SOCA (2007) 1:35–50
DOI 10.1007/s11761-007-0001-4

ORIGINAL PAPER

Accountability monitoring and reasoning in service-oriented architectures

Yue Zhang · Kwei-Jay Lin · Jane Y. J. Hsu

Received: 14 November 2006 / Revised: 22 December 2006 / Accepted: 2 January 2007 / Published online: 2 March 2007

Abstract Service-oriented architecture (SOA) provides a powerful paradigm to compose service processes using individual atomic services. When running a service process, SOA needs an efficient and effective mechanism to detect service delivery failures and to identify the individual service(s) that causes the problem. In this research, we study the model of accountability to detect, diagnose, and defuse the real cause of a problem when service errors (such as incorrect result or SLA violation) occur in a service process. Our approach leverages Bayesian networks to identify the most likely problematic services in a process and selectively inspect those services. An evidence channel selection algorithm is designed to specify which services in a service network should be monitored to achieve the best cost-efficiency. We model the channels selection as the classic facilities location problem. We also adopt a continuous knowledge learning process to manage the dynamic nature of SOA. The performance study shows that our proposed accountability mechanism is effective on identifying the root cause of problems and can achieve significant cost savings: with 50% of services' outputs monitored as evidence, the comprehensive diagnosis correctness can reach 80% after only 20% of services are inspected.

Y. Zhang · K.-J. Lin (✉)
Department of Electrical Engineering and Computer Science, University of California, Irvine, USA
e-mail: klin@uci.edu

Y. Zhang
e-mail: yuez@uci.edu

J. Y. J. Hsu
Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
e-mail: yjhsu@csie.ntu.edu.tw

Keywords SOA · Accountability · Bayesian networks · Diagnosis · Monitor

1 Introduction

Service-oriented architecture (SOA) using Web services has emerged as a major software architecture in the past few years [1,2]. Using SOA, enterprise systems can define and execute transactions across multiple server domains at distributed locations. Companies can use service-oriented computing's (SOC's) plug-and-play interoperability to compose business processes and integrate different services on the fly to enable dynamic cooperation among business partners. Collaboration protocols such as Web Service Business Process Execution Language (WSBPEL) [3] and Web services Choreography (WS-CDL) [4] have been defined and are being adopted.

The concept of software composition is not new. For many years, computer system researchers have studied the methodology of building sound and dependable systems using well-formed composition rules and deriving some formal properties that can be assured by the compositions. The goal of such a methodology is to simplify and improve system development and deployment. However, the goal is not easy to achieve; there is still a huge gap between theory and common practice. In [5], many challenges regarding reliable compositions have been identified, including multiparty incompatibilities, scalability issues, policy composability, assurance composability, certification composability, etc. These issues remain the same, or are even exacerbated, in SOA systems. This is because SOA promotes dynamic service discovery and binding. In addition to interoperability,

SOA compositions often invoke external services that may not provide an adequate or consistent level of performance and stability. A sound SOA composition methodology thus needs a more systematic measure to handle the potential discrepancy on the delivery of individual services.

In a service process involving many service partners, it is also imperative to have a mechanism to decide the *accountability* of individual services, in order to attribute credit for success or responsibility for failure in the whole process. This is because outputs from individual services have dependencies: a low output quality from a service may cause the output quality degradation of all its successors in a service process. Even worse, successive quality degradation may be accumulative or multiplicative. So we need to be able to identify the root cause of an observed service problem.

In this paper, we present the design of an accountability framework as part of an integrated SOA deployment and management solution to *detect*, *diagnose*, and *defuse* the root cause of a service deficiency (such as functional errors or service level agreement (SLA) violations). The accountability framework provides the mechanisms for: (1) QoS-based service selection [6]; (2) SLA-based real-time service monitoring [7]; (3) Bayesian network reasoning to identify the likely causes of a problem and to selectively inspect those services; (4) an evidence channel selection algorithm to find an optimized information collection structure; and (5) a broker-based trust and reputation network for prevention of future problems [8,9]. In our study, the graph model and probability theory are used as the theoretical foundations of the accountability framework.

Accountability benefits SOA since it enables a computing environment to be traceable, measurable, and configurable. The goal of accountable computing is to make all service components have transparency and controllability in order to facilitate the end-to-end quality of service (QoS) at run time. Moreover, SOA systems can be equipped with simple management tools so that users can easily control and reason with services whenever a problem is detected. These goals are above service interoperability and provide opportunities for researchers to contribute new ideas and solutions.

This paper is organized as follows. The accountability model and system architecture are defined in Sect. 2. Section 3 describes the technologies to achieve accountability, mainly focusing on the diagnosis mechanism. In Sect. 4, we present the evidence channel selection algorithm to achieve better efficiency. Section 5 shows the performance study of the accountability framework. Section 6 gives an overview of related research, followed by the concluding remarks in Sect. 7.

2 Accountability model for service-oriented computing

2.1 Accountability

Accountability has been a major concern in the financial industry, especially after ratification of the Sarbanes-Oxley Act of 2002 (also known as the Public Company Accounting Reform and Investor Protection Act of 2002), which establishes new enhanced accountability standards for all US public company management and public accounting firms. The Act has made accountability a mandatory requirement for organizations. A new agency, called the Public Company Accounting Oversight Board, is given the responsibility of overseeing, regulating, inspecting, and disciplining accounting firms in their roles as auditors of public companies. The Act provides the motivation for our research on SOA accountability as services should be similarly regulated for effective QoS delivered by a service process.

In [10], a project on results-based accountability for public institutions has been reported. It identifies the following elements for systems with accountability:

1. *Objective*: Outcomes that articulate what programs are to achieve;
2. *Quality*: Indicators to measure whether or not outcomes have been achieved;
3. *Benchmark*: Performance standards to assess how programs are progressing;
4. *Monitoring*: Data collection instruments to regularly obtain indicator data;
5. *Feedback*: Periodic collection and analysis of data for decision making and reporting.

Among the five elements of a complete accountability measure, the first three are application-dependent and should be defined by application designers. Information technology may be used to implement the other two. We therefore focus our study on the mechanisms for performance monitoring and accountability analysis.

2.2 Assumptions

Our system model describes SOA applications where multiple services form a *flow* $G = (V, E)$, such as business services networks and service supply chains. Each vertex in V represents an atomic service and each edge in E represents an interaction between two services.

In this paper, we make the following assumptions on the system under study. Some of the assumptions may be relaxed in the future.

1. The services flow $G = (V, E)$ is a directed acyclic graph (DAG). For any service s, there is no non-empty directed path starting and ending on s in the flow.
2. Each service is *atomic*. That is, the behavior of a service is independent of other services.
3. The network connection between services is error-free, even though individual atomic services may be problematic.

2.3 A motivating example

Figure 1 shows the *credit pull* service workflow in the context of the lending application studied in [11], which serves as the motivating example in our study. The business workflow provides the functionality that allows a loan sales person to obtain the electronic credit report of a customer in real time. In the figure, each rectangular node represents an atomic service running on either an internal server or any other service provider.

The lending life cycle service initiates the credit pull as a response to a user request by sending a credit request message to the credit service. The credit service listens to this message and passes the request to the external vendor interactions service which in turn places a request with credit vendors A and B who provide credit query services. The credit reports obtained from vendors A and B are used in some local processing and then transferred and stored in the document service. Finally, the credit report is sent to the UI. This business process is completely automated and is executed without human intervention.

For practical purposes, the response time for the service flow is critical to customer satisfaction. If the *Lending Lifecycle Service* initiated a credit pull and has not received the credit report back within an estimated period of time, the problem could have been in any of the several services involved in the sequence. Therefore, an accountability mechanism is useful since a slow response from an upstream node could cause a chain of downstream nodes to appear slow. For example, in Fig. 1, a delay from the *Vender A Order-Process Service* could render the *Local Processing Service* to become slow as it has to wait for the credit report from A as input, even though the *Local Processing Service* actually functions well by itself. Therefore, the very first service which breaks the SLA should be identified and repaired instead of those intermediate services affected by the root cause.

