Providing Protection and Restoration with Distributed Multipath Routing

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

***Abstract*—Multipath routing is an interesting tool to provide a fast reaction time to protect networks from failure or congestion. Indeed, a local alternate path computation allows to faster re-route the traffic without flooding the entire network with Link State Advertisements. The restoration depends therefore on the protection guaranteed by each router. Distributed techniques allow to entrust the potential restoration to each router where it is possible. We distinguish multipath routing and fast rerouting techniques to underline the possibility to use alternate routes for load balancing or just as backup solution. In this paper we first summarize our *incoming interface multipath routing* technique and then analyze its capabilities in terms of protection. We evaluate several routing techniques to achieve a good coverage. Results indicate that our hop by hop routing multipath protocol performs almost as well as the best unipath rerouting technique whereas it can also be used for proportional routing.**

***Index Terms*—Fault tolerance, Multipath Routing, Fast re-routing, Coverage, Path computation.**

I. Introduction

Protection and restoration add a new layer of reliability, integrity and availability for the network resilience on every routers they are applied to. Protection is commonly defined by pre-provisioning a backup path. This ensures a quick recovery time because the path is computed, but not necessarily activated, before failure. When a failure occurs, a restoration protocol computes an alternate path on demand, or just selects a pre-computed alternate path. IP's Interior Gateway Protocols (*IGP*), such as OSPF [12] or ISIS [14] have to flood the entire network when topology changes (through Link State Advertisements, *LSA*) and each router needs to recompute all paths with a *SPF* (Shortest Path First) algorithm. However for real-time service level requirements, the reaction time might be too long to offer a sufficient quality of service. Applications such as VoIP are particularly sensitive to packet loss, therefore a fast restoration scheme is vital to avoid disruptions. A restoration time lower than 50 milliseconds became a reference [6] to guarantee the efficiency of recovery techniques. There are two simple ways to reduce the time which is necessary to compute new routes. The first consists in segmenting the network in multiple areas because there are fewer routers concerned by failures. The second one aims at minimizing the time period between two consecutive *Hello* messages to reduce the failure detection time. Even though today links speed and *CPU* power are considerable, these two techniques are generally insufficient to reach a short reaction time. Another research area is the optimization of the time used to recompute the routes which have a failed link (for example the incremental *SPF* given in [10]) or the optimization of the *FIB* (Forwarding Information Base) updating procedure. All these techniques have to be paired with pre-computed alternate paths.

The reaction time depends on three elements :

a) Failure detection (timers issue, *SDH* alarms)
b) Failure notification (link state broadcasting, topological database updating)
c) Re-routing (path restoration and *FIB* updating)

In this article we focus on path provisioning protocols to accelerate the rerouting phase. We specifically study recovery based on *IGP* such as IP fast re-routing. First, we analyze the number of validated loop free alternate routes. Then, we weight up local recovery capabilities (on the router which detects the failure) as compared to global recovery capabilities, on routers which are located upstream from the failure. The main difficulty to achieve an efficient protection with real multipath routing, compared to unipath routing, is that an IP packet can be forwarded on several routes for a given destination even before any failure occurs. Hence, routes are not only designed for protection aspects but for load balancing too. We therefore cannot consider that, in alternate routes computation, a router forwards traffic only via the shorstest route. Conditions used to validate loop free alternate routes have to take into account the fact that routers may forward packets through different next hops for the same destination.
In the first section we introduce a brief state of the art of most common techniques using multipath routing. Then, we present our distributed multipath routing technique DT(p). In the third section, we analyze, with several topologies, different IP routing techniques characteristics and advantages. We conclude with a discussion about future extensions to provide a global mechanism of restoration and protection.

II. Resilience and fast re-routing

Different layers of protection and restoration can be used to protect Internet communications. In this section, we focus on network layer protection schemes such as *MPLS* (tunnels above link layer), multipath routing and IP rerouting techniques. Two principal kinds of routing schemes exist to achieve

protection and restoration on multiple paths at the network layer level. The first category gathers source routing methods (back-up tunnels built as an overlay network above the link layer) whereas the second category gathers distributed routing methods. Here, we do not consider the load balancing issue.

A. Multipath source routing

Source routing multipath techniques are generally made up of two components for path provisioning :

a) Path computation algorithms (such as *K's* best path [7] or *CRA* [13]) to compute efficient protection paths.
b) Path signalling protocols (such as *RSVP-TE* [5] or *CR-LDP* [9]) to position these computed paths.

The main advantage of this technique is to easily choose backup tunnels, without considering loop presence as in distributed methods. Indeed, it is important that the bypass tunnel guarantees the bandwidth requirements assumed by the primary path. However, with path protection, only ingress nodes which label or reserve path resources until the egress nodes are able to shift the traffic from one path to another. So the reaction time can be as long as the propagation delay on the return path. With link or span protection, the reaction can be faster, but does not scale very well, since the number of bypass tunnels can quickly become too large (and similary, the signalling messages overhead). Consequently, even with path protection, the extensibility in terms of Ingress/Egress routers pairs using such techniques is limited. In an *MPLS* cloud, only border routers can play this role in a reasonable perspective. Moreover, with path protection, a single link failure can affect simultaneously several primary paths, resulting in a large amount of signalling messages as depicted in [2].

