Seventh FRAMEWORK PROGRAMME
FP7-ICT-2009-5 - ICT-2009-1.6
Future Internet Research and Experimentation (FIRE)

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Deliverable D4.6:
Experimental analysis of routing schemes performance

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-D4.6 **Version**: 1.1 **Title**: Experimental analysis of routing schemes performance **Editor(s)**: D.Papadimitriou (A-LBELL) **Reviewer(s)**: W.Tavernier, D.Coudert

List of authors

Affiliation	Author
A-LBELL	D.Papadimitriou
iMinds	W.Tavernier, S.Sahhaf
INRIA	N.Hanusse, D.Ilcinkas, C.Gavoille, L.Viennot, D.Coudert
UCL	A.Kumar, J.C.Delvenne
UPC	D.Careglio
UdG	M.Camelo, L.Fabrega
UPMC	D.Bernardes, F.Tarissan

Table of Contents

1. Introduction

This deliverable documents the performance evaluation and comparison results obtained by means of simulation of the different routing models elaborated in WP2. As documented in Deliverable D4.2, we distinguish between routing model simulation and routing protocol simulation. The former avoids protocols specifics (in particular its formats) and abstracts protocol functionality (in particular its procedures) while the latter as its name indicates implements the message format and exact procedures executed by the protocol.

Deliverable D2.2 documents analysis the routing research work dedicated to i) improvements/enhancements of path-vector routing and more particularly BGP, and ii) new classes of path-based routing schemes to which geometric path routing belongs, and new routing paradigms subdivided into locator space dependent (such as geometric greedy routing) and locator space independent such as Greedy Compact Multicast Routing (GCMR) and Distributed Compact Routing (DCR). The following figure positions the different routing models/algorithms with respect to their capability to adapt to topology and policy dynamics and the distribution of the computation/decision process.

	Static	Fault-tolerant/ Incremental	Fully adaptive
Centralized	Compact routing (AGMNT)		
Distributed (synchronous/ bounded delay)	Distributed Compact routing (AGMaNT)	Greedy geometric routing /GTR (fault-tolerant)	
Distributed (asynchronous)	Distributed Compact routing (AGMaNT)	GCMR (both) Geodesic path routing/GPV (both)	BGP

The main observation out this analysis together with the feedback gathered from the project technical advisory board which can be summarized as follows: ***performance improvement alone is not a sufficient condition for migration to a new routing protocol (assuming that routing protocol would exist) but functional preservation (if not improvement) is a necessary condition*** leads us to reconsider the routing schemes initiated in the context of WP2 activities and propose geometric path routing (a.k.a. geodesic path routing) as building block of information routing in addition to ii) geometric greedy routing (and the geometric tree routing variant investigated in the project), iii) distributed compact routing (DCR) and the variant investigated in this project (AGaMNT), iv) greedy compact multicast routing (GCMR) and v) a scheme which combines a variant of the search-based discovery process together with the exploitation of the node degree to steer the search process. Note that route discovery performed by means of a search process imply selection enforcement on the reverse path towards the source/initiating node.

The design of the routing schemes (ii), (iii) and (iv) have already been reported in deliverable D2.2 together with their main limits; primarily for

what concerns adaptivity to topology dynamics and policing (meaning the capability to take routing decisions based also on arbitrary parameters (and their order) in addition to parameters qualifying the intrinsic spatio-temporal properties of the route). In this document, we focus on i) geometric/geodesic path routing (referred to as GPV) used as building block for geometric information routing as documented in [PAP2013], ii) distributed compact routing, more precisely a distributed synchronous version of AGMNT scheme and a variant relying on exploration process for path setup, and iv) the performance analysis of the multicast routing scheme developed in the project (referred to as GCMR).

2. Simulation experiments : goals, relevance and relationships to emulation

Although Internet-wide routing protocols already exist for a couple decades, research efforts in designing and evaluating such protocols largely relies on simulation. The use of simulation for evaluating routing schemes in large-scale networks with similar characteristics as the inter-AS topology stems fundamentally from the need to verify they perform “much better” than existing routing protocols, namely the Border Gateway Protocol (BGP), with an equivalent if not additional level of functionality. Indeed, simulation experiments are carried to determine the performance of a set of routing models abstracting the realization of the functionality under execution; thus providing the conditions to increase the spatial scale at which these executions can be conducted.

The EULER iterative cycles of experimentation combines simulation and emulation experiments that play a complementary role. Simulation experiments by abstracting protocol procedures and data structures, enables to evaluate and compare the spatial performance of routing models and algorithms (including, e.g., stretch of routing paths, number of routing table entries, communication complexity). Moreover, by means of simulation experiments their behavior (e.g., scaling, adaptivity metrics) can be evaluated up to large-scale graphs. Simulations (in particular, simulation by discrete event) is well suited for experiments involving spatial metrics, structure, and dimensions as simulation enables handling of large-scale topologies and produces results that are easier to tune, reproduce, and compare. When the experimental scenarios are commonly specified and their execution adapted (without introducing any bias) to the execution environment emulation experiments complement simulation results by focusing on trends and sensitivity of time and resource consumption of corresponding routing components under different conditions, and supplements simulation results by validating (when possible) the performance and behavior metrics.

More specifically, the goals for performing simulation experiments are threefold:

1. Evaluate the performances of the distributed routing models given a set of pre-defined functionality as well as behavioral properties (adaptivity, scalability, etc.);

2. Compare performance results (the use of a common simulation tool, DRMSim developed by the project, aims at facilitating the achievement of this goal) in particular against BGP/path-vector routing;

3. Eliminate models whose performances are not satisfactory when running on Internet-like graphs and/or modify the corresponding algorithm (and provide feedback to WP2). Those satisfying the performance criteria documented extensively in Deliverable D4.1 can be further elaborated by specifying the corresponding protocol design to be evaluated by protocol simulation and/or by emulation, i.e., by means of a protocol engine running on a soft-routing platform (cf. Deliverable D4.5).

3. Common simulation setup (topology and traffic) and working assumptions

3.1 Topology

The various capabilities offered by the simulation environment in terms of dynamics of the topology (generated either by Grph an efficient portable graph library tailored to network simulation and graph analysis or by an external generator, or even topologies imported from external source such as CAIDA maps) can be specified at simulation configuration time.

We can consider two types of Internet topologies: AS and router levels can be described by graphs sharing specific properties: the hop-diameter is small (around 10), the distribution of degrees follows a power law: the number of nodes of degree k is roughly k^{-c} where, c a small constant. It has been observed that c is between 2 and 3 in any internet map known so far (CAIDA, DIMES, etc). Such a graphs have 40K nodes and around 100K links for the AS level and should be multiplied by 10 for the router level.

Topology dynamics scenarios involve various link/node failure patterns. The difficulty consists in providing realistic patterns of failures of topologies comprising O(100k) links and O(10k) nodes that have no identical birth time. Despite its popularity, and wide applications, the traditional 2- or 3-parameter Weibull distribution is unable to capture the behavior of a lifetime data set that has a non-monotonic failure rate function. For this reason, many aging distributions were proposed to overcome this deficiency. Following the fundamental relationship between the reliability function $R(t)$ and its corresponding cumulative failure rate function $R(t) = exp(-a\,H(t))$ with $a > 0$ many generalized Weibull models have been proposed in reliability literature. With a suitable choice of $H(t)$, and its parameter, we can obtain a bathtub shaped failure rate distribution. In many practical applications, $H(t) = -d_t(\ln(R(t))$ is initially decreasing, followed by a period of approximately constant hazard, and ultimately increasing because of the eventual positive aging effect. This pattern is also observed for "physical links", after installation links show newborn type of failures (resulting from misconfiguration or manufacturing problems), afterwards link remain subject to constant failure until reaching their end of life period.

The following table provides an overview of the reliability functions $R(t)$ that have been proposed for some common generalized Weibull distribution function where IFR/DFR stands for increasing/decreasing failure rate and (M)BT for (Modified) Bath Tube ((M)BT)shape. Most generalizations or modifications of Weibull distributions listed in Table 1 are further detailed in D.Murthy, M. Xie, and R. Jiang, Weibull Models, Wiley).

Table 1: Reliability functions

Author	Reliability function	Characteristics
Gompertz (1825) [7]	$R(t)=\exp\left\{\frac{\theta}{\alpha}\left(1-e^{(\alpha t)}\right)\right\}$, $\theta>0,-\infty<\alpha<\infty;\ t\geq 0.$	IFR if $\alpha>0$ DFR if $\alpha<0$
Weibull (1951) [30]	$R(t)=\exp\left(-\lambda t^{\alpha}\right), \lambda,\alpha>0;\ t\geq 0.$	Exponential. if $\alpha=1$ IFR if $\alpha>1$ DFR if $\alpha<1$
Smith & Bain (1975) [28](Exponential power model)	$R(t)=\exp\left\{1-e^{(\lambda t)^{\alpha}}\right\};\alpha,\lambda>0;t\geq 0.$	BT if $0<\alpha<1$ Special cases of Chen's model
Xie & Lai (1995) [33]	$R(t)=\exp\left\{-(t/\beta_1)^{\alpha_1}-(t/\beta_2)^{\alpha_2}\right\}$ $\alpha_1,\alpha_2,\beta_1,\beta_2>0;\ t\geq 0.$	IFR if $\alpha_1,\alpha_2>1$ DFR if $\alpha_1,\alpha_2<1$ BT if $\alpha_1<1,\ \alpha_2>1$
Chen (2000) [5]	$R(t)=\exp\left\{-\lambda\left[e^{t^{\beta}}-1\right]\right\}, \lambda,\beta>0;t\geq 0$	BT if $\beta<1$ IFR if $\beta\geq 1$ Exponential power if $\lambda=1$
Pham (2002) [26]	$R(t)=\exp\left\{1-a^{t^{\alpha}}\right\};\alpha,a>0;t\geq 0.$	DFR for $t\leq t_0$ IFR for $t\geq t_0$ where $t_0=\left(\frac{1-\alpha}{\alpha\ln a}\right)^{\frac{1}{\alpha}}$
Xie, Tang & Goh (2002) [34]	$R(t)=\exp\left\{\lambda\beta\left[1-e^{(t/\beta)^{\alpha}}\right]\right\}$ $\alpha,\beta,\lambda>0;\ t\geq 0.$	Chen's model if $\beta=1$ IFR if $\alpha\geq 1$ BT if $0<\alpha<1$
Lai, Xie & Murthy (2003) [13] Gurvich, Dibenedetto & Rande (1997) [8]	$R(t)=\exp\left\{-at^{\alpha}e^{\lambda t}\right\}$ $\lambda\geq 0,\ \alpha,a>0;\ t\geq 0.$	Weibull if $\lambda=0$: Exponential if $\beta=0,\alpha=1$ IFR if $\alpha\geq 1$ BT if $0<\alpha<1$
Nadarajah & Kotz (2005) [20] †	$R(t)=\exp\left\{-at^{b}\left(e^{ct^{d}}-1\right)\right\}$ $a,d>0;b,c\geq 0;\ t\geq 0.$	Xie, Tang & Goh's model if $b=0$ Chen's model if $b=0,\ c=1$
Bebbington, Lai & Zitikis (2006) [3]]	$R(t)=\exp\left\{-\left(e^{\alpha t}-\beta t\right)\right\};\alpha,\beta>0;t\geq 0.$	IFR iff $\alpha\beta>(27/64)$ MBT if $\alpha\beta<(27/64)$

We use the following process in order to generate "failure dynamics" at each interval when new links are added a failure profile is associated such that we statistically observe a number of failures per time interval. Determining which of the distribution provides the best fitting Bath Tube (BT) shape remains to be determined. Moreover, we assume that after failure a certain time (MTTR) is used to model restoration of the corresponding link. Again there is no known model enabling to characterize this distribution we can though make use of the non-parametric technique developed in context of task 3.2 to estimate the repair time.

Another scenario is the incremental addition of nodes and links. It is important to emphasize that evolutive topology generators are capable to reproduce long term evolution of the topology but unable to reproduce this evolution from short-term / step-by-step evolution perspective. Elements driving the growth of the topology are also determined by the increasing requirement to increase the number of tangential links (transit peering links) in order to improve resilience at lowest shared cost. Remember that even if the number of nodes and links increases, the global properties of the topology in terms of diameter of the graph, characteristic path length and average path length varies logarithmically in the number of nodes.

Hence, BGP simulations experience the same trends as BGP itself in terms memory and processing consumption. Centralization of the routing table entries offers from this perspective a promising approach that would enable simulation of BGP execution on topologies comprising up to 50k nodes.

3.2 Traffic

As presented in the Deliverable D4.2, an experiment consists in simulating or emulating a running network routing according to a traffic (which nodes sends data to with others and when), the question addressed in this section consists in proposing realistic models for a traffic.

It is worth noticing that, as mentioned in the Deliverable D4.2, the current state-of-the-art of generating traffic is relatively poor, at least compared to the effort made by the community as regard models for generating realistic topologies. The usual traffic test assessed in emulation environment consists in analyzing how the network respond to specific simple scenarios, such as one nodes sending traffic towards all other nodes or even all nodes sending traffic towards all other nodes. Such scenarios are usually completed by some stress test scenarios in which specific nodes experience some burst in the traffic (see Deliverable 4.2 (section 3.4) for more details). Although very interesting in the context of EULER since it enables to confront the proposed routing paradigms to worst case scenarios, such simple scenarios do not account for realistic traffic and one might thus miss some specific properties difficult to generate by all-to-all traffic. For instance, it is clear that it is not possible to reproduce the correlations one might observe between the different traffic triggered by specific nodes during short time-windows.

