Adaptive Service Deployment using In-Network Mediation

Abdessalam Elhabbash*, Gordon S. Blair*, Gareth Tyson†, and Yehia Elkhatib*
*MetaLab, School of Computing and Communications, Lancaster University, UK
†EECS, Queen Mary, University of London, UK
{*i.lastname*}@{*lancaster.ac.uk |†qmul.ac.uk}

***Abstract*—Serendipitous peer discovery is important for emerging Internet applications, particularly in dynamic environments (*e.g.*, the IoT, ubiquitous and fog domains) where a large number of resources operate different services in any one locality and resource availability varies unpredictably over time. The current approach is to select services at design time based on offered providers and their reputation. This obviously has its limitations, particularly in terms of scalability and adaptivity, let alone the challenges of crossing vendor and operator divides. This work demonstrates how an application is better able to dynamically adapt to unforeseen environmental changes through *in-network mediation* of service requests. In our model, application developers express their service needs using *intents*. These are mapped to appropriate service providers with explicit consideration of the intermediate network. We design a general architecture and associated algorithms to realise intent formulation and processing for mapping application intents to service providers. Our results demonstrate the feasibility of adopting in-network mediation to enable adaptive application deployment using declarative intents.**

***Index Terms*—intent driven networking, adaptation, antifragility, edge services, fog computing, iot, internet of things**

I. Introduction

Networked environments and their requirements have changed significantly. With the remarkable advancements in integrated sensor-actuator design and low-power WAN technologies, the number of connected devices is rapidly expanding (50 billion by 2020 [1]). Advanced virtualisation and containerisation further expands the number of services that can be hosted on such devices, giving rise to both the Internet of Things (IoT) [2], [3] and fog computing [4], [5] paradigms.

Two crucial challenges arise in such environments: (i) better support for changes in the deployment environment [6]; and (ii) controls (especially privacy) for where and how support roles are executed [7]. Currently, most applications limit their consumers to the predetermined deployment environments they were designed for [8]. Consequently, they are brittle and susceptible to suboptimal operation when the context changes. Under such conditions, centralised (mainly cloud-based) approaches fall short especially in network-constrained conditions [9]. Instead, applications often need a way to be able to adapt at the edge without prior preparation.

We are motivated to answer these challenges by enabling edge applications to discover services in a network-aware fashion in order to dynamically adapt to changes in their environment. Our philosophy is to allow edge application developers to specify their requirements at a high level that will allow post-deployment adaptation to be both requirements- and context-aware.

We realise this vision through employing the Intent Driven Networking (IDN) paradigm [10]. This paradigm operates as an enabler for the interaction, via high-level declarative statements ('*intents*') between the end user applications and network devices. Using intents allows these players to express what they desire/provide from/to the network in a simple and abstract way. For example, one user intent might be to communicate with a particular group of users; another might be to stream a video. A service provider can also issue an intent of providing a service with certain QoS guarantees. The network is then responsible for tending to such requests.

As a consequence, IDN simplifies the development of end-user applications by eliminating the need for including 'try-all-cases' logic – Fig. 1a. This is particularly useful in ensuring lightweight end devices remain simple. For instance, IDN can provide a service-based application with an overall picture of the available network services and can support the selection of required services instead of letting the application try all available services, which can be costly. Instead, the application can formulate and declare its intent – Fig. 1b – which the network then processes, selecting a provider to satisfy it (removing such processing from the application logic). In addition, embedding this service-selection functionality into the network has the benefits of (i) **using ground truth network statistics** to be factored into the decision making, *i.e., Data-driven Network Engineering*; and (ii) allowing the use of **network instrumentation** using SDN and NFV to be better aligned with end-user application requirements.

