Charting an Intent Driven Network

Yehia Elkhatib*, Geoff Coulson*, and Gareth Tyson†
*MetaLab, School of Computing and Communications, Lancaster University, UK
Email: {i.lastname}@lancaster.ac.uk
†EECS, Queen Mary, University of London, UK

***Abstract*—The strong divide between applications and the network control plane is desirable, but keeps the network in the dark regarding the ultimate purpose of applications and, as a result, is unable to optimize for these. An alternative approach is for applications to declare to the network their abstract desires; *e.g.* "I require group multicast", or "I will run within a local domain and am latency sensitive". Such an enriched semantic has the potential to enable the network to better fulfill application intent, while also helping optimize network resource usage across applications. We refer to this approach as *intent driven networking* (IDN). We sketch an incrementally-deployable design to serve as a stepping stone towards a practical realization of IDN within today's Internet.**

I. Introduction

A key principle of the early Internet was the provision of a simple send/receive interface. As the Internet grew with new capabilities added (multicast, streaming, mobility, etc.), we have attempted to continue to live with this very simple interface, although clear downsides are emerging. Essentially, applications operate in the dark with respect to the capabilities and functioning of the underlying network and are therefore obliged to include (potentially complex) logic to handle network related events such as faults, performance fluctuations, service changes, etc. Similarly, network operators do not understand application needs beyond the issuance of seemingly isolated send and receive calls, and are thus unable to conserve resources or optimize performance for applications.

At the heart of the matter is the inability of the network to see the underlying *intent* of the application. Instead, the network only sees a series of *micro-transactions*. One solution is to extend the network API to give direct access to all the individual network elements such as caches, middleboxes, routers, etc. But this is clearly problematic for many reasons: application developers would find it too hard to understand, the API would regularly mutate in line with corresponding changes in the technologies, and it would promote unforeseen interactions between per-technology API elements, with unpredictable and undesirable consequences.

In this paper, we propose *Intent Driven Networking (IDN)* as an alternative approach. It enables the formulation of an application's *intents* as high level statements of its macro-level behaviour, *i.e.* an abstract formulation of what it desires from the network, while remaining agnostic about the means used to satisfy them (protocols, etc.). For example, an intent might be to communicate with a group of users (*e.g.* collaborative document editing); another might be to stream a video while switching between using a laptop on a 802.11 network and a smartphone on a cellular one. Whereas the current Internet sees sets of independent micro-transactions, an intent driven Internet would understand the aims and optimize accordingly.

IDN allows us to simplify application development by removing the need to provide "cover all cases" logic. Instead, user application requirements are fed down to the network, providing flexibility in how different requirements are met without predefined restrictions. For instance, one intent might implicitly ensure the availability of a certain service despite the failure of a remote server; another would ensure a certain level of Quality of Experience (QoE) even if the application is not designed to seek alternative potentially better routes; etc.

In order to attain the IDN vision, we need means by which application intents are *formulated*, *compiled*, and ultimately *reified*, *i.e.* acted upon, (Fig. 1). This paper presents the concept of an intent driven network using both a straw man design (§II) and illustrative examples (§III). We focus particularly on the practical concern of how IDN might be incrementally and partially deployed in the existing Internet without restarting at year zero. We comment on the implications (§IV) of this design, and on difference from related work (§V), before concluding (§VI).

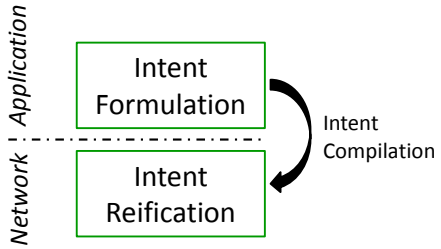

Fig. 1. High level view of an intent driven network.

II. Intents

This section defines what an intent is (§II-A), proposes an approach for the formulation of intents based on compositions of primitive verbs (§II-B), and discusses the mechanics of reifying intents using in-network *mediators* (§II-C).

A. What is an Intent?

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 but high-level operation. For example, an intent to update an Instagram feed might be composed of primitive verbs to reconfigure the application topology (connect to a service and

978-3-901882-98-2

to peers), exchange data (update the content), and uphold a certain QoE level (allocate sufficient network resources). The network, thus, carries out the necessary configuration to best serve such an intent; *e.g.* setting up direct connections between users, and allocating fair shares of router queues considering other network services.

Intent expression is based on the *verb-object-subject* sentence structure used in linguistics, supplemented by *modifiers* as an additional set of words. Primitive intents expressed using such sentences are then composed using recursive encapsulation to form a full intent.

