This project has received funding from "HORIZON 2020" the European Union's Framework Programme for research, technological development and demonstration under grant agreement no 645220

Road-, Air- and Water-based Future Internet Experimentation

Project Acronym:	RAWFIE		
Contract Number:	645220		
Starting date:	Jan 1st 2015	Ending date:	Dec 31^{2t}, 2018

Deliverable Number and Title	UNSURPASSED D1.3: Draft Software Extensions for Tasks 2,3		
Confidentiality	RAWFIE internal	Deliverable type[1]	R
Deliverable File	D13_UNSURPASSED.pdf	Date	31/3/2018
Approval Status[2]		Version	V1.0
Contact Person	Stavros Toumpis	Organization	AUEB
Phone	+30 210 8203551 +30 6974 740063	E-Mail	toumpis@aueb.gr

1 Deliverable type: P(Prototype), R (Report), O (Other)
2 Approval Status: WP leader, 1st Reviewer, 2nd Reviewer, Advisory Board

AUTHORS TABLE

Name	Company	E-Mail
Esmerald Aliaj	AUEB	aliai16@aueb.gr
Georgia Dimaki	AUEB	ginadimaki135@gmail.com
Nikos Fotiou	AUEB	fotiou@aueb.gr
Petros Getsopoulos	AUEB	pgetsopoulos@hotmail.com
Vasilios Siris	AUEB	vsiris@aueb.gr
Stavros Toumpis	AUEB	toumpis@aueb.gr

REVIEWERS TABLE

Name	Company	E-Mail

DISTRIBUTION

Name / Role	Company	Level of confidentiality[3]	Type of deliverable

CHANGE HISTORY

Version	Date	Reason for Change	Pages/Sections Affected
1.0	31/3/2018	Original Submission	

[3] Deliverable Distribution: PU (Public, can be distributed to everyone), CO (Confidential, for use by consortium members only), RE (Restricted, available to a group specified by the Project Advisory Board).

Abstract: In this deliverable:

- We present information on the preliminary (draft) software extensions for Tasks 2 and 3.
- We also present some recent developments on software extensions for Task 1.
- We present the current status of the project, recent activities not covered by the previous deliverables, and our plans for the immediate future (months 6-8).

Keywords: **EDL, Experiments, UNSURPASSED**

Table of Contents

List of Figures

List of Tables

1. Status of Software Extensions for Task 1 (Ad Hoc Routing)

We note that draft software extensions for Task 1 have also been reported in Deliverable D1.1. Here, we present recent progress, and this part of the deliverable should be read in conjunction with the related part of deliverable D1.1.

Measuring key metrics under various conditions and networking protocols (BABEL, BATMAN, OLSR, BMX7) has been continued. Some work has been done in diversifying and standardizing the metrics we monitor, with a view of automating and simplifying our work when we move to the RAWFIE platform. These metrics include

- the maximum transmission rate of transmitters,
- the total number of routes from each node to each other node,
- the received and transmitted data in bytes per packet per second,
- the total received and transmitted data in bytes per second per interface,
- and the total received and transmitted data in bytes per second per open connection.

We also record statistics for the wireless channel depending on their availability from the hardware in use, as well as more link quality metrics as calculated by the lower layers of communication and the underlying routing protocol.

Experiments are first being performed using the Core Network Emulator (for easier reproducibility and testing of different configurations and network topologies) before replicating them in the testbed. We have tested and plan to test a number of different routing protocols with widely different characteristics. Our experiments focus on monitoring the convergence speed, tolerance to disruptions, and routing overhead. We are also investigating the behavior of routing protocols when it comes to announcing the same or overlapping host networks multiple times, specifically focusing on the resulting any-cast behavior of, on the one hand, the routing protocols that allow announcements of overlapping networks and, on the other hand, protocols like BMX that do not allow this behavior.

Lastly, we plan to work on using cryptographic operations by leveraging libraries such as mbed TLS or PolarSSL, for ensuring that node identity is within a white list and trusted by the rest of the network. Also, all protocols that we have used and tested up to now are proactive; we plan to experiment with reactive ones in subsequent steps of our research if reliable implementations are found.

In the rest of this section, we briefly present some specific comments regarding our experimentation with some of the routing protocols.

