IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 18, 2017, accepted October 13, 2017, date of publication October 24, 2017,
date of current version November 14, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2766068

Workflow-Net Based Service Composition Using Mobile Edge Nodes

ISMAEEL AL RIDHAWI[1], (Member, IEEE), YEHIA KOTB[1], AND YOUSIF AL RIDHAWI[2], (Member, IEEE)

[1]College of Engineering and Technology, American University of the Middle East, Kuwait City, Kuwait
[2]School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada

Corresponding author: Ismaeel Al Ridhawi (ismaeel.al-ridhawi@aum.edu.kw)

ABSTRACT Content delivery through cloud networks has gained popularity due to the cloud's ability to provide on-demand services. However, composite services, such as customized multimedia content, introduce both delays and resource limitations if traditional cloud solutions are used. With recent advances in mobile edge computing, customized media delivery can be achieved through compositions of service specific overlays (SSOs). This paper presents a workflow-net-based mechanism for mobile edge node cooperation in fog-cloud networks to form guaranteed SSOs. The proposed solution uses a mathematical cooperation operator to turn the SSO composition problem expressed as workflow nets into algebraic representations. In turn, the minimal cost cooperative path from the workflow net is determined such that it guarantees the delivery of the requested composite media services to clients. Experimental results show that the composition process can be adequately established and carried out in a timely manner.

INDEX TERMS Cloud, fog, mobile edge computing, ontology, overlay, Petri net, service composition, workflow net.

I. INTRODUCTION

Today's service-centric paradigm of networking and QoS-based content delivery has resulted in a plethora of new services. To a large extent, most software and hardware capabilities have become deliverable and consumable services. Over the past decade, with the advances in cloud service technologies, moving computing, control, and storage into the cloud has been a favored trend. But with today's increased number and variety of powerful edge and user client devices, a new concept has emerged in which researchers are referring to as Mobile Edge Computing (MEC) or Fog Computing [1], [2]. Edge and mobile devices have become smarter and richer in functionality, in which data collection, forwarding, enhancing and decision making has become part of their capabilities. MEC is a novel paradigm that extends cloud-computing capabilities to the edge of the network. MEC can support applications and services with reduced latency and improved service quality in dense geographical hotspots by providing close proximity to mobile consumers.

Traditionally, IoT devices communicate directly with the remote cloud for task submission, leading to high delays and network bandwidth overload. A considerable amount of traffic generated by the enormous number of IoT devices can be processed before being sent to the cloud. The fog computing paradigm was introduced to solve this issue, specifically for media content that requires a plethora of media enhancement mechanisms. The Fog-to-Cloud (F2C) [3] architecture was introduced to solve communication latency issues. The fog layer is an intermediate layer between the IoT devices and the cloud (Figure 1), which extends the traditional cloud computing paradigm towards the underlying networks. By bringing the network and cloud resources closer to the network edge, substantial amount of jobs will be processed near the IoT devices instead of sending data all the way to the cloud leading to reduced communication delays.

The Internet has surpassed the point of providing host connectivity only, and has become a global platform for sharing service components and content. Mobile edge computing allows for different mobile network devices to have the capability of providing diversified composed services at the edge of the network without reliance on cloud resources. The composition of distributed service components into more complex ones in a mobile environment is an especially laborious process. This is due to the possibility of facing disruptions caused by movement of service providers and clients. Additionally, the heterogeneity of devices and resource availability

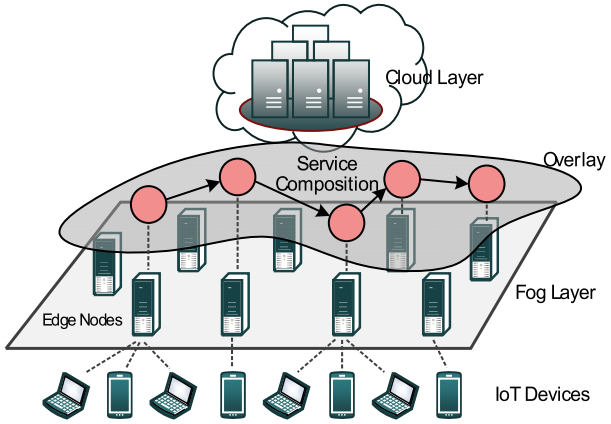


FIGURE 1. Service composition within a fog-cloud network.

is highly unpredictable in mobile networks. Furthermore, the presence of multiple service providers offering the same service results in more assorted composition processes.

With the increase in mobility and the number of available services, the problem of linking and composing isolated individual services into a distinguished complex service becomes progressively complex. Overlays (Figure 1) provide a mechanism of using lower-level infrastructure to provide higher-level services [4]. As such, it became possible to provide, through service specific overlays (SSOs), new services to end users. With today's cloud, fog, and MEC infrastructure, it has become possible to provide composable media and network-side services to help streaming applications achieve QoS guarantees. Nonetheless, service composition in an unstable mobile environment is a challenging problem.

In this paper, we address the problem of building and managing service specific overlays for media delivery in cloud networks using mobile edge devices. With the increasing number of mobile edge devices, there is a plethora of potential computational infrastructure available for providing end users with new functionality and improved services. The goal of our research is to allow end users to take advantage of dynamically available, local and remote computational infrastructure, without requiring service providers to have their services explicitly rewritten or reconfigured for each end-user service request and with minimal end-user intervention. Mobile edge devices make it possible for the dynamic creation of on-demand media services that can be composed from specific device capabilities through SSO buildup. SSOs are guided by end-user-centric abstractions capturing current user context and user intent. In contrast to data center clouds, SSO participants can include both virtualized and non-virtualized edge devices for faster service delivery. The presented work provides an abstract solution that although may seem to be adaptable to other mobile networks, the current MEC infrastructure is the most suitable for such SSO compositions.

Given the probable lack of having a single service node perform all the required media adaptations, service composition is a mainstream direction for solving such an issue. A solution towards service composition is to merge and manage the available capabilities of mobile nodes to reach the common goal (i.e. composed service). Petri-net composition is an active area of research [5], [6], where services may be composed to produce more complex ones. A cooperative mobile edge node service composition solution is introduced for cloud networks reinforced using workflow nets. Workflow nets [7], [8] constitute an extension to Petri nets such that source mobile edge nodes (SSO Media Servers) have no incoming transitions and sink mobile edge nodes (SSO Media Clients) have no outgoing transitions. Additionally, the notion of soundness associated with workflow nets implies that models are both structurally and behaviorally well-formed [9]. We integrate a cooperation operator which is used to compose mobile edge node capabilities expressed as workflow net units into a cooperative composed workflow net. We demonstrate that this solution is scalable to any number of mobile nodes given any number of capabilities and provides the minimal cost towards service composition.

Table 1 outlines the list of most used notations in this article. The remainder of the paper is organized as follows. In Section II we discuss related approaches in the literature. Section III provides an overview of the problem and the proposed solution. Section IV models the service composition problem. Section V describes the approach taken to solve the service composition problem. Section VI discusses the SSO generation algorithm. Section VII describes the composition plan learning process. Section VIII considers the soundness of the framework. Evaluation results are reported in Section IX. Finally, Section X concludes the paper.

II. RELATED WORK

SSOs were first introduced as an effective method to address some of the end-to-end QoS issues plaguing the Internet. Today, SSOs also facilitate the creation and deployment of value-added QoS-sensitive services. It allows simple services to be dynamically combined into new, more complex services. Initially composition was restricted to web services, but eventually extended to network-side functions such as media modification and synchronization. A plethora of service composition architectures have been introduced earlier and have evolved rapidly over the last decade [10], [11]. Initially, service composition architectures were designed for static environments, in which most solutions relied on a central coordinator for querying and composing services [12]. However, most of these designs presented a problem when incorporated in a dynamic mobile environment. Attempts to address these issues were solved using decentralized service composition architectures [13].

Fujii and Suda [12] presented a semantics-based service composition architecture that intuitively obtained the semantics of the requested service. The required components were then discovered and composed into a service based on the

TABLE 1. List of most used notations.

ABBREVIATION	DEFINITION
SSO	Service Specific Overlay
MS	Media Server
MC	Media Client
MP	Media Port
$\aleph$	Petri-net
WF_{net}	Workflow-net
P	Workflow-net place
T_i	Workflow-net transition
F	Arc connecting a place to a transition
W	Weight of the arc
P_{in}	Input place
P_{out}	Output place
δ	Service
λ	Service type
S	Set of service types
σ	Service capability
$w(MP_i)$	Set of service capabilities
d_j	Composition plan
$in_{i,j}^{Descriptio\ n}$	Media stream input description
$out_{i,j}^{Descriptio\ n}$	Media stream output description
$in_{i,j}^{QoS}$	Media stream input QoS
$out_{i,j}^{QoS}$	Media stream output QoS
Op_i	Function carried out by MP
C_i	MP service cost
R_i^M	Media service request
D_i^M	Media description
Q_i^M	Media QoS levels
II	Services executed in parallel
$\bullet T_i$	Set of input places to transition i
$T_i \bullet$	Set of output places for transition i
u_i	Workflow-net unit
C	Service composition plan
W^+	Input incident matrix
$\Gamma(T_i)$	Incident matrix column vector for input
$\Pi(T_i)$	Incident matrix column vector for output
g_i	Composition plan stage
A	Composition plan stage assignment matrix
P_r	Composition plan predicate
M	Composition plan execution cost matrix
$\otimes$	Cooperation operator
$\eth_i$	Workflow event
$\acute{\Lambda}$	Set of candidate events
Λ^0	Set of initial or previous events
Λ^c	Set of events that depend on already executed events
$\vec{\Lambda}$	Set of events that have already been executed
$\mathfrak{D}[\eth_i, \eth_j]$	Dependency matrix between two events
$\acute{\mathfrak{D}}$	Vector outlining the dependency of a single event from the dependency matrix
$\vec{\mathfrak{D}}$	Vector that determines the dependency of a certain event upon the set of events that have already been executed
$SS\Theta$	SSO composition workflow
Λ	Set of service requests
Σ	Set of basic service capabilities
Ω	Mapping function that maps service requests to basic service capabilities
N	Set of service nodes in the SSO
Ψ	Communication protocol used between service nodes
Υ	Set of workflows describing the service capabilities of service nodes
SIM	Similarity between two overlay nodes
$WN\text{-}SSO$	Workflow-net plan-based service composition solution
$PF\text{-}SSO$	Decentralized plan-free semantic-based service composition solution

service's own semantics and the semantics of necessary components. The architecture composed a service by creating a workflow of the requested service using discovered components and then executing the workflow. The composition solution requires a centralized implementation, which considerably limits the system's scalability. Additionally, not all service characteristics are considered when composing a service which is inadequate when providing composed media content.

The work in [14] presented a modeling approach to the web service selection problem for QoS-sensitive large processes. The service selection problem was formalized as a mixed integer linear programming problem that allowed constraints on the quality requirements to be specified. Services are selected one at a time by associating the running abstract activity to the best candidate service which supports its execution. Rudder, which was introduced in [13] provided a decentralized service composition framework that relied on software agents to provide high-level mechanisms for dealing with system adaptations and information discovery. The objective of Rudder was to enable runtime composition and coordination in P2P environments. The framework relied on a set of negotiation protocols to enable individual agents to progress towards a consensus. Adaptation plans were negotiated, decided, and enacted upon by multiple distributed cooperating agents.

