

COIN  
Internet-Draft  
Intended status: Informational  
Expires: April 3, 2021

H. Singh  
MNK Labs and Consulting  
M-J. Montpetit  
Concordia University  
September 30, 2020

Use of P4 Programs in IETF Specifications  
draft-hsingh-coinrg-p4use-00

Abstract

The IETF specifies several algorithms operating in the data plane of a network node, including liveliness detection, congestion control, network measurement, security, and load balancing. Such algorithms are commonly specified using English or flow charts. As an alternative, this document proposes that P4 programs can be used to specify some data plane algorithms. P4 is a programming language created in 2014 to program the data plane of network nodes such as switches, routers, smartNICs, and generic compute targets.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 3, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Requirements Language . . . . .	2
2. Introduction . . . . .	2
3. Example Use . . . . .	2
4. Summary of P4 . . . . .	4
5. Security Considerations . . . . .	4
6. IANA Considerations . . . . .	4
7. Acknowledgements . . . . .	4
8. References . . . . .	4
8.1. Normative References . . . . .	4
8.2. Informative References . . . . .	5
Authors' Addresses . . . . .	5

## 1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2. Introduction

Research papers for data plane algorithms already use P4 programs to specify algorithms. The MIT Domino compiler [MIT-Domino] which synthesizes hardware logic also outputs a P4 program. For example, the HULA [HULA] data plane congestion control algorithm includes P4 programming logic in the paper.

In another section, this document shows an example for how textual description and flow chart of an algorithm could be augmented or replaced by a P4 program. Lastly, This document presents a summary of the P4 programming language.

## 3. Example Use

An IETF document in [I-D.chen-nvo3-load-banlancing] discusses the flowlet algorithm for load balancing. The draft includes description of algorithm in section 4.1 and a state machine diagram in section 5. Further, if other tables are used in conjunction with the flowlet table, in what sequence does one invoke the tables? Specifying the algorithm of the draft as a P4 program is appropriate. Open source P4 compiler (p4c) already includes a P4 program which implements the flowlet algorithm. See

[https://github.com/p4lang/p4c/blob/master/testdata/p4\\_16\\_samples/flowlet\\_switching-bmv2.p4](https://github.com/p4lang/p4c/blob/master/testdata/p4_16_samples/flowlet_switching-bmv2.p4)

The program uses five tables and the ingress control block shows in what order are the tables invoked. The flowlet algorithm exists in the lookup\_flow\_map and update\_flowlet\_id P4 actions. The program uses P4 registers to maintain state. The IETF draft uses timers. The P4 code uses timestamps since P4 does not support timer yet. A rudimentary timer in a P4 program can use arithmetic to determine whether it is an even/odd minute based on data plane clock used for timestamping packets.

A portion of the P4 program listed above is shown below. The portion shows two P4 actions which implement the flowlet algorithm.

```

action lookup_flowlet_map() {
    hash(meta.ingress_metadata.flowlet_map_index,
        HashAlgorithm.crc16,
        (bit<13>)0, { hdr.ipv4.srcAddr, hdr.ipv4.dstAddr,
        hdr.ipv4.protocol, hdr.tcp.srcPort,
        hdr.tcp.dstPort }, (bit<26>)13);
    flowlet_id.read(meta.ingress_metadata.flowlet_id,
        (bit<32>)meta.ingress_metadata.flowlet_map_index);
    meta.ingress_metadata.flow_ipg =
        (bit<32>)standard_metadata.ingress_global_timestamp;
    flowlet_lasttime.read(
        meta.ingress_metadata.flowlet_lasttime,
        (bit<32>)meta.ingress_metadata.flowlet_map_index);
    meta.ingress_metadata.flow_ipg =
        meta.ingress_metadata.flow_ipg -
        meta.ingress_metadata.flowlet_lasttime;
    flowlet_lasttime.write(
        (bit<32>)meta.ingress_metadata.flowlet_map_index,
        (bit<32>)standard_metadata.ingress_global_timestamp);
}
action update_flowlet_id() {
    meta.ingress_metadata.flowlet_id =
        meta.ingress_metadata.flowlet_id + 16w1;
    flowlet_id.write(
        (bit<32>)meta.ingress_metadata.flowlet_map_index,
        (bit<16>)meta.ingress_metadata.flowlet_id);
}

```

Figure 1: Two P4 actions implement the flowlet algorithm

The ingress control block invokes table lookup using 'table.apply()'.

```
    apply {  
        @atomic {  
            flowlet.apply();  
            if (meta.ingress_metadata.flow_ipg > 32w50000)  
                new_flowlet.apply();  
        }  
        ecmp_group.apply();  
        ecmp_nhop.apply();  
        forward.apply();  
    }
```

Figure 2: Code shows order of invocation for table lookup

#### 4. Summary of P4

First, <https://p4.org> is a great resource to start with. The website includes specifications, pointers to P4 tutorials, p4c, the P4 mailer, P4 Slack Channel, and other details to P4 events, blogs. etc. The README.md file at <https://github.com/p4lan/p4c/> includes details on how to compile a P4 program. P4 started with a P4-14 version in 2014. Since, May 2017, a new version in P4-16 and compiler are available. A list of hardware targets to use for P4 programming is available here: <https://github.com/hesingh/p4-info>

#### 5. Security Considerations

Use IPsec [RFC4301].

#### 6. IANA Considerations

None.

#### 7. Acknowledgements

Thanks (in alphabetical order by first name) to Nick McKeown for encouraging this work.