This vision requires in-network entities able to receive intents and, accordingly, perform optimal selection and dynamic binding. We thus introduce the concept of *in-network*

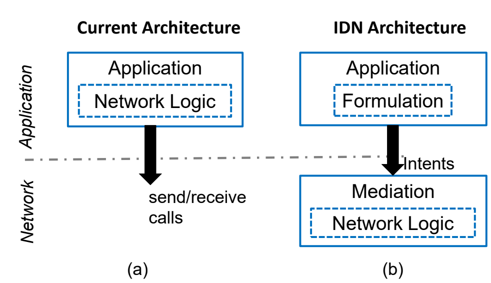

Fig. 1: A high level view of an intent driven network.

mediators: trusted middleboxes that reside in the network to receive intents from users and providers, parse them, compose them, and ultimately satisfy them, unburdening the application of logic to search for services in reaction to changes in the network. The parsing of the intents extracts the intent attributes in preparation for processing. The composition of the intents includes integrating different user intents and/or composing the requirements of a single intent (*e.g.,* the need for a composite service to satisfy the intent). The satisfaction of the intent is carried out by selecting the required services from the network. This selection should take into consideration things like network conditions, pricing, as well as time constraints associated with the intent. For evaluation, we quantitatively assess the mediation effectiveness and limitations.

Contributions: Our contributions are as follows:

1) A **design** of the internal components of the mediator as an independent agent in a given subnetwork.
2) **Two alternative approaches** to realise the selection of services by the mediator in its attempt to satisfy intents. Particularly, we present a scalable genetic algorithm that enables a mediator to serve 100,000s of simultaneous application and service intents.
3) An **evaluation** of mediator performance and limits using controlled experiments, comparing the two approaches.

Distinction from state of the art: There is a range of efforts on adapting service management, and particularly on dynamic and late binding of services. However, these challenges are seldom addressed in light of the application's current deployment environment. Furthermore, there has only been few works trying to integrate awareness of the network into such service selection process *e.g.,* [11], [12]. Numerous other efforts (*e.g.,* [13]–[15]) have focused on optimising placement of VNFs and services based on a network-wide model. Leaving aside the feasibility challenge of acquiring such knowledge and how it changes post-deployment, none of these approaches allows the application to express its runtime requirements.

II. IDN Concept

We briefly overview the intents concept we proposed in [10].

A. Definition

An *intent* is an abstract declaration of what the application desires from the network on behalf of the user. It is a composition of a set of primitive 'verbs', each describing a specific high-level operation. For example, an application intent could be to prioritise imminent VoIP streams with certain remote peers. In response to this, the network carries out the necessary configuration to best serve such an intent.

In more detail, the primitive elements that comprise intents are expressed as tuples of `<verb, object, modifiers, subject>`. A *verb* is an operation that describes the intent based on an ontology. *Object* identifies a service, process or item that is the objective of the verb. *Modifiers* are then used to parameterise this; each *modifier* can be tagged as either 'essential' or 'desirable', indicating prioritisation. *Subject* is an optional identifier of another service/process/item that is to be linked to the defined *object*. Primitive intents expressed are composed using recursive encapsulation to form a full intent.

B. Mediation

The satisfaction of intents is achieved through mediation. Mediators are responsible for 'understanding' intents by parsing them and, if necessary, compiling (*i.e.,* composing multiple intents into a composite one), and realising them. Such realisation involves the mediator taking on any of a range of roles, such as a service broker or a network manager. For instance, in the case of *Construct* intents with the verb *Discover*, the mediator finds the required services to satisfy the intent(s) and returns service access information to the intent issuer. In the case of *Transfer* intent where a content provider might decide to push copies of their contents towards edge points where there is an increase in consumption, the mediator will parse the intent and perform the required content transfer. In a third example utilising *Regulate*, a mediator will translate an intent into network configuration, *e.g.,* an intent to block `ssh` login attempts from a certain address block. The mediators can also satisfy a composition of intents. For instance, the mediator might *Discover* a cache location and *Push* the contents to that cache. These different cases (among others) call for corresponding algorithms to realise the work carried out by the mediators to satisfy the intents.

III. Mediator Design

As mentioned above, intents realisation is achieved through deploying mediators that reside in the network. In this paper, we limit ourselves to using IDN for addressing service provider selection, *i.e.,* the *Construct* verb. This would involve an intent that requests that the network *finds* a *provider* that *offers* a given *service* API. Here, the mediator is responsible for understanding client intents and exploring the network to find providers that are able to satisfy the intents, simplifying the client application logic. However, the success of the mediation role calls for achieving the following properties:

- *Independence:* Mediation should be separated from the detailed logic of the different network users and providers. The declarative intents should be sufficient for the mediator to understand *what* the intent issuer desires from the network.
- *Scalability:* The scalability of the mediation arises due to the increasing number of service providers in addition to the complexity of the intents. The scalability of IDN highly depends on the ability of the mediators to search the space of the provider(s) and to respond to the intent issuer within the specified time constraint.
- *Dependability:* Dependability is an essential property to advocate the different network users' trust and confidence to use IDN. This requires techniques to achieve dependability through load balancing, replication, fault-tolerance and secure communication, among others.

