QoS Issues in Web Services

Daniel A. Menascé • *George Mason University* • *menasce@cs.gmu.edu*

Traditionally, access to services offered over the World Wide Web has relied on the interaction between a browser and a Web server using the HTTP protocol. More recently, programmatic access to services over the Web, called Web services, has been the subject of intense activity and standardization efforts.[1-2]

In this new model, Web service providers use the Web Services Description Language (WSDL)[3] to describe the services they provide and how to invoke them. The service providers then register their services in a public service registry using universal description, discovery, and integration (UDDI).[4] Application programs discover services in the registry and obtain a URL for the WSDL file that describes the service. Then, the applications can invoke the services using the XML-based simple object access protocol (SOAP) in either asynchronous messaging or remote procedure call (RPC) mode.[1,5]

Figure 1 illustrates the concept of Web services and how the model differs from that of traditional access to a Web site. Traditional Web sites (Figure 1a) implement all components needed to carry out user transactions: user interface and navigation management, business logic, and access to persistent storage. Web services sites give users access to some or all of these services through programs that provide these services over the Web, as Figure 1b shows. In this case, the travel site uses Web services applications for airline booking, hotel reservation, and car rental reservation.

Issues in Web Services

Quality of service (QoS) is a combination of several qualities or properties of a service, such as:

- *Availability* is the percentage of time that a service is operating.
- *Security* properties include the existence and type of authentication mechanisms the service offers, confidentiality and data integrity of messages exchanged, nonrepudiation of requests or messages, and resilience to denial-of-service attacks.
- *Response time* is the time a service takes to respond to various types of requests. Response time is a function of load intensity, which can be measured in terms of arrival rates (such as requests per second) or number of concurrent requests. QoS takes into account not only the average response time, but also the percentile (95th percentile, for example) of the response time.
- *Throughput* is the rate at which a service can process requests. QoS measures can include the maximum throughput or a function that describes how throughput varies with load intensity.

The QoS measure is observed by Web services users. These users are not human beings but programs that send requests for services to Web service providers. QoS issues in Web services have to be evaluated from the perspective of the *providers* of Web services (such as the airline-booking Web service in Figure 1) and from the perspective of the *users* of these services (in this case, the travel agent site).

Service Provider Perspective

A service provider needs to consider many aspects of QoS. One of them is its *QoS policy*. Some Web services adopt a best-effort policy, which offers no guarantee that requests for services will be accepted (they could just be dropped in case of overload), and no guarantees on response time, throughput, or availability are provided. While this type of policy may be acceptable in some cases, it is totally unacceptable in others, especially when a Web service becomes an important part of an application composed of various Web services, as in the travel site example. In these cases, Web service providers may want longer-term relationships with users of their services. These relationships generate *service level agreements* (SLAs), legally binding contracts that establish bounds on various QoS metrics. Examples of conditions that an SLA may contain include these:

- The average response time for the `GetFlightAvailability` request should not exceed 0.5 seconds.
- Ninety-five percent of requests to the `BookFlight` service should complete in less than two seconds.
- The airline reservation Web service should be available at least 99.9 percent of the time.

It is not easy for Web service providers to manage their computational resources when the workload they see is unpredictable and exhibits high peak-to-average ratios in workload intensity. To ensure that all admitted requests obtain the level of service users expect, Web service providers might need to implement priority-based admission control mechanisms.[6] This might require rejecting low-priority requests. Web service providers might also offer multiple QoS levels differentiated by cost.

