A Secure Service-Specific Overlay Composition Model for Cloud Networks

Ismaeel Al Ridhawi and Yehia Kotb

*College of Engineering and Technology, American University
of the Middle East (AUM), Eqaila, Kuwait
E-mail: {Ismaeel.Al-Ridhawi; Yehia.Kotb}@aum.edu.kw*

Received 31 August 2017; Accepted 22 October 2017;
Publication 20 November 2017

Abstract

Mobile cloud service subscribers acquire and produce both simple and complex services resulting in tremendous amounts of stored data. Increasing demand for complex services coupled with limitations in current mobile networks has led to the generation of cloud service composition solutions. In this paper, we introduce a cloud mobile subscriber data caching method that allows cloud subscribers to retrieve data in a faster and more efficient way. The solution also allows the retrieval of user-specific composed data through a service-specific overlay (SSO) composition technique. Additionally, a Workflow-net based mathematical framework called Secure Workflow-net is proposed to enhance security attributes of SSOs. Simulation results show increased successful file hit ratio and demonstrate how task coverage is achieved in an adequate timely manner when constructing service composition workflows.

Keywords: Petri-net, Workflow-net, Overlay network, Service-specific Overlay, Cloud, Fog.

1 Introduction

Cloud has played a dominant role in providing both data storage and processing capabilities to its subscribers. Traditionally, cloud subscribers access data

Journal of Software Networking, 221–240.

doi: 10.13052/jsn2445-9739.2017.011

This is an Open Access publication. © 2017 the Author(s). All rights reserved.

through a remote data storage site. This incurs both high delays and network bandwidth overload. With the advances in mobile edge devices, data caching and service-specific overlay composition techniques now can be used to both store and access data for further processing when needed.

Overlay networks are created as an abstraction layer to the underlying physical network using software to run multiple virtualized network layers to provide application, networking, or security benefits [1]. With the emergence of the fog-to-cloud (F2C) computing paradigm [2], edge mobile devices are used to provide computing, storage and networking services to achieve reduced network bandwidth usage and energy consumption in data centers [3, 4]. Services are composed using edge nodes to provide composite and enhanced services acquired by cloud subscribers without reliance on remote cloud storage and computing sites. This reduces the communication delay and provides faster access to services [5].

In this paper, we introduce a distributed cloud storage and retrieval method that supports the composition of services requested by cloud subscribers. A data replication and caching technique is developed such that cloud data is decomposed into a set of files. The decomposed set is then cached into the mobile edge devices. When a data request is submitted, the query is first assessed to determine whether the request can be answered by the cache of the registered edge node. If the requested data (or part of it) is not available in the device's cache, then data is both retrieved and composed (if needed) from other nearby devices within the SSO composition path.

Petri-net provides a solution towards service composition in which the available capabilities of edge nodes are merged together to achieve the requested task [6]. Workflow-net provides an extension to Petri-net and has been adopted lately for service composition to produce a more robust and sound solution [7–9]. Information system security has been a hot topic for many decades, e.g. [10]. An information system is considered to be secure if it has well-defined security measures and characteristics such as authenticity, confidentiality and integrity [11]. Security in service composition has been considered in the literature [12, 13], but has been overlooked when Workflow-nets are used to compose services. In this paper, a closed form Workflow-net based mathematical model is proposed that ensures such security characteristics are satisfied both structurally and behaviorally. The solution is an extension to Workflow-nets in which we call it Secure Workflow-net.

This paper is organized as follows: Section 2 outlines some of the previous work in data caching and composition techniques for secure cloud services. Section 3 outlines the problem and solution overview. Section 4 discusses the

decomposition and caching process. Section 5 outlines the SSO composition process. Section 6 provides an introduction and overview to what Petri-nets and Workflow-nets are and their characteristics and how SSOs are modeled as workflow-nets. Section 7 illustrates the proposed Secure Workflow-net framework and presents the theorem of serializability and its proof. Simulations are conducted in Section 8. Finally, Section 9 concludes the paper.

2 Related Work

With the increase in the number of cloud users and data, and the improvement of mobile subscribers' resources, an enhanced cloud storage framework, preferably one that considers the use of subscribers' resources is compelling. The authors in [14] introduce a client-side cloud cache method called the cloud cache eviction strategy which relies on the contemporaneous proximity principle that combines both spatial locality and temporal affinity together. The authors compare and evaluate their proposed strategy to other different well-known caching strategies, specifically least recently used (LRU), greedy-dual size frequency (GDSF), and least frequently used dynamic aging (LFU-DA). Results show an improvement in terms of hit ratio and delay saving ratio.

