Administração de Sistemas (ASIST)

Aula Teórico Prática 11 - 2018/2019

Clustering: failover e load balancing.
Linux – HAProxy
Windows Server - NLB (*Network Load Balancing*)

Fault tolerance

Fault tolerance (tolerância a falhas) é a propriedade de um sistema que lhe permite continuar a funcionar normalmente mesmo que ocorra uma falha num ou vários dos seus componentes.

Tipicamente a tolerância a falhas implica redundância, ou seja ter vários componentes idênticos que realizam a mesma função. Implica também a existência de mecanismos que detetam os componentes em falha e garantem que o sistema deixa de os utilizar, sem que isso tenha qualquer impacto no desempenho.

Poucos sistemas são totalmente tolerantes a falhas porque para o serem teriam de eliminar todo e qualquer ponto único de falha (SPoF – *Single Point of Failure*).

Por exemplo, normalmente consideramos um sistema de discos RAID1 tolerante a falhas, mas isso só é verdade sob o ponto de vista do componente disco, não sob o ponto de vista por exemplo do controlador do disco.

Failover

Uma abordagem para proporcionar tolerância a falhas de um componente de um sistema é o ***failover***.

Na abordagem *failover* existem vários componentes semelhantes (redundância), no entanto apenas um deles está ativo (a ser usado pelo sistema), os restantes estão em *standby*, mas prontos para serem ativados.

Se o componente em funcionamento falha, um dos que se encontra em *standby* é automaticamente ativado por um sistema de controlo para o substituir.

O processo de substituição do componente em falha deve ser automático e rápido para evitar qualquer perturbação no funcionamento global do sistema. O sistema de controlo que efetua a monitorização do estado do componente ativo e desencadeia a sua substituição constitui normalmente um indesejável ponto único de falha.

Se o componente possui estado (stateful) é também necessário implementar mecanismos de sincronização que garantam que em qualquer altura os componentes em *standby* tem o mesmo estado do componente que entra em falha (exemplo: RAID1).

Load Balancing

Ter componentes em *standby* é sempre um desperdício de recursos. Se for possível o sistema usar em simultâneo os vários componentes redundantes para melhorar a sua performance e simultaneamente garantir a tolerância a falhas essa é a opção ideal. É normalmente conhecida por balanceamento de carga (***load balancing***).

Com *load balancing* todos os componentes redundantes estão em funcionamento permanente, usando um mecanismo de controlo apropriado. O sistema pode por exemplo usar alternadamente cada um dos componentes, se um componente entra em falha deixa de ser usado (*fault tolerance*).

A forma como o mecanismo de controlo define qual dos componentes vai ser usado em cada instante pode ser mais ou menos sofisticada, desde um simples ***round-robin*** até uma monitorização da performance de cada componente e utilização do componente que no momento oferece melhores garantias.

Note-se que a monitorização do estado dos componentes é geralmente obrigatória para detetar falhas. Novamente, o sistema de controlo constitui um ponto único de falha.

Redes - *Fault Tolerance / Load Balancing*

Temos já conhecimentos sobre mecanismos que garantem a tolerância a falhas e o balanceamento de carga nas redes.

Nível 2 (ex.: Ethernet) – caso existam caminhos ou ligações redundantes entre nós intermédios de nível 2 (*switches*), o STP (*Spanning Tree Protocol*) implementa *failover*, o LACP (*Link Aggregation Control Protocol*) ou outros equivalentes de *Link Aggregation* implementam *load balancing* com *fault tolerance*.

A técnica de *Link Aggregation* também pode ser usada para ligação a nós finais como servidores. No caso dos servidores Windows através de um ***NIC Team***, nos servidores Linux através de um ***bonded NIC***.

Nível 3 (ex.: IPv4/IPv6) – caso existam caminhos redundantes entre nós intermédios de nível 3 (*routers*), o encaminhamento dinâmico apoiado em protocolos de encaminhamento permite implementar *load balancing* com *fault tolerance*. Todos os protocolos de encaminhamento garantem *fault tolerance*, a eficiência do *load balancing* depende essencialmente da métrica utilizada.

Servidores e serviços de rede

Na implementação de serviços de rede redundantes, a estratégia que deve ser adotada é a utilização de servidores redundantes. Isto deriva da necessidade de dentro do possível evitar pontos únicos de falha.

Se implementarmos serviços de rede redundantes usando o mesmo servidor, em caso de falha do servidor, ficariam em falha todas as alternativas de prestação do serviço. Seria também necessário lidar com a impossibilidade de ter no mesmo servidor vários serviços redundantes a usar o mesmo número de porto.

