Routing Area Working Group
Internet-Draft
Intended status: Informational
Expires: January 5, 2012

A. Atlas, Ed.
M. Konstantynowicz
Juniper Networks
G. Enyedi
A. Csaszar
Ericsson
R. White
M. Shand
Cisco Systems
July 4, 2011

An Architecture for IP/LDP Fast-Reroute Using Maximally Redundant Trees

draft-atlas-rtgwg-mrt-frr-architecture-00

Abstract

As IP and LDP Fast-Reroute are increasingly deployed, the coverage limitations of Loop-Free Alternates are seen as a problem that requires a straightforward and consistent solution for IP and LDP, for unicast and multicast. This draft describes an architecture based on redundant backup trees where a single failure can cut a point-of-local-repair from the destination only on one of the pair of redundant trees.

One innovative algorithm to compute such topologies is maximally disjoint backup trees. Each router can compute its next-hops for each pair of maximally disjoint trees rooted at each node in the IGP area with computational complexity similar to that required by Dijkstra.

The additional state, address and computation requirements are believed to be significantly less than the Not-Via architecture requires.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference

material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2012.

Table of Contents

1. Introduction

There is still work required to completely provide IP and LDP Fast-Reroute[RFC5714] for unicast and multicast traffic. This draft proposes an architecture to provide 100% coverage.

Loop-free alternates (LFAs)[RFC5286] provide a useful mechanism for link and node protection but getting complete coverage is quite hard. [LFARevisited] defines sufficient conditions to determine if a network provides link-protecting LFAs and also proves that augmenting a network to provide better coverage is NP-hard. [I-D.ietf-rtgwg-lfa-applicability] discusses the applicability of LFA to different topologies with a focus on common PoP architectures.

While Not-Via [I-D.ietf-rtgwg-ipfrr-notvia-addresses] is defined as an architecture, in practice, it has proved too complicated and stateful to spark substantial interest in implementation or deployment. Academic implementations [LightweightNotVia] exist and have found the address management complexity high (but no standardization has been done to reduce this).

A different approach is needed and that is what is described here. It is based on the idea of using disjoint backup topologies as realized by Maximally Redundant Trees (described in [LightweightNotVia]); the general architecture could also apply to future improved redundant tree algorithms.

1.1. Goals for Extending IP Fast-Reroute coverage beyond LFA

Any scheme proposed for extending IPFRR network topology coverage beyond LFA, apart from attaining basic IPFRR properties, should also aim to achieve the following usability goals:

- ensure maximum physically feasible link and node disjointness regardless of topology,
- automatically compute backup next-hops based on the topology information distributed by link-state IGP,
- do not require any signaling in the case of failure and use pre-programmed backup next-hops for forwarding,
- introduce minimal amount of additional addressing and state on routers,
- enable gradual introduction of the new scheme and backward compatibility,

- and do not impose requirements for external computation.

2. Maximally Redundant Trees (MRT)

In the last few years, there's been substantial research on how to compute and use redundant trees. Redundant trees are directed spanning trees that provide disjoint paths towards their common root. These redundant trees only exist and provide link protection if the network is 2-edge-connected and node protection if the network is 2-vertex-connected. Such connectiveness may not be the case in real networks, either due to architecture or due to a previous failure. The work on maximally redundant trees has added three useful pieces that make them ready for use in a real network.

- Computable when network isn't 2-edge or 2-vertex connected: The maximally redundant trees are computed so that only the cut-edges or cut-vertices are shared between the multiple trees.

- Algorithm is based on a common network topology database. No messaging as has been suggested in other work is necessary.

- An algorithm [MRTLinear] is given that allows a router to compute its next-hops on each pair of maximally redundant trees to each node in the network in O(e) time - or O(e + n log n), if Dijkstra is used instead of BFS.

There is, of course, significantly more in the literature related to redundant trees and even fast-reroute, but the formulation of the Maximally Redundant Trees (MRT) algorithm makes it very well suited to use in routers.

