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

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Deliverable D3.1
“Detailed Experimental Plan and Scenarios”

Project acronym: **ECODE**
Project full title: **Experimental Cognitive Distributed Engine**
Proposal/Contract no.: **223936**

Date of preparation of Deliverable: **February 2nd, 2009**

Document properties

Document Number	ICT-2007-1.6-223936-ECODE-D3.1
Document Title	Detailed Experimental Plan and Scenarios
Document responsible	Philippe Owezarski (CNRS)
Author(s)/editor(s)	Konstantin Avratchenkov (INRIA), Chadi Barakat (INRIA), Olivier Bonaventure (UCL), Benoit Donnet (UCL), Pierre Dupont (UCL), Amir Krifa (INRIA), Yann Labit (CNRS), Imad Lassoued (INRIA), Guy Leduc (ULG), Johan Mazel (CNRS), Juan Pablo Narino Mendoza (UCL), Philippe Owezarski (CNRS), Dimitri Papadimitriou (ALB), Damien Saucez (UCL), Wouter Tavernier (IBBT), Ing-Jyh Tsang (ALB)
Dissemination level	PU
Security Type	PU
Status of the Document	Ongoing
Version & Revision History	Preliminary v1.0 - February 2009

Table of Content

1. Description of the Document

This document is the ECODE ***deliverable D3.1***, draft version-01. This deliverable is the result of the first thoughts on how to assess and validate by means of experimental scenarios (to be emulated on the IBBT ILab platform) the combination of machine learning with advanced networking techniques as proposed in deliverable D2.1. This document is nevertheless subject to periodic updates, as the ECODE project will be moving on.

As outlined is the Technical Annex B, the ECODE project comprises two experimental phases. During the first experimental phase (from M03, Dec.2008 to M16, Mar.2010), the machine learning and networking techniques are combined in the cognitive router system. During this first experimental phase, the applicability, the validity and feasibility of introducing a cognitive component/engine in the network node elements - referred to as cognitive routers - are experimented using a number of demonstrative use cases. These use cases cover different problem in representative areas, identified as Internet architectural and design challenges (such as security, controllability, routing, and accountability).

All the use cases studied in the context of the ECODE project are validated and assessed by means of one or several different experimental scenarios aiming at illustrating the benefits and applicability of our proposals, especially in term of machine learning. For this purpose, use-case driven experimentation will be performed on the cognitive engine elaborated in the context of the WP2 and described in the deliverable D21 "Cognitive Engine - Experimental Network and Architecture".

Description of the experimental scenarios intends to answer a set of questions on (1) Technical objectives, (2) Content, and (3) Description of the experimentation itself. 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, and methodology;
- Experimental report(s) including iteration/feedback on described solution.

This first experimental phase will be conducted in physical experimental infrastructures, more precisely, using the iLAB physical experimental facility, located at IBBT premises in Ghent, Belgium (see Annex 2).

2. 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 inline 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 cognitive component as part of the Internet routing system.

This first experimental phase will be conducted in physical experimental infrastructures, more precisely, using an emulation facility. The cognitive engine will be experimented and its performance evaluated by means of a dedicated and fully controlled emulation platform: the iLAB experimental facility, located at IBBT premises. The objective is to set-up a realistic environment able to reproduce as closely as possible, real network and operational conditions. In such a realistic emulated environment, it will be possible to i) run the cognitive engine designed in WP2 with the set of use cases, grouped into technical objectives, and ii) obtain feedback from its validation and performance evaluation, for improving the cognitive engine accordingly. At the end of this first experimental phase, the results will be reported back to WP2 so as to appropriately adapt the architecture outlined in the deliverable D21 "Cognitive Engine - Experimental Network and Architecture”.

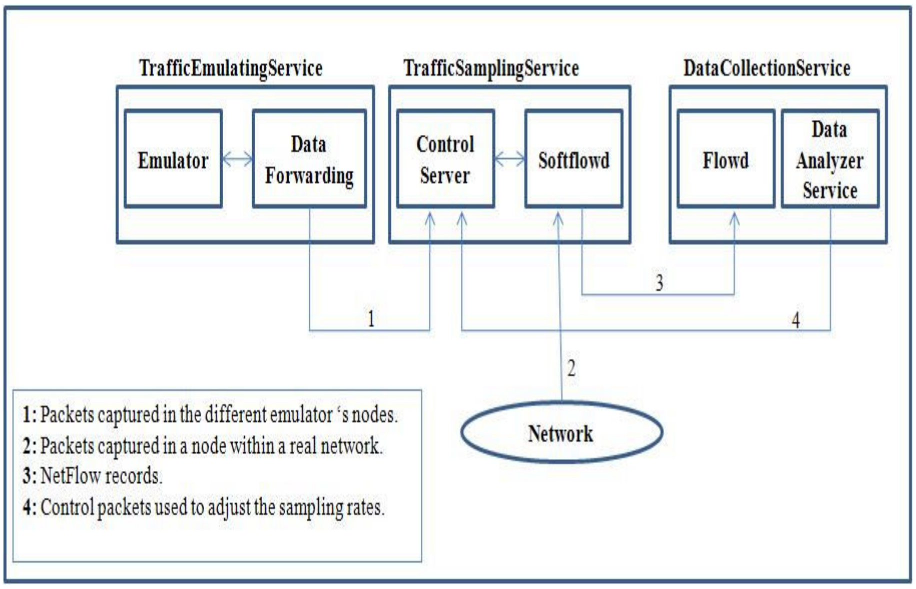
The components of the cognitive router and in particular the cognitive engine will be experimented on the ILab cluster available at IBBT offices by means of representative use cases. As for D2.1, description of the experimentation for the use cases is clustered into four groups:

- *Security and monitoring*. This 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*. This corresponds to use cases c1 and b2, i.e., the development of a solution for speeding up the BGP path exploration process and allowing fast network recovery.
- *Path selection*. This corresponds to use case b1, i.e., the development of a solution for allowing application (of any kind) to rank paths according to particular criteria.
- *Profiling and Accountability*. This 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.

This deliverable describes the experimental scenarios, and tools that will be developed and/or used to determine applicability of the proposed machine learning techniques. For each of these use cases, we describe, in this deliverable, one or several experimental scenarios for assessing and/or validating some or all the functional components of the cognitive router. Each description comes with the metrics and evaluation criteria as well as the expected results so as to assist evaluation of the obtained results. The template used to obtain this information is available in Annex 1. This deliverable provides an analysis of the experimental tools that will be used in each experiment and methodology that will be conducted for each experiment in order to facilitate interpretation of the obtained results and enable systematic re-production of results.

3. Monitoring and Security

3.1. Adaptive Traffic Sampling

Use case scenario description	
Title	Adaptive Traffic Sampling and Management
Technical objective	Our main technical objective (part of the Technical Objective 1) consists on building a distributed adaptive traffic sampling platform which will try to better reach the monitoring application requirements by optimally configuring the sampling rate in monitors, while reducing the overhead on routers and the volume of collected traffic. Our platform, as described on the figure below, will include the following services:  TrafficEmulatingService Emulator Data Forwarding TrafficSamplingService Control Server Softflowd DataCollectionService Flowd Data Analyzer Service 1 2 3 4 Network 1: Packets captured in the different emulator 's nodes. 2: Packets captured in a node within a real network. 3: NetFlow records. 4: Control packets used to adjust the sampling rates. – **TrafficEmulatingService**: since we have no access, for the moment, to traffic monitoring points and traces in a real network, we are implementing a service for traffic emulation. As input for TrafficEmulatingService, we use a real TCPDump trace collected in one or more points in a network which we re-assign to the edge routers of the emulated topology. Note that we can select different monitoring points among the emulated devices, for which we associate the SamplingService we describe in the next point. – **SamplingService**: based on the feedbacks received from the DataAnalyzerService, the SamplingService will sample the network traffic (real or emulated traffic) and send the sampled flow data to the DataCollectionService. – **DataCollectionService**: this service will just collect the received sampled flow records from the different monitoring points. – **DataAnalyzerService**: given a monitoring application, this service will analyze the collected data, estimate the network status using unsupervised machine learning techniques and use this estimation to send feedback reports to the monitoring points in order to modify their sampling rates to meet the measurement task requirements. Note that only the SamplingService will be installed in the network routers

	chosen as monitoring points, while the DataCollectionService and the DataAnalyzerService will be installed in the controller node. In our platform, we intend to use unsupervised machine learning techniques within the DataAnalyzerService in order to estimate the network status based on the collected data records. We will focus on techniques like the maximum likelihood, the Kalman filtering and the Expectation Maximization (EM) method. The performance improvement which we are expecting by the use of those ML methods will be expressed in terms of a better estimation of network status and hence a better setting of sampling rates that improves monitoring accuracy and reduces the overhead on routers.
Participant(s)	This scenario will be performed by INRIA with a possible collaboration with LAAS/CNRS and ULANC.

Content	
Short description	The purpose is 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.
Expected results	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 number of packets sent by a given AS – the greediest users – the number of packets per flow – the detection of anomalies. Our methodology is based on Machine Learning methods (the maximum likelihood, the Kalman filtering and the Expectation Maximization method) in order to improve the network status estimation accuracy. So, based on this estimation, an end user will be able to run efficiently some monitoring application.

