Copyright Notice

Portions of this article are derived from Chapter 8, "Beyond Web Surfing – Content Services" published in the book, Content Networking by Markus Hofmann and Lee Beaumont (Morgan Kaufmann Publishers, 2005). All portions of the article so derived are reprinted with permission of the publisher. www.mkp.com

Editor: Jim Whitehead • ejw@soe.ucsc.edu

Open Pluggable Edge Services

An Architecture for Networked Content Services

Markus Hofmann • *Bell Labs/Alcatel-Lucent*
Leland R. Beaumont • *Simply Quality*

The IETF's Open Pluggable Edge Services (OPES) working group focuses on rule-based, in-line transformation services of data flows between two Internet endpoints, such as Web servers and Web clients. The group has developed an architectural framework to authorize, invoke, and trace such application-level services. The framework follows a one-party consent model, which requires that at least one of the application-layer endpoints explicitly authorize each service. OPES services must also be reversible by request of the application endpoints.

Service is personal. Every individual has his or her own interests and preferences. Just as Dad might enjoy a cup of unflavored milk in the morning, for example, his kids might prefer chocolate and strawberry. Yet, every family member has quick access to the plain milk, stored locally in the fridge, and "adapts" it to their individual preferences. Similarly, computer users prefer to consume information in different forms. For instance, a European traveler might want to read New York's current temperature in the Celsius scale, whereas her American friend is probably more comfortable with Fahrenheit. Both individuals receive the same information, but the content is processed and presented differently.

With basic transport services becoming commodities and their revenue potential eroding, ISPs are increasingly complementing their offers with value-added services. The Internet's open nature facilitates the development of content networks. Providers commonly deploy Web caches, for example, to move and store content closer to the user, thus reducing server load, improving access latency, and decreasing network load. Web caches are increasingly used to implement additional features such as virus scanning, request filtering, and content adaptation. However, lack of standardized mechanisms for tracing and controlling such intermediaries causes problems with respect to failure detection, data integrity, privacy, and security. To address these problems, the IETF chartered the Open Pluggable Edge Services (OPES) working group.

The Emergence of Networked Content Services

As the functional components of content networks, *networked content services* handle processes such as creating, modifying, converting, and filtering content or requests for it. Providers have typically housed these services at centralized Web servers, but newly defined elements known as *callout servers* let them place the services on components within the networks. Powerful technical and business forces are driving content networks' evolution beyond Web caching; moving content as well as the services operating on it closer to the user is a next logical step in the evolution of content networks.

Technical Drivers

Over the past few years, service providers have used content-delivery techniques such as Web caching and server replication to distribute content across networks. This allows for faster content delivery and improves the Internet's overall scalability, but it assumes infrequently changing, static content that provides the same combination of text or images to each visitor. This assumption is inconsistent with recent trends, as users increasingly demand personalized Web experiences — reading stories relevant to their personal interests

 1089-7801/07/$25.00 © 2007 IEEE

rather than a general mix of headlines when visiting news sites, for example. Such personalization requires additional processing and dynamic page creation, typically performed at the origin Web server. Consequently, the origin server must receive and serve individual user requests, thus eliminating the basic benefits of Web caching and content distribution.

The next logical step is to distribute services operating on, and creating, personalized content. For example, to get a local weather report while sitting at your desk, you must typically type in your current location by hand. A content-delivery system can automate this step if it knows your current location, and it can customize the content and display format if it knows your preferences for using text, graphics, audio, and video. If your preferred language differs from the one in which the weather site predominantly reports, the question arises of which one the report should use and how it should translate the information. Perhaps an enhanced service is available that provides more accurate, up-to-date, or detailed information. Have you subscribed to the service, or can you pay for a single use of it? If you request a local weather report from your PDA while traveling internationally, the service must recognize that you are in a different geographic region, and it must adapt the results to fit the PDA's capabilities. The architectures, protocols, and mechanisms we discuss in this article are aimed at bringing such services to the network edge, close to the consumer.

