# FT Computation (FTC) Algorithm

## draft-cc-lsr-flooding-reduction-04

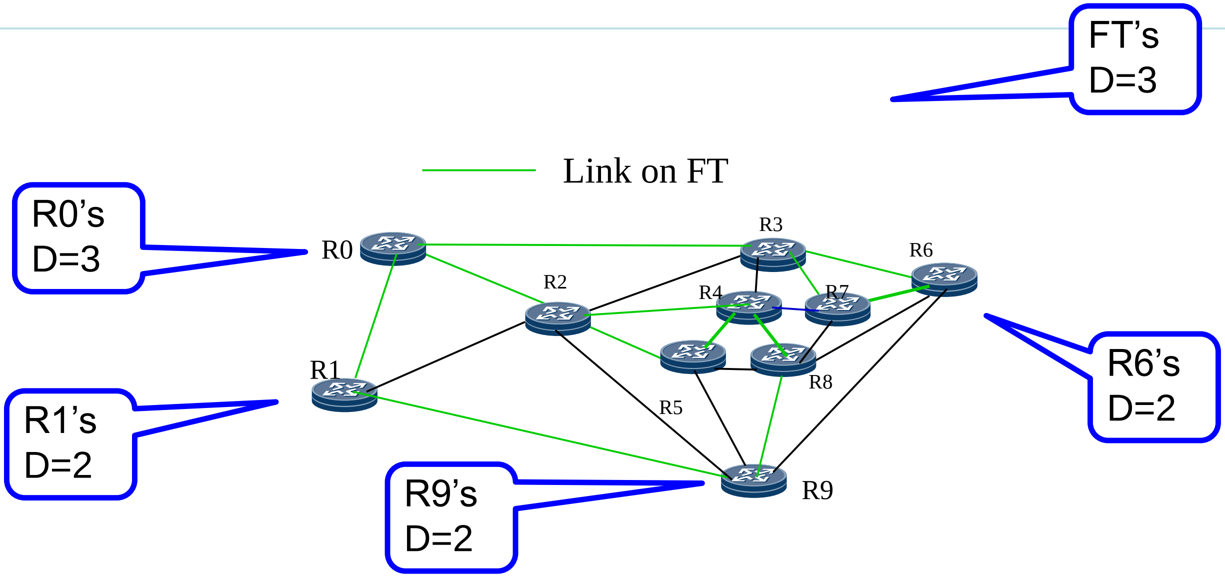
Huaimo Chen (huaimo.chen@futurewei.com)
Dean Cheng (deanccheng@gmail.com)
Mehmet Toy (mehmet.toy@verizon.com)
Yi Yang (yyietf@gmail.com)
Aijun Wang (wangaj.bri@chinatelecom.cn)
Xufeng Liu (xufeng.liu.ietf@gmail.com)
Yanhe Fan (yfan@casa-systems.com)
Lei Liu (liulei.kddi@gmail.com)

# Overview

- Removed distributed flooding reduction, and related

- Updated Algorithm for flooding topology (FT) computation to consider:
  - Degree (D for short):

  Degree of FT is the maximum degree among the degrees of the nodes on FT. The degree of a node on FT is the number of connections on FT it has to other nodes.

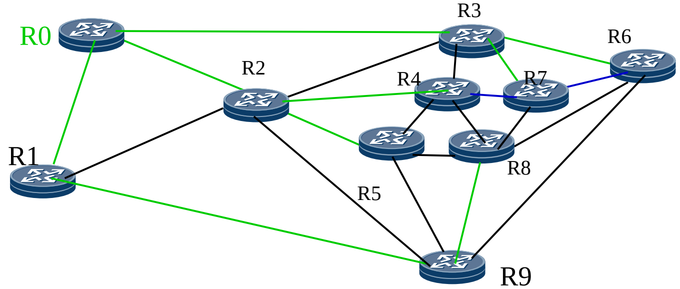


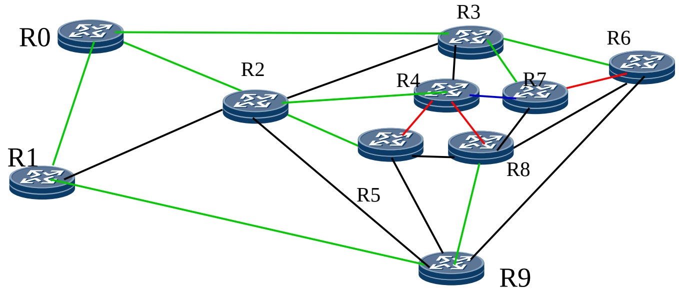
# Basic Idea of FTC Algorithm

- Select a node R0 with the smallest node ID;
- Build a tree using R0 as root breadth first;
- Connect node whose D is one to another (have FT: every node connects 2 or more nodes).

