Enhanced OSPF Graceful Restart

Carole Hounkonnou, Eric Fabre

Enhanced OSPF Graceful Restart

Carole Hounkonnou, Eric Fabre
INRIA Rennes - Bretagne Atlantique

***Abstract*—Modern OSPF (Open Shortest Path First) routers can preserve their packet forwarding activity while they reboot. This enables maintenance operations in the control plane with minimum impact on the data plane, such as the Graceful Restart (GR) procedure. This of course assumes the stability of the network topology, since a rebooting router is unable to adapt its forwarding table and may cause routing loops. The Graceful Restart standard thus recommends to revert to a normal OSPF restart as soon as a topological change is advertised. This paper proposes to be less conservative and to take full advantage of the separation between the control and forwarding functions. This is achieved by new specific functionalities: (a) the prediction of routing loops caused by a restarting router, (b) the determination of the minimal number of temporary backup forwarding actions that should be applied to prevent these loops, without reverting back to a normal OSPF restart, and (c) the design of action plans to set and remove these temporary backups in order to avoid micro-loops when the restarting router goes back to a normal functioning. This results in minimal traffic perturbations when topology changes during a maintenance operation. [1] provides a longer version of this paper, including proofs.**

***Index Terms*—OSPF, Graceful Restart, Network Maintenance.**

I. Introduction

OSPF (Open Shortest Path First) [2], [3] is a widely used link state routing protocol in the Internet. Modern router architectures separate the data plane, and thus the forwarding function, from the control plane, that runs the routing protocols such as OSPF. This creates a possibility to keep forwarding packets while the control plane is being restarted. This so-called Graceful Restart procedure has been standardized [4] and is available in commercial routers [5], [6]. Graceful Restart requires the cooperation of all routers neighboring the restarting one. Their role is to keep up the adjacency with the restarting router as long as the topology remains static. In case of any change in the topology, one must immediately stop the graceful restart and return to the standard OSPF behavior, which thus fully removes the restarting router from the topology. This intends to avoid the possible creation of routing loops resulting in packet losses and unreachable destinations.

Such an abrupt change of behavior can be temporarily harmful to the network. And, strictly speaking, it may not be necessary: not every topological change will result into a routing loop, so the forwarding activity of the restarting router could be maintained. Furthermore, even if routing loops are created, they can be temporarily fixed. The present paper studies the possibility of such smoother changes of behavior.

Since the standardization of the graceful restart procedure, few papers have examined its practical consequences. [7] examined how a general reboot of all routers could be organized, taking into account that a helper node cannot reboot until the node it is helping has completed its own reboot. To the best knowledge of the authors, however, the issue of preventing routing loops during the graceful restart of OSPF routers has only been tackled by Shaikh et al. in [8] and more recently in [9]. These contributions detail necessary conditions to the existence of routing loops, in the case of several restarting routers, and propose to remove the restarting routers from the forwarding path as soon as these conditions are detected. The present paper follows a similar approach for the detection, but relies on a necessary and sufficient condition for the existence of routing loops, in the case of a single restarting router. The developments then go further by proposing minimal temporary corrections to such loops, and by correcting simultaneously multiple problematic destinations. In our approach, when a routing loop is detected, only a few nodes are informed and apply a correction, rather than broadcasting a global warning to all nodes and returning to a standard OSPF behavior. As a result, the restarting router is maintained in the topology for all destinations to which it is not dangerous.

The paper is organized as follows. Section II illustrates the normal and graceful restarts of OSPF and explains how routing loops can occur during a graceful restart. Section III introduces the notions of source and destination graphs. These graphs are central for the detection of routing loops (Section IV). Section V characterizes the severity of such routing loops, using coloring properties of destination graphs. It then explains in detail how to correct such loops by temporary reroutings, in the case of a single problematic destination. Section VI extends the problem to several problematic destinations to correct simultaneously. Finally, Section VII evaluates, on a typical network topology, the proposed enhanced OSPF GR.