2.4 Model design and system architecture

The proposed accountability mechanism includes the Accountability Authority (AA) and Accountability Agents, as shown in the system architecture in Fig. 2. They are used in the logic flow of a service process deployment as shown in Fig. 3. Agents are used to monitor and collect status information from individual services. Agents send the information to the AA who in turn performs the diagnosis whenever an error is detected.

Our accountability mechanism applies at the service process execution step (circled by dotted line), which enables run-time exceptions to be observable and reportable. The functional steps in the service process deployment include:

– *Service Network Planning*: This is the first step in service process deployment. All applicable service process plans are collected to build a function graph, which is then used to conduct QoS-based service selection. The mapping from a user request to applicable process plans should satisfy the functional requirements of the user request without considering its QoS requirements. Parametric consistency checking between services is performed in order to integrate them together. This problem has been studied by earlier research projects such as [12,13].
– *QoS-based Service Selection*: Given a function graph, this step performs the service flow composition by selecting atomic services based on their QoS parameters and a user's QoS requirements, such as response time, cost, and reputation of a service. In [6], QoS based service selection was achieved by efficient combinatorial and graph algorithms. Furthermore, we presented algorithms to compose back-up service paths in [7]. If any component service fails or becomes overloaded during the execution of a

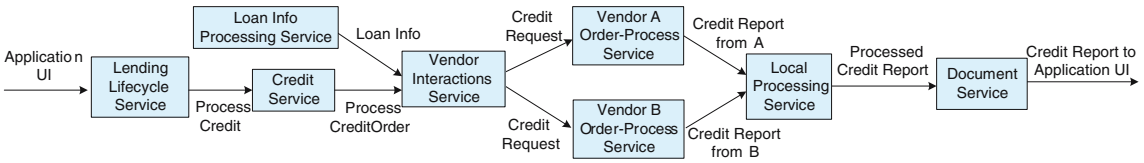

Fig. 1 The credit pull service workflow in the context of a lending application

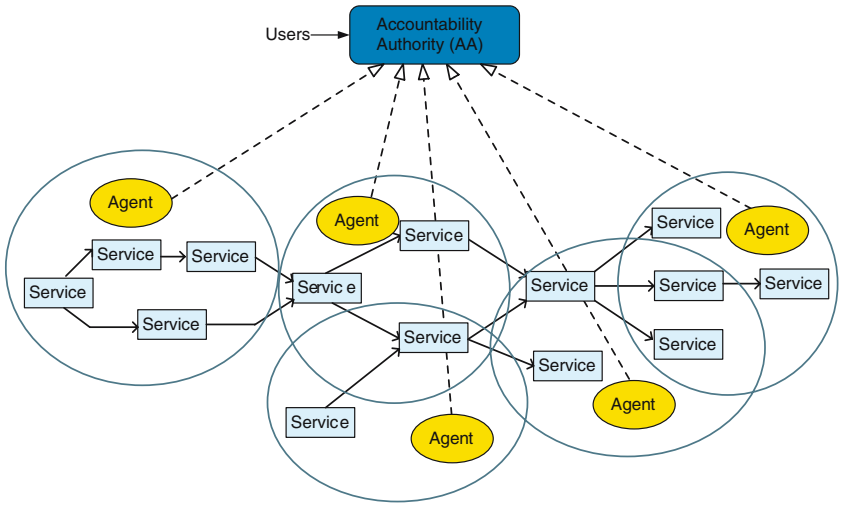

Fig. 2 The system architecture of accountability framework

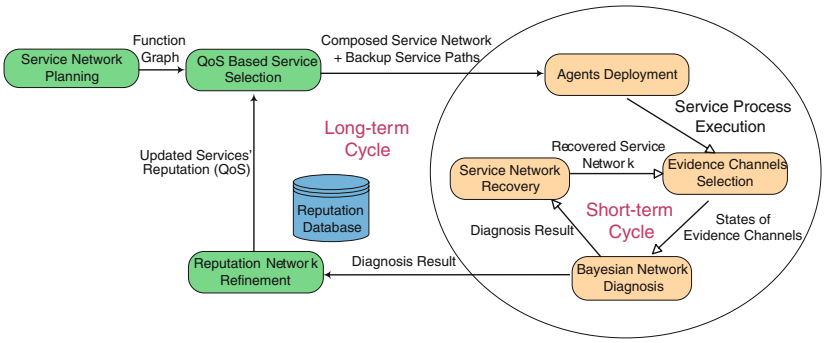

Fig. 3 The logic flow of service process deployment with accountability

service process, a back-up service path not containing the failed service is created, so that the ongoing execution will not be interrupted.

– *Service Process Execution*: This is the step where our accountability model applies. Without accountability, the service process execution appears as a black box to users. When exceptions occur at run-time, it is difficult to observe errors and discover root causes of problems in the service network. Our model achieves the accountability goal by adding several functional sub-steps described below.

1. *Agent Deployment*: Given a composed service network, agents are deployed to monitor all atomic services selected as well as the services in backup paths. Agents are responsible for reporting performance exceptions and the states of service outputs to the Accountability Authority (AA). The issue of how to efficiently deploy agents can be modeled as a set covering problem. Namely, given (1) each agent's capability on the set of services this agent can cover, and (2) locations of services, the goal is to select the minimum number of agents that are capable of covering all selected services. The set covering problem is NP-hard. However, approximation algorithms have been shown to provide feasible solutions [14].
2. *Evidence Channel Selection*: The amount of observable evidence and the number of locations from which it is generated have a big impact on the performance of diagnosing problematic services. While the agents may check the output of all selected services, it is not cost effective to collect all outputs all the time. As information collection is not cost-free, it would be better to monitor only a subset of *evidence channels*, enough to support problem diagnosis. The evidence channel selection problem will be investigated in Sect. 4.
3. *Bayesian Network Diagnosis*: AA leverages the Bayesian network reasoning mechanism to identify the root cause of a problem when violations of SLAs occur in a system. Based on the

information monitored by the agents, the AA performs an effective and efficient diagnosis to determine the services most likely to be the root causes of the problem. The Bayesian network reasoning process will be presented in Sect. 3.

4. *Service Network Recovery*: When problematic services are identified based on the diagnosis result, back-up paths can be used to replace problematic services quickly and effectively [7]. This simple fix can resume the process execution flow immediately. Since agents are deployed to cover all services including backup paths. There is no need to re-deploy the agents. This service recovery step can save significant time and cost. Once the service process is repaired, the accountability logic goes directly to the next step of evidence channel selection.

– *Reputation Network Refinement*: The AA manages and updates the reputation of each atomic service based on the diagnosis result. The reputation of a service is an important QoS parameter that affects the service network recomposition in the long term. Services that are less likely to violate its SLA are more likely to be chosen next time when a service process is composed. We have studied *DIRECT* [8,15], a distributed reputation management framework for Web services, which enables us to effectively and efficiently evaluate, aggregate, and manage the reputation information scattered in a distributed computing environment.

In summary, the flow in Fig. 3 describes two cycles in service process deployment: the *long-term* cycle and the *short-term* cycle. The long-term cycle is invoked when a service process is requested for the first time, or when the process has terminated and then is restarted. Using this cycle, the service process is composed from scratch based on the most current QoS and reputation data about all services. The short-term cycle is used during service process execution. Suppose a service process is in the middle of execution when some individual service in the process has failed to function as defined by its SLA. In that case, we will repair the process by rerouting the service flow to some other services that are able to provide substituted functionalities.

Several of the steps in service process deployment, including *service network planing*, *QoS-based service selection*, and *reputation refinement networks*, have been studied earlier [6,8,13]. In this paper, we will concentrate on *Bayesian network diagnosis* and *evidence channel selection*. *Agent deployment* issues will be left as future work.

