A probabilistic model for quantifying the resilience of networked systems

C. Queiroz
S. K. Garg
Z. Tari

Resilience is an important aspect of computing systems. Previous work on resilience has often focused on the design and architectural aspects of such systems, and not on the quantification of resilience. In addition, quantification is often restricted to a limited portion of the system. In networked systems, where multiple heterogeneous components interact in a complex manner, resilience quantification becomes a nontrivial problem. This paper proposes a model for quantifying resilience on the basis of the interdependencies of services and their adaptation. It combines performance and adaptability metrics to compute resilience of individual services that are then applied to a Markov network that computes the overall system resilience. The adaptation metric, here called adaptivity, computes how often the service adapts and evaluates the efficiency of such adaptations in terms of performance improvement. This paper also presents an evaluation that considers critical infrastructure systems.

Introduction

Resilience is considered an important aspect of computing systems. It becomes even more essential for networked systems that are responsible for operations that affect human life. For instance, a nonscheduled shutdown of power transmission lines may produce problems ranging from financial disasters to ultimately loss of lives. Likewise, a disaster-management system that is used for coordinating a disaster rescue cannot afford to have critical components unavailable, as they will affect the usefulness of the system, and consequently, this may impact the outcome of the rescue operation.

Computing systems can be affected (and compromised) in many different ways. Avizienis et al. [1] described a taxonomy of threats that may affect systems during their lifetime. They were characterized as three types: faults, errors, and failures. Faults were classified as malicious and non-malicious, where malicious faults are related to malicious attacks.

Previous work on improving resilience of systems by considering effects of such threats mostly focused on the design, architectural aspects, and performance evaluation. Proposed designs of resilient systems often aimed to provide mechanisms that allow the detection and treatment of various types of failures and faults at the design level itself.

Models proposed in the context of *dependability* [1] can be directly applied to resilience. Dependability, an umbrella term for disciplines such as reliability, survivability, fault tolerance, and resilience, has a long and rich history of research on techniques for quantitative and model-based evaluation of dependable computing systems [2, 3]. Nicol et al. [4] presented a survey on model-based evaluation techniques to quantify dependability and security. Many of the models presented could also be applied in terms of system resilience. However, often the faults that are addressed by these dependable models are clearly defined and statistically predictable. In contrast, statistical predictability of fail rates in systems subject to malicious attacks is arguably more difficult.

Even though other types of threats (faults, errors, and failures) are important, they have been exhaustively studied in previous work [5–7]. In this paper, we are interested in designing quantification models for resilience that consider faults that cannot be statistically predicable, for instance, malicious attacks. Most of the work on malicious attacks has involved detection and prevention. The goal of the

Digital Object Identifier: 10.1147/JRD.2013.2259433

proposed model is not the prevention, identification, or prediction of malicious attacks; rather, the model focuses on the consequences of such threats.

We recognize that design, architecture, and performance-evaluation techniques are important to improve the resilience of computing systems. However, given the complexity of some computing systems—such as critical infrastructure systems (consisting of multiple interdependent subsystems) and an environment where several stakeholders interact with one another—it is crucial to consider future consequences of undesired events such as faults, failures, or errors. The approach that includes such considerations can not only help in estimating extra protection or required redundancy, but also it can provide resilience against the uncertainty of such threats.

One of the aspects that the proposed resilience model considers is adaptation. Adaptation involves the capacity of adjusting system behavior for maintaining system functionality. It is a key attribute for the resilience of any system. Consider the example of biological organisms that continue to adapt to the environment they live in. Adaptation is the key for their survival. For intelligent creatures such as human beings, the adaptation becomes even more interesting. Humans learn from their past experience and use their experience to adapt to new situations and environments that they face at a particular moment. The importance of adaptation should not be different for computing systems.

Over the years, researchers in academia and industry have progressively addressed the needs for adaptation in computing systems. More recently, the attention has been shifted to evaluating adaptation. Although many technologists have designed and implemented adaptation features for computing systems, it can be an additional challenge to actually measure and quantify such adaptation measures. Reinecke and Wolter [8] introduced an adaptation metric to web services called *adaptivity*. The metric measures the capacity of the system for self-adaption. Chen et al. [9] used the time between unacceptable and acceptable state transitions to measure the adaptation capacity of a service. Sheldon et al. [10] proposed a model that takes clues from biological organisms to improve the survivability [11, 12] of systems. This model is based on the ideas of autonomic computing and self-adaptation in order to promote survival [13].

This paper presents a resilience quantification model that can be used to evaluate the functioning of such autonomic features in computing systems. To evaluate the resilience of a given system, the proposed model considers two aspects: a precise resilience definition and an accurate way of measuring resilience at any given time. It addresses both aspects by proposing a definition of resilience that includes adaptation, the importance of services, and a function to quantify resilience. The assessment takes into account service states based on adaptivity-metric and service interdependencies. Each service has an adaptivity score associated with it. Adaptivity measures how adaptable the service is—the higher the service adaptation, the better is its resilience. By considering services interdependencies, the model predicts the degradation of a service and its effects on other services in the system. Consequently, it predicts the overall system resilience.