Experimentation	
Evaluation criteria and metrics	The experimental evaluation metrics are the following: - 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	**Experimental scenario description**: Since we have no access to different live monitoring points in a real network, and as a first step, we intend to use the emulation platform we are developing (consisting on the TrafficEmulatingService which we already described) in order to replay TCPDump traces at a packet-level. The topology of the emulated scenario will be the following: – A transit network, where we intend to deploy a set of nodes, among which we will choose the monitoring points (one option is to enable all routers with monitoring capability). – To this transit network we will connect a set of AS(s). The interconnection scheme and link characteristics will depend on the

	TCPDump traces that we will replay in our emulation platform. **Experimental tools**: The following is the list of the Open Source tools we intend to use and modify in order to reach our technical and experimental objectives: – **Softflowd:** a flow based network traffic analyzer capable of Cisco NetFlow data export. Softflowd semi-state fully tracks traffic flows recorded by listening on a network interface or by reading a packet capture file. These flows may be reported via NetFlow to a collecting host. Note that Softflowd does not include support for traffic sampling and we are designing our SamplingService in such a way that it will include the Softflowd tool around which we will build other modules. – **Flowd:** a secure NetFlow collector. It offers the following features: - Understanding NetFlow protocol v.1, v.5, v.7 and v.9 (including Ipv6 flows). - Supports both Ipv4 and Ipv6 transport of flows. - Supports filtering and tagging of flows, using a packet filter-like syntax. - Stores recorded flow data in a compact binary format. Flowd works with any standard NetFlow exporter, including hardware devices or software tracking agents, such as Softflowd. Note that Flowd does not include support for performing data analysis and as we have done for Softflowd, we are designing the DataAnalyzerService in such a way that it will include Flowd around which we build other modules. Also, these modules will enable as to support different monitoring applications. The selected experimental tools (Softflowd and Flowd) need to be extended in order to reach our technical objectives and to run our experimental scenario. In particular, they should be extended to perform sampling, to learn about network status and to reconfigure the sampling rate in monitors. **Experimental configuration and running conditions**: For now, we are using different TCPDump traces at a packet-level (including traces from the traffic data repository at the WIDE project), which we will replay using our TrafficEmulatingService. Discussions are currently going on with GEANT to obtain real NetFlow and TCPDump data from the entire GEANT backbone. **Experimental execution and measurements**: As a first step, we will connect the passive measurement tools to the nodes/routers we will choose as monitoring point within our emulation platform. Those tools will sample the traffic in emulated routers and send sampled flow records to a central collector that will analyze them, estimate network status, evaluate accuracy then reconfigure monitors. Afterwards, we will be able to run the tools we are developing and enhancing on real network nodes (for example nodes within the ILAB-T platform).
Methodology	**Methodology to obtain experimental data**: As a beginning we are working on the emulation of real network traffic starting from a packet-level trace collected on one ore more points inside the network. These TCPDump traces need to be transformed as explained

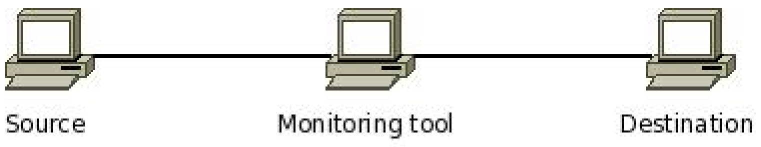
	later. In parallel we are working on the collection of network-wide traces that can be used as they are without the following transformation. **Experimental data processing methodology and analysis**: The methodology to process the experimental TCPDump packet-level traces data could be summarized in the following steps: – Selection of the transit network topology. – Choosing the number of AS(s) to inter-connect to the transit network and their characteristics (IP(s) range…). – Re-assignment of the initial TCPDump traces to the different AS(s) based on their characteristics. – Choosing the monitoring points among the transit network nodes and connecting the SamplingService to each one of them. – Running the emulation (the TrafficEmulatingService) and consequently the related services (SamplingService, DataCollectionServices and DataAnalyzerService). – Using the DataAnalyzerService to perform some monitoring application (The number of packets sent by a given AS, the greediest users, the number of packets per flow, and detection of anomalies) based on the collected data records.

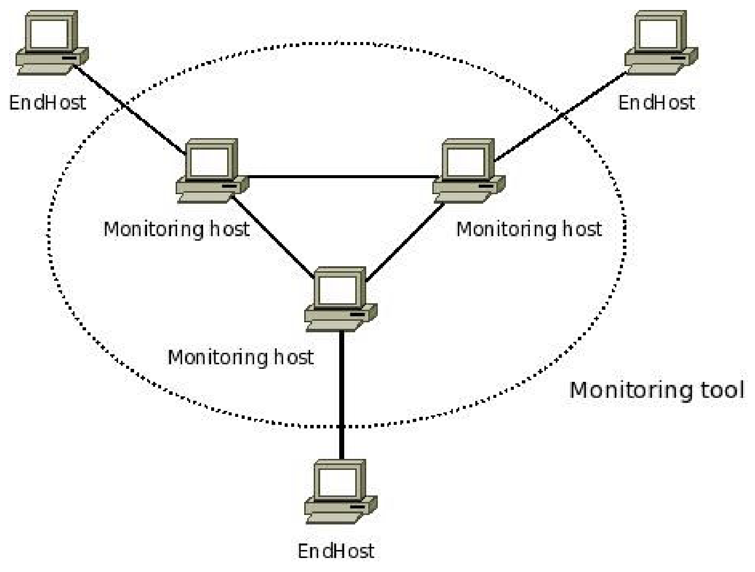
3.2. Global/Active Monitoring

Use case scenario description	
Title	Validation of the global 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 with a possible collaboration with UCL, and INRIA.

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.
Expected results	The expected outcome of this setup is to build a reliable and distributed passive monitoring system (i.e. no packet loss, accurate timestamping) that provides for accurate and fast software-based monitoring and measurement capability.

Experimentation	
Evaluation criteria and metrics	For the monitoring entity, 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. For the reporting mechanisms several other criteria and metrics will be evaluated. The criteria for the reporting mechanism must be as fast as possible (without loss) and as transparent as possible. The metrics are – Loss rate of reporting packets

	– Delays in reporting (delay between reception of reporting message on one host and the production of the measurement data) – Throughput of reporting traffic (more precisely a statistical description of the reporting traffic generation process: inter-packet times distribution, packet size distribution).
Experimental scenario: description, tools, configuration and running conditions	**Experimental scenario description**: 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. Practically speaking, our experimental scenario consists of two steps. These steps are required for validating the performance of a single monitoring entity (step 1) and for validating the global reporting mechanism between entities of the global monitoring system (step 2). For this purpose, we will have to: 1. Test the monitoring tool on one host (to assess the reliability of the capture and timestamping mechanisms). 2. Test the monitoring tool on several hosts (to assess the reliability and the performance of the reporting/communication system and the timestamps). **Experimental tools**: 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 experimental platform. They will then be used to evaluate the performance of accuracy of the global software monitoring system specifically designed and developed for ECODE and that will be then deployed on the ILab experimental facility cluster. **Experimental configuration and running conditions**: 1. For the first step of validation, the following topology will be used.  For the global reporting mechanism, a larger network is necessary. The topologies depicted on the figure will be used. We are limited by the number of available DAG systems. Hence, we won't be able to make a perfect evaluation with a larger number of machines. Nevertheless, we will evaluate delay and loss rate for larger scenarios. But in this case we will not be able to evaluate accurately the reporting packet generation process, because of the lack of DAG system (we only own 3 of them). Link characteristics will be the ones classically observed on actual networks. 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. We will apply a similar method for creating losses. 2. For the second step of validation, the following topology will be used.

	 EndHost EndHost Monitoring host Monitoring host Monitoring host Monitoring tool EndHost 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. In both cases (1 and 2), 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). **Experimental execution and measurements**: 1. For evaluating a single monitoring entity we need to use a DAG system on the same link, close from the monitoring entity to be evaluated. 2. For evaluating the global reporting with the depicted topology, we need 3 DAG systems. For larger topologies, we will reduce our ambitions in evaluating the generation process of reporting traffic. It will nevertheless be quite instructive for evaluating all other metrics aiming at evaluating the performance criteria.
Methodology	**Methodology to obtain experimental data**: Most of data will be provided by DAG systems. **Experimental data processing methodology and analysis**: We will then develop all necessary tools for analyzing them. These tool will have to: 1. 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. 2. Compute loss rate, delays, and provide characteristic of the generation process for the reporting traffic.

3.3. Anomalies Detection

Use case scenario description	
Title	Evaluation of the cooperative distributed Anomaly Detection System (ADS)
Technical objective	Technical Objective 1. This objective relates to the development of a distributed intrusion 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.

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. 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 if the collaboration mechanisms between distributed ADS entities can help improving the accuracy of the decision making process.
Expected results	We expect to find lower false alarm ratios. We also expect to differentiate the improvement due to local ML techniques which are used for continuously learning about the traffic characteristics, and then finding more accurately any anomaly, from the ones due to collaboration mechanisms between distributed ADS entities. In this later case, the collaboration and ML techniques aim at learning about the network status and traffic characteristics in a distributed way and making decision from a global view of the network and its traffic. The continuous online analysis of the traffic made by each ADS entity, and the distributed reporting of this information to all other entities should make possible to learn more efficiently and accurately from the network traffic and make more accurate decisions.

