IET Networks

Research Article

IET Journals
The Institution of Engineering and Technology

Network service orchestration in heterogeneous 5G networks using an open marketplace

ISSN 2047-4954
Received on 31st March 2017
Revised 31st July 2017
Accepted on 14th September 2017
doi: 10.1049/iet-net.2017.0058
www.ietdl.org

Shireesh Bhat[1] ✉, Robinson Udechukwu[1], Rudra Dutta[1], George N. Rouskas[1,2]

[1]*Department of Computer Science, North Carolina State University, Raleigh, NC, USA*
[2]*Computer Science Department, King Abdulaziz University, Jeddah, Saudi Arabia*
✉ *E-mail: sbhat@ncsu.edu*

Abstract: Network service orchestration across heterogeneous networks needs an open marketplace where the services advertised by the providers in different domains can be purchased for short-term or long-term time scales. The authors present two design paradigms and evaluate two corresponding prototypes which provide a framework for network services to be purchased. They compare the two prototypes from the point of view of how effective they are in addressing some of the challenges posed by heterogeneous 5G networks namely programmability, scalability and innovation. They present a network service orchestration algorithm which is advertised as a network service in the marketplace.

1 Introduction

The challenges and opportunities presented by the introduction of 5G networks can be broadly classified as below:

- Increased bandwidth and lower latency to support communication between the 'connected' devices can be achieved by dark fibre.
- Increased programmability and innovation can be supported with the help of software defined networking (SDN) and network function virtualisation (NFV).

In this work, we focus on the programmability and innovation which is brought about by the competition between the providers of network service. For any multi-domain network to function together while providing more control at the hands of the user would require opening up certain portions of the network for customisation and optimisation, which is also one of the goals of 5G networks as it transits from 4G. A open marketplace encourages competition between the service providers by allowing the services to be standardised and compared. The marketplace also allows the resellers of a service to thrive by making no distinction between a seller and a reseller and focusing only on the service. A planner/orchestration [1–5] service is an example of a reseller service which is responsible for stitching together services from different heterogeneous network to compose an end-to-end service required by the user. The prerequisite for network service orchestration is to define services in a way it can be stitched/composed. We present two design models to construct a marketplace, in the first design we extend the web services model to encompass network services and in the second model we develop a network services marketplace using a clean state design.

Some of the goals of 5G networks namely programmability and innovation is also shared by the future Internet design. The Internet in its current form fails to make room for innovation in the core of the network. To overcome this limitation one of the possible solutions being proposed is to support choice. Choice implies that users can choose from alternatives that can be deployed dynamically into the network. The notion of choice can be extended to services. To address this, we need a sustainable framework where services can be published, queried and composed. We can see that the design goals of the future Internet and 5G networks have so much in common. One of the solutions to usher in innovation is to incentivise providers at all levels in the network and to be able to do so at more granular time scales. In our earlier work, we developed the idea of ChoiceNet [6] and discussed the importance of an open marketplace. In this work, we develop the idea further and present two prototypes which are designed to encourage innovation by increasing competition between the service providers by providing a level playing field.

The biggest challenge when we try to combine different carrier networks is how we compensate or divide the revenue between the various stakeholders. The authors in [6–8] argue that the economic factors act as an impediment for the innovation and rightly so. To overcome this hurdle in our earlier work we proposed an 'economy plane' [6] which includes all the interactions leading to and signing of the economic contract between the service provider and the service user. The economy plane [6] makes dynamic deployment of network services possible by enabling setting up of economic contracts. The objective of an economy plane is to allow users to choose between a wide variety of network offerings and not be constrained by a handful of them. The economy plane benefits the providers and users equally; the providers receive compensation through economic contracts forged between them and the users, the users receive better service as it is enforced by the economic contract. The providers return tokens to use the service. The network service can be offered for a short duration which allows contracts to be signed for a short timescale thereby freeing the user from having an association with just one service provider for a longer duration.

Once we have defined the network services the intermediate step before we sign the economic contracts is network orchestration, which involves stitching together services across domains to come up with an end-to-end service. In this work, we use the term 'network service orchestration' and 'service composition' interchangeably.

Our contribution is summarised below:

- In the first prototype, we extend the web services model to network services and build a framework for advertising and purchasing network services. In this work, we define a mechanism for purchasing and authorising the services which make up a composed service.
- In the second prototype, we introduce a new semantic language for advertising network services and show how the orchestration algorithm developed by us can make use of the standardised interfaces to construct a composed service.

Following the introduction, we present the related work in Section 2 and discuss its influence on our design. In Section 3, we

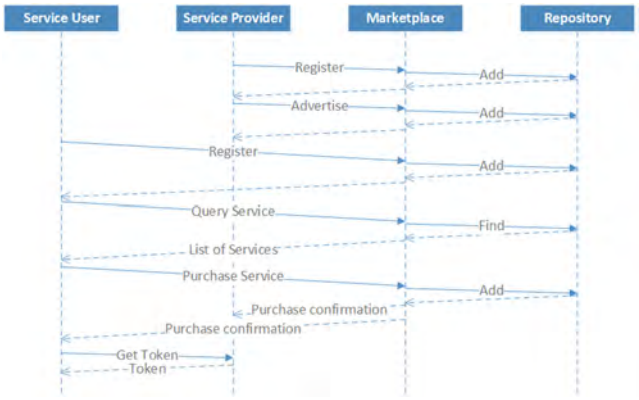

Fig. 1 *Economy plane: components*

discuss the first design and the corresponding prototype. In Section 4, we present the second design and its prototype. In Section 5, we present the network orchestration algorithm developed for the second prototype. In Section 6, we compare the pros and cons of the two prototypes and finally we conclude the paper in Section 7.

