You say "Late", I say "Delayed"

Lori MacVittie, 2008-04-03

The language of application delivery and SOA finally meets in the middle

Ronald Schmelzer at ZapThink wrote a recent ZapFlash titled, "Why Service Consumers and Service Providers Should Never Directly Communicate." Yes, I agree, the title is way too long, but you should read it anyway. The basic premise of this one is that there is a need for a service proxy to protect your investment in SOA. I'll save my kudos and comments for another post, but in general I agree with Ron and his vision of SOA and the need for a proxy/intermediary to prevent the loss of loose-coupling and destroy the reusability of services.

What I really got excited about in this ZapFlash was the use of the term "late-binding". That's because the terminology - and the technology it describes - are so very similar to what the application delivery world calls "delayed-binding".

> Therefore, the optimal solution here is to use registry-based “late-binding” to help the Service consumers find proxies and also to help them resolve Service providers to WS-Addresses.

Finally there's some terminology that not only bridges the gap between the network and the application but that is actually close enough that both camps should easily remember and recognize it. While many concepts in application delivery and applications are the same, the terms we use are often so very different - affinity vs persistence, quiesce vs bleed - that there is little chance anyone from the other side of the switch is likely to understand what we mean. But in this case, the terms are so similar that they are likely to be universally understood; if not in practice, at least in theory.

Late and Delayed Binding

Late-binding and delayed binding mean essentially the same thing: a client is not intially paired with an endpoint (service, server). There is some magic process that must occur in order to determine ***where*** the request should be ultimately be directed. In late-binding that discovery is left to the client, with delayed binding it is up to the intermediary/proxy.

In the application delivery world, delayed binding was achieved when Layer 7 routing - a.k.a. content switching, a.k.a. content based routing - was implemented. A client would make a request to a VIP (Virtual IP) and the proxy/intermediary (a.k.a. application delivery controller) examines HTTP headers or application data in order to determine the appropriate endpoint to which it should direct the request. It's called "delayed" because the decision regarding the endpoint isn't made until ***after*** the request is made.

> 1. Client sends a request to the application delivery controller
> 2. The application delivery controller examines the request and determines the endpoint (server or pool)
> 3. The application delivery controller routes the request to the endpoint (server or pool)

In the application world, late binding is achieved through the use of WSDL (Web Services Description Language). A client doesn't know which endpoint it should send to until ***after*** it requests - and receives - a WSDL. The client then parses the WSDL, finds the proper endpoint, and sends its request to that endpoint for processing.

> 1. Client requests the appropriate WSDL document, optimally from a registry
> 2. Client determines the endpoint by parsing the WSDL
> 3. Client routes the request to the endpoint

Both late and delayed binding act in essentially the same way - they simply use different information to make their decisions.

"Delate" Binding

It should be fairly simple to see how a service proxy/intermediary fits into the picture by marrying late and delayed binding:

1. The client sends a request to the intermediary/proxy (to a Virtual IP).
2. The intermediary/proxy determines the endpoint by consulting the appropriate WSDL document
3. The intermediary/proxy routes the request to the appropriate endpoint

Basically, the two concepts are exactly alike in theory, and in practice differ only on the data used to determine the appropriate endpoint.

This is exciting, because one of the early premises of SOA was that a common language between business owner and architect would ameliorate the potential problems inherent in such an expensive and lengthy endeavor. As SOA has moved from pilot to full implementation, the need for a common language between network and application architect has become evident, and for the first time it appears that perhaps we've got some momentum in that direction.

The scalability and available of services in your SOA likely depend on some form of application delivery controller, but implementation requires cooperation between application and network architects. That cooperation can be made easier and more palatable if there's some common language that can be used to communicate.

Imbibing: Coffee

Technorati tags: MacVittie, F5, application delivery controller, application delivery, SOA, services, proxy

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