The point of view adopted in the project and reported in Deliverable D2.1 consists in relying on models of traffic demands applied on an overlay network and then derive the induced traffic on the real network. The principle is illustrated in Figure 3.1:

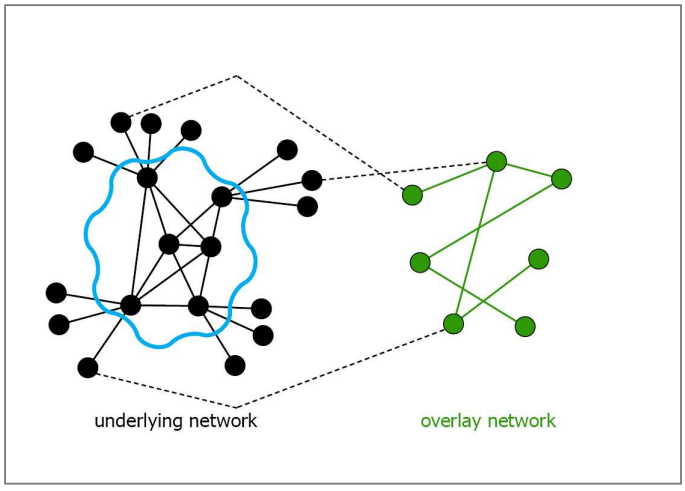

Figure 3.1: Traffic demands in overlay network

In this figure, the overlay network is represented in green on the right. On the left is the real underlying network on which the communication takes place. This figure explicates how the nodes in the overlay network coordinates in order the retrieve the information of which peers are exchanging information. Once this is done the actual communication occurs and this triggers traffic on the underlying network. This is done by mapping each green node to a real end-host black node of the real network. One can see in particular on the picture that the interaction between two neighbors in the overlay network may in turn triggers complex traffic demands on the real underlying network.

The following sections report on the proposition of realistic model for traffic demands. In order to validate the proposed model and compare the relevance of the approach compared to previous models, we tested the model on the P2P dataset (see Deliverable D3.2).

It is important to emphasize that the P2P dataset has been considered for the validation of the model. It does not mean however that the model only works for reproducing traffic generated by P2P activities (which yet account for a non negligible part of the traffic in Internet). The parameters are indeed tunable so one can use the model for different scenarios. In particular, the two parameters p and l which account for the intensity of the traffic (see Section 3.1.4 for further details) allows to propose different kinds of traffic demands.

3.2.1 Simple model for traffic demand

In a precedent work, reported in Deliverable D4.2, we provided a traffic generator able to mimic spreading effects observed in the context P2P systems. The model relied mainly on three parameters:
- An underlying (and static) network on which file diffusion occur
- A parameter $p \in [0,1]$ characterizing the amount of traffic
- A simulation bound standing for the number of step in the simulations.

Although the proposed model has proved its capability to reproduce qualitatively the properties observed in real dataset, it failed in reproducing quantitatively real values of different metrics associated to diffusion phenomena. We then investigated natural extensions to the model, which explored the key properties found in the data. We have examined improvements both in terms of the spreading dynamic (heterogeneous models according to peer behavior or file popularity) and in terms of the underlying network structure and found they did not bring, separately, major changes in the shape of the simulated cascades. These results combined weaken the case for the simple model as a pertinent spreading dynamic in the context of real-world diffusion, particularly in the case of P2P systems. The complete account of these findings was reported in deliverable D3.4 and published in [BER2013].

The main reason explaining the gap between simulations and real data stems from the fact that the model did not account for several aspects of temporal properties associated in such systems. To improve this point, we propose to integrate the interaction time directly into the model, namely transforming the original static overlay network into a dynamic overlay network. In this way, the spreading impact of transient nodes would be significantly diminished compared to the presence of mode nodes with a more steady presence in the network. Evidently this significant change in the network presupposes, first, the connection times of each node and secondly an adapted spreading model which would evolve in the order of seconds – that is in “real” time, as opposed to an intrinsic simulation time. Indeed, the spreading process is supposed to interact with the network, taking into account the nodes and links present in the system at time $t > 0$ measured in seconds and the process is supposed to evolve in the same time scale.

3.2.2 Integrating time patterns

In order to integrate real time, we decided to abandon the simple SIR and use instead a SI model with a latency between the time a node possess a file and the time it propagates such file to each of its neighbors in the overlay, namely the *inter-contagion time* (ICT). Node latency given by exponential time is a common assumption and was proposed previously in the context of P2P file sharing systems [LEI2006]. The ICT is characterized by a rate or, alternatively, by the mean (expected) ICT, since in the case of exponential random variables the mean time is the inverse of the rate. Moreover, if a peer P possesses the file F, the number of peers who received the file F from P (after P obtained it) is a Poisson process characterized by the inter-contagion time rate (or the mean ICT). We will examine SI models with homogeneous and heterogeneous inter-contagion time. In other words, in the first case we suppose all nodes have the same spreading behavior (global ICT rate) and in the second, an individual one (a different ICT rate for each node).

The introduction of contagion model featuring inter-contagion times allows us to adjust the simulation in terms of the chronological time (in seconds), as we observe in the diffusion trace. This represents a key contrast to the spreading models reported in former deliverables, whose evolution happened in a simulation intrinsic time (given by the number of steps in the algorithm). This is precisely the reason why we had to hold one cascade property constant and analyze the remaining properties: in this way we would have a comparable set of cascades with respect to a property (see Deliverable D3.4 for details). In contrast, the time bound of the simulations using the model presented above is given in seconds, so there is a more straight-forward and natural way to obtain a comparable set of simulated cascades: we impose the same time scale observed in the diffusion trace to the simulated cascades. That is, we simply simulate the diffusion of the cascades up to the time T (last time observed in the trace) and compare the three key properties of the simulated cascades to the corresponding real ones.

Another strategy to use temporal information is to rely on a dynamic overlay network. In such a network, two peers will be connected at time $t > 0$ if they are neighbors (as in a static overlay network) and if they are both online at time t. Intuitively, the dynamic overlay network is built similarly to the original overlay network, but evolves with the addition/suppression of connecting/disconnecting nodes and the respective links between these nodes and their neighbors.

3.2.3 Simulations

We have simulated the SI model with homogeneous and heterogeneous spreading behavior as outlined above on the dynamic overlay dynamic graphs for each file present in the trace. The profiles of real and simulated cascades are summarized by plotting the complementary cumulative distributions of cascades size (Fig.3.2), cascades number of links (Fig.3.3) and cascades depth (Fig.3.4). For each cascade property, we plot the same distribution in lin-log and log-log (inset) scales, which highlight respectively smaller/short cascades (most cascades) and bigger/deeper cascades (rare cascades).

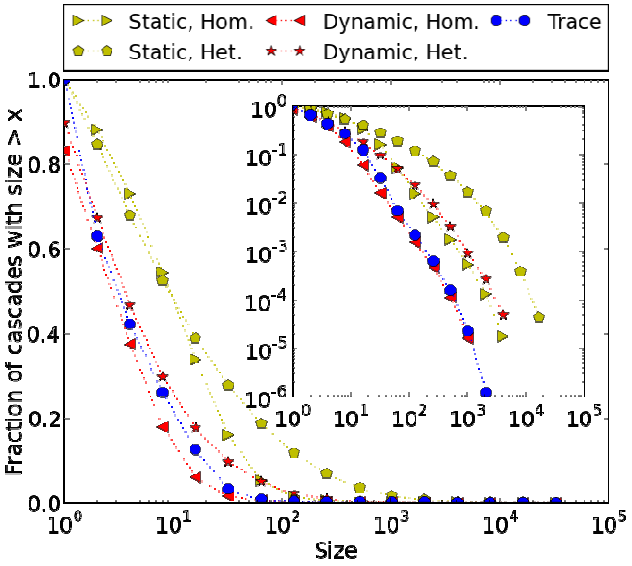

Fig.3.2: CDF of cascade vs. cascade size

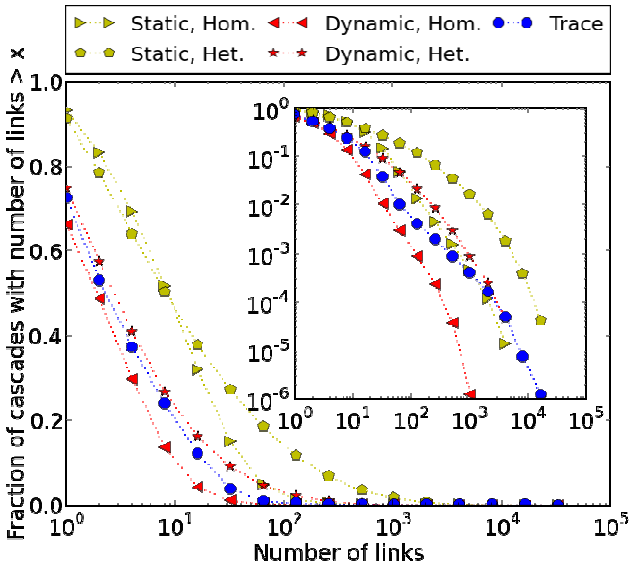

Fig.3.3: CDF of cascades vs. number links

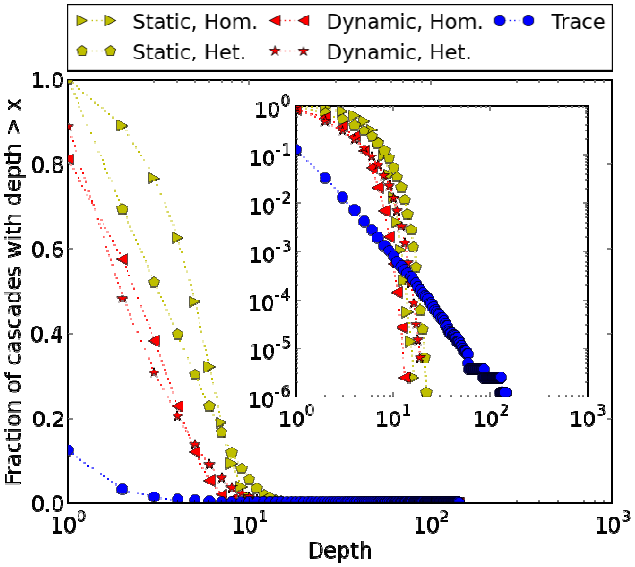

Fig.3.4: CDF cascades vs. cascade depth

In terms of the variations examined, the dichotomy static vs. dynamic graph changes has an overall impact, but affects particularly the distribution of trivial cascades. Compared to simulations on the static graph, simulations on the dynamic graph yielded a bigger proportion of small cascades which is what we observe in out measurements. The dichotomy homogeneous vs. heterogeneous SI model impacts mostly the properties' distribution tail, particularly in terms of size and number of links. All other things equal, in our setting, the homogeneous SI model yielded simulations with smaller proportion of large cascades, which is closer to the ob- served cascades in terms of size, but more distant in terms of number of nodes. In terms of depth distribution we note that none of the proposed models was able to reproduce the scale-free depth distribution featured by the real cascades: simulated cascades exhibit, in contrast to real ones, a sharp decrease in the proportion of cascades with depth greater than 10, revealing a cutoff.

In summary, in terms of size and number of links, we find encouraging results: both homogeneous and heterogeneous models perform relatively well in the dynamic setting, in the sense that simulations on the dynamic graph feature a proportion of small cascades similar to the real ones (most cascades). In terms of larger (and infrequent) cascades, the homogeneous model reproduces well the size distribution of real cascades; in terms of number of links, the heterogeneous model is superior. Although this model cannot generate artificial cascades similar to real ones in terms of all key properties, we have shown the importance of taking into account the temporal data in contagion models which aim to generate realistic cascades.

3.1.4 Summary

In this section, we have improved the model for diffusion processes, integrating the notion of inter-event times. In contrast to the models from previous work, this model adds latency in the information spread for each node of the overlay network. In order to validate the relevance of the new model, we have estimated the parameters for this model using the static and the dynamic settings for the overlay network, supposing that peers have an

homogeneous and heterogeneous behavior regarding this latency time. The proposed model now relies on the following parameters:

- An underlying dynamic network on which file diffusion occur
- A parameter p $\in [0, 1]$ characterizing the amount of traffic
- A parameter l standing for the inter-event time (expressed in seconds)
- A parameter T standing for the simulation duration (expressed in seconds)

Once traffic demands have been generated according to the model, it remains only to derive the traffic induced on the real network by mapping each node of the overlay network triggering the traffic to its corresponding node in the real network. This work has been submitted in [HOL2013] and is part of the PhD thesis [BER2014].

4. Path-Vector Routing

The Internet scale increases in terms of the number of (abstract) nodes, edges, and destinations at a rate ranging from 10 to 15% per year. These steady increases impact the Internet routing system and its underlying inter-domain routing protocol, i.e., Border Gateway Protocol (BGP). This path-vector routing protocol requires local storage and timely update of the routing states as it belongs to the class of adaptive stateful routing. Consequently, BGP progressively reaches the objective performance limits of shortest path routing (memory consumption) and adaptive routing (timely adaptation of routing states). Many research initiatives and the EULER project in particular (have) explore(d) novel dynamic routing schemes expectedly adapted to the Internet's short- and long-term evolution. These dynamic routing schemes are designed to meet the fundamental trade-off between memory space, routing path stretch and adaptation cost (communication and computational complexity). However, their evaluation faces a fundamental problem: how to verify they actually perform better than BGP.

Moreover, it becomes progressively clear as the project routing algorithmic work matures that functionality such as policing without explicit policy information exchanges (BGP does not exchange policy information) and network-wide metric (BGP performing a node-base path selection process on per-path basis) is extremely difficult to achieve without a routing model capable to exchange and process path-information units. This observation adds to the fundamental need to perform in-depth comparison with BGP/path-vector performance but also its functionality. BGP simulation can be realized without any route aggregation whereas flexibility provided by CIDR (remember that BGP has been the first routing protocol being CIDR capable) allows to reduce the communication cost and all related measures.

4.1.2 Simulation Objectives

The objective of simulating the execution of path vector routing model such as BGP is i) to calibrate the simulator (BGP being a stretch-1 routing scheme), ii) provide a comparative reference against the routing schemes developed in the context of the project, and iii) provide an upper bound in terms of simulation scale (as BGP determines the upper bound in terms of memory space consumption).

The performance analysis includes the following performance metrics:

- (Multiplicative) stretch of a routing algorithm: the maximum, over all source destination pairs (u, v) of the ratio between the routing path cost (or routing distance) from node u to v, $(u, v \in G)$ and the topological path cost (or topological distance) from node u to v, $(u, v \in G)$. The additive stretch being defined as the difference between the routing path cost (or routing distance) from node u to v, $(u, v \in G)$ and the topological path cost (or topological distance) from node u to v, $(u, v \in G)$.

- Memory complexity (in bit-space): the memory space required to store the routing information used by the routing algorithm (input of the routing algorithm) and the memory space required to store the routing tables (output of the routing algorithm).

- Communication complexity
  - in bit-message: the total number of routing update/information messages exchanged between nodes (along the edges of the graph) for the local computation of the routing table entries by the routing algorithm.
  - in time: the difference of time units between the first emission of a message and the last reception of a message during any execution of the routing algorithm assuming the slowest message uses one time unit to traverse an edge.

