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

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

Alia Atlas, Maciek Konstantynowicz (Juniper Networks)
Gábor Enyedi, András Császár (Ericsson)
Russ White, Mike Shand (Cisco Systems)

IETF 81, Quebec City, Canada

Outline

- Motivation
- Architecture
- Algorithm
- Next Steps

LFA isn't enough…

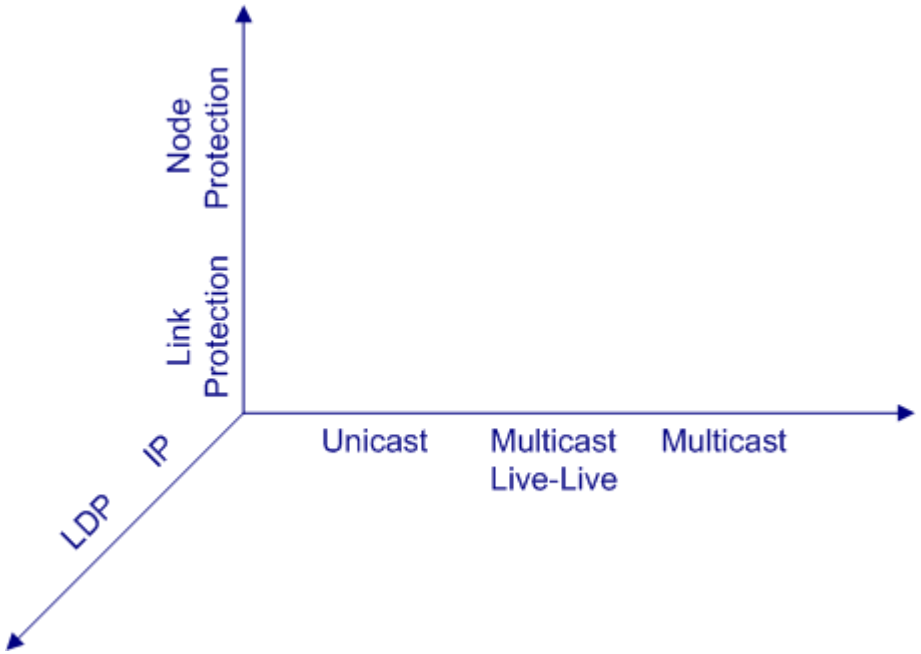
- LFA is useful and networks are being designed to improve coverage (draft-ietf-rtgwg-lfa-applicability)
- …but LFA doesn't guarantee 100% coverage.
- NotVia
  significant network state
  - Research done to reduce it, but nothing sufficiently practical & it's been years

Increasing Requirement and Demand for IP/LDP Fast-Reroute with 100% Coverage

Unicast isn't enough…

- Multicast is increasingly deployed and needs protection…
  - Need live-live as well as fast-reroute

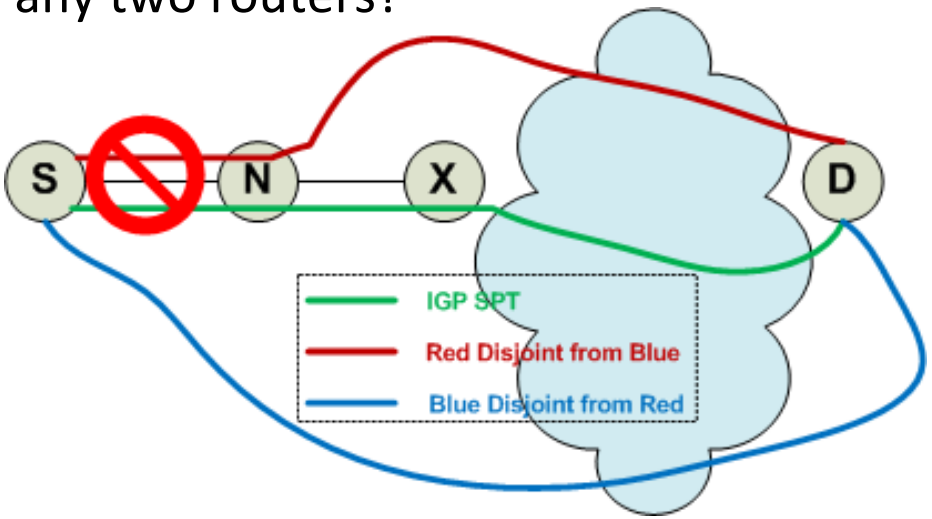
Need to consistently cover each case. *



* Still working on SRLG Protection ideas

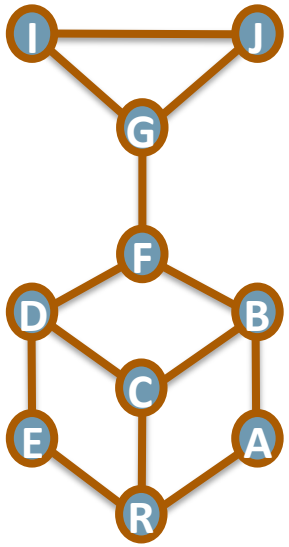
What if…

- We can always* compute two link and node disjoint paths between any two routers?