The work of this paper focuses on the first two properties. We assume that one mediator is deployed in a given subnetwork on a fog device or as a virtual network function (VNF). This would be extended to be a part of a hierarchical structure of mediators deployed in parent and sibling subnetworks. Such structure and the corresponding requirements are to be addressed in future work.

In Fig. 2 we sketch our mediator design, which has the following main components:

- *Intent Listeners:* These are interfaces that are used by the providers and clients to submit their intents to the mediator.
- *Client/Provider Intents Queue:* When a client/provider submits an intent and the mediator is busy, the intent is queued. Once the mediator becomes available an intent will be polled (if any) according to a first-come-first-served policy.
- *Provider Intents Parser:* This receives the provider intents from the *Providers Intents Listener* and extracts the intents attributes such as the service type, the service modifier values, and the information needed to access the service (*e.g.,* URL). This is then passed to the *Provider Repository* to be stored.
- *Services Repository:* This repository is simply a database that stores the service attributes to be used by the *Service Selector* component.
- *Client Intents Compiler:* This component fetches client intents from the queue, extracts the intents attributes such as the required service(s) type(s) and the modifier values and types (*i.e.,* essential or desirable). In cases where client intents need to be composed, the compiler forms a composite intent from the corresponding clients intents. Then the *compiler* passes them to the *Service Selector* component.
- *Service Selector:* This component implements the selection algorithm that will use the *Services Repository* to select a service or a composite service that satisfies the intents. In the next section, we realise this component using two different algorithms.

IV. Mediator Algorithms

This section illustrates the applicability of mediation by presenting two alternatives to realising provider selection.

Let us consider an example involving a user application that needs to use a service or a composition of services. With the increased scale of current computational environments, *e.g.,* the cloud, the decision of selecting a service is challenging to the end user. Alternatively, the users can submit their application requirements to the network, *i.e.,* to the mediator, which will select the required services. Having found a provider, the sends back a response as an XML-based metadata describing the type and modifiers of the service and how to access it.

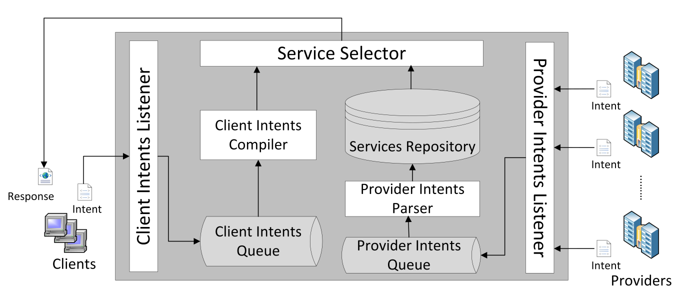

Fig. 2: The internal components of the mediator.

TABLE I: Characteristic service criteria

Criteria	Description	Typical Value Range
Storage Service		
cost	Price per GB ($)	0-10
capacity	Available storage (GB)	10-100
max_file_size	Maximum allowed file size (GB)	5-20
file_versioning	Multiple versions of a file exist at the same time?	True/False
encryption_at_rest	File is encrypted at the service provider side?	True/False
encryption_at_transit	File is encrypted during transmission?	True/False
Proxy Service		
cost	Price per month ($)	5-20
cache_size	Cache size (MB)	200-1000
addressing_type	Traffic redirection method	NATting/forwarding
visibility_type	Visibility of proxy to other network devices	transparent/visible
Intrusion Detection Service		
cost	Price per month ($)	5-20
detection_method	Method of detecting intrusion	signature-based/ specification-based/ anomaly-based
detection_time	Intrusion detection time	real-time/ non real-time
technology_type	Technology layout	network/ host/ wireless /network behaviour analysis/ hybrid
data_type	Type of input data passed to the service	host logs/application logs/wireless network traffic/network_traffic

From then onwards, it is the responsibility of the intent issuing application to parse the response and access the service.

