Avoiding transient loops during IGP convergence in IP networks

Pierre Francois, Olivier Bonaventure
Dept CSE, Université catholique de Louvain (UCL), Belgium
Email : {francois,bonaventure}@info.ucl.ac.be
http://www.info.ucl.ac.be

Abstract— **When the topology of an IP network changes due to a link failure or a link metric modification, the routing tables of all the routers must be updated. Each of those updates may cause transient loops. In this paper, we prove that by ordering the updates of the routing tables on the routers, it is possible to avoid all transient loops during the convergence of ISIS or OSPF after a planned link failure, an unplanned failure of a protected link and after a link metric modification. We then propose a protocol that allows the routers to order the update of their routing tables to avoid transient loops without requiring any complex computation.**

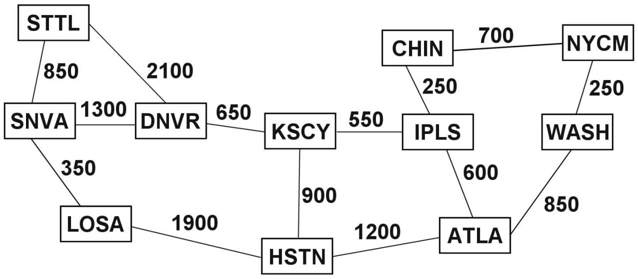

Fig. 1: Internet2 topology with IGP costs

I. Introduction

The link-state intradomain routing protocols that are used in IP networks [1], [2] were designed when IP networks were research networks carrying best-effort packets. The same protocols are now used in large commercial IP with stringent Service Level Agreements (SLA). Today, for most Internet Service Providers, fast convergence in case of failures is a key problem that must be solved [3], [4].

Vendors are actively working on improving their implementations to achieve faster convergence [4], [5]. Solving the fast convergence problem is complex as it involves detecting the failure on the attached router, producing a new Link State Packet (LSP) describing the failure, flooding this new LSP and finally updating the routing tables in all the routers using the failed resources in the network.

Achieving very fast convergence in an IP network will also require temporary tunnels to quickly reroute traffic around failures, as in MPLS networks [6]. Several solutions to establish those tunnels have been recently proposed in the literature [7]–[10]. Unfortunately, in a pure IP network, using a protection tunnel to locally reroute the traffic around the failed link is not sufficient as transient microloops may occur during the update of the routing tables of the other routers in the network. In fact, a packet that goes in the protection tunnel might not be forwarded on the shortest path to its destination and thus the routers have to update their FIB by considering the topology change in their SPT.

To understand this problem, let us consider the Internet2/Abilene backbone[1]. Figure 1 shows the IGP topology of this network. Assume that the link between IPLS and KSCY fails and that a protection tunnel is established between IPLS and KSCY via ATLA and HSTN. When ATLA receives a packet with destination KSCY, it forwards it to IPLS, which forwards it back to ATLA, but inside the protection tunnel to KSCY.

But this suboptimal routing should not last long, and thus the routers must converge, i.e. adapt to the new shortest paths inside the network and remove the tunnel. If the link is protected, the reachability of the destinations is still assured and thus the adaptation to the topological change should be done by avoiding transient loops rather than by urging the updates within each router. The new LSP generated by IPLS indicates that IPLS is now only connected to CHIN and ATLA. Before the failure, the shortest path from WASH to KSCY, DNVR, STTL and SNVA was via NYCM, CHIN and IPLS. After the failure, NYCM will send its packets to KSCY, DNVR, STTL and SNVA via WASH, ATLA and HSTN. During the IGP convergence, transient loops may occur between NYCM and WASH when KSCY-IPLS fails depending on the order of arrival of the LSPs describing the failure. If NYCM updates its routing table before WASH, the packets sent by NYCM to DNVR via WASH will loop on the WASH-NYCM link. To avoid causing a transient loop between WASH and NYCM, WASH should update its routing table

[1] This network is much smaller than large ISP backbones, but it is one of the few networks whose detailed IGP topology is publically available. We verified that similar transient loops could occur in larger ISP backbones, but the size of those backbones and NDA issues prevented us from using them as examples in this paper. Note that the IGP metricss have been rounded off to facilitate the understanding of the topology. The round off does not influence the routing tables of the network.

before NYCM for this particular failure. A detailed analysis of the Internet2 topology shows that transient routing loops may occur during the failure of most links, except STTL-DNVR and STTL-SNVA. The duration of each loop will depend on how and when the routing table of each router is updated. Measurements on commercial routers have shown that updating the routing table may require several hundred of milliseconds [3], [4]. Thus, transient routing loops of hundred milliseconds or more are possible [11]. Similar transient loops can occur in MPLS networks using LDP depending on how LDP is used [7], [9].

As shown with the simple example above, the transient routing loops depend on the order in which the updates of the routing tables are performed. In the remainder of this paper, we first discuss in section II other types of changes to an IP Network that must be handled without causing transient routing loops. In section III, we prove that the updates of the routing tables can be ordered to avoid transient loops after a link failure, a link up, or a link metric modification. Then, in section IV, we present a protocol that permits routers to respect the proposed orders, without requiring complex computations in the routers. At last, in section V, we review the solutions that have already been proposed to enhance the convergence of IGP. Note that the problem of transient routing loops has been rarely tackled in the literature.

II. Topology changes in IP Networks

Several types of changes can occur inside the topology of an IP network. The most common type of failure is the failure of a link. A network typically contains point-to-point links and LANs. Point-to-point links are typically used between Points of Presence (POPs) while LANs are mainly used within POPs. We focus on point-to-point links in this paper as there are special techniques to protect LANs [12] used in ISP networks.

When a point-to-point link fails, two cases are possible. If the link is not locally protected, the IGP should converge as quickly as possible. If the link is protected with a special tunnel or another technique [7]–[10], the IGP should converge without causing transient loops as the traffic passes through the tunnel during the IGP convergence. We will call such events *link down* events in this paper.

It should be noted that *link down* events are often caused by manual operations [13] and thus can be considered as planned events. Surveys conducted by a large ISP [3] revealed that, over a five month period, 45 % of the failure events occurred during maintenance hours. Another ISP [14] indicates that over one month, 75 % of the IS-IS events were caused by maintenance operations. Another study [15] mentions that 20 % of all *link down events* were planned. Those planned events should not cause transient routing loops [14].

It is also important to consider the increasing integration between the IP network and the underlying optical network. As the integration with the optical layer increases, the topology of IP networks will change more frequently than today. For example, [16] proposed to allow routers to dynamically establish optical links to handle traffic spikes. Several approaches have been proposed with MPLS tunnels. Once a new optical link or MPLS tunnels becomes active, an IGP adjacency will be established between the attached router and the link will be advertised in the IGP [17]. Unfortunately, the addition and removal of each of those tunnels can cause transient loops in the network.

Another source of changes in IP networks are the IGP metrics. Today, network operators often change IGP metrics manually to reroute some traffic in case of sudden traffic increase. Furthermore, several algorithms have also been proposed to automate this tuning of the IGP metrics for traffic engineering purposes [18]. Today, those algorithms are mainly implemented in network planning and management tools [19]. However, ISPs are still reluctuant to use such tools to frequently change their IGP metrics as each change may create transient routing loops in their network.

A second type of important events are those that affect routers. Routers can fail abruptly, but often routers need to be rebooted for software upgrades. For example, figure 6 of [15] shows that during September and October 2002, many links of the Sprint network “failed” once per week during maintenance hours. Those failures are probably due to planned software upgrades of all routers in the network.