#### 8. References

##### 8.1. Normative References

[I-D.chen-nvo3-load-banlancing]  
Chen, H., "Load balancing without packet reordering in NVO3", draft-chen-nvo3-load-banlancing-00 (work in progress), October 2014.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.

## 8.2. Informative References

- [HULA] Katta, N., Hira, M., Kim, C., Sivaraman, A., and J. Rexford, "HULA: Scalable Load Balancing Using Programmable Data Planes", March 2016, <<https://dl.acm.org/doi/pdf/10.1145/2890955.2890968>>.
- [MIT-Domino] Sivaraman, A., Cheung, A., Budiu, M., Kim, C., Alizadeh, M., Balakrishnan, H., Varghese, G., McKeown, N., and S. Licking, "Packet Transactions: High-level Programming for Line-Rate Switches", January 2016, <<http://web.mit.edu/domino/>>.

## Authors' Addresses

Hemant Singh  
MNK Labs and Consulting  
7 Caldwell Drive  
Westford, MA 01886  
USA

Phone: +1 978 692 2340  
Email: [hemant@mnkcg.com](mailto:hemant@mnkcg.com)  
URI: <https://mnkcg.com/>

Marie-Jose Montpetit  
Concordia Univeristy  
1455 Boulevard de Maisonneuve O  
Montreal, Quebec 01886  
Canada

Email: [marie@mjmontpetit.com](mailto:marie@mjmontpetit.com)

COIN  
Internet-Draft  
Intended status: Informational  
Expires: August 22, 2021

H. Singh  
MNK Labs and Consulting  
M-J. Montpetit  
Concordia University  
February 18, 2021

Requirements for P4 Program Splitting for Heterogeneous Network Nodes  
draft-hsingh-coinrg-reqs-p4comp-03

Abstract

For distributed computing, the P4 research community has published a paper to show how to split a P4 program into sub-programs which run on heterogeneous network nodes in a network. Examples of nodes are a network switch, a smartNIC, or a host machine. The paper has developed artifacts to split program based on latency, data rate, cost, etc. However, the paper does not mention any requirements. To provide guidance, this document covers requirements for splitting P4 programs for heterogeneous network nodes.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 22, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Requirements Language . . . . .	2
2. Introduction . . . . .	2
3. Terminology and Abbreviations . . . . .	3
4. Requirements . . . . .	4
5. Changes to P4 Compiler to Block Split . . . . .	5
6. Discussion . . . . .	5
7. Security Considerations . . . . .	5
8. IANA Considerations . . . . .	5
9. Acknowledgements . . . . .	5
10. References . . . . .	6
10.1. Normative References . . . . .	6
10.2. Informative References . . . . .	6
Authors' Addresses . . . . .	7

## 1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2. Introduction

The research paper [FLY] covers splitting a P4 program into sub-programs to run the sub-programs on heterogeneous network nodes. There are certain issues to discuss first because some P4 code cannot be split to run elsewhere. There are other issues as well. For brevity, this document uses the terms smartNIC and NIC interchangeably.

In a data center, host machines are connected to a switch. In an Enterprise network, P4 data plane replicates ARP [RFC0826] and IPv6 ND [RFC4861] messages for layer-2 address resolution. If a program split moves ARP and IPv6 code to smartNIC, the hosts should also move to smartNIC. If hosts do not move, the switch resolves layer-2 destinations and messages the NIC with ARP or IPv6 ND table update. But the switch is forwarding traffic at 12 Tbps and for any layer-2 lookup, the switch has to message the NIC which slows down switch forwarding. If hosts also move with ARP and IPv6 ND to the NIC, there are still issues. A NIC with two 100G ports will not be able to support all 25G hosts on a switch with 32 ports. So multiple NICs are used. If a switch is used in bridged mode, there is a single

link-local domain for ARP and IPv6 ND. If the switch is used as a layer-3 switch, then one interface with layer-3 addresses can operate the switch. With multiple NICs, each NIC has its own link-local domain and if configured, a layer-3 interface. So hosts on one NIC go through an additional router to communicate with hosts on another NIC. On the switch, running in bridged mode, the router is not needed.

In a public cloud, Azure resolves layer-2 destination with a central controller and thus the switch does not use any data plane broadcast or IPv6 ND multicast addresses. However, this network faces the same issue mentioned above when multiple NICs are used. Google resolves layer-2 via a proprietary Neighbor Discover protocol [GOOG]. How does Flightplan [FLY] deal with three such disparate networks?

Regarding BGP, if a CLOS network runs BGP, BGP operates between LEAF and SPINE switches. If BGP data plane table splits to a smartNIC, you have to assign an IP address for BGP peer on host CPU. Now the host CPU runs BGP control plane and NIC stores BGP data plane tables. But both Azure and AWS (Amazon Web Services) do not run any SDN or BGP control plane on host because such network activity steals key cycles from host CPU. There is another major problem. Hosts routinely move in the data center to load balance. With a host move, the BGP peer may move to a totally different subnet and break the BGP network.

The punt or divert path of a data plane processes ARP, IPv6 ND, and any routing control messages. Production quality switches (or routers) also run a punt rate-limiter in the data plane so that the switch/router CPU is not inundated. In a heterogeneous network, it is not just how close one punts packets to CPU, but also what else moves with punt path? Certainly the data plane punt rate-limiter also moves.

### 3. Terminology and Abbreviations

CPU - Central Processing Unit.