The contributions of this paper are as follows. We discuss a model for resilience quantification that takes into account service interdependencies and adaptation. The model takes a different approach from the current models based on performance metrics to analyze system behavior. By using the formalism of Markov networks, the proposed model is suitable for performing both prediction and diagnosis of the effects of failures in the system. For instance, the model may be used to infer the consequences of unavailability of a service to the overall system, which is a predictive approach. It also may be used to compute the likelihood that a certain service was responsible for the current degraded state of the system, which is a diagnosis approach. However, in this paper, only the predictive approach is demonstrated.

In addition, we discuss a metric to quantify service adaptation. The adaptivity metric is based on performance attributes of system services (e.g., service time, response time, and queuing time). It computes how often the service adapts and evaluates the efficiency of such adaptations in terms of performance improvement. The metric is sufficiently generic to support different types of performance attributes. Furthermore, it can treat one or more performance attributes in an aggregated manner.

The remainder of the paper is structured as follows. In the first section, we discuss related methodologies for evaluating resilience of networked systems. In the second section, we present the proposed quantification model, where first we define the meaning of "resilience" considered in our approach in addition to other concepts such as services, factors, and service graphs. Then, we describe the model and the proposed adaptivity metric. In the third section, we provide the details of a case study with respect to a SCADA (supervisory control and data acquisition) system [14] that is considered for evaluation. The section presents an experimental setup used for evaluation and some discussion about the results. Finally, in the last section, we conclude the paper and present future steps in this research direction.

Related work

Over the years, a wide range of resilience metrics have been proposed that consider different scenarios and domains. Accompanying those metrics were resilience definitions tailored to the specific domains. The concept of resilience is often confused with other disciplines such as dependability, reliability, survivability, and so forth. According to Trivedi et al. [15] and Jabbar et al. [16], resilience measures consider aspects that are not part of system design, whereas other dependability metrics deal with conditions within the design envelope.

Sterbenz et al. [17, 18] defined resilience based on trustworthiness of the system (dependability, security, and performability [19]) and its tolerance to disruption. Trivedi et al. [15] adopted the definition proposed by Laprie [20] and Strigini [21]. They presented a probabilistic quantification approach using Markov models for resilience in computer systems and networks. Strigini [21] defined resilience as the extent to which service can be provided continuously in the face of changes. Here, *changes* refer to unexpected failures, intrusions, or accidents. Ghosh et al. [22] also considered this definition and proposed a resilience assessment approach for IaaS (infrastructure-as-a-service) cloud systems using a stochastic reward net-based mode with respect to job rejection rate and provisioning response delay. They also considered the interdependencies of service; however, it may not be clear how their model supports complex networked systems with a higher number of services and interconnections.

There is an existing body of research that quantifies the resilience of networks. Jabbar et al. [16] developed a framework for quantifying resilience between any two layers in a network stack and applied it to mobile ad hoc networks. Resilience is quantified as a function of state transitions between different service states in the presence of disturbances in the operational state of the network. Similar to the work proposed in this paper, they considered three operational states of a service: normal operation, partially degraded operation, and severely degraded operation. Rosenkrantz et al. [23] presented a graph-theoretic model for service-oriented networks and proposed metrics that quantify the resilience of such networks under node and link failures. The metrics are based on a deterministic model that is defined as a maximum number of nodes or edges for which, in the case of network failures, the resulting sub-network is self-sufficient. Najjar and Gaudiot [24] defined network resilience in a multi-computer system environment as a measure that provides the upper bound for maximum tolerable node failure while the network remains connected with a certain probability. Whitson and Ramirez-Marquez [25] proposed a probabilistic approach for computing Category I resilience (e.g., static resilience) of a two-terminal network based on Monte Carlo simulation.

There is some work in the literature on generic resilience metrics that are applicable to a wide variety of applications. Henry and Ramirez-Marquez [2] presented a metric that takes into account three key parameters when defining resilience: disruptive events, component restoration, and overall resilience strategy. They defined resilience as a ratio of recovery time to loss suffered by the system at a previous point in time. Vugrin et al. [26] proposed a general framework for quantifying the resilience of infrastructure and an economic system by considering recovery costs and disruptive events. In general, the measures described in the literature are intended for evaluation at design time. In addition, they often either do not consider interdependencies between different services or are only applicable for specific systems.

The proposed quantification model

As mentioned, in this paper, resilience is interpreted as the system capacity to adapt. The proposed model is based on the following resilience definition: Resilience is the capacity of critical services to adapt in order to provide their functionalities in cases of undesired events compromising parts of the system (*Definition 1*).

The model quantifies resilience by taking into account the capacity of adaptation of individual services and their interdependencies. Also as mentioned, to measure service adaptation, we use a metric called *adaptivity*. The adaptivity metric considers three discrete states: high, medium, and low. It is computed for each individual service and then aggregated as a joint probability distribution that is then used to compute the overall system resilience. More details on its quantification are given in the section "Service state model: T."

The joint probability distribution over all services of the system is defined by a *resilience function* ($\mathcal{R}$). The function, represented in Equation (1), computes the probability of the critical services (s_c) not having low adaptivity state, conditional on all other services s_{nc} (non-critical services) having low adaptivity state as evidence.

$$\mathcal{R} = P(s_c = \neg\text{low} | s_{nc} = \text{low}). \tag{1}$$

To represent and compute the joint distribution efficiently (and compactly), the model relies on the formalism of Markov networks [27].