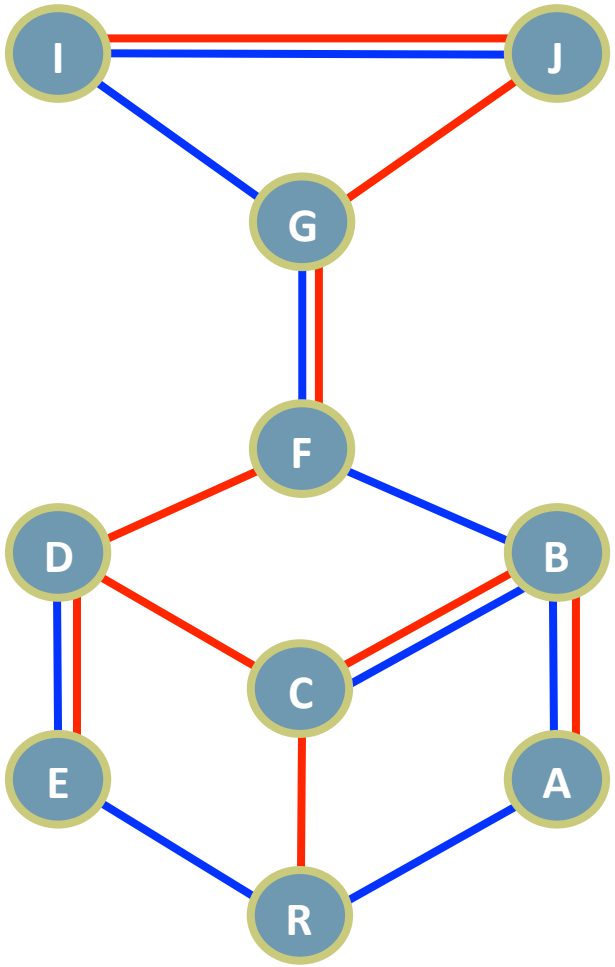
* If network doesn't have other path (e.g. not 2-link or 2-router connected), compute the most disjoint (e.g. maximally redundant) possible based on the network.

- The primary neighbor (N) **may** be on **at most** one of the red path from S to D or the blue path – but guaranteed* not both.
- Recent Research tells us how:
  - Compute destination-routed Maximally Redundant Trees
- We can plug-n-play different algorithms into this architecture.
- Avoids failure-specific paths to reduce state compared to NotVia



Maximally Redundant Trees (MRTs)

- Compute a pair of disjoint MRTs per IGP-area destination (e.g. router or multi-homed prefix).
- Trade-offs with algorithm between speed and length (**not existence**) of path. Can compute complete MRTs or just next-hops.
  - An algorithm exists that allows a router to compute its next-hops on each pair of MRTs for each node in the network in O(e) time.
- Algorithms designed to work in real networks:
  - Just as with SPF, algorithm is based on a common network topology database – no messaging is required.
  - MRTs are computable when network isn't 2-edge or 2-vertex connected – most disjoint paths are computed.
  - Need to define consistent tie-breakers to ensure identical destination-rooted MRTs computed by all routers in IGP area.



Two maximally redundant blue/red trees rooted at node R

Usability Goals for the new IPFRR scheme

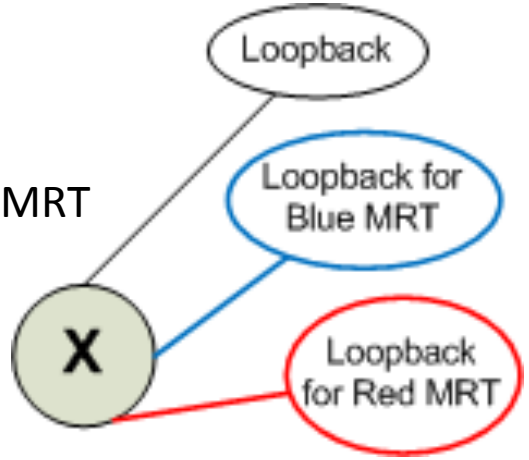
- Ensure maximum link and node disjointness for any topology
- Automatically compute backup next-hops based on the link-state database.
- Do not require any signaling in case of failure, use pre-programmed backup next-hops
- Introduce minimal additional addressing and state on routers
- Enable phased deployment and backward compatibility
- Do not impose requirements for external computation
- Handle real networks – that may not be fully 2-connected, due to previous failure or design. This also helps with phased deployment.