II. Normal and Graceful Restarts in OSPF

OSPF runs on a simple abstract vision of the network: a weighted and directed graph (Fig. 1), that we call the *topological graph*. At the core of the OSPF routing protocol

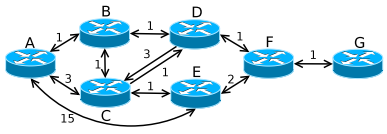


Fig. 1. An example of OSPF network

is a distributed, replicated link state database that describes the collection of routers in the domain, how they are interconnected, and the quality of each link. So each node knows

the full topological graph at any time. Given the link state database, and assuming this is a reliable description of the network state, each node/router runs Dijkstra's algorithm to derive the shortest paths to all other nodes. The shortest paths originating from (and calculated by) some router *R* organize as a shortest-paths tree (SPT) rooted at *R* that we call the *source graph* for router *R*. Fig. 2 displays this source graph for node C in the network of Fig. 1.

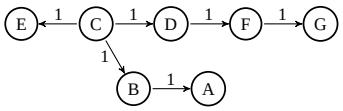
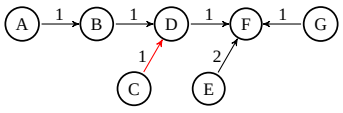


Fig. 2. Shortest paths from C to all destinations (the source graph of C).

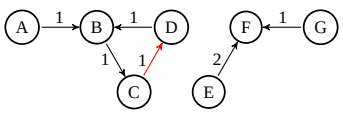
During a normal router restart, the neighbor routers break adjacency with the restarting one, i.e. they generate new LSAs that are flooded throughout the network and cause all routers to update their forwarding tables in order to avoid the rebooting node. A few minutes later, once the restart is completed, the neighbor routers re-establish adjacency with the rebooted one and the whole sequence of LSA floodings and forwarding tables updates is repeated.

With a graceful restart, a router, whose control plane is about to restart and whose forwarding plane functions normally, sends a grace LSA to its neighbors, declaring its intention to perform a graceful restart within a specified grace period. The neighbor nodes (known as helpers) continue to list the restarting router as fully adjacent in their LSAs during the grace period, but only if the network topology remains static. Once the control plane restarts, the restarting router goes through a normal adjacency establishment procedure with all the helpers, at the end of which the restarting router and the helpers regenerate their LSAs.

Any change in the network topology during the grace period would cause the helpers to abort the graceful restart and generate their LSAs showing the breakdown of adjacency with the restarting router. Indeed, the latter is unable to adjust its forwarding table in a timely manner when the network topology changes. Its forwarding table is said to be *frozen*. Since this table may no longer be consistent with the new network topology, routing loops can occur. Fig. 3 illustrates this routing loop creation for the network of Fig. 1, assuming link $D \to F$ fails while node C is restarting.



(a) before failure of $D \to F$



(b) after failure of $D \to F$

Fig. 3. Destination graphs to F before and after failure of link $D \to F$.

To prevent such routing loops, [4] takes a conservative approach and recommends to revert to a normal OSPF restart when a change in the network topology occurs. However, not every topological change will result into a routing loop even if the restarting router is unable to adjust its forwarding table. This observation underlines the need for a solution able to detect beforehand the creation of loops while a graceful restart is in progress, and possibly to temporarily fix them, in order to avoid the burden of several complete OSPF reconvergences and possible perturbations in the load balancing.

III. Properties of Routing Graphs

Definition 1. The *topological graph* of an OSPF network is a weighted directed graph $G = (V, E, w)$ where the finite set V denotes vertices (or 'nodes' or 'routers'), $E \subseteq V \times V \setminus \{(v, v), v \in V\}$ denotes the arcs (or 'links'), and $w : E \to \mathbb{R}^+$ is the weight (or cost) function on links. It is assumed that any node is reachable from any other in G (see below).

Definition 2. A *path* from u to v in $G = (V, E, w)$ is a sequence of vertices $p = (v_0, v_1, ..., v_n)$ of V such that $v_0 = u$, $v_n = v$, and each $(v_i, v_{i+1}) \in E$ for $0 \leq i < n$. When such a path p exists from u to v, v is said to be reachable from u through p, denoted $u \overset{p}{\rightsquigarrow} v$ (or simply $u \rightsquigarrow v$). v is called *descendant* of u and u is called *ancestor* of v. A *circuit* in G is a path such that $u = v$. The weight/cost of path p is the sum of the weights/costs of its arcs: $w(p) = \sum_{i=1}^{n} w(v_{i-1}, v_i)$. The distance between u and v is then $d(u, v) = min\{w(p) : u \overset{p}{\rightsquigarrow} v\}$, and a shortest path between u and v is a path reaching this bound.

We use two types of routing graphs in the sequel, source and destination graphs, attached to any node of $G = (V, E, w)$.

