Toward an Autonomic Service Management Framework: A Holistic Vision of SOA, AON, and Autonomic Computing

Yu Cheng, Illinois Institute of Technology
Alberto Leon-Garcia, University of Toronto
Ian Foster, University of Chicago

ABSTRACT

The next-generation network will enable provision of various multimedia applications, with different QoS requirements, over a common IP-based transport infrastructure. However, along with the service consolidation over IP and the rapid increase of network scale, the task of service and network management has become extremely complex and costly. Efforts are being made in various areas, for example, SOA, application-oriented networking, and autonomic computing, to achieve a scalable, secure, and self-managed service delivery framework to shorten the time-to-market of new Internet applications, as well as lower the management costs of service providers. Nevertheless, these technologies are not developed in a coordinated manner for optimal scalability, resource utilization, and QoS performance. In this article, we propose a prototype architecture and discuss related implementation issues for an autonomic service management framework, based on a holistic view of SOA, AON, and autonomic computing. In ASMF, a business or management application with end-to-end QoS requirements can be automatically composed from standard service components in a distributed manner and autonomically managed according to service level agreements.

INTRODUCTION

The Internet is experiencing a phase of dramatic evolution driven by the new distributed computing models such as grid computing, peer-to-peer (P2P) networking, and Web search. Along with the trend of service consolidation over Internet Protocol (IP), the Internet is developing into an extremely complex system, and management of the network has become one of the most challenging issues.

Efforts are being made in various areas, for example, service-oriented architecture (SOA) [1], application-oriented networking (AON[1]) [2], and autonomic computing [3], with objectives to reduce the development complexity, lower the management cost, and shorten the time-to-market of new Internet applications. Although SOA, AON, and autonomic computing can each facilitate service creation, delivery, or management from different perspectives, the technologies are not developed in a coordinated manner for optimal scalability, resource utilization, and quality of service (QoS) performance. In this article, we will take a holistic view to design an autonomic service management framework (ASMF) that integrates SOA, AON, and autonomic computing technologies for automatic, scalable, and efficient service/resource management over the next generation Internet.

The ASMF is constructed based on the concept that "everything is a service." Currently, the traditional Internet service providers (ISPs) are gradually expanding their businesses to provisioning upper-layer applications, and thus, their infrastructures are evolving beyond connectivity to encompass content, processing, and storage. In such a wide-sense ISP network, the terms *resource* and *service* will converge into the same generic understanding, which denotes any capability that may be shared and exploited in a networked environment, including both traditional computing/networking resources and virtualized services [4]. Correspondingly, service management and resource management will have equivalent meaning and will be used interchangeably in this article. The generic service perspective is also consistent with the efforts for management using Web services (MUWS) and management of Web services (MOWS), directed by the OASIS Web Services Distributed Management (WSDM) Committee.

The core functional layer of the ASMF is an

[1] *The abbreviation AON stands for either Application-Oriented Networking or Application-Oriented Network, depending on the context where it is used.*

0163-6804/08/$25.00 © 2008 IEEE

autonomic application-enabling fabric (AAEF), which is formed as a self-managed P2P overlay network consisting of autonomic service brokers (ASBs). The ASB overlay contains a distributed service registry database and enables automatic construction of end-to-end QoS guaranteed applications according to the SOA principle in a distributed manner. Furthermore, the AAEF not only supports service-level agreement (SLA)-based autonomic management but also integrates AON technology to optimize the ASB overlay design and enhance the message delivery in the system.

It is noteworthy that in this article we aim to explain the design principles behind SOA, AON, and autonomic computing; revealing the correlation between these principles; and illustrating how the principles can be seamlessly integrated to construct a scalable, secure, and self-managed service delivery framework. Our objective is for the prototype presentation of the ASMF to pave the way for further implementation designs to enable the proposed holistic vision. In the following, we describe the details of the ASMF after a review of related work on SOA, AON, and autonomic computing.

SOA, AON, and Autonomic Computing

Service-Oriented Architecture

In SOA, various software programs, computing devices, and networking resources are encapsulated via standardized common interfaces as loosely-coupled service components; each service component publishes its location and service description to a *registry* (also called a *service broker*). The service in SOA is provisioned according to the "find, bind, and execute" paradigm as shown in Fig. 1, and complex services or applications can be assembled on demand by querying the registry and locating and combining the required service components. In SOA, a service component can be readily reused/repurposed for different functionalities and conveniently upgraded with an advanced implementation [5].

The universal description, discovery, and integration (UDDI) is the current industry standard to implement the service registry in SOA. The UDDI has been proposed as a central entity, but the scalability concern motivated the study of distributed design by connecting multiple localized registries or brokers with the peer-to-peer overlay technology [6]. Our proposed application-enabling fabric also is formed as a *broker overlay network*, where we will discuss the important open issues such as how to organize the overlay topology to facilitate responsive service query, how to search for a set of correlated services that meet an end-to-end QoS requirement, and how to negotiate SLAs and manage resources in a distributed approach.

The successful deployment of SOA requires a dependable and automatic service composition mechanism. The majority of the existing composition solutions can be coarsely classified into template-based [1] and semantics-based [7] approaches, but, as far as we know, most of the existing composition solutions require a centralized implementation, which considerably limits the scalability. In particular, one of the template-based approaches, the business process execution language (BPEL) [1], is becoming the industry standard for service composition. The BPEL process is centralized in a single BPEL document that invokes the required service components according to a predefined composition workflow. In the proposed ASMF, the service composition and service invocation are to be handled by the broker overlay network in a distributed fashion.

