

TMOS: Redefinindo a solução

Visão Geral Historicamente, há dois modos de criar dispositivos de application delivery networking: para desempenho ou para inteligência. No mercado aberto, os clientes normalmente optam por soluções que exibem o melhor desempenho. Como resultado, a maioria dos fornecedores desenvolveu seus dispositivos em designs mais rápidos, baseados em pacotes, em vez da arquitetura baseada em proxy, com menor desempenho. Conforme aumenta a necessidade de inteligência nesses dispositivos, os fornecedores se encontram em uma posição precária: quanto mais adicionam inteligência ao dispositivo, a pedido dos clientes, mais eles se parecem com proxies e menor é o desempenho.

A F5 Networks escolheu o caminho baseado em pacotes, mas também começou, ao mesmo tempo, a tratar do problema principal - a criação de uma solução inteligente que também ofereça alto desempenho. O resultado é a arquitetura TMOS, um conjunto de funções e funcionalidades em tempo real, planejada e criada como uma solução de proxy completa, com o poder e o desempenho exigidos pelas infra-estruturas de rede modernas.

Pacotes X Proxy

Para entender como a TMOS é única e poderosa, é importante observar com atenção a história desses dispositivos e o dilema entre velocidade e inteligência nas soluções baseadas em pacotes e proxy.

O que é um design baseado em pacotes?

Um dispositivo de rede com design baseado em pacotes (ou **pacote-a-pacote**) fica localizado no meio de um fluxo de comunicações, mas não é um terminal para essa comunicação - ele apenas encaminha os pacotes. Normalmente, um dispositivo que opera pacote-a-pacote tem algum conhecimento dos protocolos que passam por ele, mas está longe de ser um terminal de protocolos real. A velocidade desses dispositivos é baseada principalmente na ausência de conhecimento de toda a pilha do protocolo, reduzindo a quantidade de trabalho necessária para cuidar do tráfego. Por exemplo, com o TCP/IP, esse tipo de dispositivo pode entender o protocolo somente o bastante para reescrever endereços IP e portas TCP, ou seja, apenas cerca de metade da pilha completa.

Conforme as redes se tornam mais complexas e aumenta a necessidade de inteligência, designs mais avançados baseados em pacotes começaram a surgir (incluindo os produtos BIG-IP v4.X da F5). Esses dispositivos conheciam o TCP/IP bem o bastante para entender a configuração e estrutura de conexão do TCP, modificar os cabeçalhos TCP/IP e até inserir dados em fluxos TCP. Como esses sistemas podiam inserir dados em fluxos TCP e modificar seu conteúdo, eles também tinham de reescrever os valores da seqüência (SEQ) e reconhecimento (ACK) do TCP nos pacotes sendo transmitidos entre cliente e servidor. As soluções BIG-IP v.4.x da F5 compreendiam o TCP/IP e o HTTP bem o bastante para identificar solicitações HTTP individuais e podiam encaminhar solicitações diferentes para servidores diferentes, reutilizando conexões que o dispositivo BIG-IP já tinha aberto.

Embora tudo isso seja possível usando uma arquitetura pacote-a-pacote muito sofisticada (o BIG-IP v4.x era um dos designs mais sofisticados da época), isso exigia um sistema muito complexo de rastreamento de estado para compreender os protocolos TCP/IP e HTTP bem o bastante para reescrever conteúdo do cabeçalho, inserir dados e manter suas próprias conexões com clientes e servidores.

Apesar de sua crescente complexidade, os designs baseados em pacotes ainda são menos complexos e mais rápidos do que arquiteturas tradicionais baseadas em proxy, visto que possuem a vantagem de utilizar somente uma pequena porcentagem da lógica necessária para um full proxy.

O que é um design baseado em proxy (full proxy)?

Um design de full proxy é o oposto de um design pacote-a-pacote. Em vez de ter uma compreensão mínima do fluxo de comunicações passando pelo dispositivo, um full proxy entende totalmente os protocolos e é, ele próprio, um terminal e um ponto de origem para os protocolos.

A conexão entre um cliente e o full proxy é independente da conexão entre o full proxy e o servidor, enquanto na arquitetura pacote-a-pacote o que existe é basicamente um canal direto de comunicações entre o cliente e o servidor (embora o dispositivo entre eles possa manipular os pacotes transmitidos).

