IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

# The Continuum of Computing

**Giovanni Jiayi Hu**
[1]Department of Mathematics, University of Padua, Italy I-35121

**ABSTRACT** Work in Progress.

## I. INTRODUCTION

The Internet has evolved significantly since its inception. From just a simple communication layer for information sharing between researchers, it has grown into a ubiquitous platform for complex services. This transformation is driven by many changes and innovations in terms of infrastructure, services and devices.

In the past decade, several infrastructure-related trends have emerged and consolidated. First, sharing of device, server, and networking capabilities (the cloud computing paradigm) has become a reality, giving users and companies access to virtually unlimited amounts of storage and computing power. Nowadays, there is an immense amount of seamlessly connected mobile devices, servers, and network elements, offering their virtualized capabilities to its users.

Second, as mobile computing evolved, people started bringing their devices with them and accessing the Internet anywhere anytime. Today, we are in the midst of the so-called Internet of Things (IoT), where devices (things) are connected to the Internet and each other. These things comprise a multitude of heterogeneous devices ranging from consumer devices, such as mobile phones and wearables, to industrial sensors and actuators, or electrical vehicles (EV).

Concurrently, our capacity for collecting data is expanding dramatically, and yet, our ability to timely manage, manipulate, and analyze this data to transform it into knowledge and understanding has not kept pace. Data sources, volumes, and generation rates are growing exponentially and have outpaced the ability to transport this data in a reliable and timely manner to the cloud.

One can thus envision data-driven and information-rich pervasive computational ecosystems that seamlessly combine this data and computing power to model, manage, control and make use of virtually any realizable sub-system of interest. Use cases exist in diverse application areas from managing extreme events (e.g. environmental monitoring) to optimizing everyday processes (e.g. manifacturing) and improving life quality (e.g. healthcare and smart cities).

It is also a natural consequence that, in edge networks, computational capabilities are gradually being introduced. New approaches that combine distributed services close to the data sources (i.e. edge nodes) with resources in the cloud and along the data path can be effective in processing this data. Depending on the use case and service level requirements, IoT devices may require processing and storage locally, in the cloud or somewhere in between (fog).

This trend paves the way for novel and ubiquitous services in a wide range of application domains. As a matter of fact, Haller et al. [1] define the IoT as "a world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants in business processes. Services are available to interact with these 'smart objects' over the Internet, query their state and any information associated with them, taking into account security and privacy issues."

This thesis proposes a vision which follows naturally from the previous premises - the **Continuum of Computing** - a fluid ecosystem where distributed resources and services are aggregated on-demand to support emerging services.

In the vision of this thesis, there will be service platforms and a multitude of services available over the Internet. The granularity of these services will be very different, ranging from high-level business services to low-level sensor services provided by the Internet of Things. Edge services are emerging close to the data sources to provide non-trivial data processing capabilities.

It is fundamental that services are treated as first-class citizens, allowing services to be dynamically placed by decoupling them from their location. The Continuum must also facilitate the elastic provisioning of virtualized end-to-end service delivery infrastructures, in which leased (virtual) capabilities are dynamically configured and scaled as a function of user demand and service requirements.

This vision goes beyond the traditional elasticity of clouds, since in addition to computational, network and storage resources, the managed capabilities also include end-to-end network configuration and high-level service and device functionalities.

## II. THE PROBLEM STATEMENT

In the past decade and now more than ever, cloud computing has been providing users with the potential to perform computing tasks utilizing resource physically distant to them. A popular definition of cloud computing is defined in [2] as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Conventional single provider infrastructures hosting cloud services on data centers offer undoubtedly a lot of benefits but also hide many challenges. A large data center's energy consumption is high just to keep it operational and like any other centralized computing model, in case of a failure the resulting issues would be adverse.

Another issue is that required data may have to be transferred and stored to separate places, rather than the source, because data centers are often geographically distant from the application users, especially when the data is generated at edge locations geographically distant from the data center. However, exchange of sensitive or personal data is considered critical for applications.

Moreover, the evolution of the Internet of Things has had a significant impact on cloud computing generally and stretching the limitations of conventional cloud architecture. The number of connected devices is increasing exponentially with estimations of dozens of billions of "things" going live in coming years (Gartner 2017).