In [15], the authors propose a QoS-aware service distribution strategy in Fog-to-Cloud (F2C) scenarios. The work aims at achieving low delay on service allocation by using service atomization in which services are decomposed into distinct sub-services called *atomic services* tailored to enable parallel execution. These atomic services are executed on constrained edge devices. Tasks with higher requirements are allocated on more robust resources and executed in parallel with the atomic services. A control plane within the F2C architecture exists that is responsible for the distribution of the atomic services among the available edge nodes. The authors model the service allocation problem as a multidimensional knapsack problem (MKP) [2].

Developing efficient solutions to achieve automated service composition has been a hot research area since the introduction of web services. The authors in [16] introduce a service specific overlay composition method which considers semantic similarity and semantic nearness. A decentralized solution is adopted such that each overlay node will determine how semantically similar it is to the other mobile nodes in a dynamic network environment. According to the similarity score provided by a node, a decision is made to either accept or reject a service node from the overlay composition path.

Secure service composition solutions have been discussed earlier. The authors in [17] present a privacy preserving access control model and

framework for secure service provisioning and composition. To create secure service compositions, the solution ranks possible chains of composite services according to the users' preferences and sensitivity level of their data. An access request for a service is permitted if the requester's attribute certificates, contextual conditions, and privacy preferences are in compliance with the access control policies specified by the service provider. In [18] the authors propose an information declassification mechanism used for secure service compositions. The declassification mechanism is based on cryptographic operations and information flow security requirements. Mobile service nodes then cooperate with each other to complete the secure composition procedure.

Despite the abundant existing studies on service composition for overlay networks, to our knowledge, the presented work is the first to employ a partial data caching strategy and secure Workflow-net solution used to compose SSOs to provide aid for the F2C computing paradigm.

3 Problem and Solution Overview

By disregarding costs associated with accessing remote data, latency and bandwidth become a main issue when considering cloud network performance especially when the same data is being repeatedly accessed by groups of users. It is obvious that bringing the most used data closer to cloud subscribers can improve the network performance, e.g., reduction in download latency and network congestion [19, 20]. A data caching and selection technique has been developed to overcome data accessibility issues for cloud systems. Frequently accessed data is cached on mobile cloud subscriber devices. Replicas and cached data are regularly updated through notifications sent from the cloud. A data decomposition and caching technique (Section 4) is used to decompose files into *blocks*.

Additionally, data cached on mobile cloud subscriber devices is used to compose composite services for subscribers. This process is achieved through the construction of SSOs. For instance, assume that a cloud subscriber is requesting a particular service that may not exist at a single edge node nor the cloud (e.g. a media content with certain modifications and enhancements). To provide the subscriber with the requested service, an overlay must be constructed using the edge nodes. Each edge node adds a service to the original media content (e.g. the addition of subtitles, encoding conversion, or size compression), such that a sequence of services added by multiple edge nodes will lead to the composite requested service.

Figure 1 Illustrative example of data decomposition and caching and SSO composition in cloud systems.

Figure 1 illustrates the overall process starting with file decomposition at the cloud layer, then replicating or caching the files at select edge nodes at the fog layer, then finally composing an SSO using edge nodes to deliver composite services to cloud subscribers.

4 File Decomposition and Caching

We define a *block* to be a set of files that if decomposed any further will not add any more value to the decomposition. A block is an answer to a single file query from a single data set. Any more complex queries can be answered by mathematically composing those blocks together. When a data request is acquired by a cloud subscriber, blocks are mathematically checked to see whether the fog (edge devices at the fog layer) can fulfill the request. This is achieved by decomposing the submitted request into its own blocks and then comparing those blocks with the cached ones.

Suppose a cloud subscriber is requesting a media file with English subtitles and audio enhancements. Three blocks will exist (i.e. original media file, English subtitles for the file and audio enhancements for media content). Hence, when data is cached from the original cloud storage site to fog mobile edge devices, blocks from a single file are cached separately either to the same edge device or different devices. Thus, when subscribers request access to data, the request can be either delivered using one edge device or multiple edge devices.