Experimentation	
Evaluation criteria and metrics	The performance of the ADS will be evaluated on the false negative and false positive rates. The false positive 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: description, tools, configuration and running conditions	**Experimental scenario description**: To be detailed. **Experimental tools**: There is no need for a dedicated measurement system to evaluate the ADS. However, we need to develop a tool for injecting replayed traces in the emulation network. The other tools, as the topology generators, are supposed to exist but as the

	emulated scenario will remain small and such tool might be useless. It would be useful only if the emulated scenario is large enough to force us to respect some of the actual characteristics of current real networks. **Experimental configuration and running conditions**: Several topology configurations will be necessary. The first one will only consist of a single machine running the local ADS, and a machine for injecting the traffic on the link the local ADS is monitoring. For evaluating the global ADS and its collaboration techniques, several network topologies with different numbers of machines and different interconnection topology will be necessary. We will start from 2 machines running local ADS and collaborating to a much large number. We will try to make this number as large as possible. Reaching a total of 5 machines for hosting the ADS entities (which does not mean anything on the number of traffic generators that will be necessary in the experiment – it depends on the interconnection topology we would like to experiment). Link characteristics 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. This is important in such 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.). **Experimental execution and measurements**: 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.). Such traces exist and are the property of CNRS. They were issued in the framework of the French MetroSec project. A measurement and monitoring system is the basis for the machine learning algorithms used in the local and global ADS). 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.
Methodology	**Methodology to obtain experimental data**: The experimental data produced by the ADS under evaluation are 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 traces. The number of false alarms and undetected attack is then obvious to compute.

4. Routing

4.1. BGP

Use case scenario description	
Title	Using ML techniques for reducing path exploration in BGP
Technical objective	As part of the Technical Objective T03 (see Technical Annex), the purpose is the improvement of BGP convergence time by means of the detection of path exploration event(s) and, if possible, accelerate the whole process to skip selection process directly to the actual AS_Path that will be used.
Participant(s)	Olivier Bonaventure (UCL), Benoit Donnet (UCL), Juan Pablo Narino Mendoza (UCL), Pierre Dupont (UCL), Damien Saucez (UCL), Dimitri Papadimitriou (ALB)

Content	
Short description	We want to be able to detect path exploration process in BGP and, if possible, accelerate this process (i.e., avoiding the exchange of useless BGP UPDATE messages between routers). The aim is to eliminate detrimental effects of path exploration on BGP convergence time. For this purpose, after detection of a path exploration event, a router could decide to directly select the actual (and correct) AS_Path that will be used for a given destination prefix and subsequently advertised to its downstream peers. A desired side effect of this acceleration would be to reduce the BGP churns generated during the path exploration process.
Expected results	The first step consists in detecting path exploration process (or at least a certain proportion of them) when they occur by identifying the characteristics of this process from received BGP UPDATE message sequences (patterns). In the next step, we would like to determine the right path to use before the end of the exploration period in order to stop the process and directly select the right path. The final step is to predict when an exploration process would occur and compute the correct path by limit the path exploration at its maximum (no exploration at all would be the best). For the first step, we would obtain a tool determining "for the traffic you provide me, *n* exploration processes have been found and here they are". For the second step, the tool should alert before the end of the detected exploration process, which are the right AS_Path to use. Finally, the tool should predict when path exploration would occur, anticipate the decision of the BGP route selection process, and its subsequent advertisement of the selected AS_Path to downstream peers.

Experimentation

Evaluation criteria and metrics	The following metrics will be used: 1. Number of actual path exploration process detected. 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. For instance, after how many events the sequence is labeled as path exploration. 3. The correctness of the selected path. This is a Boolean value (the selected AS_path is correct or not). 4. The proportion of correctness of selected AS_paths. 5. 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.
Experimental scenario: description, tools, configuration and running conditions	**Experimental scenario description**: We will consider inter-domain topologies of various sizes i.e. comprising various number of Autonomous Systems (AS), and various meshedness among AS but comprising at least one 3-cycle in the AS-graph (larger cycles result in longer path exploration event). Using these topologies, inter-domain link can fail and be started on demand to force a BGP re-convergence. **Experimental tools**: As a first step, our experimentation will be done through simulations so that we keep a complete control on the whole configuration and execution process. We are currently evaluating two simulators Simula BGP and SimBGP. i) **Simula BGP**: the Simula BGP tool is designed to experiment the dynamics and propagation of routing updates in large AS-Graph where each router has Adj_RIB_IN, Loc_RIB, Adj_RIB_OUTs table. However, the Simula BGP tool considers only inter-domain routing information exchanges and inter-AS topologies (each AS is an atomic node). The main advantage of this simulator is that it comes with a large number of inter-domain topologies. ii) **SimBGP**: this simulator (http://www.bgpvista.com/simbgp.php) older but it has been used to perform the simulations described in several scientific papers. Its main advantage is that it contains a more detailed BGP model than SimulaBGP since it supports Route Reflectors and internal BGP (iBGP) i.e. intra-AS BGP connections. However, its scalability properties are still under investigation. If the topologies provided by Simula BGP and the inferred topologies available on the Internet are not sufficient, we will generate additional topologies by using a topology generation such as IGen. This topology generation software uses network design heuristics such as MENTOR, MENTour, Delaunay triangulation and Two Trees for the purpose of building network topologies. For more details, see http://www.info.ucl.ac.be/~bqu/igen/. **Experimental configuration and running conditions**: For simulations, we will make use of “simple” topologies and detect, off line (so after the execution of the simulations), path exploration traces. Then, those

	traces will be post-processed and we will check whether the ML techniques can be used. **Configuration 1**: simulations using BGP simulators executed with one router per AS/external BGP (eBGP). This is the base and most simple case that will allow us to evaluate the suitability of different machine learning techniques for the BGP path exploration problem. **Configuration 2**: we will expand the AS-level inter-domain topologies by adding internal BGP (iBGP) and multiple links between AS's. Then, we will add Route Reflectors. We will first evaluate whether the successful ML techniques found in configuration 1 are still successful with configuration 2. If this is the case, we will expand the topologies to evaluate the limits of the ML techniques. Otherwise, we will consider other ML techniques. **Configuration 3**: assuming all steps are passed successfully, small topologies extracted from configurations 1 and 2 will be then executed over a real BGP implementation (XORP or Quagga) in ILab-T environment. **Experimental execution and measurements**: The simulations will be conducted on servers. The emulation experiments will be conducted in the ILab-T environment.
Methodology	**Methodology to obtain experimental data**: The incremental methodology will include the following steps: **Step 1**: simulations using BGP simulators. This step is decomposed as follows: Step 1a will be executed with one router per AS/external BGP (eBGP). Step 1b will consider at least two routers per AS/internal BGP (iBGP) and Route Reflector. The goal is to obtain early results so as to determine whether path exploration can be observed or not from the BGP monitoring/trace logs obtained by executing simulation(s). **Step 2**: same as the previous steps but using more complex topologies as those generated by IGen. **Step 3**: emulation using XORP (or Quagga). Assuming all steps are passed successfully. This step requires further evaluation resulting from the relative complexity of modifying and/or extending BGP modules to support the selected ML technique. **Experimental data processing methodology and analysis**: During the simulation phases, the incremental methodology described here above makes use of “simple” topologies so as to detect off line (so after the simulations), path exploration traces. Then, the obtained traces will be post-processed and we will check whether Machine Learning (ML) techniques can be used. With respect to machine learning investigations: the incremental approach consists in determining applicability and if this is actually the case progressively automate processing (detection) first off-line and then on-line of path exploration sequences (from the BGP update message sequences): – Steps 1 and 2 intend primarily to determine if ML is applicable to the BGP path exploration case and initial trial of ML. – Step 3 targets application of on-line machine learning.

4.2. Network Recovery and Resiliency

4.2.1. Use Case Scenario 1

Use case scenario description	
Title	OSPF event modeling and clustering
Technical objective	The objective is to model (historical) message sequence behavior of Open Shortest-Path First (OSPF) networks in order to be able to obtain faster failure detection and faster failure correlation. Machine learning coupled with the use of appropriate heuristics is expected to deliver faster reactivity to certain network events based on historical behavior. Such behavior is currently not taken into account to accelerate trigger of shortest-path tree computation.
Participant(s)	This scenario will be performed by IBBT and ALB. Additional collaboration will be possible in the course of execution of this experiment.

Content	
Short description	We want to evaluate how statistical machine-learning techniques are able to model typical protocol dynamics and capture correlations related to Shared Risk Groups and Multi-Access network failures in terms of typical message sequences and message arrival time distributions.
Expected results	By adding functionality to the ECODE cognitive engine, such as to store and model historical information using intelligent structures, we expect that several events, such as correlated failures and atypical network events (failures) can be detected significantly earlier than is currently the case. Current OSPF link-state protocol logic does not take into account the operational and historical environment in which it is running.

Experimentation	
Evaluation criteria and metrics	The evaluation criteria will be the following: - Failure detection time - Switch-over time - Resulting packet loss
Experimental scenario: description, tools, configuration and running conditions	**Experimental scenario description**: The experimental scenario is divided into two phases: 1. In a first phase, OSPF protocol message traces will be recorded for a set of network topologies representative for the different range of events that can happen in operational networks. Both normal protocol dynamics and dynamics on failure events will be generated in an emulated network using XORP and Click-based Bidirectional Forwarding Detection (BFD). For this purpose, a range of failures will be generated simulating single link failures, node failures, Shared Risk Groups (SRG), Multi-Access network failures, etc. Based on these traces, offline learning (probably in a home made simulation platform or using ns2) will allow learning methods to model messages arrival distributions, and message sequence distributions. Feedback in order to allow the ML algorithm to learn can be given once failures are either detected via existing protocol mechanics (HelloInterval expiration) or when resulting Link State (LS) Update message sequences are compared to predicted ones (delayed feedback).

