Seventh FRAMEWORK PROGRAMME
FP7-ICT-2009-5 - ICT-2009-1.6
Future Internet experimentally-driven research

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Deliverable D4.1
"Performance objectives, evaluation criteria and metrics"

Project description
Project acronym: **EULER** Project full title: **Experimental UpdateLess Evolutive Routing** Grant Agreement no.: **258307**
Document Properties
Number: FP7-ICT-2009-5-1.6-258307-D4.1 Title: Performance Objectives, Evaluation Criteria, and Metrics Responsible: D.Papadimitriou (Alcatel-Lucent Bell) Editor(s): D.Papadimitriou (Alcatal-Lucent Bell) Reviewer(s): S.Sahhaf (IBBT) Dissemination level: Public (PU) **Version: 1.0 - Date of preparation: April 30, 2011**

D4.1 - Performance objectives, evaluation, criteria and metrics

Executive Summary

In the context of the EULER project, several routing models and algorithms applicable to the Internet will be developed and designed. The range of routing paradigms that the project will investigate is relatively large; it covers a spectrum ranging from dynamic compact routing to greedy routing and its variants, e.g., updateful and updateless. Further, by relying on the results of investigations performed in WP3, these paradigms are expected to take benefit of the statistical and structural properties of the Internet topology and better characterization of its dynamics. Ultimately, the resulting routing paradigms would lead to distributed routing schemes that are specialized for the Internet while taking into account its dynamics and its continuous evolution. The aim of this document is to provide a common set of functional and performance objectives that each routing model and algorithm developed in the context of the EULER project shall meet. Next this document details the performance criteria to determine when a given routing scheme actually meets its objectives. It also details the performance metrics that will have to be measured either by simulation and/or by emulation in order to set the requirements on the corresponding measurement and scenario execution tools. Next, this document develops the experimental evaluation criteria and metrics will be exploited to determine and measure if the emulated routing protocol components that will be developed comply with both functional and performance objectives. Note that in order to consider such performance evaluation by emulation as scientifically valid, the experimental methodology shall lead to verifiable, reliable, repeatable and reproducible experimental results. Moreover, in order to perform verifiable and reliable experimental comparisons between the functionality and the performance offered by the different routing protocol components that will be emulated, we propose a framework to perform experimental comparison of routing protocols. Finally, this document concludes by detailing the requirements on the various tools that will be required to conduct the different validation, and verification phases of the routing schemes that will be developed in the context of the EULER project as well as the evaluation and analysis of their performance.

List of authors

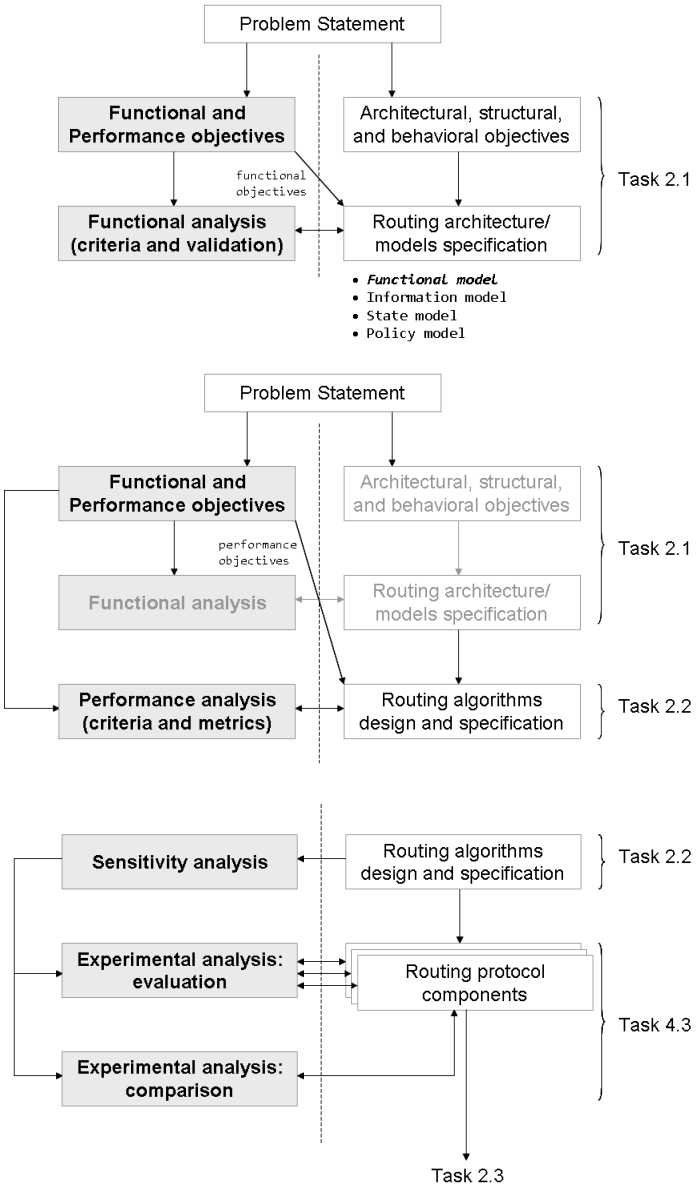
Affiliation	Author
Alcatel-Lucent Bell	D.Papadimitriou
IBBT	W.Tavernier, P.Audenaert
INRIA	N.Hanusse, A.Lancin
RACTI	I.Caragiannis
UPC	D.Careglio
UPMC	F.Tarissan

Table of Contents

Structure of the Document

This document is organized in three main sections. The first section (Section 2) details the objectives that the EULER project will realize in terms of functional evaluation and analysis. The second section (Section 3) considers the objectives in terms of the performance evaluation and analysis. The last section (Section 4), the objectives are explained in relation to experimental analysis (evaluation and comparison). Each of these parts details four elements: objectives, criteria, metrics, and comparison (when applicable). The performance evaluation and analysis will be conducted as part of task T2.2 (by means of formal verification methods) and as part of task T3.3 (by means of simulation using the simulation tools that task T4.2 will develop). The experimental evaluation and analysis will be conducted as part of task T4.3 by means of routing protocol component emulation and the measurement tools that task T4.2 will develop.

The relationships between these tasks and the specification of the routing system architecture, the routing model(s), and the design of the corresponding algorithms (as part of the task T2.2 of WP2) are depicted in Fig.1-3.



Terminology

The following terms are used throughout this document:

Architecture: the architecture of a distributed computational system is formally described by a set of function, objects/information, and state together with their behavior, structure, composition, relationships, and spatio-temporal distribution which characterize its domain of applicability. The specification of the associated functional, object/informational, and state models leads to an architectural model comprising a set of components (procedures, data structures, state machines, etc.), the description of their respective behavior and structure as well as the characterization of their interactions (messages, calls, etc.) describing their connection(s)/relation(s).

(Design) Principles: suggests normative rules/guidelines on how a designer/an architect can best structure the various architectural components. Design principles also describe the fundamental and time invariant laws underlying the working of an engineered artifact.

Pattern or scheme: refers to a set of models that are based on the same "principles" and thereby sharing the same essential and global characteristics as well as structuring/cohesive elements.

Model: a model is formally defined as a systematic and logical description of complex system by means of a simplified abstract representation. In the present context the routing model comprises the procedural model together with its associated data model, state machine model, and the data communication model which characterizes the interactions and interfaces (i.e. messages, calls, events, etc.) between the elements of the model (i.e. procedures, data structures, state machines).

Function: characteristic action (verb) a system performs to transform available inputs to the desired outputs.

Procedure: method to perform a given task/action specified as a sequence of discrete steps (each step being a finite instruction or operation, finite meaning represented by a finite number of symbols) which have to be executed in a regular definite sequence in order to always obtain the same result under the same conditions. Note that in the routing system modeling context, procedures are devised as sequences of operations that the routing function is conjectured to perform as it processes routing information.

Algorithm: consists of sequences of steps (operations, instructions, statements) for transforming inputs (pre-conditions) to outputs (post-conditions). An algorithm provides the description of the effective computational procedures composing the procedural model. The pre-conditions (of the algorithm) provide the description of the inputs, including their types as well as any relationships or properties that they must satisfy before execution. The post-conditions (of the algorithm) provide the description of the outputs, including all relationships and properties that must be satisfied (after execution).

Protocol: a set of procedures (together with their associated data structures and state machines) and message/data format.

Functional analysis: In the early design phase, functional analysis refers to the systematic process of identifying, describing, and relating the

functions a system must perform (thus the functions that need to be included in the system) in order to meet its objectives. Functional analysis does not address how these functions will be performed. Functional analysis deals with i) the top-level functions that need to be performed by the system (identification), ii) where these functions need to be performed (distribution/space); iii) how often they need to be performed (time); iv) under what operational concept and environmental conditions. In the later design phase, functional analysis proceeds to lower levels of the system decomposition by defining the system functional design, its inputs/outputs, and its various interfaces. It then refers to a methodology part of the design process that systematically describes the automated processing that a complex system must perform to transform available inputs to the desired outputs.

Performance analysis: consists in measuring (by means of well specified metrics), evaluating, and understanding system performance to decide (based on well defined criteria) if designed system meets its performance objectives.

Sensitivity analysis: attempts to identify how responsive the results of an experimental model are to changes in its parameters: this is an important tool for achieving confidence in experimentation and making its results credible. The general goal of Sensitivity Analysis is to characterize, qualitatively or quantitatively, what impact on a system a particular variable will have if it differs from what was previously assumed. In other words, by using Sensitivity Analysis, the analyst/the modeler can determine how changes in one or several parameters will impact the target variable.

Scientific validity: includes verifiability, reliability, repeatability and reproducibility. See definitions here below.

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

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

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

Reproducibility: an experiment is reproducible when the same experiment can be reproduced in different experimental setups, e.g., different platforms, different experimental facilities. Typically, reproducibility comes into play when a third party performs the same experiment in order to verify scientific validity of the outcome and results of the experimental scenario.