Business Interests

Services generate revenue — the very thing that business organizations rely on for their continued success. An open infrastructure for creating new content services in cooperation with the network provides attractive separate and complementary opportunities to content providers, network or service providers, and content consumers. The supporting architecture must allow for quick, easy development and deployment of new services to meet users' ever-evolving needs and expectations. Furthermore, content services make it possible to isolate content processing and adaptation from content delivery and storage — activities with different optimization goals that require different expertise.

OPES History

Various approaches for providing value-added services on network intermediaries emerged in the late 1990s — most of them designed to serve specific purposes. Solutions such as the Internet Content Adaptation Protocol (ICAP; www.i-cap.org) were open, but as vendors began implementing proprietary mechanisms, interoperability became a real problem. The need was clear for a standardized, open, and extensible services architecture that would let intermediaries provide services for mediating, modifying, and monitoring application messages.

Network service providers such as Akamai and AT&T recognized the increased flexibility and scalability that would come from separating content applications from specialized Web-caching devices. A standards-based interface would let them choose best-of-breed applications and infrastructures. Application providers such as Trend Micro and Symantec use open interfaces to let their network-based applications communicate with the latest caching and content-delivery systems. It provides them the necessary hooks into the network without the need for formal partnerships with caching and network equipment vendors. Infrastructure vendors such as Lucent, Nortel, Intel, IBM, Network Appliance, CacheFlow, and CacheWare were interested in open interoperability not only to meet their customers' needs but also to get into previously closed market segments. These parties all believed that working together would accelerate the availability of solutions and clarify the problems that must be addressed.

A small group of people from Intel, Novell, Lucent, and CacheFlow headed a so-called "birds of a feather" (BoF) session in December 2000 at the 49th IETF meeting. The OPES BoF aimed to form an IETF working group to develop the protocols and mechanisms for an open content services architecture. The meeting started a long, controversial, and heated discussion in the IETF community that was less about technical details than about the Internet's fundamental design principles and how the proposed architecture would affect them. The discussion focused attention on several architectural and policy issues about robustness and end-to-end data integrity — both long cited as the Internet architecture's overriding goals[1] — which an architecture that allowed elements inside the network to modify messages could potentially erode. For example, critics feared that at some future point an OPES service would perform inappropriately (a virus scanner might reject content that didn't include a virus) or an OPES element could be compromised inadvertently or with malicious intent. The discussion was helpful in identifying potential threats and focusing the proposed OPES work, although it sometimes degenerated into a very ideological and dogmatic argument. As a result, it took three more BoFs and a series of intermediate workshops before the OPES working group was finally chartered in February 2002 (www.ietf.org/html.charters/opes-charter.html).

Given the high stakes at play, the Internet Architecture Board (IAB) took the unusual step of issuing RFC 3238, which included comments and recommendations on the architectural and policy issues related to chartering the OPES working group.[2] Although it doesn't recommend specific solutions or mandate specific functional requirements, the RFC brings to the fore issues on integrity, privacy, and security that any OPES solution standardized in the IETF must address — either by demonstrating appropriate mechanisms or

making a convincing case that no concerns exist. The working group responded by producing RFC 3914, which describes how OPES solutions address those considerations.[3]

By mid-2002, the OPES working group finished its initial charter and put forward a general architecture for networked content services, as well as a comprehensive specification for a next-generation call-out protocol framework. The first specific profile for an OPES protocol was for HTTP-based services.[4] After completing that effort, the working group started on a profile for SMTP-based services. At the time of this writing, however, the OPES working group is in the process of wrapping up due to a lack of active participation. Stagnation in content-edge services deployment and the competitive landscape's thinning after the dot-com downturn lessened the interest in a generalized open solution such as OPES. With only a few deployed services and just a handful of providers and vendors left, specialized point solutions seem economically feasible, for now.

Nonetheless, the working group's documents introduce an architectural framework and a set of requirements to guide standardization of needed protocols and interfaces. The current goal is to finalize the initial work around SMTP to lay the foundation for possible individual contributions later on.