BMX7

The BatMan-eXperimental Mesh Daemon is a layer 3 routing protocol that aims at reducing network update messages, aggregating information for link and path changes that occur during the daemons’ lifetime and broadcasting them afterward. For its operation it uses IPv6 addresses exclusively. Similarly to other routing protocols, it announces IP and network information to each node. However, contrary to all other ad hoc routing protocols, duplicate or overlapping announcements of networks are not allowed.

We have tested and partly integrated the BMX v7 protocol to our existing infrastructure. Basic multi-hop experiments have been performed and its reliability has been evaluated.

Specifically in our testing with the BMX protocol, another important direction of our research is the use of the Linux kernel models supporting tunnel-based networking. Tunnel-based routing can be used, as an example, for offering a gateway to an external connection, or allowing advertisement updates from different routing protocols.

BATMAN (layer 2)

The layer 2 version of the BATMAN routing protocol has been integrated to the latest Linux Kernel, and basic functionality has been tested. BATMAN is a unique case that presents interest due to its being designed for operating on the second OSI layer. It should present a number of advantages over other layer 3 operating protocols, yet its ability to work transparently with the rest of our software stack as well as its performance on large mobile networks should be investigated before integrating it to Dedalus.

OLSR

The OLSR routing protocol has been also installed mainly for comparing it with its newer improved iterations like BABEL or BATMAN.

2. Status of Software Extensions for Task 2 (DTN Routing)

In the last two months we have integrated an implementation related to the second task on our framework, namely Delay- and Disruption-Tolerant Networking. In particular, we have installed and tested the well-known IBR-DTN standard for multi-hop and data muling scenarios. Therefore, we can have the individual nodes operate autonomously using only the DTN protocol as defined in RFC5050 for routing, or have it operate in cooperation with another ad hoc routing protocol to help it discover new neighbors. This gives us the ability to seamlessly support both delay-sensitive and delay-tolerant communication between disconnected networks, data muling operations, data streaming, and file transfer in the unsafe and unstable maritime environment.

We are able to use the bundle protocol as defined in RFC 5050, as well as the bundle security protocol as described in RFC 6257. Moreover we can limit bundle propagation based on age or hop count.

Currently we have experimented with various routing algorithms available in the IBR-DTN framework, including:

- Routing with static connections
- Epidemic routing
- Flooding routing
- PRoPHET Routing

The implementation supports a number of different storage schemes:

- In-memory storage (RAM)

- Persistent storage in the file system
- Database Storage

Using the above, we have conducted simple multihop routing experiments in the Raspberry Pi testbed.

A few issues are still being investigated and require work before they can be ironed out and integrated in the RAWFIE platform; these include the time synchronization between nodes, fragmentation occurring to the network control messages sent by the DTN protocol, interaction with the ad hoc routing protocol, and selecting the settings of the underlying TCP or UDP connection.

In future work, we aim at supporting the following applications:

- Network performance monitoring for DTN-enabled networks
- Connection keep-alive and polling for DTN-enabled networks
- Path tracing for DTN enabled networks
- File transfer functionality
- Automatic monitoring and synchronization of files (DTN-enabled distributed filesystem)
- Audio and Video streaming by leveraging the ad hoc layer for route discovery and the DTN layer for alleviating issues arising from channel quality fluctuations

Other frameworks investigated and under consideration for future implementation and use are:

- ION (Interplanetary Overlay Network): This DTN architecture was described in RFC 4838 and is designed with space communications in mind, and the ability to work on embedded devices makes it a viable choice for our network.
- DTN2: A less mature but promising implementation of a DTN routing protocol.

3. Status of Software Extensions for Task 3 (ICN Routing)

The third layer to be integrated in our framework is Information Centric Networking. We have built, installed and tested CCN-Lite in fully-connected networks and multi-hop scenarios successfully over the last few weeks. Furthermore, ICN routing based on CCN-Lite has been tested and works equally well over ad hoc networks as it does over traditional networks.

The protocol gives us the advantage of reducing the network hops and eliminating redundant requests for specific named content, as it handles the caching on every node and delivers the content from the nearest cache. This function is extremely important in order to consume as few resources as possible from the already diminished resources available due to the nature of the environment.

Thus far, we have conducted experiments using up to 5 nodes in a fully-connected network, with one node asking for content from the other four, emulating a multihop network by restricting the set of nodes that each node can reach.

The next issue to tackle will be the integration with the DTN protocol of the second layer, so that CCN-Lite will be able to work indifferently of various delays or lack of connectivity to other nodes. This requires modifying the CCN-Lite packets to support the format of the DTN protocol and to communicate using it instead of the default configuration.