No contexto da virtualização de servidores, devemos ainda optar por, não só colocar os serviços redundantes em diferentes servidores, mas também de preferência em servidores virtuais residentes em plataformas de virtualização diferentes e independentes.

Novamente a motivação é evitar pontos únicos de falha que afetem todos os serviços alternativos, neste caso se a plataforma de virtualização falhar, falham todos os servidores que lá se encontram e consequentemente todos os serviços alternativos.

Clusters de servidores

Resumindo, na implementação de serviços de rede redundantes devemos evitar que haja a possibilidade de serem todos afetados por uma mesma falha (ponto único de falha). Isso consegue-se implementando-os em máquinas distintas e evitando que haja pontos únicos de falha que possam afetar todas as máquinas.

Um *cluster* de servidores é um conjunto de servidores que trabalham em conjunto com um objetivo comum. Objetivos típicos de um *cluster* de servidores são *load balancing* e *fault tolerance* ou seja disponibilizar serviços de alta disponibilidade (*High-availability cluster* – HA-*cluster*).

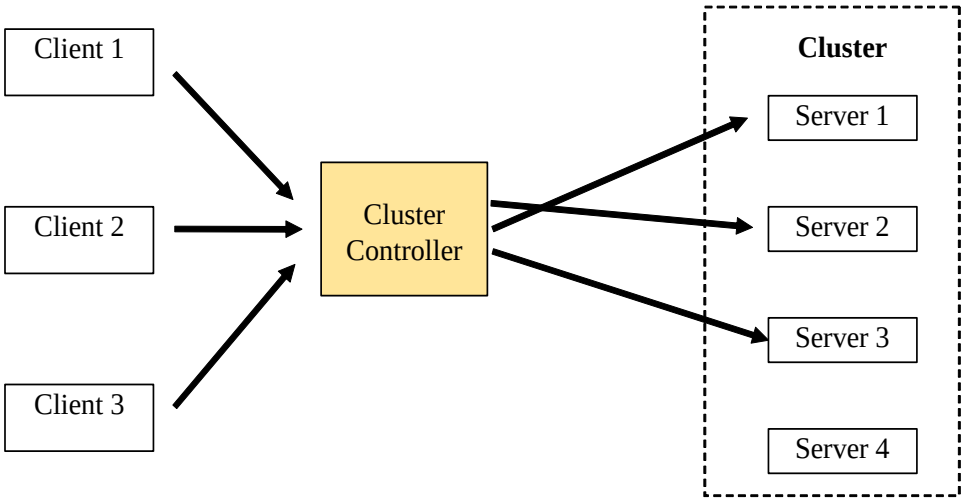
Uma outra caraterística que um *cluster* de servidores deve ter é ser usado como um sistema único, ou seja as solicitações são dirigidas ao *cluster* e não a servidores individuais do *cluster*.

Muitas vezes esta caraterística é implementada através de um controlador do *cluster* que se constitui num ponto único de acesso ao mesmo, consequentemente também constitui um ponto único de falha.

IP único de acesso ao *clusters* de servidores

Embora constitua um ponto único de falha, uma forma de implementar um ponto único de acesso a um *cluster* de servidores é através de um controlador com um endereço IP único.

Os clientes acedem ao *cluster* enviando os pedidos para o endereço IP único do controlador, o controlador redireciona os pedidos para os vários servidores do *cluster* segundo critérios de *load balancing* e *fault tolerance*.