1. Introduction

In the context of the EULER project, several routing models and algorithms applicable to the Internet will be developed and designed. The range of routing paradigms that the project will investigate is relatively large; it covers a spectrum ranging from dynamic compact routing to greedy routing and its variants, e.g., updateful and updateless. Further, by relying on the results of investigations performed in WP3, these paradigms are expected to take benefit of the statistical and structural properties of the Internet topology and better characterization of its dynamics. Ultimately, the resulting routing paradigms would lead to distributed routing schemes that are specialized for the Internet while taking into account its dynamics and its continuous evolution.

The aim of this document is to provide a common set of functional and performance objectives that each routing model and algorithm developed in the context of the EULER project shall meet. Indeed, over time sufficient experience has been acquired to devise the functionality any routing scheme should deliver and the performance objectives it is required to meet. Next this document details the performance criteria to determine when a given routing scheme actually meets its objectives. It also details the performance metrics that will have to be measured either by simulation and/or by emulation in order to set the requirements on the corresponding measurement tools. Indeed, simulation and emulation experiments serve different purposes but are also subject to different constraints. For instance, executing emulated routing protocol components on experimental facilities requires accounting for the physical limits of the facility in terms of, e.g., number of nodes, processing/CPU and memory available at each node, and number of links. Experimental evaluation criteria and metrics will be used to determine and measure if the emulated routing protocol components that will be developed comply with both functional and performance objectives. These experimental evaluation criteria and metrics drive an important part of the experimental work that will be conducted as part of this experimental project. Moreover, in order to perform fair and reliable experimental comparisons between the functionality and performance offered by the different routing protocol components that will be emulated, a section of this document is dedicated to the experimental comparison of different routing protocols.

Finally, we conclude this deliverable by documenting the requirements on the various tools that will be required to conduct the different validation, and verification phases of the routing schemes that will be developed in the context of the EULER project as well as the evaluation and analysis of their performance.

2. Functional Evaluation and Analysis

In the early design phase, functional analysis refers to the systematic process of identifying, describing, and relating the functions a system must perform (thus the functions that need to be included in the system) in order to meet its objectives. It does not address how these functions will be performed. Functional analysis deals with i) the top-level functions that need to be performed by the system (identification), ii) where these functions need to be performed (distribution/space); iii) how often they need to be performed (time); iv) under what operational concept and environmental conditions.

The basic idea of functional analysis is that the system is viewed as computing a function (or, more generally, as solving an information processing problem). Following the definition provided by [Buede2000], a function is a transformation process that changes inputs into outputs. The system itself is modeled as a single, top-level function that can be decomposed into a hierarchy of subfunctions. The top-level function is partitioned into a set of subfunctions that use the same inputs and produce the same outputs as the top-level function. Functional analysis assumes that processing can be explained by iteratively decomposing the top-level complex function into a set of simpler functions (subfunctions) that are computed by an organized sub-system. Each of these subfunctions can then be partitioned further. The decomposition process continues until atomic functions are attained. Atomic functions are functions that by definition can not be further decomposed. The expectation is that when this type of decomposition is performed, the subfunctions taken individually will be simpler than the original functions.

Classically, there are four elements to be addressed by functional analysis approach. The first one, the hierarchical decomposition of the top-level function into sub-functions enables to identify the functions that need to be performed by the routing system.

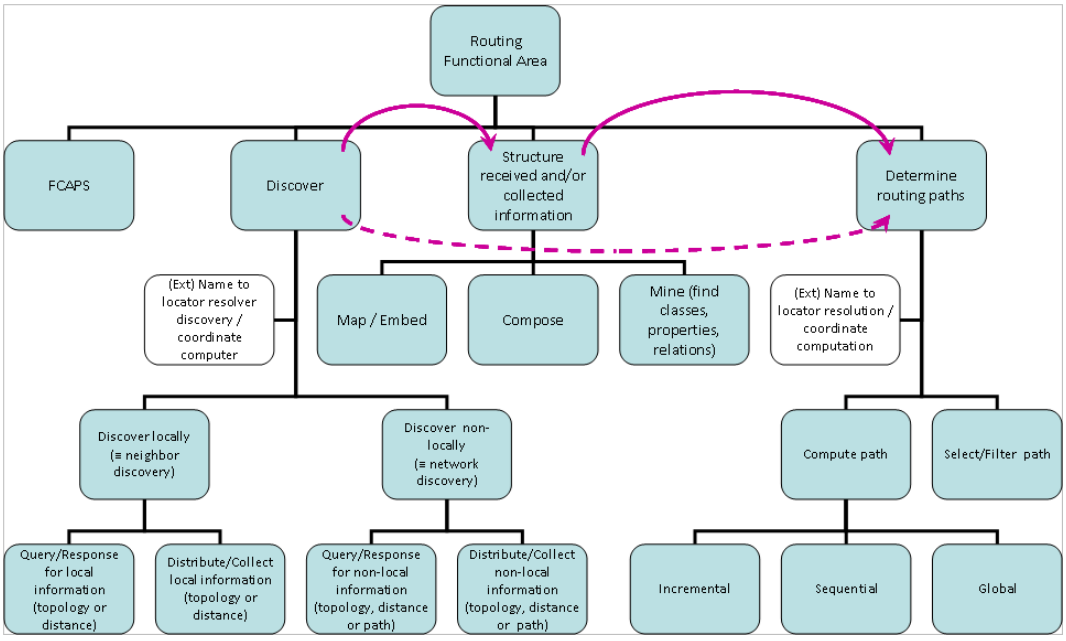


Fig.4: Hierarchical decomposition of the routing functional area

2.1 Routing functionality

Routing functionality is the combination of all functions that transform topology and/or routing information into routing paths that populates local routing tables. In turn, these entries steer the forwarding functionality. Typically, these functions are combined into a single component, referred to as the routing engine that comprises the following functions:

The FCAPS acronym denotes the set of routing system management functions including Configuration management, Accounting/Administration, Performance management, and Security management. Being mainly routing scheme specific their usage will be considered on per-protocol basis.

Discovery:

- Local discovery (also referred to as neighbor discovery): functionality enabling the acquisition/dissemination of knowledge about the local environment (neighborhood) to local entities including local and remote interfaces (and their properties), incident links (and their properties), and nodes adjacent to incident links (and their properties).
- Remote discovery (also referred to as network discovery): functionality enabling the acquisition/dissemination of knowledge about the non-local environment from/to remote entities including remote links/nodes, paths and/or distances to reachable destinations.

Structuring of topology information units and/or routing information units:

- Embedding: given metric spaces (X,d_X) and (Y,d_Y), where d_X and d_Y are distance function, a mapping function $f: X \rightarrow Y$, $x \rightarrow y=f(x)$ is called an embedding. An embedding is called distance-preserving if for all $x, y \in X$, $d_X(x,y) = d_Y(f(x),f(y))$.
- Composition: production of combinations from topology and/or routing information units so as to build more complex topology and/or routing information units (called structures).
- Mining: includes all functions enabling to find (hidden) relationships between routing and/or topology information units, features/properties and classes in these information units.

Routing path resolution:

- Computation: function applied to (structured) routing information units and/or (structured) topology information units to produce routing paths which can be used to derive routing table entries. Computation can be seen as the operation of finding the routing path that minimizes/maximizes a (multi-)constrained (multi-)objective function. Computation functions can be sub-divided into:
    - Global computation: for instance, application of the Dijkstra shortest-path tree algorithm on the entire link state topology
    - Incremental computation: by making use of the structure of the previously computed shortest-path tree, this approach minimizes the changes to the topology of an existing shortest-path tree when some link states in the network have changed
    - Sequential computation: makes use of intermediate computation steps before computing routing paths to destinations
- Selection: either by enforcing selection rules, by applying filters, or by multi-criteria decision on a set of routing information units (typically routing paths with associated attributes). By means of this processing, a limited number of routing paths is selected from which routing table entries can be derived.

Additional functions part of the routing functional area include

- Transfer of routing table entries: a mechanism allowing to export the routing table entries towards the forwarding engine component
- Trigger for renewal/update of the local routing states based on external or internal events.

Associated functionality to the routing functional area can be classified as follows:

- Identification: the functionality that assigns identifiers to nodes. These names can be either topology-dependent (locators) or topology-independent (names); a locator can take the form of a label, a topology-dependent address or a coordinate.
- Resolution: translation (or mapping) from the name of the destination to its associated locator.
- Location: the functionality allowing destinations to be located by means of the resolution function.

3. Performance Evaluation and Analysis

This section describes the performance objectives, criteria and metrics to be met by candidate routing models/algorithms. These performance criteria and metrics will drive the performance evaluation of the routing models and algorithms to be conducted in the context of the task 2.2 (by means of formal methods/techniques) and task T3.3 (by means of simulation). Indeed, tasks T2.2 and T3.3 shall be distinguished from the experimental analysis (comprising both evaluation and comparison) of routing protocol components that will be conducted by means of emulation/prototypes as part of task T4.3.

This section also analyzes the fundamental and unavoidable trade-offs that exist between some of these performance criteria and metrics. This shall be taken into account when specifying routing procedures in context of task T2.1 and designing the corresponding algorithms in the context of task T2.2. For instance the memory space - routing path stretch tradeoff is at the inception of compact routing schemes.

3.1 Performance Objectives

This section details the performance objectives that the routing algorithms that will be designed in the context of WP2 task T2.2 shall meet. We recognize the fundamental tradeoffs that exist between these objectives.

3.1.1 Scalability

Definition: ability of a computational system (hardware or software) to continue to function (without making changes to the system) under satisfactory and well specified bounds, i.e., without affecting its performance, when its input is changed in size or volume or in their respective rate of variation. Examples include increasing number of nodes/AS, increasing number of links, increasing number of hosts/prefixes.