A known disadvantage of MRT, and redundant trees in general, is that the trees do not necessarily provide shortest detour paths. The use of SPF in tree-building and some heuristics can improve this, but the length of the alternate paths is topology-dependent. Providing shortest detour paths would require failure-specific detour paths such as in [I-D.ietf-rtgwg-ipfrr-notvia-addresses], but the state-reduction advantage of MRT lies in the detour being established per destination (root) instead of per destination AND per failure.

A simple but not optimal way of computing maximally redundant trees is described in Appendix A.

2.1. Redundant Trees Overview

In graph theory, a pair of maximally redundant trees are a pair of directed spanning trees of an directed graph with a common root node

(this root can be any node of the graph), such that the two paths along the two trees to the root from any other node are as edge-disjoint and as vertex-disjoint as it is possible. It is known that such trees can be found in any connected networks for any selected root. A pair of trees for the graph depicted in Figure 1 is shown in Figure 2 considering "r" as the root.

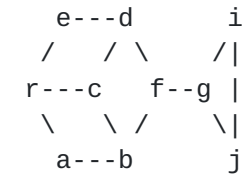

Figure 1: A non-2-connected network

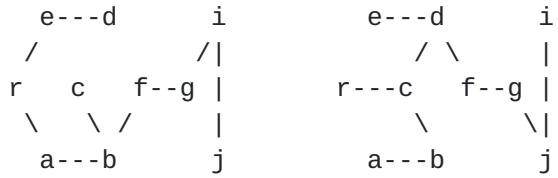

Figure 2: The two maximally redundant trees rooted at node "r".

The two paths along the two trees to a given root of a 2-connected graph are node-disjoint, while in any non-2-connected graph, only the cut-vertices and cut-edges can be contained by both of the paths. As an example consider the trees depicted in Figure 2. Here, the two paths from e.g. node "b" to "r" are node-disjoint (since there are such two paths), and the two paths from e.g. node "i" to "r" have only node "f" and "g" and link "f-g" in common.

3. Maximally Redundant Trees (MRT) and Fast-Reroute

In normal IGP routing, each router has its shortest-path-tree to all destinations. From the perspective of a particular destination, D, this looks like a reverse SPT (rSPT). To use maximally redundant trees, in addition, each destination D has two maximally redundant trees associated with it; by convention these will be called the red and blue redundant trees.

Redundant trees are practical to maintain redundancy even after a single link or node failure. If a pair of maximally redundant trees is computed rooted at each node, all the nodes remain reachable along one of the trees in the case of a single link or node failure.

When there is a link or node failure affecting the rSPT, each node

will still have a path via one of the redundant trees to reach the destination D. Consider a simple 5-node ring (S--A--B--D--C--S), with traffic sent from sources S and C to destination D. In this case, if the link C->D fails, then C can forward traffic along the blue redundant tree to reach D. Of course, if the link S->C fails, then S can simply use its LFA of A.

In a failure free network, packets are forwarded along the shortest path tree as done today. However, if forwarding the packet fails at a node, the router detecting the failure locally reroutes the packet along the blue MRT. If forwarding the packet along the blue MRT fails again, the packet is forwarded along the red MRT. If there is only a single link or node failure, the packet must get to the root along either of the trees. Therefore if forwarding along the red MRT fails, either multiple failures occurred, or the single failure split the network into two, and packets must be dropped.

The above logic gives the following basic rules for unicast use of maximally redundant trees and fast-reroute when failure of the primary is detected are:

1. If there is an node-protecting LFA, use it.

2. Otherwise, if only link-protection is acceptable and there is a link-protecting LFA, use it.

3. Otherwise, if the blue MRT next-hop shouldn't fail with the primary, send traffic along the blue MRT next-hop.

4. Otherwise, if the red MRT next-hop shouldn't fail with the primary, send traffic along the red MRT next-hop.

5. If traffic is received on the blue redundant tree and the appropriate next-hop is not available, then send the traffic on the red redundant tree.