In more detail, the primitive elements that comprise intents are expressed using <*verb, object, modifiers, subject*> tuples. A *verb* is an operation that describes the intent based on an ontology (described next in §II-B). *Object* identifies a service, process or item that is the objective of the verb. *Modifiers* are used to specialize or parameterize this; each modifier can be tagged as either 'essential' or 'desirable', indicating prioritization preference. *Subject* is an (optional) identifier of another service/process/item to be linked to the *object*.

Intents are not limited to only user applications; they extend to other network players (*e.g.* ISPs, cloud service providers, content providers) to express their own intents.

B. Formulating Intents

An intent verb is expressed using one of the basic operations in Fig. 2. The ontology is divided into three operation categories: *Construct*, *Transfer*, and *Regulate*. Each of these categories has a number of sub-operations from which the verb is chosen. Categories are just logical groupings; it is the verbs that signify the primitive intent. This is an introductory ontology not a comprehensive one; modification and expansion is possible through collaboration with the research community.

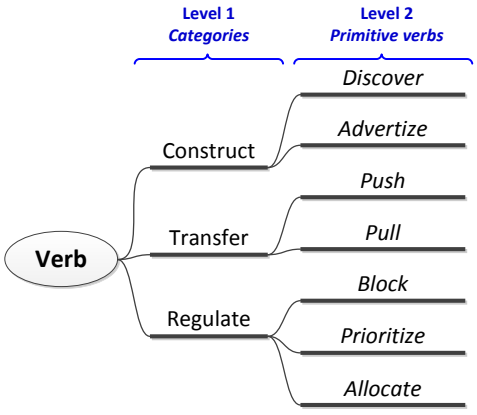

Fig. 2. A basic ontology of primitive verbs.

Construct is used when an application needs to form connections to another application (the *object*) in a peer-to-peer fashion, either locally over a broadcast address or remotely. An example is an intent for a VoIP client to connect to another. *Discover* is issued to look for certain applications, while *Advertize* allows an application to announce a new service that is able to fulfill the intents of other applications. Examples include nodes spawning a caching or load balancing service. Announcing various applications to be discovered or advertized enables an application to dynamically employ external modules without the latter being a component of the native application code. This is of particular use to applications running with scarce resources, *e.g.* a mobile gaming application offloading transcoding processes.

Transfer allow applications to pull and push content (the *object*). A *Transfer* intent is analogous to an Information-centric networking (ICN) abstraction, where the *Push* verb corresponds to a prefix announcement whilst *Pull* corresponds to an interest packet.

Finally, *Regulate* intents capture the desire of an application to have traffic handled in a certain way in the network rather than locally. This is helpful for propagating traffic management logic closer to the source, which facilitates better network management and aggregation of interests. An example is an intent to block ssh login attempts from a certain address block, or to prioritize traffic from a service like `hulu.com`.

C. Reifying Intents

Our conceptual architecture relies on a hierarchical structure of mediators deployed in the network. These are middleboxes that arbitrate between user intents, network and service operator policies, and the current state of the network. We refer to this mediation presence as *Maat* and each of the middleboxes as a *Maat* agent, in reference to the ancient Egyptian concept of conflict resolution to achieve harmonious equilibrium.

User intents are sent on a specific broadcast address to be picked up by a local Maat agent. If a Maat agent is not available as signified by the expiry of a timer since issuing the intent, the application can widen the address scope to seek another agent in the parent subnetwork (and recursively so), or alternatively it could choose to fall back to non-IDN behaviour.

If a Maat agent is available, its job is to "reify" this intent by deploying or activating the required mechanisms (such as an in-network function) or identifying candidate services (a nearby deployment), and consequently sending the relevant information back to the user application to realign itself accordingly. The Maat agent is also required to create a session to keep track of how the intent was met. This is important for auditing mediation efficiency (see §IV-B).

III. Examples

We now further illustrate the formulation of an intent through a set of use cases. We also use these to discuss how the corresponding intent reification would work.

A. Use case #1: Discovery

An important application of serendipitous peer discovery is in Internet of Things (IoT) environments where a large number of hosts would be operating different services in any one locality. Currently, discovery relies on the presence of directory or similar services, an approach with obvious limitations in terms of consistency and scalability.

In such a context an application might signal an intent to build a new overlay structure from a set of suitable nodes

(*e.g.* [1]). In other words, an intent could be used to opportunistically compose an edge system that was not statically constructed at design time. This would work in a fashion similar to ARP; a node would seek other nodes that fit certain criteria on the service(s) they operate, location, communication mode, QoS metrics, etc. The network then propagates the announcement according to criteria laid out by the modifiers.