The OPES Architecture

The OPES architecture defines a framework for distributing, authorizing, and invoking networked services at the application level that both offloads origin servers and improves the user experience.[5] Although focused mainly on HTTP-based applications, the architecture is designed to enable support for other applications such as email or multimedia streaming, as well.

OPES as an Evolution of Web Caching

Enhancing Web caches to run additional services, such as filtering Web messages, is a first step toward distributing services that transform or create content. Yet, Web caches are typically specialized devices that are highly tuned for efficient, high-performance file storage and Web retrieval. Running other processing-intensive services on the same network components is likely to degrade cache performance. A better approach is to provide services such as virus scanning and multimedia content transformation on separate elements. To that end, the OPES architecture introduces an open standard interface to callout servers, which Web caches or proxies can use when services are needed. The callout server can be collocated with the callout proxy, or it can be located across the network from it. OPES also allows services to be executed locally on the Web cache or proxy. Callout servers can be added to the network as the need for their particular services is recognized and they become available.

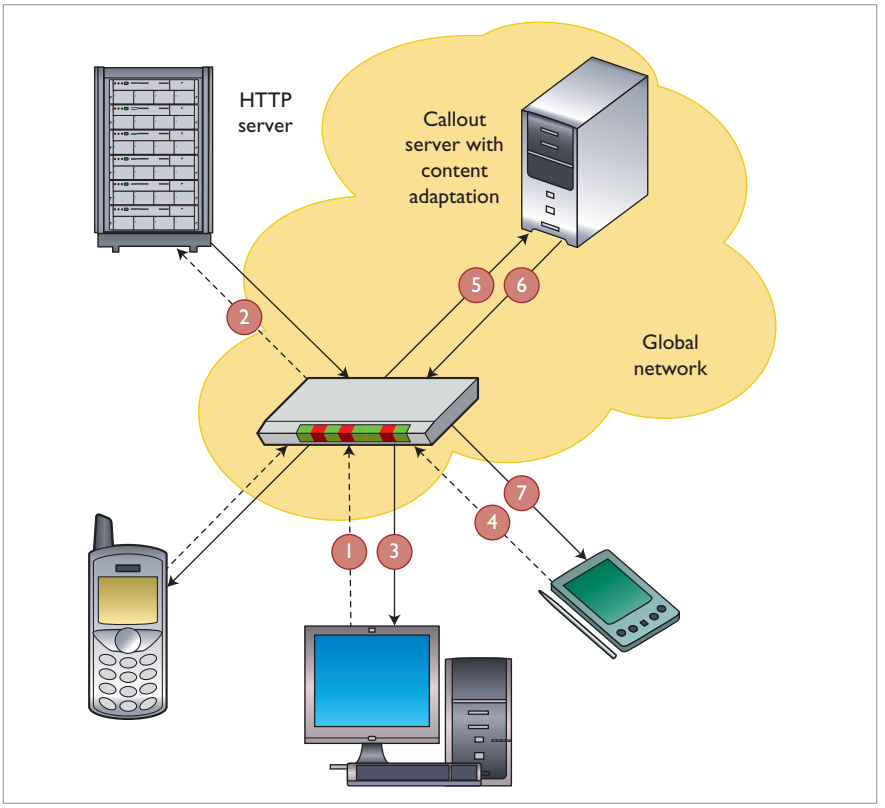

Figure 1. Content-adaptation service. A services-aware Web cache vectors client requests, server responses, or cached content to callout servers, where it is adapted to meet user preferences.

