

TMOS: 对解决方案进行全新设计 [概述](#) [挑战](#) [解决方案](#)

概述

从前，我们有两种方式部署应用交付网络设施；面向性能的部署或面向智能的部署。在开放市场上，客户传统上选择那些具有最佳性能的解决方案。结果造成，大多数厂商将其设施构建于运行速度更快、基于分组的架构；而非性能较差、基于代理的架构。由于这些设备对智能的需求有所增加，因此提供上述解决方案的各个厂商发现其自身处

不妙：这些厂商向设备添置的客户所需的智能特性越多，他们越像是在配置代理，并且性能越糟。

F5 Networks 公司最初采用基于分组的方式，但同时也致力于解决根本的问题——为用户提供一款可提供较高性能的智能解决方案。TMOS 架构因此应运而生，这一架构具备各种实时特性与功能，基于特定目的构建、是一款全代理解决方案，并具有当今网络基础设施所需的强大功能与较高性能。

基于分组与基于代理

要想了解 TMOS 架构的独特特性与强大功能，至关重要的一点是须密切关注该设备的历史情况，并在运行速度与智能特性之间做出抉择，选出基于分组和基于代理的解决方案。

基于分组的设计是什么样的设计？

基于分组（或报文到报文）设计的网络设备，虽然位于通讯流的中间部分，但并非这些通讯的端点；此类设备仅负责传递数据包。对于基于报文到报文运行的设备而言，它通常对通过其间的协议一无所知，根本不是一个真正的协议端点。这些设备速度缓慢的主要原因是因为它们对于整个协议栈一无所知，缺乏用以处理流量的各种快捷方法。例如，就 TCP/IP 协议而言，此类设备对协议的理解仅限于改写 IP 地址和 TCP 端口；仅知道整个协议栈的一半内容。

由于网络变得日益复杂，对智能的需求日益增加，更加先进的基于分组的设计开始出现（包括 F5 Networks 公司生产的 BIG-IP 4.x 版产品）。此类设备对 TCP/IP 协议有着充分的了解，既了解 TCP 连接的安装与卸载，也了解如何修改 TCP/IP 标头，甚至还知道如何插入数据至 TCP 数据流当中。由于这些系统能够向 TCP 数据流插入数据，可修改数据流中的内容，因此，他们还必须对客户端和服务器间来回传递的数据包的 TCP 序列 (SEQ) 和确认 (ACK) 值进行重写。F5 Networks 公司的 BIG-IP 4.x 版解决方案可充分理解 TCP/IP 和 HTTP 协议，从而可确认单个 HTTP 请求，能够发送不同请求至不同的服务器，并复用已经开启的 BIG-IP 设备连接。

采用一个非常复杂的报文到报文架构（BIG-IP 4.x 版是迄今为止上述方案中最全面的一款产品），所有上述功能均可实现。

尽管复杂性越来越强，但基于分组的设计仍然还不是十分复杂，并且快于传统的基于代理的方案，因为它们具有仅需一小部分逻辑用于全代理的优势。

基于代理的设计（全代理）是什么样的设计？

全代理设计是与报文到报文设计相对的一种设计。与对通过设备的通讯流一无所知不同，全代理完全了解这些协议，其自身构成该协议的一个端点，也是该协议的创作者之一。客户端与全代理之间的连接完全独立于全代理与服务器间的连接，然而在报文到报文设计中，实质上在客户端和服务器端之间存在一条直接通信信道（尽管位于其中的设备可能会充分利用来回传递的数据包）。

由于全代理是一种实际上的协议端点，它必须在客户端和服务器端进行充分部署（基于分组的设计却并非如此）。这也意味着全代理能够有其自己的 TCP 连接行为，比如缓冲、转发，以及 TCP 选项。借助全代理，每个连接均为独一无二的；每个连接都有其自己的 TCP 连接行为。这意味着与全代理设备连接的客户可能会存在不同的连接行为，而全代理可能用其与后端服务器进行通讯。因此，不论最初的源地址还是最终目标地址如何，全代理解决方案均支持单独对每个连接进行优化。此外，全代理能够理解并处理每种协议，正像真实的客户端或服务器使用层进行处理那样。以 HTTP 的使用为例，首先对 IP 协议进行处理，之后为 TCP，然后是 HTTP；每个层均对较低层一无所知。

全代理设备具备诸多优势，能够更加轻松地支持应用级协议，能够完全“讲述”协议，在主动优化这些协议时，他们具有更大的灵活性（这一点与报文到报文设计不同，后者只能被动地支持一些协议）。此外，全代理能够更加轻松地提供各项先进功能，用以检查并充分利用各种协议及其数据。尽管由于充分支持应用协议造成额外开销，以及由于必须理解每个客户端和服务器的复杂性增加，但是，由于能够与各种协议交互，全代理设备具有更加广泛的能力，这是因为，与报文到报文设计相比，其可用信息更为翔实丰富。