B. Distributed routing

The second category gathers IP hop by hop routing methods which can partially solve these problems. However they have to guarantee that IP packets in transit will not loop. Table I gives the definitions used to express the loop free routing property. Any of the conditions (1), (2), (3) or (4) can be used to avoid loops. *v* and *p* are adjacent to *s*, *v* is a downstream router, whereas *p* is an upstream router to *s* for a given destination *d* (Figure 1).

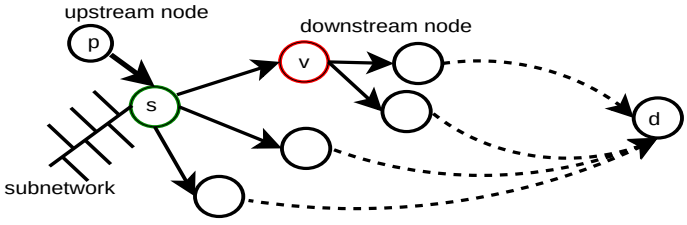

Figure 1: Loop free routing

These conditions are sufficient but not necessary to find loop free alternate routes. Consequently, on poorly connected networks, a loop free alternate route (or rather associated next hop, denoted NH in the following) cannot always be found even if one exists. The loop free routing property on *s* with for neighbor *v* can be expressed as :

$$C_j(s,d) = C_1(s,d) \wedge v = NH_j(s,d) \quad (1)$$
$$C_1(v,d) < C_1(s,d) \quad (2)$$
$$C_1(v,d) < C_1(p,d) \quad (3)$$
$$C_1(v,d) < C_1(s,d) + C_1(v,s) \quad (4)$$

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.
$\{e.x, e.y\}$	edge $e \in E$ which connects node x to node y $e^{-1} = \{e.y, e.x\}$ is in the opposite orientation.
$k^-(x), k^+(x)$	incoming and outgoing degree of node x.
$P_j(s,d) = \{e_1, ..., e_m\}$	j^{th} best path linking s to d. Recursively, this is the best path whose first edge is distinct from the first edge of the $j-1$ best paths.
$C_j(s,d) = \sum_{i=1}^{m} w(e_i)$	j^{th} best cost 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 $P_j(s,d)$.
$NH(p,s,d)$	set of next hops validated on the router s for the upstream router p as input and towards destination d.

Table I: Notations

1) Multipath routing: Condition (1), used by *ECMP* (Equal cost multipath extension of routing protocol such as OSPF and IS-IS), verifies that a j^{th} path cost computed on *s*, is equal to its best one with a very simple enhanced *SPF* algorithm. For example, in figure 2 (we consider that all link costs are equal), router 1 has two equal cost paths to reach the destination 6. Condition (2) can be verified in a distributed way, i.e *s* asks its neighbors for the cost of their best path to be strictly less than its own. Neighbors cost, $C_1(v,d)$, could be obtained with distance vectors diffusion as with the Loop Free Invariant condition (*LFI*) introduced in [17]. Condition (2) can also be verified with a local computation if *s* computes shortest path trees routed at its neighbors. A NH set activated with condition (1) on a given router s is a subset the NH set activated with condition (2). However with an an uniform link valuation, it produces equivalent sets of NHs.
Condition (2) is extended with a source path deflection computation in [18]. This article presents a set of rules whose increasing flexibility allows to widen the space of valid neighbors. Condition (3) proposed in [18] does not prevent loops at the node level but does so at the link level. Indeed, a packet can transit twice by the same router but never by the same link. Authors argue that the queue is the primary resource to save, however delays can increase if paths contain several times the same router and this consumes more resources. We do not think that the queue usage is the only resource that a network administrator has to take care of. This condition needs an enhanced SPF algorithm to compute path costs of the neighborhood. With condition (3), router 1 of figure 2 can use four paths to forward its traffic to 6 : the two best ones, $1-2-4-6$ and $1-3-5-6$, and also longer ones $1-2-3-5-6$ and $1-3-2-4-6$.

The first three conditions can be used for load balancing and traffic engineering on multiples routes. Restoration is a more complex issue with multipath routing than with unipath routing. Viable next hops are potentially simultaneously used, whereas pure restoration methods use alternate routes only for rerouting. Thus, fast rerouting topological conditions do not have to take into account the possibility of simultaneously using several routes to link a unique pair of routers.

