System of Systems for Quality-of-Service Observation and Response in Cloud Computing Environments

Paul C. Hershey, *Senior Member, IEEE*, Shrisha Rao, *Senior Member, IEEE*, Charles B. Silio, Jr., *Life Senior Member, IEEE*, and Akshay Narayan, *Student Member, IEEE*

***Abstract*—As military, academic, and commercial computing systems evolve from autonomous entities that deliver computing products into network centric enterprise systems that deliver computing as a service, opportunities emerge to consolidate computing resources, software, and information through cloud computing. Along with these opportunities come challenges, particularly to service providers and operations centers that struggle to monitor and manage quality of service (QoS) for these services in order to meet customer service commitments. Traditional approaches fall short in addressing these challenges because they examine QoS from a limited perspective rather than from a system-of-systems (SoS) perspective applicable to a net-centric enterprise system in which any user from any location can share computing resources at any time. This paper presents a SoS approach to enable QoS monitoring, management, and response for enterprise systems that deliver computing as a service through a cloud computing environment. A concrete example is provided for application of this new SoS approach to a real-world scenario (viz., distributed denial of service). Simulated results confirm the efficacy of the approach.**

***Index Terms*—Cloud computing, distributed denial of service (DDoS), enterprise systems, information assurance, net centric, quality of service (QoS), security, service-oriented architecture (SOA), systems of systems (SoS).**

I. Introduction

As ECONOMIC pressure intensifies for network and enterprise operations centers, those responsible for these centers seek a method to lower costs in the presence of data overload. This dramatic increase in the quantity of data is a product of the evolution of complex net-centric enterprise systems over which multiple disparate users in dispersed locations share gigabytes, terabytes, or even petabytes of data at high speeds over production networks. Cloud computing provides one possible method to meet this challenge by reducing cost through shared computing resources while distributing data to multiple end users efficiently [1], [2]. However, complex systems that use cloud computing, such as shown in Fig. 1, are prone to failure and security compromise in five main areas: *computing performance* (e.g., latency and time delay experienced by a system when processing a request), *cloud reliability* (e.g., network connectivity), *economic goals* (e.g., interoperability between cloud providers), *compliance* (e.g., digital forensics to discern what happened, learn how to prevent incidents, and collect information for future actions), and *information security* (e.g., to protect the confidentiality and integrity of data and ensure data availability) [1], [2]. Present approaches to mitigate these issues are not sufficient to ensure quality of service (QoS) for end users [3].

Recent survey articles [4]–[7] summarize challenges for cloud computing to provide mechanisms to mitigate the issues listed above, the need for monitoring the various layers of the cloud computing environment, and the need to provide end-to-end solutions for the problems. The architectural frameworks considered in these excellent surveys are one dimensional and principally deal with the infrastructure, platform, and software as service layers of cloud computing. Federated clouds with cross-cloud connectivity provide an opportunity mentioned in [6] to support service-oriented computing, and QoS is discussed in [8] but is limited in application to the task of virtual machine provisioning in data centers rather than to end-user satisfiability. None treats the clouds from a system-of-systems point of view as is done here. The architectural model used here to support a service-oriented architecture (SOA) is multidimensional, extending to layers of business and governance as services, while reducing complexity by combining the traditional platform and applications layers into one software layer that provides overall Software as a Service (SaaS) to end users. Furthermore, specific locations and the use of software agents for monitoring and observation in this system of systems (SoS) are presented.

Motivation: QoS specification and monitoring for cloud services is a complex and challenging issue, one where there are few universal benchmarks or standards. While the usual quality metrics such as uptime and reliability may still be considered applicable in the context of cloud systems, it is less clear what the QoS parameters unique to such systems are and how they should be applied in specific contexts. A lack of common metrics to be applied to cloud offerings from various providers is also a barrier to standardization of cloud offerings from different providers. Thus, although "federated" clouds using services from different providers are thought to

Manuscript received November 16, 2012; revised September 19, 2013; accepted December 10, 2013.

P. C. Hershey is with Raytheon Intelligence, Information and Services, Dulles, VA, 20166 USA (e-mail: Paul_C_Hershey@raytheon.com).

S. Rao is with the International Institute of Information Technology-Bangalore, Bangalore 560100, India (e-mail: shrao@ieee.org).

C. B. Silio Jr., is with the University of Maryland, College Park, MD 20742 USA (e-mail: silio@umd.edu).

A. Narayan is with School of Computing, National University of Singapore, Singapore 117417 (e-mail: anarayan@comp.nus.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSYST.2013.2295961

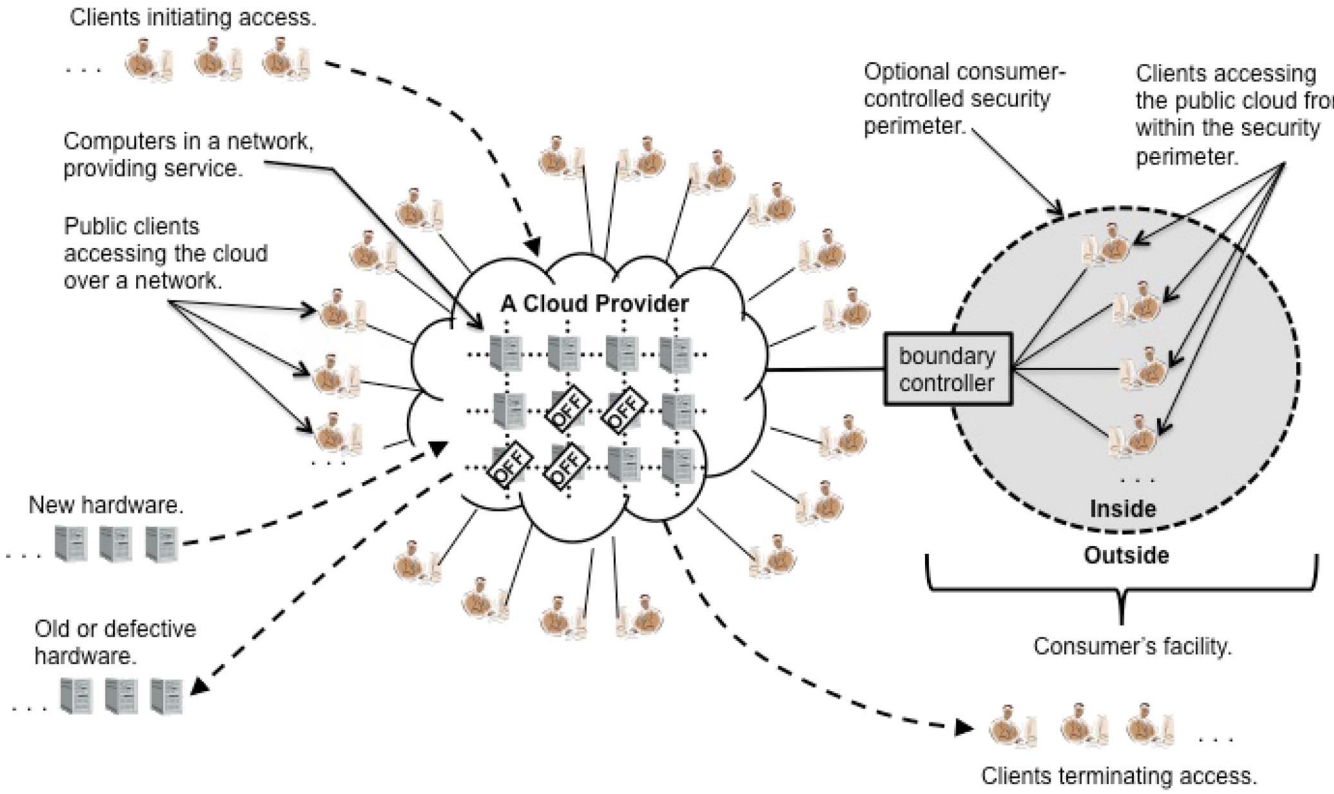

Fig. 1. Complex cloud computing environment [2].

be the way of the future, it is not yet possible to construct complex services using a mixture of services from different cloud providers [9]. QoS parameters are also often specified in semitechnical legalese, and it is a challenge to derive their appropriate mathematical formulations.