A set of files or part of files are selected for caching based on the file access requests executed by cloud subscribers. The process of file splitting is based on the file size itself and the block size. A file is defined in terms of the number of blocks it contains (S_{file}). A block is composed of the following properties: the size of the block (S_{block}), the time required to create a block (T_{block}), and the time to locate a block (T_{locate}), which considers the transmission and execution times (T_{exec}). The time required to determine which block should be accessed (T_{access_k}) is also considered when caching files partially.

The transmission time for sending a file f_m from a set of M data files $F = \{f_1, \dots, f_M\}$ which are composed of K blocks (b_k) such that $f_m = \{b_1, \dots, b_K\}$ is calculated as follows:

$$T_{trans_m} = \sum_1^K (T_{block_k} + T_{locate_k}) \quad (1)$$

For the job set $\{J_1, \dots, J_N\}$ which requires an input file f_m or part of a file b_k , the execution time for each job n is calculated as follows:

$$T_{exec_n} = \sum_1^K \left(T_{trans_m} * \frac{S_{file_m}}{S_{block_k}} \right) \quad (2)$$

The process of caching blocks into edge devices is conducted such that the cost of creating the required partial files is minimized as follows:

$$minimize \sum_1^N \sum_1^M (T_{trans_m} + T_{exec_n}) \quad (3)$$

Once files are replicated and cached at the fog layer, the next step is to construct secure SSOs to support the service composition process needed to deliver complex services to cloud subscribers.

5 SSO Composition

The service composition problem is modeled as a set of processes. A process is a series of actions performed to achieve a task. We assume that each action is performed by an edge node in the fog. Events are the driving force behind an action. For instance, a movie that exists on an edge node which has received audio enhancements is considered an event. Provided this event an action can now be performed by another edge node. We assume that there is a set

$\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_m\}$ of primitive events that cannot be fragmented into simpler events, where m is the number of distinct events in a process. We further classify Λ into two distinctive sets: Λ^0 and Λ^c . The former is the set of events in which a process begins with, such that $\Lambda^0 = \{\lambda_1^0, \lambda_2^0, \dots, \lambda_j^0\}$. The latter is the set of events that follow, such that $\Lambda^c = \{\lambda_1^c, \lambda_2^c, \dots, \lambda_k^c\}$. The properties for Λ are defined as follows:

$$\Lambda^0 \cup \Lambda^c = \Lambda \quad (4)$$

$$\Lambda^0 \cap \Lambda^c = \emptyset \quad (5)$$

$$\forall \lambda \in \Lambda^0, \bullet\lambda = \emptyset \quad (6)$$

$$\forall \lambda \in \Lambda^c, \bullet\lambda \neq \emptyset \quad (7)$$

where $\bullet\lambda$ is the set of earlier events that lead to λ .

The service composition process is based on a learning approach, such that edge node log files from previously successful composition processes are used to recreate similar compositions for similar events in the future. Edge node log files include process descriptions such as the events that lead to an action to be considered, the events which occurred after applying the action, list of edge nodes that preceded the considered node in the composition, and the edge node that followed the considered node in the composition.

A set of events that are candidates for processing $\hat{\Lambda}$ are first determined and selected according to (8), such that it contains all events that could start a particular process and do not depend on any previous events Λ^0 , union with the set of events that depend on the already executed events Λ^x .

$$\hat{\Lambda} = \Lambda^0 \cup \Lambda^x \quad (8)$$

Those candidate events provide an overview of the events' sequence in a service composition process. The set $\hat{\Lambda}$ contains all possible events that may exist in a single composition process. The probability of occurrence is then derived for each event and is calculated according to (9) and (10).

$$bel(\lambda_i) = \int_{j=1}^n P(\lambda_i|\lambda_j) \times MAX(\Lambda^x, \sigma(||\lambda_i \cap \Lambda^0||)) d\lambda \quad (9)$$

where $P(\lambda_i|\lambda_j)$ is the probability for the event λ_i to occur given the occurrence of the event λ_j . This probability is calculated according to a uniform distribution function. $||\lambda_i \cap \Lambda^0||$ is the magnitude of the intersection between event λ_i and the set of events in which a process begins with, σ is a step function which

produces 0 if the magnitude is 0 and 1 otherwise. A normalized probability score is determined for each event as follows:

$$P_N(\lambda_i) = \frac{bel(\lambda_i)}{\sum_{j=0}^n bel(\lambda_j)} \quad (10)$$

The SSO composition path is constructed by selecting the events with the highest occurrence probability. Such that, edge nodes that perform certain actions (i.e. addition of services) leading to the selected events are chosen for the SSO composition path.