2 Related work

The related work can be broadly classified into two categories. In the first category, we look at work which focuses on the services which make up the marketplace. In the second category, we look at prior work which focuses on the language which defines the network services and the interaction between the various entities which use the marketplace.

2.1 'Services'

The concept of marketplace for advertising and purchasing 'services' is not new and the first attempts at standardising can be traced back to early 2000.

Universal Description Discovery and Integration (UDDI) [9, 10] a public service entity was designed and hosted by a select group of companies. These companies were responsible for maintaining the database containing information on business registry data. Simple object access protocol (SOAP) was used for publishing and querying for information. Business data can be registered with one of the vendors and it is the responsibility of the UDDI to replicate this information across other vendors. UDDI provides separate Web Services Description Language (WSDL) files for registration and discovery services. UDDI failed to find acceptance as it was a trust deficient model and it was designed for isolated services which do not interact with other services.

Universal plug and play (UPnP) [11, 12] offers a real-time picture of the service and its state. UPnP allows a device to advertise its services to the control points in the network using the simple service discovery protocol (SSDP). Similarly when a new control point is added to the network, SSDP allows device discovery. Notification events about the changes in the service are carried to the control points using general notification event architecture. Every service maintains three URLs, ControlURL, EventSubURL and DescriptionURL, that provide the information necessary for control points to communicate with services. UPnP is primarily targeted for end devices like mobile phones, cameras and so on. Although UPnP presents a more modular architecture, the use of UPnP is confined to home area networking.

Open services gateway initiative (OSGi) [13, 14] architecture provides a flexible model for maintaining services. The basic building blocks of OSGi framework are bundles and the service registry. Bundles provide well defined services and they publish the services in the service registry. The services published in the registry can be found by other bundles who want to use this service. OSGi allows for dynamic addition, update and removal of bundles. The simple yet flexible constructs offered by OSGi is modelled for a specific programming language (Java).

Marketplaces have also been used for advertising path services [8, 15, 16], while others have also used a marketplace for advertising virtualised networks and their functions [17, 18]. Some of the cloud service providers advertise services, which can be provided as an edge service in their respective cloud domains, for example transcoding, compute, storage and so on.

2.2 Semantic language

The use of semantic language [19] has found widespread acceptance for web services. WSDL along with web ontology language and resource description framework are used to describe web services and their respective interfaces. SOAP is used to define the syntax of the messages exchanged between the sender and the receiver over HTTP.

Some of the newer semantic language designs [20, 21] have focused on SDN and NFV, respectively. Unlike the semantic language used for web services, the semantic languages being proposed for SDN and NFV are geared more towards functionality and are not quite suited for heterogeneous networks which rely on multi-domain interaction between services. For SDN and NFV to be adopted widely in the carrier networks we need to have two essential things in place. First, a marketplace to be a service repository for multi-domain network services and second, a mechanism for establishing contracts for short-term or long-term time scales. In our earlier work [22–25], we developed the idea of a semantic language for virtualised network services. We demonstrated that the new semantic language construct can be used to model network services from providers who are different domains.

3 Semantic network services with web service constructs

In the first design, we extend the web services constructs to network services.

3.1 Overview

The three main entities which participate in the economy plane are shown in Fig. 1 and described below:

- Marketplace
- Service provider
- Service user

Marketplace provides the framework where service providers can register and advertise new services and withdraw existing service advertisements. The marketplace entity provides templates for service advertisements, based on the common vocabulary/schema and can be used as reference by the service providers. However, the service provider can advertise a service which is not compliant to any of the marketplace templates by extending the common vocabulary. The service provider and the advertised service are uniquely identified with respect to a given marketplace. We assume the existence of a root marketplace which can be used for advertising services and even other marketplaces.

Service provider needs to register with the marketplace using a unique name which is crucial in identifying services belonging to a particular service provider. Once the registration is complete, the service provider advertises services in the marketplace. Each service advertisement is assigned a unique service ID by the marketplace. The marketplace stores service advertisements in a non-volatile memory and also allows withdrawing existing advertisements using the service ID assigned earlier. The only way to modify an existing advertisement is to withdraw it and to advertise as a new service. This helps in avoiding ambiguity with respect to associating an economic contract with a service ID. The advertisements are akin to claims made by service providers about the service provided by them.

Service user also needs to register with the marketplace. After registering, the service user can query the marketplace for services by specifying a filter. The marketplace applies the user specified filter against the list of services in its repository and passes the service advertisements which are a perfect match. The service user

now has a list of alternative services from which he can choose. The service user then contacts the marketplace to purchase the service(s). After getting proof of purchase confirmation, the user contacts the provider to setup the token which is to be used in the data plane for obtaining the service. The service purchase phase between the provider and the marketplace is short since the marketplace is not an auction clearing house but provides a framework where services are sold for the price as advertised. So the purchase phase does not increase the latency nor does it reduce the throughput significantly.

3.2 Service description

The services advertised in the marketplace can be broadly classified as belonging to one of the below three categories.

- *Transport*: These services are responsible for moving a packet from one location to another.
- *Transform*: These services are responsible for altering the content of the packet at a particular location.
- *Preserve*: These services neither move nor change the content of the packet.

For services to be compared against each other and for composability of services which complement each other we need a minimum common vocabulary which is extensible. The service description templates help establish the common vocabulary. These templates are offered in the marketplace as an advertised service by an entity similar to Internet Assigned Numbers Authority (IANA). The service providers advertise their services based on these templates. The IANA like authority would be responsible for adding new templates and removing the obsolete templates. By creating service templates for various services we allow for automated service composition [1]. The templates are structured in a hierarchical manner as described below.

