1) Protocolo de aplicação: São protocolos projetados para fornecer funcionalidade requeridas para as aplicações do utilizador
Protocolo de middleware: São protocolos projetados para fornecer meios de comunicação e gerenciamento de dados entre os aplicativos

No protocolo de aplicação:
- Foco apenas na passagem de mensagens/pacotes
- Várias funcionalidades desnecessárias (a muitas aplicações)
- Não provê transparência de acesso

Quando é uma coisa mais especifica, podemos implementar protocolo de aplicação colocando protocolos especificos para a aplicação. Ex: Obter temperatura, calculadora, umidade do ar, etc

Com o middleware, tudo isso é evitado:
- Um conjunto rico de protocolos de comunicação
- Marshaling/Unmarshaling dos dados – necessário para integração dos componentes em sistemas heterogêneos
- Protocolos de resolução de nomes, para facilitar a descoberta e o compartilhamento de recursos
- Protocolos de segurança, para comunicação segura
- Mecanismos de escalabilidade, tais como replicação e caching

O middleware é uma coisa mais geral inves de ser algo especifico, ele consegue tratar todas as aplicações de uma forma geral

Ex protocolo de aplicação: serviços HTTP, troca ou transferência de arquivos
Ex protocolo de middleware: cliente-servidor, enviar formulários em um navegador da Web ou permitir que o servidor Web apresente páginas dinâmicas da web com base no perfil de um usuário.

2) Sincronização na submissão da request, Sincronização na entrega da request e Sincronização depois do processamento pelo servidor, todas essas sincronizações depende se é sincrona ou assincrona ou se são transiente ou persistentes.
Sincronização na submissão da request: Seria vantagem se fosse assincrona porque depois de saber que o servidor recebeu a request, ele poderia fazer outra operação, se fosse sincrona seria desvantagem porque ele continuaria esperando a resposta do servidor pro cliente
Sincronização na entrega da request: Seria vantagem se fosse assincrona porque depois de saber que o servidor recebeu a request, ele poderia fazer outra operação, se fosse sincrona seria desvantagem porque ele continuaria esperando a resposta do servidor pro cliente
Sincronização depois do processamento pelo servidor: Seria vantagem se fosse sincrona ou assincrona porque depois de saber que o servidor respondeu a request do cliente, ele poderia fazer outra operação

3) O problema da representação de dados em sistemas distribuídos heterogêneos é que quando fala de heterogeneidade quer dizer que os sistemas podem ter sistemas completamentes diferentes como SO, processamento, memoria, são maquinas muito diferentes, diferente hardware, então quando queremos nos comunicar com esses sistemas cada um deve ter um padrão de mensagem pra receber ou uma representação de dados, para resolver essa situação usamos RPC com o Marshaling que basicamente converte essa representação de dados por padrão, empacota essa chamada em mensagem e faz a serialização da mensagem ai o sistema que pegar esses dados faz o unmarshaling e recebe a mensagem

4)

ACOPLAMENTO TEMPORAL: O destinatário (ou destinatários) deve existir nesse momento no tempo.
DESACOPLAMENTO TEMPORAL: O remetente (ou remetentes) e o destinatário (ou destinatários) podem ter tempos de vida independentes
ACOPLAMENTO ESPACIAL: Comunicação direcionada para determinado destinatário (ou destinatários)
DESACOPLAMENTO ESPACIAL: O remetente não precisa conhecer a identidade do destinatário (ou destinatários)

acoplado temporalmente + acoplado referencialmente: Direto
desacoplado temporalmente + acoplado referencialmente: Caixa de correio
acoplado temporalmente + desacoplado referencialmente: Baseada em eventos
desacoplado temporalmente + desacoplado referencialmente: Espaço de dados compartilhados

Ex de aplicação:
acoplado temporalmente + acoplado referencialmente: Jogo
desacoplado temporalmente + acoplado referencialmente: Caixa de email
acoplado temporalmente + desacoplado referencialmente: youtube(inscrito)
desacoplado temporalmente + desacoplado referencialmente: Busca online

5) Comunicação Assíncrona

Desvantagens:
- Requer uso de computador
- Sem Interação - a comunicação não é feita em tempo real
- Maior isolamento
- Falta de motivação

Vantagens:
- Baixo custo
- Eliminação da barreira da timidez
- Facilidade de uso
- Flexibilidade de horários e local
- Permite mais tempo para reflexão, pesquisa e composição da mensagem

6) O stub tem a função de isolar o programador dos detalhes referentes à comunicação através da rede. Os stubs apenas cuidam dos detalhes relativos à troca de mensagens. Na parte do cliente, a função do stub é fazer o marshaling, na parte do servidor, a funçao do stub é fazer o unmarshaling

7) Não tem como pois a comunicação que o RPC faz pode ser entre dois sistemas diferentes, ou seja, os dois podem ter linguagens diferentes entao durante a passagem de parâmetro pode provocar problemas.