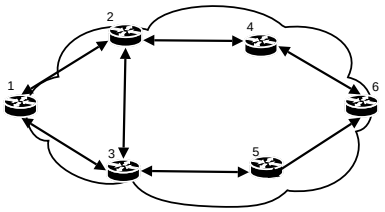

Figure 2: Simple network illustration

2) Fast rerouting techniques: Condition (4) used in [4] guarantees that a neighbor *v* does not use *s* as a possible NH in any of its shortest paths for a given destination *d*. To ensure this condition, with the Loop Free Alternate validation denoted *LFA*, *s* needs to compute $C_1(v,d)$. In figure 2, router 2 has a loop free alternate path via router 3 thanks to the path $2-3-5-6$ ($C_1(3,6) < C_1(2,6) + C_1(3,2)$).
With poorly connected topologies, the coverage achieved by *LFA* in terms of protection is low and it is even lower with the other conditions. This motivates the introduction of U-turns (denoted *UTURN* in the following) in [3]. A *UTURN* alternate to a router s (for a failure on a direct outgoing link $e = \{s, v\}$) is a router adjacent to s which does not include the link e in one of its LFA path for a given destination d. Indeed, a *UTURN* alternate is a router which does not verifies condition (4) : it uses s as a primary next hop towards d, so it has to notify s that it has a *LFA* alternate. For example, in figure 2 router 6 is called an *ECMP UTURN* Alternate of router 4 for destination 1 thanks to one of its best paths : $6-5-3-1$. Router 6 needs to have an entry in its forwarding table which takes into account the incoming link $\{4, 6\}$. Thus, it does not validate its best path $6-4-2-1$ for traffic coming from 4. The *UTURN* technique requires a cooperation between neighbors to restore the path in case of a failure. The incoming link is used to distinguish incoming traffic. This distinction effectively allows to forward traffic with a finer granularity: flows coming from different interfaces do not need to be forwarded to the same next hop. However, *UTURN* uses two modes of forwarding : the normal case where only best paths are used for routing, and the case in which a failure activates alternate paths. The condition used by *UTURN* is exclusively designed for the fast rerouting issue, whereas the first three conditions allow to share the load among several next hops. Multipath routing protocols have to verify stricter conditions to guarantee that routes can be used simultaneously without waiting for a failure detection. Indeed, when *UTURN* or *LFA* using simultaneously primary and alternate routes that may create routing loops. Another advantage of multipath routing is stability when multiple failures or congestions occur. Fast failure detection can often result in a link with a high load to be falsely classified as a failed link. As a result, if a link fails on the primary route and a congestion occurs on the only alternate route, packets could be dropped with multipath routing techniques whereas a fast rerouting technique can produce transient loops causing severe troubles.

III. The DT(p) protocol

In this section, we summarize our distributed multipath routing protocol DT(p) in its two stages of loop free paths computation and validation. The originality of our technique is the distinction made by each router on the origin of the flow (the incoming link) to forward traffic.
We distinguish two kinds of traffic:

1) local traffic : traffic coming from the router itself and/or its attached subnetworks.
2) transit traffic : traffic coming through other routers.

Transit traffic coming from different interfaces and towards the same destination does not have to be forwarded through the same set of next hops. In addition, all the sets of next hops computed for transit traffic are included in the set of next hops validated for local traffic (for a given destination).
First, all nodes compute a set of paths and their associated costs to reach all destinations of the domain. Routers then ask each others to position routing rows depending on computed costs and according to the incoming interface. A routing row on a router *s* is defined by an entry in the forwarding table which permit to route the traffic from a given incoming interface *p* and to a destination *d* through one next hop $NH_j(s,d) \in NH(p,s,d)$.
The DT section introduces our enhanced *SPF* algorithm and the DT(p) section briefly describe our loop avoiding protocol.

A. Dijkstra Transverse DT

The first stage is to construct a candidate routing table with a Dijkstra modified algorithm we call *Dijkstra-Transverse*. The *DT* algorithm considers a subset of paths with distinct first edges including at most one *transverse* edge (definitions are given in Table II). Hence, if an alternate path exists, our algorithm always computes the best one.

DT computation consists in three main steps :

a) Compute the best path tree and *simple transverse* paths.
b) Construct a *backward transverse* path set and add it to the previous set.
c) Construct a *forward transverse* path set and add it to the previous set.

The *DT* algorithm constructs a cost matrix for all destinations and for each possible next hop. This matrix is used by upstream routers to test the next hop validity in the inter-neighbor validation phase.
Complexity of the *DT* algorithm is :

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

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.
simple transverse $Pt(s,d)$	a path of m edges $\{e_1,e_2,...,e_m\}$ such that $\{e_1,e_2,...,e_{m-1}\}$ forms a best path $P_1(s,e_{m-1}.y)$ and such that e_m is a transverse edge.
backward transverse $Pbt(s,d)$	a path of m edges $\{e_1,e_2,...,e_m\}$ such that for a w i.e $0<w<m$, $\{e_1,...,e_w\}$ is simple transverse, and such that $\{e_m^{-1},e_{m-1}^{-1},...,e_{w+1}^{-1}\}$ is a best path $P_1(d,e_w.y)$.
forward transverse $Pft(s,d)$	a path of m edges $\{e_1,e_2,...,e_m\}$ such that for a w i.e $0<w<m$, $\{e_1,...,e_w\}$ is either simple transverse or backward-transverse and such that, $\{e_m,e_{m-1},...,e_{w+1}\}$ is a best path $P_1(e_w.y,d)$.

Table II: Terminology

The *DT* algorithm prunes the graph to obtain the most interesting NHs composition. Therefore, we focus only on next hops proposing shortest alternate paths. Then, DT(p) validation procedure tests individually each candidate NH (contrary to *MPDA* or *MPATH* in [17]). Our technique allows to forward differently packets depending on the incoming link, without computing best costs of upstream nodes towards all destinations.

In figure 2, if we consider router *1* as the source, edges $\{2,3\}$ and $\{3,2\}$ are *transverse* according to $branch_3(1)$ and $branch_2(1)$. $Pft(1,6) = \{\{1,3\},\{3,2\},\{2,4\},\{4,6\}\}$ is a forward transverse path which link router *1* and *6*.