(1) *Root service description template:* A root service description template needs to have the following fields.

- *ServiceProviderName*: the ServiceProviderName can be a simple character string or a bit string representing a self-certifying name [26] which is digitally signed.
- *ServiceFileLocation*: the ServiceFileLocation has the address from where the file containing the service definition can be found, this can be a public dns name or an ipaddress/port combination.
- *ServiceFileContent*: the contents of the service definition file are cached in the field ServiceFileContent in character string format.
- *ServiceDescription*: the ServiceDescription contains a brief description of the service in a simple character string format.

(2) *Transporting service description template:* A transport service template inherits the root service description template and in addition needs to have the following minimum fields.

- *fromAddress*: the fromAddress is an octet string and can represent an IP address.
- *toAddress*: the toAddress is an octet string and can also represent an IP address.

(3) *Transforming service description template:* A service which transforms the data needs to have the mandatory ‘atAddress’ filed and can have the rest of the optional fields to describe the point which perform the transformation function. Some of the services which can be described using this template are transcoding, compression and other generic computing services. This template also inherits the root service description and includes the following additional fields:

- *atAddress*: the atAddress is an octet string and can represent an IP address.
- *Architecture*: the machine architecture is represented using this field.
- *RAM*: the random access memory size is a floating point value and the unit is in Gigabytes.
- *Benchmark*: The processor benchmark is used as a reference to compare different services and it can be one of the widely accepted standard types or a custom benchmark reference.
- *ComputeUnits*: The compute units is expressed in multiples of the benchmark processors and is a floating point value.
- *VirtualCPUCores*: The number of virtual cpu cores is represented using an integer value.

(4) *Preserving service description template:* A service which neither transports or transforms the data needs to have the mandatory ‘atAddress’ field in addition it can have the similar optional field like the transformation service. Some of the services which can be described using this template are storage, content distribution, monitoring and other data preserving services. Other marketplaces can also advertise their marketplace functionality using this template and these marketplaces can run in parallel with the root marketplace competing for service providers and users. This template also inherits the root service description, and includes one additional field:

- *atAddress*: the atAddress is an octet string and can represent an IP address.

3.3 Service composability

One of the main advantages for providing a service description template and enforcing it upon services who advertise in the marketplace is to allow for services to be composed to get an end-to-end service. Four variations of service composability rules are applicable to services described using the above format.

- *Criteria 1* – transport and transform/preserve: a transport service can be composed with a transforming/preserving service provided the toAddress of the transport service matches with the atAddress of the transforming/preserving service. In some cases this is not sufficient and the format of the text/video inside the packet is also considered for composability. If the data inside the packet is intended for a particular type of transformation/ preserving function, then the transport service can carry this information from its origin.
- *Criteria 2* – transport and transport: for two transport services to be composable the toAddress of the preceding service should match with the fromAddress of the succeeding service. In addition either the two transport service types should be identical or the location where the handover is done should be able to convert from one transport service type to the other implicitly.
- *Criteria 3* – transform/preserve and transform/preserve: for two non-transport services to be composable both of them should lie in the same location and the output of the first transforming/ preserving service should act as the input to the succeeding transforming/preserving service. Further the additional considerations mentioned for a transform/preserve service in criteria 1 should also hold.
- *Criteria 4* – transform/preserve and transport: a transforming/ preserving service can be composed with a transport service provided the atAddress of the transforming/preserving service matches with the fromAddress of the transport service. Further the additional considerations mentioned for a transform/preserve service in criteria 1 should also hold.

The rules for matching can be either exact, plug-in or subsume [27]. The criteria for matching the fields of a transform/preserve service is extensive and in this document we have elaborated on two broad fields, i.e. the format of the data and the intended service type for feasible composability. The service description fields for transport, transform and preserve can be simple description in plain text or they can be extended WSDL files. This extends the triple by nesting the attribute values into more triples. So the service composition rules need to consider the nesting of the attribute values when matching services.

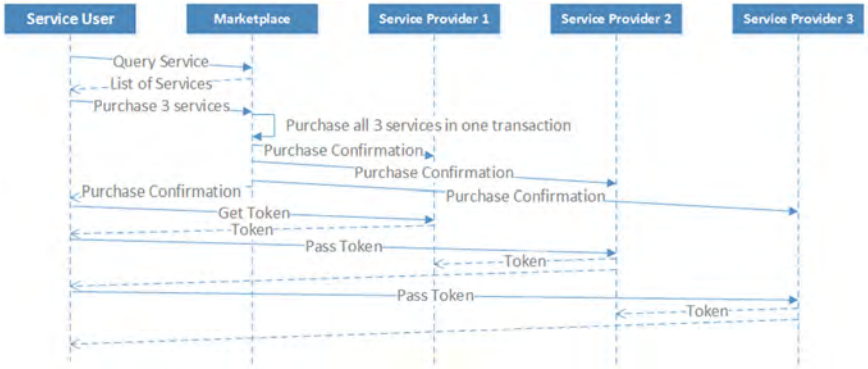

Fig. 2 *Economy plane: composite service*

3.4 Service purchase

The marketplace needs to make provisions for services to be purchased. The root marketplace needs to provide banking/credit service where registered users and providers can transact. It also needs to allow for other trusted entities to provide the banking/credit services. The entities participating in a transaction are free to choose the mode of transaction as long as all agree. Irrespective of which service is used, the marketplace keeps a record of the transaction and notifies the provider(s) and the user with a purchase confirmation ID.