Using MRT Alternates for Unicast

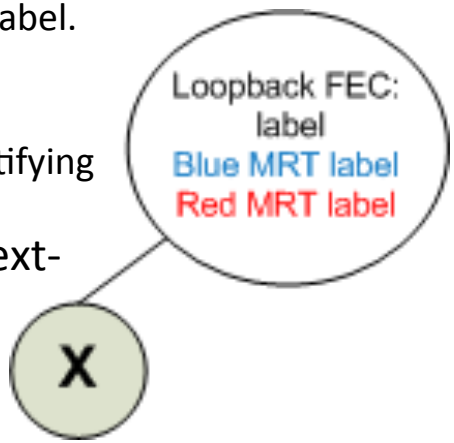
- MRTs supplement LFAs and do not replace them.
- Compute normal SPT and per-IGP-area-destination Blue & Red MRTs.
- Pre-compute whether Blue MRT or Red MRT will survive primary next-hop failure & select which to use.
  - Work-in-progress: doing this quickly for the fast algorithm that computes only next-hops
- In case of a failure:
  - If a node-protecting LFA exists, use it.
  - Otherwise, if only link-protection is needed and there is a link-protecting LFA backup, use it.
  - Otherwise, use the pre-selected MRT, whether Blue or Red.

Unicast Forwarding: For Want of 2 bits…

- IP unicast – tunneling is the only option
  - Each router supporting MRT would need to announce two additional loopbacks associated with MRT colors
  - Transit MRT routers would use these addresses to identify MRT topology to forward traffic along
  - LDP tunneling as alternative



- LDP unicast – two alternatives
  - (1) topology-id labels
    - Specify two additional labels, one per MRT color
    - Stack topology-id label on top of LDP FEC label
    - When sending packet on MRT, 1st swap FEC label, then push MRT label. Every hop, pop MRT label, swap FEC label, and push MRT label.
  - (2) topology encoded in labels
    - Routers would need to provide two additional labels per FEC, identifying MRT color
  - No new hardware capabilities needed – basic MPLS or context-label spaces.



Multicast Live-Live with MRTs

- Use MRTs rooted at the Multicast Source
- Extend PIM and mLDP to indicate which topology (e.g. blue or red MRT) to use for the multicast tree.
- Receivers join both the blue MRT and red MRT to receive traffic.
- As in draft-karan-mofrr, receivers determine which packets to keep.
- Work-in-Progress: Packets may need to identify their topology/tree for non-2-connected networks. (e.g. if a common link must be used)

Multicast and Fast-Reroute: Basic Issues

- Several basic issues with FRR for multicast
  a) PLR does not know the set of next-next-hops in the multicast tree
  b) For mLDP, the PLR doesn't know the right labels to use for the next-next-hops in the multicast tree
  c) MP does not know upon what interface to expect backup traffic.

- Basic Protocol Extensions needed
  a) Extend PIM Join Attributes and mLDP to specify the router's next-hops on the multicast tree.
  b) Extend mLDP to provide the advertised labels for the router's next-hops.
  c) Either explicitly signal Upstream Backup Joins or tunnel packets so MP knows packets are from alternate.

Multicast Traffic Handling

- When PLR detects a failure, it doesn't know if it is a link failure or a node failure.
  - For unicast, always send on a node-protecting alternate.
  - For multicast, send traffic on both a node-protecting alternate and a link-protecting alternate. The primary neighbor may have receivers too!
  - PLR sends traffic on alternates for a configurable time-out.
- MP can independently determine whether to accept alternate traffic.
  - If primary upstream link is up, keep accepting traffic via that.
    i. Either failure hasn't happened or
    ii. Link failure happened and upstream neighbor is getting and forwarding traffic on alternate.
  - Otherwise, accept and forward alternate traffic.
  - When traffic is received on a new primary upstream link, stop accepting and forwarding alternate traffic.
  - Work-in-progress: For mLDP, need to consider all direct links to upstream.