Definition 3. A *source graph* $H^\sigma = (V, E')$ is a directed graph such that every node has a unique predecessor for E', except a unique node $\sigma \in V$ (the source), which has none: $\forall v \in V \setminus \{\sigma\}$, $|\{u : (u, v) \in E'\}| = 1$ (and 0 for $v = \sigma$). H^σ is said to be *correct* iff it contains no circuit. In $G = (V, E, w)$, the source graph $G^{\sigma *} = (V, E^{\sigma *})$ of a node $\sigma \in V$ is obtained by gathering consistent shortest paths from σ to all other nodes in G: $E^{\sigma *} \subseteq E$, and $\forall v \in V \setminus \{\sigma\}$ the unique path p such that $\sigma \overset{p}{\rightsquigarrow} v$ in $G^{\sigma *}$ satisfies $w(p) = d(\sigma, v)$ in G.

Definition 4. A *destination graph* $H_\delta = (V, E')$ is a directed graph such that every node has a unique successor for E', except a unique node $\delta \in V$ (the destination), which has none: $\forall u \in V \setminus \{\delta\}$, $|\{v : (u, v) \in E'\}| = 1$ (and 0 for $u = \delta$). H_δ is said to be *correct* iff it contains no circuit. In $G = (V, E, w)$, the destination graph $G^*_\delta = (V, E^*_\delta)$ of a node $\delta \in V$ is obtained by gathering consistent shortest paths to δ from all other nodes in G: $E^*_\delta \subseteq E$, and $\forall u \in V \setminus \{\delta\}$ the unique path p such that $u \overset{p}{\rightsquigarrow} \delta$ in G^*_δ satisfies $w(p) = d(u, \delta)$ in G.

Source and destination graphs naturally appear in OSPF: by connecting the forwarding rules to destination δ in all nodes of topology G, one gets a destination graph G_δ. Ideally, each G_δ coincides with the true G^*_δ if all nodes have an accurate and

up-to-date knowledge about G. However, during a graceful restart, the G_δ in use may differ from the expected G^*_δ, due to frozen forwarding tables, and thus may contain circuits. Similarly, one could build the source graphs G^σ actually used by OSPF for topology G: for each source σ, u is the unique predecessor of v if a packet originating from σ and addressed to v reaches it through u. Ideally again, G^σ should coincide with $G^{\sigma *}$, but this may not hold during a graceful restart. Observe that source and destination graphs are dual notions: inverting the orientation of edges in a source graph yields a destination graph.

As destination graphs encode the effective forwarding rules applied by OSPF, they are instrumental in the prediction of routing loops. These simple objects have numerous properties that can help for this task.

Definition 5. In a directed graph $G = (V, E)$, the *connected to* relation on vertices, denoted by $u \sim v$, is defined as the equivalence relation on V generated by $u \rightsquigarrow v \Rightarrow u \sim v$ (this amounts to dropping the orientation of edges). A *connected component* of G is a subgraph $G_{|V'} = (V', E_{|V' \times V'})$ of G such that $V' \subseteq V$ is an equivalence class of vertices for $\sim$.

Proposition 6. *Let H_δ be a destination graph, each connected component of H_δ either contains δ or contains a unique circuit. Therefore, if H_δ contains p circuits, then it contains $p+1$ connected components.*

See [1] for proofs. As an example, the destination graph G_F in Fig. 3(b) contains two connected components. All destination graphs have a similar shape, with connected components made of a single circuit and directed trees descending towards it, plus one last tree directed toward the destination node.

Corollary 7. *Let $G^*_\delta = (V, E^*_\delta)$ be the destination graph gathering the shortest paths to δ in $G = (V, E, w)$. Let G_δ be a perturbed version of G^*_δ where k nodes have modified their successor. Then G_δ contains at most k circuits, and $k+1$ connected components.*

The perturbations above model the fact that k routers are not using the forwarding table they should follow on topology G, but rely on an outdated one. As a consequence, if the topology changes while k routers are operating a graceful restart, at most k routing loops can be created for each destination δ. And as shown in Section VI, the same loop can alter several destinations at a time. This suggests that few 'problems' should actually appear and require fixing.