Other issues that need to be investigated further and will be solved soon include the support of updating cached content on nodes after a file has changed (thus avoiding the delivery of stale data) and the automatic creation of Faces and Forwarding rules based on the neighbors the node has from the ad hoc layer.

After solving the aforementioned issues, we are interested in testing more ICN frameworks. The most notable of these is the Named Data Network (NDN) Project, a large project with participants from universities around the world that may provide us with tools for better management and monitoring of the network.

The overall target is to incorporate the ICN protocols in an as transparent as possible way with the rest of the layers in order to support file transferring and caching abilities with support for named content lookup among the nodes.

4. Visit to Skaramagas Site

On March 27th part of our team visited the Skaramagas site in order to (a) discuss some logistics of our experiments as well as (b) observe preliminary experiments conducted by the University of Glasgow team. Our team comprised of Esmerald Aliaj, Georgia Dimaki, and Stavros Toumpis (for part of the day).

We were updated on the status of the available platforms, and investigated the options we have for placing our hardware on them. The Flexus platform, in particular, has a number of advantages:

1. It is heavy enough so that our hardware can be accommodated without compromising the stability and steering of the platform.
2. It is large enough so that our antennas can have ample clearance from the water.
3. Our hardware can easily be attached on a number of points on the structure of the platform.
4. Each Flexus USV can offer a WiFi access point to nearby objects, notably our hardware (according to the manufacturers of the platform, connectivity to the shore is offered through the use of the BATMAN protocol).

None of these advantages is shared by the other platform (Niriis) of which there are 10 USVs available. However, in principle we will be able to use any combination of the available platforms.

We also observed the experiments conducted by the Glasgow team and we believe that the experience will be useful for the design of our experiments as well. In particular, the use of an independent router and Kafka server can significantly accelerate the debugging process, and we plan to adopt this option for our experiments as well.

5. Preliminary Experiments

The next two months will be important to the project, as we will use them to set up and execute our first preliminary experiments at the Skaramagas site. As specified in our proposal, the experimental results will be reported in the next Deliverable, titled “D1.4: 2nd Progress Report”, due at the end of Month M8 (31/5/2018).

The aim of these preliminary experiments is not to produce final results, but, rather, to reveal logistical problems, software bugs, and other related issues, as well as make sure that all the actors (our team, the operators of the USVs, and the rest of the consortium) are on the same page in terms of the responsibilities and capabilities of each party.

Taking these aims into account, the experiments will have the following features:

- A small number of only 3 boats will be deployed so that the manpower of the USV operators remains at a manageable level.
- There will be 4 experiments, one for each task.
- The software used in each experiment will be independent of the other experiments to the greatest possible extend. This means that, e.g., the experiment in delay-tolerant routing will not be making use of ad hoc routing, in contrast to scheduled final experiments
- Our hardware will come with its own power supply, so that the mounting of the devices will be easier (based on preliminary discussions, it appears that our power supplies can last longer than the power supply of the USVs).
- Our experiments will make use of the RAWFIE Kafka server, however we will bring our own router and Kafka server, on which the software and hardware will be already tested, as a fallback scenario.
- There will be only one type of traffic, CBR, with a relatively modest data rate.
- There will be no experiments where the nodes will be influencing the mobility of the nodes.

As the experiments will be simpler, the algorithm we will implement will be called SIMPLE_NETALGORITHM, to differentiate it from the NETALGORITHM algorithm described in full in the previous deliverable:

SIMPLE_NETALGORITHM (START, END, DEST, PROT)

The arguments are as follows:

- START: the time when the node will start creating traffic
- END: the time at which the node will stop creating traffic
- DEST: the destination of the traffic. It is set to 0 if the node creates no traffic and only used as a relay or destination of traffic.
- PROT: the protocol used, as follows:
  - ADHOC: A single ad hoc protocol will be implemented, tentatively BABEL (which is the protocol for which we have the most experience in our local testbed)
  - DTN: The IBR-DTN protocol will be implemented (with one of the flooding/epidemic/PROPHET options, this will be decided)
  - ICN: The CCN-Lite protocol will be implemented.
  - SEC: The Identity-Based encryption protocol will be implemented.

