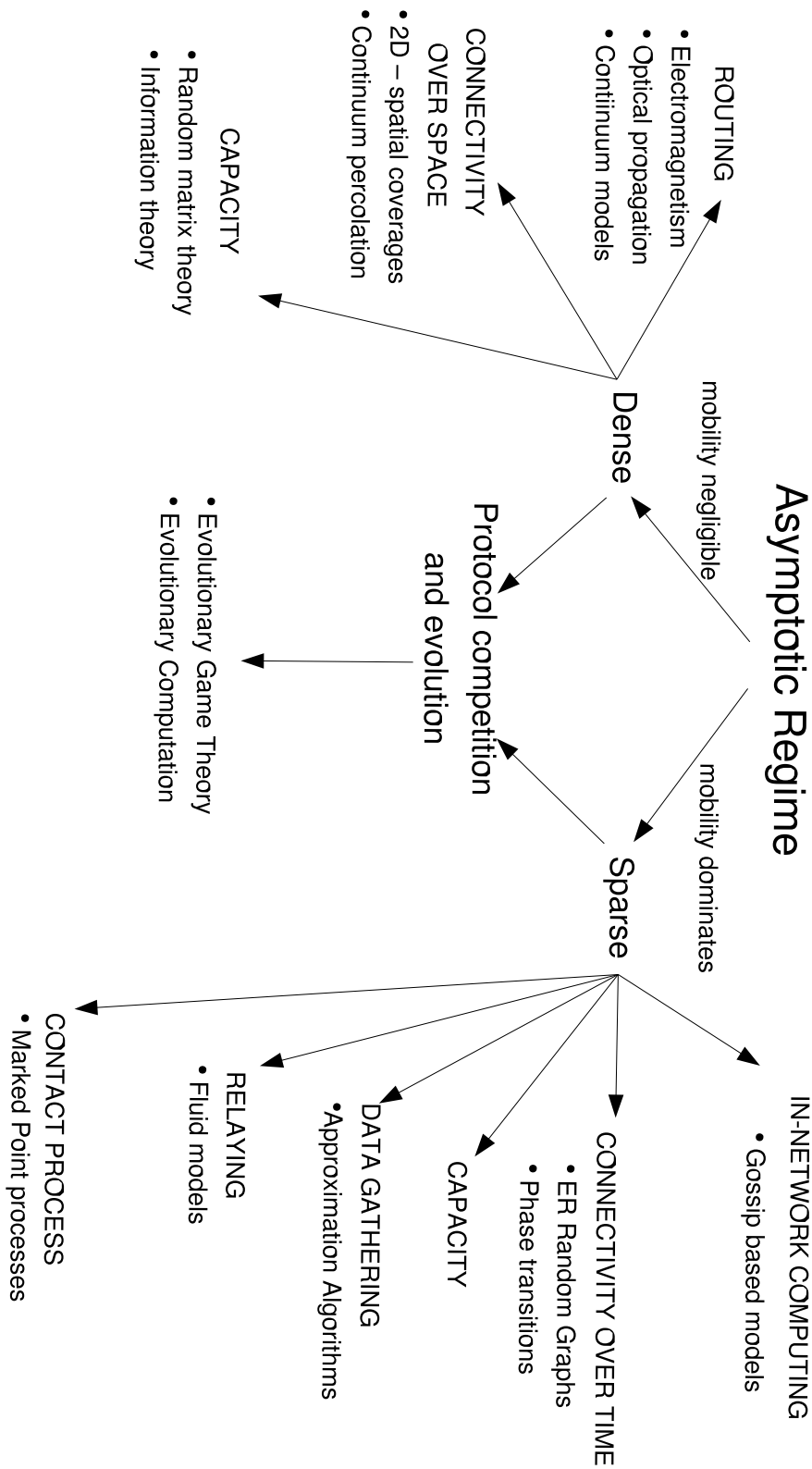


Figure 6: A taxonomy for the toolkit to assess asymptotic performance of BIONETS systems.

