**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

--- o0o ---

**BÁO CÁO MÔN HỆ PHÂN TÁN**

**ĐỀ TÀI: HỆ THỐNG CHIA SẺ FILE THEO MẠNG CHORD**

Nhóm sinh viên thực hiện:

**Nguyễn Vũ Long MSSV: 20175979**

**Minh MSSV:**

**Thắng MSSV:**

***Hà Nội, tháng 7 năm 2023***

CHORD DHT

1. Lý thuyết
   1. DHT
   2. CHORD
2. Hệ thống xây dựng dựa trên CHORD
   * Cách hoạt động
   * Demo
   * Tính trong suốt
   * Tính co giãn (scalability)
   * Nhân bản, nhất quán
   * Khả năng phục hồi

# 1. Lý thuyết

# 1.1 DHT

DHT nhấn mạnh vào các thuộc tính sau [3]:

* Không tập trung (Decentralization): Các node tham gia cấu thành hệ thống không có thành phần trung tâm làm điều phối mạng.
* Khả năng mở rộng: Hệ thống vẫn có thể hoạt động hiệu quả với hàng nghìn hoặc hàng triệu node.
* Khả năng chịu lỗi: Hệ thống vẫn có thể làm việc ổn định ngay cả khi có các sự kiện node tham gia, rời bỏ, lỗi diễn ra liên tục.

Kỹ thuật khóa được sử dụng để đạt được mục đích là mỗi node chỉ cần liên kết với một số ít các node khác trong hệ thống, thường là **O(logn)** với n là số node tham gia. Vì vậy sự thay đổi trong các thành viên chỉ ảnh hưởng đến một phần nhỏ của hệ thống.

Một số thiết kế DHT tìm đến tính bảo mật chống lại những người dùng xấu và cho phép người tham gia giấu danh tính, mặc dù điều này không phổ biến trong các hệ thống P2P chia sẻ file.

Cuối cùng, DHT phải giải quyết những vấn đề cơ bản của các hệ thống phân tán đó là cần bằng tải, tính toàn vẹn dữ liệu, hiệu năng (cụ thể là đảm bảo các hoạt động như định tuyến, lưu trữ, truy vấn phải được thực thi nhanh chóng).

# Một số ứng dụng sử dụng DHT

* BitTorrent: Phân tán file. BitTorrent lựa chọn sử dụng một DHT như một người theo dõi phân phối để cung cấp theo kế hoạch giữa client đang download một file đặc biệt.
* The Circle: Chia sẻ file và tán gẫu.
* Codeen: Web caching.
* Coral Content Distribution Network.
* CSpace: Các giao tiếp an toàn.
* Dijjer: Freenet-like mạng phân tán.
* eMule: Chia sẻ file.
* FAROO: Công cụ nghiên cứu Web Peer to Peer.
* GNUnet : mạng phân tán Freenet-like
* I2P: Mạng nạc danh.
* JXTA: Mã nguồn mở P2P.
* Limewire: P2P Chia sẻ file có gắn java applet gọi DHT.
* NEOnet: Chia sẻ file.
* Warez P2P: Chia sẻ file.
* YaCy: Công cụ nghiên cứu phân tán.

**Cấu trúc DHT**

Cấu trúc của DHT có thể được phân chia ra một số thành phần chính. Nền tảng là một **không gian khóa** trừu tượng, ví dụ như tập các chuỗi 160-bit. Cách thức phân chia không gian khóa cho từng node tham gia vào mạng. Mạng overlay kết nối các node, cho phép chúng có thể tìm kiếm được node quản lý của một khóa cho trước bất kỳ trong không gian khóa.

