ARCs for Fast Reroute

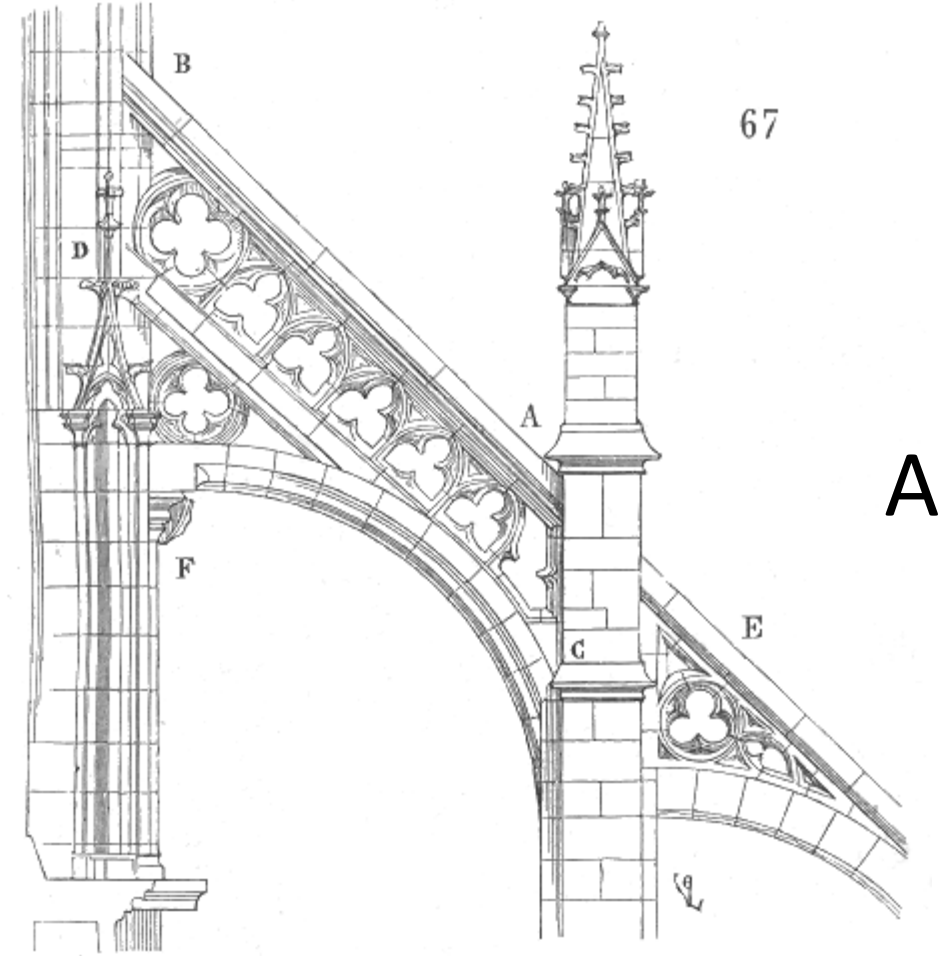
Available Routing Construct

draft-thubert-rtgwg-arc-00

Pascal Thubert
Cisco

Patrice Bellagamba
Cisco

RTG Area WG, Atlanta, 2012



ARCs for Fast Reroute

draft-thubert-rtgwg-arc-00

Pascal Thubert
Cisco

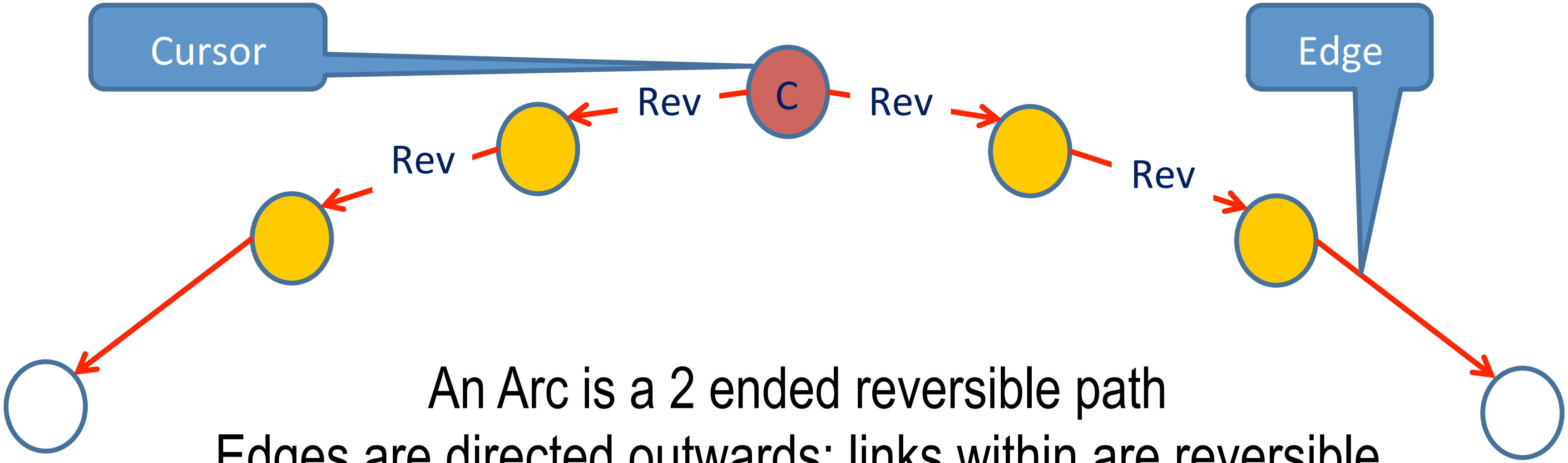
Patrice Bellagamba
Cisco

RTG Area WG, Atlanta, 2012

Problem: routing availability

- Classical trees and Directed Acyclic Graph (DAG) topologies do not provide non-congruent alternate routes for all nodes
- State of the art Fast Reroute (FRR) tolerates 1 failure but may drop traffic or loop upon 2
- Yet accidental damage to a fiber harness hits multiple links (Shared Risk Link Group)
- Same goes for interferences in wireless

Arc concept



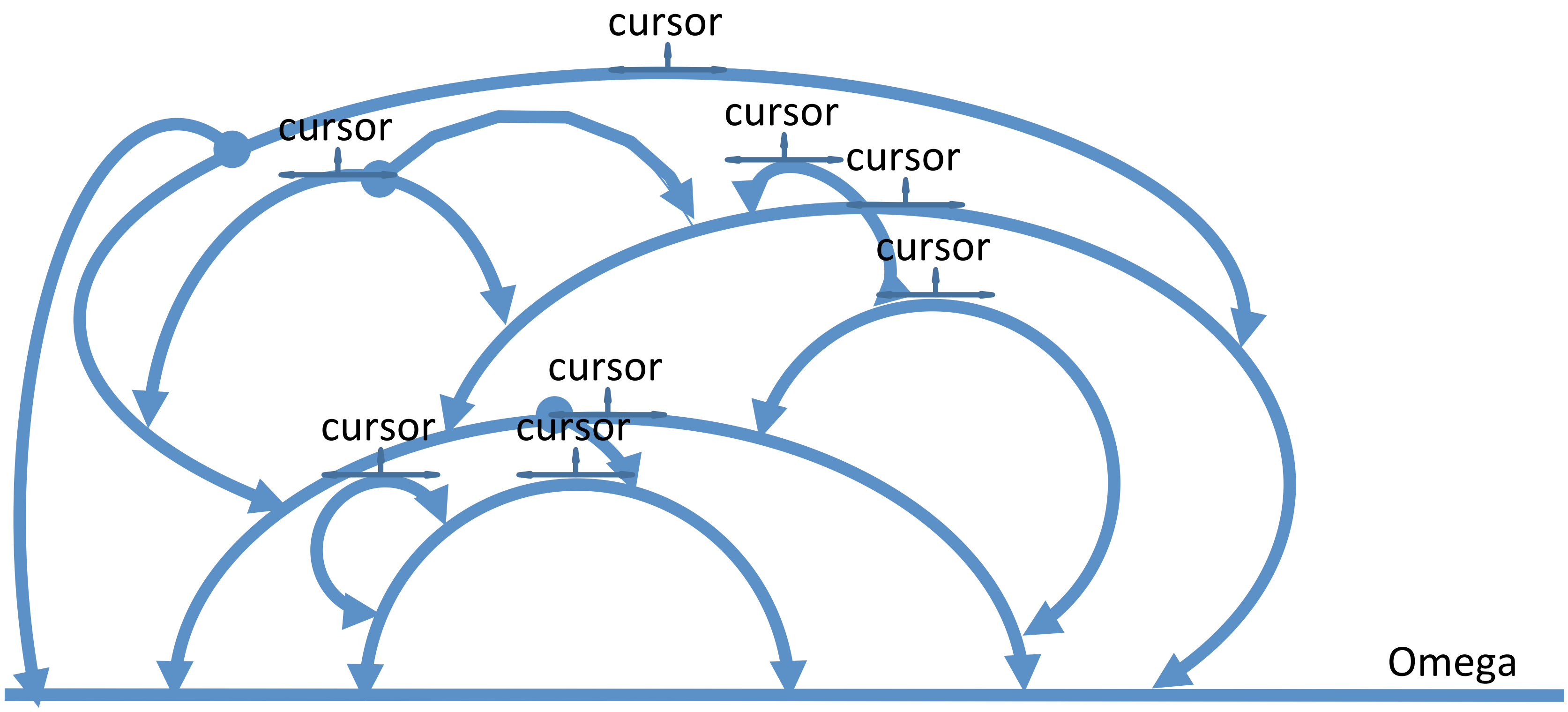
An Arc is a 2 ended reversible path
Edges are directed outwards; links within are reversible
An arc is resilient to any link or Junction break by returning links
Links are oriented from cursor to edges and returned by moving the cursor.

We build Arcs between Safe Nodes

ARC topology

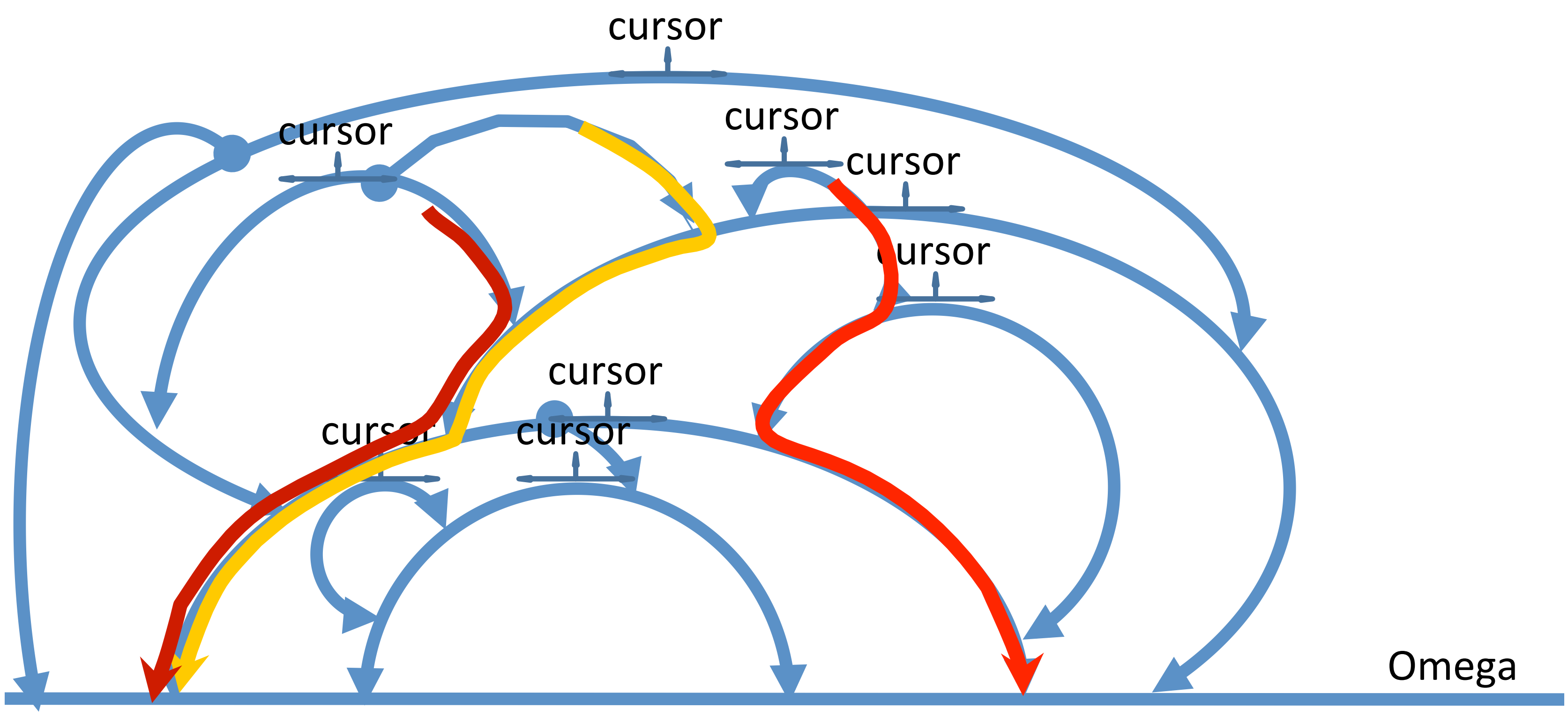
ARCs form dual or multi-ended structures

- An ARC stitches 2 SPF subpaths together
- ARCs + buttressing ARCs = Comb
- One cursor per ARC / Comb as the water separation line



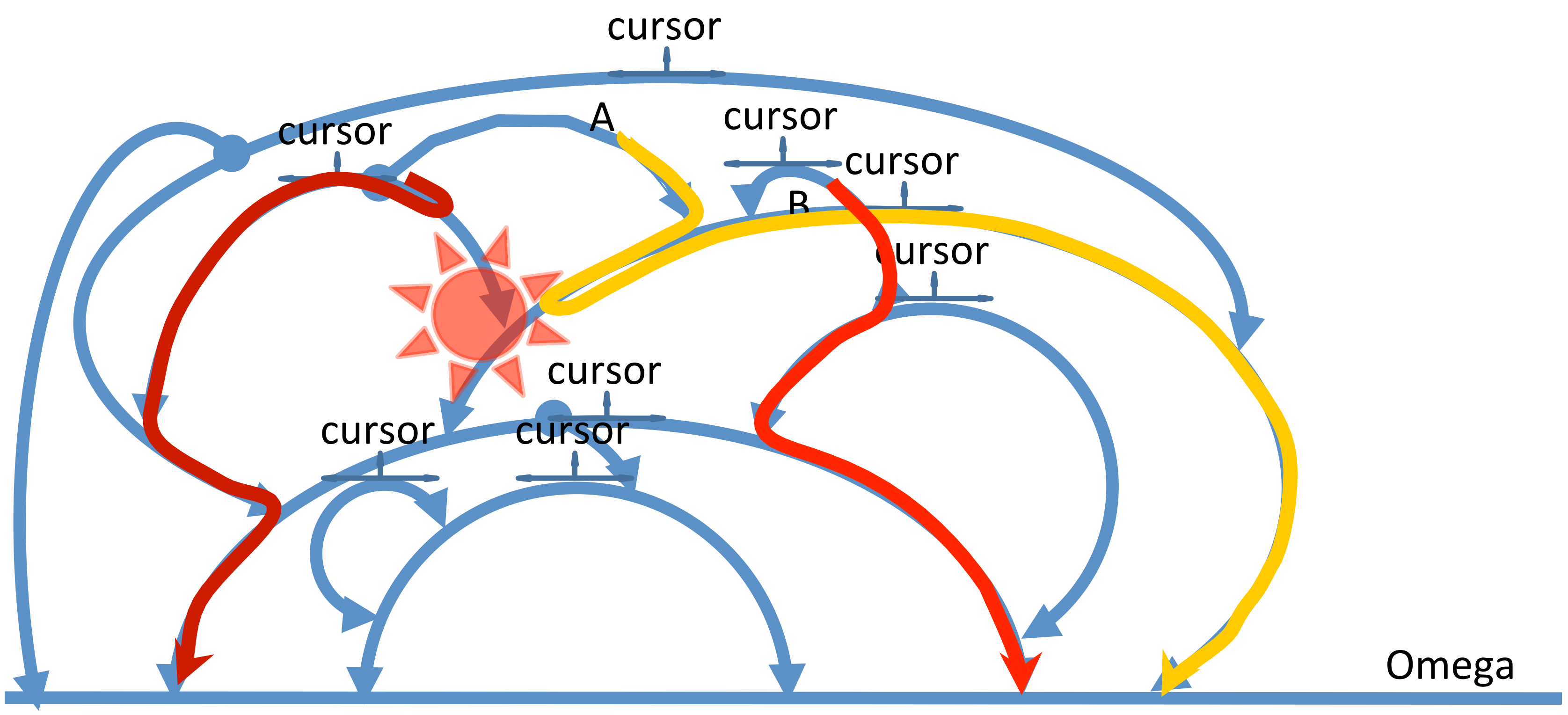
Forwarding

In normal operations, traffic flows away from the cursor and cascades from ARC to ARC along shortest path



