**RESUMO S.C. Distribuidos – AULA 23/04/2021**

Comunicação em um ambiente de computação distribuida.

Quando falamos de comunicação entre processos, deve-se lembrar que em um sistema distribuido é a partir de troca mensagens, ou seja a base dos processos distribuidos seria a própria comunicação entre os processos.

Troca de mensagens é mais difícil do que usar primitivas baseadas em memória compartilhada. Ele lida com tolerancia a falhas, com seguranças, por isso deve ser tratato dentro de um sistema distríbuido.

“*Desejável obter modelos onde a complexidada da comunicação seja transparente para o desenvolvedor*” Os sistemas modernos tem o conceito de sistema distribuidos dentro deles, o linux, mac os, várias funcionalidades de sistemas distribuidos já estão definidas dentro desses S. O’s, portanto tiramos a responsabilidade total para aplicação e entregamos uma parte dele para o proprio S.O fazer esse trabalho. O middleware também tem responsabilidade nesse processo.

*“Os processos que se comunicam têm que obedecer às regras estabelecidas”* – protocolos. Eles são regras que devem ser seguidas, o que deve ser feito para que essa comunicação aconteça. Então se escolhemos um protocolo, toda a nossa rede de s.d, todos equipamentos e aplicações devem seguir/usar o protocolo que foi definido.

O que podemos falar de protocolos? Um exemplo é a transferencia de arquivo na internet. Tendo o filezila como exemplo, é um sw que faz transferencia de arquivos de um lugar para o outro, mas usa como base o protocólo FTP, então FTP é um protocolo que tem na sua estruturação regras de como deve acontecer a transmissão em um ambiente/cenário/aplicação. Então o protocólo é isso, um conjunto de regras para que a atividade/tarefa seja executada seguindo a padronização de regras.

“*Processo A monta um msg em seu próprio espaço de enderaçamento, executa uma chamada de sistema, SO envia a msg....” Temos o processo A e o processo B*



Temos uma troca de mensagns que A se comunica com B e B se comunica com A. Então como que vai acontecer? Tendo uma mensagem escrita em string em A e B não souber ler essa string, caracteriza-se que A não teve como definição padrão do mesmo tipo que B, ou seja, não seguiram um protocolo.

“*Vários acordos diferentes necessários: Quantos volts são necessários para sinalizar um bit 0 ou 1... como o receptos..*.”

*“Para ficar mais facil lidar com os vários níveis... ISO desenvolv...”*

Modelo osi estabelece através de camadas o que deve ser feito em cada camada, são elas apresentação, enlace etc. Sendo 7 camadas/7Níveis, que fornecem uma interface que estão acima dela. Modelo Osi é um modelo de **referencia** diferente de protocolos. [Modelo de referencia para comunicação em redes do qual não é tão aplicado em relação aos sw em geral, acaba sendo mais conceitual, diferente do TCP que é mais utilizado no mercado e é a base de muitos sistemas que estão disponíveis no mercado]

*“Protocolo TCP / UDP “*

Eles têm suas características com vantagens e devantagens, conforme citado no slide. Um acaba tendo mais segurança que o outro, porém acaba sendo mais lento (TCP). (UDP) inverso, sem conexão, não confiável por não tratar as questões referentes a segurança no momento de comunicação.

Não é porque o UDP n é seguro, que n será utilizado, n é pq o TCP é lento q n sera utilizado. UDP qnd não temos uma propagação de acesso, que vai rodar dentro da empresa, que vai linkar um equipamento com uma automação, o UDP se encaixaria bem. Agora propagar algo que muitas pessoas terão acesso, é adequado trabalhar com um protocolo mais confiável, q por mais que seja mais lento garante uma segurança para o usuário.

*“camada de aplicação”*

Camada de aplicação tem toda uma comunicação com o user ...

Por trás do Fillezilla temos *o “FTP que define um protocolo para transferir arquivos entre um cliente e um servidor. Ele define as regras para as transferencias desses arquivos, o FTP é uma aplicação que o usuário utiliza para transferir esses arquivos baseados no protocolo ftp”*

A mesma coisa é no HTTP, que é um protocolo de internet para a parte de manipulação de hipertexto, paginas web, do qual permite a manipulação de páginas html. Baseado nisso podemos perceber que ao digitar o inicio do site no http, ele completa no navegador. Mas o papel do protocolo é garantir as regras para a apresentação e manipulação de páginas webs, das quais utilizam esse conceito de hyper texto.