DPDK - Data Plane Development Kit from Intel.

CLOS - leaf and spine switched network redundant topology.

FPGA - Field Programmable Gate Array.

NIC - Network Interface Card.

npu - Network Processing Unit.



smartNIC - a NIC with processor/FPGA.

TCAM - Ternary Content-Addressable Memory.

VPP - Vector Packet Processing from Cisco.

#### 4. Requirements

The requirements are:

1. If the heterogeneous network includes a switch, the ARP and IPv6 ND data plane P4 code should not be split to run outside the switch.
2. Likewise ARP or IPv6 ND Proxy data plane code should not be split to run outside the switch.
3. BGP table should not be split and move outside the switch. Distributed BGP is a research topic.
4. A switch likely includes TCAM and thus the P4 program may use P4 ternary table match kind. If such a table is moved to another node due to program split, the node the code moves to is important. A FPGA (field-programmable gate array) does not use TCAM and a host machine may not either. The FPGA and host use hash-based table lookup. Depending on the table key size, an appropriate hash is required. Either the splitting tool prompts the user for what hash to use or deduces what hash - user input is desirable. For example, for a 6-tuple IPv4 key, a 128 bit key is used and for the same 6-tuple, the IPv6 key uses 320 bits. Appropriate hashes are required for such keys.
5. Splitting algorithms should not develop High Availability. Network deployments already use dual switches, or CLOS topology for redundancy. BFD [RFC5880] is recommended for use with liveness detection.
6. Any automated tool that splits a P4 program to run on heterogeneous nodes, should provide a manual override. For example, a P4 program is compiled for a switching asic. The compiler raises an error saying code fits in N+2 pipeline stages but the asic has only N stages. In this case, an automated tool will just split the program. However, a manual override allows the programmer to tweak the code manually to fit. With manual tweaking I have been able to fit code in N-1 stages after getting an initial error from compiler for code using N+2 stages. Manual override could kick in if the number of stages used is  $(N + 16\% \times N)$ .

7. The splitting tool should define clearly what is the punt path for P4 code running on a host. The reason is because the host CPU is the data plane, so where is the punted packet to CPU sent? For DPDK, I expect Linux user space to receive punted packets. For VPP, it supports a punt node.

## 5. Changes to P4 Compiler to Block Split

Using P4 Annotations to pass information to p4c (P4 compiler) backend [P4C] to not split certain code is not desirable. This document proposes to change p4c. A new table implementation property called atomic is added to p4c. If this atomic table implementation property is configured for a table in the P4 program, then the table and its actions and any table invocation code block are not split.

## 6. Discussion

The two largest public cloud operators are Amazon AWS and Microsoft Azure [NIC]. Both operators run Software Defined Networking (SDN) in the smartNIC. The reason is running SDN stack in software on the host requires additional CPU cycles. Burning CPUs for SDN services takes away from the processing power available to customer VMs, and increases the overall cost of providing cloud services. Azure uses a FPGA on smartNIC and programs the FPGA in Verilog, not P4. Amazon uses multi-core npu (Graviton uses 64 cores) on smartNIC and does not program Graviton in P4. Both these operators do not use host CPU or network switch for SDN operations. In future, even if both operators program smartNIC in P4, the operators do not have heterogeneous nodes running SDN. Likewise, in future, the switch runs a new SDN feature, e.g. switch caching of popular lookup, then there are heterogeneous nodes to apply Flightplan to.

## 7. Security Considerations

Use IPsec [RFC4301] to secure any control plane communications.

## 8. IANA Considerations

None.

## 9. Acknowledgements

Thanks (in alphabetical order by first name) to Nik Sultana for reviewing this document.

## 10. References

### 10.1. Normative References

- [RFC0826] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <<https://www.rfc-editor.org/info/rfc826>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.

### 10.2. Informative References

- [FLY] Sultana, N., Sonchack, J., Giesen, H., Pedisich, I., Han, Z., Shyamkumar, N., Burad, S., DeHon, A., and B. T. Loo, "Flightplan: Dataplane Disaggregation and Placement for P4 Programs", November 2020, <<https://flightplan.cis.upenn.edu/flightplan.pdf>>.
- [GOOG] Singh, A., "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google Datacenter Network", September 2016, <<https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/7a2ef8424cdc3be32a4cb96bf3e3483eaf0b8949.pdf>>.
- [NIC] Firestone, D., "Azure Accelerated Networking: SmartNICs in the Public Cloud", April 2018, <[https://www.microsoft.com/en-us/research/uploads/prod/2018/03/Azure\\_SmartNIC\\_NSDI\\_2018.pdf](https://www.microsoft.com/en-us/research/uploads/prod/2018/03/Azure_SmartNIC_NSDI_2018.pdf)>.

[P4C] Community, P., "P4\_16 Reference Compiler - Github", May 2018, <<https://github.com/p4lang/p4c>>.