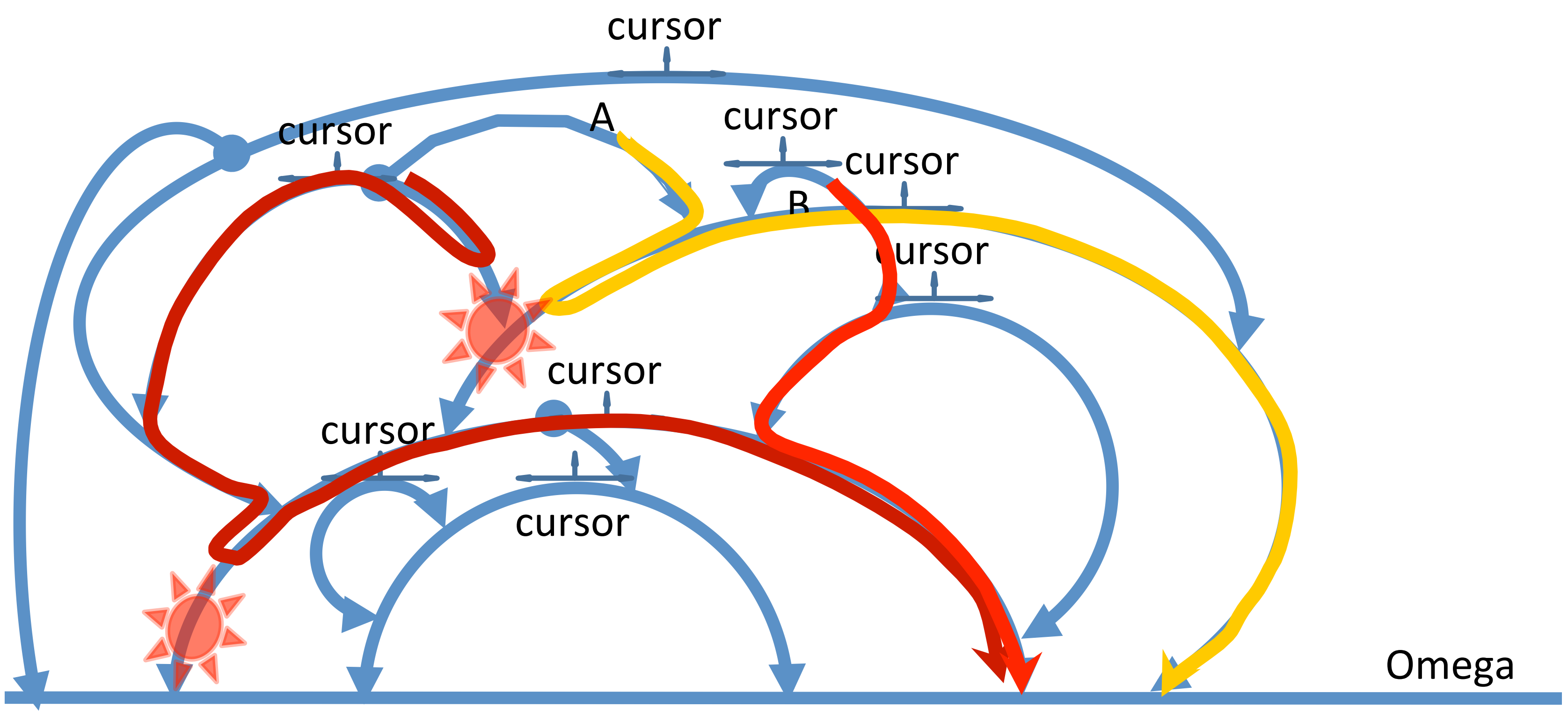
Forwarding errors

Are Addressed inside an ARC by returning the incoming link,
In order to exit via the other edge of this ARC
In control plane, this means that the Cursor is placed at the failure location



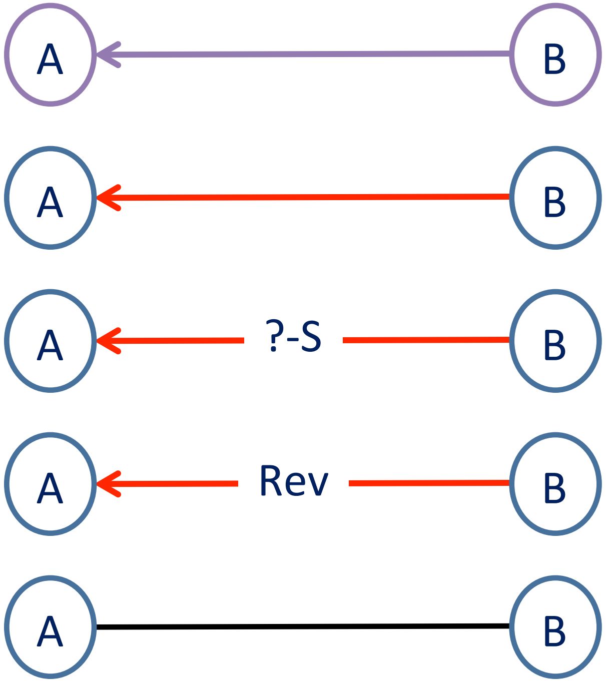
Double breakage

Each ARC is its own domain of fault recovery



Questions?

Notations for Link types



A is SPF successor of B

A is non shortest path successor of B

B -> A is unresolved for Safe Node S

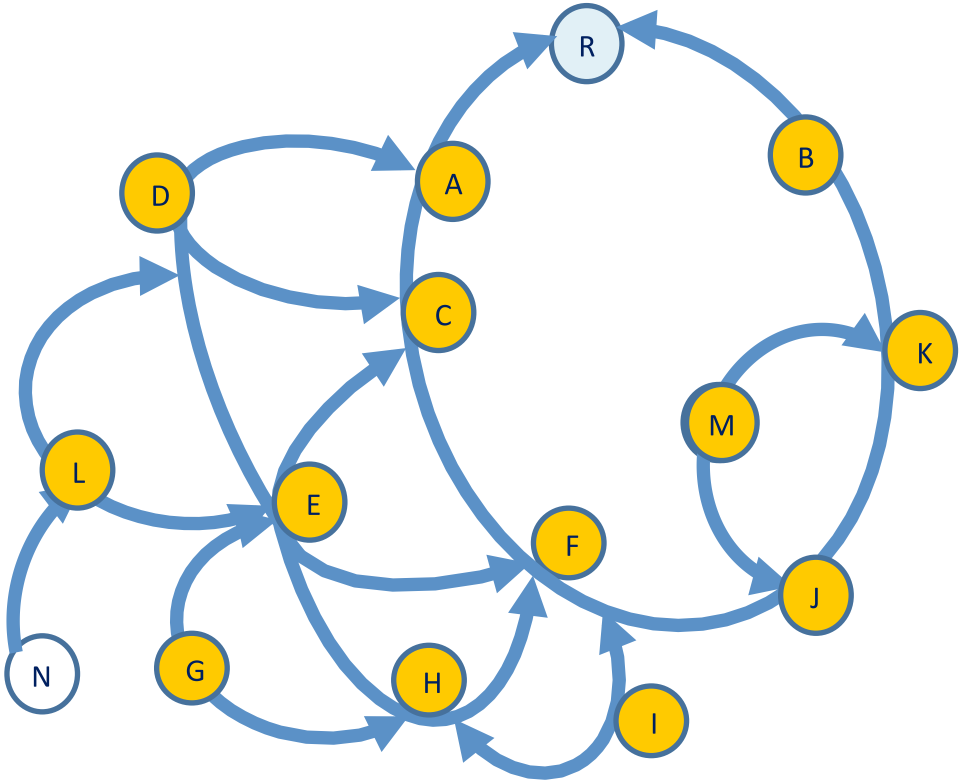
B is standby alternate on A isolation

Non SPF Link used to join an ARC

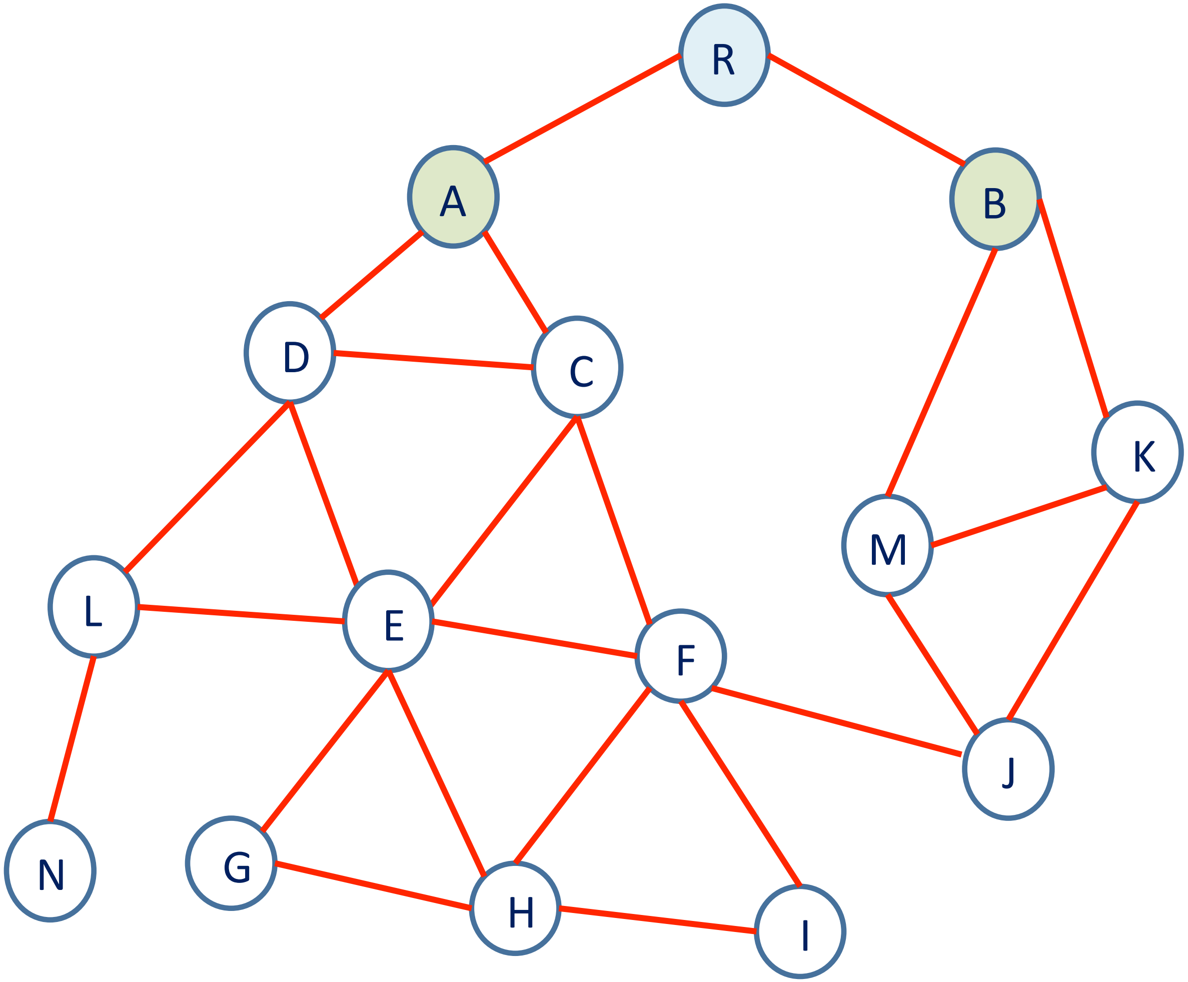
LAF (Lowest ARC First)

LAF is a SPF variation that creates ARCs by connecting SPF paths

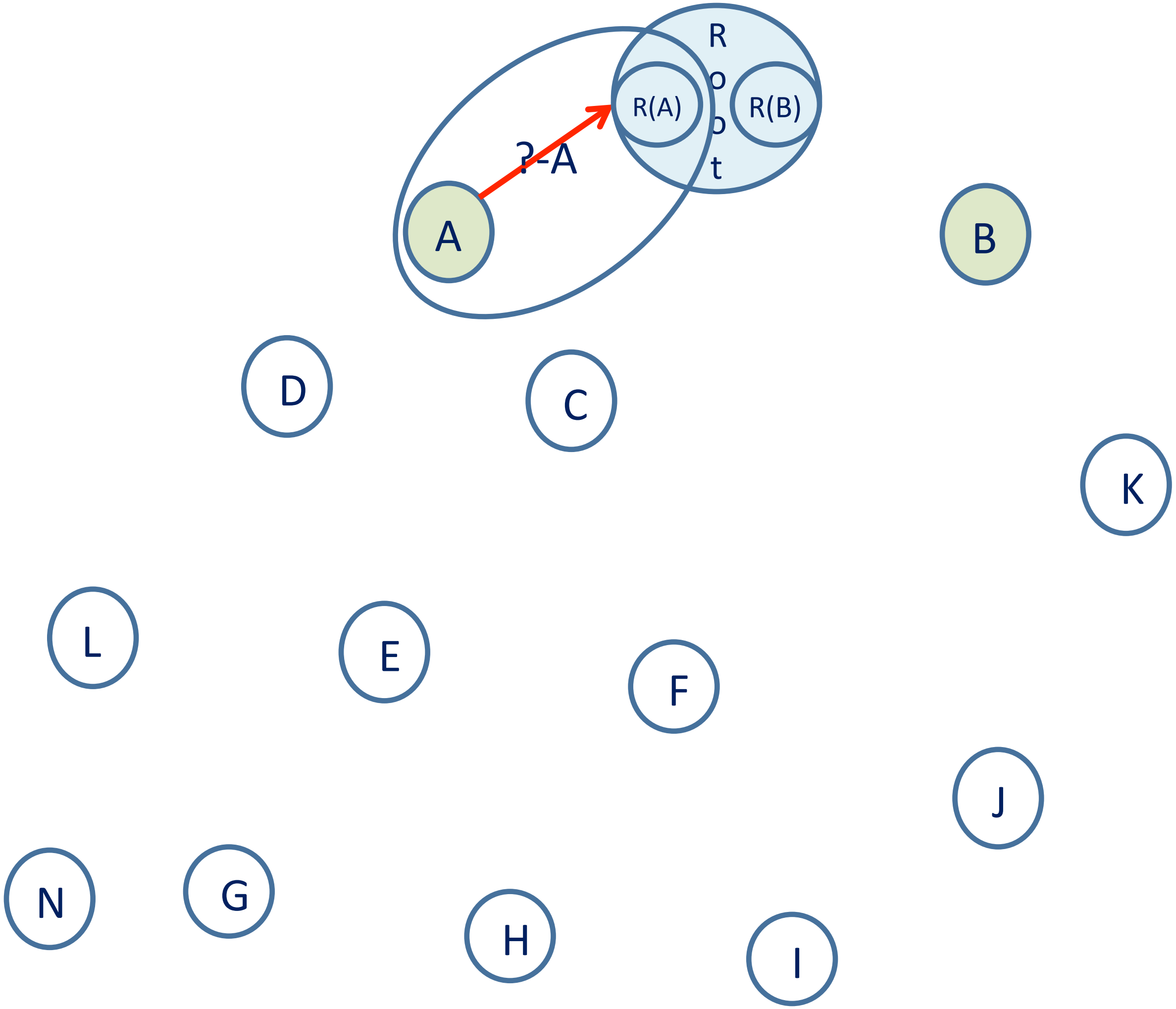
- The ARCs include the SPF tree
- The algorithm identifies the mono-connected zones
- and provides redundancy inside such zones



oLAF Example: Initial topology



Running the modified Algo, Start from R:

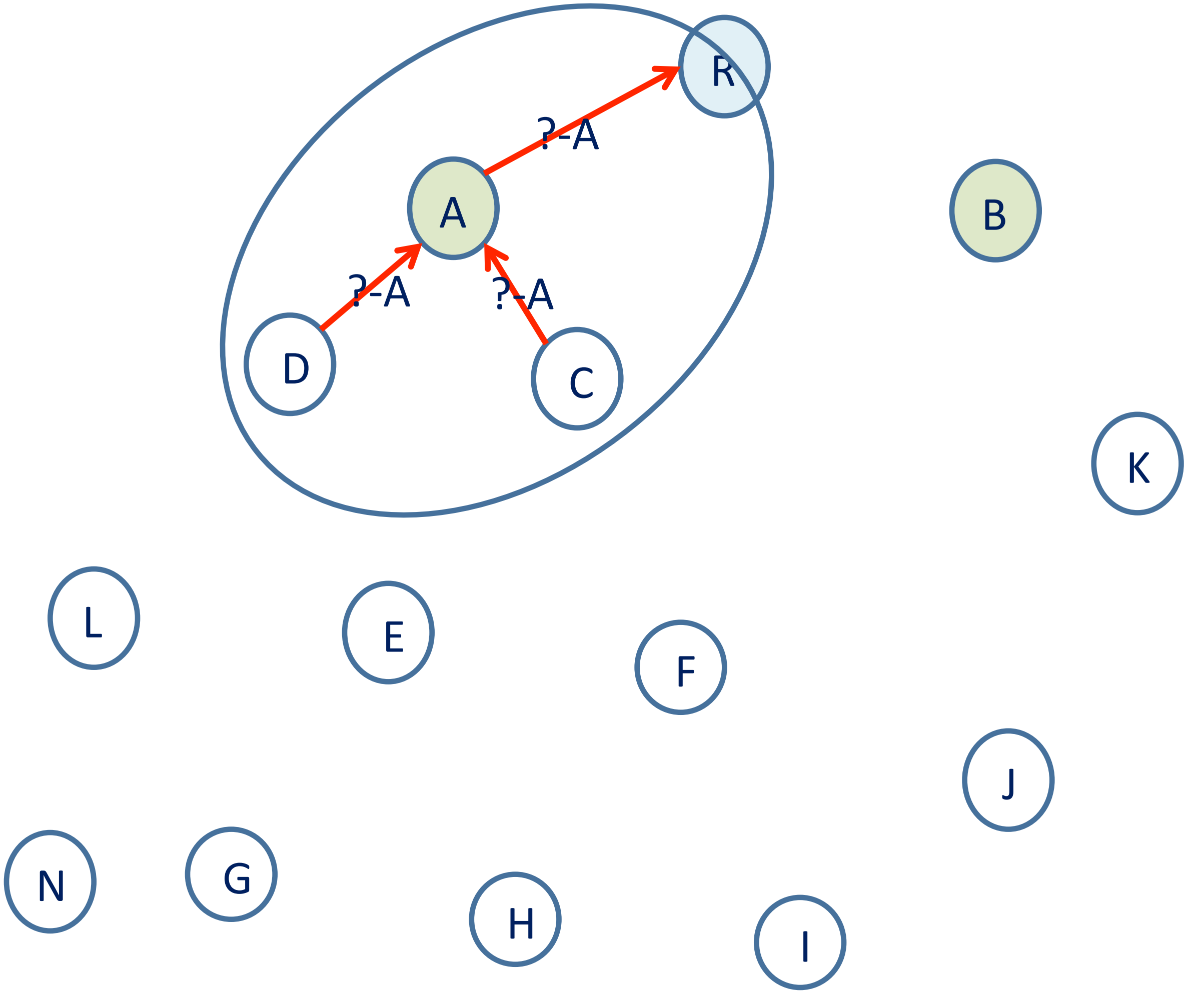


A and B are Heir

Since we have a single root we create virtual roots R(A) and R(B)