Consider for instance an actuator in an IoT deployment that wants to find a nearby node capable of running a MapReduce analytics workflow over a collection of sensor data. The intent is formed by composing a *discover* verb (for the MapReduce service and QoS requirements) within an encapsulating *push* one (with additional non-functional requirements) as follows:

```
<push,
  dataset-20170723-1800,
  (net=1.2.3.0/24,ess),
  <discover,
    hadoop,
    (rtt<50ms,des)&(rtt<80ms,ess),
    hadoop-workflow.jar>
>
```

This would be examined by Maat to allow the intent to traverse different networks (if within the specified criteria). As another example, an intent could emanate from a node in a sensor network seeking secure data storage, and use IDN to explore options as diverse as local fixed-power nodes and remote data centers. Such discovery may also include intents formed at different levels, such as the ability to choose which middlebox (proxy, anonymizer, etc.) to go through.

B. Use case #2: Edge Deployment

This example involves different stakeholders: content and service providers, both of which have a lot to gain from a strong and adaptive presence towards the edge of the network.

Consider a content provider that finds an increase in the consumption of certain content (say the 1960s "Batman" series following the death of Adam West) in a particular area (say large metropolises in continental USA). It is in the provider's interest to provide good viewing QoE for its customers and also manage increased load on backend services.

Consequently, the content provider would decide to push copies of the content to cache in different cities. An intent will be expressed as a composition of a verb that discovers suitable caching services (the *object*) in certain locales (the *modifiers*), a verb that pushes content to the discovered caching points, and a final verb to announce the new content once cached. An example of such an intent follows, where `asn` is the AS number signifying a certain customer base.

```
<push,
  Batman,
  (auth=https://www.foxmovies.com/oauth),
  <push,
    831FD96B0.mp4,
    NULL,
    <discover,
      cache,
      (asn=123456,ess),
      NULL>
  >
>
```

Other modifiers could be used to identify target locales at a finer grain. Similarly, a service provider might deploy applications to nodes offering hosting services to balance load at the edge, mitigate flash crowds, or improve user QoE.

IV. Implications

In realizing IDN we do not propose the re-writing of the network stack, but instead to overlay the concept of intents onto the existing Internet architecture. Our straw man architecture has been designed to support **backward compatibility** and **incremental/ partial deployment**. As such, IDN opens up a whole new set of opportunities in research, and also creates some challenges. We now discuss some of these.

A. Opportunities

IDN opens up self-adaptation opportunities for all players in the network space: users, developers and service providers. Users benefit from improved QoE through service provisioning that is dynamic and adaptive to their requirements and contexts. Application developers gain access to higher programming primitives that facilitate fluid application behaviour at runtime, with less reliance on ad hoc means of connecting services and mitigating failures. Service providers are empowered to provision their services in a migration-ready form to be able to provide better QoE for their end users.

IDN also opens a market for hosting services towards the edge. This is beneficial particularly for small and medium sized service providers who cannot afford a highly customized CDN presence like the Googles and Facebooks of the world. Instead, they would be able to bid for edge resource provisioning that in many parts of the world has a wider reach that traditional CDNs [2], [3], [4].

B. Challenges

The most prominent challenges relate to *trust* – specifically: *security* and *efficacy* – and *deployment*.

The **security** challenge could be summarized by the following question: *Could the application trust the network to interfere with its communications, potentially redirecting it to an unintended destination?* This is indeed a major challenge that we recognize. We should first clarify that the in-network Maat agents receive and compile intents, not the subsequent communication which is more likely to contain sensitive information. Based on this, Maat would have information about the desires of the application such as connecting with peers, advertising services or content, regulating network traffic, etc. There is potentially a lot of risk in divulging such information to outside parties. It is noteworthy that such challenges are also being faced by the current Internet architecture.

The **efficacy** concern is summarized with a subtly different question: *Would the application trust the network to not impede or interfere with its performance?* Maat will have significant influence on where the application is redirected to serve its intent. As far as the application is concerned, Maat mediators are black boxes that might have interests conflicting

with those of the application users. They could also be misconfigured, resulting in non-optimal mediation. We perceive this challenge to in fact be an opportunity for *auditing schemes* that ensure efficacy of mediation. For this, we envisage regular reporting of intent, and resulting mediation logs that could be scrutinized to ascertain efficacy. In a multi-mediator market, the mediation score resulting from such auditing mechanisms would engender competition.