When an IS-IS router needs to stop forwarding IP packets, IS-IS can flood a new LSP indicating the router as overloaded [2]. Some ISPs have even defined operational procedures [14] to bring routers down by changing link metrics and setting the `overload bit`, but those procedures are not sufficient to ensure that transient loops will not occur during the IGP convergence. The graceful restart extensions [20]–[22] could be used when a router is rebooting. However, those extensions cannot apply if the maintainance operation jeopardises traffic forwarding in the router. For example, routers that do not have the forwarding plane and control plane separated cannot use those extensions.

As shown by the above discussion, there are many different types of changes in IP networks that should be handled without risking to create transient routing loops in the network. If a protection tunnels is used in the case of a sudden link failure, the event can be also considered as non-urgent because the reachability of the nodes is still assured although the protection paths to reach them can be sub-obtimal.

III. An order for link events

In this section, we show how to handle non-urgent topology changes. In order to let routers perform the updates of their Fowarding Information Base (FIB) that correspond to those topology changes without creating routing loops, we define an order on the FIB updates on the routers that are impacted by the topology change. Conceptually, the FIB is a data structure that reflects the forwarding information contained in the IP routing table, and that is maintained in each interface of the router. When the FIB of all the routers are such that they cannot cause routing loops or loose traffic in a ”blackhole”, we say that routing is consistent.

Firstly, we introduce some general considerations about the proposed rerouting schemes. We then present an order to preserve routing consistency during the convergence phase that follows a non-urgent link down event. Next, we explain how to deal with link up events, and, at last, we explain how a link metric modification can be handled by respecting the same orders than the ones presented for link up and down events. Due to space limitations, we do not present the orders on the FIB updates that follow a router down or a router up event. However, those orders and corresponding protocols are very similar to the ones that are described for link events and thus we give the intuition of these when we present the protocols for link events.

The proposed orders always allow each router to perform all the FIB updates corresponding to the event in one shot, once it has been authorized by its neighbors to do it. In other words, a router will have to wait for clearance to update the whole set of destinations that are concerned by the event. This approach is much appreciated since it does not require to maintain and propagate state information about all the concerned prefixes in each concerned routers. For example, when a router R has to gracefully reroute packets passing through a link $(X \rightarrow Y)$ that will be shut down, R will have to wait until a particular subset of its neighbors have updated their FIB. Once R has been authorized to update its FIB for this event, it will update its forwarding information for all the prefixes that are impacted by the failure of link $X \leftrightarrow Y$, in one shot. Note that X and Y will update their FIB after all other rerouting routers, and that the link will only be shut down once both routers have updated their FIB and thus once packets are not forwarded on the link anymore.

The proposed orders are not inspired by loop-free distance-vector routing mechanisms [23]. They come from properties of the Shortest Path Trees that are computed in all the routers implementing IS-IS or OSPF. This implies that only small modifications to the current implementations are required to implement the order. In fact, the protocol presented in section IV only needs a read access to the shortest path tree that is already computed in currently deployed routers.

A. *Managing* link down *events*

When an operator shuts an interface down, link state packets are sent by the attached routers. Some routers of the network will need to update their FIB to take the topology change into account. Those routers are said to be rerouting routers for this topology change. We assume that when a link is being gracefully shut down, it could remain operational for a short while in order to let the IGP converge without causing transient loops nor packet loss. In the case of the failure of a protected link, the concerned packets arriving at the failing link will be sent in the tunnel to finally reach their destination, and thus the IGP convergence should also be performed without causing transient loops nor packet loss in that case.

Firstly, we introduce a few notations and basic properties of the IGP routing dynamics. We will then use those properties to prove the correctness of the proposed order. We always assume that routing is consistent when the considered event occurs.

SPT_N is the shortest path tree, rooted on N and computed from the current link state database of N.

$P^{init}_{A \rightarrow B}$ is the path followed by packets from A with destination B, before the event. This path does not contain a loop, as we assume routing consistency before the event.

$P'_{A \rightarrow B}$ is the path followed by packets from A to B, immediately after the FIB update of A. By using an order on the FIB updates, we will prove that this path cannot contain a loop.

$P^{final}_{A \rightarrow B}$ is the path followed by packets from A to B, after all the routers of the network have converged. This path cannot contain a loop, as the final routing state is consistent once all routers have updated their FIB.

If the event makes a node S reroute the packets with destination D to its neighbor N, then $\{S, P^{final}_{N \rightarrow D}\} = P^{final}_{S \rightarrow D}$. This is explained by the fact that S will forward packets with final destination D to N, after its FIB update, and given the definition of $P^{final}_{N \rightarrow D}$.

Let us assume that link $X \leftrightarrow Y$ is going to fail in the network. The routers having directed link $X \rightarrow Y$ or $Y \rightarrow X$ in their SPT are the affected nodes and will need to compute a new SPT. They may have to update their FIB, in order to reflect the changes in their SPT. The routers that do not use link $X \leftrightarrow Y$ in their SPT are not affected by the failure. The packets transmitted by those routers will be consistently routed before, during, and after convergence, as none of their paths to any destination contains link $X \leftrightarrow Y$. This property is rendered by Lemma III.1. From this lemma, we can deduce that any rerouting node S never creates routing loops when it reroutes packets to a node N that is not affected by the failure, as these packets will always be routed by N (and by its downstream nexthops) as if the link $X \leftrightarrow Y$ never existed. This is expressed by Proposition III.2.

Lemma III.1

$$\forall S \mid X \rightarrow Y \notin SPT_S \; and \; Y \rightarrow X \notin SPT_S :$$
$$\forall D : P^{init}_{S \rightarrow D} = P^{final}_{S \rightarrow D}$$

Proposition III.2

$$\forall N \mid X \rightarrow Y \notin SPT_N \; and \; Y \rightarrow X \notin SPT_N :$$
$$P'_{S \rightarrow D} = \{S, N, \ldots, D\} \Rightarrow P'_{S \rightarrow D} = P^{final}_{S \rightarrow D}$$

As an example, let us consider the topology of Figure 1. If the link $SSTL \leftrightarrow SNVA$ fails, traffic from $DNVR$ will not suffer as $SSTL \leftrightarrow SNVA \notin SPT_{DNVR}$. We thus have $\forall D$: $P^{init}_{DNVR \rightarrow D} = P^{final}_{DNVR \rightarrow D}$. When $STTL$ will perform its FIB update, it will reroute packets on link $STTL \rightarrow DNVR$. As $\forall D$: $P^{init}_{DNVR \rightarrow D} = P^{final}_{DNVR \rightarrow D}$, this rerouted traffic will reach its destination.

Next, let us notice that a router S, having the link $X \rightarrow Y$ in its SPT, can always reroute compromised packets (packets to $D \mid D$ is downstream of $X \rightarrow Y$ in SPT_S), to a neighbor N that has link $Y \rightarrow X$ in its SPT, without creating routing loops. In fact, if those packets passed through link $X \rightarrow Y$, N cannot use link $Y \rightarrow X$ to reach their destination. Otherwise, before the failure, those packets would have passed through both links $X \rightarrow Y$ and $Y \rightarrow X$. This is impossible since this would imply a routing loop between X and Y before the event, although routing is assumed to be consistent at that moment. We thus know that the packets rerouted by a node using directed link $X \rightarrow Y$, to a node using directed link $Y \rightarrow X$, reach a router that is not impacted by the failure, for this destination, and from which routing to this destination will thus remain consistent during the convergence phase. Combining this property with lemma III.1, we obtain the lemma III.3. It expresses the fact that if a router reaches a destination D via link $X \rightarrow Y$, then routers having link $Y \rightarrow X$ or routers that do not have link $X \leftrightarrow Y$ in their SPT are not rerouting routers for destination D, when the link $X \leftrightarrow Y$ fails, and packets with final destination D arriving to them will be consistently routed towards D.

