Seventh FRAMEWORK PROGRAMME
FP7-ICT-2007-2 - ICT-2007-1.6
New Paradigms and Experimental Facilities

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Deliverable D4.1
"Experimental scenarios including evaluation criteria and methodology"

Project description
Project acronym: **ECODE** Project full title: **Experimental Cognitive Distributed Engine** Grant Agreement no.: **223936**
Document Properties
Number: FP7-ICT-2007-2-1.6-223936-D4.1 Title: Experimental scenarios including evaluation criteria and methodology Responsible: Bart Puype (IBBT) Editor(s): Bart Puype (IBBT) Contributor(s): Chadi Bakarat (INRIA), Pedro Casas (CNRS), Benoit Donnet (UCL), Mickaël Hoerdt (UCL), Amir Krifa (INRIA), Yann Labit (CNRS), Steven Latré (IBBT), Guy Leduc (ULg), Johan Mazel (CNRS), Stijn Melis (IBBT), Philippe Owezarski (CNRS), Dimitri Papadimitriou (ALB), Bart Puype (IBBT), Damien Saucez (UCL), Wim Van de Meerssche (IBBT), Bruno Willemaers (ULg) Dissemination level: Public (Pu) **Date of preparation: June 2nd, 2010 (v1.0), December 13th, 2010 (v1.1)**

D4.1 – Experimental scenarios including evaluation criteria and methodology

Executive Summary

This document is the final version of ECODE **deliverable D4.1**, which describes **experimental scenarios including evaluation criteria and methodology**. The experimental scenarios proposed here continue from the use cases as defined in D3.1. Building on initial assessment and proof-of-concept experimentation performed within WP3, WP4 takes into account real-world evaluation criteria such as stability and scalability. Attention to scientific validity ensures the relevance of experimental outcomes.

WP4 comprises the second of two experimental phases spanning from M18 (Feb.2010) to M31 (Mar.2011) as outlined by Technical Annex I, part B. The first experimental phase (WP3) examined the applicability, validity and feasibility of a machine learning engine for several use cases. During the second phase however, experimentation is taken beyond the initial assessment and validation of machine learning combined with advanced networking techniques which were performed in WP3. For this, the experimentation converges around the common Machine Learning Engine (MLE) platform detailed in D2.2 “Cognitive Engine – Experimental low-level design”.

First phase experimentation consisted of a number of use cases distributed over three parallel technical objectives. Its results were included in D3.3, D3.5 and D3.7 (related to TO1, TO2 and TO3 respectively). Whereas WP3 tasks were defined through these technical objectives, WP4 tasks are defined by experimentation goals. Nevertheless, WP4 builds further upon these use cases.

This deliverable D4.1, as the first WP4 deliverable, describes the experimental scenarios for the second experimental phase. This identifies the scenarios considered within WP4 (some based on WP3 scenarios, some new). Also, it will allow answering questions regarding (1) Technical Objectives, (2) Content, (3) Description and (4) Scientific Validity of each of the experimental scenarios. The template used to obtain this information is available in Annex 1. Each experiment description is structured around the following elements:

- Detailed use case description, performance objectives, (technical and non-technical) constraints, and description of the expected results;
- Description of the experimental evaluation criteria and metrics;
- Description of the experimental scenario description, setup, tools, platform and methodology;
- Scientific validity in terms of verifiability, reliability as well as repeatability and reproducibility.

The second experimental phase too will be conducted in physical experimental infrastructures. For most experimental scenarios, this is the iLab.t physical experimental facility, located at IBBT premises in Ghent, Belgium.

List of authors

Affiliation	Author
ALB	Dimitri Papadimitriou
CNRS	Pedro Casas
CNRS	Yann Labit
CNRS	Johan Mazel
CNRS	Philippe Owezarski
IBBT	Steven Latré
IBBT	Stijn Melis
IBBT	Bart Puype
IBBT	Wim Van de Meerssche
INRIA	Chadi Bakarat
INRIA	Amir Krifa
UCL	Benoit Donnet
UCL	Mickaël Hoerdt
UCL	Damien Saucez
ULg	Guy Leduc
ULg	Bruno Willemaers

List of figures and/or list of tables

Table of Contents

1 Introduction

The focus of the ECODE experimental research project is to introduce a new Internet architectural component realized by means of a cognitive system and that preserves the original Internet design principles, including the end-to-end principle and its transparency, to sustain growth of an Internet that remains in line with what it supposed to deliver to the end-user and that performs in accordance to what it is expected to deliver to the end-user. As the purpose of this new architectural component is to sustain growth of an Internet that performs in accordance to what it is supposed to deliver to the end-user and performs according to these expectations in order to satisfy the end-users, large-scale testing and validation is explicitly in the scope of this research project. Combined experimentation will allow determining whether composing the Internet high-level goals - societal, economical, etc. - can be translated into lower-level objectives (in terms of functionality and performance) and constraints (both technical and non-technical) and enforced via the newly introduced machine learning component as part of the Internet routing system.

This second experimental phase will be conducted in physical experimental infrastructures. For most experimentation, the machine learning engine will be experimented and its performance evaluated by means of a dedicated and fully controlled emulation platform: the iLab.t experimental facility, located at IBBT premises.

1.1 Relation to other ECODE deliverables

This deliverable describes the experimental scenarios themselves, including methodology, evaluation criteria, platform, tools, input data etc., and this on a network (or scenario) level. Regardless of exact system architecture and tools used, the experimental scenarios included here all use the common XORP machine learning engine platform, which is detailed in **Deliverable D2.2 "Cognitive Engine - Experimental low-level design"**. Meaning, D2.2 provides a system level description of the experimentation, stating the interfaces between machine learning engine (MLE), routing engine (RE), forwarding engine (FE), management plane (MP) and translation and communication (TCI) component.

As WP3 and WP4 entail the first and second experimental phase respectively, this deliverable serves a function similar to the one of **Deliverable D3.1 "Experimental Plans and Scenarios"**. Some of the experimental scenarios are a continuation of those already included in D3.1. WP3 defined three technical objectives (TO). Evaluation of those TO1, TO2, TO3 was included in deliverables D3.3, D3.5 and D3.7 respectively. As WP4 considers a common machine learning engine platform, an integrated approach is followed for the technical objectives, while WP4 tasks are defined towards goals. Comparison of WP3 and WP4 technical objective use cases is provided in a further section. Where necessary, the experimental scenario description will refer back to relevant D3.1 descriptions.

1.2 Scientific validity

In WP4, experimentation goes beyond initial proof-of-concept of machine learning as an advanced networking technique. Implementation and evaluation of actual online machine learning is envisioned. The targeted evaluation criteria will allow assessment of stability and scalability of these techniques in realistic networking scenarios.

In order to accept experimentation as a viable track in determining performance and evaluation metrics, the experimental scenario and methodology should be **scientifically valid**. Scientific validity includes verifiability, reliability, repeatability and reproducibility.

Verifiability: an experiment is verifiable if the outcomes can be verified against a formal model, meaning they match models that describe the outcome as a function of the experiment input parameter. In the case of functional analysis, experiment flow and outcome match a prescribed list of actions and/or output.

Reliability: reliability means the experiment and outcome are valid for a certain time run. As a minimum requirement, this means that the components of the experiment remain functional (i.e., do not crash or break down) during this time period. Furthermore, results and outcomes are reliable if they remain consistent during that time period (within a certain well-defined range).

Repeatability: the term repeatability is used when repeating the experiment within the same experimental scenario, i.e., same platform, experimental facility, testbed, input parameters, etc. The experiment is repeatable when different runs of the experiment (repetitions) yield the same outcome and results. Correct experimental methodology and usage of models, algorithms and output data processing is required in order to guarantee repeatability.

Reproducibility: an experiment is reproducible when it can be reproduced within a similar, but different experimental setup. This can mean different platform, facility. Typically reproducibility comes into play when a third party performs the same experiment in order to verify scientific validity of the outcome and results of the experimental scenario.

Verifiability, reliability, repeatability and reproducibility of an experimental scenario depend heavily on outcome values, which are in some cases measured only within a certain range, and not exactly. Actual experiments –including emulation experiments- often include some form of non-determinism. The experimentation scenarios may validate only under certain constraints (this is especially the case for repeatability and reproducibility). Where necessary, this is mentioned in the scenario descriptions.

1.3 Experimental scenarios

Within WP3, three technical objectives were defined, each consisting of several use cases which cover different problems in representative areas, identified as Internet architectural and design challenges (such as security, controllability, routing, and accountability).

Table 1. List of WP3 use cases

TO	Case	Name
TO1 (D3.3)	Case a1	Adaptive traffic (packet/flow) sampling
	Case a2	Path performance monitoring
	Case a3	Distributed anomaly/intrusion detection
TO2 (D3.5)	Case b1	Informed path selection
	Case b2	Network-driven recovery and resiliency
	Case b3	Profile-based accountability
TO3 (D3.7)	Case c	Inter-domain routing/BGP

For WP4, these following use cases are defined within each of the technical objectives (as per the Technical Annex):

Table 2. List of WP4 use cases

TO	Case	Name
TO1	Case a1	Adaptive traffic sampling and management
	Case a2	Path performance monitoring
	Case a3	Intrusion and attack / anomaly detection
TO2	Case b1	Path availability
	Case b2	Network recovery and resiliency
	Case b3	Profile-based accountability
TO3	Case c	Routing system scalability
		Routing system quality

The remainder of this deliverable describes experimental scenarios each fitting within one of these use cases, clustered into four groups:

- *Monitoring and security*: corresponds to use cases a1, a2, and a3, i.e., the development of an autonomous system for network monitoring, traffic management, and anomalies detection.
- *Routing and recovery:* corresponds to use cases b1 and b2, i.e., the development of a solution for ranked path selection and fast network recovery.
- *Accountability*: corresponds to use case b3, i.e., the development of a solution for correlating profiles with subscribers' usage and their impact on the network resources.
- *Routing system:* this corresponds to use case c, the development of a solution for speeding up BGP path exploration, and determining quality (correctness) of the solution.

