F5 White Paper

TMOS: Redefining the Solution

This paper addresses the F5 TMOS architecture, a collection of real-time features and functions, purpose-built and designed as a full-proxy solution with the power and performance required in today's network infrastructure.

by KJ (Ken) Salchow, Jr.
Manager, Technical Marketing

Contents

Executive Overview

Historically, there have been two ways to build Application Delivery Networking appliances—build them for performance or build them for intelligence. In the open market, customers have traditionally selected solutions that exhibit the best performance. As a result, most vendors have built their devices on faster, packet-based designs instead of the poorer performing proxy-based architecture. As the need for intelligence in these devices has grown, vendors find themselves in a precarious position: the more intelligence they add to the devices in response to customer demand, the closer they resemble a proxy and the worse they perform.

F5 initially took the packet-based path, but simultaneously started addressing the root problem—making an intelligent solution that also delivers high performance. The result is the F5 TMOS® architecture, a collection of real-time features and functions, purpose-built and designed as a full-proxy solution with the power and performance required in today's network infrastructure.

Packet-based vs. Proxy-based

To understand just how unique and powerful TMOS is, it is important to look closely at the history of these Application Delivery Networking devices, and the dilemma between speed and intelligence with packet-based and proxy-based solutions.

What is a packet-based design?

A network device with a packet-based (or packet-by-packet) design is located in the middle of a stream of communications, but is not an endpoint for those communications; it just passes the packets through. Often a device that operates on a packet-by-packet basis does have some knowledge of the protocols flowing through it, but is far from being a real protocol endpoint. The speed of these devices is primarily based on not having to understand the entire protocol stack, short-cutting the amount of work needed to handle traffic. For example, with TCP/IP, this type of device might only understand the protocols well enough to rewrite the IP addresses and TCP ports; only about half of the entire stack.

As networks became more complex and the need for intelligence increased, more advanced packet-based designs began to emerge (including the BIG-IP® products from F5). These devices knew TCP/IP well enough to understand both TCP connection setup and teardown, modify TCP/IP headers, and even insert data into

TCP streams. Because these systems could insert data into TCP streams and modify the content of the stream, they also had to rewrite the TCP sequence (SEQ) and acknowledgment (ACK) values for packets going back and forth from the client and server. F5's BIG-IP products understood TCP/IP and HTTP well enough to identify individual HTTP requests and could send different requests to different servers, reusing connections the BIG-IP device already had open.

While all of this is possible using a very sophisticated packet-by-packet architecture (BIG-IP devices are some of the most sophisticated of such designs to date), it required a very complex state tracking engine to understand the TCP/IP and HTTP protocols well enough to rewrite header contents, insert data, and maintain its own connections to clients and servers.

Despite this increasing complexity, packet-based designs are still less complex and faster than traditional proxy-based designs, as they have the advantage of only requiring a small percentage of the logic required for a full proxy.

What is a proxy-based design (full proxy)?

A full-proxy design is the opposite of a packet-by-packet design. Instead of having a minimal understanding of the communications streaming through the device, a full proxy completely understands the protocols, and is itself an endpoint and an originator for the protocols. The connection between a client and the full proxy is fully independent of the connection between the full proxy and the server; whereas in a packet-by-packet design, there is essentially a direct communication channel between the client and the server (although the device in the middle may manipulate the packets going back and forth).

Because the full proxy is an actual protocol endpoint, it must fully implement the protocols as both a client and a server (a packet-based design does not). This also means the full proxy can have its own TCP connection behavior, such as buffering, retransmits, and TCP options. With a full proxy, each connection is unique; each can have its own TCP connection behavior. This means that a client connecting to the full-proxy device would likely have different connection behavior than the full proxy might use for communicating with the backend servers. Therefore, a full proxy allows for the optimization of every connection uniquely, regardless of the original source and the final destination. Further, a full proxy understands and processes each protocol as a real client or server would, using layers. Using HTTP as an example, first the IP protocol is processed, then TCP, then HTTP; and each layer has no knowledge of the lower layers.

