MOM – Message Oriented Middleware

Bruno Miguel de Sousa Gonçalves

1.O que é a MOM?

1.1. Conceito

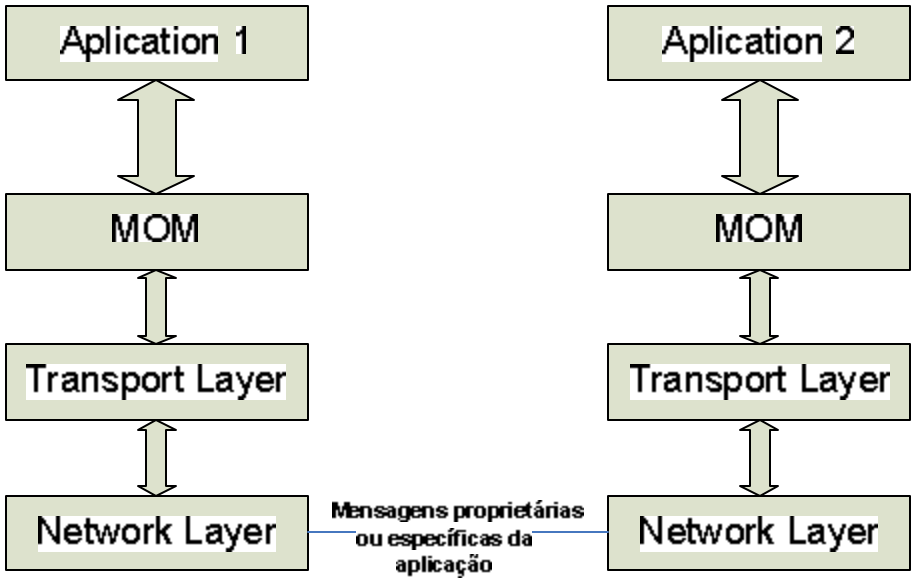

Figura 1 – Descrição geral da arquitectura MOM.

Message Oriented Middleware trata-se de uma insfraestrutura de software cliente/servidor que cria uma camada entre as aplicações de alto nível e as plataformas onde estão instaladas, substituindo a comunicação directa entre aplicações por um sistema de troca de mensagens.

1.2.O que faz?

Típicamente, uma implementação MOM fornece um conjunto de API's capaz de funcionar com um número relativamente vasto de plataformas e redes que varia entre implementações.

As API's (Aplication Programing Interface) fornecem um nível de abstração capaz de aumentar a portabilidade, interoperabilidade e flexibilidade das aplicações que correm sobre a MOM.

Usando as API's, os programadores são libertados dos detalhes das várias plataformas e protocolos, reduzindo assim a complexidade da implementação das comunicações das suas aplicações.

A comunicação é efectuada através de mensagens que são transmitidas, ou pela estrutura cliente/servidor pura (usando broadcast ou multicast), ou entre pilhas mantidas por gestores locais, sendo esta última a mais poderosa em termos de aplicabilidade e versatilidade.

2.Descrição tecnológica

2.1.Como faz?

A MOM está presente em ambos os lados de uma arquitectura cliente/servidor e serve de ligação peer-to-peer entre ambos, suportando chamadas assíncronas entre as aplicações intervenientes. É possível adicionar sincronismo à arquitectura usando, por exemplo RPC's, em conjunto com MOM.

Conseguindo-se abstrair da plataforma onde executa, uma aplicação consegue trocar mensagens com outros programas residentes em plataformas diferentes da sua.

Existem dois tipos de mensagem, as persistentes e as não persistentes. Uma mensagem importante pode ser marcada como persistente de forma a ser armazenada em memória não volátil, assim em caso de falha, os seus dados não são perdidos. As não persistentes são guardadas em memória volátil.

Os dados que podem ser transmitidos numa mensagem são:

- Dados formatados (variáveis, strings)
- Pedidos de execução (chamadas a funções)
- Ou ambos

As mensagens são transmitidas por gestores de pilhas (Message Queue Manager ou MQM). A finalidade de um gestor de pilhas é fornecer um ambiente para aplicações que usem as pilhas de mensagem ou para aplicações como a MQI (Message Queue Interface). As suas funções são providenciar armazenamento fiável para as mensagens armazenadas nas pilhas, gerir acessos concorrentes a dados, assegurar segurança e autenticidade, e providenciar funções especiais de emparelhamento de mensagens (por ex. triggers).

