BÀI TẬP TRÊN LỚP
MÔN HỌC: HỆ PHÂN TÁN CHƯƠNG 4: TRAO ĐỔI THÔNG TIN

HỌ TÊN SV: Mạc Quang Huy MSSV: 20173169
MÃ LỚP: 118663 MÃ HỌC PHẦN: IT4611

Câu hỏi 1: Trong các giao thức phân tầng, mỗi tầng sẽ có một header riêng. Vậy có nên triển khai một hệ thống mà tất cả các header của các tầng đưa chung vào một phần (gọi là header chung), gắn vào đầu mỗi thông điệp để có thể xử lý chung? Giải thích.

Trả lời:

- Không nên vì khi đó hệ thống mất đi ý nghĩa phân tầng, các tầng độc lập với nhau. Ngoài ra dùng header chung sẽ mất đi tính trong suốt giữa các tầng

Câu hỏi 2: Xét 1 thủ tục *incr* với 2 tham số nguyên. Thủ tục làm nhiệm vụ là cộng 2 tham số đó với nhau. Bây giờ xét trường hợp chúng ta gọi thủ tục đó với cùng một biến 2 lần, ví dụ *incr(i, i).* Nếu biến *i* được khởi tạo giá trị *0*, vậy giá trị của *i* sẽ là bao nhiêu sau khi gọi thủ tục này trong 2 trường hợp sau:
- Lời gọi tham chiếu
- Phương pháp sao chép-phục hồi được sử dụng.

Trả lời:

- TH1: Lời gọi tham chiếu

Mỗi con trỏ đều trỏ tới i, nếu i tăng 1 đơn vị => i tăng 2 lần => i=2

- TH2: Phương pháp sao chép-phục hồi sử dụng

Sao lưu không làm tăng giá trị. Ghi đè 2 lần => i=1

Câu hỏi 3: Một kết nối socket cần 4 thông tin nào? Tại sao phải cần đủ 4 thông tin đó?

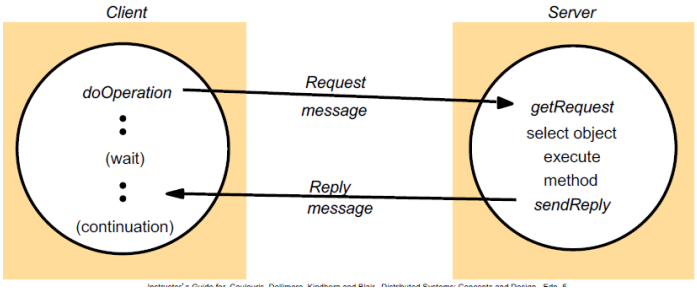
Trả lời:

- Vì một socket được đại diện bởi 2 thông tin: IP máy + Port lắng nghe. Phía server phải chọn port ứng với mỗi socket. Bên phía client port được xác định bởi OS

⇨ Một socket connection được đại diện bởi 4 thông tin:
IP máy gửi + Port gửi
IP máy nhận + Port nhận

Câu hỏi 4: Tại sao giao thức *yêu cầu-trả lời (request-reply)* lại được coi là *đồng bộ* và *tin cậy*?

Trả lời:



- Client gửi xong tự block mình chờ reply => Đồng bộ từng cặp request/reply => Giao thức đồng bộ
- Coi reply là báo nhận cho request đó => Giao thức tin cậy mà không cần báo nhận ACK

Câu hỏi 5: Hai vấn đề chính đối với giao thức RPC là gì?

Trả lời:

- Đây là hệ thống không đồng nhất: Có không gian nhớ, cách thức biểu diễn thông tin khác nhau, xày ra vấn đề với truyền tham số
- Có lỗi xảy ra khi gọi RPC:

- Máy 1 gọi máy 2, máy 2 treo, máy 1 cứ block chờ
- Kết nối đứt, thông điệp không về lại máy 1
- Máy 1 bị treo lúc máy 2 gửi kết quả

Câu hỏi 6: Vấn đề đối với truyền *tham biến* trong RPC là gì? Còn đối với truyền *tham chiếu*? Giải pháp đưa ra là gì?

Trả lời:

- Tham biến: Vấn đề biểu diễn dữ liệu khác nhau ứng với mỗi dòng máy (Kiểu little endian, kiểu big endian)
- Tham chiếu: Bộ nhớ bị phân tán

⇨ Sử dụng giải pháp sao lưu, phục hồi, bỏ ra n bytes để ghi nhận => Tốn băng thông

⇨ Giải pháp: Xây dựng 1 thủ tục để 2 bên hiểu nhau, dựa vào tính mở của RPC xây dựng một interface cho client-stub và server-stub hiểu nhau

Câu hỏi 7: So sánh RMI và RPC. Nhược điểm của RMI so với RPC là gì?

Trả lời:

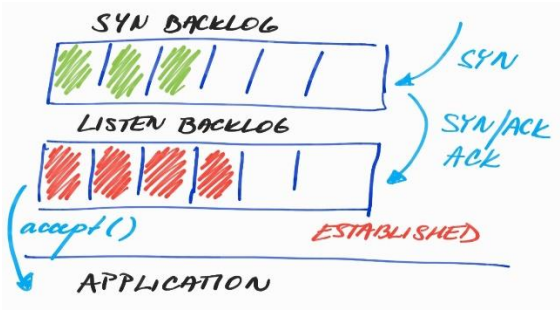
	RMI	RPC
Giống	Hỗ trợ lập trình giao diện Dựa theo giao thức request/reply Cùng mức độ trong suốt	
Khác	Lập trình viên có thể khai thác sử dụng hết điểm mạnh OOP Định danh duy nhất => Truyền tham chiếu	Tính mở cao

Nhược điểm của RMI so với RPC: Chỉ dùng cho java => Tính mở không cao như RPC

Câu hỏi 8: Hàm *listen* được sử dụng bởi TCP server có tham số là *backlog*. Giải thích ý nghĩa tham số đó.

Trả lời:

- Server quản lý 2 hàng đợi: Hàng đợi hoàn thành và hàng đợi chưa hoàn thành.
- Khi client gọi connect() => Gửi SYN lên server, server tạo 1 bản ghi trong queue chưa hoàn thành. Gửi lại SYN/ACK, khi server nhận ACK đẩy nó ra queue hoàn thành



- Hàm accept() lần lượt lấy bản ghi trong queue hoàn thành. Nếu hàng đợi đó rỗng => block chương trình Server

⇨ Int backlog: Số bản ghi lớn nhất của 2 hàng đợi

Câu hỏi 9: Trong trao đổi thông tin hướng dòng, những cơ chế thực thi QoS được thực hiện ở tầng nào? Giải thích. Trình bày một số cơ chế thực thi QoS để chứng minh điều đó.

Trả lời:

- Được thực thi trên tầng IP: Đơn giản, best-effort
- Ví dụ thực thi QOS: Differentiated