6 Modelling SSOs as Workflow-nets

6.1 Petri-nets and Workflow-nets

A Petri-net is a directed bipartite graph with two types of nodes, namely *places* (circles) and *transitions* (solid rectangles). Transitions model actions that may occur. Places are pre- or post-conditions for the transitions in which they are connected to. Places and transitions are connected via directed weighted *arcs*. If these arcs are not weighted then the *weight* is assumed to be one. These integer weights determine the number of activities that flow from places along the arcs per transition. Activities are called *tokens* (small solid circles that reside in places). The distribution of tokens over places is called a *marking*. Figure 2 depicts an example of a Petri-net structure.

When arcs run from places to a transition, these places are considered input places to the transition. On the contrary, when arcs run from a transition to places these places are considered output places to the transition. A transition in a Petri-net is enabled if and only if there are tokens in all the input places to the concerned transitions and each input place contains a number of tokens that is greater than or equal to the weight of its connecting arc. After a transition

Figure 2 Petri-net structure.

is enabled, it will eventually fire by consuming tokens from its input places and producing tokens in its output places.

Workflows have been used in the management of distributed information systems [21, 22]. Workflow-nets (WF_{net}) are used to model the structural and dynamic behaviors of workflows. The structural behavior of a workflow defines task dependencies and their structure which guarantees the desired output. The dynamic behavior of a workflow is how the structure of the workflow reacts online with activities that are handled by the workflow. A Workflow-net is a special type of Petri-net that has two special places, i and o , where i is the only place that does not have any input transitions and o is the only place that does not have any output transitions. Nodes i and o are called the *source* and *sink* nodes. Workflow-nets are preferred over normal Petri-nets in distributed systems due to the fact that they guarantee the success of a process. This supports the property of soundness [8, 9].

6.2 SSO Workflow-net Model

The composed SSO is modeled using a Workflow-net, in which workflow transitions perform actions (i.e. services added by edge nodes). As defined in Section 3 actions are represented as tokens residing in places (i.e. events preceding the appliance of actions). A transition executes (i.e. performs an action) after being enabled. The result of the execution is the removal of tokens from each of the transition's input places and the creation of tokens in each of its output places. Figure 3 depicts a Workflow-net model for the service composition process example outlined in Section 5. The figure outlines a media service composition problem that requires the addition of media enhancements to the original media content to produce a composite media service. Each transition characterizes an action that must be carried out by an edge node. Each place depicts an edge node with an action (service) awaiting to be applied

Figure 3 Workflow-net model for a service-specific overlay example.

(added) to the media content. As the media content (represented as a token residing in a place) is modified by the edge nodes, the token will eventually end up residing in the last place (i.e. last edge node to add a media enhancement).

7 Secure SSO Workflow-net Model

7.1 Security Properties

According to [8], an information system is considered secure if it satisfies certain properties. The following outlines those properties:

Confidentiality (χ): is the availability of network resources (node services) and data for only those who are entitled to access such concerned resources and data. We mathematically define confidentiality as follows:

$$\chi(a, r) = \begin{bmatrix} x_1, & x_2, & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots, & x_k \end{bmatrix} \quad (11)$$

where $X = x_1, x_2, \dots, x_n$ is a vector that represents different access levels. If $x(3, 4) = x_6$, this means that node a_3 has access level x_6 on resource r_4 .

Service Integrity (ψ): is the availability, reliability and completeness of the network. The availability of resources is demonstrated though a vector as follows:

$$V = [\nu_1, \nu_2, \dots, \nu_k] \quad (12)$$

where ν_i is the availability of a resource identified by index i , and k is the maximum number of resources.

To assign a resource to a node, two conditions must be satisfied:

1. The resource has to be available.
2. The node must have an accessibility privilege to that resource.

which is mathematically represented by the following equation:

$$\forall S(a_i, r_k), \exists \chi(a_i, r_k) \quad \text{and} \quad \nu_k \neq 0 \quad (13)$$

where S is the assignment probability, a_i is node i in the network, r_k is resource k , and ν_k is the availability of that resource. The assignment probability matrix is the product of the availability vector and the transpose of the accessibility matrix:

$$S = [V \times \chi^T] \quad (14)$$

7.2 Secure Workflow-net Framework

The secure Workflow-net framework WF_s is mathematically defined as follows:

$$WF_s = \langle WF_{net}, A, \chi, R, \Pi, \xi \rangle \quad (15)$$

where A is the set of nodes available for service composition in the network, χ is the accessibility of every node to resources, R is the set of available resources, Π is the routing mechanism that routes resources into their sub-workflows, and ξ is a sub-Workflow-net that binds the input to the output for access rejection or error handling cases.