Al Ridhawi and Karmouch [15] presented a decentralized semantic and syntactic nearness-based SSO node selection algorithm for mobile network environments. The process of constructing an SSO that meets the service requester's media content specifications and QoS requirements involves: 1) a semantic similarity measure between two connected overlay nodes, 2) a semantic nearness measure between a candidate service overlay node and the requester's service requirements against the output of the next-hop candidate service overlay node and the requester's requirements, 3) current QoS measures between the output stream of a chosen service overlay node and the input stream of the immediate next hop candidate service overlay node, and 4) expected QoS measures between the output stream of the next hop candidate service overlay node and the service requester's QoS requirements. Both semantic similarity and nearness are evaluated using ontologies. A fuzzy QoS evaluation technique is used to determine whether quality levels reflected in the provided services can meet the requester's requirements. SSO composition paths are created in a distributed decentralized fashion in which each overlay node determines whether or not it can meet the requester's requirements.

Sirin *et al.* [16] demonstrated a goal-oriented, interactive composition approach with a matchmaking algorithm to filter and select services. The system utilizes OWL-S [17] service descriptions (Service Profiles). When a service needs to be located, the system creates a service profile for the desired final service. A service registry matches requested profiles to those advertised through subsumption. A composer is responsible for choosing the best services for composition. The work suffers from an oversimplified matchmaking process where it limits the matching classification to only four classes as well as omitting the importance of QoS in service selection.

The authors in [18] introduced a solution to model the many-to-many mobile cloud service composition problem, where multiple mobile cloud edge nodes pool their

resources to compose a service requested by a mobile client. A theoretical model is presented to describe the service composition topology reconfiguration process based on a series of decisions. Three algorithms are proposed to solve the service composition process, each suitable for a different scenario. The work omits experimental evaluations to showcase the advantages of applying such a composition technique in cloud environments.

The authors in [19] introduced an orchestration mechanism to enable the re-use and selection of service components across different composite services in distributed edge clouds. Clients requesting a particular composite service would select one of the available service components. Since it is unlikely that all components of a composite service are available in a single cloud edge, session slots are used as a service availability metric to allow sharing of service components among clients.

Oueis *et al.* [20] presented a cooperative Small Cell Cloud (SCC) solution to offload mobile user computations to pools of resources that are closer to the user. Small cells cooperate for computation purposes by forming computation clusters. Clusters are composed depending on the requested task to balance the load among the cells. The presented work focuses on computation cluster provisioning and resource allocation in multi-user cases, where such provisioning and allocation is jointly and simultaneously optimized for all users to achieve optimal performance. Simulation results show that the proposed algorithm yields high user satisfaction for up to four users per small cell and reduced power consumption.

The authors in [21] outline that current cloud service composition and selection methods assume that networking resources are over-provisioned and their usage is not considered when making composition decisions. This will lead to wasteful network resource consumption and impractical end-to-end QoS optimality for cloud-based services. The authors proposed a network-aware cloud service composition approach that optimizes service composition decisions by taking into account, among other decision-making factors, network resource consumption in fat-tree cloud datacenter networks and realistic (end-to-end) QoS optimality. The solution has a linear computation time with respect to problem size and was tested using the WebCloudSim cloud simulator. Although the presented approach performs well when compared to several other service composition approaches, simulation results show that it is not suitable for scenarios with multiple data centers.

Padmavathi *et al.* [22] proposed an agent-based approach for cloud service negotiation. The solution enables the interaction among software agents to optimize the procedure of selecting the best cloud service providers that satisfy the request of consumers. Additionally, if a single request can only be satisfied by two or more providers, then a broker combines the set of services from multiple providers and delivers the combined service as one virtual service. The authors introduced Contract Net Protocol (CNP) which is a task sharing interaction protocol used for cloud service selection and composition within the system.

In [23], a cloud service composition algorithm called DE4CDSC is developed which considers mobility, QoS, and temporal constraints in mobile cloud environments. It first utilizes a constraint-based service filtering process to reduce the search space and then adopts a differential evolutionary based algorithm to form a composed service. In [24], the authors use bi-graphs to model the cloud service composition problem as opposed to mathematical definitions to provide more accurate cloud composite services.

Wu *et al.* [25] introduced a cloud service selection method, named CSSM, to mine for qualified versions of cloud services for trusted cross-cloud service composition. The method takes the utility value as the evaluation index and aims at finding optimal or near-optimal trusted service composition solutions from a set of cloud services based on users' demands. User preference on each QoS metric is formalized as the preference interval for enhancing the fitness of the service composition solution. Additionally, an extended top-k iteration composition process is performed among cloud services to get an optimal or near-optimal trusted service composition solution. Compared to other composition methods, CSSM reduces time complexity and maintains comparable optimality.

Research in the area of Petri net service composition is ongoing [5], [6], where services may be composed on the basis of complementarity. For instance, send and receive services are complementary by nature and may be composed [26]. Du *et al.* [6] presented a mediation-aided web service composition approach when dealing with incompatibilities of services. The solution first model's services as open workflow nets and then a modular reachability graph of composition is constructed and analyzed. Li *et al.* [27] introduced a formal definition of context-independent similarity and showed that web services can be substituted by an alternative service of similar behavior without intervening other web services in the composition. This will reduce the cost of verifying service suitability. Klai and Ochi [28] proposed a solution to compose cloud services through the use of Symbolic Observation Graphs (SOG) [29]. SOGs are used to abstract cloud services and check the correction of the composition with respect to behavioral properties using event- and state-based LTL formulae. A registry containing SOGs (i.e. abstraction of cloud services or public views) are created through service translation with the aid of petri-nets. A new formalism of petri-nets called resource constrained open workflow nets (RCoWF-net) is introduced to allow both asynchronous communication and sharing of cloud resources between different services. The solution is limited to only checking whether or not the resource provider is able to satisfy a user's request in terms of cloud resources.

With the increase in the number of sensors, actuators, and mobile nodes, IoT has emerged as a promising supporting

solution for seamless and ubiquitous computing. The concept of IoT has become more widely implemented with the integration of cloud systems. IoT combines different aspects and technologies from different everyday objects and turns them into a smart surrounding in which not only information is collected, processed, and controlled, but also with the aid of the cloud, this information is interconnected and exchanged to provide enhanced composite services. The delay caused by transferring data to the cloud and back is intolerable. To address this issue, fog computing was introduced to allow for the integration of edge devices and cloud resources to help overcome latency issues. Fog computing is a distributed paradigm that provides cloud-like services to the network edge [30]. Studies on fog computing are still premature leading to many emergent architectures for computing, storage, control, and networking that distribute services closer to end-users.

Despite the abundant existing studies on service composition techniques, to the best of the authors' knowledge, the presented work is the first to employ a petri net based cooperative service composition solution using mobile edge nodes in fog and cloud systems to achieve fast and stable composed service overlays to deliver media content to clients.

III. PROBLEM AND SOLUTION OVERVIEW

Composition of SSOs in fog-cloud networks is constrained by the identification of which mobile edge nodes (service nodes) should interoperate with each other in order to fulfill the subscriber's request. It is important to incorporate a solution that can model inter-service relationships to successfully form SSOs. Linking consecutive nodes in a composition involves addressing the notions of choice dependency and unit similarity, such that if two or more nodes among a set are deemed similar and fulfill part of or all of the requirements, then they can be interlinked to accomplish the required task or part thereof. Finding a service that performs a required function does not guarantee it can seamlessly link with the previous or next node in a service path without having their respective capabilities and similarities challenged.

Such SSO composition problems can be solved for using cooperating mobile service nodes. Cooperation among a group of mobile edge nodes is defined as the process of adapting and managing the available services and resources of mobile nodes to reach the goal of composing a particular service. Cooperating mobile nodes thus negotiate for services and perform task planning in order to accomplish the goal of establishing a composed service.

Figure 2 illustrates a set of composed SSOs in a fog-cloud environment. The process of service composition starts with a mobile node registering with a cloudlet (or a registered edge node). The mobile node will communicate its capabilities such as media services, hardware and software resources, etc. Nodes that are capable of adapting enhanced media services

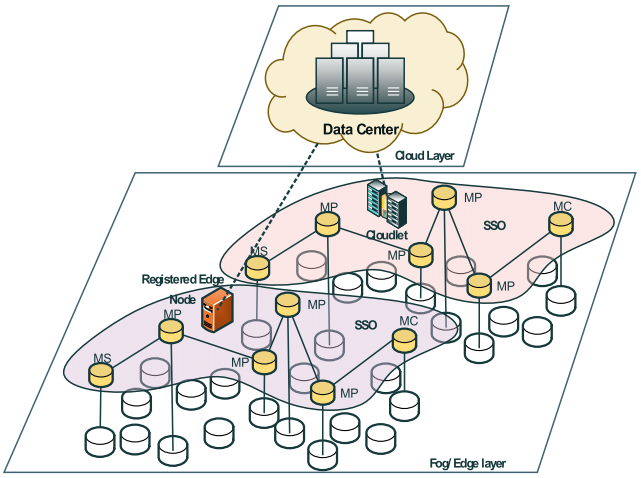


FIGURE 2. Composition of SSOs within a fog-cloud network.

such as encoding conversion, addition of subtitles, buffering, caching, routing and synchronization are registered as Media Ports (MPs). A flexible and transparent construction of end-to-end media delivery paths can be made from Media Servers (MS) to Media Clients (MC). For example, consider users (MCs) trying to view a particular video stream originating from a video server (MS) in MP4 encoding on their mobile devices. Each client requires the media content to be adapted to his/her unique device requirements. Given the possible unavailability of needed services at the MC side, one or more MPs are needed for the conversion process. Therefore, an SSO is constructed independently for each media delivery session.

Thus, given a set of MPs, we must define a formal cooperative behavior description among the nodes to provide the composed service. In our approach, we use workflow nets which constitute an extension to Petri nets to guarantee the correctness of the cooperation and reachability problem [9]. Additionally, we use workflow nets to generate workflow plans, compare them, and find the one that produces the minimal cost in terms of latency and path stability. Moreover, workflows are used to allow cloud systems to find the most optimal SSO composition solution for future user requests using a process learning approach [31].

A petri net $\aleph$ (1) is a directed graph for which the nodes are either transitions or places. A place is connected to one or more transitions. On the other hand, a transition is connected to one or more places. Nodes sharing identical types cannot be directly connected. A place can be either empty or contain activities. A transition is said to be enabled if and only if there are no empty places connected to it as input.

$$\aleph = < P, T, F, W > \tag{1}$$

P is the set of places, T is the set of transitions, F is the set of arcs among transitions and places, and W is a vector containing the weights of the arcs in F. Workflow

nets (WF_{net}) are an extension to petri nets with the following forced conditions:

- $\aleph$ has a single input place P_{in}, where $P_{in} = \emptyset$.
- $\aleph$ has a single output place P_{out}, where $P_{out}\bullet = \emptyset$.
- If a transition t^* is added to $\aleph$ such that $\bullet t^* = \{P_{out}\}$ and $t^*\bullet = \{P_{in}\}$, the petri net $\aleph^*$ becomes strongly connected. t^* is a transition which connects the input to the output of WF_{net}.

