It&rsquo;s like load balancing. On steroids.

Lori MacVittie, 2009-20-04

What is this application delivery thing that everyone keeps telling me I need? Isn't that just the latest marketing term for load balancing?

A recently released Forrester report concludes that "firms must develop and integrated strategy for application delivery." We don't disagree with that, or with the Gartner report claiming that "Load Balancing is Dead, Time to Focus on Application Delivery." Application delivery is the next step in the logical evolutionary path from the tactical solution of load balancing to a comprehensive application infrastructure strategy.

Forrester's research indicates that despite the fact that application delivery makes sense, many organizations are still operating in a very tactical, problem-resolution oriented manner.

> **Application Delivery Takes Center Stage**
>
> Top infrastructure initiatives — like consolidation and virtualization — are focused within the data center, and firms aren't paying enough attention to solving the growing need to provide anywhere, anytime access to applications. The result? Application response times don't meet expectations. The knee-jerk usual reactions are to increase network bandwidth and to deploy point solutions like WAN optimization, but these measures do not address the underlying problems. Our conclusion: To deliver acceptable application performance levels without unacceptable increases in IT costs, firms must develop an integrated strategy for application delivery.
>
> …
>
> Despite the increased focus on the network, we still don't see a lot of companies taking advantage of more purpose-built solutions that tackle application performance, availability, and scalability. An increasing number of firms are throwing hardware point solutions at the problem, as demonstrated by the 41% reporting that they are deploying such equipment as load balancers. However, we were a bit surprised to see a lower emphasis on more comprehensive solutions, with 33% and 20% indicating they are taking a more strategic approach by implementing application delivery infrastructure and application acceleration equipment, respectively.

Some of the reason for the lack of adoption of more integrated solutions is likely that organizations are simply not aware of what application delivery is. Some of the reason is certainly that there still exist silos within IT that focus on the many functions of application delivery but do so in a non-integrated fashion themselves. Some of the reason is simply that IT is overburdened at the moment; and has very little time for strategy when it is tasked with solving real problems right now.

Ironic, then, that IT doesn't have time to focus on the very strategy that could reduce the burden of siloed application delivery management and thus give IT the time they need in the first place. A Catch-22, to be certain.

WHAT IS APPLICATION DELIVERY

Analysts, press, industry pundits. All three agree that application delivery is an essential component to the efficient data center of tomorrow. But they – and I'm guilty of this too - often assume you know what application delivery is, and what it does, and why it's so necessary as part of solid foundation for emerging data center models.

The question "What is it?" is far more common than you might think. The term is one that almost always requires defining unless you've been knee-deep in the industry for a while. If I say "load balancer" to a crowd the term is immediately understood. But if I say "application delivery" the audience gets that "are-you-speaking-in-a-foreign-language-because-I-don't-know-what-the-hell-you're-talking-about" look on their face. You know, the one that makes you wonder if you just brayed like a donkey or maybe your latent Tourette's Syndrome just kicked in.

That's why I often describe it with "it's like load balancing on steroids." Mostly because application delivery grew out of load balancing because it just made sense.

Application delivery is what you do with applications. You deliver them, via some kind of network, to users. Application delivery infrastructure, then, is all the components necessary to make that happen.

Load balancing is the core of application delivery. This is because the load balancer just happened to be deployed in the perfect place in the data center to provide additional, application-focused functionality: between the client and the server. Because it usually acted as a proxy, it was able to grow from simple layer 4 (TCP) load balancing to a more flexible, intelligent and application-aware layer 7 (application) device. As it did so, developers began to see the potential benefits of adding functionality to the load balancers. Because the device could see everything from the network layer to the application data, it could optimize network and communication protocols, add security options, implement rate shaping and other QoS functionality, and be more "smart" regarding the definition of "availability" when it came to the application. And thus application delivery solutions began to appear, each one comprising more and more application-aware functionality; each one capable of providing more and more benefits.

And as applications grew more complex, so did the infrastructure. There's performance and access considerations. Reliability, scalability, and security concerns. Failover, optimization, and application-specific quirks that must be addressed in a load balanced environment. There are a lot of components required to deliver an application, keep it secure, and make sure it's fast enough to keep the user happy.

The Forrester report discusses the need to "provide anywhere, anytime access to applications." That means from home, on the road, in the office, in the wee-hours of the morning or during the middle of the day. Application delivery also concerns itself with being able to handle the myriad other factors that go into application delivery such as SLAs based on application ***and*** user ***and*** network conditions.