6. If traffic is received on the red redundant tree and the appropriate next-hop is not available, discard it.

3.1. Multi-homed Prefixes

One advantage of LFAs that would be good to preserve is the ability to protect multi-homed prefixes against ABR failure. For instance, if a prefix from the backbone is available via both ABR A and ABR B, if A fails, then the traffic should be redirected to B. This can also be done for backups via MRT.

If there exist multiple multi-homed prefixes that share the same

connectivity and the difference in metrics to those routers, then a single proxy-node can be used to represent the set. In addition to computing the pair of MRTs associated with each router destination D in the area, a pair of MRTs can be computed for each such proxy-node to fully protect against ABR failure.

3.2. Unicast Forwarding with MRT Fast-Reroute

With LFA, there is no need to tunnel unicast traffic, whether IP or LDP. The traffic is simply sent to an alternate. The behavior with MRT Fast-Reroute is different depending upon whether IP or LDP unicast traffic is considered.

Logically, one could use the same IP address or LDP FEC and then also use 2 bits to express the topology to use. The topology options are (00) IGP/SPT, (01) blue MRT, (10) red MRT. Unfortunately, there just aren't 2 spare bits available in the IPv4 or IPv6 header. This has different consequences for IP and LDP because LDP can just add a topology label on top or take 2 spare bits from the label space.

3.2.1. IP Unicast Forwarding

For IP, there is no currently practical alternative except tunneling. The tunnel egress can be the original destination in the area, the next-next-hop, etc.. If the tunnel egress is the original destination router, then the traffic remains on the redundant tree with sub-optimal routing. If the tunnel egress is the next-next-hop, then protection of multi-homed prefixes and node-failure for ABRs is not available.

In either case, each router that supports MRT fast-reroute would need to announce two additional loopback addresses and their associated MRT color. Those addresses are used as destination addresses for MRT-blue and MRT-red IP tunnels respectively. They allow the transit nodes to identify the traffic as being forwarded along either MRT-blue or MRT-red tree topology to reach the tunnel destination. Announcements of these two additional loopback addresses per router with their MRT color requires IGP extensions.

IP packets could be tunneled via LDP. This has the advantage that more routers can do line-rate encapsulation and decapsulation. If tunneled via LDP, naturally one of the LDP unicast forwarding options would need to be used. It would be possible to use just a LDP Topology-Identifier label on top of the IP packet; if done, this would avoid any need to allocate or signal additional IP addreses and is particularly interesting for multi-homed prefixes.

For proxy-nodes associated with one or more multi-homed prefixes, the

problem is harder because there is no router associated with the proxy-node, so its loopbacks can't be known or used. In this case, each router attached to the proxy-node could announce two common IP addresses with their associated MRT colors. This would require configuration as well as the previously mentioned IGP extensions. Similarly, in the LDP case, two additional FEC bindings could be announced.

3.2.1.1. Protocol Extensions and Considerations: OSPF and ISIS

This captures an initial understanding of what may need to be specified.

- Capabilities: Does a router support MRT? Does the router do MRT tunneling with LDP or IP or GRE or...?

- Topology Association: A router needs to advertise a loopback and associate it with an MRT whether blue or red. Additional flexibility for future uses would be good.

- Proxy-nodes for Multi-homed Prefixes: We need a way to advertise common addresses with MRT for multi-homed prefixes' proxy-nodes. Currently, those proxy-nodes aren't named or considered.

- Algorithm-specific Commonalities: In specifying the exact details for a common algorithm, there may be tie-breakers that are better done based on configuration than just using Router ID.

As with LFA, it is expected that OSPF Virtual Links will not be supported.

3.2.2. LDP Unicast Forwarding

For LDP, it is very desirable to avoid tunneling because, for at least node protection, this requires knowledge of remote LDP label mappings. There are two different mechanisms that could be used.