Another issue with QoS parameters and their evaluation is that there typically are multiple metrics, some positive (where a higher value means a better QoS, e.g., network bandwidth) and some negative (where a lower value means a better QoS, e.g., network latency), and it is far from obvious how, in general, the various parameters should be weighted and compared (particularly in cases where the metrics are incompatible or unavailable all at once). This, in fact, is an example of a challenging problem in multicriteria decision making, for which a method called simple additive weighting [10] has been applied in the past in the context of web services [11]. More recently, in the context of cloud computing, simple additive weighting and the analytic hierarchy process [12] have been used [13] to scale and compare QoS parameters, both positive and negative.

Proper resource allocations to different tiers of a cloud system are also known to be critical to improving the QoS, but existing approaches tend to be *ad hoc* and wasteful [14].

The approach presented in this paper introduces a SoS to provide a clear and concise view of QoS events within cloud computing environments that proactively informs enterprise operators of the state of the enterprise and, thereby, enables timely operator response to QoS problems. Section II provides a step-by-step description of the SoS approach; Section III provides the mathematical model for the QoS metrics considered in our work. Section IV describes the application of this approach to a cyber security scenario in which a complex cloud computing environment is subjected to a distributed denial-of-service (DDoS) attack. Section V details the experimental results obtained in the prototype system created to verify the SoS approach. Section VI presents benefits and conclusions.

TABLE I
SoS Characteristics

Structural	A SoS has a structure that comprises interdependent systems that integrate to form a higher order system, usually resulting in a hierarchy. This hierarchy can include monitoring and response at the highest-level system down to the smallest sub-component system (*i.e.* bit-level).
Coupling	The systems that comprise a SoS include coupling with respect to such areas as data, information, functions, state, and algorithm. A loss of any portion of the SoS will degrade the overall performance or capabilities of the higher order system; therefore, the systems are interdependent.
Behavioral	Integration of decisions and actions of systems occurs in the higher order system through governance in contrast to non-SoS where the sharing of information is the basis for collaboration.
Interoperable	Systems that comprise a SoS interface with one another and interoperate by design in contrast to non-SoS where systems are not designed to do so.

II. Approach

Step 1: *Define a SoS for monitoring, management, and response*. A SoS [15]–[18] possesses the characteristics shown in Table I. Fig. 2 presents a SoS comprising a system of multiple

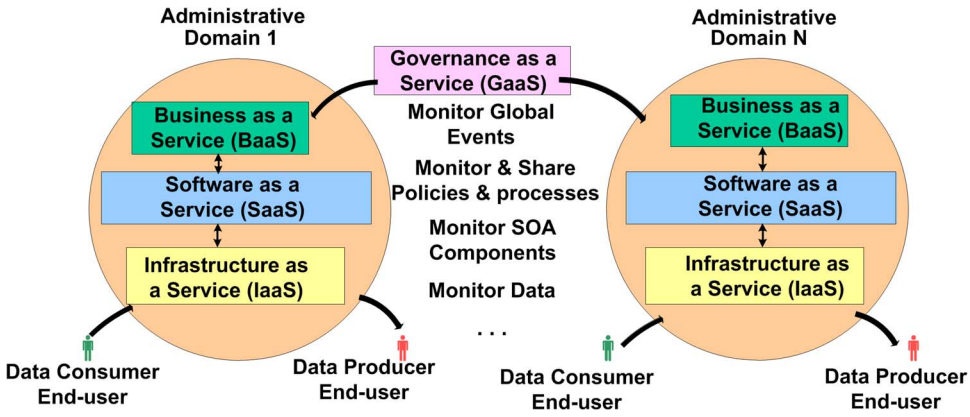

Fig. 2. Net-centric SOA-based SoS.

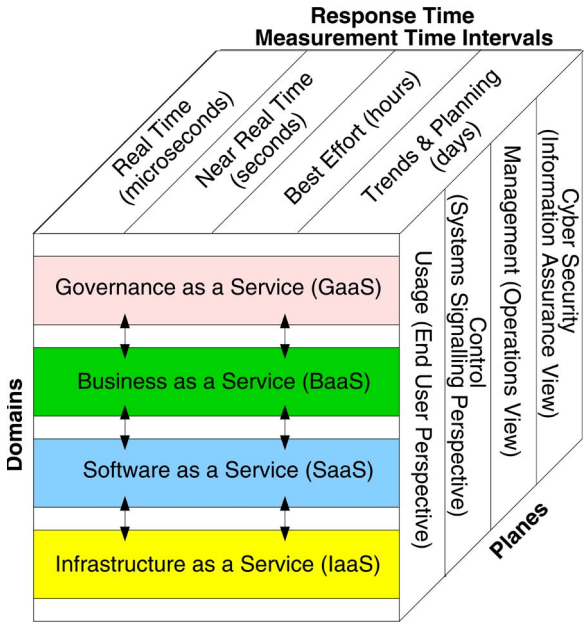

Fig. 3. EMMRA CC.

administrative domains operating within a SOA-based cloud computing system [19]. For the system depicted in Fig. 2, a single authority provides governance services to multiple heterogeneous administrative domains in which SOA-based applications enable business and collaboration services that support end users who are producing and consuming data using software and infrastructure services.

Step 2: *Derive framework for QoS monitoring, management, and response in cloud computing environments.* The Enterprise Monitoring, Management, and Response Architecture (EMMRA) for Cloud Computing Environments (EMMRA CC), as shown in Fig. 3, extends previous work [19] to provide structure from which to identify points within the administrative domains of Fig. 2 where key QoS metrics may be monitored and managed.

This framework is multidimensional to enable an end-to end view of the system where metric viewpoints may be located within traditional Open Systems Interconnection (OSI) layers (infrastructure through applications/software) and SOA-based layers (business and governance), as well as across these layers.