A forma de tornar as trocas de mensagens mais fiável é usar tratamento de pilhas transacionais. Basicamente as mensagens são trocadas em transações, ou seja, o MQM envia várias mensagens para uma aplicação, se ela não as consegue receber em parte ou no todo, volta a guardar as mensagens na pilha para entregar mais tarde.

As aplicações podem comunicar com o MQM através da MQI ou directamente. É possível a coexistência de mais que um gestor no mesmo sistema ou máquina. Nestas situações a aplicação deve especificar a qual se quer ligar, se não especificar nenhum nome, então estabelece a ligação com o gestor por omissão.

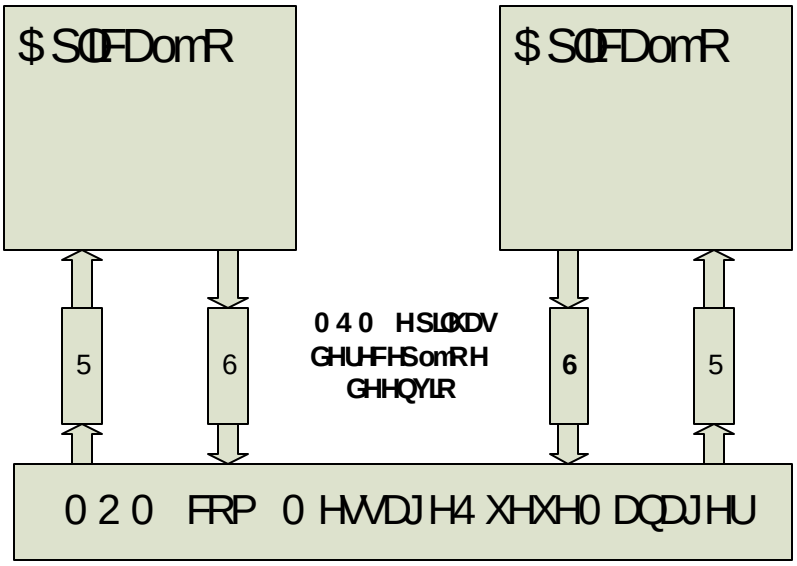

Figura 2 – Descrição detalhada da arquitectura MOM.

Nem sempre a aplicação que corre sobre MOM está disponível para receber mensagens, ou por se encontrar a processar mensagens anteriores ou por estar offline. Nestas situações, as mensagens são guardadas numa fila onde ficam à espera que a aplicação se torne disponível.

Quando ocorre um evento, a aplicação cliente delega à MOM a função de notificar a aplicação servidora acerca desse evento. Por esse motivo, as implementações MOM são mais apropriadas para aplicações orientadas a eventos. Também é adequado para sistemas orientados a objectos pois oferece um mecanismo conceptual para comunicação peer-to-peer entre objectos.

3.Prós, contras e alternativas

3.1.As vantagens

Comunicação assíncrona. A MOM usa uma aproximação “send and forget”. Deste modo o emissor não tem que esperar que o receptor receba a mensagem, basta-lhe esperar que a mensagem seja transmitida e guardada na pilha de recepção do receptor. Comunicação assíncrona permite independência temporal e execução não bloqueante, pois os processos não são obrigados a esperar uns pelos outros.

Mesmo sendo um sistema assíncrono, a MOM também pode funcionar em modo síncrono, com ajuda por exemplo, de RPC’s. Nem todas as aplicações foram desenhadas para funcionar em modo assíncrono.

A comunicação é mais fiável que em RPC pois as mensagens são transmitidas por “store-and-forward”, ou seja, mesmo quando um envio falha, é possível ao sistema mensageiro retransmitir a mensagem

O nível de abstracção oferecido pela MOM permite que ocorram alterações nos protocolos de rede, ou mesmo nas arquitecturas onde correm as aplicações, sem estas terem que ser reprogramadas. A sua complexidade também é reduzida, pois deixa de existir necessidade de criar código específico para a rede ou para a plataforma.

