# Path Computation for Incoming Interface Multipath Routing

Mérindol Pascal, Pansiot Jean-Jacques, Cateloin Stéphane
Equipe Réseaux et Protocoles
LSIIT - ULP - CNRS
Pôle API Boulevard Sébastien Brant
67400 ILLKIRCH,FRANCE
Email:{merindol,pansiot,cateloin}@dpt-info.u-strasbg.fr

***Abstract*— Currently used IP routing protocols calculate and only use a single path between two nodes of a network, or in the best case, only paths with the same cost (with OSPF2 or IS-IS extension: ECMP). If we want to use the underlying physical network with multipath routing efficiently, the loopfreeness of the used paths has to be guaranteed especially with distributed computation. Indeed different types of traffic engineering with source computation like OMP-MPLS or MATE-MPLS do not have to pay attention to loops. However the positioned paths, with CR-LDP or RSVP-TE for example, are not flexible enough to support strong load oscillations. Load balancing is only possible on the ingress node which labels the different paths (with a hash function to avoid packet mis-ordering for TCP traffic) even though the congestion spot can be very far from the ingress node. This is why distributed techniques can react more quickly to prevent congestion when possible. But such techniques do not generate enough paths in poorly connected topologies in so far as the loopfreeness condition employed (equal cost path, Loop Free Alternate or Loop Free Invariant) is stricter than necessary. In this article, we propose a multipath routing scheme able to compute more loopfree paths (with a low complexity algorithm such as Dijkstra in the worst case and a light communication protocol between directly adjacent nodes) than with existing propositions.**

**KEYWORDS:Multipath routing, QoS provisioning, traffic engineering, fast re-routing.**

## I. Introduction

Intra domain Traffic Engineering (TE) becomes a necessity in modern Internet Service Provider design. One use of TE is load balancing between operational links to avoid hot spots and increase network reliability. Furthermore, TE allows avoiding failure by faster rerouting than traditional routing using the shortest path. TE objectives are often obtained by routing traffic demands on multiple paths. Multipath routing can be divided into four tasks:

a) Compute and position paths.
b) Analyze local traffic activities and disseminate local resources availability in the network.
c) Define load balancing policy depending on the received information.
d) Split traffic among paths with granularity of a flow.

This article only focuses on the first task. The second and third issue are the subjects of several discussions in scientific papers ([15],[10],[14] for example) and generally try to minimize the maximum link use by using off-line or online (realtime) traffic measurement, but it is difficult to prevent load oscillations and guarantee system stability. The final task is to avoid packet mis-ordering of a particular flow (characterized by the same source, destination, port and other parameters depending on the needs) because TCP traffic mis-ordering implies packets re-emission and throughput decreases when it occurs. In [11], a good analysis of this issue with ECMP is given.

Several techniques are used with IP routing, such as RSVP-TE ([6] and [5]) or CR/LDP ([3] and [13]) with MPLS ([22]), in order to use the connectivity of the physical networks efficiently. In this case, path computation and load balancing are only made on the ingress router. Hence, on the one hand this kind of scheme is too rigid to achieve quick reaction in the case of a congestion. On the other hand, the extensibility in terms of number of ingress-egress *(S-D)* pairs, on which this kind of source computation protocol is used, remains limited. Indeed, without considering LSP merging, the label switching techniques may require forwarding tables proportional in size to the number of source-destination pairs in the network. In figure 1, if *S* wants to balance the load through the represented network for the destination *D*, either numbered routers are able to distribute the load by themselves with their own resource information (it is distributed routing), or *S* constructs "Label Switched Paths" (LSP) towards *D*. Hence, only source-router *S* is able to compute some distribution proportions to really share the load across the LSPs it computes and positions (it is

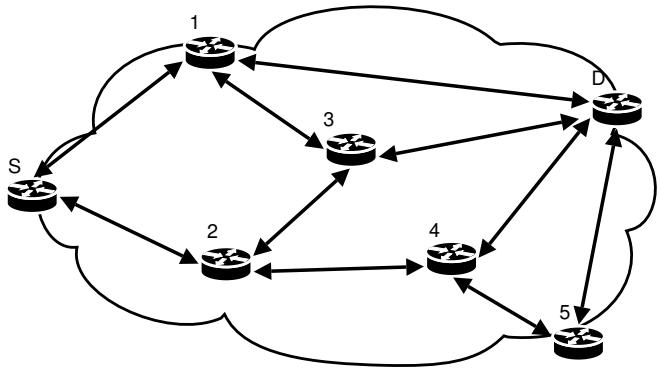


Fig. 1. How many route can link *S* to *D* without loop?

a kind of source routing without carrying the full path in the packet's IP header). In this figure, if we consider that all links have the same bandwidth, only the best path via router *1* can be used with distributed multipath existing techniques, whereas all possible paths can be used with labeling or reservation mechanisms. In this article, we will show how with our distributed proposition, *S* can benefit from six paths (*S-1-D*, *S-1-3-D*, *S-2-3-D*, *S-2-4-D*, *S-2-4-5-D*, *S-2-3-1-D*) to reach *D* whereas existing distributed techniques allow only one: *S-1-D*, the shorstest one.

In section II, we first discuss about existing propositions or actually used protocols in order to highlight their limitations. Then, in section III we present our proposition and its two main aspects to compute and validate a maximum of loopfree paths. After presenting an evaluation and simulations results of our method in section IV, we conclude with a discussion about potential advantages of our distributed technique for load balancing and fast rerouting.

## II. Existing techniques

Two kinds of methods exist to achieve load balancing on multiple paths. The first category consists of source routing methods (deployed above traditional IP-routing) whereas the second category gathers distributed route computation protocols. Source routing multipath generally contains two stages for path positioning:

a) On source path computation algorithms (like K's best path [9], CRA [19], DSPA [20] or varied heuristics) to calculate efficient paths to reduce delays and improve throughput.
b) Path positioning protocol (RSVP-TE, CR-LDP ) for deploying calculated paths.

The main advantage is the ease of optimization to ensure end to end delay reduction or throughput increase, without considering loop presence. However, only ingress nodes which label or reserve path resources until the egress nodes are able to share the load on different paths, so the reaction time can be as long as the propagation delay on the return path. In addition, the extensibility in terms of Ingress/Egress router pairs using such techniques is limited. In an MPLS cloud, only border routers play this role in a reasonable perspective. [23], [14] and [8] are good examples of multipath source routing description.

Distributed multipath routing protocols can solve this problem, but they have to guarantee that each IP packet in transit between two routers cannot loop on any router of the route linking these routers. We have now to define the properties of loop free routing for distributed policies. We distinguish between "local" traffic coming from the router itself or directly attached subnetworks, and "transit" traffic coming from other routers. For simplicity we assume that they come from distinct interfaces.