B. Loop Avoidance at depth p : DT(p)

At this stage, we have to guarantee that composition of next hops computed by *DT* does not produce loops. To guarantee the loop free routing property with next hops combination on an adjacent node *v* ($p=1$), we define this condition :

$$C_j(v,d) \leq C_1(s,d), \, 1 \leq j \leq k^+(v) \tag{5}$$

This condition is directly used between adjacent nodes with *DT(1)*. It means that $NH_j(v,d)$, the j^{th} next hop computed by *v*, is a loop free NH for traffic coming from *s* : thus v activates a routing row for transit traffic coming from s and s activates a routing row via v for its local traffic towards d ($\exists i \mid NH_i(s,d)=v$, v a viable NH). Note that each upstream router has its own set of viable NHs. The set $NH(s,v,d)$ is necessarily included in the activated set for the local traffic : $NH(v,v,d)$.

In figure 2, router *3* validates with *DT(1)* a routing row with $NH_2(3,6) = 2 \in NH(3,3,6)$ for its local traffic towards destination *6*. However, router *3* cannot, with this condition, use *1* as a viable outgoing link towards destination *6*. With DT(1), routers 2 and 3 can use each other as a possible NH for destination 6, so that traffic coming from 1 to destination 6 may be forwarded via $1-2-4-6$ and $1-3-5-6$ but also via $1-2-3-5-6$ and $1-3-2-4-6$.

To improve the number of validated path, we must increase the depth of the validation process, this is DT(p). Let us define the term of *route* as opposed to the notion of *computed path* to introduce the concept of loop avoidance at depth p.

Definition 1 (Route): Formally, we denote $R_m(s,d)$ a route of *m* hops which links a source *s* and a destination *d*. We note $NH(s,s,d)$ the set of validated NHs on *S* for its local traffic. Hence, a route $R_m(s,d)$ is a composition of validated NHs (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 $r_{i+1} \in NH(r_{i-1}, r_i, d)$ and $r_1 \in NH(s,s,d)$

With this terminology we can describe our breadth first search loop detection method with *p* nodes in depth. This is a wave of messages called *query* triggered on each downstream router $v = r_1$ where DT(1) does not succeed for the upstream node s on the k^{th} NH of r_1 : $NH_k(r_1,d)$. These messages $query(s,d,c,q,P)$ contain $c = C_1(s,d)$, the best cost for s, *q* ($1 \leq q \leq p$) the number of remaining hops and *P* the set of tested routers. In the following we describe our algorithm for fixed *s* and *d*. The aim is to determine if a NH is valid, even if it does not satisfy condition (5). With *p>1*, DT(p) cannot benefit from the granularity of the incoming interface. If $p > 1$, condition (5) has to be verified for all NHs computed by DT. However, a router has only to take care of loops coming back to itself. A router r_θ ($0 < \theta < p$) can appear twice or more in the validation phase. The wave triggered on a router r_1 which does not belong to *NH(s,s,d)* with *p=1* (or if a router $r_2 = NH_k(r_1,d)$ does not belong to $NH(r_1,s,d)$), must explore, in a radius of *p-1*, all NH compositions to test the set paths generated by DT. If r_1, a neighbor of *s*, does not verify condition (5) on $NH_k(r_1,d) = r_2$, it forwards the validation message $query(s,d,c,q-1,r_1)$ to r_2 and waits for a reply. When a node r_{i+1} receives a $query(s,d,c,q,P)$ from r_i, the pseudo code of the DT(p) algorithm can take this form :

- ▷ if $NH_j(r_{i+1},d)$ satisfies condition (5), r_{i+1} stores a VALID result for $NH_j(r_{i+1},d)$
- ▷ else if $NH_j(r_{i+1},d) = r_\theta$ with $r_\theta \in P = \{r_1,...,r_i\}$, r_{i+1} stores a SKIP result for $NH_j(r_{i+1},d)$
- ▷ else if $NH_j(r_{i+1},d) = s$, r_{i+1} replies with a LOOP result to r_i
- ▷ else if *q > 0*, r_{i+1} sends a $query(s,d,c,q-1,P)$ with $P \leftarrow P \cup r_{i+1}$ to its candidate $NH_j(r_{i+1},d)$
- ▷ else the max depth *p* has been reached without success and a LOOP result is returned to r_i

The response computed on a router $r_{i+1}, 1<i+1\leq p$, a potential NH $\in NH(r_{i+1},r_i,d)$, contains a result among :

(a) LOOP : if a loop comes back to *s*.
(b) VALID : if the NH verifies condition (5) towards *d*.
(c) SKIP : if the router creates loops not coming back to *s*.