Markov networks are based on undirected graphs, in which the nodes represent the variables and the edges represent connections between nodes. However, for Markov networks, the joint distribution consists of factors, not nodes. A factor $\phi(x)$ is defined as $\phi(x) : x \rightarrow \mathbb{R}^+$. A factor usually consists of more than one node of a graph. Markov networks are defined as products of factors (maximal cliques) of a graph. Given a set of variables $X\{x_1, x_2, x_3, \ldots, x_n\}$, a Markov network is defined as $p(x_1, \ldots, x_n) = 1/Z \times \prod_{f=1}^{F} \phi_f(X_f)$, where F is the set of factors on subsets of variables of the graph, and Z is a constant, used to normalize the distribution [28]. We refer the reader to [27, 28] for detailed discussions on Markov networks.

The proposed model consists of four components $\langle H, S, T, F \rangle$. H is the service graph that represents the system under evaluation. S is the set of services that are part of the system. As suggested, services are categorized as critical (s_c) and non-critical (s_{nc}). T defines the service state model. Services may have a variety of states. The state model defines the service states and how they are computed. It is also part of the state model to define the metrics used to compute such states. F defines factors that represent the service graph and that are used to compute the system resilience.

Service graph: H

The service graph represents how the services are connected with each other in the system. A service graph $H = (S, L)$ is an undirected graph defined by a set of services represented as nodes $S = \{s_1, s_2, \ldots, s_n\}$ and a set of links mapped as edges $L = \{l_1, l_2, \ldots, l_n\}$. Edges exist only if there is at least one transaction between two services. A transaction is defined as follows: Given two services X and Y, there exists a transaction $T(X, Y)$ if a successful communication to exchange information is carried out either between X and Y or vice-versa (*Definition 2*).

Service graphs are usually created through the analysis of network traffic interactions between services. For the cases in which it is not possible to access such information, they can be created on the basis of the expected transactions between services. An expected transaction is a possible interaction between two services that was planned during the design of the system.

It is important to note that a computer host that is hosting multiple services will have its services individually represented in the service graph. Network devices (e.g., routers and switches) that are commonly represented in a network topology are not part of service graphs.

Services: S

As mentioned earlier, a system is evaluated, and consequently quantified, through the instrumentation of services. This paper uses the following service definition: A service is software that provides a functionality consumed by other entities (services or clients) of the system. Services can have a variety of states that describe different levels of operability, for example, normal, compromised, acceptable, unacceptable, and so forth (*Definition 3*).

Critical services (s_c) are vital to the functioning of a system, and they usually have priority over other services. The identification of critical services is a challenging problem, and researchers and practitioners have resorted to different approaches [29, 30]. A common approach often adopted in the literature is to delegate the identification of critical services to system experts [31]. This proposed model resorts to this particular approach for identifying critical services.

Service state model: T

The definition of the possible states that a service can assume is necessary for the analysis of dependencies between services, and consequently, to the resilience quantification proposed in this paper. As mentioned, the state model defines three states of adaptivity: high, medium, and low. A service with low adaptivity means that it is not responding properly to the changes in its environment. A service with high adaptivity means the opposite.

The adaptivity metric has been used elsewhere [8]. In this model, we present a different version of adaptivity. For this proposed version, adaptivity is based on service performance attributes, in addition, to support the use of one or more attributes.

To compute adaptivity, the first step is to compute the service performance attributes. The attributes usually illustrate important service aspects that should be monitored. For instance, the CPU load of the machine, where a service is located, may offer important information about the service performance.

The attributes are represented as a set $Pa = \{pa_1, pa_2, \ldots, pa_n\}$, where each element represents an attribute. In addition, another set $W = \{w_1, w_2, \ldots, w_n\}$ is used to represent the weights associated with each performance attribute pa_i. The weights are used to differentiate attributes on the basis of their importance. For instance, some services can be more CPU-bound, others more memory-bound, and so forth. As shown below, the adaptivity metric is based on individual service scores. Therefore, the performance attributes must be aggregated as one final score. To create the aggregated score, the attribute values are normalized, and an average weighted computation is used to generate the final score

$$p = \frac{\sum_{i=1}^{n} w_i \times \|pa_i\|}{\sum_{i=1}^{n} w_i}.$$

Assuming a service with three performance attributes $Pa = \{2.1, 0.8, 3.4\}$ and weights $W = \{2, 1, 1\}$, the performance score computation is as follows. First, Pa is normalized $\|Pa\| = \{0.62, 0.23, 1.0\}$, then the average weight is computed $p = (0.62 \times 2 + 0.23 \times 1 + 1 \times 1)/3 \rightarrow 0.823$. It is important to notice that the attributes are computed periodically and that a set $P = \{p_1, p_2, \ldots, p_n\}$ of performance data points for individual services are used to compute adaptivity states.