*Definition 1 (Loopfreeness for distributed multipath routing):* A distribute multipath routing system is loopfree, if whenever a router *s* sends a local or transit packet to any next hop *v* towards a destination *d*, this packet never comes back to *s*.

| Notations | Definitions: |
|---|---|
| **G(N, E, w)** | Oriented graph G with a set of nodes N, a set of edges E and a strictly positive valuation w of edges. |
| **\|N\|,\|E\|** | respective cardinal of sets N and E. |
| $e = e.x, e.y$ | edge e ∈ E which connects node x to node y. |
| $k^-(x)$, $k^+(x)$ | incoming and outgoing degree of node x. |
| $succ(x)$ | direct neighbor set of x<br>y ∈ succ(x) if ∃ an edge $e = e.x, e.y \in$ E. |
| $P_j^m(s,d) = \{e_1, ..., e_m\}$ | $j^{th}$ best path of m hops linking s to d. Recursively, this is the best path whose first edge is distinct from the first one of the $j-1$ best paths according to metric C. |
| $C_j^m(s,d) = \sum_{i=1}^{m} w(e_i)$ | $j^{th}$ best cost (with lexicographical order) computed on s towards d $(1 \leq j \leq k^+(s)), (0 < m < \|N\|)$. |
| $NH_j(s,d)$ | $j^{th}$ best next hop computed on s towards d. This is the first hop $e_1.y$ of $P_j(s,d)$. |

TABLE I
NOTATIONS

Notations used in this document are given in Table I.
Several conditions[1] have been proposed for loopfreeness:

$$C_1(v,d) < C_1(s,d) \quad (1)$$
$$C_j(s,d) + c(s, NH_j(s,d)) < C_1(s,d) \quad (2)$$
$$C_j(s,d) = C_1(s,d) \quad (3)$$

Conditions 1 and 2 are similar in theory but their verification is done in a different way in practice. Condition 1 is verified in a distributed way, i.e *s* asks its neighbor for the cost of its best path to be strictly less than its own (see Loop Free Invariant condition in [24] or Loop free Alternate in [4]), whereas condition 2, described in [18], is verified on *s* for each path possibility, computed on *s* with a path finding algorithm without probing neighborhood. Condition 3, used by ECMP [16], just verifies that a $j^{th}$ path cost computed on *s* is equal to its best one (*j*=1). Each of these three conditions is sufficient but not necessary, so that the number of validated paths is usually not able to improve significantly the load distribution on a poorly connected network. Moreover, conditions 1 and 2 only validate equal cost paths if link valuation is homogeneous (number of hop metric) in the network. The condition used by ECMP does not consider all equal cost paths for a given destination, but only disjoint equal cost paths computed with Dijkstra algorithm. Condition 1 is used with Dijkstra computation too, but each router should ask its neighborhood to verify the condition, which again should ask its neighborhood to verify the condition and so on until the destination.

## III. Our proposition

In this section, we present our distributed multipath protocol in its two stages of loopfree multipath routing construction. The originality of our technique is the distinction made by the router on the flow origin (the incoming link) to direct it more precisely. Indeed, in the existing distributed propositions, the routing table does not consider where packets come from so

[1]Number of hop index *m* in table I only appears when it is useful

that routing table rows have to be the same for all packets having to reach the same destination. This is why the conditions (1),(2) and (3) have to be stricter than necessary. Figure 2 depicts our procedure to construct routing tables which take into account the interface through which packets arrive. The first step is to construct a candidate routing table with a Dijkstra modified algorithm we call "Dijkstra-Transverse" (**DT**). At this stage, we just consider the best first-edge disjoint paths for a given destination. Indeed, if computed paths on *s* have the same first hop for a destination *d*, then the router on which the routes diverge is able to benefit by itself from this disjunction.

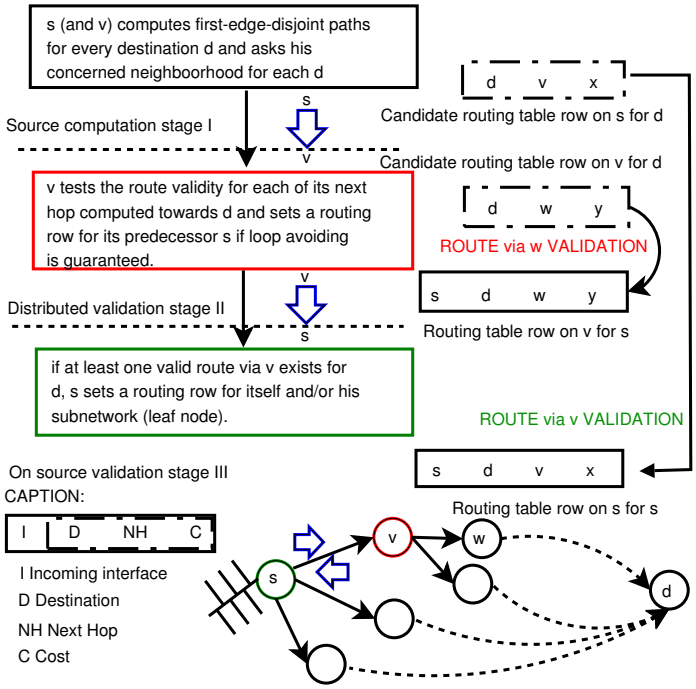


Fig. 2. Global protocol description

Figure 2 shows the three steps of the next hop validation. Every router *s* begins by computing (phase I) a set of multipath for each destination and defines, depending on these results, a candidate routing table. Then, *s* sends to its direct neighbors, *v* for example a request to test the path validity. Its concerned neighbor, here *v*, checks the feasibility of the route and returns an answer accordingly (phase II). The answer sent by *v* to *s*, allows *s* to decide if *v* can be a valid next hop for a subset of destination.

Before step II is reached, best equal cost candidate routing rows for a given destination are automatically validated for each incoming interface, so that the routing strategy is temporarily the same as ECMP. Our protocol can be implemented as an extension of a link state protocol such as OSPF while adding a path validation mechanism. The underlying Dijkstra algorithm used in OSPF to compute best paths is extended to calculate "transverse paths", as detailed in the next subsection. Routers then exchange information to validate, by successive adjacency on their direct neighborhood, transverse paths computed by Dijkstra-Transverse for their respective incoming interface.

| Terms: | Definitions: |
|---|---|
| **branch** $branch_h(s)$ | all best paths $P_1(s,d)$ in the best path tree which have the same first edge $\{s,h\}$. |
| **transverse** | an edge is transverse if it connects two distinct branches. |
| **P(n)** | function which returns predecessor of n in the best path tree. |
| **simple** transverse $Pt_j^m(s,d)$ | a path of m edges $\{e_1,e_2,...,e_m\}$ such that $\{e_1,e_2,...,e_{m-1}\}$ forms a best path $P_1^{m-1}(s,e_{m-1}.y)$ and such that $e_m$ is a transverse edge. |
| **backward** transverse $Pbt_j^m(s,d)$ | a path of m edges $\{e_1,e_2,...,e_m\}$ such that for a w i.e $0<w<m$, $\{e_1,e_2,...,e_w\}$ is simple transverse, and such that $\{e_m,e_{m-1},...,e_w\}$ is a best path $P_1^{m-w}(d,e_w.y)$. |
| **forward** transverse $Pft_j^m(s,d)$ | a path of m edges $\{e_1,e_2,...,e_m\}$ such that for a w i.e $0<w<m$, $\{e_1,e_2,...,e_w\}$ is either simple transverse or backward-transverse and such that, $\{e_m,e_{m-1},...,e_w\}$ is a best path $P_1^{m-w}(e_w.y,d)$. |

TABLE II
TERMINOLOGY

### A. Source computation

At this stage, we compute first edge disjoint paths, including at most one transverse edge, with a low complexity algorithm (like Dijkstra [7] in the worst case). The algorithm is shown below and definitions are given in Table II.

Our algorithm can be divided into three principal steps:

a) Calculate the best path tree (lines 14-17) and simple transverse paths (lines 18-20).
b) Construct a backward transverse paths set and add it to the previous set (lines 24-30).
c) Construct a forward transverse paths set and add it to the previous set (lines 31-37).