3 Diagnosis using Bayesian network reasoning mechanism

A Bayesian network is a probabilistic graphical model [16] and is leveraged to perform the root cause diagnosis in our accountability framework. Bayesian networks have been a popular model in the AI community. It has been successfully used as a diagnosis and trouble-shooting engine in the mechanical operation and medication treatments fields [17,18]. Bayesian networks use a directed acyclic graph (DAG) model for reasoning under uncertainty, where the nodes represent random variables (discrete or continuous) [19]. Each variable has a finite set of mutually exclusive states [20]. The directed edges connecting the nodes can be used to represent direct causal relationships [21].

Assume $U = \{V_1, \ldots, V_n\}$ which represents all random variables in system. The goal of reasoning under uncertainty is to calculate the *conditional* probability of a variable V_i in one of its states given the states of a set of other variables $\{V_1, V_2, \ldots, Vk\}$, where $(\{V_1, V_2, \ldots, Vk\} \subset U) \wedge (V_i \notin \{V_1, V_2, \ldots, Vk\})$. This conditional probability is formally specified as:

$$P(V_i|V_1, V_2, \ldots, V_k) = \frac{P(V_i, V_1, V_2, \ldots, V_k)}{P(V_1, V_2, \ldots, V_k)} \tag{1}$$

In probability theory, both $P(V_i, V_1, V_2, \ldots, V_k)$ and $P(V_1, V_2, \ldots, V_k)$ can be calculated if the full set of *joint* probability distributions $P(U)$ is known. However, for large and complex systems, determining $P(U)$ is a computationally expensive process. On the other hand, since Bayesian networks explicitly specify the causal relationships between variables which leads to the existence of conditionally independent nodes, determining $P(U)$ only requires knowledge of the conditional probability distributions of every random variable node given its parent set. In other words, much fewer probability distributions are required for the calculation of conditional probabilities than in the standard case, thus making it a much more computationally feasible formulation. This fact is shown via the chain rule for Bayesian networks (equation (2)).

Chain rule in a Bayesian network: Let BN be a Bayesian network over $U = \{V_1, \ldots, V_n\}$. Then, the joint probability distribution P(U) is defined as:

$$P(U) = \prod P(V_i|pa(V_i)), \quad \text{where } pa(V_i) \text{ is the parent set of } V_i. \tag{2}$$

In addition to the advantage of computational feasibility, Bayesian networks have also been selected as the diagnostic engine in our accountability framework for the following reasons:

1. Its applications are systems that are modeled as directed acyclic graphs (DAGs), such as the service networks we are targeting.
2. It is capable of dealing with causal relationships in service networks.
3. It can handle uncertainty in service networks. In a service-oriented computing environment, service nodes can be considered as random variables with probabilistic reputation values. Furthermore, the causal relationships among services nodes may not be deterministic.

3.1 Transforming a service network into a Bayesian network

In a service network, all directed edges represent the execution flow among nodes. Whereas in a Bayesian network, all directed edges connecting nodes must represent direct causal relationships among nodes. Therefore, it is necessary to transform a service network topology into a Bayesian network topology.

As shown in Fig. 4, an atomic service node $service_i$ has k inputs and one output. The result of the output set is determined by the correctness of all inputs and $service_i$'s correct operation. In other words, all of the input sets, as well as the service node's execution, are the causes of the output. Therefore, as shown in the right part of Fig. 4, to represent the causal relationship correctly in the Bayesian network, $service_i$'s operation is extracted out as a separate root variable node (the rectangular node) and the output set is represented as an individual variable node (the elliptical node) in the Bayesian network. Since we have assumed that the network connection among services is flawless, the output node of a given service becomes the input node for its children.

Independence of a service's operation nodes After the transformation, every service's operation becomes a root node (i.e., the node has no parent) in the network. All such nodes $S_1, S_2, \ldots, S_N$ are independent random variables since: (1) any Bayesian network node V_i is conditionally independent of any subset of nodes that are not descendants of V_i given the parent set of V_i; (2) each service's operation node has no parent; and (3) no service's operation node is the descendant of any other service's operation nodes. The independence of service nodes matches our assumption in Sect. 2. Since every service is atomic, whether a service can function normally and meet SLA is not affected by any impact from the outside world.

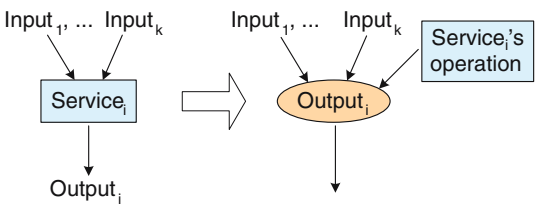

Fig. 4 Transforming a service network node to random variable nodes in Bayesian network

3.2 Configuring a Bayesian network's parameters

As a requirement for Bayesian network reasoning, *conditional probability tables* (CPTs)—the conditional probability distribution of each variable node over its parent set—are needed. For those nodes without a parent, the probabilities are not conditioned on any other node. These are called the *prior probabilities* of the variables [20,21].

Therefore, after a service network is topologically transformed into a Bayesian network, the following parameters need to be defined for the reasoning engine to execute:

1. *Prior probabilities of the nodes representing a service's operations (rectangular nodes)*. Those prior probabilities can be defined by the reputation of services, which ideally is the likelihood of a service to function correctly or to meet its SLA and is a continuous variable in [0,1]. The higher the value, the more consistent the service is. These values can be obtained from historical statistical feedback data, if available.
2. *CPTs of the nodes representing a service's outputs (elliptical nodes)*. CPTs describe the correlation among a service's inputs, operation, and output. Suppose a service V_i has k inputs. The CPT of V_i specifies the probability distribution of $output_i$'s states given the states of $input_1, input_2, \ldots, input_k$, and $service's\ operation$. The CPT of V_i should include a complete list (2^{k+1}) of probabilities in the format of

$$P(output_i | input_1, input_2, \ldots input_k, service_i's\ operation) \quad (3)$$

Our accountability framework allows the co-existence of both *probabilistic* and *deterministic* nodes. For deterministic nodes, the logic (AND, OR, k-out-of N) can be easily expressed in probabilistic formats, such as:

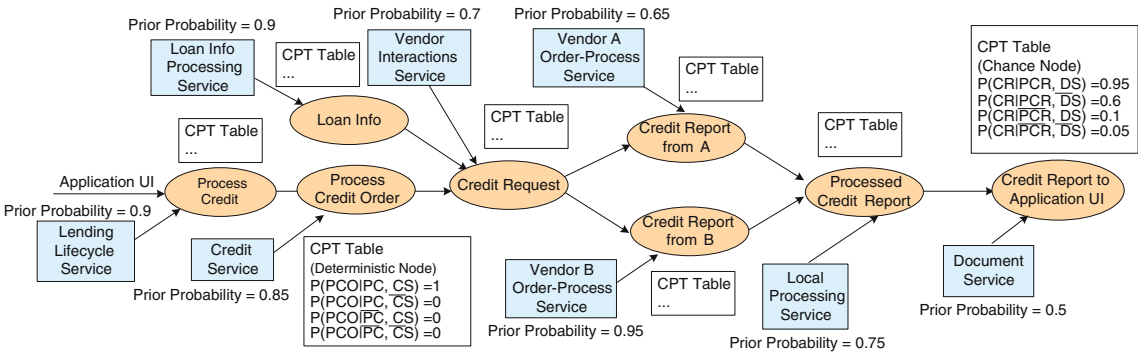

Fig. 5 The transformed Bayesian network for the credit pull service work flow with parameters configured

$$P(output_i|input_1, input_2, \ldots, input_k, service_i's\ operation) = 1 \text{ and} \quad (4)$$

$$P(output_i|input_1, input_2, \ldots, input_k, service_i's\ operation) = 0. \quad (5)$$

This makes our framework flexible for both deterministic and non-deterministic relationships among inputs, services, and outputs.