- Computational complexity in time (or time complexity): quantifies the amount of time taken by the routing algorithm to run as a function of the input size. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, where an elementary operation takes a fixed amount of time to perform.

4.1.3 Simulation and Results

In order to obtain representative and comparable results, we consider the following scale-free topologies characterized by a small diameter growing proportionally to $log(n)$, heterogeneous degree, high degree nodes are central:

- GLP topology generation model (16k and 32k nodes)
- CAIDA map (16k and 32k nodes)

The simulation of BGP should lead to the following results:

- Stretch: in absence of policing, the multiplicative stretch of path-vector routing algorithm equals to 1.

- Memory space consumption: assuming the encoding is proportional to $log(n)$, the memory space required per node to store the routing table (output of the path vector algorithm) is proportional to $n.\Delta(\mathrm{G}).log(n)$ where, n is the number of nodes and $\Delta(\mathrm{G})$ is the diameter of the graph G; the memory space required per node to store the routing information (input to the routing algorithm) is $O(n(n-1).\Delta(\mathrm{G}).\log(n))$.

- Communication complexity (in bit-message): the total number of routing information messages (input to the routing algorithm) received by each node is $O(n(n-1))$;

- Communication complexity (in time): in absence of policing, the communication complexity in time is $\Omega(\Delta(\mathrm{G}))$.

5. Geometric Routing

Besides the base performance metrics common to any routing scheme (stretch, memory cost (in terms of bit-space), communication and processing cost), geometric routing involves two additional metrics. The first is succinctness or the number of bits required to encode coordinates. The second metric is the success rate as the coordinate assignment may lead to traffic blackholes (packet following trajectory along local minima).

5.1 Geodesic Geometric Routing

Instead of assigning (virtual) coordinates and compute distances from these coordinates as in geometric greedy routing, the geodesic geometric routing operates by computing the distances between vertices. This computation is carried out on the coordinates drawn out of the negatively curved geometric space (the hyperbolic plane) that are processed as distance label at intermediate nodes performing local routing decisions. The theoretic foundations underlying the geodesic geometric routing approach (also referred to as GPV) are derived from [BRI2009].

5.1.1 Preliminaries

We first introduce the base notions used throughout this section. Let X be a set and d a positive definite metric. A metric space (X,d) is geodesic if any two points $x,y \in X$ can be joined by a geodesic (segment) of length $d(x,y)$. A geodesic segment is defined as a (not necessarily continuous) map γ from the closed interval [a,b] of length $|a-b|$ to X such that $\gamma(a)=x, \gamma(b)=y$ and $d\big(\gamma(s),\gamma(t)\big)=|s-t|,\ \forall\ s,t\in[a,b]$. In particular, $d(\gamma(a),\gamma(b))=|a-b|=d(x,y)$. Further, a metric space (X,d) is δ-hyperbolic if for any four points $u,v,x,y \in X$ the two larger of the three distance sums $d(u,v)+d(x,y)$, $d(u,x)+d(v,y)$, $d(u,y)+d(v,x)$ differ by at most $2\delta \geq 0$.

An undirected connected graph $G=(V,E)$ equipped with a distance metric d_G can be transformed into a geodesic space (X,d) by replacing every edge $e=(u,v)\in G$ by a segment $[u,v]$ of length $|v-u|=1$ that may only intersect at common ends. Let (X,d) be such geodesic space. In turn, a connected graph G equipped with a distance metric d_G is δ-hyperbolic if the corresponding metric space (V,d_G) satisfies the four-points condition. Intuitively, the δ -hyperbolic property of a graph measures its deviation from tree-likeness (obtained when δ = 0). As the value of delta increases, the deviation of the graph from a tree structure is more marked. Negative curvature $(\kappa \leq 0)$ of hyperbolic spaces and δ-hyperbolicity are related by the following equation: $\delta = ln(1+\sqrt{2})/\sqrt{-\kappa}$. We refer to the Deliverable D3.3 for further details concerning this property.

5.1.2 Description

Geodesic geometric routing relies on the assignment to each vertex v of the graph G of a coordinate (sometimes referred to as geo-locators) taken out of a metric space (X_G,d). In the present case, the metric space (X_G,d) is hyperbolic, i.e., negatively curved, following the definition provided in Section 5.1.1. This selection is not arbitrary as it closely follows the geometric properties of scale-free graphs. Following this coordinate assignment, the distance $d_G(x,y)$ between any two vertices x and y of the graph G can be determined or estimated by inspecting the coordinates assigned to nodes x and y, in particular at vertex x. At this level thus, the routing scheme performs similarly to a distance labeling schemes which

label the vertices of a graph with short labels (in the present case succinct coordinates) in such a way that the distance between any two vertices x and y can be determined or estimated efficiently by inspecting the labels of x and y, without using any other information.

The routing scheme performs as follows:

1. It first determines the ball $B(x,\rho)$ of radius ρ centered at node x such that this ball defines a ρ-geodesic space of negative curvature $\kappa < 0$. This operation ensures that the negative curvature condition of the topology is locally verified. As we will see here below, this condition is required to guarantee that quasi-geodesics remain close to geodesic paths.

2. Then, provided that the edges of the graph are properly weighted (this weight translates the distance between node pairs), the routing algorithm selects the next-hop neighbor along geodesic trajectories. However, as a routing path along the optimal trajectory derived from the coordinate(s) assigned to each vertex may not necessarily exist in the underlying topology (reaching such vertex leads to the so-called lake or river by-pass). Hence, the set of possible routing paths that can be selected in order to ensure any-to-any connectivity requires to be extended beyond the minimal set of optimal or geodesic paths. This implies that the routing algorithm along with the routing information discovery process detailed in point 4 performs its selection among the set of available near-optimal paths, referred to as quasi-geodesics. This scheme assumes thus that a trajectory can be found in the underlying graph that corresponds to (λ,ε)-quasi-geodesics. In a metric space (X,d) a quasi-geodesic is defined as a (λ,ε)-quasi-isometric map γ from the closed interval $[a,b]\to X$ such that there exist constants $\lambda \leq 1$ and $\varepsilon \leq 0$ verifying that $\forall\, s,t \in [a,b],\ \lambda^{-1}.|s-t| - \varepsilon \leq d\big(\gamma(s),\gamma(t)\big) \leq \lambda.|s-t| + \varepsilon.$ For this purpose, the routing algorithm provides (i) local detection of quasi-geodesics and (ii) mean to ensure that the quasi-geodesics remain close to geodesics for the type of metric spaces under consideration. For the latter (ii), we rely on Theorem III.H.1.7 of [BRI2009] which proves that $\forall\, \delta > 0,\ \lambda \geq 1,\ \varepsilon \geq 0$ there exists a constant $R = R(\delta,\lambda,\varepsilon)$ such that if γ_{uv} is a geodesic and γ'_{uv} is a (λ,ε)-quasi-geodesic joining the same end points u,v of the hyperbolic geodesic space X, then $d(\gamma_{uv}, \gamma'_{uv}) < R(\delta,\lambda,\varepsilon)$; in other words, this theorem proves that quasi-geodesics follow closely geodesics if the metric space is hyperbolic.

To fulfill the first condition (i), the routing algorithm performs local detection of geodesic segments. For this purpose, the routing algorithm relies on the detection of k-local geodesics, notion introduced in Theorem III.H.1.23 of [BRI2009]. A path $\gamma:[a,b]\to X$ is said to be a k-local geodesic if $d(\gamma(t),\gamma(t')) = |t-t'|, \forall\, t,t' \in [a,b]$ with $|t-t'| < k$. For the class of geometric graphs (i.e., metric spaces) under consideration, our simulation results confirm that for graphs that are quasi-isometric to trees this value can be limited until reaching $k = 2$. This means that geodesics can be detected locally at least two-hop away from the localizing node. In addition, the stability property of quasi-geodesics (following Theorem III.H.1.8 of [BRI2009]) implies that routing decisions are robust and offer the possibility to consider alternate (quasi-)geodesic path(s) to ensure destination reachability.

4. The routing information exchanged comprises the "vector of distance" to reach the "coordinate" allocated to each vertex. This coordinate is processed as distance label at intermediate nodes performing local routing decisions. For this purpose, one can distinguish two exchange modes for

what concerns the routing information discovery process: the pull mode and the distributed mode. In the pull mode, an oracle keeps tracks of the paths (segments) between pairs of vertices $u,v \in V$ and their associated distances. In the distributed mode (or push mode), each node propagates the distance to its neighbor's coordinate (similarly to the distance vector algorithm). The difference being that instead of propagating the "selected" neighbor's distance to the (set of) destination, the vector of distances is pushed until reaching k-hops (that is every node keeps a vector of distances to all nodes distant from k-hops). At the local level, this schemes operates thus similarly to a distance vector (per destination t) where the distance along a single geodesic path segment is communicated to the preceding node until reaching boundary nodes (or edges) of its neighborhood.

5. At the network-wide level, geodesic geometric routing operates as follows: edges of each partition propagate the set of all contiguous coordinates they can locally reach (being the collection of all destinations reachable via k-local geodesics) together with the minimum number of partitions to reach them. We refer to the set of contiguous coordinates as an area. In other terms, the distance metric at the network-wide level translates the number of geodesic path segments to be traversed from the source to the destination area. Routing end-to-end from source $s, s \in B(u,\rho)$ to destination t, $t \in B(v,\rho')$ runs then as follows: node s uses the geodesic path segment $[s,x]$ to the edge x of its local partition; node x then selects the next-hop neighbor y $[x \to y]$ where, y is the head-end of the geodesic path segment $[y,t]$ in the next partition towards destination t. The algorithm performs by avoiding multiplicity of paths to the same geo-locator (coordinate) though it shares some properties in common with path-segment routing.

5.1.2 Simulation Results

1. Stretch

In general, the stretch of the routing path from any source x to any destination y produced by an approximate distance or routing labeling scheme in a graph G is determined by the formula $\lambda \cdot d_G(x,y) + \varepsilon$. For simplicity, $(1,\varepsilon)$ -approximate distance or routing labeling schemes are called ε-additive labeling schemes, and (λ,0)-approximate distance or routing labeling schemes are called s-multiplicative labeling schemes.

The geodesic routing scheme exploits the geometric property of the graph by mapping (λ,ε)-quasi-geodesics from each vertex to the (set of) coordinates (contained within each non-local ball). The stretch of this routing scheme is $(\lambda = 1, \varepsilon = \delta'.h.(k-1))$ additive. The first term translates the maximum deviation to reach the destination along h geodesic segments of length at most δ each. The second term the maximum deviation inside each ball along the corresponding routing path. For δ-hyperbolic graphs, quasi-isometric to trees, i.e., $k = 2$. Moreover, Deliverable D3.3 produced by WP3 shows that the δ-hyperbolicity of the graph underlying the Internet topology ranges between 2 and 2.5, leading to an additive stretch depending on the number of geodesic segments. The important finding in this respect is that for such values of δ, the deviation of the end-to-end routing path from geodesics is the smallest. For $\lambda = 1$, we obtain following Fig.5.1 a maximum additive stretch of 4.54, and 5.59 for $\delta = 2$, and $\delta = 2.5$, respectively. Moreover, the density of geodesics per unit length is the highest for such

values of λ as shown in Fig.5.2; enabling usage of ranking-function (for policing purposes) and availability of alternate paths (in case of failure).

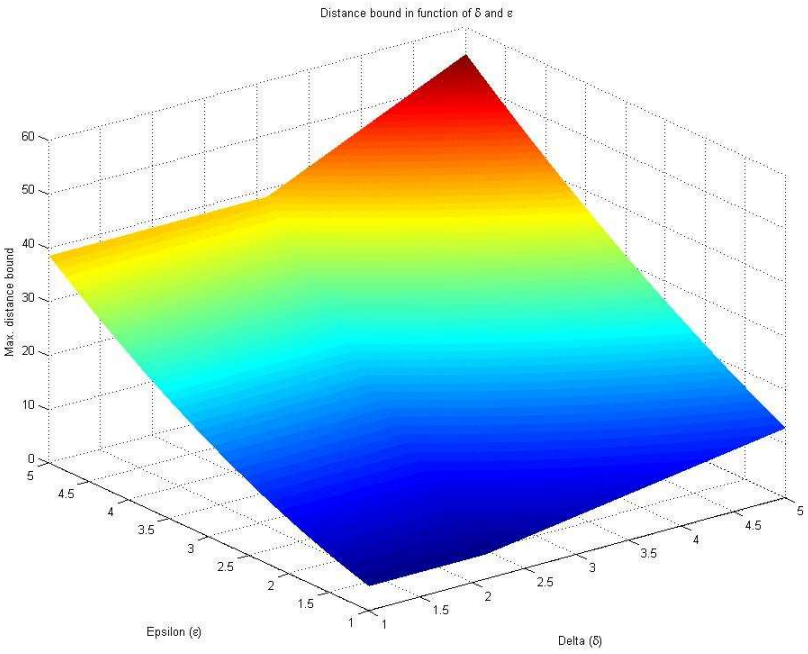

Fig.5.1: Distance bound vs. epsilon (ε)

Results reported in [TR2014] using per-AS router-topology confirm nevertheless that the value of $k = 2$ may not be reached. The consequence being that either a higher safety margin is to be considered ($k = 3$) or one need to enforce a local tree routing structure.