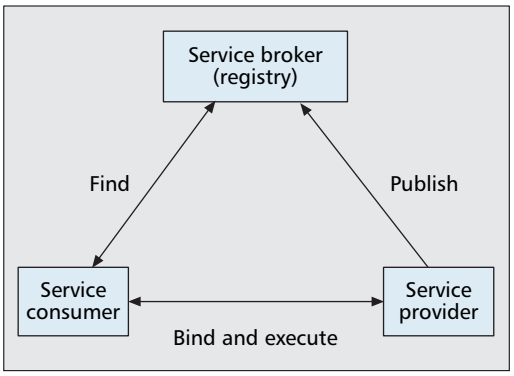


Figure 1. *Find, bind, and execute paradigm in SOA.*

SOA is an abstract reference model, implemented by different techniques. Currently, the SOA implementation based on Web services is becoming popular and being standardized in the grid computing and Web computing communities, where the service definition is described in eXtensible markup language (XML) and presented in the form of Web services description language (WSDL). In SOA, messages exchanged among service requesters, service providers, and service brokers are usually in XML format and communicated over the SOAP protocol. As XML coding is verbose, pure software-based XML parsing often leads to unfavorable overhead in a SOA system.

Application-Oriented Networking

The efficient implementation of SOA requires a powerful messaging backbone, which is one of the major motivations leading to the AON technology initiated by Cisco. An AON-based network can transparently intercept the content and context of application messages, conduct operations on those messages according to business-driven policies and rules, and deliver wire-speed inspection and processing of information. The cost and complexity associated with SOA deployment will decrease significantly by embedding the application messaging services within the network fabric, instead of into middleware stacks [2]. Although AON currently stands for a manufacture-specific solution, in this article, we take a generic interpretation of application-oriented networking: *the IP devices can intercept and process the application messages*.

Currently, it is not clear in both industry and academia how to exploit the AON capability systematically to enhance network and service management. The existing messaging middleware for SOA, running over a best-effort networking

An AON-based network can transparently intercept the content and context of application messages, conduct operations on those messages according to business-driven policies and rules, and deliver wire-speed inspection and processing of information.

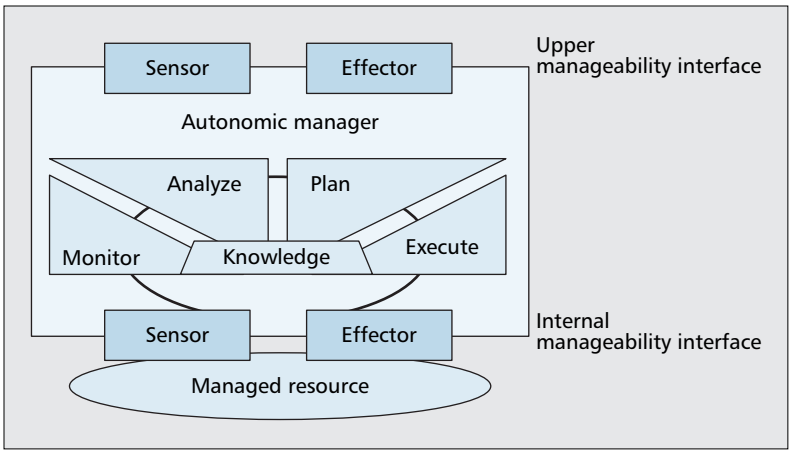


Figure 2. *Architecture for an autonomic element.*

infrastructure, must involve resource monitoring, QoS routing, and fault recovery schemes to ensure satisfactory service delivery. With AON, the messaging system is built into the backbone where the resource availability and QoS control information are much easier to access than in the application-layer, so it is expected that the message routing in AON might be simplified considerably compared to the middleware solutions. In this article, we initiate the discussion on how to utilize the AON infrastructure to facilitate SOA-based service creation and management, particularly by a distributed application-enabling fabric.

Autonomic Computing

The key feature of autonomic computing is the automated management of computing resources, encompassing the characteristics of *self-configuration*, *self-optimization*, *self-healing*, and *self-protection*. An autonomic system is a collection of autonomic elements. Each autonomic element consists of an autonomic manager (AM) and the managed resource (MR), as shown in Fig. 2. The AM operates according to a "monitor, analyze, plan, and execute" control loop, with supporting knowledge of the computing environment and management policies. The communication between the AM and the MR goes through the MR *manageability interface*, which is organized into a *sensor* and an *effector*. The sensor is used to obtain data from the MR, and the effector is used to perform operations on the MR. The autonomic manager also provides sensor and effector interfaces for other autonomic managers to use, so that each autonomic element itself can be treated as a managed resource for resource or service composition. For convenience, we differentiate the interfaces as *upper manageability interface* and *internal manageability interface*. To form a self-managed system, all the autonomic elements are orchestrated together under high-level policies [3].