Each experimental scenario description includes an experiment overview and objectives. A detailed listing of (technical and non-technical) constraints, expected results, evaluation criteria and metrics, setup, tools, platform and methodology follows. Finally, scientific validity and any constraints thereupon are stated.

The template used to obtain this information is available in Annex 1.

2 Monitoring and security

2.1 Adaptive traffic sampling and management

2.1.1 Running monitoring applications based on adaptive sampling

Experimental scenario description	
Title	Running monitoring applications based on an adaptive sampling approach
Technical objective	The scenario builds further on the use case concerning adaptive traffic sampling (TO1). Machine learning is used to optimally configure the sampling rate in monitors so as to reach the best measurement accuracy at limited overhead. The information on flows will then be used to manage them appropriately inside routers.
Participants	INRIA, IBBT

Content	
Short description	We plan to deploy our architecture components on top of IBBT iLab.t test bed. Namely our Traffic Monitoring and Sampling Service and our Data Collection and Analysis service. The later is supposed to be a centralized component that (i) collects NetFlow reports sent by the monitoring and sampling service installed in each monitor of the selected topology and (ii) runs our ML algorithms which performs a given monitoring solution and updates accordingly the monitor's configuration at runtime. Then, using a range of topologies (GEANT and Abilene like topologies), we want to evaluate different monitoring applications.
Expected result(s)	Our system is intended to provide an estimation of network and traffic status. This estimation is afterwards used to find a better configuration of monitors and controllers that reduces measurements errors and improve the management of flows inside routers. For instance, our platform can optimize the monitoring to carry out the following measurements: • The greediest users • The number of packets per flow

Experimentation	
Evaluation criteria and metrics	The following evaluation criteria will be used: • The network status estimation accuracy • The traffic collection overhead • In addition, the convergence time of the estimation and the computation time are also two important parameters that will be used as metric

Experimental scenario: description, tools, configuration and running conditions	**System structure and platform:** We will use IBBT iLab.t test bed running Linux. We will install within the different routers our traffic monitoring and sampling service. The later captures real traffic promiscuously using the Pcap library. Then, using its sampling module, it decides either to consider the captured packet or not for updating the list of maintained flows. Once some conditions (flows' timers, number of constructed flows, etc.) are satisfied our traffic monitoring services will send NetFlow reports towards our data collection and analysis service. We will need to install the later service in a separated central node. **Network topology:** We need to setup either GEANT or Abilene like topologies including 20 routers in average in addition to one central node which will run our data collection and analysis service (including the ML algorithm). **Experimental configuration:** Once the topology is available and our services are deployed, we will need to configure locally our monitoring and sampling service and to point them to the central collector node.
Methodology	**Methodology to obtain experimental data:** The experimentation methodology involves: - Setting up an experimental topology (on the iLab.t test bed). - Deploying our traffic sampling and monitoring service within all nodes considered as monitors. - Deploying our data collection and analysis service on a central separated node. - Locally configuring the different services. - Applying any traffic. We can either use a common traffic with other partners or run ours using either iLab.t test bed traffic generation tools (if existing) or some third party tools like D-ITG (the Distributed Internet Traffic Generator, http://www.grid.unina.it/software/ITG/). - Extracting experimental data. Three main methodology tracks will be followed: - Proof-of-concept: functional analysis of the NetFlow reports collection from the different monitors, the execution of our adaptive sampling algorithm within our central service, and finally the re-configuration of the deployed monitoring points. The planned public demonstration will make use of this track. - Performance analysis: by running batches of experiments for various configuration parameters, and experimental topologies, we will extract performance metrics such as the network status estimation accuracy and the traffic collection overhead.

	- Scalability analysis: for this track we concentrate on topology size and the number of interfaces to be monitored and we look at the resulting traffic collection overhead and the time taken by our ML algorithm to react and adjust monitors' configuration.

Scientific validation	
Verifiability	One of the applications that we want to verify consists on estimating the traffic generated by each Autonomous System (AS). Once done, we can verify whether the weight we have already associated to each AS is reflected in terms of the amount of the traffic it generates. We can also verify that the overhead resulting from the experimentation we run does not exceed the target overhead value that we have already fixed.
Reliability	1) Indicate key issues that may impact reliability. Which external factors may invalidate the experimental results? 2) The collector node should be powerful enough (CPU, Memory …) in order to cope with the big amount of the traffic it collects from the different monitors and process. If some delay is introduced - when processing the collected traffic and extracting results – the decisions that the collector would take in order to adapt the sampling rates within the different monitors will be out of date.
Repeatability and reproducibility	Unless the background traffic changes, we can reproduce the same experimentation on the iLab.t platform.

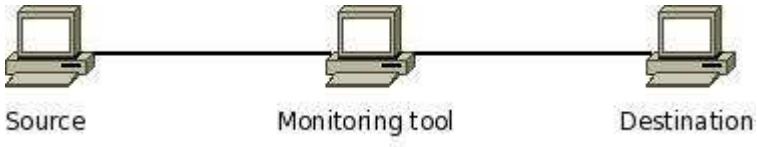
2.2 Path performance monitoring

2.2.1 Validation of the performance and accuracy of the monitoring system

Use case scenario description	
Title	Validation of the performance and accuracy of the monitoring system
Technical objective	Technical Objective 1. This objective relates to the improvement of the manageability and diagnosability of the Internet.
Participant(s)	This scenario will be performed by LAAS/CNRS for the passive monitoring system.

Content	
Short description	We want to setup an emulation network, generate traffic and verify that our monitoring system is reliable, i.e., it captures and reports every packet correctly (no missing packet, no bit error, accurate timestamp).
Expected results	The expected outcome of this setup is to build a reliable passive monitoring system (i.e. no packet loss, no bit error, accurate timestamp) that provides accurate and fast software-based monitoring and measurement capability.

Experimentation	
Evaluation criteria and metrics	The two main criteria to enforce are: packet loss (i.e. packets that are not captured) and accurate timestamps for each captured packet. The two metrics we will use are: the percentage of captured packets over the total number of packets and the error on values of timestamps.

Experimental scenario: models, platform, configuration, traffic, constraints	**Machine Learning Engine:** The passive monitoring system aims at capturing packets on a link and providing it online to the cognitive engine. **System structure and platform:** For validating the passive monitoring system, we need, for providing a ground truth, a DAG system. Because of this requirement in terms of a trustable hardware transparent traffic capturing system, the evaluation is run on the LaasNetExp experimental platform (Ilab does not provide such hardware for capturing traffic). The DAG system has been previously validated during the last decade and proved to be very fast in capturing packets (and then avoiding capture loss when the hosting machine is well provisioned) with a very accurate GPS based timestamping mechanisms. CNRS owns several of these DAG systems on its LaasNetExp experimental platform. They will then be used to evaluate the performance of accuracy of the passive software monitoring system specifically designed and developed for ECODE. **Network topology:** Practically speaking, our experimental scenario consists of testing the monitoring tool on one host (to assess the reliability of the capture and timestamping mechanisms). For the validation, the following topology is be used.  **Experimental configuration and input description:** We need to generate realistic traffic. For this purpose, we will either replay previously captured traces or generate traffic based on a realistic traffic model (as the Gamma-Farima one that was developed in the framework of the French MetroSec project by ENS Lyon and LAAS-CNRS). The Operating system running on host is a Linux distribution. The Tool to replay traces is either TCP Replay, or a tool specifically developed by ourselves: nonglrd_gen. **Constraints:** We need a ground truth when capturing traffic to compare with the one capture by our software monitoring tool. This constraint is solved by the use of DAG systems available at LAAS.

Methodology	**Methodology to obtain experimental data:** Validating a monitoring system is quite impossible without another trustable monitoring system. The principle of this validation then relies on the use of a DAG system for validating our software global monitoring and measurement system. DAG system will provide the ground truth for assessing our software monitoring tool. For evaluating the monitoring entity we need to use a DAG system on the same link, close from the monitoring entity to be evaluated. Both tools capture the traffic. We then compare the two traffic trace files, the DAG one being the ground truth, and then evaluate our software monitoring tool performance level. **Experimental data processing methodology and analysis:** We develop all necessary tools for analyzing them. These tools have to compare the traces captured by a software monitoring entity and the ones captured by DAG systems. It will consist in checking that all packets are reported, and compute difference between related timestamps.

Scientific validation	
Verifiability	DAG systems provide a ground truth about the traffic which has been captured. The DAG trace is the reference which our software based captures have to be compared against.
Reliability	The DAG system has been proved to be very accurate and reliable. It guaranties that our validation will be accurate and reliable.
Repeatability and reproducibility	As we are replaying traces or generating our own traffic based on a model, we can repeat the experiments as many times as needed.

2.3 Intrusion and attack / anomaly detection

2.3.1 Evaluation of the Anomaly Detection System (ADS)

Use case scenario description	
Title	Evaluation of the Anomaly Detection System (ADS)
Technical objective	Technical Objective 1. This objective relates to the development of an anomaly detection system to improve Internet security. By the proposed technique, we expect to improve the performance in terms of false negative and false positive rates
Participant(s)	This scenario will be performed by LAAS/CNRS with a possible collaboration with ULANC if distributed anomaly detection is considered.

Content	
Short description	The objective is to evaluate the two proposed contribution families to the improvements of the Anomaly Detection System (ADS). The latter uses ML techniques in order to improve the false alarm ratio traditionally encountered when using ADS in the actual Internet. Thanks to ML, it is also aimed at detecting 0-day anomalies. Here, the objective is thus twofold: 1. Evaluate if the local Machine Learning (ML) techniques aiming at better characterizing the traffic provide more accuracy and efficiency in decision making; 2. Evaluate whether the system is able to detect anomalies it does not know and is able to increase its knowledge database
Expected results	We expect to find lower false alarm ratios thanks to ML techniques which are used for continuously learning about the traffic characteristics. We also expect to generate new detection rule as a reaction do the detection of previously unknown anomalies.

