On Service Composition Algorithm for Open Marketplaces of Network Services

Shireesh Bhat, Robinson Udechukwu, Rudra Dutta, George N. Rouskas
Department of Computer Science, North Carolina State University, Raleigh, NC, USA

***Abstract*—Network service composition provided as a service in an Open Marketplace enables users to obtain customized end-to-end composed service(s) using the services advertised by the providers in the marketplace. By providing a semantic language for advertising services and offering choice for the composed service(s) we provide a level playing field for the providers and alternatives for the users to choose from based on their requirement. This is similar to the services offered in the cloud, but without the provider monopoly or the limitation of having to select from limited options.**

I. Introduction

Service composition [1], [2], Service orchestration and choreography [3], Service concatenation or stitching [4], [5], all point to a single objective of bringing together multiple services to form a meta-service, which can be used to satisfy a user request. The difference lies in the way these services are realized in the real world and the ease with which these services can be combined. As we extend the concept of a meta-service to a diverse set of domains: web service, domain independent planners, and network services, it is becoming difficult to separate the methodology involved in the creation of the meta-service and hence the liberal use of these terms seems to be justified. We use the term Service composition to explain the functionality of the Planner which is described in this work.

This work makes several contributions:

- Introduces a semantic language to fully automate the service composition process by describing service functionality in a way that it can be advertised in an Open Marketplace [6] and can be interpreted by the Planner. This is a paradigm shift from the earlier approach [4], which involved reinterpreting the service syntax for integrating the Planner.
- Supports aggregating service advertisements to make it more compact and manageable.
- Presents an extension to Yen's k-shortest loopless paths algorithm [7] 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.

978–1–5386–3873–6/17/$31.00 © 2017 IEEE

A new semantic language design was undertaken as some of the more established [8] and developmental [9] semantic language designs do not provide the flexibility in modeling compact network services. We do not claim our design can be used for modeling all the network services which are now being realized through SDN but we make our design extensible allowing modeling of new services.

Some of the earlier work on Service Composition has dealt with either web service composition [1], finding a shortest cost path in a layered graph [4], [5] or finding a plan based on state space [10]. This work fills in the gap of providing multiple composed services using the network service composition algorithm. The motivation for this work is online travel services, for example Orbitz, Kayak, Expedia, and TripAdvisor. They present a list of itineraries for the customer to choose from consisting of flights, hotels, cars, etc. The billions of dollars in revenue generated by the online travel agencies is a strong indicator of a successful technological and economic model. In this work, we would like to replicate some of the principles of online travel agencies and extend the notion of choice when it comes to selecting network services in an Open Marketplace.

ChoiceNet [6] was designed to provide an infrastructure, which would enable choice in an Open Marketplace to facilitate economic contracts on short term or long term time scales. Some of the related work have used the marketplace for advertising path services [11]–[13], while some have used a marketplace for advertising virtualized networks and theirs functions [14], [15]. 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, etc. In this work, we consider two broad classes of services: path (routing) and data modification service, being advertised in the marketplace. We develop a new algorithm which is inspired by Yen's [7] algorithm to complement the ChoiceNet marketplace by providing alternative service offerings, comprising of a list of end-end paths, a list of data modification services, or a combination of both of them matching the user's requirement.

Following the introduction, we present the semantics language which is responsible for shaping the marketplace in Section II. In Section III we discuss the Planner. In section IV we present numerical results and finally we conclude the paper in section V.

II. Marketplace and Graph Model

The ChoiceNet Marketplace [6] consists of Service advertisements, which are used by the Planner to construct a graph while finding a list of composed services to satisfy the user request. In this section we define the role of semantics language in the workings of a Planner, and also highlight how it encourages competition in the Marketplace, among the various service providers, and how it is advantageous to the service users.

A. ChoiceNet Semantics Language (CSL)

CSL helps define a service advertisement and the user requirement, the essential pieces in the working of the Planner. The extensible CSL schema/vocabulary enables building a consensus between the entities interacting with the Marketplace. This schema/vocabulary may be managed by a regulated and widely accepted authority such as the Internet Assigned Numbers Authority (IANA), which enforces the *vocabulary's* syntax and semantics. The attributes are fully specified using the triple: (attribute name, attribute value, *vocabulary* location), the attribute name and value are defined in the context of the *vocabulary* whose location is specified in the last part of the triple. The service advertisement and requirement are illustrated in Figure 1. The attributes which make up the CSL 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 paid for purchasing the service.
- **Alternatives:** The number of composed services expected by the user is specified in the requirement using K.