Two important issues to be decided in the immediate future are:

- The platform that will be used. Tentatively, we will use the Flexus platforms, due to its advantages mentioned above, but this will be finalized in our biweekly phone meetings.
- The connectivity of our hardware with the WiFi network of the testbed. There are two options: either the equipment connects through the USV platform, or directly with the on-shore router. It is much preferable if the connectivity is through the platform, as in this manner there will be fewer connectivity problems on the *control channel*, and it will be easier for the nodes to spread out and form a truly multihop network over the *experiment channel.*

Experiment 0.1 Ad Hoc Routing

In this experiment, three nodes will be placed such that a two-hop ad hoc network will be formed. An important issue is the quality of the channel, and in particular what is the distance between nodes such that two hops are indeed necessary. We will also make sure that the three USVs are not placed on a line, so that the link between the first and the third is not obscured by the one in the middle, but rather weakened due to the distance between the two nodes.

Table 1: Experiment 0.1

```
Node
        ID Flexus1   % This is the source of the traffic stream
                Route[
                        WP<0, +146.08, -47.00, +0.0>                    ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 300 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                 Algorithm SIMPLE_NETALGORITHM(
                                 0,              % Start  of traffic
                                 300,            % End of traffic
                                 Flexus2,        % Destination of  traffic
                                 ADHOC           % Ad hoc routing
                ]
~Node

Node
        ID Flexus2    % This is the destination of the traffic stream
                Route[
                        WP<0, +163.92, -2.41, +0.0>
                ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 300 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                 Algorithm SIMPLE_NETALGORITHM(
                                 0,              % Start  of traffic
                                 300,            % End of traffic
                                 0,              % There is no destination
                                 ADHOC           % Ad hoc routing
                ]
```

```
~Node

Node
        ID Flexus3    % This is the relay
                Route[
                        WP<0, +207.50, -8.35, +0.0>
                ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 300 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                 Algorithm SIMPLE_NETALGORITHM(
                                 0,                % Start  of traffic
                                 300,              % End of traffic
                                 0,                 % There is no destination
                                 ADHOC            % Ad hoc routing
                ]
~Node
```

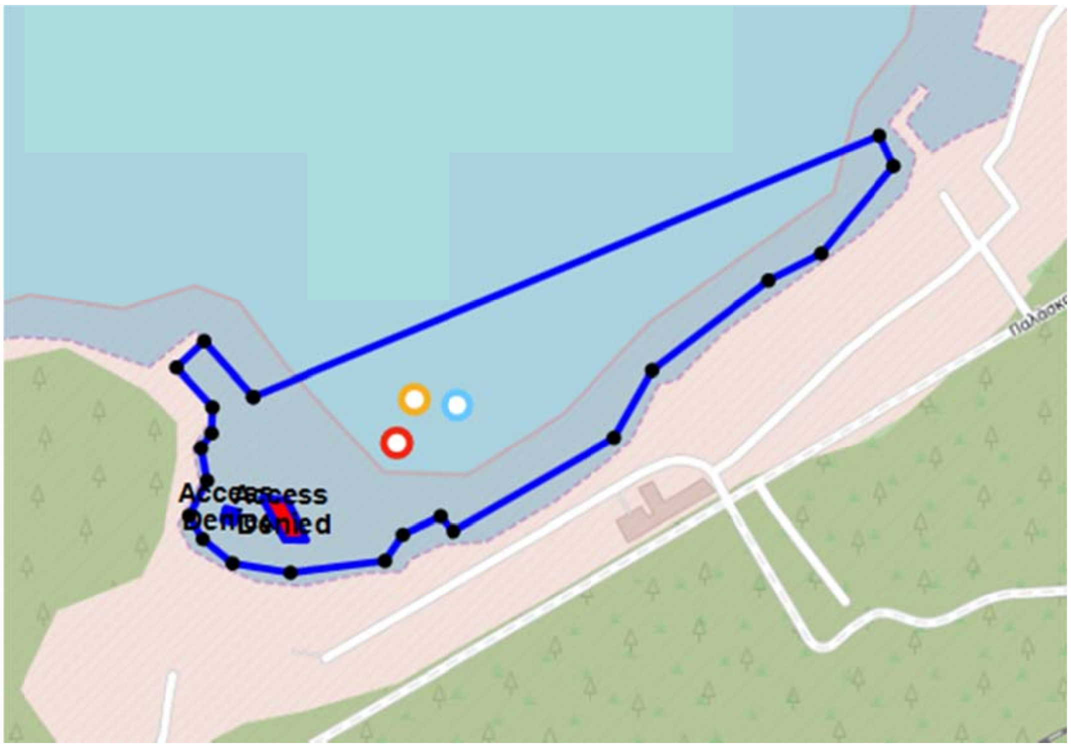

Figure 1: Topology of Experiment 0.1

Experiment 0.2 Delay-Tolerant Routing

In this experiment, there will be two stationary nodes placed at some distance to each other, and a third node will be moving between the two, acting as a data mule. One of the stationary nodes will be creating traffic for the other one.