Experimentation	
Evaluation criteria and metrics	The performance of the ADS will be evaluated on the false negative and false positive rates. This is valid for both known and 0-day anomalies. The false negative rate is the ratio between the number of undetected attacks over the total number of attacks. The false positive rate is the ratio between the number of false alarms over the total number of attacks.
Experimental scenario: models, platform, configuration, traffic, constraints	**Machine Learning engine:** We develop a system including several algorithms using ML techniques (in particular clustering techniques) for detecting anomalies (previously known or even unknown – 0day anomalies) on a single link. These system and algorithms are describes in deliverables D3.2 and D3.3. **System structure and platform:**

	We will use several machines connected to a network on which the access point is monitored by our ADS. These machines will generate anomalies, e.g. flash crowd, DDoS attacks, of various kinds, shapes and intensities. The ADS responses will then be logged and its detection performance evaluated as presented in the following. We will use the LaasNetExp platform for small topologies (and for setting up the methodology), and the Ilab cluster facility for larger ones. **Network topology:** For evaluating the ADS, several network topologies with different numbers of machines and different interconnection topology could be necessary to emulate different aggregation schemas. We will start from 2 machines generating traffic, and go up to maybe 10 sources or more if possible. **Experimental configuration and input description:** Link characteristics on the topology will be the ones classically observed on actual networks, except that we will not introduce any loss as they are already represented in the replayed traces when a packet is lacking. The delay distribution can be something very simple as a classical exponential function or any sub-exponential functions generally observed in the Internet path delays as Weibull or Pareto distributions. It is important in such a scenario to be able to evaluate the ADS based on documented traces, i.e. traces for which we know at what moment, and for what duration there are anomalies in the traffic, what kind of anomaly, their characteristics (as intensity for example, number of sources or destination, source and destination addresses, source and destination ports, etc.). We will use the MetroSec dataset for this purpose which is a set of traces containing documented anomalies of any kinds. The MAWI dataset could be used as well (despite it is not fully documented – anyway, it contains real anomalies while the ones of the MetroSec dataset are kind of artificial). We also need a tool for injecting replayed traces in the emulation network or use existing solutions as tcpreplay. **Constraints:** First, we need traces containing documented anomalies. The MetroSec dataset provides a very reliable ground truth. MAWI could also be used because the anomalies are not artificial. Anyway, the anomaly documentation work can be prone to errors. Second, a measurement and monitoring system is the basis for providing inputs to the machine learning algorithms used in the local ADS.
Methodology	**Methodology to obtain experimental data:**

	The ADS has to be evaluated with normal traffic without anomalies, and with anomalous traffic. The anomalies in this traffic can be legitimate as flash crowds or alpha flows, but also illegitimate as DoS attacks, scanning, failures, misconfigurations, etc. Among the DoS attacks or scanning strategies, it would be requested to experiment as many different kinds of attacks as possible (all kinds of flooding, smurf, etc.). We will use several machines connected to a network on which the access point is monitored by our ADS. These machines will generate anomalies, e.g. flash crowd, DDoS attacks, of various kinds, shapes and intensities. The ADS responses will then be logged and its detection performance evaluated as presented right over. The evaluation relies on calculating the false negative and false positive rate based the ADS alarms raised when analyzing the traffic of replayed documented anomalous traces. The experimental data produced by the ADS under evaluation are the log files of alarms raised by the local and global ADS. **Experimental data processing methodology and analysis:** The obtained log files will be compared with the list, time, duration and characteristics of anomalies contained in the replayed documented traces. The number of false alarms and undetected attack is then trivial to compute.

Scientific validation	
Verifiability	Verifiability is enforced by the use of traces containing documented anomalies. This documentation represents the ground truth for this evaluation.
Reliability	Given the ground truth provided by the documented traces, the evaluation process is completely reliable.
Repeatability and reproducibility	As we are replaying traffic traces, we can repeat the experiments as many times as needed.

3 Routing and recovery

3.1 Path availability

3.1.1 IDIPS

Experimental scenario description	
Title	IDIPS Use Case
Technical objective	The scenario builds further on the IDIPS use case. Machine learning is used to limit the number of measurements. The machine learning technique takes the history of measurements to predict the future performance of the paths. IDIPS is a framework to determine the best among several paths, for any definition of best.
Participant	UCL

Content	
Short description	We implemented ISP-driven informed path selection (IDIPS) and the least mean squares (LMS) filtering for path performance prediction on the integrated XORP platform. Simulations have shown that the LMS filtering is adequate for delay prediction. We plan to test it in real time with concurrent instances to see if LMS can be used in a real system with plenty of simultaneous computations. In other words, we want to be sure that the system is able to scale.
Expected result(s)	A first result we expect is the confirmation that LMS predictions are correct (i.e., the prediction error is limited). We also expect to see a reduction of the number of measurements to the different paths. Finally, we plan to see how scalable the implementation is. At a first glance it should be very scalable, but the dynamic of the path performance could stress the implementation and we have to see an upper bound for the system, regarding the load and the dynamic.

Experimentation	
Evaluation criteria and metrics	The following evaluation criteria will be used: - Scalability - Prediction accuracy - Stability - Measurement load/measurements gain Scalability is examined both in terms of the machine learning engine (LMS) and the IDIPS framework. The scalability of the machine learning engine consists of observing the amount of resources consumed by the LMS models for keeping the prediction models accurate. The resource consumption is measured both in terms of memory and time. One LMS model has to be kept for each path to predict performance for, the amount of memory could vary with the dynamic of the path, each model keeping track of the last observations. The

	time consumption comes from the necessity to compute the model parameter for each prediction by analyzing the last observations. The scalability of the IDIPS framework, if we are not considering the machine learning part, is essentially driven by the time complexity. Indeed, for each request, IDIPS has to compute all possible paths, then query the machine learning part to get the last predictions for finally ranking and sorting the paths according to the rank that has been computed for them. The spatial complexity for a given metric is given by O(p + r*p_r) where p is the total number of paths maintained by the system, r, the number of request and p_r, the maximum number of paths per request. Prediction accuracy is related to LMS. The prediction accuracy represents the quality of the prediction, i.e., the error made by the prediction with respect to the use of the observed value. Prediction is used because the cost of observing (i.e., measuring) is too important. Stability is examined for path ranking. Along the time, the performance of the path changes and requires re-computation of the prediction model which may influence the final ranking result. The stability aims at determining how ranking changes along the time. A high stability means that ranking can be cached, reducing so the time complexity (at the cost of a space complexity) and thus increasing the request rate IDIPS can handle. On the contrary, instability means that ranking re-computation has to be performed all the time, reducing so the request rate IDIPS can handle. Measurement load/measurements gain is somewhat related to stability. If a path is stable with respect to a metric, the metric can be measured less often while keeping accuracy in the predictions. On the contrary, an unstable path will require more frequent observations to keep a good fitting between the prediction and the reality. The prediction part thus helps in reducing the number of measurements as it adapts the measurement rate to only measure the path when needed. Without a prediction system, the path has to be measured every time a ranking is requested for it. The two main concerns are the stability and the scalability, which are somewhat related. In our simulation so far, each path was considered independently of the others. In the experimentation, we will perform tests with several paths to rank in parallel.

Experimental scenario: description, tools, configuration and running conditions	Machine Learning engine: The LMS filtering algorithm will be used for the experimentation. Earlier studies have shown that both LMS and ARMA-like techniques were working correctly for the delay, but it has been determined that ARMA-like technique were to costly to be able to use them in the IDIPS use case (it would not scale). System structure and platform: We will use IBBT iLab.t test bed running XORP 1.7 on top of Linux 64 bits. The testbed will be formed of two independent IDIPS instances with up to 50 clients. The experiment will only focus on the delay metric and the path availability (if a path is not available it will be presented as non reachable to the clients). The first step will consist of measuring paths. Their performance is controlled by the path performance controller facility from iLab.t. Once validated, we will do the experiment by measuring path through the real Internet via ADSL, cable, and fiber connectivity. The purpose of having two IDIPS instances is to see if the computed rankings are the same at each instance. The difference could be caused by interferences between the two IDIPS measurements parts. Some failures will be generated manually to determine how fast the system can detect them. Constraints: A major constraint for the experimentation is that the experimentation lab must be isolated of any other experiment to be sure that no interference exists between the different experiments.
Methodology	Methodology to obtain experimental data: The experimentation methodology involves: - Setting up an experimental topology (on the iLab.t test bed). - Deploying the XORP platform on iLab.t. - Configuring the IDIPS modules and the clients. - Applying any client request behavior. - Setting path performance alteration generator. - Extracting experimental data. Two main methodology tracks will be followed: - Scalability analysis; several parameters influence the scalability: the number of ranking requests to process in parallel influence the state IDIPS has to keep, the number of concurrent path that have to be ranked influence the amount of data to be stored in IDIPS but also the network load (the number of measurements to perform). - Accuracy analysis; IDIPS has to predict the delay for paths and real paths have performance changing all the time which may affect the prediction. Measuring and predicting on the wild will provide a good prediction accuracy analysis. Experimental data processing methodology and analysis: Log facilities are put in the IDIPS implementation to get all the parameters and predictions.

Scientific validation	
Verifiability	In parallel to the measurements and predictions performed by IDIPS, the path delays can be monitored with other tools. The ranking can then be performed mathematically based on the observation from the monitoring tool. The scalability can be estimated theoretically once we know the dynamic of the path. We can then compare theoretical results with the observations performed on the testbed.
Reliability	The major reliability issue will come from the XORP scheduling which does not ensure that the actions requiring time information will be perfectly accurate. IDIPS must be able to terminate ranking instances and path measurements if they have not been used for a long time. Without this, the state would increase forever and consume all the memory. The evaluation will show the best timers to put on this.
Repeatability and reproducibility	For the fully generated delay, the repeatability and the reproducibility of the experiment are trivial as all the parameters of the testbed are controlled. For measurements involving Internet paths, it is not evident that the reproducibility will be easily achievable. However, we will take care of analyzing the external parameters to see if the experiment is representative or if they correspond to a transient phase.

