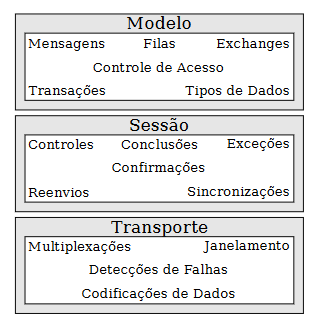
Edivency message protocol – AMQP é um protocolo aberto, assíncrono e que garante interoperabilidade entre diferentes plataformas com objetivo de realizar a transferência binaria de mensagens de forma segura, confiável, eficiente.

O AMPQ Working Group divide o protocolo em três camadas: 

*Figura 1: Camadas do Padrão AMQP*

**1-A camada de transporte visa oferecer** serviços a camada de sessão como: multiplexação de canais, detecção de falhas, representação de dados e janelamento.(framing).

* *A lista dos requisitos dessa camada inclui:*
* *Possuir uma representação de dados binária e compacta que seja rápida de mutiplexar e demultiplexar;*
* *Trabalhar com mensagens sem um limite significante de tamanho;*
* *Permitir que sessões não sejam perdidas no caso de falhas de rede ou de aplicação;*
* *Ser assíncrona e neutra em relação à linguagens de programação.*

**2-A camada de sessão** Age como uma intermediária entre as camadas de modelo e transporte. É Responsável por dar confiabilidade às interações entre as aplicações do cliente e o broker. As Sessões são interações estabelecidas entre um cliente e um servidor AMQP. Todos os comandos, como envios de mensagens e criações de filas devem acontecer no contexto de uma sessão. O ciclo de vida de alguns dos objetos da camada de modelo como filas, exchanges e bindings podem ser limitados ao escopo de uma sessão.

*Os principais serviços que a camada de sessão oferece a camada de modelo são:*

*Identificação sequencial dos comandos: onde cada comando dado pelos pares é identificado como individual e único, garantindo assim sua execução uma vez e permitindo correlacionar o comando solicitado ao resultado.*

*Confirmação que comandos serão executados: é utilizado para garantir ao cliente(solicitante) possa descartar de forma segura o estado de um comando com a certeza que o comando dado será executado. Neste caso garante a continuidade do serviço caso haja alguma falha temporária dos pares.*

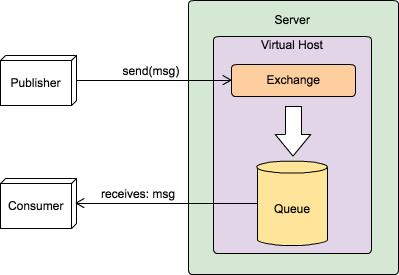
*Notificação de comandos completados: Notifica o solicitante que o comando foi executado corretamente. Objetivando assim a garantia da ordem de execução entre diferentes sessões.*

*Reenvio e recuperação no caso de falhas na rede: caso haja falha, a sessão devera reenviar o comando com rotulo. Neste caso os rótulos previnem a duplicidade dos comandos.*

**3-A camada de modelo:** Define o conjunto de comandos e objetos que as aplicações podem utilizar. O modelo AMQP que estabelece como as mensagens serão enviadas, roteadas, armazenadas, enfileiradas, recebidas.

* *Estre os objetivos desta camadas estão:*
* *Garantir a interoperabilidade das implementações,*
* *Prover o controle sobre a qualidade do serviço,*
* *Oferecer um mapeamento natural entre os comandos do protocolo e as bibliotecas do nível de aplicação.*

O modelo AMQP depende diretamente de seus 7 componentes: vamos falar deles:



*Figura 1- Diagrama da arquitetura de componentes do AMQP até versão 0.9.*

**1-Produtor**: é a entidade que envia mensagens para o broker/servidor.

**2-Consumidor**: responsável por consumir as mensagens das filas.

**3 -Servidor** (Server): denominado de Broker, um servidor AMQP aceita conexões de clientes, recebendo a mensagens, e as roteamento para diferentes consumidores através de filas.

4- Virtual Hosts: são domínios independentes no servidor que compartilham um ambiente comum para autenticação e segurança. É similar a um ambiente de trabalho a qual o usuário se conecta após se autenticar no servidor. Este ambiente Inclui um conjunto de Exchanges, filas de mensagens e vínculos (bindings).