As a consequence of connecting these sensors to the Internet, large volumes of data are being generated in unprecedented volumes, variety and velocity. This data is currently transferred and stored in the cloud in a centralized manner. Data transfer, especially in these volumes, is extremely expensive and retards computational performance. The result is that an enormous amount of data needs to be transmitted over the network, stored and/or processed by the receivers in an efficient way.

It becomes thus obvious that traditional methods of data processing where all data is collected and processed in a central instance will not suffice. A more decentralized solution is required where data processing could take place before transfer and storage. The key is to reduce the number of messages and the amount of data transmitted throughout all the layers of the system; to that end data should be evaluated locally and only where it makes sense.

With these premises, this thesis visions bringing the cloud computing definition to the Internet of Things and the network along the data path (fog). The rational is two-fold. First, it offers virtually unlimited capabilities regarding storage, memory and processing units that augment IoT devices and components with limited computation capabilities due to construction factors.

Second, scaling fog/edge infrastructure is costly and time-consuming at least as much as in cloud infrastructure, as it is difficult to predict capacity needs in advance. Underprovisioning means potentially losing business while overprovisioning means wasting money on unused infrastructure. Moreover, user demand is often very spiky, meaning that companies traditionally needed to provision for anomalous peaks like gatherings and events, investing in significant infrastructure that sits underutilized most of the time. This also has an environmental cost when underutilized infrastructure consumes significant amounts of power.

This thesis identifies the following significant challenges for the Continuum:

1) *Heterogeneity*: the heterogeneity of the connected devices is immense and can be discrete in many levels, such as computing performance, storage and network requirements, communication protocols, energy consumption amongst others
2) *Portability*: how do we program applications to run in such dynamic environment?
3) *Interoperability*: applications on a platform should be able to amalgamate services and infrastructure from another platform;
4) *Security and Privacy*: personal and sensitive user data are subjected to high risk while many users access public and ad-hoc clouds. In some instances, personal data may have to be stored closer to the users/devices in order to facilitate computing and processing on the edge or fog layer. TODO: Homomorphic encryption?
5) *Mobility*: efficient migration of each application and service has to be supported from platform to platform and follow the users' movements in the network.
6) *Reliability*: establishment of real-time communication between objects and applications with high connectivity and availability must be supported
7) *Service orientation*: consumers should only be concerned with what they want to do and accomplish, and providers with how that could be done and provided to the user.
8) *Orchestration*: advanced orchestration systems are required to support the resource management of heterogeneous devices and adapted to many applications running on the IoT connected devices. Ss their numbers increase, so do their requirements accordingly, making it more complex and far more difficult to cope with.
9) *Virtualization—the*: potential to provision resources and provide access to heterogeneous resources and hardware such as GPUs, FPGAs, etc.

The following subsections will further explore some of the stated problems.

### A. DECENTRALIZATION

A highly distributed network is the most effective architecture for the Continuum, particularly as services becomes more complex and more bandwidth hungry. Although often referred to as a single entity, the Internet is actually composed of thousands of different networks. This means that content generated at the edge must travel over multiple networks to reach its centrally-hosted data center.

Unfortunately, inter-network data communication is neither an efficient nor reliable operation and can be adversely affected by a number of factors.

Capacity at peering points where networks exchange traffic typically lags demand, due in large part to the economic structure of the Internet [8]. Money flows in at the first mile (cloud data centers) and at the last mile (end users), spurring investment in first and last mile infrastructure. However, there is little economic incentive for networks to invest in the middle mile—the high-cost, zerorevenue peering points where networks are forced to cooperate with competing entities. These peering points thus become bottlenecks that cause packet loss and increase latency.

Across the Internet, outages are happening all the time, caused by a wide variety of reasons such as cable cuts, misconfigured routers, DDoS attacks, power outages, even earthquakes and other natural disasters. While failures vary in scope, large-scale occurrences are not uncommon [9].

At scale at which the Internet is growing, it is no longer sufficient to simply have enough server and network bandwidth resources. One must consider the throughput of the entire path from IoT devices to servers to end users. The bottleneck is no longer likely to be at just the far ends of the path. It could be at a peering point or an ISP's upstream connectivity, or it could be due to the network latency between server and device.

A fog and edge-based platform, with servers in thousands of locations, can achieve the scale needed with each location supporting higher orders of throughput.

For these reasons, alternate models such as fog computing and edge computing have emerged in the recent years.