For a WF_s to be structurally valid, the following constraints must be satisfied:

1. WF_{net} is a sound workflow net,
2. $\forall r \in R$ and $a \in A, \exists(a, r) \in \chi$,
3. $\Pi \cap WF_{net} \neq 0$,
4. ξ is a sound WF_{net} that binds input with output.

The main Workflow-net modeling the composition has to be a sound Workflow-net. The nodes have some defined access level on resources. The routing mechanism is tightly bound to the Workflow-net to guarantee the flow of tokens to the right sub-workflow. There is also a sub-workflow that drives the access rejection tokens to the output.

In our previous article [9], we have demonstrated the soundness of the workflow through a set of theorems and lemmas. Next, we define the serializability of the workflow.

7.3 Workflow Serializability

In addition to the notion of soundness achieved by the workflow, serializability is also assured. Serializability is the property of executing concurrent processes in a workflow as if they are executed sequentially. This feature is very important as it signifies the feature of atomic execution. In other words, the execution of one activity of a workflow will never affect the result of execution of a concurrent activity.

Theorem 1. *A sound secure workflow-net is a serializable workflow-net.*

Proof:

- Since $\forall a \in A$ and $r \in R, \exists \chi(a, r) || P(\xi | \chi) \neq 0$ if and only if, $\exists \pi \in \Pi || \pi(a) = \xi$.

- And since $\forall M \in \Pi$, M will eventually reach the output as the secure workflow-net is sound.
- Therefore $\forall t \in T \cap \xi$, $M \in \bullet t_{r-1}$ and $M \in t_r \bullet$.
- Therefore since $\forall M_i \cap M_j = \emptyset$, markings will not affect the execution of transitions for other markings.
- Therefore, the workflow is serializable.

8 Simulation Results

8.1 SSO Composition Results

To generalize the problem and test the system's capability regardless of the type of service requested, a simulator was developed to analyze three different solutions: i) a cooperative non-secure Workflow-net, ii) a cooperative secure Workflow-net, and iii) a non-cooperative solution. The first considers a solution which uses a probabilistic learning approach where edge nodes cooperate to deliver the requested composite service. The second solution considers a similar solution to the first one but adds a layer of security as described in the previous section. The third solution disregards node cooperation and hence services are composed using a single edge node. The goal of these simulation tests is to demonstrate the correctness of the secure workflow-net and that compositions can be adequately established.

The input to the simulator consists of a process in the form of a linear logic expression with operators described in [21]. Other input parameters consist of a set of nodes, each with a set of resources, expressed as Workflow-nets. Each resource corresponds to one service defined in the process, along with the cost associated with performing that service. When a node is granted access to a resource, the cost for executing the service is randomly determined with a normally distributed variable. To simplify the simulations, the execution time is computed as the total execution cost in the Workflow-net.

The delay incurred to complete a certain number of service requests is considered in the first experiment test. Results depicted in Figure 4 show that the non-cooperative solution incurs the most delay when compared with the other two cooperative solutions. The secure composition method incurs a small delay penalty compared to the non-secure composition approach and show stabilization in the time needed to complete the service requests.

The second experiment considers a service request which requires a set of sub-services to be performed to achieve the task. Results depicted in Figure 5 show that the cooperative approach outperforms the non-cooperative method. Additionally, the secure cooperative approach incurs a small delay and shows

Figure 4 Time needed to complete a service composition request.

Figure 5 Time needed to complete a service composition request when multiple actions are required for an individual service.

similar time delays for the composition when compared to the non-secure cooperative solution.

The third experiment considers evaluating the secure cooperative method as the number of nodes used for cooperation increased. Results depicted in Figure 6 show that the secure approach does not incur an increased time burden when compared to the non-secure approach.