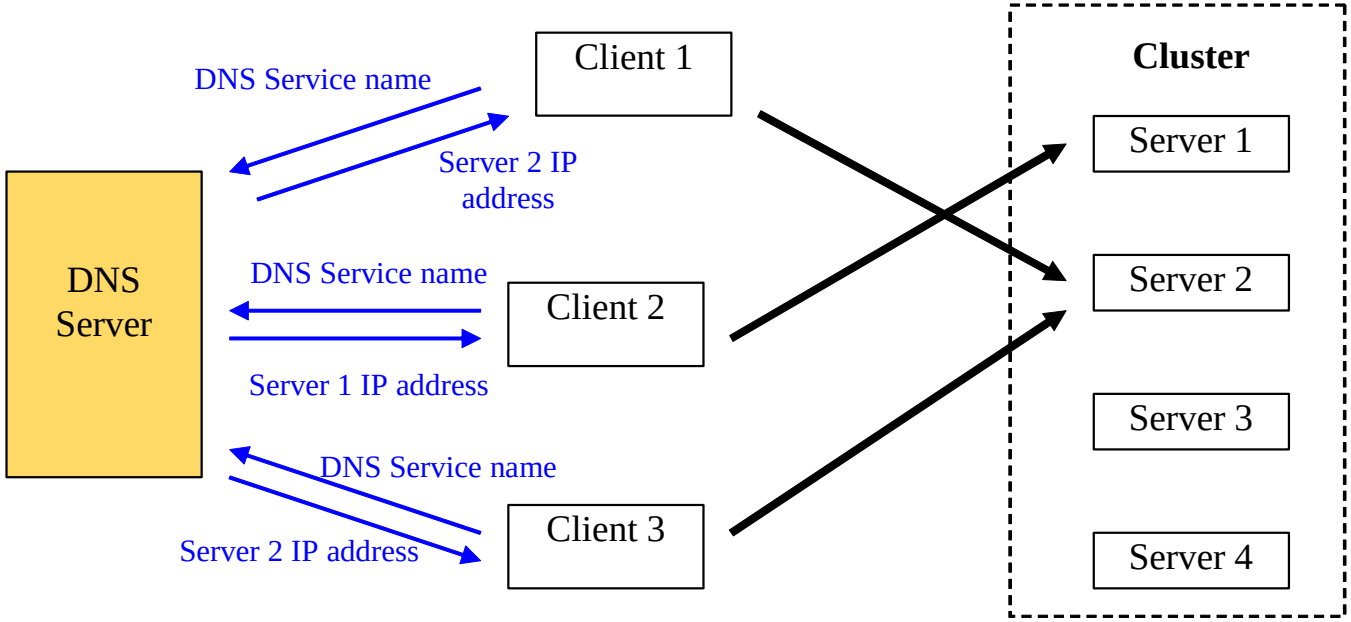


O controlador pode ser implementado através de um router a executar a função DNAT (*Destination Network Address Translation*) ou um servidor a funcionar como *proxy*, simplesmente redirecionando os pedidos que recebe.

Para muitos protocolos de aplicação será desejável que, uma vez atribuído um servidor a um cliente, essa atribuição se mantenha (ex.: cookies do HTTP). Caso contrário será necessária uma sincronização mais estrita entre os vários servidores.

Nome único de acesso ao *clusters* de servidores

Uma alternativa a um IP único é a utilização de um nome DNS único, a vantagem é que este sistema assenta sobre o funcionamento do DNS que é por conceção tolerante a falhas.



O controlador do *cluster* é o servidor DNS, é ele que distribui os clientes pelos vários servidores do *cluster* segundo critérios de *load balancing* e *fault tolerance*. Os registos DNS fornecidos pelo servidor DNS aos clientes terão necessariamente de ter um tempo de vida (TTL) muito curto pois enquanto não expirarem continuam a ser usados sem nova consulta ao servidor.

Failover e *load balancing* pelos clientes

Uma forma de evitar o ponto único de falha no acesso ao *cluster* é violar uma das características que um *cluster* deve ter e permitir aos clientes o acesso a servidores individuais do *cluster*. Neste caso o controlo passa para o cliente.

O cliente possui uma lista de servidores alternativos e implementa ele próprio o *failover*, ou seja tenta utilizar um deles, se não obtém resposta passa a utilizar outro da lista.

Em complemento ao *failover*, o cliente pode também implementar uma forma de *load balancing* básico fazendo uma rotação da lista (*round-robin*).

Na implementação do *failover* pelo cliente é fundamental definir de forma adequada o tempo máximo (*timeout*) das tentativas de acesso aos servidores.

Este método é muito usado em várias circunstâncias tais como clientes DNS, clientes SMTP, clientes LDAP (nss_ldap), etc.

Os servidores DNS podem ser usados para centralizar este processo através de registos MX (SMTP), SRV e registos A/AAAA.

DNS – Round-robin

No DNS podemos definir registos iguais do tipo NS, MX, SRV, A e AAAA, ou seja que associam o mesmo nome DNS a dados diferentes.

Exemplo:

```
ssh.dei.isep.ipp.pt.   60   A     193.136.62.80
ssh.dei.isep.ipp.pt.   60   A     193.136.62.81
ssh.dei.isep.ipp.pt.   60   A     193.136.62.82
ssh.dei.isep.ipp.pt.   60   A     193.136.62.83
ssh.dei.isep.ipp.pt.   60   A     193.136.62.84
ssh.dei.isep.ipp.pt.   60   A     193.136.62.85
```

Neste exemplo existem 6 registos A todos com o mesmo nome, quando um cliente pede ao DNS a resolução do nome **ssh.dei.isep.ipp.pt** vai obter uma lista de 6 endereços IP.