IV. Prediction of Routing Loops

Let $G_0 = (V, E_0, w_0)$ denote the topology with which a restarting router r computed its last forwarding table, and let G^r be the source graph of node r in this topology. In G^r, node r has $h_1, ..., h_K$ as successors, which are also helper nodes by design of the graceful restart procedure. Let $\mathcal{D}_r(h_k) = \{v : h_k \rightsquigarrow v \text{ in } G^r\}$ denote the descendants of h_k in G^r, $1 \leq k \leq K$. As G^r is a correct source graph, the $\mathcal{D}_r(h_k) \cup \{h_k\}$ form a partition of $V \setminus \{r\}$.

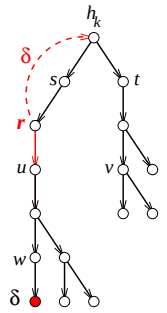


Fig. 4. The expected source graph G^{h_k} of h_k, and its misbehavior for packets addressed to δ. Instead of correctly forwarding such packets to u, node r selects a wrong neighbor and actually send them back to h_k, thus creating a circuit.

Let $G_1 = (V, E_1, w_1)$ denote the actual network topology. We assume that the links $(r, h_k) \in E_0$ are still present in E_1. Let G^{h_k} be the source graph of node h_k in this new topology, for $1 \leq k \leq K$ (Fig. 4). Let $\mathcal{D}_{h_k}(r) = \{v : r \rightsquigarrow v \text{ in } G^{h_k}\}$ denote the descendants of r in G^{h_k} ($\mathcal{D}_{h_k}(r)$ contains u and all nodes below u in Fig. 4).

Finally, for a node $\delta \in V$, let G_δ be the actual destination graph to δ when all nodes use topology G_1 except r that uses topology G_0.

Proposition 8. *There exists a (unique) routing loop in the destination graph G_δ iff there exists a (unique) k such that $\delta \in \mathcal{D}_r(h_k) \cap \mathcal{D}_{h_k}(r)$.*

This defines a simple practical test for discovering destinations δ at risk, i.e. unreachable due to the presence of a routing loop. All neighbors of r are advertised that r initiates a graceful restart, and they act as helpers, so they can store the frozen G^r used by r all along the grace period. Each successor h_k of r can then determine and announce the contents of $\mathcal{D}_r(h_k) \cap \mathcal{D}_{h_k}(r)$ if the topology changes.

V. Correction of a Routing Loop

Let $G_\delta = (V, E_\delta)$ be a destination graph over topology $G = (V, E, w)$, so $E_\delta \subseteq E$. If G_δ is correct (i.e. contains no circuit), it can reliably be used to forward packets to δ, but it may not use the shortest paths of G. Assume that k routers $r_1, ..., r_k \in V$ are performing a graceful restart in G. As seen above, the actual G_δ used for forwarding packets to δ differs from the optimal G^*_δ by at most k arcs: the arcs originating from routers $r_1, ..., r_k$ (assuming they differ from δ) can point to any node in V. Therefore G_δ contains at most k circuits, that each contain at least one node of $\{r_1, ..., r_k\}$. Given that these nodes cannot change their forwarding rule, is it possible to modify the routing choices of other nodes to turn G_δ into a correct destination graph? What is the minimal number of nodes that should be rerouted, and where are they?

A. Severity Degree of Routing Loops

Definition 9. Let $G_\delta = (V, E_\delta)$ and $G'_\delta = (V, E'_\delta)$ be two destination graphs in G such that r has the same successor in G_δ and in G'_δ. Let us denote by $C(G_\delta, G'_\delta) = |E_\delta \setminus E'_\delta| = |E'_\delta \setminus E_\delta|$ the number of arcs that distinguish them. The *color*

of node v in G_δ is defined as $C_\delta(v) = min\{C(G_\delta, G'_\delta) : v \rightsquigarrow \delta \text{ in } G'_\delta\}$.

So $C_\delta(v)$ is the minimal number of reroutings that should take place in G_δ in order to correctly forward packets from v to destination δ. Note that $C_\delta(v)$ can be infinite if no correction is possible, and $C_\delta(v) = 0$ iff v is in the connected component of G_δ that contains δ.

Proposition 10. *Let G_δ be a destination graph in topology G. If $u \rightsquigarrow v$ in G_δ, then $C_\delta(u) \leq C_\delta(v)$. And if the arc (u, v) exists in G, then $C_\delta(u) \leq C_\delta(v) + 1$.*

As a consequence, the color of nodes in each connected component of G_δ augments as one progresses towards the circuit, and it is constant on this circuit. There cannot be gaps in series of colors: vertices of color n exist only if there exist vertices of color $n - 1$.

