The Network's **NEW** Role

**Application-oriented
networks can help
bridge the gap
between enterprises.**

TAF ANTHIAS and KRISHNA SANKAR, CISCO SYSTEMS

Companies have always been challenged with integrating systems across organizational boundaries. With the advent of Internet-native systems, this integration has become essential for modern organizations, but it has also become more and more complex, especially as next-generation business systems depend on agile, flexible, interoperable, reliable, and secure cross-enterprise systems.

This article describes the various demanding scenarios in the cross-enterprise domain and offers perspectives in addressing these challenges. We look at the trajectory and locus of cross-enterprise systems, the many ways in which the various complexities are

The Network's NEW Role

addressed now, and how they can be simplified in the future. It is in this context that the network emerges as one of the alternatives, acting as an intermediary for cross-enterprise integration of federated business services, with an application orientation.

APPLICATION ORIENTATION— A NEW ROLE FOR THE NETWORK

Demands from and requirements for service-oriented application networks are changing the design, deployment, and operation of networks. Some crosscutting requirements of service-oriented applications are best housed in the network. These include service virtualization, routing based on message semantics, policy intermediation, and security.

An AON (application-oriented network), in its basic form, is either a blade in a network device (such as a router or a switch) or a dedicated appliance that serves as the front end for a network or server farm. The two main functions of an AON device are acceleration of generic application functions (such as XML processing, transformation, and digital signature verification) and control path functions (such as policy intermediation, routing, and service virtualization).

The AON takes advantage of its tactical footprint while maintaining the characteristics that make the network fabric ubiquitous and pervasive. Figure 1 shows the conceptual model of an AON.

Whereas pure transport networks are bit pipes acting as packet haulers, an AON understands not only the syntax of networks but also the semantics of applications by straddling the lower application layers and the higher network layers. Figure 2 shows the functional reach of an application-oriented network. The network in this context plays the role of an integrating service intermediary (transparent or otherwise) functioning as a protocol

translator, a policy mediator, a resolver, and a security appliance. Conceptually, in this new role as an application-aware network, an intermediary helps separate domain complexity from accidental complexity—the application layer implements domain complexity while the network handles the complexity arising from wiring them together and providing a flexible coupling—in both the real and virtual sense.

Another aspect of an AON is its role as an orthogonally extensible platform. The AON platform facilitates high-level programmability through bladelets (which are collections of logic to be applied to a message or group of messages), information models, and similar mechanisms.

SERVICE VIRTUALIZATION BASED ON AN AON

To understand the choreography, topology, and sequence of events in the AON world, let's look at service virtualization and the challenges in implementing this paradigm based on an AON substrate.

Virtualization is done at different levels, namely server virtualization (mechanisms such as InfiniBand and VFrame), process virtualization (through hypervisors and Xen), and container virtualization (through app servers). The AON adds a service virtualization layer to this mix.

Conceptual Model of Application-Oriented Network

FIG 1

Service virtualization is key for an agile, flexible, scalable, inter-organization service fabric.

To expand cross-enterprise systems faster, businesses require agility, resiliency, responsiveness, and flexibility. This makes the separation of business and infrastructure policies from programming very important. Business policies should not be hard-wired; the crosscutting business capabilities need to be separated from programming. Another business goal is to reduce TCO (total cost of ownership) by developing reusable services that can be dynamically deployed in different scenarios.

In a retail organization, for example, one requirement might be to have more POS (point-of-sale) services during business hours and more back-end data-transfer services during off hours. A medium-size store needs a total of 30 POS services and 30 back-end services during various times of the day. Without virtualization and dynamic routing, we would need a static capacity of 90 services: $(30+30) \times 1.5$. But we do not need all services at all times (i.e., there is no need for 30 POS and 30 back-end services to be permanently running). Actually, only 30 POS + 5

back-end services are needed during business hours, and 2 POS + 30 back-end services are needed after hours. Why plan for 90 services, when you can plan for 45 with service virtualization based on an application-aware network mechanism? The service virtualization functionality will start, stop, and route between the required services based on various policies. Note that this problem is one of scalability, not parallelism. We can use the same technique for solving concurrent problems that require parallel calculations.

Another example is the business SLA (service-level agreement); an organization would have partners at different levels (such as gold, silver, and normal), where the high-tier partners (gold) would have higher levels of response times. In this case, we need a virtualization layer that not only can differentiate between services but also can monitor response times of different tiers and route according to the business-level service differentiator policies. At the end of a quarter, a company might implement a service differentiator giving preference to gold partners over silver partners. By monitoring the flow, the network intermediary might decide to add more services to satisfy the SLA that an organization has with its gold partners.

