



# Mapping Service-Level Agreements in Distributed Applications

Daniel A. Menascé, *George Mason University*

In a previous column, I discussed how to build distributed systems from quality-of-service (QoS)-aware software components.<sup>1</sup> I also described a design by which individual components can engage in QoS negotiation and perform admission control so that new incoming requests don't violate committed QoS requirements.<sup>2</sup> But what happens in a distributed application made up of several QoS-aware components? Moreover, what if the application has a global service-level agreement (SLA) for maximum end-to-end execution time? The problem I discuss here is how to determine which SLAs should be negotiated at the component level so that the global SLA is satisfied at the minimum possible cost.

## A Model for the Application

Let's say that a distributed application is made up of  $M$  software components. Each component  $m$  ( $m = 1, \dots, M$ ) is visited, on average,  $V_m$  times during the application's execution. When there is no congestion at component  $m$  — that is, when only one request is executing on it — the execution time of the application per visit to the component has the minimum possible value. Let  $E_m^{\min}$  be this value. Let  $E$  be the application's execution-time SLA, and let  $E_m^{\max}$  be the execution-time SLA per visit to component  $m$ : clearly,  $E_m^{\max} \geq E_m^{\min}$ . The relationship between the application SLA and the component-level SLAs is

$$E = \sum_{m=1}^M V_m \times E_m^{\max}. \quad (1)$$

If component  $m$  charges  $C_m$  dollars per visit for a guaranteed execution-time SLA equal to  $E_m^{\max}$  per visit, the total cost of executing the application for a set of component-level SLAs  $\{E_m^{\max}, m = 1, \dots, M\}$  is

$$C = \sum_{m=1}^M V_m \times C_m. \quad (2)$$

Figure 1 shows that the component-level cost  $C_m$  is typically a decreasing function of the component-level SLA. A component charges more as it provides more stringent execution-time guarantees.

## Optimization Problem

The SLA allocation problem we want to solve is this: Given a value  $E$  for the application-level execution-time SLA, the values  $\{E_m^{\min}, m = 1, \dots, M\}$  of the minimum execution time per component, the cost functions  $C_m$ , and the visit ratios  $V_m$ , find a set  $\{E_m^{\max}, m = 1, \dots, M\}$  of component-level execution-time SLAs that minimizes the total cost

$$C = \sum_{m=1}^M V_m \times C_m \quad (3)$$

such that

$$E = \sum_{m=1}^M V_m \times E_m^{\max} \quad (4a)$$

$$E_m^{\min} \leq E_m^{\max} \leq \frac{E}{V_m}, m = 1, \dots, M. \quad (4b)$$

The upper bound on Equation 4b comes from the fact that the maximum total time spent on component  $m$ , or  $V_m \times E_m^{\max}$ , cannot exceed the application-level SLA, or  $E$ .

## Numerical Example

To illustrate the SLA-optimization problem, consider an application that has a global execution-time SLA of 20 seconds uses four components. Table 1 shows the average number of visits per component and the minimum execution times; the cost functions for each component are

$$C_1(E_1^{\max}) = 1.5 \times e^{-0.08 \times E_1^{\max}}$$

$$C_2(E_2^{\max}) = 1.3 \times e^{-0.1 \times E_2^{\max}}$$

$$C_3(E_3^{\max}) = 2.0 \times e^{-0.07 \times E_3^{\max}}$$

$$C_4(E_4^{\max}) = 1.8 \times e^{-0.075 \times E_4^{\max}}$$

We can solve the optimal SLA mapping problem by using existing iterative search techniques such as Newton’s method for solving optimization problems.<sup>3</sup> Table 2 indicates one such solution. Column two shows the component-level SLAs per visit to each component, and column three shows the cost over all visits to a component – that is,  $V_m \times C_m$ . The total and minimum cost is US\$9.25, as indicated in the last row.

### A Heuristic Solution

We can solve the optimization problem described in the previous section quite efficiently with current iterative techniques. However, in some cases, solving the optimization problem might incur significant overhead compared with the application’s target execution time. We must therefore consider a fast-to-compute heuristic SLA mapping.

Consider this heuristic:

$$E_m^{\max} = E_m^{\min} + \frac{E - \sum_{m=1}^M V_m \times E_m^{\min}}{M \times V_m} \quad (5)$$

The interpretation of Equation 5 is simple: the component-level execution-time SLA is equal to that component’s minimum execution time  $E_m^{\min}$ , plus the fraction shown in the equation. This fraction’s numerator is equal to the total SLA to be allocated to all components once the minimum execution time per component is taken into account. This value is then divided by the number of components  $M$ . Because  $E_m^{\max}$  is defined as the SLA per visit to a component, we divide the resulting value by the number of vis-