Khi các thành phần này được đặt ở đúng vị trí của nó, thì cách thức sử dụng DHT cho việc lưu trữ và truy vấn dữ liệu sẽ được diễn ra như sau. Giả sử không gian khóa là một tập các chuỗi 160-bit. Để lưu trữ một file với tên gọi filename và dữ liệu data trong DHT, ví dụ giá trị hash của filename theo thuật toán SHA1 được tính toán, tạo ra một khóa k có độ dài 160-bit,và một thông điệp put(k,data) được gửi tới một node bất kì tham gia trong DHT. Thông điệp được chuyển từ node này sang node khác thông qua mạng overlay cho đến khi nó tới node cuối cùng chịu trách nhiệm quản lý khóa k, được xác định nhờ cách thức phân chia không gian khóa, ở đó cặp (k,data) được lưu trữ. Mọi client có thể truy xuất nội dung của file bằng cách hash filename để sinh ra khóa k và hỏi một node bất kì trong DHT để tìm dữ liệu ứng với khóa k đó với một thông điệp get(k). Thông điệp lại được định tuyến qua mạng overlay tới node chịu trách nhiệm quản lý khóa k, node này sẽ trả lời với dữ liệu data được lưu trữ.

Các thành phần cách thức phân chia không gian khóa và mạng overlay được miêu tả dưới đây với mục đích đưa ra được những ý tưởng chính có tính quan trọng chung đối với hầu hết các DHT; nhiều thiết kế khác nhau ở những chi tiết cụ thể.

# Cách thức phân chia không gian khóa

Hầu hết DHT đều sử dụng các dạng hash nhất quán (consistent hash) để ánh xạ các khóa tới node. Kỹ thuật này áp dụng một hàm δ(k1, k2) định nghĩa một khái niệm trừu tượng về khoảng cách giữa k1 và k2. Mỗi node được gán một khóa đơn gọi là định danh (identifier - ID). Một node với ID i sở hữu tất cả các khóa mà i là ID gần nhất của các khóa này, ước lượng theo hàm δ.

**Ví dụ :** Chord DHT coi khóa là các điểm trên một đường tròn, và δ(k1, k2) là khoảng cách đi theo chiều kim đồng hồ xung quanh đường tròn từ khóa k1 tới khóa k2. Do đó, không gian khóa đường tròn được chia thành các cung liên tiếp mà điểm cuối của cung này là các định danh ID của các node. Nếu i1 và i2 là hai ID liền kề nhau thì node có định danh ID i2 sở hữu tất cả các khóa nằm giữa i1 và i2.

Consistent hashing có một thuộc tính quan trọng thuộc về bản chất đó là việc gỡ bỏ hay thêm vào các node chỉ làm thay đổi một tập các khóa sở hữu bởi các node có ID liền kề, và các node khác thì không bị ảnh hưởng.Vì mọi thay đổi trong quyền sở hữu đều gây ra những di chuyển tốn kém băng thông của các đối tượng được lưu trữ trong DHT từ node này đến node khác, nên việc giảm thiểu việc sắp xếp lại sẽ tăng tính hiệu quả cho việc hỗ trợ các yêu cầu vào ra của node hay các sự cố lỗi của các node.

# Mạng Overlay

Mỗi node duy trì một tập các liên kết với các node khác (hàng xóm của nó hoặc có thể gọi là bảng định tuyến). Các liên kết này sẽ tạo ra một mạng overlay. Một node kết nạp các node khác vào làm hàng xóm của nó dựa trên một cấu trúc nào đó, gọi là network topology.

Tất cả DHT topology đều có chung một thuộc tính quan trọng là: với một khóa k bất kỳ, node có chứa khóa k hoặc có liên kết tới node khác gần hơn với khóa k theo khoảng cách trong không gian khóa được định nghĩa ở trên, thì đều có thể định tuyến được thông điệp tới node quản lý khóa k sử dụng thuật toán tham lam: ở mỗi bước, chuyển thông điệp tới hàng xóm mà ID của nó gần nhất với khóa k. Khi không có hàng xóm nào như thế, thì chúng ta đã đến đúng node là node quản lý khóa k. Kiểu định tuyến này đôi khi được gọi là định tuyến dựa trên khóa (key based routing).