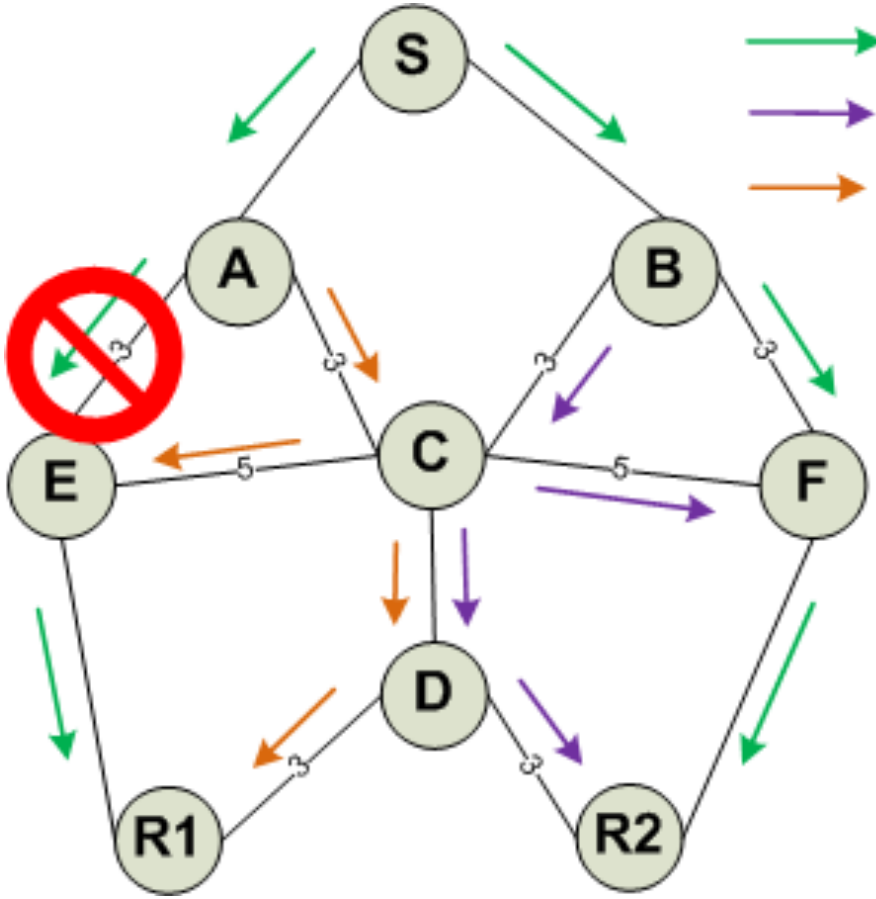
Where to Replicate? At PLR??

- PLR can use unicast alternates to the primary neighbor and next-next-hops.
  - Replicate at PLR
  - Tunnel multicast traffic in unicast (either IP or LDP) where the outer IP address or LDP label indicates the MP and the appropriate topology (SPT, blue MRT or red MRT).
- MP recognizes that traffic is via alternate because it comes in tunneled with MP as destination.
  - Could use explicit NULL or a known label for LDP.
- Standard issues with ingress replication
  - Same packet may be duplicated many times on a link.
- Requires tunneling of traffic.

Where to Replicate? Along alternate-tree?

- PLR knows its alternate (preserve independence of selection) – so needs to originate the *Upstream Backup Joins*.
  - Separate messages unicast destined to each MP
  - Indicate MP and topology (SPF, blue MRT, red MRT) to use.
  - Backup tree is the merged set of unicast alternates.
    - E.g. Alternate to E might be LFA, alternate to F might be on a blue F-rooted MRT, and alternate to G might be on a red G-rooted MRT.
- Need to merge alternate-trees from different PLRs for same (S,G).
  - Drawback is unnecessary replication
- To reduce state, could merge alternate-trees from different PLRs for same S.
  - Same candidates for next-next-hops