Redefining the Solution

It is common knowledge that proxy-based solutions, or at least the intelligence offered by them, were the ultimate solution. However, the vastly superior performance of packet-by-packet designs more than made up for their limited intelligence. For a while, this was an acceptable trade-off for most enterprise networks.

As the need for increased intelligence grows, packet-based solutions are quickly experiencing the same performance restrictions that proxy-based solutions have always suffered from. And the development complexity of packet-based solutions is quickly approaching that of proxy-based designs as well. Despite dramatic increases in hardware and software power, packet-by-packet designs are unable to keep up with the need for intelligence and performance. It is no longer acceptable to have to choose between them.

While packet-based solutions had their time, that time is gone. It is now readily apparent that short-cutting intelligence in lieu of performance did not adequately provide a viable solution. The real solution is to build a proxy-based solution with the performance of the packet-based solution.

TMOS

TMOS is a collective term used to describe the completely purpose-built, custom architecture which F5 spent years and significant investment developing as the foundation for F5 products going forward. From a high-level, TMOS is:

- **A Collection of Modules**
  Each module performs a particular function. For example, there is a networking driver module, an Ethernet module, an ARP module, an IP module, a TCP module, and so on. Every component of the system is self-contained, which helps reduce the complexity of the system and eases future development. Adding support for new protocols is simply a matter of adding a new module. This design also allows for easier reuse. If a new application-level protocol ran on top of TCP/IP, it would be a simple matter to link the modules together so that the new protocol had access to data from TCP/IP without understanding the lower-level protocols.

- **Self-Contained and Autonomous**
  Many people have observed that a TMOS-based device has a form of Linux running on it, which can be seen when administering the device using the command line. It is important to note that this Linux system is not involved in any aspect of the traffic flowing through TMOS. TMOS has its own dedicated CPU, memory, and system bus for access to peripheral devices. When a TMOS-based device receives packets, everything from the wire to the system bus, from the networking subsystem to the memory management subsystem are completely self-contained within TMOS. Linux is never involved or aware of any of it; not even the Linux kernel. The Linux system is used for management tasks, such as the command line or the web GUI only.
  The reason for this is simple: an operating system which is ideal for high speed traffic management operations is not ideal as a general purpose operating system. So it makes sense to use a general purpose operating system for general purpose tasks, like management, and leave the traffic management to the operating system designed for that purpose—TMOS.

- **A Real-Time Operating System**
  A real-time operating system means TMOS does not have a preemptive CPU scheduler. For general-purpose operating systems, having a preemptive scheduler in the kernel is very desirable, as all processes can get a fair share of CPU time, and many of them can run essentially at the same time. In a highly optimized, special-purpose operating system like TMOS, such a scheduler is not well-suited because it adds unnecessary overhead. TMOS was designed and fine-tuned so that each component in the system performs the necessary operations and then lets the next component run. This significantly reduces the overhead of CPU scheduling by eliminating interrupts, context switches, and most of the work a scheduler would typically do. This also allows full control over when, and in what order, processing occurs.

- **Both Hardware and Software**
  Because TMOS is inherently modular in design, it doesn't matter whether individual functions are handled by software or by hardware. With TMOS, everything can be done in software utilizing highly optimized and purpose-built modules; however resource-intensive operations can also be offloaded to specialized hardware. For example, TMOS has its own SSL stack and can process SSL entirely in software, but it is much faster to offload cryptographic operations to specialized SSL ASICs. Having fully-featured software technology allows for virtually infinite flexibility and having specialized offload hardware allows for cost-effective scaling to industry-leading levels of performance.