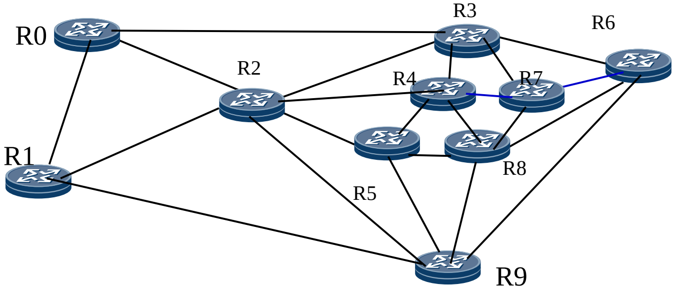
Consider Degree

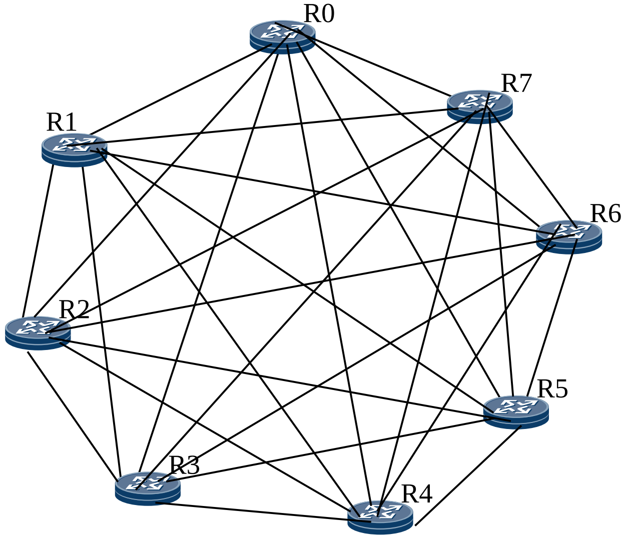
Consider Degree

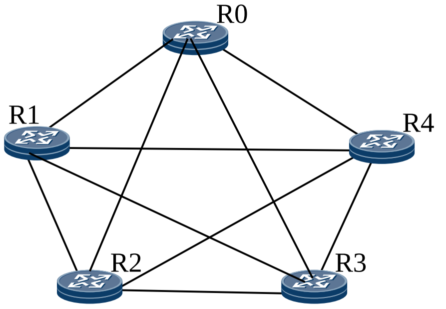


# FT Examples by Algorithm

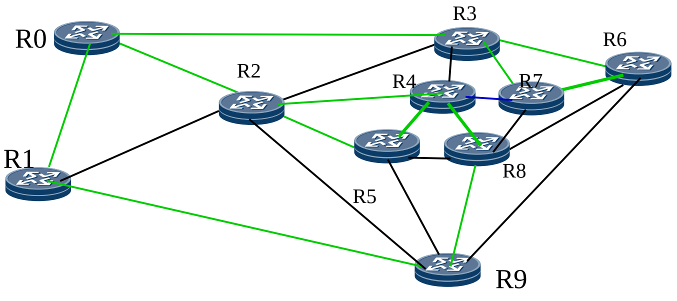


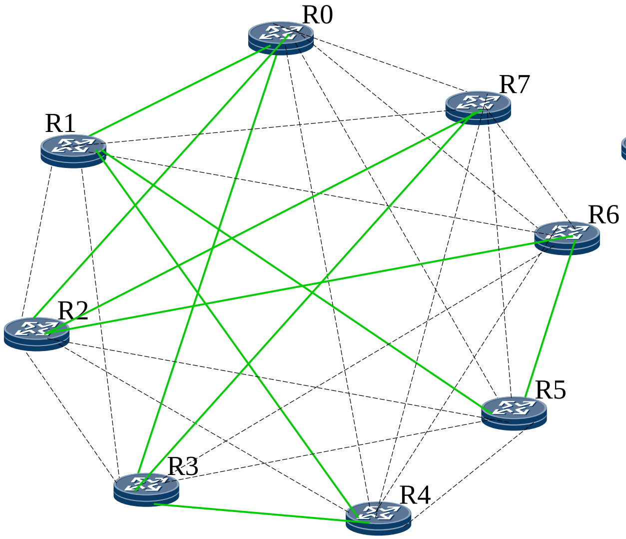



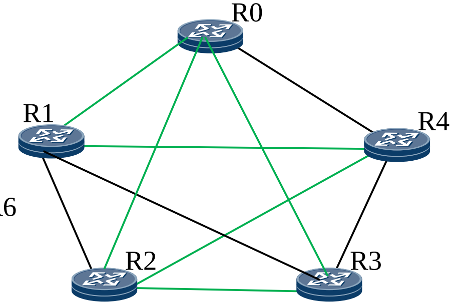




FT's
D=3

Link on FT

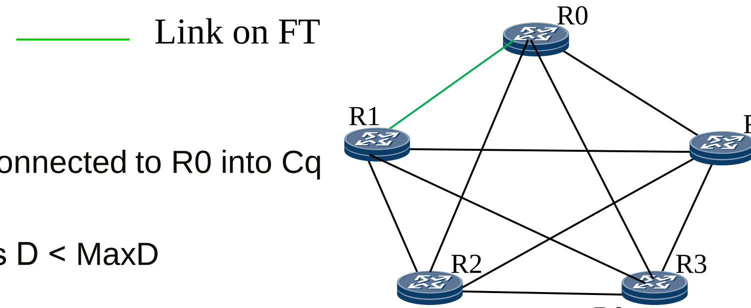



Minimum D = 2
Tolerant to any 1 link failure

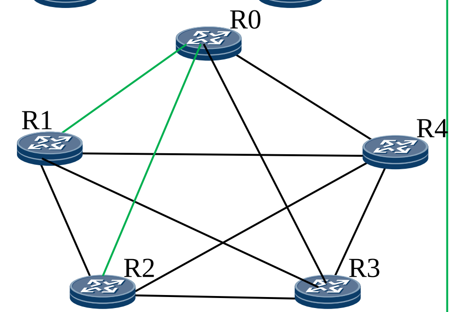
# FT Computation Details: build tree breadth first



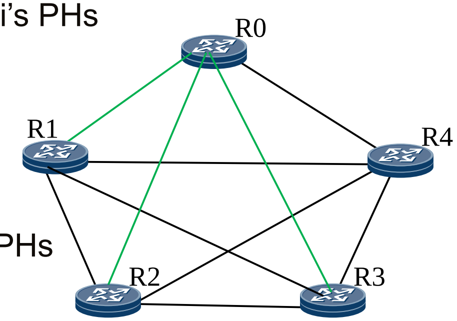

Cq = {(R0,D=0,PH={ })}, FT={ }, MaxD = 3.