Scenario	Topic	Approach	Tools	Original Results
dense	Routing	Electromagnetism	Differential Equations.	[9, 10]
dense	Connectivity over Space	Percolation Theory	2-D Covering Processes	[13]
dense	Capacity	Information Theory	Random Matrices	[16]
sparse	In-Network Computing	Gossip Algorithms	Population and Aggregation Algorithms	[21]
sparse	Connectivity over Time	Random Graphs	Erdős-Renyii graphs	[23, 24]
sparse	Contact Process	Marked Point Processes	Statistical Tests	[26, 27]
sparse	Relaying	Fluid/Discrete Models	Optimal control	[31, 25]
sparse	Relaying	Discrete Models	Stochastic approximations	[32]
sparse	Capacity	Information Theory	-	-
sparse	Data-gathering	Optimization Theory	Approximation Algorithms	[30]

Table 4: Brief view of the original results obtained

- **Connectivity over Time:** in the sparse scenario, the possibility for messages to reach nodes depends on the contact patterns among nodes. As a consequence, the notion of connectivity is different from what is seen in traditional networking literature; in the homogeneous cases, connectivity properties are described by Erdős-Renyii graphs [22]. Threshold effects related to over-time connectivity are showed in some works developed in the project [23, 24, 25].
- **Contact Process:** in the sparse scenario contacts among nodes occur at low rates; in this case, a point process model proved extremely effective in capturing the major feature of pairwise contact patterns. One such example is the MM2 model developed and validated in [26, 27] within the project.
- **Capacity:** the main related work for the capacity of mobile ad hoc networks is the paper [28]; the main result there is that mobility is proved able to increase the capacity ad hoc networks up to $\Theta(1)$ in the number of nodes. The better scalability compared to the static case (see [29]), is paid in terms of delay.
- **Data gathering:** the problem of data collection, when several T-nodes act as sources of information, is how to make optimal use of finite contact times in order to generate the maximum benefit for the U-nodes gathering the T-nodes information. The problem is discussed and proved \mathcal{NP} -hard in [30]: approximation algorithms are provided there with provable bounds on the complexity in the number of nodes.
- **Relaying:** in the sparse scenario, the strategies for messages delivery are constrained by the fact that the network is disconnected most of the time; a recent research line in BIONETS, in particular, addressed the control of the message forwarding such in a way a way to maximize the delivery probability within a given deadline and to minimize the energy consumption [31, 25, 32].

A.3 From analysis to design: the problem of control

The specific aspects of BIONETS described before characterize in depth the performance of the system with respect to some metrics. Typically, this occurs with asymptotic analysis, where transients are neglected with respect either to the time or the space domain.

In general, though, the goal of the system designer is to bring the system to a desired operating point in terms of performance. Such optimization, clearly, would require acting on some/all aspects described before in order to search a convenient (and possibly optimal) operating point of the system. Moreover, driving the system towards such a point should occur in a distributed fashion.

Related works developed within the project indicate that the control of the system, at the network level, could occur via a general bio-inspired paradigm that we call “protocol competition”. In this scenario, several variants of networking protocols could be run in parallel, and selection would be operated at run time. So far, the two major mechanisms to operate protocol competition and selection are

- **evolutionary computation:** some specific techniques described in [33, 34], showed already capable of on line optimization of forwarding policies. The assumption in order to leverage this technique, is that the selection pressure is made among variants of the same basic algorithm;
- **evolutionary game theory:** using evolutionary game theory, it is possible to study the effect of protocol competition and derive distribution of the strategies adopted by nodes in the network; a fundamental feature of the theory is that in some cases it is able to predict which variants are going to coexist in the system [35, 36, 37].

B Integration of the Situated Adaptive Forwarding

SAF can be partly considered a generalization of the K-copy algorithm. The nodes estimate the delivery probability of the neighboring nodes for a given destination. This delivery probability is estimated based on nodes' current network neighborhood as well as their past engagements with other nodes. Also, the estimation process is influenced by the category of the message payload, connectivity of the nodes and their resources. Available battery, storage and CPU resources weighted with the current selected usage profile are taken into consideration. The nodes periodically send HELLO messages which contain information described above of the sender node. The nodes store these information in a Network Situation Manager (NSM) module.

Some new features of the SAF are listed here with a brief explanation which required refactoring of the Simulation Platform:

- Two-hop neighborhood information gathering: To gain view of the situation at hand, the information from the network neighborhood is gathered within two hops. This information is stored in the newly introduced NSM module.
- Delivery probability estimation: For sending direct messages, estimations of which nodes are good candidates for forwarding are provided also by the NSM module.
- Resource information gathering: How much resources are available for use are available from the corresponding modules. Other nodes' resource situation is available from NSM.
- Direct sending of messages: As described in table 2, *send()* and *connectTo()* have been introduced to offer SAF's functionality as SerWorks primitives.