The X dimension (Response Time) defines time-based services based on measurement time intervals (MTIs) ranging from microseconds to days or beyond, depending on the time criticality of the information to the mission. In each domain, the term service refers to any program, algorithm, function, analysis technique, or monitoring activity that uses or interprets enterprise system information. A service can be a single individual or group of individuals (e.g., individuals doing problem determination or analysis), a program providing statistics, a piece of code performing network functions (such as load balancing), or another entity using network data. Different services in which network managers are interested require different time periods (i.e., MTIs) for data collection. For example, a critical mission in which lives may be at stake could require real-time (i.e., microsecond) monitoring, analysis, and response for QoS event problem determination. On the other end of the spectrum, a department store studying customer trends may be content to collect and analyze data on a daily basis as they plan for an upcoming sales event. Traditional performance measurement and analysis approaches do not have the flexibility to collect data and assess QoS performance in real time for the wide range of MTIs required by enterprise-level services. A need exists for a flexible data collection device that can collect information within complex enterprise systems for varying MTIs. EMMRA CC provides this flexibility and thereby meets critical QoS mission requirements for diverse missions with varying MTIs.

The Y dimension (Domains) detects and responds to enterprise events using similar techniques and instrumentation. A key contribution from this step is the extension of the domain dimension from Infrastructure as a Service (IaaS) and SaaS, where traditional techniques attempt to enforce QoS in cloud computing environments, to include Business as a Service (BaaS) and Governance as a service (GaaS) to meet the challenges of SOA-based net-centric enterprise systems. For BaaS, EMMRA CC focuses on monitoring business processes and managing these to ensure their uniform implementation among end users. For GaaS, EMMRA CC identifies ownership of governance services and enables the institution and enforcement of policies that influence enterprise-wide behavior.

The Z dimension (Planes) introduces structures that monitor and manage particular end-to-end events. Planes (namely, usage, control, management, and cyber security) encompass all domains and thereby provide a cross-domain solution that

enables enterprise-wide monitoring, management, and response. Planes also span multiple MTIs, enabling them to address multiple and diverse services. Implementing a plane-based approach within a SOA methodology is highly effective compared with existing domain-only QoS parameter evaluation techniques because the EMMRA CC approach fills the gaps between the domain monitoring layers. For example, the usage monitoring plane encompasses those activities that filter, collect, analyze, and disseminate information about the user data. The monitoring and management of this information could originate with an activity in the GaaS domain (e.g., dissemination, implementation, and assessment of operational polices), but this activity can then influence similar activities within the BaaS, SaaS, and IaaS domains. The usage monitoring plane provides this enterprise-wide view.

Step 3a: *Identify Cloud computing metrics.* To enable effective QoS monitoring, management, and response across and among multiple and diverse enterprise operations centers, we define relevant service-based QoS metric categories and metrics within those categories. The term service-based metric represents an object that has a name (i.e., metric name), a definition, a value (i.e., measure), an observation time period, a computing or collection method (i.e., measurement), and one or more threshold values for setting alarms (e.g., minor, major, and critical). In this paper, whenever the word metric is used, it refers to the metric name of the metric object. The term measure is a quantitative value of the metric object derived for the observation time interval. The term measurement is the computing or collection process or method of determining the value of the metric object.

To clarify the meaning of these definitions, consider the example of a toll bridge with two parameters of interest: the length of the bridge and the number of cars that have passed over the bridge. Let the service provider be the local department of transportation whose mission is to determine whether the capacity of the bridge is adequate or if a second bridge span will be required to handle the traffic at some time in the future. A common characteristic of metrics is that they change over time. The length of the bridge is static over time and does not change regardless of how often it is measured. Thus, this parameter is not a metric. By contrast, the number of cars that have passed over the bridge requires counting cars over an observation time interval. This observation time interval could be the last hour, the last day, or from the time that the bridge first opened. The number of cars counted can be reset to zero at the beginning of the observation time interval as required by the end user in order to best meet the purpose of collecting the traffic flow information, e.g., throughput analysis or capacity planning. For this example, the terminology number of cars that have passed over the bridge is a service-based metric. The value representing the counted number of cars per observation time interval provides the measure for the metric. The measurement method for the metric could be to use a sensor placed across the lanes of the bridge that would advance a counter each time a car passed over it.

This paper focuses on the QoS metrics categories of performance and security and their respective metrics, as shown in Table II. In Section V, we present results for the categories of Delay, Delay Variation, Throughput, Authentication, Authorization, and Certification and Accreditation.

TABLE II
METRICS CATEGORIES

Category	Metric
Performance	**Delay**
	Delay Variation
	Throughput
	Information Overhead
Security	**Authentication**
	Authorization
	Non-repudiation
	Integrity
	Information Availability
	Certification & Accreditation
	Physical Security

Delay is the elapsed time observed for a completed or on-going task and is caused by processing, queuing, and transmission of data. Examples of other terminology used interchangeably with delay include response time, round-trip time, and latency. Delay metrics identify when the cumulative effects of processing and transport hinder the end users ability to accomplish the mission. It is obvious to the end user and therefore influences customer satisfaction.

The comparison of delay for different observation time intervals is **Delay Variation**. Examples of delay variation include the following: the variation in application response time between peak and nonpeak hours, the edge-to-edge delay variation in the pattern of packet arrival events, and jitter. Delay variation identifies system instability that either presently prevents the end users from successfully executing their missions or that is a forewarning of upcoming problems that will do so.

Throughput metrics describe the amount of work completed over a time period [20]. Throughput metrics identify the level of work accomplished by the team, application, computer, and network. Proper interpretation of these metrics enhances productivity, efficiency, and resource allocation.

The **Authentication** metrics category includes metrics that confirm the identity of users, systems, or data sources. Authentication employs one or more mechanisms such as passwords, key exchange, digital certificates, and biometrics. Confirmation of entity identity is a fundamental requirement in establishing trust and confidence in the service and enabling other security functions.

User **Authorization** metrics report the success and failure of access to resources based on policy and permission levels. Authorization extends authentication—confirming an entity's identity—to define the entity's privileges (i.e., those functions that the entity can be trusted to perform). Authorization metrics enforce the principle of least privilege. By ensuring that entities are assigned the fewest privileges consistent with their mission, the overall service integrity is maintained, and mission effectiveness is enhanced.

Certification and Accreditation (C&A) is the comprehensive evaluation of the technical and nontechnical security features of an IT system and other safeguards, made in support of the accreditation process, to establish the extent to which a particular design and implementation meets a set of specified security requirements [21]. DoD requires C&A security