Lemma III.3

$$\forall D : \exists S \mid X \rightarrow Y \in P^{init}_{S \rightarrow D} :$$

$$\forall N \mid Y \rightarrow X \in SPT_N or X \leftrightarrow Y \notin SPT_N :$$

$$P^{init}_{N \rightarrow D} = P^{final}_{N \rightarrow D}$$

The proof is done by contradiction. Its reasoning was presented above.

From this lemma, we can deduce a second proposition, expressing that any node S using directed link $X \rightarrow Y$ can always reroute packets with final destination D to another node N that does not have the link $X \rightarrow Y$ in its SPT, without causing routing loops.

Proposition III.4

$$\forall D, S \mid P_{S \rightarrow D} = \{S, \ldots, X, Y, \ldots, D\} :$$

$$P'_{S \rightarrow D} = \{S, N, \ldots, D\} \text{ } and \text{ } X \rightarrow Y \notin P^{init}_{N \rightarrow D}$$

$$\Rightarrow P'_{S \rightarrow D} = P^{final}_{S \rightarrow D}$$

This proposition directly comes from Lemma III.1 and Lemma III.3, and Proposition III.2

As an example, let us consider the topology of figure 1. We can see that $NYCM \rightarrow WASH \in SPT_{IPLS}$ and that $WASH \rightarrow NYCM \in SPT_{ATLA}$. If link $NYCM \leftrightarrow WASH$ fails, packets rerouted by $ATLA$ to $IPLS$ have a final destination which is downstream of link $WASH \rightarrow NYCM$ in SPT_{ATLA}. Those destinations are reached by $IPLS$ without using the link $NYCM \leftrightarrow WASH$ and thus the forwarding of packets rerouted by $ATLA$ on link $ATLA \rightarrow IPLS$ will arrive at destination, as they are routed from $IPLS$ without using the link that will fail.

Thus, it is always safe to reroute traffic to a node that does not use the failing link, or to a node that uses it, but in the other direction. We thus may also say that transient routing loops can only be created between routers sharing the directed failing link in their SPT.

Now, let us look at $rSPT_{X \rightarrow Y}(Y)$, the reverse SPT of Y, cut at link $X \leftrightarrow Y$. More precisely, this graph is the graph that contains all (and only) the shortest paths to Y that terminate with link $X \rightarrow Y$. This graph represents all the paths to Y (and implicitly all the paths through Y) that are affected by the failure of link $X \rightarrow Y$. $rSPT_{X \rightarrow Y}(Y)$ is an acyclic graph[2] without oriented loops. It contains all the nodes that use link $X \rightarrow Y$ to reach Y, and possibly some destinations downstream of Y[3]. The subtree of $rSPT_{X \rightarrow Y}(Y)$, rooted on a node A, contains all the routers that use link $X \rightarrow Y$ to reach Y, *and* that reach this link via A. A router R, having no child on this tree, has thus no neighbor that uses link $X \rightarrow Y$ and that reaches it via R. Note that in that case, it is impossible to find a router of the network that uses link $X \rightarrow Y$ and that reaches it thanks to R. If R performs rerouting, after the failure, it will reroute packets to routers that

- do not use link $X \rightarrow Y$, or
- use link $X \rightarrow Y$ but do not use R to reach it

If a router that is a leaf on $rSPT_{X \rightarrow Y}(Y)$ performs a FIB update, it will not cause a routing loop. In fact, the routers that do not use link $X \rightarrow Y$ will forward the packets on paths that remain valid during and after the convergence, and the routers that use link $X \rightarrow Y$ but do not use R to reach it will forward the packets on the compromised but still consistent paths to their destination.

But, one could wonder if this loop-free property remains true when the leaves of $rSPT_{X \rightarrow Y}(Y)$ perform concurrent updates of their FIB. Moreover, it must be proved that, when a router R transiently becomes a leaf in this tree, as all the routers that were using it to reach $X \rightarrow Y$ have updated their FIB, it can update its FIB without creating routing loops.

Keeping the previous lemmas and propositions in mind, we thus need to prove the theorem III.5. This theorem completely describes the proposed order on the FIB updates to handle a link failure without causing transient loops.

Theorem III.5 *To avoid transient routing loops during the entire IGP convergence phase following the failure of the link $X \rightarrow Y$, any rerouting router should update its FIB only after all of its children in $rSPT_{X \rightarrow Y}(Y)$ have performed their own FIB update.*

Let us assume that the proposed order is respected and let us consider the forwarding of packets that were previously forwarded through link $X \rightarrow Y$ and that reach node R, which had link $X \rightarrow Y$ in its SPT, before the event.

If R has not updated its FIB yet, then all the routers on all the paths from R to X have not updated their FIB yet, as they respect the proposed order. Note that, by definition of the SPT, those routers also have the link $X \rightarrow Y$ in their SPT. R is thus forwarding compromised packets on the outdated but consistent paths, passing through link $X \rightarrow Y$, and reaching

[2] If Equal Cost Multipath is not used, $rSPT_{X \rightarrow Y}(Y)$ is a tree, otherwise it is an acyclic graph.

[3] In the case of a network with symetrical link metric assignment, $rSPT(Y) = SPT(Y)$, by the definition of a shortest path.

their final destination, according to Lemma III.3 and as Y cannot have link $X \rightarrow Y$ in its SPT. Note that in the case of ECM, R may also forward those compromised packets on paths that do not contain the link $X \rightarrow Y$. Those paths will remain consistent during the convergence phase, according to Lemma III.1, and will still be used after the IGP convergence. Proposition III.6 renders this property.

Proposition III.6 *If link $X \rightarrow Y \in SPT_S$ and S has not updated its FIB at time t, and if the order defined by Theorem III.5 is respected by the routers of the network, then*

$$\forall D, P^t_{S \rightarrow D} = P^{init}_{S \rightarrow D}$$

If R has already updated its FIB, we are sure that all the routers that were using R to reach link $X \rightarrow Y$, have already updated their FIB, as R is respecting the proposed order. R is now forwarding compromised packets to a new set of nexthops $\{N_1, N_2, \ldots, N_i\}$. Let us look at each particular nexthop N_j. If N_j has not updated its FIB yet, we are in the case depicted in the preceding paragraph. Packets rerouted to N_j will not loop back to R as this router has not updated its FIB yet and thus its paths to destinations that are reached by passing through link $X \rightarrow Y$, remain used and valid as they are composed of routers that have not yet updated their FIB (Proposition III.6). Note that those paths cannot contain R, as R has been allowed to update its FIB. The other paths from N_j, to compromised destinations, that do not pass through link $X \rightarrow Y$ remain valid during and after convergence. Note that N_j could also be a router that did not have the link $X \rightarrow Y$ in its SPT. In that case, packets rerouted to N_j will be consistently routed until the end of the convergence phase, according to lemma III.3. If N_j has already updated its FIB, its case matches the case of R, depicted in this paragraph. But we can see that packets with a given destination D cannot loop between routers that have already updated their FIB or that are non rerouting routers for destination D. It is obvious, as the contrary would mean that packet forwarding would remain inconsistent between those routers when the convergence phase is completed.

