**Lab 04: Truyền tham số với Omnetpp.ini**

*Notes:*

*+ Có deadline*

*+ SV phải chụp lại các bức ảnh để minh chứng việc mình hoàn thành các bài tập trong từng Lab*

*+ Ghi câu trả lời (và cả ảnh chụp ra file pdf hoặc doc) rồi gửi đến địa chỉ email: nguyenthanh@soict.hust.edu.vn, với tiêu đề email và tên file như sau (chẳng hạn bạn sinh viên có tên là Nguyễn Văn An, mssv: 20160001:*

***IT3920-STUDENT\_ID-NAME-Lab04***

***Vd: IT3920-2016001-AnNV—Lab04***

Ở bài Lab này, SV cần hoàn thành một ví dụ cho phép mô hình giả lập đọc được các dữ liệu truyền vào cho tham số của omnetpp.ini và các dữ liệu khác trong file NED. Mô hình giả lập này cho phép tùy chọn trong file omnetpp.ini: (i) số lượng tin nhắn tối đa được gửi từ một host và (ii) số lượng các host có trong mạng. Ngoài ra khi tin nhắn được truyền từ một host A, sẽ được forward qua các host khác và chỉ dừng lại (sau đó bị hủy) tại một host B ngay cạnh A (biết rằng các host kết nối với nhau theo hình trạng dạng ring).

**SV tiếp tục hoàn thành Lab03B, hạn cuối là trước 12h đêm chủ nhật (12/04) này. Bài Lab04 được phép chia làm 2 phần Lab04A, Lab04B. Lab04B chính là câu 4b trong bài này. Lab04A có hạn cuối là trước 12h đêm thứ 3 này. Lab04B sẽ có hạn cuối sau buổi học của thứ 4 tuần sau (15/04).**

1. Giới hạn số lần trao đổi tin nhắn từ một host:

Như ở bài Lab trước, bạn có thể thấy số lượng các tin nhắn gửi đi từ một host là vô hạn cho đến khi chương trình giả lập được lệnh dừng lại. Ở bài Lab này, chúng ta sẽ cài đặt giới hạn về số lượng tin nhắn mà một host có thể gửi đi. Để có thể dễ dàng cập nhật lại số lượng tin nhắn này, ta sẽ cài đặt ở file cấu hình omnetpp.ini.

Trước tiên, ở file cấu hình NED cần khai báo một simple mang tên là SimpleHost. Trong simple này khai báo một tham số có tên là limit, kiểu là int. SimpleHost có một cổng in và một cổng out.

Để đơn giản, ở đây ta làm việc với mạng mang tên là network, và nó chỉ có hai host bên trong, độ trễ delay trong đường truyền mạng là 1000ms.

Tiếp đó, gán giá trị của tham số limit trong file omnetpp.ini, để không quá nhiều tin nhắn cũng như không quá ít, SV có thể thiết lập giá trị là 5.

SV thực hiện việc build Project để Omnet++ sinh ra file mã nguồn CC tương ứng với SimpleHost. Trong mã nguồn file CC này, cần cài đặt các công việc sau:

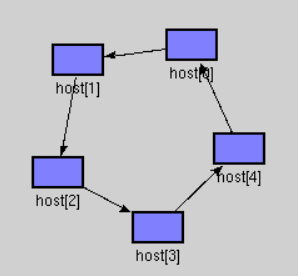
a) Đọc giá trị được gán cho tham số limits ở file omnetpp.ini (gợi ý dùng hàm par).

b) Khai báo một biến đếm mang tên là counter và thiết lập nó là biến được theo dõi trong suốt quá trình giả lập (bằng cách sử dụng hàm WATCH trong hàm initialize).

c) Với hai host trong mạng, sử dụng hàm strcmp đặt trong câu lệnh if để tiến hành thiết lập chỉ một host gửi tin nhắn đầu tiên đi.

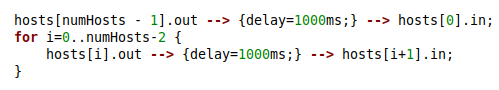
d) Mỗi khi một host nhận được tin nhắn, sẽ tăng biến đếm của mình lên thêm một và forward tin nhắn này sang host còn lại nếu biến đếm chưa đạt đến 5. Nếu đã đạt đến 5 thì xóa tin nhắn đi (bằng cách dùng hàm delete của C++).

2. Tạo ra mạng dạng vòng (ring) với số lượng host tùy ý:



Ở phần này, chúng ta cũng thiết lập ra một định dạng tin nhắn mới, thay vì sử dụng lớp cMessage. Cấu trúc message mới có tên là SimpleMessage (ở trong file có đuôi .msg) và có ba thuộc tính kiểu số nguyên là source, destination, và hopCount, giá trị của hopCount này được gán bằng 0 ở ngay phần khai báo.