	2. In a second phase, once offline learning results are satisfactory, available methods will be adapted and will be tried in an online setting using XORP. **Experimental tools**: The tools that will be used for this experiment are routing emulation using XORP and Click-based Bidirectional Forwarding Detection (BFD). Tools such as TCPDump or Wireshark will be used to capture experimental data. Data analysis will be performed using either custom made modules or XORP/Click-based software. Probably, machine-learning libraries will be used to quick start the analysis. **Experimental configuration and running conditions**: Experimentation will be executed on emulated network topologies comprising in the range of 10-80 nodes, and having a number of multi-access networks with sufficient routers connected to them. Background data traffic is of less importance because OSPF protocol messages are prioritized. Hence, processing of OSPF protocol messages, in particular, LS PDUs messaging is not impacted by data traffic - at least in the limit of the experimental variables under control in these experiments -. **Experimental execution and measurements**: Trace capturing tools such as TCPDump are available and will be used for this experiment. The XORP routing engine will probably need to be adapted in order to allow communication of received PDUs and related timing information to the cognitive engine. The cognitive engine will be extended such as to learn on the provided data. And an interface towards the XORP routing engine will need to be provided so as to trigger rerouting based on a prediction. Measurement tools will be made available in every network node such as to measure specific characteristics of arrived LS PDUs such as Inter-Arrival Time (IAT), LS PDU field values, etc.
Methodology	**Methodology to obtain experimental data**: A four-step methodology as detailed here below will be used. This process will be iterated until the results are satisfactory, experimenting with different sets of observed parameters. 1. Before-failure protocol dynamics will be modeled using a set of parameters either explicitly available in LS PDU such as the sequence number, the Link-State ID, the Link-State Age or measured such as the Inter-Arrival Time between LS PDU 2. Upon failure, differences in the value of the observed parameters are detected and modeled 3. The “distance” as measured between step 1 and 2, will serve as guideline for predicting future failures 4. A feedback/learning phase is reconciling the prediction with the actual event, being non-failure as in step 1 or failure as in step 2 **Experimental data processing methodology and analysis**: The OSPF (or BFD) protocol engine will be used as the main source of experimental data that will be processed and analyzed. Tools such as TCPDump or Wireshark will capture these data. Data analysis will be

	performed in either custom made simulation modules or XORP/Click-based software. Probably, machine-learning libraries will be used to quickstart the analysis.

4.2.2. Use Case Scenario 2

Use case scenario description	
Title	OSPF cycle detection and prediction
Technical objective	The objective is to experiment techniques to detect transient loops in the routing topology from a local router perspective. Transient routing loops can occur during transient re-convergence phases, resulting in packet losses.
Participant(s)	This scenario will be performed by IBBT and ALB. Additional collaboration will be possible in the course of execution of this experiment.

Content	
Short description	We plan to capture different types of information about active traffic streams using the ECODE monitoring engine in a node. This information will be used to detect the occurrence of cyclic patterns in data traffic. When a cycle is detected, it will be signaled to the Routing/forwarding Engine such as to rapidly take action to break the cycle.
Expected results	Early topology cycle detection will quickly detect loops during the transient phase of network re-convergence. As such, the resulting packet loss and the increased network load during these phases will be significantly lesser compared to the current situation where no cycle detection is applied.

Experimentation	
Evaluation criteria and metrics	The criteria to evaluate the solution existing solutions will be the following: – packet-loss of affected traffic streams – delay of affected traffic streams
Experimental scenario: description, tools, configuration and running conditions	**Experimental scenario description**: A learning component will be trained to detect cycles on a range of representative topologies with topological cycles of increasing size. The cognitive engine of the detecting (local) node will have a set of measured parameters at its disposal such as packet size, source, destination, and TTL. In a training phase a selection of packets being forwarded can be stored locally. This information can then be used as exact feedback to reinforce the learning process to know if there was a cycle or not. The learning will mainly consist of detecting a set of parameters and patterns of these parameters (for example based on time and sequences) representative enough to be able to quickly detect cycles. In an operational environment, the component will be activated upon arrival of an LS Update message, which triggers shortest-path tree re-computation. The goal is thus to have a fast cycle detection tool to prevent selection of routes crossing these cycles - even temporarily -. **Experimental tools**: Additional development will be probably needed in the cognitive component that will be coupled to the XORP routing engine (such that it can feed the routing engine/process of XORP, especially for learning the right sampling

	interval and set of parameters. **Experimental configuration and running conditions**: To perform these experiments, a set of network topologies with cycles of increasing size, and number of nodes between 1-50 will be used. Data traffic will be generated, using: 1. In a first phase, standard traffic generator tools (IPerf, D-ITG, etc.) 2. In a second phase, real traces will be used **Experimental execution and measurements**: Traffic monitoring functionality will be needed in every node as part of the Monitoring Engine (see Deliverable D2.1, "Cognitive Engine - Experimental Network and System Architecture") such that passive measurements can be taken on a learned sampling interval and on a learned set of parameters.
Methodology	**Methodology to obtain experimental data**: The following methodology will be used to obtain experimental data: 1. Learn in isolated environments – Use topology x from a set of topologies with possible cycle size y – Generate traffic in the given topology – Introduce cycles at scheduled random times – Let the component learn, and iterate the process with different topologies and different cycle sizes until the detection time is fast enough 2. Learn during re-convergence – Re-execute the above process only during re-convergence of the OSPF process **Experimental data processing methodology and analysis**: Experimental data processing will be performed using monitoring tools developed for the monitoring Engine.

4.2.3. Use Case Scenario 3

Use case scenario description	
Title	Minimizing packet loss during routing table switch-over
Technical objective	The goal of this experiment is to develop and evaluate a technique that optimizes the routing table update process while taking traffic volume into account during switch over events. Switching over high traffic volumes timely should result into lower packet loss during the switch over event responding on failures.
Participant(s)	This scenario will be performed by IBBT and ALB. Additional collaboration will be possible in the course of execution of this experiment.

Content	
Short description	This scenario will experiment with a traffic volume-based routing update process. Traffic will be monitored using techniques to minimize resource consumption, resulting data will be classified using ML techniques in ordered classes of traffic, and an efficient mechanism to determine the best update blocks and parameters will be evaluated.

Expected results	We expect to find a significant decrease in packet loss during switch-over events. The routing table update process is currently performed randomly independent of the traffic volumes or based on off-line / pre-configured prefix prioritization.

Experimentation	
Evaluation criteria and metrics	The main evaluation criteria will be packet loss during switch over.
Experimental scenario: description, tools, configuration and running conditions	**Experimental scenario description**: The experimental scenario will consist of three parts: 1. **Evaluating/optimizing the adaptive monitoring mechanism**: this part will consist in training the component such as to find the ideal monitoring interval. For this purpose, the component will be presented in a first phase traffic generated from standard traffic generators (iperf, D-ITG, etc.), and in a second phase the component will have to learn on captured traffic from operational networks. 2. **Evaluating the approximate sorting or traffic stream classification mechanism**: this part will evaluate several classification mechanisms such as to make an ordered set of traffic streams in terms of volume. The goal is to have the classification as fast as possible, experimenting with fewer classes, or approximate sorting algorithms. 3. **Evaluating the online quantum interval optimization**: this last part will involve using the given ordering or classification to find an optimal quantum interval. Probably this will be possible using a heuristic. However it should be tested and compared to random, constant and other online learned values with given input. **Experimental tools**: For traffic monitoring purposes a Click Router component will be developed. Evaluating different prefix-classification or prefix-ordering algorithms will require additional development in the cognitive engine. The router update process (quantum optimization) will in a first phase be modeled in simulation. If the simulation proves to be valuable, it will be implemented in XORP in a second phase. Development will need to happen in the Monitoring Engine (Click) and in the routing process of XORP. **Experimental configuration and running conditions**: Traffic monitoring functionality will be needed in every node in the Monitoring Engine (see ECODE architecture) such that passive measurements can be taken on a learned sampling interval and on a learned set of parameters.
Methodology	**Methodology to obtain experimental data**: A different methodology will be used for each part described in the scenario section: The methodology for adaptive monitoring (sub-case 1) will consist of gradually increasing the set of traffic streams presented to the monitoring component: both in terms of complexity and size. A traffic matrix will be generated for a set of networks, varying from well-meshed to poorly meshed. Experiments will run on a traffic matrix having different types of flows between the nodes: varying from uniform to strongly directed, from

	very well spread in time to very bursty, flows having increasing to decreasing volumes (using Linux-based traffic generators such as IPerf, D-ITG, and others). The node on which the sampling will happen will be placed at several locations in the network such as to present several types of traffic. In a second stage, traffic traces from a realistic network will be sent through the learning node (e.g., traces from the GEANT network). The changing rate of the ordering or classification of the volume of different traffic streams (next sub-case) will be used as a feedback such as to decide to monitor more or less frequently. For evaluating the classification or ordering component (sub-case 2), random sets of prefixes and associated numbers will be generated such as to provide the classification / ordering mechanism with data. Feedback will be provided by existing sorting algorithms. Optimizing the quantum interval (sub-case 3) will receive as input several classes of prefixes, having different sizes, and being either strongly or loosely ordered (depending on the output of sub-case 2). In a first phase, a fixed model of the quantum optimization process in simulation will be used to experiment with different quantum optimization strategies. In a second phase, the model can be implemented in emulation. The experiment will evaluate strategies using a fixed quantum (as is) and those with a variable quantum. The end-performance will be compared with random prefix ordering and fixed quantum interval. **Experimental data processing methodology and analysis**: The data analysis tools will mainly consist of custom made tools in Click to report on seen traffic patterns and trends. Reports (to be developed) will be generated from the router engine to be able to track why a certain choice of quantum interval and associated traffic streams was chosen.

