

RTGWG
Internet-Draft
Intended status: Informational
Expires: July 26, 2015

P. Thubert, Ed.
Cisco
G. Enyedi
Ericsson
S. Ramasubramanian
UA
January 22, 2015

ARC vs. MRT
draft-thubert-rtgwg-arc-vs-mrt-01

Abstract

This draft compares the capabilities offered by the Available Routing Construct (ARC) and the Maximally Redundant Trees (MRT) techniques in order to support applicability statements.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [RFC2119].

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 July 26, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

Internet-Draft

ARCVsMRT

January 2015

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Important Notice	2
2.	Introduction	3
3.	Terminology	5
4.	Reference Topology	5
5.	Normal Operation	6
6.	One breakage	7
7.	Second breakage	9
8.	Summary of differences	10
8.1.	Differences between ARCs and MRT	11
8.2.	Similarities between ARC and MRT	14
9.	ARC specific properties	14
9.1.	Multiple related breakages	14
9.2.	Multiple Destination and Load Balancing	15
9.3.	Hierarchical Routing	16
9.4.	Recovery from Node Failures	17
9.5.	Applied to bicasting (livelive)	17
10.	Manageability	17
11.	IANA Considerations	17
12.	Security Considerations	17
13.	Acknowledgements	18
14.	References	18
14.1.	Normative References	18
14.2.	Informative References	18
14.3.	References	19
	Authors' Addresses	19

[1.](#) Important Notice

This document compares MRT and ARCs based on [\[I-D.enyedi-rtgwg-mrt-frr-algorithm\]](#), [\[I-D.ietf-rtgwg-mrt-frr-architecture\]](#), [\[I-D.thubert-rtgwg-arc\]](#) and

[\[I-D.thubert-rtgwg-arc-bicast\]](#) as they were on 10/12/2012. Both solutions may have changed or may change in the future, which would make this document somewhat outdated.

[2.](#) Introduction

Traditional routing and forwarding uses the concept of path as the basic routing paradigm to get a packet from a source to a destination. A path is represented as an ordered sequence of nodes. This paradigm is greedy in that it requires that the next node represent a progress towards the destination. Greediness is maximized by Shortest Path First (SPF) techniques which optimize for a cost that relates to the amount of forwarding effort and latency, at the expense of network capacity and reliability. In particular, a path is broken as soon as a single link or a single node fails, and getting around a breakage can require path recomputation, network reconvergence, and incur delays and/or microloops till service is restored.

Maximally Redundant Trees [\[I-D.ietf-rtgwg-mrt-frr-architecture\]](#) [\[I-D.enyedi-rtgwg-mrt-frr-algorithm\]](#) (MRT), on the other hand, favors reliability over shorter path. MRT provides two spanning trees rooted at a destination, referred to as red and blue trees. The trees have the property that the red and blue paths from any node to the destination (root of the tree) are maximally-disjoint. The red and blue paths may not be the shortest path. Thus, a node may employ shortest path tree for forwarding under normal circumstances. Thus, every node may have up to three FIB entries for a destination, one for red tree, one for blue tree, and a third for the shortest path tree. Under normal circumstances, the idea is to forward the packets along the shortest path until the packet encounters a failure. If the failed link is red (blue), then the packet is forwarded along the blue (red) path. Once a packet is rerouted along the red or blue tree, it continues to be forwarded along the chosen tree until it reaches the destination. If the packet encounters another failure, then it is dropped. Thus, MRT is guaranteed to recovery from only one failure.

The maximally redundant trees are constructed using a technique called path augmentation. The approach works as follows: (1) For a

given destination node, a cycle traversing the node and two other nodes are computed. (2) One direction of the cycle is termed as red, while the other direction is termed as blue. (3) A cycle or path is computed that starts and ends at a node that has been already added to the trees. On this cycle/path, one direction is treated as red and the other direction is treated as blue. The direction of the red and blue trees is computed based on a partial order, which ensures that the red and blue assignments would result in disjoint paths along the red and blue trees. The last step is repeated until all the nodes in the network are considered. For a more detailed description of the construction, refer to [\[Jayavelu.2009.Maintaining\]](#).

MRT constructs trees that are maximally-disjoint. Thus, if the network provided two node-disjoint paths between a node and destination, MRT would provide node-disjoint paths as well. If the network does not provide node-disjoint paths, but provides link-disjoint paths between a node and destination, then MRT would provide link-disjoint paths as well.