No entanto, tendo em vista o *load balancing*, o servidor DNS não os fornece na mesma ordem a todos os clientes, vai rodando a lista entre pedidos (*round-robin*).

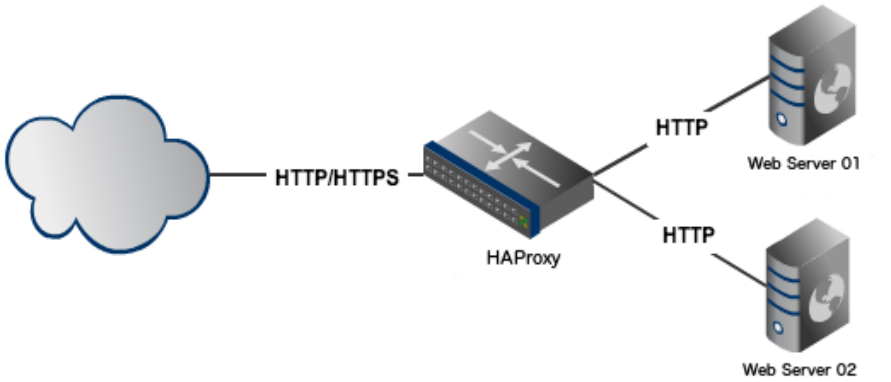
Assumindo que cada cliente tenta em primeiro lugar usar o primeiro endereço da lista fornecida, isso leva a que os clientes se distribuam pelos vários servidores.

Esta funcionalidade deve ser implementada por todos os servidores DNS sempre que existem registos com nomes repetidos.

Linux – HAProxy (The Reliable, High Performance TCP/HTTP Load Balancer)

O HAProxy (http://www.haproxy.org/) é um controlador de *cluster* para Linux capaz de redirecionar ligações TCP para um *cluster* de servidores incluindo funções de segurança e funcionalidades específicas para o protocolo HTTP.

Entre outras funcionalidades, o HAProxy permite por exemplo receber ligações protegidas por SSL/TLS como HTTPS e transformar as mesmas em ligações não seguras no interior do *cluster*.



Esta possibilidade torna-se útil porque como os clientes acedem ao *cluster* através do nome DNS da máquina onde se encontra o HAProxy, assim evita-se ter de manter certificados de chave pública iguais em todos os servidores do *cluster*.

O HAProxy permite implementar diversas funções de segurança que vão desde a limitação do número de ligações por segundo até à validação das linhas de cabeçalho HTTP antes de serem retransmitidas aos servidores do *cluster*.

Frontends e backends

No HAProxy um **frontend** representa um ponto de acesso para clientes, um **backend** é um conjunto de servidores do *cluster*.

O ficheiro de configuração, é constituído por secções contendo parâmetros de configuração, nomeadamente uma secção **global** e diversas secções **defaults**, **frontend**, **backend** e **listen**.

Declaração da secção	
global	Define parâmetros globais relacionados com o funcionamento geral do HAProxy.
defaults [NOME]	Define conjuntos de parâmetros que serão aplicados a todas as secções seguintes, a menos que sejam alterados na secção ou através de outra declaração defaults.
frontend NOME	Define um frontend, nomeadamente através do parâmetro **bind** e do parâmetro **use_backend**.
backend NOME	Define um backend, nomeadamente através de parâmetros **server**.
listen NOME	Forma de declaração mais expedita de um proxy incluindo na mesma declaração frontend e backend.

Stickiness

Quando o HAProxy é contactado por um cliente, para o **frontend** correspondente estará associado um **backend**, de entre os vários servidores definidos no **backend** será atribuído um a esse cliente.

Devido ao funcionamento dos protocolos de aplicação, o HAProxy tem de manter essa associação cliente<->servidor em pedidos subsequentes, isso é designado ***stickiness***.

O mecanismo de *stickiness* é gerido automaticamente através de *stick-tables*, mas há vários parâmetros que permitem o seu controlo.

Parâmetros ***mode tcp*** e ***mode http***

Para cada forntend/backend, são suportados dois modos de operação; **tcp** e **http**. O modo tcp (mode tcp) retransmite a informação diretamente entre o cliente e o servidor sem a analisar. No modo http (mode http) é realizada uma análise e validação das mensagens HTTP antes de serem transmitidas ao servidor.