8.2 Data Caching Results

To evaluate the effect of the data replication and caching approach on the cloud system, a number of simulation experiments were performed using GridSim [23]. Evaluations were conducted to compare the proposed Block Caching (BC) solution (i.e. partial file caching) against a Full File Caching (FFC) solution. The network is modeled as a graph $G = (N, B)$ where the set

Figure 6 Time needed to complete a service composition request when multiple edge nodes are needed.

of nodes $N = \{1, \dots, n\}$ represent storage resources on edge nodes and B represents the bandwidth. All nodes are assumed to have uniform bandwidth and processing capabilities. Different scenarios are simulated by varying the number of files, number of job requests and storage capacity of edge nodes in which there exists: three cloud storage sites, 150 edge devices, between 100 GB to 500 TB of files, up to 2GB/sec of bandwidth, cache storage size between 5 TB to 20 TB and workload size of up to 1500 jobs.

The BC technique has an advantage over the FFC method in terms of file diversification. For instance, for a cache size of 5 TB, 2000 different sets of file blocks can be stored using the BC method. Whereas using the FFC technique, we can only have up to 250 different blocks of files stored.

Performance of the two techniques was measured in terms of cache hit ratio calculated as follows:

$$\text{Cache Hit Ratio (\%)} = \frac{\text{Total cache hit}}{\text{Total number of requests}} \quad (16)$$

Results depicted in Figure 7 show that the cache hit ratio for caching a partial file is higher than the cache hit ratio of caching a full file. Moreover, when the cache size increases, the cache hit ratio of caching partial files increases compared to cache hit ratio of caching full files.

Experiments were also conducted to analyze the effects of varying the cache size. Results outlined in Figure 8 show that the adopted BC technique outperforms the FFC method such that as the cache size increases, the miss ratio (equal to $1 - \text{Cache Hit Ratio}$) is decreased by more than 10%. This provides an indication of the improvement in service response time and bandwidth utilization.

Figure 7 Comparing hit ratio using the BC and FFC techniques.

Figure 8 Comparing miss ratio using the BC and FFC techniques.

9 Conclusion

This paper introduced a distributed cloud storage and retrieval method, where files are decomposed into a set of file blocks and then cached on mobile edge devices. When a data request is submitted, the query is first assessed to determine whether the request can be answered by the cache of the registered edge node. If the requested data is not available in the device's cache, then data is retrieved from other nearby devices. Additionally, a service-specific overlay composition method is developed to provide composite services for cloud subscribers as per their request. The available capabilities of edge nodes are merged together to achieve the requested composite service. The service composition process is modeled as a workflow-net while ensuring network security characteristics are satisfied. Simulation results show that the proposed block caching technique has a significant impact on the cache hit ratio. Additionally, results show that the proposed technique incurs minimal delay overhead when composing service overlays.

References

- [1] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim (2005). “A survey and comparison of peer-to-peer overlay network schemes,” in *IEEE Communications Surveys & Tutorials*, (Second Quarter 2005), Vol. 7, No. 2, pp. 72–93,.
- [2] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan and G. J. Ren (2016). “Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems,” in *IEEE Wireless Communications*, Vol. 23, No. 5, pp. 120–128.
- [3] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu and F. Bonomi (2013). “Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture,” in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, Redwood City, pp. 320–323.
- [4] F. Jalali, K. Hinton, R. Ayre, T. Alpcan and R. S. Tucker (2016). “Fog Computing May Help to Save Energy in Cloud Computing,” in *IEEE Journal on Selected Areas in Communications*, Vol. 34, No. 5, pp. 1728–1739.
- [5] M. Aazam and E. N. Huh (2016). “Fog Computing: The Cloud-IoT/IoE Middleware Paradigm,” in *IEEE Potentials*, Vol. 35, No. 3, pp. 40–44.
- [6] Y. Xia, X. Luo, J. Li and Q. Zhu (2013). “A Petri-Net-Based Approach to Reliability Determination of Ontology-Based Service Compositions,” in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 43, No. 5, pp. 1240–1247.
- [7] Y. T. Kotb and E. Badreddin (2005). “Synchronization among activities in a workflow using extended workflow Petri nets,” in *Seventh IEEE International Conference on E-Commerce Technology (CEC’05)*, pp. 548–551.
- [8] I. Al Ridhawi, Y. Kotb, M. Aloqaily and B. Kantarci (2017). “A probabilistic process learning approach for service composition in cloud networks,” in *IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE 2017)*, Windsor, ON, Canada, pp. 1–6.
- [9] I. Al Ridhawi and Y. Kotb (2017). “A Secure Workflow-Net Model for Service-Specific Overlay Networks,” in *Mobile and Wireless Technologies 2017*, eds K. Kim and N. Joukov, ICMWT 2017, Lecture Notes in Electrical Engineering, Springer, Vol. 425, pp. 389–399.