The example content-adaptation service in Figure 1 illustrates how this architecture works to present the same information to two users with different client devices. A content consumer requests a Web page via a PC-based client (1). The request is routed to a cache in the network, which requests the page from an HTTP content server (2), stores it, and forwards a copy to the client (3). When a PDA user later requests the same page, the request goes to the Web cache (4), which already has the page stored. Recognizing the need to adapt the page for the client, the cache sends the content to a callout server (5), which returns a version adapted for PDA display (6). The cache then forwards the adapted page to the PDA-based client (7). Here, the entire request is completed locally without accessing the origin server across the network. This reduces server load, network load, and response time. It's like adding chocolate syrup to the

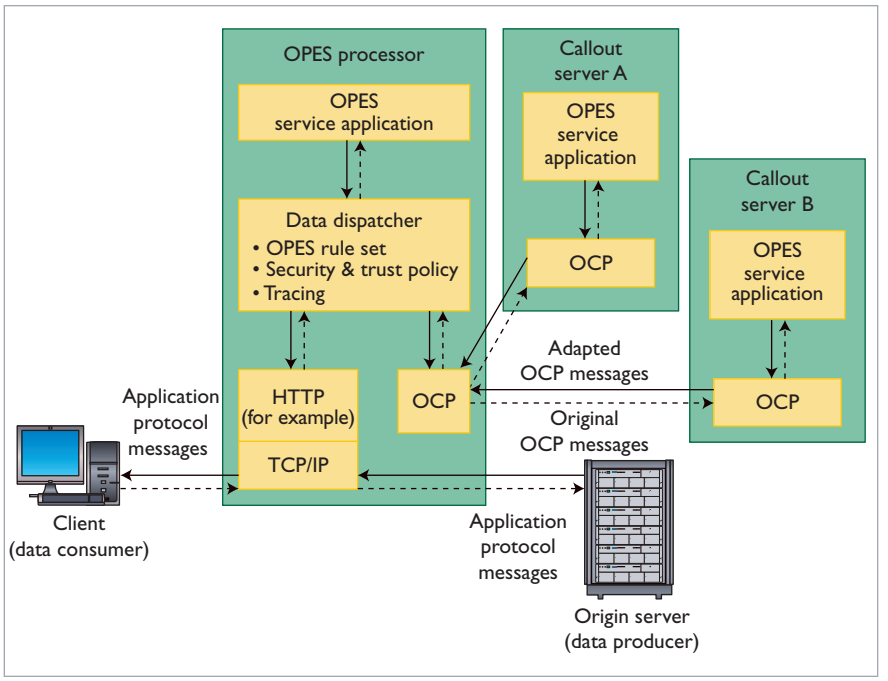

Figure 2. The Open Pluggable Edge Services (OPES) architecture. The data dispatcher within the OPES processor vectors messages to the OPES service application, which may be collocated or reached in a separate callout server via the OPES Callout Protocol (OCP).

milk from the refrigerator, rather than going back to the store for chocolate milk (or searching for a chocolate cow). These savings become more important as the number of client device types to be served increases. Finally, the Web cache has the option of whether to store the adapted page, representing a typical trade-off between space and processing time.

Authorization and Trust

Although the OPES architecture provides many exciting opportunities and promises benefits to content consumers and providers alike, it also has the potential for misuse. Concerns center on the possibility of intercepting message flows between content consumers and providers without their knowledge. Intercepting a message flow for executing services can cause problems similar to those that occur with interception proxies. Consumers and providers can quickly lose trust in networks that modify content without either endpoint being aware of it.

To address the issue, OPES follows a one-party consent model that requires each content service to be authorized by either the content provider or the content consumer. *Surrogate services* are content services, such as watermarking, content adaptation, or dynamic Web page assembly, provided on the origin server's behalf. The elements that make up the services form a surrogate overlay and are logically part of their respective origin servers' authoritative domains. Similarly, *delegate services* are those provided on content consumers' behalf or by the applications they're running — virus scanning or content filtering, for example. The elements that make up the delegate services form a delegate overlay and are logically part of the content-consumer applications' authoritative domains.

The content provider or consumer must securely delegate and transmit policy information, describing what types of services are authorized for various transaction types, from the authorizing party to the data dispatcher's policy-enforcement function.