Ngoài tính chính xác trong định tuyến, có hai yếu tố quan trọng trong một topology là số lượng hops tối đa trên đường đi thấp để các yêu cầu kết thúc nhanh; và số lượng hàng xóm tối đa thấp để việc duy trì không quá khó khăn. Tuy nhiên, để có quãng đường đi ngắn đòi hỏi số lượng hàng xóm cao và ngược lại. Có một vài lựa chọn cân bằng giữa hai yếu tố trên như sau:

* Degree O(1), route length O(logn)
* Degree O(logn), route length O(logn/loglogn)
* Degree O(logn), route length O(logn)
* Degree O(n1/2), route length O(1)

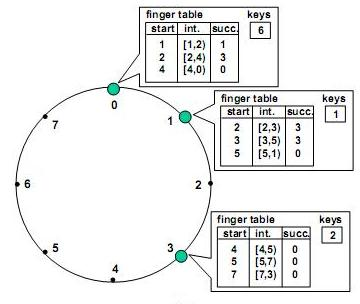
Degree: số hàng xóm tối đa; Route length: số hop tối đa trên đường định tuyến; n: số node tham gia trong hệ thống.

Lựa chọn thứ ba là lựa chọn phổ biến. Nhiều DHT sử dụng tính linh hoạt trong cách chọn hàng xóm để chọn ra những hàng xóm gần nhau về mặt độ trễ giữa các node của mạng vật lý ở phía dưới.

# 1.2 Chord

Giao thức Chord được thiết kế giống như giao thức định tuyến DHT nhằm mục đích phát triển một cách phân tán dữ liệu tốt nhất, các node được phân phối IDs và Keys với nhiều đặc trưng như Scalability(đánh giá), Complete Decentralization(phân quyền), Efficient Load Blancing(cân bằng tải), và Simplicity( đơn giản).

Chord coi các khóa Key là các điểm trên một đường tròn. Không gian khóa đường tròn được chia thành các cung liên tiếp mà điểm cuối của cung này là các định danh ID của các node. Mỗi node lưu trữ thông tin định tuyến tới các node khác trong một bảng định tuyến được gọi là **Finger Table.**



Bảng Finger table và cấp key cho từng node 0,1,3 và keys 1,2,6

Giao thức Chord hỗ trợ duy nhất một hoạt động : đưa ra 1 key, nó sẽ ánh xạ key đó vào một node. Tùy thuộc vào ứng dụng sử dụng Chord ( văn bản, hình ảnh, media..), node đó sẽ lưu trữ một giá trị kết hợp với key. Chord sử dụng kí thuật consistent hashing để cấp key cho các node. Consistent hashing dùng để cần bằng tải, mỗi node sẽ nhận được số lượng key gần ngang nhau, vào làm cả việc chuyển số lượng key khi có node tham gia hay rời khỏi hệ thống.

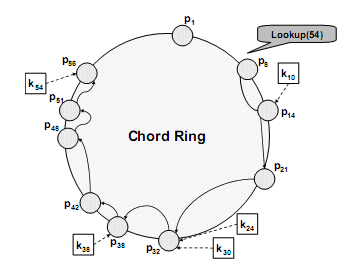
Kĩ thuật consistent hashing đầu tiên sẽ nhận biết các node trong hệ thống, tạo ra sự cân chỉnh về số lượng các node. Mỗi node trong Chord cần được "routing" để biết thông tin về một vài node khác. Vì bảng định tuyến là phân tán, một node sẽ sử dụng hàm băm để giao tiếp với các node khác. Khi mạng được thiết lập, một hệ thống gồm N-node, **trong đó mỗi node chứa thống tin về O(log N)** node xung quanh nó, và **tìm kiếm các node khác thông qua O(log N) thông điệp tới các node đó**. Chord duy trì thông tin định tuyến khi các node tham gia/rời khỏi hệ thống. Với một hệ thống có tần suất các node tham gia/ rời khỏi mạng cao, một node khi tham gia hay một node cần định tuyến lại cũng chỉ cần gửi không quá O(log2 N) thông điệp để ổn định thông tin định tuyến.