1. Create Topology-Identification Labels: Use the label-stacking ability of MPLS and specify only two additional labels - one for each associated MRT color - by a new FEC type. When sending a packet onto an MTR, first swap the LDP label and then push the topology-identification label for that MTR color. When receiving a packet with a topology-identification label, pop it and use it to guide the next-hop selection in combination with the next label in the stack; then swap the remaining label, if appropriate, and push the topology-identification label for the next-hop. This has minimal usage of additional labels, memory and LDP communication. It does increase the size of packets and

the complexity of the required label operations and look-ups.

2. Encode Topology in Labels: In addition to sending a single label for a FEC, a router would provide two additional labels with their associated MRT colors. This is simple, but reduces the label space for other uses. It also increases the memory to store the labels and the communication required by LDP.

Note that with LDP unicast forwarding, regardless of whether topology-identification label or encoding topology in label is used, no additional loopbacks per router are required as are required in the IP unicast forwarding case. This is because LDP labels are used on a hop-by-hop basis to identify MRT-blue and MRT-red forwarding trees.

3.2.2.1. Protocol Extensions and Considerations: LDP

This captures an initial understanding of what may need to be specified.

1. Topology-Identification Labels: Define a new FEC type that describes the topology for MRT and the associated MRT color.

2. Specify Topology in Label: When sending a Label Mapping, have the ability to send a Label TLV and multiple Topology-Label TLVs. The Topology-Label TLV would specify MRT and the associated MRT color.

3.3. Multicast Forwarding with MRT Fast-Reroute

There are several basic issues with doing Fast-Reroute for multicast traffic, whether the alternates used are LFA or MRT. They are given below:

1. The Point-of-Local-Repair (PLR) does not know the set of next-next-hops in the multicast tree.

2. For mLDP, the PLR does not know the appropriate labels to use for the next-next-hops in the multicast tree.

3. The Merge Point (MP) does not know upon what interface to expect backup traffic. For LFAs, this is a particular issue since the LFA selected by a PLR is known only to that PLR.

There are also issues about how to manage traffic.

a. When should the PLR stop sending traffic on the alternate? Based upon a configurable time-out is the most general answer. For PIM, an explicit Withdraw or Backup Withdraw could be sent, but they could always be lost and are not reliable. For mLDP, even if the PLR is known, for node-protection, there is no targeted LDP session and so no way for the MP to explicitly withdraw the label that was implicitly learned by the PLR.

b. Can anything be done about traffic missed due to different latencies along new primary and alternate/old primary trees? If a router is willing and able to examine traffic from both the alternate and new primary, perhaps the full set of packets could be assured. This requires more investigation, but such an option would be at most optional. A router could also continue to accept traffic from both the old alternate and the new primary for a period longer than the expected difference in latency, but this comes with a possible doubling of traffic during that period.

3.3.1. **Tunneled?** Yes

The disadvantages of tunneling unicast traffic do not fully translate to those for multicast. With MRT fast-reroute, IP unicast traffic is tunneled. With mLDP, in the suggested extensions (later), along with learning the next-next-hops on the multicast tree, the associated labels can be learned so there is no need for targeted sessions.

If multicast traffic is not tunneled along the alternates, then there is the question of what happens when unencapsulated backup traffic intersects the normal multicast tree before reaching the MPs. Resolving this is likely to introduce significant complexity and state into the routers, with the only gain being the avoidance of a tunnel.

Therefore, tunneling for IP and mLDP multicast traffic along the selected alternates is required. This means all replication is done by the PLR.

3.3.2. **PIM Forwarding**

For node-protection, the merge points would be the next-next-hops in the tree. For a PLR to learn them, additional PIM Join Attributes [I-D.ietf-pim-mtid] need to be defined to specify the set of next-hops from which the sending node has received Joins. For link-protection, of course a PLR knows the routers that have sent it Joins.

An MP must know the interface that alternate traffic should be

accepted from. To do this, a new Upstream Backup Join would be added to PIM. This Upstream Backup Join would be sent by the PLR to MPs. If the PLR has selected an LFA for a MP, then the PLR tunnels the Upstream Backup Join to the MP via the LFA. If the PLR will use MRT, then the PLR must send two Upstream Backup Joins - one transmited via the blue MRT and one transmited via the red MRT; these will also be tunneled via LDP or IP as is configured in the network.