As filas de armazenamento de mensagens permitem que os dados sejam enviados assim que estão prontos para tal, ficando à espera que a aplicação as receba. Também é possível assim introduzir novos servidores ou clientes de forma transparente, pois não existe uma ligação directa entre as aplicações.

3.2.As desvantagens

Um dos grandes problemas desta arquitectura é o facto de não existir um standard definido que regule as várias implementações. Assim, as implementações existentes são proprietárias, o que por efeito torna diferentes implementações incompatíveis. Sendo obrigado a usar apenas um produto, o cliente fica dependente do fabricante da implementação escolhida.

O núcleo MOM tem que correr em todas as plataformas da rede, ficando a escolha de uma implementação dependente do seu suporte aos sistemas e protocolos usados por essas plataformas.

Quanto maior a heterogeneidade do sistema, mais custos são associados ao mesmo por ter de comprar vários módulos, maior manutenção também lhe está associada.

3.3.As alternativas.

Como alternativa à MOM, existem outras arquitecturas que permitem a distribuição de processamento por plataformas heterogéneas. As mais importantes são:

- Object Request Broker (ORB)
- Distributed Computing Environment (DCE)
- Remote Procedure Call (RPC)
- Transaction Processing Monitor Technology
- Three Tier Software Arquitectures

4.Como é que a MOM revolve certos problemas?

4.1 Como é que um emissor consegue enviar para todos os receptores interessados?

Todas as aplicações ao arrancar são registadas nos MQM's e publicam as suas mensagens para as queues. A partir daí o MQM é o responsável por encaminhar as mensagens para as pilhas de recepção dos destinatários correctos.

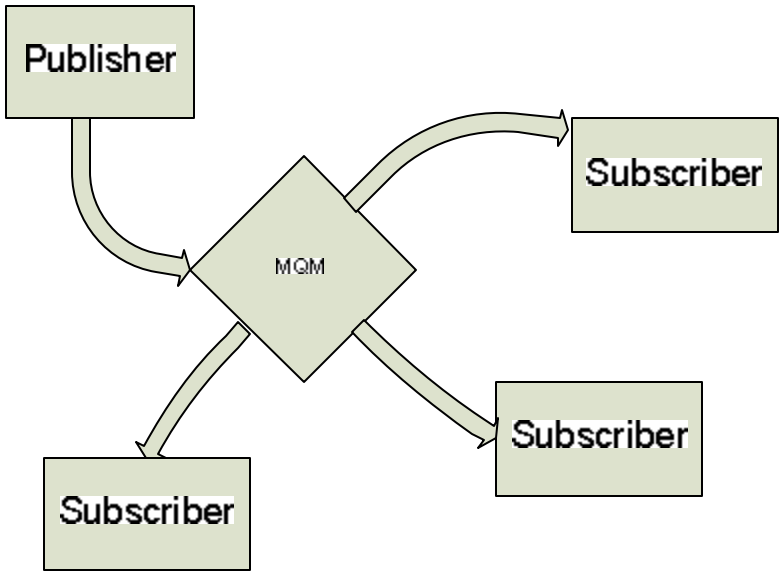

Figura3 - Canal publish-subscribe.

4.2 Como é que sistemas com formatos de dados diferentes comunicam usando messaging?

Sistemas heterogéneos comunicam através de dados diferentes, por exemplo, pacotes de dados com cabeçalhos diferentes. Se no MQM existir um módulo tradutor, então as mensagens dos emissores podem ser reformatados para o formato dos receptores.