#### 1) Fog Computing

Fog computing is a decentralized computing infrastructure which is used particularly as a complement to cloud computing. It leverages the compute resources at the edge network and brings the computational processing closer to the data source by offloading workload to edge nodes from cloud data centers. The network nodes near the edge providing these resources are called fog nodes. Overall, any device with computing, storage and network connectivity can constitute a fog node, for example switches and routers, industrial controllers, embedded servers and video surveillance cameras.

Its first level usually lays a single hop away from the edge and is an extension of edge computing.

#### 2) Mobile Edge Computing

Mobile Edge Computing, also referred to as just edge computing, takes place only on the edge of the network. Processing is executed closer to the data source and eliminates the costly data transfer to a remote data center or cloud. This significantly improves user Quality of Service (QoS) as, similar to fog computing, there is considerable network latency reduction and bandwidth consumption by the mobile subscribers.

#### 3) Serverless Computing

Serverless Computing involves building, running and providing applications and services without taking into consideration the server side. "Serverless" does not mean that there is no server usage but rather the main focus is on the application itself rather than what happens on the physical infrastructure.

Serverless Computing is synonymous with Function-as-a-Service (FaaS) and event-based programming as the execution of an application will be executed only when necessary and not all the time, thus meaning that an event can trigger the execution of a function or more than one function concurrently.

Although serverless computing is a trend generated in cloud computing per se, it can prove useful for many edge computing use cases.

### B. SERVICE ORIENTATION

This thesis visions service oriented paradigms as necessary for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. A Service Oriented Architecture (SOA) provides "a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations" [6]. According to this model, the major components of a basic SOA and their possible interactions are: a service provider publishes his service interface via a service registry where a service requester/consumer can find it and subsequently may bind to the service provider [1]. The central concept of the SOA reference model is the existence of services which provide access to capabilities by well-defined interfaces to be exercised following a service contract with constraints and policies. This enables a loose coupling of services (thereby minimizing mutual dependencies). Services are provided by entities, the service provider, and are to be used by others, the service consumers. Services may be composed on the basis of other, existing services, thereby adhering to the principle of reuse. They are autonomous in the sense that they solely control the logic they encapsulate, uniformly described and publicly retrievable via certain discovery mechanisms.

However, to enable the seamless and agile interoperation of smart items in the Internet of Things, a number of challenges still exist with respect to the organization and implementation of services.

First of all, the lack of comprehensive, trustworthy and widely accepted service intermediaries in the Internet of today still prevents the establishment of a Continuum as described above. Service providers are necessary for improved navigation by facilitating the efficient retrieval of services which match a given user need, service performance transparency through permanent monitoring of service quality and availability, and governance, i.e., controlling and enforcing the delivery of service levels as defined in Service-Level Agreements (SLA).

Secondly, the lack of means for composing diverse services towards higher-order services meeting actual user de-

mands imposes additional challenges to the emergence of the Continuum of Computing. The lack of interoperation prevents web services from being set up quickly and easily. Cloud providers and their respective systems adhere to different conventions with respect to interfaces and communication protocols. For this reason, the information objects which need to be exchanged between services are virtually guaranteed to have inconsistency in both syntax and semantics. A major effort will be required to uniformly provide meaning of services and their input as well as output objects in a comprehensible way. Semantically organized services are believed to improve retrievability and seamless composability significantly.

Thirdly, services in the Internet of today can be considered as mute and autistic. In the Future Internet, services are expected to become more open and reactive to their respective environments particularly in two respects. First, existing service interfaces (e.g., REST-based) are not designed to be interpreted and used by machines but rather by humans. This frequently prevents the systems from easily discovering and interacting with them. Methods are needed to allow to both "put a face" on services and to thus improve user-service interaction and machine-to-machine communication. Second, today's Internet-based services are rarely context-sensitive. However, in order to ..., services need to be aware of different aspects of the environment they are acting in. Context services can provide such information to the application services that implement local control loops and that trigger specific actions. Such context services will be composed of several lower-level services which deliver individual sensor readings.

#### 1) History of service-oriented architectures

The idea of service-orientation can be partially traced back to the object-oriented programming (OOP) literature [3]. However, the evolution of objects into services, and the relative comparisons, has to be treated carefully since the first focus on encapsulation and information is hidden in a shared-memory scenario, while the second is built on the idea of independent deployment and message-passing. It is therefore a paradigm shift, where both the paradigms share the common idea of componentization.

