Abstract

The application-centric deployment of 'Internet' services has increased over the past ten years with many million 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 of 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 IETF work in some 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
  3.1 Mobile Application Function Offloading ..................... 4
  3.2 Collaborative Gaming ..................................... 6
4 Enabling Technologies ....................................... 6
  4.1 Application Packaging ................................... 6
  4.2 Service Deployment ..................................... 7
  4.3 Service Routing ....................................... 8
  4.4 Service Pinning ....................................... 9
  4.5 State Synchronisation ................................... 9
5 Security Considerations .................................... 9
6 IANA Considerations ....................................... 9
7 Conclusion ................................................. 9
8 References .................................................. 10
  8.1 Normative References ................................... 10
  8.2 Informative References ................................ 10
Authors’ Addresses ........................................ 11
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 website 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 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. 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).

The emergence of AppCentres will democratise infrastructure and service provision to anyone with compute resources, while app-centric computing provides a truly pervasive user experience. Moreover, this will lead to new forms of application interactions and experiences based on cooperative AppCentres (pico-micro and large cloud data centres), in which applications are being designed, (micro-services) 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

3.1 Mobile Application Function Offloading

The App-centric model increases the ubiquity of computing elements available for execution of application functions. Most functions can be categorised into any of three, "receiving", "processing" and "displaying" function groups. Partitioning an application into micro-services allows for denoting the application as a sequence of receiving->processing->displaying, for a flexible composition and distributed execution. Any device may realize one or more of the micro-services of an application and expose them to the execution environment. When the micro-service sequence is executed on a single device, the outcome is what you see today as applications running on mobile devices. However, if any of the three functions are terminated on the device (e.g., for optimising the experience), the execution of the rest of the functions may be moved to other suitable devices which have exposed the corresponding micro-services to the environment. The result of the latter is flexible mobile function offloading, for possible reduction of power consumption (e.g., offloading CPU intensive process functions to a remote server) or for improved device capabilities (e.g., moving display functions to a nearby smart TV).

The above scenario can be exemplified in an immersive gaming application, where a single user plays a game using a VR headset. The headset hosts functions that "display" frames to the user, as well as the functions for VR content processing and frame rendering combining with input data received from sensors in the VR headset. Once this application is partitioned into micro-services and deployed in an app-centric execution environment, only the "display" micro-service is left in the headset, while the compute intensive real-time VR content processing micro-services can be offloaded to a nearby resource rich home PC, for a better execution (faster and possibly higher resolution generation).

Figure 1 shows one realisation of the above scenario, where a 'DPR app' running on a mobile device (containing the partitioned Display(D), Process(P) and Receive(R) micro services) over an SDN network. The packaged applications are made available through a localised 'playstore server'. The application installation is realized as a 'service deployment' process (Section 4.2.), combining the local app installation with a distributed micro-service.
deployment (and orchestration) on most suitable AppCentres ('processing server').

![Diagram of application function offloading example]

Figure 1: Application Function Offloading Example

Such localized deployment could, for instance, be provided by a visiting site, such as a hotel or a theme park. Once the 'processing' micro-service is terminated on the mobile device, the 'service routing' (SR) elements in the network (Section 4.3.) route requests to the previously deployed 'processing' micro-service running on the 'processing server' AppCentre over an existing SDN network.

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. Any solution SHOULD also provide methods for choosing and/or deploying the best possible instances of micro-services in the app-centric execution environment, for service routing, and for pinning the resources to the corresponding micro-services.
3.2 Collaborative Gaming

There has been a recent shift from applications that provide single-user experiences, such as the ones described in the previous section to collaborative/cooperative experiences such as multi-user gaming and mixed/virtual reality. The latter leads to increasing amounts of interaction where input (e.g., gesture, gaze, touch, movement) and output (e.g., visual display, sound, and actuation) needs to be processed within strict timing constraints and synchronized to ensure temporal and spatial consistency with local and distant users. App-centric design allows functions with high data and process coupling to be modularised, deployed and executed, such that the subset of micro-services is cooperatively executed towards optimising multi-user experiences.

The same example in previous section can be envisaged from a multi-player gaming scenario. Here the micro-services that need to be executed cooperatively are executed in a localised and synchronised manner for player coordination and synchronizing interaction and state between collaborating players.

Any app-centric execution environment MUST provide means for real-time synchronization and consistency of distributed application states.

4 Enabling Technologies

4.1 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-X 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.

4.2 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 to deploy service functions as executable code into chosen service execution points, supporting the various endpoint realizations (e.g., device stacks, COTS stacks, etc.), and service function endpoint realization through utilizing existing and emerging virtualisation 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 4.1 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.

4.3 Service Routing

Service routing within a combined compute and network infrastructure that will enable true end-to-end experiences across distributed application execution points provisioned on distant cloud, edge and device-centric resources (e.g., using ICN/name-based routing methods), is a key aspect of app-centric micro-service execution. Once the micro-services are packaged and deployed in such highly distributed micro-data centres, the routing mechanisms will ensure efficient information exchange (e.g., for satisfying application requirements) between corresponding micro-services within the multi-X execution environment.

Routing becomes a problem of routing the micro-service requests, not just packets, as done through IP. Traditionally, the combination of the Domain Naming Service (DNS) and IP routing has been used for this purpose. However, the advent of virtualisation with use cases such as those outlined above have made it challenging to further rely on the DNS. This is mainly due to the long delay in updating DNS entries to ‘point’ to the right micro-service instances. If one was to use the DNS, 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 [nSFF] with the service function chaining [RFC7665] framework as well as name-based routing [ICN5G][ICN4G] address some aspects described above. However, further extensions to those need to be considered supporting an app-
Service Pinning

Allocating the right resources to the right micro-services is a fundamental task when executing micro-services on such highly distributed app-centric micro-data centres (e.g., resource management in cloud [CLOUDFED]), particularly in the light of volatile resource availability as well as concurrent and highly dynamic resource access. Once the specific set of micro-services of an application has been identified, during the lifetime of the application, requirements (e.g., QoS) must be ensured by the execution environment. Therefore, all micro-data centres and the execution environment will realize mechanisms for ensuring the utilization of specific resources within a pool of resources (i.e., resources in all app-centric micro-data centres), for a specific set of micro-services belonging to one application, while also ensuring integrity in the wider system.

State Synchronisation

Given the highly distributed nature of app-centric micro-services, their state exchange and synchronisation is a very crucial aspect for ensuring in-application and system-wide consistency. Mechanisms that ensure consistency will ensure that data is synchronised with different spatial, temporal and relational data within a given time period.

Security Considerations

N/A

IANA Considerations

N/A

Conclusion

This draft positions the evolution of data centres as one of becoming execution centres for the app-centric experiences provided today by smartphones. 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. Although a number of activities are currently underway to address some of the challenges for realizing such AppCentre evolution, we believe that the COIN research group will provide a suitable forum to drive forward the remaining research and its dissemination into working systems and the necessary
standardization of key aspects and protocols. Future work on this
draft will focus on contributing its main points to an evolving
research roadmap that we would see as a possible outcome of the COIN
AG.

8 References

8.1 Normative References

[RFC2119]  Bradner, S., "Key words for use in RFCs to Indicate


8.2 Informative References


[SCOMPOSE] M. Hirzel, R. Soule, S. Schneider, B. Gedik, and R. Grimm,


Authors’ Addresses

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

Email: Chathura.Sarathchandra@InterDigital.com
Dirk Trossen
InterDigital Europe, Ltd.
64 Great Eastern Street, 1st Floor
London EC2A 3QR
United Kingdom

Email: Dirk.Trossen@InterDigital.com

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

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