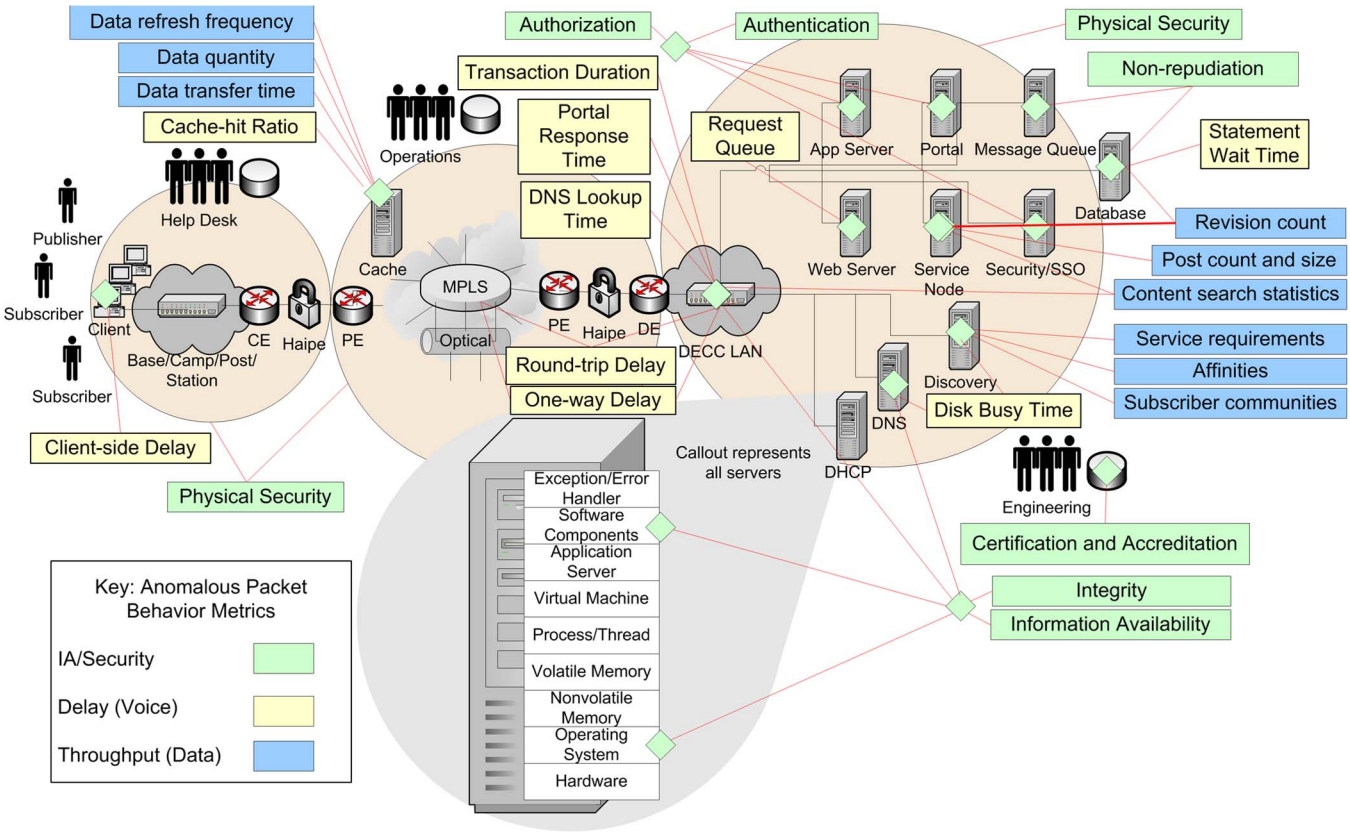

Fig. 4. Locations for IA/security, delay, and throughput metric observation.

through the DoD Information Technology Security Certification and Accreditation Process. Metrics within the C&A metrics category identify security risks and deficiencies, provide information to help ensure that steps are taken to correct these deficiencies and vulnerabilities, and provide information to help ensure the safeguarding of applications, networks, systems, data and information.

Step 3b: *Measuring performance metrics.* When considering heterogeneous resource types, having a baseline for measuring all performance and security metrics is nontrivial. In our work, we measure low-level independent performance metrics such as delay at each level of EMMRA CC. In the prototype of an online transaction processing application, we measure the response time at each of the layers. Specifically, we measure the following: 1) time taken for a database query to return the results (on the cloud instance hosting the database), 2) time taken by the application logic to execute (on the cloud instance hosting the application server), and 3) time taken for the data to be transmitted over the network between the application server and the database server. Throughput is measured as the number of transactions completed when viewed from the application perspective and hence is measured on the instance hosting the application server. A similar approach to measure performance metrics for SLA management has been used in earlier work [22].

Step 4: *Identify suitable locations within the cloud computing environment for metric detection.* This step identifies suitable locations to observe and collect the metrics in Table II. Fig. 4 shows these locations for an expanded set of metrics for security (green boxes), delay (yellow boxes), and throughput (blue boxes) for a representative cloud computing environment realized through a SOA-based net-centric enterprise system. This system includes four user communities: 1) end users shown with the client workstation, 2) help desk shown with the trouble management system database, 3) operations shown with the configuration management database, and 4) engineering shown with the project control and development tracking database. With respect to the system components, the end-user client machines that run applications over the base/camp/post/station network appear at the left side of the figure. The network edge appears as a Customer Edge (CE) router attached to a High Assurance IP Encryptor (HAIPE) device and a Provider Edge (PE) router, along with a cache. The core network includes information transported via Multiprotocol Label Switching (MPLS), Dense Wave-Division Multiplexing, and Synchronous Optical Network services. The network terminates at a Defense Enterprise Computing Center (DECC) with the connection from a second PE router to a HAIPE device and then to a DECC Edge (DE) router and a DECC LAN. To the right of the DE are the computing services to be shared through a cloud computing environment, including those for dynamic host configuration protocol (DHCP), Distributed Names System (DNS), Discovery, App Server, Portal, Message Queue, Web Server, Service Node, Security, and Database. As an example, we provide the rationale used to select the locations for IA/security metrics.

Authentication: Cyber security can be monitored at the Apps, Portal, and Security/SSO servers. For example, EMMRA CC agents can monitor Security Assertion Markup Language authentication assertions at the Security/SSO server. These

assertions facilitate the secure exchange of authentication information between systems regardless of their underlying security mechanisms.

Likewise, EMMRA CC agents can effectively monitor and respond to **Authorization** events from the Apps, Portal, and Security/SSO servers where they can monitor and respond to information such as need-to-know determination required to grant authorization to a resource. This is typically implemented in the DD Form 2875, System Authorization Access Request. For this example, EMMRA CC agents embedded within the Apps server would report the number of roles supported and the number of users assigned to each role, including the number added and the number removed over an observation period. The EMMRA CC agent would also monitor system resources access attempts to determine authorization failures.

Non-repudiation confirms that a transaction between two parties took place. Thus, a good location for EMMRA CC agents would be at the receiving end or a processing intermediary, such as the Message Queue in Fig. 4. Here, the EMMRA CC agents can observe confirmation of messages both from the sender to the agent and from the agent to the receiver.

A prime location to monitor and respond to **Integrity** events is the DNS server. For example, web applications use DNS name resolution to resolve IP addresses. An attack called DNS poisoning corrupts the DNS domain-to-IP address database such that requests are redirected from the legitimate server to the attackers' server for the purpose of presenting fraudulent content or collecting sensitive information. To prevent DNS integrity problems, EMMRA CC agents should audit all administrative updates to the master database and data transfer activity and provide a response so that operators configure DNS such that the end user never directly contacts the master database. EMMRA CC agents should search the DNS logs for particular events such as spikes in DNS traffic that could indicate a redirection. In addition, EMMRA CC agents should work with Intrusion Detection Systems, Intrusion Prevention Systems, anti-virus and configuration tracking software at the DNS servers to deter unauthorized changes to the server.

For **Information Availability** monitoring and response, EMMRA CC agents should be installed at the DECC LAN router and DHCP server to observe malicious traffic that could reduce information availability below its required threshold, thereby indicating spurious threat activity at individual servers. The commander Joint Task Force for Computer Network Defense can use this metric to set and track the Information Operations Condition (INFOCON) [23]. Based on the INFOCON status level, a Collection and Analysis (CA) node in the operations center can invoke countermeasures to uniformly heighten or reduce defensive posture, defend against computer network attacks, and mitigate sustained damage to the DoD information infrastructure.