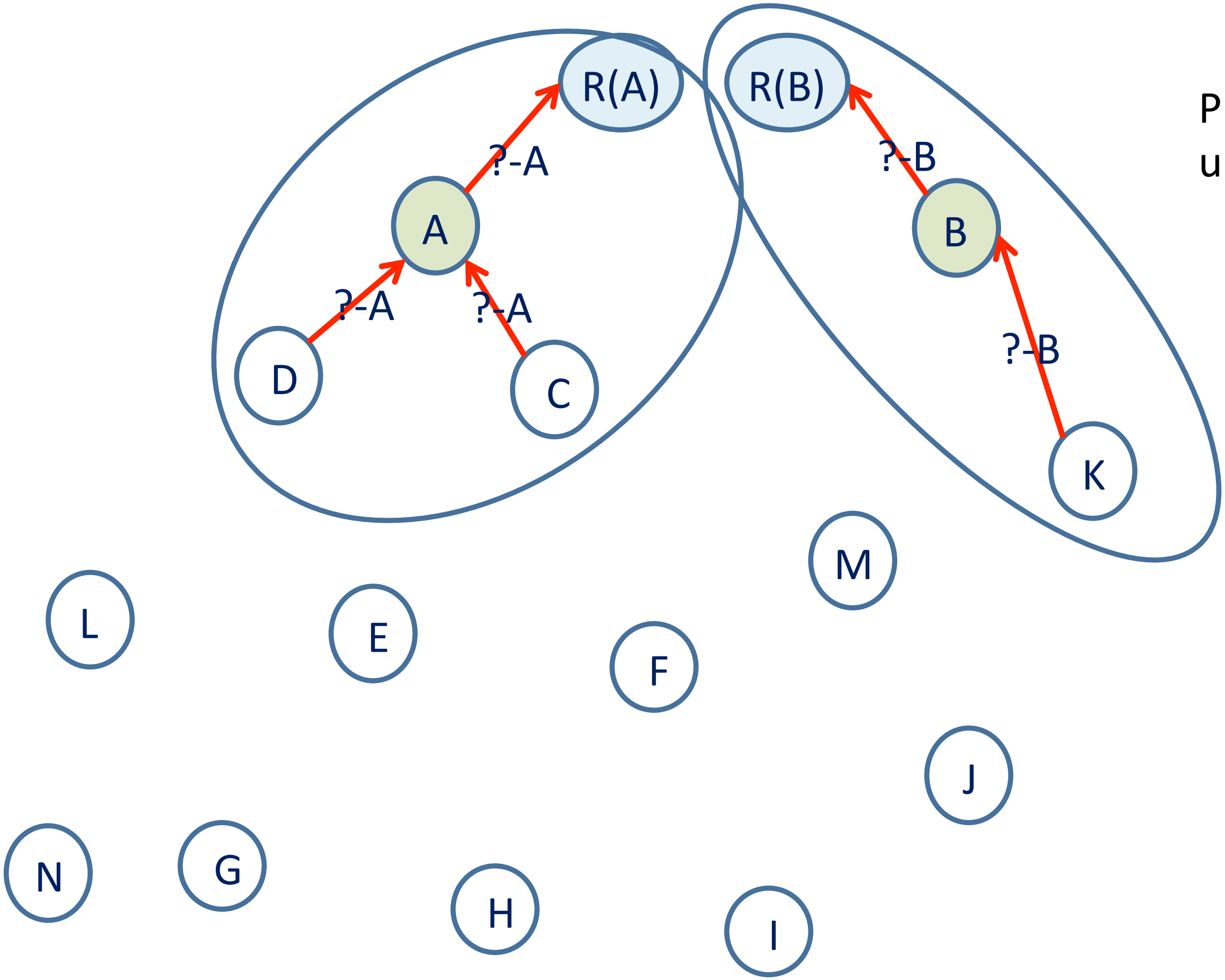
We note the set dependent on R(A) as ?-A for convenience

Picking A (closest to root), and D, and C:



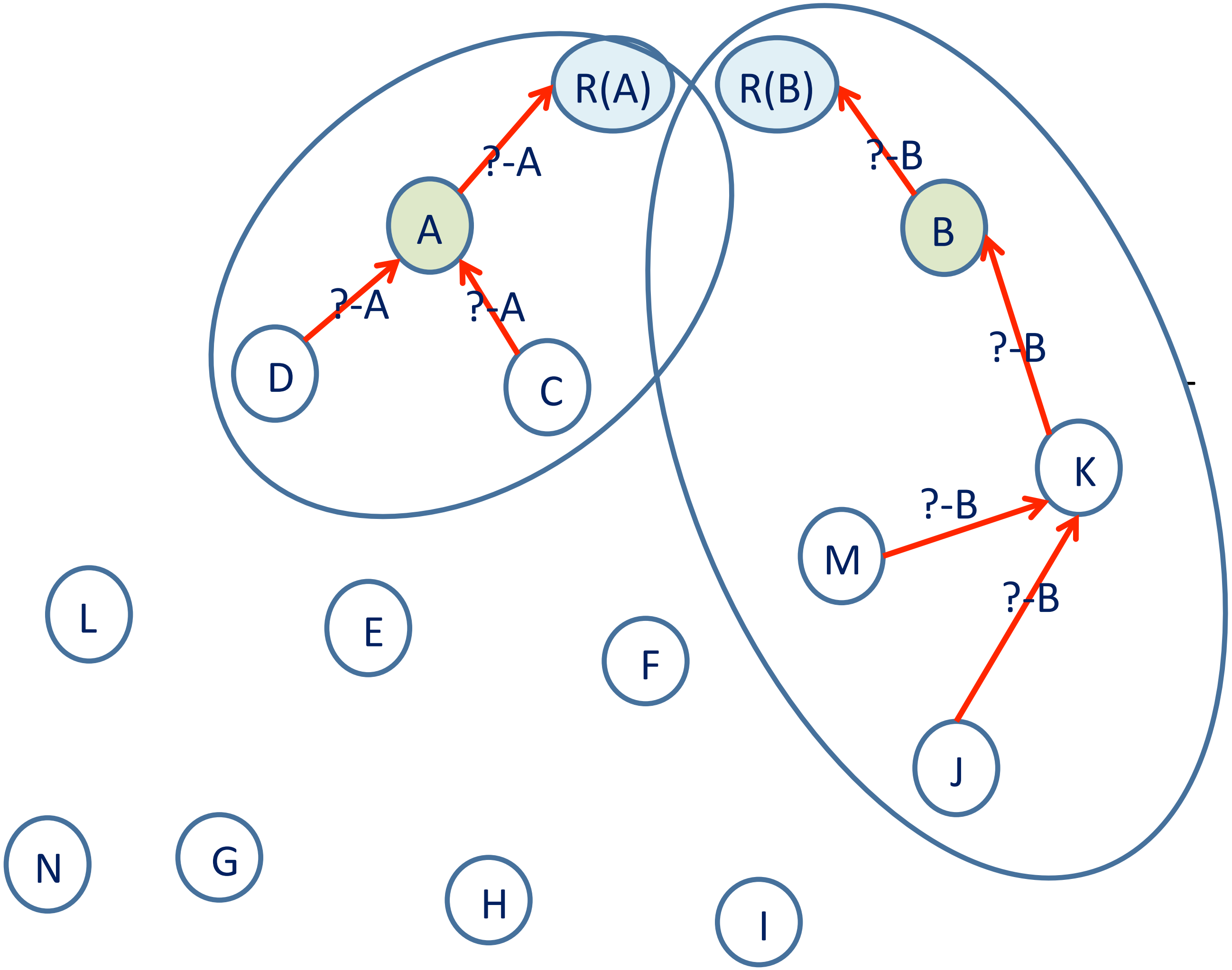
Then pick
Pick D,
Pick C,
Each time place in the parent set

Picking B:



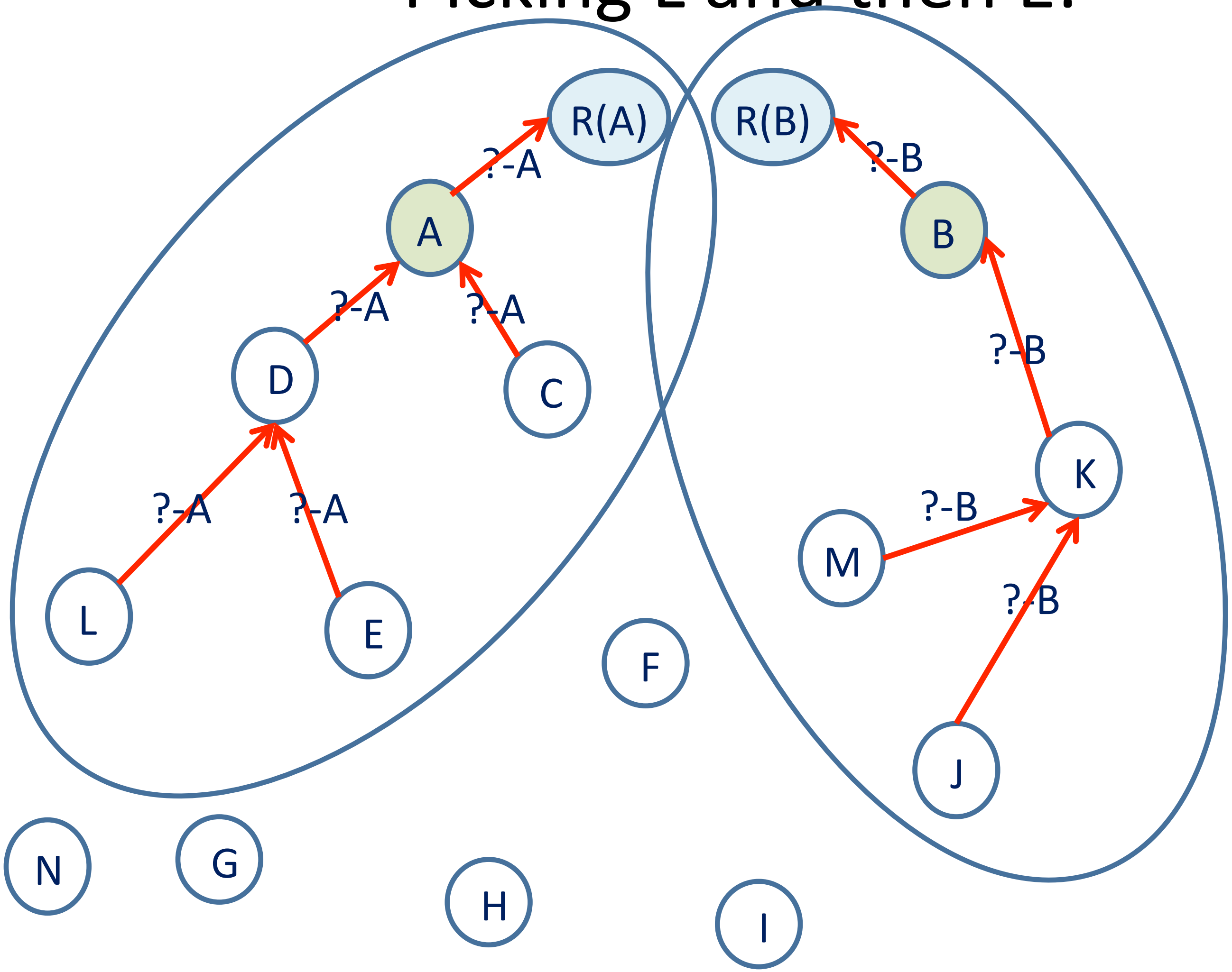
Pick K, start building up B’s dependent set

Picking M and J:

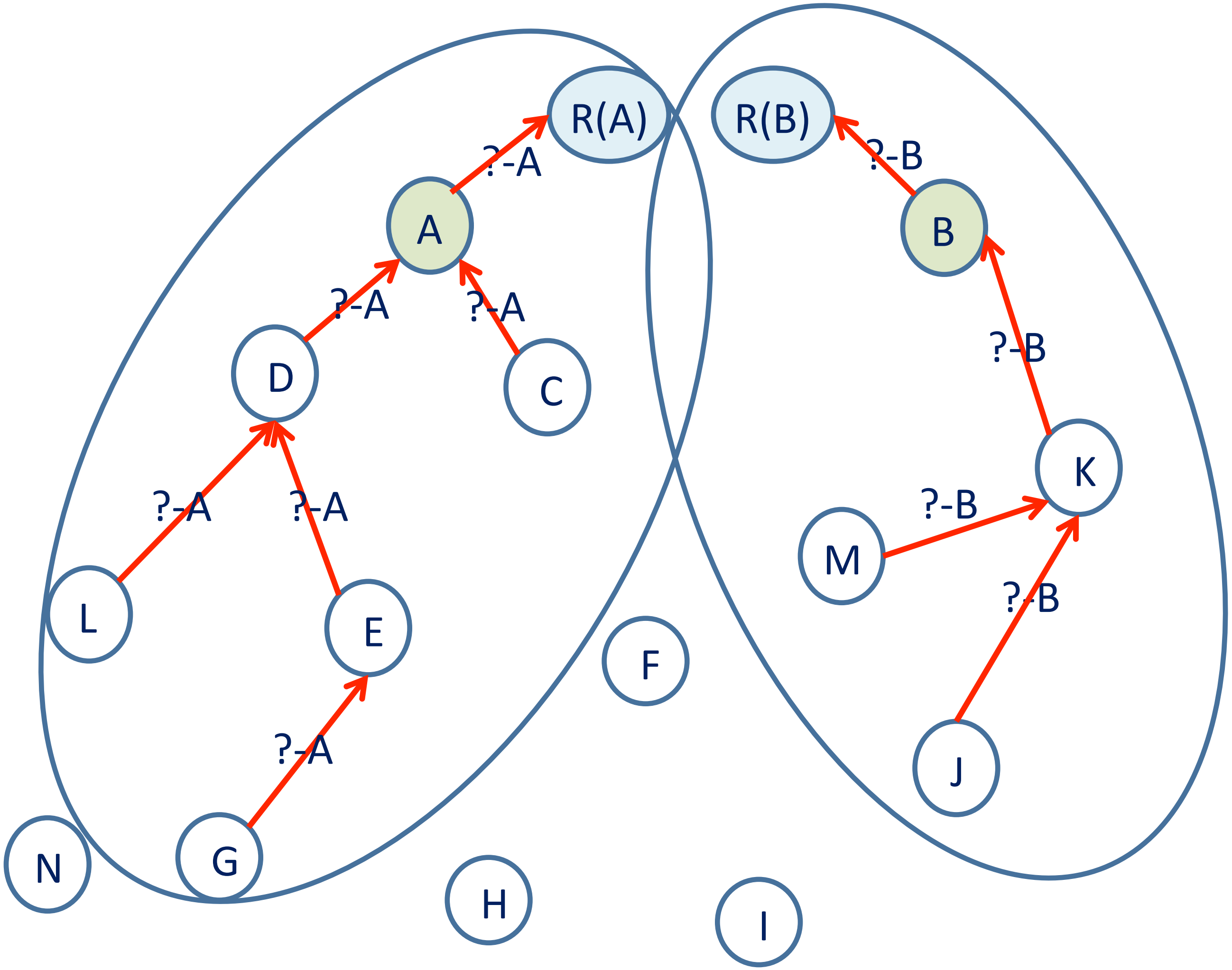


The dependent sets grow.