The autonomic computing architecture and SOA are both established on component-based reference models. By nature, the two architectures can be integrated seamlessly: each autonomic element can be encapsulated with a standard Web service interface to form an autonomic service component, and the "find, bind, and execute" SOA principle and associated messaging schemes can be used to orchestrate the autonomic service components into a self-managed system. IBM, the initiator of the autonomic computing architecture, already proposed to define the manageability interface of an autonomic element according to the OASIS WSDM standard [8]. However, to the best of our knowledge, the existing system-integration studies have considered neither the issue of distributed service composition nor the integration with AON.

Autonomic Service Management Framework

The proposed autonomic service management framework is shown in Fig. 3, where applications are created as a composition of manageable service/resource components. A service component could be directly provisioned by physical resources or virtually provisioned by combining other services. All the resources are manageable. Each manageable resource and the associated autonomic manager are wrapped as a standard Web service component and furthermore connected to form a Web services network. The service components interact with each other to form a service relationship and behave as service providers or customers, correspondingly. A service component may act both as a service provider and a customer at the same time, which enables constructing applications over a hierarchy of service components. To ensure QoS and efficient resource utilization, a service level agreement usually is contracted between the customer and the service provider.

The cornerstone to the ASMF is a service broker based application-enabling fabric, the AAEF. In AAEF, each service broker is designed as an autonomic element, termed as an autonomic service broker, and multiple ASBs form a self-managed P2P fabric that autonomically enables service composition and SLA-based resource management. We emphasize that each ASB also provides a manageability interface, but mainly for human operators to install high-level policies, for example, the P2P protocols underpinning the overlay network, the security policies, and the SLA templates. For convenience of expression, we will term the services constructed through the AAEF as applications or processes. Another important piece of the ASMF is the AON transport infrastructure, where the network-embedded application intelligence is utilized to facilitate SOA messaging, enhance the ASB overlay performance, and enforce security policies. In the following sections, we discuss the Web services network, AAEF, and AON transport network in details.

Web Services Network

In the ASMF, we follow the OASIS MUWS and MOWS specifications to consider *manageable Web services*, which publish optional interfaces and manageability interfaces in WSDL documents to facilitate business applications composition and management applications composition, respectively.[2] All the manageable Web services collectively form a Web services network (WSN),

[2] *The separation of business applications and management applications is to emphasize the service composition capability through either the optional interfaces or the manageability interfaces. In practice, both types of interfaces can be simultaneously involved in a service composition process to constitute a business or management application.*