Another challenge relates to **deployment** and scalability. The core IDN design lends itself to partial deployment through independent rollout of Maat agents most likely at the edge. There are different ways of doing this, one of which is to augment routers with additional modules, perhaps through NFaaS [5]. Such devices, however, are typically resource constrained and might suffer from performance issues if a large number of services are advertized on their local address spaces. One way of avoiding this is to deploy dedicated Maat agents instead of piggybacking on existing infrastructure. This comes with its own cost, but is feasible on commodity hardware.

V. Related Work

Bringing application awareness to networks has long been sought after [6]. We now review relevant efforts in this domain.

Resource-centric. The REST architectural principle [7] reduces network interactions to a few *verbs* (GET, POST, etc.) transitioning between states using in-request data, making infrastructure scalability and manageability easier. However, REST continues to adopt the "narrow" network API approach and, thus, continues to suffer from associated problems (§I).

Network-centric. ICN solutions are proposed to convert networks into inherent content delivery systems [8], [9]. The service-centric networking (SCN) concept [10], [11], [12], [13], [14] extends ICN principles to apply to services too. Both ICN and SCN attempt to align the application and the network, which helps to break away from statically binding to specific network resources. However, they only partly address the problems we have outlined in the specific cases of accessing content/services: they do not generalize to other scenarios, *e.g.* those involving switching of networks.

Stakeholder-centric. Some work (*e.g.* EONA [15]) has been proposed for application and content providers as well as infrastructure operators to exchange information from their respective control loops in order to improve user experience. However, we are concerned about the viability of this approach. In a world where data is the new oil, we can not imagine such cooperative exchange of information happening between parties that ordinarily have conflicting interests [16].

Policy-centric. Policy-Based Management (PBM) languages and tools are well established for defining high level policies and refining that into actionable and quantifiable network-level targets [17]. PBM is typically constructed around rule-based, goal-driven, or event-driven principles that are mapped to specific operations. Literature includes a host of work on specifying, managing, and refining DiffServ policy hierarchies (*e.g.* [18], [19]) and enabling autonomic networking (*e.g.* [20], [21]). Other PBM work is also emerging under the 'network synthesis' subfield [22], [23], [24].

Recent extensions to this philosophy include the RFC on autonomic networking [25], which defines some high level design goals of automated implementation of abstract operational goals. The RFC does not provide any indication of how to implement or deploy this. A recent paper [26] provides a solution to quantify soft goals associated with such abstract intents, and to use *Network Function Virtualization* (NFV) chains to implement them. The work focuses on middlebox configuration and deployment, but not service discovery.

However, this work is largely on facilitating malleable network management driven by QoS goals or business constraints. Hence, they cater to network operators dealing with wholesale traffic. They cannot, for instance, be used for facilitating application-defined opportunistic service binding at the edge.

Application-centric. Closer to our work are recent efforts on enabling applications to express their requirements and allowing these to percolate down to the underlying network. The Pyretic [27] framework raises the level of abstraction of writing sophisticated network policies. Merlin [28] is a declarative language for specifying global networking policies as a collection of logical predicates to identify traffic subsets and a set of statements indicating the action(s) to be taken on each subset. Both Pyretic and Merlin focus on issues relating to unifying network administration rather than identifying and addressing application requirements.

VI. Summary

We propose *Intent Driven Networking* (IDN) as a concept in which applications and other players such as content providers formulate their communication-related 'intents' in high-level terms that get transformed into network-level reifications that better support the declared intents. We put forward a straw man design that specifies how intents might be formulated (§II-A), involving an ontology of verbs to signify various application desires (§II-B) and a syntax that allows encapsulation of intents. Reification relies on the Maat system to provide in-network mediation between user intents on the one hand, and policies of network and service operators on the other (§II-C). IDN is purposefully designed to be incrementally and partially deployable. Further, we elaborated on our straw man design using different intent examples (§III), and resulting implications on research opportunities and challenges (§IV).

As a consequence, IDN facilitates more fluid development of end user applications and is conducive to better alignment of the network to application needs. No longer are applications expected to ship with intricate and brittle logic to work around unexpected network behaviour.

There is a huge body of future work required to develop IDN into a viable implementation. Therefore, we solicit contributions from the wider systems research community, architects and developers of different disciplines.

Acknowledgment

This work has been partly funded by CHIST-ERA under UK EPSRC grant reference EP/M015734/1.

REFERENCES

[1] G. S. Blair, Y.-D. Bromberg, G. Coulson, Y. Elkhatib, L. Réveillère, H. B. Ribeiro, E. Rivière, and F. Taïani, "Holons: Towards a systematic approach to composing systems of systems," in *Workshop on Adaptive and Reflective Middleware*, 2015, pp. 5:1–5:6.