Table 2: Experiment 0.2

```
Node
        ID Flexus1   % This is the source of the traffic stream
                Route[  WP<0, +100.36, -73.72, +0.0>  ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 100 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                        Time 0 Algorithm SIMPLE_NETALGORITHM(
                                  0,                  % Start  of traffic
                                  300,                % End of traffic
                                  Flexus2,            % Destination of  traffic
                                  DTN                 % Delay-tolerant routing
                ]
~Node
Node
        ID Flexus2   % This is the destination of the traffic stream
                Route[    WP<0, +361.20, +63.43, +0.0>      ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 100 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                  Algorithm SIMPLE_NETALGORITHM(
                                  0,                  % Start  of traffic
                                  300,                % End of traffic
                                  0,                  % No destination
                                  DTN                 % Delay-tolerant routing
                ]
~Node
Node
        ID Flexus3    % This is the data mule
                Route[
                        WP<0, +123.88, -60.34, +0.0>   % mule will be traveling back and for th
                                                        %between the two WPs
                        WP<1, +333.63, +48.57, +0.0>                    ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 100 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                 Algorithm SIMPLE_NETALGORITHM(
                                  0,                  % Start  of traffic
                                  0,                  % End of traffic
                                  0,                  % No destination
                                  DTN                 % Delay-tolerant routing

                ]
~Node
```