The list of attributes mentioned above provides the necessary information for composing a network service. The service advertisement and requirement can be extended to accommodate services which need more attributes to describe the service. The extension would require changes in the composition algorithm for new attributes to be considered while finding alternative composed services matching the user requirement but the underlying design and principle would still remain the same.

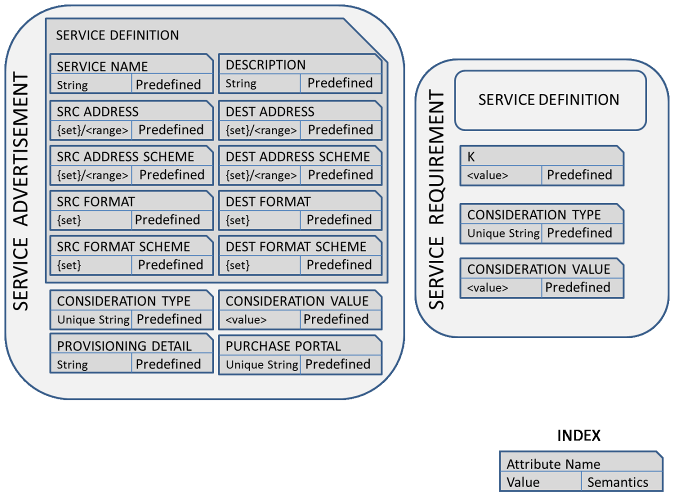

Fig. 1: Service Advertisement and Requirement Schema

B. Functionality

The objective of CSL and the purpose of the fields which makeup the service advertisement and requirement are described below:

1) Layer Abstraction: Since we are dealing with network services it becomes important to state the layer at which a particular service is being provided. We use a layering abstraction which is realized using the address type and format type fields of the service advertisement. In this work we classify network services, which offer routing, as path services, which are realized at layer 2 of the TCP/IP protocol architecture and the remaining services, as being realized at layer 2 or above of the TCP/IP protocol architecture. The layering abstraction enables us to extend this work to realize services at layer 1 of the TCP/IP protocol architecture or to any other architecture which follows the layering model.

2) Address Decode: While interpreting a service advertisement/requirement if the "SRC" and "DEST" 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.

3) Format Decode: The format schema is used to specify the functionality of the data plane service with respect to how it treats the user data. To define path services, we support wild card formats to represent them. While interpreting a service advertisement if the "SRC" and "DEST" formats are different we interpret that this service either modifies/stores/analyzes the data, if they are identical then we interpret that this may be a path service. If the "SRC" and "DEST" formats have wild card formats then we are dealing with a routing service which transports any data, else we are dealing with a routing service which transports data selectively. In this work, we use formats to refer to the data at the application layer of the TCP/IP architecture.

4) Mode of Payment: To allow multiple ways of being compensated for the service, we provide a way to specify the consideration type and the amount of consideration in the service advertisement. The consideration type and the value in the service requirement indicate the cost the user is willing to pay for a composed service. We assume all the services which are part of the "composed service" have the same consideration type and the value is the accumulated sum of the values of the services which make up the "composed service". We can extend this work, to have services with different consideration types being part of the "composed service" but we would need a service which can convert between different consideration types.

5) k-composed services: The user request has the option of specifying the number of "composed services" which should be returned by the Planner, which are below the threshold consideration value specified in the user request. The other fields in the Service advertisement including the Service Name, Service Description, Provisioning Detail, and Purchase Portal do not influence the service composition and are mentioned here for completeness.

6) Pricing Mechanism: The marketplace with inputs from the service providers determines the smallest unit of time for which a service advertised in the marketplace needs to be available. The consideration value represents the cost of using this service for at least this unit of time. Usually a user requirement cannot be accomplished just based on a service advertisement and there are other differentiating factors such as bandwidth for a path service, a continuous duration of time for which the service is needed, the time when the service is needed i.e., instantly or sometime in the future, which can be negotiated by contacting the providers of the services which make up the k-composed services. Advertising services for all combinations is not feasible for both, the providers, and the marketplace administrator. Further, revealing the pricing mechanism/modeling in the marketplace along with the complete network topology would not be something which the providers would do, mainly for two reasons. First, it gives out way too much information which can be exploited by competitors and malicious users. Second, if they want to change their pricing model they have to handle the existing advertisements which are already in the marketplace using the old pricing model. By decoupling the connectivity information from the pricing, we ensure that frequent updates to the graph are reduced.