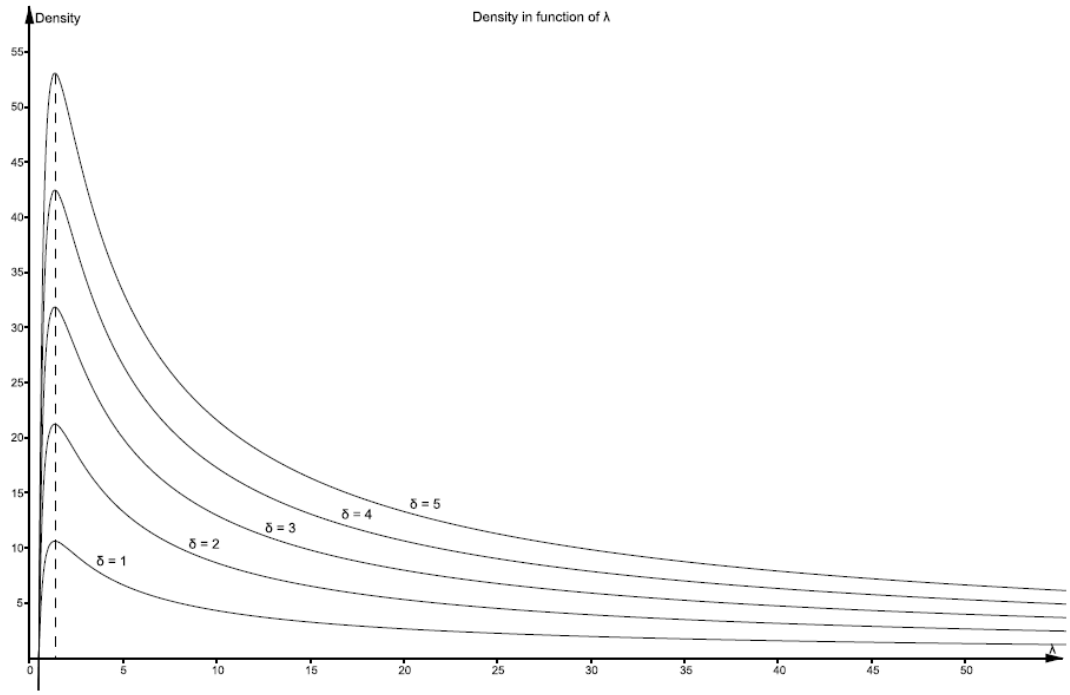

Fig.5.2: Density of quasi-geodesics per unit length

In comparison, if the hyperbolicity of an n-vertex graph G is δ ≥ 1/2, then G admits an additive $O(\delta.log(n))$ spanner with at most $O(\delta.n)$ edges, and linear time construction of distance approximating trees with an additive error $O(\delta.log(n))$. Consequently, such graphs admit a $\delta.log(n)$ -additive routing labeling scheme which uses $O(\delta.log^2(n))$ bit labels and performs routing decision in $O(log_2(4\delta))$ time. It also supports a $\delta.log(n)$ -additive distance labeling scheme which uses $O(log^2(n))$ bit labels and constant time distance decision. In general, the closer the values of δ to 0, the lower the increase of the routing path stretch. In other terms, the stretch gain trades against memory increase as each node maintains an association between the distance derived from the header and the next-hop to the corresponding routing.

2. Number of routing states and Memory

Since every node keeps a vector of distances to all nodes distant from k-hops and a vector of distance to each set of coordinates (areas), in the worst case, the number (active) routing states is $O(\sqrt{n}(1+\varphi))$ assuming $\sqrt{n}$ areas are defined. The parameter translates the forgetful property of this scheme which discards alternate routes; if a single route is kept per destination, $\varphi = 1$.

Assuming that each geodesic path segment can be represented by a succinct coordinate pair, and the port encoding is proportional to $log(n)$, the memory space required per node to store the routing table (output of the geodesic routing algorithm) is $\tilde{O}(\sqrt{n})$ memory bits where, $\tilde{O}(f(n)) = f(n)log(n)^{O(1))}$ to hide poly-logarithmic factors.

3. Communication cost

If each node advertizes the set of coordinates then the communication cost is equivalent to the one obtained for distance vector routing, i.e., $O(n.m)$ If each coordinate set (or area) is advertized by a single entity then $O(|B|.m)$ where, B determines the number of coordinate areas. The main drawback being that the advertisement of destination coordinates do not allow deriving their reachability. Thus, requiring an explicit withdraw message in case a set of coordinates becomes unreachable (similar to the processing of route aggregation with other path vector routing).

Assuming $\sqrt{n}$ areas are defined, we obtained a communication cost of $O(\sqrt{n}.m)$. In the most plausible usage scenario, every edge of each ball advertizes the set of contiguous coordinates (area) they contain to their external neighbors and redistributes the received routing information to internal neighbors by means of reflectors (similarly to BGP route reflectors).

Comparison with the Border Gateway Protocol (BGP) shows that geodesic geometric routing provides remains competitive in terms of memory-stretch tradeoff.

5.2 Geometric Tree Routing

Geometric tree routing (GTR) schemes rely on embeddings into metric spaces to attach a geometric coordinate to ASs encoding discovered routing information. In GTR, coordinates are determined by a greedy embedding based on a **spanning tree discovery component**. Rather than relying on hyperbolic coordinates, tree coordinates are used (which in simulation experiments proved to be performing equally well than other greedy embedding based

schemes (the properties of the topology as reported by CAIDA maps which does not reproduce all peering links are certainly not disconnected from this observation) in terms of stretch but better in terms of coordinate memory scaling).

Data packet forwarding relies on distance-decreasing routing decisions compared towards the destination AS coordinate of the packet in processing. Network or prefix reachability is distributed on-demand via a **mapping component or service** comparable to the DNS system. The mapping component maps networks / prefixes to AS coordinates. Robustness and adaptivity with respect to changing inter-AS topology might be achieved via **on-demand discovery component to bypass failing network elements**. The latter discovery process can be pro-actively activated (protection), or can be executed upon the moment of failure detection (restoration). If none of these techniques is applied, re-convergence of the supporting spanning tree might be needed, resulting into renewal of coordinates for a (sub-)tree of the topology. Emulation experiment results have been reported in Deliverable D4.5.

6. Compact routing

Following Deliverable D2.2, two classes of distributed compact routing schemes are considered hereafter; those specialized for unicast routing (based on the AGMNT scheme building blocks) and those for multicast routing.

6.1. Multicast routing (GCMR)

Our we close this gap by theoretically analyzing and comparing the performance of two reference multicast routing algorithms (the shortest-path tree and the Steiner tree), compact multicast routing as proposed in the seminal document of I.Abraham et al. [ABR2009] (referred to as ACMR hereafter) and the greedy compact multicast routing (GCMR) scheme recently proposed in [PED2011]. We compare the obtained results in order to determine the routing scheme which would yield the best trade-off between the stretch of the multicast routing paths, the memory space required to store the routing information and routing table as well as the communication cost. We also confront these results to those obtained by simulation on the CAIDA map of the Internet topology comprising 32k nodes as of January 2011 [CAI2011].

For this purpose, our performance analysis includes the following metrics:

- The *stretch* of the multicast routing scheme is defined as the total cost of the edges of the MDT (as produced by the routing algorithm) to reach a given set of leaf nodes divided by the cost of the minimum Steiner tree for the same leaf set. Note that this definition differs from the one used for (unicast) routing schemes. For the latter, the stretch is defined as the maximum cost of the produced routing path $p(u,v)$ over all node pairs $u,v \in V$ divided by the cost of the corresponding shortest (topological) path.
- The *memory space* (in bits) required at each node to locally store i) the information locally processed by the routing scheme to produce the routing table (RT) entries and ii) the produced RT entries.
- The *communication cost* (also referred to as the message cost) is defined as the number of messages exchanged to build the MDT. This metric is directly related to the leaf join time, i.e., the higher the message cost the longer the time needed for a leaf to join the tree.

We also define the adaptation cost of the multicast routing scheme as the number of multicast routing states changes resulting from MDT changes due to arbitrary join-leave sequences or topology changes.

6.1.1 Stretch

a) ACMR

The stretch of the ACMR scheme as determined by the Lemma.7 of [ABR2009] is $O(min\{log(n), log(\varpi)\}.log(n))$ competitive compared to the stretch of the ST algorithm. The quantity ϖ called the aspect ratio of the graph G is defined as the ratio between the maximum distance $max\, d(u,v)$ and the minimum distance $\min d(u,v)$ for any node pair $u,v \in V$ (see [ABR2009]). Note that when the minimum distance is equal to 1, then the aspect ratio corresponds to the diameter $\Delta(G)$ of the graph G.

Using the CAIDA maps of the Internet topology comprising 16k (January 2004) and 32k nodes (January 2011), the measured ratio $\varpi = \Delta(G) \simeq 10$. These

results are confirmed by the systematic routing path length measurements documented in [HUF2002]. Consequently the stretch of the ACMR scheme is $O(log(n))$. Note here that compared to other studies the present document makes a clear distinction between the average path length and the diameter of the graph (i.e., the maximum path length). Moreover, since the diameter of the unweighted graph underlying the Internet topology is of the order of $log(n)$, the stretch upper bound of the ACMR scheme is $O(\Delta(G))$.

b) GCMR

For unweighted (weighted) graphs, the stretch of the GCMR scheme is determined by Lemma 1 (respectively, Lemma 2).

Lemma_1: The stretch upper bound of the GCMR scheme is $O(\frac{\Delta(G)+1}{2})$.

Proof: cf. reference [PAP2014]

Lemma_2: The stretch increase of the GCMR scheme is dominated by the sum (over all join events) of the ratio between the minimum distances $min_v\{d(u_i,v)|v \in B(u_i) \wedge v \in T_{s,M}\}$ and $min_w\{d(u_i,w)|w \notin B(u_i) \wedge w \in T_{s,M}\}$ such that $min_w\{d(u_i,w)\} < min_v\{d(u_i,v)\}$.

Proof: cf. reference [PAP2014]

c) Comparative Analysis

The stretch upper bound of the multicast routing paths produced by the ACMR scheme even if universal (i.e., applicable to any graph) is 2 times higher than the one produced by the GCMR scheme. It also important to note that the stretch of the GCMR scheme has a second order dependence on the network size (due to its dependence on the diameter $\Delta(G)$). On the other hand, the ACMR scheme shows a first and a second order dependence on the network size (due to its dependence on the number of nodes n and the diameter $\Delta(G)$ and the number of nodes n).

Fig.6.1 depicts the routing scheme stretch obtained by simulation of the ST, the SPT, the GCMR and the ACMR scheme (for different values of the parameter k). The simulations are performed on the CAIDA map of the Internet topology comprising 32k nodes. The scenario executed simulates the construction of multicast routing paths for leaf node set of increasing size from 500 to 4000 nodes with increment of 500 nodes. Each execution is performed 10 times by considering 10 different multicast sources.

From this figure, we can observe that the upper bound for the ACMR scheme is not reached (its maximum value reaches 2.15 for $k = 1.5$). Moreover, the stretch of the GCMR scheme is in average still twice better than the stretch of the ACMR scheme with a maximum value of 1.08 (for 500 leaf nodes) and a minimum value of 1.03 (for 4000 leaf nodes). Note also that the comparative gain is weakly influenced by the value of the parameter k. This parameter characterizes the sparse tree cover construction: the higher the value of k, the lesser the number of trees in the sparse tree cover (TC).

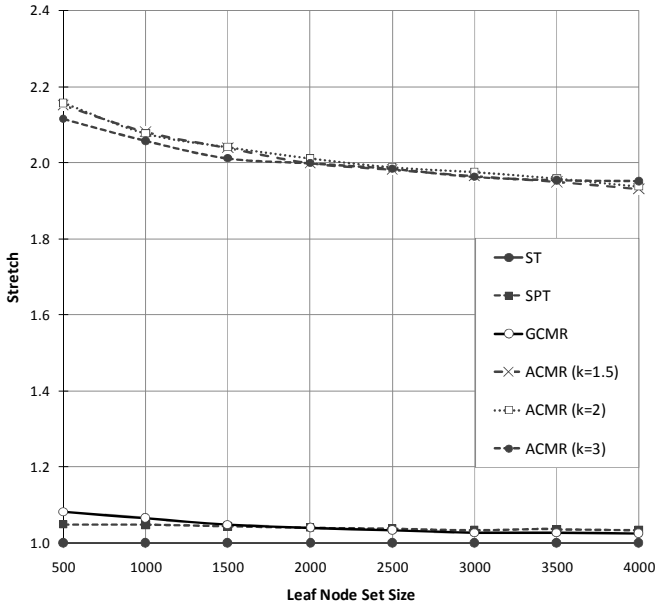

Fig.6.1: Stretch in function of the leaf node set size

6.1.2 Memory

a) ACMR

The memory space consumed by the ACMR scheme as documented in Section 6.1 of [ABR2009] comprises the space required to store the following routing information:

1) Each node $v \in V$ stores the tree routing information $\tau\ (T, v)$ for all the trees T in its own label $SPLabel(v)$[1], which yields a total memory of $O(log^3(n).log(\varpi)/\ log(log(n)))$.
2) For each radius $r \in R = \{0,1,\dots,log(\varpi)\}$ and tree T belonging to the sparse tree cover $TC_{k,2^r}(G)$, the center node $c(T_i(v))$ of node $v \in T$ stores the label of all nodes contained in the ball $B(v,2^r)$, which leads to a total memory over all $|R|$ radii of $O(kn^{1+1/k}log(\varpi))$ bits.
3) Each node $v \in V$ stores $O(log(\varpi))$ labels of size $\tilde{O}(kn^{1/k})$ to reach the center nodes $c(T_i(v))$ for all radii $r \in R = \{0,1,\dots,log(\varpi)\}$, which leads to a total memory of $O(kn^{1+1/k}log(\varpi))$.

Thus, the ACMR scheme consumes in total $\tilde{O}(kn^{1+1/k})$ bits. As the value of the parameter k ranges in the interval $[\,1, log(n)\,]$, we obtain respectively as upper bounds $\tilde{O}(n^2)$ and $\tilde{O}(n^{1+1/\log(n)})$. Note that the memory consumption of the ACMR scheme is independent of the MDT size.

b) GCMR

Per multicast source s, each node $v \in T_{s,M}$ stores in its local routing table one entry to the selected upstream node and one multicast routing entry. The memory-bit space consumed by the multicast routing entry, which indicates the outgoing ports for the incoming multicast traffic is proportional to the local tree out-degree d_k. Assuming an optimal port identifier encoding proportional to $log(n)$ at each node, the total memory space consumed by the MDT constructed by means of the GCMR scheme is

[1] The label SPLabel(v) stores the label λ(T,c(T)) given by Lemma 9 of [3] for each tree T part of the sparse tree covers containing node v.

$O(h\,log(n))$, where h is the size of the MDT. The latter equals n when the MDT covers the entire network.

c) Comparative Analysis

Depending on the value of the parameter k, the GCMR scheme (for $h = n$) is $\tilde{O}(n)$ competitive for $k = 1$ and $\tilde{O}(n^{1/log(n)})$ competitive for $k = log(n)$ compared to the ACMR scheme. The main difference between them consists in that the GCMR scheme depends explicitly on the MDT size whereas the ACMR scheme depends on the network size.