- **Event-Driven**
  The combination of modularity and real-time processing allows TMOS the unique ability to change its behavior based on real-time, real-world events. Every event, from client connection initiation through payload processing—even return traffic from the server back to the client—constitutes an opportunity for TMOS to change its behavior to match the current requirement. This functionality makes TMOS the most adaptable and flexible solution available.
- **Stateful inspection**
  The core architecture of TMOS is based on a high speed full proxy which performs stateful inspection to traffic flows as does a stateful firewall. While connection traffic is proxied through TMOS, the F5 iRules® engine has full access (layer 2-7) to that traffic and can set various rules (based on traffic characteristics, ACLs, or business logic). As an example, rule actions can be Drop, Block, Redirect, Log, or Transform.
- **Dynamic Packet Filtering**
  By default, just like a firewall, TMOS has a "deny all" policy. One important security advantage of TMOS is that its full proxy completely shadows the network stack of backend services. The F5 iRules in TMOS allow dynamic control of application traffic flow such as dynamic control of traffic shaping policies based on application (not just TCP or network protocol), request, or response content. Traffic flow can also be filtered, redirected, or blocked dynamically at the same application layer such as HTTP, or packet by packet, such as via UDP, SCP, or TCP.

All of these items make TMOS an extremely powerful and adaptable solution. The combination of modularity in a self-contained, real-time, event-driven operating system gives TMOS unprecedented capabilities. For instance, TMOS is capable of utilizing three entirely unique network stacks designed to match deployment requirements. First, there is the FastL4 stack; a limited-state TCP/UDP or traditional packet-by-packet design to handle any high-connection rate, but limited functionality (layer 4 and below) requirements. Then, there is the FastHTTP stack; a limited-state TCP/HTTP stack representing an extremely advanced packet-by-packet design to handle high-connection rates with increased HTTP (layer 7) intelligence requirements. Finally, there is the Fast Application Proxy; the default full-proxy based stack representing the pinnacle of proxy-based design. The fact that TMOS is capable of using software and hardware components interchangeably allows the entire FastL4 stack to be completely offloaded to F5's Packet Velocity ASIC

(PVA) if the system is equipped with the PVA hardware. The choice of which approach is best is entirely up to the customer.

By itself, any one of these high-level descriptors would differentiate TMOS from any existing solution, packet-by-packet or proxy-based. This architecture alone would be enough to change the application delivery networking market, but even the best architecture built with sub-standard components results in a sub-standard solution. The components, or modules, with which TMOS is built makes it a truly superior solution.

Taking it to the Next Level

What makes the TMOS architecture extraordinary are the custom-built modules created to support it. Among the most significant are some of the most widely used: TCP Express,™ Fast Application Proxy,™ and iRules.

TCP Express

Packet-by-packet designs simply cannot provide what TMOS can with the TCP Express feature-set. There are companies that have a full proxy, but it is limited by the fact that it is merely a component on a general-purpose UNIX-based system. They have no possibility of achieving the performance of a custom-written, real-time operating system that was designed to provide low-latency, state of the art networking performance. There is simply no way to effectively bolt-on this type of functionality after the product is designed—it requires a fundamental architectural commitment to high performance, low-latency networking.

TCP Express is a collection of TCP efficiency improvements, in the form of internet standards (RFCs), and hundreds of custom F5 features and tuning based on our extensive real-world experience. TMOS supports every modern TCP efficiency improvement, including:

- Delayed and Selective Acknowledgements, (RFC 2018)
- Explicit Congestion Notification ECN, (RFC 3168)
- Limited and Fast Retransmits, (RFC 3042 and 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 many more (Vegas, NewReno, etc.)

These features, and over a hundred others, are all ways of trying to achieve peak performance in every connection scenario. Every device TMOS interacts with has a different network scenario. To get peak performance for all connecting devices, TMOS has to intelligently react to the unique requirements of each and every device, for each and every connection. It is not enough to support one or two advanced TCP optimizations. It is not enough to develop your own optimizations, and to field-optimize your product. Complete TCP optimization, custom extensions, and extensive real-world testing, are all needed together, on top of a real-time, latency-optimized operating system to achieve ideal performance. All of these things must be done together to achieve significant real-world benefit, which is what F5 has done with TMOS.