Na verdade, como o full proxy é um terminal para o protocolo, ele deve implementar os protocolos completamente, como cliente ou como servidor (o que não é o caso na arquitetura baseada em pacotes). Isso também significa que o full proxy pode ter seu próprio comportamento na conexão TCP, como buffering, retransmissões e opções do TCP. Com um full proxy, cada conexão é única e pode ter seu próprio comportamento de conexão TCP. Isso significa que um cliente conectando ao dispositivo de full proxy provavelmente terá um comportamento de conexão diferente daquele usado na conexão com os servidores de segundo plano. Portanto, um full proxy permite a otimização de cada conexão de maneira individualizada, independentemente da fonte original e do destino final. Mais ainda, um full proxy compreende e processa cada protocolo como um cliente ou servidor real fariam, usando camadas. Tendo o HTTP como exemplo, o protocolo IP é processado primeiro, depois o TCP e então o HTTP; cada camada não tem conhecimento das camadas inferiores.

Os dispositivos de full proxy têm a vantagem de um suporte mais fácil aos protocolos no nível de aplicação e, como compreendem perfeitamente o protocolo, oferecem um grau de flexibilidade muito maior na otimização ativa desses protocolos (ao contrário do design pacote-a-pacote, que só tem suporte passivo aos protocolos). Além disso, um full proxy pode fornecer mais facilmente as funcionalidades avançadas para inspeção e manipulação de protocolos e seus dados. Apesar da carga adicional do suporte completo aos protocolos de aplicativos e da complexidade de ter de entender os aspectos tanto do cliente quanto do servidor, um dispositivo de full proxy é muito mais capaz de manter interações inteligentes com os protocolos, pois possui muito mais informações disponíveis do que a arquitetura pacote-a-pacote.

Desafio

Redefinindo a solução

É amplamente conhecido que as soluções baseadas em proxy, ou pelo menos a inteligência oferecida por elas, eram a solução definitiva. Entretanto, o desempenho muito superior das arquiteturas pacote-a-pacote mais do que compensou sua inteligência limitada. Por algum tempo, essa era uma troca aceitável para a maioria das redes corporativas.

Conforme cresce a necessidade por inteligência, as soluções baseadas em pacotes estão rapidamente experimentando as mesmas restrições de desempenho que sempre afetaram as soluções baseadas em proxy. A complexidade do desenvolvimento de soluções baseadas em pacotes está rapidamente alcançando a das arquiteturas baseadas em proxy. Apesar dos sensíveis aumentos da potência de hardware e software, as arquiteturas baseadas em pacotes não podem acompanhar

a necessidade de inteligência e desempenho. É inaceitável ter de escolher entre um e outro.

As soluções baseadas em pacotes tiveram seu tempo, mas esse tempo acabou. Agora, é óbvio que concentrar esforços no desempenho em detrimento da inteligência não forneceu uma solução viável. A solução real é criar uma solução baseada em proxy com o desempenho de uma solução baseada em pacotes. TMOS:

Solução

TMOS

A TMOS é um termo coletivo usado para descrever uma arquitetura personalizada e criada com um propósito específico pela F5, que investiu muito tempo e dinheiro para desenvolver os alicerces dos produtos em evolução da F5. De uma perspectiva de alto nível, a TMOS é:

Uma coleção de módulos

Cada módulo executa uma função determinada. Por exemplo, há um módulo de driver de rede, um módulo Ethernet, um módulo ARP, um módulo IP, um módulo TCP e assim por diante. Cada componente do sistema é independente, o que ajuda a reduzir a complexidade do sistema e facilita o desenvolvimento futuro. Adicionar suporte a novos protocolos é simplesmente questão de adicionar um novo módulo. Esse design também permite uma reutilização muito fácil. Se um novo protocolo do nível de aplicação é executado sobre o TCP/IP, seria simples vincular os módulos para que o novo protocolo tenha acesso aos dados do TCP/IP sem ter de compreender os protocolos de baixo nível.

Independentes e autônomos

Muitas pessoas notam que um dispositivo baseado na TMOS executa um formato do Linux, que pode ser visto enquanto se administra o dispositivo pela linha de comando. É importante observar que esse sistema Linux não está envolvido em nenhum aspecto com o tráfego transmitido pela TMOS. A TMOS tem seu próprio processador, memória e barramento de sistema (todos dedicados) para acesso a dispositivos periféricos. Quando um dispositivo baseado na TMOS recebe pacotes, tudo nele, do cabo da rede ao barramento do sistema, do subsistema de rede ao subsistema de gerenciamento de rede, é independente e completamente contido na própria TMOS. O Linux jamais se envolve ou percebe nada disso, nem mesmo o kernel. O sistema Linux é usado somente para tarefas administrativas como a linha de comando ou a interface gráfica da web. A razão para isso é simples: um sistema operacional que é ideal para operações de gerenciamento de tráfego de alta velocidade não é ideal como um sistema operacional de uso geral. Portanto, faz sentido usar um sistema operacional de uso geral para tarefas gerais, como o gerenciamento, e deixar o gerenciamento do tráfego para o sistema operacional criado para esse fim, a TMOS.