3.1.2 Internet Coordinate System

Experimental scenario description	
Title	Internet Coordinate System experimentation
Technical objective	Internet Coordinate Systems (ICS) are promising techniques to predict unknown network distances (typically delays) between any pair of nodes from a limited number of measurements. They are distributed algorithms, actually distributed Machine Learning Engines (MLEs), inferring together the missing elements in an almost empty distance matrix, where the few elements present in the matrix are the only distances actually measured.
Participant(s)	ULg

Content	
Short description	We have already implemented the classical Vivaldi ICS within XORP as a reference. We are now implementing two other modules: - Our own ICS, designed in ECODE and denoted DMF (for Decentralized Matrix Factorization), which is not based on a Euclidian embedding and can therefore capture asymmetric distances and triangular inequality violations (TIV); - A Triangular Inequality Violation (TIV) detector, which allows finding routing shortcuts based on the ICS. They will be tested in a realistic setting, to see how well they perform and compare to the simulated results.
Expected result(s)	We expect a validation of our implementations with similar behaviors and results as the simulated ones. We also expect that DMF will outperform Vivaldi, especially on topologies with asymmetric distances and TIVs.

Experimentation	
Evaluation criteria and metrics	Experimental evaluation criteria: – Distance prediction accuracy of the ICS – Gain of discovered routing shortcuts Distance prediction accuracy is the main evaluation criterion for an ICS. We want to predict distances (delays) within the network, and the more accurate the predictions are, the better. Our reference distance matrix will be obtained by active measurement over all pairs of nodes on the testbed. The internal information issued by the ICS algorithm about estimated relative errors on the coordinates only gives a first insight, which we can compare with the real errors. In this context, we can use the same metrics as during simulation, with $d_{i,j}$ and $\hat{d}_{i,j}$ being respectively the distance from node i to j and its estimation:

	-*Cumulative Distribution of Relative Estimation Error (REE)*[1], with $REE = \frac{\left\|\hat{d}_{i,j} - d_{i,j}\right\|}{d_{i,j}}$. -*Stress-1*[2]: $\sqrt{\frac{\sum_{i,j}\left(d_{i,j} - \hat{d}_{i,j}\right)^2}{\sum_{i,j} d_{i,j}^2}}$. -*Median Absolute Estimation Error:* $median_{i,j}\left(\left\|d_{i,j} - \hat{d}_{i,j}\right\|\right)$. Gain of routing shortcut is related to the TIV detection and inference of some better paths than the direct ones currently used. For detected TIVs, it will give direct insight of how well it performs. It is computed as the ratio between the best shortcut we have detected, and the best path that actually exists.
Experimental scenario: models, platform, configuration, traffic, constraints	**Machine learning engine:** Two ICS algorithms will be used for experimentation. First, we will use the Vivaldi algorithm, which is already implemented within XORP and see how it performs, and compares to other real use cases of Vivaldi. Indeed, Vivaldi was proposed some years ago and benefits from several actual implementations, for example within Azureus – a bittorent client – or Pyxida running across PlanetLab. Second, once the implementation for the DMF algorithm is ported to XORP, we will use it to perform a similar experiment, with reference and feedback from the first experimentation. Eventually, we will use TIVs and shortcuts detection on both coordinate systems. **System structure and platform:** We will use the IBBT iLab.t test bed with 64-bit Linux machines running XORP 1.7(SVN) instances. We would use as many nodes as possible on this testbed. Each one would have the implementation of our ICS module, activated on multiple network interfaces. During performance of the experimentation, each ICS module would log its coordinates and data needed to perform its update, associated with a timestamp for later comparison and analysis. After validation on the IBBT iLab.t testbed, we could conduct similar experiments over PlanetLab.

[1] Notice that it might be interesting to compare them with the errors estimated by the algorithms.

[2] Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika,29*, 1-27.

	Traffic: We will experiment the ICS under various traffic loads to evaluate the impact of the traffic on our performance metrics.
Methodology	**Methodology to obtain experimental data:** First, we will set up an experimental network topology, with XORP running on the hosts. We will activate ICS nodes successively, giving the address of the first one started as a bootstrap server. We will test various topologies, including some with asymmetric delays and TIVs, by engineering link delays and link weights on the testbed. We will let each node record within a local file the successive coordinates and error the node is computing for the experiment duration. **Experimental data processing methodology and analysis:** Experimental data collected will be processed to obtain a delay matrix which will be compared to the one measured over the experimental network topology. We will then compare both matrices wrt our performance metrics.

Scientific validation	
Verifiability	Since exhaustive data will be recorded by each Vivaldi module at each iteration, we will be able to verify the computation of coordinate updates for each node, at each step. Coordinates observed can be compared to the delays explicitly measured during experiment. Also, we can compare experimental results with results obtained by simulation with the delay matrix from the experiment.
Reliability	Experimental results might be biased if we halt our experiment soon after some failure within the network, considering that we are using final data to our analysis. In such case, the coordinate system would be in a state where some coordinates are not yet updated and, therefore, delay predictions are not accurate. To obtain reliable results, we must ensure that such a failure has not happened soon before the recording of our final data.
Repeatability and reproducibility	All the configuration parameters of the testbed (topology, delays, etc.) will be available. The boot order of the ICS nodes will be given too. While we can give our *modus operandi* to repeat our experiment, operations from individual nodes, such as picking peers at random for coordinates updates, are not reproducible. We can expect to reach similar results, but not identical from one experiment to another.

3.2 Network recovery and resiliency

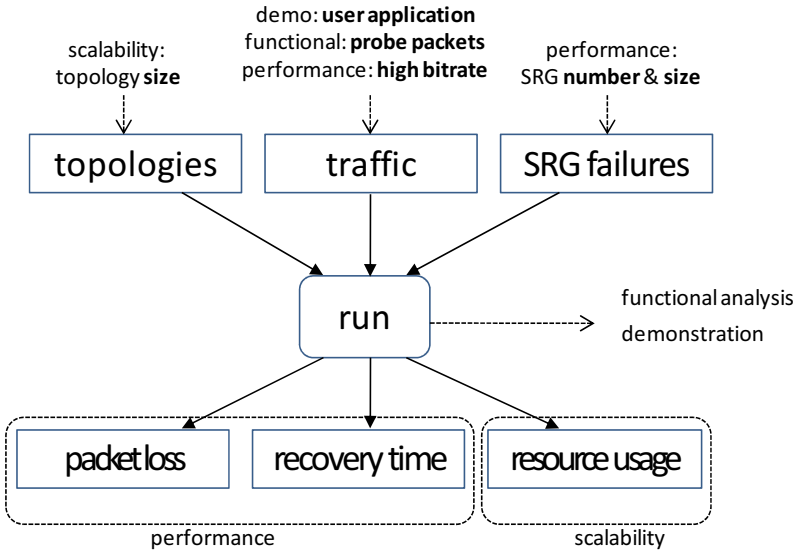
3.2.1 OSPF SRG inference

Experimental scenario description	
Title	OSPF SRG inference
Technical objective	The scenario builds further on the use case concerning data mining of OSPF updates to identify SRLGs (TO2). Machine learning is used to infer Shared Risk Groups (SRG) from correlated historic OSPF protocol messages. These SRGs will be communicated to the OSPF protocol in order to reduce failure recovery times.
Participants	IBBT, ALB

Content	
Short description	We plan to implement the algorithms developed for OSPF event modeling and clustering on the integrated XORP platform. Using this implementation on a range of topologies, we want to evaluate the modified OSPF process for failure recovery using inferred SRGs.
Expected result(s)	We expect that failures of links that are part of a SRG will be detected significantly faster using the machine learning engine, as OSPF does not have a mechanism to represent SRGs. Furthermore, we expect the routing system to be more scalable, since the SRG mechanism allows grouping, and thus reducing the number of OSPF routing recalculations and updates.

Experimentation	
Evaluation criteria and metrics	The following evaluation criteria will be used: - Failure detection time - Recovery time / convergence time - User traffic packet loss - Inference accuracy - Stability - Scalability Failure detection time is the time between occurrence of a SRG failure and the time said SRG is detected (reported) as down in a node running the machine learning engine with OSPF inference capability. Recovery time is the time between occurrence of the failure and re-establishment of end-to-end connectivity. User traffic affected by the failure is lost during recovery time. Alternatively, convergence time is the time between failure occurrence and the last routing update related to the failure; the network may or may not be recovered after convergence. Inference accuracy concerns how accurate the SRG inference (prediction) is. OSPF inference will predict links as failing; these predictions may or may not match protocol

	messages (and link state) receive shortly after the failure. Stability is to be examined both in terms of the machine learning engine itself (SRG inference accuracy should not deteriorate over time), as well as the routing system (addition of SRG inference should not destabilize routing). Our main concern with scalability for this experimentation scenario is processing time, and how it relates to topology size. We expect an OSPF processing time scaling better with number of nodes, compared to the OSPF case without SRG inference functionality. Processing time of routing updates and shortest path tree calculation after failure can be expressed as simply the number of routing recalculations (since the actual processing time is hard to measure).
Experimental scenario: description, tools, configuration and running conditions	**Machine Learning engine:** Two algorithms developed earlier for OSPF event modeling and data-mining will be used in the scenario: - SRG state-space transition probabilities and LSA event clustering - SRG prediction from link failure correlation **System structure and platform:** We will use IBBT iLab.t test bed running XORP on top of Linux. For some experiment tracks some end-user applications (e.g. video streaming) will be used in order to visually demonstrate recovery times. The OSPF SRG inference functionality can be demonstrated running on just one node in the network (receiving protocol messages from regular OSPF nodes), or running on some/all nodes of the network. Also, multiple nodes can rely on a single machine learning engine instance (which in term can be populated based on information from multiple nodes). Functionality is required in order to emulate the failure of links and SRGs (multiple concurrent failures). **Network topology:** As the experiments require recovery after failure, (at least) bi-connected topologies are needed. Topologies of 4-5 nodes are an absolute minimum in order to demonstrate basic recovery for very small SRGs (e.g., 2 links). Scalability analysis requires larger topologies. These may be implemented by running the XORP platform on the desired number of nodes of a large topology (containing the maximum number of nodes needed for the scalability experiments). **Experimental configuration and input description:** Recovery using SRG inference can be demonstrated without actual user traffic. For experiments evaluating traffic loss, constant bit rate traffic generators can be used. For a possible (public) demonstration, we will use actual traffic from e.g. a video-stream. This allows both measurement (of traffic loss) as well as visual verification of recovery times. **Constraints:** The handling of SRGs is done by failing multiple links at the same time. No effort will be taken to emulate actual SRGs by,