Authors' Addresses

Hemant Singh  
MNK Labs and Consulting  
7 Caldwell Drive  
Westford, MA 01886  
USA

Email: [hemant@mnkcg.com](mailto:hemant@mnkcg.com)  
URI: <https://mnkcg.com/>

Marie-Jose Montpetit  
Concordia Univeristy  
1455 Boulevard de Maisonneuve O  
Montreal, Quebec 01886  
Canada

Email: [marie@mjmontpetit.com](mailto:marie@mjmontpetit.com)

COIN  
INTERNET-DRAFT  
Intended Status: Informational  
Expires: July 26, 2021

D. Trossen  
Huawei  
C. Sarathchandra  
InterDigital Inc.  
M. Boniface  
University of Southampton  
January 26, 2021

In-Network Computing for App-Centric Micro-Services  
draft-sarathchandra-coin-appcentres-04

Abstract

The application-centric deployment of 'Internet' services has increased over the past ten years with many millions of applications providing user-centric services, executed on increasingly more powerful smartphones that are supported by Internet-based cloud services in distributed data centres, the latter mainly provided by large scale players such as Google, Amazon and alike. This draft outlines a vision for evolving those data centres towards executing app-centric micro-services; we dub this evolved data centre as an AppCentre. Complemented with the proliferation of such AppCentres at the edge of the network, they will allow for such micro-services to be distributed across many places of execution, including mobile terminals themselves, while specific micro-service chains equal today's applications in existing smartphones.

We outline the key enabling technologies that needs to be provided for such evolution to be realized, including references to ongoing standardization efforts in key areas.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

#### Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1	Introduction . . . . .	3
2	Terminology . . . . .	4
3	Use Cases . . . . .	4
4	Requirements . . . . .	4
5	Enabling Technologies . . . . .	5
5.1	Application Packaging . . . . .	5
5.2	Service Deployment . . . . .	7
5.3	Compute Inter-Connection at Layer 2 . . . . .	7
5.4	Service Routing . . . . .	8
5.5	Constraint-based Forwarding Decisions . . . . .	9
5.6	Collective Communication . . . . .	10
5.7	State Synchronization . . . . .	11
5.8	Dynamic Contracts . . . . .	11
6	Overview of Relevant Standardization Efforts . . . . .	11
7	Security Considerations . . . . .	12
8	IANA Considerations . . . . .	12
9	Conclusion . . . . .	12
10	References . . . . .	13
10.1	Normative References . . . . .	13
10.2	Informative References . . . . .	13
	Authors' Addresses . . . . .	16

## 1 Introduction

With the increasing dominance of smartphones and application markets, the end-user experiences today have been increasingly centered around the applications and the ecosystems that smartphone platforms create. The experience of the 'Internet' has changed from 'accessing a web site through a web browser' to 'installing and running an application on a smartphone'. This app-centric model has changed the way services are being delivered not only for end-users, but also for business-to-consumer (B2C) and business-to-business (B2B) relationships.

Designing and engineering applications is largely done statically at design time, such that achieving significant performance improvements thereafter has become a challenge (especially, at runtime in response to changing demands and resources). Applications today come prepackaged putting them at disadvantage for improving efficiency due to the monolithic nature of the application packaging. Decomposing application functions into micro-services [MSERVICE1] [MSERVICE2] allows applications to be packaged dynamically at run-time taking varying application requirements and constraints into consideration. Interpreting an application as a chain of micro-services, allows the application structure, functionality, and performance to be adapted dynamically at runtime in consideration of tradeoffs between quality of experience, quality of service and cost.

Interpreting any resource rich networked computing (and storage) capability not just as a pico or micro-data centre, but as an application-centric execution data centre (AppCentre), allows distributed execution of micro-services. Here, the notion of an 'application' constitutes a set of objectives being realized in a combined packaging of micro-services under the governance of the 'application provider'. These micro-services may then be deployed on the most appropriate AppCentre (edge/fog/cloud resources) to satisfy requirements under varying constraints. In addition, the high degree of distribution of application and data partitions, and compute resources offered by the execution environment decentralizes control between multiple cooperating parties (multi-technology, multi-domain, multi-ownership environments). Furthermore, compute resource availability may be volatile, particularly when moving along the spectrum from well-connected cloud resources over edge data centres to user-provided compute resources, such as (mobile) terminals or home-based resources such as NAS and IoT devices.

We believe that the emergence of AppCentres will democratize infrastructure and service provision to anyone with compute resources with the notion of applications providing an element of governing the execution of micro-services. This increased distribution will lead to new forms of application interactions and user experiences based on

cooperative AppCentreS (pico-micro and large cloud data centres), in which applications are being designed, dynamically composed and executed.

## 2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 3 Use Cases

Although our motivation for the 'AppCentre' term stems from the (mobile) application ecosystem, we foresee use cases that are not limited to mobile applications only. Instead, we interpret 'applications' as a governing concept of executing a set of micro-services where the 'application provider' can reach from those realizing mobile applications over novel network applications to emerging infrastructure offerings serving a wide range of applications in a purpose- (and therefore application-)agnostic manner.

Originally being described in more detail in this draft, use cases are now gathered and described in more detail in [COIN-usecases], following a common taxonomy for their description. Specifically, the use cases for immersive devices and infrastructure services have guided the following requirements and technology selection, although this draft also applies to a number of industrial use cases as well. For more detail on those use cases overall, we refer the reader to [COIN-usecases].

## 4 Requirements

The following requirements are derived from the use cases in Section 5 and 6 in [COIN-usecases], serving as a guidance for the following discussions on enabling technologies in Section 5 of this document.

Req 1 - Service Routing: Any app-centric execution environment MUST provide means for routing of service requests between resources in the distributed environment.

Req 2 - Constraint-based Forwarding Decisions: Any app-centric execution environment MUST provide means for dynamically choosing the best possible micro-service sequence (i.e., chaining of micro-services) for a given application experience. Means for discovering suitable micro-service SHOULD be provided.