Architectural Elements

Figure 2 illustrates the various architectural elements and their interactions, as described in RFC 3835. The OPES *processor* is an application-level intermediary on the path between data consumer (a Web client, for example) and data producer (such as a Web server or origin server). The OPES processor analyzes incoming application messages and invokes the appropriate applications. Integral to every OPES processor is the *data dispatcher*, which is the specific component responsible for this message analysis and application invocation. The data dispatcher bases its decisions on a rule set that specifies what applications to invoke on which messages and how. For example, a rule might specify that the services application must scan all binary HTTP response messages for viruses before forwarding them to user Markus. The OPES processor can use a function sometimes called message *vectoring* to hand messages off for further processing by OPES service applications, which can reside on the OPES processor itself or on remote callout servers. A single OPES processor can communicate with multiple callout servers (indicated by A and B in the figure), just as a single callout server can receive requests from multiple OPES processors. The OPES Callout Protocol (OCP) governs communication between OPES processors and callout servers.

The architecture isn't limited to a single OPES processor between the consumer and producer. Indeed, a message might traverse multiple OPES processors on its way between two endpoints. OPES requires, however, that the first OPES processor in such a chain be explicitly addressed at the IP layer. As a first step toward controllability, this policy prohibits deploying OPES processors as interception proxies and thus ensures that the originating endpoint is always aware of the first OPES processor its message travels through.

Controllability, Integrity, and Security Considerations

The working group focused heavily on features intended to allow only authorized content transformation. The intent is to ensure that endpoints are aware and in control of the services performed inside the network.

In an OPES system, each message flow must follow policies established by the data consumer or the data producer, identifying what parties are authorized to perform what services. (The system is responsible for implementing a mechanism to resolve possible conflicts.) The endpoints communicate their policies to their immediate, trusted service providers, thus delegating authority and letting the providers act on the endpoints' behalf, according to the policies set forth. For example, a residential customer might authorize a DSL service provider to perform virus scanning on all binary downloads. The service provider could configure the network elements accordingly by activating the appropriate rules on its OPES processors. Authority delegation can move to more distant entities in a stepwise fashion, such that entity A delegates to entity B, entity B delegates to entity C, and so forth.

User-authorized policies typically extend to include encryption requirements on the various network links, including possible communication with callout servers. Callout servers must not violate trust policies by transmitting information to servers or processes outside the trust domain. Throughout each OPES flow, the callout servers must protect customer data identified as private. They must thus be able to announce their privacy capabilities and ability to enforce privacy policies. To allow operational verification, the OPES architecture requires each OPES system to provide tracing functions. Coupled with strong end-to-end integrity checks such as digital signature techniques, this ensures that control over the services provided in the network remains with the endpoints.

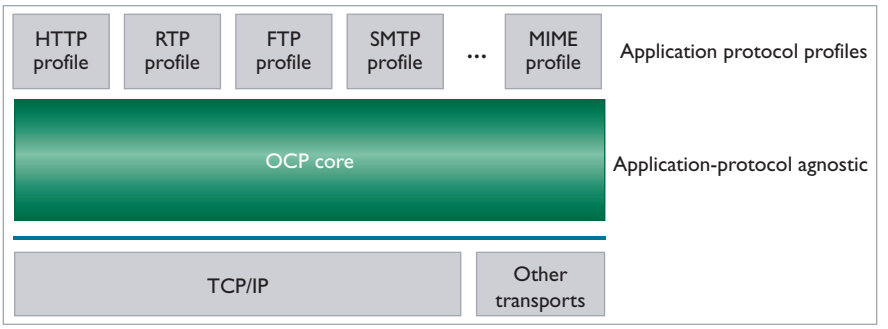

Figure 3. The OPES Callout Protocol architecture. Application-specific protocol profiles augment the protocol-agnostic OCP core.

The OPES Callout Protocol

OCP is currently a proposed IETF standard that indicates the design phase's completion. The protocol details are subject to possible modifications until they prove valid in practice after wider deployment. As such, we don't attempt a detailed description or definitive reference to the protocol. Instead, this article provides an overview of OCP's workings and some of its features; for more details, refer to RFCs 3836 and 4037.[6,7]

The protocol-specification process began by identifying and describing the functional, performance, and security requirements the protocol needed to meet.[6] The working group used this as a basis for selecting between alternate design choices.