Các đặc điểm hệ thống

* Load Balance ( phân tải) : Chord sử dụng bảng băm phân tán, phân tải trên các node, một node sẽ không chứa quá nhiều key.
* Decentralization (phân quyền): Chord là phân tán hoàn toàn, không node nào quan trọng hơn node nào, việc này cải thiện được sự vững chắc của hệ thống.
* Scalability (mở rộng) : Hệ thống hoạt động với hàng nghìn hay hàng triệu node. Giá của việc tìm kiếm tăng lên theo Log của số node : Log(n)
* Availability (tiện dụng) : Chord tự điều chỉnh các bảng định tuyến khi có node tham gia và rời khỏi mạng
* Flexible naming ( định nghĩa tên linh hoạt) : Chord không ràng buộc về cấu trúc của key mà nó tìm kiếm, không gian key là phẳng bằng việc gán cho key một cái tên và tìm kiếm. ( ví dụ phẳng tức là đưa tất cả các loại key về thành một kiểu như id, khi tìm thì chỉ cần tìm id của key)

**Giao thức cơ sở**

Chord là một giao thức tìm kiếm phân tán đánh địa chỉ. Nó cung cấp chức năng của một DHT bằng cách hỗ trợ tìm kiếm : đưa ra một key, nó ánh xạ key đó vào một nút. Mục đích của Chord là dùng kỹ thuật băm phù hợp. Consistent hashing giữ độ cân bằng tải, khi một nút nhận một số lượng khóa xấp xỉ nhau. Hơn nữa, cân bằng tải làm việc ngay cả trong một dải băm luôn thay đổi, như khi các nút lỗi hoặc rời khỏi hệ thống hay khi nút mới join vào.

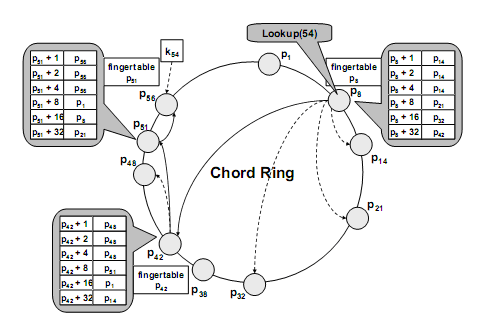


Chord không chỉ bảo đảm cho việc tìm nút có trách nhiệm cho một key đã đưa, nhưng có thể làm điều đó với nhiều ảnh hưởng trong một hệ thống vững chắc N nút, mỗi nút chỉ chứa thông tin về **O(log n)** nút khác, và giải quyết tất cả tìm kiếm nhiều **O(log n)** thông điệp tới các nút khác. Các thuộc tính cung cấp khả năng cho hiệu quả hệ thống mở rộng lớn.

Khái niệm đằng sau Chord là : tất cả các nút pi và tất cả các ki đc ánh xạ vào trong ko gian ID tuần hoàn. Dùng các khóa và số Peer nếu hàm băm được chấp nhận, nhưng không chỉ ra rõ ràng hàm băm cho các trình bày đơn giản hơn. Với mỗi khóa ki đc gán cho successor của pi trong không gian ID ( mỗi nút chịu trách nhiệm cho tất cả các key để nhận biết giữa ID predecessor của nó và ID sở hữu của nó). Trong hình 3 :10 nút đc phân tán qua ko gian ID, key k54 đc gán cho nút p56 như là successor của nó.

Một cách tiếp cận của định vị Peer có trách nhiệm cho một key được minh họa, khi mỗi Peer biết làm cách nào để tiếp xúc với successor hiện thời của nó trên vòng tròn ID, một truy vấn cho k54 khởi tạo bởi Peer p8 vượt qua vòng tròn cho đến khi nó bắt gặp một cặp nút straddle mong muốn nhận ra; cặp thứ hai p56 là nút chịu trách nhiệm đến key đó. Tiến trình tìm kiếm giống với tìm kiếm theo đường thẳng trong danh sách và có một số **O(n)** hop để tìm nút đích, trong khi chỉ yêu cầu **O(1)** thông tin về các nút khác.