5. Path Selection

5.1. Informed Path Selection

5.1.1. Use Case Scenario 1

Use Case Description	
Title	Intelligent path ranking using IDIPS
Technical objective	Technical objective T02. Active measurements are used to feed the Knowledge Base and also used by Cost Functions for ranking paths.
Participant(s)	Damien Saucez (UCL), Juan Pablo Narino Mendoza (UCL), Benoit Donnet (UCL), Olivier Bonaventure (UCL), Pierre Dupont (UCL), Guy Leduc (ULG).

Content	
Short description	The idea of the use case is to improve ISP-Driven Informed Path Selection (IDIPS) with machine learning techniques. IDIPS is a server that can be contacted by any host (applications or network stack) for ranking paths based on particular requirements, such as minimizing the delay or maximizing the bandwidth. For performing this ranking, the IDIPS server is supposed to collect path performance information, such as delay, jitter, packet loss rate, capacity/ available bandwidth or TCP throughput. Possibly, we can also imagine to determine path diversity using IDIPS. This data collection, as being made through active probing measurements, is network intrusive, especially when performed at a large-scale. In this use case, our objective is to see whether those path performance information can be predicted using machine learning techniques. At this point of the project, we believe that this is a time series problem. Thus, the goal is to reduce the impact of IDIPS on the network by avoiding sending multiple probes. In particular, bandwidth estimation is known to be very intrusive as it requires a lot of probes.
Expected results	We expect to observe a correct estimation of the path performance (we still have to define the word “correct”). The accuracy of the estimation will be evaluated during our experimentation (see the Experimentation sub-section). We also would like to characterize the metrics dynamic. From one hand, the metrics are time dependant and we want to determine how long a given prediction can be valid. On the other hand, some metrics would depend on the metric ends. For example, if the limiting factor for the metric is shared by all the destinations (e.g., the local uplink is the bottleneck), the metric will look independent of the destination. Otherwise, a correlation between the metric and the destination will exist.

Experimentation

Evaluation criteria and metrics	The following evaluation metrics will be considered: – accuracy of the prediction, i.e., how far is the prediction from – actual value of a given path performance information – computation cost of the prediction – time period during which the prediction can be considered as valid – probing savings, i.e., the amount of probes we avoid to inject in the network thanks to machine learning predictions.
Experimental scenario: description, tools, configuration and running conditions	**Experimental scenario description**: Off-line experimental scenarios based on collected data sets, some of them are publicly available while others still have to be built by our own. **Experimental tools**: An IDIPS server software release is already available but does not include any measurement tool or ML engine. Only the protocol stack is implemented and an interface to easily plug path ranking components. IDIPS is already interfaced with LISP (Locator/Idendifier Separation Protocol) [INM08] and would probably be interfaced with a P2P system in the next future. In order to influence the path performance on the testbeds, Dummynet and/or Click will be used. Metrics like delay, capacity/available bandwidth, packet loss ratio and others can be controlled with such tools. Traffic generator will be used on the testbed to generate traffic in a controlled way and thus validate the IDIPS components. **Experimental configuration and running conditions**: Experimentations will be in three steps. **Step 1**: The machine learning engine will be tested off line based on datasets (Off Line Validation). Some dataset will be homemade while others will come from well-known public datasets like RIPE and Routeviews. The homemade datasets will be passive (e.g., NetFlow, TCPdump) or active and will be collected by the partners. The need of homemade datasets comes from the fact that there are no public traces available that gives, at the same time and the same place, information about all the metrics we need. **Step 2** (Emulation validation): Validation of experimentations will rely on emulation based on traffic model extracted from the state of the art of network traffic modeling and the datasets. This step allows to validate IDIPS in a totally controlled environment. The emulation testbed will begin small with a few dozen nodes and progressively increase as much as the IBBT environment allows it. **Step 3** (Internet validation): IDIPS will be validated on the Internet as a final step. We still have to define how the testbed will be setup, e.g., IDIPS in a P2P or a web server. Like for emulation, the testbed will start with a few nodes and progressively increases in size. The size is still to be defined. **Experimental execution and measurements**: These experimental scenarios might be seen as “replays”. Using the previously collected data, we will inject them in the Cognitive Engine. Depending on how we will calibrate the system, it will use a set of previous path performance information (delay, jitter, packet loss rate, and bandwidth) to predict future value of this information.

Methodology	**Methodology to obtain experimental data**: Obtaining measurements data for performing the experiment is the most difficult part of the case. It is rather easy to collect data for the delay (the RIPE Test Traffic Measurement (TTM) Service data set looks like being a good candidate). See: http://www.ripe.net/projects/ttm/ Unfortunately, it is going to be much more difficult for the bandwidth. In particular if we want (and, at some point, we will want to) both bandwidth and delay at the same time for the same destination. Therefore, time series are of high interest for us. **Experimental data processing methodology and analysis**: Data processing and analysis will be two fold. First, every component will be evaluated alone as unitary tests. Once validated, the components will be ported to IDIPS and tested with emulated (Emulation Validation) traffic and real traffic (Internet Validation).

5.1.2. Use Case Scenario 2

Use case scenario description	
Title	Delay estimations, delay-based path selection and routing
Technical objective	Experimentation of Technical Objective 2 (B1: Path availability/IDIPS). By observing the behavior (precision, oscillations) of a Virtual Coordinate System deployed in every IDIPS server, Machine Learning is used to detect potential routing shortcuts, and thereby improve the diversity and ranking of paths of high quality.
Participant(s)	This scenario will be performed by ULG.

Content	
Short description	IDIPS servers run an Internet Coordinate System (ICS) and thereby measure some Round Trip Times (RTT) between them. These RTT measures are used by IDIPS servers to infer their coordinates, which in turn are used to estimate other unmeasured RTTs between them. We will evaluate whether observations of the dynamical behavior of this ICS (e.g. values and variations of delay estimation errors) allow IDIPS servers to find the best paths in terms of delays.
Expected results	We expect to show that the observations of the ICS (and the associated learned model) allow to detect with high confidence when there exists a routing shortcut between IDIPS servers, and then to narrow the search for relay servers leading to lower delay paths.

Experimentation	
Evaluation criteria and metrics	We will evaluate the success rate of the machine-learned models to detect the presence of shortcuts and to narrow the search for interesting indirect paths.
Experimental scenario: description, tools, configuration and running conditions	**Experimental scenario description**: The experimental scenario consists in deploying IDIPS servers that run an Internet Coordinate System (ICS).

	Realistic experimental scenario requires a large number of nodes representing IDIPS servers. For this purpose, at least a few hundreds should be considered but preferably one thousand would be required for the purposes of our experimental scenario. This can be seen as the deployment of a large peer-to-peer network. **Experimental tools**: An IDIPS server provides tools to measure path characteristics to lots of (possibly all) remote locators/prefixes. Beside IDIPS servers no dedicated tools are foreseen. The generated traffic will be modest, basically some ICMP echo request between nodes, each node pinging a few tens of neighbors. Henceforth, no dedicated traffic generator tool will be used. **Experimental configuration and running conditions**: IDIPS-equipped nodes should be geographically distributed, or some suitable delay emulators should be used between these nodes in case servers are physically co-located - the ILAB-T experimental facility allows emulation of configurable delay between nodes. Our experimental setup does not require to replay data traffic traces. The experimental setup does not require any special hardware, or specific tool. **Experimental execution and measurements**: We will measure RTTs by using active probing. The IDIPS servers themselves perform data collection.
Methodology	**Methodology to obtain experimental data**: The methodology relies on the deployment of nodes as a peer-to-peer system. Experiments will last a few hours. **Experimental data processing methodology and analysis**: From the raw delay measurements, each IDIPS-equipped node will extract several metrics, such as: – RTTs (min, mean, max, standard deviation, etc.) – Coordinates (precision, stability, drift, etc.) A Machine Learning module is used to detect presence of shortcuts (better paths in terms of delay between IDIPS servers). The quality of the paths found will be evaluated, including the false positive rate, and the delay gain (absolute and relative) of these indirect paths.

6. Accountability

6.1. Profiling and Accountability

This section describes the experimental plans and scenarios, including validation techniques and methodologies, for the technical objective (b3)-profile-based accountability. It is divided in three sections, where the first two sections describe the use case, objectives and expected results. The third section zooms in the experimentation, describing the scenarios, methodologies and evaluation criterions. It also includes the architecture of the profile-based accountability engine and the positioning toward other use cases.

Use case scenario description	
Title	Profile-based Accountability Experimental Framework
Technical objective	The objective of this use case is to define new ways to implement Fair Usage Policies (FUP). FUP regulates the way users are allowed to use network resources, which is in fact the binding contract subscribers have not only with their Internet Service Provider (ISP) but also, in a broader sense, to the collative networks that compose the Internet. At present, the mechanism in place for FUP is the maximum access speed, for example a typical DSL 12Mbps downstream with a 400kbps upstream capability. Depending on the country, some operators also cap the total volume consumed by the subscribers, such as 25 gigabytes per month. They penalize transgression by dropping the maximum access speed to, for example, 64 kbps for the remaining of the month. It is clear that these are very limiting capability for operators to control the fairness of use in their network. Technically, the goals of this use case is to define new ways to determine and impose network accountability and control, so that operators can have their network resources fairly allocated amongst their users. It is important to notice that in this context fairness does not imply in "equal allocation for everybody", but rather a respect the binding contract between the operator and the subscriber. Machine learning is expected to be used to define profiles (which will be associated to a FUP), and to monitor and control subscribers according to the assigned profiles.
Participant(s)	This scenario will be performed by Alcatel-Lucent Bell (ALB) and under the technical responsibility of Ing-Jyh Tsang (ALB) with the help of Werner Van Leekwijck (ALB), Wouter Rogiest (ALB), and Koen De Schepper (ALB). Results of the experimental use case (a1), referring to adaptive traffic sampling will directly contribute to network characterization for profile-based accountability. Chadi Barakat (INRIA), Amir Krifa (INRIA), Imed Lassoued (INRIA), Giovanni Neglia (INRIA). In addition, machine learning technical objectives will directly contribute to the experimental scenarios. Thus, we foresee additional cooperation with the experts in machine learning amongst the ECODE partners. Pierre Geurts (ULG), Louis Wehenkel (ULG), Juan Pablo Narino Mendoza (UCL), and Pierre Dupont (UCL).