The last decade has seen a further shift towards the concept of service first and the natural evolution to microservices afterwards. Service oriented architectures (SOA) have been introduced to harness the complexity of distributed systems and to integrate different software applications. In SOA, a a service offers functionalities to other components, accessible via message passing. Services decouple their interfaces (i.e. how other services access their functionalities) from their implementation.

This is in striking contrast to monolithic architectures, where the application is composed of a single program or platform, typically providing a user interface and data access through a database. However, the monolithic architecture starts to pose real challenges and difficulties when growing exponentially because of the application need to scale up.

Service Oriented Architectures (SOA) are instead build around services, which are built to work in an orchestrated manner to modularize the system. It is more challenging to divide the application in to multiple services, but it enables greater flexibility, extensibility and reusability of existing services for multiple use cases. The benefits of this model are application modularity, service reusability and enhanced security in the (re)building process and development of the application [4]. A major disadvantage is the complexity in orchestrating all the services, especially when the project is complex and the components are huge.

The first generation of SOA architectures defined daunting and nebulous requirements for services (e.g., discoverability and service contracts), and this hindered the adoption of the model. Microservices are the second iteration, introduced as a solution for the gaps in the SOA approach.

This approach divides applications into more granular components by distributing them into small independent services. Each service implements and attends to separate business functions and capabilities to maintain independency from other services. They are the mainly deployed in an automated manner, through a container and communicating through Rest APIs [5], thus making the impact of programming language insignificant. This allows microservices to be easily deployed in the cloud, offering great reusability and minimal or no centralized management and orchestration.

Consumers should only be concerned with what they want to do and accomplish, and providers with how that could be done and provided to the user. A successful interface establishment between those two actors can lead to minimal direct consumer interaction with provider's infrastructure, thus allowing full control to the provider.

Furthermore, this results in the assumption that various service implementations should already exist and the consumer himself does not have to be an expert and develop them. physical resources provided by the infrastructure should not be consumer-aware and yet there may be several diverse implementations to meet specific service demands.

These implementations can differ in hardware type and could be characterized by different price and performance attributes.

### C. ORCHESTRATION

Orchestration can also be defined as the use of programming technology to manage the interconnections and interactions among workloads on distributed edge-cloud infrastructure. This is accomplished through three main attributes of orchestration, which are closely related: service orchestration, workload orchestration, and resource orchestration.

Continuum orchestration is a crucial feature for many IT organisations and DevOps adopters as a way to speed the delivery of services, simplify optimisation, and reduce costs (Nygren et al. 2010). A cloud orchestrator automates the management, coordination, and organisation of distributed computer systems, services.

The orchestration of virtualised environments is challenging due to the scale, heterogeneity, and diversity of resource types and the uncertainties of the underlying cloud environment. The uncertainties arise from a number of factors including resource capacity demand (e.g. bandwidth and memory), failures (e.g. failure of a network link), user access pattern (e.g. number of users and location) and lifecycle activities of applications. Cloud resource orchestration is challenging because applications are composed of multiple, heterogeneous software and hardware resources, which may have integration and interoperation dependencies.

The orchestration and management of a cloud-to-thing architecture is mostly realised through virtualisation. As discussed in Chap. 2, the evolution of virtualisation has moved away from virtual machines towards more lightweight solutions such as containers. This is specifically relevant for application packaging at a software platform and application level. Different application packages such as containers have been proposed to cluster Cloud-to-Edge and solutions such as Docker container and Kubernetes architectures. Yet, there is still a need for a topology specification and a derived orchestration plan for cloud edge computing.

### D. VIRTUALIZATION

In addition to architecture modeling, different approaches have emerged regarding service modeling. In addition to conventional SaaS, IaaS, PaaS models, a new approach in virtualization is bare metal or native virtualization (Scarfone et al. 2011). Here, the hypervisor can run directly on the underlying hardware of the provider without a host operating system. Bare metal offers more security, providing that the hypervisor itself is well-secured. As the hypervisor is placed directly over the hardware, there is no host OS thus it cannot be harmed. This model is mostly used for servers in the cloud. Nevertheless, the hardware provided is limited so that a hypervisor does not consume the total available resources.

Hosted Virtualization is where the hypervisor runs over the host OS. The difference between hosted virtualization and bare metal is that the hypervisor is provided with more virtual resources but, on the other hand, the potential to harm the host OS is significantly increased.