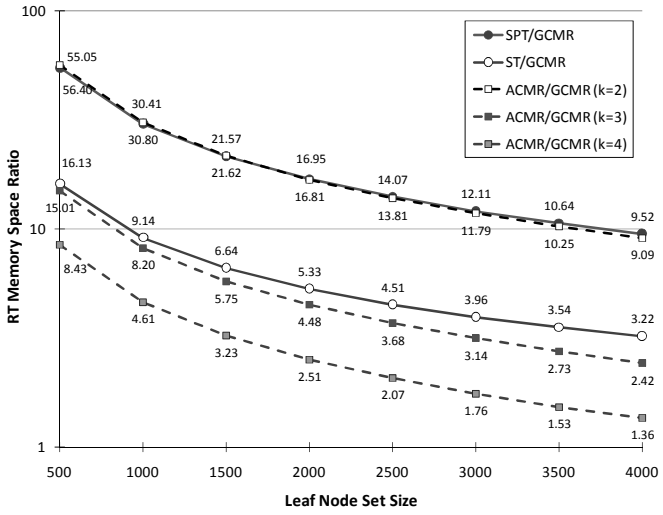

Fig.6.2: Memory space consumption ratio in function of the leaf node set size

Fig.6.2 depicts the memory consumption ratio of the ST, the SPT and the ACMR scheme (for different values of the parameter k) against the GCMR scheme. This ratio provides a good indication of the achievable reduction in terms of the memory space required to store the routing information and routing table entries produced by these algorithms. The results are obtained by means of simulation on the CAIDA map of the Internet topology comprising 32k nodes. The scenario executed simulates the construction of multicast routing paths for leaf node set of increasing size from 500 to 4000 nodes with increment of 500 nodes. Each execution is performed 10 times by considering 10 different multicast sources.

From Fig.6.2, we can observe that for a leaf set of 500 nodes the memory space consumption ratio between the ACMR and the GCMR scheme decreases from 56,40 (for $k = 2$) to 8,43 (for $k = 4$). This ratio decreases as the size of the leaf node set increases. When the size of the leaf set reaches 4000 nodes, this ratio drops to 9.09 (for $k = 2$) and 1.36 (for $k = 4$). These results confirm that the gain in memory space consumption obtained with the GCMR scheme decreases against the ACMR scheme as the size of the MDT increases. The dependency of this gain with respect to the parameter k finds its origin in the underlying sparse tree cover construction that the ACMR scheme requires: the higher the value of the parameter k, the sparser the tree cover. As the value of this parameter increases to its maximal value $log(n) \sim \Delta(G)$ and the size of the leaf node set increases to n, the gain in memory space consumption tends to 1. However, this situation is unlikely to occur in practice as it would imply that the MDT comprises all network nodes.

6.1.3 Communication Cost

In order to analyze the communication cost it is important to distinguish between adaptive and oblivious routing. A main property of the ACMR scheme variant documented in Section 6.2 of [ABR2009] is the construction of MDTs that are oblivious, i.e., the multicast routing path from the source s to a given leaf node is irrespective of the other leaves. Due to obliviousness, when other nodes join and leave the MDT, this does not affect the multicast routing path to that leaf. In contrast, the GCMR scheme is adaptive, i.e., routing decisions may be modified once there is a change in the information that has lead to that decision. This implies that even if the GCMR scheme is competitive compared to the ACMR scheme, interleaved sequences of join and leave events may increase the message cost. For this purpose, we distinguish between the "join" communication cost from the adaptation cost, i.e., the additional message cost to restore the optimal multicast routing path when nodes that previously joined the tree leave the MDT before the multicast session ends.

6.1.3.1. **Join Communication Cost**

a) ACMR

The total communication cost of the ACMR scheme can be derived from the Lemma.7 of [ABR2009]. In case of join-only events, the communication cost is $O(2^{\rho+2}.2|M|.log(\varpi).log(n))$, where $|M|$ is the size of the leaf node set.

Since the exponent ρ is at maximum equal to 1 (following the inequality $\rho \leq \log(\Delta(G))$ with $\Delta(G) \backsimeq 10$), we obtain for the total communication cost of the ACMR scheme $O(16|M|.log(\varpi).log(n))$. Moreover, as the minimum distance of the unweighted graph underlying the Internet topology is equal to 1, the aspect ratio Δ corresponds to the diameter $\Delta(G)$ of the graph G; hence, we obtain for the total communication cost $O(16|M|.log(n))$.

b) GCMR

In the GCMR scheme, each join event as initiated by a node $u_i \in V$ to reach a node $v \in T_{s,M}$ results in a communication cost equal to:

$$C(u_i) = 2\mu_i X_i + 2m(1 - X_i) \qquad (2)$$

In (2), μ_i and m are respectively the number of edges in the vicinity ball $B(u_i)$ of the joining node u_i and the total number of edges $|E|$ in the graph. The Boolean variable $X_i = 1$ when at least one node $v \in T_{s,M}$ is comprised in the vicinity ball $B(u_i)$ of the joining node u_i. Thus, when all the multicast distribution tree nodes $v \in T_{s,M}$ are outside the vicinity ball $B(u_i)$, the communication cost $C(u_i) = 2m$.

The total communication cost, i.e., the cost to build the entire MDT, is thus determined by the sum of the individual communication costs $C(u_i)$ induced by all nodes $i = 1, \ldots, |M|$ joining the multicast tree $T_{s,M}$:

$$C(T_{s,M}) = \sum_{i} [2\mu_i X_i + 2m(1 - X_i)] \qquad (3)$$

As already shown in [PED2011], defining a vicinity ball size proportional to $\sqrt{n}/log(n)$ minimizes the number of messages exchanged during the construction of the MDT and thus the communication cost. To further reduce the communication cost of the GCMR scheme, each multicast source s constructs a vicinity ball $B(s)$ whose number of edge is given by μ_s. This vicinity ball shall demonstrate the following properties i) its size at least as large as the average size of leaf node's vicinity ball, and ii) the radius locally computed from its outgoing ports is inversely proportional to the neighbor's node degree. Subsequently, when a request

message reaches the boundary nodes of the ball $B(s)$ of the multicast source s, the message is directly routed along the shortest path to the source s. This enhancement prevents searching at the neighborhood of the multicast traffic source. The total communication cost is thus determined by:

$$C(T_{s,M}) = \sum_{i} [2\mu_i X_i + 2(m - \mu_s)(1 - X_i)] \qquad (4)$$

c) Comparative Analysis

Simulations performed on the CAIDA map of the Internet topology comprising 32k nodes show that the communication cost ratio of the GCMR scheme is relatively high compared to the SPT algorithm. As depicted in Fig.3, the communication cost ratio between the GCMR scheme and the SPT algorithm increases from 2,69 (for leaf set of 500 nodes) to 8,17 (for leaf set of 4000 nodes). The ratio's slope decreases as the leaf node set increases until reaching a saturation level around 10. It is worth mentioning that the memory and the capacity required to process communication messages are relatively limited.

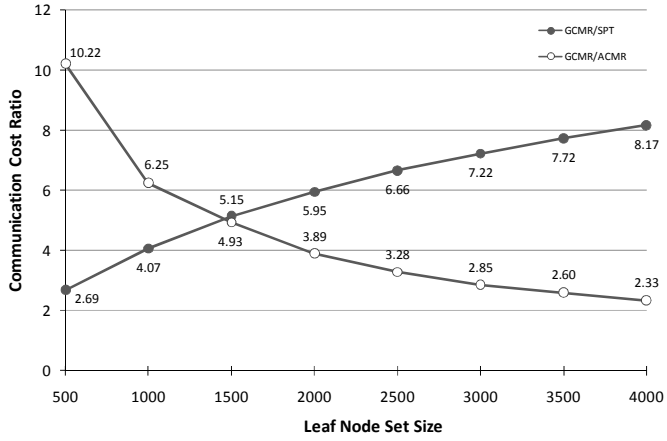

Fig.6.3: Communication cost ratio in function of the leaf node set size

When comparing the communication cost of the ACMR scheme against the GCMR scheme for the same topology, the opposite trend can be observed from Fig.3. Note here that the communication cost for the ACMR scheme accounts also for the hidden cost associated to the exchange of multicast routing information between joined branching points (for each joining node u_i) and the multicast source node s. The communication cost ratio between the GCMR scheme and the ACMR scheme decreases from 10,22 (for leaf set of 500 nodes) to 2,33 (for leaf set of 4000 nodes). The gain factor observed when decreasing the size of the leaf node set plays in favor of the ACMR scheme and underlines that improvement(s) should be further considered to reduce the join communication cost of the GCMR scheme.

6.1.4 Adaptation Cost

In order to evaluate the adaptation cost of the GCMR scheme, we are interested in determining the maximum number of re-routing events that this scheme requires to adapt the MDT upon occurrence of leave events. Remember that the ACMR scheme is oblivious (when nodes leave the MDT, the multicast routing path to the remaining leaf nodes is not affected); hence, there is no additional adaptation cost.

For the GCMR scheme, which is adaptive, the situation completely differs; in particular, when the node $v \in T_{s,M}$ leaves the MDT after an arbitrary sequence of σ $(1 \leq \sigma < |M|)$ dependent join events, each involving at least one

of the nodes along the path $p(v,s)$ from the leaving node v to the multicast source s of the MDT. In this case, a certain number of re-routing events are required to restore the optimal multicast routing path.

Theorem_1: The number of re-routing events triggered by a node leaving the MDT after an arbitrary sequence of σ $(1 \leq \sigma < |M|)$ dependent join events is $O(\sigma.(\Delta\ (G)-1))$.

Proof: cf. [PAP2014]

Since the diameter $\Delta(G)$ of the unweighted graph G underlying the Internet topology grows proportionally to $log(n)$, the number of re-routing events is relatively limited. The derivation of the corresponding message exchange depends on the aspect ratio of the multicast distribution tree. Further investigation would enable determining the total message cost depending on the aspect ratio of the multicast tree.

6.2. Distributed asynchronous AGMNT (AGMaNT)

Our main goal was to perform a comparison between several compact unicast routing schemes in order to get good trade-offs between the routing path stretch, Average/Max memory-bit capacity required per node and the communication cost. In the following, m stands for the number of links, Δ for the hop-diameter of the graph G and n is the number of nodes. Depending on the context, "nodes" can either represent AS or router.

Note that these schemes performs asynchronously (meaning that message exchanges between nodes is are non blocking, nodes do not wait each other once the algorithm produces an output thus the algorithm is capable to work with partial information and nodes perform message processing independent of the others). In the table 2, the communication cost stands for the number of messages in a synchronous setting in order to build all distributed tables from scratch. Note that modeling asynchronous behavior.

In red, the complexity represents the particular case of scale-free networks in the random power law graphs model of Li [LI2005]. These networks are assumed to mimic well AS-level networks. The complexity f stands for $O(f\ log(n))$.

Table 2: Comparison stretch - memory - communication cost

	Stretch	**Avg Memory**	**Max Memory**	**Com. Cost**
Shortest Path-Vector	(1,0)	Δn n	Δn n	mn n^2
AGMaNT	(5,0) or (7,0)	$\sqrt{n}$ $\sqrt{n}$	$\sqrt{n}$ $n^{2/3}$	m$\sqrt{n}$+$\Delta n^{\frac{3}{2}}$ $n^{3/2}$
LandOmni	(3,0)	$\sqrt{n}$ $\sqrt{n}$	n n	m$\sqrt{n}$+$\Delta n^{3/2}$ $n^{3/2}$
HDBLR	(2,2D_L) O(1)	$\sqrt{n}$+B $\sqrt{n}$	n n	m($\sqrt{n}$+B) $n^{3/2}$
Cluster	(3,4D_L) O(1)	1+B' 1	$\sqrt{n}$+B' $\sqrt{n}$	m(1+B') n

The different algorithms share several building blocks:

- A set of nodes, landmarks L, are known by every node that is all the nodes are able to route toward landmarks using shortest paths. D_L stands for the maximal distance between landmarks.
- Every node is able to route within its close neighborhood, called vicinity balls. More precisely, every node knows its B closest nodes until B contains at least one landmark and possibly one manager per color.
- Every node share a same random hash function h and has a color between range 1 to k (=$\sqrt{n}$ in our experiments). The hash function h also assign node id to the range 1,…,k.
- A manager of color c is node knowing how to route toward nodes whose node id's have hash value c.

Sketch of the different distributed constructions:

- Shortest Path Vector: based on distributed Bellman-Ford algorithm (using the hop count as distance metric)
- AGMaNT 5/7: two asynchronous and distributed versions of a universal compact routing scheme. This algorithm is further documented in [GAV2013] presented at DISC 2013. Landmarks are chosen uniformly at random and within a vicinity ball, there is at least one manager per color.
- LandOmni (LO): Landmarks are chosen uniformly at random and know how to route to every node. Vicinity Balls only have to store a landmark.
- HDLBR (compact routing scheme specialized to scale-free networks): Landmarks are the nodes of large degree or centrality. D_L turns to be constant.
- CLUSTER (specialized to scale-free networks): One landmark is the node of highest degree. Other Landmarks are connected and are the closest nodes of the first landmark.