Available Routing Construct [\[I-D.thubert-rtgwg-arc\]](#) (ARC) defines a routing construct made of a bidirectional sequence of nodes and links with 2 outgoing edges, so that, upon a single breakage, each lively node in along ARC can still reach one of the outgoing edges. The bi-directional sequence of nodes and links are similar to the paths/cycles used in the path augmentation technique in constructing MRT. The routing graph to reach a certain destination is expressed as a cascade of ARCs, each ARC providing its own independent domain of fault isolation and recovery. Unicast traffic may enter an ARC via any node but it may only leave the ARC through one of its two edges. One node along the ARC is designated as the cursor. In normal unicast operations, the traffic inside an ARC flows away from the cursor towards an edge. Upon a failure, packets may bounce on the breakage point and flow the other way along the ARC to take the other exit. [\[I-D.thubert-rtgwg-arc\]](#) calculates ARCs within the shortest path computation, and both sides of an ARC as delineated by the cursor actually represent the shortest path to the destination.

Applying Available Routing Constructs to multicasting
[\[I-D.thubert-rtgwg-arc-multicast\]](#) provides additional information on how an ARC fabric can be used for multicasting applications such as livelive. In this class of applications, at least two disjoint paths

must be established between a source and a point of consumption. Ideally, the source is diverse, for instance a distributed Content Delivery Network (CDN), but it does not have to be that way. An ARC fabric can be set up for an Omega that is a multiple destination in the exact same fashion as if Omega is a unique node.

[[I-D.thubert-rtgwg-arc-bicast](#)] then shows how an ARC set can be painted in two colors, an half of each ARC with one color and the second half of the other color in such a fashion that in most cases, an edge of a certain color ends in the half of the next ARC that is of the same color. A colored packet has the constraint to exit each ARC along its path via the edge of its own color. If the edge and the next half ARC are of the same color, the packet follows the shortest path towards Omega. It results that ARC bicasting aggressively pursues the goal that both colored paths are close to shortest even though disjoint. The quality of that closeness depends on the way the ARC set was painted, which itself depends, in the bicasting algorithm, on the starting points in Omega. A centralized computation can probably improve the result that the simple technique in [[I-D.thubert-rtgwg-arc-bicast](#)] obtains.

This draft compares the properties of the ARC and MRT techniques in a number of example situations that are crafted to highlight their particular behaviors with an educational purpose in mind, as opposed to represent the most classical real-world cases. It must be noted that though the draft focuses on differences, ARCs and MRT have a lot in common, in particular both provide maximally redundant solutions, and represent roughly the same amount of labels and states in the packets.

[3.](#) Terminology

The draft uses terminology from [[I-D.enyedi-rtgwg-mrt-frr-algorithm](#)] and [[I-D.thubert-rtgwg-arc](#)].

[4.](#) Reference Topology

This draft uses a simple reference topology, made of eight nodes A to H and ten links with costs of either 1 or 2 as represented below:

```
A--- 2 ---E      A      E
|              |      v
```

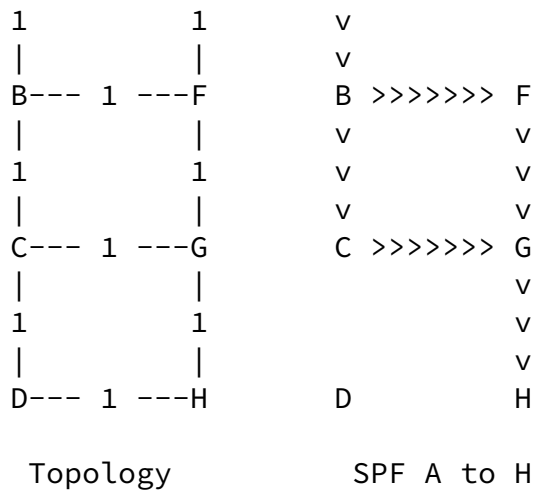


Figure 1: Reference Topology

In the following examples we will be interested in traffic flowing from A to H. The shortest path for that flow is via B, then through an equal cost multipath via F or C, and then G to finally reach H.

The oLAF algorithm forms three ARCs and MRT forms two trees:

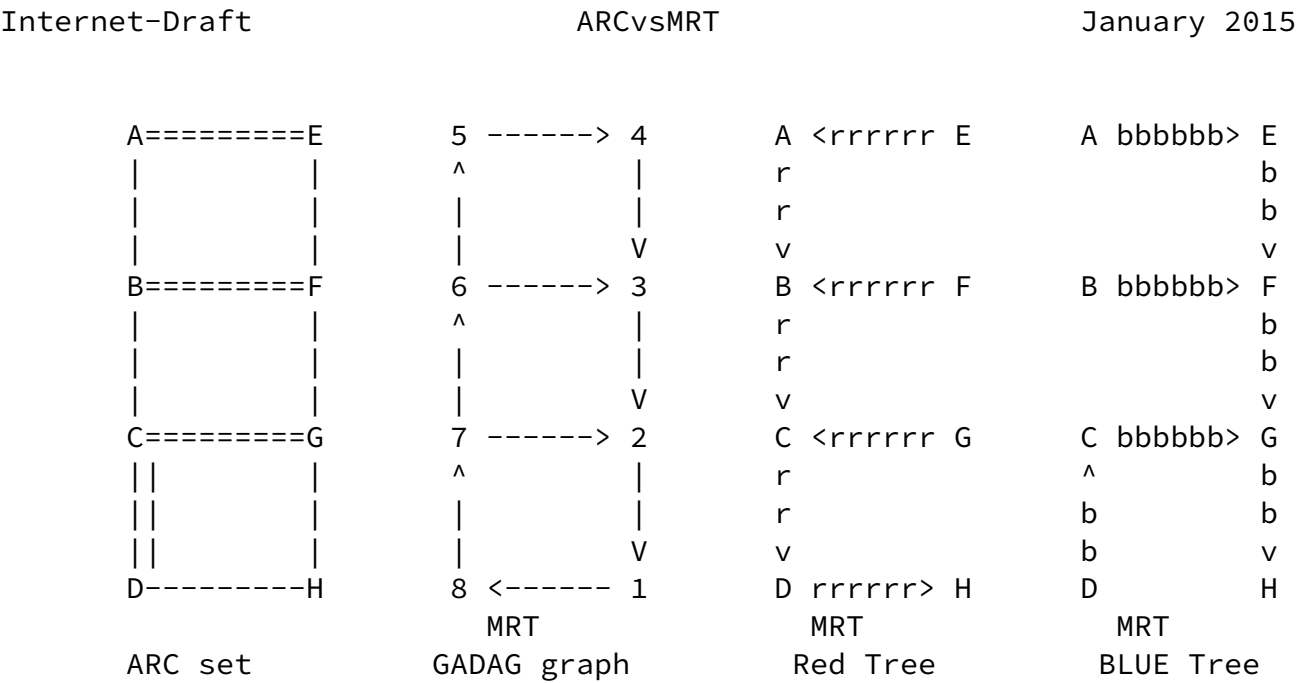
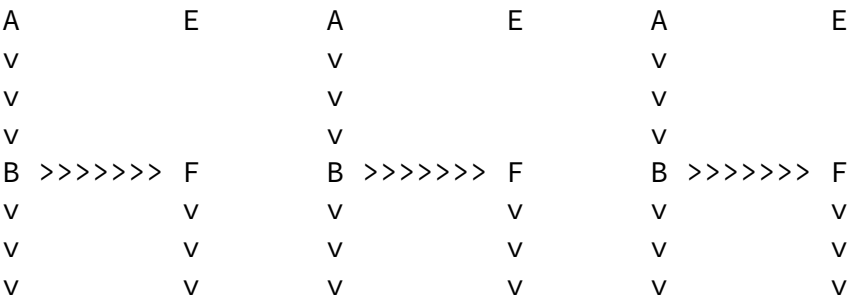


Figure 2: ARC set and MRT Trees

Bidirectional links along the ARCs are represented as doubled lines. The cursors end up on A, B and C for their respective ARCs.

5. Normal Operation

In normal operations MRT trees are not used so the MRT case is identical to the SPF case. For ARCs, the cursors A B and C may send traffic on either side but favor shortest. For A that means sending to B and for C sending to G. B sees an equal cost so it may distribute its traffic.



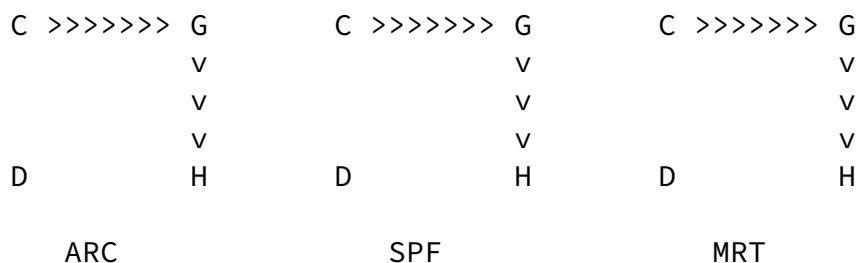


Figure 3: Normal Operation

All in all, the three approaches end up with an identical result.

6. One breakage

Upon a first breakage, SPF cannot make a greedy progress so a packet that follows the broken path is dropped. MRT converts to the packet to blue or red, and then packet is placed in the associated tree till the destination. Because blue and red have to be non-congruent to the end, the detour that MRT generates tends to route the packet away from shortest path. ARCs returns the packet along the current ARC so it exits on the other edge. As soon as the cursor in the current ARC is passed, the packet is again on shortest path.

