Foco: Solução publish/subscribe para cenários onde o desempenho e a latência sejam importantes.

Dependendo das necessidades dos diferentes cenários os diferentes tipos de publish/subscribe podem ser mais adequados que outros, sendo que o mesmo se aplica aos meios de comunicação.

Possíveis implementações de sistemas publish/subscribe:

- Message-Oriented Middleware: Fornece subscrições persistentes, garantia de entrega de eventos e expressiva semântica de entrega. Não é a melhor solução para todos os cenários, pois tem tendência para ser uma solução com custos de desempenho.

WS-Notification: Definição de um standard de uma abordagem de Web services para notificações utilizando o padrão publish/subscribe baseado em tópicos. Este tipo de sistema publish/subscribe é desenhado para interligar com outros standards WS-\*. Requere uma grande utilização de recursos e é análogo a dois recursos comunicarem através de MSMQ. Não é uma boa solução para cenários onde se procura elevado desempenho e baixa latência.

## Soluções publish/subscribe à medida

Existem diversas implementações de sistemas publish/subscribe desenhadas à medida do contexto onde se aplicam, estas soluções perdem na generalização de aplicação mas podem ganhar no desempenho e na latência.

Um exemplo de um sistema publish/subscribe à medida é o Jabber. Este sistema tem a noção de um serviço publish/subscribe que age como um servidor de eventos, que trata as subscrições e entregas de eventos. Os eventos são XML. O desempenho deste sistema é melhor que o atingido no WS-Notification, não sendo contudo atingido um alto desempenho nem uma baixa latência.

## Caracterização do domínio do problema

Cada vez mais existe a necessidade de as aplicações comunicarem de forma assíncrona. À medida que o ambiente de funcionamento se torna mais virtualizado, as aplicações estão a ficar virtualizadas. Estas características colocam maiores exigências sobre as infra-estruturas de comunicação. Os standards WS-\* existentes resolvem alguns cenários de aplicação, mas não resolvem os cenários onde o alto desempenho e a baixa latência são essenciais.

No contexto de aplicações WEB empresariais existem três tipos de cenários que exigem alto desempenho e baixa latência, sendo eles: virtualização de sistema, virtualização de aplicação e recolha de métricas.

### Virtualização de sistema

Uma WEB farm e´um exemplo de um sistema virtualizado. Um dos problemas existentes no Microsoft ISS é o facto de não virtualizar os serviços de sistema pela WEB farm. Um problema que advém desta particularidade é o caso do ASP.Net cache que é contextualizada a um AppDomain, sendo que desta forma a cache não pode ser utilizada por diferentes aplicações, e muito menos por diversas máquinas. Para muitas aplicações esta característica representa a impossibilidade de se utilizar cache. Uma vez que a cache não pode ser partilhada pelas diferentes máquinas, também não pode ser utilizada caso não se estabeleça afinidade na utilização duma WEB farm. Por razões de desempenho não é desejável a utilização de afinidade.

Uma cache distribuída é um dos exemplos onde um sistema de eventos publish/subscribe distribuído pode consistir numa boa solução para melhorar a virtualização de sistema. Ter-se alto desempenho e baixa latência traduz-se na possibilidade prática de se ter uma cache, comum, partilhada através dos limites da aplicação e inclusive uma cache distribuída através dos limites da máquina. Cada cache receberia as actualizações das outras caches com um tempo de espera de milissegundos, permitindo que as caches se mantivessem sincronizadas.

### Virtualização de aplicação

As aplicações estão-se a deslocar para um paradigma cada vez mais desacoplado. Desta forma consegue-se uma melhor escalabilidade e simplifica-se a inovação ao reduzir-se a complexidade de forte acoplamento e dependências. Quanto mais as aplicações se tornam virtualizadas, mais significante se torna a necessidade de comunicação. Tal como no mundo da interligação, não existe um único tipo de comunicação que resolve todos os cenários. Nalguns casos, um esquema de mensagens directo, como o WS-RM, pode ser a melhor solução. Um sistema de publish/subscribe construído sobre WS-RM pode ser apropriado, caso seja necessária a durabilidade e não seja um requisito o alto desempenho e a baixa latência.