Content	
Short description	Profile-based accountability requires a proper definition of who is accountable and for what, which were introduced in D2.1-"Cognitive Engine - Experimental Network and System Architecture". Hence, subscribers are accountable for their usage and on the available network resources. Profiles

	should characterize a traffic pattern/behavior, correlating the end-user conduct and the effects of such traffic pattern in the network. Consequently, profiles should reveal certain characteristics of the subscribers' usage, interaction, and impact on the network. There are two main tasks required by the system, first to define the profiles and second to classify subscribers according to profiles. The inputs for the system consist of measurements from the subscribers' network traffic flow [1, 2, 3]. The measurements will be sampled from data gathered during time-sliced intervals. In addition, elements of the data can be divided in two parts: one that conveys the information of the user network demand, and a second that relates information from the state of the network load. This opens the possibility of including measurements from different network elements to be associated with the subscribers' traffic flow. For both profile definition and profile classification, the initial step, requires data sampling and pre-processing, which consists in getting raw data and perform feature selection and extraction. The raw data are packet traces from a network element, which can be obtained and processed in real-time or in batch-mode. The dataset will be used to extract measurements, and constitute the input data for the machine-learning algorithm. This input data can be mathematically defined as a set of vectors with d-elements, each representing a feature measured from the raw data, e.g. $V = (v_1, v_2, \cdots, v_d)$. For the profile definition task, a learning dataset will be used for training the machine-learning algorithm. The main objective is to define profiles that characterize subscribers, however there are different ways to achieve this. For instance, the input data can be used to feed a machine-learning algorithm, which is designed to categorize subscribers in different profiles directly from the input vectors. Alternatively, a two-step approach can be taken and the input data can be used to characterize subscribers' *action* or their network behavior during sliced intervals. The results from this classification can be used to feed another learning system to create the final subscriber profile classes. For the classification problem, a test dataset will be used to validate the learned system. Depending on the machine-learning algorithm used for the learning phase, the input vectors will classify a subscriber as belonging to one of the possible profiles derived from the learning phase.
Expected results	It is expected that machine-learning techniques will produce a system capable of imposing network accountability based on profiles, whereas the definition of profiles was given in the previous sections. The ultimate goal is to achieve a system that controls the network resource allocation amongst the users of the Internet in a fairer way. Again, the term fair does not intent to mean equal distribution for all the users, but rather maximization of the use of the network resource, in respect to the contract in which subscribers have with their operator. It is expected that based on the network traffic and machine-learning techniques, a learning system will define profiles. Those profiles will enable the differentiation of subscribers in respect to classes of network resource consumption. Finally, a system capable of monitoring and controlling subscribers' interaction and network demand will be created to impose network accountability.

Experimentation	
Evaluation criteria and metrics	The evaluation criteria and metrics used will depend on the nature of the experimental case, i.e. batch-mode or real-time. In addition, the type of machine-learning technique will also play a role. There are many techniques in consideration for cross-validation, such as the k- fold cross-validation or

	bootstrap technique. Moreover, different metrics such as Euclidian or Mahalanobis might be applied, in case clustering techniques are used for profile classification [4,5,6]. Note: at this point, the specific techniques and metrics to evaluate the system are not closed defined. A formal definition will be included with the first results of the experimentations.
Experimental scenario: description, tools, configuration and running conditions	**Experimental scenario description**: The experimental scenario is divided in two phases one based on an off-line, centralized, batch-mode machine-learning algorithm and the other using distributed, real-time machine-learning algorithms. This division reflects the challenges and a stepwise approach towards the resolution of the objectives laid out by the profile-based accountability problem. **Scenario 1**: This is an initial scenario to allow experimentation on solutions for the challenger posed by profile-based accountability. Using an off-line, centralized, batch-mode system, will allow us to focus on the task of defining and classifying profiles. This will be based on traffic traces, gathered from a network element during a specific time interval. For example, a one-hour traffic trace from an aggregated node. **Scenario 2**: This scenario moves the experimental objectives closer to the application on a real network. The assumption is that the first phase will solve issues related to understanding the nature of the data, adaptive sampling, feature extraction, and network traffic patterns. The focus in this phase will be on perfecting the distributed, real-time machine-learning algorithms as a solution for the profile-based accountability task. **Experimental tools**: The following hardware, software and network system will be used for these scenarios. Network (1 L3 Gigabit Ethernet Switch): – 24 triple speed ports – 20 RJ-45 connectors configurable to 10/100/1000 Base-T – 4 MiniGBIC combo ports (port 21 - 24) – 2 10Gig stacking ports – 2 build-in 10 Gig ports – Capacity: 64Gbps Full Duplex or 128Gbps – Throughput: 65.5 Mbps Hardware (4 Rack Servers 2U): – 2 x Quad-core Intel Xeon E5430 2.6 GH with 12 MB cache – 32 GB DDR2-667 registered ECC (16 DIMMs) – 4 x Seagate 300 GB SAS drive with 15,000 rpm – 4 x RJ45, PCI-e Intel® PRO/1000 PT Ethernet Server Adapter Software (Open Sources): – Linux CentOS – Gnu C/C++ compiler **Experimental configuration and running conditions:** **Scenario 1**: Figure 1 depicts different network elements, and position which data traffic could be gathered. The figure also indicates some particularity of the different location, for example traffic through the edge server loses the intra-autonomous system user-to-user flows. This scenario assumes that a raw dataset is collected from an aggregate node. In an off-line mode, individual subscribers' traffic is discerned, data feature are extracted, and

pre-processed. The dataset is then split into learning and test data. This experimental scenario looks to evaluate several different techniques for data filtering, feature extraction and data reduction, together with different machine-learning techniques [9]. It is important to note that machine-learning algorithms used in this phase many not be easily used in the next scenario. The challenges of mapping a centralized, batch-mode machine learning algorithm to a distributed, real-time environment is no trivial and constitutes a well-known challenge in the machine-learning community [7,8]. However, data filtering, feature extract and data reduction techniques used in this phase will be very useful for the next phase. In this way, an efficient system can be better achieved in the next experimental scenario.

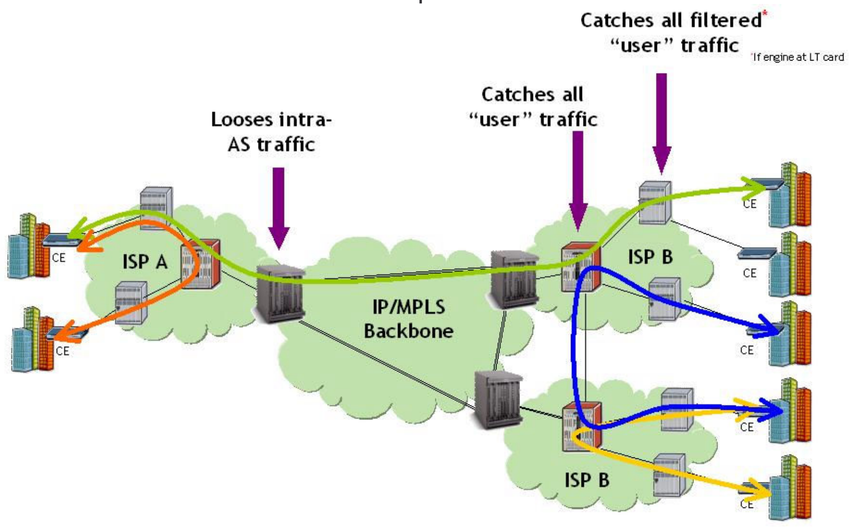


Figure 1 Possible network elements where data can be extracted for experimentation.

The computational resources required for this experimental scenario is limited. It is assumed that the raw data will come from an aggregated node from a real network. Once the raw data is available, a single server will be sufficient to perform the different steps required by this experimentation.

Scenario 2: The PBA-engine will operate in two levels, a learning domain and a classification domain, both operating concurrently and interacting with each other. While the learning stage will be responsible to keep the profile definition up to date with respect to the evolving subscribers network traffic behavior, the classification stage produces a real-time monitoring system, cross-checking and classifying in which profile each is operating and being precise enough to give a measurement on the profile deviation.

Figure 2 shows an ideal network configuration including the profile-based accountability (PBA)-engine, which is in the access node. The access node is the first element inside the operators' network where subscribers' traffic could be monitored. If a PBA-engine resides in the line card, no subscribers' traffic filtering is necessary. Furthermore, scalability issues in term of computational resources have only limited impact, because a limited numbers of subscribers are connected to each access node.