	for example, constructing OSPF links as connections that share a common point of failure in some layer 2 connection-oriented network. OSPF interfaces can be configured in slightly different ways each yielding more or less the same results in terms of end-to-end connectivity. However, the configuration method affects how LSAs are reported. Similarly, OSPF protocol messages do not report failures; earlier work on OSPF event modeling and data-mining assumed link failure messages. This will be coped with using the translation/representation functionality of the machine learning engine.
Methodology	**Methodology to obtain experimental data:** The experimentation methodology involves: - Setting up an experimental topology (on the iLab.t test bed). - Deploying the XORP platform and SRG inference machine learning instances. - Configuring the OSPF modules for the topology (tools will be used to do this automatically). - Applying any user traffic. - Generating failures according to (pre-determined) SRGs. - Extracting experimental data.  Three main methodology tracks will be followed: - Proof-of-concept; i.e. functional analysis of SRG inference, OSPF protocol message collection and SRG prediction based OSPF rerouting. The planned public demonstration will make use of this track. - Performance analysis; by running batches of experiments for various configuration parameters, SRG sets and experimental topologies, we will extract end-to-end performance metrics such as packet loss and recovery times. - Scalability analysis; for this track we concentrate on topology size mainly (choosing a limited set of configurations from the previous tracks), in order to examine network system wide stability and scalability (evaluation criteria as explained above)

	Some state changes within the OSPF process are exposed through OSPF messages. Additional monitoring points will be used inside the OSPF module in order to log how OSPF state (LSA database and pruned links) is updated when MLE SRG inferences are pushed to the OSPF module. Monitoring points may also be added in order to achieve more accurate timing of OSPF recovery actions, as the corresponding OSPF messages may be delayed for routing protocol stability reasons. **Experimental data processing methodology and analysis:** Experimental data will be gathered by examining: - OSPF protocol messages and machine learning engine information exchange; - OSPF routing recalculation occurrences and outcome. The above can be compared with unmodified OSPF; running the same experimentation scenario on a vanilla OSPF area. - Monitoring end-to-end packet loss Link-based recovery time can be inferred from link local OSPF messages (i.e., hello protocol). Generated datasets contain the above information. All these data will heavily depend on OSPF parameters and topology size and structure. It is important to note that the size and number of SRGs will also influence results. Since multiple dimensions can be identified in SRG formation, this will require particular attention. Datasets that contain traces of OSPF messages, failure messages, end-to-end packet loss etc. should be tagged with: - used topology; - set of SRGs used; - OSPF parameters; - failure occurrence parameters (e.g., IAT/HT for exponentially distributed failures); - type of traffic used in determining packet loss;

Scientific validation	
Verifiability	Recovery times can be verified using OSPF message exchange analysis. Gains seen in processing time (i.e., number or routing recalculations) can be derived from size and number of SRG, and topology size.
Reliability	1) Indicate key issues that may impact reliability. Which external factors may invalidate the experimental results? 2) As the experimental scenario aims to improve resilience, the OSPF modifications and machine learning engine processes should remain stable on a time scale which supersedes MTBF of the network links (i.e., months/years). Nevertheless, given the distributed nature of OSPF, the modified OSPF and machine learning engine should be able to cope with a node in the OSPF area rebooting; this should not impact performance of SRG inference in nodes that remain up.
Repeatability and	Some of the timing results will depend heavily on OSPF timing parameters. As such, these results will be a statistical

reproducibility	function of these timing parameters. Expected results should be stated within a range. Reproducibility relies on proper implementation of the interaction with the OSPF process; the default OSPF process (as present in XORP) has been modified to incorporate SRG inference predictions. Also, since SRGs are emulated by failing several links at a time, we should take care to construct realistic SRGs. When reproducing the experiment on an actual L3-over-L2 network scenario, with SRGs stemming from L2 routing and common points-of-failure, this poses certain limitations on the nature (number, size, span) of actual SRGs.

4 Accountability

4.1 Profile-based accountability

Experimental scenario description	
Title	Profile based accountability
Technical objective	The scenario is based on the Profile Based Accountability use case (b3), initially developed in WP3. The (machine-learning based) algorithmic options are the identification of action profiles and the computation of the subscriber profile in a profile learning and profile prediction stage, will be continued and extended.
Participant(s)	IBBT

Content	
Short description	We plan to: 1) Extend the existing clustering algorithm set with new clustering algorithms and implement a deviation function to detect a subscriber going out of profile. 2) Apply the “out of profile” information obtained from the algorithms in the previous step and previous work of case b3 (as reported in Deliverable D3.5) to the routing and forwarding plane.
Expected result(s)	When we are able to accurately classify out of profile behavior from subscriber traffic, we can detect which subscriber traffic is not behaving in a responsible way. By adapting their scheduling and congestion control/AQM parameters, we expect that resources allocation would enable a better fairness between subscribers’ traffic flows (depending on the local resource usage rate). More specifically, we expect that the changes made to the router’s scheduling algorithm will allow only penalizing (a set of) traffic flows that are behaving "badly" (e.g. inappropriately responsive or unresponsive to congestion notifications).

Experimentation	
Evaluation criteria and metrics	The following evaluation criteria will be used: - Clustering accuracy - Fairness - Reaction time - Efficiency Clustering accuracy: as during this phase also new clustering and classification algorithms will be investigated, the accuracy of those newly developed algorithms will be evaluated. Fairness: In this use case, fairness means that subscriber’s traffic (as result of TCP stack parameterization) which is not behaving according to its profile receives the resources it requested and only the subscribers’ traffic which is out

	of profile is penalized. A metric for this fairness will be defined and used as evaluation criteria. Reaction time: When a subscriber's traffic goes out of profile, he should be penalized as quickly as possible in order to statistically maintain a fair allocation of resources for the other subscribers' traffic. While occasional spikes of "behaving badly" may be allowed, we define the term going "out of profile" as behaving in such a way that a reaction is required. This reaction time will be evaluated and consists of a detection delay and reaction delay. Efficiency: The goal of the profile based accountability case is to penalize out-of-profile subscribers' traffic in favor of in-profile subscribers' traffic. The expectation is that this in-profile subscribers' traffic should be better serviced than out-of-profile traffic without unduly reduced resource sharing efficiency and link resource utility.
Experimental scenario: models, platform, configuration, traffic, constraints	**Machine learning engine:** Algorithms that were used during WP3 such as the C4.5 Decision Tree Classification algorithm will be used as well but we also plan to investigate new ones, both for the classification problem as for the deviation calculation. This allows effectively characterizing the performance of the earlier investigated algorithms through comparison. **System structure and platform:** We will use the IBBT iLab.t Virtual Wall test bed facility running XORP or the Click Modular Router on top of Linux. The experiments will be coupled with the traffic generation tool that was designed and reported on in deliverable D3.5. This traffic generation tool also foresees in the use of multiple software programs: - Apache web server - Sirannon video server, designed by the IBBT - VLC media player - etc. The profile based accountability algorithms can run on one machine. However, multiple physical machines are needed to emulate the behavior of different TCP stacks accurately. Furthermore, the algorithms can also be deployed an multiple nodes, in which each algorithm will work independently of the others. Running this scenario on multiple nodes is also part of this scenario. **Network topology:** As this scenario does not heavily rely on the routing functionality but focuses more on the scheduling functionality, there are no real requirements on the employed topology. However, for simplicity reasons, we investigate a butterfly-based topology where a set of servers are connected to a set of clients over one link. In order to have realistic data a large amount of nodes (20+) are needed to emulate this behavior. Initially, the number of servers will be fixed to 1, and the focus will be on the emulation of the clients. As such, a defective TCP stack will be emulated on the client

	side. **Experimental configuration and input description:** The evaluation and demonstration of the profile based accountability functionality requires the emulation of realistic subscriber traffic stacks. This emulation is handled by employing multiple TCP implementations that are provided by default in Linux. Typical examples include: - TCP Westwood/Westwood/+ - TCP Tahoe/Reno/New Reno - TCP BIC/CUBIC - Etc. Each of these TCP stacks can feature in-profile or out-of-profile by enabling the ECN support. If the use of ECN bits is ignored, then, although ECN is applied at the communicating side and intermediate routers, the stack can be regarded as defective (or unresponsive). In the proposed scenario, a set of TCP stacks (on the client side) will communicate with one specific TCP stack on the server side. The various TCP stacks on the client side can be turned defective, either at the start or the experiment or during the experiment. On the shared link between clients and servers, the PBA algorithm will be deployed. The goal of this algorithm is to detect the defective behavior of the TCP stacks while it lasts, and react accordingly.
Methodology	**Methodology to obtain experimental data:** The methodology can be highly automated thanks to central configuration of the traffic generation tool. This involves: - Defining the network topology and link configurations - Defining the scenarios to be investigated. The core scenarios described above (Experimental configuration and input description) are used and implemented with a varying number of subscribers. - Introducing traffic through the emulation of subscriber behavior. This includes subscribers going “out of profile” deliberately. - Extracting experimental data **Experimental data processing methodology and analysis:** This experimental data will be used to train the machine learning algorithms at which an adapted version of the process will be executed. The process that will be followed is: - Defining the network topology and link configurations - Defining the scenarios to be investigated. The core scenarios described above (Experimental configuration and input description) are used and implemented with a varying number of subscribers. These scenarios may or may not differ from the originally employed scenarios. - Introducing traffic through the emulation of subscriber behavior. This includes subscribers going “out of profile” deliberately. - Configuration of the machine learning algorithms and configuration of the routing and forwarding planes of the different routers. The profile based accountability functionality will be able to steer the scheduling functionality in the forwarding plane.