The Upstream Backup Join will specify the PLR, the MP, the (S, G), and the alternate details (e.g. LFA with neighbor address, blue MRT, red MRT). If desired, the alternate topology could be used to verify the incoming interface appropriately via a tree-appropriate RPF check. Upon receiving the Upstream Backup Join, the MP will accept specified multicast traffic from the LFA backup neighbour, blue or red MRT respectively.

3.3.2.1. Protocol Extensions and Considerations: PIM

This captures an initial understanding of what may need to be specified. This is focusing on PIM Sparse mode.

- Capabilities: New Hello Option Capabilities to indicate the ability to understand the new Join Attributes and Upstream Backup Join.

- Next-Hops: Need a new Join Attribute[I-D.ietf-pim-mtid] to send the next-hops to the PLR. This list could be updated and sent upstream every time it changes.

- Upstream Backup Join

3.3.3. mLDP Forwarding

As in PIM, in mLDP[I-D.ietf-mpls-ldp-p2mp] a mechanism must be added so that the PLR can learn the next-next-hops. The PLR also needs to learn the associated label-bindings. This can be done via a new P2MP Child Data Object. This object would include the primary loopback of an LSR that has provided labels for the FEC to the sending LSR along with the label specified. Multiple P2MP Child Data Objects could be included in a P2MP Label Mapping; only those specified in the most recent P2MP Label Mapping should be stored and used.

This will provide the PLR with the MPs and their associated labels. The MPs will accept traffic received with that label from any interface, so no signaling is required before the alternates are used.

Traffic sent out each alternate will be tunneled with a destination

of the MP.

3.3.4. Live-Live Multicast

In MoFRR [I-D.karan-mofrr], the idea of joining both a primary and a secondary tree is introduced with the requirement that the primary and secondary trees be link and node disjoint. This works well for networks where there are dual-planes, as explained in [I-D.karan-mofrr]. For other networks, it may still be desirable to have two disjoint multicast trees and allow a receiver to join both and make its own decision about what to do.

MRT allows this, but would require minor extensions to PIM or mLDP. The pair of maximally redundant trees is rooted at the multicast group source S. IF asymmetric link costs aren't a concern, then the same set of next-hops (previous-hops in this case) could be used as is used for MRT fast-reroute. The extension to PIM would be to specify which MRT tree is being joined - so instead of specifying join(S,G), it would specify join(S,G, MRT red) or join(S,G, MRT blue). Similarly, a new P2MP FEC with Tree Identifier Element would need to be defined; it would include the topology to be used which could be IGP, MRT red, or MRT blue.

The receiver would still need to detect failures and handle traffic discarding as is specified in [I-D.karan-mofrr].

3.4. MRT Algorithm Open Issues

The MTR algorithm as given in [MRTLinear] handles most of the issues that occur in real networks, but there are a few aspects that need to be considered and factored in.

- Broadcast Interfaces: While a broadcast interface can simply be represented as a pseudo-node in the graph, the rules for handling it given that there is a requirement to provide node-protection as well as link-protection need to be defined.

- Parallel Links: It is quite common to have parallel links in a real network. In the case where node-disjoint paths are possible, the parallel links just introduce the possibility of multiple next-hops along the MTR. In the case where node-disjoint paths aren't possible, having the ability to use parallel links is important.

- Improved Paths in MTRs: In the tree-building, it should be straightforward to use SPF instead of BFS. There are additional heuristics in the Finding Multiple Maximally Redundant Trees in Linear Time paper that should be evaluated for realistic benefits.

- Asymmetric Link Costs: The tree-building must consider that links may have asymmetric costs.

