Resuminho II

Arquitetura de Aplicação

Existem três modelos principais para estruturar a comunicação entre esses processos:

1. Arquitetura Cliente-Servidor

Este é o modelo mais tradicional. Nele, há um **servidor** que é um hospedeiro sempre ligado, com um endereço IP fixo e conhecido. Sua capacidade pode ser expandida através de *server farms* (conjuntos de servidores).
Os **clientes** são os processos que iniciam a comunicação para solicitar serviços a este servidor. Eles podem ter endereços IP dinâmicos, conectar-se à rede de forma intermitente e não se comunicam diretamente entre si; toda a interação é intermediada pelo servidor.

2. Arquitetura Peer-to-Peer (P2P) Pura

Na arquitetura P2P, não existe um servidor central sempre ativo. A comunicação ocorre diretamente entre pares (peers), que são sistemas finais arbitrários na rede. Esses pares se conectam de forma intermitente e podem mudar de endereço IP com frequência.
Este modelo é altamente escalável, pois cada novo par adiciona capacidade à rede. No entanto, é consideravelmente mais difícil de administrar devido à sua natureza descentralizada e dinâmica.

3. Arquitetura Híbrida (Cliente-Servidor e P2P)

Este modelo combina elementos das duas arquiteturas anteriores para aproveitar o melhor de cada uma. Um exemplo clássico é o Skype (em suas versões mais antigas) e aplicações de mensagem instantânea.
Nesse modelo, um **serviço centralizado** (servidor) é usado para tarefas de coordenação, como localizar o endereço IP de outros usuários ou detectar se eles estão online. No entanto, a troca de dados principal — como uma chamada de voz ou o bate-papo em si — acontece em uma **conexão direta P2P** entre os clientes, sem passar pelo servidor. Isso alivia a carga do servidor central e melhora a eficiência.

Comunicação de Processos

Em redes de computadores, a comunicação ocorre entre processos, que são programas em execução dentro de um dispositivo hospedeiro (como um computador ou smartphone, estamos vendo isso em SO, vejam as aulas do Norton crianças). Quando dois processos no mesmo dispositivo precisam se comunicar, eles utilizam mecanismos de comunicação entre processos,

que são gerenciados pelo sistema operacional.
No entanto, quando os processos estão em dispositivos diferentes, eles se comunicam trocando mensagens pela rede. Essa comunicação é geralmente iniciada por um **processo cliente**, que busca o serviço. Do outro lado, um **processo servidor** aguarda ser contatado para fornecer o serviço.
OBS: o autor confude processo com thread no livro, isso é corrigido em edições futuras.

Sockets: A Porta para a Rede

Para que um processo possa enviar e receber mensagens pela rede, ele utiliza uma interface chamada **socket**. Pense no socket como a porta de uma casa: é por onde as mensagens entram e saem do processo.
O processo que deseja enviar uma mensagem a "empurra" através do seu socket. A infraestrutura de transporte da rede (controlada pelo sistema operacional, com seus buffers e protocolos como o TCP) se encarrega de levar essa mensagem até o socket do processo receptor. O socket funciona, portanto, como uma API (Interface de Programação de Aplicação) que permite ao desenvolvedor da aplicação escolher protocolos de transporte e definir parâmetros para a comunicação.

Endereçando Processos

Para que um processo (um programa em execução) possa receber mensagens através de uma rede, ele precisa ter um identificador único.
Cada dispositivo hospedeiro na rede possui um endereço IP exclusivo (de 32 bits, no caso do IPv4, que é o do qual estamos conversando) que o identifica. No entanto, o endereço IP por si só não é suficiente para identificar um processo específico. A razão para isso é que muitos processos diferentes podem estar rodando simultaneamente no mesmo dispositivo hospedeiro. Portanto, para que a comunicação seja direcionada ao programa correto, o identificador de um processo deve incluir dois elementos:

- O **endereço IP** do dispositivo hospedeiro onde o processo está sendo executado.
- Um **número de porta** associado especificamente àquele processo dentro do hospedeiro. O número da porta funciona como um ramal, garantindo que, uma vez que a mensagem chegue à máquina correta (via IP), ela seja entregue ao aplicativo certo. Por exemplo, servidores web (HTTP) normalmente usam a porta 80, enquanto servidores de correio eletrônico costumam usar a porta 25.