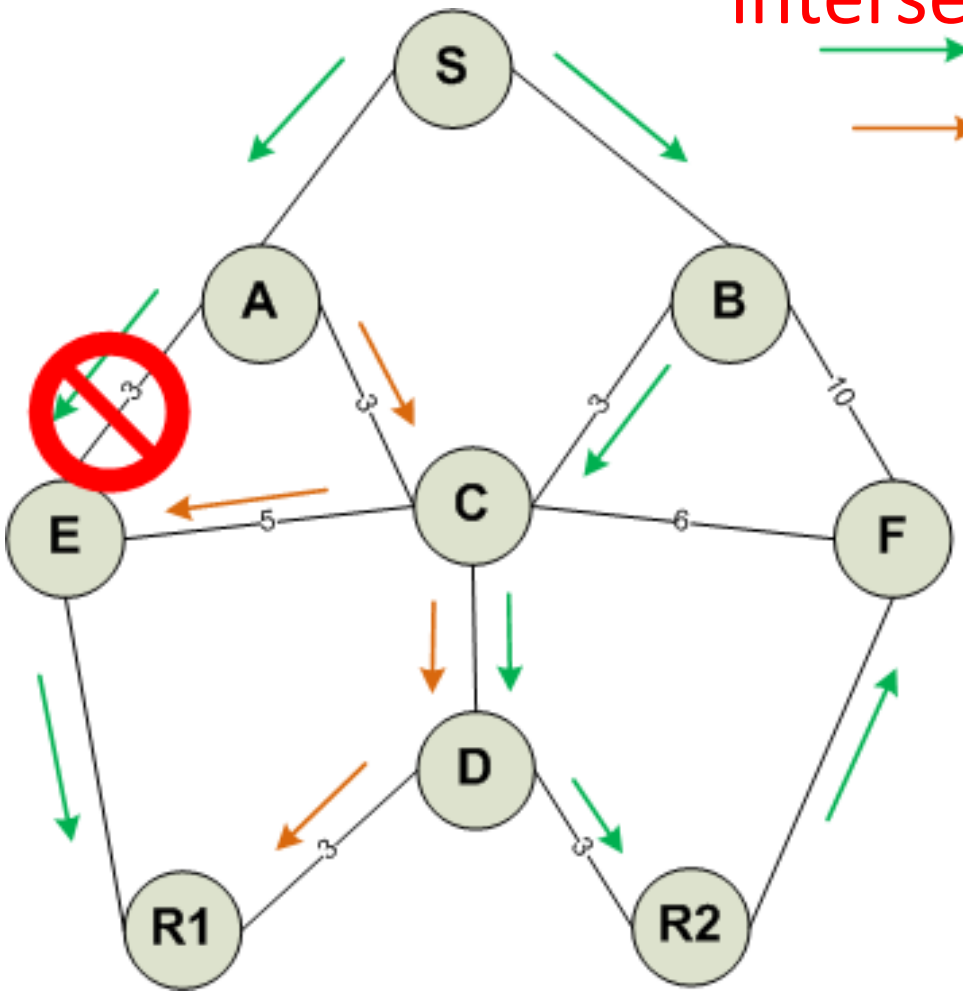
Cntd. Where to Replicate? Along alternate-tree? Merging Alternate-trees from different PLRs



- D can't tell if a packet from C should go on orange alternate tree or on purple alternate tree.
- Merge the alternate-trees.
- MPs know which traffic to keep or discard – so *just* use bw on alternate-tree links.
- Less state and simpler.

Cntd. Where to Replicate? Along alternate-tree?

Multicast Primary Tree and Alternate-Tree intersect



S Multicast tree on SPT

A's Alternate-Tree using LFA

- D can't tell if a packet should go on the primary tree or on the alternate-tree.
- Need to mark traffic in alternate-tree as being alternate.
  - LDP topology-id label
  - Different multicast address (and rewrite at MPs)
  - Other ideas??

Work in Progress

- Important Practical Problems:
  - An ABR may need to know about 2 MRTs per area (and the packet may need to indicate the area). How can we reduce this state?
  - Details on how to handle multi-homed prefixes.

- Protocol Work to Do:
  - OSPF, ISIS, PIM, LDP, mLDP
  - Capability advertisements and Management Controls