- Req 3 - Flow Affinity: Any app-centric execution environment MUST provide means for pinning the execution of a specific micro-service to a specific resource instance in the distributed environment.
- Req 4 - Deployment: Any app-centric execution environment SHOULD provide means for packaging micro-services for deployments in distributed networked computing environments. The packaging SHOULD include any constraints regarding the deployment of service instances in specific network locations or compute resources. Such packaging SHOULD conform to existing application deployment models, such as mobile application packaging, TOSCA orchestration templates or tar balls or combinations thereof.
- Req 5 - Synchronization: Any app-centric execution environment MUST provide means for real-time synchronization and consistency of distributed application states.
- Req 6 - Generic Invocation: Any app-centric execution environment MUST provide support for app/micro-service specific invocation protocols.
- Req 7 - Collective Communication: Any app-centric execution environment SHOULD utilize Layer 2 multicast transmission capabilities for responses to concurrent service requests.
- Req 8 - Orchestration: Any app-specific execution environment SHOULD expose means to specify the requirements for the tenant-specific compute fabric being utilized for the app execution. Any app-specific execution environment SHOULD allow for dynamic integration of compute resources into the compute fabric being utilized for the app execution; those resources include, but are not limited to, end user provided resources. Any app-specific execution environment MUST provide means to optimize the inter-connection of compute resources, including those dynamically added and removed during the provisioning of the tenant-specific compute fabric. Any app-specific execution environment MUST provide means for ensuring availability and usage of resources is accounted for.

## 5 Enabling Technologies

We now discuss a number of enabling technologies that address the requirements set out in Section 4.

### 5.1 Application Packaging

Applications often consist of one or more sub-elements (e.g., audio, visual, hepatic elements) which are 'packaged' together, resulting in the final installable software artifact. Conventionally, application developers perform the packaging process at design time, by packaging a set of software components as a (often single) monolithic software package, for satisfying a set of predefined application requirements.

Decomposing micro-services of an application, and then executing them on peer execution points in AppCentreS (e.g., on an app-centric serverless runtime [SRVLESS]) can be done with design-time planning. Micro-service decomposition process involves, defining clear boundaries of the micro-service (e.g., using wrapper classes for handling input/output requests), which could be done by the application developer at design-time (e.g., through Android app packaging by including, as part of the asset directory, a service orchestration template [TOSCA] that describes the decomposed micro-services). Likewise, the peer execution points could be 'known' to the application (e.g., using well-known and fixed peer execution points on AppCentreS) and incorporated with the micro-services by the developer at design-time.

Existing programming frameworks address decomposition and execution of applications centering around other aspects such as concurrency [ERLANG]. For decomposing at runtime, application elements can be profiled using various techniques such as dynamic program analysis or dwarf application benchmarks. The local profiler information can be combined with the profiler information of other devices in the network for improved accuracy. The output of such a profiler process can then be used to identify smaller constituting sub-components of the application in forms of pico-services, their interdependencies and data flow (e.g., using caller/callee information, instruction usage). Due to the complex nature of resulting application structure and therefore its increased overhead, in most cases, it may not be optimal to decompose applications at the pico level. Therefore, one may cluster pico-services into micro-services with common characteristics, enabling a meaningful (e.g., clustering pico-services with same resource dependency) and a performant decomposition of applications. Characteristics of micro-services can be defined as a set of concepts using an ontology language, which can then be used for clustering similar pico-services into micro-services. Micro-services may then be partitioned along their identified borders. Moreover, mechanisms for governance, discovery and offloading can be employed for 'unknown' peer execution points on AppCentreS with distributed loci of control.

Therefore, with this app-centric model, application packaging can be done at runtime by constructing micro-service chains for satisfying

requirements of experiences (e.g., interaction requirements), under varying constraints (e.g., temporal consistency between multiple players within a shared AR/VR world) [SCOMPOSE]. Such packaging includes mechanisms for selecting the best possible micro-services for a given experience at runtime in the multi-technology environment. These run-time packaging operations may continuously discover the 'unknown' and adapt towards an optimal experience. Such decision mechanisms handle the variability, volatility and scarcity within this multi-X framework.

## 5.2 Service Deployment

The service function chains, constituting each individual application, will need deployment mechanisms in a true multi-X (multi-user, multi-infrastructure, multi-domain) environment [SDEPLOY1][SDEPLOY2]. Most importantly, application installation and orchestration processes are married into one, as a set of procedures governed by device owners directly or with delegated authority. However, apart from extending towards multi-X environments, the process also needs to cater for changes in the environment, caused, e.g., by movement of users, new pervasive sensors/actuators, and changes to available infrastructure resources. Methods are needed to deploy service functions as executable code into chosen service execution points. Those methods need to support the various endpoint (e.g., device stacks, COTS stacks, etc.) and service function realizations, e.g., through utilizing existing and emerging virtualization techniques.

A combination of application installation procedure and orchestrated service deployment can be achieved by utilizing the application packaging with integrated service deployment templates described in Section 5.1 such that the application installation procedure on the installing device is being extended to not only install the local application package but also extract the service deployment template for orchestrating with the localized infrastructure, using, for instance, REST APIs for submitting the template to the orchestrator.

The concept of 'intent-based networking' [IB\_CONC] has been the focus of studies in the Network Management RG, allowing for declaratively stating the goals that a network shall meet. In relation to service deployment, intent-based concepts may be useful for the placement of service endpoints in a distributed environment under given service-specific constraints, e.g., on HW constraints for the execution of service endpoints or similar. This could also link into conveying service-specific constraints for the forwarding of information, as discussed in the following Section 5.5.