0. Cq = { }, // remove the first element containing R0 from Cq
   FT= { (R0,D=0,PHs={ }) }; // add the element into FT
   Cq = { (R1,D=0,PHs={R0}), (R2,D=0,PHs={R0}), // add Ri connected to R0 into Cq
   (R3,D=0,PHs={R0}), (R4,D=0,PHs={R0}) }

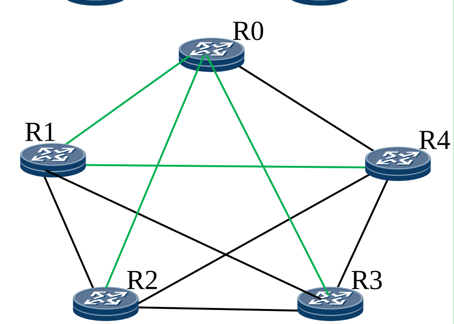
1. // remove the first element (R1,D=0,PHs={R0}) from Cq, R0's D < MaxD
   Cq = { (R2,0,{R0}), (R3,0,{R0}), (R4,0,{R0}) },
   // add (R1,0,{R0}) into FT, increase R0's D and R1's D by one
   FT = { (R0,1, { }), (R1,1, {R0}) }; // Ri -- R1 in Cq, not on FT, add R1 to Ri's PHs
   Cq = { (R2,0, {R0,R1}), (R3,0, {R0,R1}), (R4,0,{R0,R1}) }.