Algorithmic Work In Progress

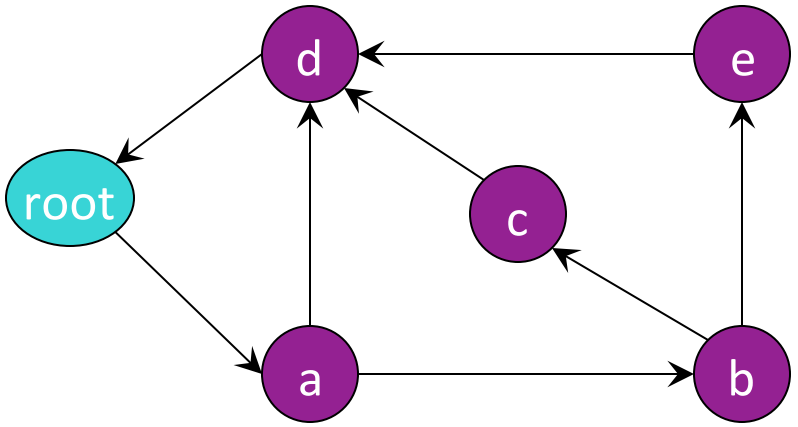
- Define rules and behavior for broadcast interfaces (straightforward).
- Pre-compute, when only MRT next-hops are known, which MRT won't fail with primary neighbor.
- Administratively Unavailable Links and Nodes
- Asymmetric Link Costs
- Tie-breaking rules
- Specification of using multiple next-hops (like ECMP) in MRTs
- Complete specification of fast-compute MRTs algorithm
- Complete specification of better-paths MRTs algorithm
- How to build MRTs that are SRLG-disjoint?

Outline

- Motivation
- Architecture
- Algorithm
- Next Steps

Finding MRT

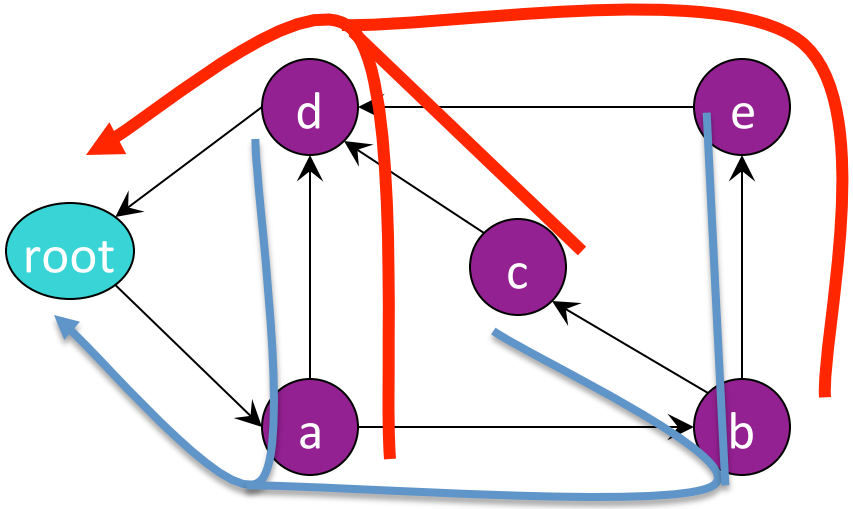
- Key: special partial order is needed
  - Can be represented by a directed graph
    - Almost Directed Acyclic Graph
    - Remove the root of it to convert it into a DAG



- root<a<b<c<d<root
- b<e<d

Why is it good?

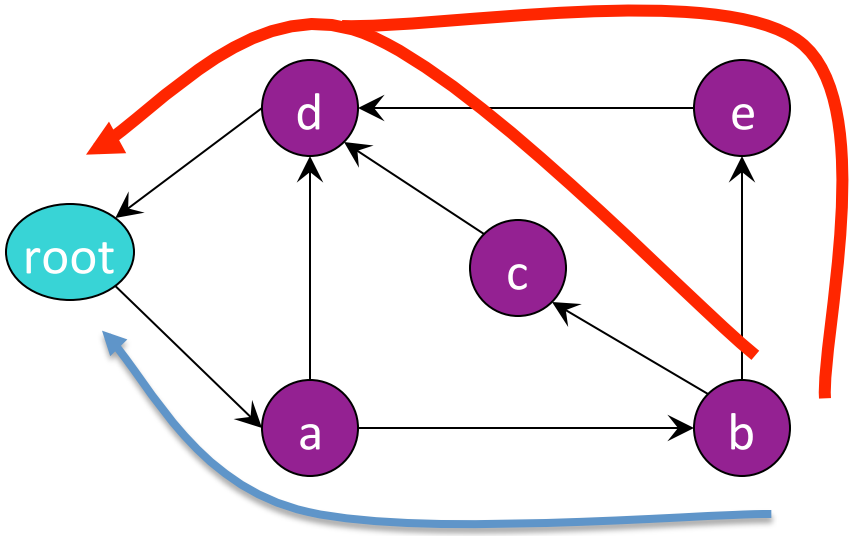
- Use an increasing and a decreasing path
  - They are disjoint!
  - Combine them, and you get two trees