Corollary 11. *G_δ contains a circuit which color is infinite iff this routing loop cannot be corrected. The color of a circuit in G_δ is the number of reroutings that is necessary to redirect to δ all nodes of the connected component containing this circuit. If G_δ contains a unique circuit, its color is the minimal (and sufficient) number of reroutings to transform G_δ into a correct destination graph.*

Fig. 5 illustrates the vertex coloring on the destination graph G_F, for our running example. Vertices E, F, G are located in the same connected component as the destination F, therefore their color is 0 (displayed in green). One has $C_F(A) = 1$ (yellow), because edge (A, E) exists in topology G, and $C_F(E) = 0$. A can easily reach F by rerouting packets through E instead of B in G_F. Finally, vertices B, C, D in the circuit all have color 2 (red). C is the frozen restarting router, so it cannot be rerouted, and neither B nor D could be directly rerouted to E, F or G (recall that link (D, F) failed). However, B can be rerouted to A, and the latter to E. These two modifications are sufficient to guarantee that all packets addressed to F actually reach it.

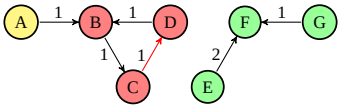


Fig. 5. Vertex colors on the destination graph G_F: green=0, yellow=1, red=2.

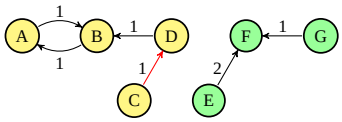
Proposition 10 reveals a simple *coloring algorithm* over G_δ. Nodes of color 0 are easily obtained by back-tracking from δ. For any remaining (uncolored) node u, if arc (u, v) exists in G and v has color 0, then u takes color 1. And one can recover all nodes of color 1 on G_δ by backtracking from such u nodes. Similarly, nodes of color 2 are the uncolored predecessors u in G of a node v of color 1, or the uncolored ancestors in G_δ of such u nodes. And so on, until no more coloring rule is applicable. The remaining uncolored nodes take ∞ as color. This algorithm has a linear complexity, similar to Dijkstra's algorithm, and it can also be distributed. It allows one to decide if routing loops can be corrected.

B. Correction of a Routing Loop

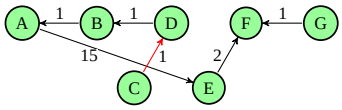
Due to space limitations, the remainder of the paper focuses on the case of a single restarting router in G. Therefore, if destination graph G_δ is incorrect, there is a single routing loop to repair.

Corollary 12. *Let the incorrect destination graph G_δ contain a unique circuit p of color n. At least one node of this circuit (different from the frozen node r) can be rerouted to a node of color $n - 1$. Performing this rerouting yields the destination graph G'_δ that again contains a unique circuit p', of color $n - 1$.*

This result derives simply again from Proposition 10. Its interest is to reveal a simple procedure to determine the n reroutings that can turn G_δ into a correct destination graph. Fig. 6 illustrates these two steps for the G_F in Fig. 5: B is first rerouted from C to A, then A is rerouted from B to E.



(a) New vertex colors once B is rerouted from C to A.



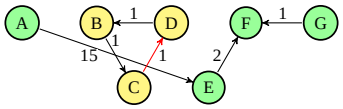
(b) New vertex colors once A is also rerouted from B to E.

Fig. 6. Successive reroutings to correct destination graph G_F.

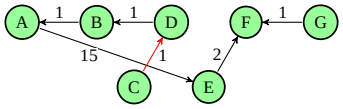
In summary, when a topological change occurring during a (single) graceful restart creates a routing loop for some destination δ, a simple procedure can determine the minimal number n of reroutings that could correct it, and the location of these reroutings. There generally exist several such temporary 'patches' of n reroutings, and one could wonder which one is the most efficient in terms of average cost, if link weights are taken into account. We conjecture that this problem is NP hard. One may wonder about situations where the color of the circuit is infinite. In that case, there is no solution to reroute messages to δ around r. Therefore a standard restart of OSPF would also be useless to resolve the problem.

C. Scheduling of Backup Routings

Assume one has determined a sequence $s_1, ..., s_n$ of vertices that should be rerouted to correct a destination graph G_δ, where the index i in s_i represents the color $C_\delta(s_i)$. In which ordering should these temporary reroutings be performed? One possibility is illustrated in Fig. 6, where $s_2 = B$ is rerouted before $s_1 = A$ in G_F. The reverse order is illustrated in Fig. 7. As one can notice, this second option offers a better transient mode: nodes are progressively rerouted correctly to $\delta = F$, whereas in the previous option all nodes suffer from the loop until the last rerouting is performed.