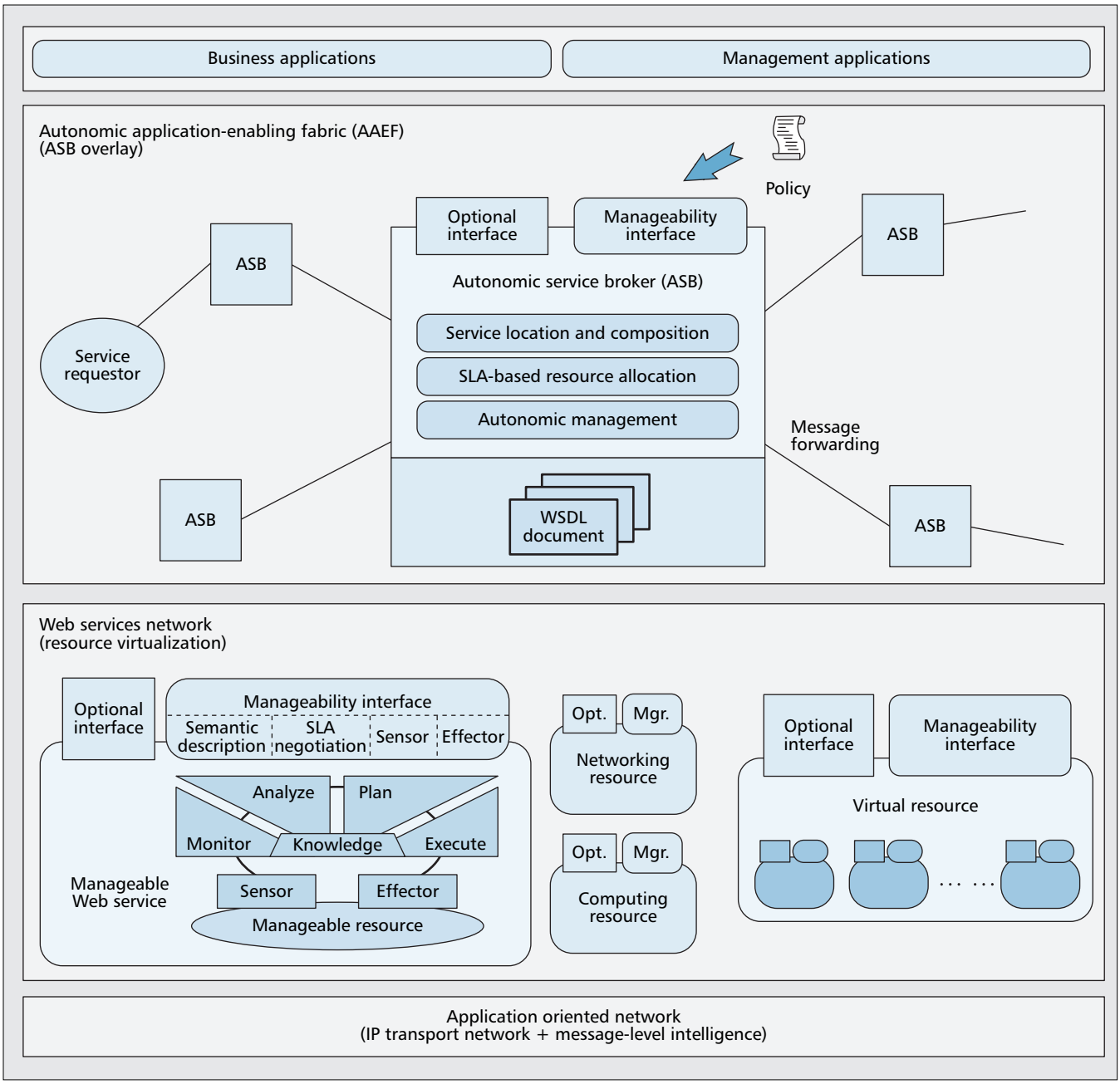


Figure 3. *Autonomic service management framework.*

which can be viewed as a *resource virtualization layer over the physical transport infrastructure*. The capability of virtualization to decouple utility from specific physical resources can greatly facilitate dynamic resource sharing and infrastructure optimization [4].

The MUWS and MOWS specifications do not define the internal operational schemes for the resource management. It is a promising design to implement an autonomic manager to manage the resource [8], which will apply necessary monitoring, resource allocation, admission control, and resource sharing techniques to ensure SLA compliance, as well as high resource utilization and to trigger the SLA renegotiation when the external workload or other dynamics exceed the adaption capability of the internal management system. We would like to emphasize that, to form a WSN, the upper manageability interface of a Web service component is required to follow the WSDM standard, whereas the internal manageability interface could be component-specific.

According to the OASIS WSDM specifications, the manageability capabilities offered through the *manageability interface* include identification, configuration, metrics, status, operations, and events generated by manageable entities for management purposes. In ASMF, we also consider distributed service location/composition and SLA-based resource allocation as important manageability capabilities. Thus, the manageability interface of each Web service is further categorized into interfaces for *semantic description*, *SLA negotiation*, and *autonomic management* (sensor and effector).

The autonomic application-enabling fabric is the core functional layer of the ASMF, which is mainly responsible for creating services or applications according to the SOA methodology, and also provide autonomic managers to those customers that do not have their own management system.

Semantic Description — A service component can be basically defined by specifying the operations that the component performs and the properties of the component that are summarized into the *optional* or *functional interface* for a manageable Web service. Within the optional interface, a specific operation is modeled as a pair of inputs to and outputs from the component, and each input and output is modeled as a pair of name and data type. The semantic description aims at adding machine-interpretable information to the Web service optional interface to enable automatic service discovery and composition. To model the semantics of a Web service component, the entities of *concepts* can be defined to represent abstract ideas that are then used to annotate the semantics of the operations, inputs, outputs, and the properties of the component [7]. A concept also can be used to specify the relationship between two concepts.

With the concept annotation, the machine-understandable description of a Web service can be in the format of a *semantic graph* that consists of nodes and labeled links. Nodes in the semantic graph represent operations, inputs, outputs, and the properties of a component, as well as their data types and concepts. Labeled links in the semantic graph represent the relationship among the nodes [7]. It is noteworthy that the semantic descriptions and the optional descriptions are in fact closely coupled in a semantic graph; we abstract the semantic capability into a separate interface for convenience in discussing the management issue. Using the semantic descriptions, the discovery process can be based on the semantic match between a declarative description of the service being sought and a description of the service being offered; the composition process can be based on integrating the semantic graphs from different service components to fulfill the semantic graph of the requested service.

SLA Negotiation — In the ASMF, the service relationship between the components involved in the delivery of a service is regulated by SLAs. SLAs are contracts between SPs and customers; defining the service performance metrics, QoS levels, authentication, authorization, and accounting related rules, and lifecycle of the service. An SLA negotiation procedure is usually executed through the SLA negotiation interface before the SLA is formally contracted. To facilitate automatic SLA negotiation, the interface must define a standard SLA template [9] to present the service contract in a machine-understandable format. The SLA-based resource allocation is critical for implementing the resource virtualization layer; the manageable resources associated with a Web service can be virtualized resources, which are in fact some kind of serving capability provisioned by other Web service components through certain SLAs.

Sensor and Effector — The sensor and effector are manageability interfaces to enable autonomic management of the Web service component in the role of a Web service resource (WS-resource). The autonomic manager can retrieve resource utilization status from the sensor interface, through which the service component also can actively report status to the manager. The effector interface will be used to deliver the configuration or control information from the manger to the service component. A typical example to illustrate the functionalities of the sensor and effector interfaces is the *admission control*. When the manager receives a new service request and successfully locates an SP component, the manager next must check the resource availability through the sensor interface. If the SP has enough leftover capacity to fulfill the new service request, the manager can then establish the resource commitment through SLA negotiation. After a successful negotiation, the manager delivers the corresponding resource configuration parameters through the effector interface to the service component to enforce the resource allocation supporting the new service. According to the OASIS Web services resource framework (WSRF) specifications, the sensor and effector interfaces of a stateful WS-resource can be implemented through the definition of a resource properties document and a standard set of message exchanges for querying or updating the property values of the WS-resource [10].