When r_i, with $i > 1$, has a result/reply for all its candidate NHs it computes its own result which is the max of all responses (the order of replies/results verifies $LOOP > VALID > SKIP$) and sends it to r_{i-1}. If r_1 receives a VALID results thanks to a router r_2, it can validate this NH and transmit a VALID result to s if $r_2 \in NH(r_1,r_1,d)$. By induction on the route $R_m(s,d)$, we obtain (each router r_i is also considered as a source) :

$NH(r_i, r_{i+1}, d) \neq \emptyset$ only if r_i verifies condition (5) on a NH of r_{i+1}, or if r_{i+1} receives a VALID result coming from a router r_{i+2} for a given couple (r_i, d).
Formally, with the DT(p) procedure, a router r_i in a route $R_m(s, d)$ guarantees, for every destination *d*, that:

(a) No NH composition in a radius of p comes back to r_i (no LOOP result).
(b) If $\{k | NH_k(r_{i+1}, d) = r_{i+2} \wedge C_k(r_{i+1}, d) > C_1(r_i, d)\}$, routers $r_{i+q}, 2 \leq q \leq p$ guarantee in a maximum radius of $p - q$, for each possible NHs composition with DT (except SKIP NH), a cost less than $C_1(r_i, d)$.
(c) $NH(r_i, r_{i+1}, d) \subset NH(r_{i+1}, r_{i+1}, d)$.

Item (a) simply prevents loops on router r_i. Item (b) uses condition (5) : all NHs computed with DT must correspond to a path whose cost verifies this condition if DT(1) does not directly succeed. If a NH computed on a router $r_{i+k}, k \leq p$ forwards back to a router $r_{i+j}, 1 \leq j < k$ this generates a SKIP result : that is its own job to avoid loops on itself. Item (c) ensures that the downstream router, r_{i+1}, does the same for its local traffic, and activates for upstream routers only the routing rows it activated for itself. These conditions ensure that the constructed routes are loop free in a stable state. For more details about scalability and routers synchronization to achieve a stable state, refer to [11] (a formal proof of the loop free routing property with DT(p) is also given).
After some experimentations it appears that it is not useful to try do validate as many next hops as possible. The more DT stores candidate NHs, the more difficult it is for DT(p), with $p > 1$, to verify the property (a). We choose as an improvement to try to validate only all best cost next hops and the best valid sub optimal next hop if any (DT has only to store these NHs). This insures that in most cases there will be at least one alternate path. The set of routes generated by this improvement is not a subset of routes validated without modifying DT.
In the example given in figure 2, router *4* is able with *DT(3)* to validate a routing row on *2* towards *6* because the best cost computed on *5* is equal to its own (VALID result). Other candidate NHs on *2* and *3* returns SKIP results. If the DT computation on *2* stores the alternate NH via *1*, $NH_3(2, 6)$, it is not possible to validate the route $R_4(4, 6) = \{2, 3, 5, 6\}$.

IV. Performance criteria and Simulation results

A. Evaluation networks

We ran simulations on four networks that we have obtained by multicast traces (see Table III [1]) with ns2. 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 [15]). Renater, Opentransit, Global Crossing and Alternet are such networks (their maps are given in [1]). We consider each link as symmetric in valuation and existence. The valuation of each link is considered equal. Therefore, the metric used to determine the cost of a route is the number of hops, even though DT(p) is able to work with a classical link state metric. To ensure concision, the following figures concern only two topologies whose characteristics are representative of the others. Nevertheless, we gather important results in table IV for an exhaustive study.
We also use a valuated topology, the GEANT network, to complete our analysis. We use an additive metric and a link valuation *w* given in [16].

Network name	# of nodes	# of edges	Diameter
Alternet	83	334	8
Global Crossing	112	340	11
Open Transit	76	206	11
Renater	79	198	9
GEANT	23	74	6

Table III: Evaluation networks

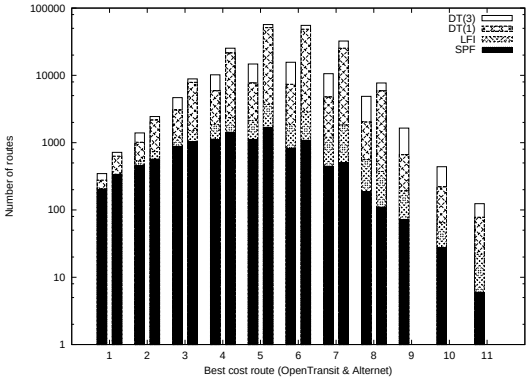

Figure 3: Route distribution according to routing protocol

In figure 3 (the scale is logarithmic), the number of loop free routes is gathered depending on their length. Route distribution is analyzed with three protocols. *SPF*, only considers strictly best routes. *LFI* only validates equal cost routes because valuation is uniform. We gathered all validated alternate routes by DT(p), with $p \in \{1, 3\}$, depending on the best one which links two routers. Left bars concern OpenTransit whereas right ones describe Alternet route distribution. Figure 3 can be used as a basis for the interpretation of figures 4 and 5 : a bar, corresponding to a route length *l*, in figure 3 is equal to the sum of all points of the curve $min = l$ in figures 4 and 5. We observe that the number of alternate routes validated with DT(p) is very high compared to the number of routes validated with *LFI* condition.

B. Number of protection routes

In this section, we present the number of routes validated by DT(p) according to route length l. Because we choose all link costs to be equal in our simulations, presented results verify this relation :

$$l \leq l_{min} \times (p + 1) \quad (6)$$

where l_{min} denotes the minimal distance between two routers. The route length is gathered depending on this parameter.