5- bindings (baydengs)/vinculações: são responsáveis por relacionar o Exchange as filas de mensagens definindo assim quais as regras a serem utilizadas para o roteamento.

6-**Exchange**: componente que recebe as mensagens e faz o roteamento para as filas (message queues). Então ao Receber as mensagens criadas pelo produtor o exchange inspeciona a mensagem verificando qual a chave de roteamento que fica no cabeçário da mensagem e com auxílio do bindings encaminha as mensagens para a fila correta.

7-Filas de mensagens: é componente responsável por manter as mensagens e encaminhar para os consumidores via política FIFO. As filas mantém as mensagens em memórias ou em disco até que sejam enviadas em sequência para as aplicações consumidores.

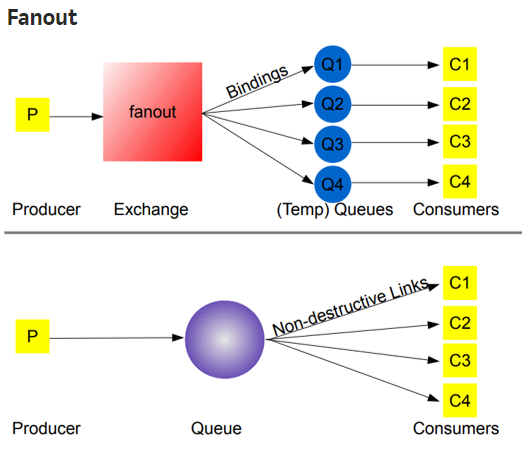
Até a versão 0.9 o protocolo AMQP é ainda é muito utilizado por brokers como o rabbitMQ, porém o AMQP na versão 1.0 passou a ter mais focos nas definições para as camadas de mensageria, conexão e de segurança com utilização de protocolo orientado a conexão denominado camada de segurança e autenticação simples(sasl) o q permitiu separar os mecanismos de autenticação dos protocolos de aplicativo *(acesso a convidado não autenticado, mecanismo de senha, autentificarão direta via ipsec ou tls).*Na versão 1 t**odas as tentativas de definir um modelo de broker foram removidas e o** protocolo deixou de ser vinculado a topologia de aplicação. Na prática, isso significa que foram retirados os Bindings e Exchanges, mas apesar disso continua a realiza a troca de mensagens. O AMQP passou a ser visto como um conjunto de Containers, Nós e Links, onde o Container: é formado por um conjunto de nos. Como por exemplo aplicações cliente e o Broker, o No: é representado por qualquer entidade endereçável (como produtor, consumidor e fila), e os Links: realizam a conexão entre os Nós e são unidirecionais para transferência da mensagem.

Para que a comunicação ocorra entre nós em diferentes contêineres, uma conexão precisa ser estabelecida. Uma única conexão PODE ter várias sessões independentes ativas de forma simultânea. Ambas as conexões e sessões são modeladas por cada par (como endpoints) que armazenam o local e último estado remoto em relação à conexão ou sessão em questão. Importante para caso há uma falha e precise voltar ao estado de onde estava.

Para transferir mensagens entre nós (por exemplo, para mover mensagens de uma fila para um consumidor), um link precisa ser estabelecido entre os nós. Os links podem ser criados por um contêiner a qualquer momento e em uma sessão existente, o que torna AMQP diferente de muitos outros protocolos. **O contêiner de link da inicialização é o responsável por solicitar que o contêiner oposto aceite um link** e escolha uma função de remetente ou de destinatário. Portanto, o contêiner pode iniciar a criação de caminhos de comunicação unidirecional ou bidirecional, com o último modelado como pares de links. As mensagens só trafegarão por um link se atenderem aos critérios de entrada. Então, estabelecida uma conexão o link passa a existir naquela sessão e o link poderá ou não possuir um filtro para verificar se a mensagem pode trafegar através dele, caso a mensagem possa trafegar a sessão então correlaciona dois canais unidirecionais para formar uma conversa bidirecional e sequencial. Ao final da sessão o link é desfeito.

Para finalizar vamos ver como seria alguns tipos de transmissão de mensagens comparando a versão 1.0 do AMQP com as anteriores.