- Combine them in any way, they are disjoint on path from node to root

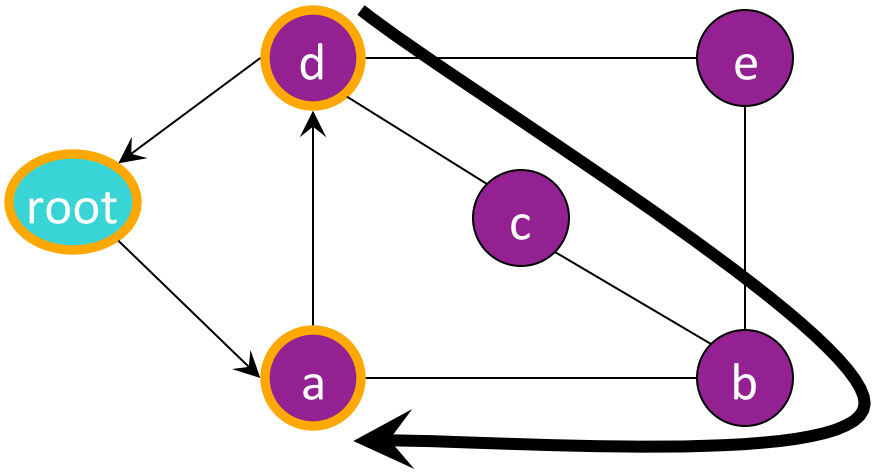
Load sharing?

- You can even vary the next hop
  - You can do load sharing like ECMP



Creating a partial order (ADAG)

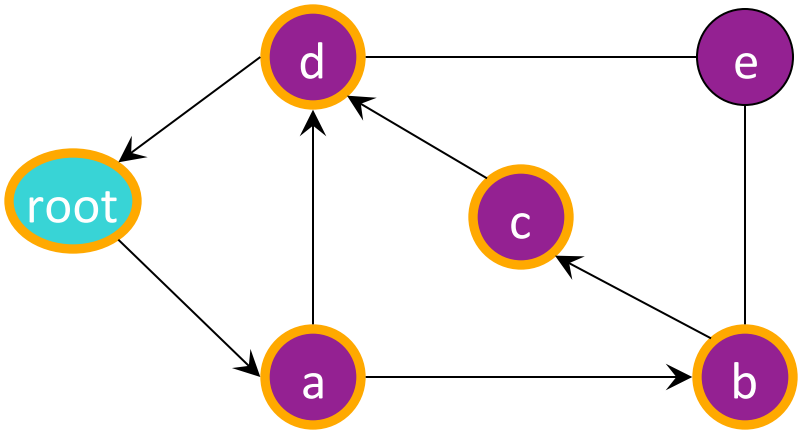
- Find a path
  - Between two vertices already in the order
  - Using only vertices not in the order
  - Add it in a direction, which keeps up current order



- root<a<d<root

Creating a partial order (ADAG)

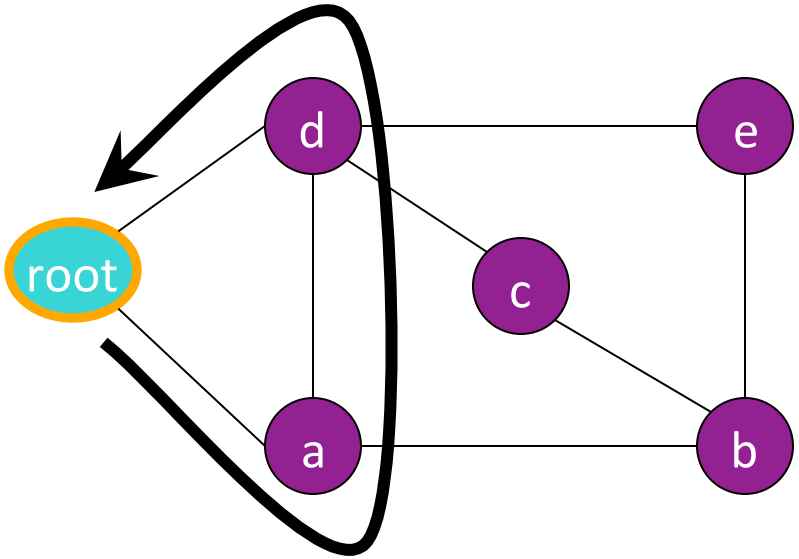
- Find a path
  - Between two vertices already in the order
  - Using only vertices not in the order
  - Add it in a direction, which keeps up current order