- Administratively Unavailable Links and Nodes: With the standard need to avoid a flag day, not all routers may participate in MTR and those that aren't must not be used in an MTR. Additionally, links can be overloaded or administratively specified as not available just as is considered with LFA. Such links may not be used in the MTR.

- ECMP: Consider the ability to use multiple equal-cost paths in building the MRT to get additional capacity along the MRT.

3.4.1. SRLG Protection

As shown in Appendix A, a straightforward way to build two redundant trees involves taking a link from a ready node to a non-ready node to provide one path and then determining the shortest-path back to a ready node that doesn't include that ready node. Creating similar redundancy with arbitrarily placed Shared-Risk Link Groups is still a challenging open problem.

3.4.2. Common Computation

In the MTR algorithm, there are some places where decisions are made as to which link to use next, which neighbor to consider, etc. The exact rules to follow and a detailed algorithm with example need to be provided. Ideally, there would be reference pseudo-code.

4. Acknowledgements

The authors would like to thank Hannes Gredler, Robert Kebler, Ted Qian, Kishore Tiruveedhula, Santosh Esale, Nitin Bahadur, Harish Sitaraman and Raveendra Torvi for their suggestions and review.

5. IANA Considerations

This doument includes no request to IANA.

6. Security Considerations

This architecture is not currently believed to introduce new security concerns.

7. References

7.1. Normative References

[I-D.ietf-mpls-ldp-p2mp]
Minei, I., Wijnands, I., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", draft-ietf-mpls-ldp-p2mp-14 (work in progress), June 2011.

[I-D.ietf-pim-mtid]
Cai, Y. and H. Ou, "PIM Multi-Topology ID (MT-ID) Join Attribute", draft-ietf-pim-mtid-08 (work in progress), June 2011.

[I-D.karan-mofrr]
Karan, A., Filsfils, C., Farinacci, D., Decraene, B., Leymann, N., and T. Telkamp, "Multicast only Fast Re-Route", draft-karan-mofrr-01 (work in progress), March 2011.

[RFC5286] Atlas, A. and A. Zinin, "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, September 2008.

[RFC5714] Shand, M. and S. Bryant, "IP Fast Reroute Framework", RFC 5714, January 2010.

7.2. Informative References

[I-D.ietf-rtgwg-ipfrr-notvia-addresses]
Shand, M., Bryant, S., and S. Previdi, "IP Fast Reroute Using Not-via Addresses", draft-ietf-rtgwg-ipfrr-notvia-addresses-07 (work in progress), April 2011.

[I-D.ietf-rtgwg-lfa-applicability]
Filsfils, C., Francois, P., Shand, M., Decraene, B., Uttaro, J., Leymann, N., and M. Horneffer, "LFA applicability in SP networks", draft-ietf-rtgwg-lfa-applicability-02 (work in progress), May 2011.

[LFARevisited]
Retvari, G., Tapolcai, J., Enyedi, G., and A. Csaszar, "IP Fast ReRoute: Loop Free Alternates Revisited", Proceedings of IEEE INFOCOM , 2011, <http://opti.tmit.bme.hu/~tapolcai/papers/retvari2011lfa_infocom.pdf>.

[LightweightNotVia]
Enyedi, G., Retvari, G., Szilagyi, P., and A. Csaszar, "IP Fast ReRoute: Lightweight Not-Via without Additional Addresses", Proceedings of IEEE INFOCOM , 2009, <http://mycite.omikk.bme.hu/doc/71691.pdf>.

[MRTLinear]
Enyedi, G., Retvari, G., and A. Csaszar, "On Finding Maximally Redundant Trees in Strictly Linear Time", IEEE Symposium on Computers and Comunications (ISCC) , 2009, <http://opti.tmit.bme.hu/~enyedi/ipfrr/distMaxRedTree.pdf>.

[Maintaining Colored Trees]
"Maintaining colored trees for disjoint multipath routing under node failures", IEEE/AC Transactions on Networking vol. 17, no. 1, pp. 346-359, 2009, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.138.6025&rep=rep1&type=pdf>.