Um sistema operacional de tempo real

Um sistema operacional de tempo real - isso significa que o TMOS não tem um agendador preemptivo de processamento. Para sistemas operacionais de uso geral, é muito desejável possuir um agendador preemptivo no kernel, visto que todos os processos podem obter uma parte justa do tempo do processador, e muitos deles podem ser executados basicamente ao mesmo tempo. Em sistemas operacionais otimizados de propósito específico como o TMOS, um agendador não seria vantajoso, pois adiciona uma carga desnecessária. O TMOS foi criado e ajustado para que cada componente do sistema execute as operações necessárias e, então, deixe o próximo componente ser executado. Isso reduz de forma significativa a carga do agendamento do processador,

eliminando as interrupções, mudanças de contexto e a maioria do trabalho executado normalmente por um agendador. Isso também permite um controle total sobre quando e em que ordem o processamento ocorre.

Hardware e software

Como o TMOS possui um design modular, não importa se funções específicas são gerenciadas por hardware ou software. Com o TMOS, tudo pode ser feito por meio de software, usando módulos altamente otimizados e de propósito específico; entretanto, operações intensivas que consomem recursos também podem ser transferidas para hardware especializado. Por exemplo, o TMOS tem sua própria pilha SSL e pode processá-la inteiramente via software, mas é muito mais rápido transferir as operações de criptografia para os ASICs SSL especializados. Ter uma tecnologia completa de software permite uma flexibilidade virtualmente infinita; ter hardware especializado para a transferência de carga permite um escalonamento acessível para níveis de desempenho líderes na indústria.

Conduzido por eventos

A combinação de modularidade e processamento em tempo real dá ao TMOS a capacidade única de mudar seu comportamento com base em eventos reais, em tempo real. Cada evento, da conexão inicial ao processamento de conteúdo - e até a devolução do tráfego do servidor para o cliente - constitui uma oportunidade para que o TMOS mude seu comportamento para atender ao requisito atual. Essa funcionalidade faz do TMOS a solução mais adaptável e flexível que existe.

Todos esses itens tornam o TMOS uma solução extremamente poderosa e adaptável. A modularidade combinada em um sistema operacional de tempo real independente e conduzido por eventos dá ao TMOS capacidades inéditas. Por exemplo, o TMOS pode utilizar três pilhas de rede exclusivas criadas para atender a requerimentos de implementação. Primeiro, há a pilha FastL4, um conceito tradicional pacote-a-pacote de estado TCP/UDP limitado que gerencia altas taxas de conexão, mas com requerimentos limitados de funcionalidade (4ª camada e abaixo). Então, há a pilha FastHTTP; uma pilha TCP/HTTP de estado limitado que representa um conceito pacote-a-pacote extremamente avançado para gerenciar altas taxas de conexão com requerimentos mais altos de inteligência no HTTP (7ª camada). Por último, há o Fast Application Proxy; a pilha padrão baseada em full proxy que representa o ápice do conceito baseado em proxy. Como o TMOS é capaz de usar componentes de hardware e software de maneira intercambiável, se o sistema estiver equipado com o hardware PVA, é possível que toda a carga da pilha FastL4 seja transferida para o Packet Velocity ASIC (PVA) da F5. A decisão sobre a melhor abordagem é toda do cliente.

Qualquer destes três descritores de alto nível bastaria para diferenciar o TMOS de qualquer solução existente, baseada em pacotes ou em proxy. Essa arquitetura já seria suficiente para mudar o mercado de application delivery networking, mas mesmo a melhor arquitetura, se montada com componentes inferiores, resultará em uma solução inferior. Os componentes ou módulos com que o TMOS é montado fazem dele uma solução superior.

O próximo nível

O que torna a arquitetura TMOS extraordinária são os módulos personalizados criados para suportá-la. Entre os mais importantes, eis alguns dos mais utilizados: TCP Express, Fast Application Proxy e iRules.