Parâmetro ***balance ALGORITHM***

O parâmetro **balance** pode ser usado nas secções **backend** e **listen**, serve para definir o modo de balanceamento de carga entre os servidores ativos dos que estão definidos no **backend.**

Existem cerca de nove algoritmos disponíveis, a maioria dos quais é parametrizável. Se não for definido nenhum, será usado o valor **roundrobin**. O algoritmo **roundrobin** é ponderado dinamicamente, isto é, a probabilidade de um servidor ser selecionado depende da sua eficiência anterior.

Outro algoritmo possível é **leastconn**, nesse caso o eleito é o servidor com menor número de ligações ativas no momento. Entre vários servidores com o mesmo número de ligações, é aplicado round-robin.

Note-se que o algoritmo só é usado quando se trata de um cliente desconhecido, ausente da *stick-table* ou quando há necessidade de lhe atribuir um novo servidor porque o anterior deixou de estar disponível.

HAProxy - exemplo de configuração

A figura representa uma secção *listen* do ficheiro de configuração:

```
listen http_proxy
     bind 10.2.0.1:80,10.2.0.1:8080
     stats enable
     mode http
     option httpclose
     option forwardfor
     balance roundrobin
     server S1 192.168.5.10:80 check
     server S2 192.168.5.11:80 check backup
     server S2 192.168.5.12:80 check
```

Disponibiliza informação estatística via HTTP no URL /haproxy?stats

Insere nos cabeçalhos HTTP Connection: close

Insere nos cabeçalhos HTTP X-Forwarded-For: IP-CLIENTE

A opção **check** na declaração server força a verificação periódica do seu estado através de ligações TCP.

Um servidor declarado como **backup** só é usado se nenhum dos outros estiver disponível.

Windows Server – NLB e *clustering*

A *feature* NLB (*Network Load Balancing*) do Windows Server permite que o servidor possa ser integrado num *cluster*, designado NLB *Cluster*.

Os clusters NLB de servidores Windows utilizam uma abordagem diferente: todas as máquinas que são membros do *cluster* vão ter o mesmo endereço de nó, que será o endereço do *cluster*.

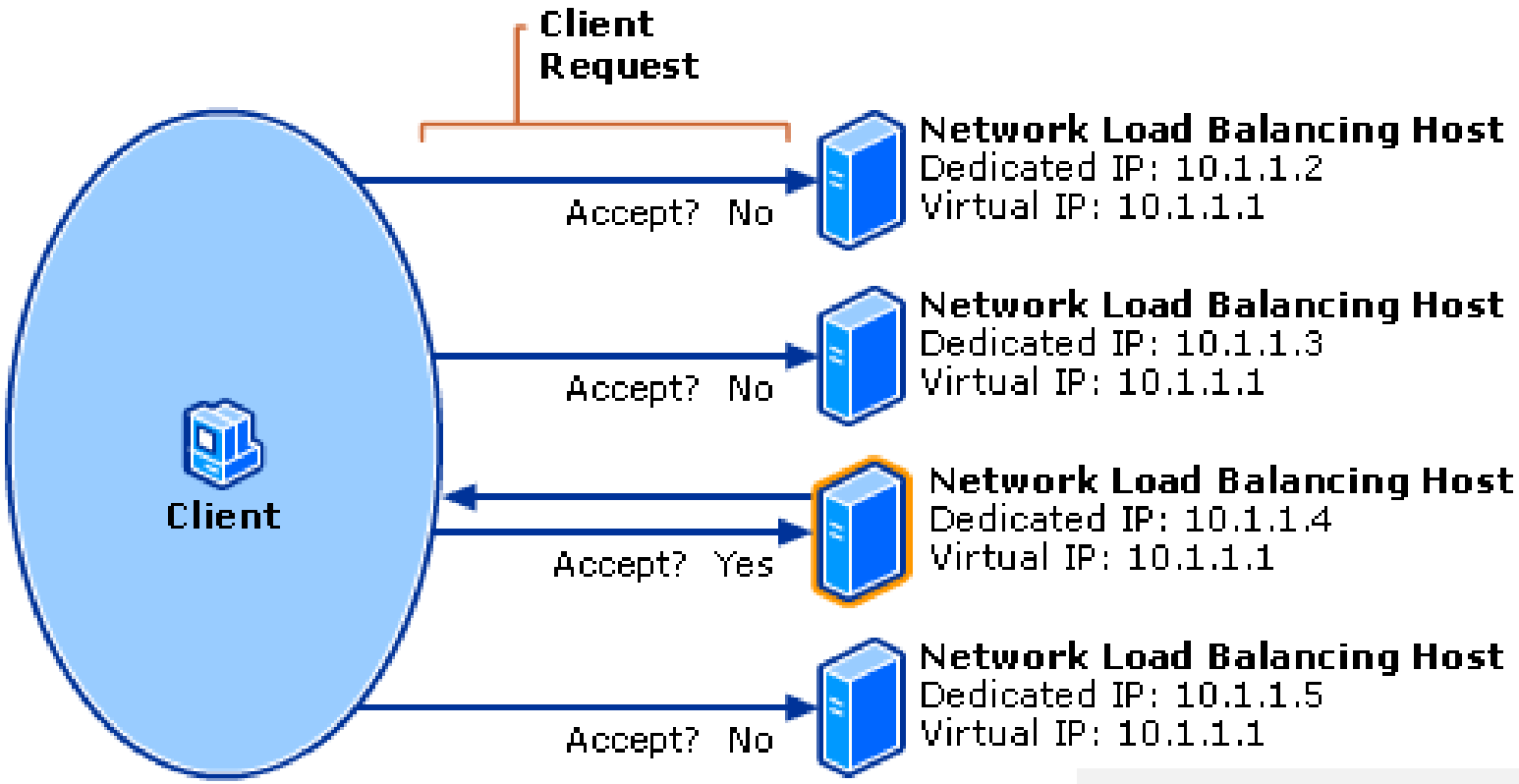
Quando um cliente envia um pedido para o endereço do *cluster*, todas as máquinas que são membros do *cluster* vão receber uma cópia do pedido. Apenas um membro pode responder, isso é estabelecido entre os vários membros do *cluster*.