Before computing such states, it is necessary to compute adaptivity scores, an intermediate stage between performance scores and adaptivity states. Let a_i represents an adaptivity score at data point i, and p_i and p_{i-1} represent performance scores at data points i and $i - 1$. The adaptivity score (a_i) is computed using p_i and p_{i-1} as inputs. Equation (2) shows how it is computed. For instance, given $p_i = 0.6$ and $p_{i-1} = 0.45$, $a_i = 0.45 + |0.45 - 0.60| \times$

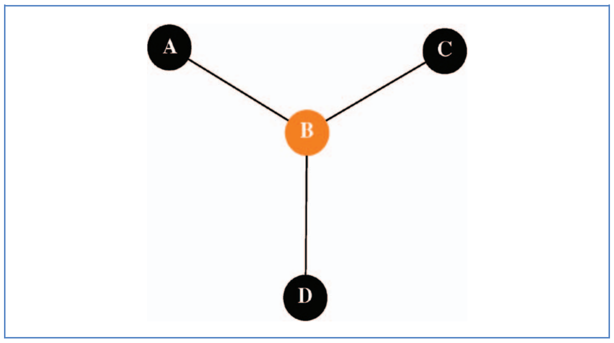

Figure 1

A simple computing system with four services A, B, C, and D. Service B (in orange) is a critical service.

$0.60 \Rightarrow 0.54$. As a result of the computation, a set $A = \{a_1, a_2, \ldots, a_{n-1}\}$ of adaptivity scores is generated.

$$a_i = \begin{cases} p_i, & \text{if } p_i = p_{i-1} \\ p_{i-1} + |p_{i-1} - p_i| \times p_i, & \text{if } p_i > p_{i-1} \\ p_i + |p_{i-1} - p_i| \times p_i, & \text{if } p_i < p_{i-1} \end{cases} \quad (2)$$

The set A is then used to generate the adaptivity states (high, medium, and low). Two thresholds t_1 and t_2, where $t_1 < t_2$, are also used to define the bounds to compute the adaptivity states [Equation (3)]. The thresholds can be either automatically computed or manually defined. In this paper, they are manually defined.

$$(a_i = L) \leq t_1 < (a_i = M) \leq t_2 < (a_i = H) \quad (3)$$

Factors: F

Factors represent the conditional probability distribution (CPD) used by the Markov network. They represent the maximal cliques of the service graph structure. For instance, the service graph represented in **Figure 1** with four services named $\{A, B, C, D\}$ has three maximal cliques $\{A - B, C - B, D - B\}$. Therefore, three factors with corresponding CPDs are used to compute the joint probabilistic distribution for that particular system.

Building the quantification model

The quantification model consists of six steps, from data acquisition to resilience quantification and inference.

1. *Assignment of critical services*—This is a manual process performed by system experts.
2. *Data acquisition*—The data acquisition process consists of acquiring the data at the service host and processing data at a central repository. Each service generates a set, with information about the other services it had communicated with and a dataset that contains the performance attributes measurements.
3. *Adaptivity computation*—The computation follows the steps defined in the section "Service state model: T." The performance attributes acquired from the services are used to compute the adaptivity scores and, consequently, the adaptivity states. The states are based on Equation (3). They are later computed to generate the CPDs used by the Markov network.
4. *Service graph*—This graph is derived from the service transactions. Every time a service communicates with another service in the system, a transaction between the two services is created and stored. The information is later retrieved to generate the graph. Each transaction describes a link between two services in the service graph.
5. *Computing CPDs*—To compute the CPDs, the first step is to define the factors of the service graph. As mentioned earlier, factors are representations of maximal cliques with values attached to them. Once the cliques are defined, the CPDs based on the adaptivity state services are created.
6. *Resilience computation*—Once the model is built, one can perform queries to determine the overall resilience (conditional probabilities on critical services) and can also perform some types of diagnosis, such as what service is responsible for the degradation on the overall system resilience. Queries to answer these types of questions about the system are essentially variations of Equation (1), taking into consideration different evidence. Equation (4) is another example of queries supported by the proposed model.

To provide a better understanding of the computation, a simple example is illustrated in Figure 1. As mentioned, the system contains four services named $\{A, B, C, D\}$, where service B is connected to services A, C, and D. The service graph has three maximal cliques $\{A - B, C - B, D - B\}$. Assuming that the service model for this particular system consists of two states, t and f, B is the critical service, and state t is the one on which the critical service B should be. The resilience function is defined as the following query:

$$\mathcal{R} = P(b^t | c^f, a^f, d^f) = \frac{\phi(a^f, b^t) \times \phi(c^f, b^t) \times \phi(b^t, d^f)}{\sum_{a,b,c,d} \phi(a^f, b) \times \phi(c^f, b) \times \phi(b, d^f)}. \quad (4)$$

For the sake of brevity, we omitted the steps to compute the service states and factor CPDs. Instead, random values are used to represent factors $\phi(A, B) = \{30, 5, 1, 10\}$, $\phi(C, B) = \{100, 1, 1, 100\}$, and $\phi(D, B) = \{1, 100, 100, 1\}$. Each value in the sets represents a combination of services' states. For instance, $b^f = 30$, $a^f b^t = 5$, and so forth. Those values are used to compute the CPDs of each factor. By applying the resilience function to this system, the result is $\mathcal{R} = (5 \times 1 \times 100)/[(30 \times 100 \times 1) + (5 \times 1 \times 100)] \Rightarrow 500/3500 \approx 0.14$.