## 5.3 Compute Inter-Connection at Layer 2

While Layer2 switching technologies have long proliferated in data centre deployments, recent developments have advanced the capabilities for interconnecting distributed computing resources over Layer2 technologies. For instance, the efforts in 3GPP on so-called '5G LAN' (or Vertical LAN) [SA2-5GLAN] allow for establishing a Layer2 bearer between participating compute entities, using a vertical-specific LAN identifier for packet forwarding between the distributed Layer2 entities. Combined with Layer2 technology in data centres as well as office and home networks alike, this enables the deployment of services in vertical (Layer2) networks, interconnecting with other Internet-based services through dedicated service gateways.

Real deployments and realizations will have to show the scalability of this approach but it points into a direction where application or service-specific deployments could potentially 'live' entirely in their own vertical network, interconnecting only based on need (which for many services may not exist). From the application's or service's perspective, the available compute resource pool will look no different from that being realized in a single data centre albeit with the possibility to being highly distributed now among many (e.g., edge) data centres as well as mobile devices.

In such a deployment, it is interesting to study the realization of suitable service routing capabilities, leading us to the next technology area of interest.

#### 5.4 Service Routing

Routing service requests is a key aspect within a combined compute and network infrastructure in order to enable true end-to-end experiences across distributed application execution points provisioned on distant cloud, edge and device-centric resources. Once the micro-services are packaged and deployed in such highly distributed micro-data centres, the routing mechanisms must ensure efficient information exchange between corresponding micro-services, e.g., at the level of service requests, within the multi-technology execution environment.

Routing here becomes a problem of routing micro-service requests, not just packets, as done through IP. This calls for some form of 'flow affinity' that allows for treating several packets as part of a request semantic. This is important, e.g., for mobility (avoiding to send some packets of a larger request to one entity, while other packets are sent to another one, therefore creating incomplete information at both entities as a result). Also, when applying constraints to the forwarding of packets (discussed in more detail in Section 5.6), it is important to apply the actions across the packets

of the request rather than individually.

Another key aspect is that of addressing services. Traditionally, the combination of the Domain Naming Service (DNS) and IP routing has been used for this purpose. However, the advent of virtualization with use cases such as those outlined in Section 3 (such as those on app-specific micro-services on mobile devices) have made it challenging to further rely on the DNS. Apart from the initial delay observed when resolving a service name into a locator for the first time, the long delay in updating DNS entries to 'point' to the right micro-service instances prohibits offloading to dynamically created service instances. If one was to use the DNS, one would be updating the DNS entries at a high rate, caused by the diversity of trigger, e.g., through movement. DNS has not been designed for such frequent update, rendering it useless for such highly dynamic applications. With many edge scenarios in the VR/AR space demanding interactivity and being latency-sensitive, efficient routing will be key to any solution.

Various ongoing work on service request forwarding [RFC8677] with the service function chaining [RFC7665] framework as well as name-based routing [ICN5G][ICN4G][ICNIP] addresses some aspects described above albeit with a focus on HTTP as the main invocation protocol. Extensions will be required to support other invocation protocols, such as GRPC or MPI (for distributed AI use cases). Proposals such as those in [DYN-CAST] suggest extensions to the IP anycast scheme to enable the flexible routing of service requests to one or more service instances. Common to those proposals is the use of a semantic identifier, often a service identifier akin to a URL, in the routing decision within the network.

Efforts existed in the IRTF, in the form of the Routing RG [RRG], to specifically study aspects of routing. The RRG concluded its work in 2014, but its possible revival has been suggested in ongoing discussions on routing evolution [FIPE] as a forum to study semantic-rich routing approaches.

## 5.5 Constraint-based Forwarding Decisions

Allocating the right resources to the right micro-services is a fundamental task when executing micro-services across highly distributed micro-data centres (e.g., resource management in cloud [CLOUDFED]). This is particularly important in the light of volatile resource availability as well as concurrent and highly dynamic resource access. Once the specific set of micro-services for an application has been identified, requirements (e.g., QoS) must be ensured by the execution environment. Therefore, all micro-data centres and the execution environment will need to realize mechanisms

for ensuring the utilization of specific resources within a pool of resources for a specific set of micro-services belonging to one application, while also ensuring integrity of the wider system. Application-layer solution frameworks, such as those developed as part of the Alto WG [ALTO], can be used for utilizing app/service-specific constraints.

In relation to the service routing capability, realized below the application layer and discussed in the previous sub-section, constraints may need to be introduced into the forwarding decisions for service requests. Such constraints will likely go beyond network load and latency, as often applied in scenarios such as load balancing in CDNs and as used in solutions such as [RFC7868]. Instead, those constraints could be app/service-specific and will need a suitable representation for the use within network nodes that are forwarding service requests, as also outlined in [DYN-CAST]. Moreover, individual router decisions (e.g., realized through matching operations such as min/max/equal over a constraint representation) may be coordinated to achieve a distribution of service requests among many service instances, effectively realizing a service scheduling capability in the network, optimized around service-specific constraints, not unlike many existing data centre service switching schemes.

As discussed already in Section 5.2, managing the constraints (for controlling the forwarding behaviour) may be linked into the concepts of intent-based networking [IB\_CONC] to declaratively describe the goals of the forwarding or steering of traffic, while specific signaling protocols will need to be used to convey the actual constraints as well as the operations performed on them in order to fulfil the stated intent (or goals).

