## Topo:

* topology mạng là đồ thị có hướng gồm có 1 tập node nguồn, 1 tập nút đích, 1 tập core node.
* trong tập core, cạnh có hướng nối giữa bất kì node nào (cạnh a -> b là phân biệt với cạnh b -> a).
* Với mỗi node nguồn, cạnh chỉ nối từ node nguồn đó tới 1 hoặc nhiều core node. Với mỗi node đích, cạch chỉ nối từ core node tới nút đích (ko có cạnh theo chiều ngược lại).
* Mỗi cạnh có bandwidth (> 0).
* Mỗi node có computing capacity (core node thì có thể >= 0, riêng node nguồn và đích thì luôn = 0).

## Demand

* Tập demand: y1, y2,..... Mỗi demand gồm có các thông tin: node nguồn, node đích, chuỗi function yêu cầu, bandwidht yêu cầu (ymin, ymax)

## Giải thuật

### B1: Chuẩn bị cho giải thuật

* tạo 1 node nguồn tổng, node nguồn tổng có 1 cạnh nối với mỗi node nguồn với bandwithd rất lớn.
* Một node đích tổng, mỗi node đích có 1 cạnh nối tới node đích tổng với bandwithd rất lớn.
* Tìm **luồng cực đại** từ node nguồn tổng tới node đích tổng. Gọi giá trị luồng cực đại tính được là **fmax**.

### alphaBetaMultipath

* Initialization Graph
* Khởi tạo bandwidth cấp cho demadn y1, y2, ... lần lượt là y1min, y2min....
* Thiết lập yMin (lb) = min (yd) và yMax (ub) = Fmax/|D|

### B2:

* Sắp xếp các demand theo thứu tự giảm dần của bandwidth đc cấp
* vào vòng lặp:
  + (1) Sắp xếp các demand theo thứu tự giảm dần của bandwidth đc cấp
  + (2) Xử lý từng demand lần lượt theo danh sách đã sắp xếp
    - Với mỗi demand yd thì:
    - (2.1) Tìm mọi đường đi ngắn nhất từ nguồn của yd tới đích
    - (2.2) Với mỗi đường đi: Tính giá trị nhỏ nhất của link capacity của mọi cạnh trên đường đi gọi là ep, tính giá trị nhỏ nhất của node capacity của mọi node trên đường đi (trừ node nguồn và node đích) gọi là vp. cp = min(ep, vp).
    - (2.3) Sắp xếp các path theo thứ tự giảm dần của cp
    - (2.4) Chọn các path nào gán cho demand: Xem 1 path có gán đc ko, duyệt danh sách path theo thứ tự đã sắp xếp tìm path cuối cùng trong danh sách mà cp > bandwidth(yd). Nếu tồn tại, gán path này cho demand yd, cập nhật lại link capacity còn lại của các cạnh. Chạy thủ tục GanNode() (mô tả sau). Nếu ko tồn tại path như vậy, thì chọn path đầu tiên trong danh sách (cp lớn nhất), gán toàn bộ path này cho yd, bỏ path này khỏi danh sách, bandwithd(yd) = bandwithd(yd) - cp(của path vừa gán). Tìm path thứ 2 cho bandwidth(yd) vừa tính. Cứ vậy tới khi gán hết bandwidth(yd).

### B3:

(3) Tính giá trị hàm mục tiêu : F = Min (yd) với mọi demand d - alpha Max ( link utilization ) của mọi cạnh - beta Max ( Node utilization ) của mọi node

link utilization của 1 cạnh = tổng băng thông qua cạnh đó / link capacity, tương tự tính node utilization

Nếu F > F thì nghĩa là tìm đc giải pháp tốt hơn, F := F, lưu lại

Nếu F > Ftoiuu thì nghĩa là tìm đc giải pháp tốt hơn, Ftoiuu := F, lưu lại

### B4:

Nếu không thỏa mãn được hết các demand thì bandwidth của mỗi demand gán lại như sau: cận trên = min(bandwitdh(yd)) với mọi yd; bandwidth(yd) = min(ymax, (cận trên + cận dưới) / 2). Lặp lại các bước (1)-(3) vòng lặp.

Nếu thỏa mãn được hết các demand thì bandwidth của mỗi demand được gán lại như sau: cận dưới = min(bandwidth(yd)) với mọi yd; bandwidth(yd) = max(ymin, (cận trên + cận dưới ) / 2). Lặp lại các bước (1)-(3) vòng lặp.

cận dưới ban đầu gán bằng min(ymin(yd)) với mọi demand yd

cận trên ban đầu gán bằng giá trị luồng cực đại Fmax / số demand

Thủ tục GanNode để gán node xử lý trên mỗi path như sau:

Xét lần lượt từng VNF trong chuỗi SFC

Với mỗi VNF duyệt từ lần lượt các node trên đường đi từ nguồn tới đích: thấy node đầu tiên có đủ capacity để gán cho VNF này thì gán luôn. Sau đó xét VNF tiếp theo, duyệt danh sách node trên đường đi từ nguồn tới đích xét bắt đầu từ node vừa gán trước đó...

Vào bước lặp

(1) Giả sử node A xử lý VNF1, B xử lý VNF2, C xử lý (VNF3, VNF4), D xử lý VNF5. Xét xem trong các node trên path đang xét nằm giữa node A tới node B, tìm node có capacity nhỏ nhất và đủ lớn để xử lý VNF1 và nhỏ hơn capacity node A ko, nếu có thì gán node này xử lý VNF1 và gọi là node A1. Tương tự lại tìm các node nằm giữa A1 và C để xem có node tốt hơn gán cho VNF2 ko, nếu có gọi là B1, nếu ko có thì B1 = B. Rồi là xét các node giữa B1 và D ....

Thu đc danh sách mới các node xử lý các funtion. Thực hiện lại vòng lặp với chuỗi node mới này, dừng lại khi với 1 chuỗi mới mà ko tìm thấy node nào tốt hơn thay thế thì thôi