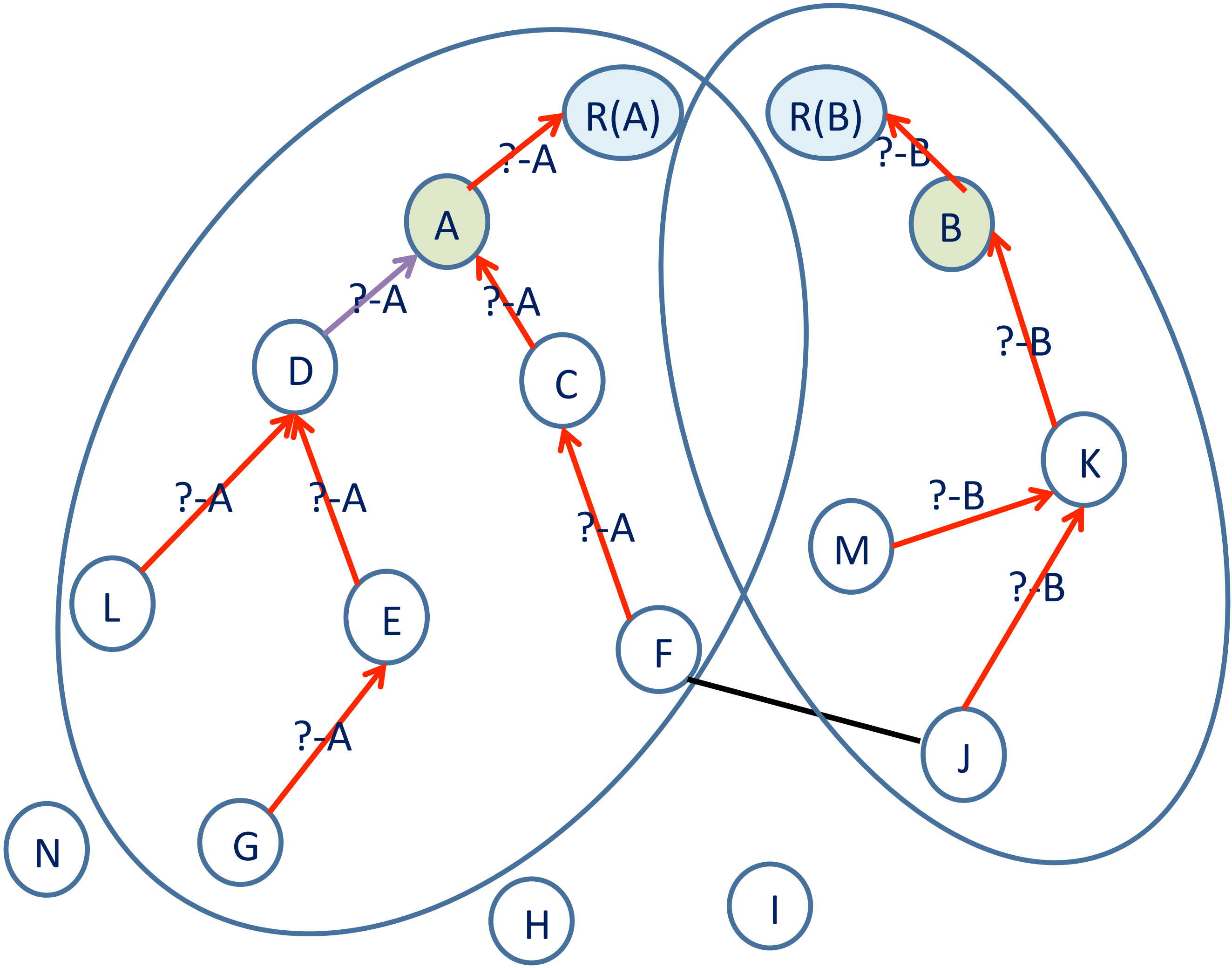
Running the Algo
Picking L and then E:



Picking G:



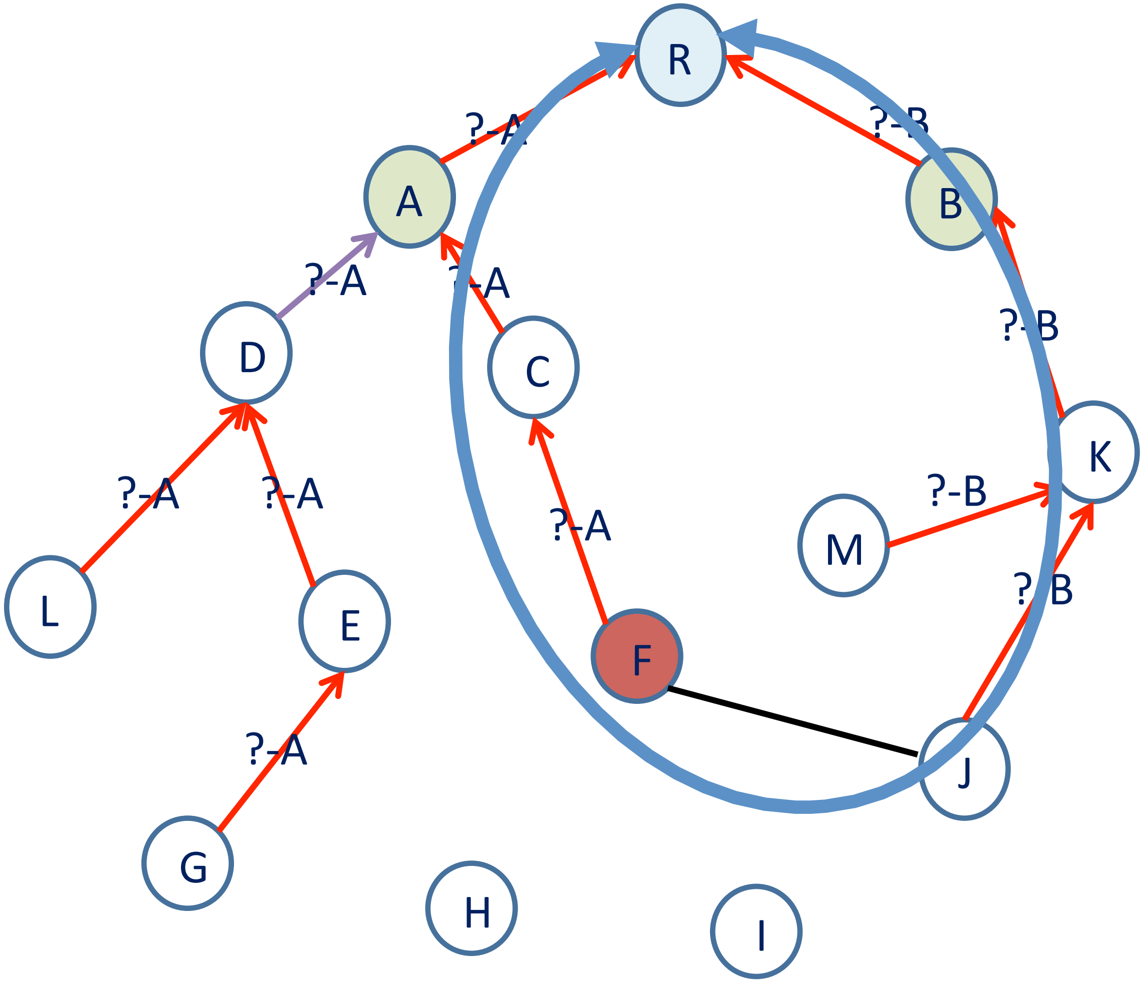
Picking F; F is a Safe node!



Examining F’s neighbors we find J that is B-dependent

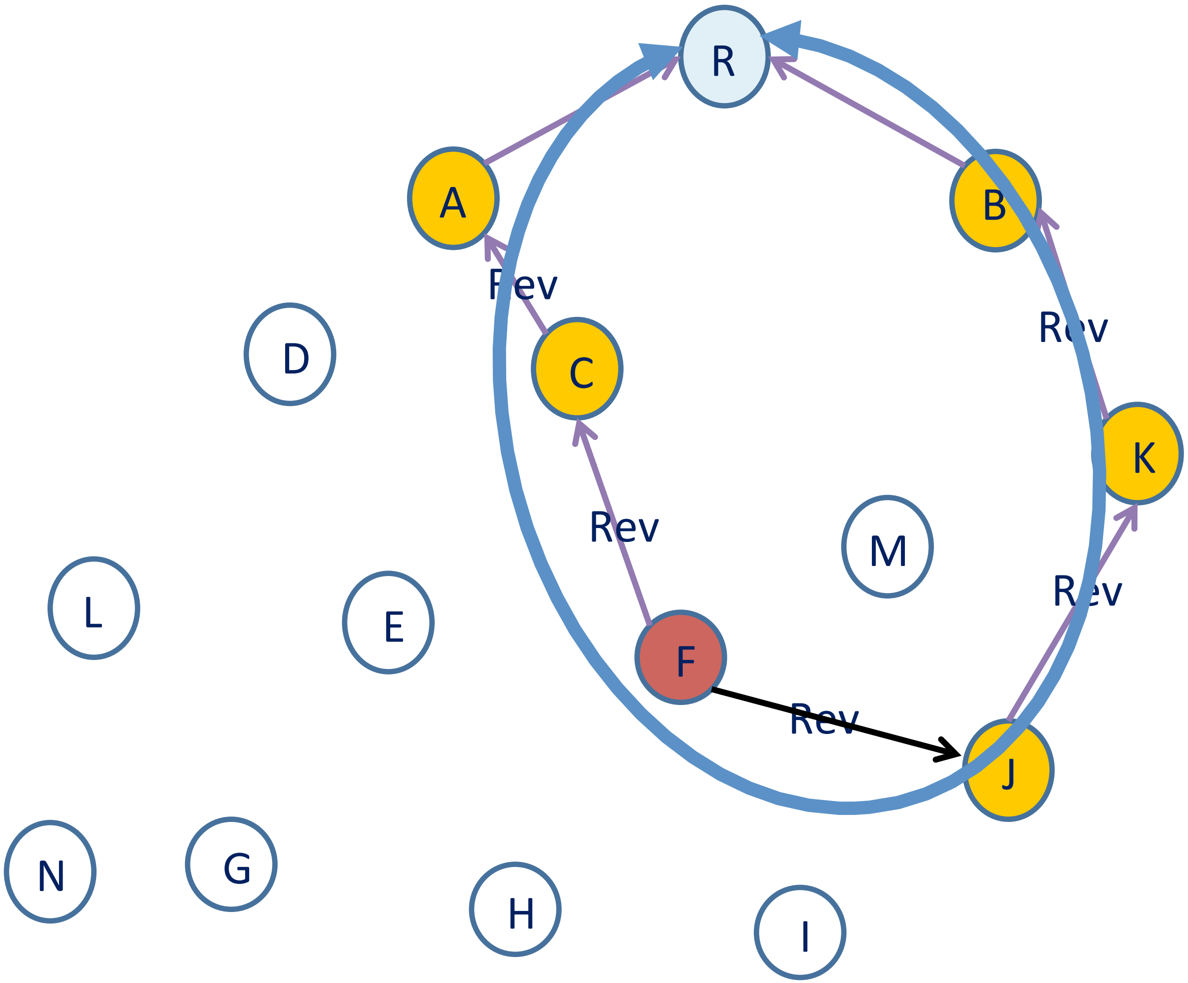
F has 2 non congruent path to 2 Safe Nodes, though virtual this time since they are R(A) and R (B)

We can form the first infrastructure ARC!



We can use F-J to tie F’s shortest path to R (A) with J’s shortest path to R(B)

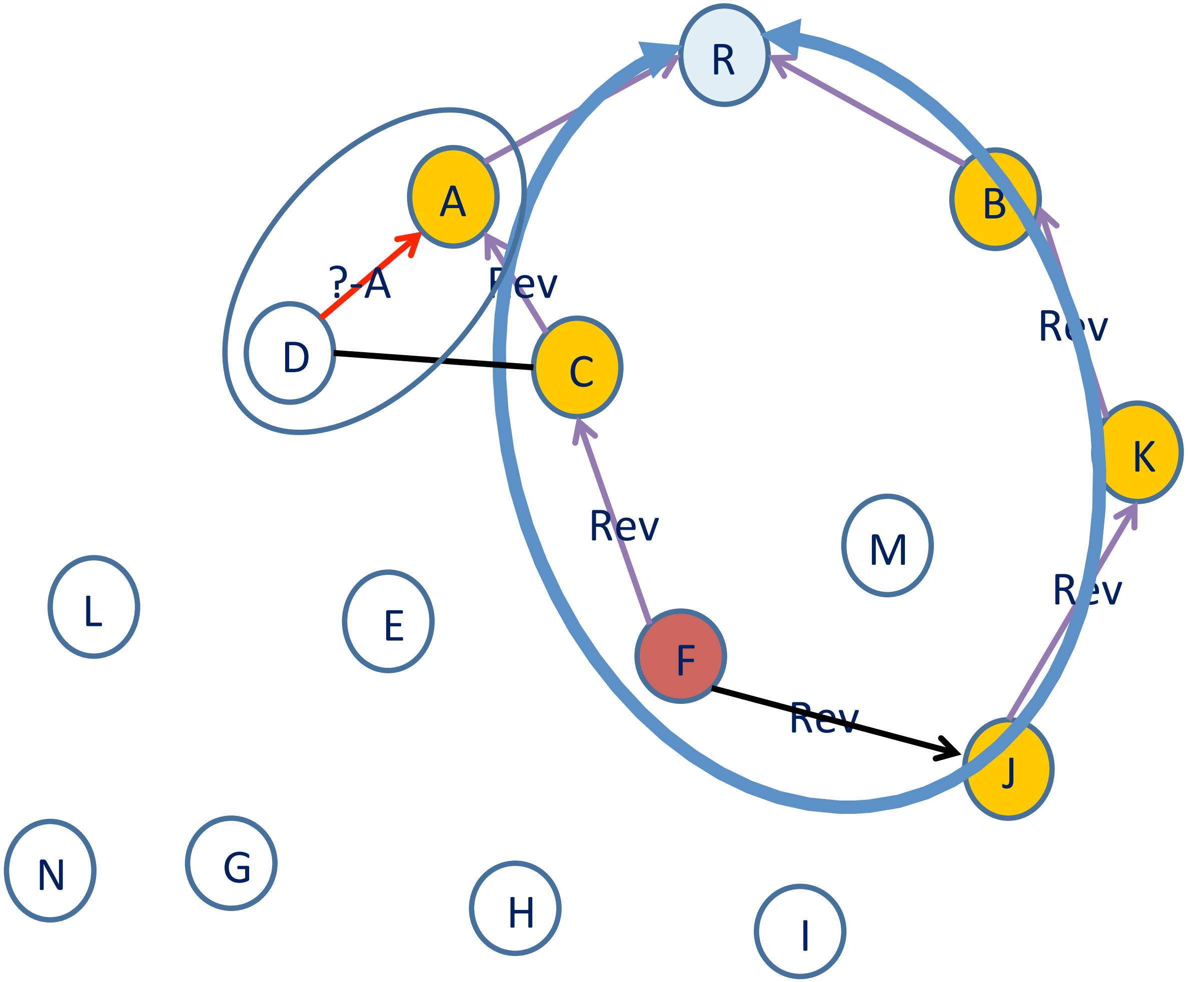
All nodes along the ARC are Safe



Nodes along the ARC are placed alone in there own dependent set
(not represented)

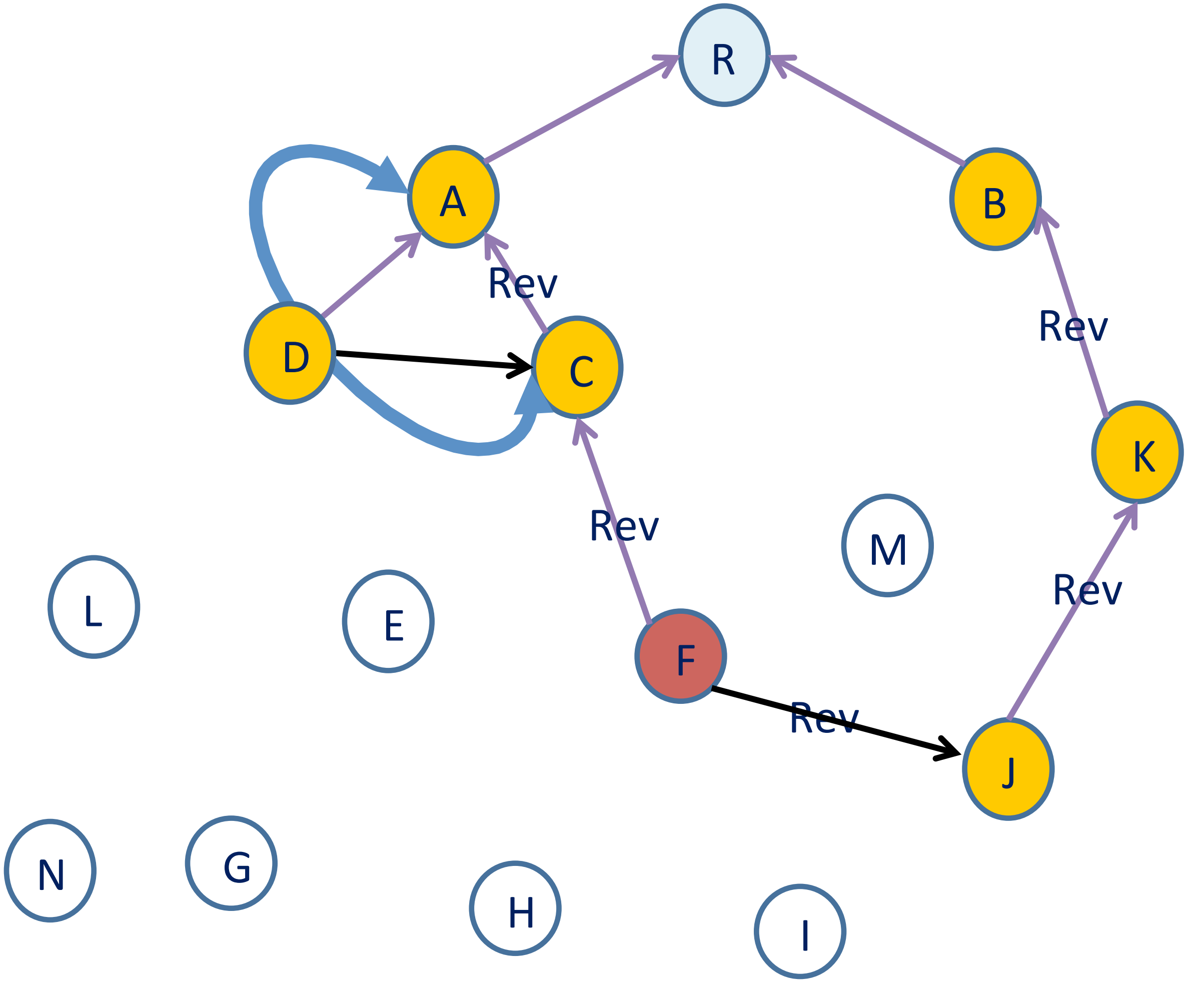
All other nodes are returned to the original set