Nessa parte é interessante entender a diferença da APLICAÇÃO e do PROTOCOLO, do qual a aplicação utiliza das regras do protocolo

*Middleware*

Aplicação q roda no meio que serve como intermediario que serve como um intermediário da aplicação e da parte física. (Parte que o usuario interage e onde terá toda a sua execução)

*“o objetivo do middleware...”*

O usuari nem sabe da existencia do middleware, porém ele tem um papel extremamente importante porque ele que gerencia tanto os serviços como os protocolos que estão snedo distribuidos, tem um conjunto rico de protocolos que não é composta apenas por um protocolo, podemos ter outros protocolos. Exemplo, podemos ter algo rodando de pagina web, podemos ter o HTTP disponível, e se transferir arquivos precisamos do FTP, e assim por diante. “normativa > ftp, http”

*“Marshing”*

Quando falamos de marshing de dados, falamos da padronização. Q para acontecer a comunicação entre 2 processos, tem que existir um contrato entre eles para definir quais são as regras de comunicação como por exemplo como os dados vão trafegar de um lado pro outro. Quando trabalhamos com o modelo cliente e servidor, os dados não poderiam ser transferidos no formato de string, tinham que ser convertidos para bytes que é a padronização definida para SOCKET, n adianta tentar ocnverter para outro padrão sendo que o padrõa para aquela situação for outro.

*“https vs http”*

https vem com a premissa da segurança encorporada.

*“mecanismos de dimensioanamento para replicação e caching”*

Middleware tem q garantir o dimensionamento para o sistema distribuido, que através de um processo de caching pode acontecer, expansão de memória etc, tudo isso pode ser trabalhado dentro do s.d e o responsável por todo esse trabalho é a camada de middleware, que é intermediaria entre a *Aplicação e a Camada Física*.

No Middleware encontramos o DNS.

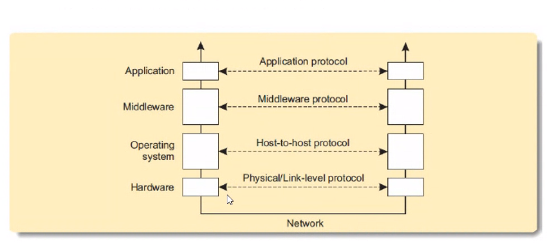
*“Procura um endereço de rede associado a um nome. No modelo osi.... “*

Então quer dizer uqe no TCP ou no UDP a parte do DNS é independente do protocolo.

*“Protocolso de autorização para garantir....”*

Middleware também é responsável para fazer o direcionamneto e o controle de qm tem acesso ao q

*“Protocolo para acessar acesso compartilhado”*



...

Temos o hw q é o instrumento de camada física.

A rede que estabelece o transporte de envio e conteúdo das mensagnes que serão trocadas entre os processos

“ as camadas de apresentação e sessão foram subs... “

As camadas de apresentação e sessão são trabalhadas como middleware

“ as camadas de transporte e rede forma agrupadas ... “

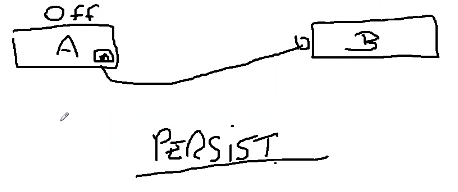
Os s. o’s modernos trazem muitos recursos relacionados a sistemas distribuidos, então oq poderia ser tratado na aplicação, já está sendo bem tratado dentro do S.O., então ele tem uma importancia mt grande no S.D pq ele permite q essas interações acontecam e de forma segura para garantir toda essa sincronia.

*Tipo de comunicação*

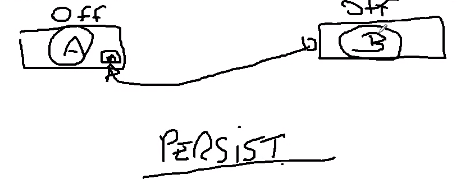
*“Persistencia”*

*“ uma mensagem é armazenada pelo middleawre..”*

Oq é uma comunicação persistente?



Ele pode ficar off dps de ter enviado a msg.



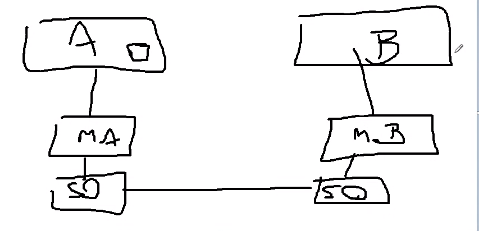
Ela pode ser enviada de um processo, entrar em uma fila, e depois o processo ser enviado quando disponível.

2. Os dois lados n precisam estar disponíveis para q a mensagem seja enviada, porém n sera recebida até q esteja disponivel, será guardada até q o processo ou aplicação esteja disponível”

*Transiente*

*“Uma mensagem é armazenada pelo middleware de comunicacao somente ... “*

É diferente da persistente q um pode estar **offline** para q a comunicação aconteça. Já na transiente acontece outro tipo de situação:



Todas devem estar disponíveis (A e B) para que a comunicação aconteça;

Se os processos n estiver online a mensagem se perde, sendo descartada no caso.