Topologically, a set of AON devices would be placed either on the client (service requester) side, in the network cloud, or on the server (service provider) side. For simplicity, let's assume that the services are defined by WSDL and use SOAP/HTTP as transport. First, the policies would be configured into the AON blade, which would say what to do when it encounters particular messages. The AON would monitor the message traffic and when it "sees" the messages it is interested in, it activates the logic. One logic would be to monitor the response times (client-to-server and back), based on network primitives (which we will not get into this time), and store them in DNS. Another logic would be to interface with other virtualization layers (such as server, process, and container) and note when particular services start and stop. A third logic would be to actively start or stop services by interacting with server/process/container software.

With these primitives in place, an AON can load-balance between multiple homogeneous services, route between heterogeneous services, and monitor business SLAs through policy mediation and dynamic endpoint resolution.

One challenge when implementing service virtualization (as opposed to server, process, or container virtualization) through the network is the design time/configuration time/runtime demarcation (i.e., who would do what, when, and how). Services are designed

The Network's NEW Role

and hosted in the application servers; application servers define policies, but the network becomes an intermediary during runtime, executing the policies by managing routing, virtualization (of the transport, server, service, and endpoints), and other related functions.

During design time, the developers and architects work toward an abstract service over a virtualized transport (i.e., transport-agnostic and location-independent). During configuration time, application and network administrators configure virtualization layers (server, process, container) and policy. Finally, during runtime, the AON dynamically resolves the endpoint references, providing a mechanism for virtualized service deployment and invocation and optimized routing between the services, resulting in agility, responsiveness, and reduced operational expenses. When the business expands and needs more services, the infrastructure can be expanded. Then, the network will route between more services. In fact, we will transcend from “brittle deterministic hard-coded” application infrastructure to a “resilient-flexible” service infrastructure.

We can further generalize this architecture and the transport protocols can change dynamically depending on the policies, but this would not result in any development change. In that sense we have separated domain complexity from accidental complexity by adding the network intermediary.

Another challenge is the transparency in the intermediary. One way to accomplish this is to hide the network intermediary from programming and let the WDSL define normal endpoints—say, `www.url1.com/service-x`. Then in the AON device we can configure a policy that says, “When you see `www.url1.com/service-x`, run the logic set, apply the virtualization policies, and route accordingly.” Another method is to ask developers to write the endpoints as `www.url-of-AON-blade.com/service-x` and

then apply the virtualization and policy logic.

A third challenge is the use of end-to-end security, especially SSL, which would encrypt the traffic and thus would be decipherable by an AON. In this case, you would need to add security mediation and an AON as a trusted security intermediary.

THE CROSS-ENTERPRISE INTEGRATION LANDSCAPE

Now let's look at some of the scenarios we might encounter in the cross-enterprise landscape. We will also look at the current approaches to dealing with these scenarios and discuss some simplifications. More importantly, we will point out what's tricky and challenging about using a network—the agonies and ecstasies of pragmatic implementation—based on our experience with our customers.

Legal and internal corporate requirements necessitating uniform policy mediation. Organizations require uniform application of business policies, regardless of the technological platform and application domain. (For example, there could be a policy that for international banking transactions going to Russia, the encryption type should be changed to GOST, which is what Russian banks use; for an international banking transaction to Korea, the encryption would be SEED.) Uniform dynamic policy evaluation and enforcement is an important business-level feature. The requirement stems from not only legal environments but also temporal and spatial business capabilities. Moreover, the pervasive policy enforcement

MORE

Related articles in ACM's Digital Library:

Arsanjani, A. 2002. Developing and integrating enterprise components and services. *Communications of the ACM* 45(10).

Sutherland, J., Van den Heuvel, W. 2002. Developing and integrating enterprise components and services: Enterprise application integration and complete adaptive systems. *Communications of the ACM* 45(10).

These articles will be available online at www.acmqueue.com for an eight-week period beginning May 1. Want full access? Join ACM today at www.acm.org and sign up for the Digital Library.

can take different forms depending on the domain, such as business-to-business, branch-office-to-head-office, DMZ, intra-enterprise, and service provider.