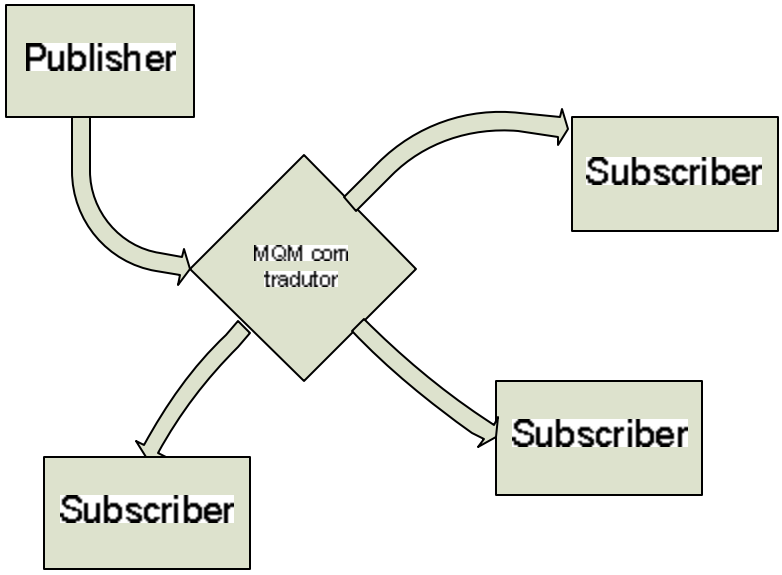

Figura 4 - Tradutor

4.3 Como encapsular o acesso ao sistema de messaging do resto da aplicação?

Se permitirmos às aplicações acesso directo aos canais de comunicação, nomeadamente as pilhas, perde-se grande parte da abstração criada pela MOM e aumenta-se desnecessáriamente a complexidade da implementação de aplicações.

Ao introduzir um gateway de mensagens na aplicação, toda a comunicação passa a ser transparente. Este gateway pode ser uma classe fornecida pela API que recebe as mensagens directamente da aplicação e as passa para o MQM ou recebe as mensagens que a MQM envia passando-as para a aplicação.

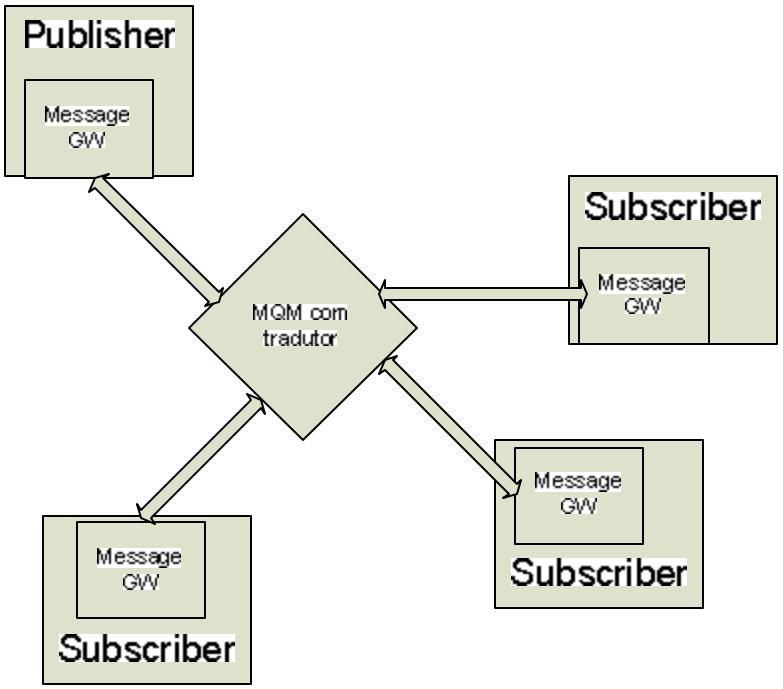

Figura 5 - Message gateway

4.4 Como fazer quando a implementação de uma função lógica está espalhada por vários sistemas fisicos?

Um pedido por parte de uma aplicação cliente pode envolver o processamento de mais que um servidor, ou pode necessitar de fazer pedidos de contextos diferentes. Implementando um encaminhador baseado em contexto é possível que o MQM encaminhe as mensagens para as pilhas correctas, não tendo em conta o destinatário mas sim o contexto da mensagem e quem a pode tratar.