Thus, remembering that compromised packets arriving at a router that has not updated its FIB is proved to be consistently forwarded to its final destination (Proposition III.6), and as compromised packets reaching a node that did not use the oriented link $X \rightarrow Y$ are proved to be consistently forwarded to their final destination (Lemma III.3), we can deduce that packets sent by a node that has already updated its FIB will follow a (possibly empty) loop-free path crossing nodes that have already updated their FIB, then it will arrive at a node that uses link $X \rightarrow Y$ and that has not updated its FIB or it will arrive at a node that does not use link $X \rightarrow Y$. In both cases, those packets are proved to be consistently routed to their final destination.

We have proved that routers respecting the proposed order cannot bring routing inconsistency during the convergence phase following an event that must be gracefully handled, when they perform their FIB update.

In section IV, we describe a protocol that let routers comply with the presented order on the FIB updates, in the case of a graceful link shutdown.

B. An order for link-up events

In order to avoid routing loops in the case of a link-up event (let us assume link $X \rightarrow Y$ goes up), routers should wait for the neighbors to which they will reroute some traffic to have completed a FIB update that considers the link-up. This order is based on the same observations of the SPT dynamics as for a link-down event. This time, the $rSPT$ that describes the order for the FIB updates on the routers is computed by taking the topology change into account. If link $X \rightarrow Y$ comes up, X and Y can update their FIB immediately. But a distant router has to wait for its parents in $rSPT^{final}_{X \rightarrow Y}(Y)$ to have completed their FIB updates. By proceeding like this, packets are only rerouted to routers that already use the upcoming link, and thus traffic inconsistencies cannot happen. This is explained by the fact that the parents of a rerouting router S in $rSPT^{new}_{X \rightarrow Y}(Y)$ are the sole routers that will receive rerouted traffic from S, after the update of S's FIB. Thus, if all of them have updated their FIB before S, the FIB update on router S will cause rerouting to routers that are already using the new link in their FIB. Thus, packets will follow a path containing ”up to date” routers, and the packets will thus reach link $X \rightarrow Y$. As X and Y are, by definition of the order, the first routers that update their FIB, rerouted packets arriving at link $X \rightarrow Y$ will be consistently routed to their final destination. The following proposition expresses this order. Its proof is quite straightfoward, it is based on the same reasoning.

Proposition III.7 *In order to avoid routing inconsistencies during the convergence phase following the link up of $X \rightarrow Y$ in a network, routers should always update their FIB after their parents in $rSPT^{final}_{X \rightarrow Y}(Y)$*

C. Considering metric updates

In the previous sections of this document, we only considered link-down and link-up events. But, in practice, simple updates of link metrics can also happen, for example, for traffic engineering purposes [18]. This kind of events can be gracefully treated, as they do not imply a direct inconsistency in the global routing state of the network. In other words, reachability is still assured between the event and the first reactions of the routers to this event.

Respecting the same orders as the one described for graceful link-up and link-down events will assure routing consistency for link metric decrease and link metric increase, respectively. In fact, the former orders can be considered as worst cases for the latter since link-down/up events are metric updates. Those updates imply rerouting for a set of destinations that is included in the set of the affected destinations if the event had been a link-up/down event. So, respecting an order that assures routing consistency for a “larger” set of destinations than the actual one, will lead to consistent transient routing states.

D. Considering router events

The FIB update order to respect when a router is shut down or comes back up, comes from the same properties of the IGP routing dynamics. When a router N is shut down, a rerouting router R should reroute only when all the routers that were using it to reach N have updated their FIB.

By proceeding like this, compromised packets reaching a router that has not updated its FIB yet will be fowarded to routers that have not performed their FIB update either. Those packets will thus reach a neighbor of N, that will send the packets to N in the case of a graceful router shutdown, or in the protection tunnnel in the case of a protected sudden router failure. As packets that do not reach an "outdated" router are proved to reach their destinations, the order to respect is thus well described by $rSPT(N)$; i.e. a router belonging to this tree is allowed to update its FIB once its children in the tree have updated their own FIB.

When N comes back up, a router R will only update its FIB when all the routers on its new paths to reach N have updated their FIB. The order to respect is thus also well described by $rSPT(N)$, but, this time, a router R is allowed to update its FIB once its parents in the tree have updated their own FIB.

IV. A PROTOCOL TO ORDER FIB UPDATES

To respect the orders on the FIB updates proposed in section III, a router does not need to compute any graph, tree, or other complex data structure. When an event occurs, it only needs to use to the SPT that it already maintains, and exchange messages with its neighbors.

We first note that it is easy to modify the encoding of the LSPs to indicate that a link change should be treated gracefully by all routers inside the network. This information can be placed in the LSP by defining a new type of TLV field or by using the syntax proposed in [24]. A router receiving a LSP containing this TLV will easily determine the change by comparing the new LSP with the previous one.

To compute the order on the FIB updates, we utilise the IS-IS[4] HELLO PDUs that are regularly exchanged between neighbouring routers. Usually, those messages are exchanged to establish the IS-IS adjacency and to detect link or router failures. This failure detection is achieved by a regular transmission of the HELLO PDUs and the associated Holding-Timer [2]. However, nothing prevents an IS-IS router from transmitting HELLO PDUs more often than required by its HelloTimer. Since HELLO PDUs can also contain sub-TLVs [2], they can be used to allow the routers to compute the ordering of the FIB updates described in the previous section.

We define a new type of "event" TLVs that can be optionally included inside HELLO PDUs. Those "event" TLVs will be used by a router to determine when it can safely update its FIB without risking to create transient loops.

The link event TLV is used to indicate a change in the metric of directed edge $upstream \rightarrow downstream$. It contains the following information : a FIB bit, the LSP id of the $upstream$ node attached to the link, the LSP id of the $downstream$ node attached to the link, the old ISIS metric (a value of `MAX_METRIC` indicates a new adjacency) and the new ISIS metric (a value of `MAX_METRIC` indicates that an adjacency is being shutdown)

Each router will analyse the FIB bit of the HELLO PDUs received from its neighbors to determine when it can safely update its FIB without causing a transient loop.

When router R receives a LSP containing one (or more) graceful link changes it shall first flood the LSP. All the graceful changes indicated in the LSP will be placed in a special memory before being inserted in the LSDB to be used to compute the SPT and to update the FIB. A router should only start to handle a graceful change once the LSP describing the change has been acknowledged by all its neighbors.

Before explaining the protocol, we need to introduce a few notations. The two types of events that we consider in this paper are :

- $N_1 \uparrow N_2$: the metric of the directed link $N_1 \rightarrow N_2$ increases
- $N_1 \downarrow N_2$: the metric of the directed link $N_1 \rightarrow N_2$ decreases

$HELLO(E, F)$ is a shorthand for a HELLO PDU that contains the event TLV for event E and the FIB bit set to F. $Neighbours(R)$ is the set of routers that are adjacent to router R. $SPT(R)$ is the shortest path tree computed by router R, based on its current LSDB. $Nexthops(R, N_1)$ is the set of neighbors that router R uses as nexthops to reach node N_1. This set may contain several nodes when Equal Cost Multipath (ECM) is used. We will also use $Nexthops(R, N_1 \rightarrow N_2)$ to indicate the set of nexthops that router R uses to reach link $(N_1 \rightarrow N_2)$. $W(E)$ is the waiting set containing all the neighbors from which our router must receive a confirmation before updating its FIB for event E. There is one waiting set per event being handled. $I(E)$ is the set of nodes to which our router must send a confirmation after having updated its FIB for event E.