Originally SAF was developed with a different simulator code base than the official Simulation Platform. As the modules were already written for OMNeT++, a complete re-write of the code was not necessary for the integration. Still, — as mentioned before — the architectures had differences in the already implemented SAF and in the existing BIONETS simulator, thus some modifications were needed for the integration.

To accommodate the SAF networking functionality, the SerWorks network primitives were created first. These represent abstract functionalities and thus are implemented as compound modules. So each SerWorks primitive contains certain implementations of the SAF framework, as it is described in table 2 and detailed below.

- Pairwise Communication Channel Unit: This module is to allow the U-nodes using SAF to send messages directly to other nodes. To implement this feature we integrated the SAF's NSM Module into this unit.
- Neighbor Discovery Unit: The primitives that serve this feature use HELLO messages to discover neighbors. As we already implemented HELLO messages for MMSBA, those were extended with the extra fields of the SAF hello messages.

The main achievement of the integration is that all networking functionalities of the information dissemination algorithms can be used in the same simulation scenario.

C Integration of the Cooperative Content Retrieval scheme

C.1 Introduction and Objective

The aim of the simulation is to model a cooperative content retrieval scheme in a BIONETS scenario, where U-nodes have different interests for information objects stored in T-nodes, and hence are not immediately interested in retrieving those objects from the T-nodes, unless they incur some other benefit. Here, we consider benefits are incurred from bilateral object exchanges. That is, a U-node may copy an unwanted object from a T-node node in order to exchange it with content of its interest from another U-node.

A theoretical analysis of such a scheme was presented in [38]. A description of the scheme and main results are also included in BIONETS Deliverable No. D2.2.5.

C.2 General Simulation Scenario

We consider a graph model of the network, as shown in Fig. 7. T-nodes are located on the vertices of the graph and collect information about states or variables of the environment. U-nodes move randomly along the edges of the graph, collecting information when they reach a T-node, or upon meeting another U-node somewhere on the graph.

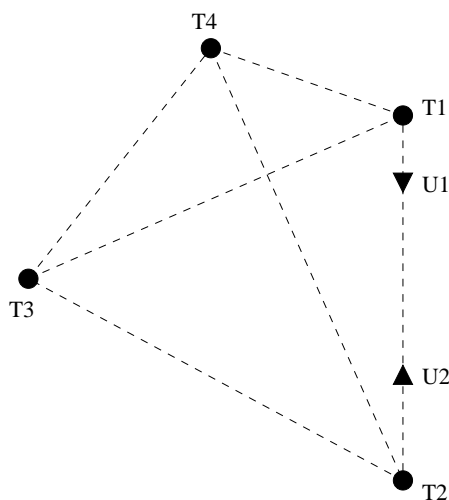


Figure 7: Example of the network graph

There exists a discrete set of U-node classes. Each class specifies a set of U-nodes with common interests for information gathered by T-nodes. We consider that each T-node collects and stores a single type of content or information object, and hence each class also specifies a subset of T-nodes that U-nodes in the class are interested in.

We postulate that the mean information retrieval cost for each U-node can be reduced, if U-nodes are cooperative and collect information not only in their own interest but pass it to other nodes. Formally, this is seen as a game of pairwise interactions between U-nodes, that lasts a (random) number of rounds.

Strategies that can enforce such cooperation are the following:

1. *Immediate punishment*: "Each U-node initially behaves cooperatively and copies unwanted content from a T-node. When two U-nodes get in communication range with each other, they exchange their lists of objects in their memory. If from this communication a U-node perceives that the other U-node has behaved selfishly and has not collected content from a T-node it has visited, it does not transmit its acquired content."
2. *Tit-for-Tat*: "In the first round of the game, each U-node makes a specific personalized move (either cooperative or selfish). Upon meeting the same U-node in a subsequent round, each U-node makes the same move the other U-node did in the previous round."

Refinements of these strategies are further discussed in the next section.