Certification and Accreditation monitoring and response supports the DoD Information Assurance Certification and Accreditation Process Instruction [24]. EMMRA CC agents distributed within the engineering project control and development-tracking database can provide the relevant information to support ongoing certification and accreditation.

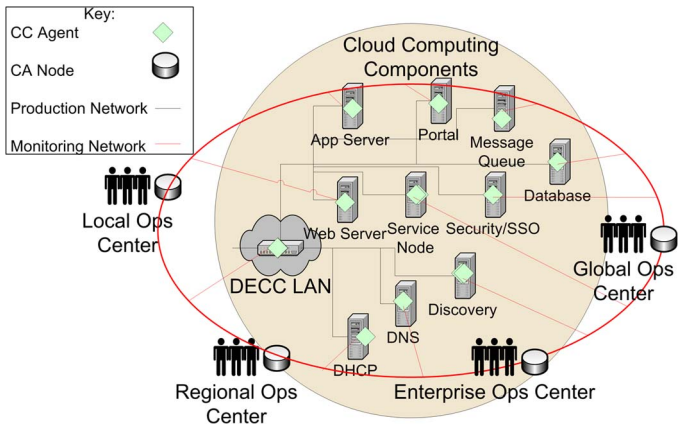

Fig. 5. Locations for EMMRA CC agents and CA nodes.

The DoD requires Physical Security for every enclave of a net-centric enterprise system. Thus, each user community could benefit from embedded EMMRA CC agents located at perimeter devices that indicate destruction, theft, or sabotage from unauthorized access to facilities, equipment, material, data, information, or documents.

Step 5: *Identify potential implementation schemes from which to collect and analyze the cloud computing QoS metrics.* One possible scheme presented in this paper embeds EMMRA CC agents within multiple diverse cloud computing components where they can continuously monitor the enterprise system for QoS metrics associated with cloud computing environments (e.g., delay, throughput, and security metrics). These agents communicate over an out-of-band (OOB) monitoring network to EMMRA CC CA nodes that are located at local, regional, enterprise, and global operations centers, as shown in Fig. 5.

III. System Model

Here, we describe the mathematical model for the QoS metrics considered in our work. Furthermore, we describe the possible response actions to be considered on a QoS breach. We describe the individual metric classes considered in Table II.

A. Performance

The *performance QoS metrics* are additive in the numerical sense. The SoS view from the top-level domain in Fig. 3 (i.e., GaaS) perceives delay as a sum of the delays experienced in the other lower domain levels of the cloud. This is also dependent on the infrastructure components used to provide the service. Hence, we must include the component-induced performance degradation. The delay metric can be represented as

$$D_{\mathrm{SoS}} = p_1 \cdot D_G + p_2 \cdot D_B + p_3 \cdot D_S + p_4 \cdot D_I \qquad (1)$$

where each p_i is a parameter that is dependent on the infrastructure component used. D_j is the delay experienced in each layer j in EMMRA, where the specific letter for j is the domain (i.e., Governance, Business, Software, and Infrastructure).

We define *throughput* at the system level as the number of transactions that are completed per unit time. Throughput can

be visualized at different levels. Throughput at the GaaS level of EMMRA is on the order of a few days. This must be captured in a different scale. However, the throughput at the lower levels of EMMRA is multiplicative in nature. The throughput at every level is a function of the throughput at a lower level in EMMRA. Hence, we have

$$\begin{aligned} T_I &= n \times \text{Transaction Throughput} \\ T_S &= m \times T_I \\ T_B &= q \times T_S. \end{aligned} \tag{2}$$

Here, m, n, and q are the numbers of transactions at the lower domain needed to complete a transaction in the higher domain. Additionally, at each level, throughput is additive in nature. For example, at the software layer, if there are p operations independent of each other (which may or may not require services from the infrastructure layer), then the throughput of the software layer is the sum total of the number of operations completed per unit time.

B. Security

Security can be thought of as a functional requirement of the system. It comprises authentication and authorization using certificates and accreditation.

The *authentication* QoS metric is the logical conjunction at each level in EMMRA. The users' access to the system ceases at the level authentication fails. Hence, the SoS view of authentication is a logical AND of the authentications at various levels in EMMRA. Security can be viewed as a top-down metric, i.e.,

$$A_{\text{SoS}} = A_G \wedge A_B \wedge A_S \wedge A_I. \tag{3}$$

The lower level EMMRA components have to be kept secure from the end user. A user at the top level can obtain service from the bottom levels but is not authorized to access the components directly. Only specific personnel are allowed access to the lower level components (viz., administrators). Hence, in order to obtain access to lower level components, the user needs to be authenticated at the top level. This is modeled in (3).

Authorization, however, is a bottom-up metric and is applicable at each level. User access to the service at any layer of EMMRA is subject to authorization. The authorization is such that the least privilege is granted sufficient to accomplish the operation. Authorization is applicable at each level in EMMRA CC. For example, in a banking application, an administrator is not authorized to access account details of the customer of the bank. Authorization at the IaaS level can be represented as

$$\text{Auth}_I = \min \left\{ \bigcap_{i \in \text{Set of actions}} p_i \right\} \tag{4}$$

where p_i is the permission to perform action i at the IaaS level. The min operator is used here as an indication of the least privilege level that is granted to the user. Authorization is defined similarly for the rest of the levels in EMMRA CC. The SoS view of authorization can be obtained using methods such as linear logic [25].

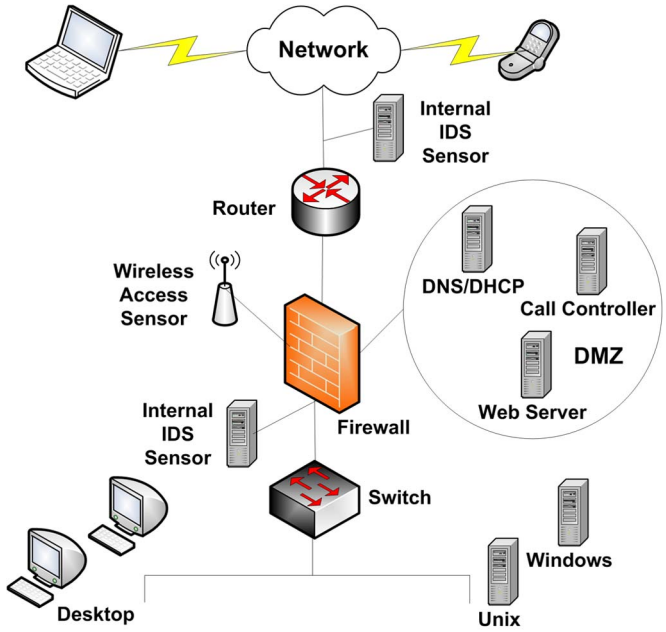

Fig. 6. Security enclave.