III. Planner

A. 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 Figure 1. The output of the Planner is a list of "composed service(s)" which is structured as shown in Figure 2. 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 CSL. 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)".

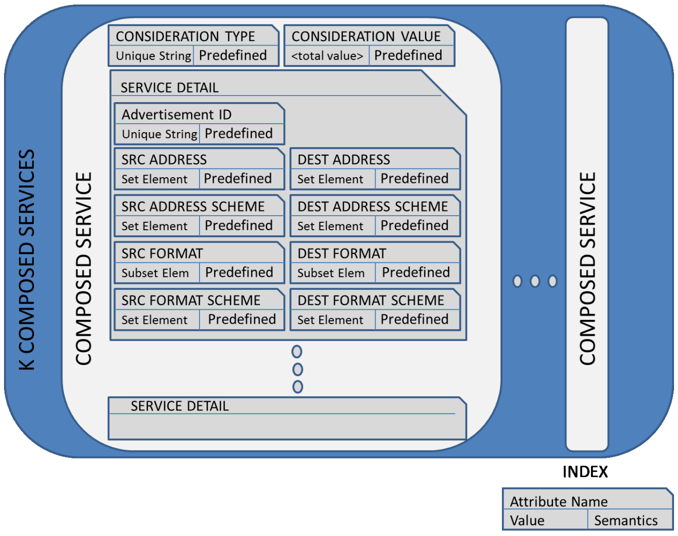

Fig. 2: Composed Service Schema

B. Notations

We denote a service advertisement by $S_{service\_id} = \{S_A, D_A, S_F, D_F\}$, S_A, D_A, S_F, and D_F are the set of source addresses, destination addresses, source formats and destination formats respectively and $service\_id$ is used to uniquely identify a advertisement. We denote the service instance corresponding to a service advertisement by $S_{service\_id,j} = \{s_a, d_a, s_f, d_f\}$, $s_a \in S_A, d_a \in D_A, s_f \in S_F, d_f \in D_F$ and $j = 1, ..., |S_A| \times |D_A| \times |S_F| \times |D_F|$. We denote the requirements specified by the user as $U = \{S_A^U, D_A^U, S_F^U, D_F^U, K, C\}$, S_A^U, D_A^U, S_F^U, and D_F^U are the set of source addresses, destination addresses, source formats and destination formats, and at least one element from the source set i.e. $\{S_A^U, S_F^U\}$ should be present in the first service of a composed service, similarly at least one element from the destination set i.e.

$\{D_A^U, D_F^U\}$ should be present in the last service of the composed service, K represents the number of such composed services required and C represents the threshold cost which shouldn't be exceeded by any of the composed service. Let $H_{service\_id}$ represent the set of service instances corresponding to $service\_id$ which are now part of the already found composed services. Let N denote the set of source service instances i.e. $\{s_a, s_f\}$ which have been deleted as part of the extended Yen's algorithm. Let $T[s_a, s_f]$ denote the cost of reaching the service instance node $\{s_a, s_f\}$ from the virtual source node. Let Q represent the set of destination service instances which have been explored i.e. $\{d_a, d_f\}$. Let $F[]$ be an array of size K consisting of up to K composed services. Let P be a set of potential composed services.

C. Pseudocode

The algorithm assumes that a virtual source node exists which can reach all nodes, represented by the (source address, source format) combination of the user request at zero cost. Similarly, there exists a virtual destination node which can be reached from all nodes, represented by the (destination address, destination format) combination of the user request at zero cost. A variation of Yen's algorithm [7] is used to find the K lowest cost composed service(s) between this virtual source, and destination node and is represented using Algorithm 1. We use the same notations as the original Yen's algorithm [7] to highlight the similarities and differences, the parts of the pseudocode which are almost identical to the Yen's algorithm have been blurred. We have included the pseudo code of the extended Yen's algorithm here to make this work self contained. We now explain the part where our algorithm deviates from the original Yen's algorithm.

- We represent a node by the tuple (address(s), format(s)) and an edge by the tuple (S_A, D_A, S_F, D_F). In the case of a composed service each edge is represented by a service instance tuple (s_a, d_a, s_f, d_f).
- A path in our case is a concatenation of service instances formed from the service advertisements.
- The place where the original Dijkstra's algorithm [16] would have been called is now replaced by a call to the modified Dijkstra's algorithm denoted by $Modified\_Dijkstra$.
- We use $H_{service\_id}$ to keep track of edges which are part of the previously calculated composed service instance(s) and we use N to keep track of the rootNodes which share the same path.