The WF_{net} consists of transitions representing the actions performed by MPs. In our model, we assume that there is a set $S = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ of all primitive service types that cannot be fragmented into simpler service types, and sets of non-primitive service types $\varepsilon \subset S$. Any service δ provided by a MP at a given time is constructed from a list of non-primitive service types. If a MP_i from the set of cooperating nodes $\{MP_1, MP_2, \ldots, MP_n\}$ has composition plan d_j from the set of plans $\{d_1, d_2, \ldots, d_k\}$, then the MP can perform its plan on its own if and only if it meets the time constraints (if any), and the following holds:

$$\forall \delta \in d_j : \delta \in w\,(MP_i) \tag{2}$$

where $w\,(MP_i) = \{\sigma_1, \sigma_2, \ldots, \sigma_s\}$ is the set of basic service capabilities σ provided by MP_i, $d_j = \{\delta_o \cup \delta_k^*\}$, where $\delta_o = \emptyset$ is a starting service and δ_k^* is a set of services that follow.

Two nodes MP_i and MP_k can cooperate to perform a desired composition plan d_j if they satisfy the MC's composed service requirements as follows:

$$\forall \delta \in d_j : \delta \in w\,(MP_i) \cup w\,(MP_k) \tag{3}$$

Node MP_k is a candidate for cooperation with node MP_i if and only if:

$$\forall \delta \in \Delta_j : \Delta_j = d_j - w\,(MP_i) \;\; such\ that\ \delta \in w\,(MP_k) \tag{4}$$

where Δ_j is the difference between the capabilities required to achieve composition plan d_j and the capabilities of node MP_i.

IV. SSO COMPOSITION MODEL

A. SERVICE DESCRIPTION

The process of node selection for SSO composition begins with a service request from the MC to the cloudlet (or a registered edge node). To provide such a service, a number of sub-services might be composed to form the MC's requested service. The composition process might involve a number of MPs in which each offers a particular type of service to users such as caching, synchronization, video encoding/decoding and content insertion. A service provided by a MP is described as follows:

$$\delta_i = \left\{ \begin{array}{c} Op_i, \\ In_i\left(in_{i,j}^{Description}, in_{i,j}^{QoS}\right), \\ Out_i\left(out_{i,j}^{Description}, out_{i,j}^{QoS}\right), \\ C_i, \end{array} \right\} \tag{5}$$

where Op_i specifies the functions carried out by the MP such as encoding; In_i defines the acceptable input requirements by the MP needed to perform its service which is described by: service input description $in_{i,j}^{Description}$ and service input QoS $in_{i,j}^{QoS}$. The former for example specifies *{encoding, size, stream length, etc.}* of a video stream. The latter specifies the level of acceptable QoS, for example *{bit rate, loss rate, accuracy, etc.}*. Out_i defines the characteristics of the output service generated from the MP. It has the same definition as In_i. C_i represents the cost of performing the MP's service.

B. SERVICE REQUEST

A media service requested by a MC is represented as a composite of a media description D_i^M parameter and the required QoS properties' levels Q_i^M.

$$R_i^M = \left(D_i^M, Q_i^M\right) \tag{6}$$

The description parameter D_i^M encompasses general media description properties such as the media stream identity D_i^{id}, type D_i^{type}, encoding format $D_i^{encoding}$, etc. While the QoS parameter identifies the MC's accepted levels of quality. Each QoS property Q_i^M, where $M = \{delay, jitter, cost, etc.\}$ is further expanded to define the acceptable range of values if not exact. These values range from a minimum acceptable value q_{min}^M to a maximum acceptable value q_{max}^M. Additionally, a priority level $P^M \in (1, 10)$ is associated with each QoS property. The priority level illustrates the significance of the property to the MC.

C. MODELING SERVICE COMPOSITION PLANS

Media requests arriving from the cloudlet (or registered edge node) to the MS may require the formation of service composition paths radiating from the MS towards the MC through a number of MPs. The service composition problem can be viewed as a planning problem. The use of plans provides a solution to automatically compose services through mapping requirements into tasks' descriptions that are potentially mapped into concrete services provided by MPs.

A plan represents a full map of the services required in every hop towards the MC along with the required levels of QoS. Such services are initially described using (5). An example of a composition plan is depicted in Figure 3. The figure illustrates a multistage composition path involving sequential and parallel services. To facilitate chaining in service composition, MPs can be described according to their input and output ports. The concept was introduced in [32] and [33]. Single input MPs take a media flow from one input port and transform it into a different output flow according to the service functions they offer. Splitters have a single input and two or more output ports. A splitter might for example take an MP4 video stream as input, and produce an AVI video and MP3 audio streams as output. Joiners are the reverse of splitters, taking in two or more input streams and producing a single output stream.

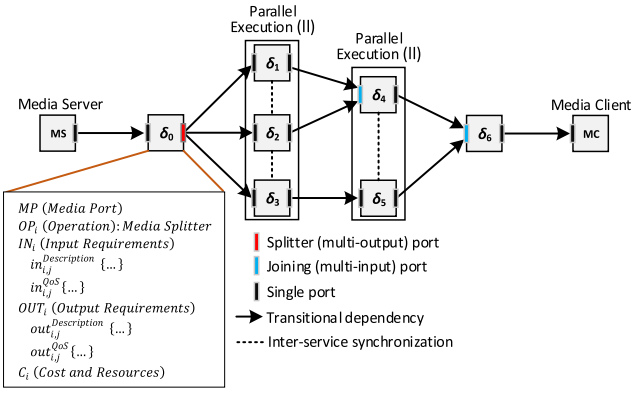


FIGURE 3. A sample SSO composition plan generated following the Media Client's requested specifications.

As such, the general model of a composition plan as described in Figure 3 is represented as follows:

$$Plan_{MS}^{MC} = \{MS, \delta_0, \text{II}(\delta_1, \delta_2, \delta_3), \text{II}(\delta_4, \delta_5), \delta_6, MC\} \quad (7)$$

where II represents services that must be executed in parallel, and $\delta_i, 0 < i \leq n$ is a detailed description of the service that must be provided by the MP. For example, δ_4 is a joiner service that merges the audio stream of service δ_1 with the video stream of δ_2. The parallel execution of services can have an effect on sequentially executed services even if it is not connected to its predecessor's parallel neighbors. For example, δ_5 cannot execute without δ_3. However, δ_3's parallel execution with δ_1 and δ_2 must be synchronized as does δ_5's execution with δ_4. This is due to the fact that δ_6's final execution is affected by all previous services. Although complete and exact parallelism cannot be guaranteed, however we make the assumption that services executed in parallel synchronize their operations as visually illustrated in Figure 3. This synchronization assures that services execute without having any conflicts with parallel or sequential services.

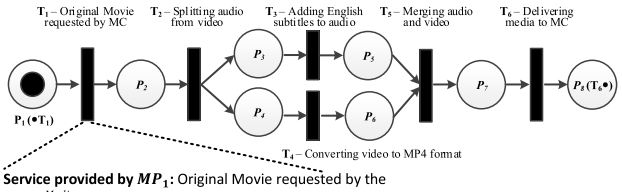
The formed plan represents an abstract plan that is later translated into a workflow-net model constructed of workflow-net units (8). A unit u_i, where $1 \leq i \leq k$, k is the number of units composing the workflow net, is a transition comprised of sets of input and output places that model an action, the conditions prior to execution, and the results of performing the action.

$$u_i = (\bullet T_i, T_i, T_i \bullet) \quad (8)$$

T_i is a transition, $\bullet T_i$ is the set of input places to T_i, and $T_i \bullet$ is the set of output places for T_i.

As discussed in the previous section, two nodes MP_i and MP_k can cooperate to perform a desired composition plan d_j if they satisfy the MC's composed service requirements as in (3) and (4). Thus, given a group of MPs, node service capabilities and requirements must be considered for cooperation to be successful and for a desired composition plan to be performed. Such cooperation is performed through workflow-net unit comparisons.

The service overlay composition problem is modeled as a set of workflow plans, where a required task can be achieved using a single workflow from the set of workflow plans. Figure 4 illustrates a WF_{net} model for a composed media service plan originally outlined in Figure 3 and equation (7). A descriptive example of the original media content (service parameter description) provided by MP_1 is outlined in the figure showing the quality of the service requested by the media client. Each transition will incorporate a similar media description outlining the service parameters after applying the media services such as splitting audio from video or adding subtitles. Transition description includes service parameters such as the media type, id, encoding, delay constraints, jitter constraints, and cost.



Service provided by MP_1: Original Movie requested by the MC (R_{MC}^{Media})

[D_{MC}^{Media} [%description of the media.
D_{type}^{Media} = "video stream";
D_{id}^{Media} = "WALL-E";
$D_{encoding}^{Media}$ = "MP4";
] Q_{MC}^{Media} [%quality of received media stream.
Q_{delay}^{Media} [
$q_{required}^{dely}$ = 80 msec;
q_{min}^{dely} = 50 msec;
q_{max}^{dely} = 160 msec;
p^{delay} = 4;
]Q_{jitter}^{Media} [%jitter or received media stream
$q_{required}^{jitter}$ = 0.8%;
q_{min}^{jitter} = 0%;
q_{max}^{jitter} = 1.3%;
p^{jitter} = 9;
] ...
Q_{cost}^{Media} [%cost unimportant to MC, no
%limits set and has lowest priority
q_{min}^{dely} = $-;
q_{max}^{dely} =$ -;
p^{delay} = 1;]
]
]

FIGURE 4. A WF_{net} model representing the SSO composition plan described in equation (7). The model represents a media service which consists of six transitions, each representing a predefined action.

The following section elaborates further on the SSO composition process. We introduce an algorithm which selects the minimal cost cooperative path from the workflow net plan set (see Section VI for details). Additionally, a composition plan learning approach is developed to compose SSOs for future user requests (see Section VII for details).

V. SSO COMPOSITION PROCESS

A. COMPOSITION PLANNING PROCESS

The composition planning process can be viewed in the summarized architecture shown in Figure 5. With the arrival of the MC's request (6) at the cloudlet (or registered edge node), the request is broken down by the *Request Translator* into three service components: input media description, output media

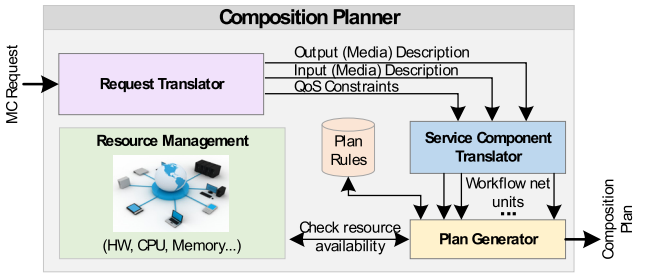


FIGURE 5. Service composition planning process performed at the cloudlet or a registered edge node.

description, and output QoS (5). With the aid of the *Service Component Translator*, the translated service request is then rendered into workflow net units (8). A *Plan Generator* will then query the network's available resources to determine its capabilities of enforcing the plan.