The protocol's primary purpose and value is to enable OPES processors to forward application messages to callout servers for processing by OPES services. OCP then lets the callout servers return the results to the processors — possibly including modified messages. Although the working group initially focused on supporting the exchange of HTTP messages, it soon shifted to specifying a generic, application-agnostic protocol core supplemented by application-specific protocol profiles. The protocol core makes no assumptions about the application-layer protocol used in the path between the data producer and the data consumer, which means that features commonly needed for all application protocols can be implemented just once in the protocol core. Providers will develop features specific to each application protocol in separate protocol profiles.

Figure 3 shows the OCP architecture. The OCP core implements application-agnostic features to support various application protocols.[7] We assume that OCP runs on top of a reliable transport protocol such as TCP. In particular, it relies on the underlying protocol to maintain packet ordering and provide congestion-control mechanisms in conformance with RFC 2914.[8] OCP doesn't require separate transport connections for each callout transaction. Instead, it allows multiple transactions over established connections, much as HTTP/1.1 uses persistent connections. OCP even allows transactions on a single connection to overlap. The OCP core is augmented by protocol profiles that are specific to the application protocol used between the content consumer and producer. The working group wrote RFC 4236 to provide a profile specification for HTTP-based applications to allow transmission of Web messages and fragments between OPES processors and callout servers.[9]

OCP offers several novel and useful features:

- *Asynchronous message exchange* allows multiple outstanding callout requests on a single transport connection and provides a method to correlate callout responses to callout requests.

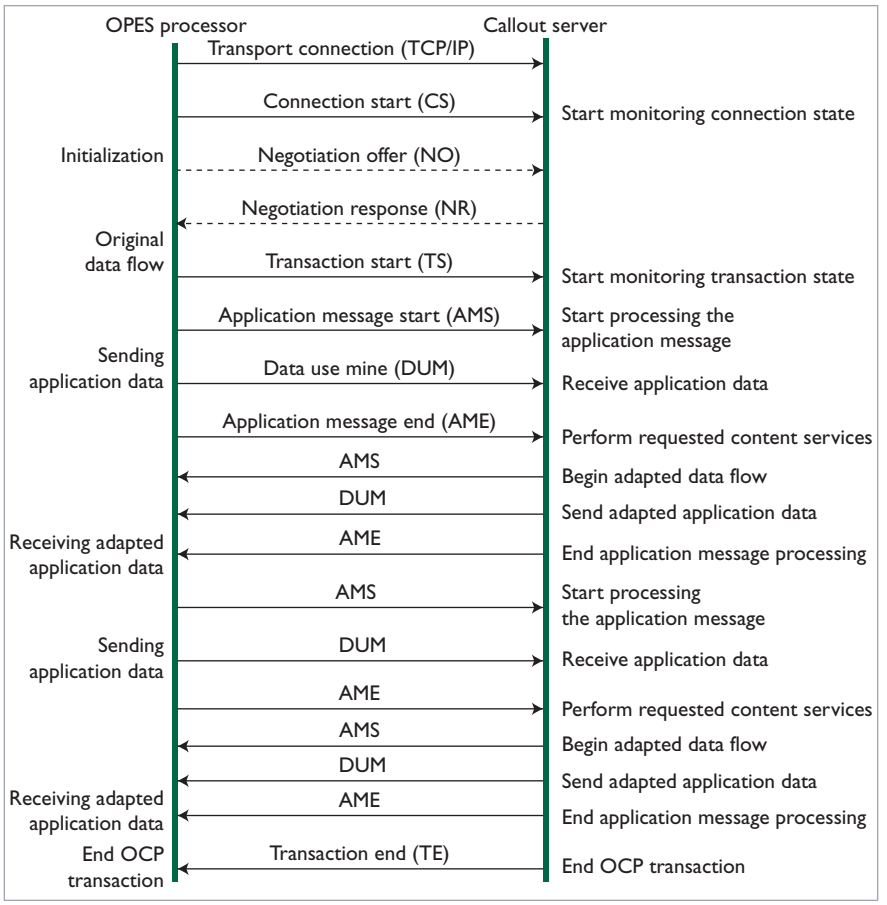

Figure 4. Example OPES Callout Protocol session. An OPES processor requests services from a callout server using an OCP session.