A. Metric increase events

Let us remember that, in the case of the failure of the link $X \rightarrow Y$, routers are not allowed to update their FIB before their children in $rSPT_{X \rightarrow Y}(Y)$. Thus, a router S must wait for its neighbor N if N is one if its child in this tree. By definition, this can only be the case if N uses link $X \rightarrow Y$ (thus if $X \rightarrow Y \in SPT_N$) and S is the nexthop from N to X. Thus, N is able to identify the neighbor(s) that will be allowed to update their FIB only after it has updated its own one. Note that N could reach link $X \rightarrow Y$ via S, but reach S via another path ($P_{N \rightarrow S}$) than the link $N \rightarrow S$. S will nevertheless wait for N as S will have to wait for the routers composing this path, and those routers will have to wait for N before updating their FIB.

The problem becomes easy to solve. When a router S receives the LSP describing the failure of link $X \rightarrow Y$, it checks if it is affected by the event ($X \rightarrow Y \in SPT_S$). If it

[4] A similar extension can be defined for OSPF.

is, it must identify the routers that it has to wait for. For this, it sends a special HELLO PDU to each of its neighbors. This message contains the description of the event, and the FIB bit which is unset if S uses the neighbor to reach $X \rightarrow Y$. A neighbor N of S will reply to this message to announce S if it uses it to reach $X \rightarrow Y$ (by setting the FIB bit or not). When S received all the answers to its messages, it is able to identify the neighbors that are its children in $rSPT_{X \rightarrow Y}(Y)$ at a distance of one hop. If it is a leaf in this tree, S will only receive answers with the FIB bit set. It can thus update its FIB. Once this update is done, it must send a new message, with the FIB bit set, to the routers to which it previously sent a message with an unset FIB bit. By doing this, S will indicate its parents in $rSPT_{X \rightarrow Y}(Y)$ that S and its children in $rSPT_{X \rightarrow Y}(Y)$ have all updated their FIB. If S received replies with an unset FIB bit from a neighbor N, S must wait for N to have updated its FIB and sent a new message with a set FIB bit before it becomes allowed to update its FIB.

Thus, the basic principle of the protocol in the case of metric increase events is that a router that uses a link whose metric increases gracefully will not update its FIB until it has received a HELLO PDU with the FIB bit set to 1 from all its neighbors.

A typical implementation of this protocol would have to react to three types of events : the arrival of the LSP indicating the change, the arrival of a HELLO PDU and the recovery from lost HELLO PDUs. Figure 2 shows how router R reacts to a metric increase for link $X \rightarrow Y$ received in a LSP. Figure 4 will explain how router R responds to the received HELLO PDUs and figure 6 describes the utilization of timers to recover from the possible loss of HELLO PDUs.

```
Metric increase event for link $X \rightarrow Y$ :
if $(X \rightarrow Y) \in SPT(R)$ then
    $W(X \uparrow Y) = Neighbours(R)$;
    $N(X \uparrow Y)$=$Nexthops(R, X \rightarrow Y)$;
    foreach $N \in W(W \uparrow Y)$ do
        if $N \in N(X \uparrow Y)$ then send($N, HELLO(X \uparrow Y, 0)$);
        else send($N, HELLO(X \uparrow Y, 1)$);
    end
    start_timer($Timer(X \uparrow Y)$);
end
else
    /*No update of the FIB required */
    insert(change($X \uparrow Y$),LSDB);
end
```

Fig. 2: Processing of link metric increase event

As shown in figure 2, two cases must be considered by a router processing a metric increase event. If the changing link is not contained in the router's SPT, then the router does not utilize this link. Thus, its FIB is by definition already up-to-date and the change is inserted in its LSDB. Otherwise, the router will send HELLO PDUs with the FIB bit set to 0 to the neighbors that it uses to reach the changing link and HELLO PDUs with the FIB bit set to 1 to the neighbors that it does not utilize to reach the changing link. The router will wait until it has received a HELLO PDU with the FIB bit set to 1 from all its neighbors. This is implemented by using the $W(X \uparrow Y)$ waiting set in figure 4. This set initially contains all the neighbors of router R. It is updated (see figure 4) each time a $HELLO(X \uparrow Y, 1)$ PDU is received.

The pseudocode of figure 2 can be better understood by considering for example the graceful failure of link $A \leftrightarrow B$ in a network with four routers (figure 3). When this link fails, A and B will flood their LSP reporting a metric of `MAX_METRIC` for this link. In this network, the RSPT of router A is $C \rightarrow A \leftarrow B \leftarrow D$ (and $D \rightarrow B \leftarrow A \leftarrow C$ for B's RSPT).

Figure 3 shows the HELLO PDUs that are sent by the four routers of the simple network when link $A \leftrightarrow B$ fails gracefully. Router C only uses directed link $A \rightarrow B$ in its SPT[5]. Thus, C can already install the change for link $B \rightarrow A$ in its LSDB. For the change of link $A \rightarrow B$, router C's nexthop is A. Router C sends $HELLO(A \uparrow B, 0)$ to router A. This HELLO PDU indicates that router C is a child of A in $rSPT_{A \rightarrow B}(B)$. Router C sends $HELLO(A \uparrow B, 1)$ to router D since C does not use D to reach the failed link. The waiting set of router C for the $A \uparrow B$ change initially contains its neighbors, routers A and D.

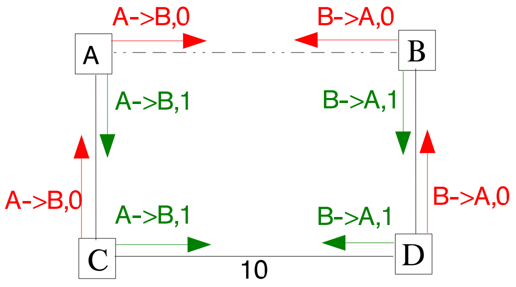

Fig. 3: Transmission of the first HELLO PDUs in the simple network

The order of the FIB updates is defined by the reception of the $HELLO(X \uparrow Y, 1)$ PDUs. Figure 4 shows how a router will reply to the HELLO PDUs that it receives. A router will remove a neighbor from its waiting set $W(X \uparrow Y)$ once it has received $HELLO(X \uparrow Y, 1)$ from this neighbor. When this set becomes empty, the router can update its FIB. After the FIB update, it sends $HELLO(X \uparrow Y, 1)$ to all its neighbors.

Figure 5 shows the first wave of the HELLO PDUs sent in response to the PDUs shown in figure 3. Router C sends $HELLO(B \uparrow A, 1)$ to router D as router C does not utilize this directed link. For the same reason, router A sends $HELLO(B \uparrow A, 1)$ to router B. As router C has received $HELLO(A \uparrow B, 1)$ from router A, it removes A from its waiting set for this event, $W(A \uparrow B)$. Router A sends $HELLO(A \uparrow B, 1)$ in response to the $HELLO(A \uparrow B, 0)$ sent by router C since C is not used by A to reach link $A \rightarrow B$.