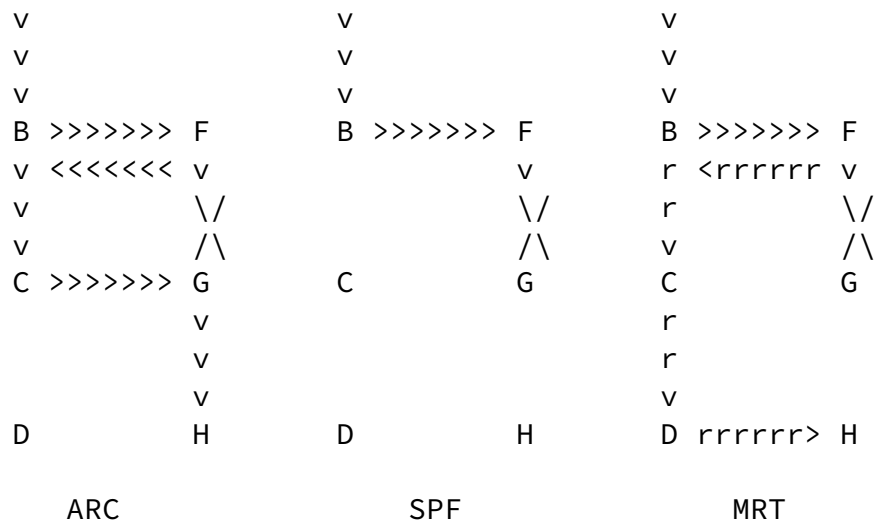


Figure 4: ARC closer to shortest path

Both MRT and ARCs enable fast reroute. MRT will generally incur a larger detour. In the example above, the ARC path after B is shortest, whereas the red path selected by MRT is not.

Because the blue and red paths of MRT are link-disjoint from any node to the destination, the recovery path does not visit a broken link again. ARCs achieve a shorter path by being more greedy. It results that in some situations, a broken node might be visited twice, once from the above, that is from an incoming ARC, and then from the side, that is from inside its own ARC. This is illustrated below though it would take an intermediate node between C and broken G to actually create the undesired bouncing that is discussed here.

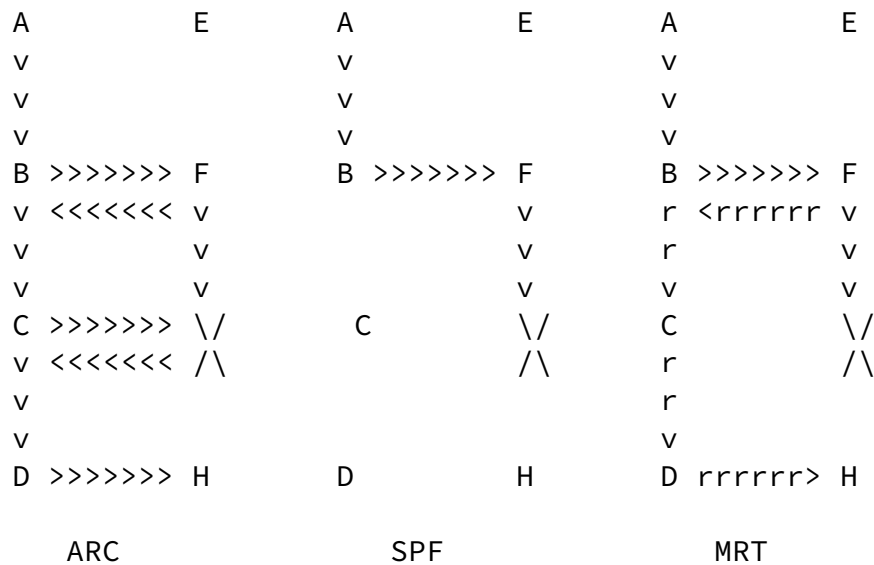


Figure 5: ARCs may revisit broken node

Note: In the particular case where a broken node is visited twice, the end to end path that ARC uses may still be shorter than the blue or red path that MRT computes, though it is not the case in the particular example above.

MRT recovers from node failure similar to that of link failure. Note that the paths provided by MRT are maximally disjoint. Thus at a node, if a link is unavailable (whether due to a link failure or node failure), the recovery path is taken. As the paths are maximally disjoint, MRT guarantees that a failed node would never be visited again.

7. Second breakage

Upon a first breakage, SPF cannot make a greedy progress so a packet that follows the broken path is dropped. Upon a first breakage, MRT selects either blue or red. If that selected path is broken down the road, the packet is dropped. In the data plane, ARCs protect against one breakage per ARC. Once a packet leaves an ARC to cascade in the next, stigmata from a previous breakage are removed. So a break on the next ARC can be resolved as well.