Currently there are multiple policy evaluation and enforcement points, and the coordination among them is difficult. To implement this policy uniformly, either all applications must be designed to understand the policy,

or the network must become the intermediary that evaluates and enforces the policy. From a scalability, flexibility, and availability perspective, the network intermediary is a good choice. Because of the pervasiveness of the network, central policy mediation at the “cloud” can achieve the uniformity, governance, and visibility required.

An AON can “know” what is in the message and also

Tenets for Network Fabric Integration

Using the network as a cross-enterprise integrator, mediator, and application-oriented network has its advantages and disadvantages. There is logic and rationale both for and against embedding application primitives in the network layer.

1. The first and foremost tenet is: “Do not do it just for the sake of it.” You should not embed a primitive just because it can be done—embedding primitives indiscriminately necessitates more logic in the upper layers. Functions such as routing, XML acceleration, and security are suitable as network intermediaries, but transactions or service containers are not appropriate for the network.

2. Embed functions at the lowest layer possible, not any lower or higher; any lower misses the context and the overall objective, and any higher necessitates compensation for lack of visibility and control at the granularity required.

3. Additional services should *not* require simultaneous changes to a network stack, a new piece of infrastructure, and applications at the same time. This subject stems from a discussion in one of the IETF (Internet Engineering Task Force) groups—“forklift” upgrades are *out*, and incremental extension is *in*.

4. When in doubt about where a functionality fits in, think in terms of interoperability, scalability, performance, and availability.

5. The capabilities of the network should provide an orthogonally extensible platform and the principle of bounded uncertainty in the network. A networked infrastructure has to support arbitrary behaviors while providing a coordinated outcome. Another reason for the extensibility is that the deployment environments vary tremendously, and we need designs that permit variation and thus flex under pressure and survive. The network as a platform should separate interfaces that will change and require refinements over time.

6. A network platform should modularize and provide isolation among subsystems that have conflicting interests.

7. Seek synergy across domains and take advantage of work already done. Work should be based on extending open standards—for interoperability as well as for using work from knowledgeable sources and for collaboration.

8. Any feature embedded in the network should increase not only scalability and availability but also the flow structure—i.e., transport agnosticism and location independence. This directive is specific to the service-oriented architecture domain.

9. Whereas traditional enterprise and intra-enterprise systems were heavyweight platforms, the Web 2.0 paradigm has ushered in an era of lightweight programming models that allow for loosely coupled systems. Platforms for Web services from Amazon, Google, and eBay are based on lightweight business models that require a platform that facilitates lightweight programming over lightweight connections.

10. In a Web 2.0 world, syndication (and federation) is more common than coordination.

11. Even though we might not think of it that way, we do need to design for “hackability” and “remixability.”

12. The ultimate networked platform will support “sloppily extensible” formats, the social dimension of software, simplicity, and, above all, loose coupling. This will become Web 2.0 and maybe even Web 3.0.

By applying these tenets, the demarcation between the application and network layer becomes crisper and clearer to the practitioner. While functions such as policy mediation and execution, security, routing, and service virtualization migrate to the network, functionalities such as policy definitions, business process orchestration, and transaction states become crucial at the application servers.

Even though we concentrate here on cross-enterprise integration, the same principles apply equally well for internal integration. In fact, most organizations have islands of systems that need cross-integration—cross-enterprise integration principles fit very well here.

The Network's NEW Role

“learn” about application protocols in many ways. The AON will be watching all the packets—either inline or in an offline mode. In the XML world, the interesting elements would be defined in a bladelet, and when the AON recognized these elements, it would perform the required action—maybe transforming the actual message or raising an event or routing the message, depending on the policy. An AON also has adapters for various application-level protocols.

Multiplicity of transports and interaction patterns.

Another scenario occurs when organizations have multiple transport mechanisms and associated SLAs and requirements. In most cases, the transports would be selected depending on the message, the source, destination, system load, and a host of other factors (e.g., Tibco for internal but HTTP for external). In this case a network-based multi-protocol application router can act as either a transport gateway or a service virtualization mechanism.

This example is similar to the service virtualization example. The SLA-based routing policy selects a different transport and might start new services based on decay of the actual SLA, but transport would also be an attribute in deciding which servers to select and where to start them.

The next layer to transport is the interaction pattern—the common ones being synchronous or asynchronous or publish-subscribe. An effect of developing complex cross-enterprise systems is that there is usually a mix of synchronous and asynchronous services; sometimes the services have long-lasting transactions as well.