3.5 Service instance tokens

The service provider issues tokens to the user of the service. In case of a simple service the user hands out the token to the provider to receive the service. In case of a composite service it becomes a bit tricky. The composite service can be compared with the Unix pipe command where a set of services are chained by their output and input, so that the output of one service acts as input to the next service. If the advertised service is owned completely by a single provider, the service user can get the token for using the service. In the case where the advertised service is a combination of several services where the individual services are offered by different service providers, the service user needs to purchase the service from the individual providers as part of one single transaction or as part of multiple transactions. The service user receives the token from the first service in the composite service and informs other services in the composite service to pass their tokens to the service provider preceding them.

The service instance participating in a composite service needs to know the token it provided and also the token provided by the service instance ahead of it. A service provider participating in a composed service needs to pass the token and also allow for receiving a token from the succeeding service to enable a composite service. Once the intermediate service provider participating in a composite service receives the token it needs to store the mapping information of its token and the next token for the corresponding composed service instance. This is illustrated with the help of Fig. 2. No service provider/user participating in the composite service has complete knowledge of the token chain or the functionality associated with the token for the succeeding service except the user who made the purchase, so the possibility of misusing a token received as part of a composite service (to gain financially) is remote. This approach also keeps the amount of tokens which a sender needs to cache relatively low.

3.6 Prototype I

One of the ways for realising the economy plane [6] is by modelling the marketplace using the concept of web services. This prototype builds a framework where services can be published and queried. It also defines the vocabulary which defines the semantics of the underlying message structure of the economy plane application programming interfaces (APIs). It implements a basic banking/credit service which is tied to the marketplace allowing for users to purchase services and obtain tokens. It also demonstrates how a composite service is obtained by defining the interactions leading to the provisioning of the individual network services.

(1) *Service differentiation:* The services offered in the marketplace can be broadly classified in two categories.

- *Atomic service*: An atomic service comprises transport, transform and preserve services.
- *Composite service*: A service can be part of a more complex service which is akin to a Unix pipe command where a set of services are chained by their input and output, so that the output of one service acts as input to the next service.

  The name atomic service is used to describe services which do not split up into individual services and can be advertised or purchased individually. Locating the service provider to establish a contract is done via the service advertisement at the marketplace. In case of a composite service the services can be offered by a single service provider, in which case the economic contract can be established for all the services with just one provider. In the scenario where the services are offered by different service providers, an economic contract needs to be established with all the service providers as part of one transaction.

  The atomic service which is part of a composite service can be either at the beginning, end or at the intermediate stage in the composite service. Based on where the atomic service is positioned it needs to provide a generic interface to
- receive the user input and handover the response to the next atomic service or
- receive the response from an atomic service preceding it and handover the response to the next atomic service or
- receive the response from an atomic service preceding it and send the response to the destination.

(2) *Marketplace interaction:* A service provider needs to register with the marketplace before he can advertise a service. The marketplace assumes the service provider can be uniquely identified by the name used while registering. When the service provider publishes a service to the marketplace he can use one of the templates for an atomic service, i.e. transport, transform or preserve template. A service provider has also the option of specifying a composite service in which case he needs to specify the atomic services which form the composite service and the ordering of these atomic services. A service user also needs to register with the marketplace before he can start using it. A service user can query for a service based on either the service provider name, the service description, the unique service id assigned by the marketplace or the type of service. The marketplace returns all the service advertisements which match the query statement. If the advertised service does not need a contract to be established before the service can be used, then the service user can construct SOAP messages based on the interface description mentioned in the WSDL file and start using the service. If there needs to be a contract in place before the service can be used, the advertisement needs to specify the price for purchasing the service. Once the marketplace presents the users with a list of choices matching his search filter, the user chooses one of the services from the list and

Fig. 3 *Search for services in marketplace*

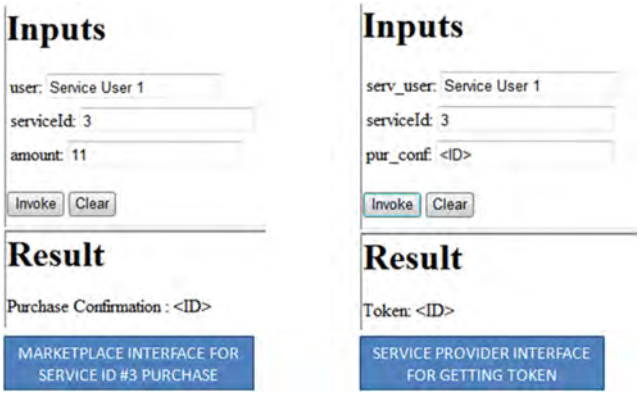

Fig. 4 *Steps in token retrieval*

initiates the purchase through the marketplace. The marketplace provides a minimal banking/credit service which allows for service users to purchase services based on the credit information provided while registering. The marketplace keeps a record of this transaction and informs the owner of the service and the user about the transaction.

(3) *Multiple tokens:* To accomplish the various interactions between the user and the service provider and between two service providers we use the concept of tokens to access and authenticate the use of service. Token is basically an authorisation given to the service user for using the service. Token is issued by the service provider to the user of the service. This is done after receiving proof of purchase from the marketplace. There are two ways we can authorise using a composite service.

- Single token: All the services instances recognise and act on a single token.
- Multiple token: Each service instance is associated with a separate token.