Internet-Draft

ARCvsMRT

January 2015

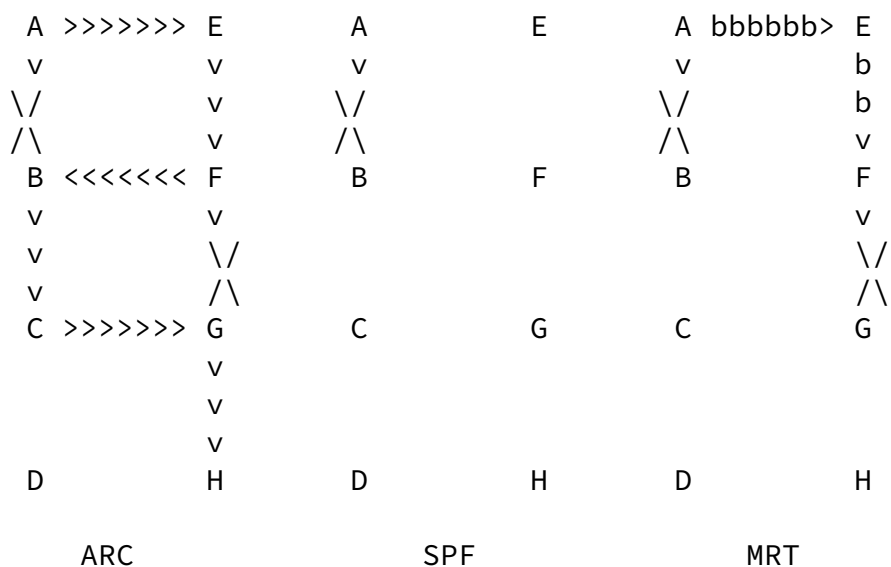


Figure 6: Each ARC its own recovery domain

It can be noted that if C to G,G or G to H is broken, ARCs will find the C, D, H path and thus fix a third breakage as well.

8. Summary of differences

In sumary:

Internet-Draft

ARCVsMRT

January 2015

MRT	ARC
1. Limited complexity - can be even $O(e)$	Complexity inherited from SPF
2. Detour, unrelated to Shortest Path	Short detour then Shortest Path again
3. Smaller chance to avoid unrelated failures	Higher chance to avoid unrelated failures => may address SRLG cases
4. Single failure: reroute at most once	Single failure may incur double reroute
5. Load balancing with traditional ECMP	Non-equal Cost Load Balancing capabilities
6. Unrelated "topologies" exists -> reconfiguration after the failure is simple	Detours are strongly bound to default paths => reconfiguration techniques are difficult
7. Non-Congruent bicasting is not addressed	Shorter Path bicasting with collision avoidance
8. Source-centric computation -> easier to distribute	Destination-centric computation => enables complex destinations (multiple equivalent ARC set terminations)

Figure 7: Comparison summary

[8.1.](#) Differences between ARCs and MRT

In this section, we elaborate on the differences listed above.

1.

ARC is a destination based technology, i.e. it compute a set of ARCs to all the destinations in the network. Constructing one such set needs several shortest path computations. This may cause a minor problem: since detours and default paths are bound, computing the new default paths after a failure may be delayed due to this slower computation.

Thubert, et al.

Expires July 26, 2015

[Page 11]

Internet-Draft

ARCsMRT

January 2015

In contrast, MRT used the traditional shortest paths in a failure-free state, thus detours can be computed separately, when all the shortest paths are up and running. Moreover, although MRT offers multiple ways to construct detours (keep in mind that MRT is only for detours, default paths are computed with SPF), even the slowest one is as fast as computing one set of ARCs for a single destination; the fastest algorithm takes only $O(e)$ time, so it is linear with the number of links in the network.

2.

In MRT, the paths from any node to the destination on the red and blue trees are link-disjoint, while in ARC, it is not. Thus, in MRT, neither the red nor the blue tree is guaranteed to provide shortest path for a node. However, in ARC, packet is forwarded along the shortest path after a short detour. Naturally, when there is no failure, both algorithms use the shortest paths, thus this difference can only be observed after a failure.

3.

MRT uses separated default paths and detours. As a consequence, one needs to have three FIB entries one for shortest path forwarding (default), one for red tree forwarding, and one for blue tree forwarding. When a failure happens, MRT puts the packet onto a

detour, which is not removed until it reaches the destination (egress router, area border router, etc..). Thus, packet is bound to either the red path, or the blue path. When the packet encounters a second failure, the packet is dropped.

Although the MRT draft [[I-D.ietf-rtgwg-mrt-frr-architecture](#)] does not discuss about recovering from multiple failures, the MRT framework allows the network to recover from multiple link failures, as long as no more than one failure is seen on a path/cycle that's used for augmentation. Thus, when a packet is forwarded from one path/cycle to another, the packet can handle one more link failure. This technique has been shown in [[Erlebach.2009.Path-Splicing](#)].

In contrast, ARC partitions the network in delimited protection areas, i.e. it puts packets back to shortest path when they leave the current ARC. Putting packets back to the shortest path as soon as they leave the ARC makes it possible to reroute again at another failure. Moreover, theoretically ARC needs only two labels/addresses per destination, one describing the shortest path, and the other describing the other direction in the ARC. However, this approach would result loops, if there are more than one failure in the same ARC, thus current version of ARC use at least three labels/addresses per destination.

ARC can protect against multiple link failures as long as the only one link failure occurs on an ARC.