C.3 Implementation Details

C.3.1 Number of U-nodes and T-nodes

The number of T-nodes should be small, to limit the number of edges in the graph. It is suggested to employ a few tens of T-nodes (up to 100 T-nodes). The simulation area should be seen as a “local” area that U-nodes frequently visit. The number of U-nodes should also be small; the aim is to simulate an environment with sporadic encounters between U-nodes. It is recommended that the number of U-nodes be an order of magnitude less than the number of T-nodes (up to 10 U-nodes).

C.3.2 Classes of U-nodes

A small number of U-node classes must be specified. To each class we shall manually append a set of affiliated T-nodes, that contains the T-nodes whose content is of interest to this class. These sets can be overlapping.

C.3.3 T-node information objects

Each T-node generates a single type of data, or information objects. Each information object will be recognized by the ID of the T-node that generates it and the time of generation.

C.3.4 Construction of the network graph

Random graphs should be employed. A usual construction is to place a number of T-nodes according to a uniform or spatial Poisson distribution, and create a link between each pair of T-nodes with a probability p . The link will represent a physical route. If n is the number of T-nodes, then we must have $p > \log n/n$ for connectedness.

Alternatively, grid topologies can be used to represent more structured paths, such as streets in a city. Link distances can then be drawn from some distribution (e.g., uniform).

C.3.5 Mobility model of U-nodes

U-nodes move according to a random waypoint model on a subgraph of the original network graph. For each class of U-nodes, the subgraph consists of the T-nodes the class is interested in, and the shortest paths between each pair of these T-nodes. All intermediate nodes in these shortest paths are also included in the subgraph.

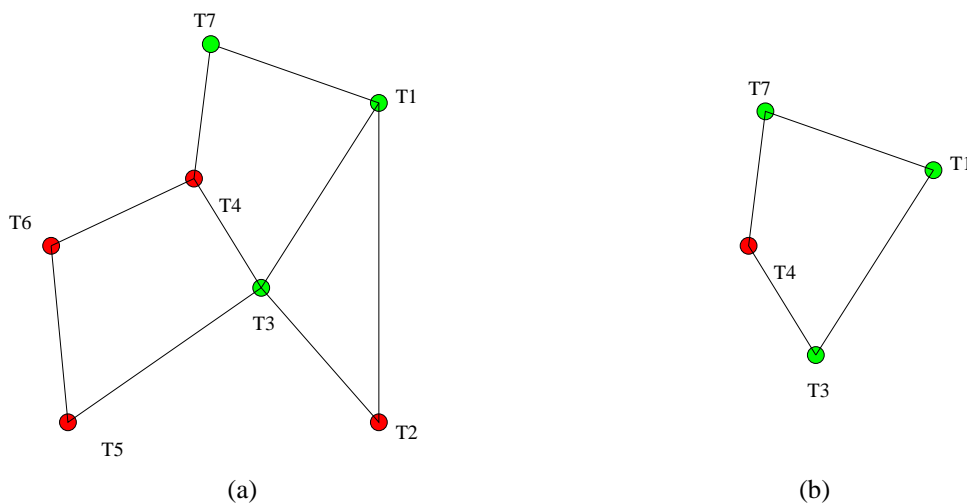


Figure 8: A network graph (a) and the subgraph (b) induced by the interesting T-nodes (in green, or lighter gray for b&w) and the shortest paths between those.

For example, consider the network graph of Fig.8(a), and suppose that a U-node is only interested in visiting T1, T3, and T7. Then it will move in the subgraph specified by these T-nodes and the shortest paths (measured in absolute distance, or number of hops) between these T-nodes. In the subgraph induced in this way, shown in Fig.8(b), the U-node will also visit T4, in whose content it is not interested.

In the random waypoint model, the T-nodes are located on the vertices of the subgraph. Transitions from one waypoint to the next are governed by a Markov chain, i.e., a U-node moves from waypoint T_i to waypoint T_j with a probability $p(T_i, T_j)$.

We shall primarily consider that U-nodes perform random walks in their subgraphs. If within the set of interesting T-nodes a U-node favors specific T-nodes, we can append higher probabilities to incident edges of these T-nodes.

Other parameters of the random waypoint model are the following:

- *Velocity of a U-node*: This can be taken a constant v . More generally it is a function $v(r, T_i, T_j)$ of the distance r from the origin point T_i to the destination point T_j .
- *Pause times at T-nodes*: These can be taken constant, or more generally a random variable drawn from a certain distribution with finite mean.