Em alguns casos de aplicações WEB empresariais existe a necessidade de existirem aplicações virtualizadas por todo WEB farm, tendo estas que se comportar como uma só aplicação lógica. O padrão geral figura-se nos componentes de um front-end alojados em servidores WEB e a enviarem eventos para componentes do back-end, que correlaciona e processa os eventos. Um exemplo desta utilização consiste num sítio WEB com comportamento dinâmico, sendo este comportamento controlado por um sistema de recomendação localizado no back-end. Neste caso os componentes WEB do front-end enviam eventos para o motor de recomendações do bak-end, sendo que este último fornece recomendações em resposta. O exemplo ilustrado exige alto desempenho e baixa latência.

### Recolha de medidas

A recolha de medidas consiste num cenário comum nas aplicações WEB. É essencial monitorizar a actividade de um utilizador num sítio WEB.

No sítio da WEB Microsoft.com, existem cerca de 2000 eventos por segundo que têm de ser recolhidos. Para recolher, de forma eficiente, tantos eventos a este ritmo, é essencial existir um sistema de eventos com alto desempenho e baixa latência.

## Descrição do sistema de eventos WSP

O sistema de eventos WSP é sistema de eventos publish/subscribe distribuído. É distribuído por a publicação e subscrição de eventos poder ser intra-máquina ou inter-máquina. Os publicadores e subscritores não necessitam de saber da existência uns dos outros, sendo contudo possível que um publicador possa escutar os eventos de subscrição, para determinar se existem subscritores interessados nos seus eventos, antes da sua publicação.

### Subscrições

Os eventos são tipificados através da propriedade EventType. Esta propriedade é um GUID que permite que sejam definidos tipos eventos sem que exista a preocupação de conflitos com tipos de evento criados por outras utilizações. As aplicações subscrevem tipos de evento, sendo que existe a opção de subscrição de múltiplos tipos de eventos.

As subscrições têm associados identificadores de subscrição. Na criação de uma subscrição é criado um evento de subscrição, que é publicado pela rede, contendo o respectivo identificador de subscrição e tipo de evento.

### Topologia



Ilustração 1 - Imagem retirada de WSP Event System

As máquinas na rede de eventos estão organizadas em hierarquia. Cada máquina tem conhecimento do seu pai e cada pai tem conhecimento dos seus filhos, à medida que estes se conectam. As comunicações entre máquinas são feitas através de TCP.

Caso os eventos sejam enviados entre irmãos, o encaminhamento é feito através da máquina pai, até ao nó destino. Independentemente do número de subscritores atingíveis através do nó pai, um evento será enviado apenas uma vez da máquina filho para a máquina pai, ou de um pai para um filho. O pai encaminha o evento para o seu pai e/ou seus filhos, caso seja apropriado.

### Encaminhamento de eventos

As tabelas de encaminhado são construídas e mantidas dinamicamente, de forma semelhante a como os routers de rede mantêm as suas tabelas para multicast IP. As subscrições são tratadas ao serem enviados eventos através da rede. Cada nó utiliza os eventos de subscrição para manter as suas tabelas de encaminhamento. Quando são publicados eventos numa máquina ou encaminhados para uma máquina, o sistema de eventos encaminha-os através da rede de ligações físicas onde existem subscrições.

### Estrutura do sistema



Ilustração 2 - Imagem retirada de WSP Event System

O Receiver e Fowarder são as interfaces de comunicação para interagir com as máquinas pai e filho. Quando um evento chega ao Receiver, este envia o evento para o RePublisher, que publica o evento no sistema local. Publicar o evento, neste sentido, significa colocar o evento no buffer de memória partilhada. Os subscritores interessados copiam o evento da memória partilhada, tal como o Listener. Este último reencaminha o evento para outros componentes, caso seja apropriado fazê-lo. Caso um evento seja de subscrição, o Listener envia-o para Subscriptions, para que sejam actualizadas as tabelas de encaminhamento. Se o tipo de evento for para ser persistido, o Listener encaminha o evento para o Persister. Quando as tabelas de encaminhado indicam que o evento precisa de ser encaminhado, o Listener encaminha o evento para o Forwarder, que por sua vez envia o evento para outra máquina.

As aplicações de publicação limitam-se a colocar eventos no buffer de memória partilhada. Quando um publicador e subscritor estão na mesma máquina, o evento é colocado na memória partilhada, pelo publicador, e recolhido desta, pelo subscritor. Nesta situação nunca existem custos acrescidos do evento passar pelo servidor de eventos. Uma vez que podem existir N subscritores à escuta de eventos colocados na memória partilhada, é possível que todos os N subscritores recolham o evento ao mesmo tempo e de forma eficiente.