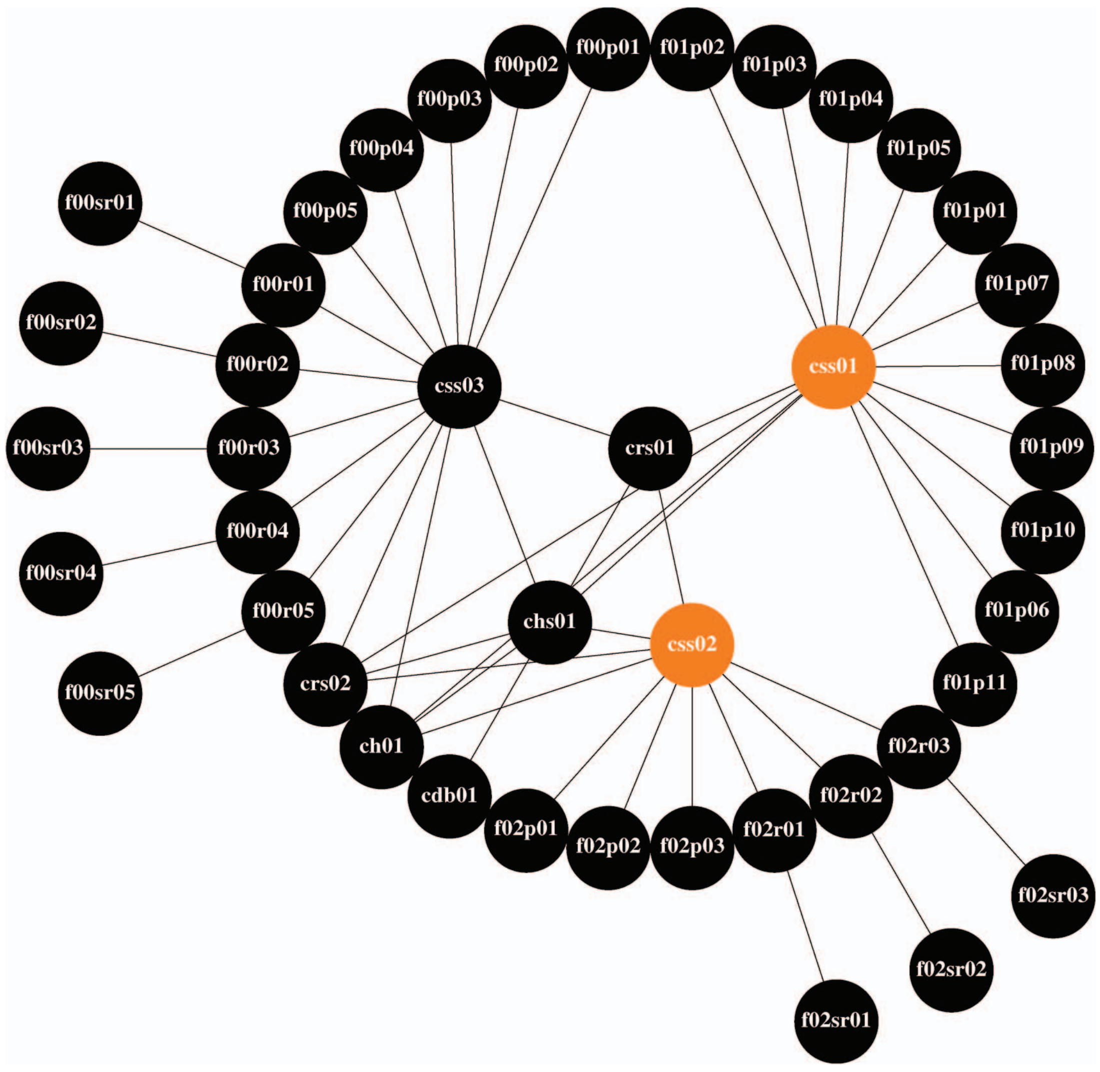

Figure 2

A SCADA system with 43 services. css01 and css02 (in orange) are critical services.

Evaluation

To evaluate the proposed model, we compute the resilience of a SCADA system [14] that controls critical infrastructures such as power grids, nuclear plants, and so forth. In a typical SCADA deployment, MTUs (master terminal units) are responsible for managing field nodes. Human-machine interaction (HMI) servers are responsible for allowing system operators to program and interact with the field devices. Remote terminal units (RTUs) and programmable logic controllers (PLCs) are field devices that control and monitor transmission lines, generators, and so forth. Historians are databases that collect logs of all actions happening in the system. Relational databases are used to store information used by HMI servers and MTUs.

To help understand the SCADA system and its services' functionalities, the following nomenclature is used: *css*01 (controller service server 01), *cdb*01 (central database server 01), *chs*01 (central HMI server 01), and *crs*01 (central remote servers 01). There are three field networks *f*00, *f*01, and *f*02. *r*01 stands for RTU 01, *sr*01 stands for sensor 01, *p*01 stands for PLC 01, and so forth. Because there are PLCs, RTUs, and sensors that belong to different field networks, service names are defined as a combination of the field network code plus the service code. For example, *f*01*r*01 indicates RTU *r*01 within field network *f*01.

The service graph that represents the system is illustrated in **Figure 2**. It consists of 43 nodes in total. To avoid referring to all services one by one, we use XX to represent the two-digit number used to enumerate services of the same type. There are three MTUs ($cssXX$), one central historian ($ch01$), one relational database ($cdb01$), one HMI server ($chsXX$), two remote servers ($crsXX$) that are used to provide access for remote users, and multiple RTUs ($fXXrXX$), sensors ($fXXsrXX$), and PLCs ($fXXpXX$).