In Algorithm 2 we use a variation of Dijkstra's [16] shortest path algorithm to compute a lowest cost composed service. We summarize the description of the modified Dijkstra's algorithm below:

- In this variation we query the Marketplace repeatedly till we find an end-to-end composed service or we are certain that no such end-to-end composed service exists which satisfies the user constraints.
- While we are extending the list of explored nodes, we decompose a service advertisement into individual service instances if any of the set elements are part of the previous composed service(s). From the perspective of the graph this involves splitting the two nodes which form the end points of a service advertisement into $|S_A| \times |D_A| \times |S_F| \times |D_F|$ nodes.

```
Initialization:
$N = Q = H_{service\_id} = \emptyset$
F[0] = Modified_Dijkstra(U)
for $k = 1, ..., K$ do
    for $i = 0, ..., |F[k-1]| - 1$ do
        $N = Q = H_{service\_id} = \emptyset$
        $spurNode = F[k-1].service\_instance(i)$
        $rootPath = F[k-1].service\_instance(0, i)$
        forall the $c \in F$ do
            if $rootPath == c.service\_instance(0, i)$
            then
                $H_{service\_id} =$
                $H_{service\_id} \cup c.service\ instance(i, i+1)$
            end
        end
        forall the
        $rootPathNode's \in rootPath\ except\ spurNode$
        do
            $N = N \cup rootPathNode$
        end
        $U = \{spurnode_a, D_A^U, spurnode_f, D_F^U, K, C\}$
        $spurPath = Modified\_Dijkstra(U)$
        $totalPath = rootPath + spurPath$
        $P = P \cup \{totalPath\}$
    end
    while $P \neq \emptyset$ do
        $composed\_service = delete\_min\{P\}$
        if $composed\_service \neq duplicatePath$ then
            $F[k] = composed\_service$
            break
        end
    end
end
```

Algorithm 1: Extended Yen's Algorithm

IV. Numerical Results

We now present some preliminary results to evaluate the composition algorithm discussed in the previous section. We have used two topologies, the 14-node NSF network and the 24-node US network [17]. We have used two different models to analyze the results corresponding to this algorithm. In the first model we run the traditional Yen's algorithm [7] which outputs K shortest cost paths on the above topologies and compare it with the service composition algorithm which outputs K lowest cost composed path services. For Yen's algorithm and service composition algorithm, nodes are represented as integers and as a tuple (addresses, format) respectively, and the edges are represented using path service advertisements. In

```
Initialization:
$vSource = \{S_A^U, S_F^U\}$
$vDestination = \{D_A^U, D_F^U\}$
$Q = Q \cup \{vSource\}$
$T[s_a, s_f] = 0 \; \forall \; \{s_a, s_f\} \in \{S_A^U, S_F^U\}$
while $vDestination \notin Q$ do
    //Query the Marketplace
    $T[s_a^w, s_f^w]$ = min$\{T[s_a^v, s_f^v] + C_{vw}\}$
    s.t. $v \in Q, w \notin Q$, $C_{vw}$ = service instance cost,
        $\{s_a^v, d_a^w, s_f^v, d_f^w\} \notin H_{service\_id}$ and $\{s_a^v, s_f^v\} \notin N$
    $Q = Q \cup \{w\}$
    if $T[s_a^w, s_f^w] > C$ then
        return error;//Cost exceeded
    end
end
//construct a composed after tracing back
return (composed service)
```