In addition, we must consider that edges in the best path tree are symmetric (duplex link) for backward transverse path computation (step b and c). In algorithm 1, the matrix Mc (Mc gathers all best paths costs $C$ for each possible next hop) has to be indexed by the outgoing interface to optimize steps 2 and 3, and by destination because in a real environment a router does not know at each instant the whole network topology, so that the concerned structure cannot be statically allocated. The complexity of our algorithm on a node *s* is in the worst case without optimized structure to modelize the first cost stack $T_f(n,0)$:

$$O(|N|^2+|E|+|N|\times k^+(s))=O(|N|^2)$$

but using a Fibonacci stack to modelize $T_f$, we may obtain:

$$O(|N|log_2|N|+|E|+|N|\times k^+(s))=O(|E|)$$

The update procedure for matrix Mc and Tf simply writes the most recently smallest distance tested (lines 15, 17, 27 and 34), on each concerned destination, considering the first next hop (necessarily a direct neighbor of *s*). The transverse paths (simple, forward or backward) contain, at the most, one transverse edge linking two branches of a best path

**Algorithm 1** Dijkstra-Transverse algorithm

1: **procedure** DIJKSTRA-TRANSVERSE($G(N, E), s$)
2: LOCAL:
3: $Mc(k^+(s), |N|)$: Cost Matrix for each next-hop.
4: $Tf(|N|, 2)$: Best path table (cost and first hop).
5: $P(|N|)$: List of father nodes.
6: $To(|N|)$: Stack of marked nodes.
7: INITIALIZATION:
8: Mc(k,d) and Tf(d,0) $\leftarrow \infty$, $\forall$ d, k.
9: Mc(k,s)$\leftarrow$ 0, $\forall$ k, Tf(s,0)$\leftarrow$ 0.
10: **while** $\exists$ n $\in$ N not marked **do**
11: $\rightarrow$Select smallest cost node x in Tf
12: $\rightarrow$Push x in To
13: **for all** $y \in succ(x)$ **do**
14: **if** Tf(x,0)+w(x,y) < Tf(y,0) **then**
15: $\rightarrow$Update Tl on y
16: $\rightarrow$ $P(y) = x$
17: $\rightarrow$Update Mc for destination y
18: **else if** Mc(Tf(x,1),x)+w(x,y)<Mc(Tf(x,1),y) **then**
19: $\rightarrow$Update Mc on y
20: **end if**
21: $\rightarrow$Mark x
22: **end for**
23: **end while**
24: **for** $i \rightarrow |N|$ to 1 **do**
25: **for all** $y \in succ(x)$ **do**
26: **if** Mc(y,To(i))+w(To(i),P(To(i)))<Mc(y,P(To(i))) **then**
27: $\rightarrow$Update Mc on P(To(i))
28: **end if**
29: **end for**
30: **end for**
31: **for** $i \rightarrow 1$ to $|N|$ **do**
32: **for all** $y \in succ(x)$ **do**
33: **if** Mc(y,P(To(i)))+w(P(To(i)),To(i))<Mc(y,To(i)) **then**
34: $\rightarrow$Update Mc on To(i)
35: **end if**
36: **end for**
37: **end for**
38: **end procedure**

tree. With this algorithm, we just compute all best paths (according to the first edge) which are able to link one branch to another in addition to the best path tree. In the next subsection we give some examples of transverse paths on the simple network drawn in figure 1. Note that node *S* also computes, with **DT**, a backward transverse path $Pbt_3^4(S,1) = \{\{S,2\},\{2,4\},\{4,D\},\{D,1\}\}$ and a forward transverse path $Pft_4^4(S,D) = \{\{S,2\},\{2,3\},\{3,1\},\{1,D\}\}$ but the matrix Mc is not updated (because router 2 already proposes a better cost: lines 26 and 34 of algorithm 1). These paths are not stored in matrix Mc because there are already better paths with the same first hop. More examples are given in [17].
In the next section, we will present our validation procedure to eliminate loops.

### *B. Distributed validation*

*1) One hop validation procedure:* Since paths are computed at the source, we enter the neighbor-node validation phase. At this moment, only the best cost equal paths are already valid by default without considering the incoming interface of flows. Now routers have to exchange best-path-cost information to validate other (simple, backward or forward) paths previously computed. A router *s* sends, for all destinations *d*, a message to its successor *v* (if *v* is a possible next-hop for *d* computed with **DT** on s) which contains the value of the best calculated cost path to reach *d*. If *v* proposes at least one valid path for *d*, it returns a positive answer to *s*, which validates the path via *v* to *d* with an incoming interface representing itself and/or its subnetwork for local traffic (see figure 2). The main advantage of our technique is the use of a candidate routing table use allowing to individually test the validity of each next-hop (corresponding to the first hop of a path $P_j$) proposed by an adjacent node towards the desired destination. Hence, loopfreeness condition can be expressed as:

$$C_j(v,d) \leq C_1(s,d), {}_{1 \leq j \leq k^+(v)} \tag{4}$$

This relation means that the first hop on a path $P_j$ (computed on *v*) is valid for packets coming from *s* to destination *d* (and so on for paths $P_k, k \leq j$). Indeed, if an adjacent node *v* to *s* guarantees a cost equal to the best one that *s* has, for a given destination, then, one hop further, *s* is sure that this path cost is strictly less than its own for the same destination. Moreover the path is validated on *v* for *s* as input (transit traffic from *s*) only if *v* validates the route for itself or its subnetwork. In figure 2, if the cost *y* proposed by *v* is less or equal to the best one on *s* (necessarily less than *x*), then the router *s* can use *v* as a possible next hop ( noted NH in the following) as soon as *v* has validated its next hop *w* in the same manner.
On the network represented in figure 1 there are two branches rooted at router *S* (by lexicographical search): $b_1(S)$=$\{P_1^1(S,1), P_1^2(S,3), P_1^2(S,D), P_1^2(S,3), P_1^3(S,5)\}$ and $b_2(S)$=$\{P_1^1(S,2), P_1^1(S,4)\}$. Then, *S* records with Dijkstra-Transverse computation two candidate routing rows [2] ($D|1|2$ and $D|2|3$ corresponding to $P_1^2(S,D)$ and $Pt_2^3(S,D)$ respectively) towards destination *D*. In the same way, router *1* has three paths (a best path $P_1^1(S,1)$=$\{1,D\}$ of cost $C_1^1$=1, a transverse path $Pt_2^2(1,D)$=$\{\{1,3\},\{3,D\}\}$ of cost $C_2^2$=2 and a path $Pt_3^4(1,D)$=$\{\{1,S\},\{S,2\},\{2,4\},\{4,D\}\}$ of cost $C_3^4$=4) for *D*. Router *2* has also three possibilities (candidate routing row $D|3|2$ corresponding to a best path $P_1^2(2,D)$=$\{\{2,3\},\{3,D\}\}$, $D|4|2$ corresponding to a transverse path $Pt_2^2(2,D)$=$\{\{2,4\},\{4,D\}\}$ and candidate routing row $D|S|3$ corresponding to path $Pt_3^3(2,D)$=$\{\{2,S\},\{S,1\},\{1,D\}\}$ ) for the same destination.