C.3.6 Transmission ranges

The transmission range of a low-power sensor (T-node) is about 50m, while that of a U-node is about 200m. The exact values depend on modulation and coding schemes, as well as channel conditions, but don't need to be specified in a simplified simulation.

Furthermore, in a simplified simulation we can ignore interference phenomena. Overhearing will not take place, even if more than two U-nodes are in the range of each other. This can be assured by appropriate security mechanisms. We can also assume that data exchanged between U-nodes or between a U-node and a T-node consist only of a few bytes; thus they are transmitted within a very small time interval which can be neglected.

C.3.7 Utility functions

Costs can be expressed in time or energy units. The main performance metric of a U-node is the delay to retrieve an object. This is the delay from the generation of an object until the U-node retrieves it. The utility function, which a U-node wants to maximize, can be defined as the number of objects of interest collected in an interval of time T .

Additionally, we can calculate the energy consumed by U-nodes and T-nodes during a specified interval of time. If U-nodes are cooperative, it is anticipated that they will have increased energy consumption, because of more interactions. However, there can be less interactions and hence less energy consumed by the T-nodes, which are more critical energy-constrained devices.

C.3.8 Updates of T-node data

T-nodes periodically get new measurements or messages about the states or variables of the environment. U-nodes are typically unaware of the times at which new measurements are generated; they revisit T-nodes at random instances according to their mobility model and get any updated content, if there is.

C.3.9 U-node memory management

U-nodes have a finite buffer in which to store objects. They can apply any buffer management scheme; typically, since T-nodes collect measurement type data which are depreciated over time, a U-node will apply a FIFO scheme.

C.3.10 U-node perception of a selfish strategy

The way a U-node realizes that another U-node has behaved selfishly is crucial, and determines the optimal strategy it should follow. A simple way one can perceive selfishness goes as follows: each U-node knows the topology of the graph, so that when two U-nodes meet somewhere on an edge of the graph, each U-node

knows (at least) the last T-node the encountered U-node has passed from. If the latter has collected the data from the last T-node it has passed from, then it is cooperative, otherwise it is selfish. For example, in Fig. 7, U2 knows that U1 has last passed from T1, and U1 knows that U2 has last passed from T2. A U-node can also check the timestamp of the T-node data a U-node delivers, to recognize misbehaving U-nodes that try to communicate stale data. If the timestamp is greater than a specified interval (e.g., the estimated period of content update), the U-node may also treat such misbehaving U-nodes as selfish.

The further history of an encountered U-node's path can also be a basis for deducing the strategies of this node. However, this history can only be known approximately (unless the graph is a straight line, or is isomorphic to a straight line). An alternative way is for each U-node to log the ID and time of visit to each T-node. When getting into communication range, they can exchange these logs in order to deduce if the encountered node is cooperative or selfish. If the history of retrievals of another U-node can be discovered, then a U-node can also discern if a U-node has been partially cooperative (or selfish), i.e., if it retrieves the data from only a few of the T-nodes it passes.

C.3.11 Strategy of a U-node

We say that a U-node is cooperative at a certain T-node if it collects the data emitted, regardless of whether it is interested in them or not. Elementary actions a U-node can take at a certain T-node are: *C* (cooperative/collect data) and *S* (selfish/not collect data). A U-node's strategy consists of its behavior each of the times it visits a T-node. More specifically, if there are n T-nodes in the network, it is a set $\{s(T_1)_i, s(T_2)_i, \dots, s(T_n)_i\}_i$, $i = 1, 2, \dots$, where $s(T_j)_i$ is the strategy followed by this U-node the i th time it visits node T_j .

From the above description, the set of possible strategies is infinite. We should limit the set of simulated strategies to just a few. To this end, a U-node will be called cooperative(selfish) if it is cooperative(selfish) at all T-nodes it visits. Primary cases that should be simulated are: all cooperative, all selfish, and a subset of the U-nodes cooperative, the remainder being selfish. Additional cases that can be explored are those concerning partially selfish U-nodes (e.g., a U-node can be cooperative with a certain probability).