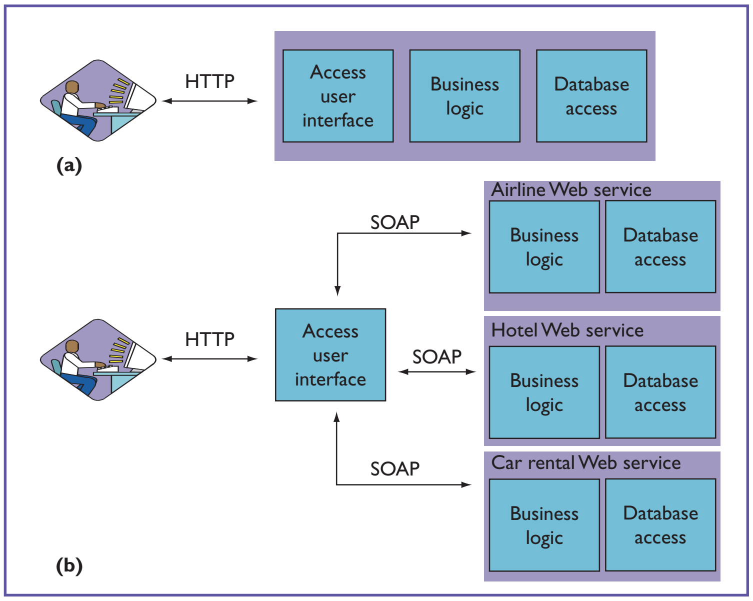

Figure 1. Implementing a site with Web services. (a) In a traditional travel site, the user accesses it through a browser, which communicates with the site via the HTTP protocol. The travel site implements the user interface, the business logic, and database access. (b) Under the Web services model, the travel site implements only the user interface, invoking airline, hotel, and car rental reservation services via SOAP.

Service User Perspective

The traditional travel site in Figure 1a did not need to rely on any third-party services to determine the quality of the services it provides to its customers. When Web services are used, as in Figure 1b, the QoS of the travel site may be strongly affected by the QoS of the various Web services it uses.

Figure 2 shows a Web service flow graph (WSFG) whose nodes are either Web sites or Web services. A directed edge between nodes a and b indicates that a uses the services of b. The label on the edge (a, b), called the *relative visit ratio*, is the average number of times node b is visited per visit to node a. So on average, each travel booking request to the travel site generates V_a requests to the airline Web service, V_h requests to the hotel Web service, and V_c requests to the car rental Web service.

We can now use an argument based on the Forced Flow Law[7] to establish an upper bound on the throughput X_{TA} of the travel site based on the throughputs of the three Web services it uses. For example, every request the travel site completes generates V_a requests on average to the airline Web service. The throughput X_a of the airline service therefore needs to be at least equal to $V_a \times X_{TA}$, since that service must be able to serve all requests it receives from the travel site as well as all requests coming from other sites that use its service. We can make the same kind of argument

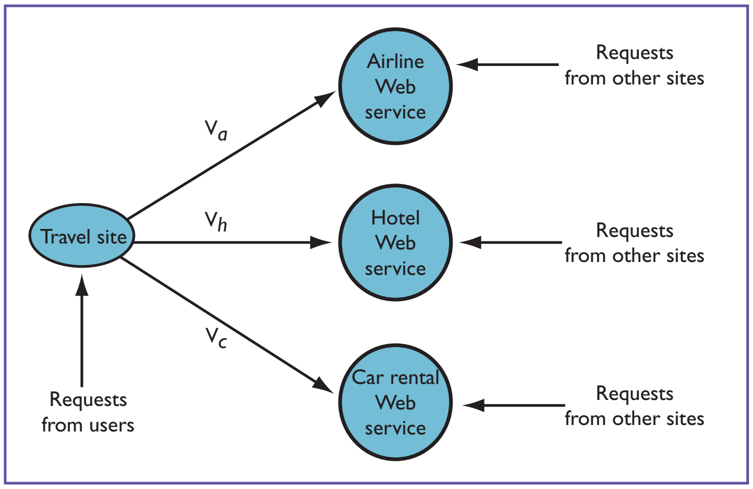

Figure 2. Web service flow graph. Arrows link the travel site to other Web services. The labels on the links indicate the average number of times a Web service is invoked per request to the travel site.