These CPTs can be obtained based on the combinations of theoretical considerations, historical cases, and subjective estimates. The service provider, as the expert of the service network's behavior, may provide such subjective estimates via CPTs of all probabilities. There may also exist a large database of cases about the service's past behavior from which these CPTs are extracted. CPT adaptation is the process of modifying the Bayesian network parameters to better reflect the experience represented by the accumulated cases. It is a very active area of research and some approaches, such as *Fractional updating* and *fading* [20], have been developed to solve this problem. In this paper, we assume that CPTs can be obtained from service providers. The adaptation of a service's CPT by using historical data is left as future work for our research.

Figure 5 shows the transformed Bayesian network for the credit pull business process with the parameters configured. To make the illustration clear, each variable's state space is set to binary, i.e., each variable's state is either *correct* or *fail*. In the example, each service's operation node (rectangular node) is specified with the prior probability (i.e., the reputation of that service) and each output node (elliptical node) is configured with a CPT. Those parameters are randomly assigned in the figure. In practice, the prior probabilities should be obtained from the services' past behaviors and the CPTs are produced by service providers and historical data.

3.3 Reasoning with Bayesian networks

Bayesian networks support any direction of inference, including *causal* (top-down) inference; and *diagnostic* (bottom-up) inference [19]. In our accountability model, diagnostic inference, i.e., reasoning about causes based on evidence, is performed since we want to know the malfunctions or SLA violations given some observed evidence. This reasoning flows in the opposite direction of the directed edges in a service network.

Given the observation of an evidence set $\{E_1, E_2, \ldots, E_n\}$ from corresponding output nodes, the *posterior probability* of every service operation node S (i.e., the conditional probability of S given an evidence set), denoted as Bel(S) can be reasoned. The inference results, which include a complete list of service operation nodes' posterior probabilities, provide the recommendations on which services are most likely to be accountable for the problem in terms of probabilities. We will choose the service with the maximum Bel(S) value and check its service logs to find out its correctness. Inspecting the service log may give us accurate knowledge of a service's operational correctness, but is costly and time-consuming. Therefore, using Bayesian network inference can save significant diagnosis time and cost since it shows the most likely problematic services. Once a service's state is diagnosed, its value can be added to the evidence set and the Bayesian network inference is conducted again, the correctness being enhanced after every iteration.

Metrics to Measure the Performance of a Diagnosis Network To check the effectiveness of the proposed Bayesian network diagnosis, we use two different metrics in our study.

1. *Service-log Checking Cost (SCC)*. Assume the network size is N and each node S_i costs C_i to perform a service log check. If the diagnosis process finishes after M services are checked, $SCC = \sum_{k=1}^{M} C_k / \sum_{k=1}^{N} C_k$. SCC quantifies the service-log checking cost.

The lower the *SCC*, the more diagnosis cost saving is achieved.

2. *Comprehensive Diagnosis Correctness* (*CDC*). Assume the number of problematic services is E_{total} and the diagnosis process confirms $E_{diagnosed}$ services to be problematic, $CDC = E_{diagnosed}/E_{total}$. The higher the *CDC*, the more comprehensively problematic services nodes have been discovered.

When to Stop The Bayesian network inference process provides the likely locations of problematic services. However, at run-time, there is no way to know exactly how many services have violated their respective SLAs. Therefore, a *stop_condition* should be defined to achieve a significant cost saving with an acceptable CDC. When the *stop_condition* is reached, the diagnosis process is terminated.

A *decision_threshold* can be used to define the *stop_condition*. When all $BEL(Service) <= decision\_threshold$, the diagnosis process stops. When the BEL(Q) for a service Q is low, it means that the diagnosis network is not certain about whether this service is problematic. Then it does not make sense to continue the service log check. However, the selection of *decision_threshold* must be careful. If the *decision_threshold* is set too high, the CDC may be too low to be acceptable. On the other hand, if the *decision_threshold* is set too low, although *CDC* is high, SCC could also be high. Thus, little cost saving is achieved. Therefore, the selection of *decision_threshold* is a trade-off for each application to decide based on its needs.

Assume there are N service nodes in the network, the diagnosis algorithm is described in Algorithm 1.

Algorithm 1 *Diagnosis Algorithm*

Input: $decision\_threshold, Evidence\_Set = \{E_1, E_2, \ldots, E_m\}$
1: Set $Error_{Diagnosis}$ to $\emptyset$
2: **repeat**
3: Infer the posterior probability Bel(S) for every service operation node
4: Look for $Service_k$ that has the maximum $BEL_{max}(Q)$ value in $\{\text{Bel}(S_1), \text{Bel}(S_2), \ldots, \text{Bel}(S_N)\}$
5: **if** $Service_k$'s log check is problematic **then**
6: $Service_k$ is one of the root cause nodes and is added to $Error_{Diagnosis}$;
7: **end if**
8: Add $Service_k$'s state to *Evidence_Set*
9: **until** $decision\_threshold > BEL_{max}(Q)$
10: **return** $Error_{Diagnosis}$

end

3.3.1 The complexity of the diagnosis algorithm

In the worst case, the complexity of the algorithm is $O(N(log(N) + O(Bayesian)))$, where $O(Bayesian)$ is the complexity of Bayesian network reasoning. Bayesian network inference algorithms are computationally complex. In the worst case, they are NP-hard. Therefore, the major complexity is from the Bayesian network reasoning. There exist several efficient algorithms [16,20], however, that make Bayesian network inference consisting of tens or hundreds of variables tractable. The inference time is between a fraction of a second and a few seconds using existing Bayesian network engines [22]. The size of most practical service networks is within the tens or hundreds range. Therefore, the diagnosis algorithm should be computationally tractable for real-world applications.

4 Evidence channel selection algorithm

The number and location of observable evidence items have a big impact on the performance of the Bayesian network diagnosis process. In the transformed Bayesian diagnosis network, all output nodes (elliptical nodes) are candidates for agents to monitor. However, it is not cost effective if agents collect all information from all output nodes since agents will be overburdened by the information collection process. In a distributed system, this could lead to very high overhead. Therefore we would like to select only the subset of nodes likely to provide the most fruitful diagnostic information to the agents.

4.1 Sensitivity analysis technology

To decide how to select evidence channels, we must decide which output nodes are the most informative to the observations of service operation nodes. *Sensitivity analysis* technology in the Bayesian network theory can be used to measure the influence of each service's output node on service operation nodes [23,19]. This technology was initially designed for medical diagnoses where there may be multiple tests available; and clinicians would like to perform a test that decreases the uncertainty of the diagnosis as much as possible. For example, if Bel(S) is 60% for a service node S, then there exists a 40% chance that this service node actually works well. While if Bel(S) is 90%, the suspicion of faulty behavior becomes small, which in turn decreases the liklihood of an unnecessary service log check.

Entropy reduction, or *mutual information*, of node Y to node X, is used to measure the influence of Y to X in Bayesian network sensitivity analysis [23,19]. Entropy uses Shannon's measure of mutual information as a measure of how much uncertainty is represented in a probability mass. Assume $x_1, x_2, \ldots$ are a set of mutually

exclusive states that X can be, the entropy of a probability distribution over variable X is defined as:

$$-\sum_{x\in X} P(x)log_2(P(x)) \tag{6}$$

Therefore, assume S is a service with a set of mutually exclusive states $s_1, s_2, \ldots, s_m$ and T is an output node with a set of mutually exclusive states $t_1, t_2, \ldots, t_n$. $ER(T,S)$, which denotes the expected benefit of obtaining the information of node T by S, i.e., the *entropy reduction (mutual information)* of T to S, is defined as:

$$ER(T,S) = \frac{Entropy(S) - Entropy(S|T)}{Entropy(S)} \tag{7}$$

Using Eq. (6) in Eq. (7), we get:

$$ER(T,S) = \frac{\sum_{t\in T}\left(\sum_{s\in S} P(s|t)\log_2(P(s|t))\right)P(t) - \sum_{s\in S} P(s)\log_2(P(s))}{-\sum_{s\in S} P(s)\log_2(P(s))} \tag{8}$$

where $P \times log_2 P = 0$ if $P = 0$.