[2]These notations are defined in figure 2

Both routers *1* and *2* only validate the paths where costs are less or equal to two hops ($C \leq 2$), for packets coming from router *S*. Furthermore, router *3* only validates its direct path to *D* for packets coming from router *1*, whereas it validates its path via router *1* to *D* for router *2* in the incoming interface. In the same manner, router *4* validates its two paths $P_1^1(4,D)$=$\{\{4,D\}\}$ and $Pt_2^2(4,D)$=$\{\{4,5\},\{5,D\}\}$ towards *D* but not its transverse path $P_3^3(4,D)$=$\{\{4,2\},\{2,3\},\{3,D\}\}$ for router *2* as input. Finally, *S* benefits from six paths towards *D* thanks to two routing rows on S ($S|D|1|2$ and $S|D|2|3$) and two routing rows on routers *1,2,3,4* (respectively $S|D|D|1$ and $S|D|3|2$, $S|D|3|2$ and $S|D|4|2$, $2|D|D|1$ and $1|D|D|1$, $2|D|D|1$ and $2|D|5|2$) and one routing row on router *5* ($4|D|D|1$).

*2) p-hop validation procedure:* We have also developed a more complex protocol which tests the validity of a path (in fact path's first hop composition between adjacent routers) up to depth *p*. This procedure triggers a wave of requests on the candidate routing graph network rooted on the trigger router. Answers obtained by the return wave will make it possible to prune this graph with the insurance of destroying packet circuits more precisely as depth *p* increases. With this depth procedure, the number of exchanged messages depends on the degree of each router located between the source and the last router tested (*p* routers away from the trigger router at most). To minimize the number of validation messages, each router aggregates its requests according to destination, whereas answers are returned directly to accelerate validation convergence. Formally, if we note *K* the upper bound in router degree, the maximum number of validation messages noted $n_{sync}$ satisfies the following inequality:

$$n_{sync} < |N| \times (K^p + K^p \times |N|)$$

In order to avoid too much load in terms of messages we have decided to limit our depth-procedure to *p=2* and merge answers by destination because it is a waste of time to answer by destination individually with $p \leq 2$. We therefore obtain only $|N| \times (2 * K^2) \rightarrow O(|N| \times K^2)$ messages at worst on the whole network. However, in the next section, we analyze some results obtained with *p=3* to highlight advantages and inconveniences of the depth procedure. At this stage, we have to introduce the notion of "route" by as opposed to the term "path". With distributed (hop by hop) routing only the first hop of a computed path is actually used for routing.

*Definition 2 (Route):* Formally we denote a route of *m* hops which links a source *S* and a destination *D* :$R_m(S,D)$, and we note $NH(S,S,D)$, the set of validated next hops of computed paths with **DT** on *S*, for its local traffic. Hence, a route $R_m(S,D)$ is a composition of validated next hops (depending on the incoming interface: the link which connects the preceding router) and takes this form :

$$R_m(S,D) = \{r_1, r_2, \cdots, r_i, r_{i+1}, \cdots, r_m\}$$

with

$$r_{i+1} \in NH(r_i, r_{i-1}, D)$$

$NH(r_i, r_{i-1}, D)$ is the set of validated NHs on $r_i$ for $r_{i-1}$ as input (transit traffic).

Now we can describe the concept of loop avoidance with a distributed *Breadth First Search* loop detection (BFS dissemination) *p* node in depth.

**p-depth Loop Avoidance (p-LA)** principle:

This is a wave of messages triggered on each router (potentially all routers in the domain). The goal is to determine when a next hop is valid even if the adjacent router does not satisfy relation (4) for this NH. These messages contain the best cost, computed on the trigger router, to reach every destinations in the domain. We do not have to consider all Internet destinations but only routers belonging to the same domain, as depicted in the next subsection concerning extensibility. With this information a router can determine if it is able to forward data without loop, and it answers according to its costs computed with **DT**. The condition is the same as relation (4) but for all possible NHs composition at most *p* router in depth (each end $r_k$ of a partial route of *p-1* elements $\{r_2, \cdots, r_k\}, _{k \leq p}$ has to verify this relation).

Indeed, each router only takes care of potential loops back to itself, so that all neighboring nodes (potentially $K^p$) have to guarantee, in a radius of $p-1$ at most, a cost for *D* less or equal to the best one calculated on *S*, for all NH's possible composition computed with **DT**. It means, that one hop further, we are sure that the cost is strictly smaller than the best one on *S* for all possible NHs , because each link valuation is strictly greater than zero. A router, except *S*, can appear twice or more in the validation phase, made with BFS, because it will be its responsibility to avoid loops coming back to itself. The BFS dissemination is triggered one hop after the neighbor, so it must look for every NHs computed by composition of **DT** on at the most $K^p$ routers. The validation search triggered by a router *S* (which really begins on a router $r_1 \in NH(S,S,D)$ ) has to explore, in a radius of p-1, all path compositions to test every possible routes (on subgraph generated by **DT**). A router $r_j$ is able to select a particular NH only with adjacent routers ($r_{j+1}$) because of the distinction done on incoming interface by one-hop neighborhood (as depicted for one hop validation).

Description of forward wave:

If an adjacent router $r_1$ does not satisfy relation (4) (validation requirement) on some of its candidate NHs computed with **DT**, for a given destination *D* and a trigger source *S*, it then sends a request to all its NHs for which the condition is not verified. The routers queried by $r_1$ do the same until depth *p* is reached or validation is obtained, and return an answer in case of loopfree success. Router $r_k$, $1 < k \leq p$, records a positive response for their candidate NHs which verify relation (4).

Description of returned answers:

The returned answer contains a null or a positive result, for a given pair *(S,D)*. These responses are gathered for all possible candidate NHs, depending on this pair *(S,D)*, and returned to the router which asked the loopfreeness for *S* (the father node in the wave triggered by *S* on its direct neighbor). The answer are returned to father node only if, for a given pair *(S,D)*, the

concerned router:

1) receives at least one positive response for one of its NHs.
2) receives responses from all its candidate NHs.

Father node can in its turn transmit a positive response if it verifies these two items, and so on until *S*. If direct neighbor $r_1$ receives a positive answer, it can compute a new valid routing row and transmit this information to *S*.

The value contained by the end-response computed on router $r_k, 1 \leq k \leq p$ is:

a) positive if the router verifies relation (4) for all its NH computed towards *D*.
b) null if the router creates a loop not coming back to *S*.
c) or there is no response (because a loop comes back to *S*).

*Properties 1 (p-LA Loopfreeness):* The p-depth Loop Avoidance procedure constructs a set of routes between each pair of routers which is loopfree in a stable state network.

*Proof 1 (p-LA Loopfreeness):* Formally with the **p-LA** procedure, a router $r_i$ in a route $R_m(S, D)$ guarantees, for every destination *D*, that:

(a) No path composition in a maximum radius of depth *p* can come back to $r_i$.
(b) Each router $r_{i+k}, 1 \leq k \leq p$ at the end of a path composition proposes, for all its NH, a cost less than its best one towards *D*. That means routers $r_{i+k+1}$ propose a best cost strictly less to the best one that $r_i$ does, because the edge valuation is strictly positive.
(c) Each direct neighbor $r_{i+1}$ activates routing rows positioned by $r_i$ (transit traffic) only if they are already (or later) activated for its local traffic.

The last item means for a given *D*, if there is a row for transit traffic there is also a row for local traffic with the same NH. Hence, demonstration is still valid if we just consider loop on *S* because each router must verify the same conditions. In figure 3 we have represented the demonstration basis with $i = 0$ ($r_0 = S$ denoted *r* in the figure) whereas in the following, we generalize the demonstration for every router on any route linking *S* to *D*.