for all other Web services and write that

$$X_a \geq V_a \times X_{TA} \quad (1)$$

$$X_h \geq V_h \times X_{TA} \quad (2)$$

$$X_c \geq V_c \times X_{TA} \quad (3)$$

where X_a, $X_{h,}$, and X_c represent the throughputs of

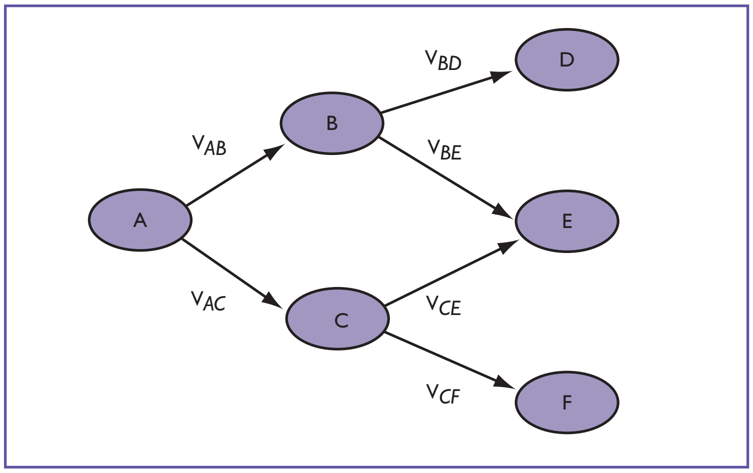

Figure 3. A more complex Web services flow graph. Web service A uses Web services B and C; B uses D and E; and C uses E and F.

the airline, hotel, and car rental Web services, respectively.

We can now combine Equations 1-3 to establish an upper bound on the throughput of the travel site:

$$X_{TA} \le \min\left\{\frac{X_a}{V_a}, \frac{X_b}{V_b}, \frac{X_c}{V_c}\right\}. \tag{4}$$

To see the usefulness of this equation, suppose that the throughput of the airline, hotel, and car rental Web services is 20 requests/sec, 15 requests/sec, and 10 requests/sec, respectively, and that on average, each travel site request will visit the airline Web service four times, the hotel Web service twice, and the car rental service only once. So, using Equation 4, we can say that

$$X_{TA} \le \min\left\{\frac{20}{4}, \frac{15}{2}, \frac{10}{1}\right\} = 5 \text{ requests/sec.} \tag{5}$$

Equation 5 says that in order for the travel site to increase the upper bound on its throughput, it would need to use a better airline Web service, because this is the Web service that limits the maximum throughput of the travel site. Alternatively, the travel site could try to reduce the number of times it has to invoke the airline Web service per transaction.

Of course, the performance of the various Web services the travel site uses depends not only on the load placed on them by the travel site, but also on the load coming from other sources.

The above computation can be generalized to any acyclic directed WSFG, such as the one in Figure 3, in which Web services B and C provide service to A. Web service B uses Web services D and E, and Web service C uses Web services E and F. An upper bound on the throughput of Web service A, in terms of the relative visit ratios and as a function of the throughputs of Web services B, C, D, E, and F, can be written as

$$X_A \le \min\left\{\begin{matrix}\dfrac{X_B}{V_{AB}}, \dfrac{X_C}{V_{AC}}, \dfrac{X_D}{V_{AB}V_{BD}}, \dfrac{X_F}{V_{AC}V_{CF}}, \\ \dfrac{X_E}{V_{AB}V_{BE}+V_{AC}V_{CE}}\end{matrix}\right\}. \tag{6}$$

Equation 6 easily generalizes to an arbitrary acyclic WSFG.

A site that uses Web services may need to consider *transactional* Web services. A transaction, in database parlance, is a sequence of actions that must be executed as a unit. For example, when a Web site sells a travel package to a customer, the site must confirm all components of the package (flights, hotels, and car rental reservations). It is common to require distributed transactions to have the ACID property in the presence of any type of site or network failures:

- *Atomicity:* Either all actions of a transaction are executed or none are.
- *Consistency:* Updates made by a transaction preserve its consistency constraints.
- *Isolation:* Concurrent transactions do not reflect the effects of a transaction until that transaction completes.
- *Durability:* The updates of committed transactions are never lost.