Quantitative objective: the number of reachable nodes/hosts the routing scheme/model shall support should be of the order of 10^9. So, at least 3 order of magnitudes higher than the current routing protocols such as Border Gateway Protocol (BGP) supports today (about $3.5\ 10^6$). The number of Autonomous Systems (AS), which is approximately of the order of $3.5\ 10^4$, should be able to scale up to $3.5\ 10^7$. In proportion, each AS containing in average 10 nodes, the number of nodes to be supported should be of the order of 10^8.

3.1.2 Complexity

Definition: by complexity we refer here to the computational complexity and algorithmic complexity of the routing algorithm.

- *Computational complexity*: a measure of the computational resources needed to solve computational problems. Computational resources are measured in terms of either time (i.e., number of elementary computational steps per second) or space (i.e., size of memory usually measured in bits or bytes) or some combination of the two.
- *Kolmogorov complexity* (a.k.a. program-size complexity) a measure of complexity that quantifies how complex a system is in terms of the length of the shortest computer program, or set of algorithms, need to completely describe the system. In other terms, this measure of complexity qualifies how small a model of a given system is

necessary and sufficient to capture the essential patterns of that system. Formally, the Kolmogorov complexity of a string x is the length of the smallest program p that outputs x, relative to some model of computation. That is, $C_f(x) = \min_p \{|p| : f(p) = x\}$ for some computer f; $C_f(x)$ is the minimal length of a program for f (without additional input) to compute the output x. This definition can be extended to account for the input string y by defining the conditional Kolmogorov complexity of a string x, relative to a string y and a model of computation f as $C_f(x|y) = \min\{|p| : C_f(p,y) = x\}$. $C_f(x|y)$ is the size of the minimal program for x when started with input y.

Quantitative objective: the computation complexity (both in time and space) should grow sublinearly with the number of reachable nodes/prefixes.

3.1.3 Routing Quality

Definition: the quality of the routing paths as produced by the routing algorithm (both in stationary and non-stationary conditions) as determined by their stretch, stability, and convergence properties

Quantitative objectives related to routing quality cover:

- *Path cost/length*: Minimize the ratio between the cost/length of the routing path(s) as produced by the routing scheme and the minimum path cost/length for the same source-destination pair. Note that resilient/fault-tolerant schemes may further deteriorate this ratio (in particular, for pre-provisioned schemes enforcing the disjointness of protected/ protecting routing path pairs before failure occurrence).
- *Stability*: the stability properties of the individual routing entries and routing table should be such that they minimize perturbation resulting from i) the exploration of the routing state space (compared to the BGP uninformed path exploration intrinsic to shortest-path vector algorithm) and ii) the routing policies interactions (compared to BGP routing policy interactions that can lead to non-deterministic and unintended but stable routing states, and "dispute wheels", i.e., non-deterministic and unintended but unstable states).
- *Convergence*: upon occurrence of an instability event, e.g., physical topology change, routing topology change or protocol change, the convergence properties of the routing system should minimize the number of operations/execution steps (expressing the convergence time) needed to reach a new routing state. This new routing state results from the local re-computation and/or re-selection of new routing paths. The properties of this new state shall verify are i) consistency (do not result in any forwarding loop due to this event) and ii) globally stable (do not lead to any subsequent re-computation of routing table entries due to this event).

3.1.4 Adaptivity

Definition: the capacity of the routing system to adapt/react in a timely and cost-effective manner when internal or external events occur that affects its value delivery. Adaptivity to topology changes (due to network engineering e.g. add/remove link or node or network failures) is referred to as structural adaptivity. Next, adaptivity is also concerned with the spatio-temporal variability of the traffic (leading to traffic engineering decisions and/or network engineering decisions) and with its ability to

support arbitrary non-technical constraints and/or decisions/rules (driven by cost minimization, profit/revenue maximization, etc.).

Quantitative objectives:

- *Structural adaptivity*: capability of the routing models/algorithms to adapt to short- and long-term topology dynamics (resulting, e.g., from the addition or removal of different percentage of links, nodes, autonomous systems/network partitions, and destination prefixes) while minimizing the convergence time to a stable and consistent routing state as well as the associated processing capacity mobilized to reach that state together with the robustness/longevity of this state over time.
- *Traffic adaptivity*: capability of the routing models/algorithms to adapt to short- and long-term traffic variability/dynamics by minimizing the convergence time to a stable and consistent routing state as well as the associated processing capacity mobilized to reach that state together with the robustness/longevity of this state over time.
- *Economic/Cost adaptivity*: capability of the routing models/algorithms to adapt to short- and long-term routing policy dynamics by minimizing the convergence time to a stable and consistent routing state as well as the associated processing capacity mobilized to reach that state together with the robustness/longevity of this state over time
- Note: a priori there is no requirement on how these objectives are to be met by the routing model/algorithm.

3.1.5 Availability

Definition: Availability is defined as the probability that the system is operating properly when it is requested for use, i.e., the probability that a system is not in a failure state or undergoing a repair action when it needs to be used. Availability is expressed as a function of reliability and maintainability.

- *Reliability*: probability P_r that a system or component fails within a given period of time. Reliability is a function of time that expresses the probability at time t+1 that a system is still working, given that it was working at time t. Reliability represents the probability of components and systems to perform their required functions for a desired period of time without failure in specified environments with a desired confidence. Reliability, in itself, does not account for any repair actions that may take place. Reliability accounts for the time that it will take the component or system to fail while it is operating. It does not reflect how long it will take to get the unit under repair back into working condition.

- *Maintainability*: probability P_m that a system or component will be retained in or restored to a specified condition within a given period of time.

The notion of availability depends on what types of downtimes are considered in the performance analysis. As a result, there are two main classifications of availability:

- Time-based classification:
  - Point (instantaneous) availability: probability that a system (or component) will be operational at any random time, t.

Unlike reliability, the instantaneous availability measure incorporates maintainability information.
  - Average up-time availability (mean availability): proportion of time during a mission or time-period that the system is available for use. It represents the mean value of the instantaneous availability function over the period T.
  - Steady state availability: limit of the instantaneous availability function as time approaches infinity.

- Engineering classification:
  - Design/system availability used in system engineering is defined as the ratio MTBF / (MTBF + MTTR) where MTBF denotes the Mean time Between Failure and MTTR the Mean Time To Repair. The MTBF is inversely proportional to the probability P_r: MTBF ~ $1/P_r$. The MTTR is inversely proportional to the probability P_m: MTTR ~ $1/P_m$
  - Operational availability used in network engineering is defined as the ratio MTBM / (MTBM + MDT) where MTBM denotes the Mean time Between Maintenance and MDT the Mean Down Time. Operational availability is a measure of the uptime that is the total time the system was functioning between two experienced sources of downtime (such as administrative downtime, logistic downtime, etc.) during the operating cycle that is the overall time period of operation being investigated.

Improving routing system availability implies thus to improve the mean time to repair and more generally the maintainability capabilities of the routing system by means of resiliency/fault-tolerance techniques. Resiliency is the ability of a system to reach (rapidly) and maintain an acceptable level of functioning and structure with one or more of its components malfunctioning. Note that resiliency does not refer to a "full" but an "acceptable" level of functioning and does not refer to the correction of these malfunctioning components. Resiliency mechanisms are characterized in terms of the level of protection they provide (e.g., full protection against single failures, best effort protection, traffic pre-emption in case of failure, etc.) and the time required to activate them in case of topological failure. Coverage, mobilization of recovery resources, and activation time enable to define quantitative objectives:

- Maximize the percentage of links (or nodes) that can be fully protected (i.e., for all destinations) while minimizing the recovery resources (protecting path length/cost). Note that some percentage of the possible failures may be identified as being un-protectable.
- Maximize the percentage of destinations that can be protected for all link (or node) failures while minimizing the recovery resources (protecting path length/cost). Note that some percentage of destinations may be identified as being un-protectable.
- Maximize the percentage of the total potential failure cases (destination x failures) that are protected while minimizing the recovery resources (protecting path length/cost).
- Minimize the difference between the number of packet flowing through the network towards their destination before and after failure occurrence, i.e., maximize the percentage of packets passing through the network that will continue to reach their destination by using the protecting paths.

3.1.6 Performance Objectives from Algorithmic Game Theory

Routing in the presence of selfish and possibly antagonistic participants is one of the first (see [KP99]) problems examined under the lens of Algorithmic Game Theory (AGT) and still remains a very active research area [NRTV07]. Routing on the Internet and Internet-like networks is a prominent example of problems within this research area and usually depends on the policy of the entities involved. This certainly holds for the existing protocols (e.g., BGP) but is also expected to be true for new protocols that may be developed within EULER. For example, the contents of the routing table of a router affect the amount of traffic this router will handle and, hence, under the hypothesis of rational behavior by the router's administrator, it is reasonable to assume that these contents are edited so that the traffic handled is minimized. In this sense, the entities involved in routing are engaged in a strategic game; each such entity acts as a player and aims to select a strategy (e.g., the contents of the routing table in our example) that minimizes her cost (or maximize her benefit) given the strategies of the other players.