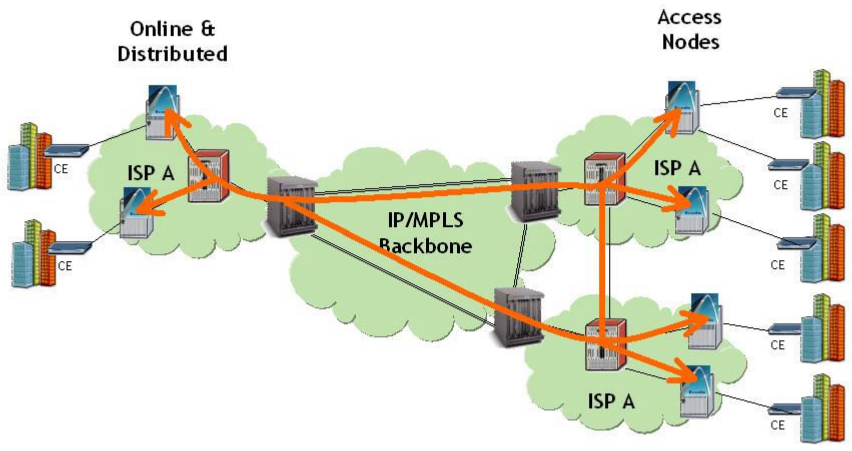


Figure 2 Diagram of an ideal configuration where PBA-engine resides in access node.

This experimental scenario will use four servers to simulate a real network environment. It is assumed that the raw data will come from different aggregated node from a real network. Each node will host the distributed PBA-engine and contain separate learning and test datasets.

Experimental execution and measurements:

In both scenarios, the raw data set is collected and traffic is pre-processed and data feature extracted. A representative data set is then split into learning and test data. Several different data sets can be produced depending on the cross-validation technique used [7,8]. A machine-learning algorithm is applied on the learning i.e. training data, and a test dataset is used at classification stage to evaluate the performance of the system. An overall evaluation of the system has to be engineered, so that we will be able to scientifically assert and verify that the profile-based accountability goals are met.

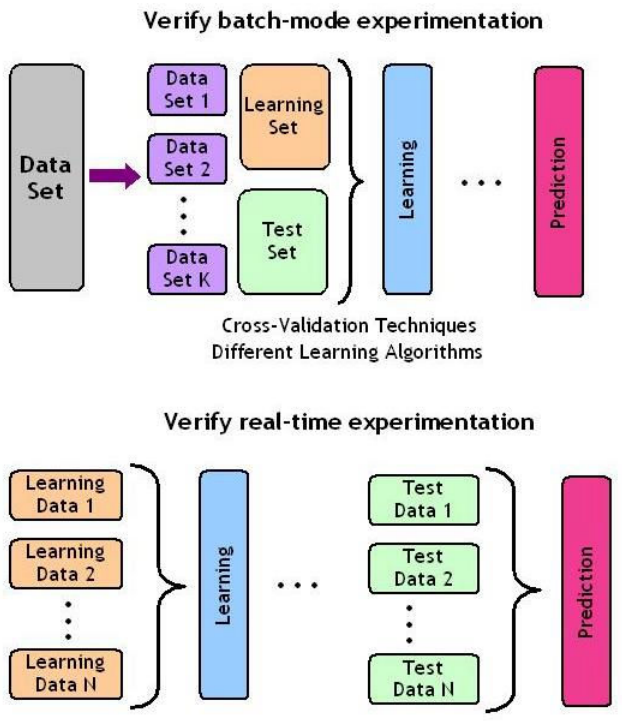
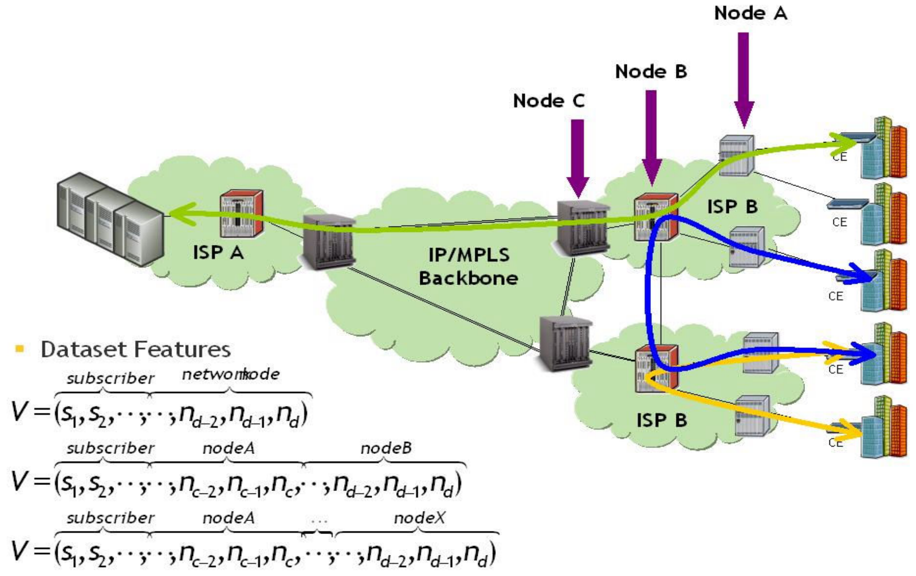


Figure 3 Dataset split for the cross-validation schemes.

Methodology	**Methodology to obtain experimental data**: The experimental dataset can be extracted from an aggregated node from a real life network. This can be obtained directed from networks that ECODE consortium has access or through known databases such as the DatCat^{SM}. The initial experimentation will be done as an off-line experiment, with the possibility of simulation of a real-time environment. In case of direct access to a real life network, where issues such as of the requirement of anonymization are solved, the data will be directly extracted from the network, using techniques from case a1, and processed in real-time. **Experimental data processing methodology and analysis**: The initial data collection block, represent a very important phase, which includes data processing and analysis. The objective of this step is to perform feature selection and extraction from the raw dataset. This will lead to a uniform feature set that can be used for the machine-learning phase. The elements of the feature vectors can be divided in two parts, one that conveys information of the user network demand and another that relates information from the state of the network load. This opens the possibility of including measurements from different network elements to be associated with subscribers' traffic flow (see Figure 4). Example of measurable are: number of flows, packet size and packet inter-arrival time statistics (average, deviation, minimum, maximum, quartiles), byte count, connection duration, the aggregated number of flows, aggregated packet size and inter-arrival time statistic at the network element, etc. Feature vectors are defined as $V=(v_1,v_2,\cdots,v_d)$, where each component v_c determines a measurement referent to some properties calculated during a certain time-slice from a subscriber. The elements of the vector v_c can be related to the subscribers' traffic demand or it can capture information over the resource allocation at the network side.  Node A Node B Node C ISP A IP/MPLS Backbone ISP B CE ISP B Dataset Features $V=(\overbrace{s_1,s_2,\cdots}^{subscriber},\overbrace{\cdots,n_{d-2},n_{d-1},n_d}^{networknode})$ $V=(\overbrace{s_1,s_2,\cdots}^{subscriber},\overbrace{\cdots,n_{c-2},n_{c-1},n_c}^{nodeA},\overbrace{\cdots,n_{d-2},n_{d-1},n_d}^{nodeB})$ $V=(\overbrace{s_1,s_2,\cdots}^{subscriber},\overbrace{\cdots,n_{c-2},n_{c-1},n_c}^{nodeA},\overbrace{\cdots}^{\ldots},\overbrace{\cdots,n_{d-2},n_{d-1},n_d}^{nodeX})$ Figure 4 Diagram depicting a subscriber traffic flow (green arrow) and the network elements the traffic flow passes. The dataset represented by vectors V with possibility of contain features from subscriber and different network nodes Figure 4 shows a network diagram, which will help in understanding the definition of the feature vectors. Assuming that a subscriber is served by the access node A and have traffic flow from the green arrow. A feature vector

	$V = (s_1, s_2, \cdots, n_{d-2}, n_{d-1}, n_d)$ will be composed of elements describing the subscribers' traffic and the features characterizing the load on the network node. This definition can be extended to add features characterizing network resource allocation from other network nodes that is part of traffic flows. Thus, $V = (s_1, s_2, \cdots, n_{c-2}, n_{c-1}, n_c \cdots, n_{d-2}, n_{d-1}, n_d)$, where the s elements represent features from the subscriber side, and the n elements from network nodes, such as node A, B and C in Figure 4. A data set containing a representative number of V will used as the input of an unsupervised machine-learning algorithm. The results would define the number of profiles (clusters) according to a certain distance measure. Alternatively, if some small set of V could be labeled such as belonging to a certain profile (defined a priori), a semi-supervised or supervised algorithm can be used to learn the profiles from the data set.

7. Observation and Analysis Grid

This section provides an analysis of the experimental tools that will be used in each experiment and methodology that will be conducted for each experiment in order to facilitate interpretation of the obtained results and enable systematic re-production of results.

8.

9. Conclusion

Conclusions will be provided in the next release of this document.

10. Annex.1: Template

Use case scenario description	
Title	A meaningful title for the use case experimentation
Technical objective	Which technical objective will be experimented in this scenario ? Which performance improvement is expected by the use of machine learning ?
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 evaluate in more detail than the title and less detail than the remainder (experimental scenario)
Expected result(s)	What do you expect to find as results ? Why would this be the case ?