TCP Express

As arquiteturas pacote-a-pacote simplesmente não podem oferecer o que o TMOS, com o conjunto de funções TCP Express, pode. Há companhias que têm um full proxy, mas limitado pelo fato de que é apenas um componente em um sistema de uso geral baseado em UNIX. Eles não podem alcançar o desempenho de um sistema operacional de tempo real personalizado, criado para oferecer baixas taxas de latência e um desempenho de rede magnífico. Não há como implementar esse tipo de funcionalidade em um produto já criado - é necessário um compromisso arquitetônico fundamental com as redes de alto desempenho e baixa latência.

O TCP Express é um conjunto de melhorias de eficiência do TCP, na forma de padrões da Internet (RFCs) e de centenas de funções e ajustes personalizados da F5, baseados em sua grande experiência no mundo real.

O TMOS suporta todas as modernas melhorias de eficiência do TCP, incluindo:

- Delayed and Selective Acknowledgements, (RFC 2018)
- Explicit Congestion Notification ECN, (RFC 3168)
- Limited and Fast Retransmits (RFC 3042 e RFC 2582)
- Slow Start with Congestion Avoidance (RFC 2581)
- Adaptive Initial Congestion Windows (RFC 3390)
- TimeStamps and Windows Scaling (RFC 1323)
- TCP Slow Start (RFC 3390)
- Bandwidth Delay Control and muito mais (Vegas, NewReno)

Essas funções e mais de cem outras são meios de alcançar o melhor desempenho em qualquer cenário de conexão. Cada dispositivo com que o TMOS interage tem um cenário de rede diferente. Para obter o melhor desempenho para todos os dispositivos, o TMOS tem de reagir de forma inteligente aos requerimentos únicos de cada dispositivo, em cada conexão. Não basta suportar uma ou duas otimizações avançadas do TCP. Não basta desenvolver suas próprias otimizações e otimizar seu produto para campos específicos. A otimização completa do TCP, as extensões personalizadas e os exaustivos testes em condições reais são necessários, todos juntos, em um sistema operacional de tempo real e com latência otimizada, para se obter o desempenho ideal. Todas essas coisas devem ser feitas juntas para se obter um benefício real significativo, e é isso que a F5 fez com o TMOS.

Fast Application Proxy

O componente Fast Application Proxy do TMOS, a pilha de full proxy, é outro diferencial importante. Isso permite ao TMOS fornecer mais funções de aceleração e otimização do que qualquer outra solução, antes ou depois dele. Normalmente, a inspeção ou lógica inteligente chega ao custo da velocidade, e isso é natural - quanto mais trabalho é exigido por conexão, menor o número de conexões que podem ser gerenciadas. O que torna o Fast Application Proxy exclusivo é que, empregando hardware de forma mais transparente quando possível, em conjunto com um sistema operacional de tempo real e alto desempenho (TMOS), ele pode atingir um desempenho inédito, oferecendo todos os desempenhos de uma arquitetura real de full proxy. Além disso, o Fast Application Proxy oferece a fluência necessária para implementar muitos outros módulos TMOS, incluindo:

- Compactação HTTP
- Multi-Store Caching
- Aceleração SSL
- Fast Cache
- Spooling de Conteúdo
- Compactação Inteligente
- QoS/ToS
- L7 Rate Shaping

Todas as partes se encaixam. O TMOS fornece uma arquitetura que permite que o Fast Application Proxy forneça inteligência e desempenho. O Fast Application Proxy, usando a arquitetura TMOS, permite o uso inteligente de vários outros módulos, de hardware e software, que oferecem um desempenho ainda maior.

iRules

Essa é uma das capacidades mais exclusivas do TMOS. As *iRules* são scripts criados usando o padrão Tool Command Language (TCL) com extensões personalizadas da F5, que permite aos usuários criar funções únicas, ativadas por eventos do TMOS. Além das regras em si serem fáceis de criar e entender, esse módulo as compila em códigos de bytes que executam ações, como ler e escrever cookies HTTP, por meio de chamadas de funções personalizadas e otimizadas no núcleo do TMOS. A combinação dessas duas tecnologias significa que o texto simples da regra TCL resulta em códigos de bytes de alto desempenho, que executam a inspeção e manipulação de maneira original no TMOS, portanto, rapidamente.