1) Existence of Equilibria

Definition: Nash equilibria [Nas50] correspond to stable states in which no participating entity has an incentive to alter her strategy/action (e.g., the routing path it has chosen) assuming that the remaining participants do not deviate. Clearly, such states may be different than the optimal state. In pure Nash equilibria, all players follow pure (deterministic) strategies, while in mixed Nash equilibria players are allowed to use probabilistic strategies (i.e., probability distributions over pure strategies). Pure and mixed Nash equilibria correspond to settings where participating entities are assumed to know in advance the strategy followed by other participants. A more natural setting is when initially the players have no knowledge of the choices made by other players and have to learn them by observing their behavior. In such a setting, the notions of correlated and coarse correlated equilibria can be viewed as generalizations of mixed Nash equilibria, where the players have a joint probability distribution instead of independent ones. Informally, in both settings, a mediator draws a strategy vector from a publicly known distribution and secretly informs each player of her suggested strategy. If no player has an incentive to deviate from the suggested strategy, then this is a correlated equilibrium [Aum74], while if no player has a pure strategy that he can always follow, irrespective of the outcome, and reduce his expected cost, then this is a coarse correlated equilibrium ([MV78], see also [You04]). Even though mixed Nash equilibria (and, hence, also correlated and coarse correlated) always exist (due to the theorem of John F.Nash [Nas50]), the existence of pure Nash equilibria is not always guaranteed. Pure and mixed Nash equilibria are solution concepts in full information settings, where it is assumed that players have complete knowledge of the game played, e.g., they are aware of the demand of other players. A more realistic concept is that of incomplete information (or Bayesian) games. Bayes-Nash equilibria [Har67] correspond to stable states in such settings where participating entities have "beliefs" about certain characteristics of other entities.

Quantitative objectives: To study the impact of the policy of AS administrators and whether the corresponding policy-induced games exhibit stable states (and of which type).

2) Time to Equilibrium

Definition: Consider an arbitrary initial state. If all players are satisfied and have no incentive to deviate, then this state corresponds to equilibrium, otherwise certain players will prefer to change their strategy. Thus, the system reaches a new state, and, similarly, this process may continue forever (in the case of loops) or may reach equilibrium. If this process is guaranteed to reach an equilibrium starting from any initial state, then the game satisfies a convergence property. Since at any moment more than one player may wish to deviate, no assumptions are made with respect to the order or priority of moves/deviations.

Quantitative objectives: the number of rounds (alternatively, moves) required to reach equilibrium.

3) Efficiency of Equilibria

Definition: the inherent selfishness of participating entities may lead to stable states that differ from what would be optimal from a designer's point of view. The notion of efficiency captures the degradation of the system performance due to this selfishness.

Quantitative objectives: the ratio of the equilibrium cost over the optimal cost.

3.2 Performance Criteria

This section details the performance criteria that will be considered for determining whether a given routing model or scheme/algorithm actually meets its performance objectives or not.

3.2.1 Scalability

The scalability of a system is measured by the rate x state x size of input that the system can sustain when running using a given number of resource units (for processing and storage). In the distributed routing context, scalability is measured by i) the memory space consumed to store the routing information, ii) the memory space consumed to store the resulting routing table entries, and iii) sustainable rate of communication messages (that result in re-computation and/or replacement of the routing table entries).

Criteria: the memory space required to store the routing table entries should scale better than n, the number of (abstract) nodes (and be preferably of the order of log n, i.e., the routing scheme should produce sub-linear routing table size. The routing state update rate is dependent on routing scheme response to external events (reachability, topology and/or traffic variations) and also on routing scheme specifics.

3.2.2 Computational Complexity

The computational complexity of a routing algorithm is measured by the complexity in time and space/resource.

Criteria: the number of operations/execution steps (expressing the computational time) needed to (re-)compute a (set of) routing table entries should be sublinear in the input size. This size of the input depends on

the topology properties such as the number of nodes/links, the number of paths/prefixes, network diameter, etc.

3.2.3 Quality

The quality of a routing system and thus the routing model/algorithm that sustains it, is measured by

- The *stretch* of the routes produced by the routing algorithm in stationary and non-stationary/variable conditions (in case of routing information or protocol operation change)
- The *stability* of the individual routing entries and routing table (the number of routing table updates before and after routing information or protocol operation change)
- The *convergence time* of the individual routing states

Criteria are respectively the following

- The routing path length/cost should be as close as possible to the shortest routing path length/cost (both in stationary and non-stationary conditions). Ideally after reaching a fraction above stretch 1, a substantial amount of the routing paths produced by the routing schemes and nodes should be covered).
- Upon local routing state change, decrease number/rate of communication/routing update message (or variation of their attributes) to downstream neighbors and decrease variation of local routing state upon reception of such message(s).
- Decrease the number of operations/execution steps (thus the convergence time) needed to reach a new stable and consistent routing state (compared to BGP behavior for the same topology and the same reachable prefix); in absolute terms, convergence to such stable state should be reached in polynomial (preferably linear) time.

3.2.4 Adaptivity

Being topology- traffic- or cost-driven, adaptivity is measured by the convergence time to a stable and consistent routing state as well as the associated processing capacity mobilized to reach that state together with the robustness/longevity of this state over time.

Criteria: when converging to a short-live state, the routing algorithm shall only require updating the affected routing state(s) and the updates performed within the time interval needed for the updated routing and/or topology information units to reach the concerned nodes. On the other hand, when converging to a long-live state, the routing algorithm may require updating not directly affected routing state(s) and may have to perform these updates during a larger time interval than the one needed for the updated routing and/or topology information units to reach the directly concerned nodes. In the latter, the routing algorithm must provide the means to converge to a new stable and consistent long-live routing state before the next internal or external event occurs.

3.2.5 Availability

The routing system availability is measured by the ratio MTBF / (MTBF + MTTR) where MTBF is the Mean time between failure and MTTR the Mean Time To Repair of the routing table entries and associated forwarding paths.

Criteria: the MTTR of forwarding paths resulting from links/nodes failure must be short enough (while minimizing the mobilization of protecting

resources) so that the difference between the number of packet flowing through the network towards their destination before and after failure occurrence is negligible and can be recovered by upper layers. Moreover, the coverage (percentage of links/nodes that can be fully protected for all destinations, percentage of that can be protected for all link/node failures) should be optimized against the recovery resources mobilized.

3.2.6 Performance Criteria from Algorithmic Game Theory

1. Potential functions

Definition: A potential function takes as input a state of the game and returns a number. More importantly, a potential function entails the incentive of all participating entities to deviate from the current state; this leads to the concept of potential games [MS96]. The potential function is a useful tool to analyze equilibrium properties of games and the set of pure Nash equilibria can be found by simply locating the local optima of the potential function.

Criteria: Existence of potential functions. Preferably, the potential functions should be computable in polynomial time.

2. Price of anarchy

Definition: The price of anarchy ([KP99, see also [Pap01]) for a class of equilibria corresponds to the worst-case efficiency of equilibria of the given class. In other words, it denotes the worst-case deterioration of the system performance due to selfish behavior and lack of coordination. The price of anarchy is measured by the ratio of the worst equilibrium cost over the optimal cost.

Criteria: Low (e.g., constant) price of anarchy.

3. Price of stability

Definition: The price of stability [ADK04+] for a class of equilibria corresponds to the best-case efficiency of equilibria of the given class. In other words, it denotes the best-case deterioration of the system performance due to selfish behavior and denotes the best efficiency that can be attained by a coordinator/designer that is able to propose stable states to the participating entities (e.g., specific paths for routing). The price of stability is measured by the ratio of the best equilibrium cost over the optimal cost.

Criteria: Low (e.g., close to 1) price of stability.

4. Time to Equilibrium

Definition: the process of reaching a state corresponding to an equilibrium starting from arbitrary initial states. Since at any moment more than one player may wish to deviate, no assumptions are made with respect to the order or priority of moves. The time to equilibrium is measured by the number of rounds/moves required in order to reach equilibrium.

Criteria: Convergence to equilibria in polynomial (preferably linear) time.

3.3 Performance Metrics

Performance metrics identified for the measurement of the routing scheme/ algorithm performance are:

3.3.1 Stretch

We distinguish between the multiplicative and additive stretch of the routing paths produced by a routing scheme:

- *Multiplicative stretch* (of a routing scheme) is defined as the ratio between the cost/length of the routing path(s) as produced by the routing scheme and the minimum path cost/length for the same source-destination pair. Intuitively, the stretch of a routing scheme provides a quality measure of the path cost/length increase it produces, compared to the shortest paths. Shortest path routing schemes either AS-path length based (path vector routing) or cost-metric based (link-state routing) are stretch-1. This metric is interesting to measure because compact routing schemes that produce reduced routing tables, are not always able to choose the minimum cost/length path for a given destination. On the other hand, the routing scheme should favor computation and/or selection of routes whose stretch remains closer to 1.

- *Additive stretch* (of a routing scheme) is defined as the difference in number cost/length between the routing path(s) as produced by the routing scheme and the minimum path cost/length for the same source-destination pair.

3.3.2 Routing Table Entries Storage

This metric measure the storage capacity required to store the routing table entries (it thus measures the number of entries and their respective size):

- Number of (local) routing table entries, equivalently the number of active routing states

- Size of the locally stored routing table entries expressed in terms of memory-bit space consumed to store these entries

3.3.3 Computational Complexity

Computational complexity measures the computational resources required for the (re-)computation of routing table entries in terms of time and space:

- *Time complexity*: number of operations/execution steps (expressing the computational time) needed to (re-)compute a (set of) routing table entries as a function of the input size.

- *Space complexity*: amount/size of memory in bits or bytes needed to (re-)compute a (set of) routing table entries.

3.3.4 Communication/Messaging

The communication cost and complexity can be measured by means of:

- Number of (routing and/or topology update) messages exchanged between nodes to maintain a coherent and timely non-local knowledge about network topology and/or reachability such that each node can (re-)compute a consistent (set of) routing table entries

- Size of communication (routing and/or topology update) messages

- Sustainable rate of communication (routing and/or topology update) messages

3.3.5 Convergence

Upon occurrence of an instability event, e.g. routing topology change or protocol change, this metric measures the number of operations/execution steps (expressing the convergence time) needed to reach a new routing state (resulting from the re-computation and/or re-selection of new routing paths) that is consistent (do not result in any forwarding loop) and globally stable (do not lead to any subsequent re-computation).

3.3.6 Repair Coverage

This metric applies specifically to fault-tolerant routing schemes by means of repair paths (a.k.a. alternate or backup paths) that are pre-computed in anticipation of topological failure and made available for invocation with minimal delay. The repair coverage provides a simple comparison metric for measuring and comparing fault tolerant routing schemes.

