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).
FT Examples by Algorithm

FT’s D=3

Minimum D = 2
Tolerant to any 1 link failure
FT Computation Details: build tree breadth first

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

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

1. // remove the first element (R1,D=0,PH=\{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

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, \( \text{MaxD} = 3 \), \( \text{Cq} = \{(R0, D = 0, \text{PHs} = \{\})\} \), and \( \text{FT} = \{\} \).

---

Start

1. Find and remove the first element with node A in \( \text{Cq} \) that is not on \( \text{FT} \) and one PH’s D in \( \text{PHs} \) < \( \text{MaxD} \).
   - If there is no element with a node in \( \text{Cq} \) whose PHs \(!=\{\}\) and one PH in \( \text{PHs} \) whose D < \( \text{MaxD} \) then \( \text{MaxD}++ \), restarts algorithm from R0, \( \text{MaxD}, \text{Cq} = \{R0,D=0,\text{PHs} = \{\}\}, \text{FT} = \{\} \);
   - otherwise (i.e., A with one PH’s D in \( \text{PHs} \) < \( \text{MaxD} \) or \( \text{PHs} = \{\} \))
     - If \( \text{PHs} = \{\} \) (i.e., A is the root), then add A with D=0 and \( \text{PHs} = \{\} \) into \( \text{FT} \);
     - otherwise (i.e., A is not the root. Assume that PH is the first one in \( \text{PHs} \) such that PH’s D < \( \text{MaxD} \) ), PH’s D++, add A with D=1 and \( \text{PHs} = \{\text{PH}\} \) to \( \text{FT} \).

2. Are all nodes on \( \text{FT} \)?

3. Step 3

4. For each node B in \( \text{FT} \) whose D is one, find a link L attached to B such that L’s remote node R whose D and ID are minimum; add L to \( \text{FT} \) (i.e., add R into B’s PHs), increase B’s D and R’s D by one.
   - Return \( \text{FT} \).

End

Suppose that node \( X_i \) (\( i = 1, 2, \ldots, n \)) is connected to node A and not on \( \text{FT} \), and \( X_1, X_2, \ldots, X_n \) are in an increasing order by their IDs (i.e., \( X_1 \)’s ID < \( X_2 \)’s ID < \( \ldots < X_n \)’s ID).
   - If \( X_i \) is not in \( \text{Cq} \), then add it into the end of \( \text{Cq} \) with D = 0, and \( \text{PHs} = \{A\} \);
   - otherwise (i.e., \( X_i \) is in \( \text{Cq} \), add A into the end of \( X_i \)’s PHs
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.

1. Find and remove first element with node A in Cq not on FT and one PH’s D in PHs < MaxD and < its ConMaxD. If there is no element with a node in Cq whose PHs != { } and one PH’s D in PHs < MaxD and < its ConMaxD then MaxD++, restarts algorithm from R0, MaxD, Cq = {R0,D=0,PHs = { }}, FT = { }; otherwise (i.e., A with one PH’s D in PHs < MaxD and < its ConMaxD or PHs = { }) If PHs = { } (i.e., A is the root), then add A with D=0 and PHs={ } into FT; otherwise (i.e., A is not root. Assume PH is first one in PHs such that PH’s D < MaxD and < its ConMaxD), PH’s D++, add A with D=1 and PHs={PH} to FT.

2. Are all nodes on FT?
   - Yes, Step 4
   - No, Step 3

3. Suppose that node Xi (i = 1, 2, …, n) is connected to node A and not on FT, and X1, X2, …, Xn are in an increasing order by their IDs (i.e., X1’s ID < X2’s ID < … < Xn’s ID).
   - If Xi is not in Cq, then add it into the end of Cq with D = 0, and PHs = {A}; otherwise (i.e., Xi is in Cq), add A into the end of Xi’s PHs

4. For each node B in FT whose D is one, find a link L attached to B such that L’s remote node R whose D and ID are minimum; add L to FT (i.e., add R into B’s PHs), increase B’s D and R’s D by one. Return FT.

End