Algorithm 2: Modified Dijkstra's Algorithm

the second model, we add transformation services to the 24-node topology on top of the existing path services added in the first model. We add stream transcoding services, [HLS_MPEG-2_H.264_AAC, HLS_MPEG-2_H.264_MP3, HDS_FMP4_H.264_AAC, HDS_FMP4_H.264_MP3, MPEG-DASH_FMP4_H.264_AAC, MPEG-DASH_FMP4_H.264_MP3, MSS_FMP4_H.264_AAC, MSS_FMP4_H.264_MP3], at nodes which have nodal degree even and $>$ 3, we add video transcoding services, [MP4_H.264_AAC, MP4_H.264_MP3, FLV_H.264_AAC, FLV_H.264_MP3, MPG_MPEG-2_MP2], and audio transcoding services, [MP3_MP3, MP4_AAC, OGG_VORBIS, OGG_FLAC, OGA_FLAC], at nodes which have nodal degree odd and $>$ 3, the reasoning is, this helps evenly distribute services in the core of the network. The cost of the transcoding is proportional to the nodal degree; the reasoning is, for a core node to be doing something other than routing, there should be an incentive for doing transcoding. In this experiment, a stream transcoding service can convert from any of the eight formats to another, while a video or audio transcoding service can convert from any of the five listed formats to another. We use the second model to compare the impact of service advertisements on the composition algorithm. In the first instance we advertise one transcoding service per node which can convert between any of the supported formats, while in the second instance we advertise one transcoding service per node which can convert from only one supported format to the other. The first instance represents a compact representation of the Marketplace, while the second instance represents a layered representation of the Marketplace. We compare the time it takes to find K composed services on these two instances for the composition algorithm. In the first model, we have 41 compact service advertisements for the 14-node network and 73 compact service advertisements for the 24-node network

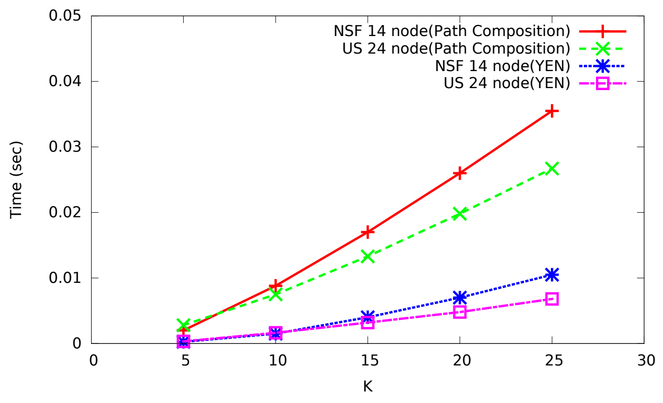

Fig. 3: Running Time Comparison

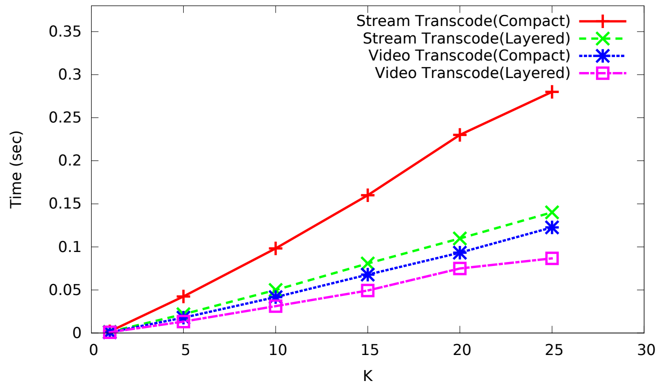

Fig. 4: Running Time for Transcoding Service

representing the path services. In the second model, we have 90 compact service advertisements for the first instance and 593 service advertisements for the second instance, for the 24-node network.

For the first model in the 14 and 24 node networks S_A^U and D_A^U consists of the IP address of one of the 14 and 24 nodes respectively, while S_F^U and D_F^U consists of a wild card which denotes a path service. For the second model, S_F^U and D_F^U consists of a combination of one of the eight streaming formats, or a combination of one of the five video formats, or a combination of one of the five audio formats. C is assigned a very high value.

We have implemented the K lowest cost service composition algorithm in C, and we run the simulation experiments on a 64bit machine with 4 cores and Intel i5 processor @ 2.90GHz. In the figures we present in this section, each data point corresponds to the average of 200 randomly generated problem instances for a 14 node network and an average of 600 randomly generated problem instances for a 24 node network.

Figure 3 plots the running time of the original Yen's algorithm against the composition algorithm for the NSF 14-node and US 24-node networks as a function of K the number of shortest/lowest cost paths/composed services respectively.

We observe that our composition algorithm adds a overhead to the original Yen's algorithm on account of supporting IP subnet addresses and set notation schema. The overhead is not constant and is a function of K as shown in the previous section. We also observe that finding composed services when $K > 10$ takes more time on the US 24-node network than the NSF 14-node network. This observation is made for both the original and extended Yen's algorithms. Although the running time of the K composed services algorithm is proportional to the size of the graph, for small graphs, the nodal degree also influences the running time such that a larger degree allows the algorithm to find a composed service faster. Since the average nodal degree of the NSF 14-node network is 3 while that of the US 24-node network is 3.58, the algorithm runs somewhat faster in the latter.