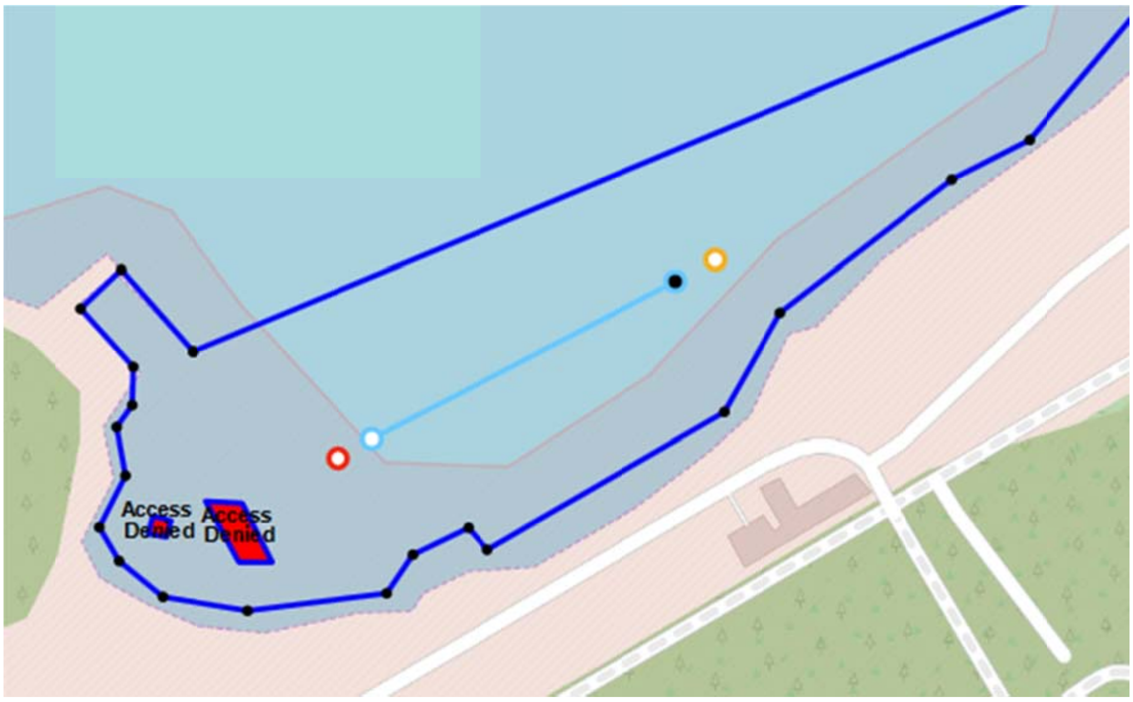

Figure 2: Topology of Experiment 0.2

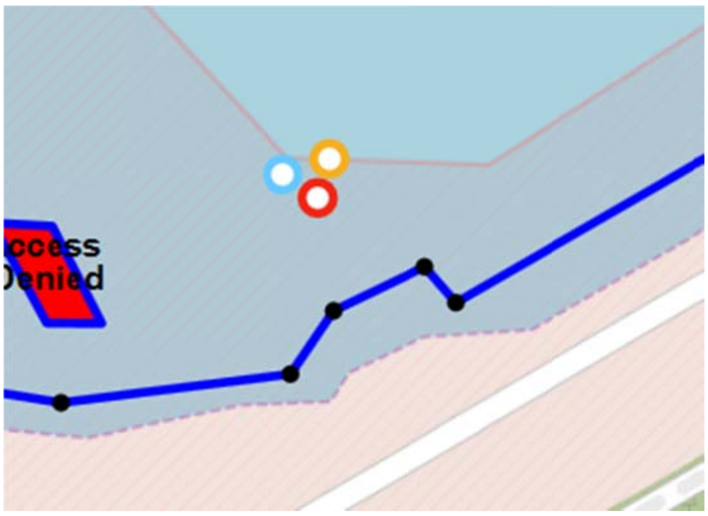

Figure 3: Topology of Experiments 0.3 and 0.4

Experiment 0.3 Information Centric Routing

In this experiment, there will be three stationary nodes close to each other. Two of them will be requesting pieces of content located in the other one, simultaneously for part of the experiment.

Table 3: Experiment 0.3

```
Node
        ID Flexus1   % This is the source of the first traffic stream
                Route[  WP<0, +145.42, -92.92, +0.0>                    ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 300 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                Algorithm SIMPLE_NETALGORITHM(
                                0,                  % Start  of traffic
                                200,                % End of traffic
                                Flexus2,            % Destination of  requesting traffic
                                DTN                 % Delay-tolerant routing
                ]
~Node
Node
        ID Flexus2    % This is the the source of the second traffic stream
                Route[  WP<0, +150.33, -76.56, +0.0>                    ]
                Sensor[
                        Time 000     Name NETSENSOR set Activated
                        Time 300 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                Algorithm SIMPLE_NETALGORITHM(
                                100,                  % Start  of traffic
                                300,                % End of traffic
                                Flexus2,            % Destination of  requesting traffic
                                DTN                 % Delay-tolerant routing
                ]
~Node
Node
        ID Flexus3    % This is the destination of the two streams
                Route[   WP<0, +130.69, -83.11, +0.0>                   ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 300 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                Algorithm SIMPLE_NETALGORITHM(
                                0,                  % Start  of traffic
                                0,                 % End of traffic
                                0,                  % No requesting traffic will be created
                                DTN                 % Delay-tolerant routing

                ]
~Node
```

Experiment 0.4 Content encryption

There will be three stationary nodes close to each other. Two of them will be requesting pieces of content located in the other one, to be transmitted securely, again simultaneously for part of the experiment.

Table 4: Experiment 0.4

```
Node
        ID Flexus1   % This is the source of the first traffic stream
                Route[WP<0, +145.42, -92.92, +0.0>          ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 300 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                  Algorithm SIMPLE_NETALGORITHM(
                                  0,                    % Start  of traffic
                                  200,                  % End of traffic
                                  Flexus3,              % Destination of  requesting traffic
                                  SEC                   % Secure traffic
                ]
~Node
Node
        ID Flexus2    % This is the the source of the second traffic stream
                Route[WP<0, +150.33, -76.56, +0.0> ]        ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 300 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                  Algorithm SIMPLE_NETALGORITHM(
                                  100,                    % Start  of traffic
                                  300,                    % End of traffic
                                  Flexus3,                % Destination of  requesting traffic
                                  SEC                     % Secure traffic
                ]
~Node
Node
        ID Flexus3    % This is the destination of the two streams
                Route[WP<0, +130.69, -83.11, +0.0>          ]
                Sensor[
                        Time 0     Name NETSENSOR set Activated
                        Time 300 Name NETSENSOR set Dectivated
                ]
                DataManagement[
                                  Algorithm SIMPLE_NETALGORITHM(
                                  0,                    % Start  of traffic
                                  300,                  % End of traffic
                                  0,                    % No requesting traffic will be created
                                  SEC                   % Secure traffic
                ]
~Node
```