- [10] K. Salah, J. M. Alcaraz Calero, S. Zeadally, S. Al-Mulla and M. Alzaabi (2013). Using Cloud Computing to Implement a Security Overlay Network. *IEEE Security & Privacy* 11, 44–53.
- [11] B. Mikolajczak and S. Joshi (2004). “Modeling of information systems security features with colored Petri nets,” in *2004 IEEE International Conference on Systems, Man and Cybernetics* (IEEE Cat. No.04CH37583), (IEEE: The Hague, Netherlands), Vol. 5, pp. 4879–4884.
- [12] I. El Kassmi and Z. Jarir, “Security Requirements in Web Service Composition: Formalization, Integration, and Verification,” in *IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2016)*, Paris, pp. 179–184.
- [13] E. Kassmi and Z. Jarir (2015). “Towards security and privacy in dynamic web service composition,” in *2015 Third World Conference on Complex Systems (WCCS)*, Marrakech, Morocco, pp. 1–6.
- [14] M. McGuire (2005). “Stationary distributions of random walk mobility models for wireless ad hoc networks,” in *Proceedings of the 6th ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc '05)*, Chicago, IL, USA.
- [15] V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramirez and S. Sanchez (2016). “Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios,” in *Proc. 2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, pp. 1–6.
- [16] Y. A. Ridhawi and A. Karmouch (2015). “Decentralized Plan-Free Semantic-Based Service Composition in Mobile Networks,” in *IEEE Transactions on Services Computing*, Vol. 8, No. 1, (IEEE: IEEE Computer Society), pp. 17–31.
- [17] M. Amini and F. Osanloo “Purpose-based Privacy Preserving Access Control for Secure Service Provision and Composition,” in *IEEE Transactions on Services Computing*, Vol. PP, No. 99, (IEEE: IEEE Computer Society), pp. 1–1.
- [18] N. Xi, C. Sun, J. Ma, Y. Shen and D. Lu (2016). “Distributed Secure Service Composition with Declassification in Mobile Network,” in *2016 International Conference on Networking and Network Applications (NaNA)*, (IEEE: Hakodate, Japan), pp. 254–259.
- [19] H. Andrade, T. Kurc, A. Sussman, E. Borovikov, and J. Saltz (2002). “On cache replacement policies for servicing mixed data intensive query workloads,” in *The Second Workshop on Caching, Coherence, and*

Consistency, with the 16th ACM Int'l. Conf. on Supercomputing, New York, NY, June 2002.

- [20] W. Bethel, B. Tierney, J. Lee, D. Gunter, and S. Lau (2000). "Using high-speed WANs and network data caches to enable remote and distributed visualization," in *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing (SC '00)*, Dallas, Texas, USA.
- [21] Y. T. Kotb, S. S. Beauchemin and J. L. Barron (2012). Workflow Nets for Multiagent Cooperation. *IEEE Trans. Automat. Sci. Eng.* 9, 198–203.
- [22] N. Tantitharanukul, J. Natwichai and P. Boonma (2013). "Workflow-Based Composite Job Scheduling for Decentralized Distributed Systems," in *16th International Conference on Network-Based Information Systems*, Gwangju, 583–588.
- [23] R. Buyya, and M. Murshed (2002). *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, John Wiley & Sons Ltd.

Biographies

Ismaeel Al Ridhawi received his BSc, MSc, and Ph.D degrees in Electrical and Computer Engineering from the University of Ottawa, Canada, in 2007, 2009, and 2014 respectively. He is currently an Assistant Professor of computer engineering at the College of Engineering and Technology, American University of the Middle East (AUM). His current research interests include quality of service monitoring, cloud network management, and overlay networks.

Yehia Kotb received his Ph.D from the Faculty of Computer Science, University of Western Ontario, Canada in 2011. He is currently an Assistant Professor of computer engineering at the College of Engineering and Technology, American University of the Middle East (AUM). Prior to joining AUM, Dr. Kotb was a senior software developer at Akira Systems in London, ON, Canada. His current research interests include probabilistic process learning and multi-agent cooperation in distributed systems.