Next is D



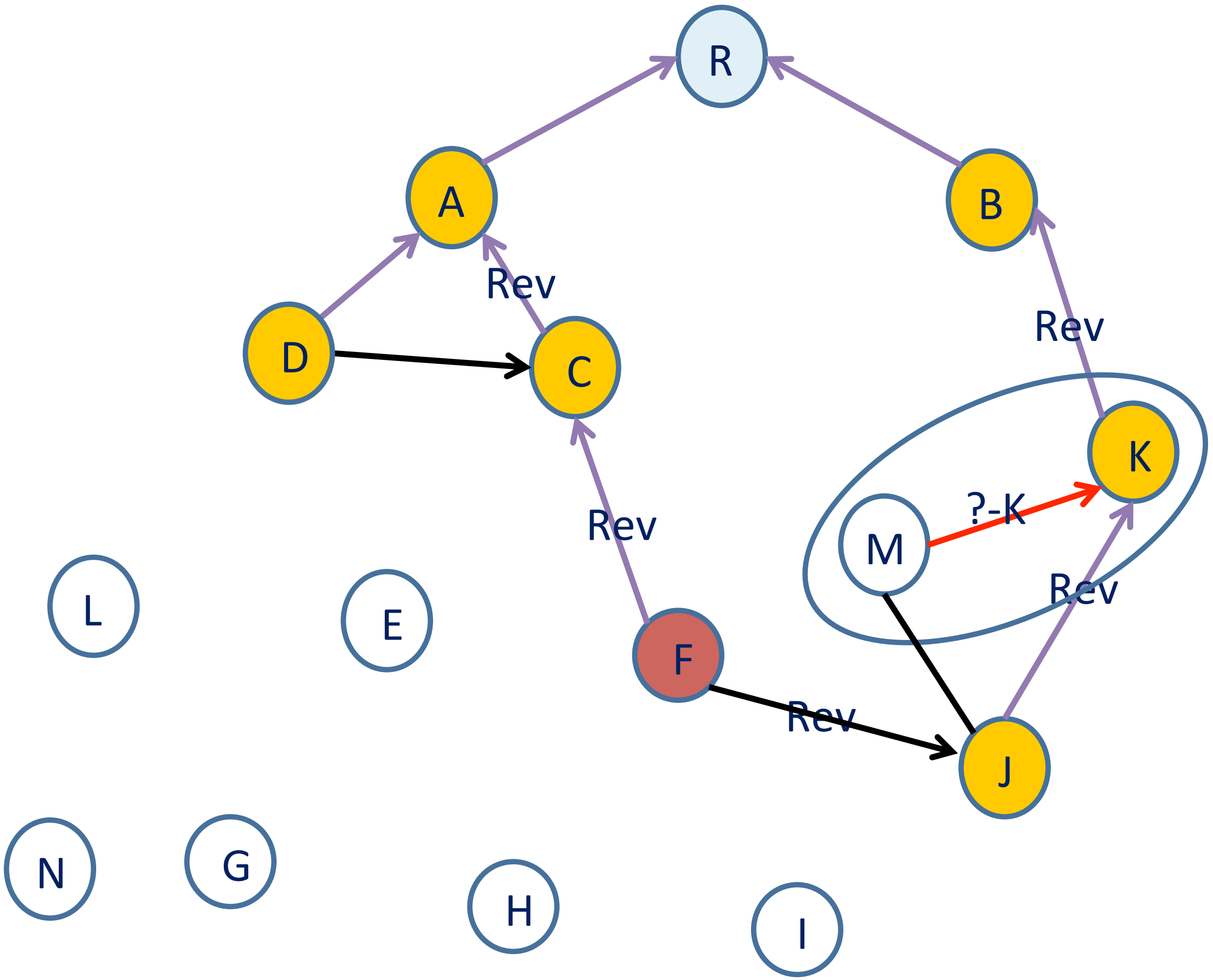
- D depends on A
- D can reach C which is in another set

D is a collapsed ARC



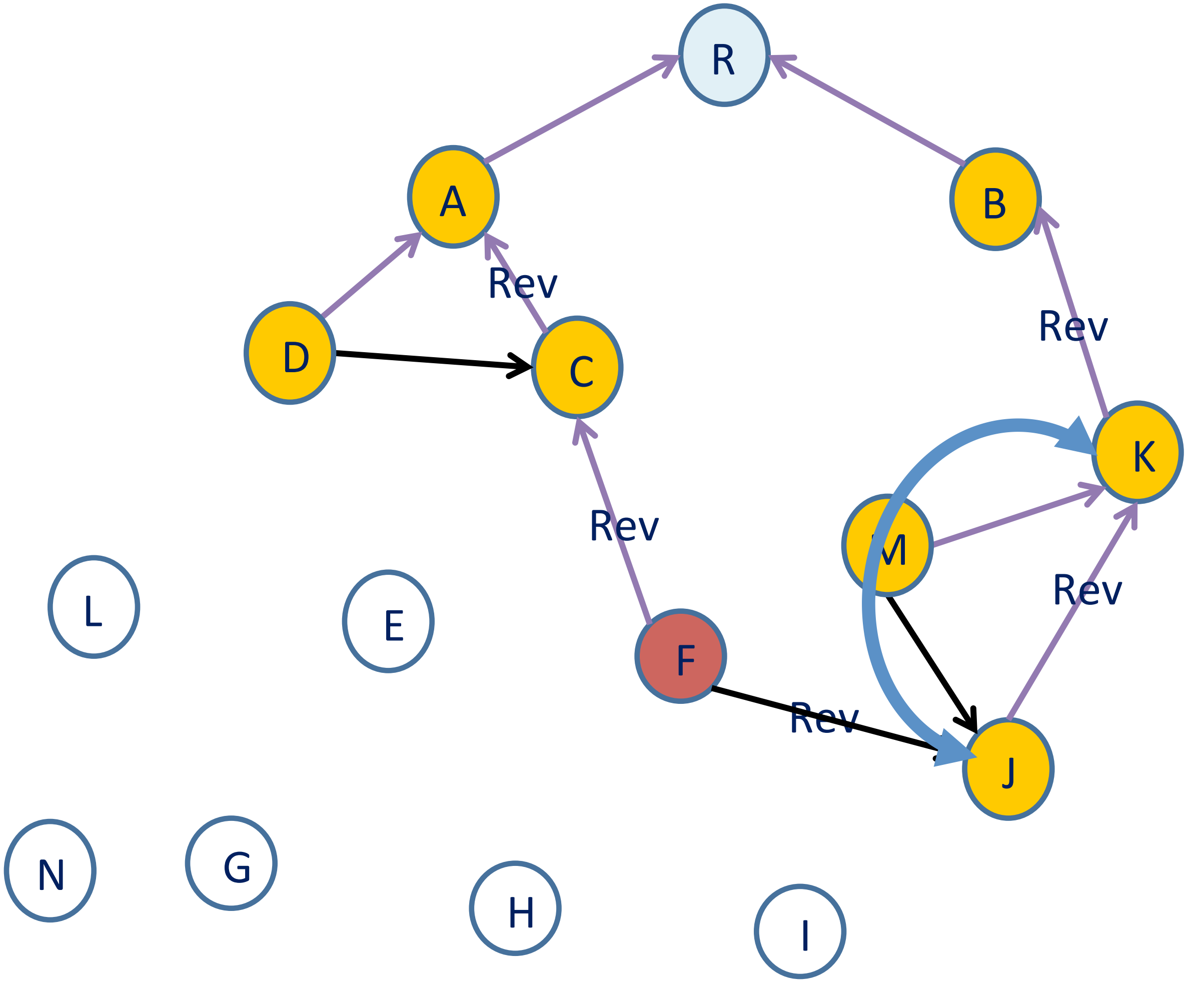
D’s parent A and D’s preferred neighbor C are both Safe Nodes