The last important parameter to be specified is how a cooperative U-node reacts to a U-node it perceives as selfish. Possible reactions are:

- When a U-node encounters a selfish U-node, it will not transmit any previously acquired data.
- When a U-node encounters a cooperative U-node, it will transmit all its previously acquired data that are requested from it.
- When a U-node encounters a cooperative U-node, it will transmit as many previously acquired information objects as it retrieves from the other node. This strategy is expected to better deal with partially cooperative nodes.
- In the tit-for-tat strategy, a U-node remembers the ID of a selfish U-node and at their next encounter also behaves selfishly.

C.4 Integration of the CCR scheme

For the integration of the Cooperation Content Retrieval the Simulation Platform had to be extended with new features like

- T-node simulation subsystem (described in section 2)
- Graph-based mobility

For easier integration a new simulation scenario was created offering easy parametrization of the above mentioned features. This scenario assumes a graph representation of the streets where U-nodes move around. T-nodes are located on the vertices of the graph and collect information about the states or variables of the environment. U-nodes move randomly along the edges of the graph, collecting information from T-nodes, or upon meeting another U-node.

The graph used for modeling the street network in this scheme is a directed graph, where a two-way road (or duplex edge) can be modeled by two parallel edges, connecting one vertex to the other and back. The parameters for this mobility model could be loaded from a standard XML file format which can be generated by hand or custom made automatized tools. The XML file describes the graph by listing the vertices with their coordinates and the edges between them.

The mobility model first read this XML file to build the internal representation of the graph which is a connectivity matrix of the vertices. For optimizing the running time a routing table is precalculated at the

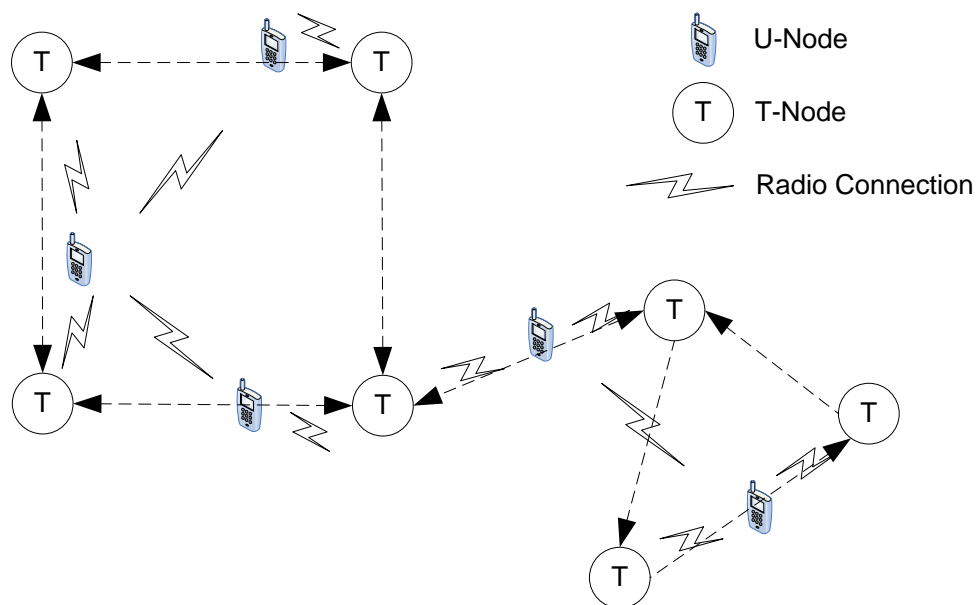


Figure 9: Graph mobility in the Simulation Platform

beginning (using the Floyd-Warshall algorithm) of the simulation for caching the shortest routes between vertices. U-Nodes choose randomly a target vertex of the graph then they calculate the shortest path to their destination.

It is also possible to simulate group movement of the U-Nodes. The model has a tunable parameter that describes the probability of joining or leaving a group. U-Nodes which are in the same group synchronize their departure from vertices in the network and move together to the next vertex effectively simulating public transportation. After reaching the target vertex, the modeled U-Node chooses another destination on the graph.

We also have a "static" graph mobility for those devices which stands in one place (T-Nodes). Those also read from the XML file the coordinates of the graph's vertices. Then it may choose randomly a vertex to stay in, or read from the XML file to know in which graph's node it is placed. This depends from the value of "random" parameter of the static graph mobility.