(a) New vertex colors in G_F once A is rerouted from B to E.



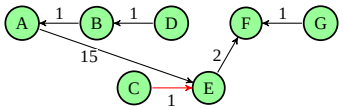
(b) New vertex colors in G_F once B is then rerouted from C to A.

Fig. 7. Successive reroutings to correct destination graph G_F.

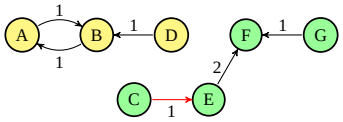
Proposition 13. *Let $s_1, ..., s_n$ be a minimal sequence of vertices that should be rerouted to correct G_δ, where $C_\delta(s_i) = i$ in G_δ, and $n = C_\delta(r)$. Rerouting only s_i to its appropriate new successor yields G'_δ where the new node colors satisfy $C'_\delta(s) = C_\delta(s)$ if $C_\delta(s) < i$, and $C'_\delta(s) = C_\delta(s) - 1$ if $C_\delta(s) \geq i$.*

A consequence of this result is that one should start rerouting nodes in the order $s_1, ..., s_n$, in order to maximize the color decrease in G_δ, i.e. to maximize at each step the number of nodes that can correctly reach δ.

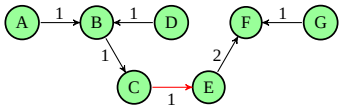
Assume now that the restarting router r has finished its graceful restart. Can it safely switch to its new forwarding table (corresponding to the actual topology G_1)? And how should one remove the temporary rerouting patches? Fig. 8 illustrates the return in function of $r = C$, now correctly connected to E, and a removal of the rerouting patches following order $s_1 = A, s_2 = B$. As one can notice, this may recreate forwarding loops, whereas the converse ordering is safe. A similar phenemenon was already observed in standard OSPF convergence, and led to the developement of ordered updates of forwarding tables, known as OFIB [10]–[12].



(a) C recovers its normal routing to F through E in G'_δ.



(b) Vertex colors once A recovers its normal route through B in G'_F.



(c) Vertex colors once B then recovers its normal route through C in G'_F.

Fig. 8. Successive removals of the temporary reroutings into the corrected destination graph G'_F, after node C returns to function.

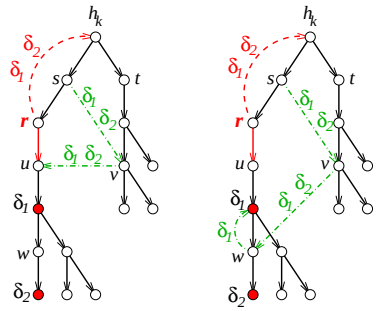


Fig. 9. Rerouting packets addressed to δ_1 with a minimal number of hops in order to go around r (and thus avoid the circuit). This rerouting path can also be (partly) used to correct the circuit on the path to δ_2 when $\delta_1 \rightsquigarrow \delta_2$ in G^{H_k}.

Proposition 14. *Let $s_1, ..., s_n$ be a minimal sequence of vertices that have been rerouted to correct the loop created by r in G_δ, where $C_\delta(s_i) = i$ in G_δ, and $n = C_\delta(r)$. Once r returns to function, it can safely switch to its expected forwarding table without recreating a loop. And removing the temporary reroutings starting from s_n to finish by s_1 guarantees that no transient routing loop appears.*

VI. Correction of Multiple Routing Loops

Assuming a single router r has a frozen forwarding table while the network topology evolves, we have shown how to detect a routing loop for some destination and how to correct it with minimal effort. This leaves open the burden of fixing *all* problematic destinations, which we address now. The idea is that fixing a problematic destination may help resolving others. Consider again the setting of Section IV, where all nodes established their forwarding table according to topology G_1 excepted node r, which used topology G_0. We rely on the criterion of Proposition 8.

