**1. Middleware là gì và vai trò của nó trong hệ thống phân tán?**

Middleware trong hệ thống phân tán là một loại phần mềm hệ thống phân tán kết nối các loại ứng dụng khác nhau và cung cấp phân tán minh bạch cho các ứng dụng của nó kết nối.

Nó được sử dụng để nối thành phần không đồng nhất xuất hiện trong các hệ thống

Giao tiếp giữa các thành phần

Quản lý giao dịch

Độc lập nền tảng

Đảm bảo an toàn

Quản lý và điều phối dịch vụ

**2. Các loại middleware phổ biến trong hệ thống phân tán là gì?**

Dựa trên các tiêu chuẩn hoặc các sản phẩm trên thị trường, middleware có thể được chia thành nhiều loại: Socket, RPC, RMI, DCE, DCOM, CORBA v.v...

**3. Socket là gì và vai trò của nó trong giao tiếp mạng?**

Socket là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ (client) và một chương trình cung cấp dịch vụ (server) trên mạng.

Vai trò của Socket trong giao tiếp mạng:

Giao tiếp trực tiếp giữa các máy tính: Socket cho phép các tiến trình trên hai máy tính khác nhau giao tiếp trực tiếp với nhau thông qua mạng.

Xử lý dữ liệu theo thời gian thực: Sockets hỗ trợ giao tiếp thời gian thực giữa client và server, giúp truyền tải thông điệp hoặc dữ liệu một cách liên tục mà không cần độ trễ lớn.

Quản lý kết nối mạng: Cung cấp các hàm API để thiết lập, quản lý và đóng kết nối TCP hoặc UDP.

Xử lý dữ liệu song song: Các tiến trình hoặc luồng xử lý có thể hoạt động song song nhờ việc quản lý kết nối và dữ liệu qua các socket.\

**4. So sánh giữa sử dụng socket và middleware trong việc xây dựng hệ thống phân tán?**

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Socket | Middleware |
| Cấp độ trừu tượng | Thấp, gần với phần cứng và mạng. | Cao, trừu tượng hóa giao tiếp và dịch vụ mạng. |
| |  | | --- | | Quản lý kết nối |  |  | | --- | |  | | Quản lý thủ công, lập trình viên phải quản lý việc tạo, duy trì, và đóng kết nối. | Được quản lý tự động bởi middleware, lập trình viên không cần lo lắng về chi tiết này. |
| |  | | --- | | Giao tiếp |  |  | | --- | |  | | Thường sử dụng mô hình client-server, hoặc peer-to-peer. | Hỗ trợ nhiều mô hình giao tiếp phức tạp hơn như publish-subscribe, request-response. |
| |  | | --- | | Hiệu năng |  |  | | --- | |  | |  | |  |  | | --- | |  | | Cao, phù hợp với các ứng dụng yêu cầu thời gian thực, tốc độ cao. | Thường chậm hơn do mức độ trừu tượng cao hơn, nhưng dễ phát triển và bảo trì. |
| |  | | --- | | Phức tạp |  |  | | --- | |  | | Phức tạp hơn do lập trình viên phải xử lý chi tiết giao tiếp. | Ít phức tạp hơn do middleware xử lý nhiều chi tiết kỹ thuật. |
| Phạm vi sử dụng | Dùng trong các hệ thống yêu cầu kiểm soát tối đa về hiệu suất và chi tiết giao tiếp. | Thường được sử dụng trong các hệ thống doanh nghiệp và phân tán phức tạp. |

**5. Quy trình thiết lập và sử dụng socket trong lập trình mạng?**

**Bên Server**:

**Tạo socket server**: Tạo một đối tượng ServerSocket với một số cổng xác định để lắng nghe các kết nối từ client.

**Chờ kết nối từ client**: Gọi phương thức accept() để chờ client kết nối.

**Thiết lập luồng giao tiếp**: Sau khi nhận được kết nối từ client, thiết lập các luồng vào (input stream) và ra (output stream) để giao tiếp.

**Gửi và nhận dữ liệu**: Dùng các luồng (stream) để gửi và nhận dữ liệu qua kết nối.

**Đóng kết nối**: Sau khi hoàn tất giao tiếp, đóng các luồng và socket.

**Bên Client**:

**Tạo socket client**: Tạo một đối tượng Socket và kết nối tới server bằng cách cung cấp địa chỉ IP và số cổng.

**Thiết lập luồng giao tiếp**: Thiết lập các luồng vào và ra để giao tiếp với server.

**Gửi và nhận dữ liệu**: Gửi dữ liệu tới server và nhận phản hồi.

**Đóng kết nối**: Đóng các luồng và socket sau khi hoàn thành.