Since there is always a reconfiguration after a failure, and FRR is in charge only for a short amount of time while this reconfiguration takes place, one may think that preparing for more than one failure is not important. However, there are the "Shared Risk Link Groups" (SRLG), groups of links sharing the same fate due to some hidden common property (e.g. they are using the same optical cable), which may cause multiple failures at the same time. Even if both MRT and ARC can deal with the most common types of SRLGs (failure of links of the same linecard, and the failure of some ethernet network under the level of IP; namely the "LAN failure"), it is better to protect as many resources as possible.

4.

For the price of having only one reroute, MRT can guarantee that the

same failure will never be visited again. In contrast, since the same node can be an exit point for multiple ARCs, and since ARC puts packets back to shortest path after a failure, it is possible that a single failed node will be visited multiple times. It is very important that ARCs will properly handle this case with multiple rerouting, so sooner or later the destination will be reached. Moreover, the probability of such situation seems to be low.

5.

ARC can provide load balancing for the traffic with the "cursor node", since ARC gives not only a backup, but a new way for default routing as well. In contrast, since MRT only provide the detours, load balancing is not in the focus of MRT, but it is done with traditional ECMP. However, it is possible that a node have multiple blue or red next-hops with MRT, and in this case load balancing is possible even for packets on detours.

6.

IPFRR techniques are for designed for protection, i.e. they are used immediately after a failure happens. Typically after a failure, traditional routing protocols like OSPF reexplore the topology and reconfigure the forwarding entries based on the new topology. Packets start using the new shortest paths, and the network prepares to protect against a new failure. Normally, the reconfiguration after a failure may cause transient forwarding anomalies like micro-loops, but these are not acceptable for IPFRR.

Thus, several loop-preventing techniques were proposed in [[RFC5715](#)]. Since MRT provides two independent routing, i.e. the default shortest paths and the detours, it can use even Farside tunnels, which does not need protocol extension (basically, MRT can use one routing, while it reconfigures the other). Although, ARC is capable for loopfree convergence with centralized loop routing, it is not clear, which technique ARC could use in a distributed environment.

7.

Since MRT is an FRR technique, bicasting is out of its scope.

However, bicasting is theoretically possible, if we use the blue and the red next-hops; naturally, these paths will be node- and link-disjoint. In contrast, ARC was designed for not only FRR but bicasting, even if it cannot automatically provide disjoint paths. The advantage is that those paths can be shorter. For further details consult [[I-D.thubert-rtgwg-arc](#)].

[8.2.](#) Similarities between ARC and MRT

MRT uses a helper graph called GADAG, and detour computation is based on that graph. This helper graph is basically a set of ARCs with some extra direction information for the GADAG root as a destination. Thanks to this common point, computation algorithm and detours are very similar.

[9.](#) ARC specific properties

ARC is not only for FRR, but a complex forwarding technique, which can help in load balancing and speed up reconfiguration. Since MRT is an only for IPFRR, it cannot change routing in any way in a failure-free state, and the following properties do not exist.

However, it is important to note that for the advantages below we need to introduce the "cursor" node. Using the concept of a cursor needs some protocol extension, and it replaces traditional shortest path routing in some sense.

[9.1.](#) Multiple related breakages

In control plane, ARCs can find an exit if one exists and in principle, can be used to solve the Shared Risk Link Group (SRLG) problem. An ARC set is a Directed Acyclic Graph of ARCs, thus link reversal techniques can be applied to recover from complex breakages. Upon a second breakage in a same ARC, a segment is isolated. In control plane, it is possible to return all the incoming links in that segment. This may recurse if incoming ARCs become isolated as well. If the process recurses, a link may be returned a number of

times and this must be stopped by some infinity count. Once the local recovery settles, the exit path stands out.

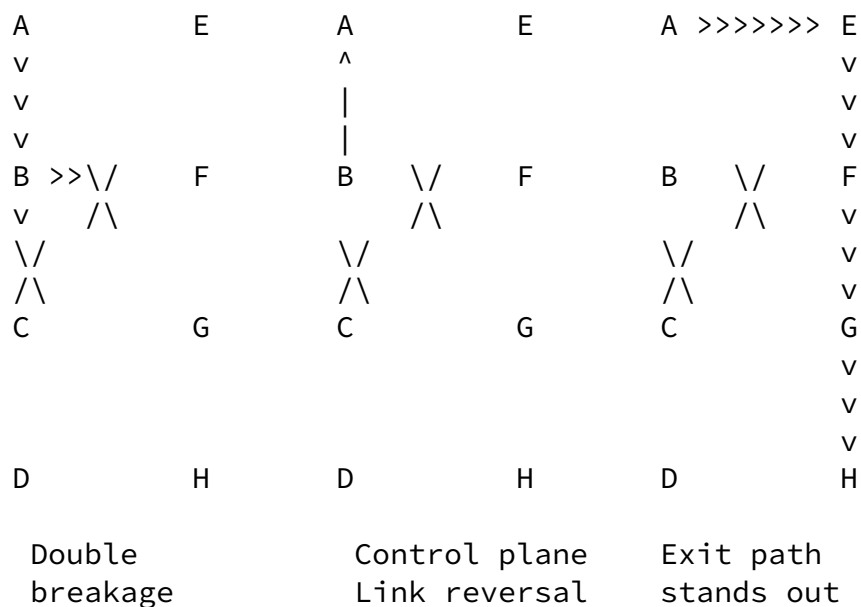


Figure 8: SRLG case

This example illustrates that the ARC segment around B is no more isolated by returning the A -> B edge into a B-> A link.