[1]The edge number is the number of simplex links

Thus, all pairs $[source, destination]$ sharing this value are analyzed on the same curve in the three following figures. The first point of the curve with $l_{min} = 1$ indicates the number of links in the network and the first point of the last curve corresponds to the network diameter. In this section, we do not focus on the *LFA* technique (with or without *UTURN* extension) given by condition (3), because the routes cannot be used simultaneously but only in case of link failures. If we consider alternate routes as viable even if there is no failure, this implies many loops with this kind of rerouting protocol.

OpenTransit is a low connected network : 2.7 simplex links

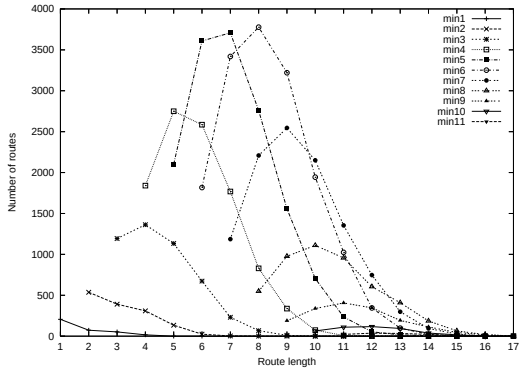

Figure 4: OpenTransit route distribution with *DT(3)*

per node in average. The number of routes validated by our technique is therefore less important than for Alternet, mainly because the length of cycles in the network are often superior to the maximal depth of search. On both networks, we notice that the length of validated routes rarely reached its upper limit (6). This observation is reassuring, especially with a great depth of validation, insofar as the validated bypass routes are not so much longer than the best one. As we notice in

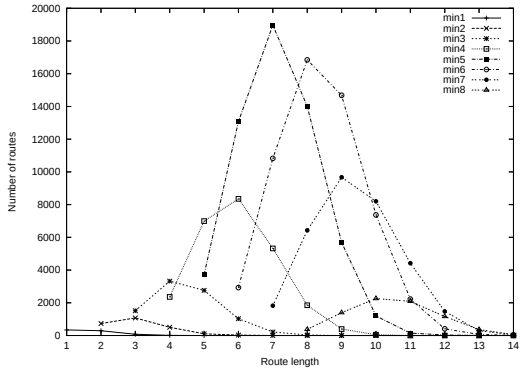

Figure 5: Alternet route distribution with *DT(3)*

the following section, the number of validated routes directly influence the protection ratio in terms of potentially bypassed links. The capabilities in term of rerouting is proportionnal to the number of alternate routes. When a local alternate route exists, a router saves the failure notification period and the path computation delay. Therefore, the reaction time depends only on failure detection and FIB updating times. We also present results obtained with the GEANT topology in figure 6. This representation contains results for SPF, LFI and DT(3). Routes are classified according to the number of hops although the valuation is not uniform. We observe that the distribution does not seem to be as regular as for a hop count metric. We do not present *DT(1)* results for lisibility and because they are very close to LFI results. The link valuation changes the topology characteristics. Thus, LFI can validate a larger set of paths than with ECMP or even with *DT(1)*. The number of validated routes is an important factor to save the notification time as we illustrate in the next section. And obviously, if the alternate route is disjoint from the primary route, the coverage capabilitie is full.

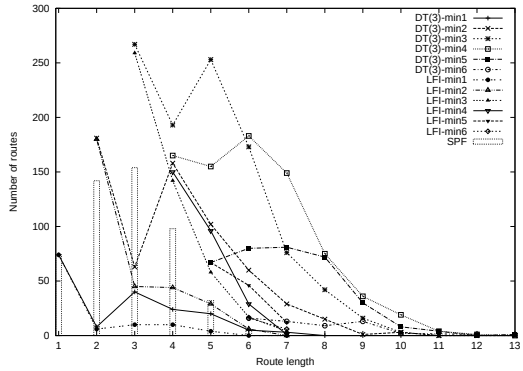

Figure 6: GEANT route distribution

C. Protected links and coverage

In this section we compare the protection capabilities of different alternate path selection techniques, in the case of a link failure. In the first two figures we calculate the average link protection ratio on each pair of nodes in the network, according to the minimal route length. Thus, if only *2* links on a best route of length *4* can be protected thanks to its set of protection path, the resulting ratio is $1/2$. We compare two different modes of protection. The first one that we call *local protection* is computed in the following way : a link $\{a, b\}$ is considered as protected only if there exists a protection path crossing the same node a, but which has an alternate next hop $c \neq b$ on this common node.

The second protection mode, that we called *upstream protection*, considers that a link is protected if there exists a protection path which does not contain this link. Some links cannot be protected at all since their failures partition the network (isthmus link). To ensure a correct analysis, we only consider links that can be protected in the following results. Hence, evaluated techniques do not take isthmus links into account. Alternet contains approximatively 10% of links that can not be protected whereas this proportion reaches 33% for OpenTransit. As illustrated in figure 7, *UTURN* is slightly more effective than *DT(3)* in terms of coverage. The results