挑战

对解决方案进行全新设计

众所周知，基于代理的解决方案，或者至少是此类解决方案所提供的智能模块，将是最终的解决方案。然而，报文到报文方案所具有的大量优异性能，弥补了它们在智能特性上的不足。对于多数企业网络而言，这种方案暂时可视为一种可接受的折衷方案。

随着对智能特性的需求日益增加，基于代理的解决方案始终面临的性能瓶颈问题，如今基于分组的解决方案很快也将面临同样的挑战。而基于分组的解决方案在开发上的复杂性，很快将与基于代理的解决方案并无二致。尽管软硬件性能正显著增长，然而报文到报文方案仍无法跟上用户对智能特性及性能的需求。此时，报文到报文方案便不再是一种可接受的可选方案。

基于分组的解决方案曾经红极一时，如今却已辉煌不再。如今，在提升性能不足以提供可靠解决方案的情况下，基于分组的解决方案在智能性方面的缺陷已使其今非昔比。比较现实的解决方案是，提供一款性能堪与基于分组的解决方案相媲美的基于代理的解决方案：而这就是 TMOS 架构解决方案。

解决方案

TMOS 为一个集合词，用于描述一种完全为特定目的建造的定制架构，该架构耗费 F5 多年经验与巨资，是 F5 产品向前发展的基石。从较高层次来看，TMOS 是：

☆ 多个模块的集合

每个模块执行一项特定的功能。例如：网络驱动模块、以太网模块、ARP 模块、IP 模块，以及 TCP 模块，等等。系统每个组件均为独立组件，用以降低系统的复杂性、减轻未来开发的负担。添加对新协议的支持，仅需添加一个新的模块即可。借助这一设计，用户能更轻松地实现重用。如果新的应用级协议在 TCP/IP 之上运行，仅需将模块连接，这样，新的协议就能访问 TCP/IP 中的数据，而无须较为低级的协议。

☆ 设备的独立特性

许多人已经注意到：基于 TMOS 的设备似乎采用 Linux 在其上运行，当我们使用命令行对设备进行管理时，可以看到这一点。但须知，采用 TMOS 对流量进行管理的任何一个部分，均未采用 Linux 系统。TMOS 有其自身专用的 CPU、内存，以及系统总线来访问外围设备。当基于 TMOS 的设备收到数据包时，无论是线路、系统总线、联网子系统，还是内存管理子系统，TMOS 架构中的每个组件均为完全独立的系统。Linux 永远不会纳入到该系统当中，或其中任何一个部分，Linux 内核也同样。Linux 系统仅用于对任务进行管理，如命令行、web GUI 等。原因很简单：一个适用于对高速流量进行管理的操作系统，并不适合作为一个通用型操作系统。因此，最好采用通用操作系统解决通用型任务（如管理任务），而将流量管理任务交给设计用于完成该任务的专用操作系统——TMOS来完成。

☆ 实时操作系统

实时操作系统意味着：TMOS 中并没有一个抢占式 CPU 调度程序。对于通用操作系统而言，在其内核中设定一段抢占式调度程序是非常有价值的，因为所有的进程能够公平地获得一个 CPU 时间段，而在这些进程当中，许多进程本质上可同时运行。在一个经过高度优化、专用的操作系统当中（如 TMOS），上述调度程序就不太适合，因为该调度程序增加了一段不必要的开销。由于 TMOS 经过精心设计与调整，因此，系统中的每个组件均用于执行必要的操作，以使下一个组件正常运行。由于去掉了中断、上下文切换，以及调度程序通常所需完成的大部分任务，因此，CPU 调度开销明显下降。这样，用户就能对进程（何时执行/以哪种排列方式执行）进行充分的控制。

☆ 硬件和软件

由于 TMOS 本质上是模块化的，因此，它并不介意单个功能是由软件还是由硬件来实现。有了 TMOS，每一步操作就能通过高度优化的软件，以及专为特定目的构建的模块来完成；当然，资源密集型操作也可卸载至专用的硬件。例如，TMOS 有其自己的 SSL 栈，完全能够用软件来处理 SSL，但是，如果卸载密码操作至专用的 SSL ASIC，则速度会更快。F5 软件技术具有的全部功能特性，使其实质上具有无限的灵活性，而专门的卸载硬件功能，又使其能够经济高效地扩充至业内领先的性能级别。

☆ 事件驱动

模块化与实时处理功能相结合，使 TMOS 具有独一无二的根据实时变化来改变其行为的能力。从客户连接开始、到有效负

载处理完成，其中的每个事件——甚至包括从服务器返回至客户端的流量，均会影响 TMOS 改变其行为，以便与当前的要求相匹配。这一功能使 TMOS 成为当前市场上最容易进行修改、最灵活的解决方案。