The two-phase commit protocol is used to guarantee the ACID property in distributed database systems. This protocol requires individual nodes to lock records while the transaction is in progress. This approach is not efficient for long-lived transactions, however, because of the inherent loss of concurrency, which degrades the QoS.

Another approach for dealing with long-lived transactions is based on *compensations*; that is, different services may commit locally, but should be ready to cancel their actions if conditions negotiated a priori require it. For example, the airline Web service may hold a seat for 48 hours and agree to accept an explicit cancellation (a compensating action) within that interval, or may decide to unilaterally cancel if a confirmation is not received within that period.[8] The hotel reservation Web service may reserve a room but accept cancellations up to 24 hours prior to the reserved date, afterward automatically charging for a first night.

Various organizations have proposed protocols to handle the demands of different kinds of transactions. The business transaction protocol, created by the Organization for the Advancement of Structured Information Systems, allows for two types of transactions: ACID (which it calls atoms) and non-ACID (which it calls cohesions).[8] IBM, Microsoft, and BEA systems just released a draft framework called WS-Coordination,[9] which provides protocols that coordinate the action of distributed applications. They also released WS-Transaction,[10] which offers two coordination types based on WS-Coordination: Atomic Transaction (AT) and Business Activities (BA). ATs are useful for short-lived transactions and BAs for long-lived ones.

Common Issues

Many providers compete to offer the same Web services, meaning that users can decide to select providers on the basis of the QoS to which they can commit. This implies that users and providers need to be able to engage in QoS negotiation.

The interaction between users and Web service providers occurs via XML-based SOAP messages. Therefore, messages tend to be longer than they would be otherwise and require XML parsers for interpretation at both sides. These two factors reduce the performance of third-party services.

Providers must monitor the load they receive from users and check whether the service they provide to them meets the agreed-upon SLAs. Users must also check on the quality of the service they obtain. QoS monitoring may be outsourced to QoS monitoring services such as the ones that monitor Web sites (such as www.keynote.com).

References

1. F. Curbera et al., "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing*, vol. 6, no. 2, Mar./Apr. 2002, pp. 86-93.
2. "Web Service Activity," World Wide Web Consortium, www.w3.org/2002/ws/.
3. R. Chinnici et al., "Web Services Description Language (WSDL) Version 1.2," World Wide Web Consortium, 2002, www.w3.org/TR/wsdl12/.
4. "Universal Description, Discovery and Integration," Organization for the Advancement of Structured Information Systems, 2002, www.uddi.org/specification.html.
5. D. Box et al., "Simple Object Access Protocol (SOAP) 1.1," W3C Note 08, World Wide Web Consortium, May 2000, www.w3.org/TR/SOAP/.
6. L. Cherkasova and P. Phaal, "Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites," *IEEE Trans. Computers*, vol. 51, no. 6, June 2002, pp. 669-685.
7. D.A. Menascé and V.A.F. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*, Prentice Hall, Upper Saddle River, N.J., 2002.
8. OASIS, "Business Transaction Protocol Primer," version 1.0, June 2002, www.oasis-open.org/committees/business-transactions/#documents.
9. F. Cabrera et al., "Web Services Coordination (WS-Coordination)," Aug. 2002, www.ibm.com/developerworks/library/ws-coor/.
10. F. Cabrera et al., "Web Services Transaction (WS-Transaction)," Aug. 2002, www.ibm.com/developerworks/ library/ws-transpec/.

Daniel Menascé is a professor of computer science, the co-director of the E-Center for E-Business, and the director of the MS in E-Commerce program at George Mason University. He received a PhD in computer science from UCLA. His published books include *Capacity Planning for Web Services: Metrics, Models, and Methods* and *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning.* He is a fellow of the ACM and a recipient of the A.A. Michelson award from the Computer Measurement Group.