$ER(T,S)$ measures the information about S that is shared by T, i.e., it measures how much uncertainty about S is reduced by knowing T. The larger the entropy reduction, the more information T contains about S. If S and T are independent, then T contains no information about S. In this case $Entropy(S) \equiv Entropy(S|T)$ and $ER(T,S)$ is zero. Knowing T does not give any information about S. If all information conveyed by S is shared with T, knowing T provides all necessary information about S. In this case $Entropy(S|T) = 0$ and entropy reduction is 100%.

4.2 Goal of evidence channel selection

If the goal of the evidence channel selection is to maximize the total entropy reduction over all services, an exhaustive simulation could be used to calculate this total given every possible combination of all output nodes. In this case, the number of entropy reduction calculations equals $N \times 2^N$, where N is the number of service nodes. This is not computationally feasible in most practical cases. Therefore, we constrain ourselves to a less complex goal:

Choose evidence channels to maximize the lower bound of the total entropy reduction over all services

It has been proved that entropy function is a convex function and the entropy reduction of obtaining one addition evidence item is never negative [24]. Since service error diagnosis is a bottom-up process, the non-descendants of a service S are independent of S unless at least one of S's descendants' state is known. Therefore, to reduce the complexity of the entropy reduction calculation, we set the lower bound of entropy reduction of S as the maximum entropy reduction contributed from one of its observable descendants. Formally, assume $T_1, T_2, \ldots, T_p$ are S's observable descendants set:

$$ER_{LowerBound(S)} = \max(ER(T_1,S), ER(T_2,S), \ldots, ER(T_p,S)) \tag{9}$$

The goal of evidence channel selection is to:

$$\max\left(\sum_{all\ services} ER_{LowerBound(S)}\right) \tag{10}$$

4.3 Mapping channel selection to the k-median facility location problem

With the goal of choosing evidence channels to maximize the total lower bound of entropy reduction over all services, this problem can be mapped to the *k-median facility location problem* (or simply the *k-median problem*). The k-median problem studies how to place facilities to serve clients effectively [25,26], such as the placement of fire stations within a city. The effectiveness is measured by the distance between a client and the facility that serves the client. In the k-median problem, we require that at most k facilities are to be placed and the total service benefit, measured as the sum of the distance of each client to the nearest open facility, is to be maximized.

More formally, in the k-median problem, we are given two sets: F, the set of facilities, and C, the set of clients. Let c_{ij} denote the benefit of serving client $i \in C$ by a facility $j \in F$; we could think of this as the distance between client i and facility j. The goal is to identify a subset of facilities $S \subseteq F$ of at most k facilities such that $\sum_{all\ clients} c_{i,j}$ is maximized.

The evidence channels selection problem can be mapped to the k-median problem as follows:

- Every output node is mapped to a facility in the k-median problem. If at most k output nodes can be selected due to the budget constraints, then at most k facilities can be open.
- Every service node is mapped a client in k-median problem.

- An output node T_j can serve service node S_i only when $ER(T_j, S_i) \neq 0$
- If T_j can serve node S_i, the distance of output T_j to service S_i is defined as:

$$D(T_j, S_i) = D(S_i, T_j) = ER(T_j, S_i), if\ ER(T_j, S_i) \neq 0 \tag{11}$$

- The goal of selecting output nodes to maximize the lower bound of the total entropy reduction over all services becomes the goal of maximizing the sum of distances of each service to its nearest facility in the k-median problem.

The k-median problem is NP-Hard. [25] provides local search heuristics for the *metric k-median problem*, which is the best known approximation algorithm for the k-median problem. The metric version of the k-median problem assumes that distances c_{ij} are symmetric and satisfy the triangle inequality. These two assumptions are satisfied in our model since:

1. The distance function is symmetric according to its definition;
2. Based on the assumptions in Sect. 2, each service is atomic. The behavior of every service is independent of other services. Therefore, since there is no edge between any two services, no triangle exists in the network. Hence the triangle inequality is also satisfied.

Algorithm 2 *The Local Search Heuristics Algorithm for the k-median Problem*

F: the set of facilities
K: the k number of facilities that are chosen
$op(K) : (K - A) \cup B$ *for* $A \subseteq K$ *and* $B \subseteq (F - K)$ *such that* $|A| = |B| \leq p$

1: $K \leftarrow$ an arbitrary feasible solution
2: **while** ∃ an operation **op** such that, $benefit(op(K)) \geq$benefit(K) **do**
3: $K \leftarrow op(K)$
4: **end while**
5: **return** K

end

As shown in Algorithm 2, the *local search heuristics algorithm* for the k-median problem first finds a feasible solution. Then it allows up to p facilities to be swapped simultaneously. It performs the swap over and over again until the solution achieves local optimality. The algorithm can achieve an approximation factor of $3 + 2/p$ with a complexity of $O(n^p)$, where n is the size of the network. The approximation factor and the complexity of this algorithm is a trade-off: the larger the p value, the lower the approximation factor and the higher the complexity.

4.4 Algorithm for evidence channel selection

In our research, the local search heuristics algorithm is used as the approximation algorithm for the metric k-median problem. The algorithm for evidence channel selection is shown in Algorithm 3.

Algorithm 3 *Algorithm for Evidence Channel Selection*

1: **for** each service S_i **do**
2: Calculate $Entropy(S_i)$
3: **for** each descendant output of S_i: T_j **do**
4: Calculate $ER(T_j, S_i)$
5: Set $D(S_i, T_j)$ according to equation 11
6: **end for**
7: **end for**
8: Perform k-median local search heuristic algorithm to find out the k number of evidence channels $T_1, T_2, \ldots, T_k$
9: Deploy agents to cover $T_1, T_2, \ldots, T_k$

end

In the worst case, the complexity of the algorithm is $(N^2O(Bayesian) + O(N^p))$, where p can be set to be a reasonably small integer and $O(Bayesian)$ is the complexity of Bayesian network reasoning. Therefore, the complexity of the overall algorithm is dominated by Bayesian network reasoning process. Again, Bayesian network inference is tractable when the network consists of tens or hundreds of variables. Therefore, our evidence channel selection algorithm is computationally tractable for real-world applications.

5 Performance study

To study the performance of our accountability framework, simulations have been conducted to analyze the effectiveness and efficiency of the Bayesian diagnosis network.

5.1 Experimental design

5.1.1 Generating services networks

We use BNGenerator [27], a generator for random Bayesian networks, to produce random service

networks. It guarantees that the generated networks are uniformly distributed in the space of graphs under consideration. BNGenerator can be tuned to accept constraints on the size of a network, the maximum degree for nodes, and the maximum number of edges in the network. These constraints limit how "connected" networks are. In our simulations, the size of randomly generated service networks ranges from 30 to 50 service nodes. The maximum in-degree (number of incoming edges) and out-degree (number of outgoing edges) for each node are both set to 2. The state space of each node variable is also set to binary.

5.1.2 Transforming a service network into bayesian diagnosis network

SMILE [22] is the Bayesian diagnosis network engine we use to perform probabilistic reasoning. Moreover, the SMILE API enables us to transform the randomly generated service network into a Bayesian diagnosis network based on the transformation rules in Sect. 3.1.