In the following, we consider three types of services that clients may need to use, namely storage services (SS), proxy services (PS), and intrusion detection services (IDS). Each service is represented as a tuple of criteria that characterises the service. Table I summarises such criteria based on relevant literature surveys [16], [17].

Using IDN, the client application will form an intent that reflects the application requirements and submit it to the mediator. The intent can express the need for an elementary or a composite service. Also each intent specifies a deadline for receiving a response from the mediator. Similarly the providers submit their intents advertising the services they provide. Figs. 3-4 show examples of a client intent and a provider intent, respectively. The mediator then compiles the intent to extract the requirements and searches for services that match the client needs. Fig. 5 shows an example mediator response.

The selection process used by the mediator can be realised through different methods. Here, we present two approaches: *Utility-based* and *Genetic-based* selection.

A. Utility-based selection

In this approach, selection is founded on quantifying the extent to which each service satisfies the intent by assigning a utility value to each service modifier. These utilities are then maximised to select the optimal service(s).

```
<CompositeIntent>
 <id>1</id>
 <deadline>323</deadline>
 <intent>
  <id>1.1</id>
  <verb>discover</verb>
  <object>storage</object>
  <modifiers>
   <max_file_size,essential,10.0/>
   <capacity,essential,5.0/>
   <file_versioning,desirable,true/>
   <encryption_at_rest,desirable,true/>
   <encryption_at_transit,desirable,false/>
   <price,essential,10.0/>
  </modifiers>
 </intent>
 <intent>
  <id>1.2</id>
  <verb>discover</verb>
  <object>ids</object>
  <modifiers>
   <price,essential,5.0/>
   <detection_time,desirable,non_real_time/>
   <technology_type,desirable,wireless/>
   <detection_method,desirable,specification/>
   <data_type,desirable,network_traffic/>
  </modifiers>
 </intent>
</CompositeIntent>
```

Fig. 3: An example of a client application intent.

```
 <ProviderIntent>
  <id>1</id>
  <verb>advertise</verb>
  <object>storage</object>
  <modifiers>
   <modifier max_file_size,20.0/>
   <modifier capacity,5.0/>
   <modifier file_versioning,true/>
   <modifier encryption_at_rest,true/>
   <modifier encryption_at_transit,false/>
   <modifier price,10.0/>
  </modifiers>
 </ProviderIntent>
```

Fig. 4: An example of a service provider intent.

In order to assign utilities, we use the functions shown in Table II, which are adapted from a utility model for quantifying volunteer services [18]. The functions assign a maximum utility of 1 to services that satisfy the corresponding intent modifier according to the service provider intent. The service modifiers that do not satisfy the intents receive a utility of 0. For cost modifiers, the utility function assigns higher values to lower costs.

After calculating the utilities, services are sorted accordingly in descending order with higher priorities given to the essential utilities first then desirable ones. Then, services with maximum utilities are selected.

B. Genetic-based selection

The utility-based algorithm ensures optimal service selection, but at the expense of computational complexity as the search space grows. As a more scalable alternative, genetic-based selection simulates the evolution process by the Genetic Algorithm (GA) [19]. It starts from a random solution and evolves it iteratively to generate slightly better ones. Each solution is represented as a set of ‘genomes’ that iteratively undergo the evolution operations of selection, crossover, mutation and fitness evaluation. In each iteration, the fittest solution will survive, and others will be ignored. Evolution stops when a pre-defined criterion is met, *e.g.,* a specific fitness value, a certain number of no-improvement in the fitness value, or a maximum number of iterations.

```
<Response>
 <storageState>found</>
 <idsState>found</>
 <proxyState>found</>
 <storageService>
  <id>153717</id>
  <price>5.0</>
  <capacity>5.0</>
  <max_file_size>10.0</>
  <file_versioning>false</>
  <encryption_at_rest>false</>
  <encryption_at_transit>true</>
  <binding>http://localhost</>
 </storageService>
 <idsService>
  <id>357162</id>
  <price>5.0</price>
  <detection_time>non_real_time</>
  <technology_type>network_behaviour_analysis</>
  <detection_method>specification</>
  <data_type>host_logs</>
  <binding>http://localhost</>
 </idsService>
</Response>
```

Fig. 5: An example of a mediator’s response.