A MAIORIA **usa persistente**. Conceito de email utiliza a persistente, já q dispara um email e fica parado lá até q cosniga ser disparado o email e a pessoa consiga receber. É esse tipo de comunicação q é estabelecida. Ou seja, **A MELHOR no caso** vai depender do q a aplicação vai fazer.

O middleware deve **gerenciar** as mensagens para q elas sejam entregadas qnd a aplicação estiver disponível. Então fica mais complicado a complexidade de implementação do middleware.

*“sincronizacao comunicação assincrona “*

Assincrona dispara a mensagem o middleware vai armazenar temporariamente aquela mensagem até q ela seje entregue ao receptor da mensagem, no caso não a pessoa mas sim o processo q vai interagir com a pessoa/usuario

*Comunicacao sincrona*

N consegue fazre outras funcionalidades até q o processamento seja retornado pra vc.

Esta relacionada a entrega da mensagem e a confirmação de que a mensagem foi recebida.

*“EXEMPLO 1 COMPUTAÇÃO CLIENTE/SERVIDOR É GERAL...”*

Os dois lados precisam estar ativos para fazer a comunicacao entre cliente/servidor. É geralmente baseada no modelo de comunicação transiente síncrona.

Tentando mandar umam ensagem pro servidor e o cliente n está rodando, daria uma mensagem q o servidor n esteja disponível

**Então para q aconteça** os DOIS lados tem q estar se comunicando. E para estar se comunicando voltamos para o lado de TRANSIENTE/Sícrona.

*“CLIENTE E SERVIDOR DEVEM ESTAR ATIVOS NO MOMENTO DA COMUNICACAO”*

*“CLIENTE SUBMETE UM PEDIDO E BLOQUEIA ATÉ Q RECEBA UMA RESPOSTA”*

**TRANSIENTE/SINCRONA**

Qnd clicamos no botao salvar e o ... o sistema fica inoperante, fica bloqueado até q receba um retorno do servidor. Qnd recebe o retorno do servidor esta liberado para continuar as proximas interações, é o conceito de comunicação sincrona. Pois o remetente é bloqueado...

A partir do momento q transmitiou a requisisao e o servidor receberu, isso é bloqeuado, o sv processa e retorna pro cliente, e a apartir de ter retornado para o cliente, o sistema é destravado para continuar a ser usado

“Servidor essencialmente fica...”

Mandava uma mensagme pro cliente, o sv mostrava na tela pro cliente e retornava pro servidor q a mensagem foi recebida. (TRANSIENTE SINCRONA) os dois tem q estar funcionando, cliente e servidor. E QND MANDAMOS ALGO PRO SV, o cliente fica travado, espernado um retorno do Servidor.

*“Falhas devem ser tratadas imediatamente .. “*

Falha deve ser tratada de forma emergencial

*“Para alguas aplicações não é apropriado “*

Onde n aplicamos transiente sincrona? Correio é um exemplo. PQ o conceito do email eh mandar uma msg, desconectar e se o receptor estivero u nconectado, ele vai receber essa msg, ent nesse caso n serve o SINCRONO/TRANSIENTE.

Tem várias vantagens e desvantagens, mas depende do q o processo vai fazer, n tendo como escapar da situação.

*EXEMPLO 2*

Processos enviam mensagens uns aos outros que são enfileiradas”

Vai entrando em uma fila onde as mensagnes vão ser distribuidas conforma a solicitações q foram feitas.

*“Remetenten envai uma mensagemm...”*

Nesse caso ai, é característico que esse tipo de comunicação assincrona. Q o remetente envia a mensagem, e difernete do cliente servidor q ambos precisam estar conectados, nesse caso não precisa q esteja conectado no caso.

*“ O middleware deve oferecer tolerancia a falhas “*

Ah temos uma aplicação q ambos n precisam estar conectados para q aja comunicação. Mas perceba q isso, tem uma sobrecarga no middleware. Então terá um trabalho maior para ficar gerenciando quando ele deve entregar ou n essas mensagens q foram enviadas para os seus receptores. Tudo isso deve ser avaliado no momento em q está trabalhando em um sistema e tem q definir q comunicação q será utilizado.

O SISTEMA DISTRIBUIDO Q VAMOS DESENVOLVER, e o tipo para q seja escolhido o tipo de comunicação.

**RPC – chamada de procedimento Remoto.**

Oq é isso? O nome já diz, quando falamos de chamada de procedimento remoto, falamos de chamar um procedimento que se encontra em outra máquna, em outro servidor, em outro equipamento, em outra aplicação que está rodando outro equipamento. Por isso a chamada de procedimento remoto.