	- Extracting the evaluation metrics as described in "Evaluation Criteria and Metrics". This will be largely automated through the use of the traffic generation tool, which is also responsible for steering the experimental setup.

Scientific validation	
Verifiability	Visualization of the experiment will be foreseen in order to verify that the semantics of the experiment correspond with the configuration of the experiment. The behavior of subscribers can be verified by checking the models used to implement this behavior. If necessary, real-time behavior can be used as input to obtain more realistic models.
Reliability	The profile based accountability functionality relies heavily on assumption made of how individual subscribers behave in practice. The use of realistic traffic patterns is therefore essential to obtain realistic results by means of the traffic generation tool(s).
Repeatability and reproducibility	As the traffic generation tool takes care of the centralization of configuration in both the learning and deployment step, the performed results are fairly easy to repeat and reproduce as long as the traffic generation tool is made available to the third party. The traffic generation tool has been designed in such a way that it operates in a closed loop system as much as possible. Therefore, the traffic generation tool configures everything starting from the operation system up to the behavior of individual subscribers' traffic and their corresponding TCP stacks.

5 Routing system

5.1 Routing system scalability

Our experiments are based on a "replay" of BGP monitoring traces. This replay is possible by injecting BGP dumped messages into our implementation of Path Exploration within XORP.

We do not collect ourselves BGP messages. Instead, as explained below, we download BGP messages from the Routeview Project and RIPE BGP traces. As a consequence, for a base experimental setting, a single machine running our implementation into XORP is enough for performing the tests. The iLab.t platform is thus not strictly necessary for re-playing the BGP traces collected off-line. This explains why we consider use of stand-alone machine(s).

Experimentation	
Platform and input data	**System structure and platform:** For experiments using the Routeview project date, a single box, with the following configuration: - Operating System: Linux 64 bits, with kernel 2.6.28 - CPU: Intel Xeon E5430, quad-core - 6 MB of L2 cache shared by pairs of cores - 32 kB of L1 cache on each core - 4.8 GB of main memory For experiments using the RIPE BGP data, two servers with the following configuration: - OS: CentOS, with Linux Kernel 2.6 - CPU: Intel Quad-Core Xeon E5405, 2.6 GHz (12MB Cache) - Memory: 32 GB DDR2-667 registered ECC (16 DIMMs) - Disk: 4 Seagate 300 GB SAS drive, 15,000 rpm - NIC: Intel® PRO/1000 PT Ethernet Server Adapter, 2x RJ45, PCI-e **Experimental configuration and input description:** BGP updates are taken from the Routeview Project. We consider one month of BGP Update (November 2009). These updates were recorded from a total of 42 peers at the Oregon Routeview monitor. The total number of Update message is about 89 millions BGP updates are taken from the RIPE Routing Information Service (RIS). RIS is a RIPE NCC project that collects and stores routing data from the Internet, on several locations around the globe. RIS offers nice tools bringing up this data to the Internet community. Raw data are collected by the RRCs using Quagga routing software, stored in MRT format. This format is described in the IETF document entitled MRT routing information export format draft-ietf-grow-mrt-11.txt). These files can be read using libbgpdump, a library written in C, currently maintained by the RIPE NCC.

The Scientific validation is the same for all experimental scenarios.

Scientific validation	
Verifiability	Decisions taken by the Learning algorithm can be verified based on the MRT files from input. As we are "replaying" BGP message traces, the decision taken can be easily verified.
Reliability	In usual programs based on MRT feeds, BGP messages are all injected at once. This creates an issue when one wants to evaluate the efficiency of the learning based on measurements that depends on time, such as the message arrival frequency. Fortunately, MRT files contain a timestamp for each message. We add a virtual clock inside the memory processing state. This clock is modified dynamically on the fly according to the timestamps contained in the MRT file. All measurements are then based on that virtual clock.
Repeatability	We use binary MRT files downloaded directly from the Routeview website and RIPE NCC website (http://www.ris.ripe.net/risreport/). Our program can read them in bzip2 format or already uncompressed. One can specify a single directory where all the MRT files are stored (in this case, they will be processed in an alphabetical order) or a single MRT file. All BGP messages recorded in the MRT will be parsed, then withdrawal instructions will be injected in the pipeline (remind that the XORP BGP update processing is implemented as a pipeline of XORP processing stages), followed by announcement instructions along with the attributes contained in the message. Raw data are collected by the RRCs using Quagga routing software, stored in MRT format. This format is described in the IETF document entitled MRT routing information export format draft-ietf-grow-mrt-11.txt). These files can be read using libbgpdump, a library written in C, currently maintained by the RIPE NCC. BGP UPDATE messages are parsed and stored in the Adj-RIB-In, processed by the machine learning algorithm. During the learning phase, the BGP UPDATE messages are grouped per time window (Max_AS-Path – Min_AS-Path) x MRAI per destination prefix, with MRAI being the minimum route advertisement interval. The result of the processing of the Adj-RIB-In entries replaces the default BGP route selection process for that prefix.
Reproducibility	Our experiments are easily reproducible as we consider well known and freely available BGP data (Routeview Project and RIPE RIS Raw BGP Data). As long as the same dataset is used in input, the results are reproducible.

5.1.1 Run-time memory cost

Experimental scenario description	
Title	Run-time memory cost
Technical objective	The objective of this scenario is to evaluate the scalability of implementing a learning algorithm to help the BGP Path Exploration process.
Participants	UCL, ALB

Content	
Short description	Evaluation of the scalability of our implementation in terms of memory usage. Confront our implementation to real BGP UPDATE messages.
Expected result(s)	We want to determine if the memory cost explodes or if it stays to a reasonable level. Ideally, the memory usage should reach a stable stage at some point, meaning that the memory usage does not grow infinitely.
Evaluation Criteria and Metrics	The following metrics are considered: - The number of distinct attributes (in terms of ML) - The number of distinct network prefixes managed/processed - The memory usage (in MB) We consider cases when the Memory processing stage is enabled or not. When enabled, we vary the history size. Considered values are: 2 attributes, 4 attributes, 5 attributes, and 10 attributes.

5.1.2 Filtering of BGP messages

Experimental scenario description	
Title	Filtering of BGP messages
Technical objective	Evaluation on how a better scalability can be reached by filtering some BGP messages.
Participant	UCL

Content	
Short description	We want to evaluate the impact of filtering some BGP messages and determine whether such a filtering leads to a better scalability. There are mainly two reasons for filtering: - Software routers (Quagga/Zebra) are often used to collect BGP updates from remote peers with multi-hop BGP sessions. They frequently suffer session resets during which the entire BGP table has to be re-transferred. - We are only interested in Path Exploration generated by AS_PATH modification. Therefore, any Path Exploration triggered by modifications in some BGP attributes (such as med or community) should be considered as noise. In our filtering process, all the updates whose AS_PATH attributes make no changes to BGP router table will be filtered out.
Expected result(s)	We expect a drop in the amount of BGP Update messages that must be processed by our Path Exploration module. Acting so, we believe we could achieve a better scalability by reducing the amount of required memory
Evaluation Criteria and Metrics	We consider the same metrics and criteria than those previously exposed. In addition, we want to determine the proportion of Updates messages filtered out.

5.1.3 BGP transient overhead reduction

Experimental scenario description	
Title	BGP transient overhead reduction
Technical objective	Evaluation on how a better scalability can be reached by removing BGP path exploration sequences
Participant	ALB

Content	
Short description	With the proposed mitigation method, the BGP decision of the route selection process is anticipated upon path exploration event detection and identification (characterization). This involves actions to suppress the churn on downstream nodes,

	such as selecting the alternate best AS_Path to be advertised to the BGP peers. Henceforth, this experiment measures, the actual number of BGP UPDATE message per prefix as received by a downstream BGP speaker.
Expected result(s)	If the HMM-based machine learning algorithm performs as expected, a downstream BGP speaker should not receive any BGP UPDATE other than the next stable routing state.
Evaluation Criteria and Metrics	The following metrics are used on the ML equipped server: 1. The number of actual path exploration events detected (true positives). Note that we might encounter false positives (i.e., a sequence of events is labeled as path exploration while it is not the case) and false negatives (i.e., a sequence of events is ignored while it should have been labeled as path exploration). 2. The time required for the detection of path exploration event. 3. The correctness of the selected path and the proportion of correctness of selected AS_Paths: for the number of path exploration events detected the number of events for which the next stable sequence is returned in the Loc-RIB and the Adj-RIB-Out. 4. The probability of selecting a wrong AS_Path and the impact of selecting a wrong AS_Path. The impact of selecting the wrong AS_Path is the deviation of the wrongly selected AS path from the AS_Path that would be selected after convergence (i.e. after full path exploration phase). From this deviation an estimate can be achieved on the number of AS's that will be affected by that decision. The following metrics are used on the downstream BGP node: 1. The number of actual path exploration events -not-detected. 2. The time required for selecting the AS_Path as received from the incoming BGP UPDATE message. 3. The correctness of the received AS_Path and the proportion of correctness of received AS_Paths (as selected by the upstream router).