To evaluate the system, we performed a simulation based on the queuing theory [32]. Each service is represented by a queue, with an exponential service time distribution attached to it. Field services have limited queue size, whereas servers such as HMI, historian, and remote servers are defined with unlimited queue sizes.

We defined two services as critical: $css01$ and $css02$ (orange color in Figure 2). Therefore, all other services are non-critical. Following Equation (1), the resilience function that represents the system is defined as

$$\mathcal{R} = P(css01 = \neg\text{low}, css02 = \neg\text{low} \mid s_{\text{nc}} = \text{low}). \qquad (5)$$

During the simulation, one performance attribute was instrumented. The actual service time (AST) attribute represents the time a service takes to process a request (service time + queuing time). It is recorded every time a request is processed. For cases in which the service is a client of another service, AST is the total time to receive the response from the service provider. The adaptivity metric is defined by computing the measurements of this attribute.

The service graph illustrated in Figure 2 contains 45 maximal cliques. Therefore, 45 factors are defined and used to compute $\mathcal{R}$, the resilience function. Because of space constraints, we omitted the equation that shows the factors.

Simulation results

The simulation ran for 86,400 seconds, which is equivalent to one day of running the system. It was run 10 times, and an average of AST readings was used. It was used as the input to compute the adaptivity states of the services. Two thresholds, $t_1 = 0.35$ and $t_2 = 0.70$, were used.

To compute $\mathcal{R}$, Equation (5) was applied. Because $css01$ and $css02$ are independent, given all other non-critical services, the computation can be expressed as $\mathcal{R} = P(css01 = \neg\text{low}|s_{nc} = \text{low}) \times P(css02 = \neg\text{low}|s_{nc} = \text{low}) \Rightarrow (0.168 + 0.117) \times (0.973 - 0.005) \approx 0.28$. This result shows that the system has a low resilience, mostly because the service $css01$ has a high probability of having low adaptivity state $P(css01 = \text{low}) \cong 0.72$, as we can see in **Figure 3(c)**.

We can also use the model to find other aspects of the system. For instance, as illustrated in **Figure 3(a)** and **(b)**, changes of the non-critical services adaptivity states show interesting results. When the evidence of non-critical services is set to high adaptivity, the resilience of the system drops even further from $\mathcal{R} \approx 0.28$ to $\mathcal{R} \approx 0.20$, as illustrated in Figure 3(a). This result shows that according to the current data, there is a very small probability of the system being

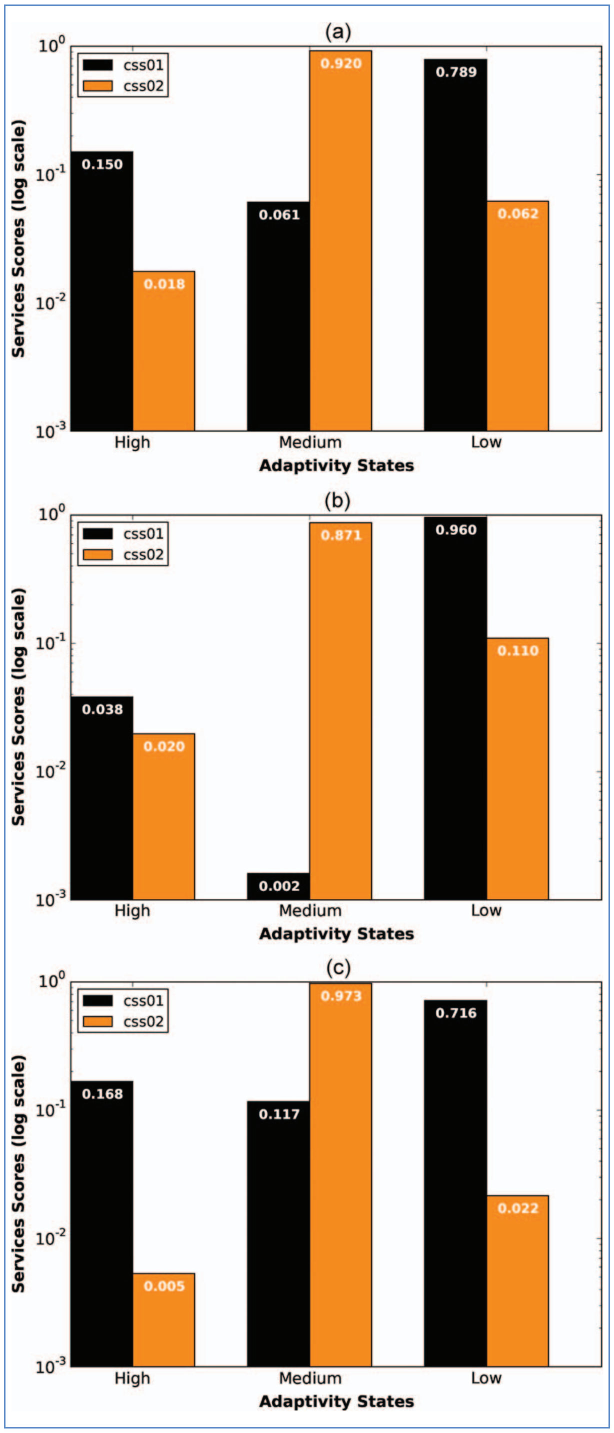

Figure 3

Charts of critical services' resilience. (a) Critical services with evidence = high. (b) Critical services with evidence = medium. (c) Critical services with evidence = low.

resilient even when the non-critical services are evidencing high adaptivity. Again, this is due to service *css*01 having a high probability of being in a low adaptivity state $P(css01 = low) \approx 0.80$. In other words, the other services of the system are not contributing to the low adaptivity of *css*01. On the other hand, *css*02 is responding to the changes, as we can see in Figure 3(a) and (b).