Every MS and MP provide one or more services. Each service is provided at a certain QoS level and require a percentage of the node's resources. Prior to sending its acceptance to join an SSO composition following the retrieved plan, the current level of available resources in a service node must be compared to those required. If acceptable levels of resources are available; these resources must be reserved to prevent any conflict with other SSO composition requests. The plan generator relies on a set of planning rules that guide the cloudlet (or registered edge node) into generating new SSO composition plans. Details pertaining to the plan generation algorithm are discussed in Section VI. Once a new plan is generated, it is passed on from the cloudlet (or registered edge node) to the MS and then to subsequent MPs to form a composition path.

B. ONTOLOGY-BASED MODELING

The reliance on the syntactic and semantic representation of transitions (i.e. services performed by MPs) and the pre- and post-conditions of such transitions (i.e. inputs and outputs of such service actions) raises the need for a use of a unified modeling method for the proposed system. The presence of multiple entities in the system along with the requirement for a common understanding of how concepts are represented and how they interact with each other, are all factors that encourage the use of a simple, sharable, and extendable approach for representing context in a mobile environment. As such, we have based our system on the assumption of the existence of an ontology [34] that unifies the view and understanding of all concepts in a fog-cloud environment.

We have assumed the presence of an OWL-based ontology [35] covering an extended area of knowledge as shown in Figure 6. This includes representations for services and all components associated with such services. Contextual information within our proposed system is restricted to bandwidth, packet loss, available memory, and other information typically used within media service-oriented systems. Through these ontologies, service request translation and syntactic

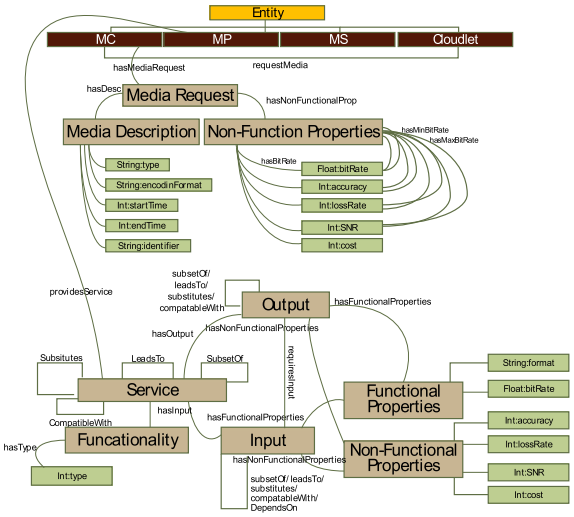


FIGURE 6. Ontology used to represent the service request and description.

representations of workflow net units can be derived. Full details pertaining to the ontology syntax, semantics, translation and inter-communication process has been introduced earlier in [34] and is out of the scope of this article.

C. COMPOSITION PLAN GENERATION REQUIREMENTS

Once service description has been rendered into workflow net units, workflow net plans are generated through the cooperation of service nodes. A service composition plan C is a set of joined units in a topology:

$$C = \{U, P, F\} \tag{9}$$

where U is a set of units, P is a set of places, and $F \subseteq U \times P \cup P \times U$. A composition $C_1 \subset C_2$ if and only if $\forall u_i \in C_1 \exists u_j \in C_2 | u_i \equiv u_j$.

The notions of choice dependency and unit similarity are crucial due to their effect on the expected level of cooperation. For example, if two units among a set of workflow nets are deemed similar, then the units can be interchanged in order to accomplish the same task or part thereof. It is thus necessary to identify similar units to minimize cooperation costs. Choice dependency occurs when two or more units share one or more input places. For instance, if unit u_1 and u_2 are choice-dependent, but unit u_3 is choice-independent, then unit u_1 cannot replace unit u_3 in its actions and vice versa.

If a unit is choice-dependent, then the set of choice-dependency is defined as:

$$\left\{T_j | T_i \in \left\{T - T_j\right\}\right\} \text{ and } \bullet T_i \cap \bullet T_j \neq \emptyset \tag{10}$$

and can be determined by satisfying the following condition:

$$W^+\left(P_j, T_i\right) - W^+\left(P_j, T_k\right) = 0$$
$$\forall_{j=1}^{m} P_j \in \bullet T_i \cap \bullet T_k \quad \text{and } \forall_{k=1}^{m} T_k \in T \tag{11}$$

where m is the number of places $P_j \in \bullet T_i$, k is the number of transitions $T_k \in T$, and W^+ is the input incident matrix of the workflow net.

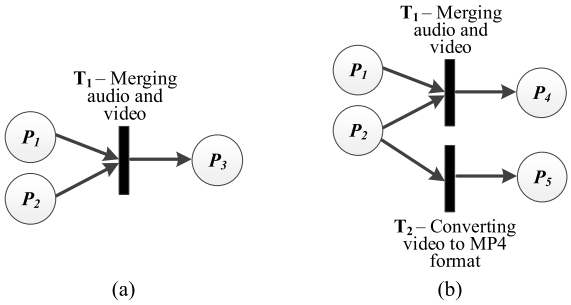


FIGURE 7. Units u_1 and u_2 represented by the figures (a) and (b) are choice-independent due to the existence of another transition input from place P_2.

Figure 7 outlines an example of two units u_1 and u_2 which are choice-independent. Although the two units are similar, such that events represented by places P_1 and P_2 will lead to events P_3 or P_4 by performing the actions represented by transitions T_1 and T_2 respectively, the units are deemed not similar since the event represented by P_2 might lead to P_5 when the action represented by transition T_2 is performed.

The input incident matrix which models the workflows represented in Figure 7 is as follows:

$$\begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{array} \begin{array}{c} \begin{array}{ccc} T_1 & T_2 & T_3 \end{array} \\ \begin{bmatrix} -1 & -1 & 0 \\ -1 & -1 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

Moreover, two units are identical if and only if they satisfy similarities in transition, input and output places. Two transitions are said to be similar if and only if $T_1 \in \lambda$ implies $T_2 \in \lambda$, where λ is an action belonging to the set of primitive service types S. Input and output places' similarities are determined respectively as follows:

$$\Gamma(T_1) - \Gamma(T_2) = 0 \tag{12}$$

$$\Pi(T_1) - \Pi(T_2) = 0 \tag{13}$$

where $\Gamma(T_i)$ and $\Pi(T_i)$ are column vectors representing the input and output places to and from transition T_i.

Accordingly, unit u_1 is said to be similar to unit u_2 (denoted $u_1 \equiv u_2$) if and only if $\exists\ T_1 \in u_1$ and $\exists\ T_2 \in u_2 | S(T_1) = S(T_2)$ and $\bullet T_1 = \bullet T_2$. Moreover, unit u_1 is said to be identical to unit u_2 (denoted $u_1 = u_2$) if and only if $\exists\ T_1 \in u_1$ and $\exists\ T_2 \in u_2 | S(T_1) = S(T_2)$ and $\bullet T_1 = \bullet T_2$ and $T_1 \bullet = T_2 \bullet$.

Figure 8 outlines an example of two similar units, namely, u_1 and u_2. Although the two units have different output places (i.e. P_3 and P_4), the two units are said to be similar since both have similar input incident vectors where the events represented by places P_1 and P_2 will lead to the same events represented by P_3 or P_4 by performing the actions represented by transitions T_1 and T_2 respectively.

Hence, the process of service node (unit) cooperation becomes feasible knowing the similarity and equality between the groups of available inter-dependent units within the cloud environment, thus allowing services to be composed and delivered to media clients. The following section describes the algorithm used to generate SSOs (workflow net plans).

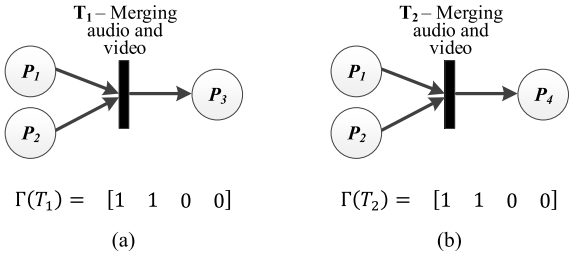


FIGURE 8. Units u_1 and u_1 represented by the figures (a) and (b) are said to be similar since both have similar input incident vectors.

VI. SSO GENERATION ALGORITHM

The SSO generation algorithm generates a cooperative, composed workflow which contains all the possible cooperative scenarios among service nodes (MSs and MPs). The algorithm implements a back-tracking scheme allowing it to determine the minimal cost cooperative path from the workflow net in terms of latency and composition path stability. We begin by defining the concepts and data structures used in the algorithm:

- Stages constitute the representation we use to express the order of computation of a plan based on standard operator precedence.
- Plan stages $G = \{g_1, g_2, \ldots, g_n\}$, where n is the number of stages. Stage i, denoted as g_i, is a set of predicates which belongs to plan ρ. A predicate is an action (service capability) performed within a composition plan to achieve the requested task. For instance, encoding of a movie to a certain format is a predicate in logic form.
- A stage assignment A is a two-dimensional array with its number of rows equal to the number of stages n. Each row has a number of tuples (columns) equal to the assignment of this stage. Example: $A(0)$ is the assignment of the first stage and is of the form $(MP_1; P_r(2)), (MP_2; P_r(1)), \ldots, (MP_n; P_r(k))$, where n is the number of node assignments in this stage and k is the last assigned predicate. $A(0, 1)$ is the second assignment of the first stage which is $(MP_2; P_r(1))$ in this example. This tuple signifies that node MP_2 is dedicated to execute predicate $P_r(1)$.
- A two-dimensional matrix M with the number of rows equal to the number of service nodes involved in the current stage, and number of columns equal to the number of predicates in the plan. Essentially, $M(i; j)$ is the cost to execute predicate $P_r(j)$ with node MP_i.
  - ➢ $M(i)$ (the i^{th} row of M) is a vector representing the costs of the capabilities of node MP_i to execute the plan predicates.
  - ➢ $M^T(i)$ (the i^{th} column of M) is a vector representing the costs of predicates when executed by different nodes.