Next is M

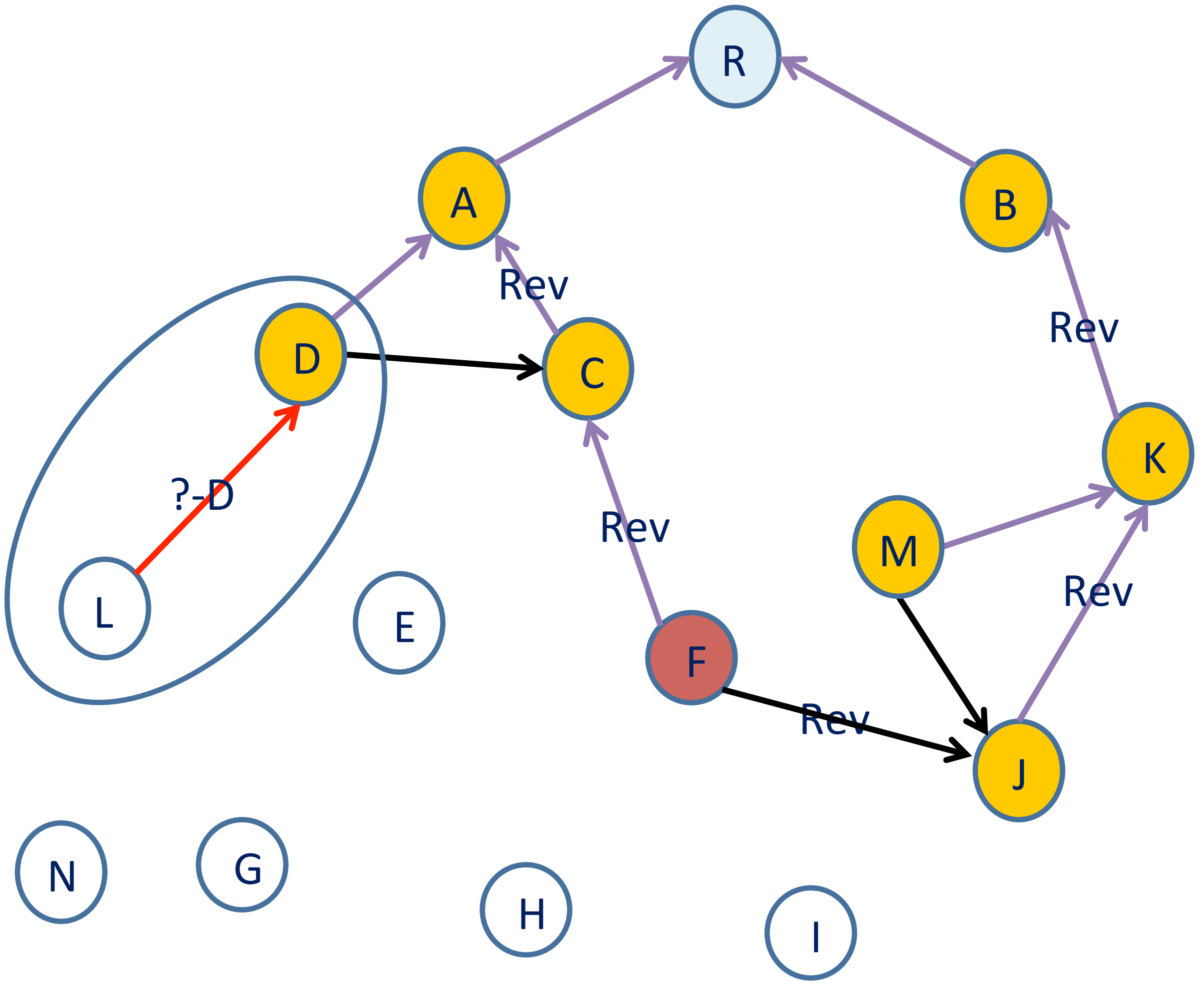


- Same goes for M

M is a collapsed ARC

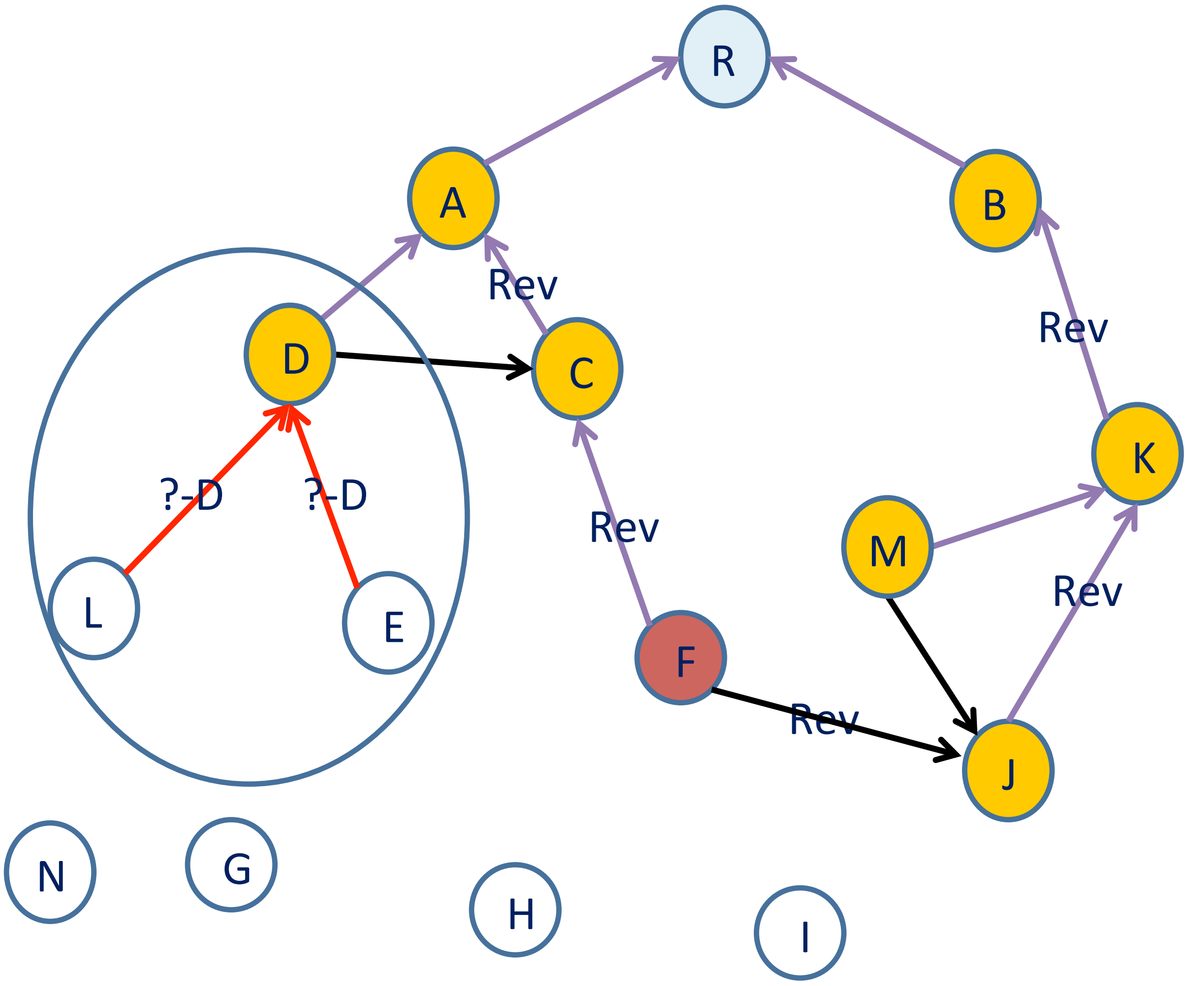


Picking L



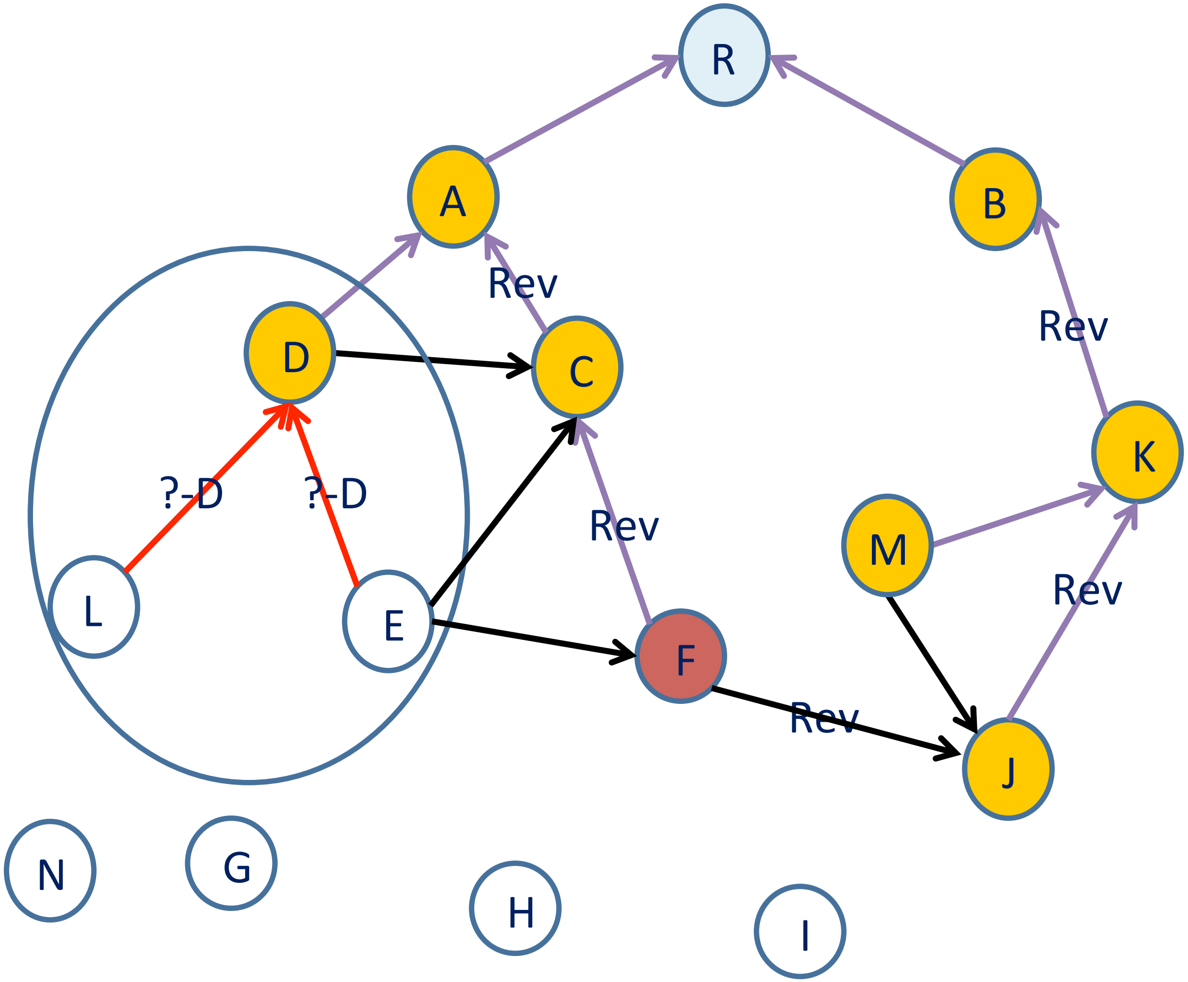
All depend on D at this point

Picking E



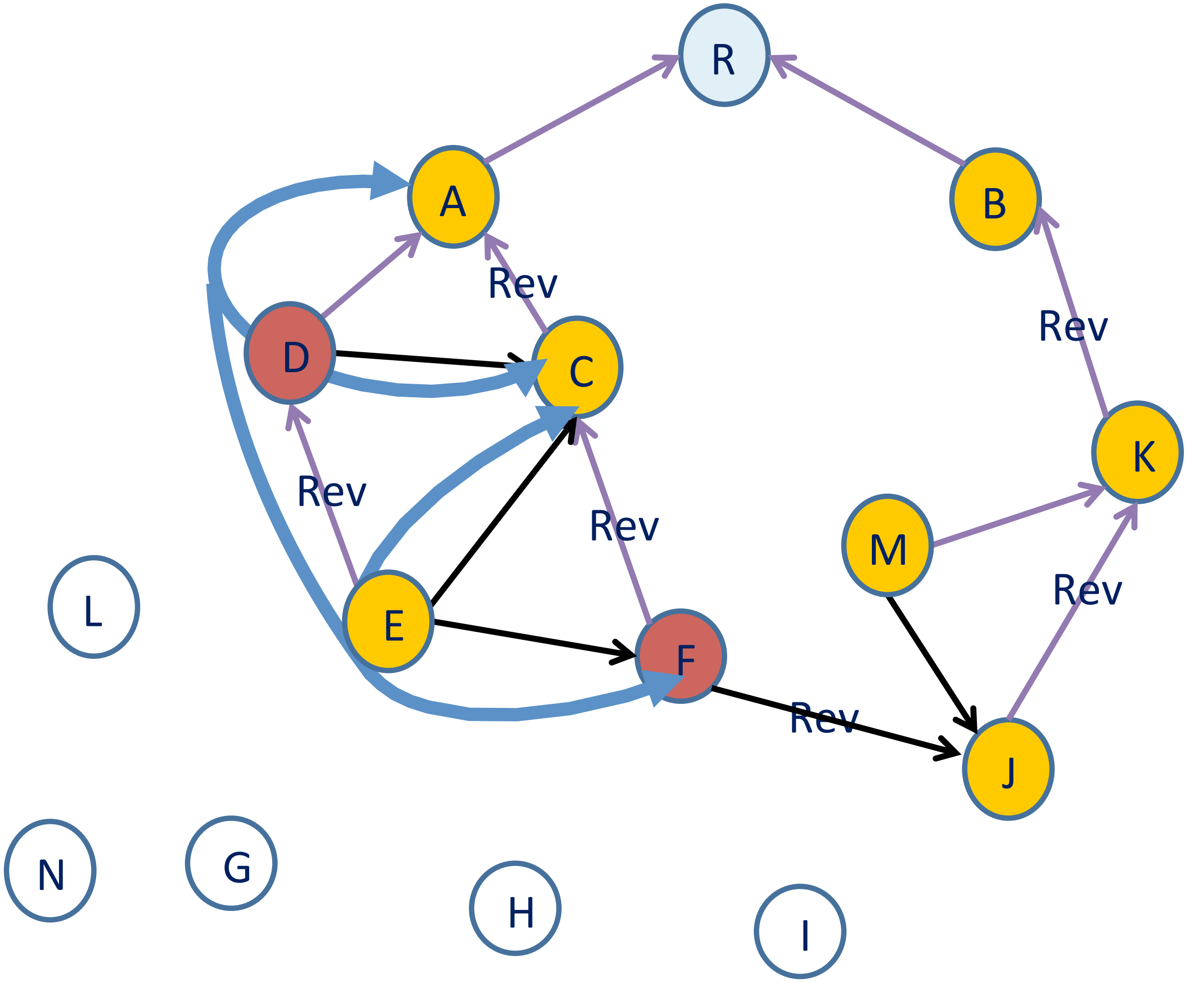
All depend on D at this point

E has links to C and F



E has links that end deeper than D's collapsed ARC

E adds a buttressing ARC



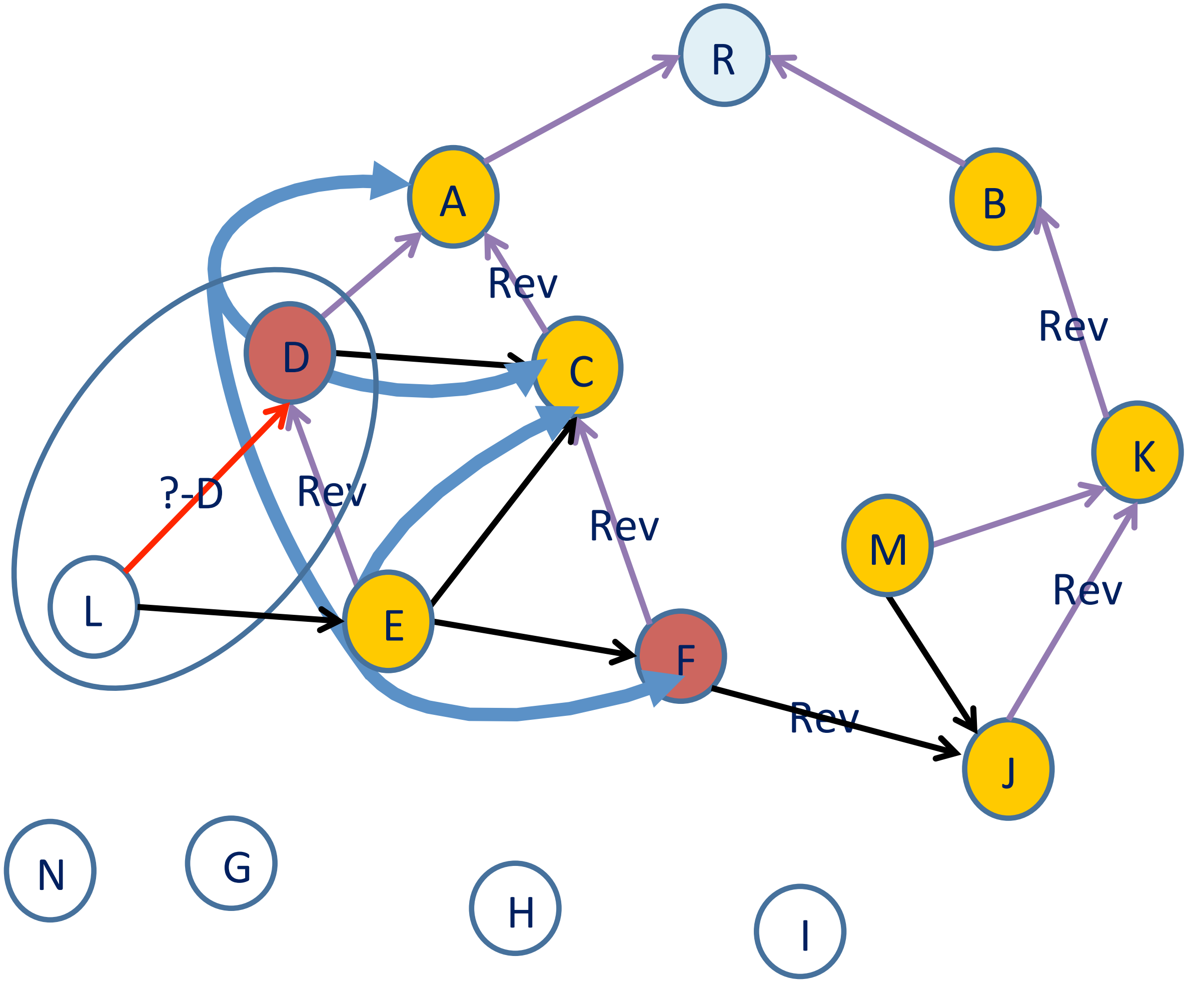
We can form a buttressing ARC keeping E’s links that end deeper than D’s collapsed ARC

E->D becomes this reversible

L returns to the set

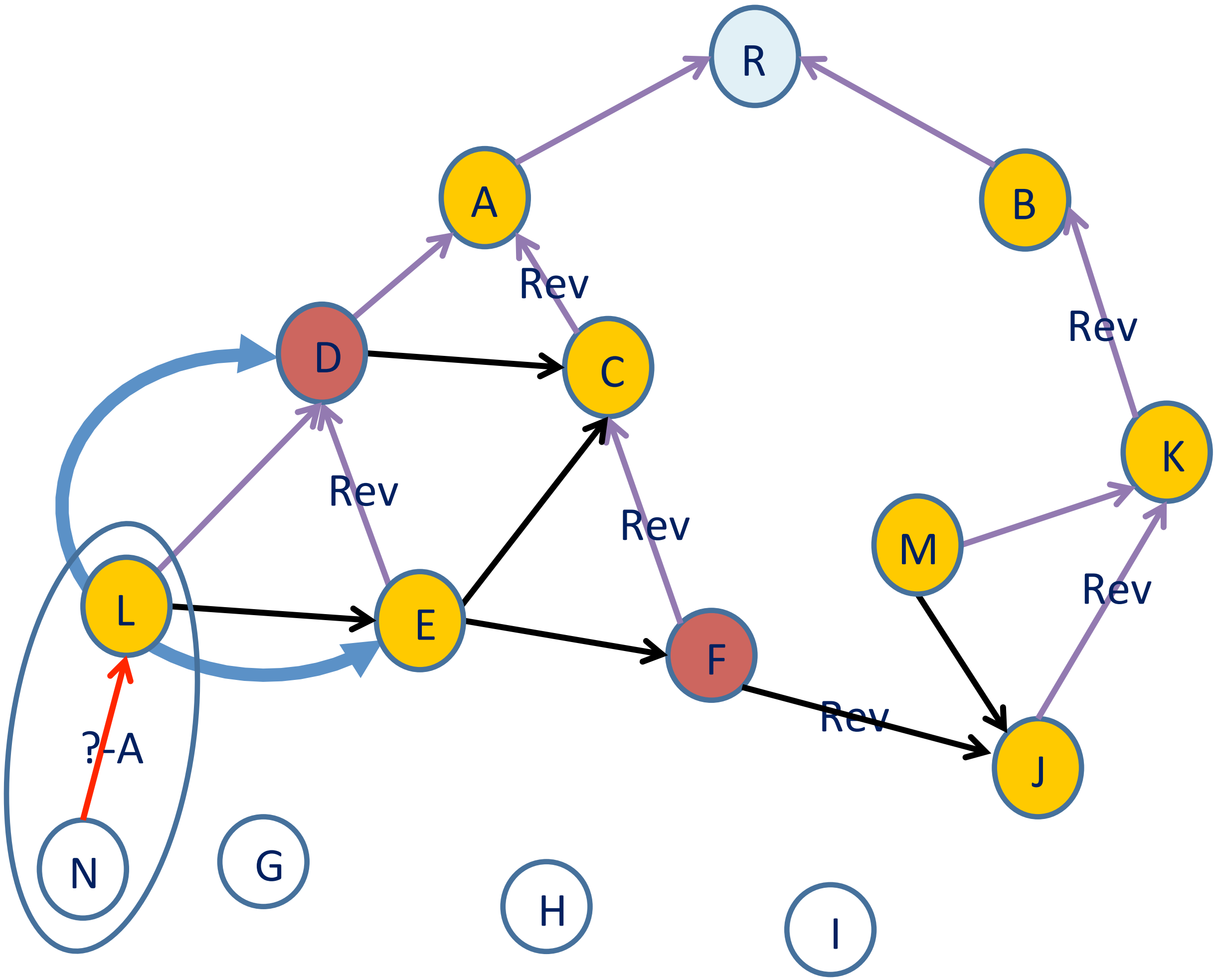
D being the Cursor of the origin ARC is cursor for the Comb