In this case, the different capabilities need to be distributed across the application and network layers. What is tricky is finding the right demarcation between the application and the network, and the right feature set. One important guideline is that any transactional state (especially the long-lasting ones) should be the responsibility of applications at the application layer. What is

embedded in the application-aware network would be content-based routing and context tracking. For example, embedded IDs in various XML documents (such as PO number, ticket ID, or even transaction number) can be used to route the messages.

Scalability and performance for thousands of services.

As the usage of cross-enterprise services multiplies and an organization has many services and service consumers, the major challenges are scalability and performance. This may require hardware acceleration and performance offloading to an application-oriented network. AON devices increase performance in many ways: through hardware acceleration, use of network processors, and use of parallelism and distribution. Many times the performance is $O(1)$ even when there are multiple operations on a set of packets.

In this case, the challenge is to separate general-purpose computing and network operations and find the right mix to be embedded in an AON.

Both general-purpose functions and specialized functions have their places. The performance acceleration, however, comes from laser focus (i.e., perform only a few common operations and do them very well). It is not feasible to embed a general-purpose function in the network and expect an increase in performance. As we saw earlier, one major challenge is the demarcation between a pure application and application-oriented networking. Finding enough common functions for achieving system-wide optimization is still an art rather than a science. At the level of current technologies (silicon, microcode, network processors, etc.), the functions for which a network intermediary can increase performance include XML transformation, XML security, and routing.

Another challenge in scaling services is to match functionalities, such as load balancing and service virtualization, with the business requirements and roles. Even though this point might be obvious, the rationale might not be. Just throwing in load balancing or virtualization will not help.

Implementing load balancing is slightly different on the client and the server sides. On the client side, many requests might be aggregated, syndicated, or federated and sent to one service—rather like inverse load balancing. On the server side, load balancing has many forms—the usual round-robin type or some weighted average scheme; there is also endpoint virtualization and adaptive load balancing (based on the context and the message content).

Internet-scale systems follow Internet rules. When cross-enterprise services embrace the Internet and are

architected for an Internet-scale or near Internet-scale paradigm, they are also bound by the three major rules of the Internet:

1. The principle of scale-free networks (i.e., some services will be more popular than others, and the usage patterns of services do not follow a normal curve).
2. The principle of long-tail consumption (i.e., applications and users who do not use the traditional applications will start consuming the distributed services).
3. The rules of disruptive technologies (i.e., as services become easily accessible and granular, they will become more pervasive and thus will be used more than planned).

The second- and third-order scaling complexities (i.e., scaling with regard to the multiple attributes of SOA, at a growth rate that might not be predictable during the design time, especially in terms of popular services) necessitate a distributed network architecture with service virtualization.

Service virtualization (and the ability to start and stop services based on dynamic policies) is a requirement when the scale-free attribute is also temporal. Examples of temporal scale-free services include authentication/login services (which are very busy in the morning), and in a cross-enterprise scenario, order-entry services in business-to-business systems (which are very busy at the end of a quarter).

THOUGHTS AND TRENDS FOR THE FUTURE

Organizations are slowly coming to grips with the reality of modern cross-enterprise systems and are finding ways of incorporating the new role of the network into their enterprise systems. Let's take a look at the trends that will shape the evolution of an application-oriented network.

Sensor networks cause asymmetric information flow.

Recent developments in the industry include applications based on RFID and sensor networks. These result in an asymmetric flow pattern from the edges; content and services generated at the edges create variations to the traditional core-distribution-access layers. This extended edge, which is challenging to handle, can be the result of intelligent edges (for example, sensor networks) or of the dichotomy in service consumer and producer roles across an ecosystem. For example, in an HR application, a service call needs to go out externally for a security check and be returned; a similar dichotomy exists in credit card validation. This situation calls for a network intermediary that acts as an intelligent aggregator at the edges.

RFID and similar applications require complex event processing. In many domains such as logistics, there is a

need to recognize and respond to events, requiring zero latency or near-zero latency organization. Most of these events would, in fact, be outside one organization—a classic cross-enterprise event-flow scenario.

The EPC (Electronic Product Code) global network is an excellent example of this domain. Newer technologies such as RFID require the following capabilities:

- Track and trace across different organizations (for example, between Procter & Gamble and Wal-Mart or Johnson & Johnson and Walgreens)
- Direction and velocity of shipments across multiple sites
- Container hierarchy, where a relationship exists between a package or container and the goods inside it
- Presence and location, which is the normal asset-tracking across organizations

In these cloud-use cases, events are happening across multiple organizations and at different times and frequencies. One cannot possibly expect hundreds of dedicated application servers. This case is for an application network substrate that can perform complex event processing (i.e., nonlinear event propagation) and network-to-application event correlation.