Appendix A. Computing Maximally Redundant Trees

This is possible but not optimal way to compute maximally redundant trees. It is included to provide some intuition about how the maximally redundant trees are built.

In Appendix A.1, we describe how to handle a network that is 2-connected. Then that is assumption is relaxed and finally how to handle distributed computation to obtain the same trees is given.

A.1. Simple pair of maximally redundant trees in 2-connected networks

Finding a simple pair of maximally redundant trees in a 2-connected network is straightforward. We call a node "ready", if it was already added to the trees. Initially, the only ready node is the common root (node r in the sequel).

When we have at least one node x in the network, which is not ready, find two node-disjoint paths from x either to r or to two distinct ready nodes. Since the network is 2-connected, there are always two node-disjoint paths from x to r. It is possible that one or both of these paths reaches another ready node sooner than r, in which case we have the two node-disjoint paths to distinct nodes. Combining the directed links of these paths makes up an *ear*.

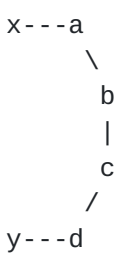

Figure 3: An *ear* connected to node x and y (x and y are ready).

Let x and y be the two ready endpoints of an ear, and first suppose that they are different nodes and none of them is r. Note that both x and y are in the two trees (since they are "ready") and if x is an ancestor of y in the first tree (x is on the path from y to r), then x cannot be the ancestor of y along the second tree at the same time. Thus, it is safe to connect the nodes of the freshly found ear to x in the first tree and to y in the second tree, if either x is an ancestor of y in the first tree, or y is an ancestor of x in the second tree. Considering the example in Figure 3, this means that links d-c-b-a-x should be added to the first tree, and a-b-c-d-y should be added to the second one.

In the case, when either x=r or y=r or when neither x is an ancestor of y nor y is an ancestor of x in any of the trees, the endpoints are not firmly bound to one of the trees, it is only important to put the links to one endpoint in one of the trees and put the links towards the other endpoint to the other tree. In our example this means that either d-c-b-a-x or a-b-c-d-y could be added to the first tree. Naturally, then the other endpoint must be selected for the second tree.

In some cases, we need to construct such (maximally) redundant trees, where there is only one edge entering to the root on one of the trees. This makes the root a leaf in that tree. To achieve this, we can add the ear to the second tree through r only if both endpoints are r. Moreover, we need to select an ear with different endpoints when it is possible (it is always possible except for the first ear, if the network is 2-connected).

Finding an ear is relatively simple and can be done in different ways. Probably the simplest way is to find a ready node q (q is not the root) with a non-ready neighbor w, (virtually) remove q from the topology, and to find a path from w to r; since the network is 2-connected, such a path either reaches r, or reach another ready node. Moreover, when only r is ready such a node q does not exist, so we select one of r's neighbors as w, and remove not r but the link between them.

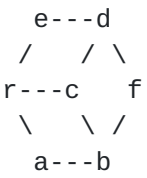

Figure 4: A 2-connected network

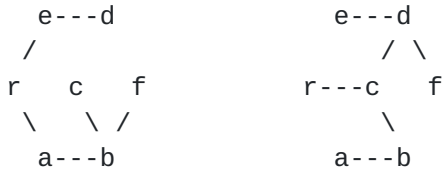

Figure 5: The two maximally redundant trees found in the network depicted in previous figure.

Now, a simple example is in order. Consider the network depicted in Figure 4, and suppose that the common root is node r. We have only r in the trees, so we select one of its neighbors, let it be a, remove the link between them, and select a path (let it be the shortest one) from a to r; this path is a-b-c-r, so the ear is r-a-b-c-r. Since both endpoints of the ear are r, selecting the right tree is not important, e.g., we can add c-b-a-r to the first tree, and a-b-c-r to the second one Figure 5. This way, r, a, b and c form the set of "ready" nodes. From the ready set, c and d are not the root and have non-ready neighbors. Let us select, e.g., c. The shortest path from d to r when c is removed is d-e-r, so we have ear c-d-e-r, we add d-e-r to the first tree and e-d-c to the second one (recall that we do not want to create a new neighbor for r in the second tree). Finally, the last non-ready node is f, and the ear is b-f-d. Since neither is b an ancestor of d nor is d an ancestor of b in any of the trees, we can connect f to the trees in both ways. E.g., add f-b to the first tree, and f-d to the second one.