5.2 Routing system quality

5.2.1 Learning results

Experimental scenario description	
Title	Learning results
Technical objective	Demonstrate the feasibility of implementing machine learning techniques within a real BGP router by using measurement's history provided by the Memory processing stage.
Participants	UCL, ALB

Content	
Short description	Evaluation of the performance of the C4.5 algorithm when confronted to real data. Evaluation of the performance of the HMM algorithm detailed in D3.6. Note that additional improvements have been proposed that account for non-exploratory withdrawal states.
Expected result(s)	The proportion of correct decisions should be high enough. A classifier is a function that maps observed AS-paths to BGP state event classes. The goal of the learning process is thus to find a function, i.e. a classifier, that correctly predicts the class of topologically correlated AS-path(s) with the minimum expected cost. The cost function assesses the penalties associated to the selection of BGP routes that contain (part of) the path exploration sequence: - missed path exploration events - false positive detections: the classification declares a path exploration event when in reality there is none; such an error may typically occur when decision is taken too rapidly - and false negative detections: the classification does not declare an event to be a path exploration event when in reality it is; such an error typically occurs when the decision is taken too slowly.
Evaluation Criteria and Metrics	We evaluate the performance of the C4.5 algorithm as follows: - First, we look at the Beacon messages coming from two different ASes (AS3549 and AS852). The machine learning algorithm, in such as case, is applied on a per peer basis. - Second, we consider the history coming from several peers at the same time. This dataset is obtained by merging the sequences of all BGP updates coming from all the 42 peers available at the Oregon Routeview monitor. Metrics considered are the learning set error ratio and the test set error ratio. The following metrics are measured on the ML equipped server: 1. The number of path exploration events detected (true positives), the rate of false positives (i.e., a sequence of events is labeled as path exploration while

	it is not the case) and false negatives (i.e., a sequence of events is ignored while it should have been labeled as path exploration). 2. The time required for the detection of path exploration event. 3. The correctness of the selected path: for the number of path exploration events detected the number of events for which the next stable sequence is returned in the Loc-RIB and the Adj-RIB-Out.

Experimentation	
Platform and input data	**System structure and platform:** Same hardware setup as for scalability experiments. **Experimental configuration and input description:** Two sources of input are considered: 1. We consider BGP messages from the Routeview project. Two months of data are used: November and October 2009. The data used to train the C4.5 machine learning algorithm is based on Beacons. A BGP Beacon is an unused prefix which has a well-defined schedule for announcement and withdrawal. The pattern used by the Beacons is very simple: a network prefix is announced at time t to be finally withdrawn at time t+2h. The beacon we consider are those announced by the RIPE NCC consortium. 2. We also consider the BGP messages from the BGP RIS Raw Data of the RIPE NIS project. Four separate months of data are used: April'09, July'09; October'09, January'10 and April 2010 as training set. These data are collected at Amsterdam (AMS-IX), Otemachi, Japan (DIX-IE) and Stockholm, Sweden (NETNOD)

6 Summary

6.1 Scheduled experimentation

The following table lists the experimental scenario described in previous sections, sorted by Technical Objective and use case. Partner(s) responsible for the scenario and related section in this deliverable is given as a quick reference.

Table 3. List of experimental scenarios by TO and use case

TO	Case	Experimental Scenario	Section	Partner(s)
TO1	**a1**	Running monitoring applications based on adaptive sampling	2.1.1	INRIA, IBBT
	a2	Validation of the performance and accuracy of the monitoring system	2.2.1	LAAS/CNRS
	a3	Evaluation of the Anomaly Detection System (ADS)	2.3.1	LAAS/CNRS (ULANC)
TO2	**b1**	IDIPS	3.1.1	UCL
		Internet Coordinate System	3.1.2	ULg
	b2	OSPF SRG inference	3.2.1	IBBT, ALB
	b3	Profile-based accountability	4.1	IBBT, ALB
TO3	**c**	Run-time memory cost	5.1.1	UCL, ALB
		Filtering of BGP messages	5.1.2	UCL
		BGP transient overhead reduction	5.1.3	ALB
		Learning results	5.2.1	UCL, ALB

6.2 Functional and Performance validation criteria and metrics

In addition to proof-of-concept experimentation, each of the experimental scenarios is used to perform functional and performance validation thanks to the definition of validation criteria and metrics, as summarized below.

6.2.1 Accuracy

Accuracy appears as both monitoring accuracy (monitoring data matches actual traffic and events, or matches third-party monitoring traces) as well as detection/prediction accuracy (detection rates, number of false positives, etc.). In the context of this deliverable, *accuracy* should not be confused with *correctness* (next section).

i) Monitoring accuracy

Running monitoring applications based on adaptive sampling (INRIA, IBBT):
Network status estimation accuracy measures how well the adaptive sampling results into an accurate estimation of actual network status.

Validation of the performance and accuracy of the monitoring system (LAAS/CNRS):
Data from the monitoring system is compared against actual traffic. A DAG system is used as base-line monitoring tool.

ii) Detection accuracy

Evaluation of the Anomaly Detection System (ADS) (LAAS/CNRS; ULANC):
Detection accuracy of anomalies is examined through detection rates, false positive/negative rates, and rate of undetected attacks. Detection of the ADS will be compared with traffic traces.

Profile-based accountability (IBBT):
The clustering accuracy of new clustering and classification algorithms is examined.

BGP transient overhead reduction (ALB) - Learning results (UCL, ALB):
Detection accuracy for path exploration events, as well as false positives and false negatives is considered. Actual occurrence of path exploration events is extracted from the fixed BGP input data.

iii) Prediction accuracy

IDIPS (UCL):
Impact of limiting the number of measurements on accuracy of delay predictions is analyzed. Predicted delays are compared with path delay measurement samples.

Internet Coordinate System (ULg):
Prediction accuracy of network distances (delays) are compared with delay measurements on the testbed topologies.

OSPF SRG inference (IBBT):
Prediction accuracy of shared risk groups is defined as rate of correct predictions, false positives and false negatives. The predictions can be analyzed by applying non-concurrent SRG failures to the scenario.

6.2.2 Correctness

The usage of a machine learning engine and executing machine learning algorithms impacts the correctness of the outcome and actions of networking techniques. The machine learning engine can be used to provide faster, more scalable solutions, in which case the outcome may suffer in correctness. Alternatively, the objective of using the machine learning engine and ML algorithms may lie in reaching functionality that performs better than traditional techniques; in this case correctness is a performance metric.

i) Correctness as requirement

BGP transient overhead reduction (ALB) – Learning results (UCL, ALB):
The correctness of selected path and AS_paths may be impacted by the removal of BGP path exploration sequences.

ii) Correctness as performance gain

Evaluation of the Anomaly Detection System (ADS) (LAAS/CNRS; ULANC):
Correctness is defined through the ability of the ADS to detect unknown (0-day) anomalies.

Internet Coordinate System (ULg):
Higher correctness (shorter paths) is reached by finding routing shortcuts which can be found from the ICS using a triangular inequality violation detector.

6.2.3 Timing

Timing of actions initiated by the machine learning engine generally has a direct impact on network user experience. Reaction times, recovery times

and detection times are the main performance metrics considered. Timing may be related to stability.

Running monitoring applications based on adaptive sampling (INRIA, IBBT):
Reaction time needed to readjust configuration of the monitors.

OSPF SRG inference (IBBT):
Firstly, SRG failure detection time is considered; this time is determined by the number of link failure events (in the same SRG) needed to predict the SRG as failed. Secondly, total failure recovery (i.e., re-establishment of connectivity) time is considered.

Profile-based accountability (IBBT):
Reaction time taken to changes in user traffic which exceed the profile.

BGP transient overhead reduction (ALB) – Learning results (UCL, ALB):
Detection time of path exploration events.

6.2.4 Stability

Stability can be expressed against a number of performance metrics.

IDIPS (UCL):
Stability in terms of measurement load and gain.

OSPF SRG inference (IBBT):
Stability in terms of prediction accuracy over time (i.e., total number of recorded failures).

6.2.5 Scalability

Scalability can be expressed against a number of performance metrics. Also scalability is expressed against one or more scenario parameters (e.g., topology, input traffic).

Running monitoring applications based on adaptive sampling (INRIA, IBBT):
Scalability in terms of traffic collection overhead, against topology size and number of interfaces.

IDIPS (UCL):
Scalability against number of requests, number of concurrent paths.

OSPF SRG inference (IBBT):
Scalability in terms of number of OSPF messages, memory usage and processing time, against topology size.

Run-time memory cost (UCL, ALB):
Scalability in terms of memory usage, number of prefixes and attributes.

Filtering of BGP messages (UCL):
As above, additionally impact of filtering on scalability is examined.

6.2.6 Quality of Service

QoS is a performance metric which is directly noticeable by the network users.

OSPF SRG inference (IBBT):
End-to-end packet loss during failures is determined.

Profile-based accountability (IBBT):
Impact of traffic exceeding the profile on QoS of traffic of other users is examined. Related to this, fairness is considered as well.

Table 4. Matrix summary of validation criteria and metrics

		Monitoring accuracy	detection accuracy	Prediction accuracy	Correctness (req.)	Correctness (gain)	timing	stability	scalability	QoS
a1	Running monitoring applications based on adaptive sampling						■		■	
a2	Validation of the performance and accuracy of the monitoring system	■								
a3	Evaluation of the Anomaly Detection System (ADS)		■			■				
b1	IDIPS			■				■	■	
	Internet Coordinate System			■		■				
b2	OSPF SRG inference			■			■	■	■	■
b3	Profile-based accountability		■				■			■
c	Run-time memory cost								■	
	Filtering of BGP messages								■	
	BGP transient overhead reduction		■		■		■			
	Learning results		■				■			

6.3 Experimental scenario relationships and dependence

The diagram in Fig. 1 shows dependence and relationships between the experimental scenarios within the ECODE project.

The diagram in Fig. 2 outlines dependencies of the experimental scenarios on software, tools, input topologies, traffic traces and testbed setups external to the ECODE project.