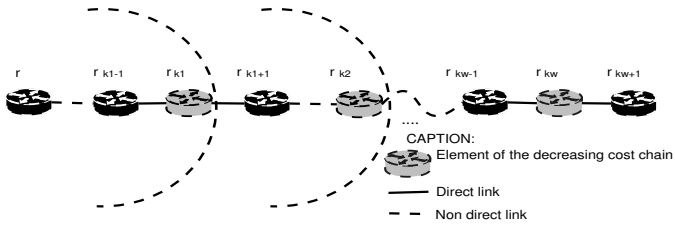


Fig. 3. Decreasing cost chain

If we admit that a route $R_m(S, D)$ contains a loop on a router $r_i, 0 \leq i \leq m$, it means there exists j on the route i.e with $i < j$: $r_i = r_j$. Yet, we know that a router $r_{i+k1}, 1 \leq k1 \leq p+1$ guarantees a cost strictly less than $r_i$ (condition (b)) so it cannot be $r_i$. In the same way, we know by transitivity of relation "$<$" that there exists a chain of strictly decreasing best costs (see figure 3), towards *D*, on routers $\{r_{i+k2}, r_{i+k3}, \cdots, r_{i+kn}\}$ ($(kw - k(w-1)) \leq p+1$, $i + kn \leq m$ ), so that none of these routers could be $r_i$, and one can deduce that the router $r_j$ is positioned somewhere behind a router $r_{i+kw}$. So router $r_j = r_i$ has to guarantee, before exploring $r_{i+k(w+1)}$ or worse $r_{i+kw}$, either a strictly lower better cost, or that all its NHs composition, (not validated NH but **DT's** computed NH) proposes a cost less or equal to the best cost on $r_{i+kw}$. It should then explore with BFS, $r_{i+k1}$, because the best cost of $r_i$ is strictly greater, and in the same manner $r_{i+k1}$ should explore $r_{i+k2}$, and so on until $r_{i+k3} \cdots r_{i+kw}$, but it is impossible because of conditions (a) and (c). The only position that could be occupied by $r_j$ is somewhere one hop behind $r_{i+kw}$ (router denoted $r_{kw+1}$ in figure 3) because this one can select the next hop for its traffic, one by one, without exploring all the subgraph generated by **DT** composition between adjacent nodes. Now, we just have to focus on the router $r_{i+kw-1}$ behavior: it cannot select $r_j$ as a viable outgoing interface for *D* on its neighbor $r_{i+kw}$. Indeed, if it puts a new entry for its traffic towards *D*, it should explore itself to prove the route validity (condition (a) is then not satisfied), because either its best cost for *D* is strictly less than $r_i$'s one for $w > 1$, or for $w = 1$ because $r_i$ has precisely done the same exploration to guarantee an equal cost for *D*. Consequently, $r_j \neq r_i$ and a validated route does not include loops. To finish, the last router of a route $R_m(S, D)$ is obviously $r_m = D$.

The routers included in the decreasing cost chain play a main role in order to control the exit of a loop. They have to guarantee a strictly decreasing best cost for *D* so that they are progressively closer to *D*. This is why a packet cannot come back to a router it already crosses.

This demonstration remains valid even if all routers choose a different value of depth *p* for their validation phase. If a router is not configured for Incoming Interface Multipath Routing, it can work with the others with its best outgoing interface only if precedent adjacent routers use it (without validation phase) through one of its best NHs.

*3) Stable state definition:* A stable state for a network can be defined by the fact that each router has the same information about the whole topology. So if topological changes (caused by failure or new link/router) are sufficiently separated in time, every router obtains all information necessary at the end of a fairly short time. We just have to ensure that signalization (Link State Advertisements and validation messages for route validation) packets always take the same route to be sure that information is passed on in the order of appearance. To ensure this property, every signalization message is always aimed towards the same next hop for each destination. We also have to check that old information is ignored by a numbering system of validation messages. Moreover, routes are available as soon as **DT** has been computed following the arrival of a new Link state advertisement (LSA) arrives, even if the incoming interface is not in use. The routers send the traffic on their best cost paths for all destination until the validation messages are back

to permit multipath routing. We also have to guarantee that the validity test is made with coherent information. Indeed, routers do not have to know more topological information than their neighbors in order to have a coherent network view. Routers can add some routing table rows between each other for their entry interfaces only if they have a same topology view. To prevent incoherence, neighbor nodes must check if the validation messages they receive take into account the last "Link State Advertisement" they have received. If not, they can discard this message. To avoid this issue, the last validation message on one-hop neighborhood removes older entries which concern the same incoming interface. And the last received LSA reset old entries for local traffic.

*4) Mechanisms for extensibility:* Hopefuly, the routing table does not have to consider every destination on the Internet. Border routers announce external destination prefixes and their exceptions to every network router. Hence, if there are multiple possible border routers for the same destination (or rather destination prefix), the multipath module can take advantage of this multiple solution associated with the multiple paths computed for concerned border routers. Our procedure just considered internal routers of its domain as a destination in **DT** computation and validation stage. Basically, as OSPF does also to route towards external destinations, we use an indirection table between IP prefixes and borders routers to perform routing. However, when several border routers announce the same prefix, we have to find a new mechanism to avoid loop because *p-LA* only works for a unique destination. This is not in the scope of this document, but we currently work on this issue. Consequently, the extensibility in terms of computation complexity and load depends of the domain size. Nevertheless, our procedure can be used independently between neighbor domain, without needing any further extensibility if border routers do not announce identical prefixes.

## IV. Evaluation and Simulation

### A. Experiments description

We used Network Simulator 2 (ns2 [1]) to evaluate our technique. We have implemented an extension of the link-state routing protocol provided by ns2 and extended the node's routing classifier attributes. We first evaluated performance of our protocol in topological terms, on the French renater4 [2] topology (see figure 4 for a simplified map) and on three other backbone networks (Opentransit, Alternet and GlobalCrossing). Our simulations and evaluations do not study data traffic, but only analyze the routing capacities on an empty network. Hence, we do not have to run several simulations to compute an average value with a given confidence interval because presented results are determinists (we do not introduce traffic with a random generator to analyse a particular traffic distribution). We present results according to node degree and we compute, with ns2 and our extension for multipath routing, the total number of loop free open routing rows (computed by

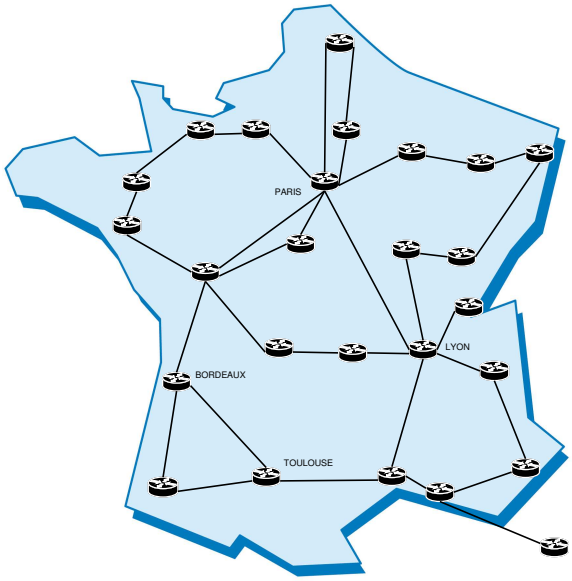


Fig. 4. An approximation of Renater4 topology

**DT** and validated with **p-depth Loop Avoidance**) depending on the node degree. We evaluate this parameters on these four networks, and give the results in the next subsection. These network topologies have been obtained through the "mrinfo" tool. For networks where native multicast routing is enabled and "mrinfo" is not filtered, this tool gives precise maps of router interconnections (see [21]). Renater, Opentransit, Global Crossing and Alternet are such networks. Network topology characteristics are gathered in table III. We have considered

| Network name | Node number | Edge number | Average degree |
|---|---|---|---|
| **Renater** | 78 | 198 | 2.54 |
| **Open Transit** | 76 | 206 | 2.71 |
| **Alternet** | 83 | 334 | 4.02 |
| **Global Crossing** | 102 | 370 | 3.62 |