所有这些因素使 TMOS 成为一款功能非常强大、适用性极强的解决方案。独立、实时、事件驱动型操作系统，以及模块化功能，使 TMOS 具有无可比拟的能力。例如，TMOS 支持利用三个完全独特的与部署要求相匹配的网络栈。首先，FastL4 栈是一种限定状态的 TCP/UDP，或是一种传统的报文到报文设计，它可处理任意高速连接，但功能存在部分限制（第 4 层或第 4 层以下）。其次，TMOS 还支持 FastHTTP 栈；这种限定状态的 TCP/HTTP 栈是一种非常高级的报文到报文设计，它能以较高的连接速率处理 HTTP（第 7 层）不断增加的智能需求。最后，快速应用代理栈；该栈是一款基于全代理的缺省栈，代表了基于代理的设计方案的最高水平。TMOS 支持使用软硬件部件进行互换之事实，使得整个 FastL4 栈可完全卸载至 F5 的 Packet Velocity ASIC (PVA) 中。选择哪套方案作为最佳方案完全取决于客户的要求。

就其自身而言，这些高级说明中的任何一项，均会使 TMOS 从所有现有解决方案、报文到报文方案，或基于代理的方案当中脱颖而出。单就该架构本身，将足以改变应用交付网络市场，但是，即使是最好的架构，也需要次标准组件的支持，这样才能有助于次标准解决方案的产生。TMOS 与这些组件，或模块相结合，才能为用户提供真正优秀的解决方案。

携手迈向新的阶段

使 TMOS 架构与众不同的因素在于：TMOS 创建的定制构建模块为其自身提供了支持。在这些明显特性当中，最明显的一些最广泛应用的特性包括：TCP Express、快速应用代理，以及 iRule。

TCP Express

报文到报文的简单设计无法像具有 TCP Express 特性集的 TMOS 那样提供众多功能。许多企业部署了全代理方案，但是却受限如下之事实：仅包含一个构建于通用 UNIX 系统上的组件。这些企业无法获得以订制编码交付、提供较低延迟和一流网络性能的实时操作系统的优异性能。产品一旦设计出来以后，没有一套简单的办法来高效地整合这种功能类型——这就要求底层架构应具有高性能、低延迟的联网特性。

TCP Express 是一套 TCP 效率提升集合，以因特网标准形式提供（RFC），成百上千个定制 F5 特性与功能，均基于我们在实际使用过程中的扩展体验获得。TMOS 支持所有的 TCP 效率提升模式，包括：

- ☆ 延迟和选择性确认 (Delayed and Selective Acknowledgements), (RFC 2018)
- ☆ 显式拥塞通知 ECN, (RFC 3168)
- ☆ 限定与快速转发, (RFC 3042 and RFC 2582)
- ☆ 缓慢启动与拥塞避免 (Slow Start with Congestion Avoidance), (RFC 2581)
- ☆ 适应性初始拥塞窗口 (Adaptive Initial Congestion Windows), (RFC 3390)
- ☆ 时间标记与窗口比例 (TimeStamps and Windows Scaling), (RFC 1323)
- ☆ TCP 缓慢启动, (RFC 3390)

☆ 带宽延迟控制, 以及更多内容 (Vegas, NewReno, ...)

这些特性以及众多其它特性, 是在每种连接情形之下尽力获得峰值性能的所有方法。与 TMOS 进行交互的每种设备, 其网络情形均不同。为了获得针对所有连接设备的峰值性能, TMOS 必须以智能方式针对每位用户、每种设备 (以及每种设备的所有连接) 的独特需求做出响应。但它不足以支持一个或两个高级 TCP 优化。也不能用它开发您自己的优化, 以及现场优化您的产品。在一个对延迟进行了优化的实时操作系统当中, 完整的 TCP 优化、昂贵的真实测试, 以及定制扩展, 均是获得理想性能之不可或缺因素。所有上述问题均获得解决, 就能取得明显的现实益处, 这正是 F5 Networks 公司针对 TMOS 架构所进行的努力。

快速应用代理

TMOS 的快速应用代理组件——全代理栈, 彰显了 TMOS 的另一项关键特性。由于 TMOS 具备上述出色特性, 因此自面世以后, 它就能提供比以往任何解决方案更多的加速与优化特性。通常, 检查或智能逻辑源于速度方面的成本, 自然, 每个连接就被赋予了更多工作, 从而系统可以对更少的连接进行处理。快速应用代理的独特之处在于, 通过透明地充分利用硬件功能, 辅之以定制高性能实时操作系统 (TMOS), 就能真正获得无可比拟的性能, 同时还可提供真实全代理设计之益处。此外, 快速应用代理还能为应用流提供必要的用于部署其它 TMOS 模块的主机, 包括:

☆ HTTP 压缩

☆ 多点存储缓存

☆ SSL 加速

☆ 快速缓存

☆ 内容池 (Content Spooling)

☆ 智能压缩

☆ QoS/ToS

☆ 第 7 层带宽调整

所有上述模块已经融为了一体。TMOS 架构支持快速应用代理提供智能与性能两种特性; 通过 TMOS 架构, 快速应用代理支持智能化地使用其它大量软硬件模块, 这将有助于提供更出色的性能。

iRules

iRules 是 TMOS 中最独特的一项能力。iRules 是由标准工具命令语言 (TCL) 创建的一种脚本语言, 并配有定制 F5 扩展, 借助 iRule, 用户就能创建可从 TMOS 事件触发的独特函数。然而, 这套规则本身却很容易创建和理解, 该模块能够将这些规则编译成字节码, 并通过定制编写的、经过高度优化的函数调用 (位于 TMOS 的核心) 来实际执行一些操作 (如: 读写 HTTP cookie)。这两项关键技术组合意味着, 简单易用的 TCL 规则就能成高性能的字节码, 由于这种程序本质上借助 TMOS 来完成实际检查与执行, 因此其速度非常快。

由于 TMOS 刚刚发布, 因此, F5 客户发现他们有数百种方式来充分利用 TMOS 中 iRule 引擎中的强大功能。例如:

☆ DNS 带宽限定。

☆ 部署 SMTP 代理以检查并发送单个消息。

☆ 对 HTTP cookie 进行重新排序, 以便更加轻松地解析后端系统。

☆ 采用 HTTP cookie 中存储的证书, 通过后端 RADIUS 服务器, 对用户连接进行认证。

☆ 部署简单 LDAP 解析器, 以检查 LDAP bind() 请求的参数。

☆ 有所选择地对后端服务器使用 SSL 再加密功能, 仅针对特定 HTTP URL 来进行。有所选择地基于 HTTP URI 要求 SSL 客户证书。

☆ 创建并插入定制会话 ID 值至 HTTP 请求, 服务器在响应时将回传, 之后检查响应并验证其是否与请求匹配。若不匹配, 将全部会话信息和最后 100 个请求记入日志。

☆ CORBA/IIOP 请求分路分解处理。

如同 F5 的产品一样, 通过其检查特性集来部署这类高度复杂的逻辑, 可谓绝无仅有、仅此一家。正是轻松易用的 iRule 技术, 通过充分利用 TMOS 架构无可比拟的灵活性、以及其基于事件的架构, 实现了上述功能。此外, 我们不能将这种属性直接归因于 TMOS, iRule 以及 iRule 开发产品, 已逐步扩充了其开发社区, 如今全球已超过 10,000 位会员 (devcentral.f5.com)。

当 TCP Express、快速应用代理, 以及 iRule 成为唯一三个 TMOS 中的独特与高度优化组件时, 它们为我们展示了这样一件事实, 即, TMOS 将不会停留在一个全新的先进架构之上, 它还将包括一些一流的组件, 从而使这一架构更为鲜活、更具生命力。

TMOS 不可能包办所有事项

正像您看到的那样, 报文到报文设计的开发源于需要提供应用交付网络装置, 它能够出色性能以及智能特性, 但与基于代理的解决方案相比, 智能性不足。此类设备满足用户网络需求的黄金时代已经一去不返, 多数厂商现在又回到了原点; 他们既要提供全代理解决方案的智能特性, 又要满足当今用户网络更加强健的性能需求。选择基于分组的架构, 已被证实是一种短视行为。

F5 Networks 公司很早就意识到, 解决方案只有基于代理架构进行设计, 才能既提供性能益处, 又具备出色的灵活性, 而在未来发展的道路上所向披靡、长期发展。由于上述原因, 通过全力开发并投入资金, TMOS 应运而生。TMOS 是首款完全基于特定目的构建的、模块化、独立式、事件驱动代理型实时架构, 能够透明地单方面利用软硬件, 获得最佳性能与智能特性。此外, TMOS 远非一套革命性的架构, 它还还为构建于该架构之上的所有组件制订了一套全新标准。从 TCP Express 推出伊始, 它就能确保以最高效的网络级连接速率连接至客户端和服务端; 到快速应用代理推出, 它能够同时提供性能与智能两项特性; 而到了 iRule 的推出之时, 已能对所有这些功能、所有的应用加速, 以及所有可用的安全性和可用性组件, 进行完全控制和定

TMOS: 对解决方案进行全新设计

制, TMOS 完全是您贴心的唯一应用交付网络解决方案。机遇千载难逢, 别再犹豫, TMOS 会是您的正确选择。