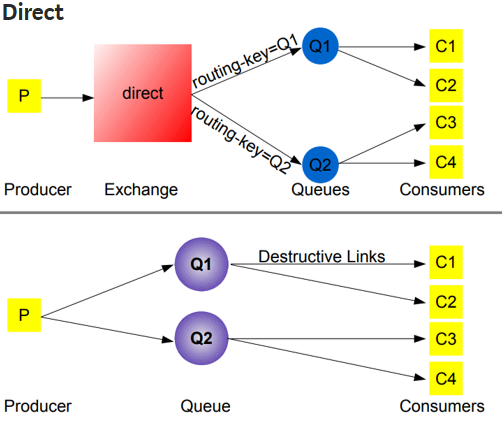
**No Fanout:** *A**Distribuição das mensagens é feita para todas as filas, independente da sua chave de roteamento. Os casos de uso para esta troca geralmente envolvem a distribuição de uma mensagem a vários clientes para fins semelhantes como:* **Utilizado para o compartilhamento de mensagens (por exemplo, servidores de bate-papo), atualizações (por exemplo, notícias) e estados do aplicativo (por exemplo, configurações).**



No caso da imagem acima para versões anteriores a 1.0 para uma mensagem fanout existe o Exchange que ao receber a mensagem do produtor tem a função de redirecionar todas as mensagens para todas as filas e cada fila redireciona para o seu respectivo cliente.

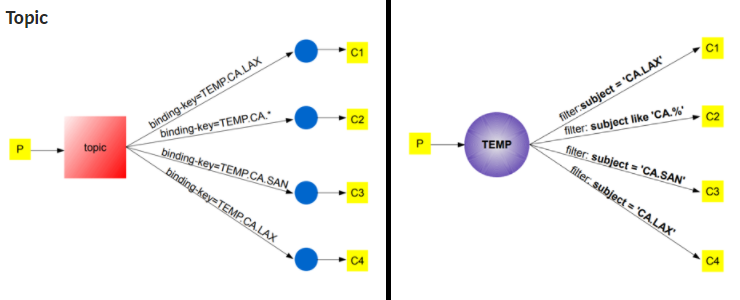
Já na figura de baixo a versão 1.0 não existe Exchange, então fica a cargo do contêiner (formado pelo produto e fila) a reponsabilidade criarem todos os links da fila para todos os consumidores para assim enviar as mensagens.

**Na Troca direta:** *a entrega de mensagens a filas é feita com base em chaves de roteamento. As chaves de roteamento podem ser consideradas como dados adicionais definidos para definir para onde uma mensagem irá. Assim a mensagem só será entregue quando a chave de roteamento houver uma correspondência exata com a****chave de vínculo****(*binding key*). O caso de uso típico para troca direta são tarefas de* ***balanceamento de carga****.*



Na figura de acima para versões anteriores a 1.0 para uma mensagem Direct o exchange analisa o cabeçário da mensagem para definir para qual fila devera direcionar a mensagem. Depois de direcionar fica a cargo da fila q1 e q2 fazerem a devida entrega ao consumidor. Já na figura de baixo a versão 1.0 como não existe Exchange, fica a cargo dos consternes formados pelo produtor e pelas filas q1 e q2 determinarem a rota, criarem os links e assim redirecionar a mensagem aos consumidores específicos.

**Troca por Topic:** A troca de tópicos direcionam as mensagens analisando a correspondência entre as chaves das mensagens e a dos vínculos das filas. Usando esse tipo de transferência, uma chave de roteamento junto com a vinculação de filas às trocas são usadas para combinar e enviar mensagens**. É usada principalmente para padrões de publicação / assinatura**, muito útil para distribuir mensagens de acordo com base em chaves e padrões. Sempre que um envolvimento especializado de um consumidor é necessário como, por exemplo, um único conjunto de trabalho para executar um certo tipo de ações.



Na figura a esquerda mostra o envio via tópicos nas versões anteriores a 1.0 , onde o exchange analisa o cabeçário da mensagem e juntamente com bindings definem para qual fila com o tópico definido devera direcionar a mensagem. Depois de direcionar fica a cargo da filas fazerem a devida entrega ao consumidores. Já na figura de baixo a versão 1.0 como não existe Exchange, fica a cargo dos consternes formados pelo produtor e pela fila TEMP criarem o link e estabelecerem a entrega com o uso de filtros presentes nos links. Assim caso a mensagem filtrada não passe pelo filtro ela não poderá ser enviada, caso contrário ela será entregue.