IV. Real-World Application Scenarios

This section describes a real-world scenario (i.e., use case) in which the EMMRA CC SoS approach is applied to a complex cloud computing environment that is exposed to a cyber security threat (i.e., DDoS) [26]. In this scenario, the Cyber Security Plane is used to observe cyber security threats across all domains in order to detect and enable proactive response to a DDoS security breach within any of these domains that could compromise the transactions and cause potentially devastating consequences to the end user.

DDoS cyber attacks are on the rise with ever increasing sophistication and costs to victims. A DDoS attack attempts to exhaust the victims' resources, including network bandwidth, computing power, and operating system data structures. To launch a DDoS attack, malicious users first build a network of computers that they will use to produce the volume of traffic needed to deny services to computer users. To create this attack network, attackers discover vulnerable sites or hosts on the network that are then exploited by attackers who install new programs (known as *attack tools*) on the compromised hosts of the attack network. The hosts that are running these attack tools are known as *zombies*, and they can carry out any attack under the control of the attacker. Many zombies together form an *army* that comprises both *master zombies* and *slave zombies*. The attacker coordinates and orders master zombies, and they, in turn, coordinate and trigger slave zombies that send a large volume of packets to the victim, flooding its system with useless load and exhausting its resources.

In order to devise and defend against DDoS attacks in a cloud computing environment, we segregate this system into security enclaves, i.e., a collection of computing environments connected by one or more internal networks under the control of a single authority and security policy. Fig. 6 depicts the main components of a security enclave [27], [28], including a Demilitarized Zone (DMZ), which is defined as a security

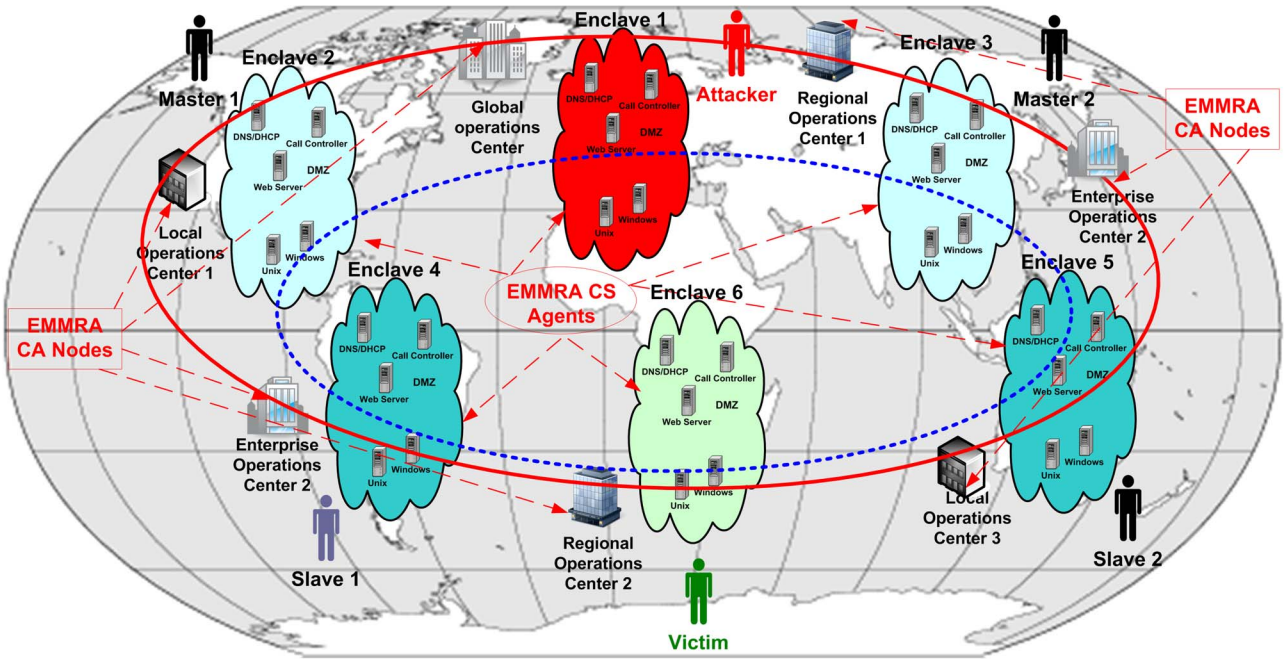

Fig. 7. Operational view of EMMRA CC agents and CA node deployment to counter DDoS attacks.

boundary created between two security policy-enforcing components. Note that the existence of a security enclave, while it reduces the chances and severity of a DDoS attack, does not guarantee attack protection.

Fig. 7 depicts a scenario in which a cloud computing system comprises six security enclaves that communicate over a global network. In this scenario, an attacker residing in enclave 1 launches a DDoS attack using master zombies (i.e., bots) in enclaves 2 and 3 and slave zombies in enclaves 4 and 5 to initiate and sustain the DDoS attack on a victim in enclave 6.

The EMMRA CC nodes are distributed among local, regional, enterprise, and global operations centers, as described Section II, and communicate in an OOB network, as shown in Fig. 7.

This distribution of CA nodes corresponds to the domain layers of the EMMRA CC framework in Fig. 3 and enables DDoS detection and response for infrastructure, software, business, and governance layers of SOA-enabled cloud computing environments [26]. EMMRA CA nodes are implemented through software or by programmable hardware devices. They are initially programmed to collect and analyze metrics from CC agents that indicate anomalous data relevant to their original mission, e.g., excessive delay on voice-based links used for military operations collaboration. Once anomalous behavior is detected, then the CA node conducts correlation and analyses to isolate the threat and provide response, viz., shut down the affected links and redistribute traffic so that the users do not lose QoS.

V. Results

Here, we describe the experimental setup and our insights into the QoS in a SoS setup. We built a prototype of an online transaction processing application. While each layer of the application was hosted on a different cloud instance, the cloud itself was deployed in a private network. This allows us to demonstrate the dependence of performance metrics on the component-induced degradation or, in our example, a lack of such degradation. For example, because the setup was deployed in a private gigabit Ethernet supported network, we can hardly feel the presence of a network-induced delay in the transactions. Network-induced delay is a significant component of induced degradation in enterprise systems that have a global presence. Our setup emulates the setup followed by many of the online transaction processing applications in the enterprise environment. The descriptive statistics of the recorded delay for application instance are shown in Table III. With a large standard deviation, the delay variations are large. In addition, we note that the data are skewed with a huge difference between minimum and maximum delays. These data are similar to real-world applications [29].