Application delivery is about making decisions based on the context of each request, rather than on one or two variables. And not just decisions like which server should respond, but which network should it be returned on and which data center should be used and should the request be scanned for malicious intent and is this request even a legitimate one for this application, for this user, from this location. It's about applying optimizations to protocols that improve the performance of applications over both WAN and LAN. It's focused on ensuring availability of applications and maintaining service-level agreements through intelligent load balancing decisions and careful monitoring of application health at the application layer, not just the network layer. Application delivery focuses on the ***application*** and on its unique quirks and behaviors that can impede performance. It provides a platform on which on-demand adjustments can be made to the application delivery process; on which functionality can be deployed to address security or architectural issues in a centralized manner.

Application delivery is the integration of solutions focused on the security, performance, and reliability of applications.

This graphic from the aforementioned Forrester report very nicely illustrates the difference between a point-product based solution and an integrated application delivery architecture.

Figure 10a: A Multivendor Application Delivery Solution

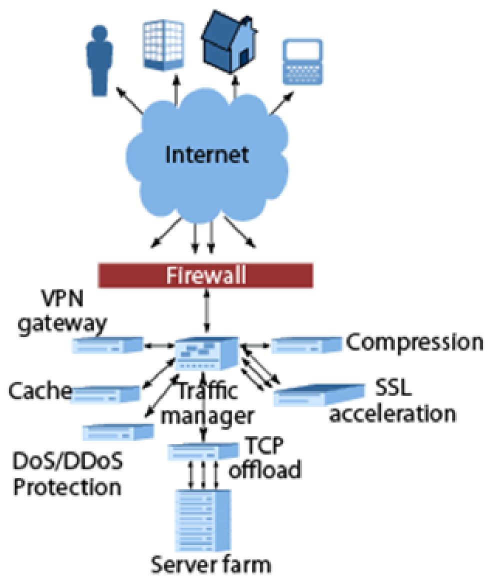
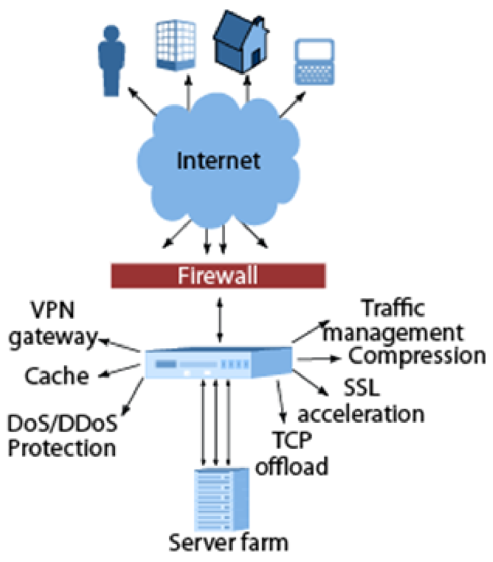


Figure 10b: A Single Vendor Application Delivery Architecture



Source: "Application Delivery Takes Center Stage," a commissioned study conducted by Forrester Consulting on behalf of Citrix Systems, December 2008

WHY APPLICATION DELIVERY

Hardware point solutions can result in sprawl. Sprawl increases operating expenses, makes it difficult to troubleshoot, introduces unnecessary complexity, and as a bonus it negatively impacts application performance – the very thing the solutions were put in place to address - by adding latency at every hop. I've explained the problem of sprawl and the proliferation of point solutions to many different types of audiences and not once has someone yelled out, "You're a liar! Does not!" because everyone knows it's true; we just may not agree on the best way to solve *that* problem.

Obviously if you integrate all the functionality normally found in point solutions so that they all work on the same data set, it's going to remove the issue of latency because all the solutions can work on the same data without needing it packaged up and delivered via a fairly expensive TCP connection.

That used to be "the big" problem application delivery solved. Today the focus is also on streamlining application delivery *processes:* the manual configuration and coordination of policies across disparate solutions designed to secure and speed the delivery of applications. That process, when using multiple point solutions, can become a nightmare. It's not just the configuration of each individual device that's the problem, it's the *coordination* across all those solutions that becomes problematic and time consuming. Policies implemented and enforced on one point solution may interfere with the application of a policy on another device, conflicting with one another even though both are equally valid and equally necessary. Resolving those conflicts takes time and can actually require a re-architecting of the network. For example, *where* in the data flow you place certain solutions such as security can change how the policies act. If an intermediary acting as a full proxy changes, in any way, the application data or headers it can trigger false positives on security devices inspecting traffic behind it.

Encrypted data has to be decrypted to be inspected by security and content filtering solutions, so it's essential to ensure that those solutions are in the flow in the right place in the network. Or you have to provide them with the proper certificates so they can decrypt the data, which means more management and tracking of certificates. This also introduces the potential for certificate theft as few devices have secure key stores and cert management. That's assuming you *can* store the cert on an intermediary device; some provide no mechanism for doing so.