AUTONOMIC APPLICATION-ENABLING FABRIC

The autonomic application-enabling fabric is the core functional layer of the ASMF, which is mainly responsible for creating services or applications according to the SOA methodology, and also provide autonomic managers to those customers that do not have their own management system. In AAEF, the service broker, which is the critical entity in SOA, is enhanced to an autonomic element defined as the *autonomic service broker*. Multiple ASBs form an overlay network implementing service composition and resource management in a distributed approach. The connectivity of the overlay network can be maintained by an existing efficient P2P technique; the service level functions are independent of the underlying P2P implementations.

The internal structure of an ASB is illustrated in Fig. 4. One of the basic functionalities of the ASB is to operate as a service registry to allow SOA-based application enabling. Each manageable Web service in a WSN will publish two service endpoints: *optional service endpoint* binding to the optional interface and *manageability service endpoint* binding to the manageability interface. The ASB overlay will serve as a distributed database to store all the published service descriptions. At the same time, the ASB overlay also will be responsible for automatic service location and composition by querying the distributed service registry database. As each ASB in the overlay is likely to interface with a customer directly, a *semantic request analyzer* is used to parse the service request, which might be in a natural language, into a semantic graph representation, and an *SLA translator* is used to translate the human-oriented QoS specifications into the a standard service level specification (SLS) based on an SLA template. The semantic graph

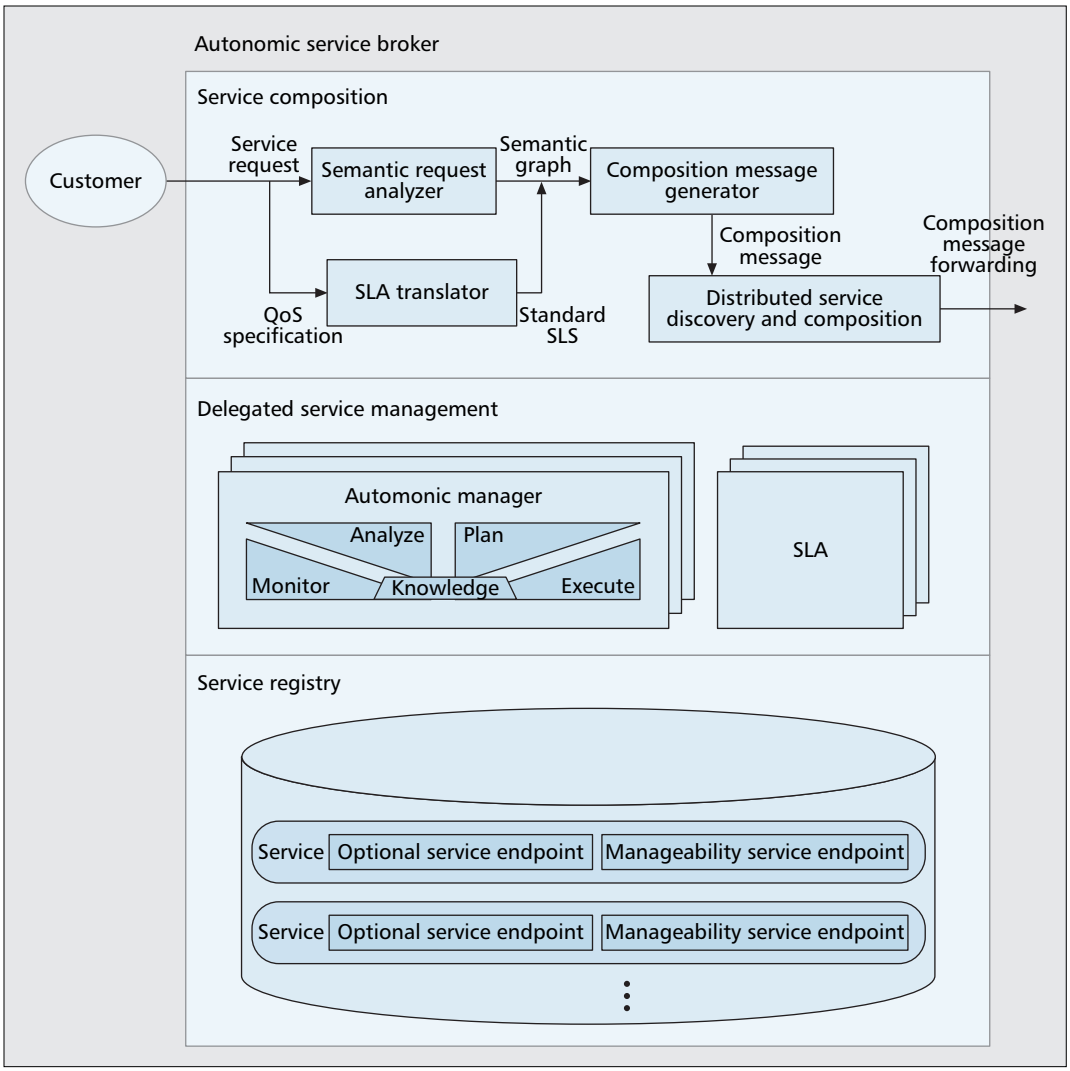


Figure 4. *Autonomic service broker functional design.*

There already exist various approaches for content location in a P2P overlay network. However, an important issue is that the ASB overlay usually must find a set of correlated services to assemble an application.

and SLS information will be wrapped into a composition message, to be delivered over the ASB overlay for service discovery and composition.