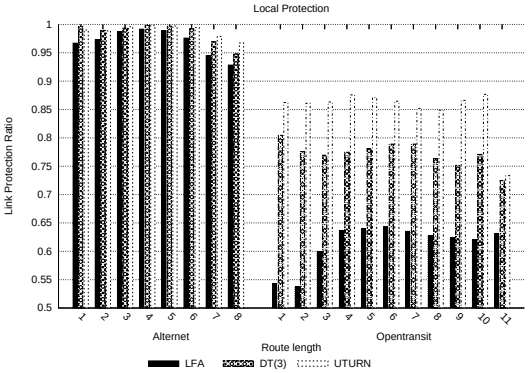

Figure 7: Local protection results

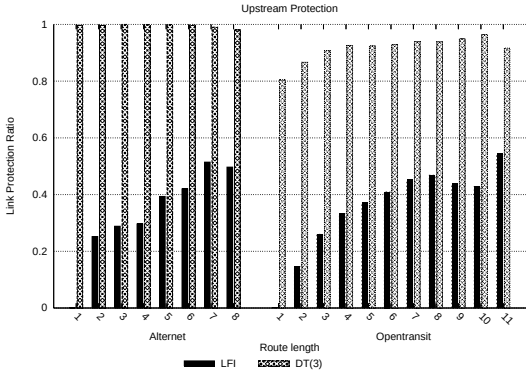

Figure 8: Upstream protection results

of our multipath routing technique almost reach the same protection ratio as *UTURN* does on the five networks (see Table IV). However, *DT(3)* has better results than *LFA* (except on GEANT). The major issue is that condition (5) has to be guaranteed on each computed NH in the DT(p) phase. With unipath rerouting protocols such as *UTURN* the validation process is required only on the best NH. To sum up, DT(p) performs almost as well as the best unipath rerouting technique. But, in addition, DT(p) is able to use alternate paths for load balancing. Routes validated by DT(p) are designed both for fast rerouting and load balancing. Moreover, if we consider the possibility to make upstream neighbors to cooperate when a failure occurs, then we can introduce the notion of upstream protection (see figure 8). Upstream link protection implies that there is a backward wave notifying the failure (possibly up to the source node). The traffic redirection could be done on upstream routers which have an alternate route which does not include the failed link. Thus, upstream protection results are obviously better than local protection results, but the reaction time can become high. Indeed, the time necessary to inform the concerned rerouting node could be large according to the topology. This notification backward wave protocol is not given here, this is not in the scope of this document. However, we discuss about possible implementations in the last section.

Network name	Direct				Upstream	
	LFI	LFA	DT(3)	UTURN	LFI	DT(3)
Alternet	17.8	98	99.2	99.4	34.2	99.8
Open Transit	15.9	62.1	77.8	86.6	33.3	91.7
Global Crossing	19.5	73.5	87.1	91.1	43.7	96.1
Renater	10	51.5	67.3	69.2	21	85
GEANT	36.6	83.7	74.6	87	63.4	94

Table IV: Coverage

LFI cannot protect routes of length 1. Indeed, if link capacities are uniform, condition (2) cannot validate a protection route for adjacent routers. Figure 10 shows the coverage capabilitie in a different way. We compute for each link the number of times, y, that a primary route use it (see figure 9). We notice on both networks that the link utilization is not homogeneous. That means that networks can be coarsely partitioned in core links and border links. Then, we compare y to the number of times, x, this link is protected by an alternate route of a given technique. The protection ratios $\frac{y}{x}$ are sorted by the number of pairs of nodes using this link within their best route. Links are gathered by groups of *7* for a better readability.

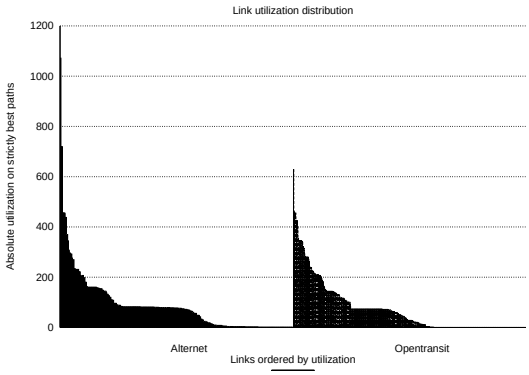

Figure 9: Links utilization distribution

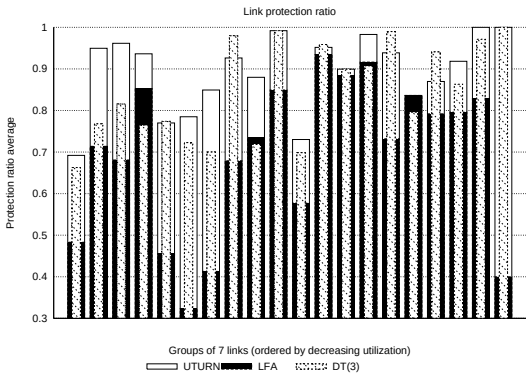

Figure 10: Link by link protection on Opentransit

The link protection results show us that the protection does

not depend on the link situation (in the core or border of the network). Moreover, we observe that *DT(3)* does not present results systematically superior to *LFA* and inferior to *UTURN*. The protection results depend on topology characteristics : the main problem for the DT(p) validation is that the number of NHs computed with DT limits the success of NHs validation even though a recovery path exists. In figure 2, if we want to protect route $5-3$, 5 cannot find a protection route because router 2 (on 2, $p=3$) has two paths and the one via 1 has a cost bigger than $C_1(5,3)=1$. Although there exists a recovery route $5-6-4-2-3$, router 5 cannot select router 6 as a viable next hop to reach 3 because router 2 cannot guarantee item (b) in section III-B. However, links which are in a simple cycle of length 5 can be protected with *DT(3)*, whereas *UTURN* cannot. The cycle lengths in the network and their overlap is a major issue for the validation procedure and the depth adjustement of DT(p). Indeed, in a simple ring with uniform valuation, the coverage achieved with *DT(3)* is better than with *UTURN*. DT(p) is able to give good protection results with, in addition, the possibility to use alternate routes for load balancing.