2. // remove the first element (R2,0, {R0,R1}) from Cq, R0's D < MaxD
   Cq = { (R3,0, {R0,R1}), (R4,0,{R0,R1}) },
   // add (R2,0,{R0}) into FT, increase R0's D and R2's D by one
   FT = { (R0,2, { }), (R1,1, {R0}), (R2,1, {R0}) };//Ri -- R2 in Cq, not on FT, add R2 to Ri's PHs
   Cq = { (R3,0, {R0,R1,R2}), (R4,0,{R0,R1,R2}) }.




3. // remove the first element (R3,0, {R0,R1,R2}) from Cq, R0's D < MaxD
   Cq = { (R4,0,{R0,R1,R2}) },
   // add (R3,0,R0) into FT, increase R0's D and R3's D by one
   FT = { (R0,3, 0), (R1,1, R0), (R2,1, R0), (R3,1, R0) }.//Ri – R3 in Cq, add R3 to Ri's PHs
   Cq = { (R4,0,{R0,R1,R2,R3}) }.

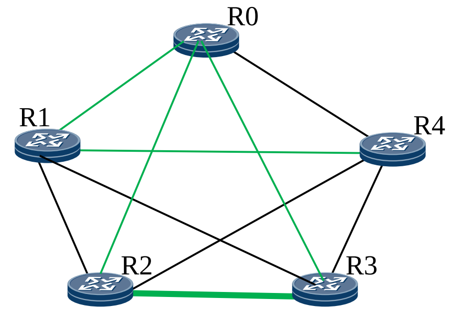



4. // remove the first element (R4,0, {R0,R1,R2,R3}) from Cq, R1's D < MaxD
   Cq = { },
   // add (R4,0,R1) into FT, increase R1's D and R4's D by one
   FT = { (R0,3, 0), (R1,2, R0), (R2,1, R0), (R3,1, R0), (R4,1,R1) }.
   Cq = { }.

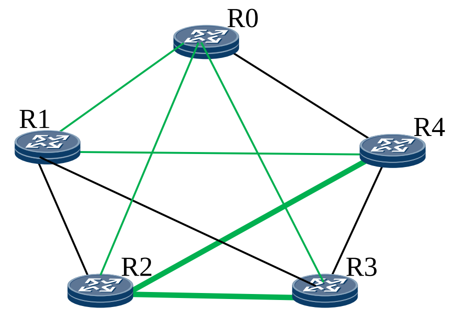
# FT Computation Details: connect node whose D=1

Link on FT

5. Cq = { },
   // Get the first node R2 whose D=1
   FT = { (R0,3, { }), (R1,2, {R0}), (R2,1, {R0}), (R3,1, {R0}), (R4,1, {R1}) }.
   // Add link R2-R3 to FT,
   // where R2-R3 is not on FT and R3's D=1 is minimum and R3's ID is minimum
   // increase R2's D and R3's D by one
   FT = { (R0,3, { }), (R1,2, {R0}), (R2,2, {R0}), (R3,2, {R0, R2}), (R4,1,{R1}) }.
   Cq = { }.




6. Cq = { },
   // Get the first node R4 whose D=1
   FT = { (R0,3, { }), (R1,2, {R0}), (R2,2, {R0}), (R3,2, {R0, R2}), (R4,1,R1) }.
   // Add link R4-R2 to FT,
   // where R4-R2 is not on FT and R2's D=2 is minimum and R2's ID is minimum
   // increase R2's D and R4's D by one
   FT = { (R0,3, { }), (R1,2, {R0}), (R2,3, {R0}), (R3,2, {R0, R2}), (R4,2, {R1, R2}) }.
   Cq = { }.




FT = { (R0,3, 0), (R1,2, {R0}), (R2,3, {R0}), (R3,2, {R0,R2}), (R4,2,{R1,R2}) }.