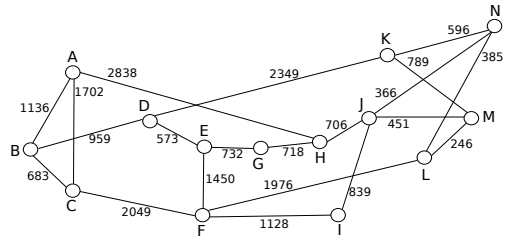
Proposition 15. *Let δ_1, δ_2 be two destinations in $\mathcal{D}_r(h_k) \cap \mathcal{D}_{h_k}(r)$ where h_k is one of the successors of r in its source graph G^r. Consider the source graph G^{h_k} of node h_k (in topology G_1). If $\delta_1 \rightsquigarrow \delta_2$ in G^{h_k}, then the routing loop to δ_1 and to δ_2 goes through the same nodes. The node reroutings that correct the destination graph G_{δ_1} can be used to correct as well G_{δ_2} (see Fig. 9).*

This also proves that the color $C_{\delta_2}(r)$ of the routing loop (to δ_2) in G_{δ_2} is lower than the color $C_{\delta_1}(r)$ of the routing loop (to δ_1) in G_{δ_1}. But as illustrated by the second case discussed above, it can be strictly lower.

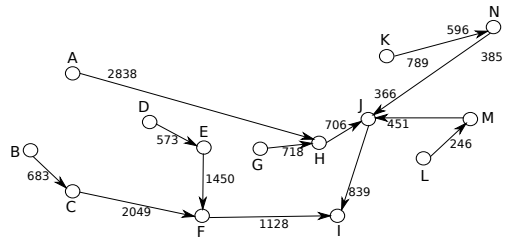
VII. Evaluation of the Enhanced Graceful Restart

To illustrate the potential gains of the proposed enhanced graceful restart, we consider the NSFNET (Fig. 10(a)), a US network based on a former NSF network topology used in many studies, e.g. [13].

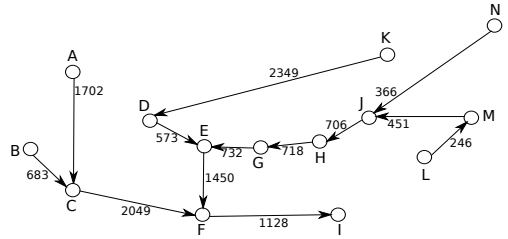
In the destination graph G_I (Fig. 10(b)), router L is supposed to be restarting and thus has a frozen forwarding table. If any link in $\{A-B, A-C, B-D, D-K, E-G, K-M, F-L, N-L\}$ fails, no routing loop will occur for destination I



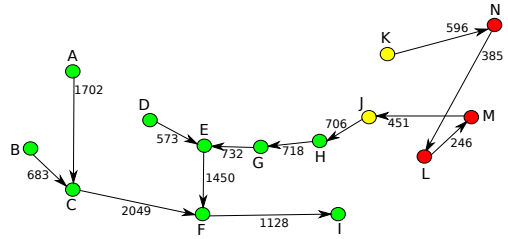
(a) 14-node NSFNET topology (link distances in km).



(b) Destination graph G_I.



(c) New destination graph G'_I, loop avoided using standardised GR.



(d) New destination graph G''_I, loop avoided using enhanced GR. Colors are 0=green, 1=yellow, 2=red.

Fig. 10.

if L keeps its frozen routing table instead of adopting the new one expected from it. Therefore, removing L from the forwarding path as it is recommended by the standardised graceful restart is unnecessary. L can safely update its routing table (towards destination I) after it completes its restart and re-establishes adjacency with its neighbors. By contrast, graceful restart is pessimistic and demands to advertise the disconnection of L, and later to announce its return in the topology, which incurs an extra round of flooding, routing table calculations and forwarding table updates, besides some unnecessary temporary reroutings. Our proposal can avoid this second round by detecting that no routing loop is about to occur, even if the rebooting router does not behave exactly as expected for some short period of time.

Now suppose that link $I-J$ fails while router L is restarting. The new shortest paths to destination I require that nodes J, K, M, N route their packets through L, the latter being expected to forward them to F. Keeping L in the topology with its frozen routing table would create the loop $M \rightarrow L \rightarrow M$. The standardized graceful restart avoids this by removing L from the topology, which results in the destination graph G'_I (Fig. 10(c)). Observe that J routes its traffic to I through H instead of M and L, and K routes its traffic through D instead of N and L. Once L completes its restart, J, K, M, N will reorient their traffic for I through L. This represents in total six modifications in their routing tables.