Picking L

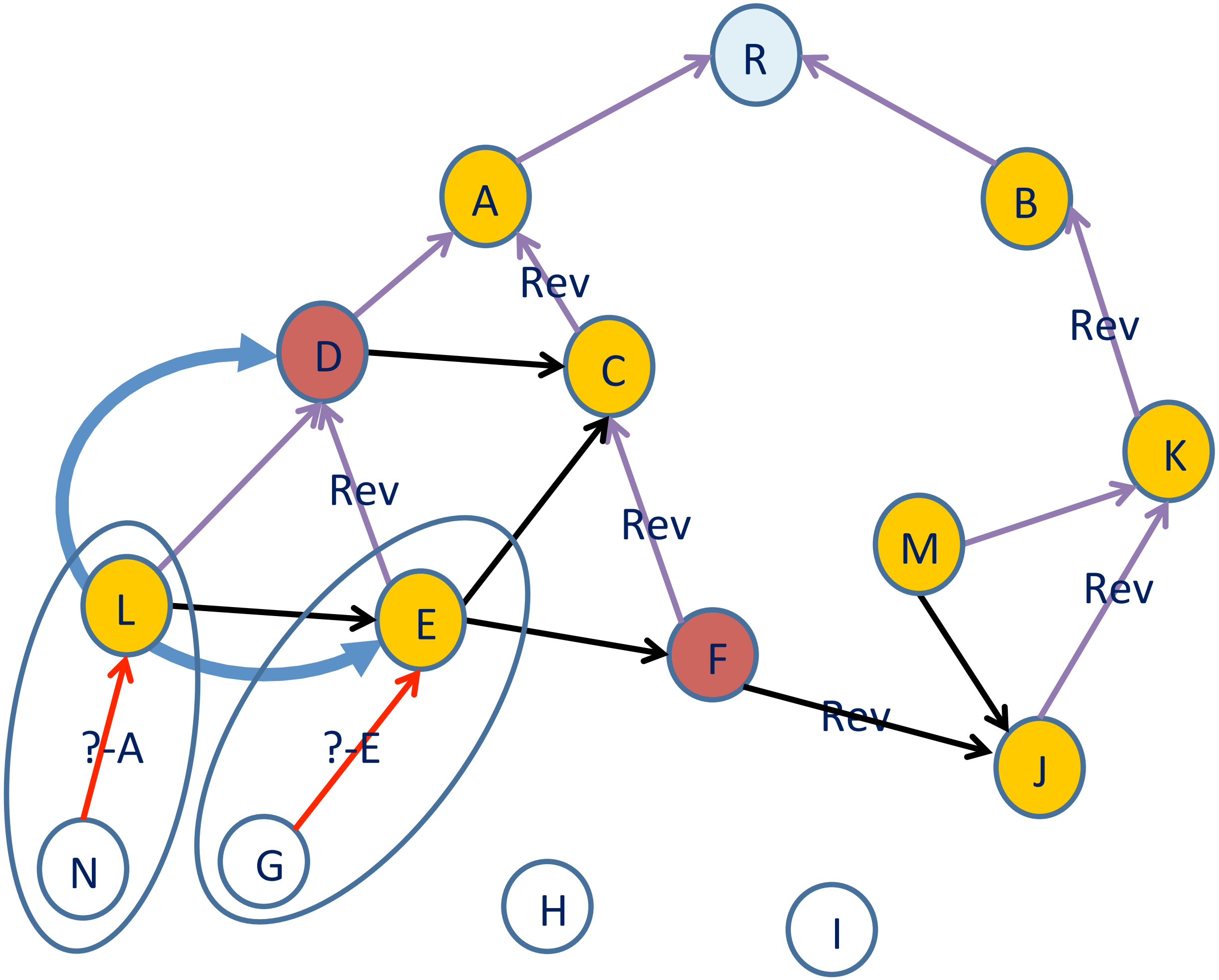


L forms its own collapsed ARC

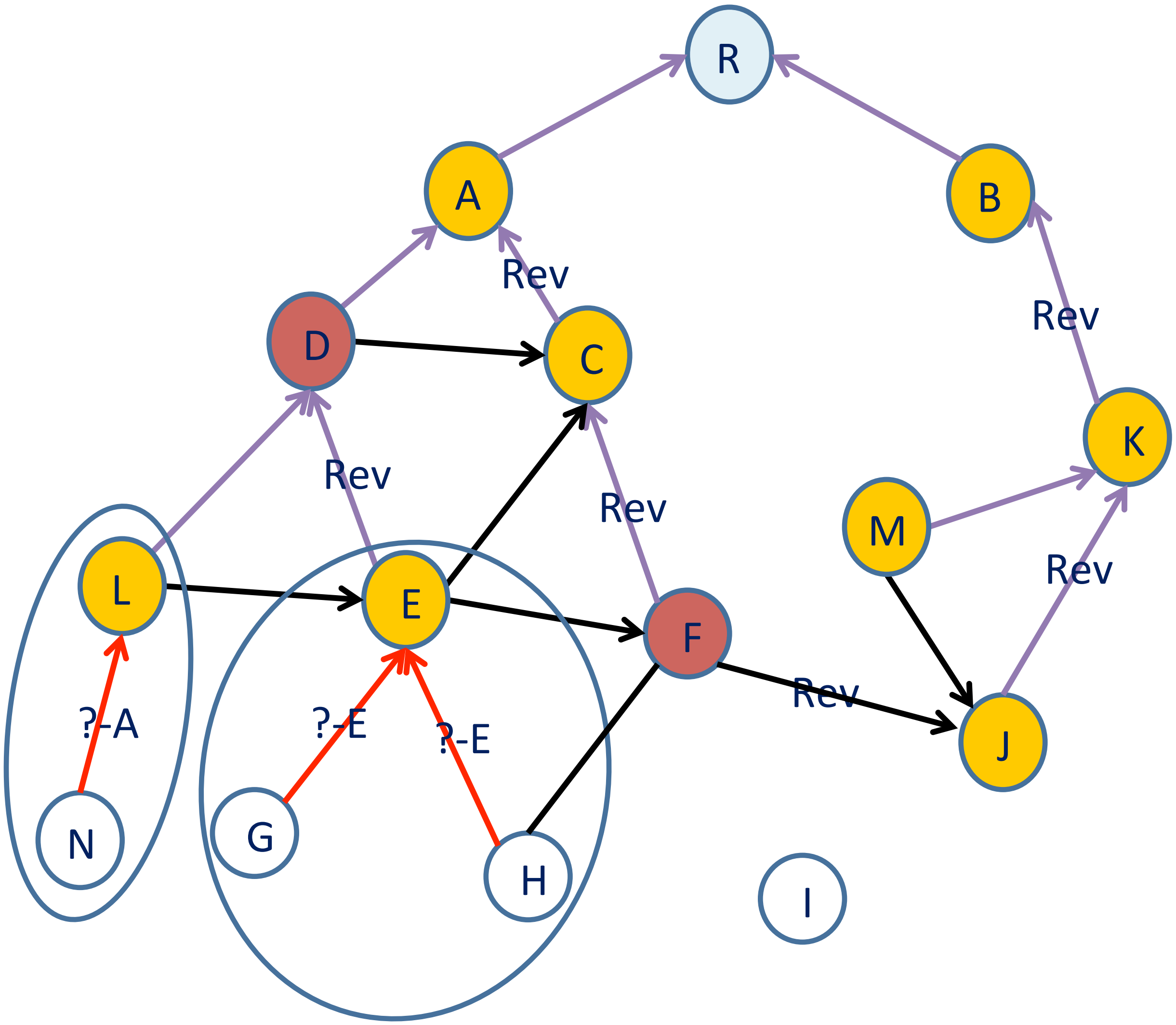
Picking N



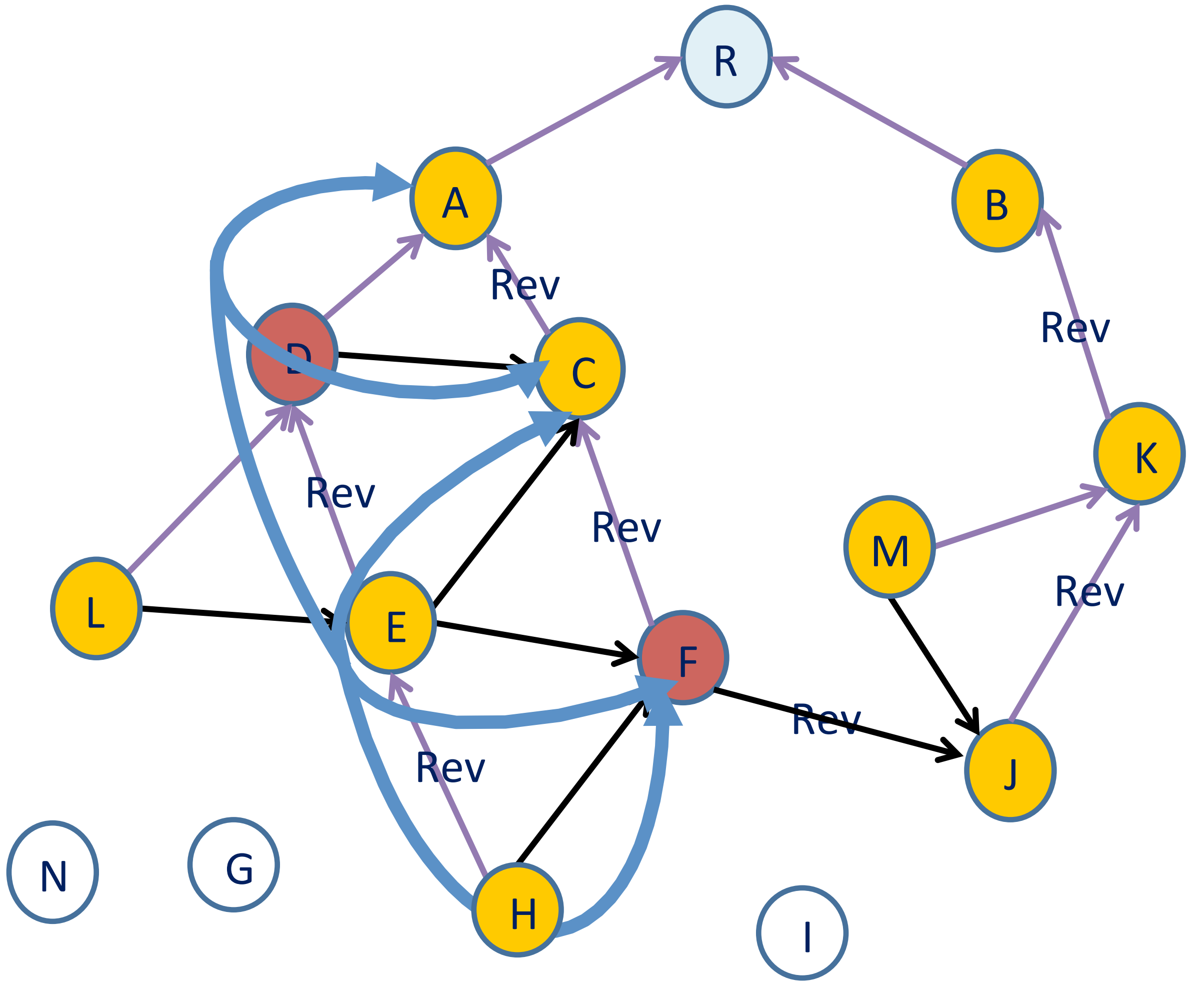
Picking G



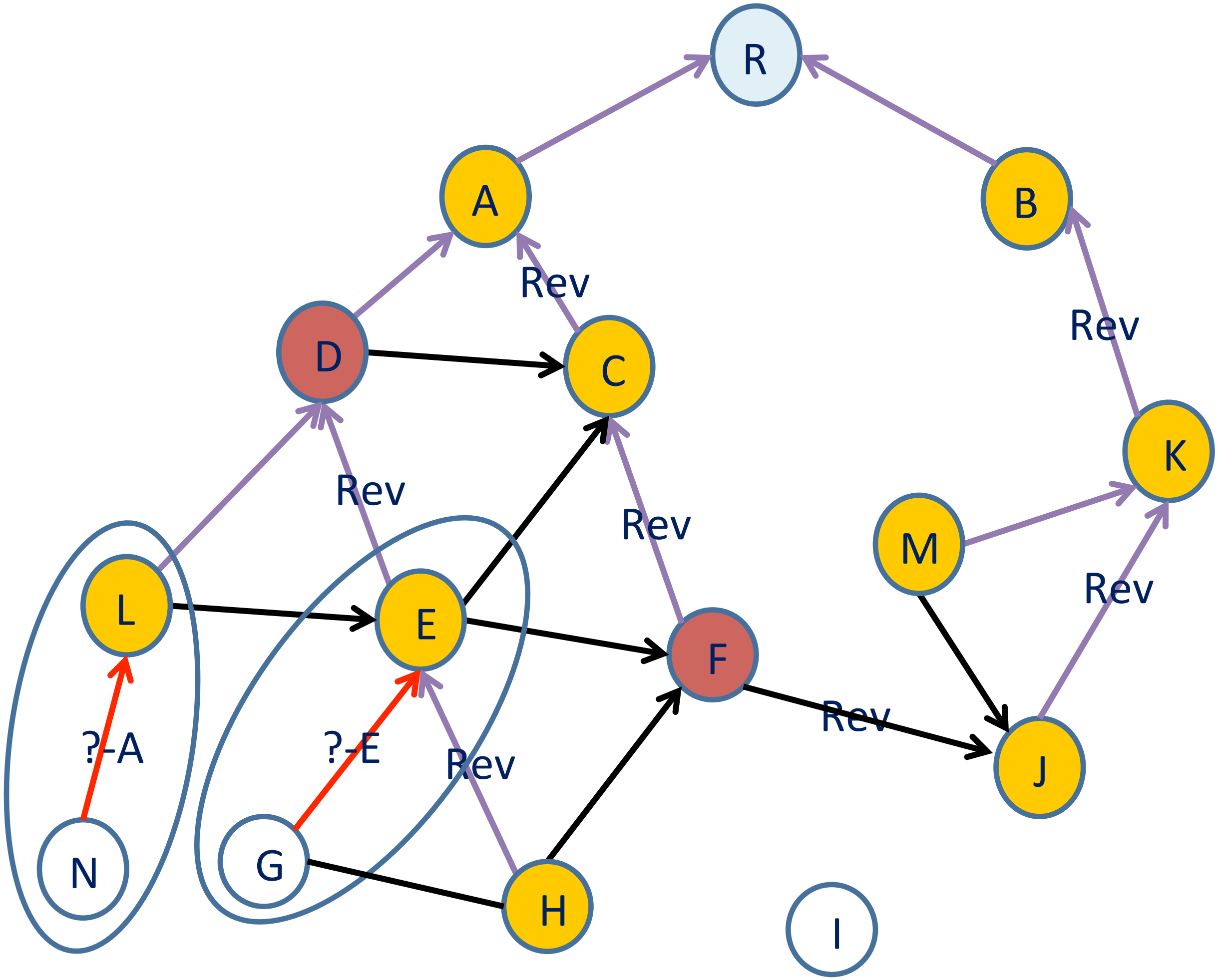
Picking H



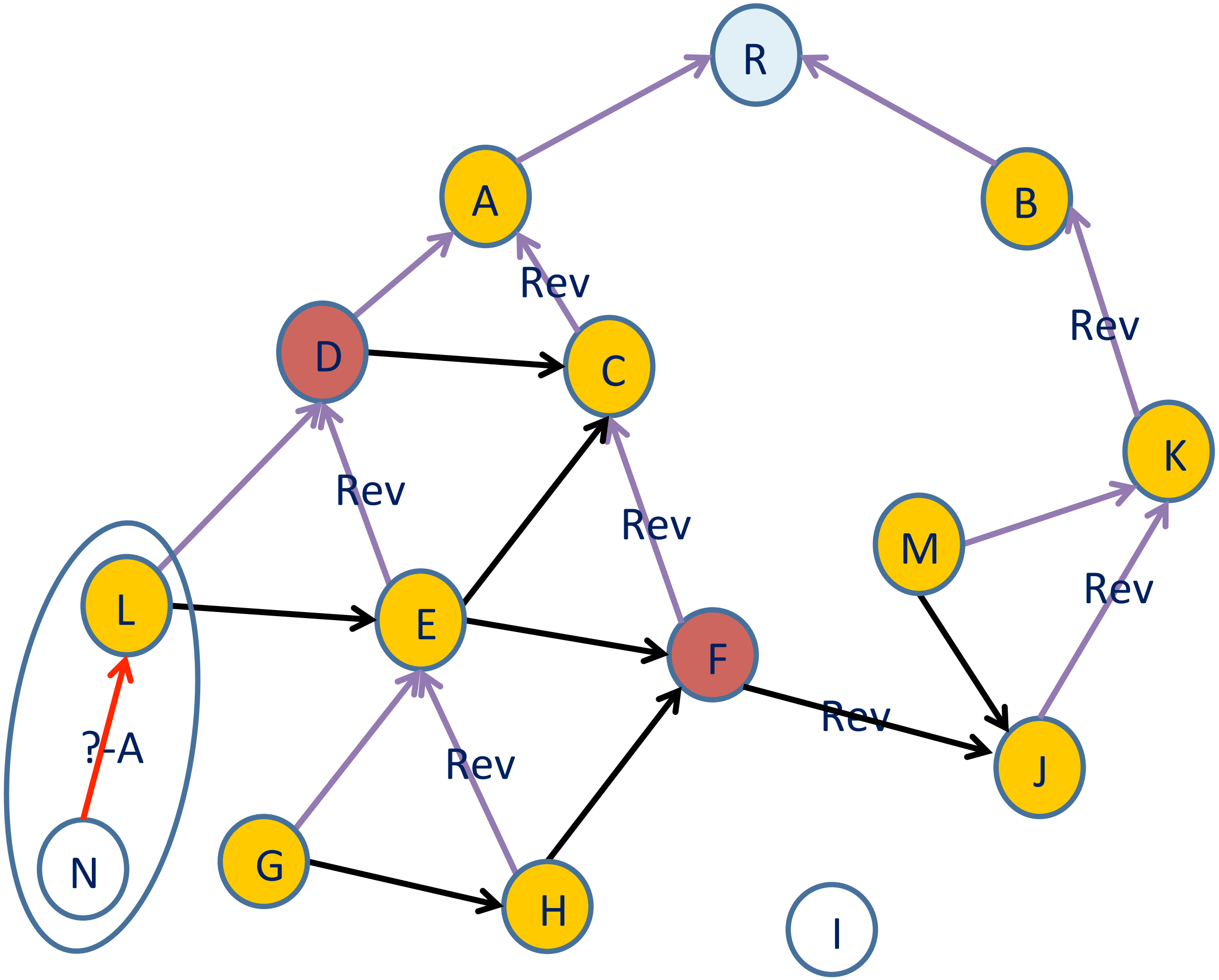
H adds a buttressing ARC



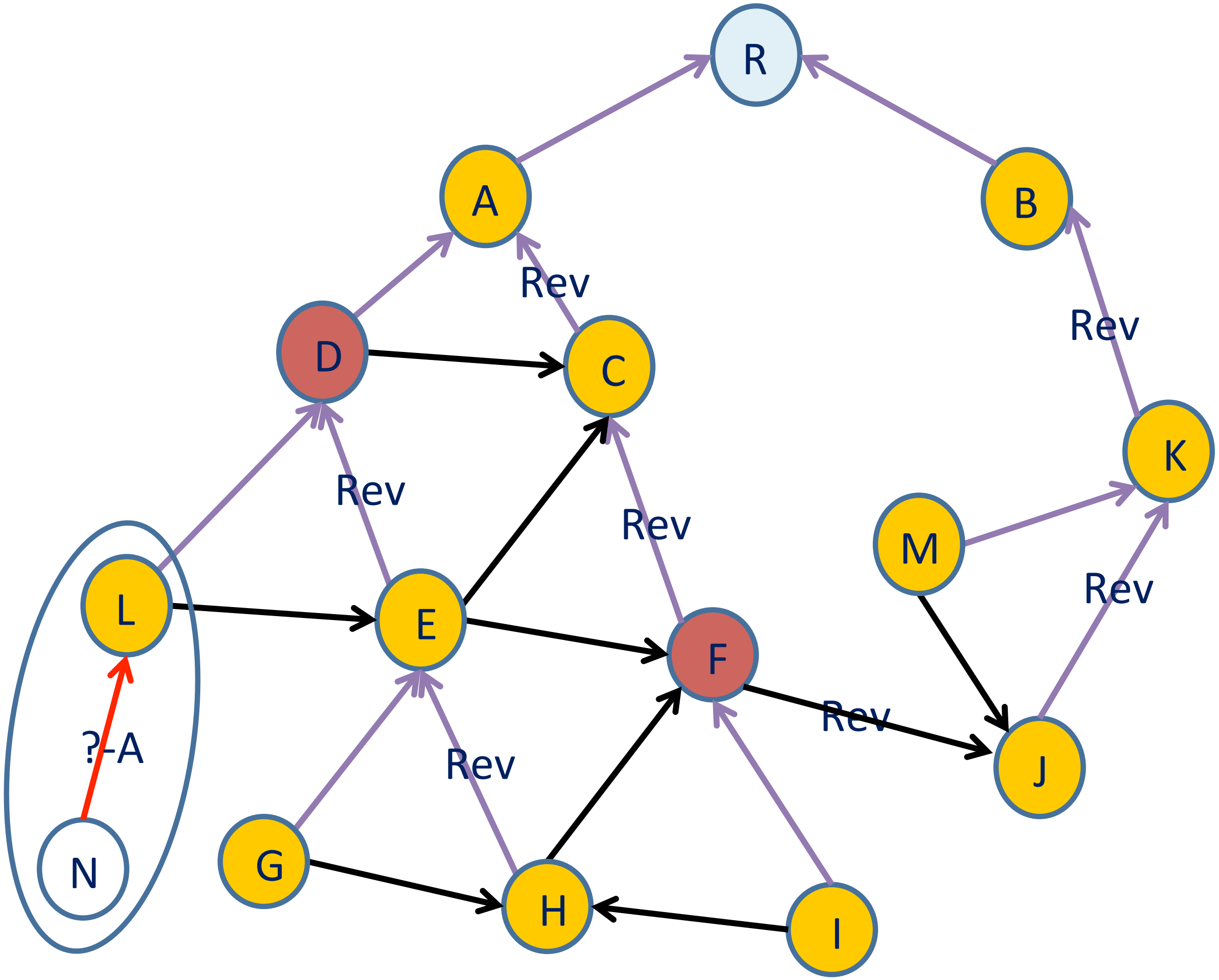
Picking N and G again



Picking N again



Picking N again and then I



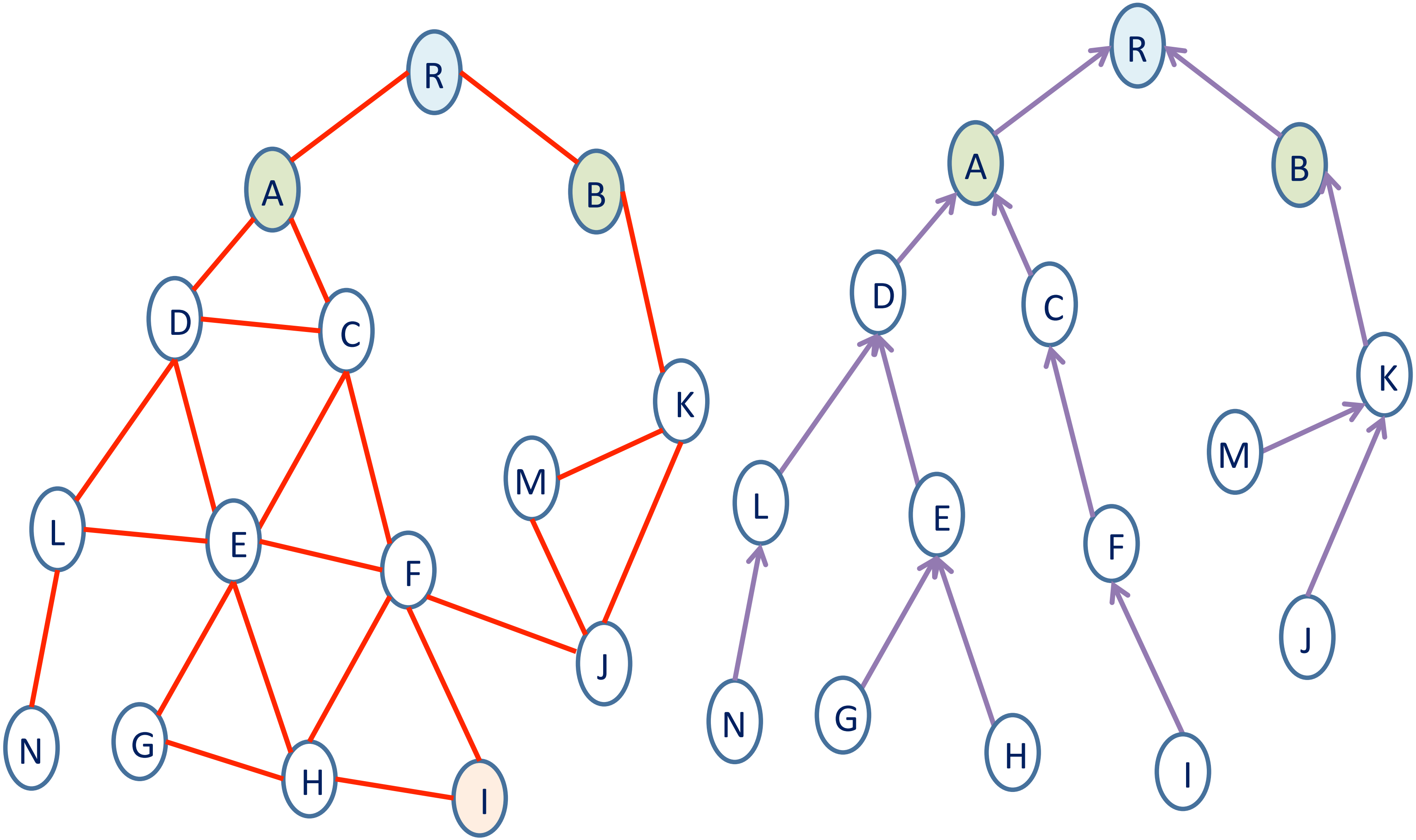
We’re done with the set

N is still dependent