Se um membro deixa de estar disponível, os pedidos continuam a ser respondidos por outros membros.

A grande vantagem dos *clusters* NLB é a inexistência de um ponto único de falha (SPOF), não existe um controlo centralizado do *cluster*, ele é gerido de forma distribuída por todos os membros. Desde que pelo menos um membro esteja a funcionar, o *cluster* está disponível.

Clusters NLB – Funcionamento da rede

Num *Cluster* NLB todos os membros possuem o mesmo endereço de nó, mas apenas um dos membros responde a cada pedido.



(Fonte: technet.microsoft.com)

Os membros do *cluster* podem manter o seu endereço IP único (*Dedicated IP*), mas todos partilharão um endereço comum entre eles (*Primary / Virtual IP*) que será o endereço do NLB *Cluster*.

Clusters NLB – Unicast nível 2

Estão disponíveis duas formas de garantir que os pedidos dirigidos ao endereço do *cluster* NLB chegam a todos os membros:

- Unicast nível 2 (MAC)
- Multicast no nível 2 (MAC)

A abordagem recomendada pela Microsoft é unicast de nível 2. Nesta configuração, os endereços de nível 2 (MAC/Ethernet) das interfaces de rede de todos os membros do *cluster* são substituídos por um endereço único igual em todos os membros (***inbound destination address***).

Os *frames* emitidos pelos membros do cluster nunca usam este endereço como origem pois isso levaria os comutadores de rede a inclui-lo na sua tabela MAC e os pedidos seguintes deixariam de ser entregues em todos os membros do *cluster*.

Quando um membro do *cluster* emite uma resposta usa como endereço MAC de origem o ***outbound source address*** que é único para cada membro do *cluster*.

Clusters NLB – Multicast nível 2

Esta abordagem é semelhante à anterior, a única diferença é que são utilizados endereços MAC multicast.

Nas interfaces de rede de todos os membros do *cluster* é adicionado um endereços de nível 2 (MAC/Ethernet) multicast, igual em todos os membros (*inbound destination address*). Como é um endereço multicast, o endereço original da interface pode manter-se. Sendo um endereço MAC multicast já não existem problemas com o funcionamento dos comutadores de rede porque eles são sempre obrigados a retransmitir estes *frames* em todas as suas portas.

A tabela anexa apresenta os endereços MAC usados. Enquanto o endereço *inbound* é igual para todos os membros do *cluster*, os endereços de origem (*source outbound*) contêm o valor P que é único para cada membro.

IP Mode	MAC Address	Explanation
Unicast inbound	02-BF-W-X-Y-Z	W-X-Y-Z = IP address Onboard MAC disabled
Multicast inbound	03-BF-W-X-Y-Z	W-X-Y-Z = IP address Onboard MAC enabled
Source outbound	02-P-W-X-Y-Z	W-X-Y-Z = IP address P = Host priority

(Fonte: technet.microsoft.com)

Clusters NLB – modos

A implementação de um *cluster* NLB nunca pode combinar diferentes formas de operação, todos os membros do *cluster* operam da mesma forma, unicast ou multicast.