The results included in the table are nevertheless representative of worst cases and the algorithms should be executed in different asynchronous scenarios. The effective and/or average communication cost can not be derived from our complexity analysis. It is the reason why, our algorithms have been implemented and simulated using DRMSim, our high-level routing scheme simulator. The executed version of our algorithm is available at https://gforge.inria.fr/projects/vizroute/

In order to get a variety of cases, we decided to mainly consider 3 types of topologies (see Fig.6.4):

- Scale-free networks (synthetic and from real data sets: CAIDA, DIMES): small diameter growing proportionally to $log(n)$, heterogeneous degree, high degree nodes are central
- Networks of large diameter (UDG): homogeneous degree, existence of central nodes
- Random graphs of small diameter (Erdös-Renyi $G(n,p)$: homogeneous degree, no central nodes

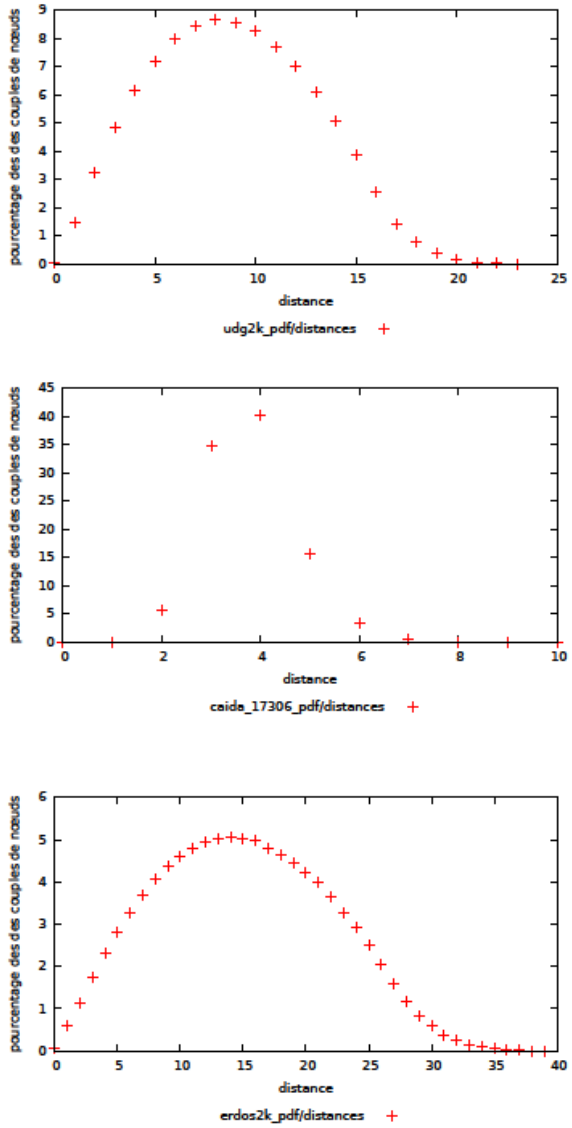

Figure 6.4: Topologies

6.2.1 Communication Cost

The asynchronous model used in our experiment is the bounded delay model where the delay (drawn out of random distribution over the interval 1 to MAX-DELAY). More MAX-DELAY increases more the communication cost grows but very slowly, that is not linearly with respect to MAX-DELAY. It confirms that the synchronous time model is not inappropriate to approximate the average communication cost. Figure 6.5 shows the real difference for the communication cost. This figure shows the real benefit of compact routing schemes with respect to standard shortest path-vector schemes. We obtain a gain of a factor 10 to 1000 depending on the compact routing scheme and/or topology.

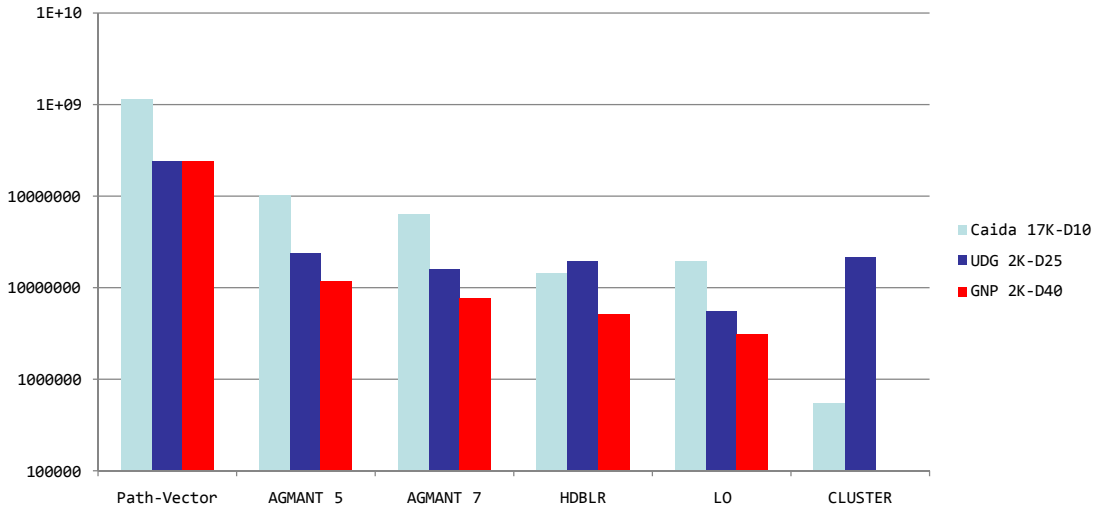

Figure 6.5: Communication cost comparison

6.2.2 Stretch

For the routing stretch, we do not obtain a significant difference. Compact schemes specialized for scale-free graphs (HDLBR, CLUSTER) do not get a real advantage on CAIDA (average multiplicative stretch around 1.6, the maximum is 10) with respect to universal compact routing schemes: AGMaNT 5/7 or LO for which the maximum stretch is 5 or 7 as shown in Figure 6.6.

The results obtained for CLUSTER (and to a lesser extend HDBLR) prove that exploiting node degree distribution does not lead to a significant gain in terms of stretch.

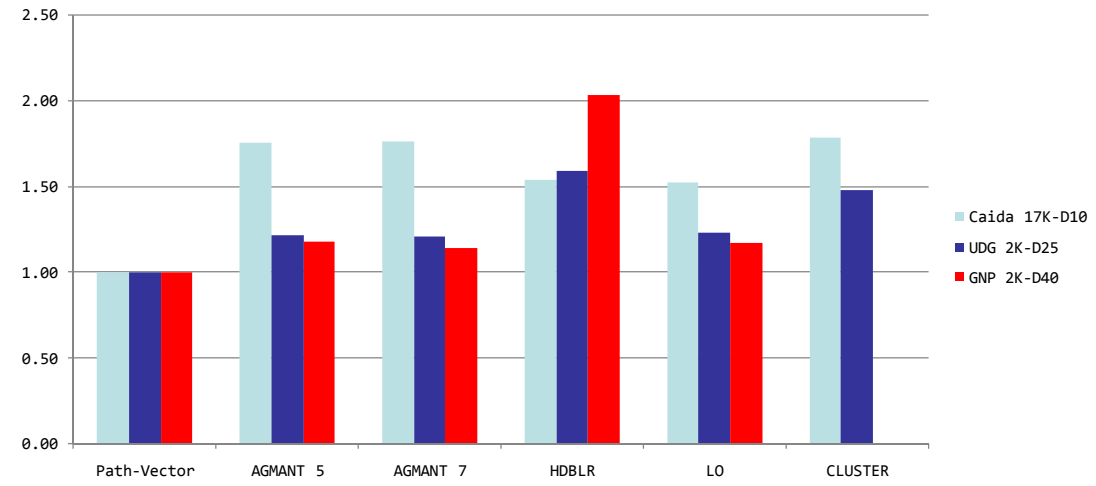

Figure 6.6: Stretch comparison

Moreover, the stretch can be very large for some pairs of sources and destinations. The corresponding behavior is provided in Fig.6.7.

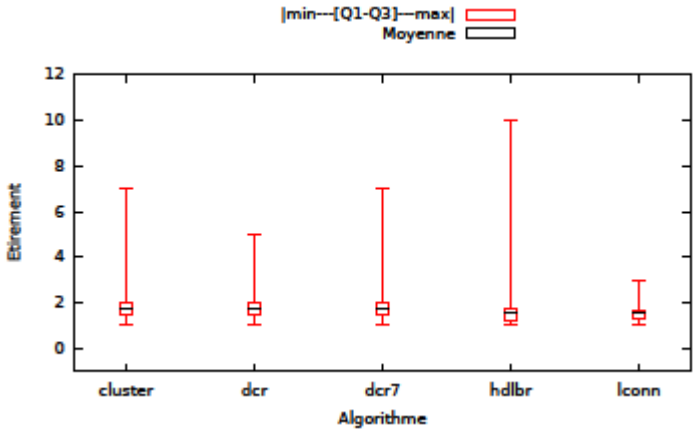

Fig.6.7: Mean and Min-Max stretch vs. compact routing algorithm

6.2.3. Memory

It turns out that the whole memory-space consumption (not only the number of routing entries) to compute the compact routing tables are of the same order that the compact routing tables. There is no need to exchange/know the full topology description. Beside the CLUSTER algorithm specialized for scale-free graphs (optimized for memory savings on scale-free graphs) all other variants require a similar range of memory space complexity.

The next figure shows the real advantage of Cluster for scale free networks (8 entries on average for CAIDA) with respect to the average number of entries per node. However, as for the stretch, if we take a closer look at the detailed measures some nodes can require a lot of memory.

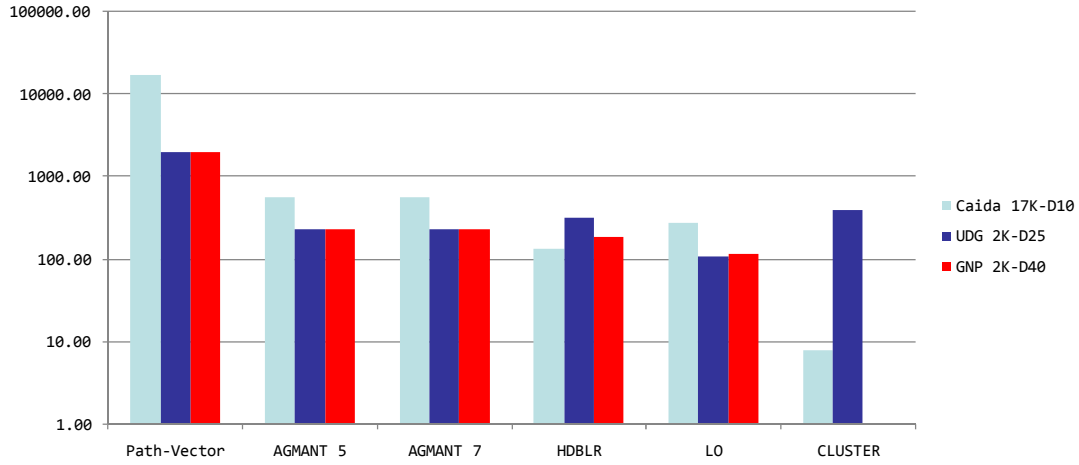

Fig.6.8: Memory complexity comparison

To conclude, our simulation results confirm that compact routing schemes provide a real gain in terms of communication cost and memory complexity without degrading paths stretch (1.75 see Figure 6.6) with respect to shortest path routing.

Assuming that the network topology is scale-free, specialized schemes performs better (in particular the Cluster algorithm) but universal schemes (AGMANT and LO) have reasonable performances even for scale-free networks.

6.3 Greedy 2-phase Reduced-state (G2RR) Routing Scheme

6.3.1 Network Model

We consider a network as a graph $G = (V, E)$ where V and E are the finite set of node and bidirectional links, respectively. The number of elements of set V and E is n and e, respectively. The network is partitioned into g connected disjoint groups where, $1 \leq g \leq n$. Each node in the network should identify by a unique node identifier (id) and a group identifier ($groupid$) and all nodes should participate in the routing process. We assume that each node knows its one-hop (NL) and two-hop neighbors information base (IB) which includes their id, $groupid$ and node degree (Δ). The one-hop (NL_x) and two-hop neighbors' information (IB_x) considered as local information for particular node x. We assume that only source nodes s know the t_{id} and $t_{groupid}$ of the destination t and other subsequent intermediate nodes can only know the id and $groupid$ of the destinations by receiving the packet p from source nodes. Every intermediate node for a specific s and t during path searching can modify the searched path.

6.3.2. Greedy 2-phase Reduced-state Routing (G2RR) Scheme

The routing problem is modeled as a decision making task with uncertainty in which the objective is to minimize the length of the path traveled by the data packets. The G2RR scheme is a reactive routing scheme which includes separate route discovery and data packet forwarding tasks and consists of following mechanisms:

1. Route discovery: The route discovery process initiated by the source node that issue a discovery packet. At the source node, and subsequently at each node along the path, the optimal decision rule (with the local information) is to forward the discovery packet to the neighbor from which the message will reach the target in the smallest number of hops, assuming that all future nodes will make their decision similarly by using local information. Here, we suggest that an effective (although not necessarily optimal) decision would be forward the packet to the two-hop neighbor which is more similar to the destination, where similarity is consider as either it share same $groupid$ or id of the destination. If similarity does not exist in local information than the decision would be forward the packet to the two-hop neighbor which is more connected, i.e., highest degree node. The selection of next hop at each intermediate node x for the destination t from source s is carried-out in two phases: (1) if the $groupid$ of destination exists in two-hop neighborhood, then select highest degree node with same $groupid$ of destination as a next hop or (2) if the $groupid$ of destination does not exist in two-hop neighborhood, then select anode with highest degree in two-hop neighborhood without considering the group. Once the discovery packet reaches at any node which shares the same $groupid$ with destination, than process tries to search destinations id in that particular group. This procedure should be followed by subsequent nodes. The discovered path by aforementioned scheme showed in the Figure 6.9 as dark arrow.

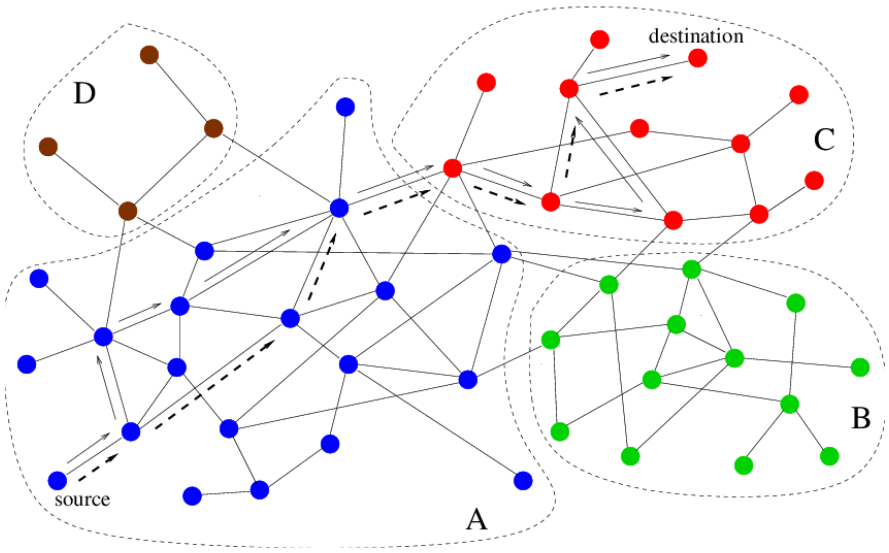

Fig.6.9: The discovered path by route discovery scheme showed as dark arrow. The optimized path is shown in dashed arrow.

After route discovery, the destination sends acknowledgement with discovered path information to the source node for data packet delivery.

2. Run-time path optimization: During the route discovery process, our scheme uses a run-time path optimization mechanism, which reduces path length, if possible. In this process, the current node of a discovery packet checks the every element in the path which existing in its 2-hop neighbor list. An element of path exists in 2-hop neighbor and path length between found element and current node larger than two that it eliminates the in-between elements and put the connecting node. The optimized path form discovered path in Figure 1 showed in Figure 2.