Fast Application Proxy

The Fast Application Proxy component of TMOS, the full-proxy stack, is another key differentiator. This is what enables TMOS to provide more acceleration and optimization features than any solution before or since. Typically, inspection or intelligence logic comes at the cost of speed, and naturally so—the more work that is required per connection, the fewer connections that can be handled. What makes the Fast Application Proxy so unique is that by leveraging hardware transparently when possible, coupled with a custom high-performance real-time operating system (TMOS), it is able to achieve truly unparalleled performance, while still offering all the benefits of a real full-proxy design. Additionally, the Fast Application Proxy provides for the application fluency necessary to implement a host of other TMOS modules, including:

- HTTP Compression
- Multi-Store Caching
- SSL Acceleration
- Fast Cache
- Content Spooling
- Intelligent Compression
- QoS/ToS
- L7 Rate Shaping

The pieces all fit together. TMOS provides an architecture that allows the Fast Application Proxy to provide intelligence with performance; and the Fast Application Proxy, using the TMOS architecture, allows for the intelligent use of numerous other modules, both hardware and software, which provide even greater performance.

iRules

iRules are one of the most unique capabilities of TMOS. iRules are scripts created using the standard Tool Command Language (TCL) with custom F5 extensions that enable users to create unique functions triggered from TMOS events. While the rules themselves are easy to create and understand, this module compiles the rules into byte-code which actually performs actions, such as reading and writing HTTP cookies, through custom-written, highly optimized function calls in the core of TMOS. The combination of these two key technologies means that the simple TCL rule text results in high performance byte-code that does the actual inspection and manipulation in a way that is native to TMOS, and therefore fast.

Since the initial launch of TMOS, F5 customers have found hundreds of ways of taking advantage of the power of the iRules engine in TMOS. For example:

- DNS rate limiting.
- Implementing an SMTP proxy to inspect and direct individual messages.
- Re-ordering HTTP cookies for easier parsing on the back-end systems.
- Authenticating user connections with a back-end RADIUS server using credentials stored in a HTTP cookie.
- Implementing a simple LDAP parser to inspect parameters of LDAP bind() requests.
- Selectively using SSL re-encryption to the back-end servers only for certain HTTP URLs. And selectively requiring SSL client certificates based on HTTP URLs.
- Creating and inserting a custom session ID value into an HTTP request, which the server will echo back in the response, then inspect the response and verify it matches the request. If not, logging the full session information, and the last 100 requests.
- CORBA/IIOP request multiplexing.

There is no other device in the world that can implement this type of highly sophisticated logic via their inspection feature-set. These are simple iRules that take advantage of the unparalleled flexibility of TMOS and its event-driven architecture. Furthermore, while we can't attribute it directly to TMOS, iRules, and the development of iRules has spawned its own development community which exceeds 7,000 members worldwide: DevCentral.

While TCP Express, the Fast Application Proxy, and iRules are only three of the many unique and highly optimized components built on top of and into TMOS, they present a wide view into the fact that TMOS does not stop at a groundbreaking architecture, but also includes the leading-edge components to make that architecture come to life.

The development of packet-by-packet designs originated out of a need to provide application delivery networking appliances which provided exceptional performance as well as intelligence, albeit less intelligence than their proxy-based counterparts. The ability of these devices to satisfy the need of today's network has run its course and most vendors are right back were they started; needing to provide the intelligence of a full-proxy with the even more increased performance requirements of today's networks. The choice of a packet-based architecture has proven short-sighted.

F5 realized early on that the only long-term solution was a proxy-based architecture designed to provide both performance and unparalleled flexibility to handle anything that came down the road. The result of that vision, as well as significant development time and investment, is TMOS. TMOS is the first, completely purpose-built, modular, self-contained, real-time, event-driven proxy architecture with the ability to transparently utilize hardware and software unilaterally to achieve the best performance and intelligence. In addition, TMOS goes beyond a revolutionary architecture and provides a new standard for each and every component used to build it. From TCP Express, that ensures the most efficient network-level connections to both the client and the server, to the Fast Application Proxy that provides intelligence as well as performance, to iRules that allow for the complete control and customization of all these functions and all of the numerous application acceleration, security, and availability components available, TMOS is simply the only Application Delivery Networking solution that isn't starting all over again. TMOS is already the right solution.

F5 Networks, Inc. 401 Elliott Avenue West, Seattle, WA 98119 888-882-4447 www.f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com