Figure 4 plots the running time of the composition algorithm for the US 24-node network which now has both path and transcoding services and compares the compact and layered notation for stream transcoding and video transcoding services. We observe that the compact notation although saves space by expanding selectively it definitely increases the running time of the composition algorithm. We conclude that advertising services in the compact notation has its pros and cons. The flip side of using the compact notation is the slight increase in the running time of the algorithm but the advantage is the decreased overhead of managing fewer services which is realized by both the service providers and the marketplace administrator.

V. Concluding Remarks

We have presented a new semantic language and a Planner for advertising network services and constructing composed services respectively in an automated way. The semantic language is extensible allowing for a wide variety of network services to be modeled. The novel algorithm presented in the planner to work with set notation to find multiple composed services can be extended to domains beyond network services.

Acknowledgement

This work was supported by the National Science Foundation under Grant number 1111088.

References

[1] Y. Syu, S.-P. Ma, J.-Y. Kuo, and Y.-Y. FanJiang, "A Survey on Automated Service Composition Methods and Related Techniques," in *Services Computing, 2012 IEEE Ninth International Conference on*, June 2012.

[2] S. Schmid, T. Chart, M. Sifalakis, and A. Scott, "A highly flexible service composition framework for real-life networks," *Computer Networks*, vol. 50, no. 14, pp. 2488 – 2505, 2006, active Networks.

[3] L. Mostarda, S. Marinovic, and N. Dulay, "Distributed orchestration of pervasive services," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, April 2010.

[4] X. Huang, S. Shanbhag, and T. Wolf, "Automated Service Composition and Routing in Networks with Data-Path Services," in *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, Aug 2010, pp. 1–8.

[5] S. Shanbhag and T. Wolf, "Automated composition of data-path functionality in the future internet," *Network, IEEE*, Nov 2011.

[6] T. Wolf, J. Griffioen, K. L. Calvert, R. Dutta, G. N. Rouskas, I. Baldin, and A. Nagurney, "ChoiceNet: Toward an Economy Plane for the Internet," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, Jul. 2014.

[7] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network," *Management Science*, vol. 17, no. 11, pp. pp. 712–716, 1971.

[8] D. Fensel, F. M. Facca, E. Simperl, and I. Toma, *Semantic Web Services*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[9] C. J. Anderson, N. Foster, A. Guha, J. baptiste Jeannin, D. Kozen, and D. Walker, "Netkat: Semantic foundations for networks," in *In POPL*, 2014.

[10] A. Gerevini and I. Serina, "LPG: A Planner Based on Local Search for Planning Graphs with Action Costs." in *AIPS*, M. Ghallab, J. Hertzberg, and P. Traverso, Eds. AAAI, 2002, pp. 13–22.

[11] V. Valancius, N. Feamster, R. Johari, and V. Vazirani, "MINT: A Market for INternet Transit," in *Proceedings of the 2008 ACM CoNEXT Conference*, ser. CoNEXT '08. New York, NY, USA: ACM, 2008.

[12] K. Lakshminarayanan, I. Stoica, and S. Shenker, "Routing as a Service," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-04-1327, 2004.

[13] I. Castro, A. Panda, B. Raghavan, S. Shenker, and S. Gorinsky, "Route Bazaar: Automatic Interdomain Contract Negotiation," in *15th Workshop on Hot Topics in Operating Systems (HotOS XV)*. Kartause Ittingen, Switzerland: USENIX Association, May 2015.

[14] D. Yu, L. Mai, S. Arianfar, R. Fonseca, O. Krieger, and D. Oran, "Towards a network marketplace in a cloud," in *Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 84–89. [Online]. Available: http://dl.acm.org/citation.cfm?id=3027041.3027055

[15] G. Xilouris, E. Trouva, F. Lobillo, J. M. Soares, J. Carapinha, M. J. McGrath, G. Gardikis, P. Paglierani, E. Pallis, L. Zuccaro, Y. Rebahi, and A. Kourtis, "T-nova: A marketplace for virtualized network functions," in *2014 European Conference on Networks and Communications (EuCNC)*, June 2014, pp. 1–5.

[16] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[17] N. Charbonneau and V. M. Vokkarane, "Tabu search meta-heuristic for static manycast routing and wavelength assignment over wavelength-routed optical wdm networks," in *Communications (ICC), 2010 IEEE International Conference on*, May 2010, pp. 1–5.