After the transformation is completed, the size of the diagnosis network is twice the size of the service network. Therefore, the size of the diagnosis network ranges from 50 to 100 in our experiments. For each service node, its prior probability is randomly generated in the range of [0.8, 0.9] based on a uniform distribution. For each outcome node, a conditional probability table (CPT) is also randomly generated. The CPT of an outcome node could represent the deterministic or non-deterministic relationship among inputs, service operation, and outputs. Moreover, a mixture of deterministic and chance (i.e., non-deterministic) service nodes can also be configured.

5.1.3 SLA violations generation and propagation

Simulated functional errors or SLA violations for a service node are determined by comparing its prior probability with a randomly generated number D in the range [0,1]. A service operates incorrectly if $D >= priori\_probability$; otherwise, the it operates correctly. For each service's outcome node, its state is also decided in the same way based on the comparison of a randomly generated number with a *probability* decided by states of its parent's services's outputs, this service operation's correctness, and this service's CPT. We topologically sort all nodes in a Bayesian diagnosis network before deciding the states of services' outputs. This ensures that the states of outputs from this service's parents are always known before this service's output state needs to be decided.

The states of all outcomes are recorded. However, only a certain percentage of the outcome states are selected and treated as observable evidence to perform Bayesian network diagnosis. The k in the k-median approximation algorithm is calculated by multiplying $Size_{service\ network}$ by *watch_percentage*. The *watch percentage* ranges from 30 to 70% in the simulation.

5.2 Experimental results

In the simulation, for each test case, 10 different network topologies were randomly generated. For each network topology, the diagnosis simulation was run 100 times. *SCC* and *CDC* are recorded in every round when a service node is being checked for its correctness. The number of data points of *SCC* or *CDC* equals the number of service nodes, where each data point is the average value of 100 simulation runs. We assume the $Cost_{(service\ log\ check)}$ for each service is the same. Therefore, as the network size is fixed, the *SCC* curve is always linear. We also calculate *Benefit*, which is defined to be $(CDC - SCC)$. It is used to indicate the cost-efficiency of performing the diagnosis process.

5.2.1 Performance with different evidence channel selection strategies

This experiment is designed to demonstrate the effectiveness and efficiency of the diagnostic network and evidence channel selection algorithm. With the same evidence limit k, three channel selection strategies are compared:

1. k-median-based channel selection algorithm, as defined in Sect. 4.4.
2. Top-cluster channel selection strategy, which selects channels from the top level of the flow, then one from each level down until k is reached. The channels selected using this strategy are clustered around the beginning of the flow.
3. Bottom-cluster channel selection strategy, which selects channels from the bottom level output nodes; then one from each level up until k channels are selected. The channels selected using this strategy are clustered near the end of the flow.

Figure 6 shows the performance comparison of the three channel selection strategies when the service network size is 50 and 50% of services' outputs are monitored as evidence. The x-axis is the number of

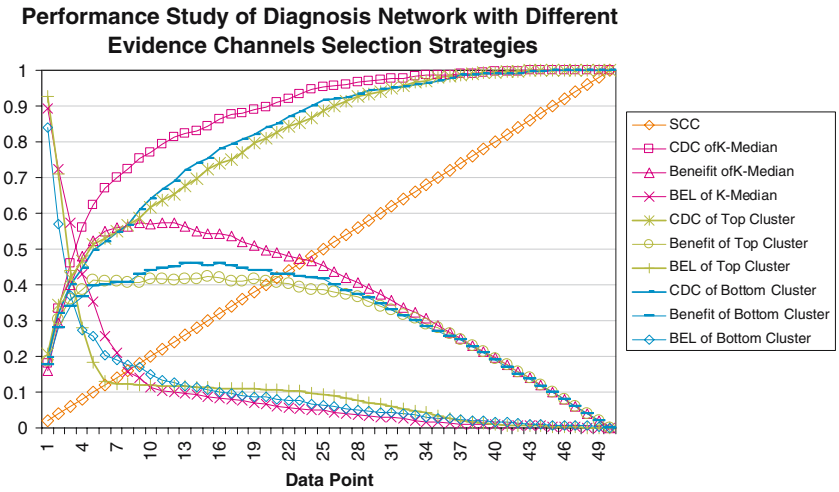

Fig. 6 Comparison of system performance with different evidence channel selection strategies. Service network size = 50, watch percentage = 50% ($K = 25$), $p = 1$, 30% probablistic nodes

service nodes inspected so far in each run. The top 3 curves report the *CDC* of various strategies and the bottom 3 curves report the *Benefit*. As the figures show, with the same number of evidence channels selected, both the *CDC* and *Benefit* of the evidence channels selection algorithm outperforms the other two strategies. For example, after 10 service nodes are inspected, the *CDC* of the evidence channel selection algorithm reaches 80%, while the *CDC* of top cluster and bottom cluster channel selection strategies are only about 60%. This demonstrates that the locations of evidence channels plays an important role for the performance of diagnosis network and our evidence channel selection algorithm can significantly improve it. Moreover, the results also show that our accountability model can achieve high diagnostic correctness and efficiency with low monitoring and inspection cost: with 50% of services' outputs monitored as evidence, the *CDC* can reach 80% after only 20% of service nodes are inspected.

In the following experiments, only the *k*-median-based evidence channel selection algorithm is used to select evidence channels.

5.2.2 *Differing service network sizes*

Figure 7 shows *CDC* and *Benefit* for service network size of 30, 40, and 50. The average problematic services for the three different network sizes are 2.91, 4.19, and 5.35, respectively. Since the reputation of each service node is set in the same range, the smaller the network size is, the fewer the problematical services there are. Therefore, as expected, *CDC* of smaller networks appear higher than the larger ones because the denominator $Error_{total}$ is small. In fact, the diagnostic network performs well in all network sizes. As shown in Fig. 7, when 20% of services (6, 8, 10 services for service network size of 30, 40, 50, respectively) are inspected, the *CDC* can reach 87, 78, and 77% for service network size of 30, 40, and 50, respectively.

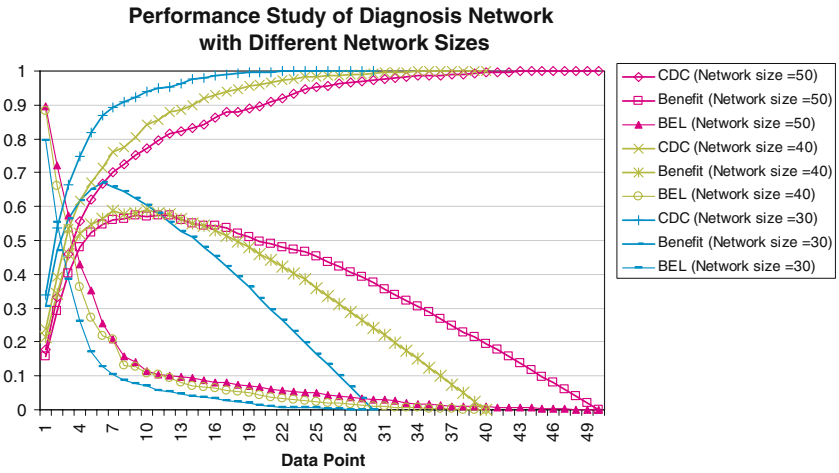

Fig. 7 Comparison of system performance with different service network sizes. Watch percentage = 50%, $p = 1$, 30% probablistic nodes

5.2.3 Different watch percentage

Figure 8 shows the diagnosis performance when the watchable evidence percentage is set to 30, 50, and 70%. The service network size is 50 for all runs. As expected, more watchable evidence can provide a better result, which explains the better *CDC* with same *SCC* as the watch percentage increases. After 20% of services are inspected, the *CDC*s of 30, 50, and 70% watchable evidence percentage reach 71, 77, and 85%. It is noticable that even when the watchable evidence percentage is quite low (30%), the *CDC* can still reach 71%, which is acceptable.