# Algorithm in Details (1)

Algorithm starts from node R0 as root with

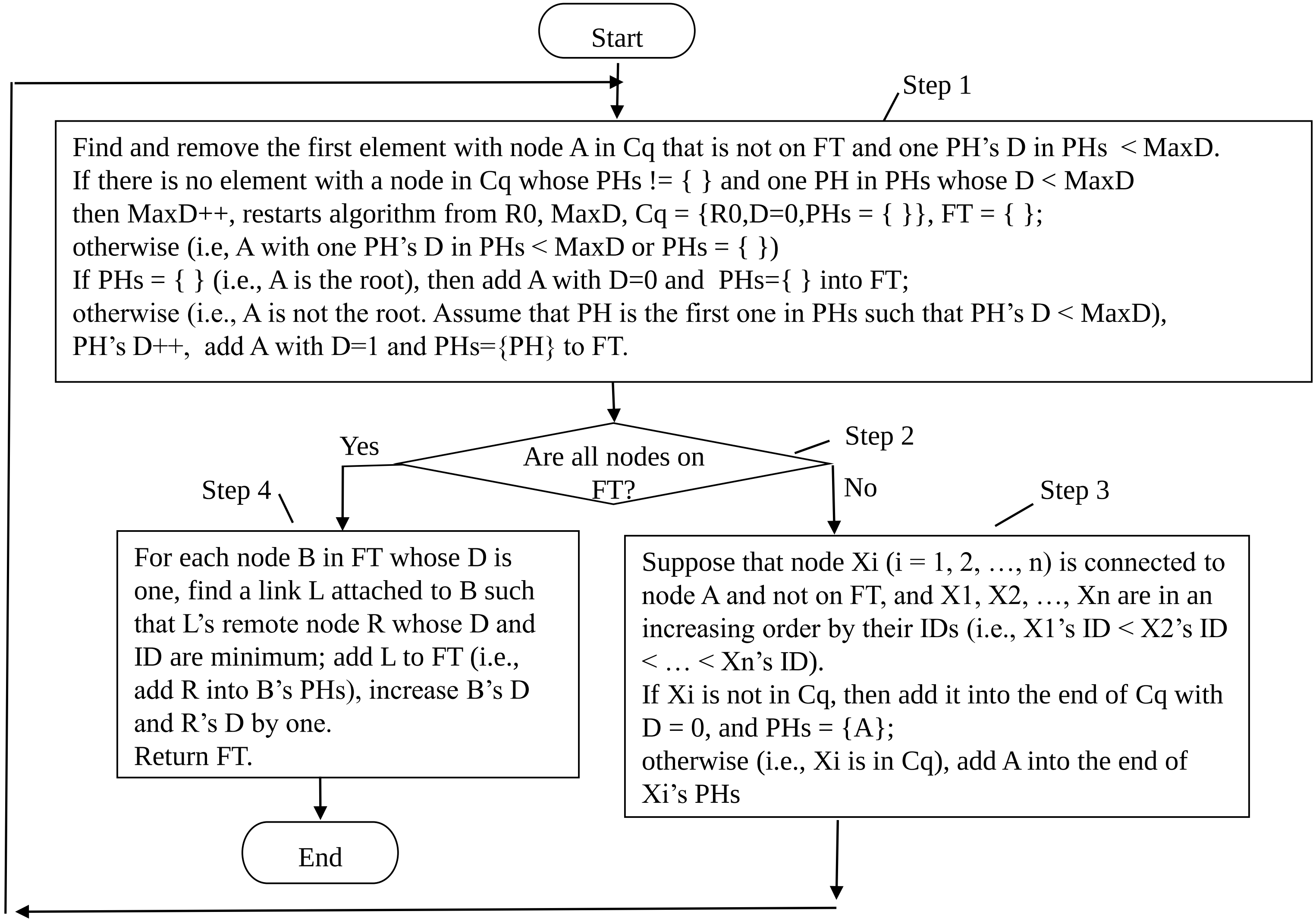
- a given maximum degree MaxD,
- a candidate queue Cq = {(R0, D = 0, PHs = { })},
- an empty flooding topology FT = { }.

Cq contains one element (R0, D = 0, PHs = { }), where

- node R0 is the root,
- D = 0 indicates Degree of R0 is 0 (i.e., the number of links on FT connected to R0 is 0),
- PHs = { } indicates that the Previous Hops (PHs for short) of R0 is empty.

# Algorithm in Details (2)

Algorithm starts from R0, MaxD = 3, Cq = {(R0, D = 0, PHs = { })}, and FT = { }.



# Next Step

Welcome comments

Request for adoption

# Algorithm Considering Degree and Others (3)

Algorithm starts from R0,  MaxD = 3,  Cq = {(R0, D = 0, PHs = { })},  and  FT = { }.

Some nodes such as leaves in spine-leaf network have constraints on their degrees of 2 (i.e., each of leaf node has a degree of 2 at maximum, which is represented as ConMaxD.