The single token approach requires all the services to agree on a unique token but then requires additional intelligence to be built in the service instance to provide service to a request routed directly from a service immediately preceding it and not from any other service part of the composite service. In case of the multiple token approach we can guarantee that the authorisation for using the service instance is given to only the service immediately preceding it. The service instance participating in a composite service needs to know the token it provided and also the token provided by the service instance ahead of it. The token produced during the economy plane contract establishment is given to the previous service provider instance in the composite service if it exists or else is given to the service user who requested the composite service. The interactions are illustrated in Fig. 2.

(4) *Implementation:* For the prototyping work we use Apache Tomcat which is an open source implementation of the J2EE web container. The marketplace is represented as a web service which is hosted on the Tomcat Server (version 7). We do not use the representation state transfer architecture style but the one which is based on SOAP and WSDL for building the marketplace and the web services which are advertised in the marketplace. There are two specifications for developing web services.

- Java API for XML-based RPC (JAX-RPC)m
- Java API for XML Web Services (JAX-WS) which is the successor of JAX-RPCm

We use Apache Axis which is an implementation of JAX-RPC specification. If we have to migrate to Apache Axis2 or Apache CXF or Spring WS which supports JAX-WS specification, we would not be changing the underlying code for the marketplace. The Log4J logging infrastructure has been integrated with the marketplace to provide provenance. This helps to log all the transactions which are done via the marketplace. We use MySQL a relational database server to store the service advertisements at the marketplace. We use standardised API Java Database Connectivity to interact between the marketplace and the MySQL database. The marketplace server and its clients, i.e. the service providers and service users are written in Java.

(5) *Example of an atomic service:* Here we demonstrate the working of the prototype to obtain and establish an economic contract with an atomic service. Suppose two service providers advertise transcoding service at different locations with slightly different functionality and consideration. Now, if a service user has a specific requirement and is searching for a transcoding service, he will search under transforming services with the specific input and output semantics in the marketplace. The marketplace queries the database and returns the list of service providers and their detailed WSDL files which contains the required interfaces for using the service and establishing an economic contract to obtain proof of purchase. In Fig. 3, we present the GUI which shows the services fetched from the marketplace. In this figure, we have omitted the WSDL fields due to space constraint. The service user can choose from a list of services based on the input and output semantics of the service and the consideration involved. The service user contacts the marketplace to purchase one of these services and is shown in left half of Fig. 4. The marketplace returns a purchase ID as confirmation of the purchase. The user then contacts the service provider and shows the purchase ID and retrieves the token as shown in the right half of Fig. 4. Its mandatory for any paid service advertised through the marketplace to provide an interface similar to the one described in this prototype to retrieve the token, hence it is added as part of the template definitions referred to in the design. The token is then used in the dataplane and is considered as an authorisation to use the service instance.

(6) *Example of a composite service:* Suppose we need transport service to get to the transcoding service. We introduce two more service providers who provide transport service to the two transcoding services defined in the earlier example. The user needs to purchase the transport service in addition to the transcoding service. Our prototype allows for purchasing these services as part of one transaction. After purchasing the service through the marketplace the service user contacts the individual service providers to obtain both the transport and transcoding service to build an end-to-end service and also informs the individual service providers of the immediate service provider preceding them to send the token as they will be using the service on behalf of the user.

4 Semantic network services with clean slate design

In the second design, we develop the semantic language from scratch and we will state the reasons in Section 6. In this design, the principal components and the interaction remain the same as in the earlier design. The major difference is in the flexibility it offers while defining the services. We are no longer confined by the web services language boundary while advertising network services.

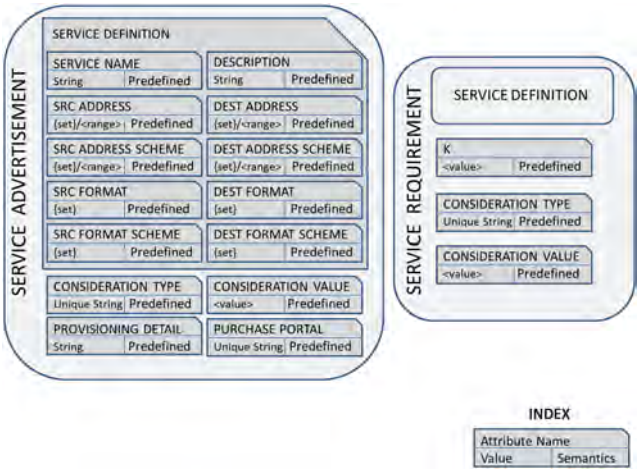

Fig. 5 *Service advertisement and requirement schema*

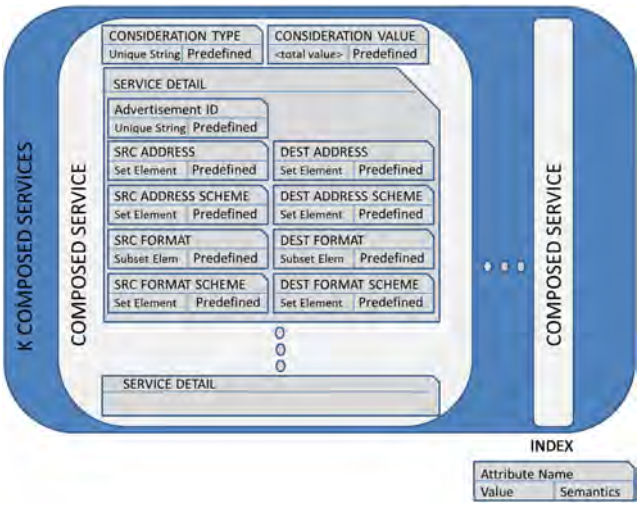

Fig. 6 *Composed service schema*

Since we are starting on a clean slate we can define the network service more concisely yet providing more flexibility at the same time.