➢ The following is an example of M with three nodes and four service capabilities, in which the minimal cost to perform a service task involving predicates 1, 2, and 4 is achieved through the cooperation of nodes 1, 2 and 4. P_r (1) is performed by MP_2, P_r (2) is performed by MP_3, and P_r (4) is performed by MP_1.

$$\begin{array}{c} \\ MP_1 \\ MP_2 \\ MP_3 \end{array} \begin{array}{c} \begin{array}{cccc} P_r(1) & P_r(2) & P_r(3) & P_r(4) \end{array} \\ \begin{bmatrix} 5 & 2 & 8 & 1 \\ 3 & 6 & 7 & 1 \\ 4 & 1 & 1 & 3 \end{bmatrix} \end{array}$$

- Van der Aalst *et al.* [36] shows that to describe the dynamic behavior of a workflow, few logic operators are needed. Among those operators are AND, OR and implication. Van der Aalst also shows that this is the complete list needed to define the workflow behavior. In our work, we use those operators to describe a plan of execution which in turn gets translated into a workflow. The *and* operator $\bigwedge$ joins the incident matrices of its predicates yielding a new incident matrix describing the workflow net resulting from applying the operator. The *or* operator $\bigvee$ joins the incident matrices of two predicates to form a new incident matrix describing the resulting workflow net to allow one part or another of the cooperative plan to be executed. The *then* operator $\rightarrow$ creates the sequential sections of cooperative plans between predicates. It joins the incident matrices of its predicates creating a new incident matrix describing the workflow net resulting from applying the operator.
- The *cooperation* operator $\otimes$ joins (composes) two workflow net units into a workflow net. The operator is also used to join (merge) two cooperative workflow nets (SSOs) into a single workflow net (SSO). The process of merging two workflow nets into one is out of the scope of this paper and has been discussed earlier in [37].
- We denote an operator $P_r\ (i)$ affected by a *then* operator using the symbol $P_r\ (i)^{\rightarrow}$ and $P_r\ (i)^{\nrightarrow}$ if it is not. A predicate that is affected by the $\rightarrow$ operator has its execution delayed by predicates prior to the operator in the plan. If $P_r\ (i) \rightarrow P_r\ (j)$ then $P_r\ (i)$ is denoted as $P_r\ (i)^{\leftarrow}$.
- A predicate $P_r(i)$ is affected by the *then* operator if and only if there is a rule in the plan that is in the form of $\left(P_r\ (k)\left\{\bigwedge \mid \bigvee \rightarrow \right\} P_r(j)\right)^* \rightarrow P_r\ (i)$.
- $|S_t(i)|$ is the cardinality of $S_t(i)$.
- NodeIndex is a temporary node index value in A.
- (NodeIndex,$*$) $\in A(i)$ represents all tuples in stage $i \in A$ that has a node with an index of NodeIndex.

Algorithm 1 outlines the steps involved in generating a composed SSO which are summarized as follows:

- For all nodes that belong to (served by) a fog, determine the similarities in node capabilities to identify parallel threads.
- Test the composition plan according to task coverage (i.e. test the collective capabilities of *MPs* and determine if it is sufficient to execute the plan). The complexity of this step is an order of $O(MP)$.
- For all predicates P_r that are a result of implication, calculate the collective incident matrix of the predicate. The complexity of this step is an order of $O(P_r)$.
- After calculating the dependencies and implications, build matrix M for stage g_i.
- For every stage g_i, select the minimum matrix M that would satisfy the plan. The complexity of this step is an order of $O(g_i)$.
- Test implications that satisfy all predicates.
- If a predicate is missing, then try to find a node that supports the missing predicate and add it to the assignment.
- The predicates are then converted into a workflow.
- The workflow is developed by merging and fusing the new construct with the framework. The merging process is out of the scope of this paper. Readers may refer to [37] for futher details.
- The overall complexity of the system is an order of $O\,(MP) \times [O\,(P_r) + O(g_i)]$.

VII. COMPOSITION PLAN LEARNING PROCESS

The composition process for future service requests is based on a learning approach, such that node log files from previously successful composition processes are used to recreate similar compositions for similar events in the future. Service node log files include plan threads' descriptions such as the events that led to an action to be considered, the events which occurred after applying the action, list of nodes that preceded the considered node in the composition, and the node that followed the considered node in the composition. A plan thread is a description of the process flow from the MS to the MC. Such that, a thread provides a description of the events that are produced in a plan following the actions performed by service nodes. More specific towards media service composition scenarios, threads constitute the different node composition paths taken to provide the composite service.

Three steps are involved in the proposed service composition plan learning method: *selecting candidate events*, *calculating the probability of event occurrence*, and *merging threads into a composition plan*. The first step derives a list of candidate events that may occur when applying a particular action. The second step calculates the probability that the selected candidate events will occur. The last step involves merging multiple threads into a plan. A service composition plan which may be made up of different or somewhat similar plan threads (i.e. a series of events that occur following the actions performed by multiple service nodes) is formed by merging those threads together. These steps are repeated for all plan threads of newly composed service log files until all events are handled.

A. SELECTING CANDIDATE EVENTS

The set of candidate events $\acute{\Lambda}$ are selected according to (14), such that it contains all events that could start a particular

Algorithm 1 SSO Composition

INPUT: A plan $P = \left\{P_i\left(\left(\bigwedge \mid \bigvee \mid \rightarrow\right)P_j\right)^*\right\}$ where P_i and P_j are predicates, and a group of mobile edge nodes $MP = \{MP_1, MP_2, \ldots, MP_n\}$ each with a set of capabilities.

OUTPUT: A cooperative service specific overlay $SS\Theta =< S, MP, \Omega(MP), D, SIM, \xi >$, where S is the set of primitive action types, MP is the set of cooperating service nodes, $\Omega(MP) = \{w(MP_1), w(MP_2), \ldots, w(MP_n)\}$ is the set of all service node capabilities, D is the set of plans to be performed by the set of service nodes, $SIM = \left(SIM_1, SIM_2, \ldots, SIM_{n(n-1)}\right)$, $\xi = \{\xi_1, \xi_2, \ldots, \xi_z\}$ is the set of workflows that bind two or more different workflows from two or more service nodes.

for all $MP_i, MP_j \in MP$ **do**
  Calculate $SIM\left(MP_i, MP_j\right)$
end for
Test SSO composition plan according to (3), end if not satisfied.
$i = 1$
for all $P_r(i)$ and $P_r(i)^{\nrightarrow}$ **do**
  $g_i = g_i \cup P_r(i)$
end for
Label: 1
Build matrix M for stage g_i
for $k = 1$ to $|g_i|$ **do**
  Label 2:
  NodeIndex = index of $\{\min M^T(k)\}$ where $\min\{M^T(k)\} \neq \infty$
  if (NodeIndex, $*$) $\in A(i)$ **then**
    M^T(NodeIndex, index of $\{\min M^T(k)\}) = \infty$
    Go to Label 2
  else if NodeIndex $= \emptyset$ **then**
    Backtrack to $k-1$
    Choose the next least cost
  else
    $A(i) = A(i) \cup$ (NodeIndex, index of $\min M^T(k)\}$)
  end if
end for
$i++$
if $\exists P_r(i), P_r(k) \mid P_r(i)^{\rightarrow}$ and $P_r(k)^{\rightarrow}$ and $P_r(k) \in g_{i-1}$ **then**
  $g_i = g_i \cup P_r(i)$
  Go to Label 1
end if
if $\exists MP_u \in MP \mid (u, *) \notin A$ and $\exists MP_k \mid (k, w(MP_k)) \in A$ and $SIM(MP_u, MP_k) = w(MP_k)$
  $A(index(k)) = A(index(k)) \cup (r_u, w(MP_k))$
end if
for all $P_r(i) \in g_i$ **do**
  Create $u_j = (\bullet T_i, T_i, T_i \bullet)$
end for
for all $u_p, u_q \mid \left(P_r(p)\left(\left(\bigwedge \mid \bigvee \mid \rightarrow\right)P_r(q)\right)\right) \cap P \neq \emptyset$ **do**
  Compute $SS\Theta = SS\Theta \otimes (u_p \otimes u_q)$
end for
Output $SS\Theta$

process and do not depend on any previous events Λ^0, union with the set of events that depend on the already executed events ($\overrightarrow{\mathfrak{D}} \times \Lambda^c$).

$$\acute{\Lambda} = \Lambda^0 \cup (\overrightarrow{\mathfrak{D}} \times \Lambda^c) \tag{14}$$

$\overrightarrow{\mathfrak{D}} = \overrightarrow{\Lambda} \times \acute{\mathfrak{D}}$ is a vector that determines the dependency of a certain event upon the set of events that have already been executed ($\overrightarrow{\Lambda}$). $\acute{\mathfrak{D}} = \eth_i \times \mathfrak{D}$ is a vector that outlines the dependency of a single event $\eth_i$ from matrix $\mathfrak{D}$, where $\mathfrak{D}$ is a two-dimensional matrix with length equal to the number of existing primitive events that cannot be fragmented into simpler events (Λ). The matrix $\mathfrak{D}$ defines the dependencies between events as follows:

$$\mathfrak{D}\left[\eth_i, \eth_j\right] = \begin{cases} 0 & \text{if event } \eth_i = \text{event } \eth_j \mid \eth_i \text{ does not depend on } \eth_j \\ 1 & \text{if event } \eth_i \text{ depends on event } \eth_j \end{cases} \tag{15}$$

The set of candidate events $\acute{\Lambda}$ provide an overview of the events' sequence in a service composition process beginning with the event leading to the selection of source node MS and sink node MC. The set $\acute{\Lambda}$ contains all possible plan threads' events that may exist in a single composition plan and thus multiple event sequences are generated, which in turn form multiple threads.

B. PROBABILITY OF EVENT OCCURRENCE

The second step in the composition plan learning process involves determining the probability for an event to occur from the set of candidate events $\acute{\Lambda}$ found in the first step. This is achieved by first calculating a belief value for an event occurrence according to (16).

$$bel(\eth_i) = \int_{j=1}^{n} P(\eth_i|\eth_j) \times MAX(\overrightarrow{\mathfrak{D}}_j, \mathfrak{H}(\|\eth_i \cap \Lambda^0\|))d_{\eth} \tag{16}$$

where $P(\eth_i|\eth_j)$ is the probability for the event $\eth_i$ to occur given the occurrence of the event $\eth_j$. This probability is calculated according to a uniform distribution function. $\mathfrak{H}$ is a step function which produces 0 if the magnitude is 0 and 1 otherwise.

Once the believe values have been derived for the set $\acute{\Lambda}$, a normalized probability is calculated for each event according to (17).

$$P_N(\eth_i) = \frac{bel(\eth_i)}{\sum_{j=0}^{n} bel\left(\eth_j\right)} \tag{17}$$

C. MERGING THREADS

The previous two steps of the plan learning method develops a composition path (thread) from the set of candidate events. In this step, the composition plan which may be composed of multiple threads is built by merging those threads together.

Merging threads is the process of redefining the relationship between threads and merging them into a bigger entity (plan).

We assume that thread events are defined through a sequence of levels in which events are categorized by their hierarchy level. Two different threads having the same set of events occurring at the same hierarchy level can be merged into the same plan. Therefore, the operation of thread merging is achieved through thread hierarchy level comparison. For instance, different composition plan paths can exist to compose the requested service. Such that the same requested service may be composed through different set of nodes and paths (i.e. different nodes offering similar services). Both solutions may create similar thread events that belong to the same execution sequence level. Given such circumstances, we can merge the two threads into a single plan which can be adopted for future service composition requests.

The steps involved in the plan merge process are as follows:

1. Create a common start event $\eth_0$ which is the starting event for all different composition paths (i.e. threads). E.g. availability of the original non-enhanced movie on a node would be considered the starting event before enhancing the movie.
2. Create a common end event $\eth_{end}$, which is the final event for all different composition paths. E.g. movie enhancements are completed, would be considered the final event.
3. $\forall(lev_i \in \epsilon_n$ and $\forall lev_j \in \epsilon_m)$, if $i = j$ then $\forall(\eth_s \in lev_i$ and $\eth_t \in lev_j)$, $\eth_k = \eth_s \cup \eth_t$.
4. $bel\,(\eth_k) = \int P\,(\eth_k) = \int P\,(\eth_s) + P\,(\eth_t)\,d_{\eth}$.
5. After the merge is complete for all threads, obtain the normalized probabilities from the belief values found in step 4.

lev_i is a level in thread ϵ_n, lev_j is a level in thread ϵ_m, and k is the level that results from merging lev_i and lev_j. In other words, the resulting composition plan would have a single start event, a single end event, and for each level the common events are modelled once with probability of occurrence equal to the sum of the two occurrences before merging.