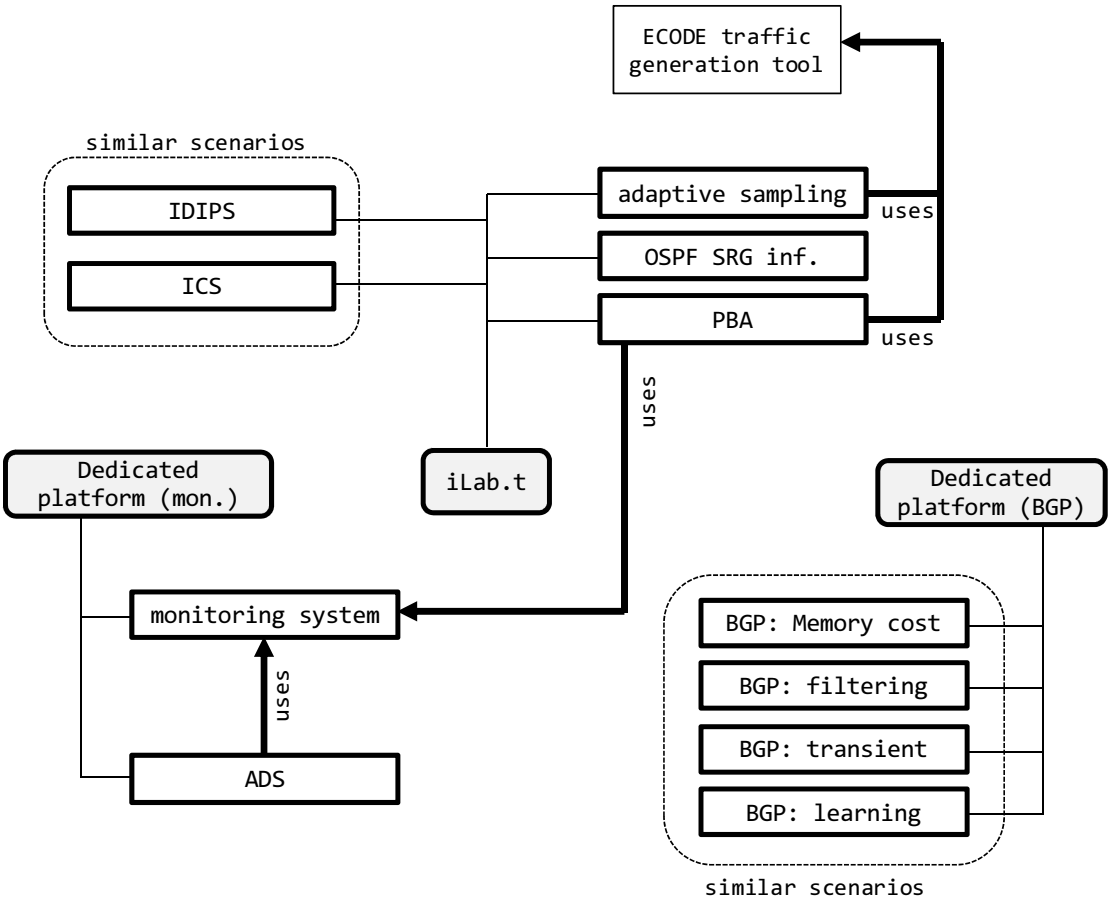


Fig. 1. Experimental scenario interdependency

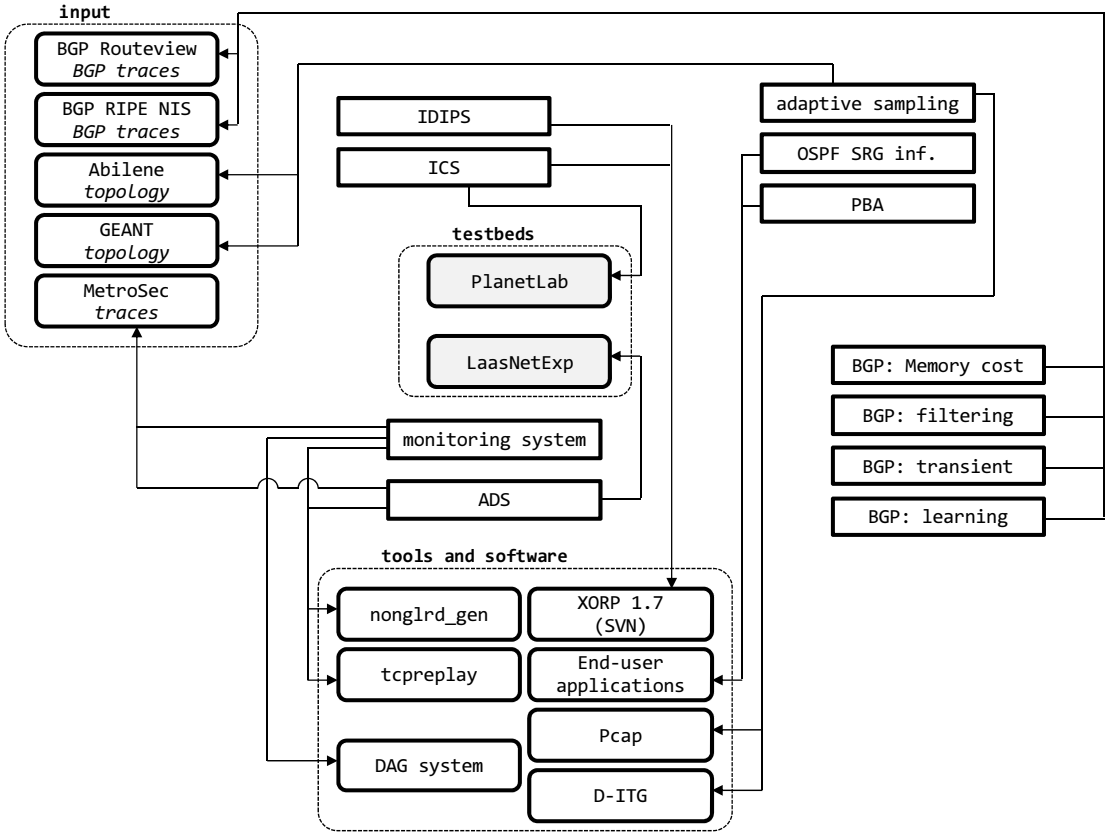


Fig. 2. External dependencies

Annex.1: Template

Experimental scenario description	
Title	A meaningful title for the experimentation
Technical objective	Indicate which technical objectives (use cases) will be used for the experimental scenario.
Task	Indicate which WP4 task the experimental scenario will contribute to.
Participant(s)	Indicate who will be performing this scenario and whether they would prefer or need to interact with other partners for accomplishing this scenario.

Content	
Short description	What do you plan to experiment in more detail than the title and less detail than the remainder (experimental scenario)
Expected result(s)	1) What do you expect to find as results? 2) Why would this be the case?

Experimentation (possibly refer to D3.1)	
Evaluation criteria and metrics	Describe the experimental evaluation criteria, and metrics.
Experimental scenario: models, platform, configuration, traffic, constraints	**Machine learning engine:** List the algorithms and models to be used in the machine learning engine implementation for the scenario. Refer to earlier work done for use cases/technical objectives; explain machine learning approaches that will be introduced in this experimentation track **System structure and platform:** 1) Describe the experimentation platform (test bed) used for the scenario. Include any hardware and/or applications that interact directly with the machine learning engine during the experimentation. 2) Does the implementation interact with implementations from other use cases (by other ECODE participants)? 3) Is the machine learning engine process distributed over multiple machines? 4) Does the experimental scenario require any machine to be terminals? 5) Is there any functionality missing from the test bed platform that must be implemented for the experimental scenario? **Network topology:** 1) What types of network topologies are required for the experiment? 2) What type of connectivity is needed between machines? 3) How many machines are needed? **Experimental configuration and input description:** 1) For the experiment list:

	- background traffic - events (including failures) to be emulated. 2) What are their characteristics? 3) Is the traffic available as a model, as a traffic trace, or will a user application be used instead ('real traffic')? **Constraints:** 1) Does the choice of the above (platform, topology, traffic, etc.) pose any constraints? 2) Does the test bed platform require some abstractions or short-cuts to be taken in the emulation? 3) Are there any special configuration issues to take into account? 4) Is there a confidentiality requirement for some of the experimentation data? 5) Does the XORP platform itself pose any limits concerning accurate timing, throughput, etc.? 6) Are there any protocol particularities that collide with abstractions made during earlier work?
Methodology	**Methodology to obtain experimental data:** 1) Flowchart of the experimental scenario with the functional blocks indicated. 2) Provide an indicative description of the experimentation running time (per step if possible). 3) If the experimentation scenario will be used for multiple experimentation tracks (e.g. functional analysis, performance evaluation, etc.), does each track require a different methodology? 5) Does obtaining the evaluation metrics and performance metrics require external measurement and tools? On the other hand, if these are obtaining from the XORP platform, do the measurements require any extensions to the platform? **Experimental data processing methodology and analysis:** 1) Describe here the methodology for processing and comparing the experimental data (e.g. against reference scenario). 2) If experimental data is collected as a dataset for further processing, describe what type of data is collected (dataset format), and mention how datasets should be identified for processing and analysis (tagging/metadata: input parameters, topology, experiment run time, person doing the experiment, etc.) 3) Provide detailed description of the analysis to be performed on the obtained data (including sensitivity analysis). - Note in case data analysis requires use of a specific tool, please indicate which tool.

Scientific validation	
Verifiability	1) Which formal models will be used in order to verify outcomes of the experiment? 2) How do the experiment scenario and evaluation criteria map to clear definitions that can be used in a formal model.

Reliability	1) Indicate key issues that may impact reliability. Which external factors may invalidate the experimental results? 2) What kind of reliability (time scale) is aimed for in the experimental scenario?
Repeatability	State which steps are taken to ensure that the experimental scenario can be repeated by the scenario participants.
Reproducibility	State which steps are taken to ensure that the experimental scenario can be reproduced by third parties. Which blocking issues may prevent other ECODE partners or external parties from reproducing the experimental scenario?