5.2.4 Differing heuristic p values

The parameter p is important in the local search heuristic k-median algorithm. It decides the algorithm's complexity and the approximation factor. Figure 9 shows the diagnosis performance when p is 1, 2, and 3. The network size is 50. As shown in the figure, the performance difference is not significant when p changes. It is because the approximation factor is the upper bound of $\frac{ApproximationSolution}{OptimalSolution}$. The actual approximation solution achieved is about 1.1 to 1.2 times of the optimal solution in simulation. Therefore, we should select p to be 1 to minimize the computational complexity.

5.2.5 Different percentages of probabilistic nodes

Figure 10 shows the diagnosis network performance when the percentage of probabilistic service nodes are set to 30, 50, and 70%. The network size is 50 for each run. As shown in the figure, all metrics remain almost unchanged as the percentage of probablistic service

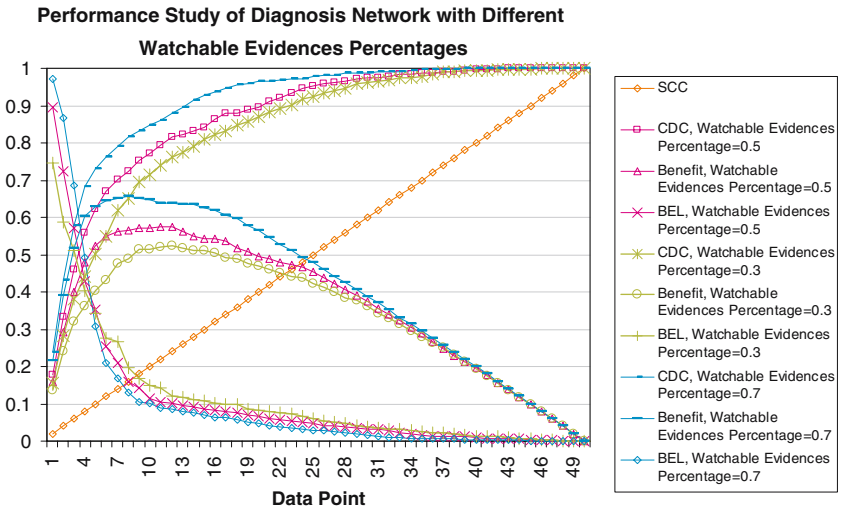

Fig. 8 Comparison of system performance with different watchable evidence percentages. Service network size = 50, $p = 1$, 30% probablistic nodes

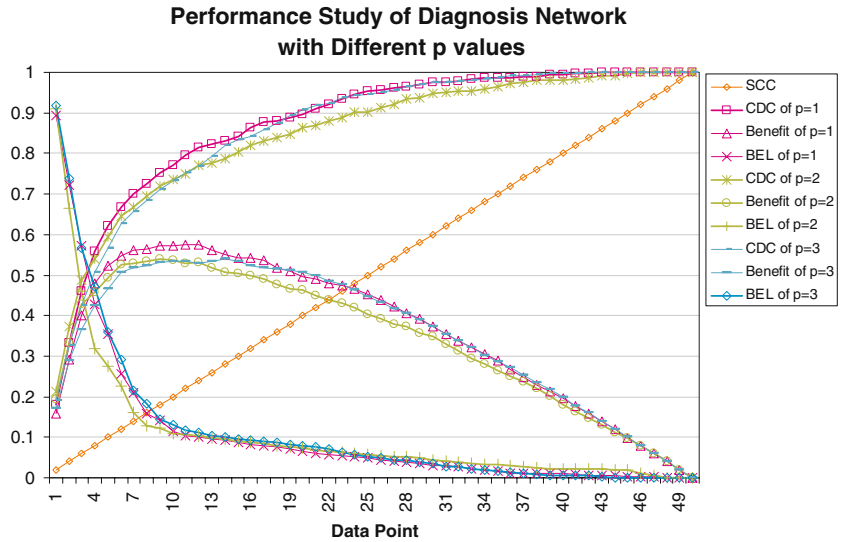

Fig. 9 Comparison of system performance with different p values in the k-Median heuristic local search algorithm. Service network size = 50, watch percentage = 50, 30% probablistic nodes

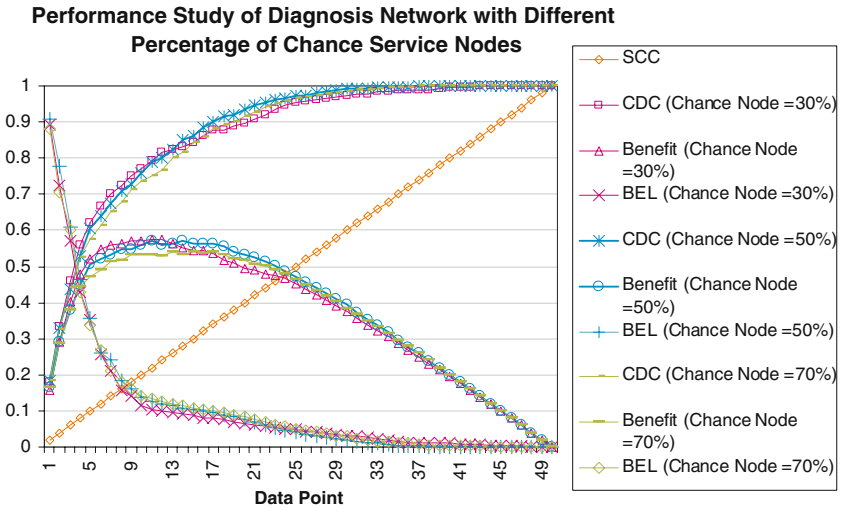

Fig. 10 Comparison of system performance with different percentages of probabilistic nodes. Service network size = 50, $p = 1$, watch percentage = 50%

nodes varies. The figure demonstrates that the percentage of probablistic service nodes does not affect the Bayesian diagnosis network performance, which suggests that our diagnosis framework has a broad application area, whether the service logic is deterministic or non-deterministic.

6 Related work

Error diagnosis and troubleshooting in SOA have become an active research topic in recent years. This section briefly describes the latest research, noting that none of which has reached the level of detail achieved by the accountability model introduced in this paper. Our accountability model is the first to leverage Bayesian networks to achieve a high diagnosis cost-efficiency in SOA when the correlations of a service's inputs, operation, and output are certain or uncertain. Services' historical reputation is novelly used to shed light on the diagnosis process and can be continuously learned based on diagnosis results. Moreover, an evidence channel selection algorithm is designed to discover the optimized deployment of information collection resources, which had not been explored into this level of detail in previous literature.

- In [28], monitoring and diagnostic services are incorporated into the QoS management framework. The diagnostic service reasons about the causes of degradation conditions in the networked enterprise system using a graphical model-based approach, i.e. causal networks. However, causal network reasoning is less than ideal as a QoS diagnosis reasoning engine. Without the capacity to deal with uncertain and incomplete information, causal network reasoning may produce ambiguous results when evidence collected is insufficient.
- In [29], a consistency-based diagnosis approach that spans across individual services is proposed to enhance fault analysis in SOA. A local diagnoser is added to each web service S. The global diagnoser coordinates the local diagnosers by exchanging messages. Hypotheses about incorrect outputs of a web service S may be related to a misbehavior of the service S itself, or to incorrect inputs from other services. The concept of the global diagnoser is similar to the *accountability authority* in our model. However, the consistency-based diagnosis can not deal with the uncertainty existing in the service network and does not leverage the historical reputation information of services. Moreover, the consistency-based diagnosis approach presented requires multiple message exchange steps between a global diagnoser and a local diagnoser before the root cause is ruled out, while our accountability framework can reason out the root cause with a one-time message exchanged between the AA and the agents. Furthermore, the diagnostic correctness of their approach relies on the ability to collect data from every service. Our accountability framework requires evidence collection at a subset of locations and therefore reduces the monitoring cost.
- In [30], a common architecture for the distributed diagnosis of Internet faults using autonomous agents is presented. CAPRI diagnoses faults using probabilistic relational models (PRMs) to combine the strengths of probabilistic Bayesian inference with the descriptive power of first-order logic. This

approach is quite similar to the our Bayesian network diagnosis where every service could be a deterministic or chance node. However, their research does not study how to select evidence channels to collect information. Furthermore, our accountability model provides a more comprehensive framework targeting the whole service deployment process including service network planing, selection, monitoring, diagnosis, and recovery.