Recovery from multiple failures may also be performed using a variant of redundant trees, called independent directed acyclic graphs, which aims to utilize all the links in the network [Choi.2012.Independent-DAG]. With this approach, every node has one or more forwarding edges in the red/blue trees. Upon failure of all the red forwarding edges, the packet is transferred to the blue tree.

9.2. Multiple Destination and Load Balancing

The MRT and ARC techniques may be used for routing to redundant destination nodes. Given two nodes R1 and R2, MRT can construct two trees, tree T1 rooted at R1 and tree T2 rooted at R2. The path from any node to R1 on T1 and to R2 on T2 are still maximally-disjoint [Jayavelu.2009.Maintaining]. Similarly, the oLAF algorithm used for constructing ARCs is destination-oriented, in that it forms ARCs from the standpoint of the destination. This makes ARCs more complex to compute in a distributed fashion, but on the other hand enables to group a set of nodes.

One example of employing multiple destinations is to employ multiple border routers between one area and another. Thus, the routing in

one area may exit to the other area through one of the border routers.

With multiple destinations, it is possible not only to allow high availability but also to dynamically load balance between the member nodes. In an ARC, the node that has the logical responsibility of cursor is the only node that may send packets towards both edges in normal conditions. It is possible to create a control loop between a congestion point on one side and the cursor, in order to balance more traffic towards the other edge. If the cursor sends all its traffic towards the other edge, then the cursor responsibility may be transferred to the next node towards the congested edge to balance more traffic. If both edges are congested, then the congestion indication can be recursively injected in incoming ARCs.

An ARC based control loop does not need to involve metric changes or other routing advertisements, so it can be kept separate from the routing protocol. A simple control loop protocol can be used, that can be kept from oscillating with appropriate periods and adaptive filters.

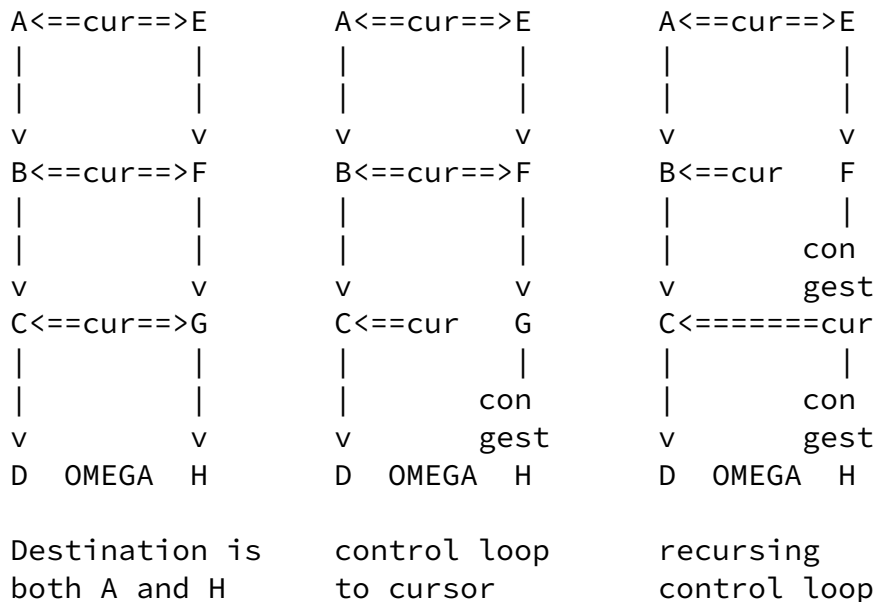


Figure 9: Complex Destination and Load Balancing

9.3. Hierarchical Routing

In a hierarchical network, routing happens within cells and then between cells. The collection of border routers between this cell and the next cell forms a complex destination for which an ARC set

can be formed within this cell. This can be done as soon as the

cells are formed, before any end-to-end route between cell is computed and whatever the routing protocol between cells is. This model applies in particular for such cells as areas, autonomous systems and optical rings.

[9.4.](#) Recovery from Node Failures

ARCS may be used to achieve fast recovery from node failures as well. The construction of ARC is similar to the construction of backup forwarding arcs forwarding arcs, as discussed in [\[Xi.2007.Fast-Reroute\]](#). The placement of cursors may still be performed once the backup paths (forwarding edges) are computed after failing each node.