Figure 9 provides an illustrative example of two composition plans (threads) considered for the addition of media enhancements requested by a MC to an original movie content found at a MS. Figure 9(a) illustrates the thread events involved in the composition process. Figure 9(b) illustrates different but similar thread events in the composition process. Figure 9(c) provides an illustration of merging the two threads in which common events are modelled once only.

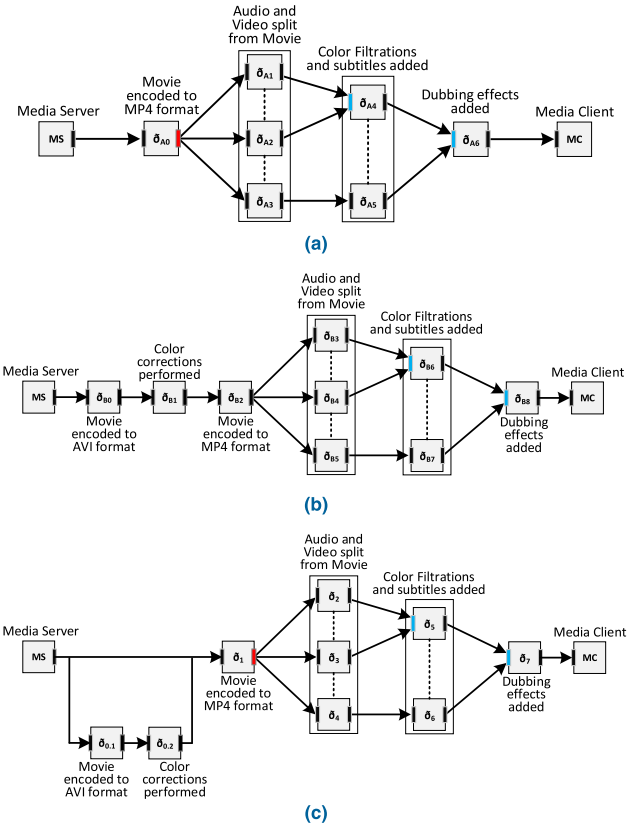


FIGURE 9. (a) Composition plan thread 1 – events for service overlay composition path 1, (b) Composition plan thread 2 – events for service overlay composition path 2, (c) Plan merging – merged threads for the same service request.

VIII. SOUNDNESS OF THE SSO

The presented composition method considers the correctness of the composed overlay in terms of soundness. Soundness is the property which indicates that for a process with a start marking of a certain number of tokens in its source place, can reach the termination state marking with the same number of tokens in its sink place [7]. In other words, for an SSO composition workflow to be sound, a planned path from the MS to the MC through MPs should exist to compose a service. The original non-enhanced service at the MS will be rendered at each MP to deliver the requested composite service to the MC. The service should not be rendered without assurance that the requested service task will be completed and delivered to the MC at some point. Soundness has been addressed in the literature where some of the work focuses on the assurance of workflow-net behavior correctness and complexity [38], [39]. The following framework describes the SSO composition workflow:

$$SS\Theta = \langle \Lambda, \Sigma, \Omega, N, \Psi, \Upsilon \rangle \tag{18}$$

where $\Lambda = \{R_1^M, R_2^M, \ldots, R_\ell^M\}$ is the set of service requests described in (6). $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_s\}$ is the set of basic service capabilities σ provided by the service node, such that $w\,(MP_i) \subseteq \Sigma$ where $w\,(MP_i) = \{\sigma_i, \sigma_j, \ldots, \sigma_k\}$ is the service capability set for a service node where $\forall\sigma \in w\,(MP_i)$, $\sigma \in \Sigma$. $\Omega = \{w\,(MP_1), w\,(MP_2), \ldots, w\,(MP_k)\}$ is a mapping function that maps every service request R_ℓ^M to its basic service capabilities σ_s. $N = \{MP_1, MP_2, \ldots, MP_k\}$ is the set of service nodes in the SSO. Ψ is the communication protocol used between the service nodes, and $\Upsilon = \{WF_{net_1}, WF_{net_2}, \ldots, WF_{net_l}\}$ is the set of workflows describing the service capabilities of every service node.

Theorem 1: A cooperative service specific overlay $SS\Theta$ is sound if and only if:

1. $\forall R^M \in \Lambda, \exists \sigma \in \Sigma | \Omega(R^M, \sigma) \neq \emptyset$.
2. $\forall \left(MP_i, MP_j\right) \in N \left| \exists \left(MP_i \times MP_j\right) \right| MC \in [MP_i\rangle$.
3. $\forall MP_i, MP_j | MC \in [MP_j\rangle$ and $MP_j \in [MP_i\rangle$ and $MP_i \in [MS\rangle$.
4. $\exists \Psi | MP_i \times MP_j$ is always true and $MP_i \times MS$ is always true and $MP_j \times MC$ is always true.
5. $\forall \Upsilon \in w(MP)$, Υ is sound at any point in time t if $\exists MP_i | MP_i \in SS\Theta_{t-k}$ and $MP_i \notin SS\Theta_t$ then $\exists MP_j | MP_j \in SS\Theta_t$ and $\Upsilon(MP_j) \equiv \Upsilon(MP_i)$.

To proof the theory, we need to proof that if $SS\Theta$ is sound, then all five conditions are satisfied, and vice versa.

Proof of the "If-part": For $SS\Theta$ to be sound then $\forall R^M \in \Lambda, R^M \in MS_{t-k}$ and $R^M \in MC_t$ where t is time and $k \geq 0$. Therefore $\exists MP_l \in [MS\rangle$ and $MC \in [MP_l\rangle$ where $0 \leq l \leq m$ where m is the path length from MS to MC. Therefore, $\forall MP_l \in [MS\rangle$ and $\forall MP_j | MC \in [MP_i\rangle, \exists \Psi | MP_i \times MP_j \neq \emptyset$. Additionally, since the task has to be covered for it to be performed, therefore $\forall R^M \in \Lambda, \exists \sigma \in \Sigma | \Omega(R^M, \sigma) \neq \emptyset$. Since $MC \in [MS\rangle$, therefore, $\exists \left(MP_i \times MP_j\right) | MP_i \in [MC\rangle$ and $MC \in [MP_j\rangle$ and $MP_i \times MP_j \neq \emptyset$ and, therefore, $\forall \Upsilon \in w(MP)$, Υ is sound for $R^M \in MC$.

Proof of the "Only-If-part": Since $\forall R^M \in \Lambda \exists \sigma \in \Sigma | \Omega(R^M, \sigma) \neq \emptyset$, therefore, $\forall R^M \in \Lambda, R^M$ can be performed using the composed overlay $SS\Theta$. Since $\forall \left(MP_i, MP_j\right) \in N \left| \exists \left(MP_i \times MP_j\right) \right| MC \in [MP_i\rangle$, therefore, $MC \in [MS\rangle$ and therefore, $\exists$ path $MS \cdot MP_i \cdot MP_j \cdot \ldots \cdot MP_k \cdot MC | R^M \in MS_{t-k}$ and $R^M \in MC_t$. Since Ψ is sound, then if $\exists \left(MP_i \times MP_j\right)$ and $MP_j \in [MP_i\rangle$, therefore, $R^M \in MP_{i_{t-k}}$ and $R^M \in MP_{j_t}$. Since $\forall \Upsilon \in w(MP)$, Υ is sound, then $\forall R^M \in MS_{t-k}, R^M \in MC_t$ therefore $SS\Theta$ is sound.

Lemma 1: If $SS\Theta$ is sound, then $\exists MP_i, MP_j, \ldots, MP_k | \bigcup_{i,j,\ldots,k} w(MP_i) \equiv (R_1^M, R_2^M, \ldots, R_k^M)$ where $R_k^M \in \Lambda$.

IX. EXPERIMENTAL SIMULATIONS

System evaluations were conducted using two simulation setups. The first simulator setup was used to demonstrate the effectiveness of node cooperation and parallelism in workflow net plan execution regardless of the type of service being provided. The second simulator setup integrates a realistic network scenario which focuses on the creation of overlays that provide multimedia services to clients. The evaluations were performed to demonstrate the effectiveness of applying the SSO composition process in terms of stability and experienced delay.

A. FIRST SIMULATOR SETUP AND RESULTS

The first simulator used was built using C++ for the SSO composition framework, in which the algorithm described in Section VI was implemented. The input of the simulator consists of a plan in the form of a linear logic expression which constitutes a set of service nodes (agents), each with a set of capabilities expressed as workflow nets. Each capability corresponds to one or more actions defined in the plan, along with the cost associated with performing that action. Capability costs are assigned as follows: first, a uniformly distributed random variable is used to determine the initial set of capabilities for each service node. When a node is assigned a capability, the cost for its execution is randomly determined with a normally distributed variable. Once the node capabilities are set, the simulation evaluates the MC's composed service requirements. If the generated node capabilities are insufficient to provide a complete composed service, the simulator terminates. Otherwise, the cooperative plan is constructed and executed.

A set of 50 mobile service nodes (agents) in 100 simulations were used such that each node possessed different number of service capabilities. We assumed a set of seven different service capabilities, in which we controlled the probability for a service node to process each capability. For example, given a task coverage probability of 20%, each one of the 50 nodes has a 20% chance of processing all seven service capabilities. Experiments were conducted with task coverage probabilities in the range of 1-100%. Time units are expressed in terms of transition costs in the workflow nets. The plan to be executed by the experiment is outlined in Figures 3 and 4 and is expressed as follows:

$$MS \rightarrow \delta_0 \rightarrow \left(\left(\left(\delta_1 \bigwedge \delta_2\right) \rightarrow \delta_4\right) \bigwedge \left(\delta_3 \rightarrow \delta_5\right)\right) \rightarrow \delta_6 \rightarrow MC \quad (19)$$

where each predicate represents a unique service capability. The cost of performing those predicates is outlined below through matrix M. From the matrix, it is clear that the minimal cost of performing the service composition is achieved through the plan outlined in equation (19).

$$M = \begin{array}{c} \\ MS \\ MP_1 \\ MP_2 \\ MP_3 \\ MP_4 \\ MC \end{array} \begin{array}{c} \begin{array}{ccccccc} \delta_0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 \end{array} \\ \begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 2 & 3 & 8 & 5 & 3 & \infty & \infty \\ \infty & 1 & 1 & 2 & 7 & \infty & 3 \\ \infty & 3 & 3 & 8 & 1 & 2 & 5 \\ \infty & 5 & 9 & 7 & 1 & \infty & 1 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix} \end{array}$$

The first set of experimental results obtained are used to demonstrate the way by which our solution exploits cooperation and parallelism. Figure 10 compares two workflow net plans WF_1 and WF_2, in which WF_2 does not associate service node cooperation, such that tasks are performed sequentially, whilst WF_1 associates full node cooperation. It is evident from the figure that WF_1 outperforms WF_2 in terms of execution time reduction as the number of achieved service tasks increase. Additionally, results show that when the two workflows work in parallel to achieve the required composed service, time needed to achieve the required task is reduced significantly.