[5] As noted earlier, a router cannot utilize both directed links $A \rightarrow B$ and $B \rightarrow A$ in its SPT.

```
  Arrival of $HELLO(X \uparrow Y, F)$ from neighbor $N$:
  if $FIB \subset change(X \uparrow Y)$ then
1     if $F$==$0$ then  send($N$,$HELLO(X \uparrow Y, 1)$);
  end
  else
      /*change($X \uparrow Y$) is being handled */
      if $F$==$1$ then
          remove($N$,$W(X \uparrow Y)$);
          if $W(X \uparrow Y) = \emptyset$ then
              insert(change($X \uparrow Y$),LSDB);
              compute($SPT(R)$);
              update($FIB(R)$);
              foreach $N \in N(X \uparrow Y)$ do
              send($N$, $HELLO(X \uparrow Y, 1)$);
          end
      end
      else
2         if $N \notin N(X \uparrow Y)$ then
              send($N$, $HELLO(X \uparrow Y, 1)$);
          end
          else
          /*If such a HELLO PDU is received, the
            LSDBs are inconsistent and the change
            should be handled as a normal urgent
            change */
      end
  end
```

Fig. 4: Processing of a HELLO PDU containing a metric increase link-event TLV

Besides the reception of the $HELLO(X \uparrow Y, 1)$ PDUs that empty the waiting set, two other cases must be considered. First, router R could receive a (retransmitted) HELLO PDU from one of its neighbors after having updated its LSDB and FIB. In this case, it should reply only if the received HELLO PDU had its FIB bit set to 0 (line 1 of figure 5). Second, router R may receive a retransmitted HELLO PDU with the FIB bit set to 0 (line 2 of figure 5). If this PDU is received from a neighbor that router R does not use as a nexthop to reach the changing link, R should reply with $HELLO(X \uparrow Y, 1)$. If router R receives $HELLO(X \uparrow Y, 0)$ from a neighbor that it uses as a nexthop to reach the changing link, then the LSDBs are inconsistent since there is a potential routing loop between R and this neighbor. This can happen if routing was inconsistent when the considered event occured. For example, this can happen if another event is being treated urgently, causing transient routing loops, meanwhile routers are consistently rerouting for the considered event.

Upon reception of $HELLO(A \uparrow B, 1)$ from router D, C will remove D from its waiting set. Since this set becomes empty, C is allowed to update its FIB. After this update, router C will send $HELLO(A \uparrow B, 1)$ to router A. Upon reception of $HELLO(B \uparrow A, 1)$ from router C, D will remove C from its waiting set. Since this set becomes empty, D is allowed to update its FIB. After this update, router D will send $HELLO(B \uparrow A, 1)$ to router B. Upon reception of $HELLO(A \uparrow B, 1)$, the waiting set of router A will become empty and A will update its FIB. After this update, A will send $HELLO(A \uparrow B, 1)$ to router B. This last HELLO PDU sent by A will indicate that the $A \uparrow B$ event has been gracefully handled. At that time, A knows that it will not send anymore packets on this link. Before physically shutting down the link, A should indicate to B that it does not utilize this link anymore. This could be achieved by sending a HELLO PDU with the Hold time set to $zero$ as proposed in [25]. A would then wait until it has received a similar HELLO PDU from router B to physically shut down the interface.

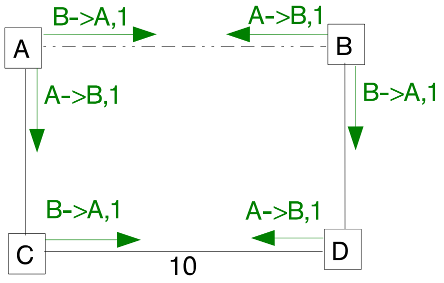

Fig. 5: The first wave of reply HELLO PDUs when link $A \rightarrow B$ fails gracefully

The last part of the protocol to be considered is the recovery from the losses of HELLO PDUs. Those losses can be recovered by protecting the waiting set for each event with a timer. The default value for this timer should be larger than the time required to compute the SPT and update the FIB. Figure 6 shows that when this timer expires, the router will retransmit HELLO PDUs to its neighbors that are still inside its waiting set. It will send $HELLO(X \uparrow Y, 0)$ to the members of the waiting set that it uses as nexthops to reach link $X \rightarrow Y$ and $HELLO(X \uparrow Y, 1)$ to the other members of $W(X \uparrow Y)$

```
Expiration of $Timer(X \uparrow Y)$ at router $R$:
foreach $N \in W(X \uparrow Y)$ do
    if $N \in N(X \uparrow Y)$ then  send($N, HELLO(X \uparrow Y, 0)$); else
    send($N, HELLO(X \uparrow Y, 1)$);
end
start_timer($Timer(X \uparrow Y)$);
```

Fig. 6: Expiration of the timer associated to a metric increase event

B. Protocol Extension for router events

Such a protocol can be easily adapted to handle router down events. Roughly, the reverse shortest path tree that defines the order on the FIB updates in the case of a link down, is ”cut” at a particular link, as described in III-A. In the case of a router down event, the whole reverse shortest path tree describes the order that must be respected. Thus, a router event TLV can be defined, and the same protocol as the one proposed for a link event can be applied. In this case, a router R can compute the subset of its neighbours that will have to wait for it by computing the value of $Nexthops(R, X)$. In the case of the link down of $X \leftrightarrow Y$, it was an empty set, $Nexthops(R, X \rightarrow Y)$ or $Nexthops(R, Y \rightarrow X)$, according to the presence and direction of this link in SPT_R.

C. *Metric decrease events*

In the case of a metric decrease of the link $X \rightarrow Y$, a router S is allowed to update its FIB once all its parents in $rSPT^{final}_{X \rightarrow Y}(Y)$ have updated their FIB. Those parents are the nodes to which S will forward traffic that will pass through link $X \rightarrow Y$. The neighbors which are parents in this acyclic graph are thus easy to identify; if $X \rightarrow Y$ is in the updated SPT (the SPT that considers the link up), then those parents are the nexthops used to reach X in this SPT. As those parents will also wait for their parents, S is allowed to update its FIB once it knows that its neighbouring parents have updated their FIB. To be aware of those updates, S can send a message to its parents. This message contains the description of the event, and a FIB bit which is unset to announce that the router has to wait for this neighbor. A neighbor that receives a message with the FIB bit unset from S will have to advise S once it has updated its FIB. Once all the parents of S for this event have sent a message to announce that they have updated their FIB, S can update its FIB and send a message of the same kind to the routers that announced to S that they had to wait for it.

When the metric of a directed link decreases or a link comes up, two cases must be considered. In the case of a simple metric decrease, the procedure defined below shall be applied upon reception of the LSP describing this change. If a link comes up, the situation is slightly different due to the two-way connectivity check [2]. An IS-IS router will only consider the link as up upon arrival of the second LSP describing the link has been received. In the case of a *link up* event, the procedure defined below shall be applied to both LSPs describing the new link upon reception of the second LSP.

The basic principle of the protocol used to handle a metric decrease event is that each router that uses the changing link will first recompute its SPT, according to IS-IS specification. This new SPT allows a router R to determine whether it will utilize the changing link or not. If it will, R will also compute the neighbours that it utilizes as nexthops to reach this link. The router will send $HELLO(X \downarrow Y, 0)$ to those neighbours since it must wait for their confirmation before updating its FIB. It will only update its FIB once it has received $HELLO(X \downarrow Y, 1)$ from all those nexthops. After having updated its FIB, the router will send $HELLO(X \downarrow Y, 1)$ to all its neighbours that sent it a $HELLO(X \downarrow Y, 0)$ PDU.