Another interesting finding is the one illustrated in Figure 3(b). The changes on the non-critical services adaptively state, from high to medium, make services *css*01 and *css*02 have higher probabilities for the low adaptivity state $P(css01 = low) \approx 0.96$, and $P = (css02 = low) \approx 0.11$. Again, the change has much more effect on *css*01. We can conclude that *css*01 is the main responsible service for the resilience of the system.

Conclusion

This paper presented a model to quantify the resilience of computer systems. The model uses adaptivity of service states to represent adaptation. It aggregates them as CPDs into a Markov network, which is used to quantify the overall resilience of the system. The paper also introduces an adaptivity metric that is used to measure the capacity of adaptation of services. The metric relies on performance attributes to its measurements, and it is sufficiently generic to support any performance attribute defined by the user. Thus, it can be used in different types of computing systems.

Once the service graph is built, and the adaptivity scores are computed, a Markov network is created. It is used to compute the probabilities of critical services not having low adaptivity state. Such computation follows the definition of resilience introduced earlier in the paper.

Further, the model is evaluated through the simulation of a SCADA system. The evaluation shows the model is a useful tool for quantifying resilience. For instance, it can find discrepancies as the one shown between services *css*01 and *css*02.

Even though not demonstrated in this paper, the model can be quickly automatized and further extended to perform other tasks such as being utilized as a tool to diagnose issues in computing systems. One shortcoming aimed to be addressed as future work is the model implementation to be deployed as an assistant tool that could be easily used by computing system administrators.

Another issue we aim to address in the future is the automatic generation of thresholds per service. In this paper, we used only one pair of thresholds to all services. In the future, we want to compare an automatic generation of thresholds per service with the current approach of global thresholds. Finally, as part of future work, we aim to define heuristics to address cases with disconnected service graphs.

References

1. A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Depend. Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.
2. D. Henry and J. Ramirez-Marquez, "Generic metrics and quantitative approaches for system resilience as a function of time," *Reliab. Eng. Syst. Saf.*, vol. 99, pp. 114–122, Mar. 2012.
3. K. Wolter, A. Avritzer, M. Vieira, and A. Moorsel, *Resilience Assessment and Evaluation of Computing Systems*. New York, NY, USA: Springer-Verlag, 2012.
4. D. Nicol, W. Sanders, and K. Trivedi, "Model-based evaluation: From dependability to security," *IEEE Trans. Depend. Secure Comput.*, vol. 1, no. 1, pp. 48–65, Jan.–Mar. 2004.
5. M. Al-Kuwaiti, N. Kyriakopoulos, and S. Hussein, "A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 2, pp. 106–124, Second Quarter, 2009.
6. P. Neuman, "Practical architectures for survivable systems and networks," SRI Int., Menlo Park, CA, USA, Jun. 2000, Tech. Rep.
7. R. Ellison, D. Fisher, R. Linger, H. Lipson, T. Longstaff, and N. Mead, "Survivable network systems: An emerging discipline," Carnegie-Mellon Softw. Eng. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-97-TR-013, 1997, revised 1999.
8. P. Reinecke and K. Wolter, "Adaptivity metric and performance for restart strategies in web services reliable messaging," in *Proc. 7th Int. Workshop Softw. Perform.*, New York, NY, USA, 2008, pp. 201–212.
9. L. Chen, Q. Wang, W. Xu, and L. Zhang, "Evaluating the survivability of SOA systems based on HMM," in *Proc. IEEE Int. Conf. Web Services*, Miami, FL, USA, 2010, pp. 673–675.
10. F. Sheldon, T. Potok, M. Langston, A. Krings, and P. Oman, "Autonomic approach to survivable cyber-secure infrastructures," in *Proc. IEEE ICWS*, San Diego, CA, USA, 2004.
11. J. Sterbenz, R. Krishnan, R. Hain, A. Jackson, D. Levin, R. Ramanathan, and J. Zao, "Survivable mobile wireless networks: Issues, challenges, and research directions," in *Proc. ACM WiSE*, New York, NY, USA, 2002, pp. 31–40.
12. J. Knight, E. Strunk, and K. Sullivan, "Towards a rigorous definition of information system survivability," in *Proc. 3rd DISCEX*, Washington, DC, USA, 2003, pp. 78–89.
13. J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.
14. J. Weiss, *Protecting Industrial Control Systems from Electronic Threats*. New York, NY, USA: Momentum Press, 2010.
15. K. Trivedi, D. Kim, and R. Ghosh, "Resilience in computer systems and networks," in *Proc. Int. Conf. Comput.-Aided Des.*, San Jose, CA, USA, 2009, pp. 74–77.
16. A. Jabbar, H. Narra, and J. Sterbenz, "An approach to quantifying resilience in mobile ad hoc networks," in *Proc. 8th Int. Workshop DRCN*, Krakw, Poland, 2011, pp. 140–147.
17. J. Sterbenz, E. Etinkaya, M. Hameed, A. Jabbar, Q. Shi, and J. Rohrer, "Evaluation of network resilience, survivability, and disruption tolerance: Analysis, topology generation, simulation, and experimentation," *Telecommun. Syst.*, pp. 1–21, 2011.
18. J. Sterbenz, D. Hutchison, E. Etinkaya, A. Jabbar, J. Rohrer, M. Scholler, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Comput. Netw.*, vol. 54, no. 8, pp. 1245–1265, Jun. 2010.
19. R. Smith, K. Trivedi, and A. Ramesh, "Performability analysis: Measures, an algorithm, and a case study," *IEEE Trans. Comput.*, vol. 37, no. 4, pp. 406–407, Apr. 1988.
20. J. Laprie, "Resilience for the scalability of dependability," in *Proc. 4th IEEE Int. Symp. Netw. Comput. Appl.*, Cambridge, MA, USA, 2005, pp. 5–6.
21. L. Strigini, "Resilience assessment and dependability benchmarking: Challenges of prediction," in *Proc. DSN Workshop Resilience Assess. Depend. Benchmarking*, Anchorage, AK, USA, 2008.