[9.5.](#) Applied to bicasting (livelive)

MRT is not designed to compute an operational path but rather an alternate fast reroute solution that avoids any potential single breakage in the operational path that is computed otherwise, e.g. using the Shortest Path First algorithm. On the other hand, An ARC fabric joins together branches of a shortest path trees, so that most of the way along an ARC Set is shortest. Coloring the ARC Set for live live forces a colored packet to exit an ARC at an edge that is not always the shortest, but often is. It results that by design, both colored path are close to shortest, making ARCs a potential solution for livelive. Additionally, the support of an Omega that is formed of multiple destinations is a valuable asset for some applications such as video distribution. An ARC Set can be designed so that the lowest ARCs are between CDNs. It results that with a single ARC Set, all nodes in the network would have a pair of path that enable distribution from a relatively close CDN.

[10.](#) Manageability

This specification describes a generic model. Protocols and management will come later

[11.](#) IANA Considerations

This specification does not require IANA action.

12. Security Considerations

This specification is not found to introduce new security threat.

Thubert, et al.

Expires July 26, 2015

[Page 17]

Internet-Draft

ARCVsMRT

January 2015

13. Acknowledgements

The authors wishes to thank Alia Atlas, Patrice Bellagamba, Stewart Bryant, Robert Kebler, and Andras Csaszar for their participation to this particular work, and Dirk Anteunis, IJsbrand Wijnands, George Swallow, Eric Osborne, Clarence Filselfs and Eric Levy-Abegnoli for their continuous support to components of the work presented here.

14. References

14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC5715] Shand, M. and S. Bryant, "A Framework for Loop-Free Convergence", [RFC 5715](#), January 2010.

14.2. Informative References

- [I-D.enyedi-rtgwg-mrt-frr-algorithm]
Envedi, G., Csaszar, A., Atlas, A., cbowers@juniper.net, c., and A. Gopalan, "Algorithms for computing Maximally Redundant Trees for IP/LDP Fast- Reroute", [draft-enyedi-rtgwg-mrt-frr-algorithm-04](#) (work in progress), October 2013.
- [I-D.ietf-rtgwg-mrt-frr-architecture]
Atlas, A., Kebler, R., Bowers, C., Envedi, G., Csaszar, A., Tantsura, J., and R. White, "An Architecture for IP/ LDP Fast-Reroute Using Maximally Redundant Trees", [draft-ietf-rtgwg-mrt-frr-architecture-05](#) (work in progress), January 2015.

[I-D.thubert-rtgwg-arc]

Thubert, P. and P. Bellagamba, "Available Routing Constructs", [draft-thubert-rtgwg-arc-02](#) (work in progress), July 2014.

[I-D.thubert-rtgwg-arc-bicast]

Thubert, P. and I. Wijnands, "Applying Available Routing Constructs to bicasting", [draft-thubert-rtgwg-arc-bicast-01](#) (work in progress), October 2013.

[RFC5921]

Bocci, M., Bryant, S., Frost, D., Levrau, L., and L. Berger, "A Framework for MPLS in Transport Networks", [RFC 5921](#), July 2010.

Thubert, et al.

Expires July 26, 2015

[Page 18]

Internet-Draft

ARCVsMRT

January 2015

[14.3](#). References

[Cho.2012.Independent-DAG]

Cho, S., Elhourani, T., and S. Ramasubramanian, "Independent Directed Acyclic Graphs for Resilient Multipath Routing", IEEE/ACM Transactions on Networking, vol. 20, no. 1, February 2012, <<http://dx.doi.org/10.1109/TNET.2011.2161329>>.

[Erlebach.2009.Path-Splicing]

Erlebach, T. and A. Mereu, "Path splicing with guaranteed fault tolerance", Proceedings of IEEE GLOBECOM, November-December 2009, <<http://dx.doi.org/10.1109/INFCOM.2005.1498553>>.

[Jayavelu.2009.Maintaining]

Jayavelu, G., Ramasubramanian, S., and O. Younis, "Maintaining colored trees for disjoint multipath routing under node failures", IEEE/ACM Transactions on Networking, vol. 17, no. 1, February 2009, <<http://dx.doi.org/10.1109/TNET.2008.919323>>.

[Xi.2007.Fast-Reroute]

Xi, K. and H. Chao, "IP fast rerouting for single-link/node failure recovery", Proceedings of IEEE BROADNETS, September 2007, <<http://eeweb.poly.edu/~chao/docs/public/>

[ipfrr_single1.pdf](#)>.

Authors' Addresses

Pascal Thubert (editor)
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
FRANCE

Phone: +33 497 23 26 34
Email: pthubert@cisco.com

Gabor Sandor Enyedi
Ericsson

Thubert, et al.

Expires July 26, 2015

[Page 19]

Internet-Draft

ARCVsMRT

January 2015

Srinivasan Ramasubramanian
University of Arizona
1230 E. Speedway Blvd.
Tucson, AZ 85721
USA

Phone: +1 520 621 4521
Email: srini@ece.arizona.edu