- The percentage of links (or nodes) that can be fully protected (i.e., for all destinations). Note that some percentage of the possible failures may be identified as being un-protectable.
- The percentage of destinations that can be protected for all link (or node) failures. Note that some percentage of destinations may be identified as being un-protectable.
- The percentage of the total potential failure cases (destination x failures) that are protected.
- The percentage of packets normally passing through the network that will continue to reach their destination by using the pre-computed repair paths.

3.3.7 Multicast support

This section details a set of specific metrics of interest for the evaluation of the performance of multicast routing schemes. Indeed, the performance evaluation of multicast routing schemes requires either different definitions or additional metrics with respect to (unicast) routing schemes.

- *Stretch*: in the particular case of multicast routing algorithms, the stretch metric is obtained as the maximum ratio over all sets of nodes between the cost of the point-to-multipoint routing path (that defines the multicast tree) as obtained by the proposed scheme and the cost of a minimum-cost tree, i.e. the Steiner Tree (ST), between the same set of destination nodes.

- *Tree cost*: is defined as the sum of the edge cost composing the point-to-multipoint routing path. The number of tree nodes and tree levels as well as the number and degree of the branching nodes may complement this metric.

- *Storage (memory space)*: the memory space required to store the various routing table entries required for the multicast routing algorithm to properly operate or the routing table entries produced by the multicast routing algorithms:
    - Unicast Routing Information Base (URIB): the URIB stores the unicast routes to the possible destination nodes in the network. For multicast routing schemes relying on the unicast routing topology, this table includes the entry pointing to the source of multicast traffic in any-source multicast (ASM) or source-specific multicast (SSM).
    - Multicast Routing Information Base (MRIB): when relying on the underlying unicast routing, the multicast routing scheme makes use of the MRIB derived from the URIB. The MRIB is used to route the multicast control messages: each entry stored in the MRIB is used to determine the next-hop neighbor to which any Join/Prune message is to be sent to join/leave a multicast group <*,G> in ASM or multicast source - multicast group pair <S,G> in SSM.
    - Tree Information Base (TIB): the TIB is the multicast routing table, built from every Join/Prune message received. It holds the state of the multicast distribution trees at a router. The entries in the TIB are used to forward multicast traffic though forwarding operations by direct TIB lookup is inefficient (due to the indirection such operation implies). Henceforth, multicast forwarding in high-performance routers is performed directly along the forwarding path by means of Multicast Forwarding Information Base (MFIB) constructed from the information stored in the TIB.

- *Computational complexity*: measures the computational resources required for the (re-)computation of multicast routing table entries in terms of time and space, e.g., due to a join/leave message
    - Time complexity: quantifies the number of operations/execution steps needed to (re-)compute a (set of) routing table entries as a function of the input size.
    - Space complexity: the amount/size of memory in bits or bytes needed to (re-)compute a (set of) multicast routing table entries.

- *Communication cost*: the number of routing update messages that needs to be exchanged between nodes to converge after a non-local network topology change (e.g., a branching node failure) or a multicast tree change (e.g., a node joining or leaving a tree) together with their size. The communication cost provides a measure of the processing capacity required at each node to process messages.

- *Convergence*: measures the amount of time it takes to converge (i.e., to re-compute the multicast tree) upon a network topology change. This metric becomes of paramount importance as it reflects the total amount of time that the network routing functionality may remain inoperative due to link/node failures for example.

3.3.8 Effects on traffic spatio-temporal properties

Several metrics can be defined to measure the effects on the spatio-temporal properties of the traffic. Indeed routing table entries are used to derive forwarding table entries whose sequence determines the forwarding

path followed by the traffic. On the other hand, these metrics can be used to evaluate the gain of traffic engineering capabilities associated to the routing scheme:

- End-to-end (forwarding path) bandwidth x delay product
- Per destination (inverse tree)/per forwarding path (point-to-point) congestion
- Per destination (inverse tree)/per forwarding path (point-to-point) throughput
- Per forwarding path (point-to-point) aggregated load
- Statistical multiplexing gain with respect to the spatio-temporal traffic distribution
- Betweeness centrality (link or nodal): fraction of routing paths (as produced by the routing scheme) crossing a given link or a given node divided over the total number of paths in the network.

3.3.9 Game Theoretic Metrics

This section details game theoretic specific performance metrics that can be assessed by means of simulation.

- Best response strategies and assessment of the quality of the outcomes. For any game it is possible to define a graph that captures the dynamics emerging by the selfish behavior of the players. This graph has a node for each state and a directed edge (with label i) connects two states if player i has incentive to deviate; the two states have the same strategies for all remaining players and differ only in the strategy of player i. For a given state, there may exist more than one outgoing edges per player, since more than one improving deviations per player may be possible. A best response walk on these states is defined by considering only outgoing edges that model a best response strategy for a given player. This includes determining the efficiency of outcomes, studying approximation of equilibria (i.e., which is the smallest decrease in cost that is attainable by deviations) and computing the time required to reach equilibria (or approximate equilibria) through such best response walks.
- Fictitious play is an instance of model-based learning, in which the learner explicitly maintains beliefs about the strategy of other participating entities. Then, the learner plays a best response to the assessed strategy of the other entities, observes the actual play and updates her beliefs accordingly. This could include studying whether it is possible for a malicious entity to learn the routing pattern and harm routing.
- It is also interesting to study how much the change in a player’s strategy can affect the whole system. This is closely related to dynamic properties of networks.

3.4 Performance Comparison

The goal of this section is to specify a framework enabling to perform by simulation fair comparisons between different routing models and algorithms by means of reproducible scenarios. This framework should guarantee the reproducibility and reliability of the simulation experiments and to get relevant comparison between different routing models and algorithms.

Such performance comparison involves the combination of the choice of:

- A dataset that is the network on which we run an experiment
- A routing scheme
- A network dynamics model
- A model of the routing requests
- An execution model

Then we analyze the traces that are assumed to be stored.

3.4.1 Dataset

Different types of dataset can be considered

- Real data: as provided for instance by CAIDA
- Synthetic data: synthetic networks of different size. Some networks have to mimic real-life networks (scale-free graphs) and we also have to consider very different topologies (dense/sparse graphs, etc.) in order to characterize each individual routing scheme.

Concerning the size of the dataset it is important to compare networks of same size. Ideally, the order of magnitude of the number of nodes goes from 10K to 100K nodes. Note that the smallest CAIDA map has 16K nodes and the largest around 35K nodes.

3.4.2 Models

Once selected, we have to specify the topology dynamics models, the workload model, and the execution model. Each of them are detailed in the below paragraphs.

1. Topology dynamics model

Also referred to as "dynamic network or topology model", such model defines how the underlying topology changes/evolves over time. More precisely, the following types of models are considered:

- Evolving model without constraint: online insertion and/or suppression of links and/or nodes. In this model, if G(t) represents the graph of the network topology at time t then G(0) and G(t) can be quite different.
- Failure model: G(t) is a subgraph of G(0). In practice, we consider that few nodes/links are removed from G(0). This model [KB2010] was recently considered in several studies dealing with shortest paths computations whenever some errors are detected in the network. For instance: how to design a data structure capable to report an s-approximate shortest path between node u and v upon failure of node z (that sits along the path from u to v).

In both models, the event list should be stored before the beginning of each experiment. Note that we can afford any model of events generation: the so-called "edge Markovian" process model, set of links chosen with respect to a given distribution, etc.

2. Workload model

The workload model determines the set of new routing queries at the beginning of each step of the execution. This set is essential whenever we aim to measure and to analyze the load and resulting congestion at each node. Importantly, all the choices should be stored in order to make a fair comparison.

Given a distribution (probability to choose to given target), each node runs a list of routing queries. For instance, the uniform distribution corresponds to routing towards all nodes. It is possible that a node instantaneously runs several routing queries towards the same node and decides to run no new routing. More formally, for any time step t and every node u, the query workload $Q_u(t)=\{v_1,v_2,...,v_i\}$ determines the set of routing queries, i.e., node u asks to run a routing from u to v_1, v_2, ..., v_i. In order to make a clear comparison between different routing schemes with exactly the same setting, the set of routing queries should be computed and then stored. For an experiment, one of the inputs will be the pre-computed workload.

3. Execution model

The routing scheme and the network are modeled as a synchronous distributed system. At each time step t, each entity (link/node) of the network can potentially change its current state (activation, deletion, etc.) in order to define the graph G(t). Note that some messages (being data traffic and/or routing updates) can be lost during this phase whenever a node is deleted.

Then each node executes the following sequence:

- FORWARD the requests received in the previous time step: forward routing, discovery messages, control messages, etc. depending on the routing scheme.
- RUN new routing requests: this depends on the workload model.
- SEND new control messages: this depends on the routing scheme.

3.4.3 Traces Analysis

Within an experiment, the set of routing paths should be stored and can be analyzed with respect to the performance measures. However, we can consider two levels of analysis:

- First glance analysis: for each performance measure, the average and the distribution is computed
- Deeper analysis: the goal is to find some correlations between the performance measures and some local/global characteristics of the network. For instance, are the most central nodes the most used in the routing paths? This would not be too long to compute it. However, if we want to determine the most frequent paths, this task corresponds to the problem of computing the frequent item sets in data mining. Because the exponential number of potential combination of graph parameters and the performance measures, it is recommended to first do a first glance analysis and then to carefully select the combinations of attributes to analyze.

4. Experimental Analysis

Experimentation by emulation/prototyping of the (protocol) components of the routing scheme(s) designed in WP2 (task T2.2) will be performed as integral part of the WP4 activity. For this purpose, the iLab.t experimental facility and, where possible and suitable, larger experimental facilities such as OneLab2/PlanetLab Facility or Ofelia will be used.