Seja qual for a opção, o facto de todos os membros do *cluster* usarem o mesmo endereço IP, obriga a que se encontrem todos na mesma rede de nível 2 (domínio de broadcast).

No **modo unicast** o endereço MAC original da interface de rede é desativado e substituído por um endereço na forma 02:bf:…, comum a todos os membros, em consequência da desativação do endereço MAC original o membro deixa de ficar acessível individualmente e fica apenas acessível como *cluster*. Para poder comunicar com individualmente com um membro de um *cluster* unicast é necessário dotar o membro de uma interface de rede adicional, caso contrário para aceder individualmente a um membro é necessário primeiro retira-lo do *cluster*.

No **modo multicast** o problema anterior não ocorre porque um endereço MAC multicast pode ser adicionado à interface sem desativar o endereço MAC original. Assim a mesma interface de rede pode ser usada para o endereço virtual do *cluster* e para o endereço dedicado do membro.

Clusters NLB – multicast com IGMP

Devido à forma de operarem com os *frames*, garantido que chegam a todos os membros do *cluster*, os *clusters* NLB impedem segmentação pelos comutadores.

Um *cluster* NLB inunda todo o domínio de broadcast com *frames*, o ideal é portanto limitar o máximo possível o domínio de broadcast, ou seja a rede do *cluster* deve ser de preferência uma rede dedicada.

No modo multicast pode ser usado o IGMP (*Internet Group Membership Protocol*). Usando a opção ICMP o *multicast inbound destination address* é definido de forma diferente e os membros emitem mensagens IGMP para se registrem em grupos de multicast.

Usar o IGMP apenas tem interesse se no domínio de broadcast existem comutadores de rede que o suportem. Através das mensagens IGMP esses comutadores evitam a propagação de *frames* multicast para locais desnecessários.

Clusters NLB - ARP

Quando um nó localizado na rede do *cluster* (tipicamente o router desse rede) tenta aceder ao endereço IPv4 do *cluster* (*primary or virtual address*) vai tentar obter o endereço MAC correspondente através do ARP (*Address Resolution Protocol*). A forma como o *cluster* NLB usa os endereços MAC pode então causar problemas.

Quando os membros do *cluster* respondem, a resposta ARP contém o endereço MAC *destination inbound unicast* ou *multicast* (igual em todos os membros), no entanto a resposta ARP é transportada num *frame* que tem como origem o endereço MAC *source outbound* (único para cada membro).

Este facto pode ser interpretado por alguns nós como uma incoerência e pode levar o nó a ignorar a resposta ARP por motivos de segurança.

Uma solução possível para esta situação é inserir nesses nós um entrada estática na tabela ARP, de forma a mapear o endereço MAC *destination inbound unicast* ou *multicast* para o endereço IPv4 do *cluster* (*primary or virtual address*).

Clusters NLB – Priority

Na implementação atual, um *cluster* NLB pode conter até 32 membros, cada membro tem de usar um número de prioridade diferente de 1 a 32.

O número de prioridade é definido pelo administrador no momento em que adiciona um membro ao *cluster*, o valor 1 significa prioridade mais elevada. Dependendo das regras de portos (*port rules*), o membro de prioridade mais elevada pode receber todos os pedidos dos clientes.

O número de prioridade também identifica o membro no *cluster*, sendo por exemplo colocado nos endereços MAC *source outbound* dos *frames* emitidos por esse membro.

O *cluster* NLB tem como objetivo receber tráfego TCP e UDP destinado a serviços de rede e distribuir esse tráfego pelos seus membros, a forma como essa distribuição é realizada é definida nas *port rules*.

As *port rules* apenas se aplicam a UDP e TCP, por exemplo o tráfego ICMP é sempre enviado a todos os membros do *cluster* (apenas o de prioridade mais elevada responde aos pedidos).

Clusters NLB – *Port rules*

Um *cluster* pode ter vários endereços (*primary or virtual address*), para cada um deles podem ser definidas diferentes *port rules*, no entanto todos os membros de um *cluster* têm de ter as mesmas *port rules*, quando se adiciona um novo membro ele adquire as *port rules* do *cluster*.

Para alterar as *port rules* num membro ele terá de ser retirado do *cluster*, as regras alteradas e novamente adicionado, estas operações podem ser realizadas sem perturbar o funcionamento do *cluster* sob o ponto de vista dos utilizadores.

Para cada endereço do *cluster* podemos definir várias *port rules*, cada uma especifica um intervalo de números de porto UDP e/ou TCP. Os intervalos em diferentes regras aplicadas ao mesmo endereço não se podem sobrepor.