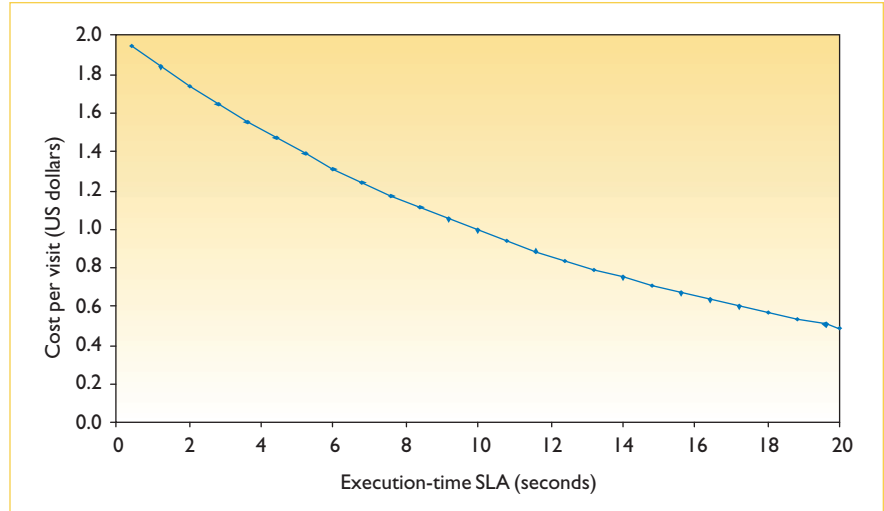


Figure 1. Component-level costs. The cost per visit to a component ( $C_m$ ) versus the component’s execution-time SLA ( $E_m^{\max}$ ) is typically a decreasing function. The cost drops as the guarantees for execution time become less strict.

Table 1. Input data for the service-level agreement optimization problem.

Component	$E_m^{\min}$	$V_m$
1	0.6	2.0
2	0.5	3.0
3	0.4	1.0
4	1.5	1.5

Table 2. Solutions to the service-level agreement optimization problem using Table 1’s input data.

Component	$E_m^{\max}$	$V_m \times C_m$ (in US dollars)
1	1.83	\$2.59
2	2.26	\$3.11
3	4.29	\$1.48
4	3.52	\$2.07
Total		\$9.25

its  $V_m$  to the component. Given Equation 5, the expression for  $E_m^{\max}$  satisfies Equation 1.

Table 3 (next page) compares optimal results with those obtained by using Equation 5 for four different values of the global SLA,  $E$ : 10, 20, 30, and 40 seconds, respectively. The input parameters and cost functions are the same as earlier, and column two shows

the component-level SLAs obtained by solving the optimization problem. Column three shows the component-level SLAs computed with Equation 5. Columns four and five give the optimal cost and the cost yielded by Equation 5, respectively. The last column indicates the percentage-relative error between the heuristic and the optimal solution. The table shows the heuristic

Table 3. Heuristic versus optimal solution.

Global SLA	Optimal SLA	SLA heuristic	Optimal cost	Cost heuristic	Relative error (percent)
10	0.60	1.33			
	1.09	0.99			
	2.61	1.86			
	1.95	1.68	10.356	10.367	0.107
20	1.83	2.58			
	2.26	1.82			
	4.29	4.36			
	3.52	3.34	9.258	9.266	0.085
30	3.23	3.83			
	3.38	2.65			
	5.89	6.86			
	5.01	5.01	8.277	8.291	0.160
40	4.63	5.08			
	4.50	3.49			
	7.49	9.36			
	6.50	6.68	7.400	7.425	0.331

**IEEE**  
Computer  
Society  
members

save  
**25%**

Not a member?  
Join online today!

on all  
conferences  
sponsored  
by the  
**IEEE**  
Computer Society

[www.computer.org/join](http://www.computer.org/join)

solution is very close to the optimal (less than 0.5 percent error) in all cases.

### Final Remarks

With the emergence of technologies such as Web services and other service-oriented technologies like Grid computing,<sup>4</sup> many future distributed applications will be built from components discovered on the fly. These components will be able to negotiate QoS goals with consumers, so QoS-aware middleware is a must. The specific functions of this type of middleware should include QoS-based component discovery, QoS negotiation at the component level, and QoS-level monitoring. I've discussed here the underlying models that could be used by such middleware to perform QoS negotiation to match application-level QoS requirements with component-level QoS commitments. However, there is still significant work to be done in this area of research, including how to describe and negotiate multidimensional QoS requirements that include execution time, availability, and throughput. ☐

### References

1. D.A. Menascé, "QoS-Aware Software Com-

ponents," *IEEE Internet Computing*, vol. 8, no. 2, 2004, pp. 91–93.

2. D.A. Menascé, H. Ruan, and H. Goma, "A Framework for QoS-Aware Software Components," *Proc. 2004 ACM Workshop on Software and Performance*, ACM Press, 2004, pp. 186–196.
3. C.T. Kelly, *Iterative Methods for Optimization*, SIAM Press, 1999.
4. D.A. Menascé and E. Casalicchio, "QoS in Grid Computing," *IEEE Internet Computing*, vol. 8, no. 4, 2004, pp. 85–87.

### Acknowledgments

Grant NMA501-03-1-2022 from the US National Geospatial-Intelligence Agency partially supports this work.

**Daniel A. Menascé** is a professor of computer science, codirector of the E-Center for E-Business, and director of the MS in E-Commerce program at George Mason University. He received a PhD in computer science from UCLA. Menascé is author of the books *Performance by Design*, *Capacity Planning for Web Services*, and *Scaling for E-Business* (Prentice Hall, 2004, 2002, and 2000). He is a fellow of the ACM and a recipient of the 2001 A.A. Michelson Award from the Computer Measurement Group. Contact him at [menasce@cs.gmu.edu](mailto:menasce@cs.gmu.edu).