TABLE II: Utility Functions

Utility Function	Used for
$U_{ic} = \begin{cases} 1 + \frac{c_i}{V(m_c)}(\delta - 1), \text{if } c_i \geq V(m_c) \\ 0, \text{ otherwise} \end{cases}$	Cost
$U_{ij} = \begin{cases} 1, \text{if } V(m_j) \geq S_{ij} \\ 0, \text{ otherwise} \end{cases}$	Capacity, max_file_size, cache_size
$U_{ij} = \begin{cases} 0, \text{if } m_j \text{ is 'essential' \&} \\ \quad V(m_j) \neq S_{ij} \\ 1, \text{ otherwise} \end{cases}$	file_versioning, encryption_at_rest, encryption_at_transit, addressing_type, visibility_type, detection_method, detection_time, technology_type, data_type

where U_{ij} is the utility of attribute j of service i, S_{ij} is the value of the attribute j of service i, c_i is the cost of service i, $V(m_c)$ is the value of cost modifier, and $V(m_j)$ is the value of the modifier specified in the intent

In our case, genetic-based selection starts from a random service (or a random composite service, based on the intent) and evaluates its fitness in satisfying the intent. In order to define the fitness functions (shown in Table III), we need also to consider that an intent modifier can be either essential or desirable. For this purpose, we define a variable y_j for each modifier j where the value of y_j equals 0 if the modifier m_j is essential and the value of m_j is not equal to the corresponding service attribute and 1 otherwise. Then in each iteration, a new service will be selected randomly from the available services, resulting in a new solution. The new solution replaces the current one if the fitness of the former is higher than that of the latter. The search process stops after a maximum of 100 iterations or a 20 times of no-improvement in the fitness value; returning the fittest solution. Note that genetic-based selection

TABLE III: Fitness Functions

Service	Fitness Function
Storage	$\prod_{j=1}^{n} y_j \times U_{i,c} \times U_{i,capacity} \times U_{i,file\_size}$
Proxy	$\prod_{j=1}^{h} y_j \times U_{i,c} \times U_{i,cache\_size}$
Intrusion Detection	$\prod_{j=1}^{h} y_j \times U_{i,c}$

may not necessarily find the optimal solution; however, it scales better when trying to satisfy a high number of intents.

V. Quantitative Evaluation

We first use simulation-based experiments to compare the performance of the mediation selection process using both utility- and genetic-based selection algorithms.

Objectives and methods: The objectives of the experiments are to evaluate:

1) *Mediation time* – the end-to-end time from an application submitting an intent until receiving the response.
2) *Percentage of satisfied intents (PSI)* – the number of intents that the mediator found services/composite services for within the specified deadline divided by the total number of submitted intents.

The experiments are conducted on a PC with Intel Pentium D 3.0GHz, 1GB RAM, Linux Ubuntu, Java SE v1.8.0. We vary the number of clients, the number of services, and the depth of intents *i.e.,* the number of different services required to be composed to satisfy an intent. As an example, an intent that expresses the need for only one service (*e.g.,* an SS) will have a depth of 1; an intent that expresses the need for two services (*e.g.,* an SS and a PS) will have a depth of 2, and so on. The intents arrive according to a Poisson process. The deadlines of the intents are generated randomly between 100ms and 300ms. Also, the values of intent and service modifiers are generated randomly according to the values shown in Table I.

Results: Fig. 6 portrays the cost of mediation, which includes the intent and response transmission times, the parsing time, and the service selection time for a varied number of services and varied depth of intents. Mediation time increases proportionally with increase in either of the dimensions of the number of services and the depth of intents. The increase exhibits a linear trend in the case of utility-based selection, which can be acceptable especially in small networks. In comparison, the figure shows the benefit of reducing the mediation time in the case of genetic-based selection, which makes it a more suitable option in large networks. Fig. 6(b) shows also that the mediation time exhibits a constant trend with high number of services. The reason refers to the way the genetic algorithm works. As explained in Section IV the search stopping condition is either reaching a certain number of iterations or a certain number of no-improvements in the fitness functions. The high number of services means that more offerings that satisfy the intent are available and hence the no-improvements threshold is always reached.