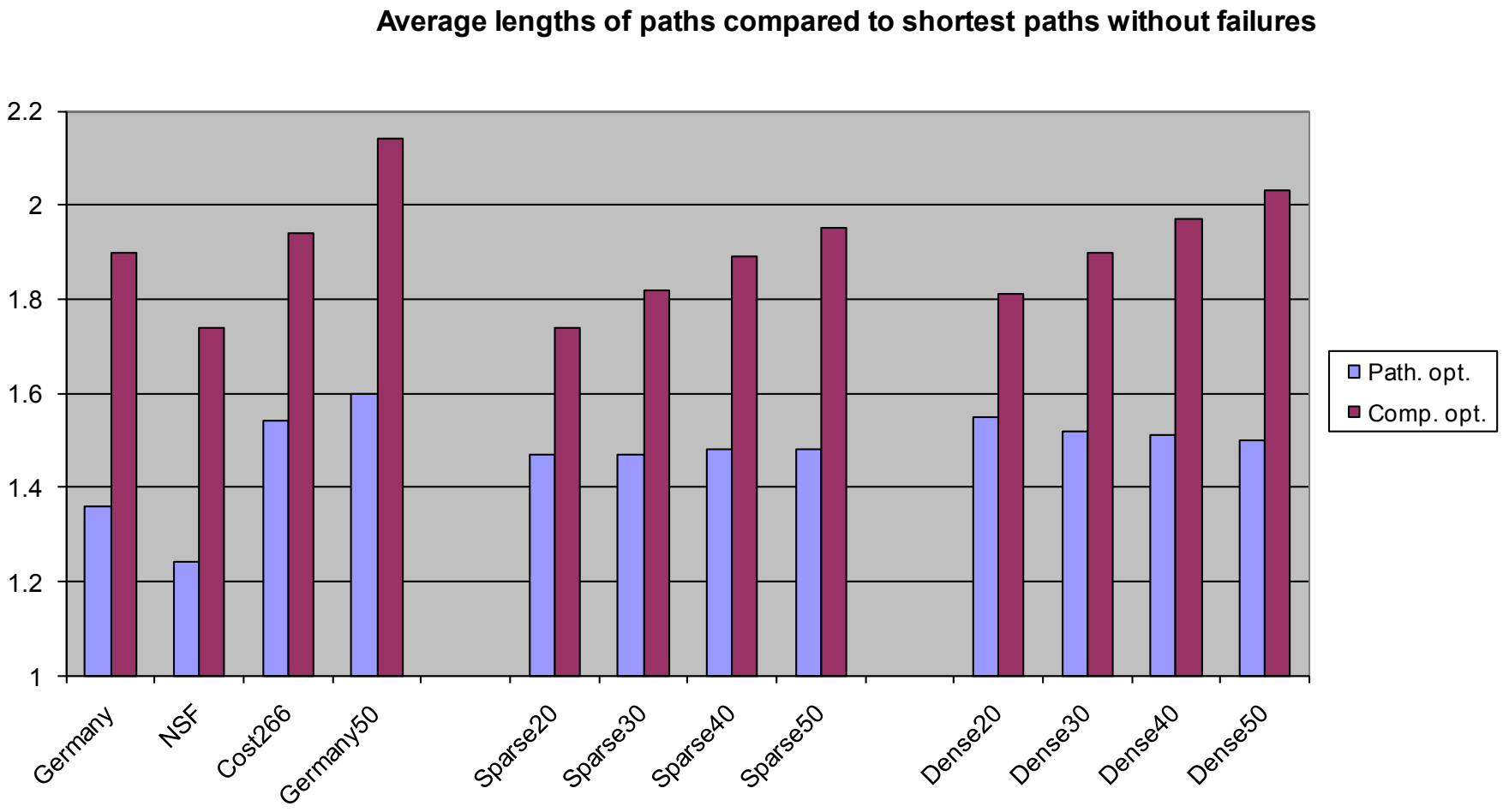
- root<a<**b<c**<d<root

What is with the first path?

- Find a path from the root to the root
  - Actually, that is a cycle… ☺



Short vs. Fast?



Technology has matured

- Very well studied area (at least 20-30 papers)
- Various optimizations for
  - Sensor networks
  - Optical networks
- Works based on link state database
- Finding 3/4 trees in 3-/4-connected networks
- Algorithm complexity less than SPF

Goal: Find the best trade-offs for an algorithm in real networks.

Next steps

- Continue work
- Get feedback on preferred trade-offs
- Interested in becoming a RTGWG draft
  - Good starting point with architecture

Questions & Comments?