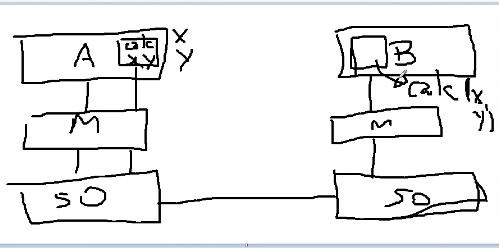
Vamos descrever uma classe conexão q será utilizada tanto para o cliente qnt para o servidoro, porém elas não vão conversar para compartilhar esses métodos.

Então pegamos a conexão q escrevemos no cliente e vamos fazer uma cópia entre eles, porém sendo iguais e n temos trabalhado o conceito de RPC, n tem problema.

Copia da classe conexão tanto qnt no cliente e no servidor, as classes eram iguais, um send e enviar, q recebia as mensagens e enviava as mensagens. Agora quando falamos de chamada de procedimento remoto, podemos chamar uma função/metodo de uma aplicação para q ela seja usada em outra aplicação. Essa é a ideia de RPC.

*“Permite a processos chamar procedimentos localizados em outras máquians “*

N precisa ser só em outras máquinas, mas tb em outras aplicações. No processo X caso queira ter acesso no processo Y, entra no conceito de RPC, mas fazendo uma chamada remota de um metodo.



B recebe a mensagem, desempacota, calcula e gera um resultado. Dps gera or esultado para o remetente. Então é dessa forma q é acontece o encaminhamento do RPC.

O processo, consegue se comunicar com o outro processo, para que aquela função seja executada. N teriamos nocaso, a duplicidade de funções, porq um chama o outro e conseguimos trabalhar com eles.

*“UM PROCESSO da máquina A chama um procedimento da Máquina B (CALC “*

O Q ELE FLW DO CALC

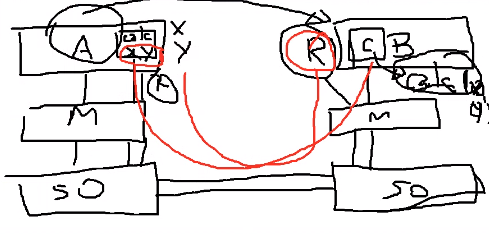
*“ Informações .... “*

O X e o Y q passou como parametro e o R q trouxe como resultado, são informações...

Se fizermos uma alteração no modo de envio, se tivermos nos dois lados, mexemos nos dois lados, AGORA SE TIVER O MÉTODO APENAS EM UM LUGAR, e lee sendo chamado pelo RPC, vamos fazer a alteração localmente só onde esta o metodo onde esta armazenado, o rstante só consome aquele metodo.

*“STUB DO CLIENTE”*

Qual é o papel desse apendice do cliente?



O processo A cria o pacote, q cria uma função q vai ser executada e coloca os parametros que vão ser passados para serem utilizados na execução do procedimento da função, então qm faz esse trabalho dentro do processo, é o stub do cliente.

*“STUB DO SERVIDOR”*

Então do outro lado chega o processo q é enviado pelo processo A + os parametros, o stub do servidor vai desempacotar os parametros que chegaram com os parametros e funções para q possa ser executado.

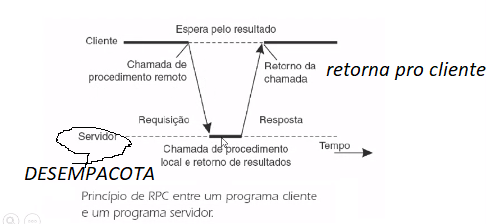
Então temos o Stub do cliente (A) e stub do Servidor (B)

É um apendice q entra dentro da execução desses processos, pensando no conceito de transparencia para o desenvolvedor.

*“ Em um sistema tradicional... read é um procedimento q faz uma chamada de sistema..“*

Entao quando usamos o **read** normal, envia uma mensagem para um outro processo, nesse caso, do RPC essa forma de envio, esse método de envio, é adaptado para que essa mensagem seja empacotada e transferida para o servidor.

Ent no stub qnd envia o pacote, ele fica bloqueado, enquanto n voltar o retorno pro A, ele não é desbloqueado, então tem q ser enviado o retorno para o A para que seja desbloqueado.



*“ qnd a mensagem chega ao servidor, o SO... “*

Então ela sai do A , vai pro middleware, vai pro SO, o SO envia pro outro SO dai ela é encaminhada para o Stub do servidor, q é equivalente ao stub do cliente, transformando requisições q ve pela rede desempacotando-as e executando a chamada no servidor.

*“O resultado é armazenado em um buffer”*

Stubb faz o papel de empacotar o conteudo e transporta para o outro lado. Desempacota, processa e devolve o resultado. [stubb dentro da rpc]