In the ASMF, some service components may not have enough storage, hardware, or software capabilities to execute SLA negotiation, store the contracted SLAs, or implement an autonomic manager. Such service components involved in an application delivery can delegate the AAEF to implement service management functions. For delegated management, autonomic managers must be created on the fly for the composed services, which will be greatly facilitated by the ongoing standardization efforts, including the modular design of an autonomic manager, elaboration of manageability interfaces, and composition of management applications. One possible implementation of an SLA-driven autonomic manager is presented in [9].

Distributed Service Location and Composition

In AAEF, when a service request comes in, the ASB overlay must deliver the service query messages to locate appropriate service components that can be combined to meet the service requirement. There already exist various approaches for content location in a P2P overlay network. However, an important issue is that the ASB overlay usually must find a set of correlated services to assemble an application. For example, when a customer requests to watch an online movie, the ASB overlay tries to locate a content provider for that movie, a network provider to provide a delivery path between the content provider and the customer, and an authentication/billing agent associated with the content provider. There is *temporal correlation* among the required service components: the content provider may start delivering the contents only after the customer passes the identity authentication and a proper bill has been charged. There also exists *spatial correlation:* different combinations of content providers and network providers can provide the movie at different view quality with different charges, and the customer usually would like the ASB overlay to enable the application in the most cost-effective manner.

While the current BPEL approach cannot meet the requirements on automatic, distributed, and QoS-enabled service location and com-

The initiator receives multiple returned composition messages, each of which carries a semantic graph partially or fully filled along a certain branch of the tree. The composition initiator completes the service composition by combining the multiple partial graphs.

position, the semantic-based dynamic service composition is a possible solution. According to the ASB design shown in Fig. 4, a customer can send a service request in an intuitive form, which will be parsed into a semantic graph by the first-hop ASB, termed as the *composition initiator*. The graph and the associated SLS for the service then will be wrapped into a composition message to be passed around in the ASB overlay. When the composition message reaches a certain ASB in the overlay, the semantic-enabled service descriptions stored in the ASB is examined against the service semantic graph. If matched service components are found, they are selected to form a partially-filled semantic graph. The access points binding to the selected components are attached to the composition message, and the message is then delivered to the next-hop ASB by the overlay. As the composition message travels along, the service semantic graph is gradually filled. When the composition message returns back to composition initiator ASB, the graph is fully filled, with all the service components to constitute the service located.

Circling the composition message around the P2P ASB overlay may lead to large overhead in composing a service. Instead, the composition initiator may set up a tree, by exploiting the AON messaging capability, to broadcast the composition message to all the other ASBs along the tree, as illustrated in Fig. 5. Each ASB still tries to fill the semantic graph after receiving the message and then continues the forwarding along the tree. When each leaf node in the tree finishes its processing, it returns the composition message to the composition initiator. The initiator receives multiple returned composition messages, each of which carries a semantic graph partially or fully filled along a certain branch of the tree. The composition initiator completes the service composition by combining the multiple partial graphs.

SLA Negotiation and Instantiation

The service composition procedure can be incorporated with SLA negotiation functionality. With SLA-based resource management, a customer submits a service request associated with a preferred QoS range and acceptable cost range, according to Fig. 4. Correspondingly, each service component also includes in its service description the set of provisioned QoS levels and the associated service charges, expressed in a standard SLS format. In AAEF, while the composition message is delivered within the ASB overlay, the SLS information of all the candidate components is collected.