Fig. 7 plots the PSI for the same range of client and service populations. PSI, a proxy for efficacy, decreases with the increase in the number of services and with the depth of the intents in the case of utility-based selection. The decrease

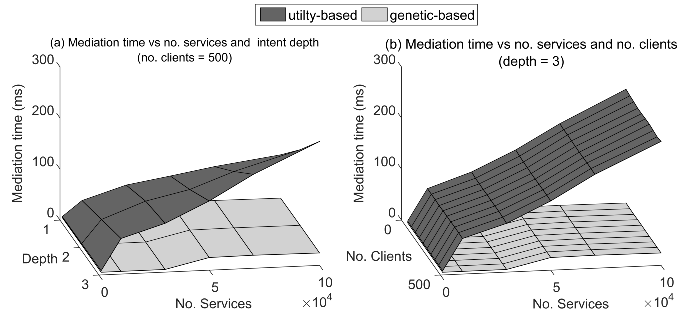

Fig. 6: Mediation time vs. the number of services, and vs. the depth of intents (left) and the number of clients (right).

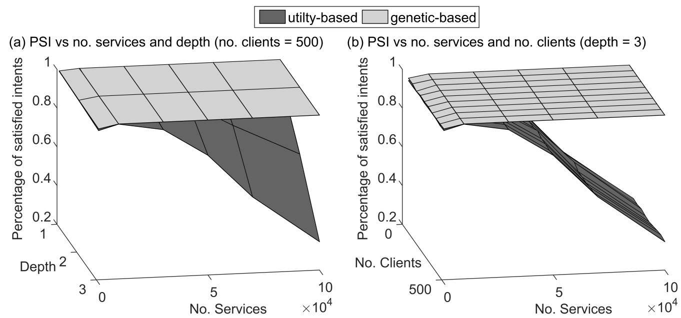

Fig. 7: Percentage of satisfied intents as we increase the number of services, as well as the depth of intents (left) and the number of clients (right).

exhibits a linear trend which can be acceptable in small cases. However, the PSI reaches low values when the number of services is high especially when intent depth equals 3. The PSI is likely to decrease further with higher intent depths. The reason is that the higher the scale of the network the more time needed to rank the utilities and select the services, which results in exceeding deadlines set in intents. Such decrease is significantly reduced in the genetic-based selection compared to the utility-based counterpart where it is almost equal to one. Clearly, the quickness of genetic-based selection enables the meeting of intent deadlines.

VI. Discussion and Future Work

The presented mediation approach illustrates that through high level specification of user intents, the network can be made aware of the application requirements and consequently involved in satisfying those requirements. Developers who need to integrate services in their applications gain many advantages from using mediators. They just need to replace the development of the logic required for finding (and negotiating) providers in a scalable environment with the simple intents formulation. In such a way, much of the development overhead is mitigated and is addressed at the network level. This in turn simplifies and accelerates application development and reduces application logic errors. However, it should be clear that the task of monitoring services and making adaptation decisions when required are not part of the mediators' responsibilities.

To reach our goal of in-network mediation, several challenges need to be addressed. We now outline these.

- *Negotiation*. Mediation is a highly complex task as it is likely that many conflicts will emerge. For example, a user streaming content would want high quality delivery, a publisher would wish to have their content viewed as many times as possible, and an ISP would prefer to only have low-cost (locally available) content viewed. Such potentially conflicting viewpoints will need to ensure thorough negotiation to ensure that all stakeholders are incentivised to cooperate. This requires the development of a negotiation protocol that should be independent of any particular set of verbs and rely on generic notions of utility and priority as derived from intent specification.
- *Mediator interaction*. We demonstrated how a mediator would assist end-user applications operating in its subnetwork. The next step is to enable mediators in different subnetworks to intercommunicate in order to satisfy intents across different domains.
- *Dependability.* The capability of the mediators to satisfy the intents regardless any events that may affect the performance is essential to 'convince' the users to trust and use the IDN. Undoubtedly, introducing mechanisms to achieve dependability (*e.g.,* replication, load-balancing, and encryption) will affect the performance of the mediation, especially in high-scale environments. This makes the desire of achieving both scalability and dependability a substantial challenge that hinders the adoption of IDN.
- *Brokerage and reification.* Marketplace brokerage is an area with potential for reifying spontaneous and strategic intent. Reification is likely to create the need for running in-network services towards the edge. Marketplaces of resources to host such services might benefit from the operation of brokerage and arbitrage agencies. For this, thorough investigation is required to alleviate concerns regarding trust and security. Efforts are also sought for reifying mediation outcomes in the form of adjusting the network control plane or providing information that could be used for late-binding.
- *Realising other intent verbs*. We have only implemented a mechanism for satisfying one type of intents; *i.e.,* the *Construct* intents which are used either to *Advertise* services by the providers or to *Discover* services by the users. More work is needed for other intent types; namely *Transfer*, which allows applications to pull and push content, and *Regulate*, used to express an application's desire to have traffic handled in a certain way in the network.