TABLE III
Evaluation networks

each link as symmetric in valuation and existence (all links are duplex links). The valuation of each link is considered equal so that the metric is equivalent to the number of hops. The presented results for the "Loop Free Alternate" (LFA [4] or "Loop Free Invariant" in [24]) solution are evaluated as our incoming interface proposition. We have considered, for each pair of source and destination, the incoming link used for packet transit even if it is not actually taken into account. Indeed, for LFA proposition, there is no difference in treatment of transit traffic or local traffic. In figure 5, we have represented a particular case where routers noted *A* and *D* (which have a degree of 3) cannot serve for transit traffic with LFA technique. If we consider that all links are equally valuated, then no router can route its traffic via *A* or *D*. Alternate paths via these

routers cannot, for any destinations in the network except their own, verifies the relation (2). Hence, when LFA is used on a

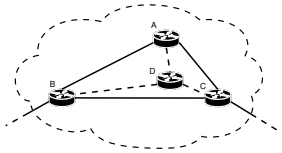


Fig. 5. An example of router triangle configuration

network, some routers cannot route traffic for neighbor routers. That is why, in figure 6 given in next subsection, some routers do not appear in router distribution with LFA utilization. Our validation protocol permits to use each router (with degree at least 2) for transit traffic in the four evaluation networks.

### B. Results

Our evaluation topologies tries have different kinds of degree distribution (see figure 6). The first two (Renater and Opentransit) seem to have a power law distribution whereas the last two seem to be more uniform. The dashed boxes represent the number of nodes, whose degree is given in x, which accept validation for transit traffic with condition (2). Our procedure allows using every router for packet transit even with *p=1*.
The first evaluation (whose results are illustrated in figures 7, 8, 9 and 10) computes the average of the total number of routing table entries by router, having the same degree, for all pairs {destination, incoming interface}. When the relation (2) is used, each valid row in the routing table of a router *r* is the same for local traffic and for routers linking to *r* and which have selected router *r* as a valid NH for transit traffic. That is why to be fair with lfa, we compare this value only to the open rows for transit traffic rather than the routes open for local traffic.

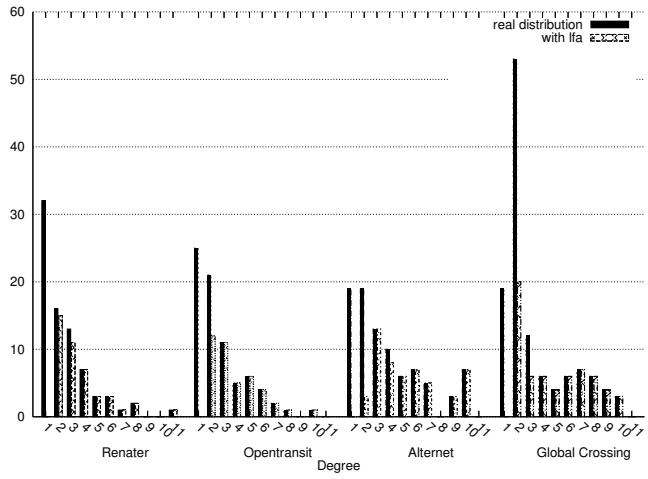


Fig. 6. Degree node distribution for each tested network

On figure 6, we can already notice that our technique allows using much more routers for multipath routing than with condition (2), especially for routers which have only two duplex links (except for Renater). The reason is that every router of degree 2 which is an element of a router triangle (see figure 5) can never be used by its neighbor for multipath routing if the metric is the number of hops as in our study. The number of routes linking a source *S* and a destination *D* depends entirely on the number of valid next hops on intermediary routers between *S* and *D*. This is why we focus our evaluation on the number of entries in the routing table for preceding router (transit traffic).

*1) Number of validated next hops:* This study mostly compares the use of condition (2) presented in section 2 against our more flexible condition in order to highlight, in various topological cases, the advantages of our method (in next four figures, boxes are superposed). For these preliminary results we do not take propagation delays into account. We note a couple (incoming interface, destination) : $(I, D)$, the y-axis represent the average of the total amount of validated next hops for all couple $(I, D)$ on each nodes whose degree is given on x-axis. Moreover, when LFA is used, the routing table does not include as many entries as shown as the figures, but we have multiplied the real number of entries computed for local traffic by the number of incoming interfaces of concerned routers.
This average can reach at most:

$$k^{-}(x) \times k^{+}(x) \times |N|$$

This means, on the network considered in figure 7, that a router of degree 8 has one the average 1200 possible routing entries for all couple $(I, D)$. It can be interpreted in the following way: a router of degree 8 has $\frac{1200/8}{77} \approx 2$ valid routing entries by couple $(I, D)$ (*77* is the number of destinations in Renater). This does not necessarily concern all destinations and incoming link because a router is not necessarily a valid outgoing link for a couple $(I, D)$. This is why, in reality, a router of degree 8 has much more than two valid next hops by couple $(I, D)$ with our procedure. In the same way, the LFA proposition validates less than one next hop by couple $(I, D)$. On the four networks we can observe that the number of entries in routing table increases depending on the router degree. More precisely, we notice that our technique increases faster according to router degree than with LFA solution. This means that our method does not multiply routing table size only because of the number of incoming interface considered, but also because it benefits from the number of possible outgoing links.
In the one hop validation (*p*=1) version, we already notice, for all simulated networks, an impressive increase of routing possibilities compared to LFA solution. In figure 7 and 8, we notice that our method computes much more valid routing entries although these networks are poorly connected (less than 3 duplex links per router on average). Moreover, the ratio between routing entries computed with our method and LFA technique increases approximatively in a linear way according to router degree.