Desde o lançamento inicial do TMOS, os clientes da F5 descobriram centenas de modos de aproveitar o poder do sistema *iRules* no TMOS. Por exemplo:

- Limitação de taxas no DNS.
- Implementação de um proxy SMTP para inspecionar e direcionar mensagens individuais.
- Reorganização de cookies HTTP para facilitar a análise nos sistemas de back-end.
- Autenticação de conexões de usuários com um servidor RADIUS de back-end, usando credenciais armazenadas em cookies HTTP.
- Implementação de um inspetor LDAP para verificar parâmetros de solicitações LDAP bind().
- Uso seletivo da re-criptação SSL nos servidores de back-end, somente para certas URLs HTTP, e requisição seletiva de certificados de cliente SSL com base em URIs HTTP.
- Criação e inserção de valores personalizados de identificador de sessão em uma solicitação HTTP, que será respondida pelo servidor, seguida pela inspeção da resposta, para avaliar se ela corresponde à solicitação.
- Caso isso não aconteça, registrar todas as informações da sessão e as 100 últimas solicitações.
- Multiplexação de solicitações CORBA/IIOP

Não há outro dispositivo no mundo que possa implementar uma lógica tão sofisticada em seu conjunto de funções de inspeção. Essas são *iRules* simples, que fazem bom uso da incomparável flexibilidade oferecida pelo TMOS e de sua arquitetura orientada por eventos. Além disso, embora não possamos atribuir isso diretamente ao TMOS, as *iRules* e o seu desenvolvimento acabaram gerando sua própria comunidade, com mais de 10.000 membros no mundo inteiro (devcentral.f5.com).

Embora o TCP Express, o Fast Application Proxy e as iRules sejam apenas três dos componentes mais exclusivos e otimizados criados para e no TMOS, eles representam uma visão abrangente do fato de que a TMOS não se limita a uma arquitetura inovadora, mas também inclui os componentes mais modernos para dar vida a essa arquitetura.

Começando novamente

Como você pode verificar, o desenvolvimento de conceitos pacote-a-pacote foi originado pela necessidade de fornecer dispositivos de redes de distribuição de aplicativos que oferecessem um desempenho excepcional e também inteligência, ainda que menos inteligência do que suas contrapartes baseadas em proxy. A capacidade desses dispositivos em atender às demandas das redes modernas se esgotou e a maioria dos fornecedores hoje se encontra exatamente onde começaram: tendo de oferecer a inteligência de um full proxy combinada com os ainda maiores requerimentos de desempenho das redes de hoje. A escolha da arquitetura baseada em pacotes provou ser precipitada.

A F5 Networks percebeu muito cedo que a única solução em longo prazo era uma arquitetura baseada em proxy, criada para fornecer desempenho e flexibilidade incomparáveis no gerenciamento de tudo o que o futuro possa trazer. O resultado dessa idéia, bem como um significativo investimento em tempo de desenvolvimento e fundos, é o TMOS. A TMOS é a primeira arquitetura de proxy de tempo real modular, independente, orientada a eventos e de propósito específico, com a capacidade de usar hardware de forma transparente e software de modo unilateral para obter o melhor desempenho e a maior inteligência. Além disso, a TMOS vai além de uma arquitetura revolucionária, estabelecendo um novo padrão para cada um dos componentes usados para criá-la. Do TCP Express, que garante as conexões mais eficientes no nível da rede, tanto para o cliente como para o servidor, ao Fast Application Proxy, que oferece inteligência e desempenho, e às iRules, que permitem o controle completo e a personalização de todas essas funções e de todos os numerosos componentes disponíveis para aceleração de aplicativos, segurança e disponibilidade, a TMOS é a única solução de redes de distribuição de aplicativos que não está começando tudo de novo. A TMOS já é a solução certa.

Sobre a F5

A F5 Networks é a líder global em Application Delivery Networks. A F5 fornece soluções que tornam os aplicativos seguros, rápidos e disponíveis para todos, ajudando as companhias a obter o maior retorno pelo seu investimento. Ao implementar inteligência e gerenciabilidade na rede para transferir a carga de aplicativos, a F5 os otimiza, permitindo que eles trabalhem mais rápido e consomam menos recursos. A arquitetura expansível da F5 integra de forma inteligente a otimização de aplicativos, protege os aplicativos e a rede e oferece confiabilidade aos aplicativos - tudo em uma plataforma universal. Mais de 10.000 companhias e provedores de serviços em todo o mundo confiam na F5 para manter seus aplicativos funcionando. A companhia tem sede em Seattle, Washington, com escritórios no mundo todo. Para mais informações, visite www.f5.com (em inglês).