Function as a Service (FaaS) is synonymous with serverless computing. Basically, FaaS enhances the microservices model development. During the development process, server operations are not taken into account, as services are hosted externally. Compared to conventional cloud models, where at least one server is utilized, FaaS triggers a server only when a function is conducted, executes the expected operations and then terminates. The major advantages of this model are increased scalability and independency of the applications and lower costs. As costs are based only on per used functionality, expenses from inactive resources are eliminated. A major disadvantage is the reduced transparency as FaaS is managed externally.

## III. TECHNICALS

### A. ARCHITECTURE

Architecture is what allows systems to evolve and provide a certain level of service throughout their lifecycle. In software engineering, architecture is concerned with providing a bridge between system functionality and requirements for quality attributes that the system has to meet. (TODO: Architecture definition from REST)

### B. DECENTRALIZATION

To incorporate resources located outside the cloud data centers. Cloudlet [7] approaches offer a more decentralized computing infrastructure, as they are located closer to the edge of the network, thus minimizing transfer cost and communication latency as they are closer, sometimes a mere single hop, to the users and the generated data. Nevertheless, their computing power is much less than a conventional cloud infrastructure as they are composed of less powerful processors and are significantly smaller in size.

### C. INTEROPERABILITY

A multi-cloud utilizes resources from multiple providers, thus providing the potential to make applications portable, meaning that data from it or even the whole application can migrate from one cloud to another in case of failure or cost-effectiveness

### D. SERVICE ORIENTATION

The concept of supporting loosely coupled, business-aligned and networked services as introduced above can be realized with the help of numerous different technologies such as the Web Services stack. This stack comprises WSDL as a uniform format for service interfaces, UDDI as standard for specifying publicly available service registries, SOAP as data exchange protocol and BPEL as language for orchestrating services according to specific business logic.

### E. COAP

Inefficient communications protocols: Although it was designed for reliability and congestion-avoidance, TCP carries significant overhead and can have suboptimal performance for links with high latency or packet loss, both of which are common across the wide-area Internet. Middle mile congestion exacerbates the problem, as packet loss triggers TCP retransmissions, further slowing down communications.

Additionally, for interactive applications, the multiple round trips required for HTTP requests can quickly add up, affecting application performance.

## IV. CONCLUSION

The challenge with cloud and IoT is not that one form will supplant another, but that we lack a programming and execution model that is inclusive and capable of harnessing the entire computing continuum to program our new intelligent world.

.

Appendixes, if needed, appear before the acknowledgment.

## ACKNOWLEDGMENT

Work in Progress

## REFERENCES

[1] Haller S., Karnouskos S., Schroth C. (2009) The Internet of Things in an Enterprise Context. In: Domingue J., Fensel D., Traverso P. (eds) Future Internet – FIS 2008. FIS 2008. Lecture Notes in Computer Science, vol 5468. Springer, Berlin, Heidelberg.

[2] Mell, P., and Grance, T. 2011. The NIST definition of cloud computing, Recommendations of the National Istitute of Standards and Technology, NIST Special Publication 800-145.

[3] Dragoni N. et al. (2017) Microservices: Yesterday, Today, and Tomorrow. In: Mazzara M., Meyer B. (eds) Present and Ulterior Software Engineering. Springer, Cham.

[4] T Lynn, JG Mooney, B Lee, PT Endo. 2020. The cloud-to-thing continuum: opportunities and challenges in cloud, fog and edge computing.

[5] Roy Fielding. 2000. Representational State Transfer (REST). https://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm. Accessed on: Mar. 15, 2021.

[6] MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R. and Hamilton, B.A., 2006. Reference model for service oriented architecture 1.0. OASIS standard, 12(S 18).

[7] Satyanarayanan, M., Bahl, P., Caceres, R. and Davies, N., 2009. The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing, 8(4), pp.14-23.

[8] Nygren, E., Sitaraman, R.K. and Sun, J., 2010. The akamai network: a platform for high-performance internet applications. ACM SIGOPS Operating Systems Review, 44(3), pp.2-19.

[9] Prolonged AWS outage takes down a big chunk of the internet. 2020. [Online]. Available: https://www.theverge.com/2020/11/25/21719396/amazon-web-services-aws-outage-down-internet. Accessed on: Mar. 15, 2021.