- *Message segmentation* lets the OPES processor forward an application message to a callout server in a series of smaller message fragments and provides a method to reassemble the fragments into the original message.
- *Keep-alive mechanism* lets each endpoint detect if the other has failed — even in the absence of callout transactions.
- *Capability and parameter negotiations* provide support for OPES processors and callout servers to negotiate capabilities and callout connection parameters, including callout protocol version, fail-over behavior, heartbeat rate for keep-alive messages, and security.
- *Metadata mechanism* lets callout-transaction endpoints include instructions for the OPES processor and callout server in callout requests and responses. For example, the processor can include an ordered list of services to be performed on forwarded application messages, as well as instructions for tracing and keeping local copies of the application messages.
- *Premature termination* allows a callout server to abort an ongoing transaction at any time. This is helpful for situations in which the callout server determines that no further actions are required and that transmitting the remaining parts of the application message is unnecessary. Virus scanners, for example, can use this feature to stop data transmission after detecting that the transmitted application message is a text file and, therefore, not in need of virus scanning. This feature is similar to message preview defined in ICAP, but offers more flexibility and a more fine-grained control over when to terminate a transaction.

Details on these features and on how they're used are available in the protocol specifications.[7,9]

Figure 4 illustrates an example OCP session, which begins when the OPES processor establishes a transport connection with the callout server (typically using TCP) and sends a connection start (CS) message to request that the callout server begin monitoring the connection state. The OPES processor then makes a negotiation offer (NO). Requiring one of the parties to initiate the negotiation process avoids possible deadlocks, such as both sides waiting for the other to make an offer. The negotiation mechanism lets the processor and callout server agree on a mutually acceptable set of features, including optional and application-specific behaviors, as well as OCP extensions. For example, they could negotiate transport encryption, data format, and support for new messages. The initialization portion of the session ends when the other party sends a negotiation response (NR).

A transaction start (TS) message begins the original data flow, at which point the callout server begins monitoring the transaction state. An application message start (AMS) then notifies the callout processor to begin processing the application message. The OPES processor sends the application data as the payload portion of one or more data-use-mine (DUM) messages and indicates the application message's completion with an application message end (AME). The callout processor receives the application data and performs the requested content services. It then begins the adapted data flow with an AMS, followed by one or more DUM messages, including the adapted application data. The callout server sends an AME message to inform the OPES processor that application message processing is

complete and it will send no further data for the corresponding message. The server can adapt any number of application messages in a single session, indicating each by the second series of AMS, DUM, AME, AMS, DUM, and AME messages.

Although this example doesn't illustrate the process, the callout server can also initiate a session with an AMS message. OPES allows the parties to send messages asynchronously as well as overlap requests and responses. Finally, either side can send a transaction end (TE) message to end the OCP transaction.

When work on OPES began during the heyday of the dot-com boom, considerable enthusiasm existed around value-added services for enhanced Web-content delivery. Many small start-ups were pushing into the market, typically focusing on specific problem areas and relying on interworking with others to provide comprehensive end-to-end solutions. For example, some companies specialized in content filtering and transformation services. Because they didn't own and operate the network infrastructure, these providers needed standardized methods to connect their services with existing networks. They also wanted to be able to develop and reuse their services in various application environments — to implement and deploy a generic virus-scanning service once and use it for HTTP downloads, email, and instant messaging, for example. In this environment, a strong desire existed for a generalized, flexible, and open framework into which such services could fit.