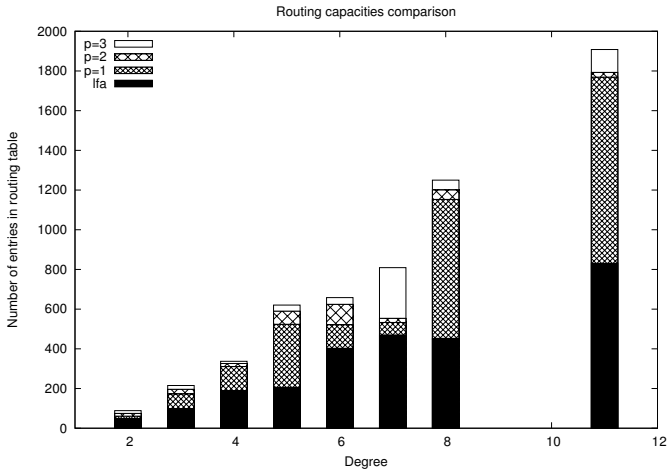


Fig. 7. Renater: 78 routers, 198 links

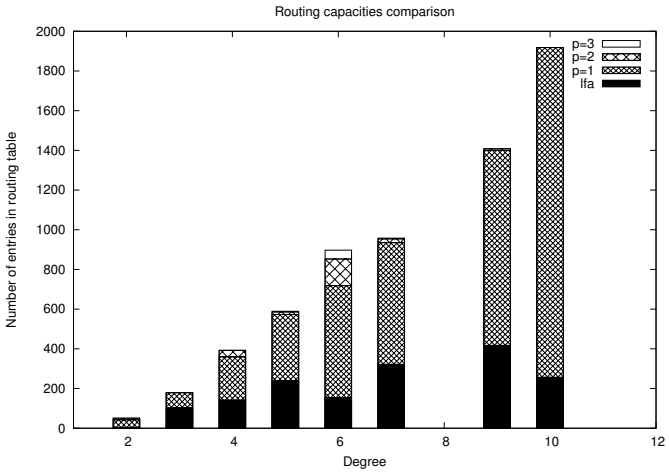


Fig. 9. Alternet: 83 routers, 334 links

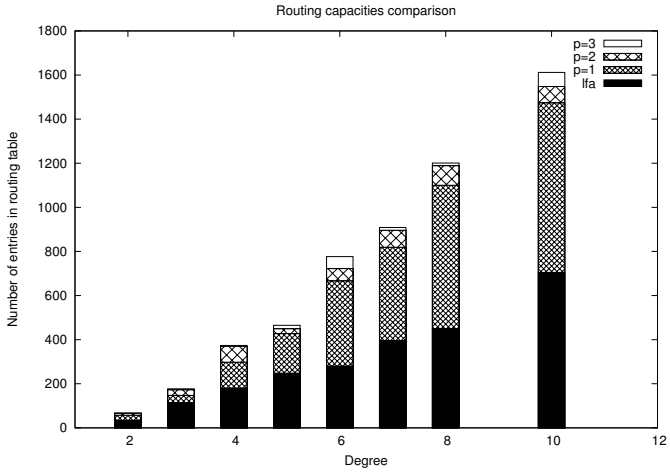


Fig. 8. Opentransit: 76 routers, 206 links

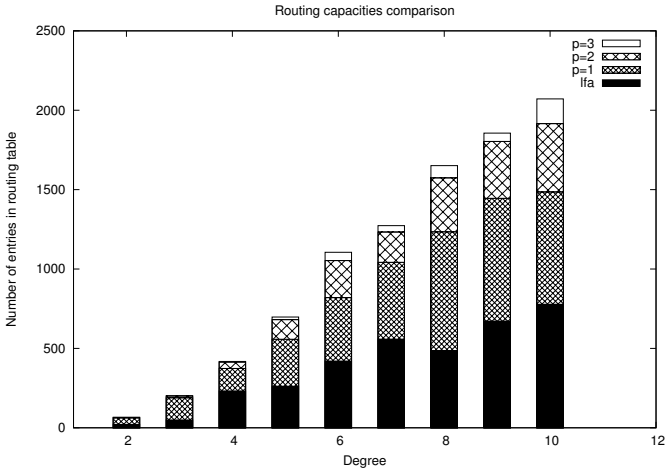


Fig. 10. Global Crossing: 102 routers, 370 links

The last two networks (figures 9 and 10) are larger and more connected backbones (an average larger than 3 duplex interfaces per router). Hence, in comparison with less connected networks, the number of routing table entries is often larger for a given degree.
We also notice on these networks a greater benefit in routing entries validation with the depth validation procedure (p>1) especially with routers of high degree (see figure 8 and 10). However, computation complexity and signalization overheads generated by p-depth validation does not bring sufficient advantages to be really useful particulary on networks such as Renater and Alternet.

Our protocol significantly improves the number of validated outgoing NHs and improves the distributed redirection capacities of traffic in the case of congestion or failure. Furthermore, maximum flow can potentially increase as compared with existing propositions, so it ensures a greater potential throughput between each couple *(S,D)*. Indeed, for each pair *(S,D)* the number of routes validated with our method is always larger than with existing distributed propositions. The number of routes between routers *S* and *D* depends on the number of validated NHs (for concerned transit traffic) positioned on routers which are able to join *S* and *D*. For example, on Renater network, our method (with *p=3*) validates 17 routes from Strasbourg to Pau ranging from 6 to 11 hops. For routers close to each other, some validated routes can be very much longer than the shortest one. This kind of routes is probably only useful for security and restoration aspects whereas the shortest routes are more suitable for load balancing.

*2) Convergence time:* We have worked out two scenarios to test the convergence time to achieve stability after failure on Renater 4 topology. Obviously, we choose failures keeping

a connected network. The first one is a central link failure (see figure 11) whereas the second one is a border link failure (figure 12). The payload of each signalization messages, LSA and validation messages, is simulated with ns2 according to the network size. We consider in our simulations that the validation packet size is proportional to |N|. We have also simulated the real propagation delays of each link. For that, we have used an orthodromic method which allows us to determine the physical length of each link fairly precisely. We consider that each link bandwidth on the network is equal to 2,5 Gigabits/s. However, we have not emulated the computation time either for **DT** overhead compared to "Dijkstra" or for validation tests because we consider it negligible. Both graphs in figure 11 and 12 represent the simulation time on the x-axis and each horizontal line (on the y-axis) represents a router ordered by LSA reception time. A row represents the detection period and validations periods for an unique router. The arrows with heads on both figures 11 and 12 show the duration of the validation periods whereas the simple arrow shows the LSA reception period. Considering that validations are distributed between neighbors (each router writes some new routing entries for its neighborhood), we have considered all validation messages for both local traffic and transit traffic. This is why validation of depth 1, received by a router r, can fairly begin "at the same time" (just after thanks to a synchronization mechanism to ensure that the LSA is treated first) that r receives its first LSA. Duration times given in presented results ignore the unsuccessful validation messages.

In the first scenario the connection (two duplex links) linking Lyon to Paris falls down after 5 seconds whereas in the second scenario there is just one duplex link failure between Bordeaux and Toulouse. The total validation phase lasts less than 20 ms to produce the entirely new routing table on every router in both scenarios.

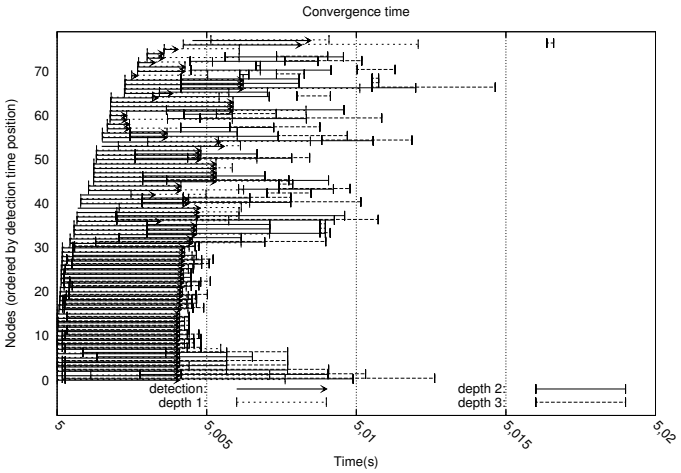


Fig. 11. Convergence time on Renater after link PARIS⇔LYON fails

We observe that there are more successful validations in small depth as we have already noticed in the precedent subsection. Moreover, we also observe that the validation procedure only slightly lengthens the time of convergence: approximately and at worst, 3ms for a depth of 1, 6ms for a depth of 2 and 10ms for a depth of 3. Two main phases should interest us. The time between the failure and the last LSA received, and the time which separates this one from the end of the last validation phase according to the search depth. These periods will be more sensitive to propagation delays in large networks especially for detection period because the network maximal distance, its diameter, is generally higher than the maximal depth we use for the validation protocol. On the Renater network, computed link propagation delays are always less than 2ms and often still much smaller. However, for $p > 2$ the validation convergence time already begins to become important for not so many validated routes added. This is why we can conclude that *p=2* is a good setting to obtain the best ratio between the number of validated NHs and the amount of time to achieve stability. The main difference between the two simulation experiments is the failure localization. Indeed, detection period and position is conditioned by failure learning following a LSA reception. The LSA reception order depends as well on the position of the link which fails and on the network topology characteristics as well. However, the failure localization does not seem to affect the validation phase period.