3. Loop Avoidance: A routing path found by path discovery process in our scheme cannot have loops because scheme uses a loop avoiding mechanism in which we restricted to revisit a node which had been already visited during path discovery process. This mechanism can lead to path discovery failure in few cases.

4. Network Partitioning: the scheme requires partitioned network for reduces the problem size, therefore the partitioning methods has vital role in the quality of the discovered route. Here, we uses two types of partitioning methods: 1) Biased partitioning method, which partitions the graph into a few large groups and many small groups, and 2) Unbiased partitioning method, which partitions the graph into groups of approximately same sizes.

6.3.3 Simulation Results

Performance Metrics: to evaluate the performance of the proposed route discovery scheme compute several performance metrics.

The first metric is the success ratio, which is the percentage of discovered paths that successfully reach their destinations. The second metric is stretch, which tells us how much longer the discovered paths are, compared to the shortest paths in the Internet topology.

Simulation setup 1: We performed the path level simulation for validating our scheme for real networks. The simulation exploits the CAIDA maps as an input topology which available in public domain. These CAIDA maps are approximate snap shots of internet topology at particular time, therefore the dynamics of policies are not considered in the simulation. As we known, each node in the network periodically sends the messages to acquire updated topology information of network either locally or globally. In our implementation, each node in the network is aware of its two-hop topology information with node degree and group information.

In the simulation, we use two topologies extract from CAIDA map which capture in different time shown in Table 3. For the group formation, we exploited both partitioning methods; biased and unbiased partition, which are described in previous section.

Table 3: CAIDA maps for simulation

	Scenario-1		**Scenario-2**	
Network	**CAIDA 2012 topology**		**CAIDA 2007 topology**	
Number of nodes	41203		26475	
Number of links	121309		106762	
Partitioning method	Biased partitioning	Unbiased partitioning	Biased partitioning	Unbiased partitioning

Figure 6.10 shows the performance of our routing scheme for scenario-1 with metrics, path stretch and success rate. As Figure 2(a) shows, the average path stretch is always less than 1.2 which means it performs near to shortest path with the success rate of more than 98% with biased partitioning. The Figure 2(b) shows the average path stretch is always less than 1.4 with the success rate of more than 96% with unbiased partitioning.

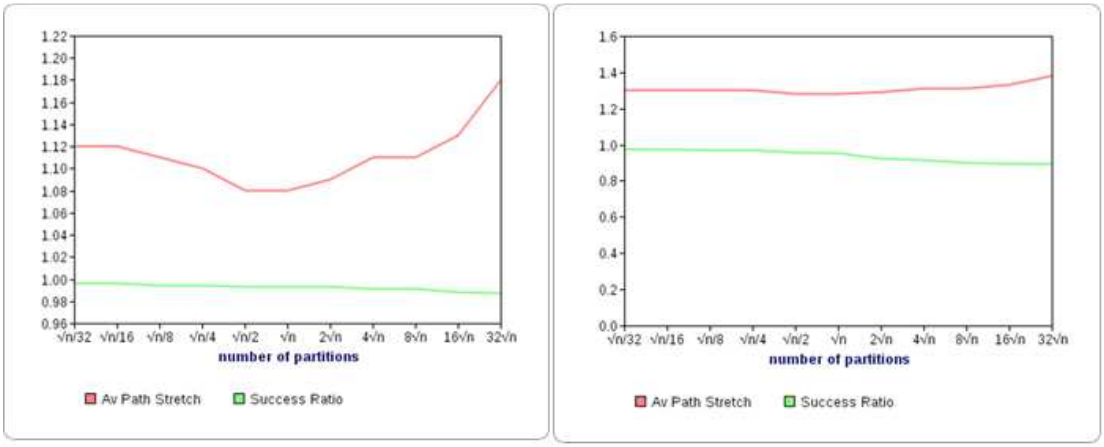

Fig.6.10: (a) average path stretch and success rate of G2RR scheme with biased partitioning for scenario-1 (b) average path stretch and success rate of G2RR scheme with unbiased partitioning for scenario-1

Figure 6.11 shows the performance of our routing scheme for scenario-2 with metrics, path stretch and success rate. As Figure 6.11(a) shows, the

average path stretch is always less than 1.2 which means it performs near to shortest path with the success rate of more than 99% with biased partitioning. The Figure 6.11(b) shows, the average path stretch is always less than 1.3 with the success rate of more than 89% with unbiased partitioning.

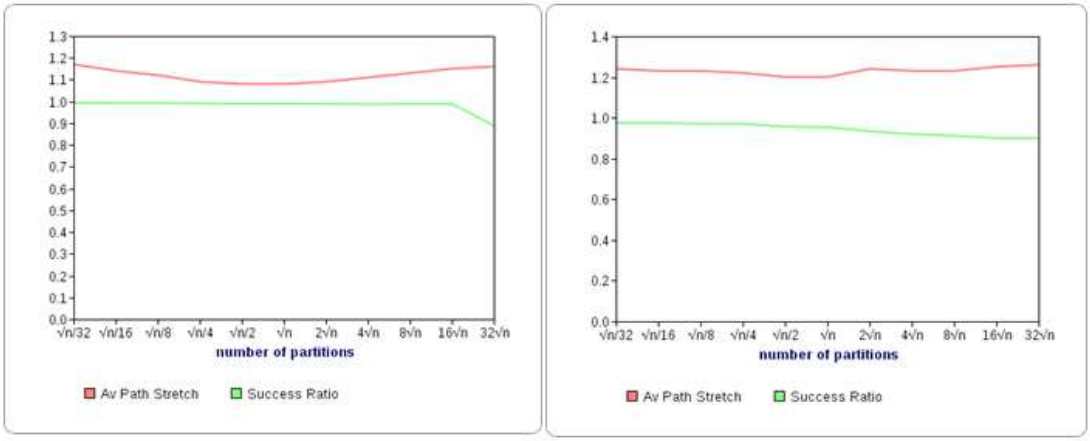

Fig.6.11: (a) average path stretch and success rate of G2RR scheme with biased partitioning for scenario-2 (b) average path stretch and success rate of G2RR scheme with unbiased partitioning for scenario-2

The Table 4 shows the average storage requirement of G2RR scheme if it needed 1-, 2- or 3-hop topology information for the route discovery process. The above result (Figure 6.10 and 6.11) obtained with use of 2-hop information requirement.

Table 4: Average storage requirement

	CAIDA 2012 topology	CAIDA 2007 topology
Average storage, if 1-hop information needed	5.888	4.032
Average storage, if 2-hop information needed	1405.416	1016.469
Average storage, if 3-hop information needed	13885.292	9090.711

Simulation setup 2: Apart for the different partitioning methods, we also conduct simulation with the country code (ISO 3166) of AS as group. The real topology snapshots of the Internet map have been used for the experiments, i.e., CAIDA map. An AS holding a packet reads its destination AS number and country code, computes the next hop according to the route discovery scheme, and forwards the packet to the computed next hop. The average success ratio of the route discovery scheme in Internet topology is remarkably high, 98% with use of the country code as group and the average stretch is low, 1.07. The average hop-wise length of the shortest paths between selected sources and destinations is 3.54, so that the average length of discovered paths by our scheme is 3.79. The low value of stretch indicates that discovered paths by our scheme are close to optimal, i.e., shortest paths.

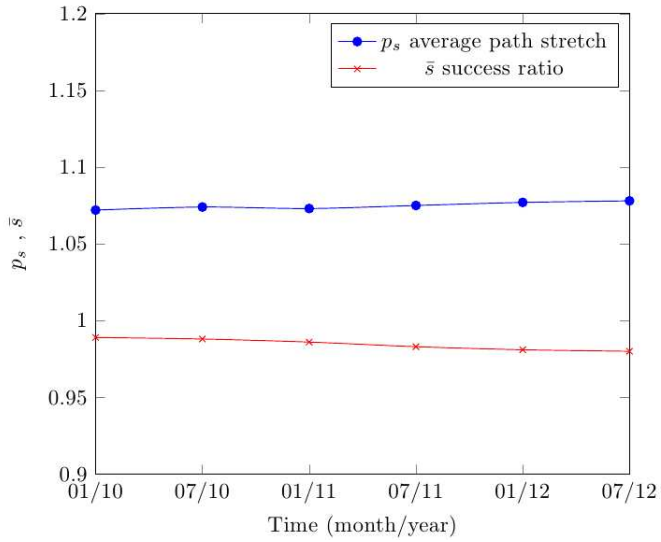

Fig.6.12: The route discovery scheme performs almost optimally in the AS topology growing from Jan.2010 to Jul.2012 as indicated by the success ratio, $\bar{s}$, and average stretch, p_s.

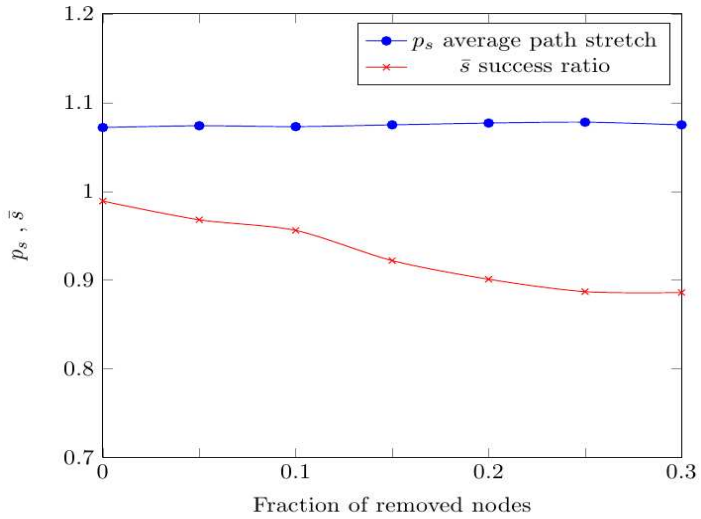

Fig.6.13: Proposed route discovery scheme performs almost optimally in the Internet, as indicated by the success ratio, $\bar{s}$, and average stretch, p_s, after removal of a given fraction of AS nodes

The two metrics above characterize the performance of route discovery in the static Internet topology. As important is how route discovery performs in the dynamic topology, where links and nodes can fail. We randomly select a percentage of links and nodes, remove them from the mapped Internet, re-compute the success ratio and stretch after the removal, and present the result in the Figure 6.13 and 6.14.

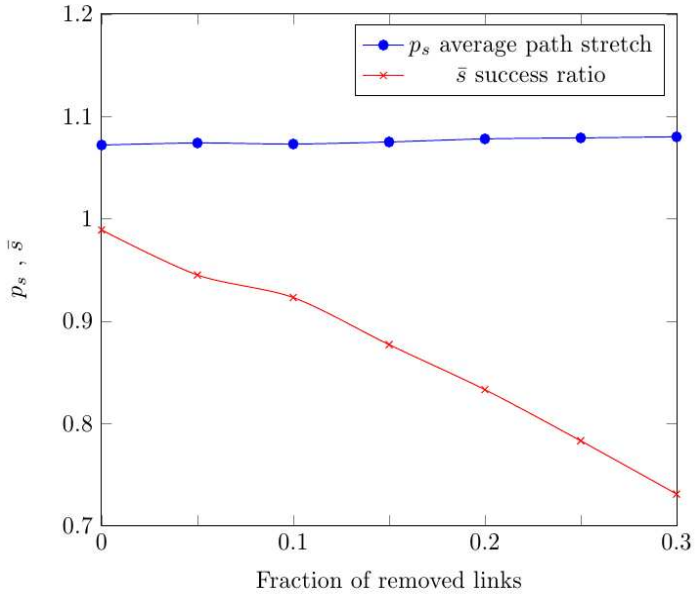

Fig.6.14: The success ratio, $\bar{s}$, and average stretch, p_s of proposed route discovery scheme, after removal of a given fraction of AS nodes

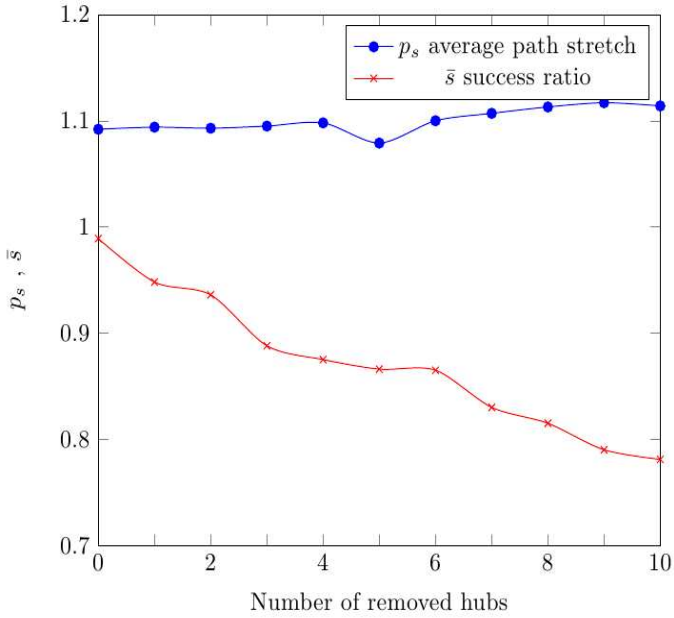

Fig.6.15: The success ratio, $\bar{s}$, and average stretch, p_s of proposed route discovery scheme, after removing a number of the highest-degree nodes

Even upon simultaneous failures of up to 10% of AS links or nodes– catastrophic events never happened in the Internet history, we observe only minor degradation of the performance of route discovery. The behavior of the scheme in the dynamic scenario is shown in Figure 6.15 and 6.16.