The present section is sub-divided in three parts. One dedicated to the experimental evaluation of routing protocol components, the second to their comparison and the last one to observation experiments.

4.1 Experimental Evaluation

This section specifies the experimental evaluation framework in order to evaluate the performance of the considered routing protocol components. This framework should guarantee the reproducibility, reliability and verifiability of the emulation-based experiment in order to get relevant results.

Here below we present the detailed description of the experimental evaluation objectives, evaluation criteria, and associated metrics for the routing protocols to be experimented. It is to be noted that some of the functional and performance analysis objectives described in the previous sections are also considered here for experimental analysis purposes (simply because the experimental methodology can also be considered for performing functional and performance analysis). However, the experimental evaluation criteria and metrics are different since the methodology is basically different. On the one hand, the network size is different; while a simulated scenario can (easily) simulate a routing model in a 10k-nodes network (or even larger networks), only smaller networks can be emulated in the experimental platform (e.g., of the order of 100 nodes). On the other hand, certain measures can be performed on emulation platforms that are not (or more difficult) to realize with simulators such as metrics that involve traffic, e.g., traffic aggregation effects, system resource consumption, forwarding packet delay. Hence, experimental analysis will cover objectives that can not be realized with routing model simulation.

It is worth mentioning that the purpose of experimental analysis of routing protocols is not to measure the performance of the forwarding plane as such but the resulting effects of the routing protocols on the data traffic. That is, the effects on spatio-temporal distribution of traffic and its properties from the locally computed routing paths.

4.1.1 Objectives

Overall, the goal of experimental evaluation is to evaluate by means of emulation experiments, the routing protocol components derived from WP2 design task (Task T2.2). This evaluation will be conducted to determine if the targeted functional and performance objectives specified in Section 2.1 and 3.1 are met.

Taking into account the physical constraints of an emulation environment (in terms of resources e.g. number of nodes, CPU/memory per node) as provided by the IBBT's ilab.t experimental facility, this section translates the performance objectives measurable by emulation.

- *Scalability*: the scalability of a routing scheme (ability to cope with larger networks) up to the order of 10^9 can only be experimented in a simulation environment. Indeed no experimental facility provides the means to emulate the spatial distribution of destination networks as currently observed on the Internet; hence, there is no foreseeable mean by which one could experiment by emulation even larger distributions. However, it is the objective to measure scalability of the routing protocol components in isolated parts of the network in interaction with a larger part of network being simulated.

- *Computational complexity*: the objective is to determine the computational complexity of the routing path computation component on an individual network node or part of the network in interaction with a larger part of network being simulated.

- *Routing quality*: using a network comprising hundreds of emulated nodes, the objective is to measure the quality of a routing protocol as determined by
    - The cost/length of routing paths as produced by the routing protocol in stationary and non-stationary/ variable conditions (in case of routing information or protocol operation change).
    - The stability of the individual routing entries as produced by the routing protocol (the number of routing table updates before and after routing information or protocol operation change).
    - The convergence time of the individual routing table entries and the entire routing table of the node being experimented.

- *Adaptivity*: using a network comprising hundreds of emulated nodes, the objective is to measure the effects of
    - Short- and long-term topology dynamics on the local and global convergence time to a stable and consistent routing protocol state.
    - Short- and long-term traffic variability/dynamics on the local and global convergence time to a stable and consistent routing protocol state.
    - Short- and long-term routing policy dynamics on the local and global convergence time to a stable and consistent routing protocol state.

- *Availability*: using a network comprising hundreds of emulated nodes, the objective is to measure the properties of the resiliency/fault-tolerance mechanisms provided by the routing protocol in terms of:
    - Coverage (percentage of links/nodes that can be fully protected for all destinations, percentage of that can be protected for all link/node failures).
    - Mobilization of recovery resources, i.e., the amount of pre-provisioned resources (thus before failure occurrence) in order to ensure this repair coverage.
    - Time to repair of routing paths under different network running conditions/failure events (including the detection, the notification, and the activation time).

4.1.2 Criteria

The evaluation criteria are the same as those exposed in Section 3.2 taking into account the experimental environment and running conditions.

4.1.3 Metrics

The following evaluation metrics will enable to measure if the performance criteria (detailed in Section 4.1.2) can be met by the routing protocol and its components.

1. General metrics for routing protocols

- *Routing path length*: the number of nodes along the routing path from source to destination as produced by the routing protocol. The routing path length can be exactly measured both in network emulation. This measure allows computing the resulting stretch of a routing scheme.
- *Routing table size*: the number of routing table entries and the total size (expressed in terms of memory space) required to store them per node. Note this number/size should be sub-linear with respect to the number of nodes/reachable prefixes.
- *Computational complexity* (of routing paths): this metric can be measured by determining i) the number of CPU cycles and the memory space required to compute each routing path and ii) the time required to locally compute each routing path with respect to the input size.
- *Communication cost*: the rate x size of routing protocol messages exchanged between nodes that are needed by the routing protocol to properly operate. Note that routing protocol messages may include topology information and/or routing information; both types of information are referred to as routing protocol information. This metric can be measured in emulation.
- *Connection time* (node-up event): time needed for a new network node to connect to the existing topology (connected component of the topology). This time includes the local interface configuration time and neighbor discovery time as well as network discovery time (discovery of that node for the rest of the topology and discovery of the rest of the topology by that node). This metric can be measured in emulation.
- *Connectivity time*: the time needed for a newly added destination to become reachable through the existing network topology (connected component of the topology). In case of routing protocol information push: this time includes the propagation of the routing protocol messages and thus the corresponding information across the network topology. In case of routing protocol information pull: this time accounts for the query/response delay (and associated resolution if any) of this new routing protocol information by an existing node.
- *Recovery time*: upon network failure (link/node failure), routing states need to automatically reconverge. The time between failure occurrence and failure recovery results into several destinations becoming unreachable (loss of connectivity) until reconvergence of the routing states on the new topology. This is an indirect way to deduce the convergence time of routing protocol states. This metric can be measured in emulation.
- *Configuration time*: number of actions to perform off-line configuration of newly added elements into the topology being network partition(s), node(s), link(s), or destination(s). The associated operations can be modeled in emulation.

The below metrics refers to forwarding plane measures. The purpose is not to measure the performance of the forwarding plane as such but the

resulting effects on spatio-temporal distribution of traffic and its properties from the locally computed routing paths:

- Forwarding table size: the number of forwarding table entries and their total size per node (that should be sublinear with respect to the number of nodes/reachable prefixes).
- Forwarding delay: the time needed to determine the outgoing interface/port of a packet from its incoming attributes (including destination coordinates, destination address, label, etc.) using the local forwarding table. This metric is of particular importance for routing protocols that assume more than one lookup/online computation operation to forward packets.
- Throughput: average rate of successful message delivery over a network path. This metric can be approximated in emulation.
- Delay: the time between sending a packet from a source node, and the arrival of the packet at the destination node (in ms). The delay can be approximated in emulation (half round-trip time).
- Jitter: the difference in end-to-end delay between selected packets in a sequence of data packets flowing from the source to the destination. This metric can be measured in emulation.
- Packet loss: this occurs when one or more packets of data travelling across a network fail to reach their destination. This metric can be measured in emulation.

2. Specific metrics for multicast routing protocols

In addition to the above metrics considered for general (unicast) routing protocols, the following metrics are related with the experimental evaluation of the multicast routing protocol:

- Tree cost: is defined as the sum of the edge cost composing the point-to-multipoint routing path from the source of multicast traffic (root) to the set of destination nodes (leaves). This metric can be exactly measured in network emulation. It allows computing the resulting stretch of a multicast routing scheme. Note that the number of tree nodes and levels as well as the number and degree of the branching nodes may complement this metric.
- Communication cost: the rate x size of routing protocol messages exchanged between nodes as needed by routing protocol mechanisms to operate. In multicast routing, this includes the messages to join/leave a tree. This metric can be measured in emulation.
- Multicast update time: includes the time for a node to join a multicast tree and the time for a node part of a multicast tree to leave it. This metric can be measured in emulation.
- Routing table size: the number of multicast routing table entries and their total size per node. Note that in multicast routing, additional routing tables (introduced in the previous section) can be required if the multicast routing scheme relies on the underlying unicast routing topology. This metric can be measured in emulation.
- Routing path computation complexity (in time): time needed to compute the least cost path between a joining node and the tree.
- Multicast forwarding table size: the number of multicast forwarding table entries (that should be sublinear with respect to the number of nodes/reachable prefixes)
- Multicast forwarding delay: the time needed to forward an incoming multicast packet to the outgoing interfaces leading to the leaf nodes part of the multicast tree.

4.2 Experimental Comparison

The goal of this section is to specify an experimental comparison framework in order to perform a fair comparison between routing protocol components. This framework should guarantee the reproducibility and reliability of the emulation-based experiment in order to get relevant comparison.

In particular, the goal is to measure the various performance metrics gain with enabling certain functionality/capability (including additional routing metrics, additional functionality, etc.). Indeed, experimental comparison doesn't (only) consist in measuring the same metrics for the different protocol components running in the same (finite) conditions and environment. It also aims to compare the "gain" of certain capabilities and functionality part of/added to routing protocols but not in others knowing also that the experimental environment induces certain limits of the scale of the experiment itself. As such it can be considered as a differential comparison assuming that each protocol provides what it is designed for (functional reliability).

For instance, measuring the link load distribution resulting from the routing path computed and selected by each protocol opens the following question: how to compare a protocol that has no traffic engineering capability against a protocol that provides such capability without biasing the conclusion? Indeed traffic-engineering implies additional computation at setup time but lesser at recovery time (since routes are spatially distributed) whereas shortest-path routing implies lesser computational cycle at setup time but may imply more computation if the failure is affecting a "central" node or link of the topology.