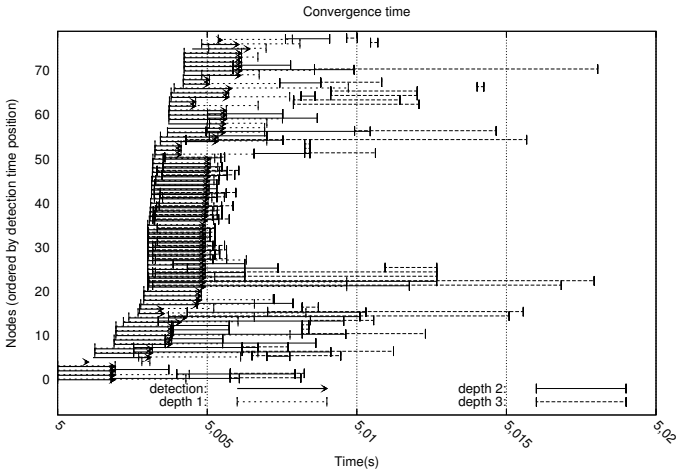


Fig. 12. Convergence time after link BORDEAUX⇔TOULOUSE fails

*3) Discussion:* During validation time and especially during detection period, load balancing cannot be optimal because a router does not know entirely its routing table. In detection phase the presence of loops is still possible as with OSPF transient state. Nevertheless, we can easily imagine a mechanism of traffic balancing which shifts the load on an alternate NH even before the best NH for a destination is computed. Indeed, if routers have good probing measures for each of their NH's, they can decide to shift the load before receiving

a LSA. Hence, two mechanisms for re-routing can coexist. The first one allows avoiding failure closest to where it occurs, and the second one optimizes route computation and use. These mechanisms are complementary and allow the network to react quickly to congestion and to recompute optimal routes in a longer time frame. We just have to lay down the adequate load distribution policy according to measured load oscillations. [12] is a good analysis of convergence time issue for fast rerouting in order to achieve IP restoration. A main goal of TE is also to use different routes for different Quality of service (QoS) requirements. The diversity of validated routes can really be useful in spite of their length. Indeed, our method computes routes which are sometimes much more costly than the best one, but which can really appear important for protection and restoration as well as for some TE aspects as QoS.

## V. Conclusion and perpectives

In this article, we have presented our distributed multi-path routing scheme. Our technique computes and validates a number of routes larger than with existing propositions because of the distinction made on the flow entries: the incoming interface. Potentially with a good load balancing policy, the throughput between a pair of nodes can improve thanks to the larger number of routes. The associated overhead is proportional to the routers degree so that the routing table is a function of the number of incoming interfaces and the number of viable next-hops. The validation phase is a very light procedure in its basic version (one or two hop validation procedure), but can greatly increase the number of viable next hops for each destination. We have shown with ns2 that the convergence time to achieve routing stability when failures occur is fairly close to an unipath link-state protocol. We have also demonstrated the loopfreeness of our procedure when routers are not set with the same depth parameter or even not set with multipath routing at all. Our future work will focus on router communication to provide online load distribution with real-time traffic measurements. In this perspective, we will try to develop some mechanisms for sharing traffic over multiple paths without leading to oscillations and TCP mis-ordering issue. Another goal for our work is to see if it is possible to extend it to wireless environment (such as ad-hoc network) which may benefit even more from the multipath routing advantages, but is more sensitive to the convergence time issue because of mobility.

## References


[1] "The network simulator- ns2, http://www.isi.edu/nsnam/ns."

[2] "Réseau national de télécommunications pour la technologie, l'enseignement et la recherche, http://www.renater.fr."

[3] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, "LDP Specification," RFC 3036, Jan. 2001.

[4] A. Atlas and A. Zinin, "Basic specification for ip fast-reroute: Loop-free alternates draft-ietf-rtgwg-ipfrr-spec-base-05," Internet Engineering Task Force," Internet Draft, Feb. 2006.

[5] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "Rsvp-te: Extensions to rsvp for lsp tunnels," Internet Engineering Task Force, Request For Comments 3209, Dec. 2001.

[6] R. Braden, L. Zhang, S. Berson, and S. Herzog, "Resource reservation protocol (rsvp)," Internet Engineering Task Force, Request For Comments 2205, Sept. 1997.

[7] E. Dijkstra, "A note on two problems in connection with graphs," *In Numerische Mathematik, vol. 1, pages:269-271*, 1959.

[8] A. Elwalid, C. Jin, S. H. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," in *INFOCOM*, 2001, pp. 1300–1309. [Online]. Available: citeseer.ist.psu.edu/elwalid01mate.html

[9] D. Eppstein, "Finding the k shortest paths," in *IEEE Symposium on Foundations of Computer Science*, 1994, pp. 154–165. [Online]. Available: citeseer.ist.psu.edu/eppstein97finding.html

[10] I. Gojmerac, T. Ziegler, and P. Reichl, "Adaptative multipath routing based on local distribution of link load information," in *Proc. 4th COST 263 International Workshop on Quality of Future Internet Services*, 2003.

[11] C. Hopps, "Analysis of an equal-cost multi-path algorithm," Internet Engineering Task Force, Request For Comments 2992, Nov. 2000.

[12] G. Iannaccone, C.-N. Chuah, S. Bhattacharrya, and C. Diot, "Feasibility of ip restoration in a tier-1 backbone," *IEEE Networks Magazine, Special Issue on Protection, Restoration and Disaster Recovery*, March 2004.

[13] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, and N. Feldman, "Constraint-based lsp setup using ldp," Internet Engineering Task Force, Request For Comments 3213, Jan. 2002.

[14] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2005, pp. 253–264.

[15] Z. Ma, P. Zhang, and R. Kantola, "Influence of link state updating on the performance and cost of qos routing in an intranet," in *IEEE Workshop on High Performance Switching and Routing*, Dallas, Texas, USA, 2001.

[16] J. Moy, "Ospf version 2," Internet Engineering Task Force, Request For Comments 2178, Apr. 1998.

[17] P. Mérindol, J. J. Pansiot, and S. Cateloin, "Mutiroutage par interface d'entrée," in *ALGOTEL '06*, 2006.

[18] P. Narvaez and K. Y. Siu, "Efficient algorithms for multi-path link state routing," in *ISCOM'99*, Kaohsiung, Taiwan, 1999.

[19] S. Nelakuditi and Z.-L. Zhang, "On selection of paths for multipath routing," in *Proceedings of IWQoS*, 2001.

[20] H. S. Palakurthi, "Study of multipath routing for qos provisioning," October 2001.

[21] J. J. Pansiot, "Local and dynamic analysis of internet multicast router topology," *Annales des télécommunications*, vol. to appear, 2006.

[22] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching (mpls)," Internet Engineering Task Force, Request For Comments 3031, Jan. 2001.

[23] C. Villamizar, "Mpls optimized multipath (mpls-omp): draft-villamizar-mpls-omp-01.txt," Internet Engineering Task Force," Internet Draft, Feb. 1999.

[24] S. Vutukury, "Multipath routing mechanisms for traffic engineering and quality of service in the internet," Ph.D. dissertation, University of California, Santa Cruz, Mar. 2001.