O objetivo de uma *port rules*, é definir a forma de lidar com o tráfego correspondente que chega ao *cluster*, existem três alternativas (*filtering modes*):

- ***Multiple Host*** – o tráfego é distribuído pelos membros
- ***Single Host*** – modo de *failover*
- ***Disable*** – o tráfego não é permitido

Clusters NLB – *filtering mode single host*

O modo *default* é *single host*, isto é, aplica-se a qualquer tráfego que não corresponda a nenhuma *port rule*. Isto inclui desde logo tráfego que não pode ser definido numa *port rule*, como por exemplo ICMP.

O *filtering mode* ***single host*** corresponde a um comportamento de ***failover***, todo o tráfego é dirigido para o membro disponível com prioridade mais elevada, se ele fica indisponível passará a ser utilizado o membro disponível com prioridade mais elevada seguinte. Depois de adicionado ao *cluster*, para cada membro é possível definir um valor de prioridade (1 a 32) específico para cada *port rule*.

O *filtering mode* ***multiple host*** permite a distribuição do tráfego por todos os membros disponíveis.

Neste modo é também possível definir para cada membro a percentagem de pedidos que lhe são dirigidos, por omissão os pedidos são distribuídos de forma equitativa.

A definição da percentagem de pedidos atribuídos a cada membro só pode ser realizada nas propriedades do membro depois de adicionado ao *cluster* e não nas propriedades do *cluster*.

Clusters NLB – *multiple host e affinity*

No *filtering mode* ***multiple host*** é fundamental definir a persistência das associações entre clientes e membros do *cluster*. A *affinity* permite garantir que quando um primeiro pedido de um cliente é atribuído a um membro, os pedidos seguintes desse mesmo cliente também serão atribuídos ao mesmo membro. Sem esta garantia muitos protocolos de aplicação como por exemplo os que operam sobre TCP não podem funcionar.

Os modos de *affinity* disponíveis são:

none – todos pedidos são distribuídos pelos membros, só pode ser usado para serviços sem ligação muito simples. Por exemplo serviços UDP com um pedido e uma resposta.

single – todos os pedidos provenientes do mesmo endereço de origem (cliente) são enviados para o mesmo membro.

network – todos os pedidos provenientes da mesma rede de classe C (prefixo de 24 bits) são enviados para o mesmo membro.

Nos modos ***single*** e ***network*** pode ser especificado um *timeout* em minutos. Se for definido, quando o cliente não comunica durante esse período de tempo, a persistência é eliminada e numa próxima comunicação será atribuído um novo membro ao cliente.

Sincronização dos membros do *cluster*

Como os membros de um *cluster* devem aparentar ser um único servidor para os clientes, os serviços disponibilizados por cada um deles devem de ser dotados de mecanismos de sincronização que garantam que todos os membros possuem o mesmo estado.

A *affinity* ou *stickiness*, garante que durante uma sessão de trabalho de um cliente ele vai trabalhar sempre com o mesmo membro, mas isso pode não ser suficiente.

Por exemplo quando um utilizador faz um *upload* de um conteúdo para um site web que está implementado sob a forma de *cluster*, espera que esse conteúdo fique disponível a todos os utilizadores do *site/cluster*.

A sincronização dos membros do *cluster* deve ser implementada caso a caso para cada tipo de serviço disponibilizado, por exemplo podem ser usados sistemas de ficheiros em rede (NAS) para esse efeito, os servidores de bases de dados possuem normalmente mecanismos próprios de replicação.

Sincronização de certificados de chave pública

Desejavelmente, sob o ponto de vista de segurança, muitos protocolos de aplicação usam atualmente SSL/TLS. Nesse contexto o servidor fornece ao cliente um certificado de chave pública a ser validado pelo cliente para efeitos de autenticação do servidor.

O certificado de chave pública contém o nome DNS do servidor, num *cluster*, para que o certificado seja considerado válido pelo cliente, esse nome DNS tem de corresponder ao endereço IP do *cluster* (endereço usado pelos clientes no acesso ao *cluster*).

Consequentemente, todos os membros do *cluster* terão de usar esse mesmo certificado.

No caso do HAProxy a instalação e manutenção do mesmo certificado em todos os membros pode ser evitada. Quando o HAProxy opera em modo HTTP, pode ele próprio fornecer o certificado ao cliente, neste caso a ligação entre o cliente e o HAProxy usa HTTPS enquanto a ligação correspondente do HAProxy ao membro pode usar HTTP.