With our proposal, the loop $M \rightarrow L \rightarrow M$ is detected and temporarily patched, resulting in destination graph G''_I (Fig. 10(d)). Observe that routers K and N are directly set to their correct final routing. Only M and J are temporarily rerouted to patch the routing loop. Once L returns in function and updates its table, M, J can safely adopt their final value. This represents a total of four modifications in the routing tables of J, M, N, since two of them are directly positioned to their final value.

VIII. Discussion

This work shows that it is possible, at low complexity, to preserve the graceful restart procedure of OSPF routers even if the topology changes during this operation. To this end, the helper nodes of the rebooting router simply have to check if routing loops will appear, and in that case to compute the optimal patches (temporary reroutings) for all problematic destinations. They then ask the selected nodes to apply these patches, and later to remove them when the rebooting routed is back, all this in an appropriate ordering. Helper nodes are also in charge of moving from one set of temporary patches to another set, in case the topology evolves again during the reboot. This is thus a minimal extension to the existing graceful restart standard, which incurs smoother traffic perturbations since no massive rerouting is involved to bypass the potentially dangerous router. These ideas extend to several simultaneous graceful restart operations: n frozen routers can cause at most n loops toward some destination. However, patching optimally these loops will require the coordination of the n sets of helper nodes. This will be examined in a forthcoming paper, together with an extensive evaluation of this enhanced graceful restart.

Modern IP networks implement fast corrective mechanisms, as IP Fast ReRoute (IPFRR) [14], that precompute bypassses for all single link or single node failures, and then rely on ordered updates of forwarding tables (OFIB) to move to the new routing rules computed by OSPF [10]–[12]. These fast protection ideas are of course compatible with the work presented here, provided their computations take into account the frozen routing table of a rebooting router, and the patches that have been applied. Notice however that they serve a different purpose since their scope is to quickly and harmlessly isolate a faulty or dead element, while an enhanced graceful restart aims specifically at maximally exploiting a not yet dead element, despite its non optimal behavior.

Acknowledgement. This work was supported by the *High Manageability* joint research group of Alcatel-Lucent Bell Labs France and INRIA.

REFERENCES

[1] C. Hounkonnou and E. Fabre, "Enhanced OSPF graceful restart (long version)," *available at http://people.rennes.inria.fr/Eric.Fabre or on request to the authors*, 2012.

[2] J. Moy, "OSPF version 2," RFC 2328, April 1998.

[3] ——, *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[4] J. Moy, P. Pillay-Esnault, and A. Lindem, "Graceful OSPF restart," RFC 3623, November 2003.

[5] Juniper. Configuring graceful restart for OSPF. [Online]. Available: http://juniper.net/techpubs/en_US/junos/topics/topic-map/ospf-graceful-restart.html

[6] Cisco. NSF-OSPF (RFC 3623 OSPF graceful restart). [Online]. Available: http://cisco.com/en/US/docs/ios/12_0s/feature/guide/gr_ospf.html

[7] S. Ghamri-Doudane and L. Ciavaglia, "Domain-wide scheduling of OSPF graceful restarts for maintenance purposes," in *Int. Conf. on Network and Service Management, CNSM'10*, 2010.

[8] A. Shaikh, R. Dube, and A. Varma, "Avoiding instability during graceful shutdown of OSPF," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2002, pp. 883–892.

[9] ——, "Avoiding instability during graceful shutdown of multiple OSPF routers," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 3, pp. 532 –542, 2006.

[10] P. Francois and O. Bonaventure, "Avoiding transient loops during IGP convergence in IP networks," in *Proceedings of IEEE INFOCOM 2005*, vol. 1, 2005, pp. 237–247.

[11] ——, "Avoiding transient loops during the convergence of link-state routing protocols," *IEEE/ACM Trans. Netw.*, vol. 15, pp. 1280–1292, 2007.

[12] D. Hock, M. Hartmann, T. Neubert, and M. Menth, "Loop-free convergence using ordered FIB updates: Analysis and routing optimization," in *DRCN*, 2011, pp. 156–163.

[13] R. Hülsermann, S. Bodamer, M. Barry, A. Betker, M. Jäger, J. Späth, C. Gauger, and M. Köhn, "A set of typical transport network scenarios for network modelling," in *Beiträge zur 5. ITG-Fachtagung Photonische Netze*, May 2004, pp. 65–72.

[14] M. Gjoka, V. Ram, and X. Yang, "Evaluation of IP fast reroute proposals," in *Proceedings of IEEE COMSWARE '07*, 2007.