D. Discussion

Our protocol allows us to benefit from a temporary efficient solution for fast rerouting while the failure is broadcasted with *LSA* and optimal routes are recomputed. However our method efficiency depends on topology. Therefore, in some particular cases, there is no local alternative to directly reroute traffic. An intuitive solution is to alert upstream neighbors which are concerned by the failure. Let *s* be a router which detects a failure on a link $\{s,v\}$, where *v* is the primary next hop, concerning a subset of destinations $D=\{d_1,d_2,...,d_i,...,d_n\}$ that it cannot reroute locally. Either *s* cannot shift traffic on an alternate NH, or the alternate link does not support the induced load. Then *s* must inform its upstream routers set with concerned destination that it cannot reroute their traffic. Let consider a single upstream router p related to the given destination d_i. This router is adjacent to s and is present in its forwarding table : $NH(p,s,d_i)\neq\emptyset$, $\{p,s\}$ is an incoming link towards d_i. Then s ask p to stop forwarding traffic to itself towards d_i. When $s\in NH(p',p,d_i)$ where p' is an upstream router of p for d_i, either p will locally reroute the traffic via an alternative NH ($\mid NH(p',p,d_i)\mid\geq 2$), or if it cannot, it will forward the failure notification to each of its own upstream routers p' (and recursively potentially up to traffic sources). This sketch of notification protocol should perform an average protection rate superior to 90% (last column in Table IV). However, it has to ensure in practice that the notification time is lower than the best route reconfiguration. Indeed, it is not useful to transmit the failure notification to a remote upstream router if the time needed to transmit this notification is close to the convergence time of a classical routing protocol (such as OSPF or ISIS) when the topology changes. The failure notification can stop when all notified routers can use a route which does not contain the failed link. With this upstream rerouting technique, we save the time of the *SPF* re-computation (see [8]) and *FIB* updating, so the recovery time only depends on the failure detection period and on the failure notification delay. This period mostly depends on the distance between the failure and the rerouting point.

V. Conclusion

In this paper, we have presented a multipath routing approach which protects from link failures almost as well as with unipath rerouting techniques. Results given in this paper guarantee a purely local coverage close to that generated by best unipath fast rerouting methods such as *UTURN*. Although multipath routing seems to fit well with protection and restoration issues, we have insisted on the difficulty to ensure a protection scheme when multiple routes can also be used for load balancing. Our proposition allows to use several routes for proportional routing, whereas protection issues are only a subcase when proportions are integers in $\{0,1\}$. We have also discussed ways to achieve better global coverage results. A simple notification protocol which informs close neighbors about failures can still add a considerable degree of protection to raise a nearly full coverage.

References

[1] "Multicast map, http://clarinet.u-strasbg.fr/ merindol/maps.tar.gz."

[2] M. Alichery and R. Bhatia, "Pre-provisioning networks to support fast restoration with minimum over-build," in *IEEE INFOCOM*, march 2004.

[3] A. Atlas, "U-turn alternates for ip/ldp fast-reroute draft-atlas-ip-local-protect-uturn-03," Internet Engineering Task Force," Internet Draft, Feb. 2006.

[4] A. Atlas and A. Zinin, "Basic specification for ip fast-reroute: Loop-free alternates draft-ietf-rtgwg-ipfrr-spec-base-06," Internet Engineering Task Force," Internet Draft, mar 2007.

[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] O. Bonaventure, C. Filsfils, and P. Francois, "Achieving sub-50 milliseconds recovery upon bgp peering link failures," in *CoNEXT'05*. New York, NY, USA: ACM Press, 2005, pp. 31–42.

[7] D. Eppstein, "Finding the k shortest paths," in *IEEE Symposium on Foundations of Computer Science*, 1994, pp. 154–165.

[8] G. Iannaccone, C.-N. Chuah, S. Bhattacharrya, and C. Diot, "Feasibility of ip restoration in a tier-1 backbone," *IEEE Networks Magazine*, March 2004.

[9] 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.

[10] J. McQuillan, I. Richer, and E. Rosen, "ARPANET Routing Algorithm Improvements," BBN 3803, Apr. 1978.

[11] P. Merindol, J. Pansiot, and S. Cateloin, "Path computation for incoming interface multipath routing," in *ECUMN'07*, Toulouse, France, 2007.

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

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

[14] D. Oran, "Osi is-is intra-domain routing protocol," Internet Engineering Task Force, Request For Comments 1142, 2001.

[15] J. J. Pansiot, "Local and dynamic analysis of internet multicast router topology," *Annals of telecommunications*, vol. 62, no. 3-4, 2007.

[16] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre, "Providing public intradomain traffic matrices to the research community," vol. 36, no. 1, January 2006.

[17] 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.

[18] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," in *SIGCOMM vol.36*, october 2006, pp. 159–170.