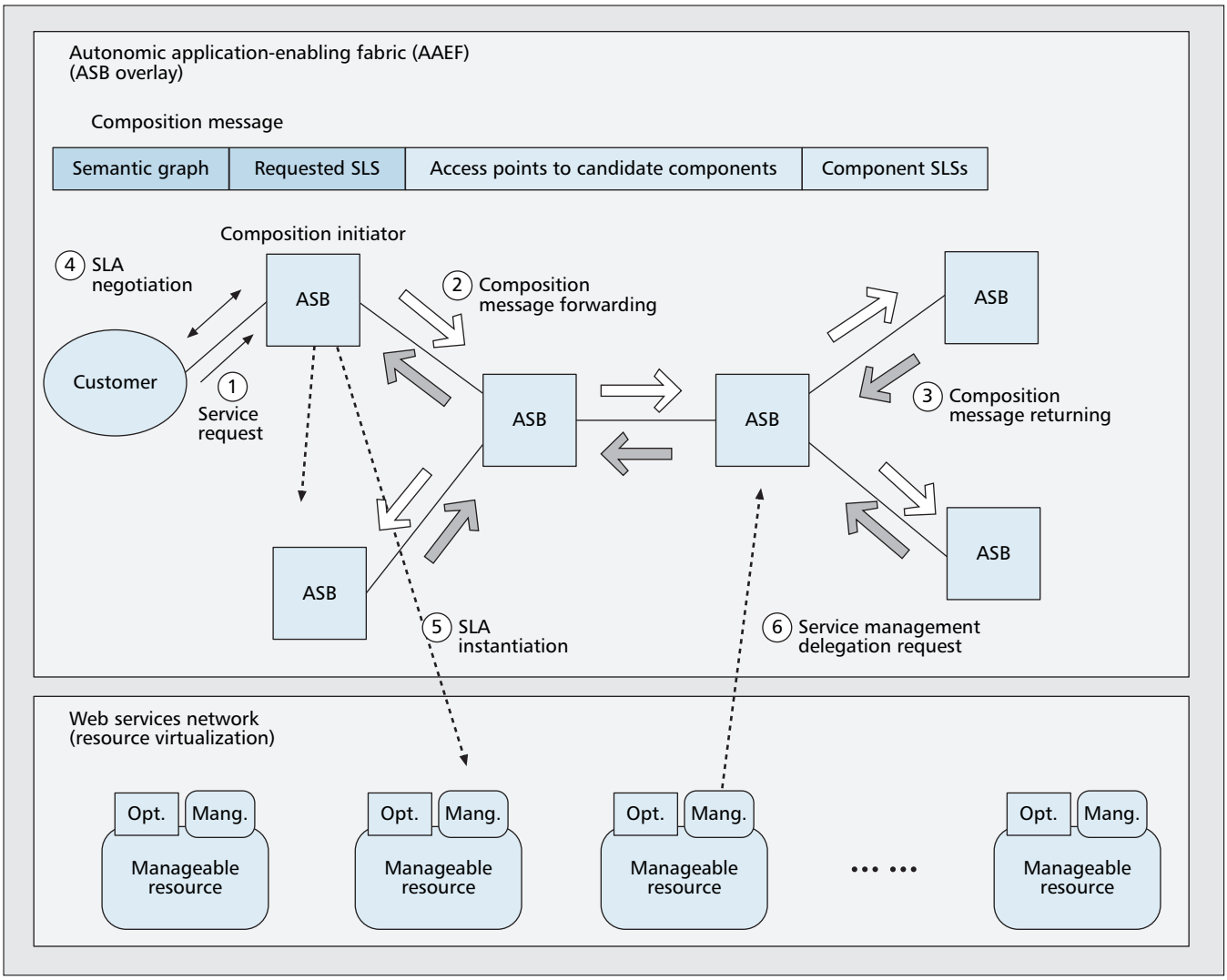
When the service semantic graph is fulfilled at the composition initiator after the composition procedure, it can be annotated by the QoS and charge information from the candidate component SLSs. The composition initiator ASB can forward the annotated service graph to the customer. The customer can compute a QoS/cost solution from the graph based on a certain utility model and then, directly contact the selected service components to contract SLAs. Nevertheless, the customer may delegate the initiator ASB to find the optimal solution from the service graph. In a case where a valid solution that meets the customer's QoS/cost requirements is not available, the ASB may make an adjustment recommendation to the customer and then restart a new composition and negotiation procedure under the customer's new service request. After a service solution is successfully achieved, the ASB sends service instantiation messages to each selected service component to contract SLA resource commitments, instantiate services, and activate the associated autonomic managers. If a service component must delegate the AAEF to store the contracted SLAs and implement the associated autonomic manager, it sends a *service management delegation request* message to the ASB where its service description is published; this way, the ASB can trace the resource usage of the service components under its management. The SLA negotiation/instantiation procedure also is illustrated in Fig. 5, in addition to the service location/composition procedure.

Application-Oriented Transport Network

In ASMF, we consider an AON transport infrastructure, which embeds the service-oriented messaging backbone within the network. According to the information from Cisco [2], the AON fabric can support security schemes from the network layer up to the application layer; this also brings the convenience of enforcing consistent security policies across services and applications. Here, in particular, we discuss how to exploit the AON technology to improve the performance of the ASB overlay network.

In AAEF, the service descriptions published by service components are distributively stored in the ASB overlay network. The peer-to-peer technology is an attractive approach to construct the ASB overlay due to its properties of self-organizing and load balancing. An important issue in P2P networking is how to exploit *proximity* or *locality* in the underlying transport network. With an inappropriate overlay topology, nearby nodes in the overlay network may actually be far apart in the transport network. Because the P2P network is an application-layer overlay, exploitation of network locality usually involves network measurement, as well as complex locality maintenance mechanisms.

When the ASB overlay network is built over an AON, with each ASB connected to an AON router, the AON capability to access both the application-layer and network-layer information can be utilized to establish a *locality-aware* overlay topology and strengthen the overlay performance through network layer QoS or traffic engineering techniques. Through the AON routers, the shortest path(s) from the network-layer routing table will be readily available for the overlay construction procedure to exploit the network proximity. At the same time, being aware of the contents, the AON routers underpinning the ASB overlay can apply the network-layer DiffServ schemes to accelerate the delivery of those more important messages. Moreover, when a logic link in the



■ **Figure 5.** *Distributed service location and composition with SLA-based management.*

overlay is found to be broken due to various reasons, the network-layer rerouting can be resorted to first reestablish the underlay link before reconstructing the overlay topology. Such a lower-layer fixing capability may improve the stability of the overlay network considerably, especially when a simple overlay topology (e.g., a Chord ring) is used.

