Bài tập 1

Sinh viên thực hiện:

* Nguyễn Đăng Khoa 0512175

# Phân tích và chọn mô hình cho bài toán

* Trong quá trình trao đổi giữa client và server, không có việc đóng kết nối giữa chừng 🡪 chọn kiểu kết nối Connection-Oriented cho cả server và client. Nếu dùng Connectionless thì mỗi lần client thực hiện tính toán xong sẽ ngắt kết nối, và khi server cần đến hàm của client đó thì bắt buộc client phải kết nối lại từ đầu, trái với đề bài vì theo yêu cầu bài toán server chỉ chấp nhận kết nối từ client ở bước đầu tiên, không cho kết nối lại.
* Trong quá trình làm việc client chỉ giao tiếp với một server duy nhất, tại một thời điểm chỉ nhận tối đa 1 yêu cầu từ phía server 🡪 chọn mô hình xử lý tuần tự.
* Tại server
  + Tại bước chờ các client kết nối vào, sau khi client đăng nhập thành công, server cần phải chờ client gửi giấy chứng thực, mà tại thời điểm đó mà có client kết nối vào thì client đó phải chờ (hoặc deadlock nếu server không nhận được giấy chứng nhận) nếu dùng mô hình xử lý tuần tự 🡪 chọn mô hình xử lý giả song song (do việc xử lý sau khi nhận chứng thực không đáng kể)
  + Tại bước chờ nhận kết quả của các client gửi về, server phải chờ nhận đủ kết quả mới thực hiện tiếp được và không có xử lý sau khi nhận 1 kết quả 🡪 chỉ cần chọn mô hình xử lý tuần tự. Nếu có một client không gửi được kết quả về thì sẽ xảy ra tình trạng deadlock, nhưng chọn những mô hình còn lại cũng không giải quyết được vấn đề trên.
  + Chọn nghi thức TCP để đảm bảo input đến được client, các kết quả tới được server vì input, output phải chính xác, không cho phép bị sai lệnh.

# Kịch bản trao đổi giữa Server - Client

|  |  |  |
| --- | --- | --- |
| Bước | Client | Server |
|  | Lặp  {  Nhận thông tin đăng nhập từ người dùng: password  Gửi thông tin đăng nhập gồm (địa chỉ MAC và password) tới SERVER  Nhận thông tin xác nhận đăng nhập (TTXN) từ SERVER  } thỏa TTXN = “thất bại”  Nếu TTXN = “từ chối” hoặc TTXN = “thất bại”  {  Ngắt kết nối  Kết thúc  } | count = 0 (số máy đã kết nối tới máy chủ)  TB = {} (Tập hợp các máy đăng nhập thất bại, mỗi phần tử gồm {địa chỉ MAC, số lần đăng nhập thất bại})  MangCT = {} (tập hợp các giấy chứng nhận được phép sử dụng client)  Lặp  {  Nhân thông tin đăng nhập từ CLIENT (MAC và password)  Nếu MAC có trong TB và số lần thất bại >= 3  {  Gửi thông điệp “từ chối” cho CLIENT  Ngắt kết nối  }  Nếu thông tin đăng nhập hợp lệ (MAC và password đúng với thông tin đã được đăng ký trước đó)  {  Tạo thread  {  Gửi thông điệp “thành công” cho CLIENT  Nhận CT từ CLIENT  MangCT = MangCT ∪ {CT}  count = count + 1  }  }  Ngược lại  {  Ghi nhận máy đăng nhập thất bại (TB = TB ∪ {MAC, số lần thất bại})  Gửi thông điệp “thất bại” cho CLIENT  }  } thỏa (count < số node đã qui định hay còn deadline) |
|  | Gửi thông tin chứng thực được phép sử dụng client (CT) lên SERVER  Chờ nhận hàm từ server | Tách hàm Fractal thành m hàm (m: số node đã kết nối thành công) |
|  | Lặp  {  Chờ nhận yêu cầu (yc, yc = {“đóng kết nối”, input + thông tin chứng thực MangCT[p]}) từ SERVER  Nếu yc = “đóng kết nối”  Thoát khỏi vòng lặp  Nếu MangCT[p] khác với CLIENT  Gửi thông điệp “từ chối thực thi” tới SERVER  Ngược lại  output = kết quả của hàm sử lý với đầu vào là input vừa nhận  Gửi output về cho SERVER  } thỏa còn thỏa deadline  Đóng kết nối | Lặp m lần  {  Gửi hàm i tới máy i tương ứng (i = 1, 2, …, m)  } |
|  |  | Chọn k máy (0 < k <= 3)  Lặp k lần  {  Gửi input + MangCT[p] cho CLIENT (MangCT[p]: thông tin chứng thực của CLIENT tương ứng)  } |
|  |  | Dem = 0 (số máy đã trả về output cho SERVER)  MangOutput = {} (tập hợp các output)  Lặp  {  Nhận kết quả output từ CLIENT  MangOutput = MangOutput ∪ output  Dem = Dem + 1  } thỏa Dem < k |
|  |  | Vẽ 1 phần đồ thị Fractal  Nếu đã vẽ xong  {  Gửi thông báo “đóng kết nối” đến các CLIENT  Đóng tất cả kết nối đến các CLIENT  }  Ngược lại  {  Chuyển các output thành input tiếp theo  Thực hiện bước 4  } |

# Giải pháp chịu lỗi khi có một máy node bị đứt kết nối

* Tại bước 5 của server, nếu mà quá deadline thì server sẽ xem như client đó bị mất 🡪 đóng kết nối với client đó, lấy hàm ứng với client đó gửi cho 1 client khác (client này phải rảnh), rồi gửi input cho client đó rồi quay lại bước 5. Tại thời điểm này server sẽ luôn giữ 1 hoặc 1 số hàm không có client đảm trách, do đó nếu server cần dùng những hàm như thế thì server sẽ gửi hàm đó cho một client nào rảnh, rồi làm tương tự như phần trên.
* Tại bước 3 của client, ngoài chờ nhận những yêu cầu đã nêu, client còn chờ nhận thêm hàm mới, nếu client nhận được hàm mới thì nó tiến hành nạp lại hàm mới và quay lại bước 3.