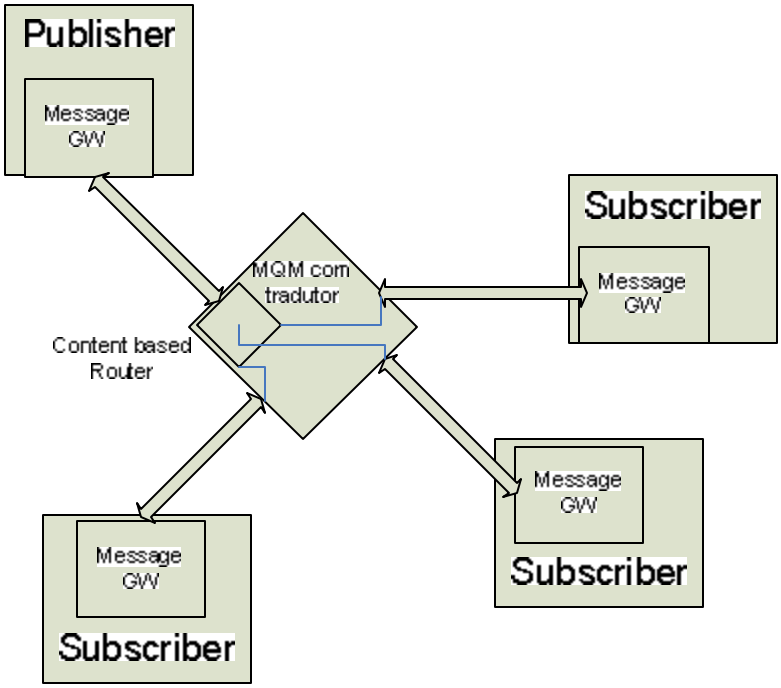

Figura 6 - Content based routing

4.5 Como efectuar o processamento complexo sobre uma mensagem?

A forma natural de tratar problemas complexos é repartir esse mesmo problema por passos mais simples. Tendo em conta uma mensagem complexa, é possível passá-la por filtros mais simples ligados em sequência por pipes, cada um tratando uma parte da mensagem, de forma a esta ser tratada numa sequência de passos.

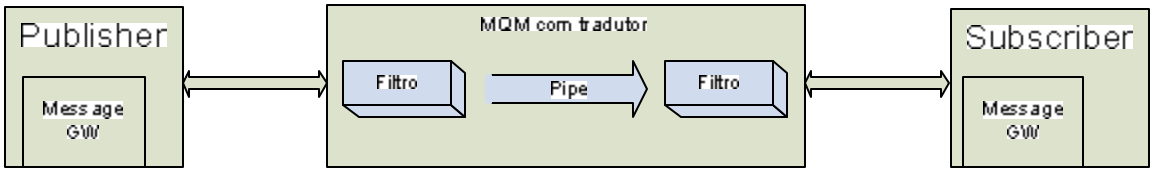

Figura 7 - Pipes e filtros

4.6 Como fazer para passar uma mensagem por vários passos de processamento quando estes são desconhecidos em tempo de design ou não são sequenciais?

Para muitas soluções o esquema pipes-filtros resulta, mas nos casos em que não sabemos a sequência correcta de passos, o problema volta a aparecer.

Nestas situações podemos introduzir um gestor de processos na arquitectura com conhecimento sobre para que processo encaminhar as mensagens e capaz de interpretar resultados intermédios por forma a determinar qual o próximo processo para onde enviar a mensagem.