22. R. Ghosh, F. Longo, V. Naik, and K. Trivedi, "Quantifying resiliency of IaaS cloud," in *Proc. 29th IEEE Symp. Rel. Distrib. Syst.*, New Delhi, India, 2010, pp. 343–347.
23. D. Rosenkrantz, S. Goel, S. Ravi, and J. Gangolly, "Resilience metrics for service-oriented networks: A service allocation approach," *IEEE Trans. Services Comput.*, vol. 2, no. 3, pp. 183–196, Jul.–Sep. 2009.
24. W. Najjar and J. Gaudiot, "Network resilience: A measure of network fault tolerance," *IEEE Trans. Comput.*, vol. 39, no. 2, pp. 174–181, Feb. 1990.
25. J. Whitson and J. Ramirez-Marquez, "Resiliency as a component importance measure in network reliability," *Reliab. Eng. Syst. Saf.*, vol. 94, no. 10, pp. 1685–1693, Oct. 2009.
26. E. Vugrin, D. Warren, M. Ehlen, and R. Camphouse, "A framework for assessing the resilience of infrastructure and economic systems," in *Sustainable and Resilient Critical Infrastructure Systems: Simulation, Modeling, and Intelligent Engineering*. Berlin, Germany: Springer-Verlag, 2010, pp. 77–116.
27. R. Kinderman and K. Snell, *Markov Random Fields and Their Applications*. Providence, RI, USA: American Mathematical Society, 1980.
28. D. Koller and N. Friedman, *Probabilistic Graphical Models Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.
29. S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 1994.
30. S. Borgatti, "Identifying sets of key players in a social network," *Comput. Math. Org. Theory*, vol. 12, no. 1, pp. 21–34, Apr. 2006.
31. P. Heegaard and K. Trivedi, "Survivability quantification of communication services," in *Proc. 38th IEEE Int. Conf. DSN*, Anchorage, AK, USA, 2008, pp. 462–471.
32. C. Harris and D. Gross, *Fundamentals of Queuing Theory*. Hoboken, NJ, USA: Wiley, 1974.

Received November 18, 2012; accepted for publication December 16, 2012

Carlos Queiroz *IBM Research - Australia, Melbourne Laboratory, Carlton, VIC 3053, Australia (carque@au1.ibm.com).* Dr. Queiroz recently received his Ph.D. degree at RMIT University, Melbourne, Australia. Before receiving his Ph.D. degree, he worked in the industry, building distributed systems for mission-critical solutions. His research interests are machine learning, probabilistic graphical models, and distributed systems.

Saurabh Kumar Garg *IBM Research - Australia, Melbourne Laboratory, Carlton, VIC 3053, Australia (skgarg@au1.ibm.com).* Dr. Garg is a Research Fellow at IBM Research - Australia. He is one of the few Ph.D. students who submitted his thesis in less than three years at the University of Melbourne. During his Ph.D. work, he has been awarded various special scholarships for his Ph.D. candidature. His research interests include stream computing, resource management, scheduling, utility and grid computing, cloud computing, and green computing.

Zahir Tari *RMIT University, Melbourne, VIC 3001, Australia (zahir.tari@rmit.edu.au).* Prof. Tari received an honors degree in operational research from the Universite des Sciences et de la Technologie Houari Boumediene (USTHB), Algiers, Algeria, and a Master's degree in operational research from the University of Grenoble I, France. He received his Ph.D. degree in artificial intelligence from the University of Grenoble II, France. He is presently with the School of Computer Science and Information Technology, RMIT University, Melbourne. His research interests include a special focus on performance in distributed systems, security (in SCADA systems), and web services (in general). Recently, Prof. Tari has established himself significantly in the nascent area of Smart Grid Systems, leading his SCADA team in an exploration of the security issues in such infrastructures. He is a senior member of the Institute of Electrical and Electronics Engineers and a regular organizer of international conferences, and he frequently contributes to renowned journals.