– In [31], a formal, process-oriented ontology of an accounting information system is developed to ensure the reliability of data in information systems. In accounting processes, each control can cover certain error classes and eliminate errors to make accounting data reliable. The authors formulate a key control selection problem as a set-covering problem: to choose the fewest set of controls, while still being effective to ensure that the general ledger accounts are free of the types of errors in the target assertions. The idea is adopted to perform agent deployment in our accountability model: agents are selected and deployed using set covering algorithms to monitor exceptions occurring in service networks with minimum cost.

7 Conclusions

In this paper we present a novel accountability framework to make service process deployments manageable and dependable in Service-Oriented Computing (SOC). The Bayesian network model is leveraged to diagnose the root causes of malfunctions and service level agreement (SLA) violations in service networks when uncertainty exists in the correlations of services' inputs, operation, and output. Given failure alarms and evidence detected by monitoring agents in a service process, our proposed accountability mechanism is able to identify the individual service(s) that most likely causes the problem. Services' historical reputation information provide information for the diagnosis process and can be continuously updated based on diagnosis results. Furthermore, an evidence channel selection algorithm is designed to specify which locations in the service network agents should collect information that are most informative. This algorithm uses entropy reduction as the metric and models channel selection as the classic facilities location problem. The complexity analysis shows both the Bayesian network reasoning process and the evidence channel selection algorithm are computationally tractable. The performance study shows the efficiency of the diagnosis mechanism such that it can save significant diagnosis cost and fit a broad range of applications. Our integrated SOA deployment and management solution will benefit SOA management in terms of end-to-end QoS enforcement and efficient problematic service detection, diagnosis, and defusion.

Acknowledgements This research was partially supported by UC MICRO Grant 04-051, California Institute for Telecommunications and Information Technology (Calit 2) Emulex Fellowship, GeoSpatial Technologies, Inc. and was partially done while the author was visiting the Institute of Information Science, Academia Sinica, Taiwan in 2006.

References

1. Bichler M, Lin KJ (2006) Service-oriented computing. IEEE Computer 39(3):99–101
2. Huhns MN, Singh MP (2005) Service-oriented computing: Key concepts and principles. IEEE Internet Computing
3. Curbera F, Goland Y, Klein J, Leymann F, Roller D, Thatte S, Weerawarana S (2003) Business process execution language for web services, version 1.1. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel
4. Ross-Talbot S, Fletcher T (2006) Web services choreography description language: Prime. http://www.w3.org/TR/ws-cdl-10-primer/
5. Neumann PG (2006) Risks relating to system compositions. Commun of ACM 49(7)
6. Yu T, Lin KJ (2005) Service selection algorithms for composing complex services with end-to-end QoS constraints. Proc. 3rd International Conference on Service Oriented Computing (ICSOC2005), The Netherlands Amsterdam
7. Yu T, Lin KJ (2005) Adaptive algorithms for finding replacement services in autonomic distributed business processes. The 7th International Symposium on Autonomous Decentralized Systems, Chengdu, Jiuzhaigou, China
8. Lin KJ, Hsu JY, Zhang Y, Yu T (2006) A distributed reputation broker framework for web service applications. J E-Commerce Res 7(3):164–177
9. Lin KJ, Lu H, Yu T, Tai CE (2005) A reputation and trust management broker framework for web applications. IEEE Conference on e-Technology, e-Commerce and e-Service (EEE'05), 262–269
10. Horsch K (1996) Results-based accountability systems: Opportunities and challenges. The Evaluation Exchange II(1) http://www.gse.harvard.edu/hfrp/eval/issue3/theory1.html
11. Acharya M, Kulkarni A, Kuppili R, Mani R, More N, Narayanan S, Patel P, Schuelke KW, Subramanian SN (2005) SOA in the real world experiences. Proceedings of the 3rd International Conference on Service-Oriented Computing 437–449
12. Ponnekanti SR, Fox A (2002) SWORD: A developer toolkit for web service composition. 11th World Wide Web Conference, Honolulu, Hawaii
13. Yu T, Lin KJ (2005) A broker-based framework for QoS-aware web service composition. In: IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-05), Hong Kong, China
14. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) The set-covering problem. Introduction to algorithms, second edition, MIT Press, McGraw-Hill, pp 1033–1038
15. Zhang Y, Lin KJ, Klefstad R (2006) DIRECT: arobust distributed broker framework for trust and reputation management. IEEE Conference on e-Technology, e-Commerce and e-Service (EEE'06)

16. Pearl J (1988) Probabilistic reasoning in intelligent systems. Morgan Kaufmann, San Mateo
17. Heckerman D, Breese JS, Rommelse K (1995) Decision-theoretic troubleshooting. Commu ACM 38(3):49–57
18. Lerner U, Parr R, Koller D, Biswas G (2000) Bayesian fault detection and diagnosis in dynamic systems. AAAI/IAAI, pp 531–537
19. Korb KB, Nicholson AE (2004) Bayesian artificial intelligence. Chapman & Hall/CRC, London, UK
20. Jensen FV (2001) Bayesian networks and decision graphs. Springer, Heidelberg
21. Nilsson NJ (1998) Artificial intelligence: a new synthesis. Morgan Kaufmann Publishers, San Francisco
22. DSL (2006) SMILE (structural modeling, inference, and learning engine). Decision Systems Laboratory(DSL), School of Information Sciences, University of Pittsburgh, Pittsburgh, http://genie.sis.pitt.edu/
23. Jensen FV, Liang J (1994) drHugin: a system for value of information in bayesian networks. Proceedings of the 1994 Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pp 178–183
24. Jensen FV, Liang J (1992) A system for hypothesis driven myopic data request. Technical Report R-92-2021, Department of Mathematics and computer Science, Aalborg University, Aalborg
25. Arya V, Garg N, Khandekar R, Munagala K, Pandit V (2001) Local search heuristic for *k*-median and facility location problems. ACM Symposium on Theory of Computing, pp 21–29
26. Charikar M, Guha S, Tardos E, Shmoys DB (1999) A constant-factor approximation algorithm for the *k*-median problem (extended abstract). ACM Symposium on Theory of Computing, pp 1–10
27. Ide JS, Cozman FG (2002) Generating random bayesian networks. Proceedings on 16th Brazilian Symposium on Artificial Intelligence (SBIA 2002), Advances in Artificial Intelligence, Springer, Berlin, pp 366–375
28. Wang G, Wang C, Chen A, Wang H, Fung C, Uczekaj S, Chen YL, Guthmiller W, Lee J (2005) Service level management using QoS monitoring, diagnostics, and adaptation for networked enterprise systems. EDOC Enterprise Computing Conference, 2005 Ninth IEEE International, pp 239–248
29. Ardissono L, Console L, Goy A, Petrone G, Picardi C, Segnan M (2005) Enhancing web services with diagnostic capabilities. Third IEEE European Conference on Web Services (ECOWS 2005)
30. Lee G (2006) CAPRI: a common architecture for autonomous, distributed diagnosis of internet faults using probabilistic relational models. In: the First Workshop on Hot Topics in Autonomic Computing (HotAC I) in conjunction with the 3rd IEEE International Conference on Autonomic Computing (ICAC-06)
31. Krishnan R, Peters J, Padman R, Kaplan D (2005) On data reliability assessment in accounting information systems. Inform Systems Res 16:307–326