D BIONETS Simulation Platform Documentation

BIONETS Simulation Framework Documentation Guide An index of all BIONETS Simulator related documentation — <http://bionets.hit.bme.hu/documentation/html/index.html>

Getting Started With The BIONETS Simulator Framework A tutorial showing how install the necessary tools and set up the Simulator Framework — <http://bionets.hit.bme.hu/documentation/html/gettingstarted.html>

Simulation details of the Networking Framework Specification of the implementation of networking level functions in the simulator — <http://bionets.hit.bme.hu/documentation/html/networkingfw.html>

BIONETS Simulator Framework Reference Manual detailed reference documentation of the source code of the Simulator — <http://bionets.hit.bme.hu/documentation/html/api/index.html>

References

- [1] D. Miorandi, J. Huusko, F. D. Pellegrini, H. Pfeffer, D. Linner, C. Moiso, and D. Schreckling, "D1.1.3/3.1.3 serworks architecture v1.0," BIONETS Deliverable (D1.1.3/3.1.3), 2008.
- [2] V. Simon and S. Szabo (Editors), "Bionets simulation framework and initial performance analysis," Bionets Deliverable (D1.3.2), August 2007. [Online]. Available: http://www.bionets.eu/docs/BIONETS_{D1}_{3}_{2}.pdf
- [3] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *ACM SIGMOD International Conference on Management of Data*, 1984.
- [4] N. Beckmann, H.-N. Beckmann, H.-P. Kriegel, and B. S. R. Schneider, "The r*-tree: An efficient and robust access method for points and rectangles," in *SIGMOD Conference*, 1990.
- [5] M. Hajieleftheriou, "Spatial index library." [Online]. Available: <http://research.att.com/~marioh/spatialindex/>
- [6] L. Tassiulas and S. Toumpis, "Optimal deployment of large wireless sensor networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, July 2006.
- [7] R. Catanuto, S. Toumpis, and G. Morabito., "Opti{c,m}al: Optical/optimal routing in massively dense wireless networks," in *Proc. of IEEE INFOCOM*, Anchorage, USA, May 7–9, 2007.
- [8] E. Altman, P. Dini, D. Miorandi, and D. Schreckling (Editors), "D2.1.1 Paradigms and Foundations of BIONETS research," BIONETS (IST-2004-2.3.4 FP6-027748) Deliverable (D2.1.1), August 2007. [Online]. Available: http://www.bionets.eu/docs/BIONETS_{D2}_{1}_{1}.pdf
- [9] P. Bernhard, E. Altman, and A. Silva, "The mathematics of routing in massively dense ad-hoc networks," in *Proc. of AdHoc-NOW Conference, Sophia-Antipolis, France, Septembre 10-13 2008*.
- [10] A. Silva, P. Bernhard, and E. Altman, "Numerical solutions of continuum equilibria for routing in dense ad-hoc networks," in *Proc. of Valuetools 2008, Workshop Inter-Perf, Athens, Greece, October 20-24 2008*.
- [11] E. Altman, M. Debbah, A. Silva, and P. Bernhard, "Continuum equilibria for routing in dense ad-hoc networks," in *Proc of the 45th Annual Allerton Conference on Communication, Control and Computing, Urbana-Champaign, Illinois, September 2007*.
- [12] R. Meester and R. Roy, *Continuum Percolation*, ser. Cambridge Tracts in Mathematics. Cambridge University Press.
- [13] D. Miorandi, E. Altman, and G. Alfano, "The impact of channel randomness on coverage and connectivity of ad hoc and sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 3, pp. 1062 – 1072, March 2008.
- [14] b. Ryan and M. Debbah, "Random Vandermonde Matrices-Part I: Fundamental results," *ArXiv e-prints*, Feb. 2008.
- [15] Øyvind Ryan and M. Debbah, "Random vandermonde matrices-part ii: Applications," *CoRR*, vol. abs/0802.3572, 2008, informal publication. [Online]. Available: <http://dblp.uni-trier.de/db/journals/corr/corr0802.html\#\{abs-0802-3572>
- [16] N. Bonneau, M. Debbah, and E. Altman, "Spectral Efficiency of CDMA Downlink Cellular Networks with Matched Filter," *EURASIP Journal on Wireless Communications and Networking*, pp. 1–10, 2006.
- [17] S. Barbarossa, G. Scutari, and A. Swami, "Distributed detection and estimation in decentralized sensor networks: an overview," in *EURASIP EUSIPCO*, Florence, 3-7 September 2006.
- [18] A. Giridhar and P. R. Kumar, "Towards a Theory of In-Network Computation in Wireless Sensor Networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 98–107, April 2006.
- [19] M. Jelasity, A. Montessor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Trans. Comput. Syst.*, vol. 23, no. 3, pp. 219–252, 2005.
- [20] J. Aspnes and E. Ruppert, "An introduction to population protocols," *Bulletin of the European Association for Theoretical Computer Science*, vol. 93, pp. 98–117, Oct. 2007.
- [21] I. Carreras, F. De Pellegrini, D. Miorandi, A. Montessor, and A. Guerrieri, "Distributed estimation of global parameters in delay-tolerant networks," CREATE-NET, Tech. Rep. 200800022, 2008.
- [22] B. Bollobas, *Random Graphs*. Cambridge: Cambridge Univ. Press, 2001.