Experimentation	
Evaluation criteria and metrics	Describe the experimental evaluation criteria, and metrics.
Experimental scenario: description, tools, configuration and running conditions	**Experimental scenario description**: Provide as precisely as possible the experimental scenario to be emulated. For example: What is the topology of the emulated scenario ? Indicate the number of PCs/hosts, network nodes and which configuration is necessary for running this experiment (OS, emulation tool, experiment tools, etc.). Indicate the interconnection schema. Give all characteristics of links and network devices in terms of delays, loss rates: indicate any model. **Experimental tools**: Do you need specific measurement tools, hardware, equipments ? Are these tools available without additional development or use of these tools require specific development for the purpose of this experiment ? **Experimental configuration and running conditions:** What are the background traffic and events to be emulated during the experiment ? What are their characteristics ? Is there a model to describe them and a tool for generating them ? Are there traces to replay ? Where to put monitoring, measurement or traffic capture tools/ equipments? **Experimental execution and measurements:** What are the requirements in terms of measurement ? What kind of measurements (active, passive, sampled, etc.)? What parameters to measure/estimate ?
Methodology	**Methodology to obtain experimental data:** Flowchart of the experimental scenario with the functional blocks indicated. Provide an indicative description of the experimentation running time (per

	step if possible). **Experimental data processing methodology and analysis**: Describe here the methodology for processing and comparing the experimental data (e.g. against reference scenario). 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.

11.

12. Annex.2: ILab-T Experimental Facility

The iLAB virtual wall, experimental facility at IBBT, will be available to the project experimenters to execute the tasks associated to the experimental phases 1 and 2. This experimental emulation facility consists out of 100 servers connected through a non-blocking Ethernet switch. The 100 high-end servers composing the experimental facility cluster come each with dual CPU dual core 2.0GHz, 4GB RAM, 4 or 6 Ethernet network interface cards (Gb/s) – PCI express, and 4x80GB disks – software RAID 0 configuration. The servers are interconnected by a Force10 E1200 non-blocking Ethernet switch (1.68 Tb/s – 1Billion pps) that scales up to 672 ports. Each server is connected with 4 or 6 gigabit Ethernet links to the switch.

Remote access to the iLAB control system is possible without any restriction on functionality. As such, large dedicated experimental setup can be built very fast using the Emulab control software (see Section 1.3.2.2). Advantages are the repeatability and the possibility of dedicated experiments within confined environment where experimental parameters can be controlled. Disadvantages are the complete isolation and lack of real Internet connectivity for the experiments.

This experimental facility will be used as follows (however other scenarios are also possible and will be developed during the project time period):

- Large number of nodes for use in control plane experiments: e.g. after a simulation step, real code could be tested in a control plane emulation. Some of the nodes can even be connected to real network elements;
- Each node is fully dedicated and under full control of one experiment, so the users can test all kind of operating system images, configurations. As such, the good properties of Emulab (dedication and control) and PlanetLab/OneLab (distribution) are merged;
- Some of the nodes may be used for network emulation (bandwidth, delay, packet loss) in order to make the experiments with data traffic between servers and clients realistic;
- Moreover, it is also possible to connect special hardware (storage networks and storage clusters, traffic generators, real network elements, etc.) to the emulation clusters. Therefore, Gb Ethernet or even 10Gb Ethernet ports can be used on the force 10 switch. iLAB has also a good connectivity (1Gbps) to BELNET, the Belgian NREN, with both IPv4 and IPv6 connectivity.

The main advantage of such an experimental facility within this project is the relatively large experimental setup (100 nodes, but even more if virtualization is considered), which can be used in a dedicated mode. As such, the experimenters have full control of dedicated nodes and the performance experiments can be performed in a repeatable way.

13. Annex.3: Experimental Tools

This Annex describes experimental tools outlined in the main sections of this document:

1. XORP (eXtensible Open Router Platform)

XORP (http://www.xorp.org) is the industry's only extensible open source routing platform. It is in broad use worldwide, with thousands of downloads by companies and educational institutions and an active international developer community. Designed from the start for extensibility, XORP provides a fully featured platform that implement IPv4 and IPv6 routing protocols (such as BGP and OSPFv2) and a unified platform to configure them. It is the only open source platform to offer integrated multicast capability (such as PIM-SM and MLD). XORP's modular architecture allows rapid introduction of new protocols, features and functionality, including support for custom hardware and software forwarding. XORP is available for free download under a BSD-style license.

The ECODE project will use XORP as the experimental routing platform for three main reasons. First, XORP is a high-quality and recent implementation of the main routing protocols that combined with Click or operating systems is used to build real routers. Second, several of the partners are experienced with development on XORP. Third, in contrast with Quagga/Zebra or other commercial router platforms, XORP is distributed under a BSD-style license. This implies that both research labs can easily obtain and modify its code but also that industry will be able to build products more easily from this code base than from a code based distributed under a GPL license.

2. Emulab

The University of Utah has developed Emulab (www.emulab.net), which is control software for a very flexible emulation experimental facility based on servers, which can run a variety of operating system images and can be interconnected dynamically in topologies as needed for the experiments. In this setup, network links (bandwidth, delay) can be emulated through software. This software has been developed for controlling local PC clusters. The software makes it possible to do automatic and remote configuration, by swapping in and out operating system images. As such, large dedicated experimental facility can be built very fast.

The Emulab software is used to control the iLAB virtual wall experimental facility. It makes possible to fully control the servers and the interconnecting network. As such, it installs the desired operating system image on each server and configures the Ethernet switch with the right VLAN configuration. The Emulab software itself cannot perform experiments on its own. It also incorporates multiple users, a web interface and an archive for experiment setups. However, the users still need to define and execute their experiments once the iLAB virtual wall is configured and the user has access.

The Emulab software provides the following features to the user (reasons for using it in the context of the ECODE project):

- Web-based interface with dedicated user accounts;
- The user can graphically draw a testbed with nodes and interconnections (or provide this through an NS2 configuration file);
- The user can select an OS image (Linux, BSD, Microsoft Windows) for each node;
- After the user confirms: all nodes are installed and the necessary interconnections are made;
- After this, the user has full root access to the nodes for executing the experiments;

- Afterwards, the testbed is swapped out again, but available for future experiments with the same setup. As such, the nodes can be reused by another user;
- Nodes are installed in parallel in a very fast way (couple of minutes).

3. Measurement equipments to set-up on the experimental emulation platform

Measuring and monitoring network traffic and performance is a key objective of ECODE for assessing the cognitive router. Several measurement and monitoring equipments and methodologies exist. They can range from technical solutions based on dedicated hardware to generic software tools.

In terms of accuracy, using dedicated devices for tapping the link (to get traffic traces) and measuring performances online is certainly the best solution, despite it is also the most expensive solution. The most famous and commonly used monitoring device is the DAG card from ENDACE located in Waikato, New Zealand. This card aims at transparently tapping the link for capturing packets or packet headers, adding a very accurate timestamp for each of them, and either storing it on a large hard drive for future analysis, or applying a particular computing for analyzing the traffic online. The transparency is reached thanks to a well-provisioned machine with fast memory, fast busses, fast and large hard drives. Clock accuracy is enforced by using a GPS system and its PPS (Pulses Per Second) signal, which makes possible to avoid clock drift and to reach a perfect synchronization of the clocks of all monitoring equipments. Several other capture cards exist, but the DAG card is dominating the market, and as such, benefits from a large set of analysis and conversion tools.

On the other side, software tools for network measurements and monitoring exist. Their advantages are their genericity and their low price. On the other side, they lack from accuracy, and are not able to perform as well as DAG cards on high-speed links. Most known and used tools are based on the libpcap library (as Ethereal or Wireshark, for instance). Such inexpensive solution can fit technical measurement requirements for traffic having throughputs less than few hundreds of megabits per second, in case the tapping machine is powerful enough and equipped with a well performing network adapter. The synchronization using Network Time Protocol (NTP) can also fit when running on LANs. In addition, the library of existing tools is quite large. Obviously, such software solution does not integrate a bypass system and is thus less transparent than a DAG system. On a platform like ILAB, shared between many users, this can be damageable by introducing some performances decreases when experiments use monitored links.

14. References

[1] Lili Yang; Michailidis, G., "Sampled Based Estimation of Network Traffic Flow Characteristics," *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE* , vol., no., pp.1775-1783, 6-12 May 2007

[2] Crotti, M.; Gringoli, F.; Pelosato, P.; Salgarelli, L., "A statistical approach to IP-level classification of network traffic," *Communications, 2006. ICC '06. IEEE International Conference on* , vol.1, no., pp.170-176, June 2006T.

[3] Sung-Don Joo, Chae-Woo Lee and Yeon Hwa Chung , " Analysis and Modeling of Traffic from Residential High Speed Internet Subscribers LNCS 3090, pp. 410-419, Springer Berlin / Heidelberg 2004.

[4] Xiang, S., Nie, F., and Zhang, C. 2008. "Learning a Mahalanobis distance metric for data clustering and classification". *Pattern Recogn.* 41, 12 (Dec. 2008), 3600-3612.

[5] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit Dhillon. Information-theoretic metric learning. *Proc. 24th International Conference on Machine Learning*, 2007.

[6] Liu Yang, Distance Metric Learning: A Comprehensive Survey, Tech Report, Carnegie Mellon University. May 2006.

[7] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney, "Integrating Constraints and Metric Learning in Semi-Supervised Clustering", in Proc. of ICML-2004, pp. 81-88, Banff, Canada, July 2004.

[8] Christopher M. Bishop. Pattern Recognition and Machine Learning, Springer, 2006

[9] Ethem Alpaydin. Introduction to Machine Learning, MIT press, 2004.

[10] Huan Liu, Hiroshi Motoda. Computational Methods of Feature Selection, CRC Press, 2008