When the dot-com bubble burst, most of the smaller companies disappeared, and standards-based interoperability was no longer the focus. Big players naturally have less interest in enabling interoperability because they can often charge more for vendor-specific solutions. In addition, deployment of content services stagnated. With only a few popular services being deployed, specialized point solutions are still economically feasible; no real incentive exists to adopt a more complex generalized framework unless the variety of services increases and separate point solutions becomes too costly. Practitioners also appear not to see a need for some of OPES's advanced features, such as tracing or bypass. Although important from an architectural perspective, these features don't seem to provide enough immediate practical value to spur wide adoption.

Consequently, very few known OPES implementations exist. Instead, providers continue to use application-specific point solutions such as ICAP for HTTP-based applications. Only after a larger number of applications with more sophisticated features is deployed will the benefits of a generalized framework such as OPES provide incentives for practitioners to adopt it.

As of this writing, the OPES working group has produced 10 RFCs, discussing use cases and a threat analysis, describing the generalized OPES architecture, and specifying a generic protocol core and a specific HTTP profile. An Internet draft on integrity, privacy, and security in OPES for SMTP is under "working group last call" status to lay a foundation for future individual submissions on SMTP or other application areas.

Acknowledgments

Portions of this article are derived from Chapter 8, "Beyond Web Surfing — Content Services," published in our book, *Content Networking* (Morgan Kaufmann, 2005). All such portions are reprinted with the publisher's permission (www.mkp.com).

References

1. J.H. Saltzer, D.P. Reed, and D.D. Clark, "End-to-End Arguments in System Design," *ACM Trans. Comm.*, vol. 2, no. 4, 1984.
2. S. Floyd and L. Daigle, "IAB Architectural and Policy Considerations for Open Pluggable Edge Services," IETF RFC 3238, Jan. 2002; www.ietf.org/rfc/rfc3238.txt.
3. A. Barbir and A. Rousskov, "Open Pluggable Edge Services (OPES) Treatment of IAB Considerations," IETF RFC 3914, Oct. 2004; www.ietf.org/rfc/rfc3914.txt.
4. A. Rousskov and M. Stecher, "HTTP Adaptation with Open Pluggable Edge Services (OPES)," IETF RFC 4236, Nov. 2005; www.ietf.org/rfc/rfc4236.txt.
5. A. Barbir et al., "An Architecture for Open Pluggable Edge Services (OPES)," IETF RFC 3835, Aug. 2004; www.ietf.org/rfc/rfc3835.txt.
6. A. Beck et al., "Requirements for Open Pluggable Edge Services (OPES) Callout Protocols," IETF RFC 3836, Aug. 2004; www.ietf.org/rfc/rfc3836.txt.
7. A. Rousskov, "Open Pluggable Edge Services (OPES) Callout Protocol (OCP) Core," IETF RFC 4037, Mar. 2005; www.ietf.org/rfc/rfc4037.txt.
8. S. Floyd, "Congestion Control Principles," IETF RFC 2019, Sept. 2000; www.ietf.org/rfc/rfc2914.txt.
9. A. Rousskov and M. Stecher, "HTTP Adaptation with Open Pluggable Edge Services (OPES)," IETF RFC 4236, Nov. 2005; www.ietf.org/rfc/rfc4236.txt

Markus Hofmann is director of multimedia networking research at Bell Labs/Alcatel-Lucent. His research interests include next-generation content-networking solutions, multicast technologies, and network architectures and protocols for converged voice, data, and IPTV services, including signaling and services control. Hofmann has a PhD in computer engineering from the University of Karlsruhe, Germany. He is cochair of the IETF OPES working group and coauthor of *Content Networking: Architecture, Protocols, and Practice* (Morgan Kaufmann, 2005). Contact him at www.mhof.com.

Leland R. Beaumont is the principal at Simply Quality. His research interests include new product development, quality management, and content networking. Beaumont has an MS in electrical engineering from Purdue University. He is coauthor of *Content Networking: Architecture, Protocols, and Practice* (Morgan Kaufmann, 2005). Contact him at lee@simplyquality.org.