The second set of experiments involved the use of four plans. Plan[0] incorporates full workflow net parallelism, Plan[1] incorporates full service node cooperation, Plan[2] incorporates some degree of cooperation, and Plan[3] provides no mechanism for cooperation in which tasks are performed sequentially. Results depicted in Figure 11 show that

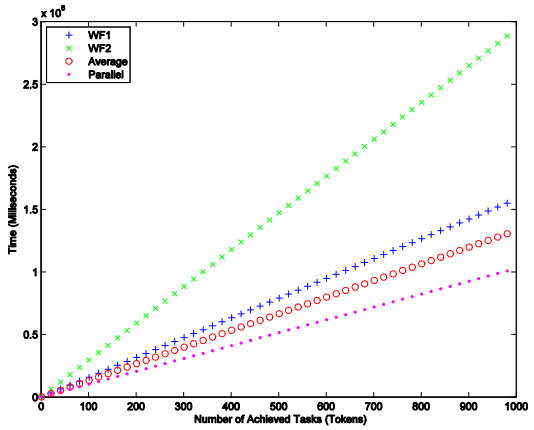


FIGURE 10. Comparing execution time for two workflow nets, WF_1 - with cooperation and WF_2 - without cooperation.

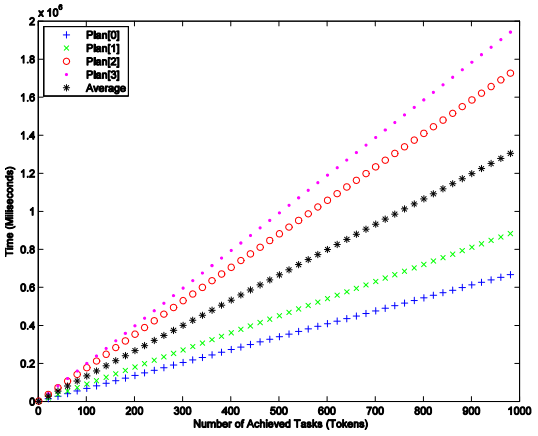


FIGURE 11. Comparing execution time for different plans while varying degree of cooperation and parallelism.

Plan[0] outperforms all other plans in terms of time required to achieve the requested tasks. On the contrary, Plan[3] provides the least favorable result. Under a realistic network scenario, it is well-known that full parallelism is impossible to achieve and hence Plan[1] provides the most optimal solution provided that all service nodes are willing to cooperate.

The third set of experiments explored the effects of varying the number of service nodes (agents) against a plan's required execution time. Figure 12(a) shows the cases in which probabilistic task coverages are insufficient to complete a plan. It is clear from the figure that such cases occur with a very low number of service node availability or low task coverage probability. On the contrary, Figure 12(b) shows the cases of sufficient task capability coverages. Results demonstrate that extended service node capabilities reduce execution time more drastically with the increase in the number of service nodes. In this particular case, the solution favors nodes with better task coverage than node availability.

B. SECOND SIMULATOR SETUP AND RESULTS

In the second phase of experiments, we adopt the successfully tested workflows from the first phase of experiments and

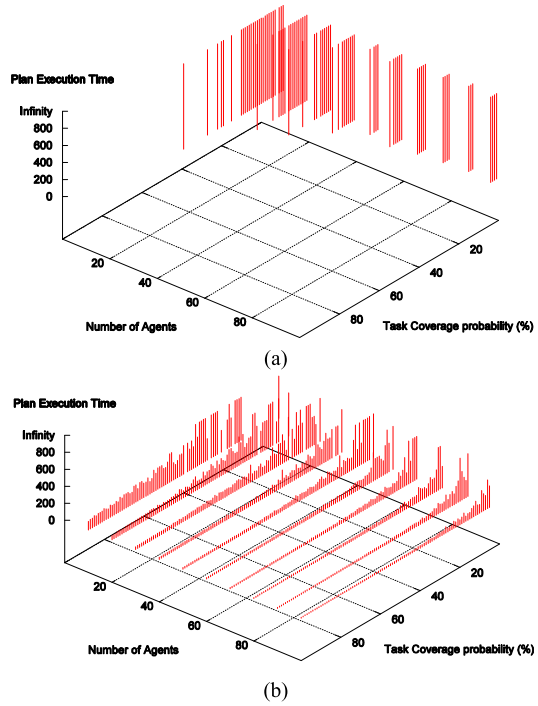


FIGURE 12. (a) Cases of insufficient task coverage to complete a plan. (b) Cases of sufficient task coverage to complete a plan.

test it on a mobile network oriented scenario. The simulation experiments conducted in this phase were performed using the C++ based OMNeT++ [40] discrete event network simulator. OverSim [41] was used to model the proposed service-specific overlay composition technique in a mobile network environment. In the performed tests, nodes were placed in a 2-dimensional Euclidean space where the delay between any two nodes was assumed to be proportional to the distance between them. Two thousand nodes were randomly distributed and divided into 5% MCs, 5% MSs, and 90% MPs. Each MP is associated randomly with a set of services where each service is provided at specific QoS levels. A lifetime based churn model of exponential distribution was used to model the arrival and departure of nodes in the network. A fully-recursive KBR protocol is used to model the network traffic, where messages are encapsulated and forwarded to the next service node according to the composition process. Message sizes range from 32 bytes to 100 bytes and are generated at a 60 seconds' interval with a failure latency of 10 seconds. Nodes are added to the network according to a Poisson process at a rate of 1 node/second. OntoCAT [42] was used to parse, search though, and compare acquired service description files. It provides a high-level of abstraction for interacting with ontology resources in the standard OWL format.

The set of experiments conducted compares our proposed workflow-net plan-based service composition solution, referred to as *WN-SSO* henceforth, to our earlier work presented in [15] referred to as *PF-SSO* and the work presented in [43] referred to as *limited PF-SSO*. The work in [15]

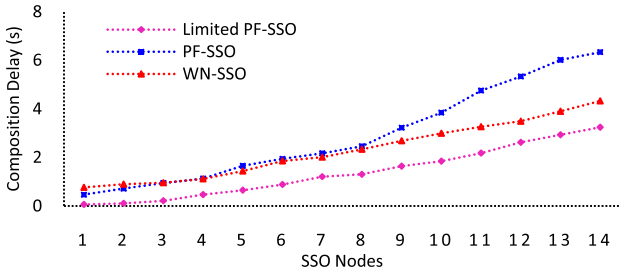


FIGURE 13. Composition delay experienced while increasing composition path length.

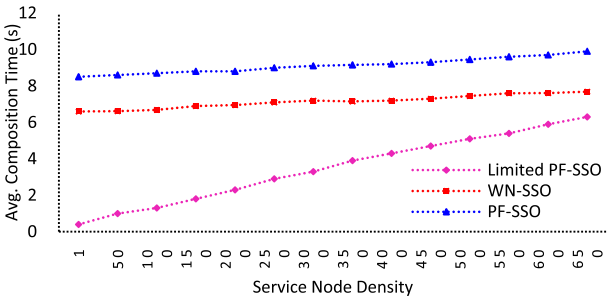


FIGURE 14. Average composition delay while varying service node density.

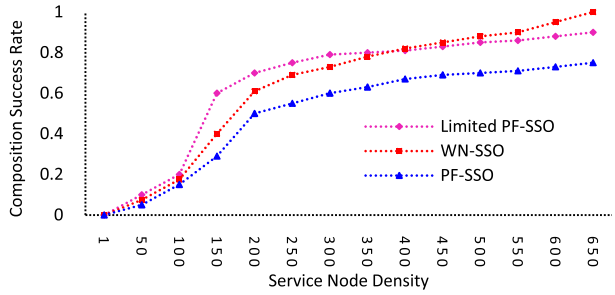


FIGURE 15. Composition success rate while varying service node density.

uses a decentralized plan-free semantic-based service composition technique, in which semantic similarity and semantic nearness of overlay nodes are considered when creating overlay paths to deliver a particular composed service. The solution semantically advances composition paths towards users' needs with each service hop while guaranteeing a user-acceptable QoS level. On the contrary, the work presented in [43] is also a decentralized service composition technique but with an assumption that each MP has a distance function used to produce the list of required adaptations for a media flow based on its input and output. A methodology for quantifying the compatibility and similarity between service descriptions is not used, but similarity between MPs is considered. However, the similarity function relies on a globally accessible directory of MP descriptions that is not regularly updated and hence optimal SSO compositions are not achieved.

A comparison of the SSO composition delays for the three techniques, namely: WN-SSO, PF-SSO, and limited PF-SSO are presented in Figure 13. Results illustrate how delay increases steadily as the number of MPs involved in the SSO path grows. The service composition time of both WN-SSO and PF-SSO is slightly higher than that of the limited PF-SSO. This increase is due to the lack of detailed QoS and semantic comparison evaluations in the limited PF-SSO technique. Limited PF-SSO thus composes SSOs faster, however, it lacks semantic and QoS precision. Additionally, the figure shows that WN-SSO outperforms PF-SSO in terms of SSO composition speed while achieving the same high level of semantic and QoS precision. Furthermore, the flooding approach used in the PF-SSO and the limited PF-SSO results in a number of messages exchanged that far exceeds the plan-based method, where messages are only forwarded to specific overlay nodes, hence resulting in a reduced composition delay.

Figure 14 depicts a comparison of the average composition time between the three methods. The average composition time is the time difference between the start of the composition request and the completion of the SSO path. Results show that the limited PF-SSO technique experiences less delay in low service node density environments. As node density increases, the average composition time increases almost exponentially. On the contrary, both the WN-SSO and PF-SSO techniques experience more of a somewhat stable average composition delay in both low and high service density environments. Although limited PF-SSO provides the least average composition time overall, limited PF-SSO incorporates less realistic and less complex service description models, as well as the utilization of semantic and QoS comparisons when selecting MPs. Comparing WN-SSO to PF-SSO, we see that our technique provides a more stable average composition time as node density increases. Additionally, WN-SSO provides the same level of semantic and QoS comparison precision with almost a twofold decrease in average composition time.

Figure 15 compares the composition success rate for the three methods. Success rate is defined as the number of service requests that receive positive responses divided by the total number of queries. The proposed WN-SSO technique outperforms both the limited PF-SSO and PF-SSO composition methods in highly dense environments. In less dense environments, although success rates in both WN-SSO and PF-SSO slightly fall below those of limited PF-SSO, yet this decline is justified by the increase in the accuracy of node selection. Both WN-SSO and PF-SSO better meet the MC's media flow format and QoS requirements. WN-SSO achieves this with a higher success rate compared to PF-SSO.