## 5.6 Collective Communication

Many micro-service scenarios may exhibit some form of collective communication beyond 'just' unicast communication, therefore requiring support for 1:M, M:1, and M:N communication. It is important to consider here that such collective communication is often extremely short-lived and can even take place at the level of a single request, i.e., a following request may exhibit a different communication pattern, even at least a different receiver group for the same pattern, such as in the case of an interactive game. It is therefore required that solutions for supporting such collective communication must support the spontaneous formation of multicast relations, as observed in those scenarios.

Solutions at Layer 2 have been discussed in [ICNIP], enabling the delivery of service requests over a Layer2 forwarding solution. The

solution in [BIER-MC] utilizes the capabilities introduced by the BIER multicast overlay [BIER] to form such spontaneous multicast relations. Both approaches, however, are limited to the reachability of the respective transport technology, i.e., the Layer 2 or BIER overlay. Solutions over Layer 3 are currently limited to long-lived IP multicast groups or will need to rely on application-level solutions, mapping the group communication to replicated unicast forwarding operations at the network layer, such as done in the message passing interface [MPI], leading to significant inefficiencies through high peak-to-average ratios for the required transport network deployments.

### 5.7 State Synchronization

Given the highly distributed nature of app-centric micro-services, their state exchange and synchronization is a very crucial aspect for ensuring in-application and system wide consistency. Efforts such as those in [GAIA-X] aim at developing solutions for application areas such as distributed storage and data repositories. For this, mechanisms that ensure consistency will ensure that data is synchronized with different spatial, temporal and relational data within a given time period. From the perspective of support through in-network compute capabilities, such as provided through technologies like P4, it is important to consider what system and protocol support is required to utilize such in-network capabilities.

### 5.8 Dynamic Contracts

NOTE: left for future revision

## 6 Overview of Relevant Standardization Efforts

Requirement	Standardization Efforts
1-Service Routing	former Routing RG [RRG], possibly re-instated Dyncast [DYN-CAST] APN BoF [APN]
2-Constraint based Fwd Decision	Dyncast [DYN-CAST] EIGRP [RFC7868] Alto WG [ALTO] Intent-based Networking [IB_CONC]
3-Flow Affinity	Dyncast [DYN-CAST]

4-Deployment	ETSI NFV MANO Intent-based Networking [IB_CONC]
5-Synchroni- zation	GAIA-X [GAIA-X]
6-Generic invocation	Internet Services over ICN [ICNIP]
7-Collective Communication	BIER WG [BIER] Internet Services over ICN (ICNIP) Multicast for HTTP over BIER [BIER-MC]
8-Orchestr.	3GPP 5GLAN [SA2-5GLAN] and ETSI MANO

Figure 1: Mapping of Requirements to Standardization Efforts

## 7 Security Considerations

The use of semantic (or service) identifiers for routing decisions, as mentioned in Section 5.4, requires methods to ensure the privacy and security of the communication through avoiding the exposure of service semantic (which is realized at the application layer) to the network layer, therefore opening up the opportunity for traffic inspection, among other things. The use of cryptographic information, e.g., through self-certifying identifiers, should be investigated to mitigate potential security and privacy risks.

## 8 IANA Considerations

N/A

## 9 Conclusion

This draft positions the evolution of data centres as one of becoming execution centres for the app-centric experiences provided today mainly by smart phones directly. With the proliferation of data centres closer to the end user in the form of edge-based micro data centres, we believe that app-centric experiences will ultimately be executed across those many, highly distributed execution points that this increasingly rich edge environment will provide, such as smart glasses and IoT devices. We have listed and discussed a number of enabling key technologies that address some of the challenges for realizing such AppCentre evolution.

Based on the requirements relevant to those key technologies, derived



from the COIN use cases, we have further provided an evaluation of ongoing and related efforts in the relevant areas of study. We believe that this analysis can be useful for positioning work discussed and pursued in COIN against those ongoing efforts. Furthermore, it may guide those interested in the respective key technologies to create appropriate linkages to those ongoing efforts elsewhere.

## 10 References

### 10.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.
- [RFC7665] Halpern, J., Ed., and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

### 10.2 Informative References

- [MSERVICE1] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. In *Present and Ulterior Software Engineering* (pp. 195–216). Springer, Cham.
- [MSERVICE2] Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software*, 33(3), 42–52.
- [SRVLESS] C. Cicconetti, M. Conti and A. Passarella, "An Architectural Framework for Serverless Edge Computing: Design and Emulation Tools," 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Nicosia, 2018, pp. 48–55. doi: 10.1109/CloudCom2018.2018.00024
- [TOSCA] Topology and Orchestration Specification for Cloud Applications Version 1.0. 25 November 2013. OASIS Standard. <http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html>.
- [ERLANG] Armstrong, Joe, et al. "Concurrent programming in ERLANG." (1993).