Figure 7 shows the procedure used by a router R to handle a metric decrease event. Router R first computes its new SPT taking the change into account. If the changing link does not belong to this SPT, router R's FIB is already up to date. Otherwise, router R will send $HELLO(X \downarrow Y, 0)$ to all the neighbours that it uses to reach link $X \rightarrow Y$ in its new SPT. Those PDUs indicate that R is waiting for a confirmation from these nodes before updating its FIB. $W(X \downarrow Y)$ is the set of neighbours from which a $HELLO(X \downarrow Y, 1)$ must have been received to perform the FIB update. $I(X \downarrow Y)$ is the set of neighbours that are waiting for a confirmation (i.e. $HELLO(X \downarrow Y, 1)$) from this router to update their FIB.

```
Metric decrease event for link $X \rightarrow Y$ :
compute( $SPT(R)$ with change($X \downarrow Y$));
if ($X \rightarrow Y$) $\in SPT(R)$ then
    $W(X \downarrow Y)$=$Nexthops(R, X \rightarrow Y)$;
    $I(X \downarrow Y) = \emptyset$;
    foreach  $N \in W(X \downarrow Y)$ do
    send($N, HELLO(X \downarrow Y, 0)$);  start_timer($Timer(X \downarrow Y)$);
end
else
    /*No update of the FIB required */
    insert(change($X \downarrow Y$),LSDB);
end
```

Fig. 7: Processing of a link metric decrease event

Figure 8 shows the HELLO PDUs that are sent by the four routers in our simple network, when link $A \leftrightarrow B$ comes up. In this example, router C sends $HELLO(A \downarrow B, 0)$ to router A since A is its nexthop to reach the new link. Router A sends $HELLO(A \downarrow B, 0)$ to ensure that the new link is correctly up.

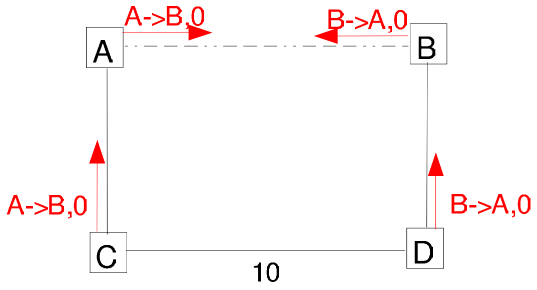

Fig. 8: The first wave of HELLO PDUs when link $A \rightarrow B$ comes up in the simple network

The processing of the $HELLO(X \downarrow Y, F)$ PDUs by router R is described in figure 9. If the change indicated in the HELLO PDU has already been installed in R's LSDB and FIB, it shall reply with $HELLO(X \downarrow Y, 1)$ if the FIB bit was set to 0. This case can happen if R is closer than N to the changing link. Otherwise, if the HELLO PDU received from node N had its FIB bit set to 1, it indicates that R does not need to wait for N to update its FIB. Router R updates its FIB as soon as its waiting set, $W(X \downarrow Y)$, is empty. After the update, router R sends $HELLO(X \downarrow Y, 1)$ to all the neighbours that it must inform. Those neighbours are contained in the $I(X \downarrow Y)$ set which is updated every time router R receives $HELLO(X \downarrow Y, 0)$.

In our example, the HELLO PDUs shown in figure 8 will be handled as follows. Router A will place router C in its $I(A \downarrow B)$ set upon reception of $HELLO(A \downarrow B, 0)$. The waiting set of A for the new link, $W(A \downarrow B)$ initially contains router B. When router B receives $HELLO(A \downarrow B, 0)$ from router A, it immediately replies with $HELLO(A \downarrow B, 1)$ since router B does not utilize link $A \rightarrow B$. For the symetrical reason, router A sends $HELLO(B \downarrow A, 1)$ to router B. The arrival of $HELLO(A \downarrow B, 1)$ allows router A to remove B from its waiting set. Since its waiting set is empty, router A can update its FIB and sends $HELLO(A \downarrow B, 1)$ to router C

```
Arrival of $HELLO(X \downarrow Y, F)$ from neighbour $N$:
if  $FIB \subset change(X \downarrow Y)$ then
    if $F$==0 then  send($N$,$HELLO(X \downarrow Y, 1)$);
end
else
    /*change($X \downarrow Y$) is being handled */
    if $F$==1 then
        remove($N$,$W(X \downarrow Y)$);
        if $W(X \downarrow Y) = \emptyset$ then
            insert(change($X \downarrow Y$),LSDB);
            compute($SPT(R)$);
            update($FIB(R)$);
            foreach $N \in I(X \downarrow Y)$ do
            send($N, HELLO(X \downarrow Y, 1)$);
        end
    end
    else
        /*F==0 */
        insert($N$,$I(X \downarrow Y)$);
    end
end
```

Fig. 9: Processing of a HELLO PDU containing a link-event TLV indicating a metric decrease event

which belongs to $I(A \downarrow B)$. Upon reception of this HELLO PDU, router C is allowed to update its FIB. For the symetrical reason, router B will send $HELLO(B \downarrow A, 1)$ to router D. At that time, all routers have updated their FIB without causing transient routing loops.

As for the metric increase events, the last issue to consider is how to recover from losses of the HELLO PDUs. As shown in figure 10, this can be achieved by protecting the waiting set with a timer and retransmitting the $HELLO(X \downarrow Y, 0)$ to the nodes belonging to this set upon expiration of this timer.

```
Expiration of $Timer(X \downarrow Y)$ at router $R$:
foreach $N \in W(X \downarrow Y)$ do
    send($N, HELLO(X \downarrow Y, 0)$);
end
start_timer($Timer(X \downarrow Y)$);
```

Fig. 10: Expiration of the timer associated to a metric decrease event

V. Related Work

The problem of avoiding transient loops during IGP convergence has rarely been studied in the literature although many authors have proposed solutions to provide loop-free routing. An existing approach to loop-free rerouting in a link-state IGP [23] require that the rerouting routers take care of routing consistency for each of their compromised destinations, separately. In fact, those mechanisms were inspired by distance-vector protocols loop-free routing mechanisms. With this kind of approach, a router should ask and wait clearance from its neighbors for each destination for which it has to reroute. This implies a potentially large number of message exchanged between routers, when many destinations are impacted by the failure. Every time a router receives clearance from its neighbors for a given destination, it can only update forwarding information for this particular one. Thus, those mechanisms require to design routers that are able to perform incremental updates of their FIB. Our solution is much simpler, uses fewer messages and can be easily integrated in existing IGPs. Morevover, they also disregard the problem of traffic loss in the case of a planned link shutdown.

In [26], a new type of routing protocol allowing to improve the resilience of IP networks was proposed. This solution imposes some restrictions on the network topology and expensive computations on the routers. Moreover, they do not address the transient issues that occur during the convergence of their routing protocol. In [27], extensions to link-state routing protocols are proposed to distribute link state packets to a subset of the routers after a failure. This fastens the IGP convergence, but does not solve the transient routing problems and may cause suboptimal routing.

While finalising our protocol, we learned about another solution [10] that avoids transient micro loops after a link failure by ordering the computation of the SPF on the routers. Our solution has two advantages compared to the one described in [10]. First, we do not need to compute any additional reverse spanning tree. Second, [10] suggests that routers statically compute a rank for the update of their routing tables. In fact, the suggested order is the same as ours, but it has to assume that all the routers concerned by the event can be rerouting routers, and thus the time at which a FIB is updated in a router is a fixed worst-case FIB update time multiplied by the rank of the router. Roughly, the rank of a router R is the depth of the tree under R in the rSPT centered on the failure, and thus the children of R will have a lower rank and thus reroute before it.

Our protocol can be considered as an optimization for this proposal. Routers that do not have to spend the whole worst-case FIB update time will send HELLO messages with the FIB bit set before their static rank time has elapsed, and thus unlock the FIB updates of their parents earlier, leading to a reduced global convergence time.