Để đẩy nhanh việc tìm kiếm, Chord chứa thêm thông tin định tuyến: mỗi Peer pi chứa một bảng định truyến gọi là Finger table... Entry thứ m trong bảng của nút pi chứa một con trỏ tới nút pi đầu tiên nối tiếp pi bởi ít nhất 2^m -1 trên vòng tròn tạo. Trong sơ đồ : đầu tiên mỗi nút chỉ chứa thông tin về một số lượng nhỏ các nút khác, và biết nhiều hơn về các nút xa hơn bằng cách hỏi các nút gần nó trên vòng tròn định danh, thứ hai: một nút finger table ko cần thiết chứa đầy đủ thông tin để xác định trực tiếp nút có trách nhiệm đến một khóa ki . Tuy nhiên, khi mà mỗi Peer có các entry finger bằng lũy thừa 2 khoảng cách vòng tròn định danh, mỗi nút có thể đẩy một truy vấn tại nửa cuối dọc theo khoảng cách giữa nó với nút đích. Thuộc tính này được minh họa trong hình 4 của nút p8. Số các nút bị tiếp xúc để tìm kiếm nút đích trong một hệ thống N nút là O(log n)



**Tìm kiếm khóa sử dụng bảng FingerTable**

Chord thực thi một giao thức ổn định với mỗi Peer chạy định kỳ trong nền và nó cập nhập bảng finger table và con trỏ successor trong thứ tự để đảm bảo tìm kiếm thực thi chính xác thiết lập thay đổi của các Peer liên quan.

Chord có thể cung cấp dịch vụ tìm kiếm cho nhiều ứng dụng, như phân tán file hệ thống hay các dữ liệu cộng tác. Tuy nhiên Chord ko phải là một công cụ tìm kiếm, nó chỉ hỗ trợ các truy vấn single-term exact-match và không hỗ trợ bất kỳ form thứ bậc nào.

## 2. Hệ thống xây dựng dựa trên CHORD

Ngôn ngữ sử dụng: Python

Lợi thế: đơn giản cài đặt, thư viện nguồn mở

**Cách hoạt động**

Vì đây là hệ thống phân tán phi tập trung (decentralized distributed system), chúng ta không phân biệt server và client, thay vì đó mỗi Node vừa là server vừa là client, cho phép kết nối dạng p2p với Node khác (insert ví dụ về bitorrent). Bất cứ node nào cũng có thể join vào nhưng phải biết trước IP và Port của Node đã ở trong sẵn network.