4.1 Overview

The network services semantics language is used mainly to define the service advertisement and the user requirement and for defining the interaction between the main entities described earlier in Section 3. The service and requirement abstraction is presented in Fig. 5. The attributes which make up the network services semantic language are described below:

- *Service overview:* The service name and description mentions in brief the service being advertised.
- *Address:* The source and destination addresses along with the addressing schemes are used for specifying the location(s), where the service is being offered. The values for the address fields are specified using a set consisting of host or network (range) addresses.
- *Format:* The source and destination formats along with the format schemes (types) are used for specifying the handling of application data. The values for the format fields are specified using the set syntax.
- *Consideration:* The consideration attribute denotes the cost of purchasing or spending ability for a service advertisement and requirement, respectively.
- *Provisioning:* The provisioning field has information on using the service post purchase.
- *Purchase:* The purchase portal has details of the site, where the consideration amount needs to be paid for purchasing the service.
- *Alternatives:* The number of orchestration services expected by the user is specified in the requirement using *K*.

4.2 Prototype II

We reuse the design philosophy from the first prototype for developing the second prototype. In the second prototype, we do not explicitly categorise a service as being atomic or composite but we leave the interpretation based on the fields which define the service. We move away from WSDL and SOAP by defining new constructs for defining the schema and the messages exchanged across entities. Further, we simplify the token management by doing away with multiple tokens and placing the onus on the composite service to agree on just one token. We refer the reader to our earlier work [24, 25] for a more elaborate discussion on the design and the prototype.

4.2.1 Implementation: Since all the entities are on equal footing we no longer need to differentiate between a marketplace, a service provider and a service user. All these entities can be represented as Unix processes. Depending on the role, these processes perform different tasks based on the messages received. We again refer the reader to our earlier work [24, 25] which provides more insights into the implementation details.

5 Orchestration algorithm

We provide a brief outline of the orchestration algorithm which is an extension to Yen's k-shortest loopless paths algorithm [28] which has been modified to run on a graph where the nodes are sets of addresses and formats while the edges are service advertisements. The modified algorithm is responsible for providing multiple composed (meta) services sorted in non-decreasing order of cost. The composed (meta) services are formed using the services advertised in the marketplace. We use the term 'planner' to describe the model which implements the orchestration algorithm.

5.1 Input and output

The input to the planner is the service requirement from the user and the set of advertisements in the marketplace which is illustrated in Fig. 5. The output of the planner is a list of 'composed service(s)' which is structured as shown in Fig. 6. In a *K-composed service*, the '*K*' stands for the number of 'composed service(s)' which are returned by the planner. The number of 'composed service(s)' returned may be less than or equal to the number of 'composed services(s)' requested by the user. Each composed service consists of the consideration type which is uniform across all the services in the 'composed service' and the accumulated cost. This is followed by the service advertisement instances arranged sequentially in the order the services need to be executed. Each service advertisement instance consists of the service advertisement identifier, which corresponds to a service advertisement in the marketplace, followed by the address and format information, each of which contains one of the set elements from the original service advertisement. The format values in the instance need to take into account wild card formats, which is the universal set containing all formats supported by network services semantic language. This can be achieved by replacing the wild card formats, if present, in a service advertisement with a specific format value and type being requested by the user. So, it is possible for two composed service(s) to have the same advertisement identifiers but what sets them apart is the service instance which is returned by the planner. If the planner cannot find a 'composed service' which matches the user request it returns an empty list of 'composed service(s)'.