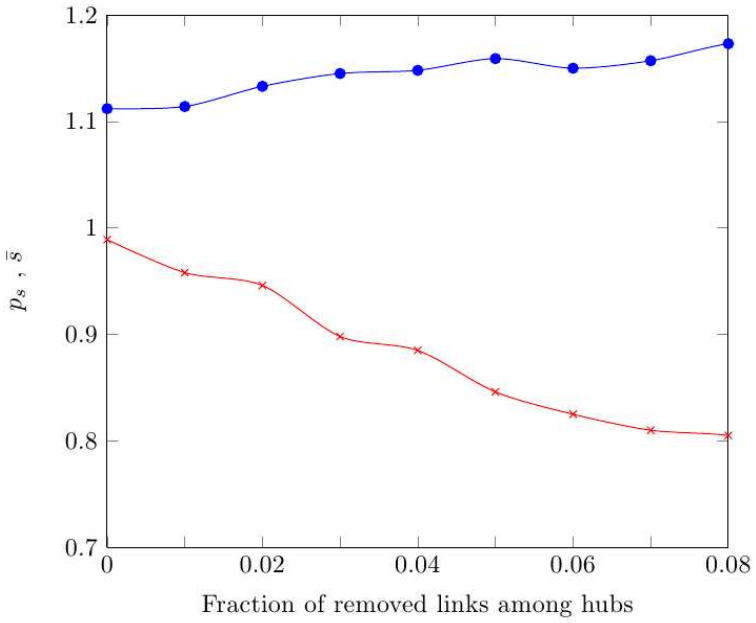

Figure 6.16: The success ratio, $\bar{s}$, and average stretch, $p_s$of proposed route discovery scheme, after removing a fraction of links among highest-degree nodes

6.4 Multi-path Routing via Independent Trees

Several routing schemes studied in this document make use of tree construction (in particular, compact routing and GTR). It is thus appropriate to further investigate multi-path routing via independent trees in order to extend these schemes with fault-tolerance.

A possible approach for controlling the dynamics of the topology while routing consists in using multiple paths. If a topology change invalidates the current path, an alternative path already built can be used. If the alternative path is disjoint from the former, then it is guaranteed that a change in the former path will not incur the alternative path. Having two edge-disjoint paths allows to cope with one edge deletion in the topology. Having two internal-node-disjoint paths allows to cope with one node failure. More generally, k edge-disjoint paths (internal-node-disjoint paths resp.) allows to cope with k edge failures (k node failures resp.). Multi-path routing can thus allow to delay the recomputation of routing tables until a certain number of topology changes have occurred. Alternatively, multi-path routing can also be used to increase the bandwidth capacity between two nodes.

Recently the design of compact routing tables enabling multi-path routing [CHA2014] has been investigated. We present in the following the approach and the results obtained.

6.4.1 Simple labels

A basic approach for routing along multiple paths consists in using path labels $1,\ldots,p$ for distinguishing the paths for a given destination t. This results in defining p next hops $NH_1(u),\ldots,NH_p(u)$ at each intermediate node u when routing towards t: if node u receives a packet for destination t with path label i, it forwards the packet to $NH_i(u)$. In other words, using simple path labels amounts to define p partial trees $T_1,\ldots,T_p$ rooted at t for each destination t. We let $T_i(u)$ denote the branch of T_i from u to t, that is the path $u, NH_i(u), NH_i\big(NH_i(u)\big),\ldots$ followed by a packet for t with label i.

6.4.2 Independent trees

We are interested in obtaining disjoint paths. Given a network whose topology defines a graph G, a node u is k-edge-connected (k-node-connected resp.) to t if there exists k edge-disjoint paths (internal-node-disjoint paths resp.) from u to t in G. We will say that a collection $T_1,\ldots,T_p$ of partial trees rooted at t is k-edge-independent (k-node-independent resp.) if for any node u which is k-edge-connected (k-node-connectivity resp.) to t, there exists $i1,\ldots,ik$ such that the paths $T_{i1}(u),\ldots,T_{ik}(u)$ are edge-disjoint (internal-node-disjoint resp.). The existence of such trees is not guaranteed for $p = k$ as detailed in the following section.

6.4.3 Related work

From [EDM1969], there exists k k-edge-independent trees rooted at t in G iff every vertex u is k-edge-connected to t. Concerning node-disjoint path, Whitty proves (for $k = 2$) that any 2-node-connected graph G contains two 2-node-independent trees rooted at any arbitrary vertex t [WHI1987]. On the other hand, A.Huck shows for $k \geq 3$ that there exists k-node-connected graphs that do not contain k k-node-independent trees rooted at one of their vertex [HUC1995]. However for $k \leq 4$, every symmetric directed graph which is k-node-connected admits k k-node-independent trees rooted at any arbitrary vertex t [HUC1994]. The concept of independent trees has been introduced in [ITA1984] for enabling reliable broadcast.

6.4.4 Variable multi-connectivity

As demonstrated in [CHA2014], any graph admits two 2-edge-independent trees rooted at any arbitrary vertex t. Interestingly, this result implies the existence of two 2-node-independent trees rooted at any arbitrary vertex t in any graph (through a simple graph reduction), generalizing the result of Whitty [WHI1987] for arbitrary graphs. The result of Huck [HUK1995] implies that some graphs do not admit k k-edge-independent trees rooted at some vertex t for $k \geq 3$.

6.4.5 Shortest path routing with an alternative path

As an application of [CHA2014], we propose a scheme for shortest path routing such that routing is still guaranteed in case of node failure. The result of [CHA2014] can be used to construct for every destination t two 2-node-independent trees T_1 and T_2 rooted at t; thus, defining $NH_1(u)$ and $NH_2(u)$ for each intermediate node u. Construct additionally a shortest path tree T_0 rooted at t; thus, defining $NH_0(u)$ for each intermediate node u. When originating a packet, u sets its path label to 0. If there is no failure, the packet will follow the shortest path $T_0(u)$. If an intermediate node v has detected that $NH_0(v)$ is under failure, it can switch the path label to 1 if $NH_1(v) \neq NH_0(v)$ or 2 if $NH_2(v) \neq NH_0(v)$, and routing will still proceed up to t if there is at most one failure.

7. Conclusion

The routing schemes evaluated by simulation are reproduced in the following table. This table summarizes their main properties with respect to the addressing space properties on which they perform their decision and the distribution process of the corresponding information.

	Locator (coordinates)		Locator (logical addresses)	Name independent
	Topology dependent (exploit topology properties)	Topology independent		
Pull		Greedy geometric routing/GTR (tree-metric)		GCMR (link metric)
Push	Geodesic geometric routing/GPV (link metric)		BGP (no metric)	BGP (no metric) Distributed Compact routing (part. tree metric)

The consolidation of the algorithmic underling each of these schemes will be the object of a specific deliverable in WP2 (Deliverable D2.3).

We can already observe nevertheless some invariants in all BGP alternatives being explored in this project: all schemes combine two types of routes (routes for destination in their close neighborhood and routes outside their neighborhood). The main difference in the corresponding discovery process results from the exchange of routing information: pull (search, route servers, etc.) vs. push (dissemination). Moreover, all alternatives involve the use of a distance metric/spatial routing metric which subdivide between local and global metrics but also between metrics derived from the properties of the topology such as the node degree or the δ-hyperbolicity (leading to specialized schemes) and universal metrics (leading to universal schemes that may show acceptable performance for the topologies under consideration). These dimensions are tightly related and our results corroborate why routing schemes such as BGP being independent of global or link metrics (the AS path length being a route selection parameter among others) but driven by local policy decisions will be extremely difficult to replace as long as the Internet domains are operated organically.

The addition of the distribution dimension leads also to reconsider the usual memory vs. stretch tradeoff (often considered in combination with the computational complexity). First, the memory complexity is not limited anymore to the routing table size (routing algorithm output) but it comprises the routing algorithm output (routing information base). Next, the communication complexity in bit-message (total number of routing update/information messages exchanged between nodes along the edges of the graph) influences the memory complexity or in time (the difference of time units between the first emission of a message and the last reception of a message during any execution of the routing algorithm) influences the convergence time. In turn, both influence the routing path stretch.

Moreover, the communication complexity in time shows a specific dependence to the messaging or communication model. Indeed, modeling the communication exchange in Internet requires to consider asynchronous message exchange models that are able to cope with local processing delay, transmission and propagation delay but also the message losses (communication channel reliability). Reproducing such conditions in simulation experiments remains however challenging due to the heterogeneity in the properties of the communication channels but also the number of relationships/routing adjacencies per node.

On the other hand, adaptivity to topology dynamics leads also to consider routing schemes providing intrinsic adaptivity of the routing decision process to non-stationary conditions instead of additional fault-tolerance/protection mechanism. More precisely, we can distinguish between mechanisms aiming at the mitigation of the perturbation effects and the avoidance of the perturbation. In first case, one can identify quantitative bounds to qualify the classes of events (at both routing and forwarding level) which initiate a sequence of actions aiming at minimizing the perturbation on the current routing/forwarding states; thus, the objective is to minimize the number of state changes while minimizing the degradation it induces. Route Flap Dampening (RFD) is a good example of such mechanism as it drops (spatial) input to prevent change of decision once a certain threshold is reached (in case of flapping links for instance), the same reasoning applies to the stability criteria developed in the project by adding a temporal dimension to the feedback control law. On the other hand, we refer to qualitative triggers to qualify classes of events (at both routing and forwarding level) which initiate a sequence of action aiming at reaching as fast as possible a new (pre-determined) state once a perturbation is detected; thus, the corresponding mechanisms aim at adapting the decision to the qualitative properties of the event itself instead of its quantitative effects. In other terms, the process consists in minimizing the time to move from the current state to the new state once the qualitative properties of the perturbation are determined. Fault tolerance mechanisms fall basically in this category as they react based on certain event types mostly spatial events independently of their duration, impact, etc. dynamic re-routing as designed nowadays too but with advantage of being robust to the event type. In comparison, BGP provides such distinction though limiting robustness to perturbation only based on spatial criteria but not temporal criteria.

Fully-adaptive routing schemes aim at supporting the best tradeoff between alternate routing states maintenance (thus, in particular, memory space consumption), stretch increase (whether the stretch of the alternate path remain close to 1), and adaptation time (which depends on the computational complexity as well as the communication complexity in time and message). Schemes such as GPV, GCMR and G2RR show adaptability to link and node failures by only requiring recomputation of the routes affected by the failure. Note that the number of routes affected by a failure is proportional to the centrality of the failing entity. GTR provides fault-tolerance/protection capability to overcome pre-determined failure patterns (and thus, pre-provisioning). Schemes such as AGMaNT on the other hand do not provide dedicate processing for information state changes (and subsequent communication) and thus require the full recomputation of the routing tables.

References

[ABR2009] Abraham,I., Malkhi,D. and Ratajczak,D., *Compact multicast routing*, Proc. of 23rd Int'l Symposium on Distributed Computing DISC'09, Elche, Spain, pp.364-378, Sep.2009.

[BER2013] Bernardes,D.F., Latapy,M. and Tarissan,F., *Inadequacy of SIR model to reproduce key properties of real-world spreading cascades: experiments on a large-scale P2P system*, Proc. of Social Network Analysis and Mining, vol.3, no.4, pp.1195–1208, 2013.

[BER2014] Bernardes,D., *Information Diffusion in Complex Networks: Measurement-Based Analysis Applied to Modelling*, PhD Thesis, 2014.

[BRI2009] Bridson,M.R. and Haefliger,A., *Metric spaces of non-positive curvature*, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences] 319, Berlin, Springer-Verlag, First Ed.1999.

[CHA2014] Charbit,P., Mathieu,F., de Montgolfier,F. and Viennot,L., *Multi-path routing via independent trees with variable multi-connectivity*, Submitted.

[EDM1969] Edmonds,J., *Submodular functions, matroids and certain polyhedral, Combinatorial Structures and Their Applications*, R. Guy et al, Eds. Gordon and Breach, New York, pp.69-87, 1969.

[GAV2013] Gavoille,C., Glacet,C., Hanusse,N. and Ilcinkas,D., *On the Communication Complexity of Distributed Name-Independent Routing Schemes*, Proc. of 27th International Symposium on DIStributed Computing (DISC) 2013, Jerusalem, Israel, Oct.2013.

[HOL2014] Romain,H., Bernardes,D., Bivas,M., Jungers,R.M. and Delvenne,J.-C., and Tarissan,F., *Data-driven traffic and diffusion modeling in peer-to-peer networks: A real case study*, Network Science, 2014.

[HUC1995] Huck,A., *Disproof of a conjecture about independent branchings in k-connected directed graphs*, Journal of Graph Theory, vol.20, no.2, pp. 235-239, 1995.

[HUC1994] Huck,A., *Independent trees in graphs*, Graphs Combin., vol.10, pp. 29–45, 1994.

[HUF2002] Huffaker,B., Fomenkov,M., Plummer,D., Moore,D. and claffy,k.c., Distance Metrics in the Internet, IEEE Int'l Telecommunications Symposium (ITS), Brazil, pp.200-202, Sep.2002.

[ITA1984] Itai,A., Rodeh,M., *The multi-tree approach to reliability in distributed networks*, Proc. 25th Annual IEEE Symposium on Foundations of Computer Sciences, pp.137-147, 1984.

[KUM2014a] Kumar,A. and Delvenne,J.-C., *A new route discovery scheme*, Internet-draft, Work in progress, Feb.2014.

[KUM2014b] Kumar,A. and Delvenne,J.-C., *Route discovery in the Internet*, Submitted at European Conference on Networks and Communications, Bologna, Italy, June 23-26, 2014.

[LEI2006] Leibnitz,K., Hossfeld,T., Wakamiya,N. and Murata,M., *Modeling of epidemic diffusion in peer-to-peer file-sharing networks*, Proc. of the Second International Conference on Biologically Inspired Approaches to Advanced Information Technology (BioADIT) 2006, pp.322–329, Berlin, Heidelberg, Springer-Verlag, 2006.

[LI2005] L.Li, D.Alderson, J.C.Doyle and W.Willinger, *Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications*, Internet Mathematics, vol.2, no.4, Aug.2005.

[NS2014] D.Papadimitriou, *Geometric analysis of flow betweenness centrality*, To be published in Proc. of SIAM Network Science, Jul.2014.

[PAP2013] D.Papadimitriou, D.Colle, P.Audenaert and P.Demeester, *Geometric Information Routing*, Proc. of Seventh IEEE International Conference on Advanced Networks and Telecommunication Systems (ANTS), Chennai, India, Dec.2013.

[PAP2014] D.Papadimitriou, D.Careglio and P.Demeester, *Performance analysis of multicast routing algorithms*, Proc. Int'l Conf. on Computing, Networking and Communications (ICNC) 2014, Hawaii (HI), Feb.2014.

[PED2011] P.Pedroso, D.Papadimitriou and D.Careglio, *Dynamic compact multicast routing on power-law graphs*, 54th IEEE Globecom, Houston (TX), USA, Dec.2011.

[WHI1987] R.W.Whitty. *Vertex-disjoint paths and edge-disjoint branchings in directed graphs*, J. Graph Theory, vol.11, pp.349-358, 1987.