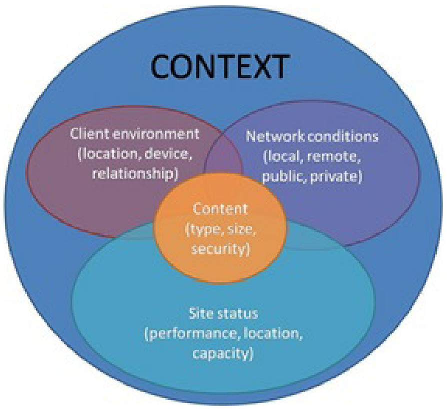
These problems are solved with an integrated application delivery solution because the policies are designed to collaborate with one another; to work in concert with each other rather than conflicting with one another. They have access to each other's data if necessary and understand their relationship with one another. And when there is still a conflict – and there invariably is for some situations – then the answer is to reorder policies, not *re-architect* the network. The former is a much simpler solution that requires less time and fewer headaches.

An integrated solution also ensures the reuse of knowledge. If you know how to configure the application acceleration components you also know how to configure the application security, and the core load balancing features. The interfaces are the ***same*** and the processes (and terminology) are the same, which means less time spent learning the nuances of each product and becoming familiar with each product's unique view of how the product should be configured and managed. This streamlines the application delivery process and makes it more efficient, which translates into reduced operating expenses.

THE PROBLEMS OF ACCURACY and CONTEXT

In an architecture comprised of multiple solutions, the only real way to share that context is to pass it around somehow between devices. The only exception to this is in the case of some security solutions that can be deployed in a bridged mode. IDS and web application firewalls are the most common example, where the solutions are deployed in such a way that the original requests are essentially broadcast to the devices, usually through the use of mirroring on the switch. This solution does solve the problem, but it also results in duplication of data on the network and increases the bandwidth used in the process.

The passing around of context between devices doesn't happen for a number of reasons. Foremost is the lack of a communication protocol to do so. There is no "context-sharing" standard, no best practices, no agreed upon method of sharing that context between disparate devices. While there may be a way to do so among products from a ***single*** vendor, anyone who builds an application delivery network based on individual components rarely sources from a single vendor, so any "sharing" of context that is possible is generally lost.


CONTEXT
Client environment (location, device, relationship)
Network conditions (local, remote, public, private)
Content (type, size, security)
Site status (performance, location, capacity)

The other issue with a multiple-solution architecture is that many solutions are full proxies. That means that it is not the user that appears to be the client, it is the last intermediary in the chain of proxies that appears to be the client. If the flow of data is *client* –> *SSL accelerator* –> *load balancer* –> *server* then the load balancer sees the SSL accelerator as the client, and not the end-user. That means data regarding the network conditions for the client are not accurate. The load balancer sees the local segment of the network as the "client link" and any decisions made based on that will be based on incorrect data.

This problem is particularly prevalent in Web 2.0 applications which provide APIs for integration. Requests via the API have different requirements; they are treated differently than requests for the same data arriving via the web application itself. Without an intelligent infrastructure, the handling of these requests is spread across multiple pieces of infrastructure – and often in the application itself. A change in policy requires changes across multiple devices, which can not be only be time-consuming but is prone to error introduction based on the sheer volume of changes required.

In an integrated application delivery network the myriad functions are integrated and deployed on the same platform. This means that what one solution (e.g. security) does to data is understand and recognized by other solutions (e.g. caching and application acceleration). The ***context*** is preserved as requests and responses flow through the disparate functions. It solves the second issue – accurate data upon which to make decisions – by having access to the original request, from the network layer up to the application layer.

START SIMPLE, GROW LATER

Most organizations necessarily turn to application delivery solutions because they are in need of a high availability architecture; they need a load balancer. As scalability through virtualization (horizontal scalability) continues to rise in popularity as a more efficient means of achieving goals, load balancing will continue to be a more strategic part of the data center. It behooves network and system architects, then, to consider the long-term ramifications associated with virtualization and increasing demand on applications in terms of access, performance, and security. Doing so should, according to analysts, lead those architects to determine that an application delivery networking solution will serve their needs best as it is these very issues that are addressed by such platforms.

Choosing a modularized, extensible application delivery platform allows architects to start with load balancing and add additional functionality as they need and in such a way as to allow them to truly design a solution that fits their specific needs rather than simply acquire and deploy more devices that may dictate changes in the network and infrastructure architecture.

Related articles & blogs

- Load Balancing is Dead, Time to Focus on Application Delivery
- Architects Need to Better Leverage Virtualization
- What's good for the network isn't always good for the application
- Why you still need Layer 7 persistence
- Layer 7 switching + Load balancing = Layer 7 Load Balancing
- The Web 2.0 Botnet: Twisting Twitter and Automated Collaboration
- API Request Throttling: A Better Option

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | 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