The experimental running conditions are also important to consider. Indeed, knowing that the number of routing paths is finite in an emulation environment the effects of hyper-aggregation might not be necessarily observable if the number of routing entries is limited by processing capability of such central node. On the other hand, the benefits of traffic engineering wouldn't be observable too.

4.2.1 Comparison Objectives

Different routing schemes rely on different protocol components to accomplish the functionality described in Section 2.1. Therefore, the evaluation of the realization of a given performance objective by one scheme requires different experimental evaluation methods compared to the evaluation to be performed for another routing protocol. The goal of this section is to detail the experimental objectives that can be realized in an emulation environment when comparing routing protocol performances with respect to the functionality they have to accomplish.

- *Scalability*: the comparison of the scaling behavior of different routing protocols will be performed by evaluating their ability to continue to function under satisfactory and well specified bounds. (i.e., without affecting its performance). The following set of varying parameters will be considered for this purpose:
    - Various topologies showing different properties including the number of nodes, the minimum/maximum/average degree, the clustering coefficient, the diameter (length of the longest shortest path), and other specific topological properties.
    - Variable number of nodes/AS, number of links, number of hosts/destination prefixes.

  - Various network traffic spatio-temporal distributions (by means of traffic generators) that are representative of the Internet traffic properties.

  For all routing protocols, a performance profile will be determined for best, worst and reference scenario in an emulated network.

- *Computational complexity*: the computational complexity of different routing protocols running in an emulation environment will be compared when they perform the following functions i) local and network discovery, ii) topology-/routing-information structuring, and ii) routing path resolution.
  For all routing protocols, a complexity profile will be determined for best, worst and reference scenario in an emulated network.

- *Routing quality*: the routing quality properties of the different routing protocols will be compared against i) a set of topologies representative of the Internet topology, and ii) different spatio-temporal traffic distributions and traffic mixes. For all routing protocols, a quality profile will be determined for best, worst and reference scenario in an emulated network.

- *Adaptivity*: the adaptivity properties (with respect to the topology dynamics and traffic dynamics) of the routing protocols will be compared by measuring their capability to adapt
  - The routing table entries produced when dealing with the addition/removal of different percentage of links, nodes, autonomous systems/network partitions, and destination prefixes,
  - The routing paths properties (in terms of, e.g., bandwidth x delay product) against variable spatio-temporal traffic distributions.
  - Concerning adaptation to policy dynamics, specific scenarios shall be developed to determine the ability of the routing protocol to avoid instabilities resulting from policy interactions as observed in BGP such as "wedgies", i.e., non-deterministic and unintended but stable routing states, and “dispute wheels”, i.e. non-deterministic and unintended but unstable states.

  For all routing protocols, an adaptivity profile will be made for best, worst and reference scenario in an emulated network.

- *Availability*: the objective is to compare the characteristic properties of the resiliency / fault-tolerance mechanisms provided by different routing protocols in terms of their
  - Coverage (percentage of links/nodes that can be fully protected for all destinations, percentage of destinations that can be protected for all link/node failures).
  - Mobilization of recovery resources for this coverage.
  - Time to repair their routing paths under different network running conditions/failure events (including the detection, the notification, and the activation time).

4.2.2 Comparison Criteria and Metrics

This section details the experimental comparison criteria that will be considered to verify whether one routing scheme performs better in an experiment than another, according to a specific experimental objective detailed in Section 4.2.1. This section also details the experimental

comparison metrics that will provide the measures in order to perform a systematic comparison. Note that except when explicitly mentioned the running conditions and experimental environment are assumed to be identical throughout the various executions required to perform such comparisons.

- *Scalability*: routing protocol's scalability will be qualified as better than another one's, if its scaling (as measured by the sustainable rate of routing state updates, number of routing states and the memory space required to store them) perform better in the reference scenario of the other. Indeed, suppose two routing protocols, each of them will show best case scaling performance on reference scenario that can be different. Thus, comparison shall be performed by running one protocol using the scenario for which the other performs the best and the reference scenario for which it performs the worst (and vice versa). The protocol that will show the least performance degradation and the best performance improvement on the other's reference will be qualified as scaling better.

- *Computational complexity*: the computational complexity of a routing protocol qualifies as lower than another one's (when performing the functionality described in Section 2.1) in case its complexity both in time and resources is lower than the one obtained for the reference scenario.
    - Time complexity measures used for comparison include i) the number of computational steps/operations required to compute routing table entries with respect to the input size, ii) the number of computational steps/operations needed for each routing state to converge to a stable state.
    - Space complexity measures used for comparison include i) the memory space required to store the resulting routing table entries, ii) the memory space required to store the routing state updates.

- *Routing quality*: the routing quality properties (cost/length/cost of the routing path, the stability of its routing states, and their convergence properties) of a routing protocol will be qualified as better than another one's, in case:
    - The lengths/costs of the routing paths that it produces are all closer to the shortest routing path lengths/costs (as measured by the stretch) in all scenarios.
    - Upon occurrence of an instability event, e.g., topology change or protocol change, the number of routing state updates and the number and rate of the topology-/routing- update messages exchanged is minimal.
    - The number of operations/execution steps (thus the convergence time) needed to reach a new stable and consistent routing state is minimal.

- *Adaptivity*: the capability of a routing protocol to adapt to topology dynamics (resulting from addition/removal of different percentage of links, nodes, autonomous systems/network partitions, and destination prefixes), traffic variability (that induces variability in the properties of the routing paths), and policy dynamics will be qualified as better than another one's if the following conditions are met:
    - Its convergence time to a stable and consistent routing state is shorter,
    - The processing capacity mobilized to reach that state is lesser, and

  - The robustness of this new state over time is longer.

- *Availability*: the characteristic properties of the resiliency /fault-tolerance mechanisms provided by different routing protocols will be qualified as better than another one's if the following conditions are met:
  - Its coverage (percentage of links/nodes that can be fully protected for all destinations, percentage of destinations that can be protected for all link/node failures) is higher,
  - Its mobilization of recovery resources for this coverage is lesser, and
  - The time to repair its routing paths under different network running conditions/failure events (including the detection, the notification, and the activation time) is lower.

4.3 Experimental Observation

In the work related to WP3 Task 3.2, we observed the properties of routing trees captured by ego-centered measurements with tracertee from PlanetLab monitors, and their dynamics captured by radar measurements (iterated ego-centered measurements) [LMOradar].

This work led to the identification of key features that should be reproduced in measurement-based models of the Internet, of routing and of their dynamics [HLMevent], and will be used as criteria for assessment of future routing protocols and emulated environments. We summarize them briefly below.

4.3.1 Node-based properties

The first salient trait of ego-centered measurements is that the number of nodes (and links) in successive ego-centered measurements is very stable: it slightly oscillates around a mean value, with no notable change. The mean value itself may change during time, leading to different regimes, but it remains stable for wide periods. Fig.5 shows a typical example.

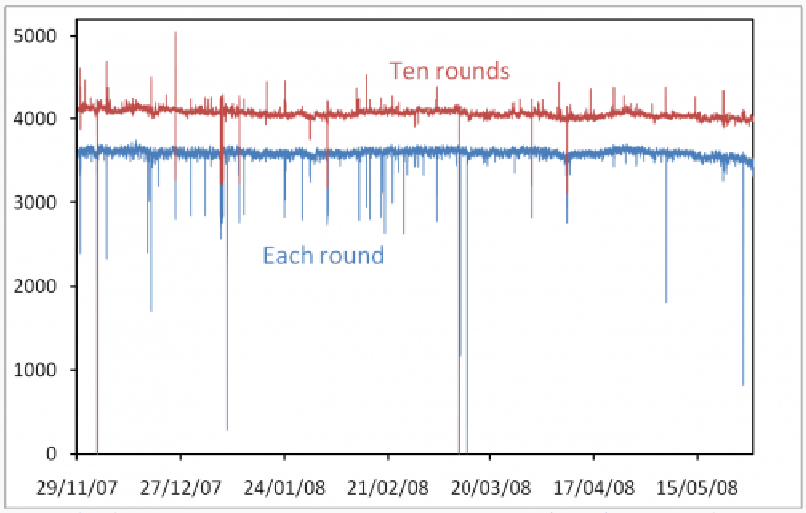


Fig.5: umber of distinct IP addresses seen at each round (blue) and during ten consecutive rounds(red)

This does not mean that the nodes observed in each ego-centered view are always the same, though, as demonstrated by the plot of the number of distinct nodes in consecutive rounds of measurements (see Fig.5) and the plot of the number of distinct nodes observed since the beginning of the measurement (see Fig.6).

This stability of the number of nodes of routing trees certainly is an important and desirable feature for protocol design. The growth of the total number of observed nodes with iterated measurements seems to be due to routing dynamics, and will be used as a lower bound for performances of future protocols, as current ones succeed in reaching this level of stability.

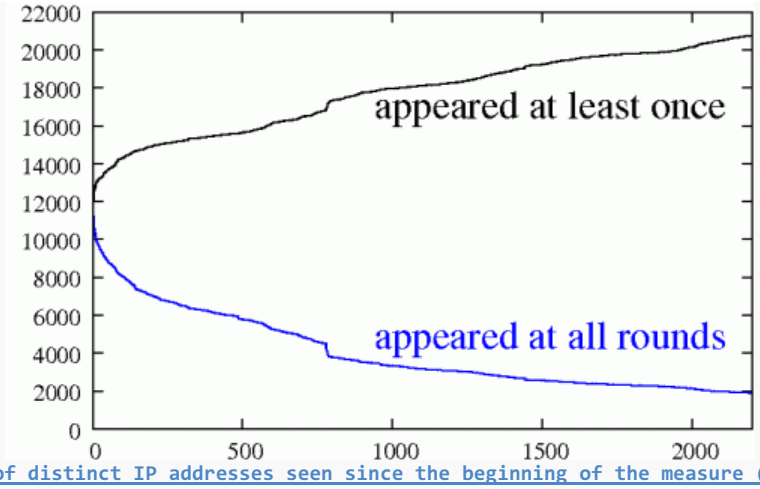


Fig.6: umber of distinct IP addresses seen since the beginning of the measure (in black) and the number of IP addresses which were present at all rounds (in blue)

4.3.2 Distance-oriented properties

A key feature of the dynamics is the appearance of new links in the topology and their discovery (apparent appearance) by ego-centered measurements. In typical radar measurements, this may be quantified by computing the distribution of the distances between two nodes immediately before a link is discovered between them (a link which was previously not there or that we had not discovered). This distribution clearly shows that appearing links are between nodes at small distances from each other (and uneven distances are overrepresented because of the presence of diamonds). This is not surprising, but some appearing links are also between nodes which were previously very far from each other (up to 30 hops).