TABLE III
Descriptive Statistics of Application Instance

Property	Application Instance
Mean	481.17
Median	407.98
Mode	411.33
Standard Deviation	316.75
Minimum	1.29
Maximum	997.74

A. Delay

As discussed in Section II, Step 3, delay is one of the primary parameters considered while evaluating the QoS of a system and can be introduced into the system because of various reasons, including misconfiguration of the software stack, blocked ports in the network, and data processing delays. In our work,

TABLE IV
DELAY RECORDED IN TEN SAMPLE TRANSACTIONS

Application	Database	Network	Total
262	1323	16.4	1601.4
200	1335	48.9	1583.9
101	763	4.5	868.5
261	838	17.5	1116.5
216	1329	23.0	1568.0
260	1350	0.8	1610.8
249	1231	15.4	1495.4
297	1341	41.6	1679.6
251	1171	94.0	1516.0
249	1344	14.3	1607.3

All delay measurements are in ms

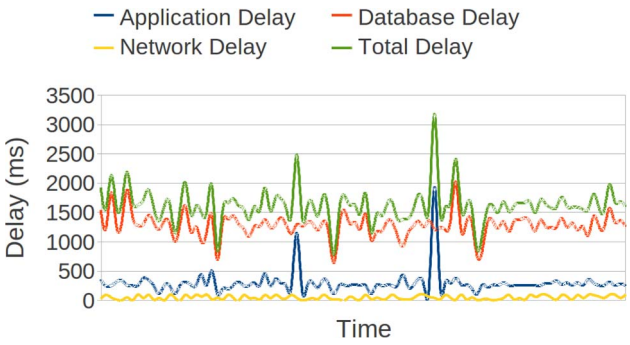

Fig. 8. Variation in delay recorded over time.

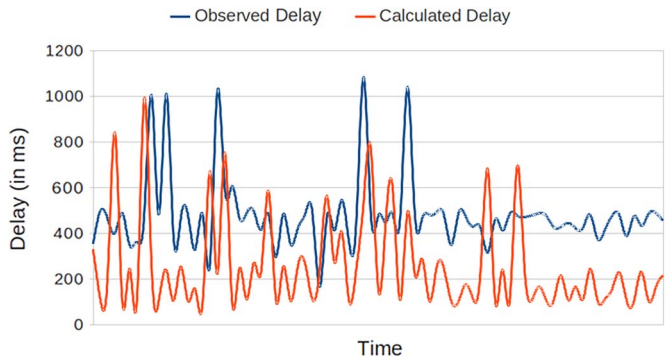

Fig. 9. Variation in observed delay and calculated delay.

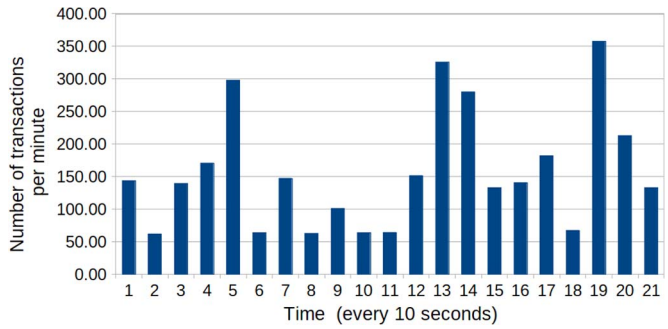

Fig. 10. Throughput: Number of transactions per minute.

we monitor delay at various levels of EMMRA CC. We notice that the delay perceived at the business or governance level in our work is the sum total of delay experienced at every level of EMMRA. This follows the relation established in (1). Table IV indicates that the SoS perspective of delay metric is additive in nature, as discussed in Section III.

Monitoring and characterizing delay helps determine the necessary configuration for a particular type of application stack to be deployed on the cloud. Delay modeling in Section III is an attempt to achieve the same. The experimental evidence provided here supports the model defined earlier.

Although characterization of delay in a deployment is advantageous, it has been noted that that delay is not a constant in the system. The following section describes the variation in delay and its implications.

B. Variability in Delay

Although delay is an important performance metric, characterizing and monitoring delay alone is not sufficient in a system. Due to various factors, the delay in the system varies with time and operation being performed. For instance, when processing a data stream, the delay may vary based on the size and type of data being processed. In addition, there are various component-induced factors that affect delay in the system. As a simple example, consider Ethernet cards manufactured by different manufacturers, although the cards have same rating, owing to the raw material used or difference in manufacturing technologies, the delay induced by these cards may be different on deployment.

Observing the delay variation over time (see Fig. 8) helps to identify a particular set of operations that cause maximum delay in the setup. For example, a regular pattern in spikes clearly indicates a set of operations that cause repeated delay in the system. Such operations can be isolated, and remedial actions can be suggested to the users to avert a QoS breach. Another relevant variation in delay is the variation in the observed delay and the delay computed using the modeling in (1) (without the correction factors p_i). This variation accounts for the component-induced performance degradation due to which delay is induced in the system. In Fig. 9, we notice this variation in the observed delay and the computed delay value. To account for this component-induced performance degradation, we need to include some correction factors in delay modeling, and hence, it is apt to have p_i at each level in EMMRA CC. It may be necessary to perform a calibration run to determine the accurate values of p_i in a particular setup.

C. Throughput

Throughput is an indicator of the overall system performance. It is desired to have a high-throughput system in the cloud. In our work, we measured throughput at the application and the database layers. The overall system throughput is represented in Fig. 10. Like delay, throughput is a variant in the system. Analyzing the throughput variation pattern allows us to have an insight into the behavior of the deployment (physical and software components).

Our application comprises two phases: the first phase is independent of the database transactions, whereas the second phase needs three database transactions. We tabulated the findings independently and presented them in Table V. Since all these operations are at the same layer in EMMRA CC, we notice that throughput is additive in nature. The factor in the

TABLE V
THROUGHPUT OF TEN SAMPLE TRANSACTIONS

Application Phase 1	Application Phase 2	Application Throughput	Database Throughput	Factor	Total Throughput
(The throughput values recorded here are number of transactions per second)					
1.272	2.545	3.817	0.756	3.000	4.573
1.667	3.333	5.000	0.850	3.000	5.850
1.967	3.934	5.901	1.311	3.000	7.212
1.944	3.888	5.831	1.193	3.000	7.025
1.543	3.086	4.630	0.800	3.000	5.430
1.282	2.564	3.846	0.741	3.000	4.587
1.339	2.677	4.016	0.812	3.000	4.828
1.122	2.245	3.367	0.746	3.000	4.113
1.328	2.656	3.984	0.854	3.000	4.838
1.339	2.677	4.016	0.744	3.000	4.760
1.323	2.646	3.968	0.729	3.000	4.697
1.339	2.677	4.016	0.713	3.000	4.729

table represents the number of database transactions needed to complete phase 2 of our application. The total throughput shown in Table V follows the additive rule presented in Section III.

VI. CONCLUSION

The new approach presented in this paper enables cloud computing service providers and operations centers to meet committed customer QoS levels using a trusted QoS metric collection and analysis implementation scheme that extends traditional monitoring, management, and response for IaaS and SaaS to a complete SOA stack that includes business logic (BaaS) and governance (GaaS). This paper includes real-world scenarios that describe the applications of this approach to voice and data systems for performance metrics and to DDoS for security metrics. Next steps include simulating these scenarios to quantify the effectiveness of this approach with respect to operations center response time to restore QoS in the presence of anomalous enterprise events.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," Natl. Inst. Standards Technol. (NIST), U.S. Dept. of Commerce, Gaithersburg, MD, USA, NIST Special Publication 800-145, Sep. 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] L. Badger, T. Grance, R. Patt-Corner, and J. Voas, "Cloud computing synopsis and recommendations," Natl. Inst. Standards Technol. (NIST), U.S. Dept. of Commerce, Gaithersburg, MD, USA, NIST Special Publication 800-146, May 2012. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf

[3] M. F. Mithani, M. Salsburg, and S. Rao, "A decision support system for moving workloads to public clouds," in *Proc. Annu. Int. Conf. CCV*, Singapore, May 2010, pp. 3–9.