There are discussions within the IETF about mechanisms that avoid transient loops during convergence. Our solution and [10] are appreciated for their full coverage. There also discussions to shortcut the proposed order by combining it with the discovery of loop free alternates that come from path locking [28]. There are also discussions to perform synchronized FIB updates in the routers, to minimize the duration of the forwarding inconsistencies, rather than avoiding them.

VI. Conclusion

In this paper, we have first described the various types of topology changes that can occur in large IP networks. Recent measurements indicate that many of those changes are non-urgent. Furthermore, the failure of all protected links can also be considered as non urgent. When such a non-urgent change occurs, the routing tables of all routers must be updated. Unfortunately, those updates may cause transient routing loops and each loop may cause packet losses or delays. Large ISPs require solutions to avoid transient loops after those non-urgent events.

The first important contribution of this paper is that we have proved that it is possible to define an order on the updates of the routing tables that prevents the network from transient loops. We have proposed an order applicable for the failures of protected links and the increase of a link metric and another order for the establishment of a new link or the decrease of a link metric. We also proposed orders that are applicable in the case of a non-urgent router down or up event.

Furthermore, we have shown that it is possible to slightly change the current link state intradomain routing protocols to allow each router to determine when it can safely update its routing table after each non-urgent change. The main advantages of our protocol are that it does not force the routers to perform complex computations and furthermore the proposed changes can be easily added to existing link state routing protocols. Thus, our proposal could be easily used in large IP networks.

We are currently working on a protocol extension that permits to handle the cases of SRLG failure. We are also planning to investigate the problem of performing consistent FIB udpates when OSPF or IS-IS areas are used.

Acknowledgments

This work was supported by Cisco Systems within the ICI project. Any opinions, findings, and conclusions or recommandations expressed in this paper are those of the authors and do not necessarily reflect the views of Cisco Systems.

We would like to thank Mike Shand, Clarence Filsfils and Stefano Previdi for their suggestions and comments.

References

[1] J. Moy, “OSPF version 2,” Internet Engineering Task Force, Request for Comments 1247, July 1991. [Online]. Available: ftp://ftp.isi.edu/in-notes/rfc1247.txt

[2] ISO/IEC, “Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the connectionless-mode network service (ISO 8473),” ISO/IEC, Tech. Rep. 10589:2002(E), April 2002.

[3] G. Iannaccone, C. Chuah, S. Bhattacharyya, and C. Diot, “Feasibility of IP restoration in a tier-1 backbone,” *IEEE Network Magazine*, January-February 2004.

[4] C. Filsfils, “Fast IGP convergence,” January 2004, presented at RIPE47, http://www.ripe.net/ripe/meetings/ripe-47/presentations/ripe47-routing-isis.pdf.

[5] C. Alaettinoglu, V. Jacobson, and H. Yu, “Towards millisecond IGP congergence,” November 2000, internet draft, draft-alaettinoglu-ISIS-convergence-00.ps, work in progress.

[6] P. Pan, G. Swallow, and A. Atlas, “Fast Reroute Extensions to RSVP-TE for LSP Tunnels,” March 2004, internet draft, draft-ietf-mpls-rsvp-lsp-fastreroute-05.txt, work in progress.

[7] N. Shen and P. Pan, “Nexthop Fast ReRoute for IP and MPLS,” December 2003, internet draft, draft-shen-nhop-fastreroute-00.txt, work in progress.

[8] M. Shand, “”ip fast reroute framework”,” June 2004, internet draft, draft-ietf-rtgwg-ipfrr-framework-01.txt.

[9] A. Atlas, R. Torvi, G. Choudhury, C. Martin, B. Imhoff, and D. Fedyk, “IP/LDP local protection,” February 2004, internet draft, draft-atlas-ip-local-protect-00.txt, work in progress.

[10] S. Bryant, C. Filsfils, S. Previdi, and M. Shand, “IP Fast Reroute using tunnels,” May 2004, internet draft, draft-bryant-ipfrr-tunnels-00.txt, work in progress.

[11] U. Hengartner, S. Moon, R. Mortier, and C. Diot, “Detection and analysis of routing loops in packet traces,” in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM Press, 2002, pp. 107–112.

[12] S. Spadaro, J. Sol-Pareta, D. Careglio, K. Wajda, and A. Szymanski, “Positioning of the RPR standard in contemporary operator environments,” *IEEE Network Magazine*, vol. 18, March-April 2004.

[13] J. Vasseur and S. Previdi, “Definition of an IS-IS Link Attribute sub-TLV ,” February 2004, internet draft, draft-vasseur-isis-link-attr-00.txt, work in progress.

[14] N. Dubois, B. Fondeviole, and N. Michel, “Fast convergence project,” January 2004, presented at RIPE47, http://www.ripe.net/ripe/meetings/ripe-47/presentations/ripe47-routing-fcp.pdf.

[15] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, “Characterization of failures in an IP backbone,” in *IEEE Infocom2004*, Hong Kong, March 2004.

[16] P. Pongpaibool, R. Doverspike, M. Roughan, and J. Gottlieb, “Handling IP traffic surges via optical layer reconfiguration,” *Optical Fiber Communication*, 2002.

[17] N. Shen and H. Smit, “Calculating IGP routes over Traffic Engineering tunnels,” December 2003, internet draft, draft-hsmit-mpls-igp-spf-01.txt, work in progress.

[18] B. Fortz, J. Rexford, and M. Thorup, “Traffic engineering with traditional IP routing protocols,” *IEEE Communications Magazine*, October 2002.

[19] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, “Netscope: Traffic engineering for IP networks,” *IEEE Network Magazine*, March 2000.

[20] A. Shaikh, R. Dube, and A. Varma, “Avoiding Instability during Graceful Shutdown of OSPF,” in *Proc. IEEE INFOCOM*, June 2002.

[21] M. Shand and L. Ginsberg, “”restart signaling for IS-IS”,” January 2004, internet draft, draft-ietf-isis-restart-05.txt, work in progress.

[22] J. Moy, P. Pillay-Esnault, and A. Lindem, “ Hitless OSPF Restart,” October 2002, internet draft, draft-ietf-ospf-hitless-restart-03.txt, work in progress.

[23] J. J. Garcia-Luna-Aceves, “A unified approach to loop-free routing using distance vectors or link states,” *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 212–223, 1989.

[24] Z. Ali, J. Vasseur, and A. Zamfir, “Graceful Shutdown in MPLS Traffic Engineering Networks ,” June 2004, internet draft, draft-ali-ccamp-mpls-graceful-shutdown-00.txt, work in progress.

[25] J. Parker, D. McPherson, and C. Alaettinoglu, “Short Adjacency Hold Times in IS-IS,” July 2001, internet draft, draft-parker-short-isis-hold-times-01.txt, work in progress.

[26] G. Schollmeier, J. Charzinski, A. Kirstdter, C. Reichert, K. Schrodi, Y. Glickman, and C. Winkler, “Improving the resilience in IP networks,” in *High performance switching and routing (HPSR'03)*, Torino, June 2003.

[27] P. Narvez, K.-Y. Siu, and H.-Y. Tzeng, “Local restoration algorithms for link-state routing protocols,” in *Proceedings of the 1999 IEEE International Conference on Computer Communications and Networks*, 1999.

[28] A. Zinin, “Analysis and Minimization of Microloops in Link-state Routing Protocols,” October 2004, internet draft, draft-zinin-microloop-analysis-00.txt, work in progress.