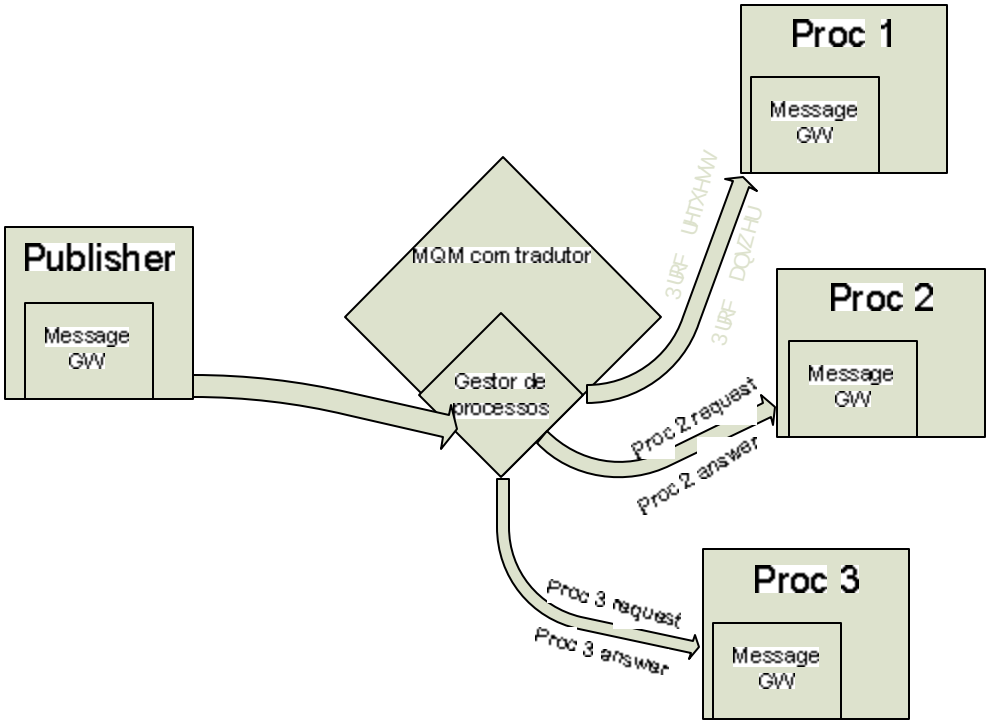

Figura 8 – Gestor de processos

4.7 O que fazer com uma mensagem que não pode ou não deve ser entregue?

Uma mensagem pode ser enviada com um determinado fim, mas por várias razões pode perder a utilidade, por exemplo no caso de expirar um prazo de entrega.

Nestas situações, o MQM pode redireccionar a mensagem em questão para uma pilha de mensagens rejeitadas.

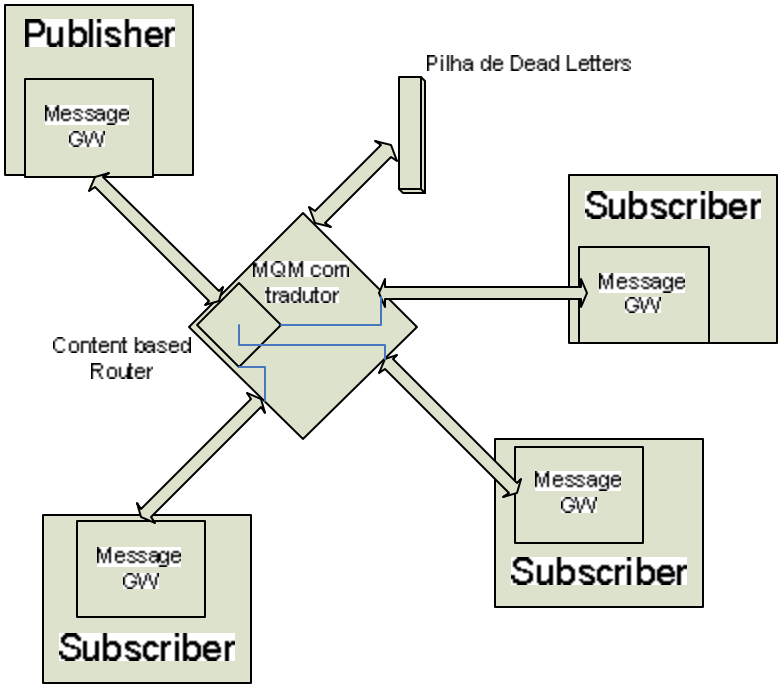

Figura 9 – Pilha de Dead Letters

4.8 Como é que um consumidor pode seleccionar da pilha apenas as mensagens que deseja receber?

O MQM quando recebe uma mensagem para um determinado destinatário, coloca essa mensagem na pilha de recepção correspondente. Mas e se o destinatário não quiser receber todas as mensagens que lhe são destinadas? Acoplando um módulo de selecção à aplicação, esta poderá escolher apenas as mensagens que lhe interessem.