Currently, such use cases are not that common, but businesses have already implemented isolated Web services and are progressing toward a federated SOA model, both inside an organization and across its partner ecosystems.

Network integration requires appropriate interfaces in the network layer. Most network protocols are designed to work in one layer (this is a side effect of specialization) and do not provide enough interfaces to use applications to their best advantage. To use the network to its best advantage from an application perspective, we need new protocols.

One such example is SCTP (Stream Control Transmission Protocol), a newer protocol than TCP. TCP is excellent as a packet-hauling protocol, but for a protocol that requires application transport semantics, SCTP is better. SCTP handles message streams instead of packets; it can carry multiple streams, whereas TCP carries only one stream; SCTP can multi-home and thus provides more resiliency than TCP. In addition, SCTP has a control channel not available in TCP. This example shows how different an AON needs to be from a pure packet-oriented network. Also, remember that SCTP works on IP—like TCP—so we still need the hard-metal wire-speed packet-hauling capabilities, but we also need application-oriented semantics in network protocols.

The AON fabric has enough visibility over topology,

The Network's NEW Role

various optimum paths available, congestion, and other factors while exercising enough control over network artifacts such as QoS bits and routing protocols such as OSPF (Open Shortest Path First) and BGP (Border Gateway Protocol). An AON can twiddle enough bits to achieve synergy with the network as well as the applications. We have been working on taking advantage of the networking protocols in an application-oriented world, but that is a topic for another discussion.

The concept of application-oriented networks will leverage domains other than enterprise. Most of this article has concentrated on the enterprise domain, but other domains and classes of problems will be solved by the same cross-enterprise patterns. For example, we are working on applying the patterns and ideas to the non-transactional Web, such as the networked entertainment and mobile ad hoc networks.

In the area of networked entertainment, opportunities exist in content distribution, content mobility, disintermediation, “transcription” (that is, *transcoding* and *encryption*), and other similar functions. In home networking, users want transparent experience and “stuff that just works”—not that different from the expectations of corporate system users.

Mobile ad hoc networks involve huge challenges, and the paradigms we are using in the cross-enterprise domain can be applied here as well. For example, managing a VoIP directory in a mobile environment requires services, interfaces, virtualization, uniform policy, and security.

CONCLUSION

Developing and maintaining cross-enterprise systems poses many challenges, but there are opportunities for simplification at the application and network layers to achieve the interoperability, scalability, availability,

and security required by business. Application-oriented networking leverages the positional and operational characteristics of networks while embedding selected application primitives into the network. Of course, there are limitations. All functions cannot be and should not be migrated to the network. In that sense, there is a delicate balance between the various mechanisms at the application and network levels. Q

SUGGESTED FURTHER READING

Bosworth, A. *mySQL Conference*; <http://www.itconversations.com/shows/detail571.html>.
<http://oakleafblog.blogspot.com/2005/11/adam-bosworth-learning-from-web-and.html>.
<http://weblog.infoworld.com/udell/2003/04/01.html>.
<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>.
<http://www.adaptivepath.com/publications/essays/archives/000547.php>.

ACKNOWLEDGMENTS

We want to thank John McDowall, Dan Kearns, and Sunil Potti for their ideas, concepts, and insights; and Peter Linkin, Neal Novotny, and Karen Brighton for taking time, in spite of busy schedules, to review and edit the paper. Many thanks to Nancy Darma for facilitating the quick turnaround.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

TAF ANTHIAS is vice president and general manager of application-oriented networking at Cisco Systems. He is a recognized industry leader in messaging and integration middleware. He has held key technical positions and helped to shape a number of software technologies at IBM, including messaging, transaction processing, distributed systems, graphics subsystems, and programming languages. Anthias holds several patents in messaging, clustering, object models, display processors, and windowing systems. He graduated from Cambridge University, UK, with a degree in mathematics.

KRISHNA SANKAR is a Distinguished Engineer with Cisco Systems. His experience ranges from software architecture and development to industrial engineering. He is an author, speaker, entrepreneur, and technology evangelist. He has been part of several standards groups, including W3C, OASIS, JCP Executive Committee, WS-I, ZigBee, and security bodies in the European Union.

© 2006 ACM 1542-7730/06/0500 \$5.00