Tiếp theo ta cần thực hiện việc khai báo mạng có số lượng các host phụ thuộc vào tham số. Trong network, ta khai báo một biến trong parameters với tên là numHosts, kiểu là int. Sau đó trong submodules, khai báo một mảng các host với số lượng là numHosts. Mảng này, có tên là hosts và có thiết lập kết nối giữa các phần tử như sau:



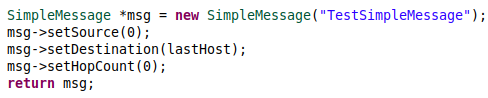
3. Thiết lập việc truyền tin trong mạng ring:

Nhắc lại mạng ring này có yêu cầu truyền tin như sau: host có chỉ số 0 trong mảng sẽ truyền gói tin đi ra khỏi nó, đến nút có chỉ số 1, tiếp tục cho đến khi gặp host có chỉ số (numHosts - 1). Host cuối cùng này sẽ dừng việc forward gói tin, báo cho host có chỉ số 0 rằng nó đã nhận được gói tin, in ra số hopCount đã đi qua và hủy gói tin. Biết rằng cứ mỗi lần gói tin đến được một nút sẽ tiến hành tăng giá trị hopCount của gói tin thêm một. Như vậy ở đây ta có các tình huống khác nhau khi nhận được gói tin như sau:

1. Nếu nút host có chỉ số 0, chỉ nhận được tin nhắn từ host có chỉ số [numHosts - 1] và sẽ tạo ra gói tin mới gửi đi ngay với hopCount = 0
2. Nếu host có chỉ số [numHosts - 1], thì nó sẽ hiển thị ra số hopCount và xóa đi tin nhắn, ngoài ra host này sẽ tạo ra một gói tin mới, gửi đi cho host 0 để báo rằng nó đã nhận được.
3. Các host khác chỉ đơn giản cập nhật số hopCount và forward gói tin

Trong mã nguồn CC (để truy cập được file này cần thực hiện build Project) ứng với SimpleHost, ta khai báo thêm hai hàm mới như sau:

Hàm generateMessage( ) sẽ tạo ra một gói tin với source là 0, và nút đích là chỉ số cực đại của các phần tử trong mảng hosts. Dưới đây mô tả gợi ý một phần nội dung của hàm này



**Hàm generateMessage( )** cần phải xác định được số lượng các host có trong mạng, từ đó biết được đích đến của mình là host có chỉ số (numHosts - 1). Để làm được điều đó, trong hàm cần có câu lệnh xác định chỉ số hiện tại của host trong mảng.

Gợi ý 1: nếu trong file .NED, mảng các host có tên là hosts thì hàm getFullName( ) sẽ trả về được tên mảng cộng với chỉ số của host đó.

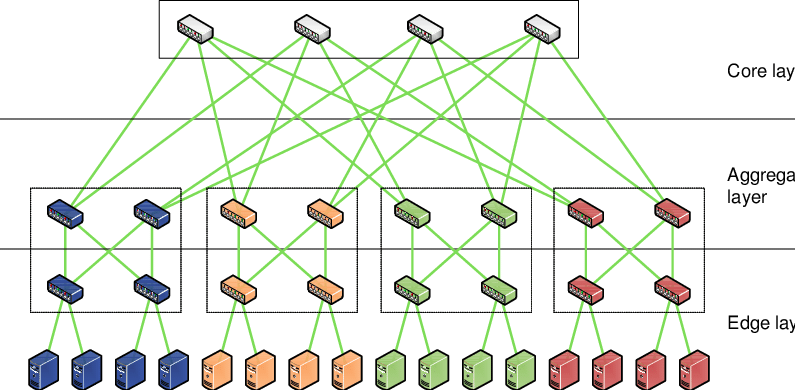
Gợi ý 2: cần lấy được giá trị numHosts của network trong mã nguồn của SimpleHost, vốn là một module con của network. Vậy có thể cần hàm getParentModule để truy cập được đến network từ SimpleHost.cc

**Hàm forwardMessage( )** sẽ được gọi bên trong hàm handleMessage, hàm này có nhiệm vụ cập nhật hopCount của gói tin lên thêm một.

4. Xây dựng file NED cho mạng Fat-tree:

Fat-tree là một cấu trúc mạng được sử dụng phổ biến trong các trung tâm dữ liệu, một mạng fat-tree được chia làm ba tầng khác nhau: Core, Aggregation, và tầng edge. Hình dưới mô tả kiến trúc của mạng fat-tree với số cổng của một switch là k = 4.

Ở kiến trúc Fat-tree, mỗi host (máy tính) kết nối với một switch. Mỗi switch ở tầng Core kết nối với k switch khác. Mỗi switch ở tầng Aggregation kết nối với k switch khác, còn mỗi switch ở tầng Edge thì kết nối với (k/2) máy tính và (k/2) switch khác. Số máy tính trong mạng Fat-tree là k\*k\*k/4 và số switch sẽ là: 5\*k\*k/4



a) Hãy tạo ra một project Omnetpp trong đó file NED là một mạng fat-tree với k = 4.

b) Với một ngôn ngữ tự chọn (C hoặc Java) hãy xây dựng chương trình nhận đầu vào là giá trị k, và dataRate, đầu ra là mã nguồn file NED của mạng Fat-tree với giá trị k đó.

Gợi ý: xem ở đây để biết cách tạo kết nối giữa các node (host và switch) của fat-tree trong trường hợp tổng quát.

<https://github.com/bkhnk48/SSiNET/blob/master/java/custom/fattree/FatTreeGraph.java>