VII. Related Work

Bringing application awareness to networks has been a long sought after goal of a number of network architectures.

Clear synergies lie between IDN and existing models of service-centricity, namely Service Oriented Architectures (SOA) [20], where systems are composed from a number of loosely coupled services that adhere to shared APIs. The technologies underpinning SOA, namely SOAP and REST, adopt the narrow network API approach. Thus, they continue to suffer from all of the associated problems discussed in Section I.

Information-centric networking (ICN) [21] proposes to convert networks into inherent content delivery systems. Service-centric networking (SCN) [22] extends ICN to apply to services. Both approaches attempt to align the application and the network, which helps to break away from statically binding to specific resources. However, they only partly address the problems we have outlined in the specific cases of accessing content/services: they do not naturally generalise to other scenarios, *e.g.,* those involving switching of networks.

Policy-Based Management (PBM) enables the definition of policies that can be refined into quantifiable network-level targets [23]. PBM is typically constructed around rule-based, goal-driven, or event-driven principles that are mapped to specific operations. Other PBM work is also emerging under the 'network synthesis' subfield, to translate a high-level forwarding policy into confluent network-wide OSPF and BGP rules [24], [25]. Extensions to this philosophy include the RFC on autonomic networking [26], which discusses intents as abstract operational goals, but does not indicate how to implement operations accordingly. Recent works provide solutions to quantify such soft goals using *Network Function Virtualization* (NFV) chains [27] and SDN-based topologies [28].This small but growing body of work is about facilitating malleable network management that is driven by QoS objectives or business constraints. As such, they are geared towards those dealing with wholesale traffic (*i.e.,* network operators). They cannot be used for facilitating application-defined opportunistic service binding at the edge.

Closer to our work are efforts on enabling applications to express their requirements and percolating the down to the underlying network. Declarative languages like Pyretic [29] and Merlin [30] enable the definition of sophisticated network policies through a high-level language. Both, however, focus on issues relating to unifying network administration rather than identifying and addressing application requirements.

VIII. Conclusion

We proposed an approach for realising network consciousness for service selection and application adaptation through employing Intent Driven Networking (IDN). This is achieved through deploying 'mediators', *i.e.,* trusted middleboxes that receive client intents and process them to satisfy their requirements. Apart from enjoying network service levels that better match their intents, applications also benefit from IDN in that some of their selection logic could be pushed to the network. No longer are user applications expected to ship with intricate conditional logic to work around unexpected network behaviour. We put forward and evaluated two approaches to realise the selection process carried out by the mediators. Our results reveal that in-network mediation is feasible to unburden the application from the costly search for providers in large-scale environments.

Acknowledgments

This work has been supported by CHIST-ERA under the DIONASYS Project (UK EPSRC grant EP/M015734/1).

References

[1] D. Evans, "The Internet of Things: How the next evolution of the internet is changing everything," Cisco Internet Business Solutions Group, Tech. Rep. IBSG 0411, Apr 2011. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[2] C. Floerkemeier, M. Langheinrich, E. Fleisch, M. Friedemann, and S. E. Sarma, Eds., *The Internet of Things*. Springer-Verlag Berlin Heidelberg, 2008.

[3] D. Giusto, A. Iera, G. Morabito, and L. Atzori, Eds., *The Internet of Things*. Springer-Verlag New York, 2010.

[4] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct 2014.

[5] Y. Elkhatib, B. F. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir, and E. Rivière, "On using micro-clouds to deliver the fog," *Internet Computing*, vol. 21, no. 2, pp. 8–15, Mar 2017.

[6] I. Yaqoob, E. Ahmed, I. A. T. Hashem, A. I. A. Ahmed, A. Gani, M. Imran, and M. Guizani, "Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges," *IEEE Wireless Communications*, vol. 24, no. 3, pp. 10–16, June 2017.