In an application oriented network, a P2P overlay may not be the best approach to organize the ASB network. The P2P overlay was developed to distribute contents (in a load-balancing manner) into hosts that reside at the network edge. With AON, each router (at the edge or in the core) can interpret and process both application-layer messages and IP packets, so the application-layer routing and the network-layer routing may be seamlessly integrated. For example, application-oriented multicasting can be conveniently achieved at the network layer in AON. Specifically, the overlay network may assemble the list of the application-layer identifications or addresses of the destination nodes into a multicast message. The AON router will intercept the message and retrieve the corresponding IP addresses of the destinations, by which the messages then can be delivered readily through an IP multicast scheme. According to our previous discussions, an efficient multicast scheme will considerably facilitate distributed service location and composition.

Conclusion

In this article, we present a prototype autonomic service management framework to shed light on how to streamline the technologies from SOA, AON, and autonomic computing to facilitate the next-generation network management. The ASMF is designed according to a generic service perspective, where all the networking, computing, or logical resources are encapsulated into manageable Web services. All the manageable Web service components together virtualize the physical transport network into a manageable Web services network, which facilitates the SOA-based service creation and autonomic service management. The core functional layer of the ASMF is an autonomic application-enabling fabric, where the semantic Web service description is exploited and a composition message-

The core functional layer of the ASMF is an autonomic application-enabling fabric, where the semantic Web service description is exploited and a composition message-based approach is proposed to achieve automatic and distributed service location and composition.

based approach is proposed to achieve automatic and distributed service location and composition. The end-to-end QoS performance of the composed service can be ensured by SLA-based management. In particular, we discuss how to exploit the AON technology to improve the robustness and locality of the ASB overlay network that implements the distributed service location and composition.

References

[1] J. Pasley, "How BPEL and SOA are Changing Web Services Development," *IEEE Internet Comp.*, vol. 9, May–June 2005, pp. 60–67.

[2] Cisco Systems, "Cisco Application-Oriented Networking," white paper, 2005; http://www.cisco.com/ application/pdf/en/us/guest/products/ps6438/c1650/cdccont 0900aecd802c1f9c.pdf

[3] IBM, "An Architectural Blueprint for Autonomic Computing (4th ed.)," white paper, June 2006; http: //www-03.ibm.com/autonomic/pdfs/AC Blueprint White Paper 4th.pdf

[4] K. Czajkowski, I. Foster, and C. Kesselman, "Agreement-Based Resource Management," *Proc. IEEE*, vol. 93, Mar. 2005, pp. 631–43.

[5] ITU-T Rec M.3060/Y.2401, "Principles for the Management of the Next Generation Networks," Mar. 2006.

[6] G. Koloniari and E. Pitoura, "Peer-to-Peer Management of XML Data: Issues and Research Challenges," *SIGMOD Record*, vol. 34, June 2005, pp. 6–17.

[7] K. Fujii and T. Suda, "Semantics-Based Dynamic Service Composition," *IEEE JSAC*, vol. 23, Dec. 2005, pp. 2361–72.

[8] H. Kreger and T. Studwell, "Autonomic Computing and Web Services Distributed Management," IBM white paper, June 2005; http://www.ibm.com/developerworks/autonomic/library/ac-architect/

[9] A. Dan *et al.*, "Web Services on Demand: WSLA-Driven Automated Management," *IBM Systems J.*, vol. 43, no. 1, 2004, pp. 136–58.

[10] OASIS Web Services Resource Framework (WSRF) TC, Web Services Resource Properties 1.2 Committee Specification, Jan. 2006; http://docs.oasis-open.org/wsrf/wsrf-ws resource properties-1.2-spec-cs-01.pdf

Biographies

YU CHENG (cheng@iit.edu) received his Ph.D. degree in electrical and computer engineering from the University of Waterloo in 2003. Since August 2006 he has been an assistant professor in the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago. His research interests include application-oriented networking, Internet performance analysis, and wireless networks. He received a postdoctoral fellowship award from the Natural Sciences and Engineering Research Council of Canada (NSERC) in 2004 and a best paper award from the International Conference on Heterogeneous Networking for Quality, Reliability, Security, and Robustness 2007. He is an Associate Editor for *IEEE Transactions on Vehicular Technology*.

ALBERTO LEON-GARCIA [F] (alberto.leongarcia@utoronto.ca) holds the Jeffrey Skoll Chair in Computer Networks and Innovation. He also holds a Canada Research Chair in Autonomic Service Architecture. His current research interests are focused on application-oriented networking and autonomic resources management. From 1999 to 2002 he was founder and CTO of AcceLight Networks in Ottawa, which developed an all-optical fabric multiterabit multiservice core switch. He holds several patents and has published research extensively in the areas of switch architecture and traffic management. He is recognized as an innovator in networking education. In 1986 he led the development of the University of Toronto-Northern Telecom Network Engineering Program. In 1997 he also led the development of the Master of Engineering in Telecommunications program and the communications and networking options in the undergraduate computer engineering program. He is the author of leading textbooks *Probability and Random Processes for Electrical Engineering* and *Communication Networks: Fundamental Concepts and Key Architecture*.

IAN FOSTER (foster@cs.uchicago.edu) is director of the Computation Institute at Argonne National Laboratory, where he is also an Argonne Distinguished Fellow. At the University of Chicago he is the Arthur Holly Compton Distinguished Service Professor of Computer Science. His research interests are distributed, parallel, and data-intensive computing technologies and applications. He has published six books, and over 300 articles and technical reports on these and related topics.