8) O modelo de Chamada Remota de Procedimento é similar ao modelo de chamadas locais de procedimentos, no qual a rotina que invoca o procedimento coloca os argumentos em uma área de memória bem conhecida e transfere o controle para o procedimento em execução, que lê os argumentos e os processa. Em algum momento, a rotina retoma o controle, extraindo o resultado da execução de uma área bem conhecida da memória. Após isso, a rotina prossegue com a execução normal.

Uma chamada remota de procedimento difere das chamadas locais em alguns pontos:

- Tratamento de erros: falhas do servidor ou da rede devem ser tratadas.
- Variáveis globais e efeitos colaterais: Uma vez que o servidor não possui acesso ao espaço de endereços do cliente, argumentos protegidos não podem ser passados como variáveis globais ou retornados.
- Desempenho: chamadas remotas geralmente operam a velocidades inferiores em uma ou mais ordens de grandeza em relação às chamadas locais.
- Autenticação: uma vez que chamadas remotas de procedimento podem ser transportadas em redes sem segurança, autenticação pode ser necessário.

9) Ao invocar um método com uma referência de objeto como parâmetro, essa referência é copiada e passada como parâmetro de valor apenas quando se refere a um objeto remoto. Nesse caso, o objeto é passado literalmente por referência. Porém, quando a referência se refere a um objeto local, que é um objeto no mesmo espaço de endereço que o cliente, o objeto referido é copiado como um todo e passado junto com a invocação. Em outras palavras, o objeto é passado por valor

10) O papel da fila de mensagens para promover o desacoplamento ente os processos comunicantes é o seguinte quando um comunicante manda uma mensagem, ele fica armazenado em uma fila. A funçao do middleware é pegar cada uma dessas mensagens dessa fila e mandar para outros comunicantes. Se vocẽ reparar o comunicante que enviou essa mensagem em nenhum momento teve conhecimento da identidade do destinatario e o destinatario tem um tempo de vida independente porque mesmo se ele nao existir mais, o middleware pode mandar essa mensagem para outro destinatario, com isso, concluimos que houve o desacoplamento

11) Espaço de dados baseado em eventos, eles sao processos que se comunicam por meio da propagação de eventos porque basicamente eles não vao utilizar nenhuma fila ou broker, eles simplesmente publica o evento e o middleware assegura que somente os processos assinantes irão receber esses eventos. Podemos usar também o ZeroMQ que escolhe os assinantes que vão receber a mensagem

12) A camada de middleware é muito importante por causa da comunicação por meio de mensagens, eles garantem transparencia, troca de mensagens entre sistemas heterogêneos.

13) A função das quatro operações básicas de sistemas de filas de mensagens são:
put - Anexa uma mensagem a uma fila especificada, ou seja, coloca uma mensagem
get - Bloqueia até que a fila especificada tenha alguma mensagem e remove a primeira mensagem, ou seja, pega a mensagem
poll - Checa a fila especificada por mensagens e remove a primeira mensagem. Nunca bloqueia, ou seja, verifica se tem mensagem na fila
notify - Instala um tratador a ser chamado quando uma mensagem for colocada na fila especificada, ou seja, o consumidor é notificado que tem mensagem na fila

14) Dentro do broker podemos ter vários protocolos diferentes implementados dentro dele, como a ideia do publish-subscribe em que podemos escolher so alguns assinantes que vão consumir a mensagem

15) O papel do broker para resolver a heterogeneidade das aplicações é:
- Transformar as mensagens recebidas para o formato alvo
- Frequentemente atua como gateway de aplicação
- Pode prover capacidade de roteamento baseado em assunto
- Mensageiro responsável pela transmissão das requisições, respostas e exceções
- Deve prover algum mecanismo de localização (identificador)
- Oferece APIs aos clientes e servidores (registro, invocação de métodos etc)

- Pode prover serviços adicionais (serviço de nomes, suporte a serialização/marshaling etc)