Our final test involved measuring the delay experienced in forming each hop of the SSO composition as depicted in Figure 16. Both PF-SSO and limited PF-SSO maintained higher levels of delay for each composed hop compared with WN-SSO. Although our proposed technique achieves lower levels of delay overall, it can be noticed that per hop composition delay increases per hop as opposed to a decrease in per hop composition for both PF-SSO and

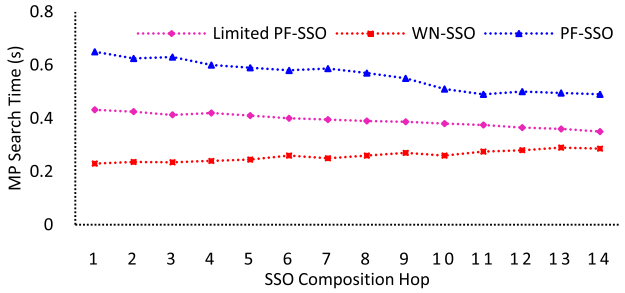


FIGURE 16. Composition success rate while varying service node density.

limited PF-SSO. This is in fact due to the use of a decentralized MP search mechanism in the PF-SSO techniques which limits the number of MPs contacted in each hop, and thus, the acceptable QoS levels set by the MC in its request becomes less restrictive due to the limited set of MP availability. On the contrary, since the WN-SSO mechanism uses a plan-based composition method which is incorporated within the cloudlet or registered edge node, the MP search process becomes more restrictive by considering not only the MC's request, but also the restrictions placed by the MPs in the path.

Consistently, our proposed workflow-net plan-based SSO composition technique provides performance levels that outpace both PF-SSO and limited PF-SSO in terms of composition delay and success rate. Additionally, it provides guaranteed QoS levels that meet the MC's service requirements

X. CONCLUSION AND FUTURE WORK

This paper proposed a workflow-net based mechanism for mobile edge node cooperation in cloud networks to form guaranteed SSOs for media content delivery. The solution integrates a cooperation operator used to compose node service capabilities expressed as workflow net units into a cooperative composed workflow net. The service composition algorithm implements a back-tracking scheme allowing it to determine the minimal cost cooperative path from the workflow net. Simulation results demonstrate that the system is scalable to any number of mobile nodes given any number of service capabilities and provides the minimal cost towards service composition. Further evaluations are planned to investigate the behavior of the proposed system in real-time under different network conditions.

REFERENCES

[1] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[2] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, "Foggy clouds and cloudy fogs: A real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Commun.*, vol. 23, no. 5, pp. 120–128, Oct. 2016.

[3] V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramirez, and S. Sanchez, "Towards distributed service allocation in fog-to-cloud (F2C) scenarios," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6.

[4] Z. Duan, Z.-L. Zhang, and Y. T. Hou, "Service overlay networks: SLAs, QoS, and bandwidth provisioning," *IEEE/ACM Trans. Netw.*, vol. 11, no. 6, pp. 870–882, Dec. 2003.

[5] Y. Xia, X. Luo, J. Li, and Q. Zhu, "A petri-net-based approach to reliability determination of ontology-based service compositions," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1240–1247, Sep. 2013.

[6] Y. Du, X. Li, and P. Xiong, "A Petri net approach to mediation-aided composition of Web services," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 2, pp. 429–435, Apr. 2012.

[7] W. M. P. van der Aalst, "The application of Petri nets to workflow management," *J. Circuits, Syst. Comput.*, vol. 8, no. 1, pp. 21–66, 1998.

[8] F. L. Tiplea and C. Bocaneala, "Priority workflow nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 2, pp. 402–415, Mar. 2013.

[9] Y. T. Kotb and E. Badreddin, "Synchronization among activities in a workflow using extended workflow Petri nets," in *Proc. 7th IEEE Int. Conf. E-Commerce Technol. (CEC)*, Jul. 2005, pp. 548–551.

[10] Q. Duan, Y. Yan, and A. V. Vasilakos, "A survey on service-oriented network virtualization toward convergence of networking and cloud computing," *IEEE Trans. Netw. Serv. Manage.*, vol. 9, no. 4, pp. 373–392, Dec. 2012.

[11] I. Paik, W. Chen, and M. N. Huhns, "A scalable architecture for automatic service composition," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 82–95, Jan./Mar. 2014.

[12] K. Fujii and T. Suda, "Semantics-based dynamic service composition," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 12, pp. 2361–2372, Dec. 2005.

[13] L. Zhen and M. Parashar, "Rudder: A rule-based multi-agent infrastructure for supporting autonomic grid applications," in *Proc. Int. Conf. Auto. Comput.*, 2004, pp. 278–279.

[14] D. Ardagna and B. Pernici, "Global and local QoS constraints guarantee in Web service selection," in *Proc. IEEE Int. Conf. Web Serv.*, Jul. 2005, pp. 1–2.

[15] Y. A. Ridhawi and A. Karmouch, "Decentralized plan-free semantic-based service composition in mobile networks," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 17–31, Jan./Feb. 2015.

[16] E. Sirin, B. Parsia, and J. Hendler, "Filtering and selecting semantic Web services with interactive composition techniques," *IEEE Intell. Syst.*, vol. 19, no. 4, pp. 42–49, Jul. 2004.

[17] I. W. Kim and K. H. Lee, "A model-driven approach for describing semantic Web services: From UML to OWL-S," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 6, pp. 637–646, Nov. 2009.

[18] H. Wu and D. Huang, "MoSeC: Mobile-cloud service composition," in *Proc. 3rd IEEE Int. Conf. Mobile Cloud Comput., Serv., Eng.*, San Francisco, CA, USA, Mar. 2015, pp. 177–182.

[19] P. Simoens, L. Van Herzeele, F. Vandeputte, and L. Vermoesen, "Challenges for orchestration and instance selection of composite services in distributed edge clouds," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, Ottawa, ON, Canada, May 2015, pp. 1196–1201.

[20] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, Glasgow, U.K., May 2015, pp. 1–6.

[21] S. Wang, A. Zhou, F. Yang, and R. N. Chang, "Towards network-aware service composition in the cloud," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2016.2603504.

[22] S. Padmavathi, V. P. Dharani, S. Maithrreye, M. M. Devi, and S. Durairaj, "Multi-agent framework for cloud service composition," in *Proc. Int. Conf. Emerg. Trends Eng., Technol. Sci. (ICETETS)*, Pudukkottai, India, Feb. 2016, pp. 1–5.

[23] S. Deng, L. Huang, H. Wu, and Z. Wu, "Constraints-driven service composition in mobile cloud computing," in *Proc. IEEE Int. Conf. Web Serv. (ICWS)*, San Francisco, CA, USA Jun. 2016, pp. 228–235.

[24] Z. Benzadri, N. Hameurlain, F. Belala, and C. Bouanaka, "A theoretical approach for modelling cloud services composition," in *Proc. Int. Conf. Adv. Aspects Softw. Eng. (ICAASE)*, Constantine, Algeria, 2016, pp. 1–8.

[25] T. Wu, W. Dou, C. Hu, and J. Chen, "Service mining for trusted service composition in cross-cloud environment," *IEEE Syst. J.*, vol. 11, no. 1, pp. 283–294, Mar. 2017.

[26] P. Baldan, A. Corradini, H. Ehrig, and R. Heckel, "Compositional semantics for open Petri nets based on deterministic processes," *J. Math. Struct. Comput. Sci.*, vol. 15, no. 1, pp. 1–35, Feb. 2005.

[27] X. Li, Y. Fan, Q. Z. Sheng, Z. Maamar, and H. Zhu, "A Petri net approach to analyzing behavioral compatibility and similarity of Web services," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 41, no. 3, pp. 510–521, May 2011.

[28] K. Klai and H. Ochi, "A formal approach for service composition in a cloud resources sharing context," in *Proc. 16th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid)*, Cartagena, Colombia, May 2016, pp. 458–461.
[29] S. Haddad, J.-M. Ilié, and K. Klai, "Design and evaluation of a symbolic and abstraction-based model checker," in *Proc. ATVA*, 2004, pp. 196–210.
[30] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, Aug. 2016.
[31] I. Al Ridhawi, Y. Kotb, M. Aloqaily, and B. Kantarci, "A probabilistic process learning approach for service composition in cloud networks," in *Proc. IEEE 30th Can. Conf. Electr. Comput. Eng. (CCECE)*, Windsor, ON, Canada, Apr. 2017, pp. 1–6.
[32] S. Herborn, Y. Lopez, and A. Seneviratne, "A distributed scheme for autonomous service composition," in *Proc. 1st ACM Int. Workshop Multimedia Serv. Compos. (MSC)*, vol. 5. 2005, pp. 21–30.
[33] S. Herborn and A. Seneviratne, "Service composition for mobile personal networks," in *Proc. 3rd Annu. Int. Conf. Mobile Ubiquitous Syst., Netw. Serv.*, 2006, pp. 1–8.
[34] Y. Al Ridhawi and A. Karmouch, "Ontology-based negotiation protocol and context-level agreements," in *Proc. 4th IET Int. Conf. Intell. Environ.*, Jul. 2008, pp. 1–8.
[35] G. Antoniou and F. van Harmelen, "Web ontology language: OWL," in *Handbook on Ontologies* (International Handbooks on Information Systems). Berlin, Germany: Springer, 2009, pp. 91–110.
[36] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distrib. Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.
[37] Y. T. Kotb, S. S. Beauchemin, and J. L. Barron, "Workflow nets for multiagent cooperation," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 1, pp. 198–203, Jan. 2012.
[38] G. Liu, W. Reisig, C. Jiang, and M. Zhou, "A branching-process-based method to check soundness of workflow systems," *IEEE Access*, vol. 4, pp. 4104–4118, 2016.
[39] G. Liu, "Some complexity results for the soundness problem of workflow nets," *IEEE Trans. Serv. Comput.*, vol. 7, no. 2, pp. 322–328, Apr./Jun. 2014.
[40] A. Varga. (2012). *OMNeT++ User Manual, Version 4.1*. [Online]. Available: http://www.omnetpp.org/doc/omnetpp/manual/usman.html
[41] I. Baumgart, B. Heep, and S. Krause, "OverSim: A flexible overlay network simulation framework," in *Proc. 10th IEEE Global Internet Symp. (GI)*, May 2007, pp. 79–84.
[42] T. Adamusiak *et al.*, "OntoCAT—Simple ontology search and integration in Java, R and REST/JavaScript," *BMC Bioinform.*, vol. 12, no. 1, p. 218, 2011.
[43] I. Al-Oqily and A. Karmouch, "SORD: A fault-resilient service overlay for MediaPort resource discovery," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 8, pp. 1112–1125, Aug. 2009.

ISMAEEL AL RIDHAWI (M'09) received the B.A.Sc., M.A.Sc., 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 with the College of Engineering and Technology, American University of the Middle East. His current research interests include quality of service monitoring, cloud network management, and overlay networks.

YEHIA KOTB received the Ph.D. degree from the Faculty of Computer Science, University of Western Ontario, Canada, in 2011. He was a Senior Software Developer with Akira Systems, London, ON, Canada. He is currently an Assistant Professor of computer engineering with the College of Engineering and Technology, American University of the Middle East. His current research interests include probabilistic process learning and multiagent cooperation in distributed systems.

YOUSIF AL RIDHAWI (M'07) received the B.Sc., M.Sc., and Ph.D. degrees in electrical and computer engineering from the University of Ottawa, Canada, in 2006, 2008, and 2013, respectively. His current research interests include autonomic service discovery and composition, quality of service management, and overlay wireless network communications.