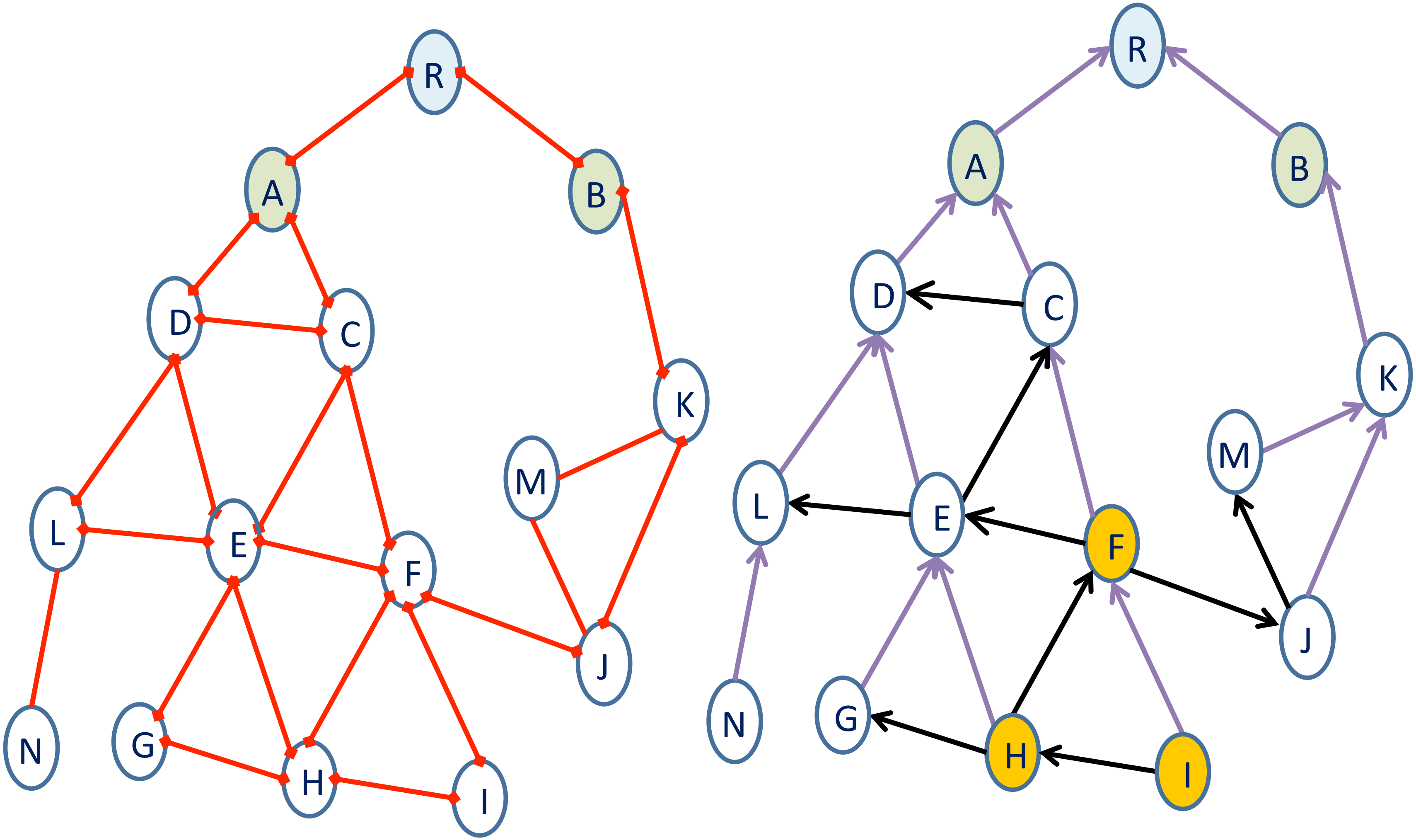
N’s subgraph is monoconnected

If N has a dependent set we run the algorithm in that set using N as root.

Original Graph and Classical rev-SPF

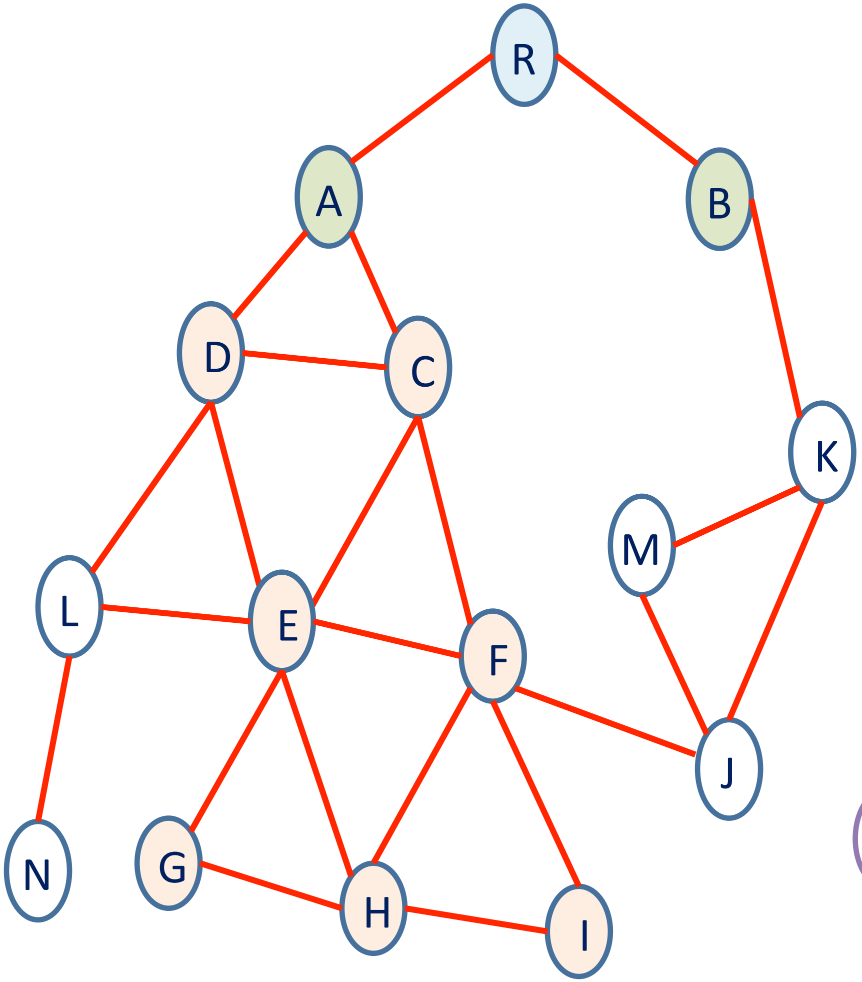


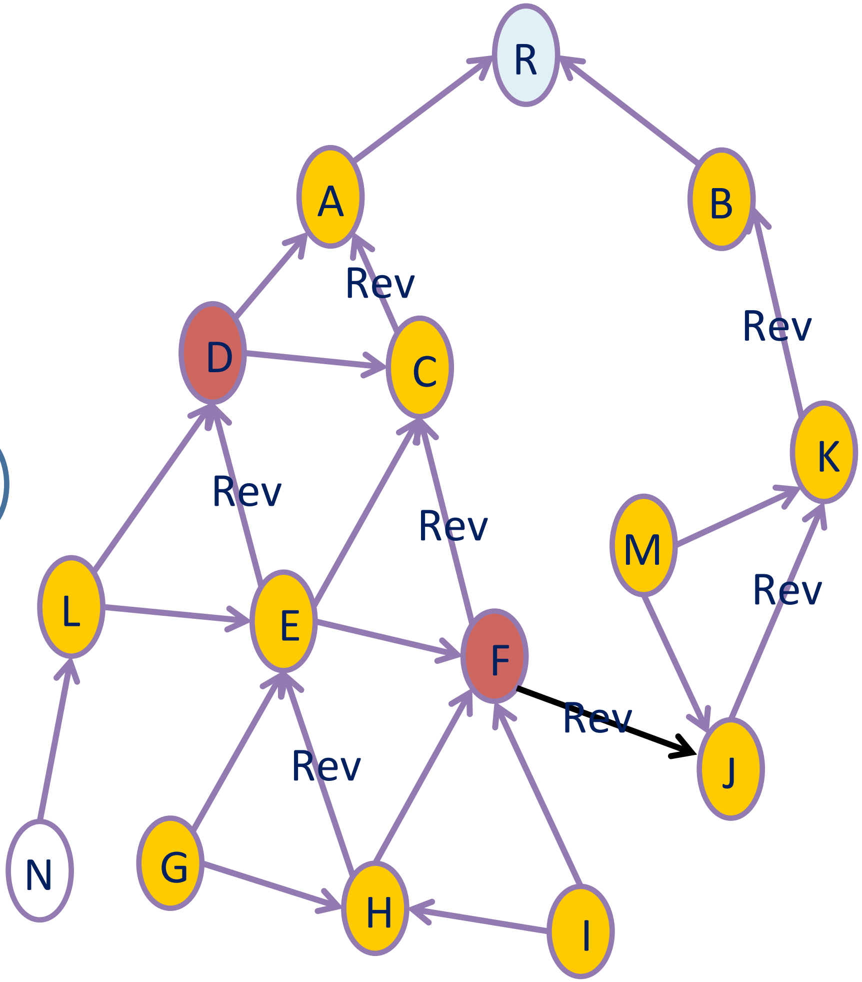
Original Graph and SPF-based DAG



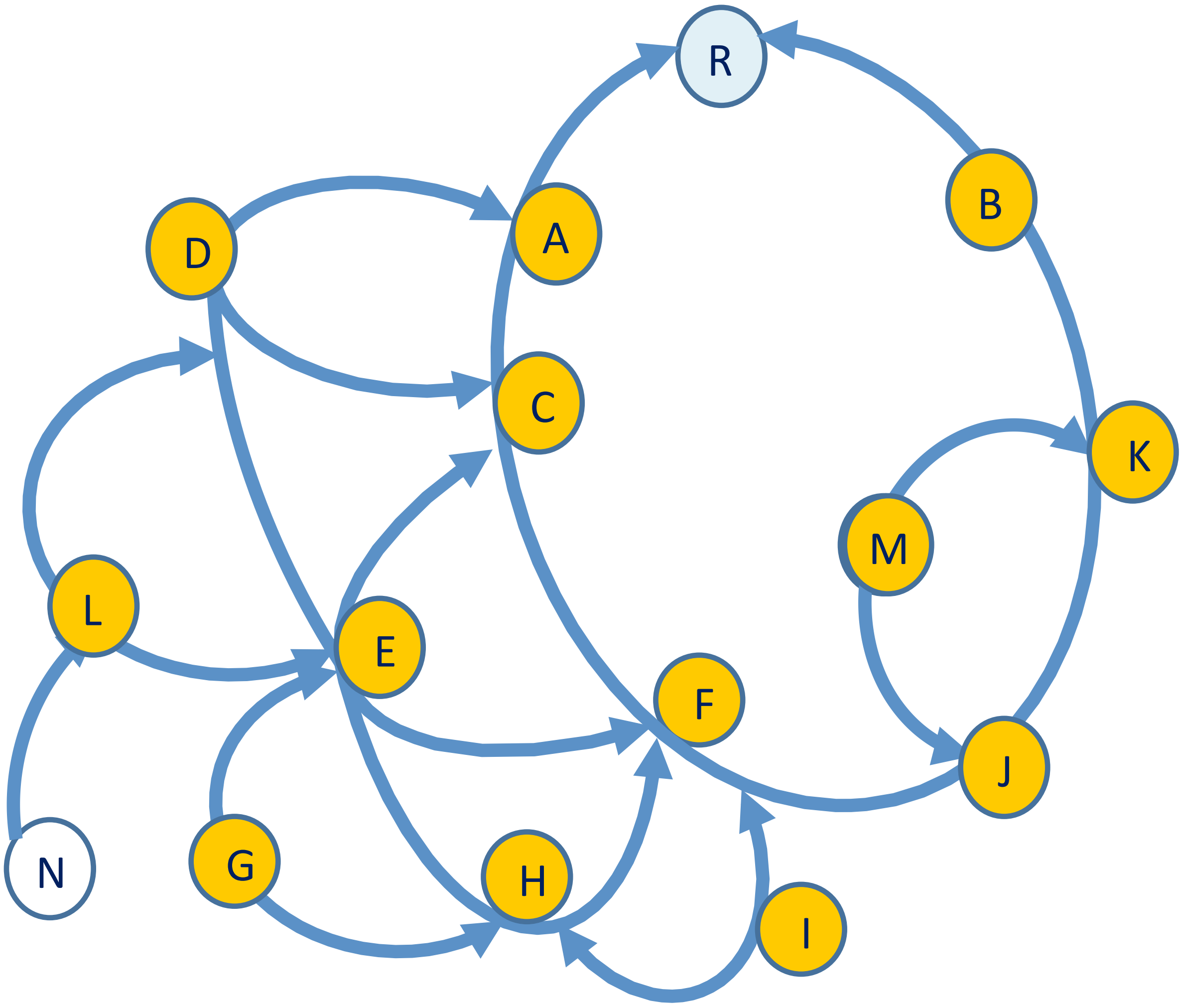
Only 3 nodes are Safe but in all cases packet end in Single point of failure waterbasins

Original Graph and resulting construct





Constructed ARCs


R
D
A
B
C
K
M
L
E
F
J
N
G
H
I