Such features show that routing changes mostly are local (probably due to load balancing), which will be an important criteria for performance of new routing protocols. However, non-local changes also occur, and, again, real-world observations provide indications of performances attained by current protocols to this regard.

5. Conclusion

We conclude this deliverable by documenting the requirements on the various tools that will be required to conduct the different validation, and verification phase of the routing schemes that will be developed in the context of the EULER project as well as the evaluation and analysis of their performance.

1. Tools requirements for formal verification and analysis (that will be conducted in Task 2.2)

Formal verification aims to rigorously prove that some piece of soft- or hardware actually meets its requirements. Typically, this task is performed by formally specifying the intended behavior using mathematical notation and verifying that the code actually fulfills the formulas. As an aside, we would like to mention that we do not focus on model checking in the text below. The reason stems because model checking is most of the time either not exhaustive, or it takes too long time. Hence, performing model checking often boils down to extensive random testing.

Thus, the work is split up in two parts: the first part describes exactly what the software should do by specifying its design. This seems rather straightforward but is actually the hardest part of the entire work. Indeed, most of the time, when we are designing complex systems from scratch, we actually do not know what we are aiming at. With hindsight, e.g. when designing software in a problem domain with which one has large experience, this is often conceived to be trivial but for new domains and without a priori insight this phase is typically very hard.

Secondly, once the intended behavior of the program is specified, one can actually prove that a piece of code actually fulfills the requirements. Most of the time, this work is not too difficult as the specification should be clear by itself, such that corresponding code matches the specification easily. Most of the tools of common use support automatic code-generation from specifications, in the case that these specifications are constructive (also called intuition-driven). This means that one never uses proofs which show that some element fulfilling some condition does exist without actually exhibiting the element itself.

Two remarks should be made at this point: both phases described here above typically go hand-in-hand, so that one never writes down a full specification without at least trying to prove some typical statements about it at the same time. Secondly, it is absolutely wrong to first design and to implement an algorithm, and only afterwards specifying it and trying to prove some properties about it. This is often an impossible task as one almost never programs in a way that lends itself to easy mathematical specification. In other terms, one should strive to an elegant mathematical formalization from the start on, instead of forcing code into such a model.

To fulfill our needs we aim at selecting an appropriate formal specification and verification tool. Here below, we provide an overview of the current state-of-the-art tools.

- PVS, the Prototype Verification System, is a freely available system of industrial strength to specify and verify soft- and hardware using a classical tableaux-style interface. It is very intuitive, both in its specification-language as well as in its prover-commands and -interface. It is programmed in LISP and can be fully extended

by your own strategies. It also allows the automatic execution of constructive specifications, although this feature is not officially supported. Everybody does it all the time, however. Please note that the code of the system's core is very large intricate, and allegedly nobody of the designers actually understands it anymore.

- HOL, one of the first higher-order-logic implementations and for years the de-facto standard for formal verification using higher-order-logics. The tool has grown very big and almost unmanageable, and most people using it do not understand the full proof-structure which was built up by the community anymore. This is not a problem, as everybody can easily let run the proof-checker which shows that the edifice in its entirety is still mathematically correct. Nevertheless, this uncomfortable situation has led part of the community to pursue new alleys, like HOL-Light. The system is programmed in Meta-Language (ML).

- Isabelle is a generic framework for designing new proof tools with new logics, and so the instantiation we are talking about in this text should actually be written out fully as Isabelle/HOL. It is quite comparable to the mainstream HOL-system, has both its advantages and disadvantages, but also suffers from obscurity by size. Typically, it takes a long time before you actually get acquainted to these tools.

- Perfect Developer aims at an easy-to-use, industrial proof tool for the masses. It does succeed to its design goals very well, although unfortunately the wider non-mathematically oriented community does not seem to value it like that. It can generate java, c# and c++ out of specifications. It is a commercial package, although for research purposes it's free of charge.

- HOL-Light [HolL11] is another member of the HOL-family of proof tools, and aims explicitly at a clear codebase and lean implementations. The current version is implemented in OCaml, and boasts beautiful packages chock-full with nice mathematical theorems which are very handy while developing your own proofs. It has famously been used for validating the floating-point arithmetic procedures of INTEL-chips, and is currently used to validate the alleged proof of the Kepler Conjecture. As it is fully reprogrammable and easy-to-read, it is not difficult to get insight in its libraries in a short time. One can easily develop in it and extract executable code, and even change its underlying logic to another equivalent logic.

As a conclusion, it is suggested to adopt HOL-Light for the task at hand. Nevertheless, the tool alone is not alone beatifying, as one should definitely start working on the specification and verification from the start on, before actually designing and implementing the algorithms. This being said, the level of specification is left up to each routing scheme.

2. Tools requirements for performance evaluation and analysis (that will be conducted in Task 3.3)

In order to evaluate the behavior (in particular, the performance) of the routing schemes through simulation, the following tools are required:

- A generator of topologies representative of the Internet. This generator should be able to generate topologies satisfying a given

set of properties, either with a proof or with high probability. Multiple topology sizes are necessary:

  - Topologies of 10k nodes in order to observe rare events that might be hidden in smaller instances.

  - Topologies comprising of the order of 100 nodes in order to compare simulation and experimental results on the same topologies and scenario.

  - Topologies ranging from 100 to 10k nodes in order to evaluate the evolution of the performances under various scale of topology growth.

- A set of tools to measure the properties of a given topology. This tool will be used for first testing if a given topology satisfies the properties it is claimed to satisfy, and second to evaluate the evolution of the topological properties under dynamic scenarios (addition/deletion of nodes/edges) and routing policy changes.

- A dynamicity scenario generator. A dynamic scenario is a list of events (topology or policy changes). The tool should allow generation of realistic scenarios in order to evaluate the quality and evolution of the above mentioned metrics (e.g., convergence time after an edge deletion).

- A dynamic routing model simulator (such as DRMSim) that given a routing scheme and topology allow to:

  - Build the routing tables starting from scratch

  - Load pre-computed routing tables and store computed one

  - Measure the performance metrics documented in this report

  - Perform experimentations with dynamic scenarios (the dynamic scenario is also an input)

  The tool should thus allow comparing the behavior of various routing schemes under the same topology and dynamicity scenario.

3. Tools requirements for emulated protocol component evaluation and analysis/ experimentation (that will be conducted in Task 4.3). Due to the dependency on the routing protocol specifics, the emulation platform, and the properties of the experimental facility, these requirements will be documented once the routing protocol components will be designed following task T2.2 outcomes and the experimental execution scenarios determined as part of task T4.2.

References

[ADK+04] E.Anshelevich, A.Dasgupta, J.Kleinberg, E.Tardos, T.Wexler and T.Roughgarden, The price of stability for network design with fair cost allocation, SIAM Journal on Computing, 38(4):1602–1623, 2008.

[Aum74] R.J.Aumann, Subjectivity and correlation in randomized strategies, Journal of Mathematical Economics, 1(1):67–96, 1974.

[Har67] J.C.Harsanyi, Games with incomplete information played by Bayesian players, Parts I-III, Management Science, 14:159–182, 330–334 and 486–502, 1967–1968.

[KB2010] N.Khanna and S.Baswana, Approximate Shortest Paths Avoiding a Failed Vertex: Optimal Size Data Structures for Unweighted Graphs, STACS 2010, pp.513-524.

[KP99] E.Koutsoupias and C.Papadimitriou, Worst case equilibria, Proceedings of the 16th International Symposium on Theoretical Aspects of Computer Science (STACS'99), LNCS 1563, pp.404–413, Springer, 1999.

[HLMEvent] A.Hamzaoui, M.Latapy and C.Magnien, Detecting Events in the Dynamics of Ego-centered Measurements of the Internet Topology, Computer Networks, 2011.

[HolL11] John Harrison, HOL Light Tutorial (for version 2.20), January 2001. http://www.cl.cam.ac.uk/~jrh13/hol-light/tutorial_220.pdf

[LMORadar] M.Latapy, C.Magnien, and F.Ouédraogo, A Radar for the Internet, Journal of Complex Systems, 2011.

[MS96] D.Monderer, and L.Shapley, Potential games. Games and Economic Behavior, 14:124–143, 1996.

[MV78] H.Moulin and J.P.Vial, Strategically zero-sum games: the class of games whose completely mixed equilibria cannot be improved upon, International Journal of Game Theory, 7:201–221, 1978.

[Nas50] J.Nash, Equilibrium points in n-person games, Proceedings of the National Academy of Sciences, 36:48–49, 1950.

[NRTV07] N.Nisan, T.Roughgarden, E.Tardos and V.Vazirani, Algorithmic Game Theory, Cambridge University Press, 2007.

[Pap01] C.Papadimitriou, Algorithms, games and the Internet, Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01), pp.749–753, 2001.

[Ros73] R.Rosenthal, A class of games possessing pure-strategy Nash equilibria, International Journal of Game Theory, 2:65–67, 1973.

[You04] H.P.Young, Strategic Learning and its Limits, Oxford University Press, 2004.