[4] J. Spring, "Monitoring cloud computing by layer, Part 1," *IEEE Security Privacy*, vol. 9, no. 2, pp. 66–68, Mar./Apr. 2011.

[5] J. Spring, "Monitoring cloud computing by layer, Part 2," *IEEE Security Privacy*, vol. 9, no. 3, pp. 52–55, May/Jun. 2011.

[6] Y. Wei and M. Blake, "Service-oriented computing and cloud computing: Challenges and opportunities," *IEEE Internet Comput.*, vol. 14, no. 6, pp. 72–75, Nov./Dec. 2010.

[7] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, May 2010.

[8] R. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and QoS in cloud computing environments," in *Proc. ICPP*, 2011, pp. 295–304.

[9] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Muoz, and G. Tofetti, "Reservoir—When one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44–51, Mar. 2011.

[10] T. Gwo-Hshiung, G. H. Tzeng, and J.-J. Huang, *Multiple Attribute Decision Making: Methods and Applications*. Boca Raton, FL, USA: CRC Press, Jun. 2011.

[11] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[12] T. L. Saaty, *Multicriteria Decision Making: The Analytic Hierarchy Process: Planning, Priority Setting Resource Allocation*. Pittsburgh, PA, USA: RWS Publications, 1990.

[13] A. S. Prasad and S. Rao, "A mechanism design approach to resource procurement in cloud computing," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 17–30, Jan. 2014.

[14] M. F. Mithani and S. Rao, "Improving resource allocation in multi-tier cloud systems," in *Proc. 6th Annu. IEEE Int. SysCon*, Vancouver, BC, Canada, Mar. 2012, pp. 356–361.

[15] G. A. Lewis, E. Morris, P. Place, S. Simanta, and D. B. Smith, "Requirements engineering for systems of systems," in *Proc. 3rd Annu. IEEE Int. SysCon*, Mar. 2009, pp. 247–252.

[16] S. M. White, "Modeling a system of systems to analyze requirements," in *Proc. 3rd Annu. IEEE Int. SysCon*, Mar. 2009, pp. 83–89.

[17] *Defense Acquisition Guidebook (DAG)*, Jan. 2012. [Online]. Available: https://dag.dau.mil/Pages/Default.aspx

[18] *Systems Engineering Guide for Systems of Systems, Version 1.0*, ser. OUSD (A & T) SSE, Aug. 2008.

[19] P. Hershey and D. Runyon, "SOA monitoring for enterprise computing systems," in *Proc. 11th Int. IEEE EDOC Conf.*, Oct. 2007, pp. 443–450.

[20] P. J. Denning and J. P. Buzen, "The operational analysis of queueing network models," *ACM Comput. Surv.*, vol. 10, no. 3, pp. 225–261, Sep. 1978.

[21] *DoD instruction 5200.40: DoD Information Technology Security Certification and Accreditation Process*, Dec. 1997. [Online]. Available: http://csrc.nist.gov/groups/SMA/fasp/documents/c&a/DLABSP/i520040p.pdf

[22] V. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level metrics to high level SLAs—LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," in *Proc. 2010 Int. Conf. HPCS*, 2010, pp. 48–54.

[23] *DoD instruction 5200.80: Security of DoD Installations and Resources, 2005*. [Online]. Available: http://www.dtic.mil/whs/directives/corres/pdf/520008p.pdf

[24] IETF, "A one-way delay metric for IPPM," Fremont, CA, USA, RFC2679. [Online]. Available: http://www.ietf.org/rfc/rfc2679.txt

[25] J.-Y. Girard, "Linear logic," *Theoretical Comput. Sci.*, vol. 50, no. 1, pp. 1–101, 1987.

[26] P. Hershey and C. Silio, "Procedure for detection of and response to distributed denial of service cyber attacks on complex enterprise systems," in *Proc. 6th Annu. IEEE Int. SysCon*, Mar. 2012, pp. 85–90.

[27] *Network Infrastructure Technology Overview. Version 8, Release 5*, Apr. 27, 2012, (ArchiveEntry=U_Network_V8R5_Overview.pdf). [Online]. Available: http://goo.gl/Ja8Ajk

[28] *Enclave Security Technical Implementation Guide*, Jan. 3, 2011, (ArchiveEntry=U_Enclave_V4R3_STIG.pdf). [Online]. Available: http://goo.gl/Xhy4L2

[29] A. Iyengar, M. Squillante, and L. Zhang, "Analysis and characterization of large-scale web server access patterns and performance," *World Wide Web*, vol. 2, no. 1/2, pp. 85–100, Jun. 1999.

Paul C. Hershey (S'80–M'84–SM'99) received the A.B. degree in mathematics from the College of William and Mary, Williamsburg, VA, USA, and the Ph.D. and M.S. degrees in electrical engineering from the University of Maryland, College Park, MD, USA.

He is currently a Senior Engineering Fellow (with honors) and the Chief Engineer of Global Hawk Ground Segment programs at Raytheon Intelligence, Information and Services, Dulles, VA. He is an Adjunct Professor with George Washington University, Washington, DC, USA, where he serves on the Curriculum Advisory Board. He has authored or coauthored more than 45 technical articles and has published 28 patents (issued) with six additional patents filed and pending. His current research focuses on real-time information collection and analysis, data to decision, autonomous systems, cloud computing, and cyber security.

Shrisha Rao (M'08–SM'13) received the M.S. degree in logic and computation from Carnegie Mellon University, Pittsburgh, PA, USA, and the Ph.D. degree in computer science from the University of Iowa, Iowa City, IA, USA.

He is currently an Associate Professor with the International Institute of Information Technology-Bangalore, Bangalore, India, a graduate school of information technology. His research interests are in distributed computing, specifically algorithms and approaches for concurrent and distributed systems, and include solar energy and microgrids, cloud computing, energy-aware computing ("green IT"), and demand-side resource management.

Dr. Rao is a member of the IEEE Computer Society, the Association for Computing Machinery, the American Mathematical Society, and the Computer Society of India.

Charles B. Silio, Jr. (S'62–M'72–SM'89–LSM'09) received the B.S.E.E., M.S.E.E., and Ph.D. degrees in electrical engineering from the University of Notre Dame, Notre Dame, IN, USA.

He is currently an Associate Professor of electrical and computer engineering with the University of Maryland, College Park, MD, USA. His research interests include performance evaluation and reliability of computer networks.

Prof. Silio is a member of Eta Kappa Nu, Tau Beta Pi, and Sigma Xi and a retired Lieutenant Colonel of the U.S. Army. He served as an IEEE Computer Society Treasurer, chaired its technical committee on multiple-valued logic, and has been an NRC Research Associate at the Naval Postgraduate School and the Army Research Laboratory.

Akshay Narayan (S'13) received the M.Tech. degree in information technology from the International Institute of Information Technology-Bangalore, Bangalore, India. He is currently working toward the Ph.D. degree in the School of Computing, National University of Singapore, Singapore.

He is currently working on sequential decision making. His research interests include artificial intelligence and its application to cloud computing.

Mr. Narayan is a Student Member of the Association for Computing Machinery.