- [SCOMPOSE] M. Hirzel, R. Soule, S. Schneider, B. Gedik, and R. Grimm, "A Catalog of Stream Processing Optimizations", ACM Computing Surveys, 46(4):1-34, Mar. 2014
- [SDEPLOY1] Lu, H., Shtern, M., Simmons, B., Smit, M., & Litoiu, M. (2013, June). Pattern-based deployment service for next generation clouds. In 2013 IEEE Ninth World Congress on Services (pp. 464-471). IEEE.
- [SDEPLOY2] Eilam, T., Elder, M., Konstantinou, A. V., & Snible, E. (2011, May). Pattern-based composite application deployment. In 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops (pp. 217-224). IEEE.
- [RFC8677] Trossen, D., Purkayastha, D., Rahman, A., "Name-Based Service Function Forwarder (nSFF) Component within a Service Function Chaining (SFC) Framework", RFC 8677, November 2019.
- [ICN5G] Ravindran, R., Suthar, P., Trossen, D., Wang, C., White, G., "Enabling ICN in 3GPP's 5G NextGen Core Architecture", <https://www.ietf.org/archive/id/draft-irtf-icnrg-5gc-icn-04>, (work in progress), January 2021.
- [ICN4G] Suthar, P., Jangam, Ed., Trossen, D., Ravindran, R., "Native Deployment of ICN in LTE, 4G Mobile Networks", <https://tools.ietf.org/html/draft-irtf-icnrg-icn-lte-4g-08>, (work in progress), January 2021.
- [CLOUDFED] M. Liaqat, V. Chang, A. Gani, S. Hafizah Ab Hamid, M. Toseef, U. Shoaib, R. Liaqat Ali, "Federated cloud resource management: Review and discussion", Elsevier Journal of Network and Computer Applications, 2017.
- [GRPC] High performance open source universal RPC framework, <https://grpc.io/>
- [MPI] A. Vishnu, C. Siegel, J. Daily, "Distributed TensorFlow with MPI", <https://arxiv.org/pdf/1603.02339.pdf>
- [FCDN] M. Al-Naday, M. J. Reed, J. Riihijarvi, D. Trossen, N. Thomos, M. Al-Khalidi, "fCDN: A Flexible and Efficient CDN Infrastructure without DNS Redirection of Content Reflection", <https://arxiv.org/pdf/1803.00876.pdf>
- [DYN-CAST] P. Liu, P. Willis, D. Trossen, "Dynamic-Anycast (Dyncast) Use Cases and Problem Statement",

<https://tools.ietf.org/html/draft-liu-rtgwg-dyncast-ps-usecases-00>, (work in progress), January 2021

[SA2-5GLAN] 3gpp-5glan, "SP-181129, Work Item Description, Vertical\_LAN(SA2), 5GS Enhanced Support of Vertical and LAN Services", 3GPP,  
[http://www.3gpp.org/ftp/tsg\\_sa/TSG\\_SA/Docs/SP-181120.zip](http://www.3gpp.org/ftp/tsg_sa/TSG_SA/Docs/SP-181120.zip)

[COIN-usecases] I. Kunze, K. Wehrle, D. Trossen, "Use Cases for In-Network Computing", <https://tools.ietf.org/html/draft-kunze-coin-industrial-use-cases-04>, (work in progress), January 2021.

[APN] Application-Aware Networking (APN), IETF BoF,  
<https://datatracker.ietf.org/group/apn/about/> (work in progress), January 2021.

[RFC7868] D. Davage et al. , "Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)", RFC 7868, May 2016,  
<https://tools.ietf.org/html/rfc7868>

[ALTO] Application-Layer Traffic Optimization, IETF Working Group, <https://datatracker.ietf.org/wg/alto/about/>, January 2021

[GAIA-X] Gaia-X, "GAIA-X: A Federated Data Infrastructure for Europe", accessed January 2021, <https://www.data-infrastructure.eu/GAIA-X/Navigation/EN/Home/home.html>, January 2021

[ICNIP] D. Trossen, S. Robitzsch, M. Reed, M. Al-Naday, J. Riihijarvi, "Internet Services over ICN in 5G LAN Environments", <https://tools.ietf.org/html/draft-trossen-icnrg-internet-icn-5glan-04>, (work in progress), January 2021

[BIER-MC] D. Trossen, A. Rahman, C. Wang, T. Eckert, "Applicability of BIER Multicast Overlay for Adaptive Streaming Services", <https://tools.ietf.org/html/draft-ietf-bier-multicast-http-response-05>, (work in progress), January 2021

[BIER] Bit Indexed Explicit Replication, IETF Working Group,  
<https://datatracker.ietf.org/wg/bier/about/>, January 2021

[RRG] Routing RG (concluded), IRTF Research Group,  
<https://trac.ietf.org/trac/irtf/wiki/RoutingResearchGroup>, accessed January 2021

[FIPE] Future Internet Protocol Evolution (FIPE) side meeting,  
<https://github.com/FIPE-Study/IETF109-Side-Meeting-FIPE>,  
November 2020

[IB\_CONC] A. Clemm, L. Ciavaglia, L. Granville, J. Tantsura,  
"Intent-Based Networking - Concepts and Definitions",  
<https://datatracker.ietf.org/doc/draft-irtf-nmrg-ibn-concepts-definitions/>, (work in progress), September 2020

#### Authors' Addresses

Dirk Trossen  
Huawei Technologies Duesseldorf GmbH  
Riesstr. 25C  
80992 Munich  
Germany

Email: Dirk.Trossen@Huawei.com

Chathura Sarathchandra  
InterDigital Europe, Ltd.  
64 Great Eastern Street, 1st Floor  
London EC2A 3QR  
United Kingdom

Email: Chathura.Sarathchandra@InterDigital.com

Michael Boniface  
University of Southampton  
University Road  
Southampton SO17 1BJ  
United Kingdom

Email: mjb@it-innovation.soton.ac.uk