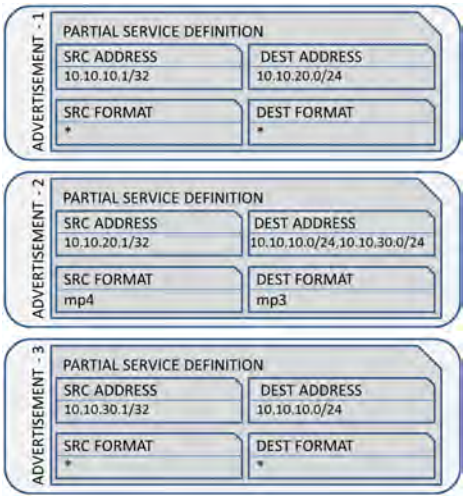

Fig. 7 *Round trip example: service advertisements*

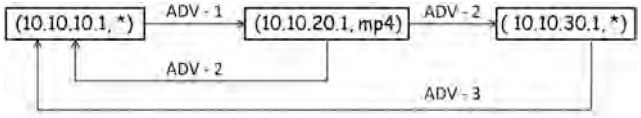

Fig. 8 *Round trip example: network topology*

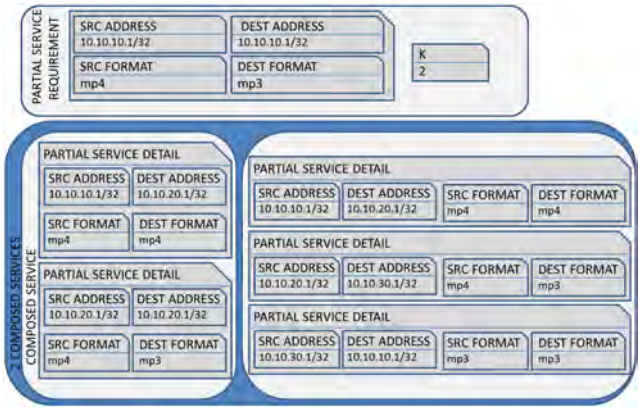

Fig. 9 *Round trip example: input and output*

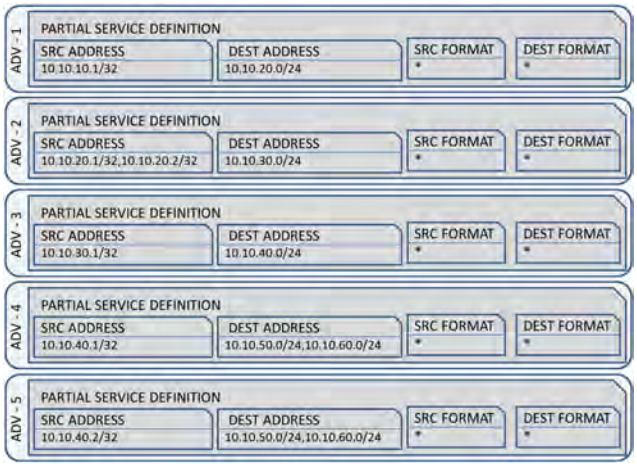

Fig. 10 *Routing example: service advertisements*

5.2 Example

While interpreting a service advertisement/requirement if the 'from' and 'to' addresses are different and the format fields are identical, we interpret this as a path service. If the address fields are identical and the format fields are different, or if both the address fields and the format fields are different then we interpret this as a non-path service.

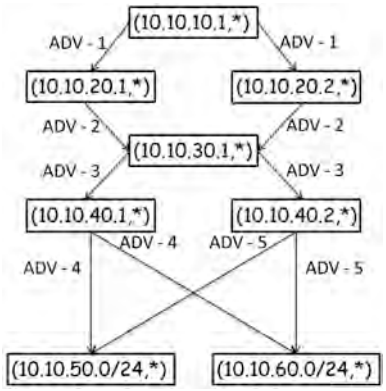

Fig. 11 *Routing example: network topology*

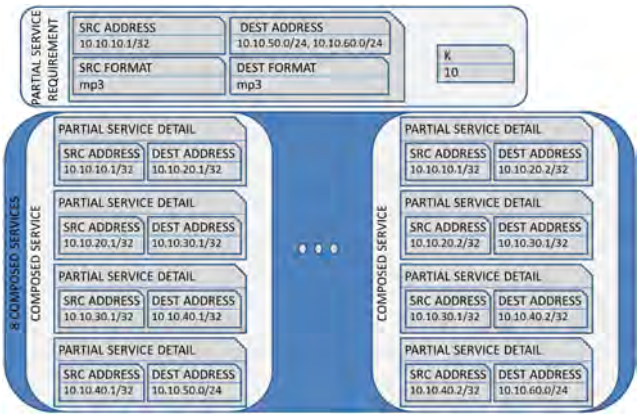

Fig. 12 *Routing example: input and output*

We present two examples; the first example describes constructing a composed non-path service while the second example describes constructing a composed path service. In the below examples, we use '*' to denote the wild card format and ',' to denote the logical 'OR' operator.

Round trip: Fig. 7 shows a set of path and non-path service advertisements for the first example. Fig. 8 shows the path service and non-path service advertisements represented in a network topology diagram with each node represented using the tuple (Addr, Fmt) and each edge represented using the advertisement ID. Fig. 9 shows two composed non-path services being output corresponding to the input.

Routing: Fig. 10 shows a set of path service advertisements for the second example. Fig. 11 shows the network topology diagram and Fig. 12 shows eight composed path services being output corresponding to the input.

6 Comparison

We compare the two design methodologies from the perspective of how effective they are in addressing the challenges of programmability and scalability in 5G heterogeneous networks.

- *Programmability*: Both the design methodologies provide programmability by separating the economy plane from the control and data planes. By allowing the user to choose from alternative network service offerings we provide flexibility to the user to customise the end-to-end service. This could be achieved because we allowed services to be modelled using a common vocabulary/schema which allows for services to be compared.
- *Scalability*: Although both designs allow the infrastructure to be scalable, the second design is better suited for the following reasons:

- In the first design, the financial system software needs to be integrated in the marketplace. This could become a potential bottleneck. In the second design, the financial system software is managed by respective providers and is not maintained by the marketplace on behalf of all the service providers.
- In the first design, the marketplace acts as an intermediary for signing the economic contracts, while in the second design the economic contacts are signed between the user and the provider without involving the marketplace.

Since the first design is based on web services constructs there are existing tools which would aid in deployment and evaluation. Since the second design starts with a clean slate, we have to build all the supporting tools. However, both the designs encourage competition among the service providers which will in turn lead to innovation in the carrier cloud.

7 Concluding remarks

We have presented two design philosophies and discussed two prototypes for advertising network services. We have highlighted the subtleties of the two designs and how they measure up against solving some of the challenges posed by heterogeneous 5G networks namely programmability and scalability while encouraging competition to spur innovation. We have also presented a network service orchestration algorithm and explained its working with the help of an example.

8 Acknowledgment

This work was supported by the NSF under Grant CNS-1111088.