[7] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[8] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT middleware: A survey on issues and enabling technologies," *Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, Feb 2017.

[9] Y. Elkhatib, "Building cloud applications for challenged networks," in *Embracing Global Computing in Emerging Economies*, ser. Communications in Computer and Information Science, R. Horne, Ed., 2015, vol. 514, pp. 1–10.

[10] Y. Elkhatib, G. Coulson, and G. Tyson, "Charting an intent driven network," in *International Conference on Network and Service Management (CNSM)*, Nov 2017.

[11] S. Y. Jeong, H. G. Jo, and S. J. Kang, "Remote service discovery and binding architecture for soft real-time QoS in indoor location-based service," *Journal of Systems Architecture*, vol. 60, no. 9, pp. 741–756, 2014.

[12] J. Santos, T. Wauters, B. Volckaert, and F. D. Turck, "Resource provisioning for iot application services in smart cities," in *International Conference on Network and Service Management (CNSM)*, Nov 2017.

[13] C. Sandionigi, D. Ardagna, G. Cugola, and C. Ghezzi, "Optimizing service selection and allocation in situational computing applications," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 414–428, July 2013.

[14] L. Qu, C. Assi, K. Shaban, and M. Khabbaz, "Reliability-aware service provisioning in nfv-enabled enterprise datacenter networks," in *International Conference on Network and Service Management (CNSM)*, Oct 2016.

[15] S. Aydi, M. F. Zhani, and Y. Elkhatib, "On improving service chains survivability through efficient backup provisioning," in *International Conference on Network and Service Management (CNSM)*, Nov 2018.

[16] A. Patel, M. Taghavi, K. Bakhtiyari, and J. C. Jnior, "An intrusion detection and prevention system in cloud computing: A systematic review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 25–41, 2013.

[17] G. Tziakouris, M. Zinonos, T. Chothia, and R. Bahsoon, "Asset-centric security-aware service selection," in *IEEE International Congress on Big Data*, June 2016.

[18] A. Elhabbash, R. Bahsoon, P. Tino, and P. R. Lewis, "A utility model for volunteered service composition," in *International Conference on Utility and Cloud Computing (UCC)*. IEEE/ACM, Dec 2014.

[19] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, Jun 1994.

[20] E. Ramollari, D. Dranidis, and A. J. Simons, "A survey of service oriented development methodologies," in *2nd European Young Researchers Workshop on Service Oriented Computing*, vol. 75, 2007.

[21] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2009.

[22] M. J. Freedman, M. Arye, P. Gopalan, S. Y. Ko, E. Nordstrom, J. Rexford, and D. Shue, "Service-centric networking with SCAFFOLD," Princeton University, Tech. Rep. 885-10, Sep 2010.

[23] R. Boutaba and I. Aib, "Policy-based management: A historical perspective," *Journal of Network and Systems Management*, vol. 15, no. 4, pp. 447–480, 2007.

[24] R. Beckett, R. Mahajan, T. Millstein, J. Padhye, and D. Walker, "Don't mind the gap: Bridging network-wide objectives and device-level configurations," in *Annual conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2016.

[25] A. El-Hassany, P. Tsankov, L. Vanbever, and M. T. Vechev, "Network-wide configuration synthesis," in *Conference on Computer-Aided Verification (CAV)*, Jul 2017.

[26] M. H. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. Carpenter, S. Jiang, and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals," RFC 7575 (Informational), Jun 2015.

[27] E. J. Scheid, C. C. Machado, M. Franco, R. L. dos Santos, R. Pfitscher, A. Schaeffer-Filho, and L. Z. Granville, "INSpIRE: Integrated NFV-baSed Intent Refinement Environment," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, May 2017.

[28] Y. Han, J. Li, D. Hoang, J. H. Yoo, and J. W. K. Hong, "An intent-based network virtualization platform for SDN," in *International Conference on Network and Service Management (CNSM)*, Oct 2016.

[29] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN programming with Pyretic," *Technical Report of USENIX*, 2013.

[30] R. Soulé, S. Basu, R. Kleinberg, E. G. Sirer, and N. Foster, "Managing the network with merlin," in *Workshop on Hot Topics in Networks (HotNets)*. ACM, 2013.