- [23] F. De Pellegrini, D. Miorandi, I. Carreras, and I. Chlamtac, "A graph based model for sparse ad hoc networks," in *Proc. of IEEE INFOCOM*, Anchorage, USA, May 7–9, 2007.
- [24] F. D. Pellegrini, D. Miorandi, I. Carreras, and I. Chlamtac, "Modeling connectivity over time in disconnected ad hoc networks," CREATE-NET, Tech. Rep. 200800020, 2008.
- [25] E. Altman and F. De Pellegrini, "Forward correction and fountain codes in delay tolerant networks," 2008. [Online]. Available: <http://www.citebase.org/abstract?id=oai:arXiv.org:0808.3747>
- [26] I. Carreras, D. Miorandi, and I. Chlamtac, "A framework for opportunistic forwarding in disconnected networks," in *Proc. of Mobiquitous*, July 17-21 2006.
- [27] S. Szabo and V. Simon (Editors), "D1.3.1 the initial mathematical models of new bionets network elements and algorithms," Bionets Deliverable (D1.3.1), February 2007. [Online]. Available: http://www.bionets.eu/docs/BIONETS_D1_3_1.pdf
- [28] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Trans. on Networking*, vol. 10, no. 4, pp. 477–486, Aug. 2002.
- [29] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. on Information Theory*, vol. 46, no. 2, pp. 388–404, March 2000.
- [30] M. Aezlادن, R. Cohen, and D. Raz, "Locally vs. globally optimized flow-based content distribution to mobile nodes," submitted to Infocom 2009.
- [31] E. Altman, T. Bacsar, and F. De Pellegrini, "Optimal monotone forwarding policies in delay tolerant mobile ad-hoc networks," in *Proc. of Inter-Perf2008*, Athens, Greece, Oct 24 2008.
- [32] E. Altman, G. Neglia, F. D. Pellegrini, and D. Miorandi, "Decentralized stochastic control of delay tolerant networks," INRIA, Tech. Rep. 6654, August 2008.
- [33] S. Alouf, I. Carreras, D. Miorandi, and G. Neglia, "Evolutionary epidemic routing," Research Rep. RR-6140, INRIA, 2007, <http://hal.inria.fr/inria-00130803>.
- [34] —, "Embedding evolution in epidemic-style forwarding," in *Proc. of IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS 2007)*, Pisa, Italy, October 2007.
- [35] E. Altman and Y. Hayel, "A Stochastic Evolutionary Game Approach to Energy Management in a Distributed Aloha Network," in *Proc. of INFOCOM 2008*, 2008.
- [36] —, "Stochastic evolutionary games," in *Proceedings of the 13th Symposium on Dynamic Games and Applications*, 30th June-3rd July 2008.
- [37] E. Altman, R. Elazouzi, Y. Hayel, and H. Tembine, "An evolutionary game approach for the design of congestion control protocols in wireless networks," in *Proc. of Physicomnet, Berlin, April 4, 2008*, 2008.
- [38] I. Koukoutsidis, E. Jaho, and I. Stavrakakis, "Cooperative content retrieval in nomadic sensor networks," *Computer Communications Workshops, IEEE INFOCOM 2008*, pp. 1–6, April 2008.