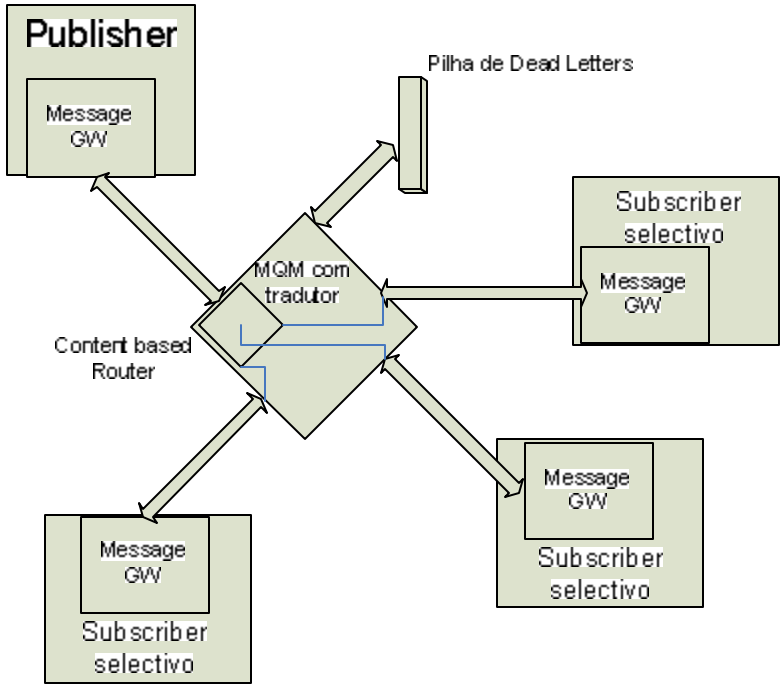

Figura 10 – Subscriber selectivo ou consumidor selectivo

4.9 Como é que analizamos e depuramos o percurso de uma mensagem num sistema de messaging?

Nem sempre o percurso de uma mensagem é linear, nem sempre mensagens do mesmo tipo são processadas da mesma forma. Então como é que sabemos se uma mensagem foi bem processada?

Ao adicionarmos uma histórico à mensagem ficamos a saber porqueprocessos ou aplicações é que ela passou e quais os resultados do seu processamento.

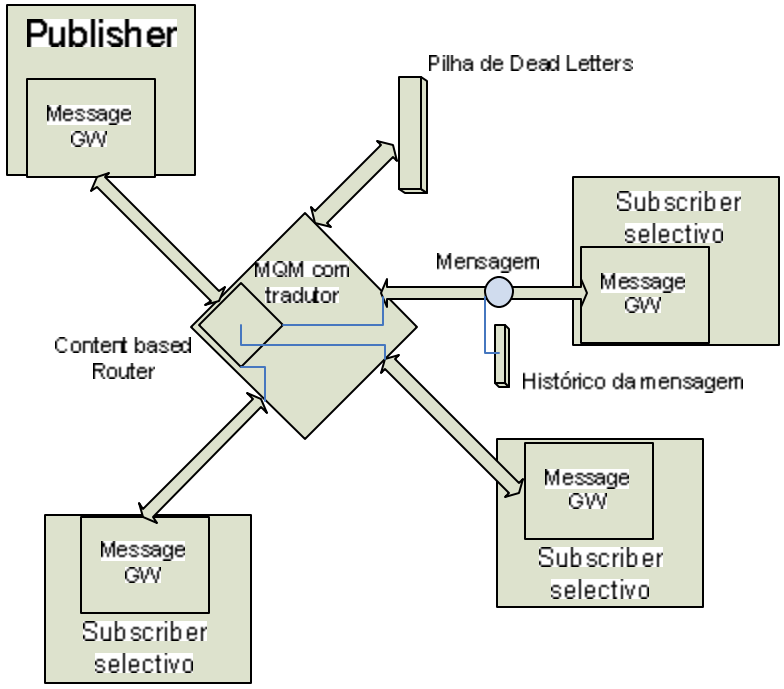

Figura 11 – Histórico de mensagem