9 References

[1] Syu, Y., Ma, S.-P., Kuo, J.-Y., *et al.*: 'A survey on automated service composition methods and related techniques'. IEEE Ninth Int. Conf. on Services Computing (SCC), 2012, June 2012, pp. 290–297

[2] Schmid, S., Chart, T., Sifalakis, M., *et al.*: 'A highly flexible service composition framework for real-life networks', *Comput. Netw.*, 2006, **50**, (14), pp. 2488–2505, active Networks

[3] Mostarda, L., Marinovic, S., Dulay, N.: 'Distributed orchestration of pervasive services'. 2010 24th IEEE Int. Conf. on Advanced Information Networking and Applications, April 2010

[4] Huang, X., Shanbhag, S., Wolf, T.: 'Automated service composition and routing in networks with data-path services'. Proc. of 19th Int. Conf. on Computer Communications and Networks (ICCCN), 2010, August 2010, pp. 1–8

[5] Shanbhag, S., Wolf, T.: 'Automated composition of data-path functionality in the future internet'. IEEE Network, November 2011

[6] Wolf, T., Griffioen, J., Calvert, K.L., *et al.*: 'Choicenet: toward an economy plane for the internet', *SIGCOMM Comput. Commun. Rev.*, 2014, **44**, (3), pp. 58–65

[7] Ratnasamy, S., Shenker, S., McCanne, S.: 'Towards an evolvable internet architecture', *SIGCOMM Comput. Commun. Rev.*, 2005, **35**, (4), pp. 313–324

[8] Valancius, V., Feamster, N., Johari, R., *et al.*: 'MINT: a market for INternet transit'. Proc. of the 2008 ACM CoNEXT Conf., ser. CoNEXT '08, New York, NY, USA, 2008

[9] Walsh, A.E. (Ed.): '*Uddi, Soap, Wsdl: the web services specification reference book*' (Prentice Hall Professional Technical Reference, 2002)

[10] Curbera, F., Duftler, M., Khalaf, R., *et al.*: 'Unraveling the web services web: an introduction to soap, WSDL, and UDDI', *IEEE Internet Comput.*, 2002, **6**, (2), pp. 86–93

[11] Miller, B.A., Nixon, T., Tai, C., *et al.*: 'Home networking with universal plug and play', *IEEE Commun. Mag.*, 2001, **39**, (12), pp. 104–109

[12] Jeronimo, M., Weast, J.: '*UPnp design by example: a software developer's guide to universal plug and play*' (Intel Press, 2003)

[13] Marples, D., Kriens, P.: 'The open services gateway initiative: an introductory overview', *IEEE Commun. Mag.*, 2001, **39**, (12), pp. 110–114

[14] de Castro Alves, A.: '*OSGi in depth*' (Manning Publications Co., Greenwich, CT, USA, 2011, 1st edn.)

[15] Lakshminarayanan, K., Stoica, I., Shenker, S.: 'Routing as a service'. Technical Report UCB/CSD-04-1327, EECS Department, University of California, Berkeley, 2004

[16] Castro, I., Panda, A., Raghavan, B., *et al.*: 'Route bazaar: automatic interdomain contract negotiation'. 15th Workshop on Hot Topics in Operating Systems (HotOS XV). Kartause Ittingen, Switzerland, USENIX Association, May 2015

[17] Yu, D., Mai, L., Arianfar, S., *et al.*: 'Towards a network marketplace in a cloud'. Proc. of the 8th USENIX Conf. on Hot Topics in Cloud Computing, ser. HotCloud'16, Berkeley, CA, USA, USENIX Association, 2016, pp. 84–89. Available at http://dl.acm.org/citation.cfm?id=3027041.3027055

[18] Xilouris, G., Trouva, E., Lobillo, F., *et al.*: 'T-nova: a marketplace for virtualized network functions'. 2014 European Conf. on Networks and Communications (EuCNC), June 2014, pp. 1–5

[19] Fensel, D., Facca, F.M., Simperl, E., *et al.*: '*Semantic web services*' (Springer Publishing Company, Incorporated, 2011, 1st edn.)

[20] Anderson, C.J., Foster, N., Guha, A., *et al.*: 'Netkat: semantic foundations for networks'. POPL, 2014

[21] Hoyos, L.C., Rothenberg, C.E.: 'Non: network function virtualization ontology towards semantic service implementation'. 2016 8th IEEE Latin-American Conf. on Communications (LATINCOM), November 2016, pp. 1–6

[22] Udechukwu, R., Dutta, R.: 'Service definition semantics for optical services on a choice-based network'. 2015 Int. Conf. on Optical Network Design and Modeling (ONDM), May 2015, pp. 98–103

[23] Chen, X., Wolf, T., Griffioen, J., *et al.*: 'Design of a protocol to enable economic transactions for network services'. 2015 IEEE Int. Conf. on Communications (ICC), June 2015, pp. 5354–5359

[24] Bhat, S., Udechukwu, R., Dutta, R., *et al.*: 'Inception to application: a geni based prototype of an open marketplace for network services'. 2016 IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS), April 2016, pp. 1043–1044

[25] Udechukwu, R., Bhat, S., Dutta, R., *et al.*: 'Language of choice: on embedding choice-related semantics in a realizable protocol'. 2016 IEEE 37th Sarnoff Symp., September 2016, pp. 31–36

[26] Ghodsi, A., Koponen, T., Rajahalme, J., *et al.*: 'Naming in content-oriented architectures'. Proc. of the ACM SIGCOMM Workshop on Information-centric Networking, ser. ICN '11, New York, NY, USA,2011, pp. 1–6. Available at http://doi.acm.org/10.1145/2018584.2018586

[27] Lécué, F., Léger, A.: 'The semantic web – ISWC 2006: 5th international semantic web conference'. ISWC 2006, Athens, GA, USA, 5–9 November 2006. Proc., ch. A Formal Model for Semantic Web Service Composition, Berlin, Heidelberg, 2006, pp. 385–398

[28] Yen, J.Y.: 'Finding the *k* shortest loopless paths in a network', *Manage. Sci.*, 1971, **17**, (11), pp. 712–716