[2] 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.

[3] R. Fanou, G. Tyson, P. Francois, and A. Sathiaseelan, "Pushing the frontier: Exploring the African web ecosystem," in *WWW*, 2016.

[4] Y. Elkhatib, "Mapping Cross-Cloud Systems: Challenges and Opportunities," in *HotCloud*, 2016.

[5] M. Król and I. Psaras, "NFaaS: Named function as a service," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 134–144.

[6] C. Rotsos, D. King, A. Farshad, J. Bird, L. Fawcett, N. Georgalas, M. Gunkel, K. Shiomoto, A. Wang, A. Mauthe, N. Race, and D. Hutchison, "Network service orchestration standardization: A technology survey," *Computer Standards & Interfaces*, vol. 54, no. Part 4, pp. 203 – 215, 2017, sI: Standardization SDN&NFV.

[7] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[8] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *CoNEXT*, 2009.

[9] G. Tyson, S. Kaune, S. Miles, Y. Elkhatib, A. Mauthe, and A. Taweel, "A trace-driven analysis of caching in content-centric networks," in *Proceedings of the 21st International Conference on Computer Communications and Networks (ICCCN'12)*. IEEE, AUG 2012.

[10] M. J. Freedman, M. Arye, P. Gopalan, S. Y. Ko, E. Nordstrom, J. Rexford, and D. Shue, "Service-centric networking with SCAFFOLD," http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA571380, Princeton University, Tech. Rep. 885-10, September 2010.

[11] T. Braun, A. Mauthe, and V. Siris, "Service-centric networking extensions," in *Symposium on Applied Computing*, 2013, pp. 583–590.

[12] D. Griffin, M. Rio, P. Simoens, P. Smet, F. Vandeputte, L. Vermoesen, D. Bursztynowski, and F. Schamel, "Service oriented networking," in *European Conf. on Networks and Communications*, 2014.

[13] C. Tschudin and M. Sifalakis, "Named functions and cached computations," in *CCNC*, 2014, pp. 851–857.

[14] A. Sathiaseelan, L. Wang, A. Aucinas, G. Tyson, and J. Crowcroft, "SCANDEX: Service centric networking for challenged decentralised networks," in *DIYNetworking*, 2015, pp. 15–20.

[15] J. Jiang, X. Liu, V. Sekar, I. Stoica, and H. Zhang, "Eona: Experience-oriented network architecture," in *HotNets*, 2014, pp. 11:1–11:7.

[16] D. D. Clark, S. Bauer, W. Lehr, K. C. Claffy, A. D. Dhamdhere, B. Huffaker, and M. Luckie, "Measurement and analysis of internet interconnection and congestion," in *Telecomm. Policy Research*, 2014.

[17] 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. [Online]. Available: http://dx.doi.org/10.1007/s10922-007-9083-8

[18] R. Rajan, D. Verma, S. Kamat, E. Felstaine, and S. Herzog, "A policy framework for integrated and differentiated services in the internet," *IEEE Network*, vol. 13, no. 5, pp. 36–41, Sep 1999.

[19] A. K. Bandara, E. C. Lupu, A. Russo, N. Dulay, M. Sloman, P. Flegkas, M. Charalambides, and G. Pavlou, "Policy refinement for ip differentiated services quality of service management," *IEEE Transactions on Network and Service Management*, vol. 3, no. 2, pp. 2–13, April 2006.

[20] B. Jennings, S. V. D. Meer, S. Balasubramaniam, D. Botvich, M. O. Foghlu, W. Donnelly, and J. Strassner, "Towards autonomic management of communications networks," *IEEE Communications Magazine*, vol. 45, no. 10, pp. 112–121, October 2007.

[21] F. Meyer and R. Kroeger, "A framework for autonomic, ontology-based it management," in *11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 78–84.

[22] 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 *Proceedings of the ACM SIGCOMM Conference*. ACM, 2016, pp. 328–341.

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

[24] R. Beckett, R. Mahajan, T. Millstein, J. Padhye, and D. Walker, "Network configuration synthesis with abstract topologies," in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. ACM, June 2017, pp. 437–451.

[25] 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), RFC Editor, Fremont, CA, USA, pp. 1–16, June 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7575.txt

[26] 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 *Proceedings of the 16th IFIP/IEEE International Symposium on Integrated Network Management (IM 2017)*. IEEE, May 2017.

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

[28] R. Soulé, S. Basu, R. Kleinberg, E. G. Sirer, and N. Foster, "Managing the network with merlin," in *HotNets*, 2013, pp. 24:1–24:7.