A.2. Non-2-connected networks

When, however, the network is not 2-connected, it is not always possible to find a pair of node-disjoint paths from any node x to root r, which makes our previous algorithm unable to find the trees. However, while the network is connected, it is made up by 2-connected components bordered by "cut-vertices" (naturally, some of these components may contain only one node). A node is a cut-vertex, if removing that node splits the network into two.

A simple algorithm to find the components and the cut-vertices can be to (virtually) remove each vertex one by one, and check connectivity

with BFS or DFS. Moreover, nodes a and b are in the same 2-connected component, if a remains reachable from b after removing any single node. Note that linear time algorithms do exist that find both the 2-connected components and the cut-vertices.

Now, we can build up redundant trees in each component. In components containing r, the root of such trees must be r. Otherwise, in the remaining components the root must be the last node in the component along a path to the root. Recall, that this must be a cut-vertex, so it is the same for each path emanating from that component.

At this point, we are ready, if there is no cut-edge in the network. However, if some 2-connected components are connected by a cut-edge, we must add that edge to both of the trees.

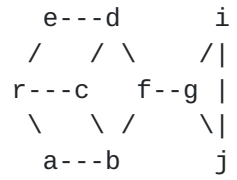

Figure 6: Non-2-connected network

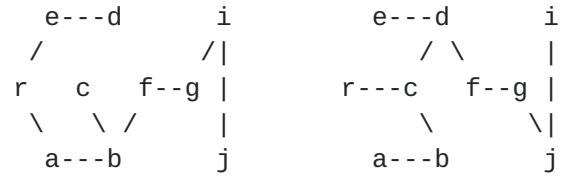

Figure 7: The two maximally redundant trees found in the network depicted previously.

As an example consider the network depicted in Figure 6. Observe that now we have two 2-connected components, one contains r, a, b, c, d, e, f and the other contains g, i, j. Moreover, these components have no common node, they are connected with a cut-edge.

Finding the trees in the component containing r is already described; these trees are the same as previously. Moreover, the other component is a cycle, so it will be covered by a single ear. Finally we must add link f-g to both of the trees, to get the trees depicted in Figure 7.

A.3. Finding maximally redundant trees in distributed environment

If we need to compute exactly the same maximally redundant trees at each of the routers, consistency needs to be ensured by tie-breaking mechanisms. Observe that the previous algorithm has multiple choices when it selects how to connect nodes to the trees when only r is ready, how to select ready node q and non-ready node w for a later ear and when neither of the endpoints is an ancestor of the other one.

All of the previous decision points can be handled in a consistent fashion. E.g., the first ear should be connected in such a way, that the neighbor of r with the lowest ID must be directly connected to r in the first tree. Moreover, later we should choose ready router with non-ready neighbor as q and its non-ready neighbor with the lowest ID as w. Finally, when neither of the endpoint is an ancestor of the other one, connect the ear to the endpoint with the lower ID in the first tree.

Authors' Addresses

Alia Atlas (editor)
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: akatlas@juniper.net

Maciek Konstantynowicz
Juniper Networks

Email: maciek@juniper.net

Gabor Sandor Enyedi
Ericsson
Irinyi utca 4-10
Budapest 1117
Hungary

Email: Gabor.Sandor.Enyedi@ericsson.com

Andras Csaszar
Ericsson
Irinyi J ut 4-10
Budapest 1117
Hungary

Email: Andras.Csaszar@ericsson.com

Russ White
Cisco Systems

Email: russwh@cisco.com

Mike Shand
Cisco Systems

Email: mshand@cisco.com