Script nhận 2 tham số đầu vào, IP và Port, `python Chord.py 127.0.0.1 10000`

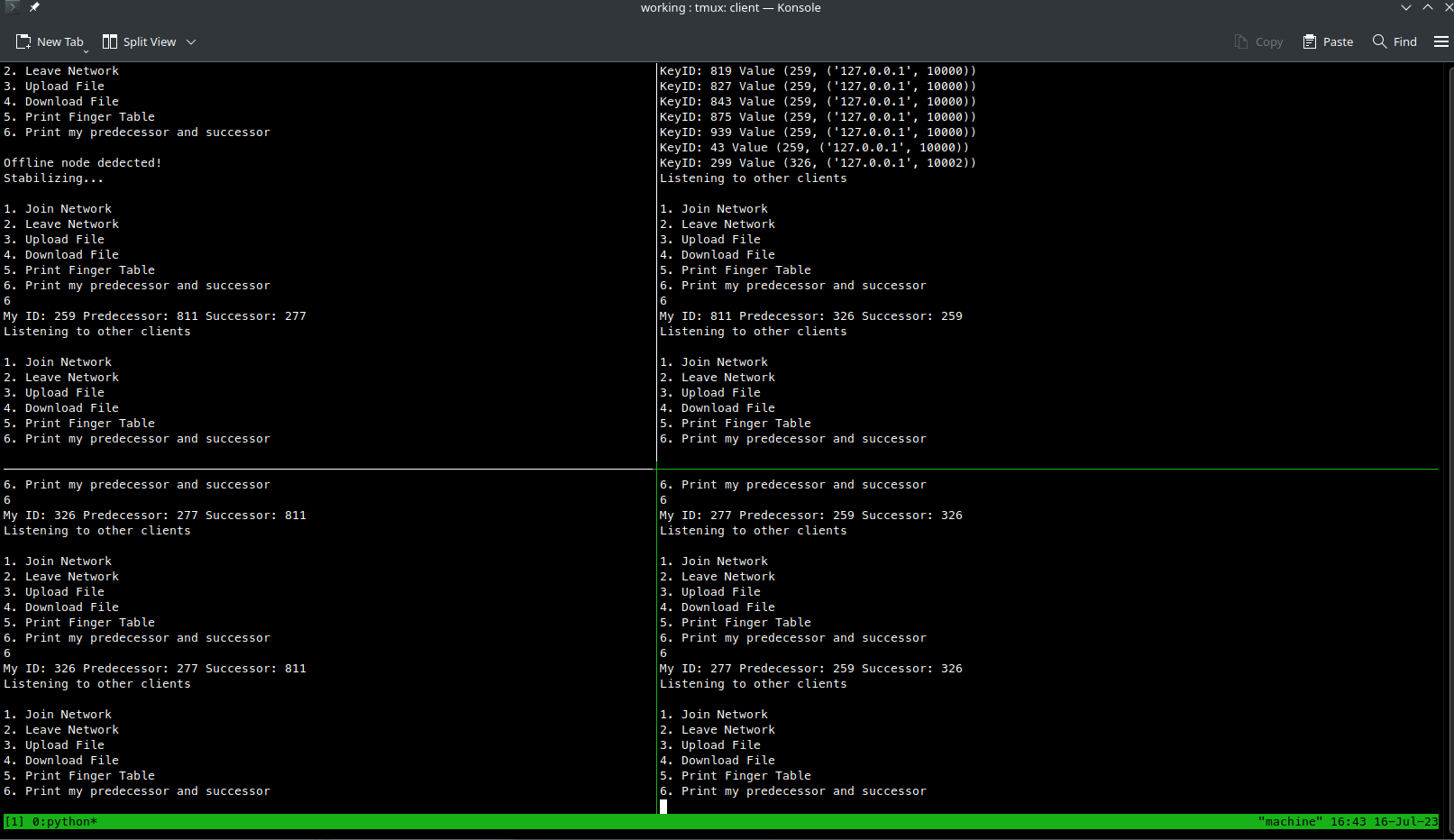
Khi khởi động, nếu không có tham số truyền vào ở đầu vào, script sẽ tự động sử dungj localhost 127.0.0.1 ở port 9999 để mở node đầu tiên trong mạng.

**Join**

Muốn thêm một Node mới vào mạng chỉ cần biết IP và Port của 1 Node bất kỳ đã có sẵn trong mạng, hệ thống sẽ tự động cân bằng.

**Leave**

Một hay nhiều Node có thể rời mạng bằng cách thông báo cho các Node khác rằng nó sẽ rời đi qua phương thức leaveNetwork. Khi một Node gặp sự cố mà không thể thực hiện thông báo, mạng CHORD cũng có thể biết được qua phương thức pingSuccessor, từ đó tự cân bằng.



**Upload/Download File**

Sử dụng phương thức uploadFile và downloadFile

**Tính co giãn**

Mở rộng được tối đa đến khi hàm băm vắt can kiệt số hash có thể bởi hàm 2^m-1 trong trường hợp này m = 10. Để đơn giản hóa quá trình kết nối, IP sử dụng trong bài này sẽ chỉ sử dụng localhost thức 127.0.0.1 và phân biệt các Node với nhau bởi Port. Vậy có 2^16-1 số port có thể mở, vẫn lớn hơn số hash mà hàm băm có thể tạo được.

**Nhân bản, nhất quán**

Có hỗ trợ nhân bản qua giao mạng hàng ngang, mỗi Node khi tham gia mạng sẽ chưa cần thiết phải được nhân bản. Nhưng sẽ tiến hành nhân bản số key của mình cho các Node lân cận để tránh mất mát.

=> chưa hoàn toàn có khả năng nhất quán theo chu kỳ

**Khả năng phục hồi**

Hiện tại script chỉ có thể tạo ra Network có khả năng phục hồi được nếu chỉ có 1 hoặc 2 Node gặp sự cố cùng một lúc. Khi gặp sự cố bất chợt tất cả những key hay những file chưa được nhân bản qua phương thức leaveNetwork. Mạng có thể tự cân bằng trở lại nhưng dữ liệu của Node gặp sự cố thì không

=> khả năng phục hồi chưa ổn định